UNMDP - Facultad de Ingeniería - Departamento de Informática

# Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería


## Autores

Emanuel Ponce - manuponce1993@gmail.com

Manuel Nucci - manuucci96@gmail.com

Sebastián Canónaco - sebastiancanonaco@gmail.com

## Director

Fernando Soriano


**Proyecto final para optar al grado de Ingeniero en Informática**

**Mar del Plata, 21 de Diciembre de 2021**

UNMDP - Facultad de Ingeniería - Departamento de Informática

# Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería


## Autores

Emanuel Ponce - manuponce1993@gmail.com

Manuel Nucci - manuucci96@gmail.com

Sebastián Canónaco - sebastiancanonaco@gmail.com

## Director

Fernando Soriano


**Proyecto final para optar al grado de Ingeniero en Informática**

**Mar del Plata, 21 de Diciembre de 2021**

# ÍNDICE

# 1. Agradecimientos

En primer lugar, queremos agradecer a la Facultad de Ingeniería y a todos los docentes que la integran por habernos logrado transmitir una gran cantidad de conocimientos a partir de los cuales nos basamos para poder resolver la problemática elegida. En especial queremos hacer mención a Felipe Evans quien estuvo presente gran parte del proyecto y cuya ayuda fue imprescindible para atacar algunos de los desafíos más complejos que tuvimos que afrontar.

En segundo lugar queremos agradecer a nuestro director Fernando Soriano por todo el apoyo y útiles consejos brindados durante la ejecución del proyecto y posterior escritura del presente documento. Sus aportes fueron sumamente valiosos para poder estructurar de manera organizada y comprensible todo el contenido expuesto en el trabajo desarrollado.

Por último queremos agradecer a nuestras familias por la contención y apoyo que nos han brindado durante todo este tiempo, aún cuándo nuestros ánimos decaían por ver aquella meta tan lejana. En particular a nuestros padres y parejas quienes estuvieron siempre presentes dándonos fuerzas y energía para poder cumplir el objetivo deseado.

Muchas gracias a todos.

## 2. Resumen del Proyecto

Con la filosofía de “el camino correcto no siempre es el más fácil”, este escrito integra los tres pilares fundamentales que los autores consideramos indispensables al momento de evaluar y transitar por el desarrollo de un proyecto de gran magnitud: Proyecto, Proceso y Producto. En el mismo, se detallarán y fundamentarán las prácticas y metodologías llevadas a cabo en cada una de estas aristas. Además, este documento también pretende transmitir cómo un pensamiento crítico e ingenieril puede sortear las dificultades y los imprevistos constantes que amenazan la vitalidad de un proyecto.

El presente proyecto, nació a partir de una iniciativa del Centro de Estudiantes de Ingeniería (CEI) con la búsqueda de desarrollar un sistema informático que automatizara el proceso de gestión de impresiones de la Facultad de Ingeniería de Mar del Plata.

Si bien en una primera instancia los requerimientos planteados por el cliente parecían ser acertados y alcanzables, tenían una visión sesgada de sus necesidades, debido a que apuntaban a una solución cortoplacista y poco escalable, manteniendo intactas las bases y el funcionamiento de un proceso deprecado y desgastado, en el que se consideraban cómodos ya que se encontraban dentro de su zona de confort.

El equipo de desarrollo tuvo la capacidad para observar esta situación en una etapa temprana del proyecto, alertando que el desarrollo aislado de un sistema informático no daría respuesta a las raíces de sus necesidades y problemáticas. Es por esto que luego de elicitar los principales requerimientos transversales, adentrándose en el contexto y el dominio del problema, se realizó un análisis profundo y detallado del labor que se estaba llevando a cabo y se detectó la necesidad de realizar una reingeniería del proceso para dar una solución abarcativa, robusta y con vistas a un mayor horizonte temporal.

De esta forma, el producto informático pasó de ser el objetivo principal del proyecto a ser un medio para dar respuesta y soporte a una transformación cultural y digital que generaría un impacto, no sólo en el CEI, sino en toda la comunidad de estudiantes de la Facultad de Ingeniería de Mar del Plata.

Ningún proyecto está exento de riesgos y dificultades, y éste, sin dudas, no fue la excepción. A pesar de contar con un plan de implementación inicial y del esfuerzo constante por consolidar el pilar de índole administrativo y de gestión, los plazos y las estimaciones originales se vieron fuertemente afectadas por causales ajenas (tales como la modificación de requisitos, cambios en el alcance del proyecto o adelantamiento de plazos) e imprevistos de índole mundial como el contexto pandémico que atravesó el proyecto en sus fases iniciales.

Ante este conjunto de situaciones amenazantes, el equipo de desarrollo supo encontrar soluciones en el momento oportuno para cada una de ellas y así solventar la vitalidad y completitud del proyecto. En el aspecto funcional, se pudo realizar una priorización de los requisitos iniciales y lanzar en una etapa muy temprana un Producto Mínimo Viable (MVP) que permitiera dar respuesta a la urgente necesidad de disponibilizar el acceso al material académico de manera online a los

estudiantes, en un contexto donde la presencialidad no era posible. Este primer despliegue productivo logró descomprimir una situación de exigencia en cuanto a términos temporales del despliegue de la versión final. A raíz de esto el equipo logró realizar una migración de la tecnología inicialmente seleccionada con el fin de construir la versión definitiva de un sistema informático robusto y extensible que estuviera a la altura de la magnitud del proyecto.

El haber transitado este camino nos ha dejado experiencias sumamente valiosas y nos ha dotado de herramientas y competencias que marcarán no sólo nuestro perfil técnico-profesional como ingenieros, sino también de cualidades y virtudes que hacen al aspecto humano.

El equipo está orgulloso de poder aportar valor a través de una solución tecnológica al Centro de Estudiantes de Ingeniería y a toda la facultad en sí, y devolver parte del conocimiento que le fue transmitido mediante la concreción de un proyecto que beneficiará a toda la comunidad.

# 3. Introducción

El acceso a material educativo es una necesidad básica de todo estudiante. Es importante que el mismo esté disponible y actualizado en base a las últimas modificaciones que pudiera haber sufrido. Este conjunto de requisitos es el principal desafío que encuentran los centros de copiado asociados a una institución educativa, y con el que lidian día a día para satisfacerlos.

Por otro lado, muchos alumnos dependen (según la técnica de estudio que apliquen) de contar con el material físico. Esto les permite afianzar los conceptos a aprender a través del resaltado, subrayado, toma de apuntes, etc. Los centros de copiado están conscientes de este requisito y para ello ofrecen servicios de impresión, en ocasiones a bajo coste, lo cual es sumamente útil en contextos de índole pública, en los cuales no todos los estudiantes pueden permitirse el lujo de invertir cualquier cantidad de dinero en el material deseado.

Todo centro de copiado presenta, en mayor o menor medida, numerosos problemas relacionados a errores en cualquier etapa de la gestión de pedidos, la existencia de largas colas en horas de alta concurrencia, la imposibilidad de emitir pedidos de forma remota por los usuarios al no manejar dinero digital y tantos otros. Estos inconvenientes atentan contra la comodidad del usuario y la salud financiera de la organización en cuestión, puesto que un usuario que ha tenido una mala experiencia difícilmente vuelva a utilizar los servicios con tanta competencia y alternativas de por medio.

El Centro de Estudiantes de Ingeniería (CEI) es un órgano democrático que representa a los estudiantes y es el responsable de administrar y gestionar el centro de copiado de la Facultad de Ingeniería de Mar del Plata. Dicho ente brinda servicios de impresión focalizados en el sector estudiantil, permitiendo la consulta e impresión de material educativo (cuyo acceso, previo a la transformación digital que abarca este proyecto, se encontraba limitado y restringido al uso de los propios servidores físicos alojados en la sala de impresión) y además posee un sistema de beneficios para los estudiantes, otorgando becas estudiantiles a aquellos alumnos que más lo necesiten.

# 4. Proyecto

Para que un proyecto esté bien diseñado y formulado se debe explicar cuál es su finalidad, sus objetivos, beneficiarios, productos, actividades, cronograma, presupuesto, etc. En esta sección se desarrollarán algunas de las partes que determinaron su forma y que posibilitaron tener un marco contextual en el cual perseguir los objetivos definidos.

## 4.1. Motivación y objetivos

### 4.1.1. Objetivo general

En una primera instancia, el objetivo general planteado por el CEI fue el de implementar un sistema informático que digitalizara los servicios y actividades del centro de impresión, manteniendo las formas y los procesos tales como los venían llevando a cabo en aquel momento. Dicho sistema daría respuesta a la posibilidad de solicitar pedidos de impresión por parte del estudiante (que contaría además con un saldo digital que lo habilitaría a realizar dichas solicitudes) y a gestionar un listado de dichos pedidos por parte de los encargados del centro de copiado.

Sin embargo, a medida que fue avanzando la etapa de elicitación y relevamiento (por medio de entrevistas, observaciones del comportamiento in situ e investigaciones de mercado), se fue tomando mayor contexto de la problemática y las necesidades reales e implícitas que poseía el CEI. A partir del modelado de los flujos y procesos que se estaban llevando a cabo por los encargados del centro de impresiones y realizando un análisis detallado sobre los mismos, se identificó que el desarrollo de un sistema informático aislado carecería de sentido tal como estaba planteado desde un comienzo, invirtiendo tiempo y recursos en una solución a corto plazo, que lo único que generaría sería ocultar la problemática base por un breve período de tiempo. Debido a esto, surgió la necesidad de realizar una reingeniería de los procesos previamente analizados con el fin de lograr una solución a largo plazo que atacara al causante real del problema.

Con este contexto, se puede observar que el objetivo mutó considerablemente, de manera tal que la reingeniería del proceso nació como una necesidad identificada por el equipo de desarrollo a partir de los requisitos inicialmente planteados. En síntesis, el objetivo general del proyecto queda enmarcado en brindar una transformación digital en el proceso llevado a cabo por el centro de copiado, implementando un sistema informático que diera respuesta a dicho proceso.

### 4.1.2. Objetivos específicos

Si bien el objetivo general es bastante amplio y marca un horizonte a seguir, las diferentes etapas que se fueron atravesando tuvieron que estar guiadas por metas concretas que se pudiesen ir cumpliendo y verificando.

En concreto, los objetivos que se pretenden alcanzar son los siguientes:

- Rediseñar los procesos llevados a cabo tanto por los estudiantes y becados que consumen de los servicios del CEI como también de los encargados que administran y gestionan los pedidos y actividades del centro de copiado.
- Implementar una plataforma digital centralizada que cubra todos los aspectos y posibilidades que brindan los nuevos procesos diseñados.
- Documentar y disponibilizar todos los activos del proyecto que aporten a una satisfactoria transferencia del conocimiento.
- Lograr consolidar una gestión y administración ágil y dinámica del proyecto, que permita adaptarse a posibles modificaciones del cliente y que existan planes de acción ante determinados riesgos.
- Disponibilizar la plataforma digital de manera masiva a través de los principales canales sobre los cuáles los usuarios puedan acceder y operar.
- Agilizar la operatoria diaria por parte de los empleados del CEI, ya sea eliminando aquellas tareas que no se ajusten al proceso propuesto o automatizando otras repetitivas y tediosas a través de una herramienta. Con esto último se busca también minimizar la cantidad de errores que puedan surgir por malentendidos en los pedidos.
- Eliminar o reducir en su máxima expresión los tiempos de espera que existen y padecen actualmente los usuarios del CEI.
- Lograr una mejor asignación de recursos respecto de las becas que se otorgan a los estudiantes.
- Disponibilizar de manera online el repositorio de archivos académicos a todos los estudiantes de la facultad.
- Proveer a las cátedras un repositorio común de información en el cual subir y disponibilizar su contenido, facilitando su acceso e impresión de ser necesario.
- Permitir un seguimiento en tiempo real de los pedidos, de forma tal de reducir la cantidad de consultas por parte de estudiantes, eliminando tareas repetitivas y haciendo más eficiente la utilización del tiempo por ambas partes.
- Posibilitar la gestión de un saldo virtual para el consumo de los servicios del CEI, prestando especial atención a las prácticas de seguridad adecuadas.
- Lograr un comportamiento dinámico y flexible de la plataforma a través de la parametrización en su configuración.

## 4.2. Alcance del sistema

El alcance del sistema a desarrollar estuvo condicionado fuertemente por el CEI en sus primeras etapas, sufriendo varias redefiniciones hasta poder definir lo que sería la plataforma a construir para el Trabajo Final.

Dado que las definiciones sobre el producto deseado fueron bastante vagas en un comienzo, el alcance sufrió recurrentes modificaciones hasta llegar a su versión final, fruto de la adquisición de experiencia sobre la solución a desarrollar a través de algunas elicitaciones.

El sistema a desarrollar comprende todas aquellas funcionalidades básicas que se considera que debe tener la plataforma para poder, por un lado, sentar las bases de una solución extensible y mejorable a futuro y, por el otro, permitir una utilización punta a punta del nuevo proceso propuesto al CEI.

De esta forma, a continuación se encuentran listadas las capacidades que fueron acordadas con el CEI:

- Registro y autenticación de usuarios en el sistema.
- ABM de todos los tipos de usuarios que soporta el sistema.
- ABM de las principales entidades con las cuales operará el sistema. Entre ellas podemos nombrar: sede, carrera, materia, grupo de anillado, archivo, pedido y movimiento.
- Permitir la carga de saldo y su transferencia entre cuentas de billetera.
- Disponibilizar y gestionar un repositorio de información para poder generar nuevos encargos.
- Crear nuevos pedidos, gestionar y poder dar un seguimiento en tiempo real sobre los mismos.
- Parametrizar el sistema para modificar su comportamiento en función de las necesidades del Administrador.
- Permitir el acceso al sistema a través de una aplicación web y móvil (multiplataforma).

La solución a implementar será un sistema totalmente funcional a partir de las capacidades mencionadas anteriormente y que sentará las bases para desarrollos posteriores que se deseen realizar, siempre tomando como input todo lo analizado, diseñado e implementado en el presente trabajo.

## 4.3. Antecedentes del equipo

La distribución de conocimientos dentro del equipo es uno de los principales fuertes que se contó desde el comienzo del proyecto. En parte por suerte y casualidad, los intereses de cada uno de los integrantes son diversos. Esto permite a cada uno de ellos poder focalizarse en una parte de la

solución y profundizar sus habilidades en esa temática, contando con la seguridad que otra área estará siendo desarrollada por una persona más capacitada para la tarea.

De esta manera, el equipo está integrado por estudiantes interiorizados en las temáticas de UX, UI, desarrollo en frontend y backend, arquitectura, bases de datos, seguridad, Cloud, automatización, Business Intelligence, procesos, gestión de proyecto y otras áreas más. Todo esto es posible gracias a que el grupo ya contaba con experiencias previas en la materia, a través de una o más PPS y un constante deseo de aprender.

Sumado a todo esto, la sinergia que existe hoy en el equipo comenzó desde mucho tiempo antes de la concreción del Trabajo Final, afianzando la confianza entre uno y otro a través de sucesivas cursadas y trabajos prácticos en conjunto.

## 4.4. Análisis FODA

En las siguientes subsecciones se desarrollan los resultados del análisis FODA realizado para relevar tanto las características internas (fortalezas y debilidades) como externas (oportunidades y amenazas) que tuvo el equipo en cuenta a la hora de encarar la solución propuesta.

### 4.4.1. Fortalezas

- Sinergia preexistente en el equipo.
- Experiencia profesional y habilidades del equipo tanto en gestión como en desarrollo.
- Conocimiento de los procesos actuales llevados a cabo por el CEI.

### 4.4.2. Oportunidades

- Contacto cercano con el cliente.
- Contacto con referentes técnicos para salvar dudas y problemas durante el desarrollo.
- Inexistencia en el mercado de soluciones a la altura de lo deseado.
- Existencia de responsable de infraestructura que asumirá el rol de mantener el sistema operativo una vez desplegado.
- El contexto de cuarentena permite no tener plazos de entrega tan ajustados y rígidos para el proyecto.

### 4.4.3. Debilidades

- El equipo se encuentra con una alta demanda para tareas ajenas al proyecto.

#### 4.4.4. Amenazas

- Al no tener el cliente una clara visión sobre el producto a entregar, el proyecto podría sufrir redefiniciones que extiendan los plazos del proyecto.

### 4.5. Viabilidad

En el **análisis de la viabilidad** llevado a cabo por el equipo se propuso estudiar la factibilidad de éxito o fracaso del proyecto a partir de una serie de datos empíricos: contexto del proyecto, necesidades identificadas, aceptación cultural del cliente y la comunidad, recursos disponibles.

A continuación se exponen tres de las variables que se tuvieron en cuenta a la hora de la evaluación del proyecto que determinaron su factibilidad y su posterior ejecución.

#### 4.5.1. Técnica

Como bien se ha planteado en secciones anteriores, el equipo cuenta con conocimientos y experiencia en diferentes tecnologías que hacen posible la concreción del producto deseado.

La solución planteada se encuentra dentro de las soluciones más demandadas por las organizaciones para poder disponibilizar sus servicios de forma digital. Esto se logra a través de un sitio web y/o una aplicación móvil. El stack tecnológico manejado puede fácilmente acoplarse a estos requerimientos y llegar a la meta deseada.

#### 4.5.2. Económica

Para la construcción del sistema no se necesita ningún servicio o componente que requiera una inversión por parte del cliente. Todo el proceso de desarrollo será llevado a cabo utilizando herramientas gratuitas provistas por la comunidad, o accediendo a capas gratuitas cuando se requiera utilizar algún tipo de servicio externo en particular.

#### 4.5.3. Temporal

La viabilidad temporal es un desafío latente dado que la falta de visión de producto y proceso por parte del cliente introducen un alto grado de incertidumbre, factor que puede afectar seriamente la planificación del proyecto.

Dadas las expectativas y necesidades del cliente se estima que el proyecto estará enmarcado en el plazo de un año. Dada la complejidad de la solución que se percibe se decide estimar el costo del proyecto en un total de ***1500 horas***, las cuales abarcarán *todas* las etapas del trabajo.

Basándose en un esquema de dedicación de **3 horas diarias** por integrante del equipo esto arroja una duración de proyecto de ***ocho meses y medio***, por lo cual desde el punto de vista de horas-hombre no existe ninguna alarma temprana que levantar en este aspecto.

A partir de dicha evaluación, se realiza una estimación más granular sobre las diferentes tareas a llevar a cabo, la cual puede verse a continuación:

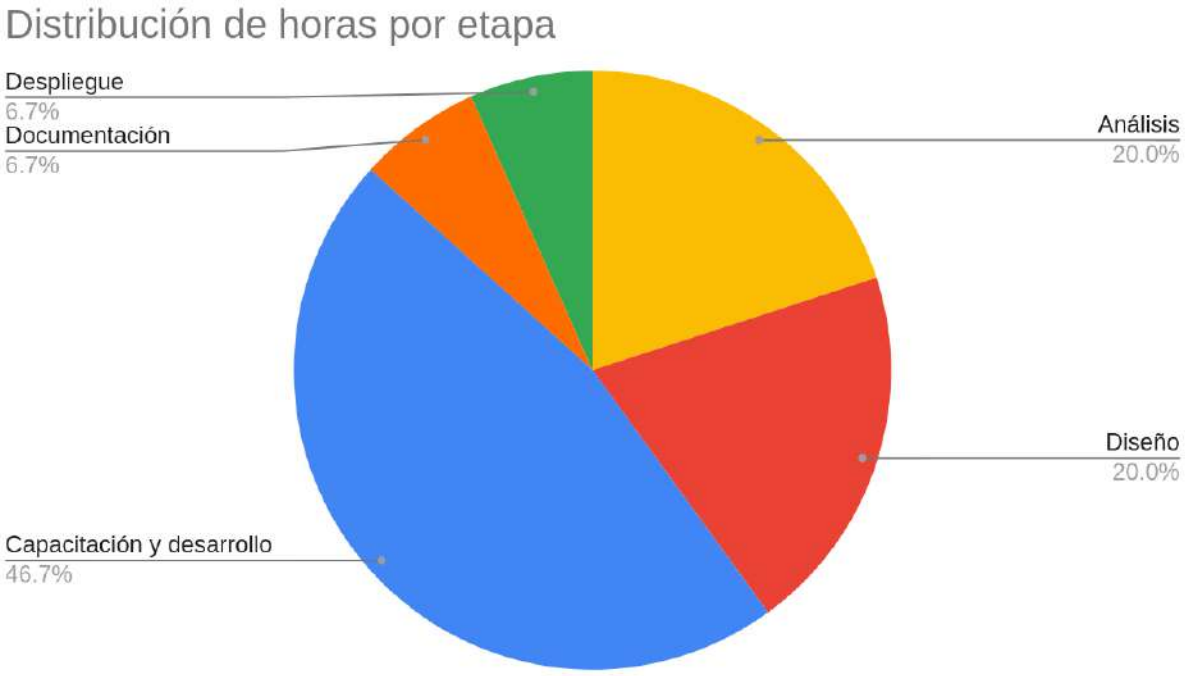



**Figura 1**. Distribución de horas por etapa

A partir del gráfico se puede apreciar que las etapas de **Análisis** y **Diseño** representan el 40% del trabajo estimado, dado que gran parte de los objetivos planteados en el proyecto consiste en entender el modelo de negocio del CEI y trabajar en un rediseño de sus procesos, junto con la arquitectura de la solución que sustentará a los mismos.

La etapa de **Capacitación y Desarrollo** es la que mayor cantidad de recursos se estima que consumirá, puesto que no solo incluye la implementación de la solución per se sino también la capacitación y el aprendizaje de todas las herramientas necesarias para ello. Esto incluirá los frameworks a utilizar en cada una de sus capas junto con las librerías y servicios externos adicionales.

Las etapas de **Despliegue** de la solución y la **Documentación** de la misma sumado a la escritura del Trabajo Final son consideradas equivalentes y con bajo coste.

A partir de esta estimación se deduce que el proyecto es **factible** de realizarse y a su vez califica como una propuesta viable a ser presentada como **Trabajo Final** de la carrera, a pesar de tener los integrantes del equipo otras obligaciones externas que atender.

# 5. Proceso

En esta sección se expondrán los diferentes pilares que moldearon el proceso y metodologías de trabajo llevadas a cabo por el equipo y que posibilitaron contar con la velocidad y el orden necesarios para la estructuración, ejecución y verificación de las distintas actividades, acciones y tareas.

## 5.1. Metodología de trabajo

### 5.1.1. Marco de trabajo de referencia

Debido a que el equipo de desarrollo está conformado por 3 integrantes, fue preciso desde un primer momento contar con una marco de trabajo que permitiera organizar las tareas a realizar. Para tal fin, existen numerosas metodologías en la industria que permiten esto, tales como: Kanban, Scrum, Extreme Programming (XP), etc.

En base al conocimiento de cuáles son las metodologías más utilizadas por diferentes software factories del mercado, se decidió tomar como base de referencia la metodología **Scrum**, adaptándola a las necesidades del equipo y volcando las tareas en un tablero **Kanban** para un fácil seguimiento del estado del proyecto.

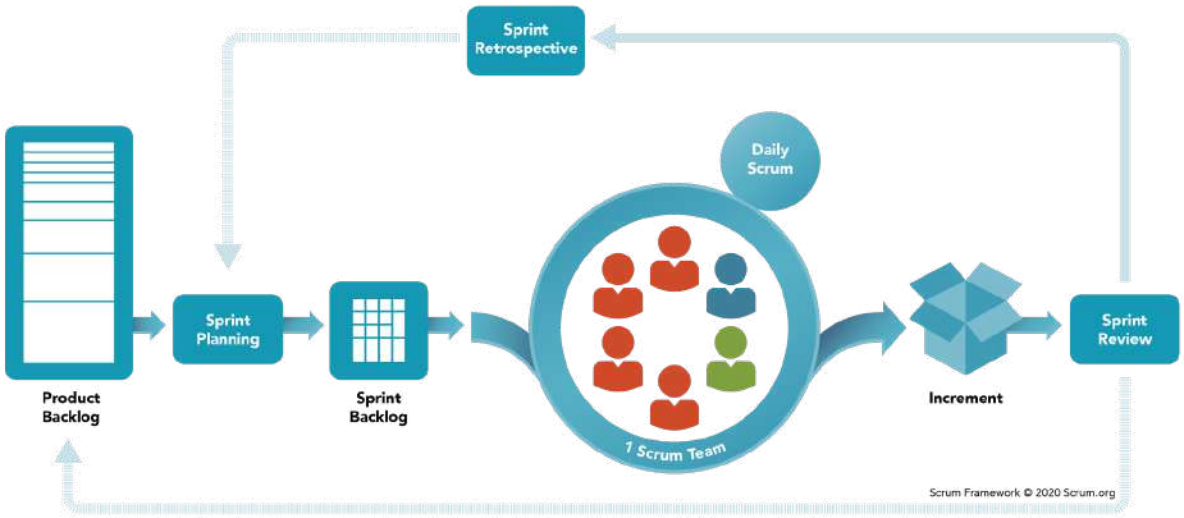



**Figura 2**. Scrum framework

### 5.1.2. Proceso y gestión de tareas

En este apartado se pretende abarcar y detallar los flujos y estándares utilizados asociados a la metodología de desarrollo del equipo.

En primera instancia, es preciso mencionar que debido a que los integrantes del equipo tuvieron durante toda la vida del proyecto una agenda ajustada respecto a otras responsabilidades ajenas al Trabajo Final, se decidió optar por utilizar un tablero Kanban para no depender rigurosamente de deadlines tan estrictos como requiere la práctica purista de Scrum. En consecuencia, se propuso enmarcar todo el trabajo por hacer bajo ***épicas***, las cuáles debieron irse avanzando a medida que las tareas fueron saliendo del ***backlog*** y pasaron al tablero activo.

Como punto de partida, es posible diferenciar dos fuentes originarias de creación de tareas: Producto y Tecnología. La primera enmarca aquellas tareas asociadas de manera directa al modelo de negocio, que pretenden dar respuesta a los requisitos funcionales detectados en la etapa de análisis y que su realización otorga un valor agregado de cara al usuario final. La segunda, en cambio, involucra aquellas asignaciones que, si bien aportan valor al proyecto, no están relacionadas directamente con el modelo de negocio, sino que apunta a trabajos internos del equipo de desarrollo.

A partir de esta aclaración se muestra un diagrama asociado al proceso diseñado por el equipo para la creación de tareas:

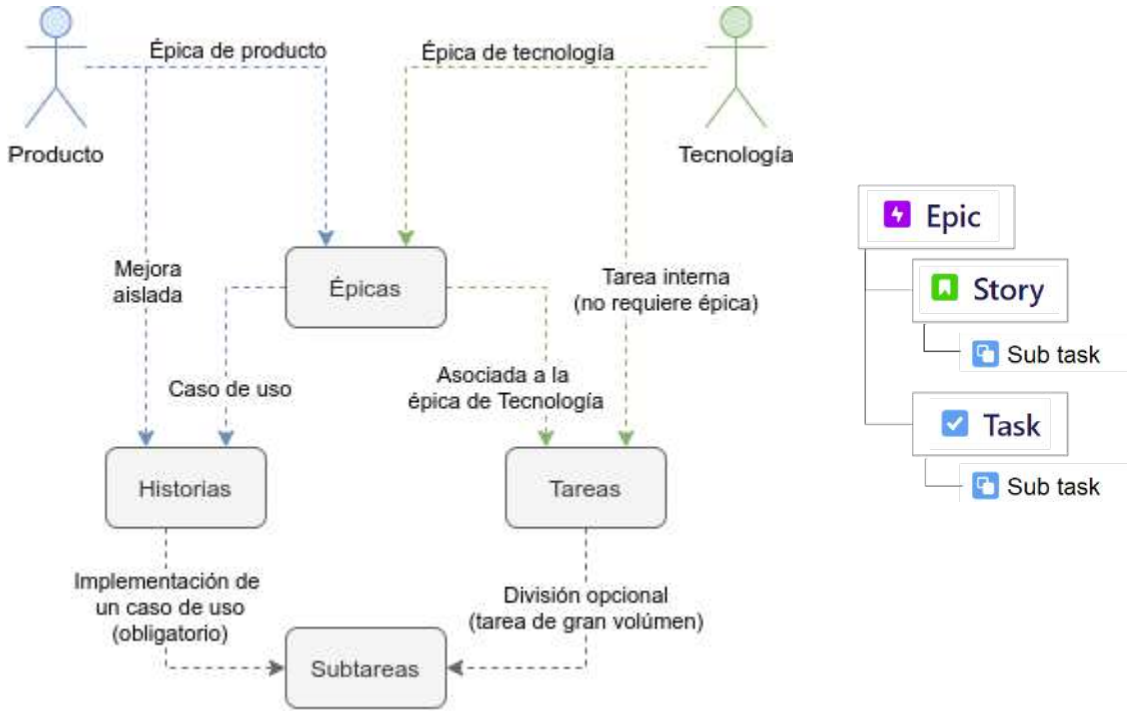



**Figura 3**. Proceso de creación de tareas

A continuación se explica el diagrama anterior, detallando la utilización y las causas de la creación de cada uno de los tipos de **Issues**:

- **Épicas:** una épica representa una funcionalidad transversal de la aplicación, la cual requiere de más de una iteración para su finalización. El proceso de creación distingue dos tipos de épicas:
  - **Épica de producto:** se asocia a un feature funcional de la plataforma orientado al modelo de negocio que se respalda y fundamenta en un documento de análisis donde se detalla el requisito funcional vinculado.
  - **Épica de tecnología:** nacen a partir de un requisito de desarrollo interno como pueden ser mejoras técnicas, documentación, testing, etc.
- **Historias:** una historia siempre tendrá asociada una épica y al igual que esta, refleja un feature funcional de la aplicación, pero con un grado de granularidad más pequeño. Su creación se dará por dos causas:
  - **Caso de uso:** recordando el ítem anterior, una épica de producto siempre responderá a un requisito funcional del sistema, el cual expone una cierta cantidad de casos. Cada caso de uso se ve representado por una historia.
  - **Mejoras:** una historia también se puede crear a partir de una mejora asociada a un flujo determinado o a la refactorización de cierto feature.
- **Tareas:** este tipo de issue hace referencia a tareas técnicas detectadas y solicitadas internamente por el equipo de desarrollo, como pueden ser: tareas de documentación, refactorización de componentes de la arquitectura según cierto estándar, migraciones, tareas de investigación, etc. Las tareas pueden tener una épica asociada (épica de tecnología) o bien crearse de manera huérfana si el issue es aislado e independiente.

  Como se detalla a continuación, si la tarea requiere un desarrollo considerable, la misma se puede dividir en subtareas más pequeñas y concretas, con el objetivo de lograr un desarrollo incremental, ordenado y también facilitar la depuración y los code reviews asociados.

- **Sub tareas:** las subtareas son el nivel de granularidad más pequeño en cuanto a los tipos de issues y como se definió anteriormente, representan tareas concretas de desarrollo. Son cargadas a partir de una historia o de una tarea padre que requiera ser dividida en múltiples tareas más pequeñas y concisas.

Como se nombró anteriormente, se optó por la implementación de un tablero de tipo **Kanban** donde cada issue siempre tiene un estado asociado. A continuación se puede observar el ciclo de vida que se definió para una tarea:

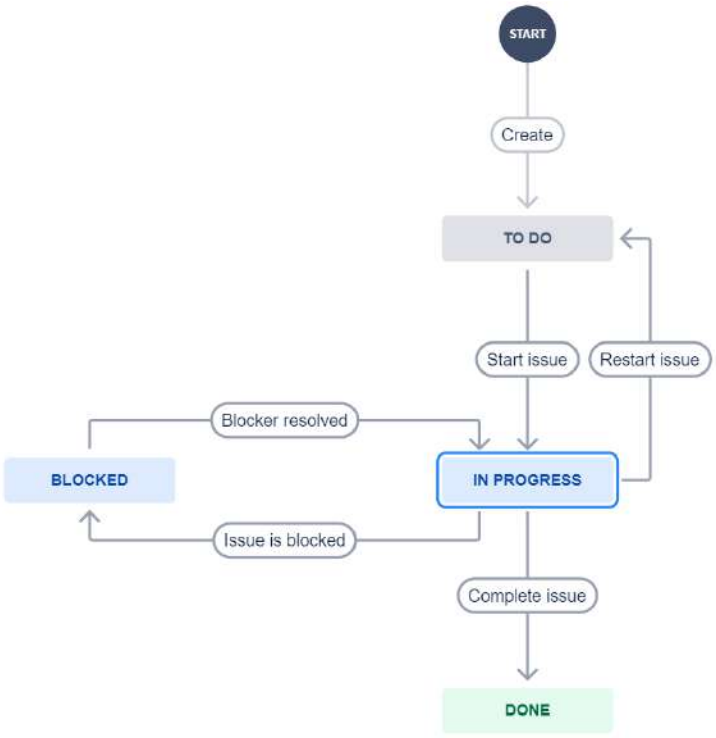



**Figura 4**. Workflow de un tarea

Es preciso mencionar que una tarea de desarrollo pasaría al estado “DONE” únicamente luego de haber realizado un test funcional y de aceptación basado en el flujo de interacción definido en el caso de uso asociado. Sumado a esto, se realizaron tests integrales de la plataforma durante el momento previo de realizar una demostración con el cliente. Si bien el equipo de desarrollo analizó la viabilidad de implementar tests unitarios automatizados, esta posibilidad fue descartada debido a la complejidad intrínseca del proyecto en cuanto a la necesidad de completitud y vitalidad del resto de las fases en los tiempos previstos.

Por otro lado, se promulgó desde un primer momento el trabajo ***distribuido*** y ***asíncrono*** de los integrantes del equipo, dado el contexto de pandemia en el cual se enmarcó gran parte del proyecto.

A continuación se puede observar el conjunto de épicas, tareas en el tablero y backlog, respectivamente:

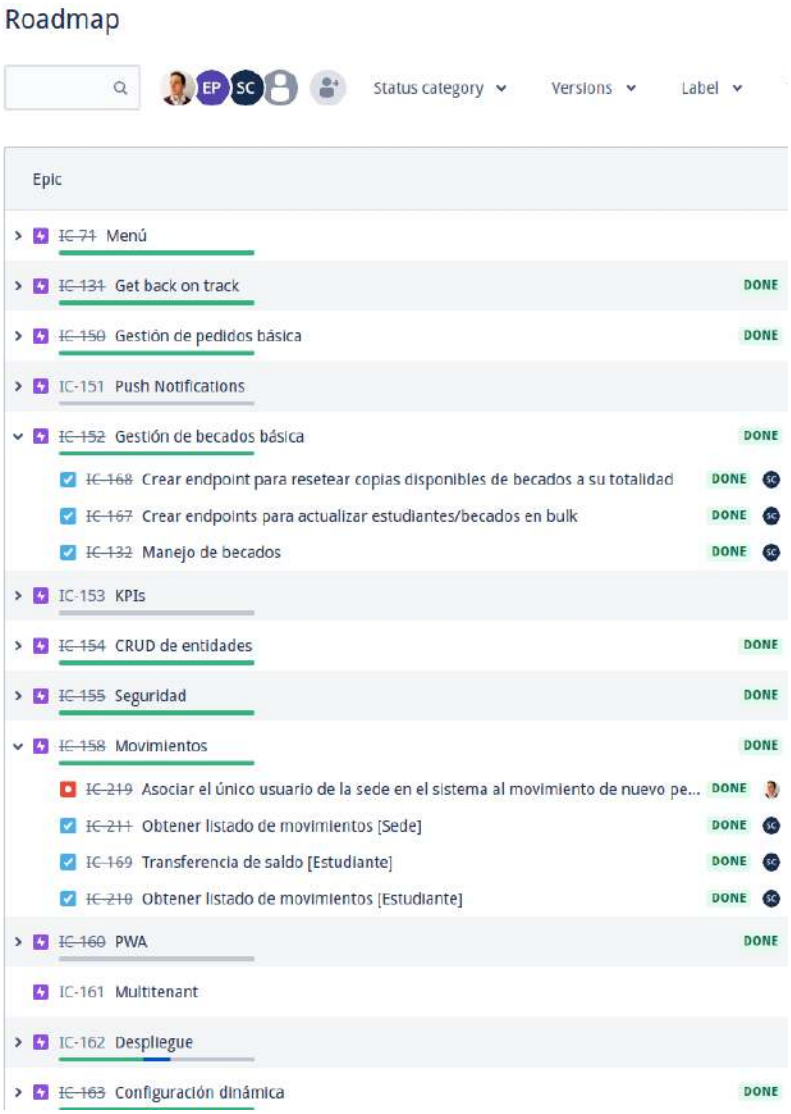



**Figura 5**. Épicas del proyecto

Cada **épica** representa un conjunto de historias de usuario y/o tareas a ejecutar hasta alcanzar su objetivo. A través de una vista global como la evidenciada fue sencillo para los miembros del proyecto identificar cuáles eran los próximos objetivos a perseguir estableciendo las prioridades necesarias.

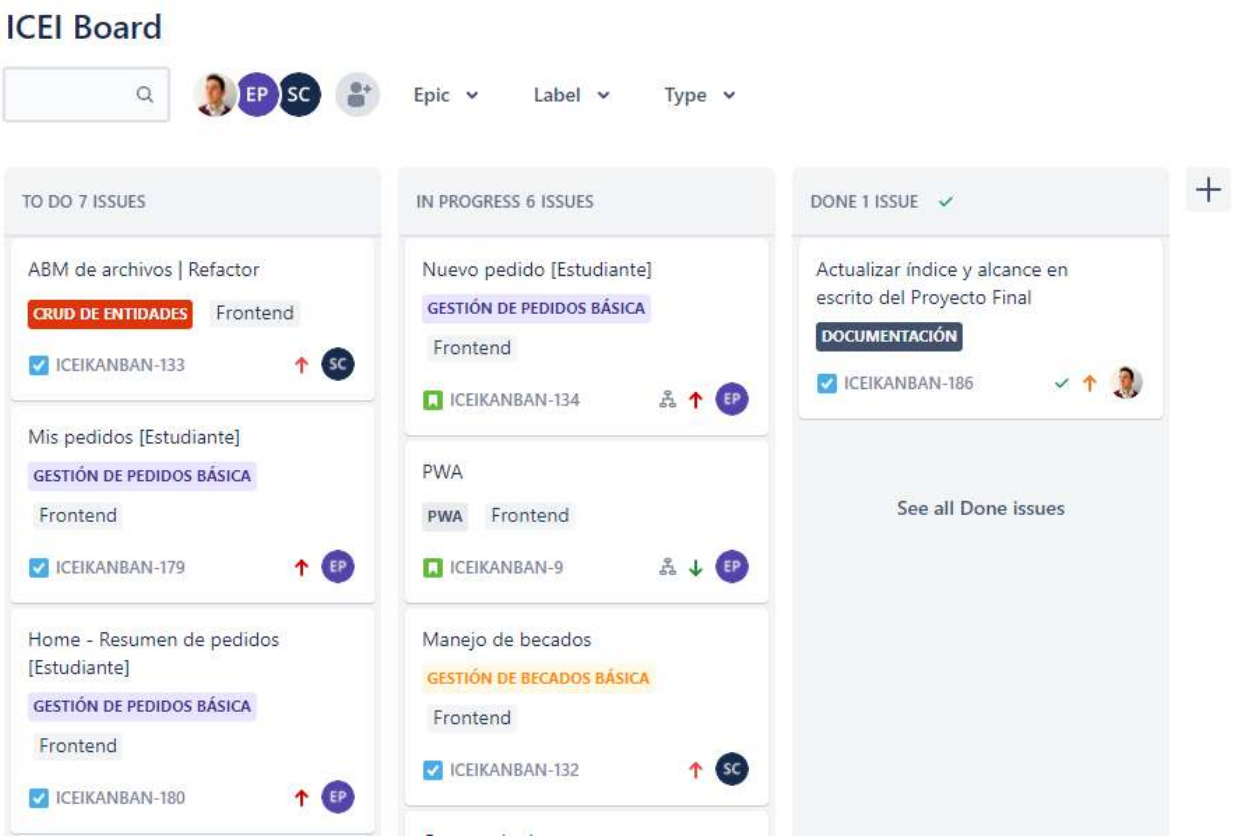



**Figura 6**. Tablero Kanban

El **tablero Kanban** representa un nuevo **punto de vista** de las mismas tareas que se aprecian en la figura 5, agrupadas allí bajo épicas. En el tablero Kanban sólo son visibles aquellas tareas que han sido extraídas del **backlog** y que han pasado a formar parte del tablero activo. De esta manera, el equipo pudo tener rápidamente acceso a cuáles eran las próximas tareas a ejecutar dado que las mismas ya habían sido depuradas y priorizadas para su ejecución.

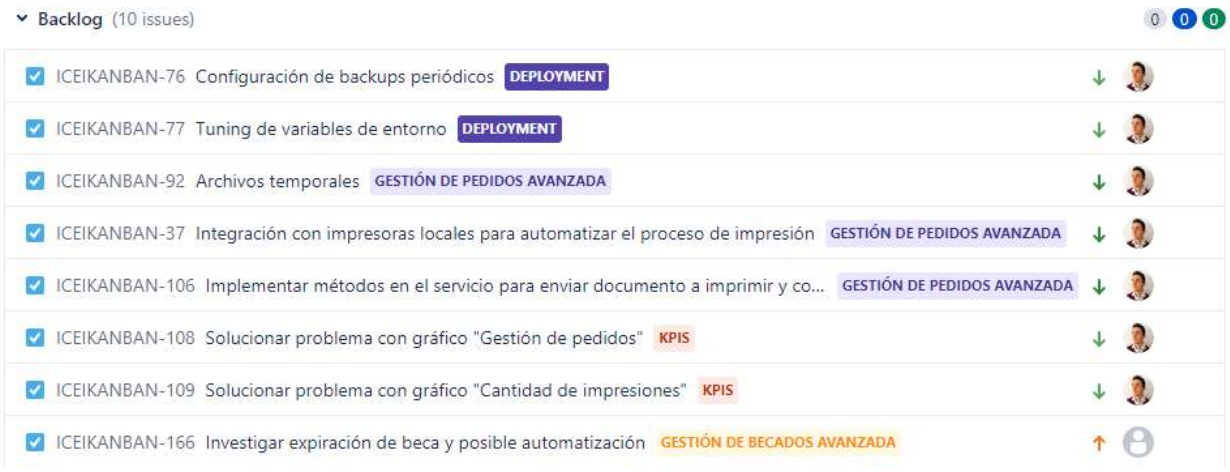



**Figura 7**. Backlog

El **backlog** es aquel conjunto de tareas que el equipo de desarrollo consideró necesarias para alcanzar el producto deseado, pero las cuales todavía se encontraban pendientes de ser depuradas y priorizadas. No todas estas tareas llegaron al tablero o sprint, sino que algunas fueron reformuladas, disgregadas en dos o más tareas más pequeñas o eliminadas si la idea inicial ya no era viable.

Por otro lado, durante el desarrollo del proyecto fueron relevados muchos requisitos que no eran estrictamente necesarios para la versión acordada con el CEI. Todas estas necesidades detectadas fueron organizadas en historias, tareas, épicas y releases posteriores a llevar a cabo. De esta manera, todo este trabajo pendiente fue quedando documentado para su futuro abordaje.

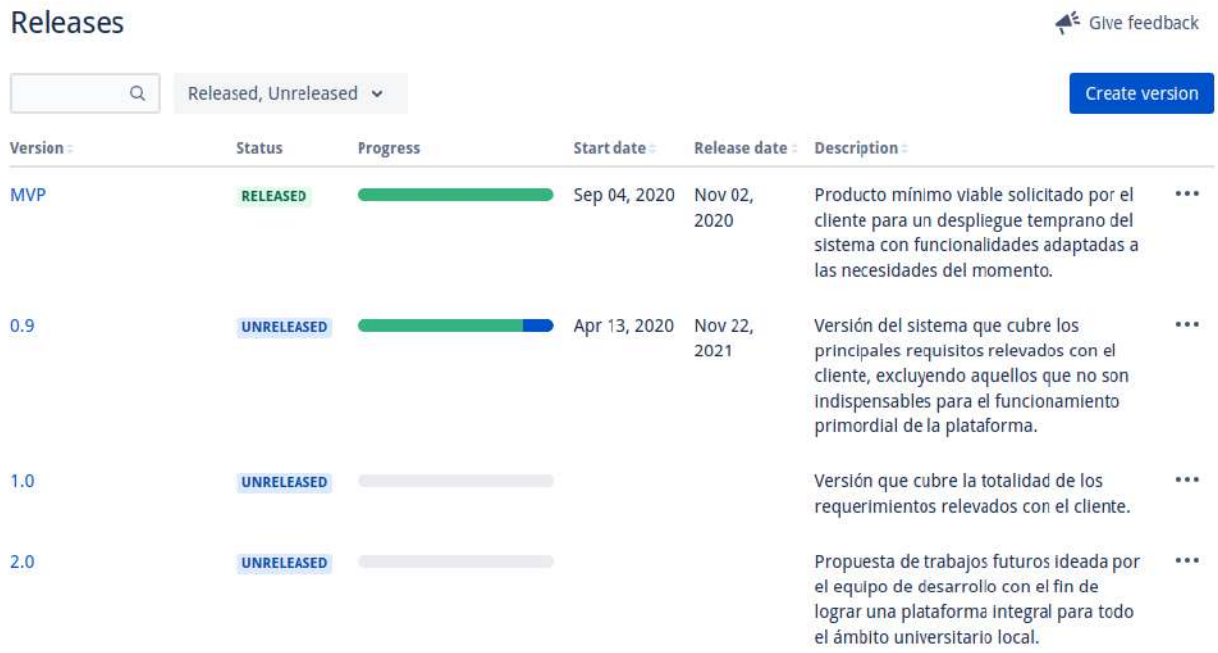



**Figura 8**. Lanzamientos planificados

### 5.1.3. Flujo de la metodología de trabajo

Como se mencionó anteriormente, debido a que los integrantes del proyecto contaban con una disponibilidad horaria limitada y sumado al contexto de pandemia en el que se desarrolló gran parte del proyecto, fue necesario contar con una gestión de la planificación organizada y formalizada, que delineara el flujo de trabajo a seguir. A continuación se muestra un diagrama con el formato de reuniones internas implementado:

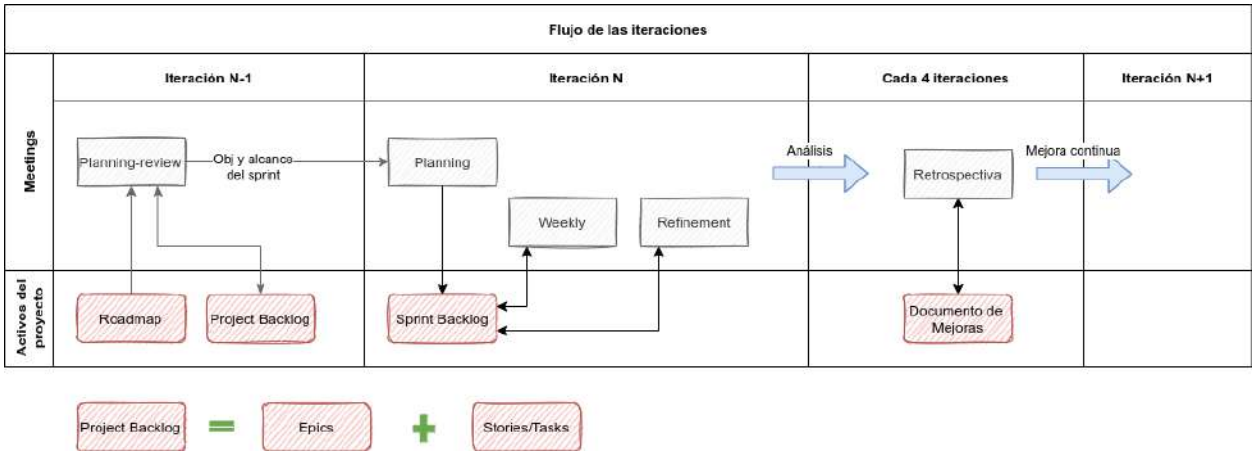



**Figura 9**. Flujo de la metodología de trabajo (adaptación de Scrum)

Cabe destacar que si bien se optó implementar un tablero de tipo Kanban, en donde por motivos de flexibilidad en los tiempos de entrega no se contaba con la entidad de Sprint como tal dentro de la herramienta, el flujo implementado sí consideró un grupo de tareas que engloban un objetivo en común y que su finalización se da en conjunto, lo cual se ve representado en el gráfico como una “Iteración”.

A continuación se explica el diagrama anterior, detallando cada una de las ceremonias llevadas a cabo en el proyecto:

- **Planning-review**: ceremonia llevada a cabo al finalizar la iteración actual con el objetivo no sólo de cerrar, inspeccionar y reveer el avance de la última iteración, sino también para determinar objetivos y alcances de la siguiente. En ella se efectúan las siguientes acciones:
    - Analizar y validar el Roadmap.
    - Cargar o actualizar las épicas, historias, tareas y/o subtareas.
    - Analizar y validar implementaciones técnicas para lograr definición y detalle en cada una de los issues.
    - Determinar el alcance y los objetivos de la siguiente iteración, priorizando las épicas e historias a incluir en concordancia con los requisitos funcionales del producto.
- **Planning**: reunión efectuada al comienzo de la iteración con el fin de seleccionar las tareas a realizar durante la misma, dividiendo la carga de trabajo entre los integrantes y especificando estimaciones en cada tarea. A continuación se detallan las actividades llevadas a cabo:
    - Revisar y asignar las historias y tareas priorizadas en la Planning-review.
    - Cada responsable estima el esfuerzo asociado a sus tareas.
    - Determinar viabilidad de la carga de trabajo de cada integrante.
- **Weekly**: reunión semanal cuyo objetivo es actualizar el avance del equipo. Cada integrante expone el trabajo realizado en los últimos 7 días, indicando si tuvo algún inconveniente durante su desarrollo y qué asignaciones tomará durante la próxima semana.
- **Retrospectiva**: ceremonia llevada a cabo de manera flexible, generalmente mensual, cuyo objetivo general es plantear las experiencias vividas en las últimas iteraciones (cómo fue la dinámica del trabajo, qué problemas existieron, qué asuntos se manejaron correctamente, etc.) identificando los aspectos tanto del proceso como del trabajo diario que funcionaron bien y los que no.

Como se mencionó anteriormente, cada iteración representa un conjunto de tareas con un objetivo de desarrollo en común, como lo puede ser la implementación de un módulo específico de la plataforma. A pesar de ello, la validación de lo desarrollado con el cliente no se realizó de manera estricta y metodológica luego de la finalización de cada iteración, sino que se recurrió a validarlo en reuniones bajo demanda, donde en cada una de ellas se realizaron demostraciones incrementales del sistema, según la cantidad de módulos funcionales implementados desde la última validación. Si bien la planificación original del equipo de desarrollo contaba con la posibilidad de validar con el cliente lo desarrollado en cada iteración de manera inmediata luego de su completitud (para evitar así la dificultad exponencial que representa realizar modificaciones ante la detección tardía de diferencias o errores en módulos funcionales), esto no fue posible por diferentes restricciones, imprevistos y el

contexto general del proyecto. Además, el proyecto sufrió picos de desarrollo intensivos y estacionales, en algunos casos muy notorios, lo que representó un desafío coordinar reuniones de este tipo de manera frecuente. Esta dificultad se vió contrarrestada por el hecho de que gran parte de la fase de elicitación y análisis no se realizó de manera iterativa e incremental, sino que se efectuó con una metodología de cascada, provocando que los requisitos funcionales primordiales identificados del sistema fueran validados en una etapa temprana del proyecto.

Como bien se mencionó, Scrum fue la metodología de trabajo en la cual se enmarcaron todas las tareas del proyecto. **Jira** fue la herramienta elegida para soportar las principales ceremonias y estrategias que propone este framework. En la sección anterior se pueden apreciar varias de las funcionalidades que ofrece la plataforma. Sin embargo, otras tecnologías también fueron utilizadas para suplir varias de las necesidades del equipo.

Complementaria a esta herramienta, **Google Meet** fue la plataforma elegida para llevar a cabo muchas de las ceremonias de la metodología implementada, tales como planificaciones, sincronización de avances, resolución de bloqueos, etc. El uso de esta herramienta se vió incrementado drásticamente durante el contexto de la pandemia. Las reuniones tuvieron como foco ser un punto de encuentro y puesta en común para debatir ideas, comparar lo planificado con lo ejecutado y decidir los próximos pasos a seguir.

Sumado a esto, muchos encuentros con el cliente fueron llevados a cabo de forma virtual. A través de los mismos se buscó identificar requerimientos, clarificar dudas y notificar sobre el avance del proyecto, entre otros.

### 5.2. Gestión de código

Para la evolución del código se optó por utilizar **Git** como control de versiones distribuido, potenciado a través del uso de repositorios privados de **GitHub**. A través de estas dos herramientas se pudo trabajar en forma distribuida y asíncrona, realizando ceremonias ocasionales de puesta en común para combinar (mergear) los avances sobre la rama principal de código. A continuación se detallan tres aspectos principales que conformaron las prácticas de gestión de código seguidas por el equipo:

- *Formato de commits*: se utilizó la especificación de **Conventional Commits** con el fin de simplificar y estandarizar la lectura del historial de cambios que se realizó en el desarrollo de la plataforma. Además, se utilizó la **Especificación del Versionado Semántico** (SemVer) con el objetivo de controlar y declarar explícitamente y de manera estandarizada los detalles y dependencias que estaban contenidas en cada incremento realizado al sistema informático.
- *Modelo de ramas*: con el fin de conseguir un desarrollo en paralelo de forma organizada se utilizó **GitFlow** como flujo de trabajo aplicado al repositorio Git de la plataforma.
- *Pull request:* se decidió implementar la práctica de **Pull Request** soportada nativamente por la plataforma de GitHub para asegurar la calidad del desarrollo validando con el resto del equipo los cambios introducidos a nivel código.

En la sección “Gestión de código” del anexo “Prácticas aplicadas en el desarrollo” es posible acceder a una descripción detallada del uso de los estándares utilizados y mencionados anteriormente.

## 5.3. Comunicación

### 5.3.1. Comunicación escrita

Más allá de la herramienta que permita la gestión y seguimiento de las tareas del proyecto, la comunicación diaria del equipo existirá de todas formas. Para poder llevarla a cabo, se decidió recurrir a **Slack**, la cual es una plataforma ampliamente extendida y que permite mantener varios hilos de conversación en paralelo, en pos de una mejor organización de los diálogos.

Para poder consultar dudas, sugerir nuevas ideas, debatir sobre temas en específico y otros tantos casos de uso se crearon diferentes **canales** que agruparon y permitieron satisfacer estas necesidades. De esta manera se posibilitó mantener intercambios entre los individuos del equipo relacionados a frontend, backend, arquitectura, infraestructura, requerimientos, elección y debate sobre tecnologías y algunos otros tópicos. A su vez, esta plataforma brinda la posibilidad de **integrarse con otras herramientas** utilizadas durante el desarrollo. Algunos ejemplos de esta capacidad es la conexión con Jira que permite poder ver la evolución de las tareas en el canal pertinente o la integración con GitHub para conocer los cambios producidos en los repositorios de código, los cuales se visualizan también en su canal específico. En la siguiente imagen se puede observar cómo fue configurada la plataforma:

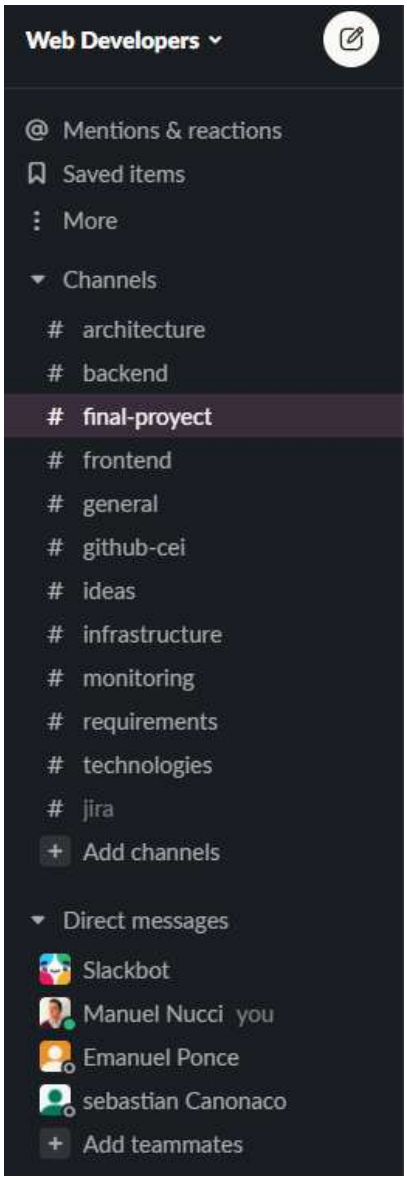



**Figura 10**. Canales en Slack

### 5.3.2. Comunicación entre capas

La práctica o filosofía adoptada para el desarrollo de la API que se necesitó diseñar e implementar durante el proyecto es **contract-first**. A través de esta práctica los contratos entre las partes son acordados en primera instancia para luego poder independizar al consumidor del productor del recurso y acelerar el desarrollo.

Las ventajas de adherirse a esta filosofía fueron las siguientes:

- **Los equipos pueden trabajar en paralelo**: dado que el desarrollo se basa en el contrato, el proveedor y el consumidor del servicio (desarrolladores de backend y frontend

respectivamente) tienen claro el enfoque y los detalles. De esta forma, cada uno puede desarrollar su implementación al mismo tiempo.
- **Los equipos saben qué esperar**: dado que el desarrollo se basa en el contrato, cada parte tiene una idea clara de las expectativas del otro. Como resultado, si el testing cruzado no es posible debido a las diferentes velocidades de los equipos, software que simule el comportamiento de la otra parte puede ser utilizado para efectuar las pruebas necesarias, siempre basado en el contrato definido.
- **Compatibilidad multiplataforma**: el proveedor y consumidor del servicio pueden utilizar diferentes tecnologías dado que los parámetros del servicio dependen exclusivamente del contrato.
- **Reutilización de esquemas**: los esquemas que son utilizados para definir el contrato del servicio pueden ser reutilizados en otros servicios que hagan uso de los mismos.

Para poder tener una comunicación clara respecto a los contratos definidos se hizo uso del estándar **OpenAPI Specification (OAS) 3**, herramienta de facto en la industria a día de hoy. La especificación de uso de este estándar se encuentra en la sección “Documentación de la API” del anexo “Prácticas aplicadas en el desarrollo”.

## 5.4. Integración continua

Para facilitar la disponibilización del backend al momento de realizar las integraciones con el frontend se decidió utilizar la práctica de **integración continua**. A partir de la misma se pudo acelerar el proceso de desarrollo eliminando aquellas tareas repetitivas que entorpecían la tarea diaria del equipo.

**GitHub Actions** fue la herramienta seleccionada como ejecutor de workflows para automatizar el proceso y eliminar los inconvenientes identificados. En la sección “Integración continua” del anexo “Prácticas aplicadas en el desarrollo” se puede obtener más detalles sobre el flujo implementado.

# 6. Producto

Junto con los pilares de **Proyecto** y **Proceso** expuestos anteriormente, aparece finalmente el último de ellos: el **Producto** a desarrollar. El producto o entregable cubierto por esta primera versión consta de una actualización del esquema de procesos actual que posee el CEI, sumado a una plataforma que basada en el mismo permite satisfacer las necesidades identificadas.

En esta sección se detallarán todas las acciones llevadas a cabo para poder relevar las características del producto deseado, identificar los requerimientos a satisfacer y las restricciones existentes en el contexto, diseñar el nuevo proceso de modelo de negocio y analizar alternativas para luego diseñar la arquitectura de solución de aquellos componentes que sustentan al mismo.

## 6.1. Análisis

Durante la fase de análisis, la atención se centró en definir los requerimientos del sistema, es decir, descripciones detalladas de lo que la plataforma debe hacer, el servicio que debe ofrecer y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los usuarios involucrados, por lo que su definición parte del proceso previo de elicitación, mediante el cual se intenta descubrir y entender cómo funciona el proceso actual, y cuáles son las problemáticas y necesidades de los distintos tipos de usuarios.

### 6.1.1. Proceso de elicitación

#### 6.1.1.1. Contexto y objetivos

La primera fase asociada a la etapa de análisis, se centró en el descubrimiento de las necesidades de los usuarios involucrados, mediante la utilización de diversas herramientas y estrategias, con el objetivo de comprender la forma en que interactúan en el contexto laboral con sus procesos y sistemas de información actuales.

Para esta tarea se llevó a cabo un proceso de elicitación en donde se procuró disminuir la brecha existente entre los usuarios, que tienen necesidades y requieren soluciones a sus problemas, pero que no poseen los conocimientos para especificar los requisitos, y el equipo de desarrollo, que sí tiene dichos conocimientos técnicos pero desconoce en profundidad el contexto y las circunstancias del usuario, haciendo hincapié en establecer una buena comunicación entre ambas partes y en lograr un conocimiento extensivo del mundo del cliente. Cabe destacar que una ventaja que se tuvo en este aspecto, fue que el equipo de desarrollo, además de ocupar dicho rol, también empleó un papel de stakeholder, al estar integrado por un grupo de estudiantes que de manera frecuente utilizaban los servicios del CEI. Esto provocó que cada uno de los miembros del equipo expresara sus problemas y necesidades como estudiantes y clientes de los servicios de impresión.

A través del proceso de elicitación se recuperaron todas las definiciones necesarias y luego la parte técnica quedó a cargo del equipo para elegir las mejores arquitecturas, diseños, productos y servicios externos necesarios que permitieran cumplir con las expectativas del CEI y a su vez ser parte de una solución escalable, mantenible y resiliente. Esto último estuvo condicionado principalmente

por el hecho de que del lado del cliente no existió ningún referente técnico que pudiera contribuir con este aspecto.

Los objetivos de la elicitación de requerimientos fueron:

- Identificar y priorizar las fuentes de información.
- Adquirir conocimiento de los procesos y sistemas de información actuales.
- Identificar el producto que se desea obtener en términos generales.
- Una vez conceptualizada la idea, descomponer el producto en funcionalidades más concisas a fin de determinar que requiere el sistema.
- Identificar los actores que harán uso del sistema, sus casos de uso y cuáles serán las interfaces a través de las cuales interactuarán con el mismo.
- Entender cuáles son los requerimientos no funcionales más importantes que deberá soportar la Arquitectura de la solución, estableciendo un trade-off entre ellos.
- Identificar las restricciones y limitantes a las cuales estará expuesto el sistema, a fin de minimizar su impacto y garantizar que se alcancen las metas deseadas.
- Trazar un mapa del ciclo de vida de la aplicación, cómo será su mantenimiento y extensión, y quiénes serán los encargados de llevar a cabo esto.

#### 6.1.1.2. Identificación de stakeholders

Los stakeholders o entidades involucrados en el proyecto son:

- ❏ **Presidente del CEI**: cliente del producto y conexión principal o *focal point* para elicitación de requerimientos sobre el mismo.
- ❏ **Empleado del CEI**: becado por parte del CEI que desempeña funciones dentro de la fotocopiadora. Es la persona que día a día gestiona los pedidos por parte de los estudiantes, mantiene actualizados los repositorios de información y aclara cualquier tipo de dudas que puedan surgir por parte de los estudiantes.
- ❏ **Estudiante de la facultad**: principal usuario del proceso actual de impresión. Utiliza el sistema para imprimir archivos personales o aquellos disponibles por parte del CEI.
- ❏ **Becado**: estudiante que temporalmente posee una beca otorgada por el CEI. La misma le otorga una cierta cantidad de copias disponibles que puede utilizar en un período de tiempo asignado (es decir, mientras dure la beca).
- ❏ **Cátedra**: órgano de la facultad formado por distintos profesores y asociado a una asignatura. Actualmente la cátedra brinda al CEI el material que desea que esté disponible a través de la plataforma. Muchas veces las cátedras utilizan repositorios personales para tal fin, generando que el estudiante deba llevar este material al sistema del CEI para poder imprimirlo.
- ❏ **Equipo de desarrollo**: grupo de estudiantes encargado de llevar a cabo la transformación digital del proceso de impresión actual, identificando los puntos de dolor actuales y diseñando la nueva plataforma para mitigar los mismos.

#### 6.1.1.3. Técnicas

Las estrategias llevadas a cabo durante el proceso de elicitación se basaron principalmente en la utilización de **métodos interactivos**, como lo son las entrevistas, en donde se tuvo un alto grado de interacción con el cliente. En conjunto con este enfoque, se utilizaron **métodos discretos** para la recopilación de información inicial, como investigaciones, observaciones del comportamiento de los encargados de la toma de decisiones, así como observaciones in situ del entorno físico. Estas últimas estrategias presentaron un enfoque complementario interesante, ya que si bien no requirieron el mismo grado de interacción con el usuario como las entrevistas, a diferencia de estas, no generan perturbaciones en los stakeholders ni en el ambiente de trabajo, por lo que se extraen resultados realistas que omiten todo tipo de especulación, malos entendidos o intereses que puedan tener los relatos de los usuarios.

A continuación se detalla las estrategias de recopilación de datos llevadas a cabo a partir de la utilización de una metodología mixta entre métodos interactivos y discretos:

1. **Entrevistas** con los principales stakeholders: se coordinaron diferentes encuentros con propósitos específicos, en los cuales se realizaron una serie de preguntas a contestar. Se comenzó con aquellos individuos que tienen una visión más global sobre el producto deseado (el Presidente del CEI, por ejemplo) y, luego, se avanzó con otras reuniones con personas que están interactuando en el día a día con los procesos (empleados del centro de copiado y becados por nombrar algunos ejemplo). En todas ellas se trató de entender cuáles eran las opiniones del estado actual de los procesos y las expectativas de lo nuevo por desarrollarse.
2. **Investigaciones**: se realizaron investigaciones personales sobre el sistema actual de impresiones del CEI, con el fin de describir y analizar con mayor profundidad las problemáticas planteadas por los stakeholders asociadas al proceso de trabajo.
3. **Observaciones del comportamiento de los encargados de la toma de decisiones**: gran parte de la comprensión del proceso actual fue gracias a la observación directa del funcionamiento del CEI. A través de esta inmersión se pudieron registrar varios inputs útiles para etapas posteriores.
4. **Observaciones del entorno físico**: además de observar el comportamiento de los encargados del CEI, se realizaron observaciones in situ del entorno físico, con el fin de determinar cualitativa y cuantitativamente los recursos actuales con los que se contaban. Esto permitió brindar un panorama de las capacidades de hardware disponibles que determinarían la viabilidad del sistema a implementar.

#### 6.1.1.4. Resultados

Luego de aplicar las diferentes estrategias de elicitación mencionadas anteriormente, se llegó a identificar los principales problemas y expectativas para cada uno de los actores. Las mismas se encuentran reunidas en la siguiente tabla:

| Stakeholder | Rol | Problemas | Expectativas |
|---|---|---|---|
| Presidente del CEI | Líder del órgano que representa al grupo estudiantil de la Facultad.<br>Busca facilitar y mejorar la experiencia diaria de todos los alumnos. | 1.1 Ineficiencia del proceso (velocidad sobre todo)<br>1.2 Picos estacionales<br>1.3 Desactualización de material<br>1.4 Alto riesgo de error en gestión de pedidos<br>1.5 Ineficiencia del proceso de asignación de copias a los becados. | Digitalizar y mejorar el proceso para brindar un servicio de mayor calidad.<br>Disminuir al mínimo el riesgo en la gestión de pedidos.<br>Modernizar el servicio del CEI a fin de entrar en la nueva economía 4.0 y mejorar la experiencia de los usuarios.<br>Mejorar la asignación de copias a los becados. |
| Empleado del CEI | Integrante del grupo de representantes de los estudiantes.<br>Entre sus funciones una de ellas es brindar el servicio de fotocopiado e impresión para los alumnos. | 2.1 Tareas repetitivas, tales como registro de configuración de pedidos, subida de material, cobro de pedidos, etc.<br>2.2 Picos de trabajo en períodos específicos del cuatrimestre.<br>2.3 Riesgo de errores en manipulación de pedidos y dinero. | Automatizar al máximo tareas repetitivas.<br>Delegar al sistema la carga de solicitudes durante picos estacionales.<br>Disminuir la tasa de preguntas sobre material, costos, etc.<br>Delegar al sistema el cálculo de totales y gestión de los pedidos. |
| Estudiante | Alumno de la facultad que realiza uso de los servicios que brinda el CEI, entre ellos el servicio de fotocopiado e impresión. | 3.1 Proceso engorroso de encargo de pedidos: colas, asistir presencialmente a la fotocopiadora.<br>3.2 Errores en la emisión de pedidos.<br>3.3 Desconocimiento de cuándo están listos para retirar los pedidos.<br>3.4 Material desactualizado en el repositorio.<br>3.5 Contenidos inconsistentes que | Poder llevar a cabo todo el proceso a través de un proceso digital, convirtiéndolo en un self-service.<br>Disminuir el tiempo invertido en el proceso al mínimo.<br>Trackear en tiempo real el estado de los pedidos.<br>Disminuir la tasa de errores generada por malentendidos. |

| | | | |
|---|---|---|---|
| | | dependen de la sede a la que se concurra. | |
| Becado | Estudiantes que gozan de una beca otorgada por el CEI que les permite contar con copias gratuitas en el servicio. | 4.1 Desconocimiento de la cantidad de copias disponibles (algunos incluso desconocen directamente del beneficio). | Contar con un sistema con el que puedan acceder al servicio y, a su vez, hacer uso del beneficio otorgado. |
| Cátedra | Cada una de las materias que se dictan en la facultad. Integrada por un grupo de docentes que utilizan diverso material, el cual es disponibilizado muchas veces a través del servicio del CEI. | 5.1 La gestión del material a disponibilizar a los estudiantes requiere de una interacción innecesaria con el CEI. | Contar con un portal desde el cual poder gestionar todo el material a estar disponible en el servicio del CEI. |

### 6.1.2. Proceso de negocio actual

A partir de las distintas estrategias de elicitación llevadas a cabo, destacando principalmente las entrevistas y las observaciones in situ del comportamiento de los encargados de la toma de decisiones, fue posible confeccionar el **esquema de procesos** que rige actualmente el funcionamiento del CEI. A continuación se presenta el mismo, tomando como **punto de vista** primero al **estudiante** y luego al **empleado** de la fotocopiadora.

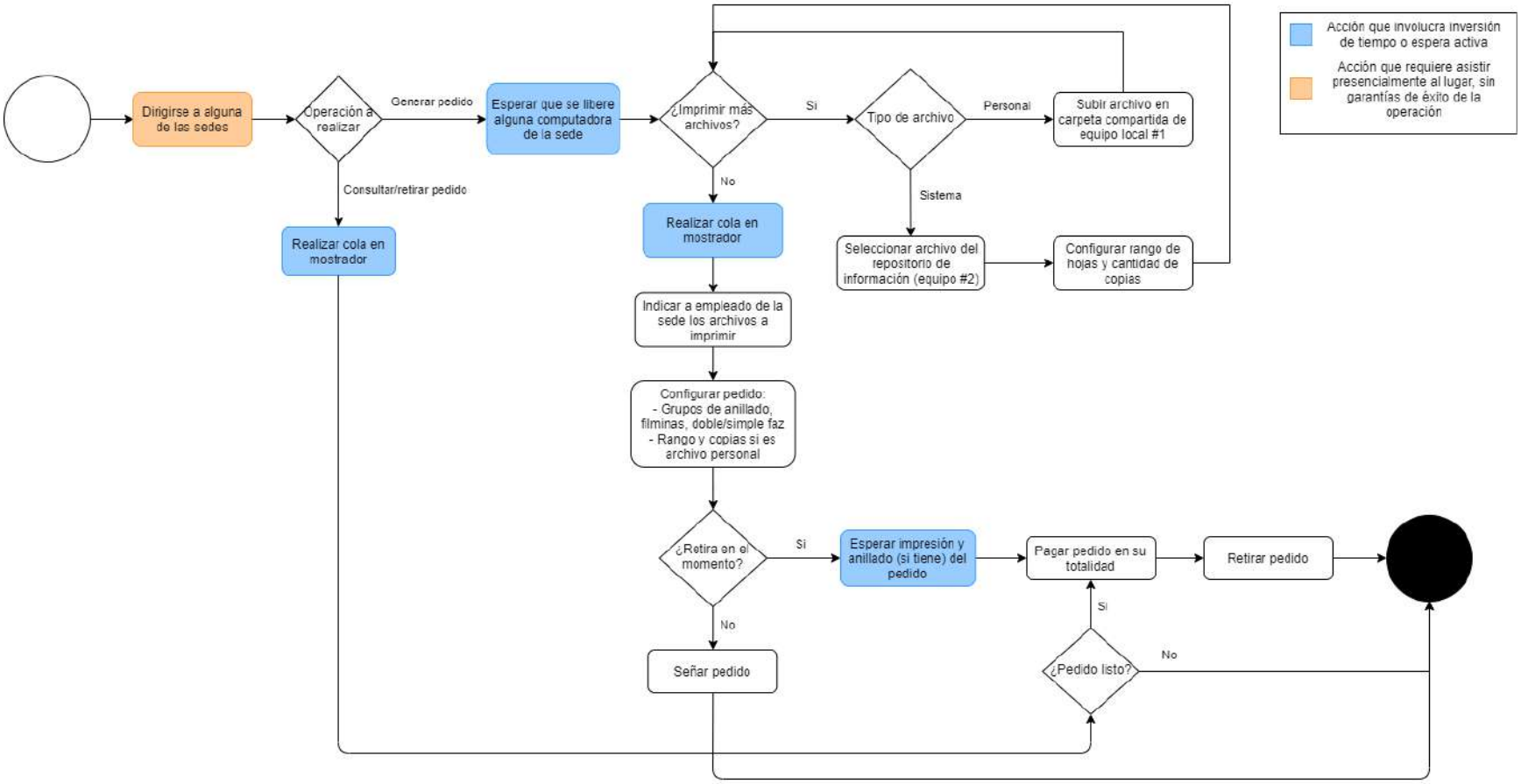



**Figura 11**. Flujo actual del estudiante para utilizar el servicio de impresión del CEI

Se aprecia que existen muchos pasos en el flujo que requieren de una **espera activa**. Esto equivale a decir que el estudiante debe estar invirtiendo parte de su tiempo sin poder ejecutar el próximo paso de su flujo hasta tanto otro actor destrabe su accionar.

El diseño del nuevo proceso tuvo en cuenta estos nodos como el principal foco para mejorar dicho flujo:

1. El usuario debe dirigirse a la sede para efectuar ***cualquier*** tipo de operación, incluso si la misma involucra una simple consulta.
2. En cualquiera de las operaciones a realizar por el usuario, el mismo siempre debe realizar una **cola** y **esperar** que el recurso o actor al que desea acceder se libere. Esto se ve acrecentado si el usuario realiza varias operaciones en una visita al centro de copiado (imprimir archivos personales y de sistema, consultar o emitir pedido en mostrador, etc.).
3. En caso de retirar el pedido en el momento, el estudiante debe **esperar activamente** su impresión y posterior anillado (si fuera necesario). Esto inhabilita el poder atender a otros usuarios en la cola dado que la práctica que usualmente siguen los empleados es operar de a un usuario por vez, a fin de no mezclar o generar confusión por su parte. El tiempo que demora la impresión es **tiempo muerto** que también podría ser evitado.

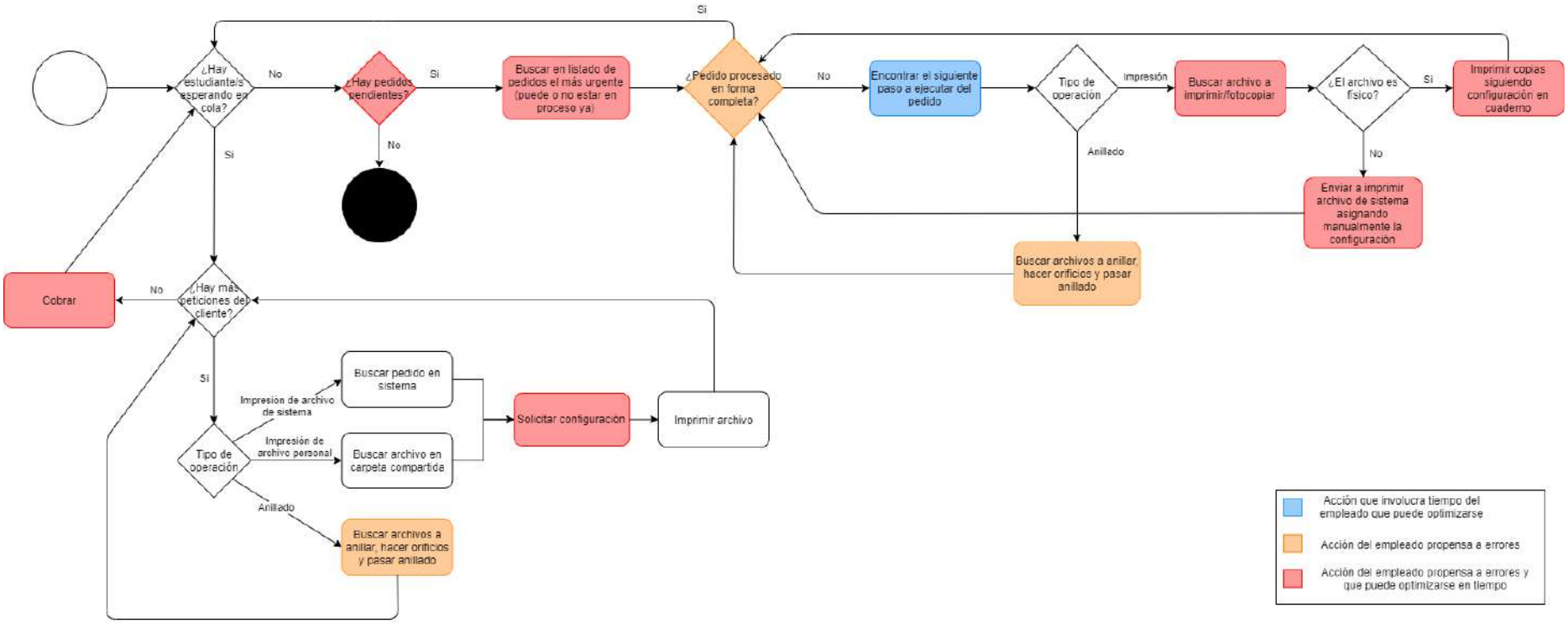



**Figura 12**. Flujo actual del empleado de la sede para brindar los servicios

En el caso del empleado de la sede, si bien no existen esperas activas, lo que ahora se presenta es una multitud de pasos propensos a **errores** dada la **intervención manual** necesaria. Muchas de estas tareas repetitivas son incluso optimizables en tiempo, posiblemente a través de una solución tecnológica:

1. Verificar si existen pedidos pendientes. El empleado podría omitir alguno que deba ser procesado.
2. Procesar pedidos pendientes en un orden que no respeta la urgencia o antigüedad de los mismos. Esto pudiera afectar el orden de procesamiento de los mismos y retrasar la gestión de un pedido que tiene mayor prioridad que otros.
3. El procesamiento de un pedido está atado a varios factores de error:
    a. Omitir la impresión o anillado de algún elemento del pedido.
    b. Error en la configuración de impresión de un pedido.
    c. Error en el anillado, ya sea por seleccionar copias equivocadas o por no respetar el orden deseado.
4. El cobro de pedidos es un factor más de error al tener que manipular dinero muchas veces fraccionario o de baja denominación. Esto obliga a depender siempre de tener “cambio” en la sede y no cometer fallas en la manipulación del mismo.

La conclusión de este análisis arroja que el proceso actual **no es escalable** y tiene una fuerte componente manual en toda su aplicación, la cual es propensa a multitud de errores humanos. Todos estos problemas identificados fueron atacados en la etapa de diseño para proponer una modificación y mejora al mismo, la cual viene acompañada de la implementación de un sistema que la sustente.

### 6.1.3. Modelado de requerimientos

Luego del proceso de elicitación, la cual fue una primera fase de adquisición de conocimiento y contexto de las problemáticas y necesidades de los stakeholders involucrados, se procedió a analizar en profundidad todos los hechos y datos recolectados, para transformar los requisitos de usuarios, que son requerimientos abstractos de alto nivel y declarados en lenguaje natural, en requisitos del sistema, que a diferencia de los anteriores, establecen detalle en las funciones, servicios y restricciones operativas del sistema a implementar.

Esta diferenciación en los niveles de especificación del sistema fueron de utilidad debido a que comunican la información recopilada a diferentes tipos de lectores, lo que permitió realizar las validaciones en la elaboración de los mismos a cada stakeholder asociado según sea su interés y asociación con el proyecto.

Para llevar a cabo esta especificación es necesario identificar tres tipos de requerimientos que condicionarán el desarrollo y comportamiento del aplicativo:

1. **Requerimientos funcionales**: son las declaraciones de los servicios que debe proveer el sistema y la manera en la que debe reaccionar a entradas particulares. Describen de manera detallada las funcionalidades que el sistema debe soportar y que los stakeholders perciben directamente. Entre ellas podemos nombrar como ejemplos: tener una sección para administración de usuarios, poder gestionar pedidos, etc.

   Este tipo de requerimientos se destacan por ser los más fáciles de identificar puesto que surgen en su mayoría de los deseos del cliente y su visión del producto.

2. **Requerimientos no funcionales (atributos de calidad)**: los atributos de calidad son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste. Estas restricciones o propiedades afectan a todo el sistema de manera transversal.

   La importancia en su detección y definición radica en que generalmente son más críticos que los requisitos funcionales, ya que si bien los usuarios pueden adaptarse alrededor de una función del sistema que cumple de manera parcial con sus necesidades, el incumplimiento de un atributo de calidad puede significar que el sistema en su totalidad sea inutilizable.

   Este tipo de requisitos surgen a partir de las necesidades intrínsecas del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas o hardware, o a factores externos como regulaciones de seguridad o legislaciones sobre privacidad.

   Como ejemplos tenemos: el sistema debe ser seguro, rápido, intuitivo en su uso, etc.

3. **Restricciones**: son todas aquellas restricciones que deben tenerse en cuenta a la hora de diseñar la solución y que condicionan fuertemente tanto un requerimiento funcional como no

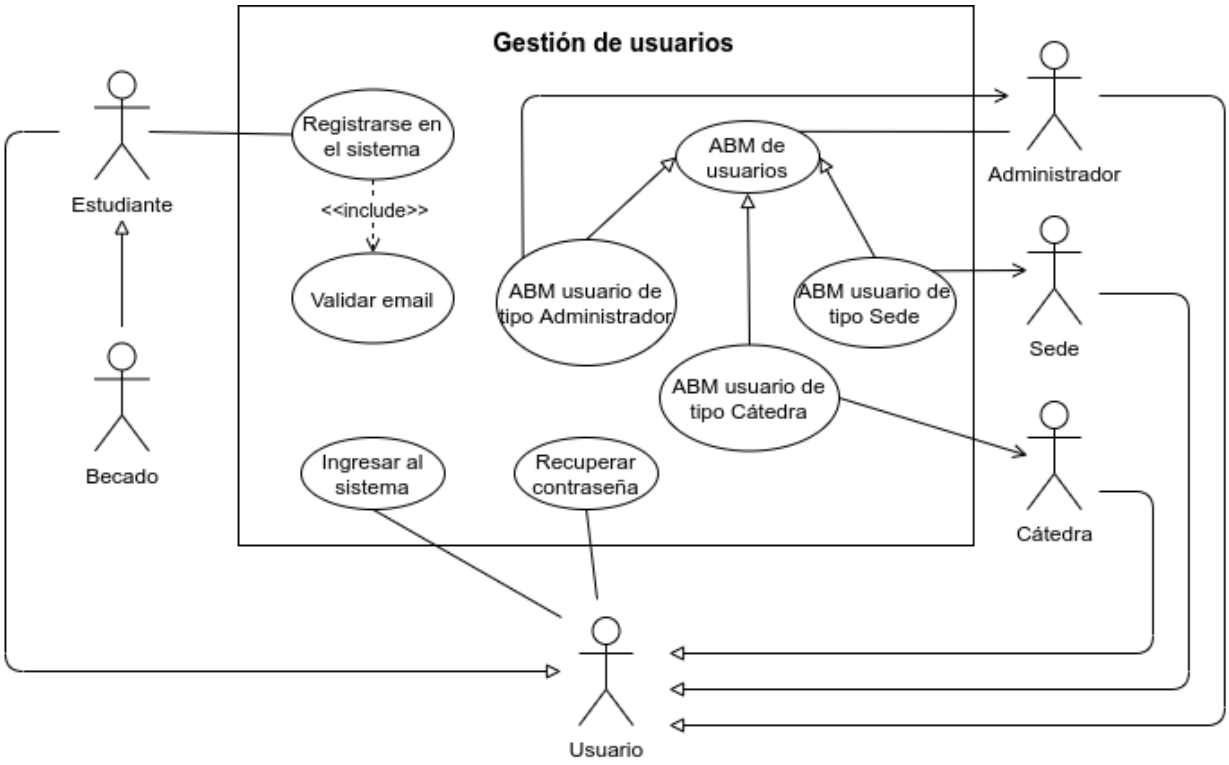
funcional. Por ejemplo, los usuarios en su mayoría tendrán conexión de internet lenta al sistema dado que utilizarán redes 3G.

El conjunto de estos tres grupos conforma los **drivers de arquitectura**, los cuales condicionarán todas las decisiones que deberán ser tomadas durante la etapa de diseño de la solución.

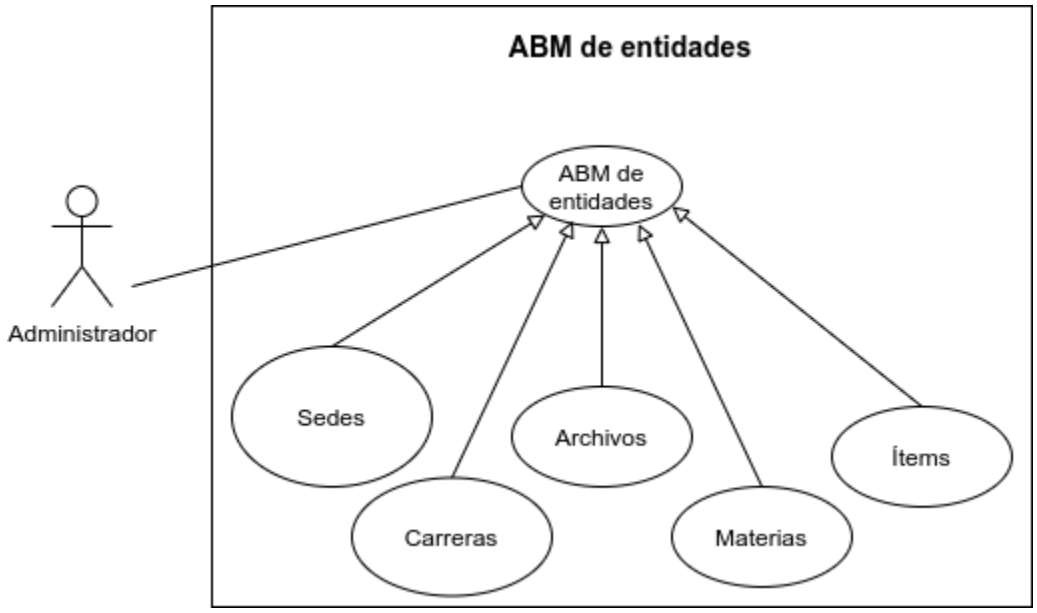
#### 6.1.3.1. Requerimientos funcionales

A partir del análisis de los resultados de las elicitaciones llevadas a cabo se identificaron los principales requisitos funcionales que debía proveer el sistema. Los mismos son listados a continuación:

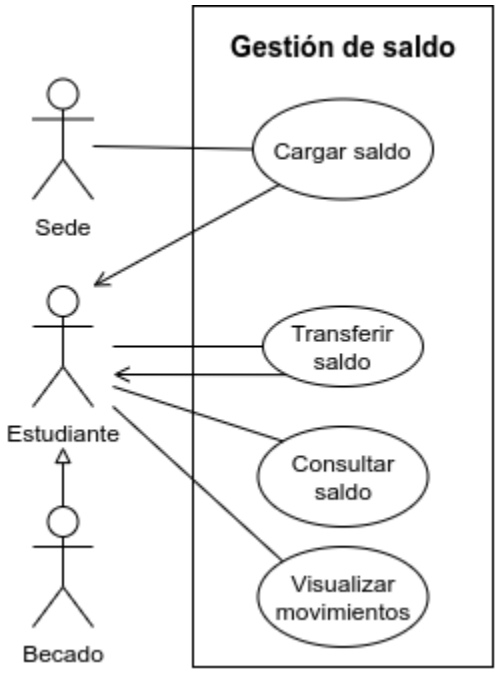
- Módulo de gestión de usuarios, flujo de registración (con validación de email), login y recuperación de contraseña:



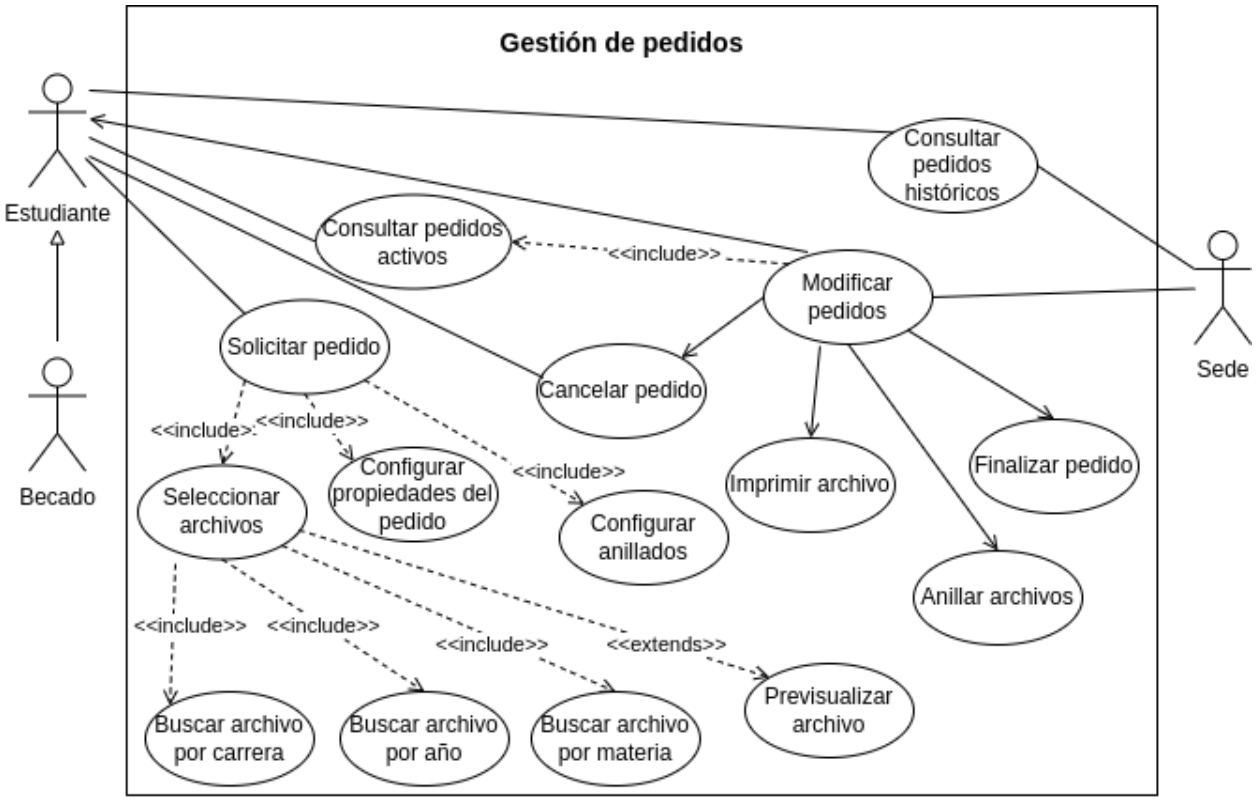

- Módulos de ABM para las siguientes entidades:
  - Sedes
  - Carreras
  - Materias
  - Archivos
  - Ítems



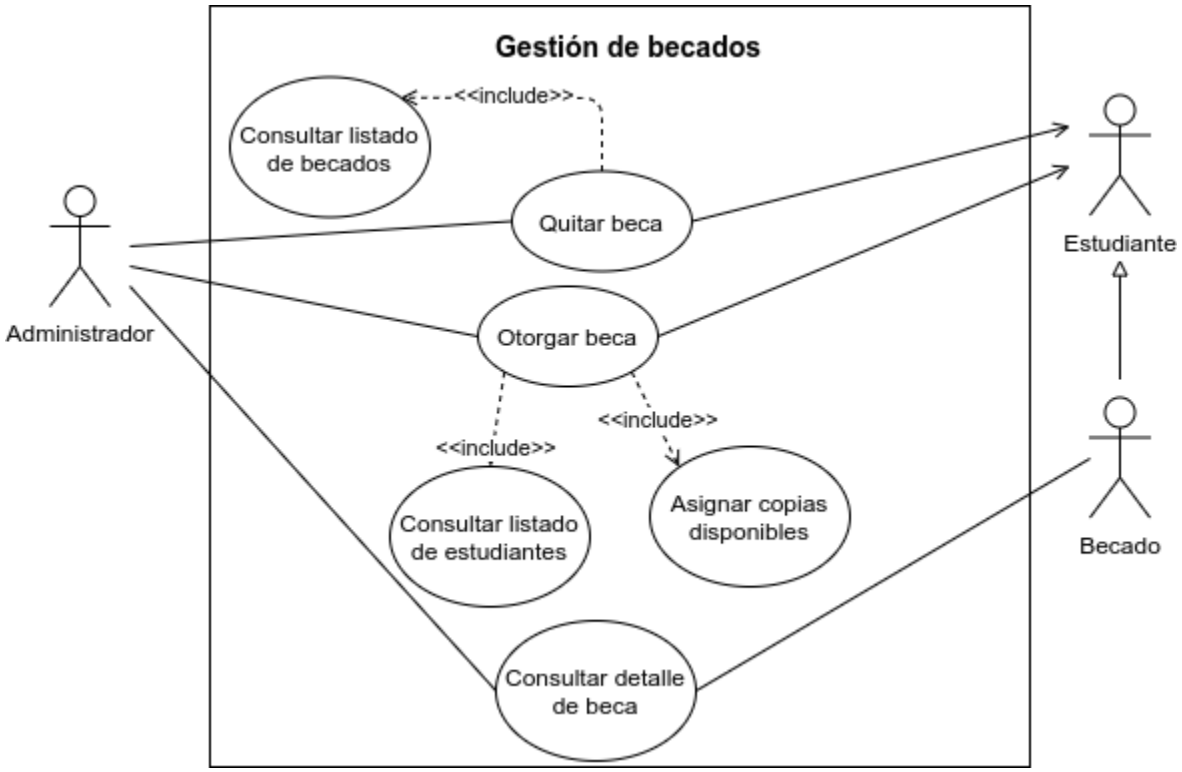

- Módulo asociado a la carga, gestión y transferencia de saldo y visualización de movimientos.



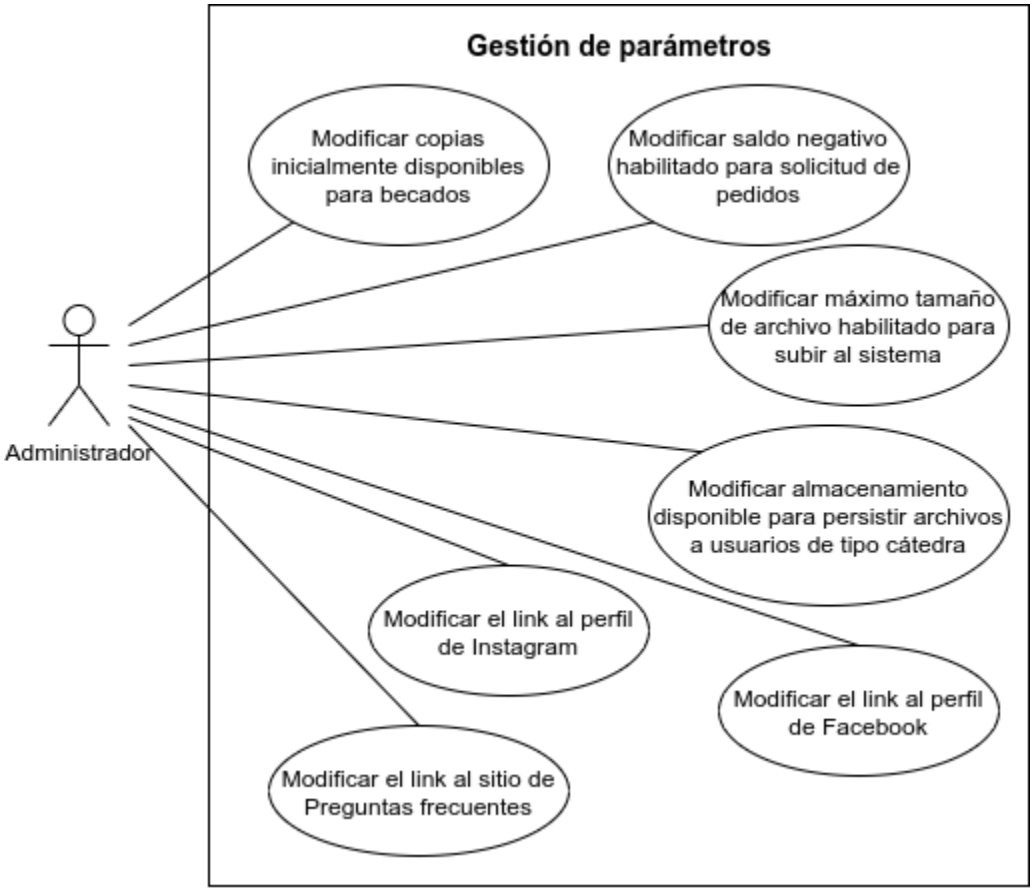

- Módulo asociado a la emisión, seguimiento y gestión de pedidos.



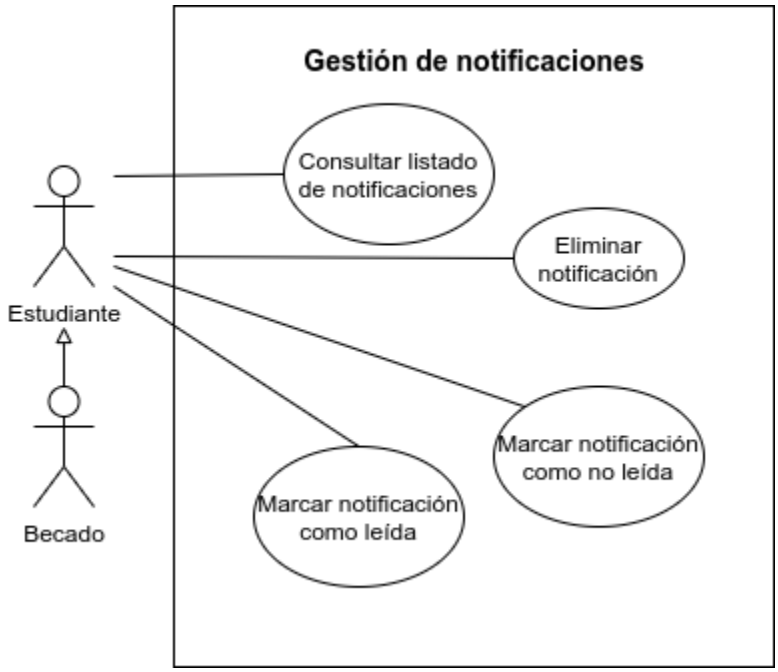

- Módulo de gestión y administración de becados.



- Parametrización del sistema para soportar comportamiento dinámico en las siguientes funcionalidades:
    - Almacenamiento inicialmente disponible para los archivos persistidos por los usuarios de tipo cátedra
    - Copias inicialmente disponibles para los usuarios becados
    - Máximo tamaño de archivo habilitado para subir al sistema
    - Saldo negativo habilitado para solicitud de pedidos
    - Redirecciones a recursos externos:
        - Link al perfil de Facebook del CEI
        - Link al perfil de Instagram del CEI
        - Link a un sitio externo con las preguntas frecuentes (FAQs)



- Módulo para emisión y gestión de notificaciones:




En el anexo “Casos de uso del sistema” es posible acceder a una descripción detallada de los mismos, pudiendo identificar los actores que participan, las problemáticas que resuelven cada uno de ellos a partir de los resultados de la elicitación y los cursos normales y alternos a considerar en el desarrollo de la plataforma.

#### 6.1.3.2. Requerimientos no funcionales

Entre los requerimientos no funcionales encontrados se listarán los que a opinión del equipo son los más importantes y sobre los cuales se hizo foco durante la implementación:

- **Mantenibilidad**: dado que el producto pasará a manos del CEI una vez desplegado, debe ser fácil de comprender, mantener y extender en caso que fuera necesario. Es por ello que se hará hincapié en los siguientes aspectos para viabilizar un correcto mantenimiento por un equipo ajeno al que desarrolló el sistema en primera instancia:
    - *Documentación*: se generarán activos del proyecto asociados a documentación técnica (la API de la plataforma será documentada siguiendo la especificación OpenAPI y a su vez se proveerá documentación del código en el proyecto de Backend y de Frontend) y funcional (existirá un manual de usuario detallado que especificará, con el suficiente nivel de detalle, las reglas de negocio del sistema).

    - *Frameworks y arquitectura*: se seleccionarán cuidadosamente los frameworks a utilizar, limitando las opciones a aquellos que cuenten con una documentación oficial clara y detallada y con comunidad masiva y activa que brinde el soporte necesario.
    - *Buenas prácticas*: se procederá a realizar una capacitación interna previo al desarrollo del sistema con el fin de seguir las buenas prácticas y recomendaciones oficiales de los frameworks seleccionados.
- **Seguridad**: el sistema manipulará, por un lado, datos personales e información confidencial y por otro lado, soportará transacciones monetarias, por lo tanto se tendrán en cuenta los siguientes aspectos:
    - *Confidencialidad*: el sistema limitará su acceso únicamente a usuarios identificados, por lo que se proveerán mecanismos de autenticación robustos y consolidados basados en la utilización del servicio externo Firebase Authentication. Por otro lado, se implementarán esquemas de autorización por rol para especificar los privilegios de acceso o modificación de los recursos.
    - *Integridad*: la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción e inconsistencia de estados. Cada movimiento o transacción realizada en el sistema (asociada a información sensible) será auditada mientras que toda comunicación entre cliente y servidor será a través de un protocolo cifrado (HTTPS).
- **Disponibilidad**: si bien no es necesario que el sistema esté online 24/7, se busca que el mismo tenga el menor downtime posible. Se buscará cumplir con los siguientes aspectos:
    - El sistema correrá en un servidor dedicado para tal fin, sin procesos externos que puedan afectar a la performance del host y degradar la operatoria de la aplicación o la base de datos.
    - Se deberá configurar una alarma que permita detectar en forma inmediata cuando el sistema no esté reportando que está saludable.
- **Usabilidad**: el sistema será utilizado por diversos perfiles de usuarios finales, por lo que debe ser sencillo de entender y utilizar, con la mínima curva de aprendizaje posible. Para ello, será necesario cumplir con los siguientes objetivos:
    - El sistema deberá proveer interfaces que respeten las normativas de diseño que existen en el mercado actualmente y que son adoptadas y aceptadas por gran parte de la comunidad.
    - El sistema deberá proporcionar mensajes de error informativos con un texto claro y orientado al entendimiento del usuario final.
    - Se deberá contar con manuales de usuario con una estructura adecuada y de sencilla lectura.
    - En el sistema deberá existir una sección de preguntas frecuentes para despejar cualquier tipo de duda del usuario final así como medios de contacto con el CEI.
- **Portabilidad**: debido a que el sistema será de uso masivo y orientado a usuarios finales, el mismo debe ser accesible tanto desde navegadores webs como dispositivos móviles, otorgando interfaces responsive principalmente para aquellas vistas asociadas al estudiante.

A pesar de considerar a los atributos anteriores como prioritarios no hay que dejar de mencionar que otros atributos de calidad son también atacados de alguna u otra manera en el diseño del sistema.

#### 6.1.3.3. Restricciones

A nivel restricciones se tuvieron las siguientes:

- Presupuesto
    - Todos aquellos servicios que se evalúen para satisfacer alguno de los requerimientos funcionales deben ser gratuitos o tener una capa free, puesto que el CEI no puede permitirse el pago de una suscripción particular.
    - No es viable contar con una aplicación nativa por el costo de mantenerla en las Stores.
- No existe un equipo de desarrollo del lado del cliente que una vez producido el despliegue de la aplicación tome control sobre el producto.
    - Esto obliga a que el sistema sea configurable y flexible mediante la interacción de un usuario final de carácter no técnico.
    - No existe tampoco un equipo de soporte que ante la identificación de errores pueda actuar en consecuencia.

### 6.1.4. Estado del arte

Esta investigación tiene como objetivo dos aspectos principales. Por un lado, interiorizarse en el dominio de la problemática actual, estudiando las diferentes alternativas que existen en el mercado y así lograr manejar un lenguaje común con el resto de los stakeholders. Por el otro, utilizarlo como inspiración y referencia previo al diseño del sistema, buscando aquellos diferenciales y puntos de mejora que puedan identificarse.

El mercado de Mar del Plata no cuenta con muchas soluciones a la altura del producto deseado. En cuanto a la UNMdP, la Facultad de Ciencias Económicas y Sociales posee un sistema que da solución a ciertos aspectos de la problemática planteada. Se observa que las funcionalidades implementadas son las mínimas y necesarias para poder gestionar pedidos, pero carecen de muchas otras que hemos listado en el alcance del proyecto actual. La UI no es del todo amigable y los flujos no son siempre claros para el usuario, al menos desde nuestra perspectiva.

Por otro lado, en la Facultad de Ciencias Exactas y Naturales este proceso es aún mucho más manual, teniendo el alumno que ir directamente al mostrador e indicar qué archivo quiere imprimir y cómo será su configuración.

Saliendo del mercado local es posible identificar un producto similar propiedad de la Facultad de Ingeniería de la Universidad de Buenos Aires. Este sistema se encuentra orientado a la impresión de objetos 3D, sin embargo, hay aspectos a resaltar. Aunque la interfaz no tiene un aspecto agradable, posee un formulario sencillo y de un único paso, al que es posible acceder tanto para alumnos de la UBA como para usuarios externos. Una vez realizado el pedido, se envían los datos al correo

electrónico proporcionado junto con un código de orden. Dicho código puede ser utilizado para realizar el seguimiento del pedido en la aplicación, indicando estado del pedido, fecha de entrega estimada y tiempo total de la impresión.

### 6.1.5. Análisis técnico

En los siguientes subapartados se justificará la elección de las tecnologías, herramientas, prácticas, protocolos e implementaciones utilizados en los componentes del sistema. Para la toma de decisiones el equipo se basó en su experiencia previa en cada una de las tecnologías, las necesidades del sistema y en cómo las elecciones cubren las necesidades que los drivers de arquitectura citan.

#### 6.1.5.1. Cliente

El objetivo de este apartado es determinar el stack tecnológico utilizado para el desarrollo del frontend de la aplicación. Para ello se decidió realizar un análisis técnico compuesto por secciones incrementales que toman como premisas el cumplimiento de los requisitos funcionales, no funcionales y restricciones determinadas anteriormente. De esta manera se buscó lograr una coherencia técnica/funcional que dote al proyecto de una consistencia estructural y que permita identificar con claridad los requisitos técnicos involucrados para dar respuesta a las necesidades de los stakeholders.

Como se nombró anteriormente, el análisis partió de la identificación de aquellos requisitos y restricciones que pudieron tener un impacto en la implementación de esta vertical. Seguido de ello, se procedió a realizar una investigación de la actualidad del mercado del frontend, cuyo objetivo fue identificar los stacks/frameworks a evaluar para luego detallar las bases del funcionamiento, fortalezas, debilidades y los contextos de utilización de cada uno de ellos. Por último, se provee una sección destinada a la elección del stack, considerando las opciones disponibles, el contexto del equipo y del proyecto y los requisitos a los que se deben dar respuesta.

**Actualidad del desarrollo Frontend**

Con el correr del tiempo, las necesidades y las formas de resolver las dificultades que se han ido presentando en el mundo del desarrollo web fueron variando. Sin embargo, el objetivo central siempre ha sido evolucionar hacia la excelencia en la experiencia del usuario, logrando una usabilidad sencilla, intuitiva e inmersiva.  Para dar respuesta a un mercado cambiante y cada vez más exigente nacieron frameworks y librerías, cuyo utilización resulta imprescindible debido a que estandarizan y facilitan el desarrollo orientado a la performance y a la usabilidad.

Con este panorama, existen 3 frameworks/librerías de Javascript que acaparan prácticamente todo el mercado frontend: React , Vue y Angular. A continuación se detallan las bases funcionales, contextos de utilización y propuestas de cada tecnología:

***React***

ReactJS se define como una biblioteca para construir interfaces de usuario interactivas. React, a diferencia de los otros dos competidores, no es un framework, sino una librería de gestión del renderizado. Esta diferencia, se fundamenta en los siguientes características:

- **React no define un flujo de trabajo**: esta biblioteca únicamente proporciona métodos, hooks y recomendaciones para realizar todo el trabajo de renderizado a través de ella, sin embargo no define una arquitectura ni ninguna forma concreta de uso. Una misma tarea en React puede llegar a realizarse de varias formas alternativas y perfectamente funcionales.
- **React únicamente gestiona el renderizado**: el resto de funcionalidades esperables en un framework, como son los módulos de ruteo, gestión de formularios o manejo de estado, deben de ser agregados como dependencias adicionales externas, debido a que no están incluidos en la librería principal.

El hecho de que React no sea un framework puede ser visto como una ventaja, sobre todo para los desarrolladores más apegados al desarrollo Javascript tradicional y para aquellos que deseen tener una libertad de trabajo mayor, ya que entre sus ventajas encontramos:

- **Programación funcional:** su implementación se alinea fuertemente con la filosofía de la programación funcional. Nociones como funciones puras o funciones de orden superior son conceptos claramente arraigados en React y aplicados a sus propios componentes. Es por ello que en React podemos encontrar los llamados componentes puros, cuyos procesos de renderizado únicamente dependen de sus propiedades, e incluso componentes que reciben componentes para crear nuevos componentes (High Order Components o HOCs).
- **Motor de templating flexible**: React renderiza a partir del uso de JSX, una sintaxis de etiquetas que basa su funcionamiento y se asimila en gran medida a HTML pero que se diferencia en el uso con una serie de extensiones que se basan en la sintaxis de Javascript.
- **Mecanismo de renderizado granular:** el mecanismo de detección de cambios es implementable por componente. Los componentes funcionales y de clase tienen implementados métodos para escribir las condiciones necesarias para que se produzca un nuevo ciclo de renderizado en la vista.
- **Facilidad para la integración de librerías:** pese a utilizar un DOM virtual, la creación de referencias a componentes virtuales permite acceder a los elementos del DOM real vinculados e integrar de forma sencilla otras librerías de Javascript que realizan modificaciones directas sobre el propio DOM.

***Vue***

Vue es un framework con orientación MVC utilizado para el desarrollo de Single Page Applications. Fue creado por un ex-ingeniero de Google, Evan You, que tras trabajar con AngularJS en un gran número de proyectos llegó a la conclusión de que sería posible crear un framework similar, pero más ligero. Hizo hincapié en mantener sus características más relevantes e incorporar nuevas

implementaciones que permitieran sanear las problemáticas identificadas en la primera versión del framework de Google.

Vue introduce ciertas novedades interesantes que lo diferencian de otras tecnologías, y que definen su ecosistema:

- **Incremental:** a diferencia de Angular y React, Vue no limita su uso a la implementación de un proyecto exclusivo, sino que es posible integrar este framework de manera sencilla a un proyecto ya existente y, poco a poco, ir ampliando sus capacidades y su alcance.
- **Progresivo:** Vue partió con el objetivo de crear una solución muy ligera que soportara las funcionalidades de gestión de renderizado y componentes. Sin embargo es posible ampliar sus funcionalidades con facilidad añadiendo una serie de módulos o bibliotecas adicionales. De esta forma se logra que si el proyecto a implementar es ligero, no tendrá que cargar con un bundle excesivo, así como si es más completo, se podrá seleccionar exclusivamente las características extra que necesita.
- **Sistema de renderizado performante:** Vue es sin duda más rápido a la hora de renderizar sus propios componentes en comparación con frameworks como React o Angular. Esto se debe a una gestión inteligente del cambio de propiedades en Vue que asegura que los cambios sólo se apliquen cuando sean necesarios y se eviten renderizaciones innecesarias.

### *Angular*

Angular, desarrollado por Google, lanzó su primera versión en 2010 (AngularJs), lo que lo convierte en el más antiguo de los 3 frameworks identificados. Al momento de su salida, logró revolucionar el desarrollo web (siendo el principal precursor del desarrollo web moderno), implementando un framework basado en el patrón MVC (Model-View-Controller) y aportando conceptos innovadores como lo son:

- **Inyección de dependencias:** la arquitectura MVC planteada por la primera versión de Angular, propone una clara separación entre la presentación, la lógica y el manejo de datos, generando tres capas de abstracción con el fin de mantener un código organizado y cumpliendo el principio de responsabilidad única (SRP). Los *Servicios* son clases cuyo propósito es contener lógica de negocio, implementaciones para acceso a datos o utilidades de infraestructura. Si bien estas clases son perfectamente instanciables desde cualquier otro archivo que las importe, Angular provee (y sugiere) una implementación del patrón de diseño de inyección de dependencias. Dicho sistema basa su funcionamiento en una serie de convenios y configuraciones que controlan la instancia concreta que será inyectada al objeto dependiente. La utilización de este patrón permite desligar la responsabilidad de la creación de instancias de un servicio para delegar directamente al motor del framework.
- **Two-way data binding:** este mecanismo permite generar enlaces de datos bidireccionales entre el componente y su template asociado, en donde, si un dato es modificado en uno de ellos, el cambio se verá reflejado de manera automática en el otro extremo.

- **Routing:** otra de las novedades que trajo consigo la primera versión de AngularJS, que luego sería tomada como un estándar en el desarrollo web moderno, fue la inclusión de un robusto sistema de gestión de rutas. Esto marcó una disrupción en las arquitecturas del momento, en donde la gestión de rutas es administrada totalmente por la aplicación web, a diferencia de la gestión habitual llevada a cabo por el servidor.

Sin embargo, en 2016 se produjo un cambio radical con el lanzamiento de Angular 2. Partiendo de la experiencia adquirida por el desarrollo de la primera versión y teniendo claramente identificados aquellos aspectos a mejorar a partir de la interacción con la comunidad, Google decidió crear un proyecto totalmente nuevo (para marcar dicho cambio, se eliminó el sufijo “JS” de su nombre, pasando a denominarse simplemente Angular), manteniendo aquellos conceptos arquitectónicos que marcaron un antes y un después en el desarrollo web, pero simplificando las APIs brindadas a los desarrolladores e incorporando una gran cantidad de features y cambios novedosos, entre los que se pueden destacar:

- **Typescript:** uno de los cambios más importantes de la nueva versión de Angular fue dejar de utilizar Javascript Vanilla como lenguaje base para comenzar a emplear Typescript. Este lenguaje es un “superset” de Javascript que incorpora muchos de los mecanismos más habituales de la programación orientada a objetos. Esta transición marcó un cambio de paradigma, transformando la arquitectura del framework a partir de la utilización de un lenguaje fuertemente tipado y fundamentos basados en la programación orientada a objetos.
- **Componentes:** la arquitectura interna también sufrió transformaciones importantes, desechando los clásicos aspectos de controladores y scope de AngularJS para pasar a una orientación basada en la jerarquía de componentes como concepto arquitectónico transversal. De esta forma, los componentes, formados por un template, un archivo de estilos y una clase, pasaron a ser la unidad fundamental de la capa de presentación del framework.
- **CLI (Command Line Interface):** junto con la nueva versión se introdujo un conjunto de herramientas para la consola de comandos que permite, entre otras características, generar las estructuras básicas de nuevos proyectos, creación de componentes, directivas o diversas clases específicas de Angular como servicios y módulos.

**Elección del Framework**

A partir de la investigación y análisis de los frameworks identificados en la sección anterior, se decidió utilizar Angular como herramienta de desarrollo del Frontend. Uno de los fundamentos de dicha elección radica en la gran compatibilidad con ciertos atributos de calidad determinados en la sección de análisis funcional:

- **Mantenibilidad**: Angular se destaca por ser un framework completo que brinda todas las herramientas y dependencias necesarias para solventar un proyecto complejo y escalable. Esto se traduce en que el propio framework define la arquitectura base y la manera en que debe ser utilizado. Para ello, además de contar con una extensa y extremadamente detallada documentación y guías de desarrollo, como se nombró anteriormente, posee un CLI a partir del cual es posible generar componentes, directivas, servicios y otras clases de una manera sencilla y unificada. Estas características otorgan de una estandarización universal a la

implementación del proyecto, lo que permite un entendimiento trivial por parte de los desarrolladores. Además, el uso de Typescript como lenguaje de desarrollo simplifica el entendimiento de métodos y modelos del sistema, explicitando sus firmas y propiedades respectivamente. Esto marca una clara ventaja debido al hecho de que el producto será mantenido por un equipo de programadores diferente luego del despliegue.

- **Usabilidad**: al ser desarrollado y mantenido por Google, Angular cuenta con una integración nativa con la librería de estilos Angular Material, que sigue con los principios y estándares definidos por Material Design. Este protocolo de diseño fue creado con el fin de lograr una coherencia estética y funcional que otorgue una experiencia de uso unificada con diseños de componentes intuitivos y atractivos.
- **Robustez**: Angular se destaca por ser un framework robusto y consistente que otorga todas las herramientas necesarias para realizar aplicaciones web complejas y escalables. En línea con esta característica, provee un entorno de prueba automático para cada componente o servicio creado.

Además, los integrantes del equipo ya contaban con experiencia previa en dicho framework, lo que supuso una disminución en la curva de aprendizaje inicial. Este aspecto cobró aún mayor importancia al momento de realizar la viabilidad temporal del proyecto, teniendo en cuenta la necesidad de cumplir con los tiempos de entrega que propuso el cliente.

**Progressive Web Application**

Como se identificó anteriormente, uno de los requisitos no funcionales más importantes que se identificaron en el proyecto fue el de lograr una aplicación con un alto grado de usabilidad, con el fin de que cualquier tipo de usuario pueda acceder y utilizar a la plataforma de manera sencilla e intuitiva. Si bien la utilización de Angular, junto con Angular Material, respetan los estándares de Material Design que permiten un diseño y una usabilidad universalmente probada y validada, no otorgan APIs ni funcionalidades para lograr esta misma experiencia en dispositivos móviles. Debido a esto, surgió la obligación de investigar en el mercado posibles soluciones que den respuesta a las necesidades intrínsecas del sistema para lograr no sólo una usabilidad aceptable en dispositivos móviles sino también para soportar push notifications, con el fin de lograr una comunicación de eventos en tiempo real y de fácil acceso.

Durante esta investigación surgió la posibilidad de implementar una aplicación nativa para dispositivos Android y iOS, o bien una aplicación híbrida a partir de frameworks como React Native o Ionic que permitieran unificar el desarrollo para los Sistemas Operativos nombrados anteriormente y además evitar una curva de aprendizaje inicial muy marcada, ya que estos últimos basan su desarrollo en tecnologías web. Sin embargo estas opciones fueron descartadas debido a la complejidad temporal y técnica que requería desarrollar y mantener proyectos en paralelo e independientes al de la plataforma web realizada en Angular.

Finalmente, luego de una ardua investigación de alternativas se logró obtener una solución tecnológica que además de ser innovadora, se ajustaba exactamente a las necesidades y limitaciones

existentes. La elección fue la implementación de una Progressive Web Applications (PWA) mediante la utilización de Service Workers.

PWA no es una tecnología, sino un concepto creado por Google en el año 2015. Este concepto pretende disminuir la brecha existente entre una aplicación web tradicional y una aplicación nativa, definiéndose como un punto medio entre ambas y tomando los beneficios de los dos tipos de aplicaciones.

Esta nueva filosofía propone una serie de requisitos indispensables que toda aplicación debe cumplir para entrar en la categoría de PWA:

- **Rápida:** todas las acciones que se lleven a cabo por el usuario, tanto de carga inicial, transiciones y navegaciones, se tienen que realizar en unos tiempos considerables y que la experiencia fluya para conseguir una interacción dinámica con el usuario. Esto permite que los contenidos se muestren al usuario prácticamente al instante, ya que se apoyan en el almacenamiento en la caché. Las interacciones, tales como clics o scroll, también deben ser inmediatas. El menor peso de la Progressive Web App en comparación a la App nativa es un factor decisivo para ello.
- **Segura:** la aplicación debe implementarse mediante el protocolo seguro HTTPS. Esto posibilita asegurar que el acceso sea seguro y que el contenido servido no haya sido sujeto a manipulaciones. Se emplean tecnologías como TLS para el cifrado web.
- **Responsive:** en la actualidad, la mayoría de webs cuentan con diseño responsive que las permite adaptarse a diferentes dispositivos, algo imprescindible con el papel predominante de los smartphones. A pesar de que las PWA van más allá del simple diseño responsive, este se puede seguir mencionando como una de sus características principales. Estas Apps deben adaptarse automáticamente a cualquier formato, navegador o dispositivo (con los consecuentes cambios de medidas y resolución).
- **Actualizada:** las PWA siempre mostrarán su última versión al usuario con el empleo de actualizaciones automáticas, de manera constante e instantánea y sin necesidad de descargarlas. Esto es posible gracias al empleo de Service Workers y porque no deja de ser una Web App, independiente de la publicación (y todo el proceso de revisión e instalación por parte del usuario que conlleva) en los markets de aplicaciones.
- **Offline:** una PWA debe permitir continuar siendo funcional (de manera parcial o incluso total) a pesar de que no haya conexión a Internet (o esta sea de baja calidad). Para que se pueda servir contenido a los usuarios que estén offline, se utilizan los service workers y el almacenamiento en caché de la información esencial para iniciar la App, que se realiza desde la primera vez que esta se abre. Así, en las visitas posteriores, se puede disponer de cierto contenido independientemente de la red. Esto se basa, a la vez, en la “App shell”, es decir, la estructura básica de la App, que se podrá mostrar aunque existan problemas con el contenido. Todo ello deriva en una mejor experiencia de usuario y evita la frustración que genera la imposibilidad de acceso.
- **Multiplataforma:** en su desarrollo, la tecnología utilizada contempla su ejecución en diversos dispositivos, sistemas operativos y navegadores. Esto, además de ser clave a la hora de ofrecer una experiencia de usuario satisfactoria y alcanzar a más público potencial, supone facilidades

para los desarrolladores y permite abaratar costes, puesto que no se requieren programaciones diferenciadas (algo que sí ocurre con las Apps nativas). Esto es posible ya que la aplicación se ejecuta directamente

- **Con acceso directo:** las webs a las que se acceda desde el navegador que dispongan de una versión PWA suelen informar al usuario, invitándole a "añadirla a su pantalla de inicio". Estas aplicaciones se pueden utilizar desde el navegador, pero también se pueden instalar en el dispositivo. Esta instalación no requiere de una "descarga" tal y como usualmente se conoce, sino que se basa en la inclusión de un acceso directo en la pantalla de inicio o escritorio del dispositivo. Este se muestra como un icono más, prácticamente idéntico al de cualquier App nativa.
- **Apariencia nativa:** la interfaz de usuario y, en general, la apariencia de una PWA es muy similar a la de las Apps nativas, tanto en estética como en la manera de interactuar y navegar por ella. A esto contribuyen elementos como una pantalla de inicio, que se ejecute en una ventana de aplicación propia, totalmente responsive, que ocupe la pantalla por completo (sin la barra que muestra la URL), etc.
- **Funcionalidades propias de una App nativa:** con la evolución de las PWA, las mismas han ido adquiriendo opciones que antes se reservaban únicamente a las Apps nativas, como el acceso a distintas funciones del dispositivo. Las Progressive Web App pueden, por ejemplo, acceder a la geolocalización del dispositivo, al Bluetooth, sincronizarse en segundo plano o enviar notificaciones push (incluso cuando no está abierta la PWA). Estas notificaciones son una potente herramienta de comunicación que permite informar al usuario e invitarle a acceder, aumentando las visitas y, en consecuencia, las conversiones. Se debe considerar que estas posibilidades no están disponibles para todos los navegadores.

**Workers**

Como se nombró anteriormente, una PWA a pesar de ser una aplicación web que se ejecuta sobre un navegador, utiliza tecnologías que hacen que su estética y funcionamiento se asemejen enormemente a una App nativa, por ejemplo, mediante la ejecución en segundo plano, la posibilidad de implementar notificaciones push o el acceso offline. Estas características son posibles en su mayoría gracias a la utilización de los Service Workers. Estos proporcionan una API que revolucionaron la forma en la que se entiende el desarrollo frontend y en cómo se gestionan y administran los recursos del navegador.

Antes de profundizar en el funcionamiento de los Workers, es necesario plantear por qué nacieron. JavaScript es un lenguaje monohilo, esto significa que toda la interpretación del código de la aplicación va a ser atendida en un único hilo de ejecución. La forma de realizar diferentes acciones es por medio del uso de la entrada y salida de manera asíncrona y controlando todas las acciones por medio de un bucle (event loop) que gestiona qué tareas asíncronas deben realizarse en cada momento. El problema surge cuando una Web requiere mucho recursos de la máquina y se desea posibilitar la ejecución y paralelización de tareas concurrentes.

Para dar respuesta a esta problemática, en JavaScript se ha implementado lo que se conoce como Workers, que como se nombró anteriormente, brindan la posibilidad de ejecutar tareas en otros hilos de ejecución distintos al principal. Este cambio de paradigma, posibilita ejecuciones en segundo plano, lo cuál permite monitorizar distintos aspectos sin que el usuario final note un degradamiento en la performance o el renderizado de la aplicación, descargar información que el usuario posiblemente utilice luego o intentar sincronizar información del usuario con el servidor que en otro momento fue imposible debido a la conexión de datos

Estas situaciones son resueltas con dos tipos diferentes de Workers:

- **Web Workers:** brindan una API orientada a posibilitar y facilitar la ejecución en paralelo de scripts que requieren un uso intensivo del procesador. Generan dos contextos globales dentro del navegador para que se puedan realizar tareas que no bloqueen la UI.
- **Service Workers:** brindan una API que permite realizar tareas que tienen que ver con la solicitud de red.

Estos últimos, como se detalló anteriormente, son la base tecnológica que permiten la existencia de las PWA. Los Service Workers actúan esencialmente como un proxy ubicado entre las aplicaciones web, el navegador y la red (cuando está accesible). Están destinados, entre otras cosas, a permitir la creación de experiencias offline efectivas, interceptando las peticiones de red y realizando la acción apropiada si la conexión de red está disponible y hay contenidos actualizados en el servidor. Consisten en un fichero JavaScript que controla la página web con el que está asociado, interceptando y modificando la navegación y las peticiones de recursos, y cacheando los recursos de manera muy granular para ofrecer un control completo sobre cómo la aplicación debe comportarse en ciertas situaciones.

De esta forma, utilizando los Service Workers con el fin de lograr implementar una PWA, se logró dar respuesta a los requisitos funcionales y no funcionales detectados previamente.

#### 6.1.5.2. Servidor

A continuación se detallarán los motivos por los cuales se propuso una solución de backend para la lógica del sistema e intermediario entre la capa de datos y la capa de presentación.

Desde la concepción del proyecto y regido por los requerimientos funcionales y no funcionales, se hizo necesario la creación de un componente capaz de lidiar con toda la lógica y algoritmia que éste conllevaba. Dentro de las alternativas se decidió por la implementación de un servicio de backend completamente codificado por el equipo.

Para la elección de una tecnología específica dentro de las que ofrece el mercado actual se tuvieron en cuenta diversos factores y requisitos que debían cumplirse. Principalmente, el foco estuvo en utilizar una herramienta que satisfaga la naturaleza del sistema, es decir, una aplicación de tres capas con una arquitectura cliente-servidor (el componente en cuestión forma parte de la capa lógica, desde la parte del servidor). Por lo tanto, se limitó a las tecnologías que soporten el desarrollo de un componente que se pueda comunicar vía web.

De dicha manera, se optó por la utilización de **Node.js**, entorno de ejecución que nos permite correr código JavaScript desde el lado del servidor. Como se mencionó previamente, los desarrolladores ya contaban con experiencia tanto en el lenguaje como en la tecnología, tanto desde la parte teórica y educativa, como desde la práctica.

En primera instancia, teniendo el entorno de ejecución mencionado, se optó por elegir **Express** como framework de desarrollo, el cual es uno de los líderes en cuanto a creación de aplicaciones web y APIs.

Estas elecciones trajeron varias ventajas al proyecto:

- Eficiencia y escalabilidad del componente gracias a la arquitectura guiada por eventos de Node.js y su asincronismo provechoso para operaciones de I/O.
- Rápido desarrollo de una API.
- Gran cantidad de librerías y comunidad que continuamente crece a través de su gestor de paquetes NPM, el cual posee un gran ecosistema de desarrolladores y herramientas muy útiles a disposición de todos.
- La unificación de un único lenguaje de programación para ambas capas, lógica y de presentación, con JavaScript/TypeScript.

A medida que el proyecto avanzaba, el equipo trabajó con ambas herramientas en conjunto, potenciadas por otras externas proporcionadas por su gestor de paquetes (NPM). Sin embargo, mientras que el componente se fue desarrollando se comenzaron a notar complejidades que obstaculizaron su implementación. Estos inconvenientes se hacían notorios a la hora de codificar una nueva funcionalidad y de trabajar con la capa de datos. Las principales problemáticas consistían en generar mucho código para agregar funciones a la aplicación web, crear código hardcodeado si se requería consultar la base de datos o, al momento de definir una arquitectura específica para el backend, su modularización y cómo se estructuraría.

De tal forma, y luego de una búsqueda de una herramienta que socorriera estos inconvenientes, el equipo se encontró con **NestJS**. Este framework para Node.js contiene tantas ventajas y beneficios para el proyecto en cuestión que se tomó la dura decisión de realizar una migración a esta herramienta, más allá de que se encontraba avanzada la aplicación.

NestJS proporcionó como ventajas:

- El aprovechamiento del lenguaje TypeScript, con todo lo que esto implica, nombrado en el apartado anterior.
- Una arquitectura fuertemente basada en el framework de frontend que el equipo utilizó, Angular, lo cual benefició la estructuración y arquitectura del componente.
- La interfaz de línea de comandos (Nest CLI) que posee, aspecto que aceleraría y simplificaría el desarrollo.
- La integración nativa que posee y facilita la utilización de librerías que eran necesarias.
    - TypeORM, para comunicarse con la capa de datos.

    - WebSockets, utilizado para mantener una comunicación activa entre cliente y servidor.
    - Entre otros (Logging, Caching, Object Validation).
- Una detallada documentación, lo que disminuyó la curva de aprendizaje.
- Incorporación de componentes y clases (decorators, interceptors, filters, entre otros), que son altamente reutilizables y simplifican el desarrollo de nueva funcionalidad.

#### 6.1.5.3. Base de datos

Para poder tener una plataforma completamente funcional se requiere la persistencia de ciertos valores y registros. Estos datos pertenecen tanto a la naturaleza del problema como a la plataforma en sí desde un punto de vista más técnico.

Como se elicitó en las etapas previas, el modelo de datos tiene un fuerte aspecto relacional y estructurado, en el que se deben recopilar ciertos datos, lo que inclinó la balanza a elegir un motor de base de datos **relacional** en principio. A pesar de ello, existen en el mercado otras alternativas de motores como son los **NoSQL**, los cuales se utilizan en casos donde los datos vienen de manera desestructurada, en gran cantidad y continuamente pueden variar. Como se mencionó previamente, la estructura de datos que se definió es suficiente para el problema en cuestión y no recibirá modificaciones, por lo que un modelo relacional es más ajustable a este caso.

Para el modelo de datos se tomaron en cuenta las entidades que entran en juego y sus actores correspondientes, analizando los campos propios de cada uno y las interrelaciones que tienen entre sí. Así también para darle cierta funcionalidad requerida al sistema se incorporaron ciertas entidades inherentes a la parte técnica y funcional de ciertos componentes del sistema, tales como paramétricas o variables que se utilizan en el entorno de ejecución.

Los principales competidores dentro del campo de las bases relacionales fueron MySQL (y/o MariaDB), SQL Server y PostgreSQL. A la hora de elegir el motor específico a utilizar, **MySQL** fue la elección obvia. Esta decisión estuvo condicionada por la experiencia previa del equipo, por ser uno de los motores líderes del mercado, licenciamiento y uso gratuito, bajo uso computacional, y la relativa sencilla integración con la capa del servidor.

Avanzado el proyecto surgió nuevamente una migración hacia otro motor de base de datos relacional, producto también líder en el mercado y de código abierto, **PostgreSQL**. Esta determinación estuvo influenciada por una funcionalidad clave de cara al futuro de la plataforma. La causa de la misma se sustenta en que PostgreSQL tiene la capacidad de manejar *múltiples schemas* en una misma base de datos, lo que brinda la posibilidad de replicar las mismas estructuras para distintos **tenants**, clientes o usuarios. Este esquema multi-tenant que PostgreSQL trae integrado, otorga una ventaja competitiva en caso de una futura extensión de la plataforma para su uso masivo (funcionalidad que el equipo tuvo en mente a lo largo del desarrollo).

De cara a esta capa de datos se puede destacar que se hizo foco en estos atributos de calidad principales:

- **Mantenibilidad:** este atributo está relacionado con el componente de backend, debido a que a través de la integración con el framework elegido se pudieron fácilmente gestionar los cambios en la estructura de datos, las consultas, inserciones y actualizaciones a la base de datos.
- **Portabilidad:** la base de datos de tipo SQL, y su motor, PostgreSQL (la cual es reconocida en la industria) permitió instalarla en los equipos donde se realizó el desarrollo y en donde efectivamente se desempeñó productivamente. Aun así, se puede resaltar que en la actualidad este motor se puede instalar en múltiples sistemas operativos, y hasta hacer su uso en plataformas de tipo cloud, lo cual aumenta su portabilidad.

#### 6.1.5.4. APIs

Esta sección del escrito se abocará a las decisiones tomadas por el equipo en relación con la comunicación con los servicios web y su interfaz correspondiente.

Como se mencionó en varias oportunidades, se cuenta inicialmente con una aplicación de tres capas. La capa de presentación, desarrollada en Angular, brindó la posibilidad de desacoplar el desarrollo del backend. De esta manera cada componente es independiente, lo que deja totalmente libre elegir su interfaz de comunicación.

Para poder lograr interacción entre todos los componentes desarrollados fue necesario la intervención de distintos protocolos y la definición de su interfaz de comunicación o comúnmente llamado Application Programming Interface (API).

En los siguientes apartados se expondrán las alternativas analizadas según los distintos protocolos identificados y componentes que los utilizan, casos de uso y requerimientos.

##### 6.1.5.4.1. HTTP

Para lograr una interacción bidireccional entre el servicio web y la capa de presentación, la elección del protocolo HTTP es trivial. Protocolo de capa de aplicación, por defecto utilizada para la comunicación en internet. Actualmente el mercado cuenta con dos arquitecturas líderes que operan sobre este: **API RESTful** y **GraphQL**.

- **API RESTful:** basado en una serie de restricciones arquitecturales definidas como REST se disparan diversas maneras de implementar una API bajo estas definiciones. Sus fundamentos residen en realizar solicitudes a ciertos endpoints, transfiriendo o recibiendo el estado de los recursos pertinentes al sistema y mundo del problema. Esta información viaja mediante HTTP en un formato estándar previamente definido, por ejemplo, JSON, XML, texto plano, entre otros.

  Es importante tener en cuenta que también los verbos, cabeceras, parámetros y códigos de estado propios del protocolo son importantes en esta filosofía. Esto es debido a que permiten referenciar metadata del usuario, manejar la autenticación y autorización, el acceso a recursos y su acción a tomar sobre ellos, gestionar diversos

flujos de la aplicación basándose en las respuestas del servidor, sus códigos de respuesta, etc.

Otro dato a resaltar sobre REST es que es **stateless o sin estado**, por lo cual ningún dato de la sesión cliente es almacenado en el servidor en cuanto a solicitudes previas a este. Es decir, cada solicitud se realiza independientemente de las futuras o anteriores y es el cliente el encargado de utilizar y manejar tanto la API y aplicación para hacer referencia a su sesión y almacenar sus datos.

- **GraphQL:** haciendo uso del protocolo HTTP y con un solo endpoint, GraphQL es un lenguaje de consulta y entorno de ejecución del lado del servidor para el desarrollo de APIs. Su particularidad y fuerte radica en que el cliente puede solicitar exactamente las propiedades o datos que necesita dentro del recurso en cuestión. Esto último, agiliza el desarrollo y lo hace mucho más flexible y amigable para el programador. A su vez permite también obtener datos de varios recursos y fuentes en una sola solicitud al servidor. Maneja consultas y mutaciones, para recuperar recursos y realizar cambios sobre estos respectivamente.

Se realizó una comparativa más a fondo entre estas dos técnicas plasmada en el siguiente cuadro:

| | GraphQL | REST |
|---|---|---|
| **Arquitectura** | Client-driven | Server-driven |
| **Organización** | Schema/Type | Endpoints |
| **Operaciones** | Query<br>Mutation<br>Subscription | Verbos HTTP |
| **Obtención de datos** | Datos específicos y personalizados con una sola request (flexible, eficiente y eficaz) | Datos fijos con múltiples requests (inflexible, ineficiente e ineficaz) |
| **Tiempo de respuesta** | Rápida | Lenta en ciertos casos: múltiples requests requieren mayor tiempo |
| **Comunidad** | Intermedia (creciente) | Grande |
| **Velocidad de desarrollo** | Rápido | Lento |
| **Curva de aprendizaje** | Moderada / acentuada | Moderada |
| **Autodocumentado** | Sí | No (aunque es posible integración con Swagger/OpenAPI 3.0) |
| **Casos de uso** | Sistemas complejos y arquitecturas de microservicios (client-driven apps) | Homogeneidad en clientes / entidades<br>(resource-driven apps) |

A la hora de optar por una alternativa se tuvieron en cuenta también el contexto, casos de uso del sistema, tiempos y conocimientos del equipo de desarrollo.

El modelo de negocio y su respectivo modelo de datos del CEI está planteado en base a recursos propios de éste, lo cual facilita el diseño de la API en base a endpoints y verbos HTTP. Las entidades del sistema ya se encuentran definidas y difícilmente muten a futuro, por lo que tanto solicitudes como respuestas se encuentran definidas y pueden ser fijadas, lo que no da lugar a una fuerte necesidad de tener respuestas o consultas flexibles. Metodologías como la paginación y el diseño eficiente para aminorar la cantidad de solicitudes realizadas al servidor, acompañado de un bajo volumen de datos, desembocan en bajos tiempos de respuesta siguiendo la filosofía REST, dejando en manos del cliente esta lógica. En la técnica de GraphQL el flujo para obtener datos de varias fuentes y recursos en una sola solicitud recae en el servidor, desembocando en una sobrecarga al mismo.

Por otro lado, el equipo se encuentra mucho más familiarizado con el diseño e implementación de APIs bajo la filosofía REST, lo cual agiliza los tiempos de desarrollo. El uso de Swagger (OpenAPI Specification 3.0), en conjunto con el servicio web, solventa la problemática de la documentación.

De tal manera la elección para la interfaz de comunicación entre el cliente y servicio web fue API RESTful. De cara a los atributos de calidad, los siguientes se vieron beneficiados:

- **Seguridad:** mediante el uso de una API REST, en conjunción con el estándar JWT (JSON Web Tokens) provisto por la herramienta de autenticación de Firebase, permite una relativa sencilla implementación junto con el protocolo HTTP a través del uso de cabeceras y códigos de estado, manteniendo protegidos los endpoints para los respectivos usuarios del sistema y roles dentro del mismo.
- **Usabilidad:** siguiendo las buenas prácticas de diseño y potenciado a través de una herramienta como Swagger, esto permite tener una API bien definida y especificada en una documentación estándar, donde yacen los parámetros, valores, cabeceras, códigos de estado de cada una de las solicitudes y respuestas del servidor facilitando así el entendimiento y uso de la misma.

#### 6.1.5.4.2. Web Sockets

A la hora de definir el proceso de gestión del pedido surgió un requerimiento particular sobre el mismo. En el contexto dado, es necesaria una interacción y comunicación en tiempo real en el que se vieran reflejados los nuevos pedidos que se iban solicitando de manera automática para el cliente, en este caso, las máquinas utilizadas por los empleados encargados de realizar las impresiones. Ante tal situación el equipo llegó a la conclusión que la alternativa que cubría todas las necesidades era la utilización de Web Sockets.

Gracias a esta tecnología es posible establecer una comunicación bidireccional, la cual posibilita recibir eventos del servidor, realizando una suscripción del lado del cliente, sin la necesidad

de volver a realizar solicitudes a éste. Este contexto de uso se alinea de gran manera con los requerimientos identificados. Con la incorporación de WS, el personal encargado de la fotocopiadora podrá tener la aplicación y el panel de solicitudes de pedidos actualizado en tiempo real sin tener que refrescar el navegador, de forma tal de poder realizar una gestión más cómoda y eficiente. Esto es una mejora significativa de cara a la usabilidad de la aplicación.

El protocolo funciona estableciendo una conexión entre cliente y servidor a través de lo que se denomina **Web Socket Handshake**. Una vez realizado, el cliente y servidor coordinan lo que se denominan **namespaces** dentro de los cuales se puede dividir la lógica de la aplicación, dependiendo del espacio de nombres que se utilice. Una vez establecida, el servidor puede enviar eventos y mensajes sin que el cliente lo requiera para que los tenga a su disposición y aplique el flujo correspondiente.

#### 6.1.5.4.3. IPP

Es notable que el sistema requirió conexión con terminales de impresión, de manera tal que la configuración de pedidos puede enviarse directamente a las impresoras y realizar las impresiones de la forma que el estudiante o usuario las requirió. Los desarrolladores realizaron una investigación a fin de soportar esta funcionalidad, guiados con asesoramiento técnico de parte de docentes de la Facultad de Ingeniería. A partir de ella se pudo poner en conocimiento del protocolo **Internet Printing Protocol (IPP)**. El mismo permite obtener datos sobre las terminales de impresión, sus características, estado actual y por supuesto, poder enviar el contenido a imprimir, delegando la gestión en el protocolo, todo a través de la red.

### 6.1.5.5. Servicios externos

En los primeros análisis que se hicieron en cuanto al sistema se detectaron ciertas funcionalidades las cuales debían ser satisfechas. Dentro de ellas se encontraban poseer autenticación de usuarios, activación de cuentas, reseteo de contraseñas (los dos últimos sujetos al envío de emails). También al analizar cómo se hizo en el apartado de frontend, la PWA debía manejar notificaciones. Todas estas funciones debían ser provistas por algún componente.

Luego de una adecuada búsqueda el equipo decidió hacer uso de **Firebase**, un Backend as a Service (BaaS) propiedad de Google, que daba respuestas a lo que debía ser resuelto. Esta herramienta cuenta con un sistema de notificaciones push y autenticación de usuarios, además de los respectivos envíos de email para activar y resetear cuentas. Además posee muchas otras funcionalidades tales como funciones en la nube (más conocido como Function as a Service), almacenamiento NoSQL, hosting, entre otros productos. Dichas herramientas podrían ser provechosas en caso que se hubieran necesitado o el proyecto hubiera mutado en esa directriz. Otra causa de su elección fue el plan gratuito que provee que resulta muy provechoso para la cuota de uso que se debe satisfacer.

Cabe resaltar que se analizó otra herramienta para la autenticación y autorización de usuarios, Auth0, la cual también es líder en ese nicho del mercado. Aunque la misma cumple con los requisitos solicitados, quedó fuera como elección por el plan de uso que ofrece, el cual permite un uso gratuito

hasta los 7000 usuarios. Por razones de costos, el equipo se decidió por Firebase debido a que su plan de uso gratuito nos garantiza cómodamente la cuota de usuarios esperada.

La elección de estos servicios tanto de backend como externos que se encargan de manejar la lógica de la aplicación impacta sobre estos atributos de calidad:

- **Mantenibilidad:** tanto la utilización del framework, como su arquitectura y la forma en que se organizó el código permite fácilmente desarrollar tanto nuevas funcionalidades como corregir errores durante el proceso de implementación. Esto contribuyó a mantener un bajo impacto entre componentes ante los cambios.
- **Seguridad:** la plataforma de autenticación utilizada, sumada a las librerías que se utilizan del lado del backend, permiten delegar la mayor parte de la lógica de seguridad tanto sobre Firebase como en NestJS. Esto permite abstraerse de su implementación y tener la confianza de utilizar un protocolo seguro y actualizado, el cual una empresa reconocida como Google y la comunidad de desarrolladores da soporte.
- **Disponibilidad:** las tecnologías en la nube que se utilizan durante el proyecto (Firebase) permiten poder hacer uso de ellas tanto en el desarrollo como en el ámbito productivo con un gran índice de disponibilidad ofrecido por la compañía.

## 6.2. Diseño

En la fase de diseño se tomó como input todo lo analizado en etapas anteriores y se avanzó en la definición de diferentes soluciones de software que solucionen los problemas identificados. Está conformada por diseños a **alto** y **bajo nivel**, siendo el Diseño de Arquitectura y el Diseño de Componentes ejemplos de cada uno respectivamente.

Además, en esta etapa se tomó como input el **proceso de negocio actual** para efectuar mejoras sobre el mismo de forma tal de estar alineado con las necesidades y requerimientos identificados. Gracias a esta evolución muchos problemas latentes en el CEI pueden ser ahora resueltos a través de una solución informática que basa estrictamente su comportamiento en el nuevo modelo de negocio.

### 6.2.1. Diseño funcional

Una vez definida y analizada la situación actual del CEI en cuanto a los procesos y roles de cada uno de los stakeholders se procedió a diseñar un **nuevo proceso** que disminuyese las complejidades, tiempos e interacciones de índole manual, a fin de acelerar los tiempos, minimizar los errores, mejorar la experiencia, etc.

En base al esquema de procesos actual relevado se decidió reestructurar el mismo, eliminando pasos innecesarios o directamente cambiando el orden, todo esto gracias a la existencia de un sistema informático de por medio.

#### 6.2.1.1. Innovación del proceso

El proceso final diseñado se encuentra representado desde dos puntos de vista diferentes: el estudiante y el empleado de la sede. En las siguientes sub secciones se detallarán los rediseños realizados para cada uno de estos flujos.

#### 6.2.1.1.1. Gestión de pedidos visto desde la perspectiva del estudiante

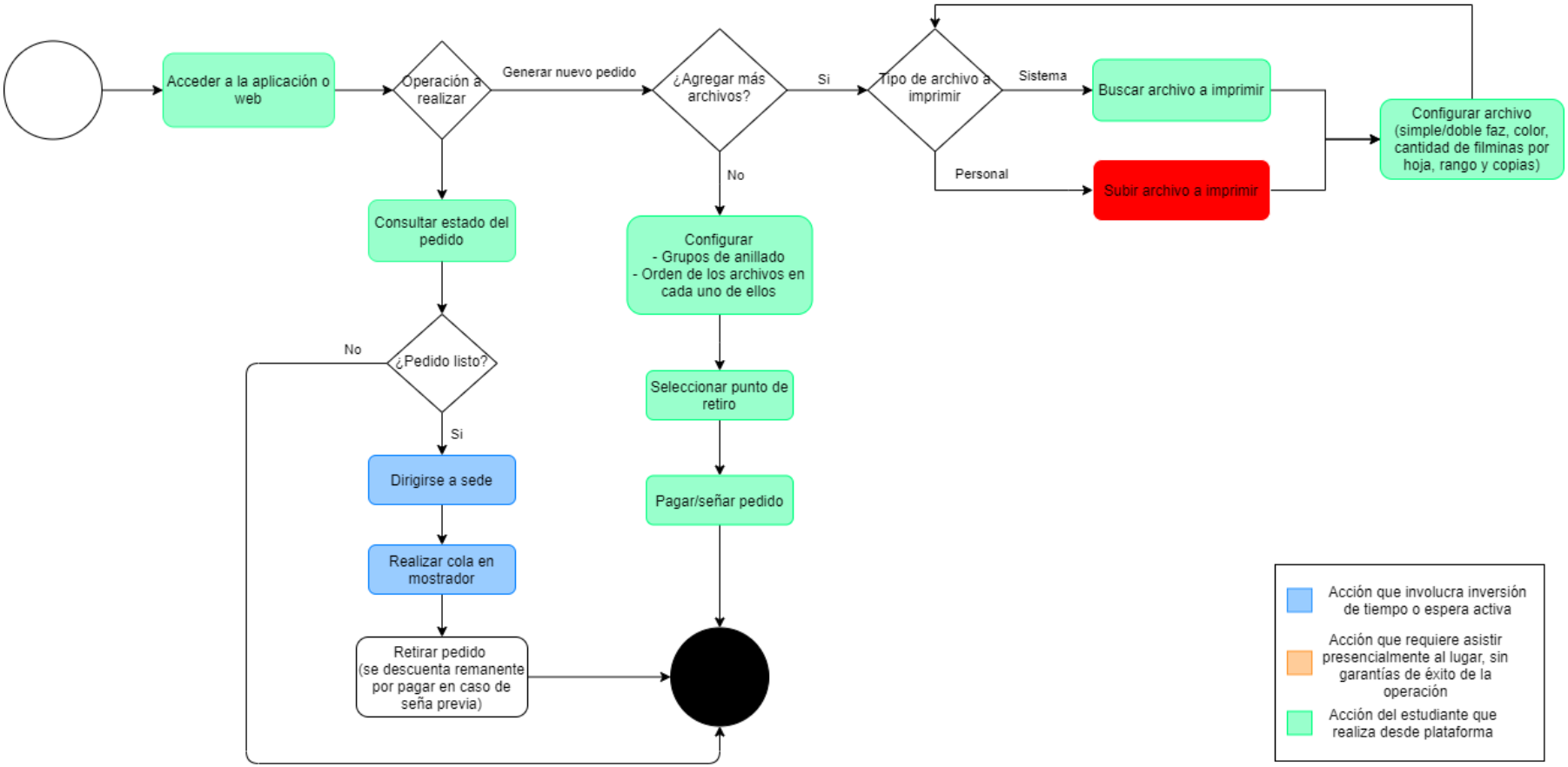



**Figura 13**. Proceso actualizado del estudiante para emisión, consulta y retiro de pedidos

En comparación con la Figura 11, el nuevo proceso **elimina** los tiempos de:

- Asistir a la sede para emitir el pedido

- Esperar que se libere alguno de los equipos de la sede para poder realizar el pedido
- Hacer la cola en el mostrador
- Esperar por la impresión/anillado del pedido.

Además, la **consulta del estado** puede ahora realizarse directamente vía sistema, ahorrando nuevamente viajes, colas y consultas en persona (que implica tiempo del empleado).

Por otro lado, tanto la **configuración** como el **pago** de los pedidos quedan ahora bajo la responsabilidad del sistema, dejando tiempo extra a los becados para que puedan abocarse plenamente a la impresión de los pedidos. Cabe mencionar nuevamente que todo esto contribuye a una **disminución de errores** en todo el ciclo de vida del pedido.

### 6.2.1.1.2. Gestión de pedidos visto desde la perspectiva del empleado de la sede

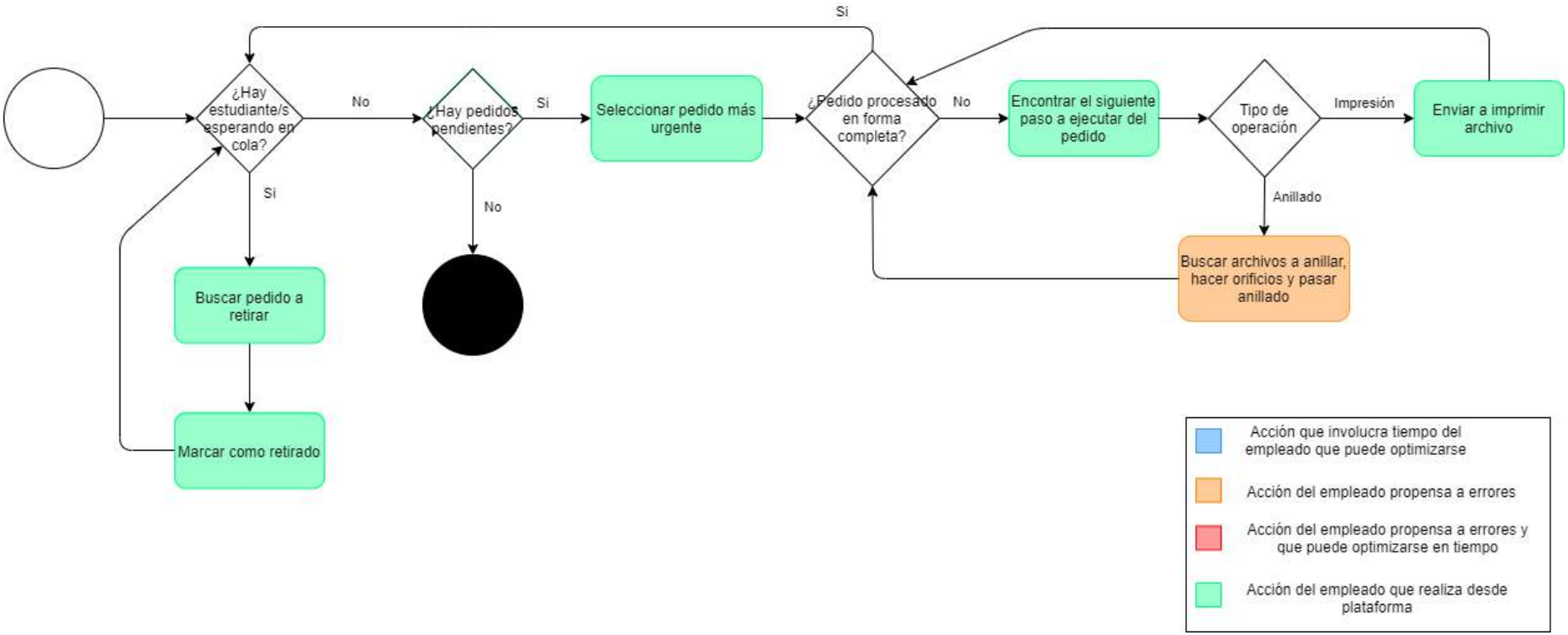



**Figura 14**. Proceso actualizado del empleado para gestión y entrega de pedidos

Comparando ahora los flujos AS IS y TO BE desde el punto de vista del empleado de la sede se puede apreciar de forma inmediata que la mayoría de las tareas manuales propensas a errores han sido **eliminadas** o **delegadas** completamente al sistema. De esta manera, todo lo concerniente a decidir el siguiente pedido o paso del pedido a procesar, la configuración de los archivos y el cobro del pedido ha quedado fuera de la responsabilidad del empleado.

Por otro lado, una consecuencia inmediata del nuevo workflow es que los estudiantes *sólo deberán* asistir a la sede para retirar sus pedidos, sabiendo que los mismos están listos. De esta manera, las colas que un alumno debiera encontrarse al asistir al establecimiento debieran ser nulas o con una reducida cantidad de personas.

El único punto que todavía presenta un **desafío** y sigue siendo una **fuente de error** es el anillado. El sistema se encarga de indicar claramente cuáles son los archivos que deben formar parte del grupo de anillado y en qué orden deben ordenarse los mismos. Sin embargo, a pesar de todas las mejoras introducidas todavía queda como responsabilidad plena del empleado hacer coincidir la configuración con el ordenamiento que se lleva a cabo.

El tiempo que el empleado ahora no invierte en configurar impresiones, tomar pedidos o incluso el buscar si el pedido está finalizado es ahora un recurso que puede ser invertido en otras operaciones que den mayor **valor** a la operatoria del CEI.

#### 6.2.1.1.3. Ciclo de vida del pedido

Para poder realizar correctamente la gestión de pedidos fue necesario diseñar un **diagrama de estados** que determine los **estados** y **transiciones** que estarán permitidas durante todo su ciclo de vida.

El siguiente diagrama tiene reunidas las acciones que puede ejecutar tanto el empleado de la sede como el estudiante sobre el pedido, limitadas de acuerdo al estado particular que se encuentre en un determinado momento.

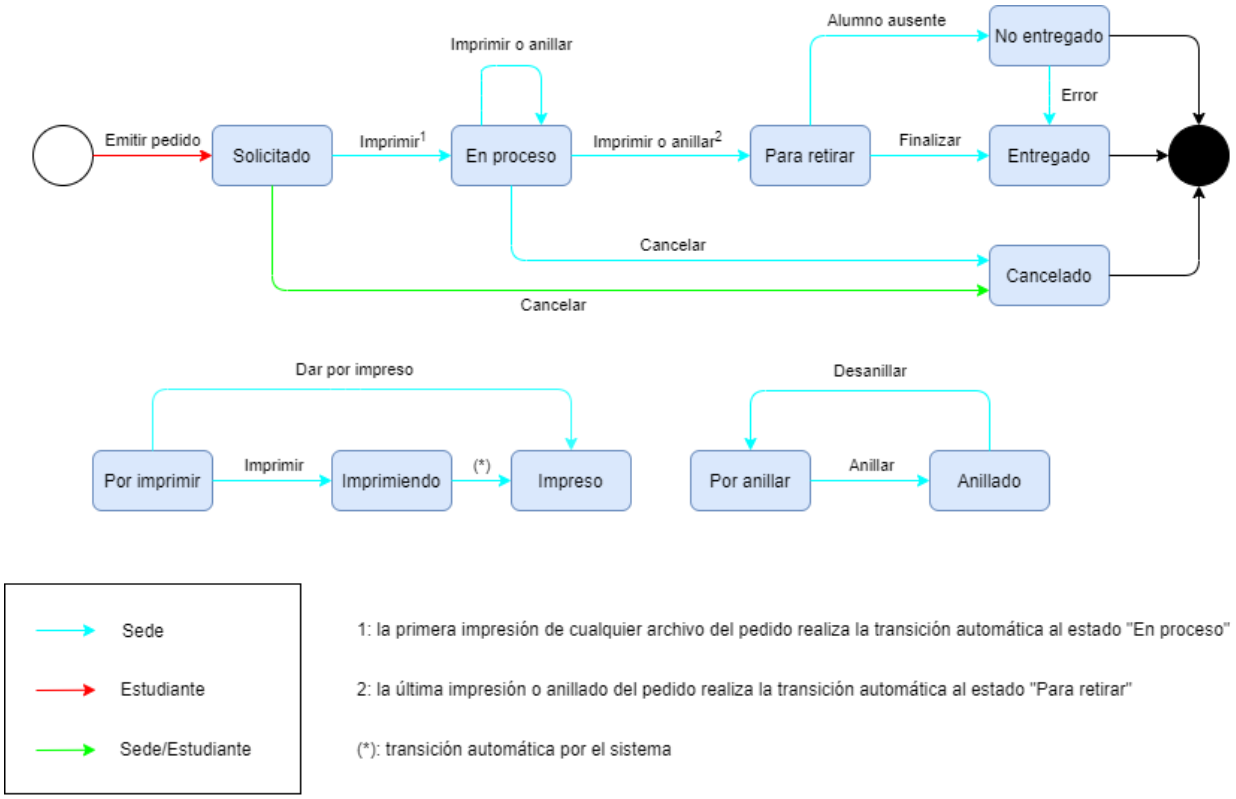



**Figura 15**. Diagrama de estados del ciclo de vida del pedido

Las transiciones de estado respecto al **pedido** son originadas a través de los siguientes eventos:

| **Estado** | | **Evento** |
|---|---|---|
| **Origen** | **Destino** | |
| Inicio | Solicitado | El estudiante crea un pedido en el sistema. |
| Solicitado | En proceso | El empleado imprime el primero de los archivos del pedido. |
| Solicitado | Cancelado | El estudiante o el empleado cancela el pedido. |
| En proceso | Para retirar | El empleado imprime el último archivo o se anilla el último grupo de anillado del pedido. |

| | | |
|---|---|---|
| En proceso | Cancelado | El empleado cancela el pedido por alguna condición dada. |
| Para retirar | Entregado | El estudiante retira el pedido en la sede designada. |
| Para retirar | No entregado | El empleado califica al pedido como “No entregado” dado que el estudiante no ha pasado a retirarlo en un determinado tiempo. |
| No entregado | Entregado | El estudiante pasa a retirar un pedido calificado como “No entregado”. |

Respecto a la **impresión de archivos** las transiciones de estados se rigen por el siguiente comportamiento:

| **Estado** | | **Evento** |
|---|---|---|
| **Origen** | **Destino** | |
| Por imprimir | Imprimiendo | El empleado envía a imprimir un archivo a la impresora seleccionada. |
| Imprimiendo | Impreso | La impresión finaliza correctamente y el sistema actualiza el estado del archivo. |
| Por imprimir | Impreso | El empleado da por impreso un archivo al tenerlo ya disponible en forma física en la sede. |

Por último, para los **grupos de anillados** las transiciones disponibles son las siguientes:

| **Estado** | | **Evento** |
|---|---|---|
| **Origen** | **Destino** | |
| Por anillar | Anillado | El empleado da por anillado un conjunto de archivos del pedido. |
| Anillado | Por anillar | El empleado anula un anillado producto de una confusión o error de su parte. |

Algunos puntos a destacar:

- Un pedido solicitado puede ser cancelado por un estudiante mientras no haya sido puesto en proceso. Una vez comenzado, solo usuario sede podrá cancelar el mismo.
- Para minimizar posibles errores por parte de los empleados se permite deshacer el anillado de un conjunto de archivos vía sistema. Por el contrario, esto no es posible para el caso de impresiones de archivos, dado que el sistema gestiona todo de su lado y no requiere intervención del usuario.
- Se permite dar por impreso un pedido en caso que el empleado tenga en su poder un archivo que cumpla con la configuración solicitada. Esto puede ser realmente útil en épocas de picos estacionales en los cuales puede ser conveniente imprimir cierto material de forma anticipada.

#### 6.2.1.2. Parametrización del sistema

La plataforma cuenta con una gestión de paramétricas que le permiten tener un comportamiento dinámico en función del contexto y las necesidades actuales. En particular abarca tres ejes principales:

- Gestión de pedidos:
  - Saldo negativo disponible para el estudiante y adaptable según el contexto inflacionario actual.
- Gestión de archivos:
  - Tamaño máximo que un archivo puede tener para ser subido a la plataforma.
  - Espacio de almacenamiento disponible para un usuario de tipo cátedra.
- Gestión de becados:
  - Copias inicialmente disponibles para el becado, factibles de ser actualizadas de un cuatrimestre a otro.
- Gestión de links a redes sociales y material del CEI que se desea volver accesible desde la plataforma.

### 6.2.2. Diseño técnico

Una vez seleccionadas las tecnologías que pasaron a formar parte del stack tecnológico de la solución fue necesario comenzar a darle forma a la misma. Para ello se realizó un diseño iterativo que buscó definir, en primera instancia, la **arquitectura** sobre la que el producto final se sustenta y, posteriormente, hacer foco en cada uno de sus componentes para aplicar las mejores prácticas existentes para su implementación.

En esta etapa fueron tenidos en cuenta multitud de **patrones arquitectónicos y de diseño** que permitieron no reinventar la rueda y poder canalizar esfuerzos en aquellos aspectos que se escaparon de lo estándar. El resultado de este proceso permitió establecer una base sobre la cual cada integrante del equipo luego podría seguir implementando la solución.

#### 6.2.2.1. Arquitectura

La base de todo sistema comienza con la definición de la **arquitectura** que le dará marco a la misma. A través de las decisiones de alto nivel que se tomen se podrá garantizar o imposibilitar los objetivos que persigue el producto a desarrollar. Es por ello que un buen análisis previo es un requisito casi excluyente para poder tomar partido en estas tareas.

En esta sección se expondrán aquellas **arquitecturas de referencia** que fueron tomadas en cuenta dados los requisitos y restricciones identificados en etapas anteriores. Luego, teniendo claros los conceptos y patrones de diseño que sirven como base se procede a describir la arquitectura diseñada justificando cada una de sus partes, explicando cómo interactúan entre sí y qué capacidad requerida satisfacen.

##### 6.2.2.1.1. Arquitectura de referencia

Como resultado del análisis realizado se determinó rápidamente que la arquitectura en la cual basarse es bastante estándar y conocida: **Multitier architecture** o **Arquitectura de múltiples capas**, una especialización de la más genérica **Arquitectura Cliente-Servidor** (Tanenbaum, 2006).

A nivel macro, una típica arquitectura multicapa puede verse así:

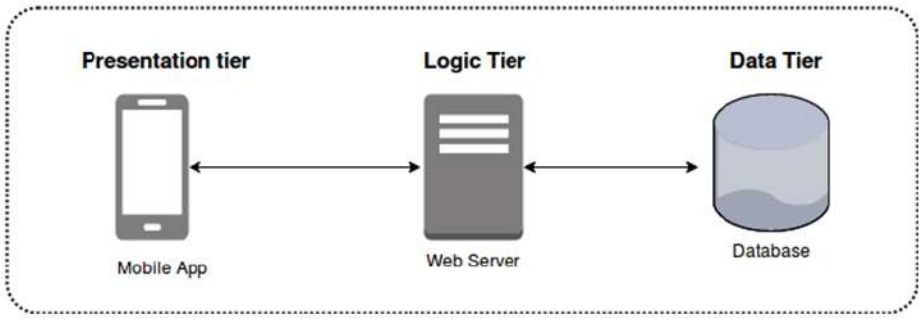



**Figura 16**. Multilayer Architecture

En este tipo de arquitectura se decide separar las capas de **presentación**, **procesamiento** y **almacenamiento** en tres componentes que están físicamente separados. Cada capa puede ser desarrollada y mantenida por separado, dándole a los desarrolladores la suficiente flexibilidad para su modificación e incluso la posibilidad de añadir nuevos componentes entremedio.

Como bien se mencionó, la Arquitectura Multicapa es una especialización de la clásica Arquitectura Cliente-Servidor, la cual se basa en dos componentes bien diferenciados conectados a través de un medio.

- **Cliente**: demandante de servicios, este cliente puede ser un ordenador como también una aplicación de informática, la cual requiere información proveniente de la red para funcionar.
- **Servidor**: proveedor de servicios, este servidor a su vez puede ser un ordenador o una aplicación informática la cual envía información a los demás agentes de la red.
- **Red**: conjunto de clientes, servidores y base de datos unidos de una manera física o no física en el que existen protocolos de transmisión de información establecidos.
- **Protocolo**: conjunto de normas o reglas y pasos establecidos de manera clara y concreta sobre el flujo de información en una red estructurada.

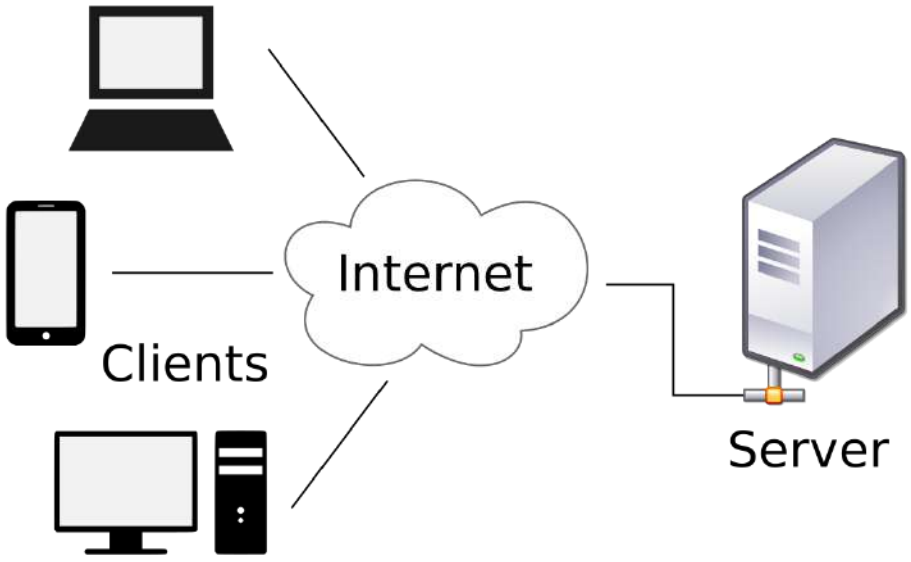



**Figura 17**. Arquitectura Cliente-Servidor

En la imagen anterior Internet funciona como la red o medio a través de la cual los clientes se comunican con el servidor. Los protocolos utilizados por la solución diseñada son HTTP, WS e IPP.

Los clientes pueden ser de diferentes tipos, tales como se aprecia: laptops, PCs de escritorio, celulares, etc.

#### 6.2.2.1.2. Arquitectura de solución

En base a la arquitectura de referencias se procedió al diseño de la misma. A continuación se presenta el **Diagrama de Componentes** que permite identificar los principales elementos que sustentan la solución.

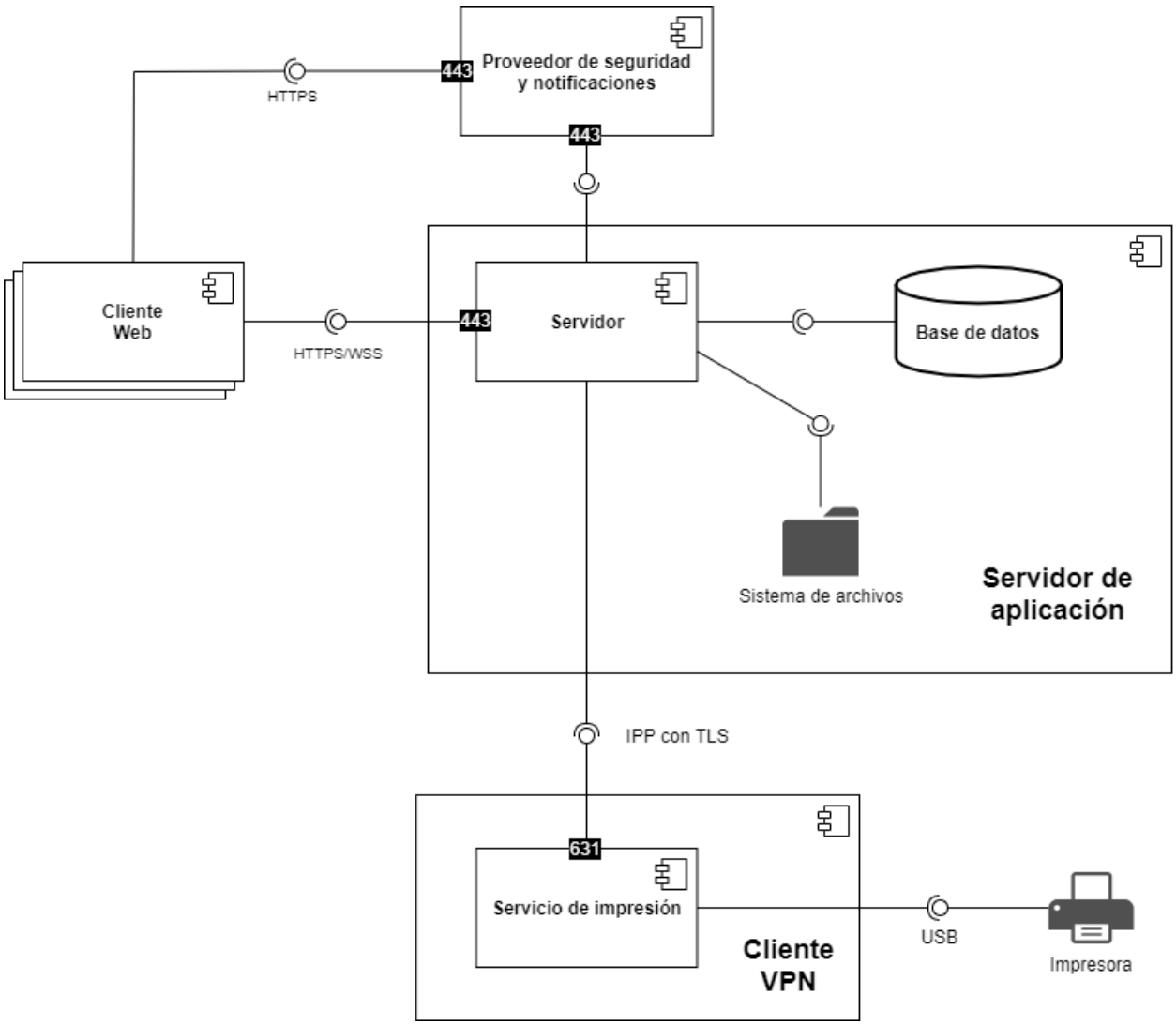



**Figura 18**. Diagrama de Componentes

Tanto los **clientes web**, el **servidor** y la **base de datos** tienen una correspondencia uno a uno con la Arquitectura Multi-Capa estándar. Los clientes y el servidor se comunican a través de los protocolos seguros HTTPS y WSS, los cuales soportan la APIs RESTful y de Web Sockets que expone la capa de procesamiento. La comunicación interna del servidor se realiza vía sockets, en lugar de viajar por una red.

El **sistema de archivos** aparece como un nuevo componente que se encarga exclusivamente del almacenamiento de los archivos que conforman el repositorio de información del sistema. Se optó por esta alternativa dado que el espacio necesario de almacenamiento ronda los **37 GB**, motivo suficiente para asegurar que será más performante y eficiente su almacenamiento en disco que en la propia base de datos, la cual resuelve otro tipo de necesidades. Los archivos serán depositados en un directorio especial con **acceso exclusivo** por determinados usuarios. Por otro lado, una **copia de respaldo** será generada cada cierto tiempo en un dispositivo externo para solventar cualquier tipo de problema que pueda originarse en el repositorio.

El **proveedor de seguridad y notificaciones** que posteriormente será ocupado por Firebase se encuentra externo al servidor. Tanto las capas del cliente como del backend se comunican con el mismo a través de HTTPS, ya sea para hacer uso de las funcionalidades de gestión de usuarios y tokens y el envío de notificaciones a los dispositivos móviles.

Por último, el caso más emblemático está dado en cómo se diseñó la arquitectura para soportar la funcionalidad de impresión. La principal restricción que condiciona todo este proceso fue el hecho de que el servidor y las impresoras no estarán en el mismo segmento de red, por lo cual algún componente extra debió ser añadido para sortear esta incomunicación.

La **topología de red** de la Universidad Nacional de Mar del Plata (UNMdP) es compleja y la conexión entre dos segmentos de red de diferentes facultades es un desafío que necesita involucrar a varios actores extras para poder habilitar la visibilidad entre las mismas. Luego de un análisis de trade-offs se encontró que este problema de infraestructura puede ser resuelto sin depender del auxilio de personal extra. A través de una conexión VPN *sobre internet* es posible sortear todos estos obstáculos de red y directamente comunicar “impresoras clientes” con el servidor que guiará sus tareas.

En pos de que el diseño anterior sea viable es necesario que aquellos segmentos de red en los cuales están las impresoras cuenten con un dispositivo físico que haga de **middleware** entre el servidor y la propia fotocopiadora. Este dispositivo será el encargado de hacer de **cliente de la VPN** (la cual es servida desde el servidor principal quien es el único que posee una IP pública) y permitirá transmitir los trabajos de impresión a la impresora destino.

El middleware estará ejecutando un **servidor de impresión**, el cual permite aceptar trabajos de impresión, procesarlos y luego enviarlos a la impresora elegida. Este componente cuenta con la ventaja que puede adaptarse a los múltiples **drivers** de impresoras que existen y permite centralizar toda la gestión de trabajos en un único punto, de forma tal de independizar a los clientes de conocer el formato específico para comunicarse con cada dispositivo de impresión.

La comunicación entre el servidor y el servicio de impresión se realiza por sobre la **VPN** utilizando el protocolo **IPP**, de modo tal que toda la comunicación es cifrada. El último tramo de la solicitud viajará vía **USB** entre el servicio y la impresora, por lo que es necesario tener este tipo de conexión física previamente establecido.

A través de esta arquitectura es posible satisfacer todos los requerimientos funcionales y restricciones identificados en la etapa de análisis.

En el caso particular del sistema diseñado, la **arquitectura instanciada** con varias de las tecnologías definidas durante la etapa de análisis se puede ver en el siguiente diagrama:

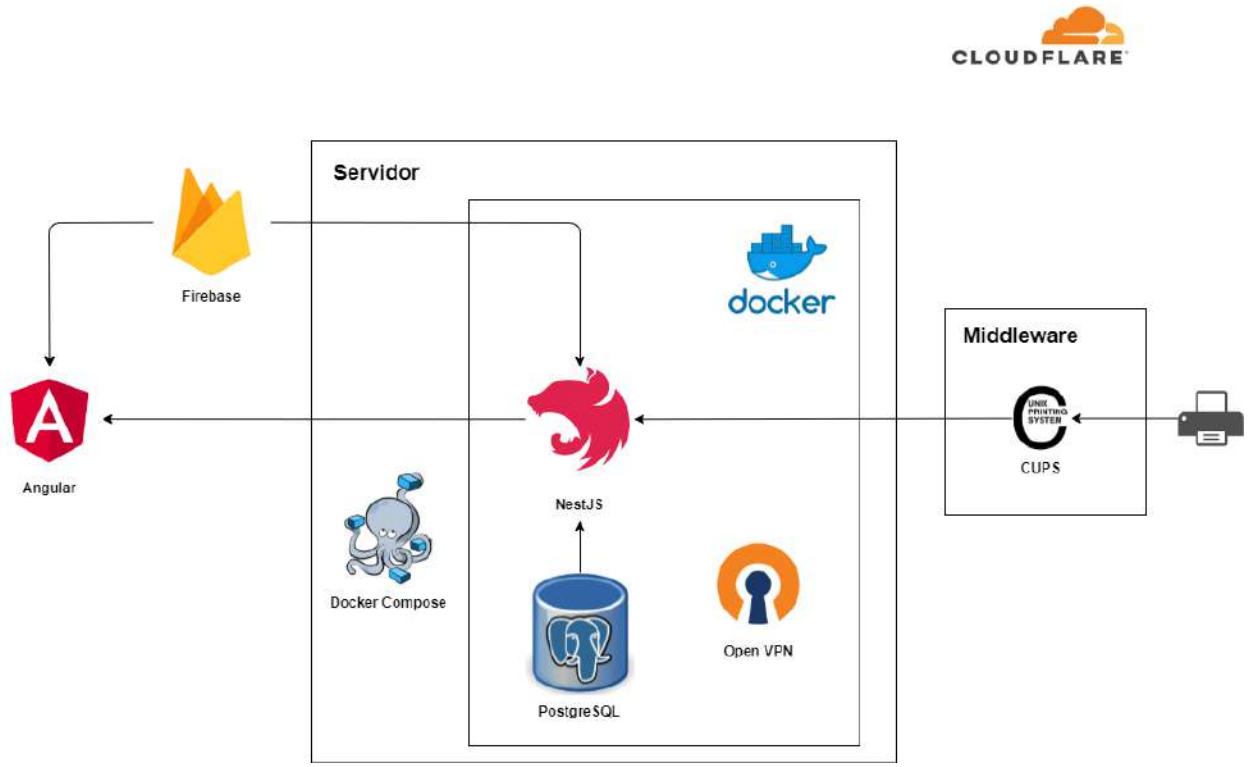



**Figura 19**. Runtime Architecture

Haciendo una comparación entre las **arquitecturas de referencias** presentadas anteriormente y la **arquitectura en tiempo de ejecución** podemos enumerar qué función cumple cada componente:

- **Angular** representa al **cliente** que hace uso de los servicios del **servidor**.
- **NestJS** ocupa el lugar del **servidor**, más específicamente la parte de cómputo del mismo.
- **PostgreSQL** cumple el rol de la **capa de almacenamiento** que soporta la operatoria de la aplicación web.
- **Firebase** es el servicio externo utilizado para autenticación, autorización y notificaciones.
- **CUPS** es el servicio de impresión que permite unificar en un único punto toda la gestión de trabajos y colas de impresión.

El resto de los componentes están vinculados con el runtime de la aplicación (**Docker**, **Docker Compose**), DNS (**Cloudflare**) o conectividad con las impresoras (**OpenVPN** y **CUPS**, cuya implementación no está alcanzada por el presente escrito).

En el anexo “Diseño de componentes” se encuentra una explicación detallada de la arquitectura interna de los componentes que forman los pilares de la arquitectura de solución. Además, en dicho anexo también se proporciona información sobre las decisiones e implementaciones llevadas a cabo en el diseño de las APIs que conectan la capa del Cliente y del Servidor.

# 7. Retrospectiva del proyecto

## 7.1. Resultados a nivel metodología, planificación, estimación y plazos

### 7.1.1. Metodología

La metodología de trabajo del proyecto que se decidió implementar ha originado como resultado el siguiente gráfico:

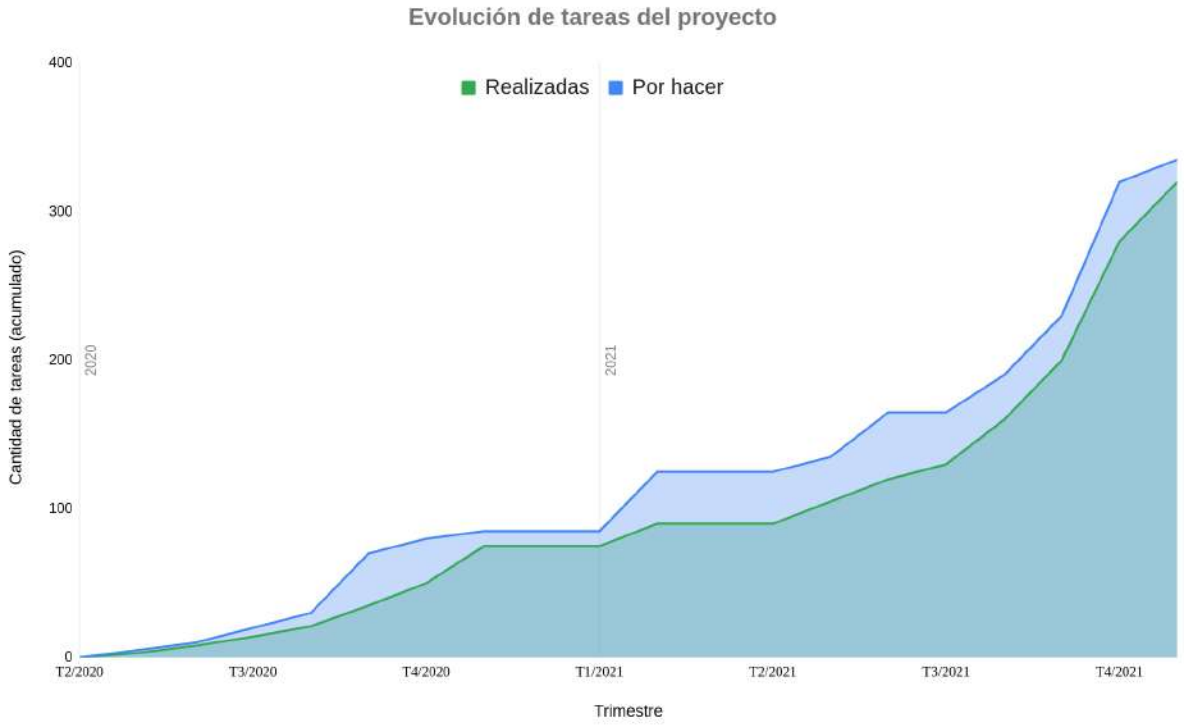



**Figura 20**. Evolución de tareas del proyecto

Dado que el modelo de trabajo estuvo influenciado fuertemente por Scrum, es preciso destacar que en el comienzo del proyecto se pudieron tener relevados los requisitos funcionales (traducidos a épicas) que regirían el desarrollo del proyecto. Sin embargo, la planificación de tareas más fina se sucedió iteración a iteración. Esto se observa en la figura ya que el número de tareas acumulado por hacer crece a medida que el tiempo avanza[1].

[1] El gráfico es un acumulado en el cual la tarea original siempre existirá y en algún momento evolucionará a un estado realizado y contribuirá ahora también a la curva verde. En cualquier instante de tiempo, las tareas por realizar surgen de la diferencia en el eje Y entre la curva azul y verde.
El objetivo final es que la curva verde se aproxime lo máximo posible a la azul y las tareas realizadas acumuladas igualen a las pendientes acumuladas.

Por otro lado, también se aprecia como la curva verde intenta en todo momento aproximarse a la azul a fin de poder cumplir con todas las tareas planificadas para cada iteración. Esto evidencia el trabajo iterativo que el equipo siguió durante el proyecto: identificar las tareas necesarias a llevar a cabo en un período de tiempo determinado, ejecutarlas, verificar lo realizado y luego volver a iterar.

Es posible concluir que la metodología de trabajo fue implementada exitosamente siguiendo los lineamientos que se definieron inicialmente, refinando la técnica iteración a iteración y logrando cada vez tareas más granulares y alcanzables dentro de un lapso de tiempo acordado con el resto del equipo.

### 7.1.2. Planificación

La planificación del proyecto puede ser contrastada contra la estimación inicial que se realizó (figura 1) a partir del siguiente gráfico:

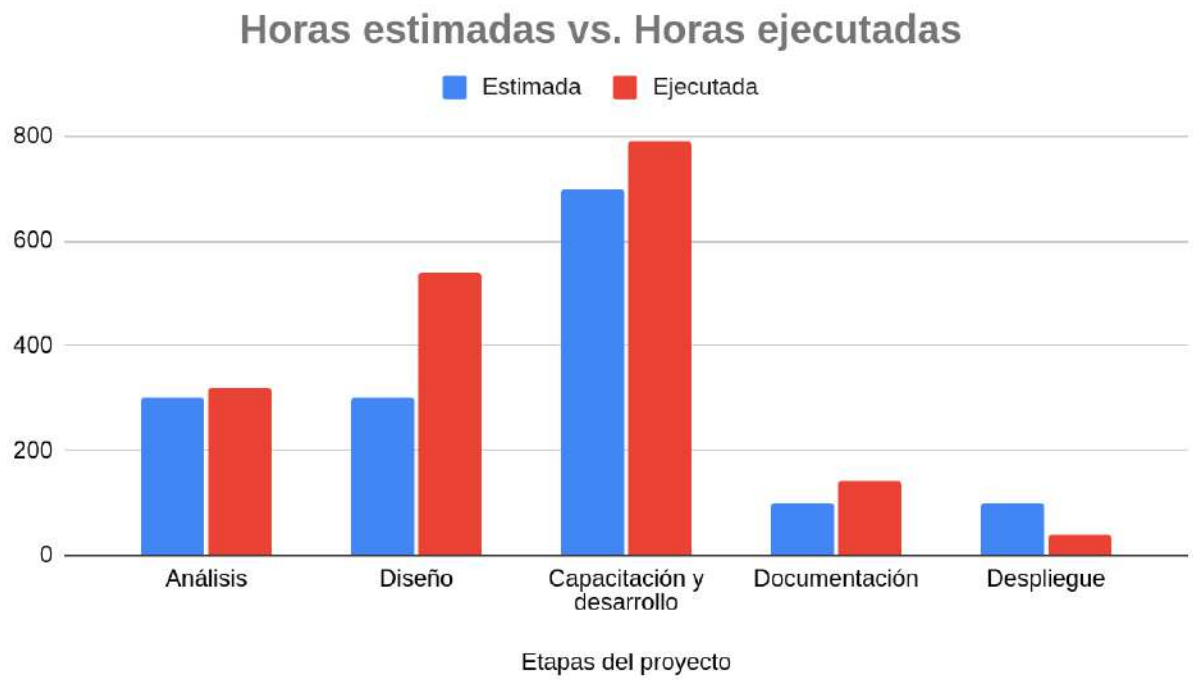



**Figura 21**. Variación entre horas estimadas y ejecutadas

Se puede observar que la etapa de **Análisis** estuvo cercana a lo planificado inicialmente, mientras que para la etapa de **Diseño** claramente se subestimó el esfuerzo necesario. Esto se debe en gran medida al hecho de que toda la reingeniería de procesos sumado al diseño de la plataforma, flujos de ejecución, diagramas de secuencia y otras temáticas ocupó gran parte del tiempo planificado.

La capacitación y desarrollo de la plataforma fue también subestimada, y si se considera el hecho de que algunas funcionalidades se han dejado para trabajos futuros también, da cuenta de que esta etapa hubiese requerido más horas de las dedicadas y hubiese afectado a la distribución final de porcentajes.

Por otro lado, la documentación también fue planificada por debajo de lo que realmente se necesitó, producto de generar multitud de diagramas y documentos para ayudar a la transferencia de conocimiento y posterior mantenimiento de la plataforma.

Finalmente, el despliegue no fue una etapa tan compleja como se esperaba, consiguiendo alcanzar los objetivos de esta fase para el lanzamiento del MVP con una poca cantidad de horas invertidas.

Se puede concluir que la planificación inicial no fue del todo acertada, identificando como principales causas de los desvíos la inexperiencia del equipo en este tipo de proyectos y el hecho de que se desconocía la complejidad del problema en una primera etapa. Todas estas circunstancias condujeron a alterar la dedicación que cada fase del proyecto requirió en última instancia.

Para la etapa de **Capacitación y Desarrollo** se puede mencionar un dato interesante. A continuación se puede apreciar la distribución de horas invertida en cada capa de la solución:


**Figura 22**. Porcentaje de tiempo invertido en cada capa de la arquitectura

El componente de **backend** es el que más trabajo requirió por parte del equipo, producto de la migración necesaria que debió efectuarse de una librería a un framework más robusto, tal como se comentó en la sección de Diseño Técnico.

El **frontend** tuvo la ventaja que parte del equipo ya tenía experiencia con las tecnologías utilizadas, por lo que su dedicación fue sensiblemente menor.

Por último, la **capa de datos** está representada como un componente extra y en ella se incluyen toda las tareas que comprenden a la creación de índices, análisis de consultas realizadas a la misma y su posterior tuning.

Se toma como aprendizaje de esta distribución de horas que el análisis inicial realizado sobre el framework de trabajo a utilizar para la capa del servidor no fue del todo eficiente, y que muchas de las horas extras invertidas pudieron ser salvadas si la solución hubiera estado basada desde un comienzo en **NestJS** como software base.

### 7.1.3. Estimación de tiempos

En cuanto a estimación de tiempos se puede observar en la siguiente figura la distribución final:


**Figura 23**. Estimación real vs ejecutada

La planificación inicial del proyecto estimó un total de 1500 horas para completarlo. Esta estimación contempló todas las tareas de **Análisis**, **Diseño**, **Implementación** y **Despliegue** de la solución, más la debida **Documentación** y escritura del Trabajo Final.

Como ya se ha mencionado, el inicio del proyecto contó con una ejecución organizada e incremental, con validaciones funcionales recurrentes con el cliente y con tiempos de entrega según lo pactado y planificado. Sin embargo, este flujo se vió seriamente perturbado con la aparición de la pandemia. Durante esta etapa, surgió la necesidad por parte del cliente de disponibilizar de manera inmediata un sistema con variaciones sustanciales en cuanto a los requisitos iniciales. Este producto dio origen al **MVP** que el equipo ya ha desplegado productivamente.

La decisión del CEI generó un impacto en la etapa de implementación puesto que parte del frontend tuvo que sufrir una **refactorización** para adecuar lo ya desarrollado con las necesidades específicas que debieron satisfacerse, con el consecuente aumento de horas de proyecto. Por otro lado, todas las tareas de despliegue debieron ser adelantadas para poder llevar a cabo la liberación del software pertinente.

Producto de todas las circunstancias previamente mencionadas, una decisión de alcance respecto a la completitud del Trabajo Final tuvo que ser tomada a fin de evitar una sobre ejecución de horas invertidas en la misma. Una vez superada la estimación inicial de 1500 horas y sabiendo que todavía estaba pendiente la etapa de escritura del presente documento, se decidió, a fin de poder dar un cierre al Trabajo Final, dejar pendientes algunas tareas de implementación finales de la plataforma, y, en su lugar, continuar con el escrito.

La decisión estuvo consensuada junto al CEI que gracias al reconocimiento del esfuerzo realizado para la entrega del MVP accedió a no contar con todas las funcionalidades pactadas inicialmente en el producto alcanzado por el presente trabajo y, en consecuencia, evaluar su implementación en futuros desarrollos junto a otro equipo de personas interesadas.

En base a todas las circunstancias anteriormente mencionadas se puede afirmar que el total de horas invertido en la ejecución del producto alcanzado y posterior escritura del Trabajo Final ronda las ***1800 horas***, presentando un desvío del 20% aproximadamente. Se deduce con este valor que la estimación inicial no estuvo acertada ya que la curva de aprendizaje necesaria en algunas etapas junto con la liberación de una versión preliminar de la misma incrementaron ligeramente los tiempos de proyecto estipulados.

### 7.1.4. Plazos y ritmo de trabajo

La comparativa de plazos se puede realizar a partir del estudio del siguiente gráfico:


**Figura 24**. Cantidad de horas acumuladas en el proyecto

En el análisis de viabilidad temporal realizado al inicio del proyecto se ha mencionado que el plazo del mismo debía estar cercano a un año, lo cual determina que la fecha ideal de cierre del presente trabajo hubiera estado rondando el segundo trimestre de 2021. Este ideal no pudo ser alcanzado por el equipo, quien para ese instante de tiempo llevaba un poco más de un tercio del proyecto realizado.

El diagrama además muestra un área roja por sobre la línea horizontal de 1500 horas que representa el exceso de horas invertidas en el proyecto producto de los desvíos e imprevistos experimentados.

En este segundo diagrama se puede observar cómo fue el **ritmo** del equipo, evidenciando la cantidad de horas dedicadas durante la ejecución del proyecto:

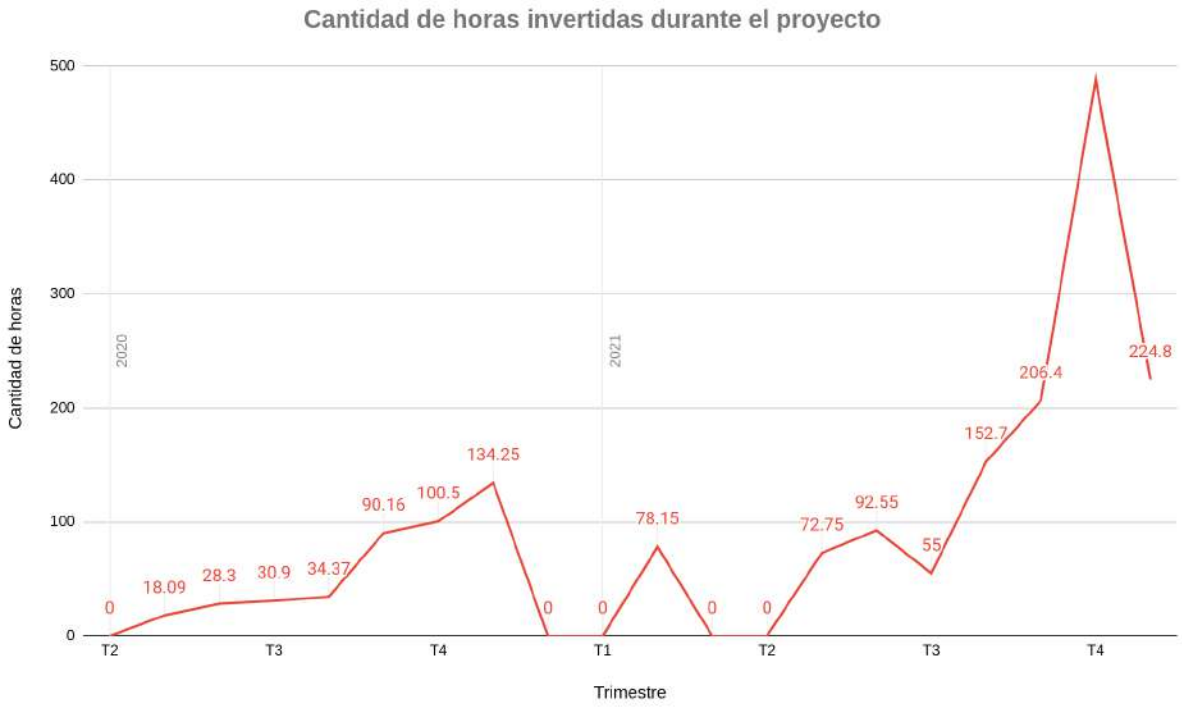



**Figura 25**. Cantidad de horas ejecutadas por período

Se puede observar como en la primera parte del proyecto se trabajó a un ritmo constante que fue aumentando hasta el **lanzamiento oficial del MVP** (cuarto trimestre de 2020), el cual como se ha mencionado fue una necesidad urgente del CEI.

Posterior a este primer despliegue se sucedieron dos **períodos de inactividad** que detuvieron en gran medida los avances que se venían generando (intervalos con cero horas ejecutadas). Esto estuvo fuertemente influenciado por **obligaciones externas del equipo** e **imprevistos** como vacaciones que no se tuvieron en cuenta inicialmente. Recién en el segundo trimestre de 2021 se observa una reactivación fuerte del proyecto que aumenta considerablemente durante el tercer trimestre, período en el que se lograron la mayor cantidad de tareas ejecutadas.

Se puede afirmar que el ritmo de trabajo del equipo no fue constante y la dedicación inicial prevista de 3 horas diarias no pudo ser cumplida en todo momento, experimentando períodos de fuerte, mediana y baja o nula actividad.

## 7.2. Cumplimiento de objetivos planteados

A partir de la comparación de los resultados obtenidos por la ejecución del proyecto con los objetivos generales y específicos planteados inicialmente se evidencia que la mayoría de ellos han sido cumplidos.

### 7.2.1. Proyecto

En base a los resultados conseguidos a nivel proyecto es posible afirmar que la metodología de trabajo fue implementada satisfactoriamente, logrando el equipo encontrar un marco de trabajo que le pudiera organizar las tareas por realizar y permitir un seguimiento de las mismas.

En cuanto a la planificación, estimación de horas y plazos de entrega se puede afirmar que no fueron los ideales. El ritmo de trabajo no fue constante ni el deseado, los plazos finales de entrega se extendieron más allá de lo previsto y la cantidad de horas invertidas también superó lo estimado inicialmente.

Si bien el equipo pudo haber realizado mejores estimaciones y previsto algunas situaciones, la realidad es que el desafiante contexto al cual se enfrentó (pandemia y trabajo exclusivamente remoto) sumado a redefiniciones y requisitos imprevistos del cliente fueron factores que influenciaron en gran medida a los resultados obtenidos. Es por esta razón que, dadas las circunstancias especiales que se sucedieron, se puede concluir que el desempeño y versatilidad de cada uno de los integrantes estuvo a la altura de lo solicitado y el desenlace final conseguido está dentro de lo que el equipo considera un buen proyecto de ingeniería.

### 7.2.2. Reingeniería de procesos

El rediseño de los procesos internos del CEI ha posibilitado minimizar la cantidad de errores y mejorar los puntos de dolor identificados. Esto permite reducir los tiempos de espera y maximizar los recursos disponibles en la organización estudiantil.

Si bien todavía no se cuenta con datos empíricos sobre el desempeño del nuevo proceso en la dinámica diaria del CEI, algunas mejoras se pueden destacar producto de la simple comparativa del antes y después. Si bien gran parte de las mismas ya han sido expuestas en la sección “Innovación del proceso”, aquí se dará una visión resumida de los logros.

Por parte del estudiante se ha logrado:

- Disminuir la cantidad de errores en los pedidos emitidos y las rectificaciones necesarias.
- Reducir al máximo los tiempos que invierte el estudiante, pasando de 4 tiempos de espera (dirigirse a la sede, esperar por un equipo de la sede, hacer cola en el mostrador y esperar la impresión) a 2 (dirigirse a la sede y hacer la cola el mostrador) en el caso de un pedido instantáneo, y aún mayor esta diferencia en el caso de señar el pedido.

- Reducir el número de interacciones entre el estudiante y el encargado del centro de copiado de 4 (solicitar y configurar el pedido, pagar/señar el mismo, consultar el estado y retirar) a 2 (carga inicial de saldo en la cuenta que se termina aplicando a varios pedidos y el retiro del propio pedido).
- Disponibilizar el repositorio de archivos pudiendo ser accesible desde cualquier dispositivo con acceso a Internet. Además, también es posible consultar el estado de un pedido por este mismo medio.
- Permitir a los estudiantes transferir dinero a fin de posibilitar los casos de uso más típicos que existen actualmente en la facultad: prestar dinero para luego ser devuelto y consolidar en un solo solicitante la emisión de un pedido grupal.
- Manejar saldo negativo para considerar el caso más usual en el que el monto total del pedido supera levemente el saldo disponible (una carga de saldo aquí arruinaría la experiencia de usuario).

Del lado de la sede se ha conseguido:

- Delegar la mayoría de las tareas operativas a la plataforma, reduciendo de esta manera las responsabilidades asignadas al empleado.
- Simplificar el proceso a través de la reducción de pasos en cada uno de los flujos que sigue el empleado. Esto ha generado, por un lado, una optimización de tiempos y, por otro, una disminución en la cantidad de errores que puedan originarse.
- La optimización realizada en el proceso permite lograr hacer un uso más eficiente de los recursos.
- Delegar a los integrantes de cada cátedra la responsabilidad de mantener el repositorio de archivos actualizado de su asignatura, evitando una gestión interna innecesaria con el CEI y posibles inconvenientes.
- La manipulación del dinero se ve minimizada a su mínima expresión producto de la acción concreta de carga de saldo.

Todos estas mejoras ponen de manifiesto el hecho de que la reingeniería de procesos era una etapa crucial en el proceso de Transformación Digital que necesitaba el CEI y que fue debidamente identificada, planteada y ejecutada por el equipo de desarrollo.

### 7.2.3. Producto

A través de la plataforma tecnológica implementada se ha logrado conseguir un producto que cumple con los requerimientos funcionales, no funcionales y restricciones identificadas.

Respecto a los pedidos se ha logrado la gestión punta a punta de los mismos. La gestión de los usuarios en general y de los becados en particular también fue resuelta exitosamente a través de una buena selección de servicios externos para no reinventar la rueda. El manejo de dinero dentro de la plataforma es otro pilar importante que conforma al sistema y que también se le ha dado solución satisfactoriamente, proporcionando los mecanismos de seguridad que tanto son necesarios.

Por otro lado, la plataforma está disponible para ser accedida desde dispositivos móviles y la web, cuidando especial atención a la UI y UX en cada uno. A través de la misma se ha democratizado todo el acceso a la información provista por el CEI, la cual es ahora accesible desde cualquier dispositivo remoto con acceso a internet.

Por último, la principal restricción identificada en el CEI dada por la falta de un equipo de soporte y operación que permita seguir manteniendo el sistema posterior a su despliegue, ha sido mitigada al máximo posible. Esta debilidad ahora puede ser atacada gracias a algunas decisiones de diseño efectuadas y contar con varios recursos útiles para entender, ajustar y evolucionar la plataforma:

- Transferencia de conocimiento con el CEI incluyendo un manual de usuario, capacitaciones, diagramas y documentación de la plataforma.
- El sistema informático es parametrizable a fin de que el administrador pueda alterar su funcionamiento sin recurrir a habilidades propias de un desarrollador.
- Del lado técnico se ha recurrido a tecnologías estándares del mercado con amplia comunidad y buena documentación, uso de las mejores prácticas, comentarios en el propio código, documentación de API, uso de arquitecturas validadas y reconocidas en el mercado, beta cerrada en un entorno similar a producción para identificar la mayor cantidad de errores posibles previo al despliegue final.
- Recomendaciones sobre los próximos pasos a seguir luego del lanzamiento oficial.

Todo esto contribuirá a poder realizar una entrega ordenada y segura del sistema.

En resumen, los objetivos planteados han sido alcanzados de forma satisfactoria a pesar de los desafíos y contratiempos experimentados. El producto alcanzado resuelve las necesidades analizadas y queda abierto y extensible para futuras mejoras al mismo.

# 8. Conclusiones

La ejecución del proyecto ha generado una multitud de resultados que pueden ser analizados para extraer conclusiones enriquecedoras.

## 8.1. Planificación, estimación y gestión del proyecto

La planificación y gestión del proyecto fueron aspectos desafiantes que el equipo tuvo que saber dar respuesta durante toda su ejecución. Condicionado en gran medida por situaciones ajenas al mismo, sucesivos ajustes fueron necesarios efectuar para llegar al mejor resultado posible.

Respecto a la planificación inicial y su contraste con el resultado final se puede concluir que la inexperiencia del equipo en cuanto a la dedicación que involucraría cada etapa sumado al desconocimiento del dominio del problema en profundidad, fueron factores cruciales que ocasionaron que los plazos y horas estimadas no puedan cumplirse. El contexto pandémico, las redefiniciones y modificación de prioridades por el cliente y la aparición de imprevistos como vacaciones, obligaciones externas o modificación de tecnologías atentaron de igual manera a la concreción de los objetivos planteados. A pesar de todo ello, el grupo de trabajo supo adaptarse satisfactoriamente y poder cumplir con cada una de las metas propuestas, ya sea identificando nuevas líneas de trabajo necesarias como la reingeniería de proceso o lanzando en el momento oportuno un MVP para toda la comunidad de la facultad.

En cuanto a la gestión del proyecto y sus tareas, es importante evidenciar que fue bastante desordenada en las primeras iteraciones. La imposibilidad de seguir iteraciones de trabajo tan rígidas afectaron al rendimiento del equipo. Este problema fue oportunamente detectado y se pudo adaptar el modelo de trabajo a tiempo para mejorar la dinámica y lograr la concreción de tareas que darían como resultado la reingeniería y producto alcanzados.

La conjunción de todas las circunstancias mencionadas anteriormente han permitido a los integrantes del equipo evolucionar en sus **competencias** respecto a la **gestión**, **planificación** y **ejecución** de proyectos de ingeniería focalizados en sistemas informáticos. Esta experiencia se convirtió en un excelente espacio para que los principales problemas que afligen a estas iniciativas puedan manifestarse y el equipo de trabajo pueda dar respuesta de la forma más apropiada.

## 8.2. Desarrollo profesional del equipo

En primera instancia es importante destacar que este proyecto fue la **primera experiencia real** que tuvo el equipo trabajando con un cliente. Esto obligó a sus integrantes a desarrollar varias **competencias** que permitieron que la etapa de análisis haya generado los activos conseguidos. Por otro lado, la buena relación con el CEI y el hecho de que este sistema beneficiaría a toda la facultad fueron importantes motivadores durante todo el desarrollo.

El hecho de haber **trabajado en equipo** y no individualmente ocasionó que cada integrante aprenda a tomar responsabilidad por sus tareas y rendir cuentas del trabajo realizado. Los constantes

encuentros para debatir y definir ciertas cuestiones permitieron desarrollar las **habilidades de análisis, diseño, planificación y ejecución** tan necesarias en el ámbito ingenieril. Por otro lado, la adecuación de la **comunicación** en función del receptor fue otro impulsor hacia poder transmitir las ideas de forma eficiente y coherente.

El aprendizaje obtenido durante el desarrollo del presente Trabajo Final es digno de mencionar. Los miembros del equipo tuvieron que recurrir en gran medida a un **aprendizaje continuo y autónomo** para aprender multitud de tecnologías, estándares, buenas prácticas y metodologías de trabajo utilizadas en la industria. Esto sumado a los conocimientos que cada integrante fue trayendo de su propia experiencia laboral fue sumamente valioso para todo el equipo y el sistema en sí, obteniéndose un producto robusto, usable, mantenible y extensible a futuro.

Además, a partir de todos los desafíos mencionados anteriormente se proporcionó el espacio para aplicar una gran cantidad de **conceptos adquiridos durante la carrera**. Sistemas distribuidos, bases de datos, comunicación entre sistemas, redes, análisis y diseño y otros tantos conceptos son algunos de los pilares en los que el equipo se basó para poder construir la solución obtenida.

Asimismo, el hecho de haber mitigado junto al CEI las principales debilidades que afligen su actividad y la generación de diferente documentación y capacitaciones sobre la plataforma que se han llevado a cabo dan cuenta de la **actitud profesional ética y responsable** que ha tenido el equipo con el órgano estudiantil.

Por último, se puede afirmar que con el desarrollo del proyecto fue factible adquirir y ejercer varias de las competencias ingenieriles inculcadas durante la carrera:

- Identificar, formular y resolver problemas de ingeniería
- Concebir, diseñar y desarrollar proyectos de ingeniería
- Gestionar, planificar, ejecutar y controlar proyectos de ingeniería
- Usar de manera eficaz las técnicas y herramientas de la ingeniería

El equipo se encuentra honrado de tener la oportunidad de devolver a la facultad todo el conocimiento transmitido a lo largo de estos años a través de una solución que beneficie a toda su comunidad.

# 9. Trabajos futuros

Tal como se ha mencionado en secciones anteriores, algunas funcionalidades han quedado pendientes y podrían ser implementadas posterior a la presentación de este proyecto. Dichas funcionalidades fueron consideradas en una primera instancia del proyecto, pero luego quedaron relegadas debido a las variaciones e imprevistos surgidos en el proyecto. De tal manera, es relevante destacar que estas capacidades han sido analizadas y diseñadas, fruto del entendimiento del modelo de negocio del CEI. A partir de la comparación con otras soluciones y prácticas del mercado se ha propuesto su adición como trabajos futuros. Por lo tanto, en el caso que los autores o partes ajenas a este proyecto deseen y sean autorizadas a implementar estos nuevos desarrollos podrán remitirse a este escrito y a la documentación para realizar dichas tareas.

Las funcionalidades que se propusieron en las primeras etapas del proyecto comprenden la gestión del pedido con las terminales de impresión, notificaciones por la plataforma, soporte para pago parcial de las órdenes o seña y capacidad para realizar encargos de impresión de archivos temporales subidos por los usuarios.

En primer lugar, se puede hacer foco en la integración de la gestión de los pedidos con las terminales de impresión. A través de los protocolos ya enunciados a lo largo del escrito se planeó realizar un desarrollo que tenga como fruto la conexión del servicio web con las impresoras, de manera tal que se gestione completamente la etapa de impresión y el ciclo de vida de la orden. Esto incluye manejo de errores (falta de papel y/o tinta en las impresoras, por ejemplo) y cambio de estado del pedido en base al resultado de la impresión.

Por otro lado, se detectó la opción de dotar al sistema de notificaciones. Estas tendrían el objetivo de comunicar a sus usuarios, en primera instancia, el estado de sus pedidos y cargas y transferencias de saldo. A partir de esta implementación sería posible comunicar dichos eventos en tiempo real a los consumidores.

Otro apartado que se debatió en la etapa de relevamiento es el soporte de pago parcial o seña de pedidos, generalmente encargos relativamente grandes y en consecuencia con un precio elevado (tales como libros, módulos, trabajos prácticos extensos o solicitudes de material que involucran gran cantidad de copias). Esta funcionalidad permitiría cobrar un monto parcial del total del pedido mediante el saldo virtual y luego abonar el restante al momento del retiro.

Cuando se analizó el proceso que se tiene actualmente se identificó que algunas órdenes emitidas están compuestas por archivos propios del usuario, no pertenecientes al sistema. En esta nueva adaptación del proceso, los usuarios podrían subir sus propios archivos que quisieran imprimir, configurarlos a su deseo y realizar el seguimiento en tiempo real de los mismos.

A diferencia de las mejoras nombradas anteriormente, existen dos funcionalidades a brindar por el sistema y su ecosistema que podrían traducirse en un mayor valor agregado del producto: la capacidad de ofrecer indicadores de performance sobre distintas métricas a registrar en el sistema y la posibilidad de que el producto pudiera servir a más facultades o centros de copiado que tengan un modelo de negocio similar al relevado.

Los **indicadores claves de rendimiento** (o **KPIs** en inglés) son un conjunto de valores medibles que demuestran cuán efectivamente se están alcanzando los objetivos de negocio de la organización a la cual el sistema sirve. En general, las empresas utilizan estos KPIs en múltiples niveles para evaluar su éxito en el cumplimiento de metas.

Para el caso particular del CEI, una necesidad detectada pero aún no cubierta por el sistema es la capacidad de ofrecer este tipo de indicadores para conocer el estado de algunas variables clave que pueden ayudar a entender el comportamiento de los usuarios y estar mejor preparados para el futuro. Para nombrar algunos ejemplos:

- Conocer la cantidad de pedidos solicitados en un intervalo de tiempo dado. Esto puede ayudar a distribuir el personal en función de la afluencia de pedidos, ya sea a nivel día de la semana o incluso en diferentes turnos durante la jornada.
- Conocer cuáles son los archivos (módulos, libros, parciales, etc.) más solicitados a fin de tenerlos impresos de antemano.
- Monitorear el uso de copias gratuitas por los becados a fin de mejorar la asignación de las mismas.

Como puede observarse, el conocimiento de estos valores podría ayudar en la detección de patrones, mejorar la toma de decisiones y permitir definir objetivos de forma más clara y precisa.

Por otro lado, una característica que no beneficiaría directamente al CEI pero si a otras instituciones, consta de adaptar la arquitectura del sistema al modelo **Multi-tenant**. Bajo esta nueva arquitectura sería posible que una única instancia del servidor sirva a más de un cliente (representado por un centro de estudiantes, un centro de copiado o cualquier otro tipo de organización con características y necesidades similares).

A través de esta modificación al sistema original, el mismo podría ahora hacer disponibles sus servicios a múltiples clientes a costa de un leve incremento en el consumo de sus recursos. Recordar que la instancia y su base de datos seguirán siendo las mismas, pero ahora adaptando su respuesta en función de quién es el consumidor.

Se encuentra que esta segunda oportunidad es un buen nicho de mercado todavía no explotado por ningún producto similar en el contexto local, por lo cual podría ser bastante factible su adopción.

Finalmente, se deja en claro que el sistema es totalmente extensible y, en caso de requerirse nuevas funcionalidades o la necesidad de implementar las ya analizadas anteriormente, las mismas podrán realizarse sin ningún inconveniente. Esto es fruto de la mantenibilidad con la que se diseñó toda la plataforma y el hecho de que el CEI cuenta con el código fuente de la misma.

# 10. Glosario

**Alfa**: primera versión de un programa. Es la fase donde un producto todavía es inestable, aguarda todavía a que se eliminen los errores o a la puesta en práctica completa de toda su funcionalidad, pero satisface gran parte de los requisitos.

**API**: del inglés Application Programming Interface, se trata de un conjunto de funciones y procedimientos que permiten la creación de aplicaciones que acceden a las características o datos de un sistema operativo, aplicación u otro servicio.

**BaaS**: modelo de servicio en la nube en el que los desarrolladores subcontratan todos los aspectos detrás de escena de una aplicación web o móvil para que solo tengan que escribir y mantener la interfaz. Los proveedores de BaaS proporcionan software previamente escrito para actividades que tienen lugar en servidores, como autenticación de usuarios, administración de bases de datos, actualización remota y notificaciones push (para aplicaciones móviles), así como almacenamiento y alojamiento en la nube.

**Becado**: estudiante de la facultad que posee una beca otorgada por el CEI, la cual se traduce en copias gratis disponibles por cuatrimestre.

**Contenedor**: unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable de un entorno informático a otro.

**Driver de arquitectura**: consideraciones que deben tenerse en cuenta para el sistema de software que son importantes desde el punto de vista arquitectónico. Conducen y guían (*drive*) el diseño de la arquitectura del software. Los drivers de arquitectura describen lo que se está haciendo y por qué se está haciendo. Entre ellos encontramos: restricciones técnicas, requerimientos funcionales, atributos de calidad, restricciones de negocio.

**Kanban**: herramienta para mapear y visualizar el flujo de trabajo, dentro del cual las tareas pueden encontrarse en uno de tres estados posibles generalmente (“Por hacer”, “En progreso”, “Hecha”).

**MVP**: Minimum Viable Product o su traducción “producto viable mínimo” es un producto con suficientes características para ser lanzado y satisfacer a los clientes iniciales, y proporcionar retroalimentación para el desarrollo futuro.

**On-premise**: software que se instala y se ejecuta en computadoras en las instalaciones de la persona u organización que usa el software, en lugar de en una instalación remota, como una granja de servidores o la nube.

**ORM**: técnica de programación que consiste en la transformación de las tablas de una base de datos en una serie de entidades utilizando un lenguaje de programación orientado a objetos. Esto crea, en efecto, una base de datos virtual que puede ser utilizada dentro del lenguaje de programación para facilitar las consultas a la base de datos.

**PWA**: tipo de software de aplicación entregado a través de la web, construido utilizando tecnologías web comunes que incluyen HTML, CSS y JavaScript. Está diseñado para funcionar en cualquier plataforma que utilice un navegador compatible con los estándares, incluidos dispositivos móviles y de escritorio.

**SaaS**: modelo de distribución de software en el que un proveedor de nube aloja aplicaciones y las pone a disposición de los usuarios finales a través de Internet.

**Scrum**: marco de trabajo que define un conjunto de eventos, prácticas y roles que puede tomarse como conjunto base para definir el proceso de producción que usará un equipo de trabajo dentro de un proyecto

**Sprint**: período corto, encuadrado en el tiempo, en el que un equipo de Scrum trabaja para completar una cantidad determinada de trabajo.

**Stakeholder**: persona, organización o empresa que tiene interés en el sistema en desarrollo.

**Tenant**: el término "software multitenancy" se refiere a una arquitectura de software en la que una única instancia de software se ejecuta en un servidor y atiende a múltiples tenants. Los sistemas diseñados de esta manera a menudo se denominan compartidos (en contraste con dedicados o aislados). Un tenant es un grupo de usuarios que comparten un acceso común con privilegios específicos a la instancia de software.

**UI**: “User Interface" o su traducción “interfaz de usuario” es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

**UX**: término en inglés que se expande a “User Experience” y se traduce como “experiencia de usuario”. Es el conjunto de factores y elementos relativos a la interacción del usuario con un entorno o dispositivo concretos, dando como resultado una percepción positiva o negativa de dicho servicio, producto o dispositivo.

**Web Socket**: tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.

# 11. Bibliografía


- Kendall K. y Kendall J. (1988). *Análisis y Diseño de Sistemas*. Octava edición. Pearson.
- Sommerville I. (1988). *Ingeniería del Software*. Novena edición. Pearson.
- Pressman R. (2010). *Ingeniería del Software. Un enfoque práctico*. Séptima edición. McGraw-Hill.
- Project Management Institute (2014). *Guía de los Fundamentos Para la Dirección de Proyectos (Guía del PMBOK®)*. Quinta edición. Project Management Institute
- Van Steen M. y Tanenbaum A. S. (2006). *Sistemas Distribuidos. Principios y Paradigmas*. Segunda edición. Prentice Hall.
- Coulouris G., Dollimore J. y Kindberg T. (2001). *Sistemas Distribuidos, conceptos y diseño*. Tercera edición. Addison-Wesley.
- Larman C. (2001). *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. Segunda edición. Prentice Hall.
- Elmasri R. y Navathe S. B. (1989). *Fundamentos de Sistemas de Bases de Datos*. Quinta edición. Addison-Wesley.
- Gamma E., Vlissides J., Helm R. y Johnson R. (1994) *Patrones de Diseño. Elementos de software orientado a objetos reutilizable*. Primera edición. Addison-Wesley.
- Top 10 Most Common Requirements Elicitation Techniques (6 de agosto de 2021). Recuperado el 10/08/2021 de https://www.softwaretestinghelp.com/requirements-elicitation-techniques/
- NestJS Docs (2017-2021). Recuperado el 15/12/2020 de https://docs.nestjs.com/
- Models for hierarchical data (20 de mayo de 2010). Recuperado el 07/08/2021 de https://www.slideshare.net/billkarwin/models-for-hierarchical-data
- Firebase Authentication (2021). Recuperado el 04/11/2020 de https://firebase.google.com/docs/auth
- Firebase Cloud Messaging (2021). Recuperado el 04/11/2020 de https://firebase.google.com/docs/cloud-messaging
- Node.js Best Practices (2021). Recuperado el 05/12/2020 de https://github.com/goldbergyoni/nodebestpractices
- Best practices for writing Dockerfiles (2021). Recuperado el 23/01/2021 de https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- Containerized development with NestJS and Docker (22 de enero de 2020). Recuperado el 23/01/2021 de https://blog.logrocket.com/containerized-development-nestjs-docker/
- API Design Standard (2021). Recuperado el 17/01/2021 de https://api.gov.au/standards/national_api_standards/index.html
- OpenAPI Specification (2021). Recuperado el 28/12/2020 de https://swagger.io/specification/
- Brian Mulloy (2021). *Web API Design*.
- Google Cloud (2018). *Web API Design: The Missing Link*.
- Web Sockets (2021). Recuperado el 25/05/2021 de https://socket.io/docs/v3/
- Multitier architecture (28 de abril de 2021). Recuperado el 25/07/2021 de https://en.wikipedia.org/wiki/Multitier_architecture

- Progressive Web Apps (2021). Recuperado el 22/12/2020 de https://web.dev/progressive-web-apps/
- Scrum Framework (2021). Recuperado el 08/02/2021 de https://www.scrum.org/resources/what-is-scrum
- What is a REST API? (8 de mayo de 2020). Recuperado el 03/08/2021 de https://www.redhat.com/en/topics/api/what-is-a-rest-api
- What is GraphQL? (2021). Recuperado el 03/08/2021 de https://www.redhat.com/en/topics/api/what-is-graphql
- The WebSocket API (WebSockets) (12 de agosto de 2021). Recuperado el 29/07/2021 de https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- What is IPP? (2021). Recuperado el 11/08/2021 de https://www.pwg.org/ipp/ippguide.html#what-is-ipp
- Service Worker API (21 de julio de 2021). Recuperado el 09/08/2021 de https://developer.mozilla.org/es/docs/Web/API/Service_Worker_API
- Introduction to progressive web apps (23 de julio de 2021). Recuperado el 17/08/2021 https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction#advantages_of_web_applications

# Proyecto Final

## Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería

# Casos de uso

Mar del Plata, 21 de Diciembre de 2021

# ÍNDICE

# 1. Objetivo

El objetivo del presente documento es exponer los casos de uso identificados y documentados para el desarrollo de la plataforma tecnológica en cuestión.

# 2. Listado de casos de uso

A continuación se expone el conjunto de casos de uso desarrollados, ordenados en función del requisito funcional al cual corresponden.

### 2.1. Flujo de registración, login y recupero de contraseña.

| **Requisito funcional** | **#1: Flujo de registración, login y recupero de contraseña.** |
|---|---|
| **Actores** | Estudiante y becado (registración), todos los actores para login y recupero de contraseña. |
| **Descripción** | Los estudiantes y becados deben poder registrarse y acceder a los servicios del sistema. La plataforma deberá proveer un módulo de registro flexible tal que permita autenticar a los usuarios con sus redes sociales con el fin de lograr un proceso inicial simple y accesible. |
| **Problemáticas que soluciona** | 1.4, 2.3, 3.2: el sistema disminuirá la tasa de errores. El registro del usuario es el primer paso para acceder a este servicio.<br>3.1: el sistema agilizará todo el proceso actual.<br>3.3: a través del registro se podrán generar y consultar los pedidos. |

| **Caso de uso** | #1.1 El usuario estudiante/becado desea registrarse en el sistema para acceder a las funcionalidades del mismo. |
|---|---|
| **Curso normal de eventos** | |

| **Acción del actor** | **Respuesta del sistema** |
|---|---|
| Usuario accede al sitio web del sistema | Devuelve y muestra la página inicial. |
| Usuario selecciona para registrarse. | |
| Usuario ingresa email y contraseña deseada. | Se realizan las validaciones necesarias. En caso de no existir el usuario en el sistema:<br>- Se crea usuario en el sistema asignándole el rol adecuado.<br>- Se envía un mail de confirmación al usuario.<br>Se informa al usuario de esta última acción. |
| Usuario ingresa a su casilla de correo, busca y abre el correo pertinente. Luego, clickea el link de verificación. | El sistema verifica la validez del token embebido en el link y devuelve un mensaje de éxito. |
| Usuario ingresa al portal de login, ingresa sus credenciales e intenta loguearse. | Se verifican las credenciales. En caso de éxito, se devuelven tokens de sesión. |
| Usuario es dirigido a la home del sistema. | |
| **Cursos alternos** | |
| **Mail ya registrado en el sistema** | |
| | Se muestra un mensaje de error al usuario para que intente con un nuevo correo electrónico. |
| **Token embebido en link de confirmación de email es inválido** | |
| | Se muestra mensaje de error al usuario indicando que el email no ha podido ser verificado.<br>Se le brinda la posibilidad al usuario de reenviar el mail de confirmación. |
| Usuario hace click en enviar mail de confirmación. | |
| Continúa el proceso en el flujo normal de eventos. | |
| **Credenciales inválidas** | |
| | El sistema informa que la combinación de email y contraseña suministradas son incorrectas. |

| | |
|---|---|
| El usuario reingresa las credenciales e intenta loguearse nuevamente. | |
| Continúa el proceso en el flujo normal de eventos. | |

| **Caso de uso** | #1.2 El usuario desea loguearse en el sistema para acceder a las funcionalidades del mismo |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| Usuario ingresa al portal de login, ingresa sus credenciales e intenta loguearse. | Se verifican las credenciales. En caso de éxito, se devuelven tokens de sesión. |
| Usuario es dirigido a la home del sistema. | |
| **Cursos alternos** | |
| **Credenciales inválidas** | |
| | El sistema informa que la combinación de email y contraseña suministradas son incorrectas. |
| El usuario reingresa las credenciales e intenta loguearse nuevamente. | |
| Continúa el proceso en el flujo normal de eventos. | |

| **Caso de uso** | #1.3 El usuario desea recuperar su contraseña para posibilitar nuevamente el acceso a la plataforma |
|---|---|
| **Curso normal de eventos** | |

| **Acción del actor** | **Respuesta del sistema** |
|---|---|
| Usuario ingresa al portal de login e indica que desea recuperar su contraseña. | |
| Ingresa el correo electrónico donde recibirá un mail para continuar con el proceso y presiona sobre el botón “Recuperar”. | Se valida que el email ingresado corresponda a un usuario existente en el sistema. |
| | Se envía un mail a la casilla de correo del usuario y se informa al usuario. |
| Usuario hace click en el link embebido en el mail recibido. | |
| Es redirigido a una pantalla donde puede ingresar la nueva contraseña deseada. Confirma la operación. | Se actualiza la contraseña en la base de datos y se devuelve un mensaje de éxito.<br>Se redirige al usuario a la pantalla de login. |
| **Cursos alternos** | |
| **Email no existe en el sistema** | |
| | El sistema informa al usuario que en caso de existir el email en el sistema, un mail ha sido enviado.<br>Al no existir el usuario asociado a ese email, ningún correo es enviado. |
| **Correo no llega a la casilla y el usuario no cerró la vista de recupero de contraseña** | |
| El usuario vuelve a presionar sobre el botón “Recuperar” sin haber borrado el email asociado | Se valida que el email ingresado corresponda a un usuario existente en el sistema. |
| | Se envía un mail a la casilla de correo del usuario y se informa al usuario. |
| **Correo no llega a la casilla y el usuario cerró la vista de recupero de contraseña** | |
| El usuario vuelve a realizar el flujo siguiendo el curso normal de eventos descrito anteriormente | |

### 2.2. ABM de usuarios

| **Requisito funcional** | **#2: ABM de usuarios** |
|---|---|
| **Actores** | Admin |
| **Descripción** | Los administradores del sistema podrán dar de alta, modificar e incluso eliminar usuarios.<br>Los estudiantes y becados se dan de alta exclusivamente a través del flujo de registración.<br>Las cátedras, usuarios sede y admins soportan ABM completo. |
| **Problemáticas que soluciona** | 1.3: las cátedras y usuarios sede podrán acceder al sistema y actualizar el material periódicamente.<br>1.5: poder realizar asignación de copias a los becados y controlar su uso.<br>2.1: los usuarios sede ya no deberán realizar algunas tareas repetitivas, que quedarán a cargo de otro tipo de usuarios.<br>5.1: el alta de la cátedra es el primer paso hacia poder gestionar el material directamente a través del sistema. |

| **Caso de uso** | #2.1 El usuario administrador desea modificar los datos del estudiante para actualizar su información |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El administrador busca el usuario de tipo estudiante que desea modificar. | Despliega listado de usuarios que cumplen con el patrón de búsqueda. |
| Selecciona estudiante a modificar. | Despliega detalles de los datos del usuario con campos editables. |
| Admin modifica aquellos campos que considere necesarios. Intenta guardar los cambios. | Verifica que tanto el email como el DNI no estén repetidos. |
| | Devuelve mensaje de éxito al usuario, haciendo |

| | |
|---|---|
| | efectivos los cambios. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |
| Continúa el proceso en el flujo normal de eventos. | |
| **Email o DNI repetido** | |
| | Se informa al usuario que el email o DNI ya están registrados. |
| El usuario modifica los campos necesarios e intenta guardar nuevamente los cambios. | |
| Continúa el proceso en el flujo normal de eventos. | |

| | |
|---|---|
| **Caso de uso** | #2.2 El usuario administrador desea promocionar a un estudiante para otorgarle la beca |
| **Precondición** | Debe existir el usuario estudiante que se desea promover a becado. |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El administrador busca el usuario que desea convertir a becado, a través del email, DNI o nombre y apellido. | Devuelve listado de los usuarios que coinciden con los parámetros de búsqueda. |
| Selecciona aquel usuario que desea convertir a becado, pudiendo ingresar cantidad de copias iniciales disponibles que tendrá. | Se promueve el estudiante a becado y se informa al usuario del éxito de la operación. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |

| | |
|---|---|
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |
| Continúa el proceso en el flujo normal de eventos. | |

| | |
|---|---|
| **Caso de uso** | #2.3 El usuario administrador desea modificar los datos del becado para actualizar su información |
| **Curso normal de eventos** | |
| **Acción del actor** | |
| El administrador busca el becado al cual desea modificar sus atributos (básicos del usuario + específicos del estudiante y becado). | Devuelve listado de usuarios que cumplen con el patrón de búsqueda. |
| Selecciona becado a modificar. | Despliega detalles de los datos del usuario con campos editables. |
| Admin modifica aquellos campos que considere necesarios. Intenta guardar los cambios. | Verifica que tanto el email como el DNI no estén repetidos. |
| | Devuelve mensaje de éxito al usuario, haciendo efectivos los cambios. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |
| Continúa el proceso en el flujo normal de eventos. | |
| **Email o DNI repetido** | |
| | Se informa al usuario que el email o DNI ya están registrados. |
| El usuario modifica los campos necesarios e intenta guardar nuevamente los cambios. | |

| | |
|---|---|
| Continúa el proceso en el flujo normal de eventos. | |

| **Caso de uso** | #2.4 El usuario administrador desea degradar un becado al rol de estudiante para quitarle la beca |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El administrador busca el becado que desea degradar a estudiante, a través del email, DNI o nombre y apellido. | Devuelve listado de los usuarios que coinciden con los parámetros de búsqueda. |
| Selecciona aquel usuario que desea degradar a estudiante. | Se degrada el becado a estudiante y se informa al usuario del éxito de la operación. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |
| Continúa el proceso en el flujo normal de eventos. | |

| **Caso de uso** | #2.5 El usuario administrador desea crear un usuario de tipo cátedra para darle acceso al sistema |
|---|---|
| **Precondición** | Debe existir la materia a la cual se desea vincular el nuevo usuario de tipo cátedra. |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| Se dirige a la sección para dar de alta un nuevo usuario cátedra. | |
| Busca la materia a la cual será asociada la cátedra. | |

| | |
|---|---|
| Ingresa campos requeridos e intenta crear el usuario. | Verifica que no exista ya un usuario asignado a la asignatura elegida. |
| | Devuelve mensaje de éxito confirmando la operación. |
| **Cursos alternos** | |
| **Asignatura ya asignada a otro usuario cátedra** | |
| | Se devuelve un mensaje de error indicando la causa. |
| Modifica la asignatura a la cual vinculará el usuario a crear. | |
| Continúa el proceso en el flujo normal de eventos. | |

| | |
|---|---|
| **Caso de uso** | #2.6 El usuario administrador desea modificar los datos de un usuario cátedra para actualizar su información |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El administrador busca la cátedra a la cual desea modificar sus atributos (básicos del usuario + específicos de la cátedra). | Devuelve listado de usuarios que cumplen con el patrón de búsqueda. |
| Selecciona cátedra a modificar. | Despliega detalles de los datos del usuario con campos editables. |
| Admin modifica aquellos campos que considere necesarios. Intenta guardar los cambios. | Verifica que el email no esté asignado ya a otro usuario. |
| | Devuelve mensaje de éxito al usuario, haciendo efectivos los cambios. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |

| | |
|---|---|
| Continúa el proceso en el flujo normal de eventos. | |
| **Email repetido** | |
| | Se informa al usuario que el email ya está registrado. |
| El usuario modifica los campos necesarios e intenta guardar nuevamente los cambios. | |
| Continúa el proceso en el flujo normal de eventos. | |

| | |
|---|---|
| **Caso de uso** | #2.7 El usuario administrador desea eliminar un usuario de tipo cátedra para quitar su acceso al sistema y evitar futuras referencias al mismo |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El administrador busca la cátedra que desea eliminar, a través del email o nombre. | Devuelve listado de los usuarios que coinciden con los parámetros de búsqueda. |
| Selecciona aquel usuario que desea eliminar. | Se elimina el usuario del sistema, desvinculando los archivos del mismo. |
| | Devuelve mensaje de éxito confirmando la operación. |
| **Cursos alternos** | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | Se devuelve un listado vacío. |
| Modifica los parámetros de búsqueda. | |
| Continúa el proceso en el flujo normal de eventos. | |

### 2.3. Cargas y transferencias de saldo

| **Requisito funcional** | **#3: Cargas y transferencias de saldo** |
|---|---|
| **Actores** | Empleado para la carga de saldo.<br>Estudiante y Becado para la transferencia de saldo. |
| **Descripción** | El sistema deberá soportar la existencia de un saldo virtual por el cual los estudiantes y becados registrados podrán abonar las transacciones realizadas dentro del sistema.<br>El crédito en la cuenta del estudiante/becado podrá provenir de una carga emitida por un empleado del CEI o bien por una transferencia de una cuenta de otro estudiante/becado registrado en el sistema. |
| **Problemáticas que soluciona** | 1.1: la eficiencia del proceso se verá beneficiada debido a que se digitaliza el cobro de los pedidos por parte de los usuarios sede.<br>2.1: dicha sistematización eliminará por completo la tarea repetitiva de cobrar un pedido por parte del usuario sede.<br>2.3: los usuarios sede únicamente manipularán dinero al momento de la carga de saldo a un estudiante/becado, lo que reduce considerablemente el intercambio monetario disminuyendo el riesgo de errores asociados.<br>3.1: la posibilidad de utilizar un saldo virtual para abonar los pedidos realizados por parte de los estudiantes/becados, agiliza y simplifica el proceso, evitando la necesidad de asistencia presencial así como la manipulación de dinero físico en el acto. |

| | |
|---|---|
| **Caso de uso** | #3.1 El estudiante/becado desea cargar saldo en su cuenta para poder abonar los pedidos realizados en el sistema |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El estudiante/becado se dirige físicamente a una de las sedes | |
| El estudiante/becado le solicita al empleado de la sede que desea cargar saldo en su cuenta y deberá brindarle sus datos personales (nombre, apellido y/o dni) | |
| El empleado accede al sistema como usuario de tipo sede y se dirige a la vista de Gestión de estudiantes | |
| El usuario sede busca al estudiante/becado filtrando por nombre, apellido y/o dni | El sistema devuelve el listado de estudiantes/becados que cumplen con los filtros establecidos |
| El usuario sede identifica el registro del estudiante/becado y presiona “Cargar saldo” | El sistema despliega un modal detallando los datos del estudiante/becado seleccionado y una entrada para ingresar el saldo a cargar |
| El empleado le solicita al estudiante/becado que le indique y abone la el monto deseado | |
| El estudiante/becado le indica el monto y le abona dicho monto al empleado con dinero físico | |
| El usuario sede ingresa el monto a cargar indicado por el estudiante/becado | El sistema verifica que el monto ingresado sea un número positivo. |
| El usuario sede presiona el botón “Cargar” | El sistema aumenta el saldo del usuario estudiante/sede según la el monto indicado |
| **Cursos alternos** | |
| **El monto ingresado no es un número positivo** | |
| | El sistema muestra un cartel de error indicando que el monto del saldo a cargar debe ser positivo |

| | |
|---|---|
| El usuario sede modifica el monto ingresado para que cumpla con los requisitos | |
| **Ningún usuario coincide con los parámetros de búsqueda** | |
| | El sistema devuelve un listado vacío de estudiantes/becados |
| El usuario sede corrige los filtros aplicados | |
| El usuario sede vuelve a realizar la búsqueda del estudiante/becado | |

| | |
|---|---|
| **Caso de uso** | #3.2 El estudiante/becado desea transferir saldo a la cuenta de otro estudiante/becado para que pueda abonar los pedidos realizadas en el sistema |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se dirige a la vista de Transferencia de saldo | El sistema devuelve un listado de estudiantes/becados registrados en el sistema |
| El estudiante/becado filtra el usuario por dni, nombre o email | El sistema devuelve el listado de estudiantes/becados que cumplen con los filtros establecidos |
| El estudiante/becado identifica el registro del usuario buscado y presiona “Transferir saldo” | El sistema despliega un modal detallando los datos del estudiante/becado seleccionado y una entrada para ingresar el saldo a transferir |
| El estudiante/becado ingresa el monto a transferir y presiona transferir | El sistema verifica que el monto ingresado sea positivo y que el usuario emisor posea saldo suficiente para realizar la transferencia |
| | El sistema descuenta el monto ingresado del saldo del emisor y le deposita dicha cantidad al usuario receptor |
| **Cursos alternos** | |
| **El monto ingresado no es un número positivo** | |
| | El sistema muestra un texto de error indicando |

| | |
|---|---|
| | que el monto del saldo a transferir debe ser positivo |
| El estudiante/becado modifica el monto ingresado para que cumpla con los requisitos | |
| **El usuario no posee suficiente saldo para realizar la transferencia** | |
| | El sistema muestra un modal de error indicando que el usuario no posee suficiente saldo para realizar la transferencia |
| El estudiante/becado modifica el monto ingresado tal que la transferencia respete el saldo disponible | |

### 2.4. Emisión y gestión de pedidos

| **Requisito funcional** | **#4: Emisión y gestión de pedidos** |
|---|---|
| **Actores** | Empleado para la gestión de pedidos.<br>Estudiante y Becado para la emisión de pedidos. |
| **Descripción** | El sistema permitirá la gestión absoluta del proceso asociado a la solicitud y gestión de los pedidos.<br>Por un lado, la plataforma disponibilizará todo el material dispuesto por cada una de las cátedras mediante un repositorio virtual para que los estudiantes/becados puedan visualizarlo de manera online o bien generar solicitudes de impresión a partir de la selección de uno o más archivos. En contraparte, los usuarios sede tendrán la posibilidad de visualizar en tiempo real la lista de pedidos activos, pudiendo gestionar y modificar el estado de cada uno de ellos. Además, los estudiantes/becados podrán realizar un trackeo del estado de sus pedidos. |
| **Problemáticas que soluciona** | 1.1, 1.2, 2.1, 2.2, 3.1, 3.2: la eficiencia del proceso se verá ampliamente beneficiada debido a que se sistematiza la generación de nuevos pedidos. Esto, junto con la posibilidad de trackear el estado de los pedidos de manera online, |

| | |
|---|---|
| | eliminará la necesidad de la presencialidad física por parte de los estudiantes/becados tanto para generar nuevos pedidos como para consultar su estado.<br>1.4, 2.3: el riesgo en la gestión de pedidos se ve disminuído ya que la configuración del pedido parte de la solicitud directa del estudiante, sin posibilidad de errores de interpretación o malos entendidos<br>3.3: los estudiantes conocen en tiempo real y de forma online el estado de un pedido |

| **Caso de uso** | #4.1 El estudiante/becado desea acceder al repositorio de archivos para visualizar de manera online el material proveído por las cátedras. |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se dirige a la vista de Nuevo pedido | El sistema devuelve un listado de las carreras registradas en la plataforma |
| El estudiante/becado selecciona la carrera deseada | El sistema devuelve un listado con todos los años de la carrera seleccionada |
| El estudiante/becado selecciona el año deseado | El sistema devuelve un listado con todos las materias de la carrera y año seleccionado |
| El estudiante/becado selecciona la materia deseada | El sistema devuelve todos los archivos de la carrera y año seleccionado, indicando el nombre y su tamaño |
| El estudiante/becado hace clic sobre uno de los archivos | El sistema descarga el archivo seleccionado por el usuario |
| El estudiante/becado visualiza el archivo descargado | |

| **Caso de uso** | #4.2 El estudiante/becado desea acceder al repositorio de archivos para crear una nueva solicitud de impresión |
|---|---|

| **Curso normal de eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se dirige a la vista de Nuevo pedido | El sistema devuelve un listado de las carreras registradas en la plataforma |
| El estudiante/becado selecciona la carrera deseada | El sistema devuelve un listado con todos los años de la carrera seleccionada |
| El estudiante/becado selecciona el año deseado | El sistema devuelve un listado con todos las materias de la carrera y año seleccionado |
| El estudiante/becado selecciona la materia deseada | El sistema devuelve todos los archivos de la carrera y año seleccionado, indicando el nombre y su tamaño |
| El estudiante/becado selecciona los archivos deseados y presiona Configuración de archivos. | El sistema muestra un listado con todos los archivos seleccionados. Por cada uno de ellos, asocia campos de configuración para determinar la cantidad de copias, las diapositivas por hoja, el rango de páginas a imprimir, la opción de doble o simple faz y la opción de impresión a color o blanco y negro. |
| El estudiante/becado ingresa la configuración deseada por cada archivo previamente seleccionado y presiona Configuración de anillados. | El sistema muestra un listado con todos los archivos seleccionados. Además muestra otra sección para crear nuevos grupos de anillados. Por cada grupo anillado posibilita asociar uno o más archivos, indicando tipo de anillado y orden de anillado. |
| El estudiante/becado selecciona opcionalmente aquellos archivos que desea anillar, indicando cantidad de anillados y el orden interno por cada uno de ellos y luego presiona Confirmar pedido. | El sistema muestra un resumen del pedido, indicando por cada archivo o anillado solicitado, su nombre, cantidad, páginas totales, precio unitario y precio total. Además, en caso de que el usuario sea de tipo becado, se muestra cuál es el descuento aplicado y la cantidad de copias utilizadas del total disponible de la beca (si es que hubiere). Por último se muestra un campo para determinar en cuál de las Sedes registradas se retirará el pedido |
| El estudiante/becado selecciona la sede donde lo retirará y finaliza su pedido. | El sistema guarda el pedido solicitado y descuenta el precio final del saldo del estudiante/becado.<br>En caso de ser becado, descuenta la cantidad de |

| | |
|---|---|
| | copias utilizadas en el pedido del total disponible con el que cuenta la beca. |
| **Cursos alternos** | |
| **El estudiante/becado utiliza el saldo negativo disponible para abonar el pedido** | |
| | El sistema verifica que el saldo adeudado no sobrepase la paramétrica que establece el máximo saldo negativo que puede tener un estudiante |
| | El sistema guarda el pedido solicitado y descuenta el precio final del saldo del estudiante/becado, indicando al usuario la utilización del margen de saldo negativo que brinda el CEI. |
| El usuario visualiza en pantalla su saldo negativo. | |
| **El estudiante/becado no posee saldo suficiente en su cuenta** | |
| | El sistema notifica al usuario que el saldo disponible no es suficiente para poder realizar el pedido |
| El estudiante/becado modifica la configuración del pedido o cancela el mismo. | |

| | |
|---|---|
| **Caso de uso** | #4.3 El estudiante/becado desea acceder a la información de los pedidos ya realizados para poder visualizar un resumen de sus solicitudes activas |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se dirige a la vista de Mis pedidos | El sistema devuelve, por defecto, un listado con los pedidos activos del usuario. Por cada pedido se mostrará la fecha de solicitud, la sede donde se retira, el estado, el subtotal, el descuento (si lo hubiere) y el precio final. |
| El estudiante/becado ordena el listado de sus | El sistema devuelve el listado de pedidos |

| | |
|---|---|
| pedidos por el campo Fecha de solicitud (de manera ascendente o descendente) | ordenados por Fecha de solicitud (de manera ascendente o descendente) |
| El estudiante/becado ordena el listado de sus pedidos por el campo Estado | El sistema devuelve el listado de pedidos ordenados y agrupados por el campo Estado |

| **Caso de uso** | #4.4 El estudiante/becado desea acceder a la información de los pedidos ya realizados para poder visualizar un resumen de sus solicitudes históricas |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se dirige a la vista de Mis pedidos | El sistema devuelve, por defecto, un listado con los pedidos activos del usuario. Por cada pedido se mostrará la fecha de solicitud, la sede donde se retira, el estado, el subtotal, el descuento (si lo hubiere) y el precio final. |
| El estudiante/becado selecciona la opción de Pedidos históricos | El sistema devuelve un listado con los pedidos históricos del usuario. Por cada pedido se mostrará la fecha de solicitud, la sede donde se retira, el estado, el subtotal, el descuento (si lo hubiere) y el precio final. |
| El estudiante/becado ordena el listado de sus pedidos por el campo Fecha de solicitud (de manera ascendente o descendente) | El sistema devuelve el listado de pedidos ordenados por Fecha de solicitud (de manera ascendente o descendente) |
| El estudiante/becado ordena el listado de sus pedidos por el campo Estado | El sistema devuelve el listado de pedidos ordenados y agrupados por el campo Estado |

| **Caso de uso** | #4.5 El estudiante/becado desea acceder a la información de un pedido finalizado para poder visualizar los detalles del mismo |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se encuentra en la vista de Mis pedidos históricos. | |

| | |
|---|---|
| El estudiante/becado selecciona el pedido deseado | El sistema devuelve los datos del pedido finalizado, incluyendo el número de pedido, la sede donde se retiró, el estado final del mismo, la fecha de solicitud, el descuento aplicado (si lo hubo), el precio total y un trackeo con todos los cambios de estado del pedido. |

| | |
|---|---|
| **Caso de uso** | #4.6 El estudiante/becado desea acceder a la información de un pedido activo para poder visualizar el resumen y el estado actual del mismo |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se encuentra en la vista de Mis pedidos activos. | |
| El estudiante/becado selecciona el pedido deseado | El sistema devuelve los datos del pedido activo, incluyendo el número de pedido, la sede donde se retirará, el estado actual del mismo, la fecha de solicitud, el descuento aplicado (si lo hubo), el precio total y un trackeo con todos los cambios de estado del pedido hasta el momento |

| | |
|---|---|
| **Caso de uso** | #4.7 El estudiante/becado desea acceder a la información de un pedido activo para poder visualizar el estado de los archivos solicitados |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se encuentra en la vista de un Pedido activo | |
| El estudiante/becado hace click sobre el botón Ver detalles | El sistema devuelve un listado de todos los archivos que fueron solicitados en el pedido |
| El estudiante/becado hace click sobre uno de los archivos | El sistema muestra la configuración del archivo seleccionado para dicho pedido, indicando la cantidad de copias, si es doble o simple faz, si es color o blanco y negro, el rango de páginas a |

| | |
|---|---|
| | imprimir y el estado actual de dicho archivo |

| **Caso de uso** | #4.8 El estudiante/becado desea acceder a la información de un pedido activo para poder cancelarlo |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo estudiante/becado se encuentra en la vista de un Pedido activo | El sistema muestra y habilita la opción de cancelar en caso de que el pedido se encuentre en estado Solicitado |
| El estudiante/becado hace click sobre el botón Cancelar | El sistema cancela el pedido, le quita el atributo de activo y lo clasifica como pedido histórico |

| **Caso de uso** | #4.9 El usuario de tipo sede desea acceder a la información de los pedidos realizados para poder visualizar un resumen en tiempo real de las solicitudes activas |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo sede accede a la vista de Pedidos en curso | El sistema devuelve un listado de pedidos activos registrados en la plataforma detallando el Id del pedido, su fecha de solicitud, el DNI y nombre completo del emisor, estado, subtotal (incluyendo el descuento si lo hubiere) y total. |
| El estudiante/becado realiza un pedido desde la página de Nuevo pedido o se modifica el estado de un pedido existente por cualquiera de las causas posibles | El sistema debe actualizar el listado de pedidos activos de la página Pedidos en curso asociada a la sesión del usuario tipo sede en tiempo real, sin interacción alguna por parte del usuario. |
| El usuario de tipo sede ingresa un valor en el campo de búsqueda, filtrando por Id de pedido, fecha de solicitud, estado, DNI o nombre del cliente | El sistema realiza una búsqueda acorde al valor ingresado por el usuario, filtrando aquellos pedidos que coincidan con uno o más campos de búsqueda. |

| **Caso de uso** | #4.10 El usuario de tipo sede desea acceder a la información de un pedido activo para poder visualizar los detalles del mismo |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo sede se encuentra en la vista de Pedidos en curso visualizando el pedido al que desea acceder | |
| El usuario de tipo sede hace click sobre el ícono de detalle en la fila del pedido deseado | El sistema despliega un modal con los detalles del pedido indicando el estado del mismo y un listado con todos los archivos que fueron solicitados |
| El usuario de tipo sede hace click sobre uno de los archivos | El sistema muestra la configuración del archivo seleccionado para dicho pedido, indicando si es doble o simple faz, si es color o blanco y negro, el rango de páginas a imprimir y el estado actual de dicho archivo.<br>Además muestra una sección destinada a gestionar la impresión del archivo |

| **Caso de uso** | #4.11 El usuario de tipo sede desea acceder a los detalles de un pedido activo para poder modificar el estado de uno de los archivos |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo sede se encuentra en la vista de Pedidos en curso con el modal abierto indicando los detalles de un pedido y la configuración de un archivo desplegada | El sistema muestra una sección para gestión de impresión brindando un campo para determinar el método de impresión a utilizar, un botón para realizar la impresión y un ícono para abrir el archivo con el previsualizador de pdf del navegador en una pestaña aparte |
| El usuario hace click sobre el campo de método de impresión | El sistema devuelve una serie de ítems, entre los que se encuentran Impresión manual y un ítem por cada impresora registrada |
| El usuario hace click sobre la opción Impresión manual | El botón de impresión cambia su texto principal a “Cambiar a impreso” |

| | |
|---|---|
| El usuario hace click sobre el botón Cambiar a impreso | El sistema modifica el estado del archivo a Impreso y también modifica el estado del pedido si el mismo se encontraba en Solicitado (pasando a En proceso) o si se encontraba en el estado En progreso y todos los archivos en Impreso (pasando a Para retirar) |
| **Cursos alternos** | |
| **El usuario de tipo sede abre el archivo en el previsualizador de PDF del navegador** | |
| El usuario de tipo sede presiona sobre el ícono Ver archivo | El sistema descarga el archivo y abre en una ventana aparte el previsualizar de PDF del navegador con el archivo cargado |
| **El usuario de tipo sede imprime un archivo por medio de una impresora registrada** | |
| El usuario de tipo sede hace click sobre el método de impresión | El sistema retorna el listado de impresoras disponibles en la red |
| El usuario de tipo sede selecciona una impresora de las opciones disponibles y hace click sobre el botón imprimir | El sistema se conecta con la impresora seleccionada y envía la solicitud de impresión |
| | El sistema retorna el estado “Imprimiendo” |
| El usuario de tipo sede visualiza que el archivo se encuentra imprimiendo | Cuando la impresora finaliza la impresión del archivo, el sistema cambia en tiempo real el estado del pedido a “Impreso” |
| El usuario de tipo sede visualiza el estado final del archivo como “Impreso” | |

### 2.5. Gestión de becados

| **Requisito funcional** | **#5: Gestión de becados** |
|---|---|
| **Actores** | Usuario de tipo sede para la gestión de becados. Estudiantes y Becados como usuarios finales. |
| **Descripción** | El sistema permitirá la administración y gestión absoluta de las becas otorgadas por el CEI. Deberá otorgar un módulo destinado a la promoción de los becados así como la baja de los mismos y la configuración de las becas. |
| **Problemáticas que soluciona** | 1.5: la sistematización de la otorgación de becas a estudiantes aumentará la eficiencia del proceso, debido a que los datos se encontrarán centralizados, serán de fácil acceso y tanto la promoción como la baja de becas se asociarán a estudiantes registrados previamente en el sistema, lo que perfeccionará el sistema de referenciado.<br>4.1: los estudiantes que sean promovidos a becados serán notificados a través de la plataforma. Además, podrán acceder en tiempo real a la cantidad de copias disponibles que posea su beca. |

| **Caso de uso** | #5.1 El usuario de tipo administrador desea visualizar un listado de becados para identificar aquellos estudiantes que tienen otorgada una beca |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo administrador accede a la página de gestión de usuarios. | El sistema muestra un listado de usuarios indicando Id, nombre completo, rol, si el correo fue verificado y si es un usuario habilitado en el sistema |
| El usuario de tipo administrador accede al menú de control de la tabla de usuarios y hace clic | El sistema devuelve un desplegable con las opciones “Todos los usuarios”, |

| | |
|---|---|
| sobre el icono “Opciones” | “Administradores”, “Estudiantes”, “Becados”, “Cátedras” y “Sedes”. |
| El usuario de tipo administrador selecciona la opción “Becados” | El sistema filtra el listado de usuarios por aquellos que únicamente poseen el rol de “Becado”. |
| El usuario de tipo administrador visualiza el listado de aquellos estudiantes que poseen una beca | |
| **Cursos alternos** | |
| **No existen usuarios becados registrados en el sistema** | |
| | El sistema muestra el listado vacío con el texto: “No se han encontrado resultados a partir del filtro de búsqueda establecido” |

| | |
|---|---|
| **Caso de uso** | #5.2 El usuario de tipo administrador desea visualizar un listado de estudiantes para promover a becado a uno de ellos |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo administrador accede a la página de gestión de usuarios. | El sistema muestra un listado de usuarios indicando Id, nombre completo, rol, si el correo fue verificado y si es un usuario habilitado en el sistema |
| El usuario de tipo administrador accede al menú de control de la tabla de usuarios y hace clic sobre el icono “Opciones” | El sistema devuelve un desplegable con las opciones “Todos los usuarios”, “Administradores”, “Estudiantes”, “Becados”, “Cátedras” y “Sedes”. |
| El usuario de tipo administrador selecciona la opción “Estudiantes” | El sistema filtra el listado de usuarios por aquellos que únicamente poseen el rol de “Estudiante”, indicando además de los datos anteriores, el DNI y si el usuario es o no es becado. |
| El usuario de tipo administrador visualiza el listado de aquellos estudiantes que poseen una beca y filtra a los estudiantes por nombre o DNI | El sistema devuelve un nuevo listado de estudiantes según el filtro establecido |

| | |
|---|---|
| El usuario de tipo administrador se posiciona sobre la fila del estudiante al que quiere promover y hace clic sobre el ícono “Otorgar beca” | El sistema lanza un modal de advertencia para confirmar si se le otorgará la beca al estudiante |
| El usuario de tipo administrador presiona “Confirmar” | El sistema guarda los cambios, otorgándole la beca al estudiante seleccionado |
| **Cursos alternos** | |
| **El usuario de tipo administrador cancela el otorgamiento de la beca** | |
| El usuario de tipo administrador presiona el botón “Cancelar” que se muestra en el modal de advertencia | El sistema cierra el modal de advertencia, cancela la acción de otorgamiento de beca y muestra la vista de gestión de usuarios en el mismo estado que se encontraba antes de lanzar el modal |

| | |
|---|---|
| **Caso de uso** | #5.3 El usuario de tipo administrador desea visualizar un listado de becados para degradar a estudiante a uno de ellos |
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo administrador accede a la página de gestión de usuarios y visualiza la lista de becados registrados en el sistema | |
| El usuario de tipo administrador filtra a los becados por nombre o DNI | El sistema devuelve un nuevo listado de estudiantes según el filtro establecido |
| El usuario de tipo administrador se posiciona sobre la fila del becado al que quiere degradar y hace clic sobre el ícono “Quitar beca” | El sistema lanza un modal de advertencia para confirmar si se le quitará la beca al estudiante becado |
| El usuario de tipo administrador presiona “Confirmar” | El sistema guarda los cambios, quitándole la beca al estudiante seleccionado |
| **Cursos alternos** | |
| **El usuario de tipo administrador cancela la quita de la beca** | |
| El usuario de tipo administrador presiona el botón “Cancelar” que se muestra en el modal de | El sistema cierra el modal de advertencia, cancela la acción de quita de beca y muestra la |

| | |
|---|---|
| advertencia | vista de gestión de usuarios en el mismo estado que se encontraba antes de lanzar el modal. |

| **Caso de uso** | #5.4 El usuario de tipo administrador desea gestionar las configuraciones de la beca para asignar las copias disponibles |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo administrador accede a la página de gestión de paramétricas del sistema | El sistema devuelve un listado de paramétricas gestionables, entre las que se encuentra el ítem “Copias inicialmente disponibles para el usuario becado” con el valor por defecto del sistema |
| El usuario de tipo administrador se posiciona sobre dicho ítem y presiona el ícono de editar | El sistema habilita el campo numérico asociado |
| El usuario de tipo administrador presiona sobre el campo numérico y modifica su valor | |
| El usuario de tipo administrador hace click sobre el botón guardar | El sistema guarda los cambios, asignando la cantidad de copias iniciales, ingresado por el administrador, a todas las becas que se otorguen en el futuro |

| **Caso de uso** | #5.5 El usuario de tipo becado desea visualizar los detalles de la beca para consultar la cantidad de copias disponibles |
|---|---|
| **Curso normal de eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| El usuario de tipo becado a la página “Home” de la plataforma | El sistema muestra una barra de información superior, y las secciones de “Accesos directos”, “Saldo en la cuenta” y “Pedidos en curso” |
| El usuario de tipo becado visualiza la sección Saldo en la cuenta | El sistema muestra el saldo que posee el estudiante en la cuenta, junto con la cantidad de copias de copias disponibles de la beca otorgada |
| **Cursos alternos** | |

| **Visualización de la cantidad de copias a partir de la barra superior** | |
|---|---|
| El usuario de tipo becado navega por cualquiera de las páginas disponibles en su sesión | El sistema muestra en la barra superior de información el saldo que posee el estudiante en la cuenta, junto con la cantidad de copias de copias disponibles de la beca otorgada |

# Proyecto Final

## Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería

# Prácticas aplicadas en el desarrollo

Mar del Plata, 21 de Diciembre de 2021

# ÍNDICE

# 1. Objetivo

El objetivo del presente documento es describir cómo fue la gestión del desarrollo de la plataforma. En el mismo se detallarán las prácticas y estándares utilizados por el equipo de trabajo.

# 2. Gestión de Código

### 2.1. Formato de commits

La gestión de cambios a nivel código fue controlada a través del uso de Git. Sin embargo, una necesidad que se volvió latente fue la documentación propia de dichos cambios, a fin de ir teniendo una noción de qué commits resolvían una tarea del proyecto en Jira. Para ello se hizo uso de la especificación conocida como **Conventional Commits**, la cual es una práctica cada vez más extendida en la industria del software.

La especificación es una ligera convención por sobre los mensajes de **commits**. Proporciona un conjunto sencillo de reglas para crear un historial de commits explícito, lo que facilita la escritura de herramientas automatizadas por encima de ellos. Esta convención encaja con **SemVer**, al describir las nuevas funcionalidades, correcciones y cambios no retrocompatibles realizados en los commits.

Un mensaje de commit debe estructurarse de la siguiente manera:

```
<tipo> [alcance opcional]: <descripción>
[cuerpo opcional]
[pie de página opcional]
```

**Figura 1**. Estructura de commit

Una vez aplicada la especificación es posible generar un **changelog** de forma automática cuando se desee lanzar una nueva versión del software, generando toda la documentación referida a los cambios sucedidos automáticamente.


**Figura 2**. Changelog

## 2.2. Modelo de ramas

Utilizar Git por sí sólo no garantiza contar con una buena organización sobre cómo desarrollar e integrar nuevos cambios sobre el código base existente. Para ello es necesario contar con un flujo de trabajo al que todos los desarrolladores puedan adherirse y permitir una evolución ordenada. **GitFlow** es un modelo de ramificación para Git, creado por Vincent Driessen. Ha atraído mucha atención porque se adapta muy bien a la colaboración y al escalado de los equipos de desarrollo. A continuación se muestra un diagrama asociado al modelo Gitflow, con pequeñas adaptaciones para el caso de uso del proyecto:

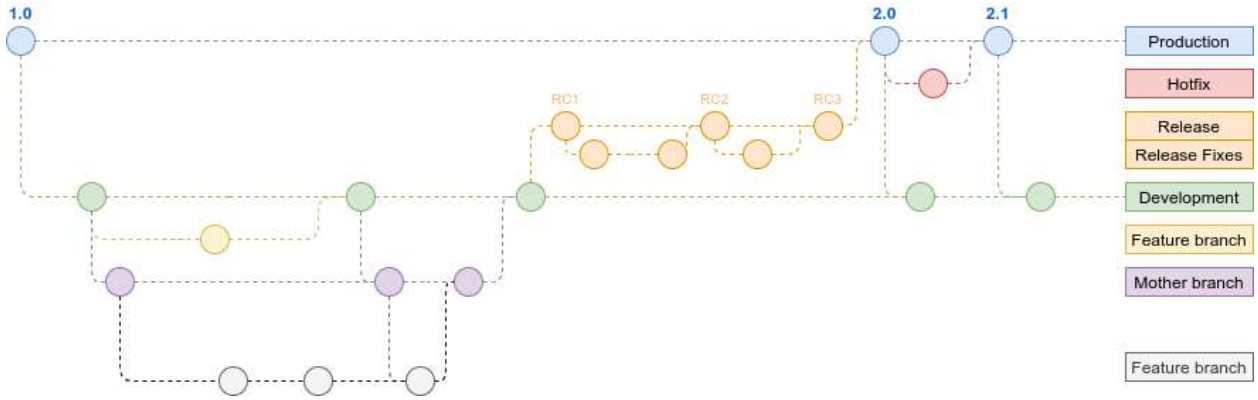



**Figura 3**. GitFlow

Los principales beneficios que trajo GitFlow son:

**Desarrollo paralelo**

Uno de los aspectos más destacables de GitFlow es que hace que el desarrollo paralelo sea muy fácil al aislar el nuevo desarrollo, del trabajo terminado. El nuevo desarrollo (como funciones y correcciones de errores que no son de emergencia) se realiza en las **ramas de funcionalidades** y solo se fusiona con el cuerpo principal del código cuando los desarrolladores están satisfechos de que el código esté listo para su lanzamiento.

**Colaboración**

Las ramas de funcionalidades también facilitan que dos o más desarrolladores colaboren en la misma función, debido a que cada rama de funciones es una **caja de arena** donde los únicos cambios son aquellos necesarios para que la nueva funcionalidad opere correctamente. Eso hace que sea muy fácil ver y seguir lo que está haciendo cada colaborador.

**Liberar el área de preparación**

A medida que se completa el nuevo desarrollo, se vuelve a fusionar en la **rama de desarrollo**, que es un área de preparación para todas las funcionalidades completadas que aún no se han lanzado. Entonces, cuando la próxima versión se bifurque de la rama de desarrollo contendrá automáticamente todos los nuevos cambios que se hayan terminado.

**Soporte para arreglos de emergencia**

GitFlow admite **ramas de revisión**, ramas creadas a partir de una **versión etiquetada**. Pueden utilizarse para realizar un cambio de emergencia, con la seguridad de saber que la revisión sólo contendrá la solución de emergencia. No hay riesgo de que se fusione accidentalmente en un nuevo desarrollo al mismo tiempo.

A través de la aplicación de GitFlow fue posible evitar varios errores y confusiones que se sucedieron en las primeras iteraciones del desarrollo del producto.

### 2.3. Pull Requests

Hasta ahora se presentó GitFlow como flujo a seguir para organizar los cambios en el código. Una funcionalidad brindada por GitHub (que no es nativa de Git) es la posibilidad de realizar **Pull Requests** sobre ramas sobre las cuales se desea fusionar una funcionalidad desarrollada. Los equipos hacen uso de esta herramienta para poder realizar **peer reviews** sobre el código que desean integrar a la rama destino. En este nuevo espacio se puede conversar sobre los cambios, proponer mejoras, actualizar pequeños errores detectados y finalmente aceptar los cambios.

Un requisito que se impuso sobre los Pull Requests es la necesidad de llenar una serie de secciones específicas a fin de proveer contexto sobre la naturaleza del cambio y ayudar al resto del equipo a entender qué se pretendía agregar al código base. La estructura base del mismo se puede observar en el siguiente diagrama:

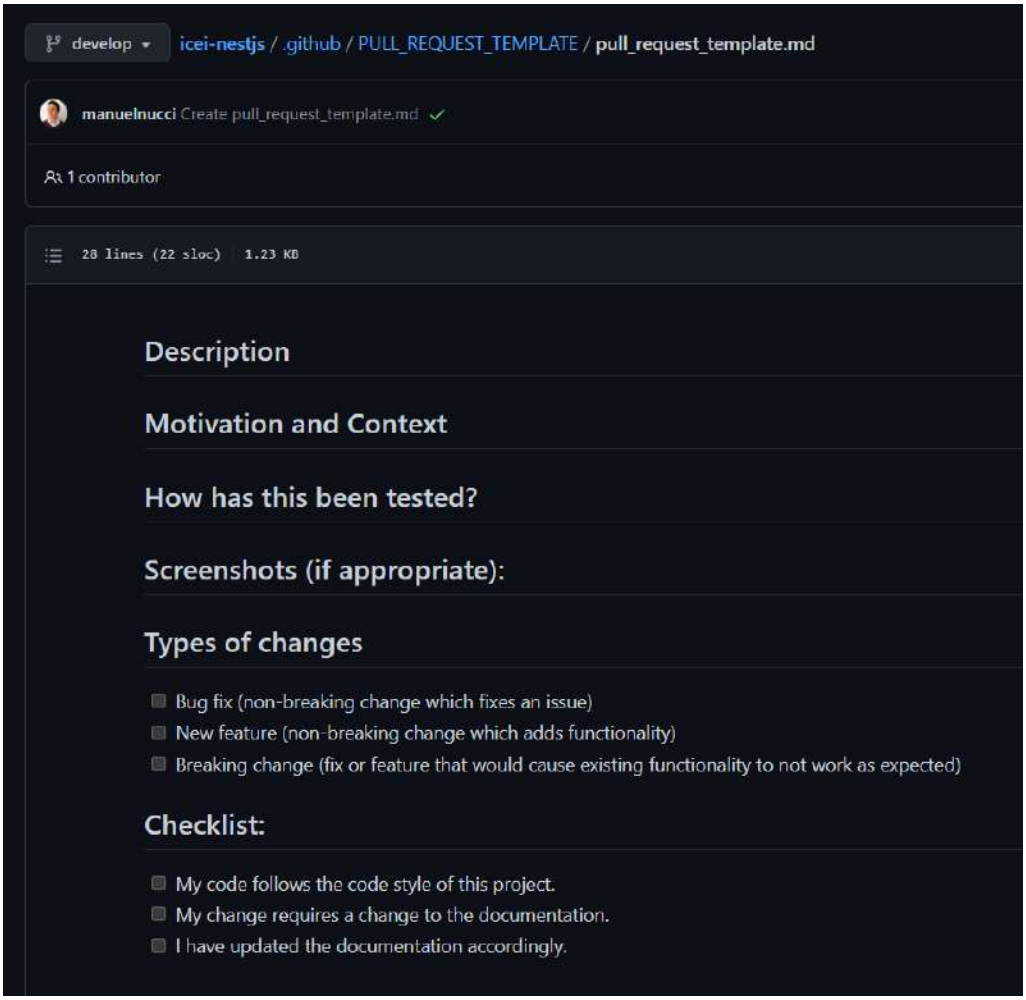



**Figura 4**. Pull Request template

De esta manera, todo cambio a integrar debe contener una breve descripción de la funcionalidad a integrar, la motivación y el contexto del cambio, si y cómo ha sido testeado, capturas de pantalla en caso de corresponder a un cambio en una aplicación con interfaz gráfica y algunos checklists extras. A partir de este requisito adicional el equipo de desarrollo pudo identificar rápidamente la naturaleza del cambio y proseguir con su integración a la rama destino.

## 3. Documentación de la API

La especificación OpenAPI, originalmente conocida como la especificación Swagger, es una especificación para archivos de interfaz legibles por máquina para describir, producir, consumir y visualizar servicios web RESTful.

La especificación de OpenAPI es independiente del idioma. Con la especificación de recursos declarativos de OpenAPI, los clientes pueden comprender y consumir servicios sin conocimiento de la implementación del servidor o acceso al código del servidor.

**Swagger** fue la herramienta que posibilitó implementar el estándar y exponer una interfaz gráfica en la cual consultar cada uno de los componentes de la API.

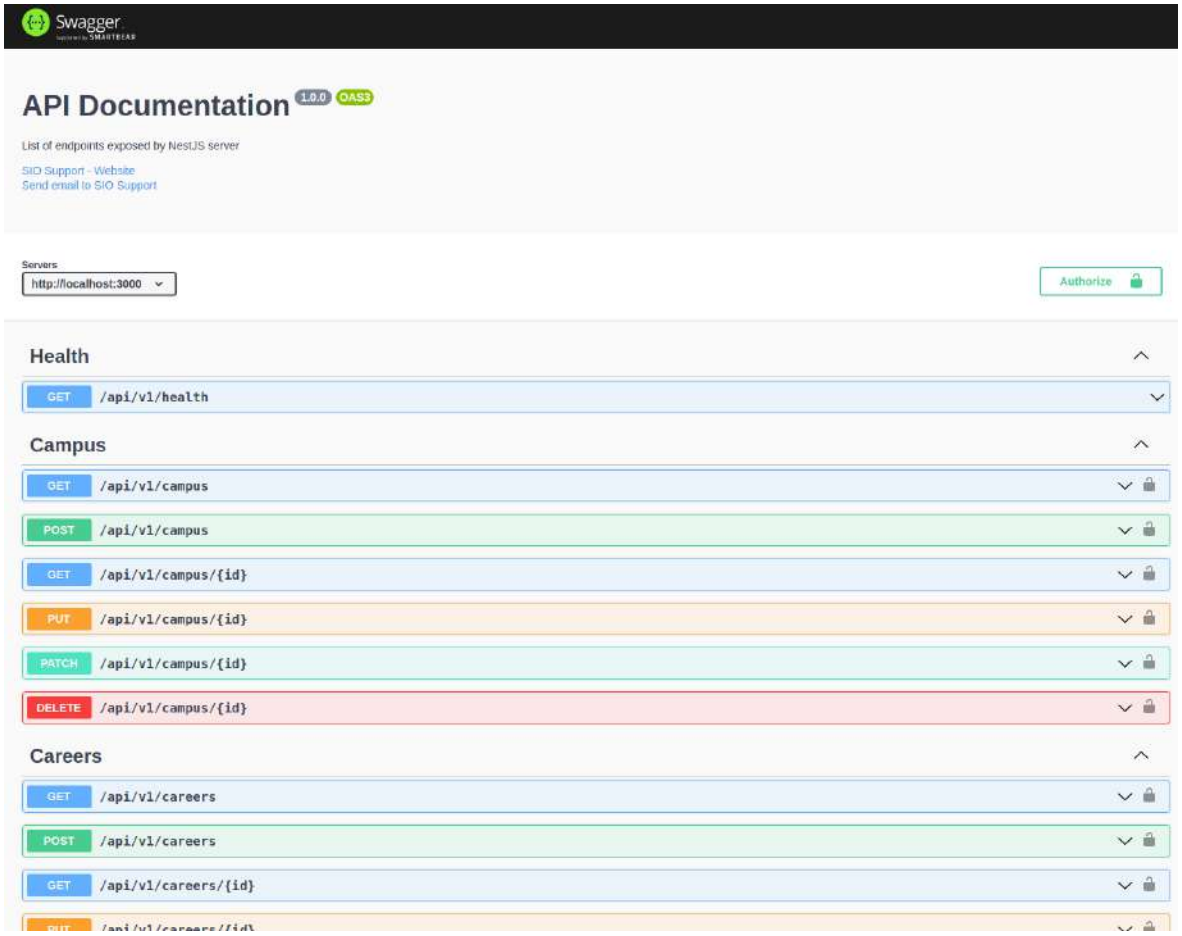



**Figura 5**. Portal de Swagger

Al acceder al portal expuesto por Swagger, los integrantes del equipo de desarrollo pudieron tener acceso al listado completo de endpoints que expuso la API. Para cada recurso, se contó con el acceso a sus endpoints implementados, los cuales se definen a través de su *verbo*, *path*, *query parameters*, *headers* y *body* necesarios para comunicarse con el mismo, entre otros.

También fue posible conocer cuáles esquemas de autorización eran soportados y proveer las credenciales necesarias en caso de querer acceder a un endpoint protegido.

En la siguiente imagen se aprecia en detalle la estructura de un endpoint implementado en el servidor:

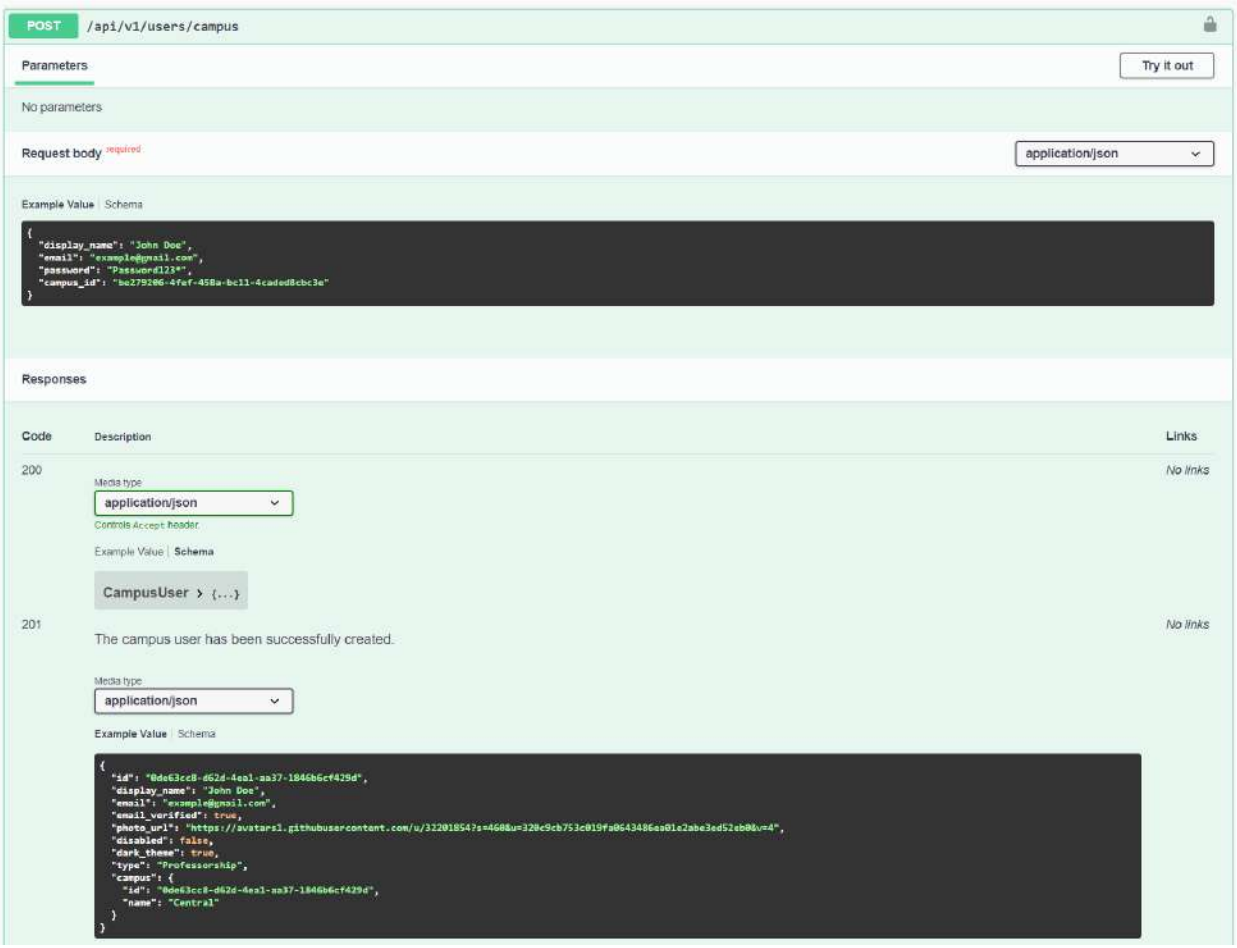



**Figura 6**. Creación de usuario para la sede

# 4. Integración Continua

Para facilitar la disponibilización del backend al momento de realizar las integraciones con el frontend se decidió empaquetar todo el software desarrollado en un **contenedor**. Un contenedor es una **unidad estandarizada** que incluye todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. **Docker** fue la empresa que popularizó esta estrategia de empaquetamiento y que hoy se establece como la alternativa de-facto para ejecución de aplicaciones.

Siguiendo esta técnica, cualquier miembro del equipo que deseara poder ejecutar el servidor para consultar o incluso verificar la integración desarrollada con el mismo, sólo debió descargarse la imagen con la versión adecuada y ejecutarla, junto con la base de datos.

Desde el punto de vista del desarrollador del backend, la creación y disponibilización de la **imagen** fue identificada como un **proceso repetitivo y tedioso**. Es por esto que se decidió tomar acción y para ello se programó su **automatización** de forma tal que ante la aprobación de un Pull Request a la rama *develop* se dispararía un **proceso de integración** con el fin de construir la imagen.

Para llevar a cabo esta tarea se utilizó la propia herramienta que brinda GitHub para tal fin: **GitHub Actions**. Este producto permite definir en un **archivo YAML** todos los pasos que deben ejecutarse cuando una condición es cumplida en el repositorio de código. Para el caso puntual en cuestión, como previamente se mencionó, un merge a la rama *develop* ejecutaría este set de tareas.

A continuación se puede observar la estructura del mismo:

```yaml
name: Build and publish Docker image

on:
  push: # every pull request is followed by a push
    branches:
      - develop

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@master

      - name: Read version from package.json
        uses: nyaayaya/package-version@v1

      - name: Decrypt all managed files
        uses: jrmcdonald/blackbox-github-actions@v0.2.0
        with:
          bb_actions_subcommand: "postdeploy"
        env:
          BLACKBOX_PUBKEY: ${{ secrets.BLACKBOX_PUBKEY }}
          BLACKBOX_PRIVKEY: ${{ secrets.BLACKBOX_PRIVKEY }}

      - name: Publish to Docker Registry
        uses: elgohr/Publish-Docker-Github-Action@master
        with:
          name: manuelnucci/nestjs-dev
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_ACCESS_TOKEN }}
          default_branch: develop
          dockerfile: Dockerfile.dev
          cache: true
          tags: "latest,${{ env.PACKAGE_VERSION }}"

      - name: Safely delete any decrypted files
        uses: jrmcdonald/blackbox-github-actions@v0.2.0
        with:
          bb_actions_subcommand: "shred_all_files"
        env:
          BLACKBOX_PUBKEY: ${{ secrets.BLACKBOX_PUBKEY }}
          BLACKBOX_PRIVKEY: ${{ secrets.BLACKBOX_PRIVKEY }}
```

**Figura 7**. Archivo de configuración de GitHub Actions

En la figura se puede observar un único job llamado ***build*** que consta de cinco pasos:

1. Descargar todo el código de la rama que disparó el job en el entorno aprovisionado para llevar a cabo el CI.
2. Leer la versión definida en el archivo *package.json* del repositorio.
3. Desencriptar el archivo que contiene la API Key para conectarse a un servicio externo.
4. Construir la imagen mediante el archivo *Dockerfile.dev* y publicarla en **Docker Registry**.

5. Eliminar todos los archivos desencriptados en el entorno de CI.

Esta simple automatización ayudó a los integrantes del equipo encargados del desarrollo del backend a acelerar sus tareas al no depender de este proceso repetitivo.

# 5. Bibliografía


- A successful Git branching model (5 de marzo de 2020). Recuperado el 13/02/2021 de https://nvie.com/posts/a-successful-git-branching-model/
- Introducing GitFlow (2021). Recuperado el 15/02/2021 de https://datasift.github.io/gitflow/IntroducingGitFlow.html
- Conventional Commits (2021). Recuperado el 20/03/2021 de https://www.conventionalcommits.org/en/v1.0.0/\
- OpenAPI Specification (2021). Recuperado el 28/12/2020 de https://swagger.io/specification/
- Designing REST API. What is contract-first? (21 de noviembre de 2019). Recuperado el 03/01/2021 de https://dzone.com/articles/designing-rest-api-what-is-contract-first
- GitHub Actions (2021). Recuperado el 04/04/2021 de https://docs.github.com/en/actions

# Proyecto Final

## Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería

# Diseño de componentes

Mar del Plata, 21 de Diciembre de 2021

# ÍNDICE

# 1. Objetivo

El objetivo del presente documento es describir de manera detallada el diseño interno de cada componente identificado en la arquitectura de solución de la plataforma, entre los que se encuentran: Cliente, Servidor y Base de datos. A su vez, también se describe cómo fue diseñada la API que comunica las capas de Cliente y Servidor.

# 2. Cliente

Como se detalló en la sección de Análisis técnico, Angular fue el framework utilizado para el desarrollo del frontend. Angular se autodefine como una plataforma y un marco para crear Single Page Applications utilizando HTML y TypeScript. Al ser un framework, a diferencia de React que se define como una librería, incluye todas las dependencias y utilidades para proveer una arquitectura intrínseca y robusta, que la propia documentación oficial menciona en detalle. Es por esto que, más allá de ciertas variaciones según el caso de uso y las necesidades propias de cada circunstancia, el desarrollador que utiliza este framework no debe tomar grandes decisiones en cuanto a patrones arquitectónicos e implementaciones particulares transversales a la aplicación.

## 2.1. Diseño de la Arquitectura

La arquitectura de una aplicación realizada en Angular se basa en ciertos conceptos y entidades fundamentales, entre las que se encuentran los Módulos, Componentes, Templates, Directivas y Servicios. La unidad básica de desarrollo en Angular son los Componentes, los cuales definen vistas, que son conjuntos de elementos visuales entre los que Angular puede elegir y modificar de acuerdo con la lógica y los datos del programa. Los componentes también utilizan servicios que proporcionan funciones específicas que no están directamente relacionadas con las vistas. Los proveedores de servicios pueden inyectarse en componentes como dependencias, lo que hace que su código sea modular, reutilizable y eficiente.

Los componentes a su vez están asociados y organizados en Módulos. Cada uno de los Módulos encapsula y recopila el código asociado a un conjunto funcional determinado. Además existe un único Módulo raíz que permite realizar el bootstrapping de la aplicación. De esta manera, el desarrollo de la aplicación queda definido a partir de la organización de un conjunto de Módulos.

A continuación se definen en detalle cada una de las entidades mencionadas anteriormente:

- **Módulos:** los módulos en Angular declaran un contexto de compilación para un conjunto de componentes que está asociado a un dominio de la aplicación, un flujo de trabajo o un conjunto de capacidades estrechamente relacionadas. Esto permite formar unidades funcionales y reutilizables, declarando componentes, servicios o directivas y exportando únicamente aquellas funcionalidades que se deseen. Al igual que los módulos nativos de

Javascript pueden importar funcionalidades que otros módulos exportan, formando así una jerarquía de módulos organizada y con un fin determinado.

La organización del código en módulos funcionales permite administrar el desarrollo de aplicaciones escalables y complejas y a diseñar en base a la reutilización.

- **Componentes:** cada componente define una clase que contiene data y lógica, y está vinculada a un template HTML y a un archivo de estilos SCSS.
- **Templates, directivas y data binding:** un template es una combinación de la sintaxis clásica de HTML con Angular markup, los cuales brindan tags y directivas personalizadas de Angular. Las directivas de un template proveen lógica de programación a la vez que proveen un mecanismo de sincronización y renderizado denominado data binding, el cual enlaza la lógica que contiene los datos de la aplicación con las vistas asociadas.

  Hay tres tipos de data-binding:

  - **Event binding**: responden a la interacción del usuario al modificar algún input en la aplicación, lo que actualiza automáticamente el valor del dato asociado en el modelo.
  - **Property binding**: permite agregar o modificar valores desde el modelo hacia la vista.
  - **Two-way data binding**: este tipo de enlace actúa en ambos sentidos entre la vista y el modelo. Es decir, si el usuario interactúa con la vista modificando un input, el valor automáticamente se refleja en el dato asociado del modelo. Y esto actúa de la misma manera, pero en sentido contrario: cuando un dato del modelo es modificado por una lógica interna de la aplicación, el DOM se sincroniza para mostrar dicho valor actualizado.
- **Servicios e inyección de dependencias:** toda la lógica transversal que no está asociada directamente a una vista y que requiere ser utilizada en diferentes partes de la aplicación y entre diferentes componentes es declarada e implementada en un servicio. Los servicios, a diferencia de los componentes, cuentan con el decorator **@Injectable** que provee una determinada metadata, permitiendo que sean inyectados en componentes como dependencias. La inyección de dependencias (DI) es un patrón de diseño en el que una clase solicita dependencias de fuentes externas en lugar de crearlas. El mecanismo de DI que implementa Angular permite aumentar la flexibilidad y modularidad de la aplicación.

Luego de haber identificado y mencionado el funcionamiento de las entidades básicas que provee la arquitectura de Angular, se muestra a continuación un diagrama que muestra la manera en que interactúan todas estas piezas:

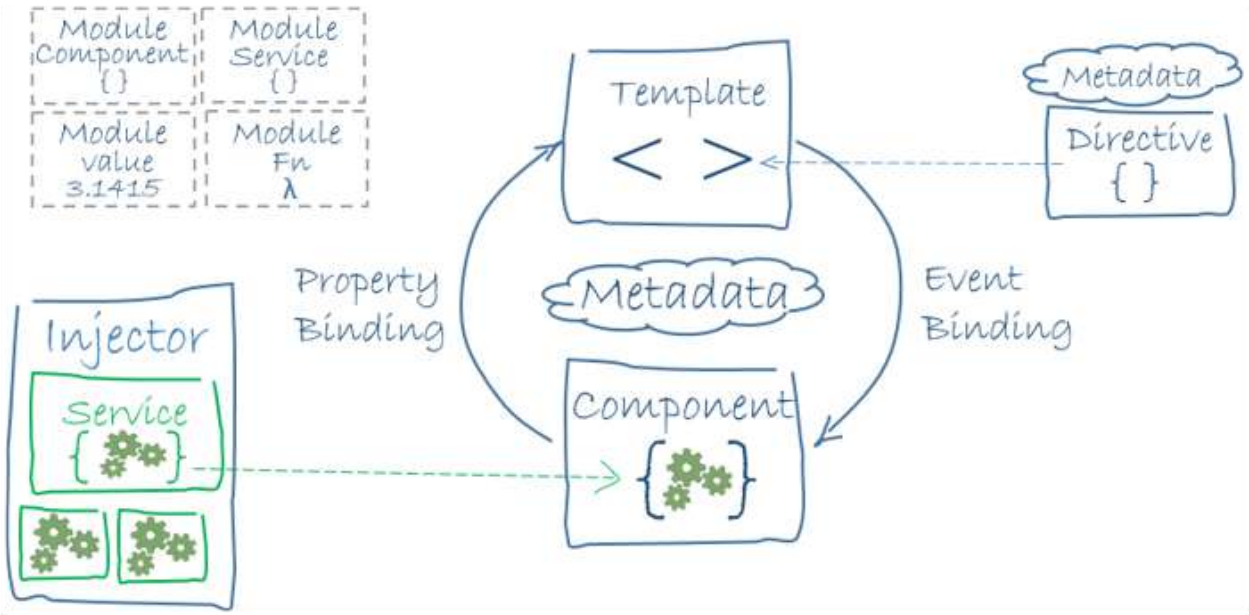



**Figura 1**. Interacción de componentes en Angular

Como se muestra en la figura anterior, una vista se forma a partir de un componente y un template. El decorador asociado a una clase de un componente agrega la metadata necesaria para la definición del mismo, incluyendo un puntero al template vinculado. Además, las directivas y el enlace de datos permiten modificar dicha vista en función de los datos y la lógica del programa. Por otro lado, el inyector de dependencias proporciona servicios que son transversales a la aplicación a un componente.

## 2.2. Estructura de archivos del proyecto

Teniendo en cuenta el contexto teórico y la arquitectura modularizada explicada en la sección anterior, se definió la siguiente estructura de directorios para el proyecto:


**Figura 2**. Estructura de directorios en Angular

Como se observa en el diagrama anterior, existe una carpeta ”modules” donde se encuentran los módulos funcionales principales del sistema. En ella se distinguen dos jerarquías principales: “logged” y “unlogged”. Esto permite separar los módulos asociados a las vistas y funcionalidades principales

del sistema del flujo de registración y login. Dentro del módulo principal “logged” se pueden diferenciar un módulo secundario por cada rol existente en el sistema.

Para fundamentar dicha decisión es necesario recordar la naturaleza de Single Page Application que poseen las aplicaciones de Angular. Este tipo de aplicaciones son conjuntos de páginas enrutadas en el propio navegador, liberando al servidor de una parte del trabajo al reducir la cantidad de llamadas y mejorando la percepción de velocidad del usuario. Esta característica es debido a que durante la primera llamada que realiza el navegador al servidor se carga en un único archivo todo el código HTML, Javascript y CSS de la aplicación. A simple vista, este funcionamiento parece ser una ventaja, ya que luego de la primera carga inicial se logra una experiencia fluida y dinámica, en la cual los ruteos son llevados a cabo directamente por el cliente, sin intervención del servidor. Sin embargo, en aplicaciones grandes, con numerosas vistas y módulos, esto puede resultar muy ineficiente, con un tiempo de carga inicial prolongado (incluso solicitando módulos y vistas que quizás el usuario no termine visitando).

Para solucionar el problema anterior Angular brinda un mecanismo de **Lazy Loading**, en el cual los módulos son solicitados a demanda, cuando se activa su ruta asociada. De esta manera se fundamenta la implementación de módulos granulares según los roles existentes en el sistema: cada uno de ellos serán cargados a demanda según el tipo de usuario que se encuentre logueado. Esta arquitectura junto con la carga diferida de los módulos ayuda a mantener el tamaño de los paquetes iniciales más pequeños, lo que a su vez ayuda a disminuir los tiempos de carga y la experiencia del usuario final.

# 3. Servidor

NestJS fue el framework elegido para desarrollar toda la implementación del lado del servidor. La arquitectura en la que se encuentra basado está inspirado fuertemente en Angular. De este modo, mucho de lo ya expuesto en otras secciones será tomado como base para los apartados siguientes del escrito.

A continuación se describirán las principales decisiones de diseño que se tuvieron que abordar a la hora de organizar e implementar funcionalidades, integrarse con sistemas externos o mismo con el almacenamiento de datos.

## 3.1. Organización de directorios y funcionalidades

La organización de las diferentes funcionalidades se llevó a cabo respetando los siguientes principios:

- Agrupar funcionalidades comunes en **módulos.**
- Cada módulo podrá declarar **controladores** si exponen sus servicios a través de uno o más endpoints.
- Cada módulo puede declarar **servicios** propios, los cuales contienen la lógica de negocio. Estos servicios pueden ser expuestos como interfaz del módulo para ser consumidos por otros módulos.
- Además, un módulo puede hacer uso de uno o más **repositorios** que abstraen las consultas a la base de datos a través de un ORM.

A nivel organización macro, los módulos implementados se pueden observar en las siguientes figuras:

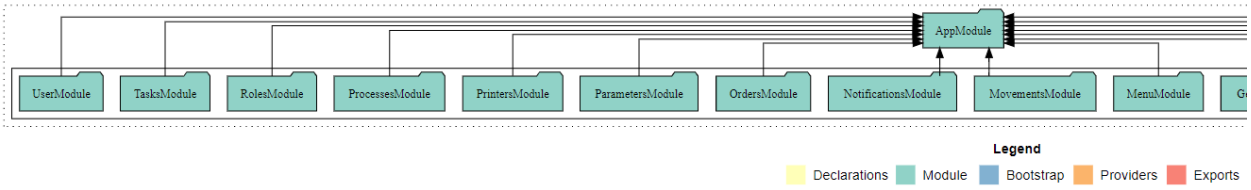



**Figura 3**. Organización de módulos - Parte 1

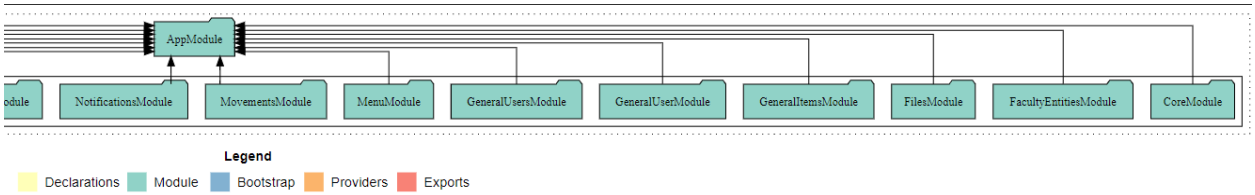



**Figura 4**. Organización de módulos - Parte 2

Como se puede apreciar, cada módulo tiene definidas claramente sus responsabilidades y esto se traduce a nivel código en un directorio único para cada uno de ellos. Toda comunicación entre módulos debe ser a través de las interfaces expuestas por el módulo al que se desea consultar. No están permitidas las importaciones específicas de servicios de otro módulo, esto violaría el principio de encapsulamiento que se desea obtener.

En su versión más simple, un módulo y sus componentes internos se pueden ver como lo siguiente:

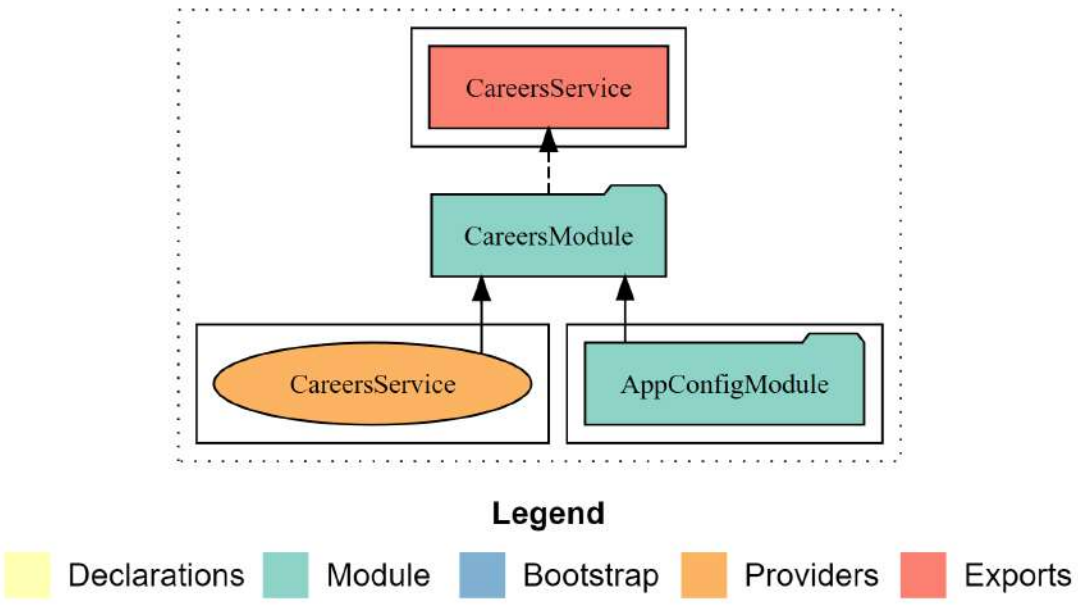



**Figura 5**. Módulo de carreras

En este caso podemos mencionar que:

- El módulo **CareersModule** hace uso de un segundo módulo **AppConfigModule**, el cual expone uno o más servicios en su definición
- Se ha implementado un servicio **CareersService** que, en este caso, contiene toda la lógica para el ABM de las carreras.
- El servicio anterior es expuesto para que otros módulos hagan uso de él. Esta es su interfaz hacia el mundo exterior.

### 3.2. Ciclo de vida de una solicitud

Otro punto importante que se atacó durante el diseño fue cómo sería el **ciclo de vida** de una **petición HTTP o WS**. NestJS provee varios componentes que permiten aplicar cierta lógica *antes* de que la petición llegue al **controller** que le corresponde. Estos son:

- Middlewares
- Interceptors
- Guards
- Pipes
- Filters

Si bien puede parecer desafiante contar con varios componentes que pareciera que hacen lo mismo, la verdad es que cada uno de ellos está diseñado para atacar un tipo de problemática específica durante el procesamiento de la petición. Si bien no ahondaremos en más detalle sobre su funcionamiento, lo importante es conocer cuál es su orden de evaluación, el cual se encuentra resumido en el siguiente gráfico:

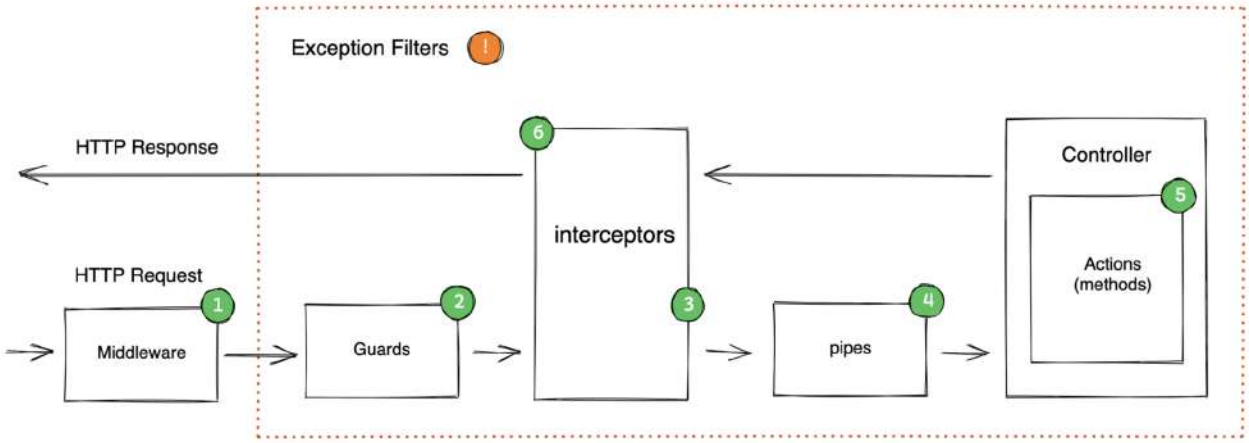



**Figura 6**. Request lifecycle en NestJS

En base a este modelo y las necesidades detectadas para el proyecto se diseñaron varios de estos componentes que simplifican enormemente la implementación, ya sea reuniendo lógica común en componentes únicos o permitiendo acceder a atributos específicos de la petición que permiten rutear, generar errores, validar inputs de entrada y otras tantas tareas.

A continuación se muestra el resultado al que se llegó:

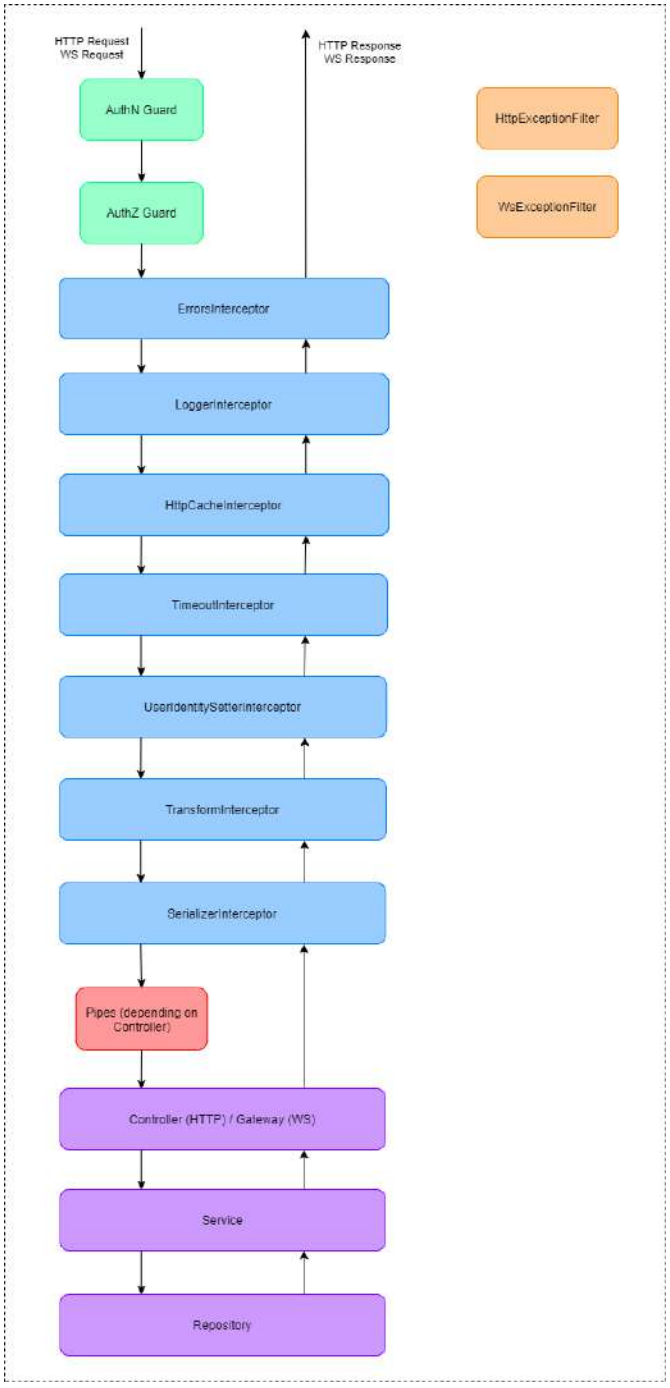



**Figura 7**. Ciclo de vida de una solicitud HTTP o WS

Se listan a continuación brevemente las responsabilidades de cada uno:

- **Guards**
    - **AuthNGuard**: guardia que se encarga de autenticar al usuario, a través de la decodificación del Bearer token que viene en la request.

    - **AuthZGuard**: guardia que se encarga de aplicar autorización sobre el recurso que se desea acceder (endpoint y método), para aquellos endpoints que hayan establecido que son protegidos.
- **Interceptors**
    - **ErrorsInterceptor**: se encarga de interceptar cualquier tipo de error que pueda originarse en las sucesivas capas inferiores, y lo convierte en un error manipulable por alguno de los filtros generales que se encargan de atrapar todas las excepciones.
    - **LoggerInterceptor**: interceptor que se encarga de loguear datos simples de la request, tal como endpoint consultado (path y verbo) y el timestamp de la consulta.
    - **HttpCacheInterceptor**: para aquellos endpoints cuyas respuestas pueden ser catcheadas, este middleware consulta si existe una caché disponible y en tal caso devuelve el contenido.
    - **TimeoutInterceptor**: se encarga de establecer un límite máximo de procesamiento de la request, a fin de estar siempre dentro de los términos de los posibles reverse-proxies anteriores al backend o incluso para garantizar una buena UX por parte del usuario.
    - **TransformInterceptor**: cuenta con dos responsabilidades. Por un lado, convierte todos los payload de las peticiones a objetos con atributos siguiendo la convención **CamelCase** (utilizada por JS/TS). Por otro lado, convierte todos los payloads de las respuestas HTTP o WS al formato **snake_case**, de forma que siempre el cliente reciba los atributos en esta convención.
    - **SerializerInterceptor**: se encarga de conservar y (por lo tanto esconder) aquellos atributos de la entidad que el usuario de X rol tiene acceso a visualizar.
- **Pipes**
    - **PaginationLimitPipe**: en caso que el límite de paginación exceda el límite configurado en el sistema, establece este límite como dicho valor ignorando el proporcionado por el usuario.
    - **EntityExistsPipe**: en caso de recibir en algún payload el id de alguna entidad que será utilizada posteriormente en la lógica de negocio, este pipe valida que dicha entidad exista en la base de datos, a fin de evitar tener que realizar esta validación posteriormente.
- **Filters**
    - **HttpExceptionFilter**: filtro general que atrapa cualquier excepción originada en cualquiera de los componentes anteriores, Controllers, Services o cualquier librería utilizada. Convierte la excepción en un error con un formato establecido de forma tal de contar con un contrato fijo con el cliente.
    - **WsExceptionFilter**: similar al filtro anterior pero se encarga específicamente de excepciones vinculadas a WebSockets.

### 3.3. Proveedor de autenticación (IDaaS)

Como bien se mencionó en el Análisis Técnico, Firebase fue la herramienta elegida para la autenticación de usuarios, gestión y reseteo de contraseñas, envío de mails, etc. Dado que estamos utilizando un subset de todas las herramientas que ofrece la plataforma, podemos verla como un BaaS (Backend as a Service) o más específicamente como un IDaaS (Identity as a Service), dado que se están utilizando las capacidades que ofrece para este tipo de problema en específico.

Si bien la plataforma resolvió la mayoría de las necesidades que el equipo encontró que debían ser resueltas por un producto externo (y no volver a reescribir la rueda), una en especial requirió una decisión de diseño particular. Firebase cuenta con la posibilidad de crear usuarios en su plataforma, pero con la restricción que no puede manipular atributos custom definidos por el cliente. Para nuestro caso en particular era crucial contar con propiedades ligadas al rol del usuario, las cuales debían ser almacenadas en algún repositorio.

Para resolver el problema se decidió almacenar estos atributos en la base local, y vincular a la tupla en cuestión con el usuario correspondiente en Firebase. Esto requirió encapsular la lógica de comunicación con la aplicación en un servicio y luego implementar otro servicio que se encargue de reunir la información de ambas fuentes, de forma tal de entregar una visión unificada (una entidad a fin de cuentas) que represente al usuario.

A nivel comunicación entre servicios del backend esto puede verse de la siguiente manera:

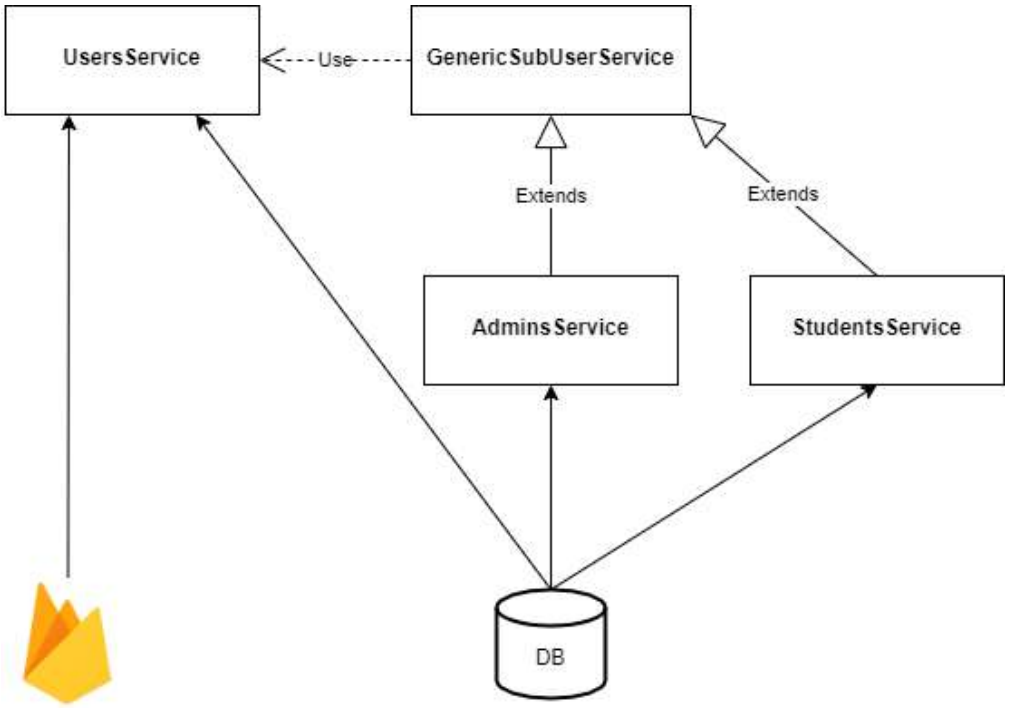



**Figura 8**. Comunicación entre servicios para manipulación de entidades de usuarios o derivadas

**UsersService** es el servicio encargado de proveer la visión unificada sobre un usuario, abstrayendo toda la lógica sobre dónde está almacenado y cómo recuperarlo. Para las demás sub-entidades del usuario (administradores, cátedras, estudiantes, etc.), se implementó una lógica común a todos los servicios específicos de cada sub-entidad. La misma se encuentra reunida en el componente **GenericSubUserService**, cuya responsabilidad es mergear la lectura del sub-usuario de la DB con el usuario proveniente del servicio **UsersService**.

A través de esta estrategia se logró encapsular lógica bien específica en un servicio en particular y con ello favorecer su reutilización a lo largo de toda la solución de backend. Si bien podemos verlo como un pequeño overhead del lado de la BD al realizar algunas consultas extras, la simplicidad de la solución supera con creces los pequeños problemas de performance que podrían aparecer cuando el volumen de datos o la concurrencia de usuarios crezca a niveles significativos.

### 3.4. Manejo de entidades con herencia

El diseño de las clases necesarias para manipular entidades que cuentan con herencia (Item y Binding, User y todas sus clases derivadas) fue resuelto a través de una abstracción que proporciona el ORM (Object Relational Mapping) que se integró a NestJS: **TypeORM**. Esta librería permite definir una herencia a nivel entidades que luego se traduce en una estructura particular a nivel tabla en la base de datos. Más de esto se hablará en la sección siguiente.

A nivel backend, el aspecto crucial a comentar es el hecho de que la recuperación de entidades heredadas a través de los métodos propios del ORM se vuelve completamente trivial, sin necesidad de realizar JOINS extras con la entidad padre a fin de recuperar sus atributos también.

```
async findAllSortedBySheetsLimit(manager: EntityManager) {
  return this.getBindingsRepository(manager).find({ order: { sheetsLimit: 'ASC' } });
}
```

**Figura 9**. Recuperación de grupos de anillados ordenados por límites de hojas

Como se mencionó, los anillados (Binding) son una subclase de una entidad más general: el Item. A través del uso del método ***find*** provisto por TypeORM toda la lógica necesaria para recuperar los atributos extras del Item que también son parte del Binding vienen en la entidad devuelta por el método, sin necesidad de añadir lógica extra a la consulta.

En las siguientes figuras se puede apreciar lo sencillo que es generar este tipo de comportamiento, simplemente agregando el decorador **@TableInheritance** en la clase padre y **@ChildEntity** en la clase hija.

```ts
@Entity('items')
@TableInheritance({ column: { type: 'enum', enum: ItemType, name: 'type' } })
export class Item extends BaseEntity {
  @Column()
  name!: string;

  @Column({ type: 'enum', enum: EItem, update: false, nullable: true })
  readonly code?: EItem;

  @Column({ type: 'decimal', precision: 8, scale: 2 })
  price!: number;

  @Column({ type: 'enum', enum: ItemType, update: false })
  readonly type!: ItemType;

  constructor(partial: Partial<Item>) {
    super(partial);
    Object.assign(this, partial);
  }
}
```

**Figura 10**. Definición de entidad Item

```ts
@ChildEntity()
export class Binding extends Item {
  @Column({ name: 'sheets_limit' })
  sheetsLimit: number;

  constructor(partial: Partial<Binding>) {
    super(partial);
    Object.assign(this, partial);
  }
}
```

**Figura 11**. Definición de entidad hija Binding

# 4. Base de datos

El diseño de la solución a nivel modelo de datos estuvo condicionado por el tipo de problemática que se atacó. El sistema de gestión de pedidos tiene una fuerte componente **transaccional** y es por esa razón que se optó por una base como PostgreSQL. Dada esta decisión fue necesario identificar todas las entidades que harían al sistema y traducirlas a una o más tablas en la base de datos.

## 4.1. Modelo Entidad Relación (MER)

Para poder llegar al **Modelo Relacional** en el cual se visualizan las tablas que serán creadas en la base de datos es preciso primero construir el **Modelo Entidad-Relación**, el cual es una abstracción o modelo de alguna situación de interés presente en el mundo real. El **MER** se realizó utilizando la técnica **Diagramas de Entidad Relación (DER)**. Para construirlo fue necesario modelar las cosas u objetos existentes, sus características y sus relaciones.

El MER se basa en los siguientes cuatro conceptos:

- Entidades
- Atributos
- Interrelaciones
- Reglas de de dominio adicionales

A continuación se encuentra el resultado del diseño realizado:

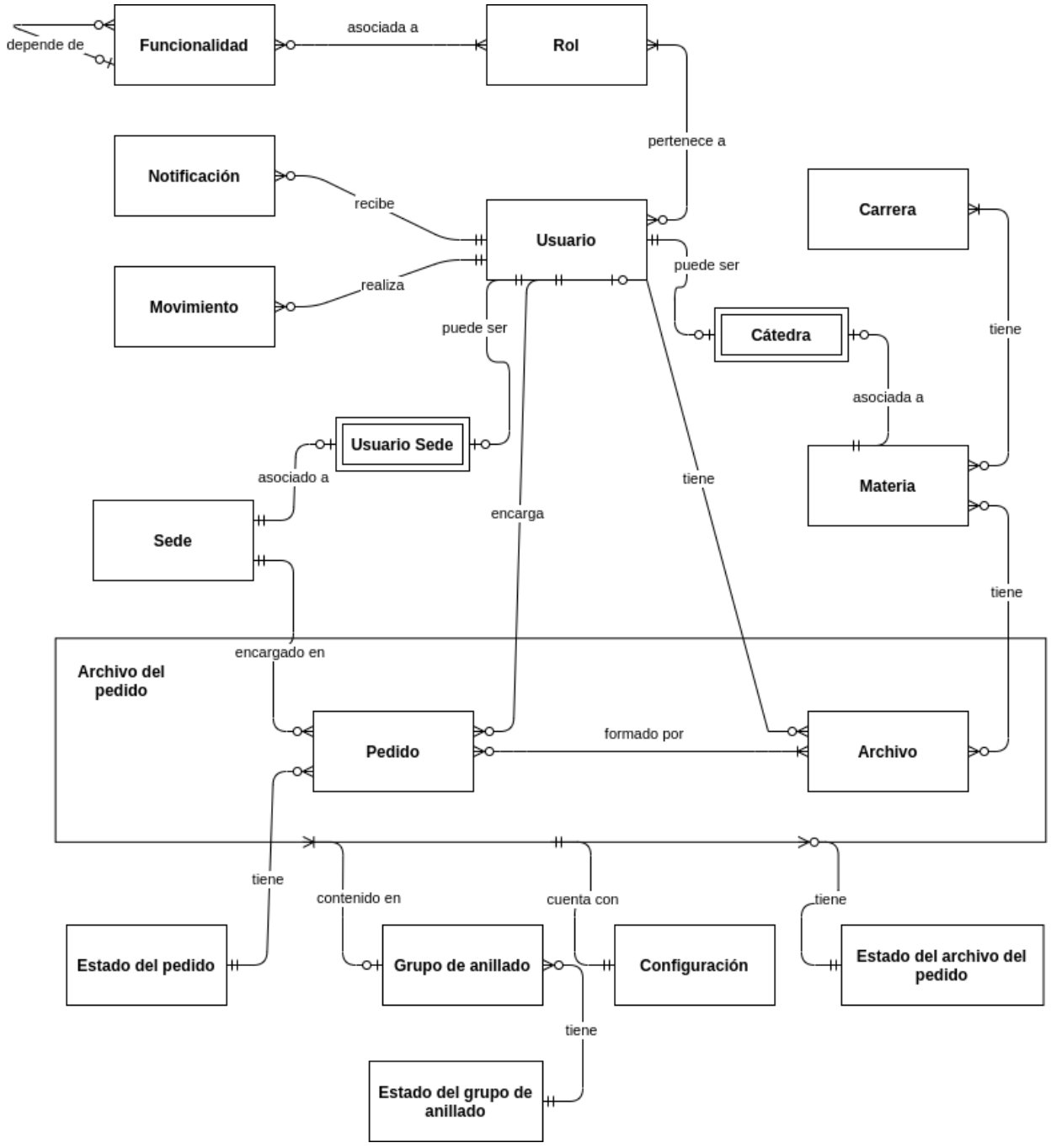



**Figura 12**. Modelo Entidad-Relación

A destacar:

- Existe una entidad **Usuario** que soporta entidades débiles asociadas (**Usuario Sede, Cátedra**). Estas sub-entidades están asociadas específicamente con **Sedes** o **Materias**, es decir que, cuando el Usuario es de un tipo particular la relación se establece, caso contrario no.
- Un **Usuario** cuenta con **Roles**, **Notificaciones**, **Movimientos**, **Pedidos** y **Archivos**, los cuales sirven de base para los principales casos de uso soportados por el sistema.
- Una **Funcionalidad** (que representa cada uno de los menús que el usuario puede acceder desde la interfaz web) puede depender de una **Funcionalidad padre**, y tener varias **Funcionalidades hijas** dependientes de ella.
- Las **Carreras** y **Materias** están relacionadas a través de una relación **M:N**, puesto que algunas asignaturas pertenecen a varias carreras.
- Una **Materia** puede contar con varios **Archivos** asociados a ella, los cuales no necesariamente son exclusivos de la misma (se permite que el mismo archivo sea asociado a otras materias a fin de maximizar el espacio de almacenamiento). De este modo, un **Archivo** puede pertenecer a una o más materias, o incluso a ninguna en caso de tratarse de un *archivo temporal*.
- Un **Pedido** está siempre asociado a una **Sede** en la cual se gestiona todo su ciclo de vida.
- Un **Pedido** está conformado por uno o más **Archivos**. La asociación entre ambos da origen a una nueva entidad llamada **Archivo del pedido**. Esto permite que la **Configuración** esté asociada a esta agregación y no a la entidad **Archivo** en sí misma.
- El **Archivo del pedido** cuenta además con un **Estado** que denota si el mismo ha sido impreso o no.
- El **Archivo del pedido** puede estar relacionado con un **Grupo de anillado** en caso que se haya optado por esta configuración adicional. El **Grupo de anillado** tiene su propio **Estado**.
- El **Pedido** cuenta con un **Estado** para hacer un seguimiento general del mismo, el cual dependiendo de la etapa que se encuentre en su ciclo de vida podrá estar supeditado a los estados de los múltiples **Archivos del pedido** y **Grupos de anillado** que lo conforman.

A partir del **MER** se pudo avanzar en la conformación de un **Modelo Relacional**, el cual representa una estructura mucho más similar a la versión final implementada en la base de datos. La **cardinalidad** y **participación** no son observables en el diagrama, pero las mismas son traducidas luego en restricciones a nivel base de datos.

Si bien la mayor parte de las tablas no presentaron una complejidad elevada respecto de su diseño y entendimiento se listan a continuación algunas decisiones tomadas que merecen ser nombradas dado que:

- Facilitan al consumidor de la base de datos la búsqueda de información.
- Se aplica alguna estrategia fuera de lo común, en base a necesidades específicas.

### 4.2. Decisiones de diseño

1. Añadir un atributo ***code*** en aquellas entidades de tipo **Enum** que son consumidas por el backend como tuplas únicas que luego son vinculadas con otra tabla.

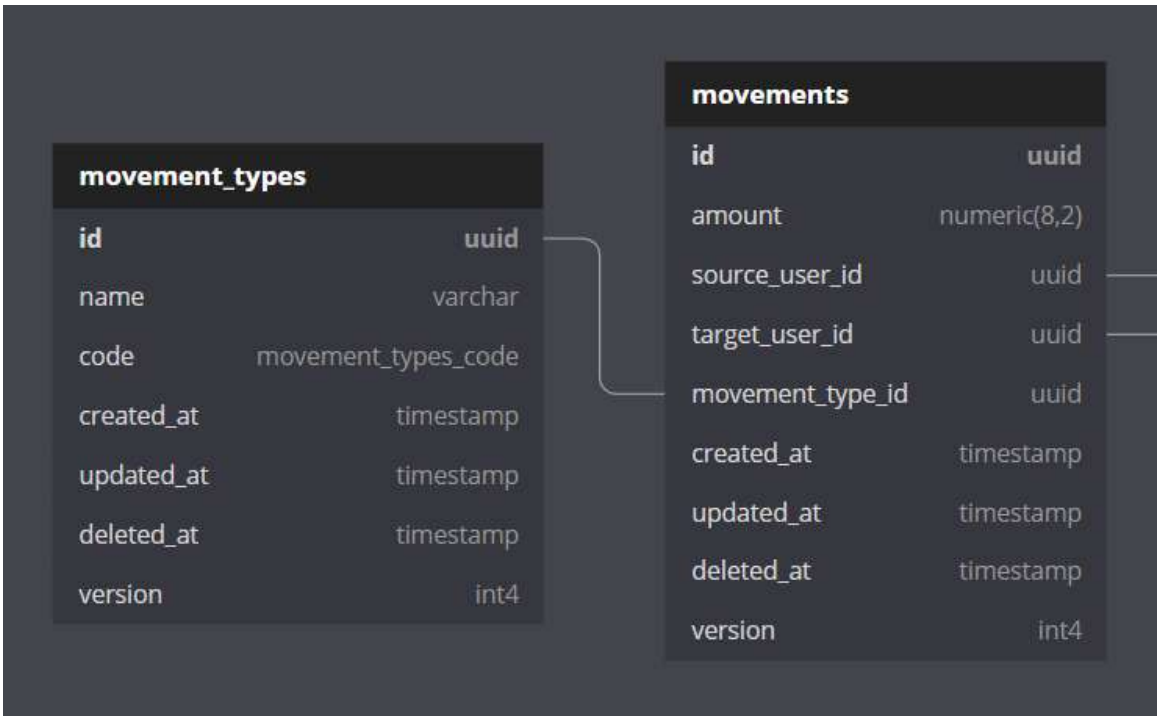



**Figura 13**. Diseño de tablas de movimientos y sus tipos

Al querer encontrar el **tipo de movimiento** que debe asignarse a un **movimiento**, el backend podría tener el **id** hardcodeado a nivel código. Sin embargo, ante un cambio eventual del id en la DB, un despliegue del código con la actualización sería requerida. Para evitar este impacto se decide designar un atributo específico que el cliente tiene seguridad de siempre encontrar en la base y con ello se evita acoplar la tupla a buscar con el código implementado.

2. Utilización de una **relación ternaria** para vincular carreras, materias y su entidad vinculante.

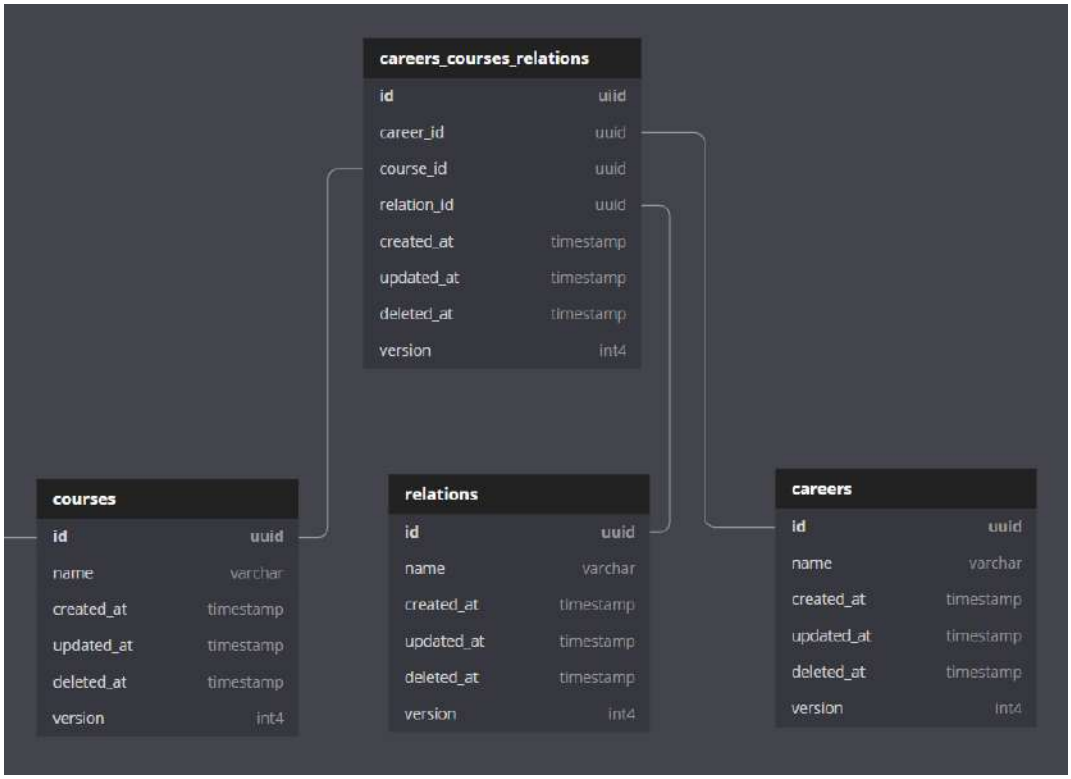



**Figura 14**. Diseño de tablas para gestionar relaciones entre carreras y materias

Dado que una **materia** puede ser compartida entre muchas **carreras** y ser cursada en diferentes años (o incluso ser optativa), la decisión tomada fue generar una entidad **Relation** que representa esta conexión. Luego, la vinculación entre carreras, materias y relaciones se realiza a través de una ternaria que siempre debe tener cada una de sus entidades relacionadas.

3. Utilización del modelo de herencia de datos **Closure Table** para gestionar un menú dinámico

Para soportar un menú dinámico que se genere en base a los roles asignados al usuario fue necesario conceptualizar una entidad genérica denominada **Functionality** que representa cada uno de los nodos de un menú organizado con estructura de árbol.

Se hace evidente que esta entidad tendrá nodos hijos y un padre, con lo cual la recursividad aparece en acción. A continuación se muestra cómo se ve en el **Modelo Relacional**:

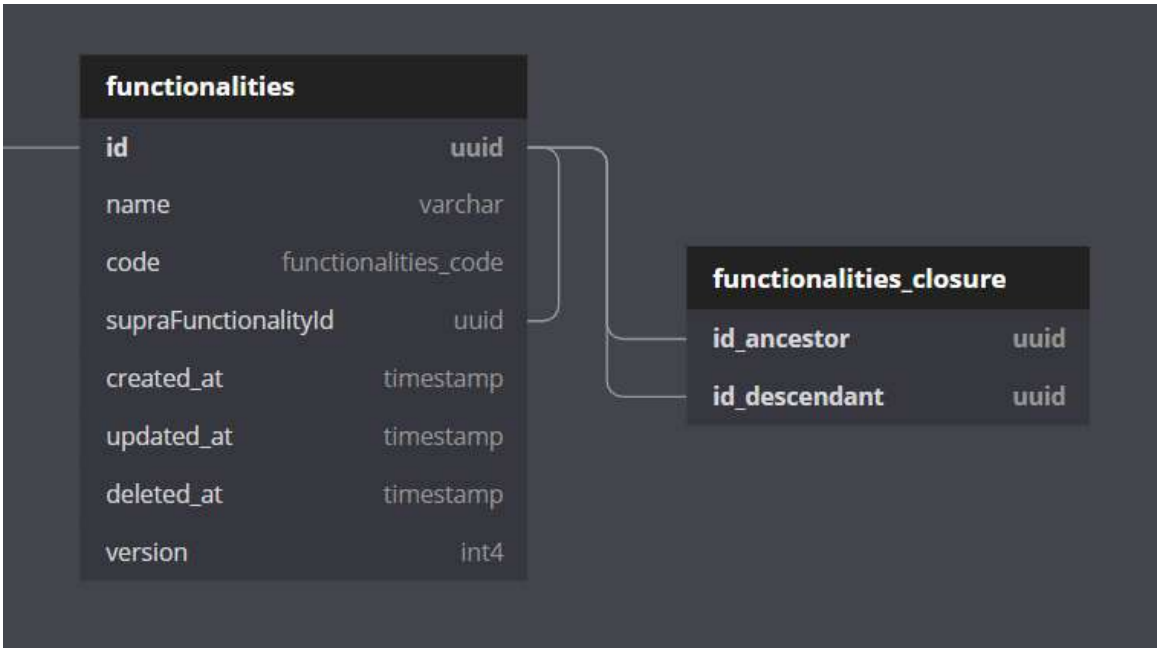



**Figura 15**. Diseño de la entidad Functionality para soportar menú dinámico

A través de la estrategia **Closure table** se necesita que cada nodo tenga la referencia a su padre y luego una tabla anexa auxiliar para manejar todos sus descendientes. Se optó por esta estrategia por sobre otras en base a su eficiencia, a pesar de requerir añadir una tabla extra al modelo.

| Design | Tables | Query Child | Query Subtree | Delete Node | Insert Node | Move Subtree | Referential Integrity |
|---|---|---|---|---|---|---|---|
| Adjacency List | 1 | Easy | Hard | Easy | Easy | Easy | Yes |
| Path Enumeration | 1 | Hard | Easy | Easy | Easy | Easy | No |
| Nested Sets | 1 | Hard | Easy | Hard | Hard | Hard | No |
| Closure Table | 2 | Easy | Easy | Easy | Easy | Easy | Yes |

**Figura 16**. Cuadro comparativo de estrategias para manejar entidades de tipo árbol en bases de datos

En la bibliografía se puede acceder a la presentación de Bill Karwin en la cual se detalla el funcionamiento de cada algoritmo.

4. Única tabla para representación de todos los tipos de usuarios del sistema

Como bien se mencionó en la sección de Diseño Técnico del backend, se aprovechó la capacidad del ORM utilizado para manipular entidades heredadas. A nivel base de datos este mapeo se ve de la siguiente manera:

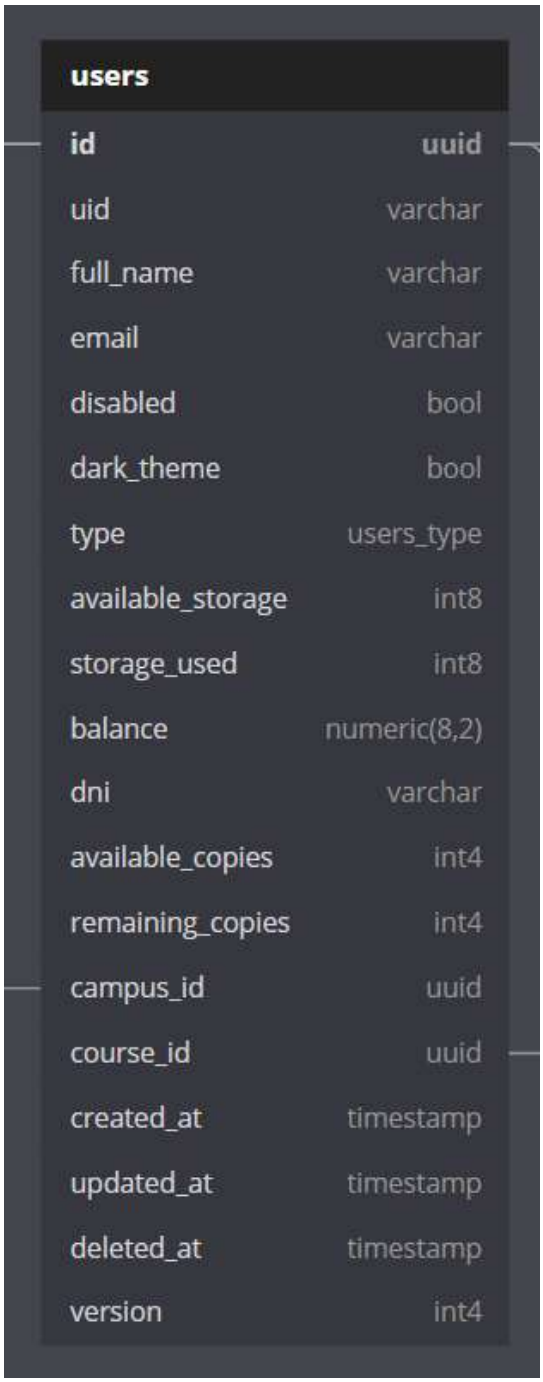



**Figura 17**. Mapeo de entidades a tabla de usuarios

Los pocos atributos de cada tabla especializada son factibles de ser nulos y el ORM traduce la lectura de la tupla a la entidad que es consultada.

Si bien la problemática puede ser resuelta con una tabla y varias hijas (siguiendo una estrategia de entidades fuertes y débiles), la complejidad en este caso queda delegada al ORM que debe tomar sólo los atributos necesarios en función del atributo **type**. Si se optase por la otra estrategia mencionada, la lectura de cada entidad hija debiera hacerse teniendo en cuenta el o los JOINS extra a realizar para traer los atributos de las entidades fuertes de las que dependen.

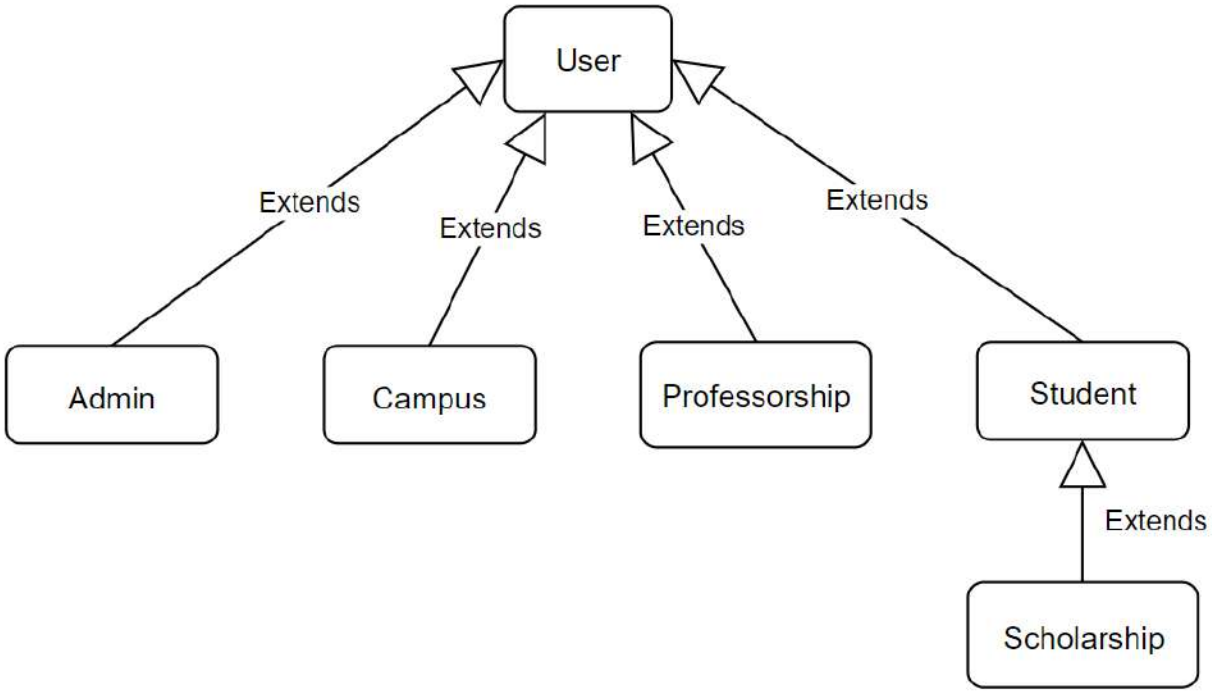



**Figura 18**. Herencia de entidades de tipo Usuario diseñada

# 5. APIs

La mayor parte de interacción entre componentes se realiza entre la aplicación web cliente y el servicio web. Para esta comunicación se utilizaron dos tecnologías: **API RESTful** soportado por el protocolo **HTTP** y **WebSockets** con su protocolo pertinente, **WS**. En cuanto al intercambio de datos con las terminales de impresión se optó, tal cual se mencionó en el apartado de análisis, por **IPP.**

## 5.1. API RESTful

La API fue diseñada según las buenas prácticas de la arquitectura REST. Algunos de los puntos tomados en cuenta fueron:

- **Utilizar sustantivos para referenciar a los recursos**: al dividir la API en base a recursos, se siguió una estructura jerárquica para hacer referencia a las entidades del sistema, nombrándolas con sustantivos para un mejor entendimiento e intuitividad. Ejemplo de algunos endpoints son:
    - api/v1/files
    - api/v1/users
    - api/v1/users/students

- **Utilizar sub-recursos**: cuando fue necesario trabajar relaciones entre recursos se organizó la API de forma jerárquica. Por ejemplo, si se necesitara obtener las materias de una cierta carrera con el identificador {careerID} el endpoint en cuestión se formaria de la siguiente forma:
    - api/v1/careers/{careerID}/courses

- **Realizar un buen uso de los verbos HTTP**: para la definición de endpoints y métodos se relacionó los verbos HTTP con las operaciones CRUD (Create, Replace, Update, Delete) para obtener una correlación entre ellos.
    - POST: utilizado para crear nuevos recursos en una colección determinada.
    - GET: obtener el/los recursos.
    - PATCH: actualizar/modificar el recurso en cuestión.
    - PUT: reemplazar por completo el recurso solicitado.
    - DELETE: borrar el recurso referenciado.

- **Proveer filtros, orden y paginado**:

  Para brindar a la API de tales funciones el equipo desarrolló una serie de técnicas a través de **query parameters (qp)**. Para explicar la lógica detrás de estas partimos del siguiente ejemplo:

  **Endpoint:**
  **api/v1/files?filter={"AND":[{"file_course.course_id":{"is":"0605a097-7702-40ae-8d0b-2cef5cf9e53a"}}]}&page=2&limit=3&sort=file.name ASC**

- **Filtrado**: en el caso de los filtros se implementó el qp **filter**, la función que cumple es acotar los datos a obtener siempre y cuando se cumplan las condiciones que yacen en este campo. Para el ejemplo en cuestión, se obtendrán solo los archivos que cumplan la condición que la materia o **course** asociado a ellos tenga el id **0605a097-7702-40ae-8d0b-2cef5cf9e53a**, esta es la forma implementada para disponer de los archivos de una materia específica.

  Esta implementación es muy versátil y permite filtrar los datos a solicitar según el recurso y su relación con otros. Como se puede observar está fuertemente basada en la sintaxis de consultas SQL, y se conforma de la siguiente manera:

  - Operador lógico:
    - Posibles valores: AND | OR
    - Función: evalúan dos condiciones a partir de un campo del recurso en cuestión y según el valor deben cumplirse ambas o al menos una de ellas.
  - Operadores:
    - Posibles valores: 'is' | 'not' | 'in' | 'not_in' | 'lt' | 'lte' | 'gt' | 'gte' | 'contains' | ‘not_contains' | 'starts_with' | 'not_starts_with' | 'ends_with' | 'not_ends_with'
    - Funciones: determina la operación que se hará sobre el campo que se especifique y el valor o valores contra los que se quiere comparar.

- **Ordenado**: para ordenar los datos provenientes del servidor el parámetro sort fue utilizado. Este parámetro tiene la utilidad de devolver los objetos en un orden ascendente o descendente según el campo que se especifique. En el caso de ejemplo, se recuperan en orden ascendente por el nombre de archivo o file. Se compone especificando en el qp sort el campo por el cual se realizará el ordenado y si es de manera ascendente o descendente a través de los valores ASC o DESC, respectivamente.

- **Paginado**: a la hora de limitar el número de objetos devueltos del servidor y así conseguir mejores tiempos de respuesta se hizo uso de una serie de parámetros para paginar. Estos fueron page y limit. Para explicar su funcionamiento partiremos del ejemplo propuesto donde los parámetros involucrados cobran los siguientes valores: page=2, limit=3. Dicha solicitud retorna la siguiente respuesta:

```
{
    "data": {
        "items": [
            {
                "id": "a167c8fa-baa9-4489-a8cc-d8b8b98b5c74",
                "name": "Ingeniería Industrial"
            },
            {
                "id": "8b9f1d36-55d4-479b-a615-24ee79aa0037",
                "name": "Ingeniería en Materiales"
            },
            {
                "id": "69c48521-219e-47a1-a801-6225d4c577b2",
                "name": "Ingeniería en Computación"
            }
        ],
        "meta": {
            "total_items": 10,
            "item_count": 3,
            "items_per_page": 3,
            "total_pages": 4,
            "current_page": 2
        },
        "links": {
            "first": "http://localhost:3000/api/v1/careers?limit=3",
            "previous": "http://localhost:3000/api/v1/careers?page=1&limit=3",
            "next": "http://localhost:3000/api/v1/careers?page=3&limit=3",
            "last": "http://localhost:3000/api/v1/careers?page=4&limit=3"
        }
    }
}
```


**Figura 19**. Ejemplo de formato de respuesta del servidor

Como podemos ver el objeto de respuesta contiene tres partes principales:

- **data**: en donde se encuentran los datos solicitados. Al limitar la solicitud con el valor limit=3 solo retornará tres objetos.
- **meta**: metadata asociada a la cantidad de elementos que se retornaron, el total de objetos que cumplen con los parámetros establecidos, la cantidad que contiene la página, el total de páginas, y la página actual. Dichos campos son útiles para la capa de presentación, de forma tal de conformar una interfaz que contenga la información necesaria para que el usuario tenga un óptimo uso de esta funcionalidad de paginado.
- **links**: ver siguiente apartado, "***Uso de HATEOAS***".

- **Uso de HATEOAS**: Hypermedia as the Engine of Application State (HATEOAS) se utilizó para proveer al cliente el acceso a otros recursos de manera dinámica a través de hipervínculos. En el paginado de recursos se denota un claro uso de esta técnica. En el objeto links retornado en la respuesta del servidor puede verse varios hipervínculos relacionados al paginado, estos son: la primera página, la anterior y siguiente, y la última página. Todos estos vínculos contienen la URL completa, dominio, endpoint y query parameters especificados. Dicha implementación facilita el uso en el frontend, en donde ya está provista la URL a la que debe hacer la solicitud según la acción que quiere tomar.

- **Versionar API**: con el objetivo de mantener la retrocompatibilidad con los clientes de la API, en el caso de realizar cambios disruptivos en base a recursos se optó por versionar la API con un indicador como "v1", por lo que todos los endpoints parten sobre este:
    - api/v1/

  De tal manera el contrato establecido se inmortaliza, y en el caso de encontrarse en la necesidad de realizar cambios no retrocompatibles que alteren dicho contrato, se deberán desarrollar en otra versión, por ejemplo ***api/v2***.

- **Validación de parámetros de entrada**: al momento de recibir los objetos provenientes del cliente se realizó una validación de los parámetros previamente establecidos por el equipo. Dicha validación radica en verificar por un lado si los tipos de los atributos coinciden con los esperados y por el otro si se encuentran acotados a un conjunto de valores específicos. De esta forma se redujo la posibilidad de errores que se pueden haber producido luego de procesar estos valores. En caso de encontrarse errores, se le informa al cliente de los fallos en la solicitud realizada.

- **Manejar errores en base a códigos de respuesta del servidor**: con el fin de manejar acciones predeterminadas ante ciertas respuestas y eventos del servidor se operó del lado del servicio web con el envío de códigos de estados y respuestas estandarizadas para ciertos errores habituales. Todos los errores provenientes del servidor son retornados con el siguiente formato de respuesta, además de una cabecera de respuesta que indica el código de estado:

```
{
    "status_code": number,
    "name": string,
    "message": string,
    "timestamp": Date,
    "path": string
}
```

**Figura 20**. Formato de respuesta para un error

De esta manera, en la capa de presentación se pudo proveer al usuario de mayor información y contexto en base al error. Este objeto de respuesta tiene varios atributos, en el orden de presentación estos son: el código de estado, el nombre del error asociado el código de estado, un mensaje con la causa del error, fecha y hora el que se ocasionó, y el endpoint al cual se solicitó en un principio la respuesta. A su vez, a través de los códigos de estado se pudo tomar acciones y flujos predeterminados, como cerrar la sesión del usuario si obtiene un error de no autenticado.

## 5.2. Web Sockets

El servicio web soporta comunicación por el protocolo WebSockets, el mismo fue de vital utilidad para realizar una interacción en tiempo real entre cliente y servidor, de cara a la gestión de pedidos como se explicita en el apartado de análisis correspondiente a este tema.

Para la implementación del lado del servidor se optó por utilizar la integración nativa de NestJS con **Socket.IO**. Dado que esta librería es una implementación *personalizada* por sobre el protocolo WS se utilizó la versión cliente de la misma para el frontend.

Para aplicar el protocolo y librería se siguió el esquema que se encuentra en la siguiente imagen:

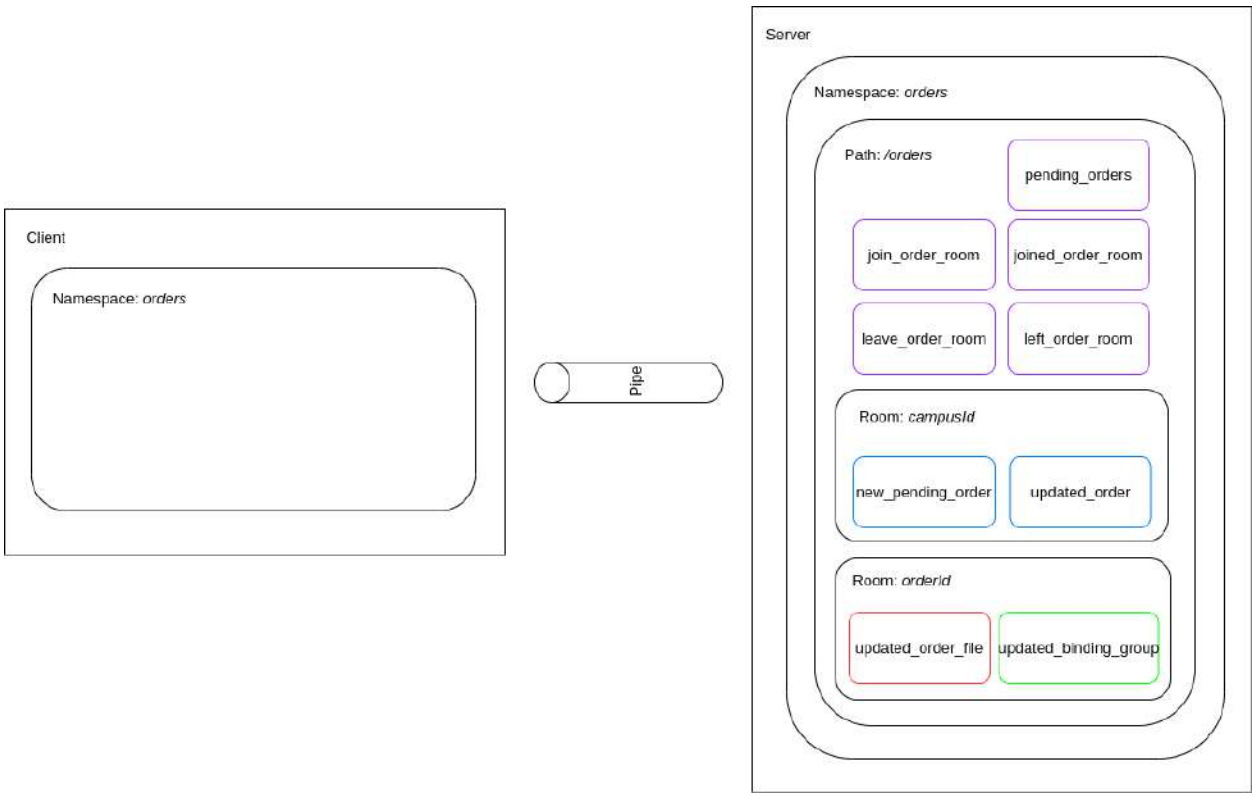



**Figura 21**. Esquema del diseño para protocolo WebSockets

Como canal de comunicación se implementó el uso de un solo Namespace llamado **orders**. Dentro de este namespace se utilizará el path **/orders** para comunicar los componentes.

Dentro de la implementación específica de la librería **Socket.IO** se encuentra la posibilidad de implementar el concepto de **salas** o **rooms**, el cual permite dividir los mensajes para cada cliente específico que lo requiera. Al encontrarse el equipo con más de una sede al momento de haber desarrollado el sistema, cada cliente web que corre en la sede correspondiente deberá ingresar a su sala con su identificador pertinente (*campusId*).

Sumado al caso de uso anterior también se presenta la necesidad de contar con una room para una orden en particular (identificada por *orderId*). En este caso se realizará la actualización automática del pedido en caso de cambio del detalle del mismo, siempre que el cliente se encuentre visualizando la configuración específica de la orden.

A continuación se explican el uso de los eventos presentados en el diagrama:

- **pending_orders**: evento utilizado para comunicar todos los pedidos que se encuentran en un estado pendiente.
- **join_order_room**: con este evento que se transmite desde el cliente a servidor se da la orden de unirse a la sala de una orden específica y así esperar cambios en la misma.
- **joined_order_room**: evento que se transmite al cliente para dar aviso que se unió correctamente a la sala de un pedido en particular.
- **leave_order_room**: evento que dispara la salida de la sala del pedido, para que el cliente asociado no esté vinculado a eventos posteriores sobre la orden.
- **left_order_room**: evento que da aviso al cliente que ya se retiró de la sala de la orden.
- **new_pending_order**: dentro de la sala según ***campusId*** este evento permite que nuevas órdenes que se generan en el backend se transmitan al cliente que se suscribió anteriormente.
- **updated_order**: este evento se dispara para dar aviso al cliente de los cambios en alguna orden en particular, para que el usuario pueda actualizar en la capa de presentación los nuevos datos.
- **updated_order_file**: este evento al cual el cliente se suscribe ocurre dentro de la sala específica de la orden, y tiene utilidad cuando se visualiza la configuración de un pedido. El servidor envía los cambios de archivos que se produzcan y el cliente los actualiza.
- **updated_binding_group**: similar al evento anterior, este ocurre cuando se realiza un cambio en cuanto a la configuración de anillado del pedido.

# 6. Bibliografía


- Web Sockets (2021). Recuperado el 25/05/2021 de https://socket.io/docs/v3/
- TypeORM (2021). Recuperado el 23/12/2020 de https://typeorm.io/#/
- HTTP Status Codes (2021). Recuperado el 22/02/2021 de https://httpstatuses.com/
- TypeScript Documentation (2021). Recuperado el 16/03/2021 de https://www.typescriptlang.org/docs/
- The WebSocket API (WebSockets) (12 de agosto de 2021). Recuperado el 29/07/2021 de https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- 10 Best Practices for Better RESTful API (5 de junio de 2014). Recuperado el 23/01/2021 de https://medium.com/@mwaysolutions/10-best-practices-for-better-restful-api-cbe81b06f291
- Páginas y rutas Angular SPA (14 de abril de 2020). Recuperado el 19/02/2021 de https://academia-binaria.com/paginas-y-rutas-angular-spa/
- Lazy-loading feature modules (2021). Recuperado el 18/08/2021 dehttps://angular.io/guide/lazy-loading-ngmodules
- NestJS Docs (2017-2021). Recuperado el 15/12/2020 de https://docs.nestjs.com/

# Proyecto Final

## Transformación Digital del proceso de gestión de impresiones del Centro de Estudiantes de Ingeniería

# Manual de Usuario

**Versión 0.9**

Mar del Plata, 21 de Diciembre de 2021

# ÍNDICE

# Introducción

El Sistema de Impresiones Online (SIO) es una plataforma en la que se puede llevar a cabo toda la gestión de pedidos que realiza el CEI, pudiendo administrar además los archivos, materias, carreras, usuarios y otras tantas entidades que hacen a su funcionamiento.

En el presente manual se describirán todas las funcionalidades implementadas en el sistema, de forma tal que cualquier nuevo usuario pueda consultar el instructivo actual y comenzar a utilizar la plataforma sin una gran curva de aprendizaje inicial.

# Acceso al sistema

El sistema cuenta con cinco tipos de usuarios o  roles que pueden acceder al mismo. Estos son:

- Estudiante
- Becado (estudiante con una beca otorgada)
- Cátedra
- Sede
- Administrador

## Inicio de sesión

Todos los usuarios pueden acceder al sistema a través de la pantalla de login.

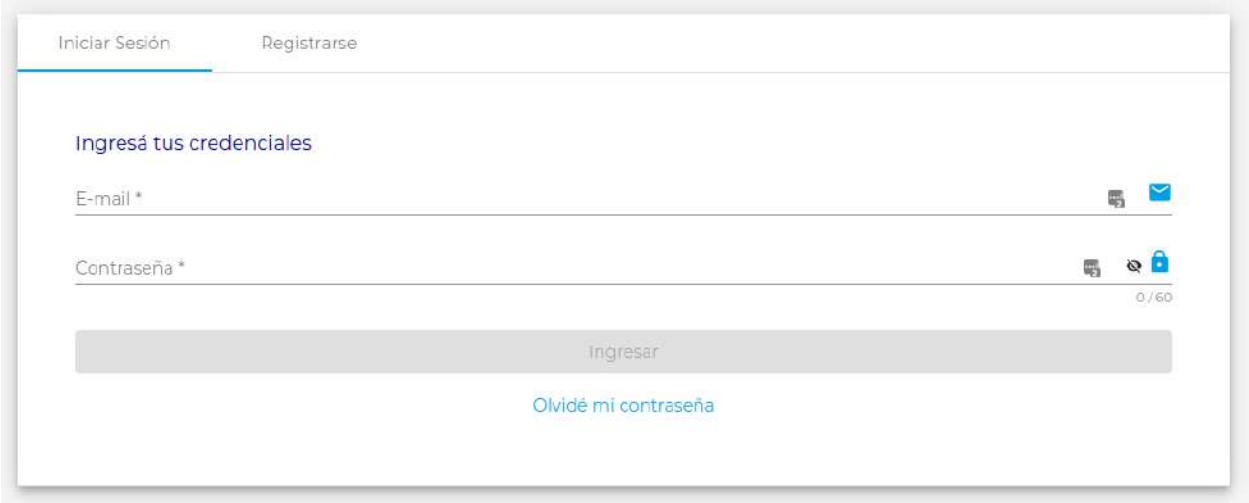



**Figura 1**. Inicio de sesión

Al ingresar su email y contraseña el sistema verificará si las credenciales son válidas. En caso de serlo, el usuario podrá acceder al sistema. Caso contrario será presentado con un mensaje de error.

## Registro de nuevo estudiante

La pantalla de registro de usuarios está enfocada solo para aquellas personas que deseen crearse una cuenta como estudiante. Para todos los demás roles (cátedra, sede y administrador), su creación será desde el rol de administrador en la pantalla correspondiente.

El estudiante que desee registrarse en el sistema deberá completar su nombre, email y contraseña deseados. En caso que todos los campos sean válidos se procederá con la creación del mismo.

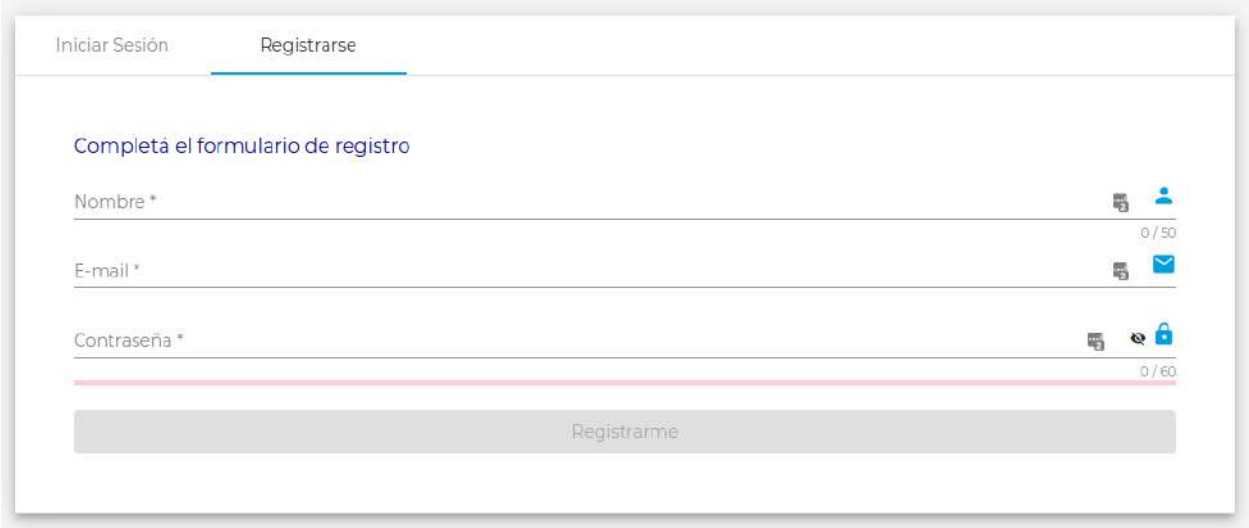



**Figura 2**. Registro de nuevo estudiante

Una vez generado el usuario, una pantalla aparecerá informando al usuario que debe verificar el email proporcionado.


**Figura 3**. Notificación para confirmar dirección de correo electrónico

El mail recibido en la casilla de correo contendrá un hipervínculo que permitirá validar al mismo y proseguir con el flujo de registro.


**Figura 4**. Mail recibido para verificar la dirección de correo electrónico

El primer ingreso al sistema del estudiante requerirá que el mismo proporcione su email para poder comenzar a operar con el mismo.

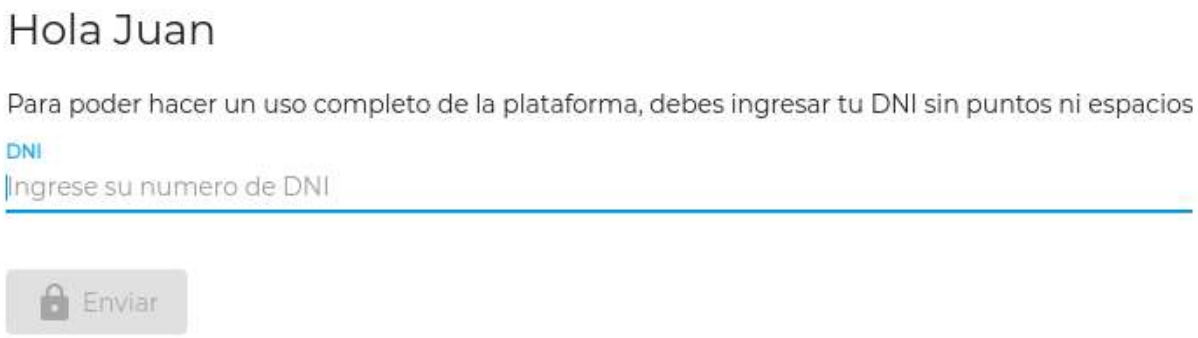



**Figura 5**. Formulario a completar para registrar DNI del estudiante

Una vez introducido un DNI válido que no haya sido registrado en el sistema, el estudiante accederá finalmente a la pantalla de inicio desde dónde podrá comenzar a operar con la plataforma.

## Olvidé mi contraseña

En caso que el usuario olvide su contraseña es posible resetear la misma a través del botón “Olvidé mi contraseña” accesible en la pantalla de inicio de sesión. Al hacer click será redirigido a una nueva sección desde la cual podrá proporcionar su email y recibir un correo electrónico de reseteo.


**Figura 6**. Reseteo de contraseña

El mail recibido contendrá un hipervínculo desde el cual se redirigirá a una nueva pantalla desde la cual podrá configurar una nueva contraseña.


**Figura 7**. Mail recibido para resetear la contraseña del usuario

La pantalla a la cual es redirigido el usuario para configurar una nueva contraseña es la siguiente:

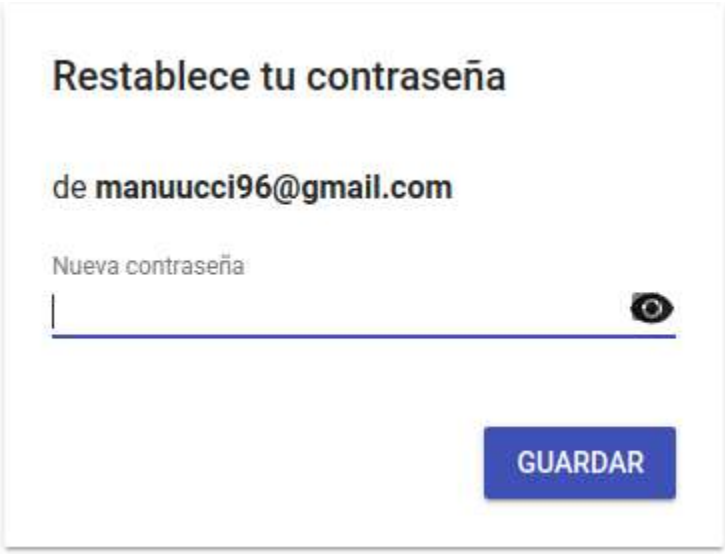



**Figura 8**. Configuración de nueva contraseña

# Pantalla de inicio

Una vez que el usuario accede al sistema será presentado con la pantalla principal o “Home”.

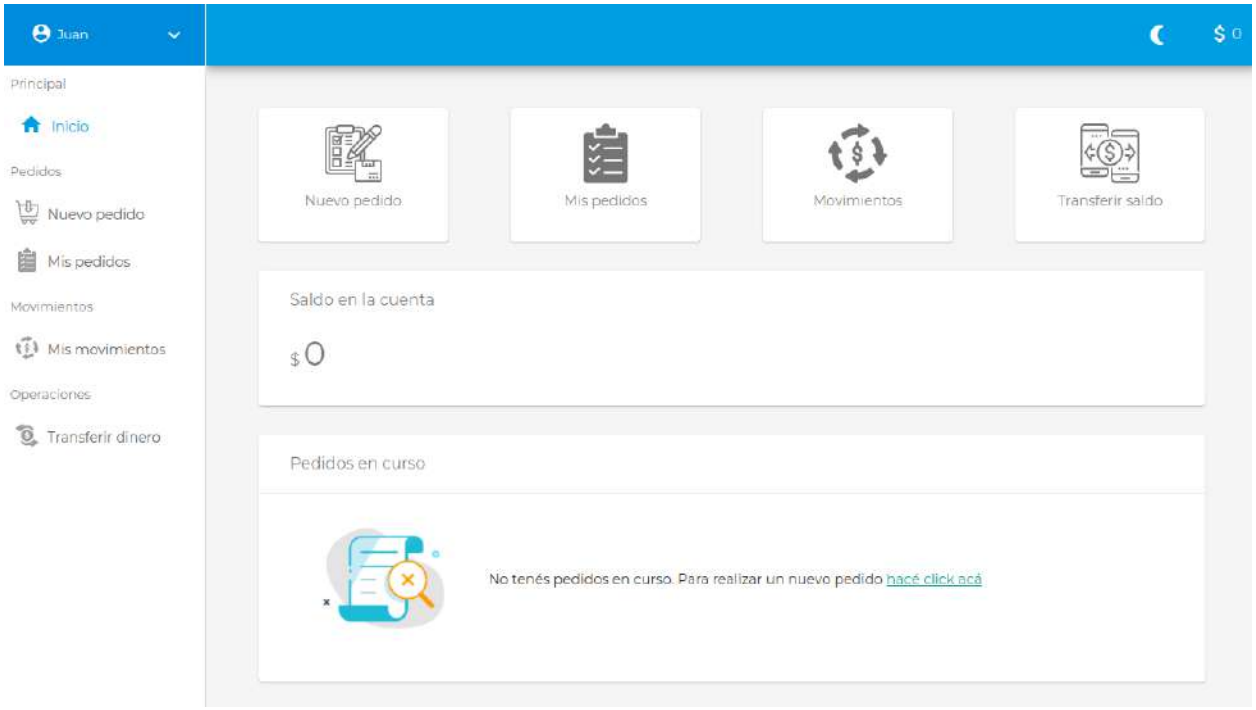



**Figura 9**. Home del estudiante

La pantalla contiene un par de componentes comunes a todos los roles.

## Menú

En primer lugar es posible observar el menú lateral izquierdo que contiene todas las funcionalidades a las que el usuario puede acceder. Haciendo click en cada una de ellas será redirigido a la pantalla correspondiente, conservándose visible el menú lateral en todo momento.

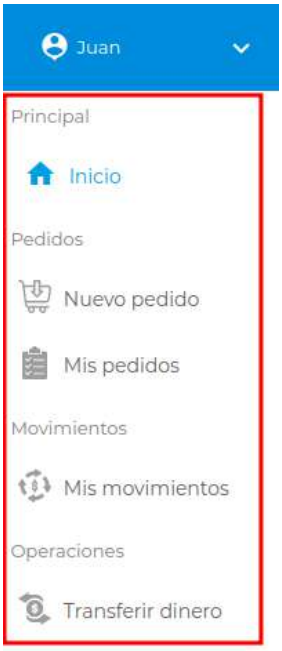



**Figura 10**. Menú del estudiante

## Submenú superior

En la esquina superior izquierda se podrá visualizar el nombre del usuario y, al hacer click en el botón con forma de flecha, un pequeño menú será desplegado que permitirá ver y alterar algunos datos del usuario, cambiar su contraseña o finalizar la sesión actual.

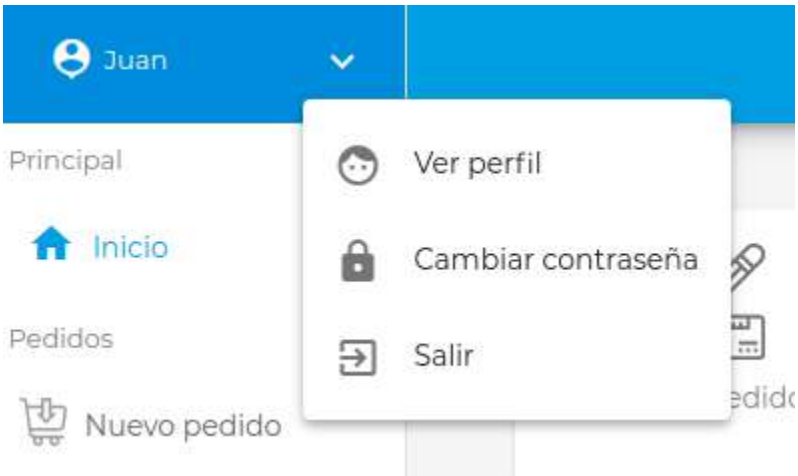



**Figura 11**. Submenú del estudiante

### Edición de perfil

Desde el botón “Ver perfil” es posible visualizar los datos del usuario y editar aquellos campos habilitados.


**Figura 12**. Edición de datos del usuario

Es importante destacar que los atributos mostrados dependen del rol que está logueado actualmente. Tanto el nombre como el email son comunes para todos los usuarios, y se agrega el DNI para el caso de los estudiantes y becados.

### Cambio de contraseña

Todo usuario puede cambiar su contraseña a través del botón de “Cambiar contraseña”, proporcionando una nueva y confirmando la misma.


**Figura 13**. Actualización de contraseña del usuario

### Fin de sesión

El usuario podrá cerrar su sesión a través del último botón del submenú accedido. Una vez que clickea sobre el botón la sesión es finalizada y es redirigido a la pantalla de inicio de sesión del sistema.

## Datos específicos del rol

En la esquina superior derecha cada usuario podrá tener acceso a ciertos atributos específicos que se busca que estén siempre visibles para el mismo, sin importar en qué pantalla se encuentre.

Para el caso puntual del estudiante podrá visualizar su saldo. En caso que fuera becado podrá además observar la cantidad de copias restantes brindadas por el CEI que tiene disponibles para consumir. Finalmente, el usuario cátedra podrá evidenciar cuánto almacenamiento disponible tiene todavía disponible para subir archivos.

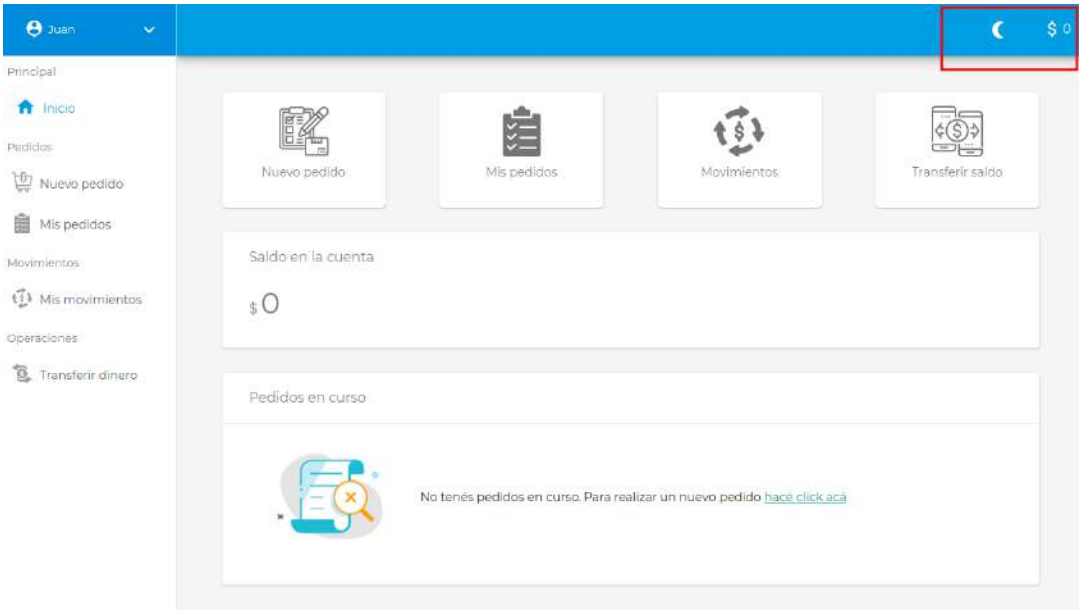



**Figura 14**. Visualización de datos específicos del usuario estudiante

Por último cabe mencionar también que el primer botón de este apartado permite alterar el tema del sistema, cambiando alternadamente entre los temas Light y Dark implementados.

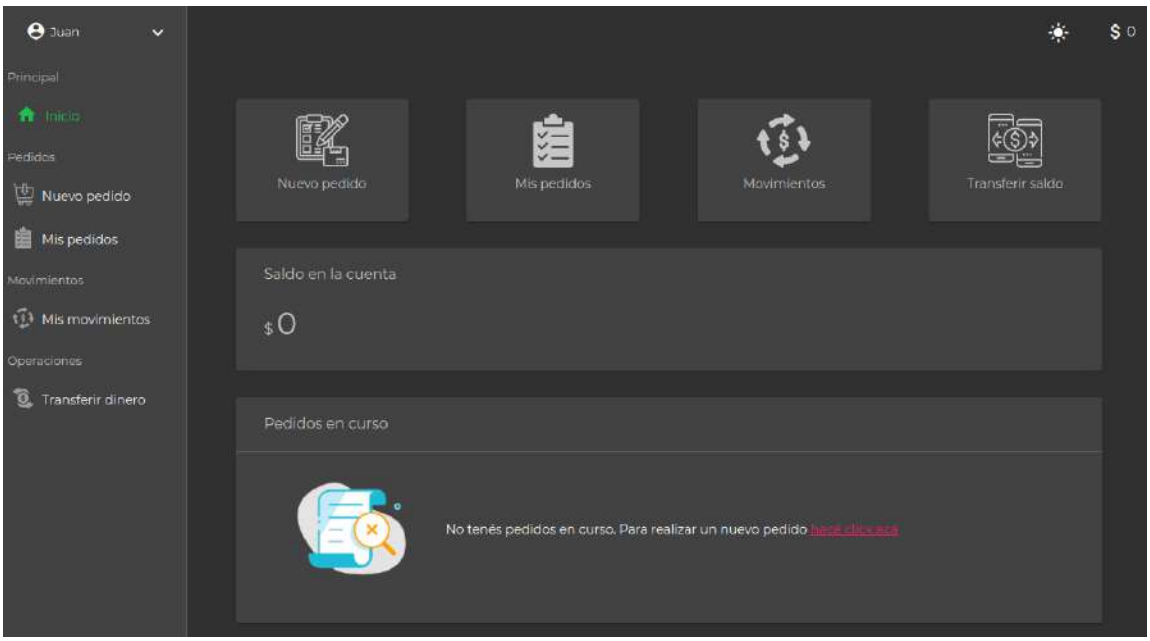



**Figura 15**. Tema Dark en funcionamiento

# Pedidos

## Solicitud de pedido

Tanto los usuarios estudiantes, como aquellos que tengan una beca otorgada, podrán acceder a la funcionalidad de solicitar un nuevo pedido de impresión dentro de la plataforma. Para ello, el usuario deberá dirigirse al ítem “Nuevo Pedido” situado en el menú principal. Al hacer click sobre el mismo, el sistema mostrará la siguiente vista:

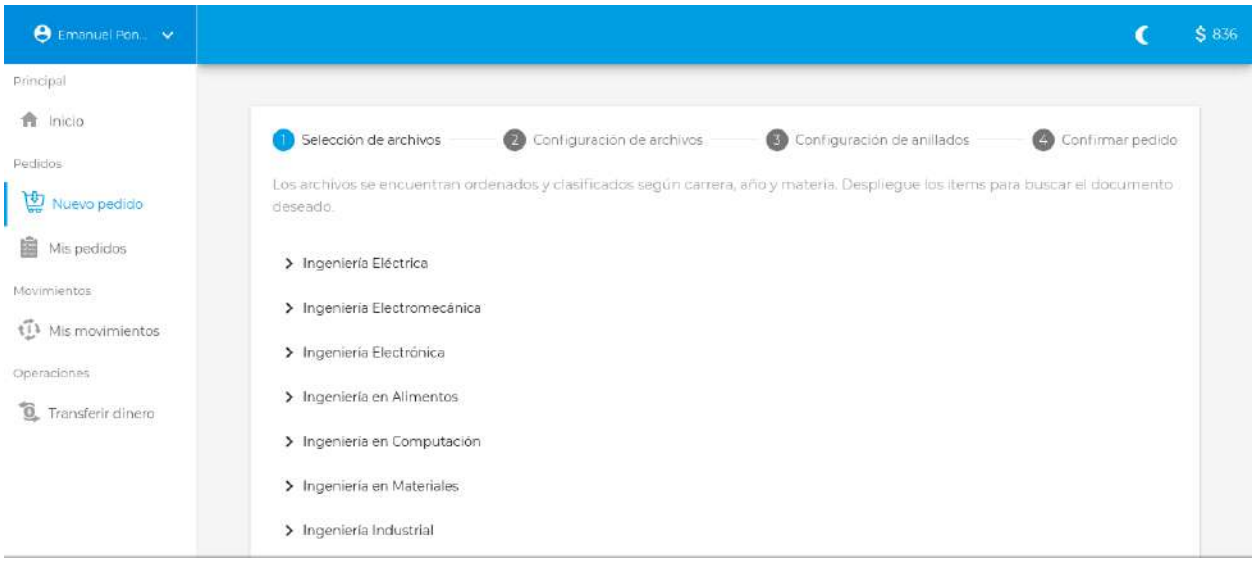



**Figura 16**. Pantalla de nuevo pedido

Como se observa en la imagen anterior, la pantalla consta de un wizard con cuatro pasos, en donde cada uno de ellos fue diseñado para personalizar una configuración específica, logrando en forma conjunta cubrir todos los aspectos necesarios del pedido.

**Paso 1: Selección de archivos**

El primer paso brinda la posibilidad de identificar y visualizar todo el repositorio de archivos y a su vez, seleccionar aquellos que se deseen incorporar en el pedido. Para ello se dispone de un componente interactivo con formato de árbol, clasificando los archivos según la carrera, año y materia a la cual pertenecen.

De esta manera, el usuario podrá realizar la búsqueda seleccionando, en primer lugar, la materia deseada:

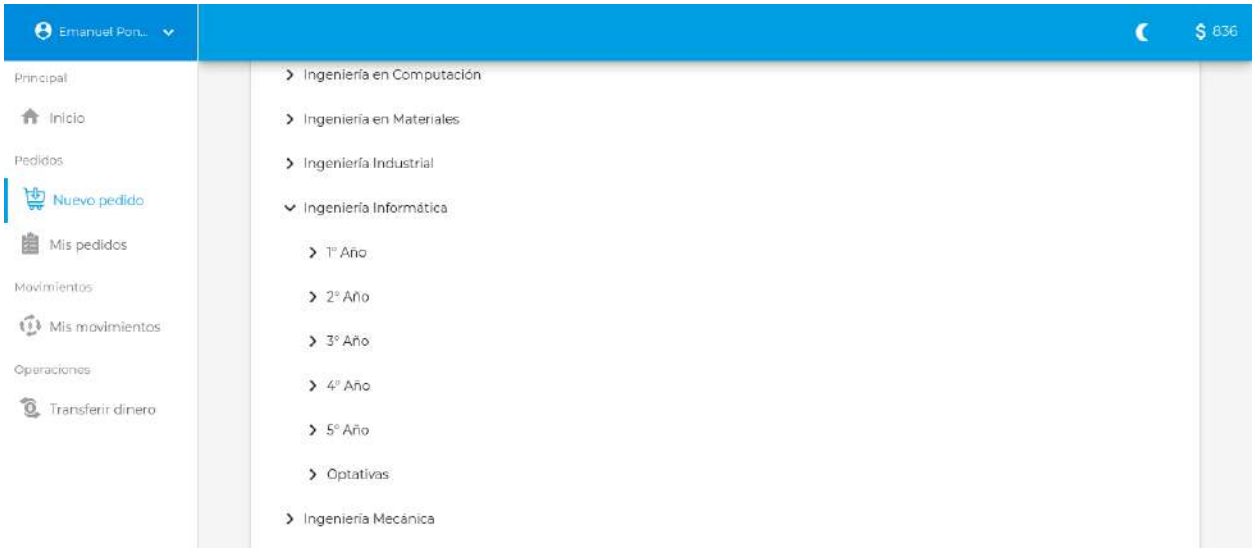



**Figura 17**. Árbol de carreras, años y materias

Al realizar esta acción, el sistema desplegará los años asociados al plan de estudio de dicha carrera, incluyendo además un ítem para aquellas materias optativas. De la misma manera, el usuario seleccionará uno de ellos y se desplegarán las materias asociadas:


**Figura 18**. Materias disponibles para una carrera y año específico

Repitiendo el mismo patrón, luego de seleccionar la materia deseada, se mostrarán todos los archivos vinculados a la clasificación indicada:

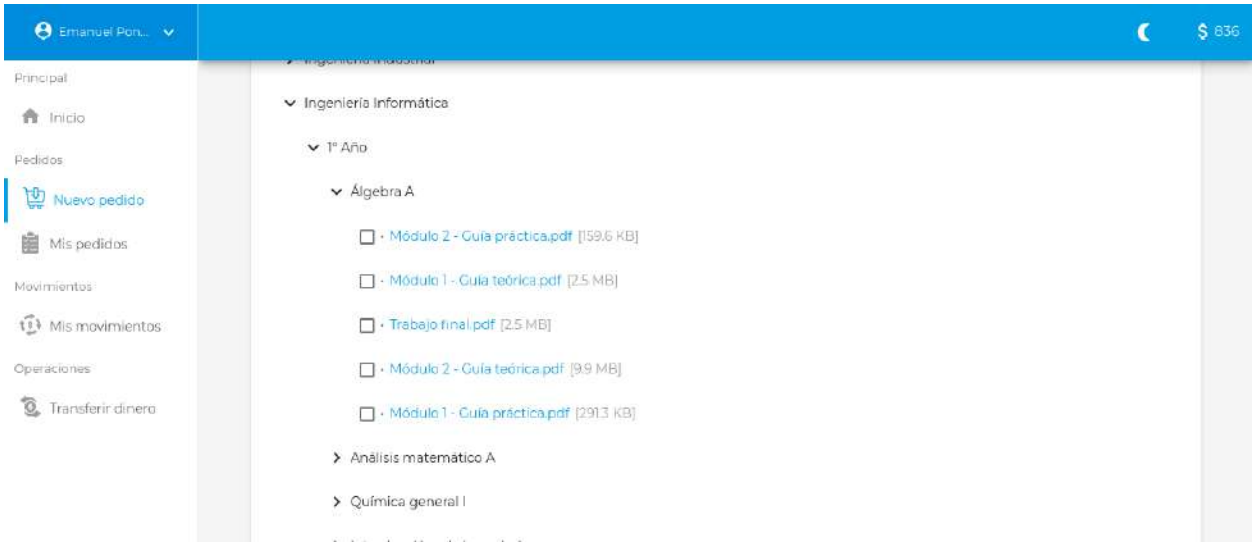



**Figura 19**. Archivos disponibles para la materia Álgebra A

Como se puede observar en la siguiente captura, cada ítem se compone en 3 partes:

1. **Campo de tipo checkbox**: este input, al ser clickeado, alternará su estado mostrando un ícono chequeado si el archivo ha sido seleccionado para ser incorporado en el pedido, o bien un cuadrado vacío en caso contrario.
2. **Nombre del archivo**: siguiente al campo de tipo checkbox se muestra el título asociado al archivo. Haciendo click sobre el mismo, el sistema abrirá en una pestaña nueva el previsualizador de pdf integrado del navegador con el fin de visualizar el archivo.
3. **Tamaño del archivo**


**Figura 20**. Estructura visual de un archivo

Luego de seleccionar al menos un archivo, se habilitará el botón con el texto “Siguiente” ubicado en la esquina inferior derecha del paso 1.

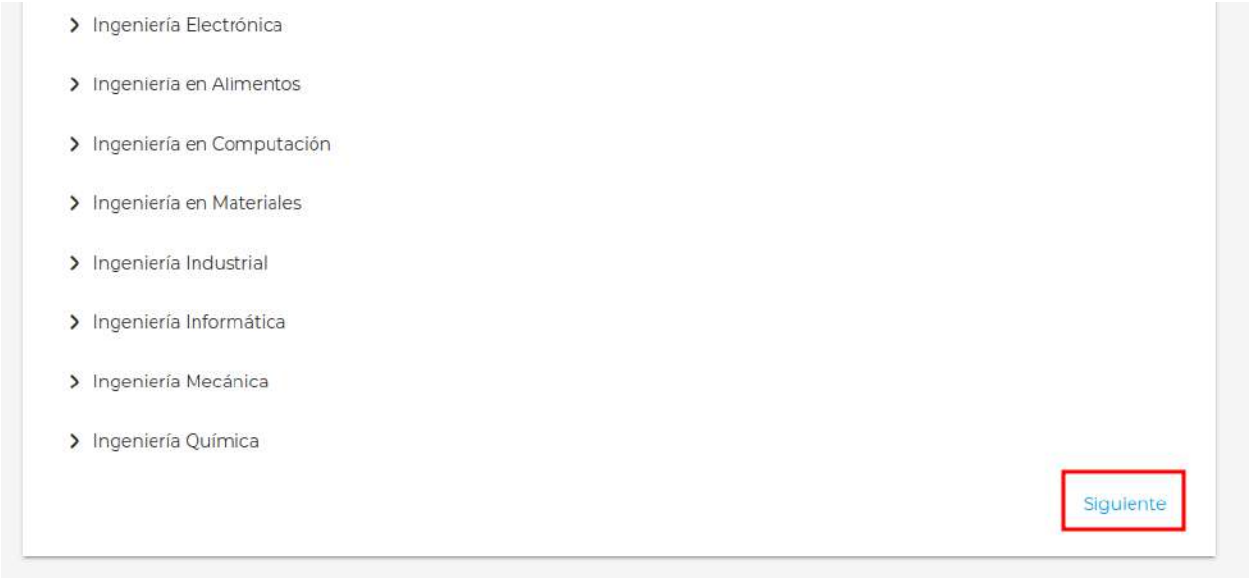



**Figura 21**. Botón para avanzar a la siguiente etapa

Al hacer click sobre el mismo (o bien sobre el paso 2 perteneciente al header del wizard), el sistema mostrará el siguiente paso: “Configuración de archivos”.

**Paso 2: Configuración de archivos**

Este apartado está destinado a personalizar todas las configuraciones necesarias a cada archivo. En primer lugar, el sistema mostrará un listado desplegable con los archivos seleccionados en el paso anterior:

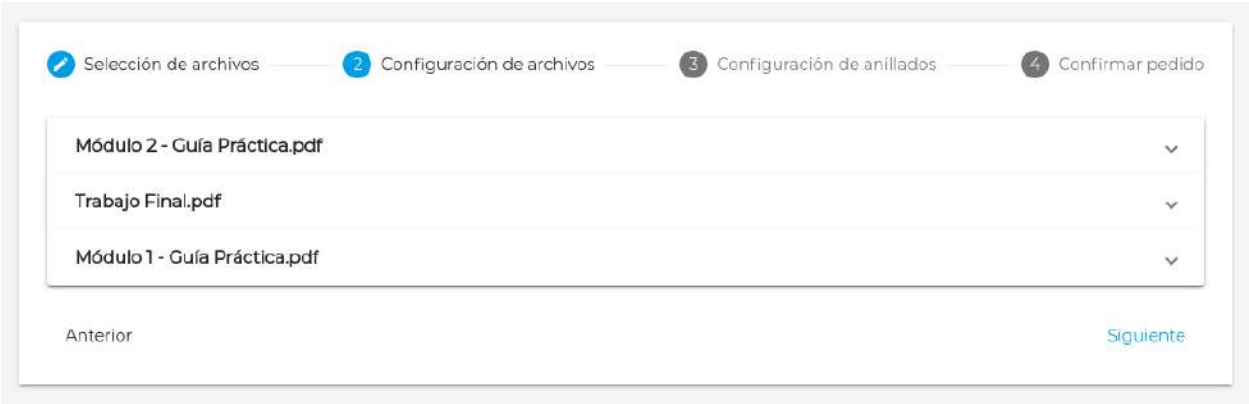



**Figura 22** Listado de archivos seleccionados en el paso 1

Al hacer click sobre uno de ellos, se desplegarán las opciones de configuración asociadas a dicho archivo:

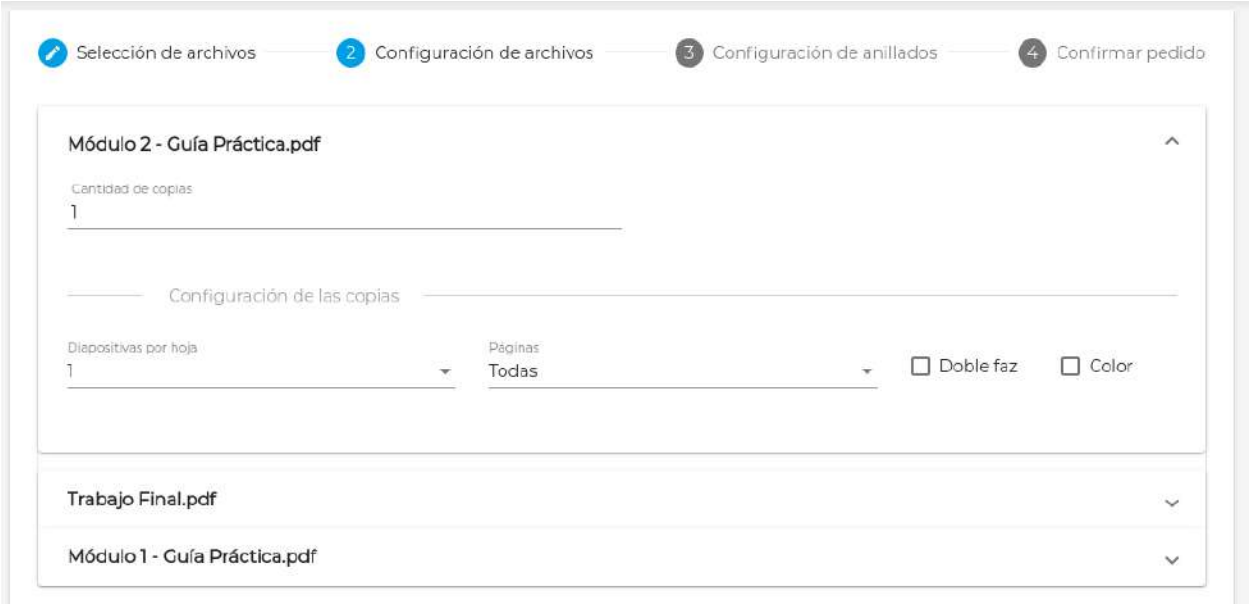



**Figura 23**. Configuración de archivo

La configuración consta de un campo principal destinado a indicar la cantidad de copias que se desean realizar de dicho archivo. Por defecto este campo vendrá inicializado con el valor 1. En caso de que el usuario aumente la cantidad de copias, el sistema mostrará un nuevo componente junto a éste, para determinar si todas las copias tendrán la misma configuración:

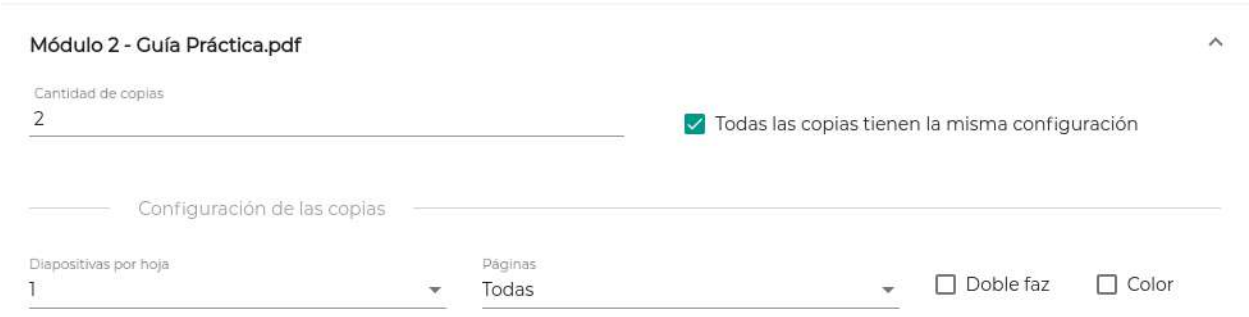



**Figura 24**. Aumento en cantidad de copias del archivo a configurar

En caso de que el usuario desactive esta opción, el sistema dinámicamente mostrará una fila de configuración según la cantidad indicada (en lugar de tener una única fila con el texto “Configuración de las copias”), permitiendo personalizar cada copia de manera independiente:

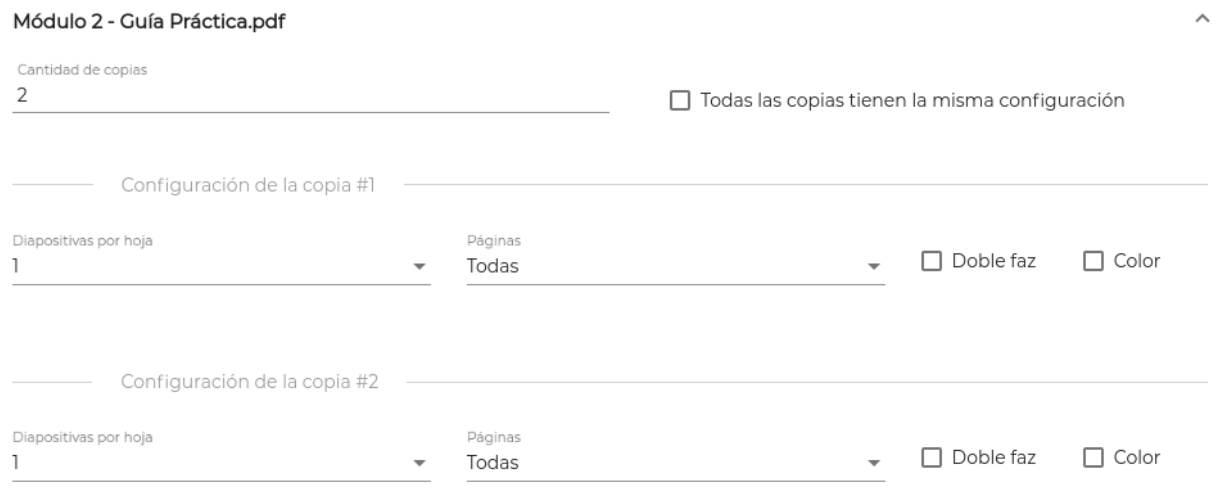



**Figura 25**. Configuración individual para las diferentes copias del archivo

La configuración asociada a cada copia (o al conjunto de copias asociadas al archivo en caso de que el usuario haya activado la opción “Todas las copias tienen la misma configuración”) contendrá los siguientes campos

- **Diapositivas por hoja:** componente de tipo Select que permite elegir 1, 2, 4 o 6 páginas por hoja.
- **Páginas:** componente de tipo Select que permite elegir entre las opciones “Todas” (se seleccionarán la totalidad de páginas asociadas al archivo) o “Personalizado” (en caso de querer seleccionar ciertas páginas o rangos específicos). En caso de que el usuario seleccione la opción “Personalizado”, el sistema mostrará un campo de texto debajo del Select, que será utilizado para indicar las páginas que se desean seleccionar. Para ello, se deberá ingresar una expresión regular estandarizada, admitiendo el ingreso de número de hojas específicas separadas por coma (ej, “1,4,8,11”), un rango de hojas (“1-5”, lo que dará como resultado las páginas 1,2,3,4 y 6) o bien la combinación de ambas sintaxis (“1-5,8,10,13-15”).


**Figura 26**. Configuración personalizada de rango de hojas

Además, el campo posee un texto explicativo del formato aceptado, y junto a él, se le informa al usuario la cantidad de páginas totales que posee el archivo seleccionado. En caso de que el formato ingresado sea erróneo o bien se ingrese un valor fuera del rango soportado, el sistema informará de la situación con un mensaje de error explicando el motivo:

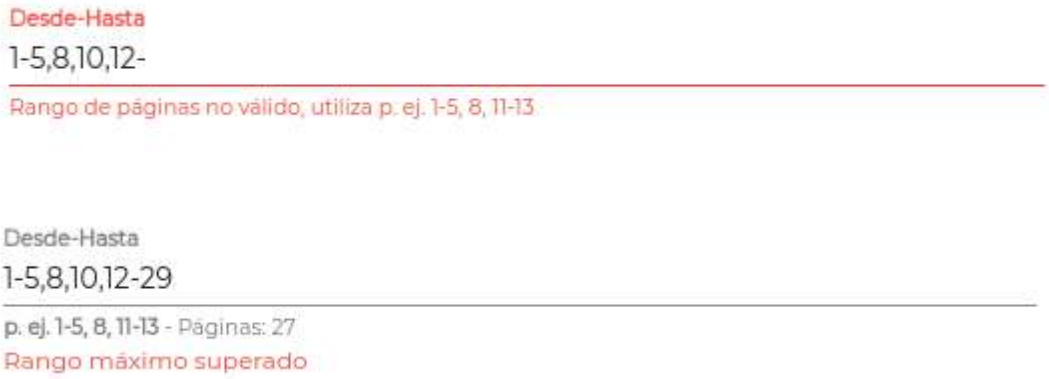



**Figura 27**. Errores comunes en configuración de rango de hojas

- **Doble faz:** campo de tipo checkbox para indicar si el rango de páginas ingresado asociado al archivo seleccionado, deberán imprimirse en formato doble faz (en caso de que el checkbox esté activado) o en formato simple faz (en caso de que el checkbox esté desactivado).
- **Color:** campo de tipo checkbox para indicar si el rango de páginas ingresado asociado al archivo seleccionado, deberán imprimirse en formato color (en caso de que el checkbox esté activado) o en formato blanco y negro (en caso de que el checkbox esté desactivado).

  Luego de haber configurado correctamente todas las copias de todos los archivos seleccionados, se habilitará un botón con el texto “Siguiente” ubicado en la esquina inferior derecha. Al hacer click sobre el mismo (o bien sobre el paso 3 perteneciente al header del wizard), el sistema mostrará el siguiente paso: “Configuración de anillados”.

**Paso 3: Configuración de anillados**

Esta sección tiene como objetivo brindarle al usuario la posibilidad de configurar, opcionalmente, los archivos que desea anillar. El sistema tiene la capacidad de seleccionar la cantidad de grupos de anillados que se desean, así como la posibilidad de asociar archivos a cada grupo de anillado en el orden que se crea conveniente.

Para ello, se cuentan con dos secciones internas dentro del paso. A la izquierda se puede observar el listado de archivos que el usuario seleccionó y configuró en los pasos anteriores, y a la derecha, un apartado para visualizar los grupos de anillados configurados así como los archivos y el orden que contiene cada uno de ellos (esta funcionalidad se verá a continuación).


**Figura 28**. Configuración de grupos de anillado

**Listado de archivos (sección izquierda)**

Como se mencionó anteriormente, este listado refleja todos los archivos que se hayan seleccionado en el paso 1 “Selección de archivos”. Cada ítem muestra el nombre del archivo asociado, la materia a la cuál pertenece (encerrada entre paréntesis) y por último un botón con un ícono (flecha orientada hacia la derecha) cuya acción concluye en la inserción del archivo dentro del grupo de anillado activo.

***Nota**: en caso de que la cantidad de copias (campo configurado en el paso 2 “Configuración de archivos”) asociadas a un archivo sea mayor que 1, cada una de ellas podrá tener su propia configuración de anillado. Para ello, los ítems pertenecientes a ese archivo indicarán un dato numérico incremental extra para identificar cada copia. Como ejemplo, la siguiente imagen muestra el caso en que se configuraron 2 copias del archivo “Trabajo final.pdf” de la materia “Álgebra A”. Notar el identificador incremental separado por guión medio de la materia.*


**Figura 29**. Múltiples copias de un mismo archivo a ser anilladas

### Grupos de anillados (sección derecha)

En esta sección se pueden visualizar, agregar y eliminar grupos de anillados (mostrando además el tipo de anillado que se utilizará a partir de la cantidad de hojas a anillar), así como qué archivos están asociados a cada uno de ellos y el orden en los que se anillarán.


**Figura 30**. Grupos de anillados configurados para el pedido a generar

A medida que el usuario selecciona archivos del listado izquierdo, los mismos son incorporados en el grupo de anillado seleccionado:


**Figura 31**. Adición de archivos a grupo de anillado

De la imagen anterior podemos destacar algunos aspectos importantes:

- El tipo de anillado se calcula automáticamente a partir de la cantidad de hojas totales asociadas a los archivos pertenecientes al grupo de anillado seleccionado. Por ejemplo, si se aumenta la cantidad de copias, el tipo de anillado correspondiente al cálculo realizado por el sistema será “Anillado mediano”


**Figura 32**. Adaptación de grupo de anillado seleccionado en función de cantidad de hojas

- Los ítems reflejan su nombre con la misma estructura que su homónimo del listado izquierdo. Seguido de ese texto, sobre el sector derecho, se encuentra el orden en que se procederán a anillar los archivos (los números respetan el orden en que el usuario seleccionó los archivos del listado izquierdo).
- El botón rojo ubicado a la izquierda de cada ítem sirve para quitar dicho archivo del grupo de anillado. Notar en la primera imagen que al agregar un archivo al grupo de anillado el botón con forma de flecha del sector derecho se deshabilita.
- En caso de querer agregar un grupo de anillado nuevo, se debe seleccionar el botón con un ícono “+” ubicado en la parte superior del listado. Al presionar dicho botón, el sistema agregará un tab asociado al nuevo grupo de anillado (cada grupo de anillado posee un

número incremental automático). Notar además, que al tener más de un grupo de anillado creado, el botón de eliminar (ubicado a la derecha del botón “+”) se habilita.


**Figura 33**. Creación de un segundo grupo de anillado

Al igual que con el primer grupo de anillado, el usuario puede seleccionar el/los archivo/s que desee anillar (con un anillado independiente al configurado en primera instancia):

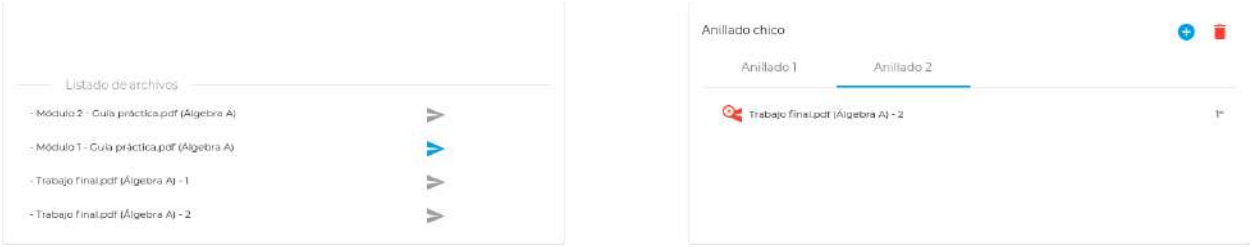



**Figura 34**. Configuración de nuevo grupo de anillado

En caso de querer eliminar un grupo de anillado, presionando el botón de eliminar ubicado a la derecha del botón “+”, el sistema quita ese grupo de anillado, eliminando el tab asociado y activando nuevamente los botones del listado izquierdo asociados a aquellos archivos que se encontraban en dicho grupo.

***Nota**: en caso de que el usuario ingrese cierta cantidad de archivos cuya sumatoria de hojas supere el rango permitido por el anillado de máxima capacidad, el sistema informará de dicha situación, impidiendo realizar la acción:*


**Figura 35**. Error por superar límite máximo de hojas permitido para el grupo de anillado

Una vez que se finaliza con la configuración de los grupos de anillados, el usuario puede seleccionar, al igual que en los pasos anteriores, el botón “Siguiente” ubicado en la esquina inferior derecha (o bien sobre el paso 4 perteneciente al header del wizard). El sistema mostrará el siguiente paso: “Confirmar pedido”.

**Paso 4: Confirmar pedido**

Este último paso tiene como finalidad seleccionar la sede donde se retirará físicamente el pedido, así como visualizar y confirmar toda la información configurada en los pasos anteriores.

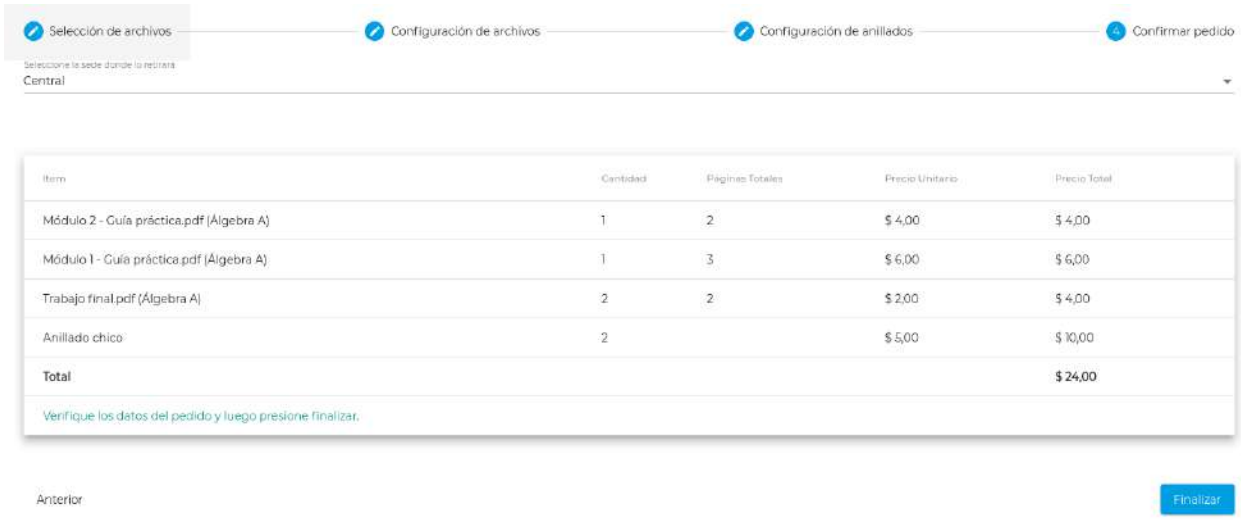
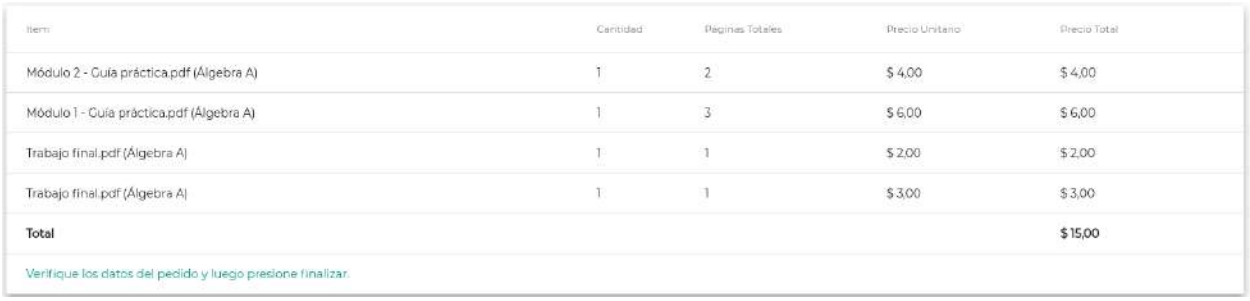



**Figura 36**. Confirmación de pedido

El usuario podrá seleccionar entre las sedes disponibles en el componente Select que se encuentra en la sección superior de la vista (en la imagen anterior, se seleccionó la sede “Central”).

La tabla muestra los archivos y anillados seleccionados en los pasos anteriores. Por cada ítem, se muestran los campos “Cantidad” (cantidad de unidades del ítem correspondiente), “Páginas totales” (sumatoria de páginas pertenecientes al archivo, este campo no mostrará ningún valor para los ítems de tipo anillado), “Precio unitario” y “Precio total”.

***Nota**: en caso que el usuario haya seleccionado más de una copia para un determinado archivo y esté activa la opción “Todas las copias tienen la misma configuración” (opción perteneciente al paso 2), la tabla mostrará un único ítem asociado al archivo con la columna “Cantidad” reflejando el valor ingresado en el campo “Cantidad de copias” (ejemplo que se observa en la imagen anterior con el archivo “Trabajo final.pdf (Álgebra A)”). En caso de que la opción “Todas las copias tienen la misma configuración” se haya desactivado, la tabla mostrará dos ítems independientes (ya que cada uno puede tener configuraciones diferentes):*



| Ítem | Cantidad | Páginas Totales | Precio Unitario | Precio Total |
|---|---|---|---|---|
| Módulo 2 - Guía práctica.pdf (Álgebra A) | 1 | 2 | $ 4,00 | $ 4,00 |
| Módulo 1 - Guía práctica.pdf (Álgebra A) | 1 | 3 | $ 6,00 | $ 6,00 |
| Trabajo final.pdf (Álgebra A) | 1 | 1 | $ 2,00 | $ 2,00 |
| Trabajo final.pdf (Álgebra A) | 1 | 1 | $ 3,00 | $ 3,00 |
| **Total** | | | | **$ 15,00** |

Verifique los datos del pedido y luego presione finalizar.

**Figura 37**. Descripción de ítems para mismo archivo con diferente configuración

En la imagen anterior se configuraron dos copias independientes del archivo “Trabajo final.pdf (Álgebra A)”. Se observan distintos precios ya que la primera se seleccionó con un formato doble faz y la segunda con un formato simple faz.

Para confirmar el pedido, luego que el usuario revisó la información brindada por el sistema y seleccionó la sede donde lo desea retirar, se debe presionar sobre el botón “Finalizar” ubicado en la esquina inferior derecha de la vista. Al realizar dicha acción el sistema corrobora que el usuario tenga saldo disponible para realizar el pedido. En caso favorable, se le mostrará al usuario un modal de confirmación, como se muestra en la siguiente imagen.


**Figura 38**. Mensaje de solicitud de pedido correcto

Al hacer click sobre el botón “Ok”, el sistema redirige al usuario a la pantalla Home, la cual indica el nuevo saldo y el estado del pedido recientemente solicitado:

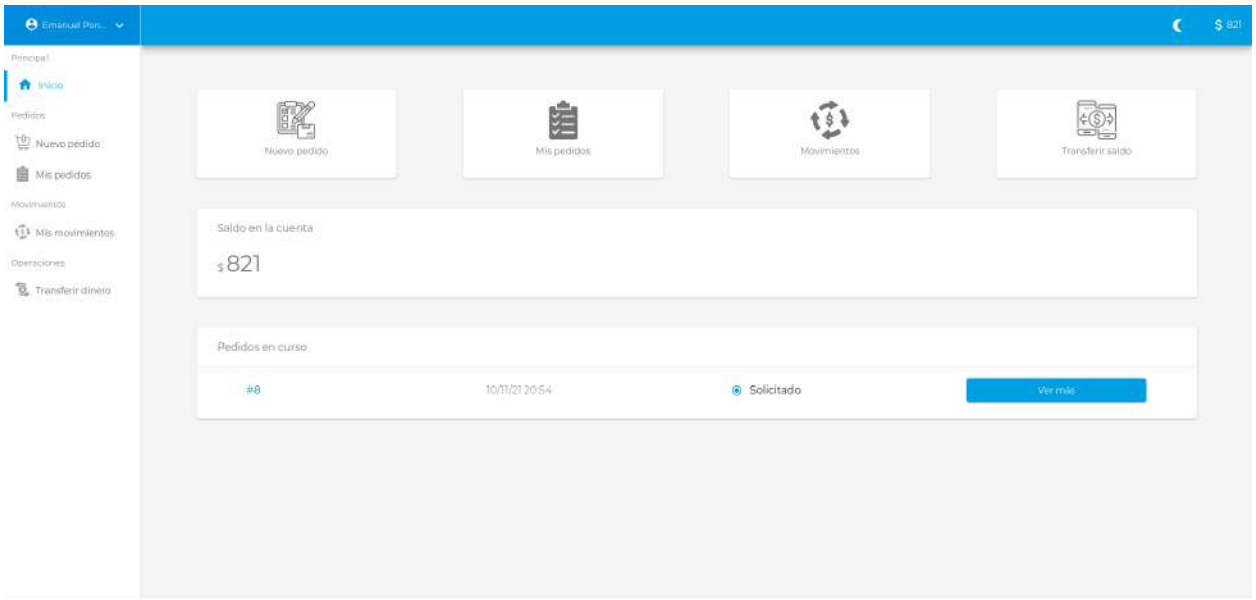



**Figura 39**. Home del usuario con nuevo pedido en curso

En caso que el usuario no tenga saldo disponible para efectuar el pedido, al confirmar el mismo, el sistema lanzará el siguiente modal de error:


**Figura 40**. Mensaje de error por saldo insuficiente

*Nota: en caso que el estudiante posea una beca otorgada por el CEI, el paso 4 "Confirmar pedido" se verá de la siguiente manera:*

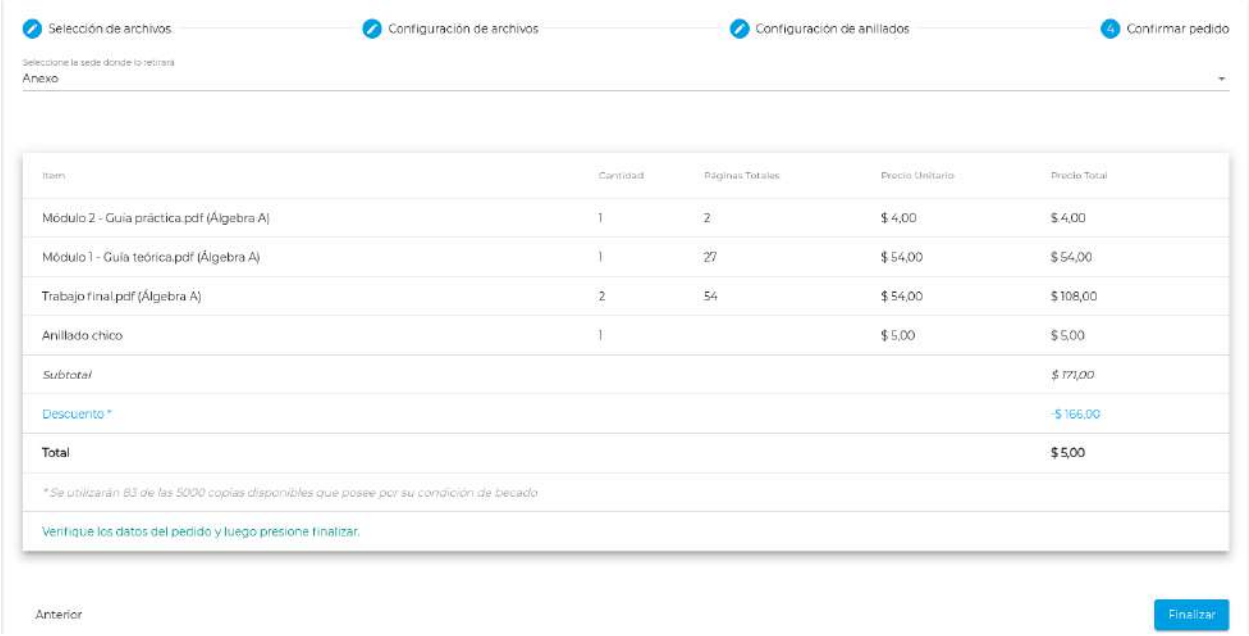



**Figura 41**. Pantalla de confirmación de pedido para usuario becado

*La tabla, en comparación con la mostrada en el caso anterior asociado a un estudiante que no posee una beca, muestra las filas "Subtotal" (indica el dinero que representa la solicitud del pedido sin aplicar*

*descuentos) y "Descuento" (indica el monto que se le descuenta al usuario por ser poseedor de una beca). En este caso de ejemplo, el sistema descontó $166 que representa el monto total asociado a las copias solicitadas (la beca sólo se aplica a copias, no a anillados) consumiendo 83 de las 5000 copias disponibles que posee el becado. El sistema siempre intentará consumir tantas copias como sean posibles según la cantidad disponible que posea el becado. De esta manera el campo "Total" refleja el subtotal menos el descuento aplicado, que será lo que finalmente deberá abonar el estudiante.*

## Seguimiento de pedidos

Luego de que un usuario estudiante o becado haya emitido uno o más pedidos, el sistema brinda una herramienta destinada a realizar el seguimiento de los mismos en tiempo real.

Para visualizar dicho seguimiento, el usuario debe dirigirse a la vista "Mis pedidos", la cual muestra por defecto un listado con todos los pedidos activos asociados al estudiante:

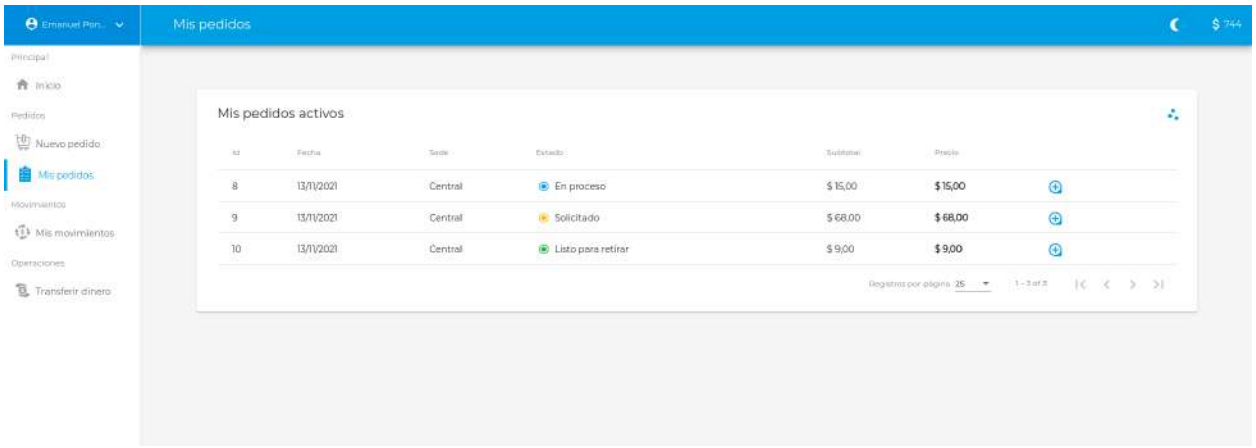



**Figura 42**. Listado de pedidos activos asociados a un estudiante

La tabla presentada anteriormente muestra, por cada pedido, los siguientes datos:

- **Id**: valor numérico que identifica unívocamente al pedido.
- **Fecha**: fecha de solicitud del pedido.
- **Sede**: la sede que el usuario seleccionó para retirar el pedido físicamente.
- **Estado**: estado actual del pedido. Los posibles valores de los pedidos activos son los siguientes:
    - **Solicitado**: estado inicial del pedido luego de que el usuario confirmó su solicitud.
    - **En proceso**: al menos un archivo ha finalizado su impresión, pero aún restan acciones asociadas a la impresión o al anillado para finalizar el pedido solicitado.

    - **Listo para retirar**: se han impreso todos los archivos (y anillados correspondientes en caso de que se hayan solicitado) y el pedido se encuentra listo para retirar en la sede que haya indicado el usuario.

  Por otro lado, aquellos pedidos finalizados pueden tomar uno de los siguientes estados:

    - **Entregado**: el estudiante retiró correctamente el pedido por la sede indicada.
    - **Cancelado**: el usuario estudiante o el usuario sede canceló el pedido.
    - **No entregado**: el pedido se encontraba listo para retirar pero el usuario sede indicó que por algún motivo el mismo no fue entregado al estudiante.
- **Subtotal**: precio del pedido antes de realizar los descuentos otorgados por la condición de becado del estudiante (en caso de que así sea)
- **Precio**: precio final del pedido abonado por el usuario luego de haber realizado los descuentos correspondientes.
- **Detalles**: botón para ingresar al detalle de un pedido.

Si el usuario desea ver el detalle de un pedido, debe presionar sobre el botón "+" asociado a la fila correspondiente. Esta acción lo redirigirá a una nueva vista de detalle:

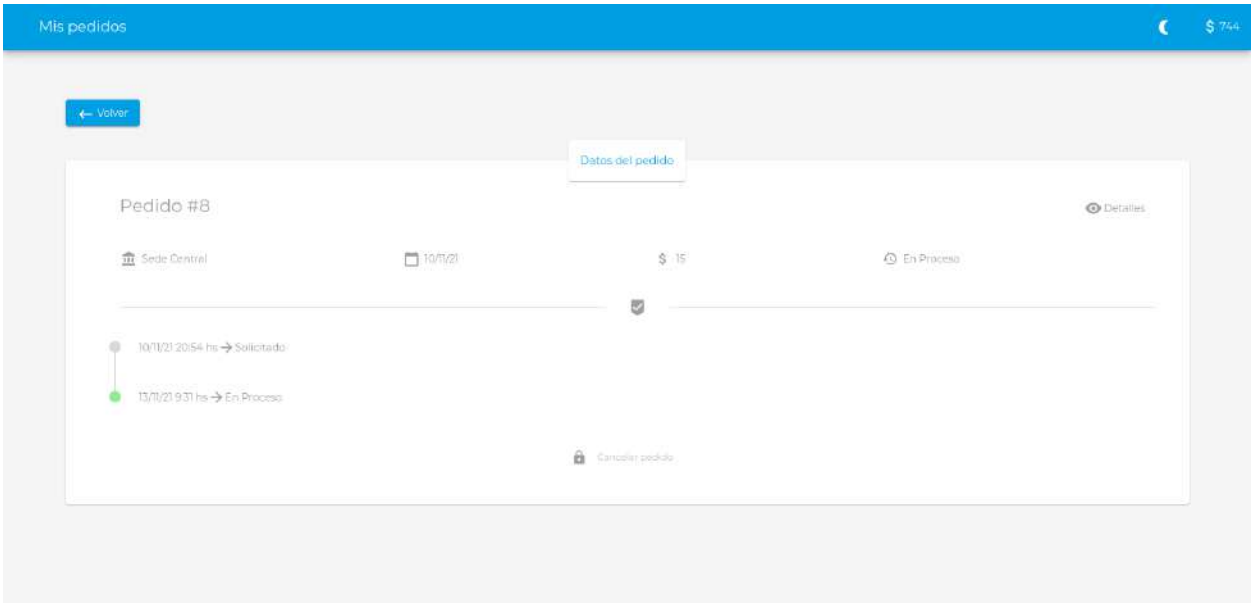



**Figura 43**. Página de detalles de un pedido para el usuario estudiante o becado

En el header de la sección principal, se encuentra la información indicada previamente en la tabla (sede en la cual se retirará el pedido, fecha de solicitud, monto final abonado por el estudiante y estado actual del pedido, respectivamente). En la sección central, se le indica al usuario el trackeo del pedido, detallando la fecha y hora en la que se produjo cada cambio de estado. En la sección inferior

se ubica un botón que le posibilita al estudiante cancelar el pedido. Esta acción pasará el pedido a estado “Cancelado” y retornará el total del saldo abonado a la cuenta del usuario solicitante. Dicho botón únicamente estará habilitado en caso de que el pedido se encuentre en estado “Solicitado”, es decir, una vez que el pedido comience su proceso de impresión, el estudiante no podrá cancelarlo para reincorporar el saldo abonado.

Si el usuario desea ver la configuración del pedido seleccionado, debe presionar sobre el botón “Detalles” ubicado en la esquina superior derecha. Esta acción mostrará el siguiente modal:

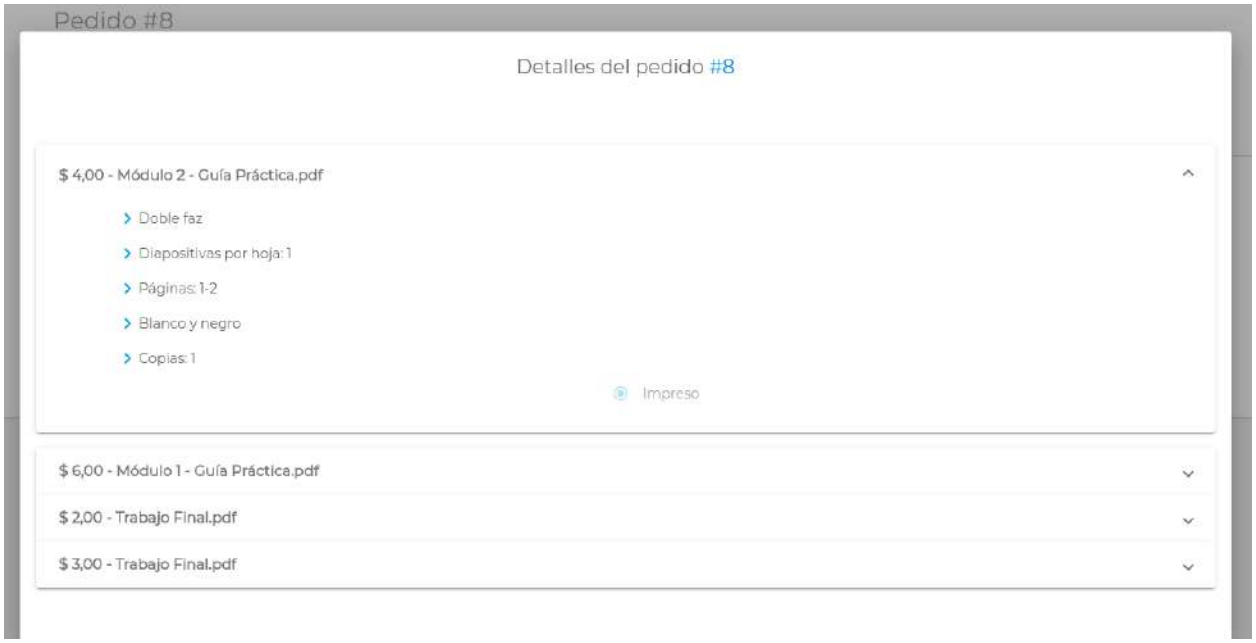



**Figura 44**. Modal que detalla la configuración y el estado de un pedido

Este componente contiene información detallada asociada a la configuración de cada archivo perteneciente al pedido así como los anillados solicitados. Cada uno de estos ítems pueden desplegar su información haciendo click sobre la flecha asociada que se encuentra en el lateral derecho.

En el header de cada ítem se detalla su precio asociado y el nombre del mismo separados por un guión medio. Para el caso de los archivos, se detalla la siguiente información:

- Si el archivo fue configurado con un formato de doble o simple faz.
- Cantidad de páginas por hoja.
- Rango de páginas seleccionado.
- Si la impresión debe ser a color o en blanco y negro.
- Cantidad de copias solicitadas.

Por último, se detalla en tiempo real el estado que se encuentra cada archivo (Impreso/Por imprimir).

En caso de los anillados, se le indica al usuario los archivos que lo componen así como el orden en el que serán ubicados:

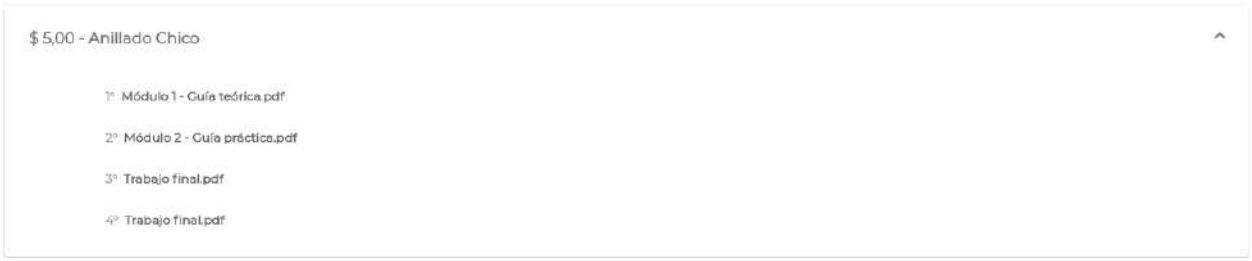



**Figura 45**. Detalles de un ítem de tipo Anillado

***Nota:*** *el sistema además brinda la posibilidad de visualizar el listado de pedidos históricos realizados por el estudiante. Para ello, el usuario debe presionar sobre el ícono que se encuentra en la esquina superior derecha y seleccionar la opción "Mostrar el historial de pedidos"*

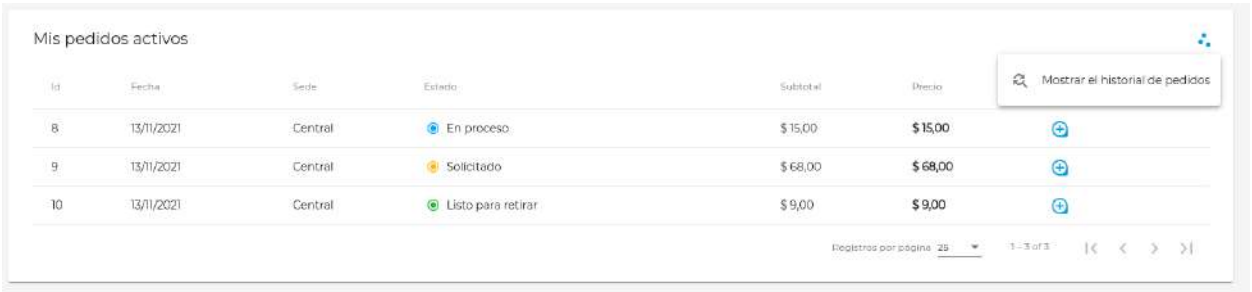



**Figura 46**. Botón para mostrar pedidos históricos

Luego de haber seleccionado dicha opción, el sistema mostrará por pantalla la misma tabla pero contenida por aquellos pedidos que se finalizaron (los datos y métodos de acceso al detalle de cada uno de ellos es idéntico a lo que se detalla para los pedidos activos).

**Figura 47**. Listado de pedidos históricos realizados por el estudiante o becado

Para volver al listado de pedidos activos, el usuario debe seguir el mismo procedimiento nombrado anteriormente, presionando sobre el ícono que se encuentra en la esquina superior derecha y seleccionando la opción “Mostrar sólo los pedidos activos”

***Nota:*** *el usuario estudiante o becado podrá también ver un resumen de sus pedidos activos en la pantalla Home (sección ubicada en la parte inferior de la vista principal):*

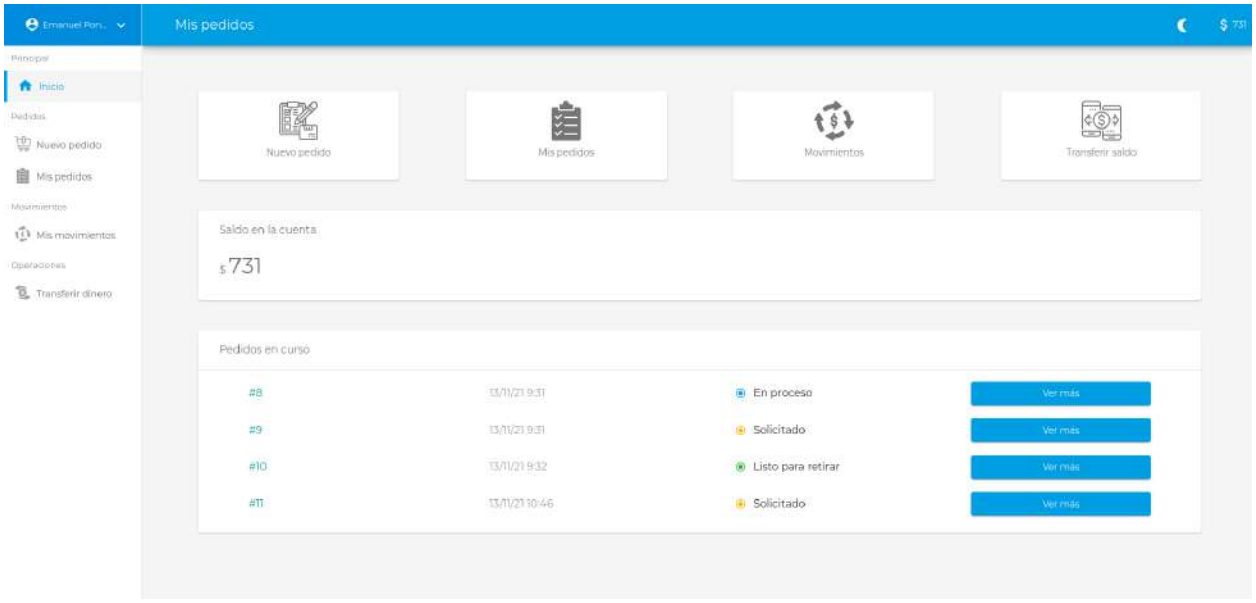



**Figura 48**. Listado de pedidos activos en la pantalla Home

*Como se observa en la imagen anterior, se le ofrece al usuario a modo de resumen el número de identificación del pedido, la fecha de solicitud y el estado del mismo.*

*El sistema además brinda la posibilidad de acceder al detalle de cada pedido haciendo click sobre el botón “Ver más”.*

## Gestión de pedidos

El usuario de tipo sede es el encargado de gestionar los pedidos solicitados por los estudiantes. La vista “Pedidos activos” tiene como finalidad brindar las herramientas necesarias para llevar a cabo la administración de los mismos.

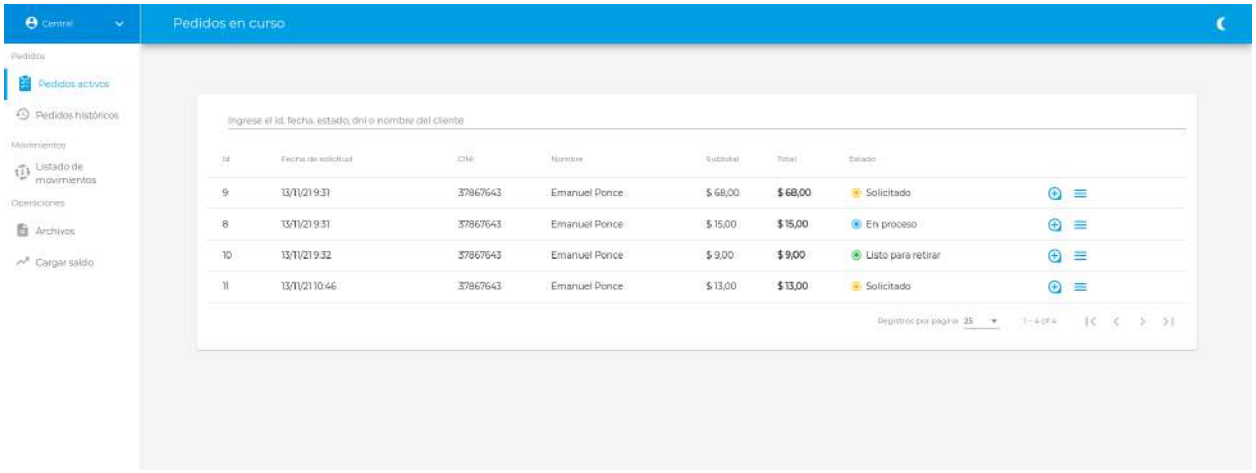



**Figura 49**. Listado de pedidos activos asociados al usuario sede

Esta pantalla muestra una tabla con los pedidos activos pertenecientes al usuario sede que han solicitado los estudiantes. Dicho listado de pedidos se actualiza en tiempo real, es decir, no es necesaria ninguna acción del usuario (tales como refrescar la página o interactuar con las vistas) para que el listado refleje los nuevos pedidos entrantes, los cambios de estado, o bien quite de esta vista aquellos pedidos finalizados.

En la sección superior se muestra un campo para que el usuario filtre los pedidos activos por el número identificatorio, fecha de solicitud o estado del pedido así como por el DNI o nombre/apellido del estudiante solicitante.

La tabla de pedidos muestra las siguientes columnas:

- **Id**: valor numérico que identifica unívocamente al pedido.
- **Fecha de solicitud**: fecha y hora de solicitud del pedido.
- **DNI**: número del Documento Nacional de Identidad del estudiante que solicitó el pedido.
- **Nombre**: nombre completo del usuario estudiante.
- **Subtotal**: precio del pedido antes de realizar los descuentos otorgados por la condición de becado del estudiante (en caso de que así sea).
- **Total**: precio final del pedido abonado por el usuario luego de haber realizado los descuentos correspondientes.
- **Estado**: estado actual del pedido.

Además, cada fila en la sección derecha contiene un primer botón para acceder al detalle del pedido y un segundo botón destinado a gestionar las entregas de los mismos.

**Detalles de un pedido**

Luego de que el usuario sede presione sobre el botón con un ícono “+”, el sistema mostrará un modal con información detallada del pedido.


**Figura 50**. Detalles de un pedido para el usuario sede

El título principal indica el número identificatorio del pedido precedido por un numeral. Debajo del mismo, se indica el estado general en el que se encuentra el mismo (en la imagen anterior se observa que el pedido está en estado “Solicitado”). Por último, el modal contiene un listado desplegable con los archivos y anillados pertinentes. El header de cada uno de estos ítems indica el estado actual asociado (“Por imprimir” o “Impreso” en caso de archivos y “Por anillar” o “Anillado” en caso de los grupos de anillado) y el nombre del mismo.

Para desplegar la información de un ítem particular, el usuario debe presionar sobre la flecha con sentido descendente ubicada en la esquina derecha o bien sobre el propio header.

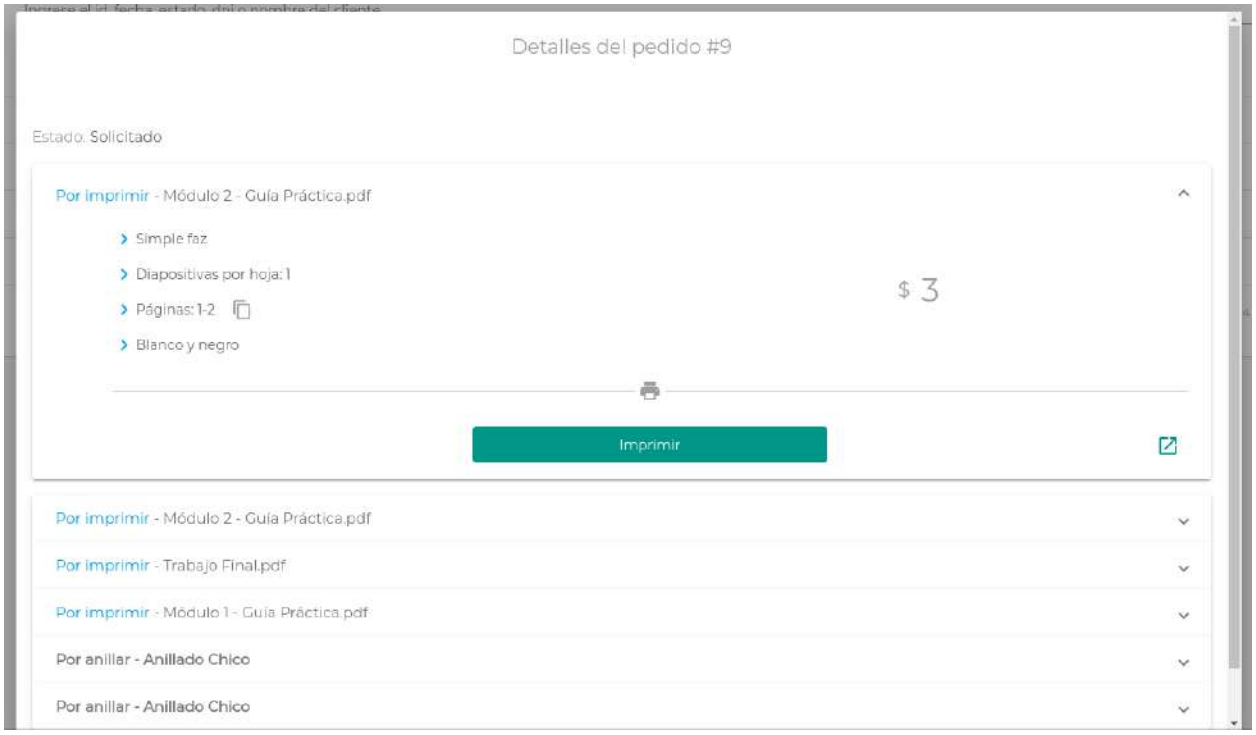



**Figura 51**. Detalles de un archivo y panel de acciones que visualiza el usuario sede

En caso de los archivos, se mostrará un ítem independiente por cada copia solicitada y reflejará la siguiente información respectivamente:

- Si el archivo fue configurado con un formato de doble o simple faz.
- Cantidad de páginas por hoja.
- Rango de páginas a imprimir. El usuario puede hacer click sobre el ícono que se encuentra a la derecha del valor para proceder a copiar al portapapeles el rango indicado (acción que facilitará el ingreso de datos al momento de la impresión del archivo).
- Si la impresión debe ser a color o en blanco y negro.
- Monto asociado al archivo (ubicado en el sector derecho).

En la sección inferior se encuentra el panel de acción asociado al archivo. En el mismo, es posible previsualizar (e imprimir) el archivo en el lector de PDF integrado en el navegador al hacer click sobre el botón que se encuentra en el sector derecho.

Además, en esta sección es posible cambiar el estado del ítem a impreso al presionar el botón “Imprimir”.

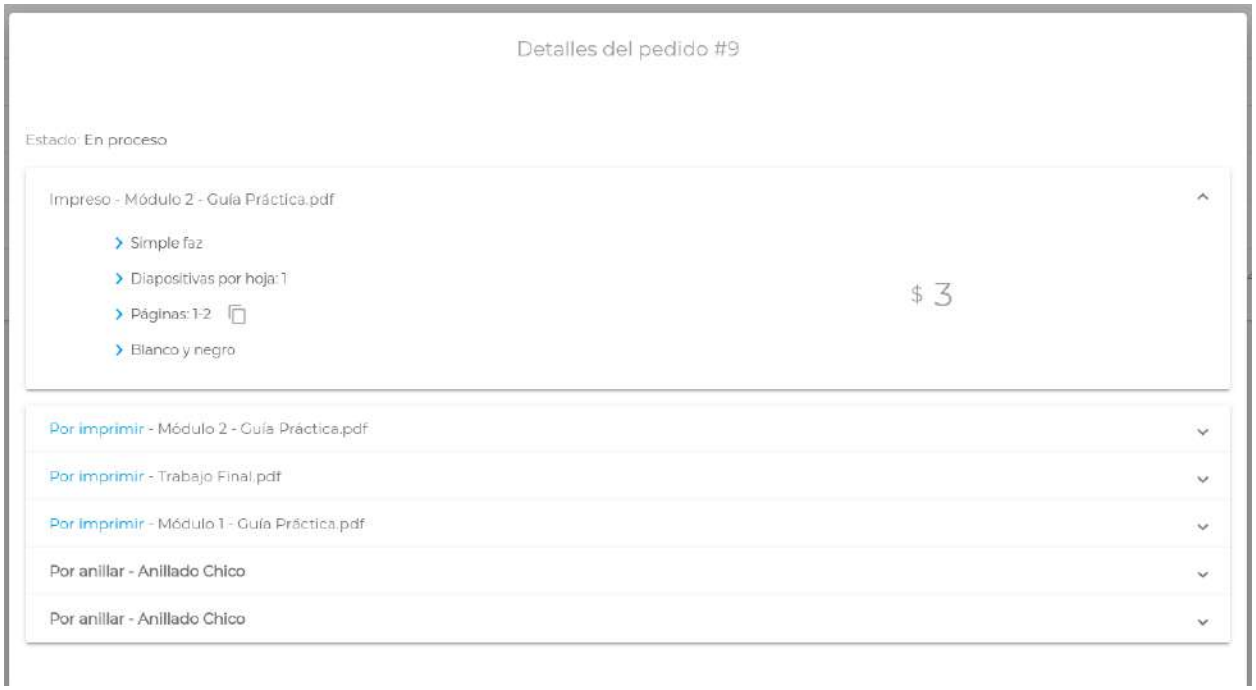



**Figura 52**. Visualización del detalle de un archivo impreso

Luego de esta acción se modifica el estado asociado al archivo a “Impreso” y se oculta el panel de acción.

***Nota:*** *luego de que se modifica el estado de al menos un archivo a “Impreso”, el estado general del pedido, que antes se encontraba en estado “Solicitado”, se modifica automáticamente al estado “En proceso” (estado visible en tiempo real por el usuario estudiante).*

En el caso de los anillados, luego de desplegar el ítem asociado, se muestra la siguiente información:


**Figura 53**. Detalles de un ítem de tipo Anillado

En el sector izquierdo se detallan los archivos que pertenecen al grupo de anillado y su orden correspondiente, mientras que en el sector derecho se indica el monto asociado.

Por último, en el sector inferior se muestra un panel de acción en el que es posible cambiar el estado del anillado desde “Por anillar” a “Anillado” al presionar el botón “Anillar”.

Ésta acción es posible únicamente si todos los archivos asociados al grupo de anillado se encuentran en estado “Impreso”. Caso contrario, el sistema indicará el siguiente error:

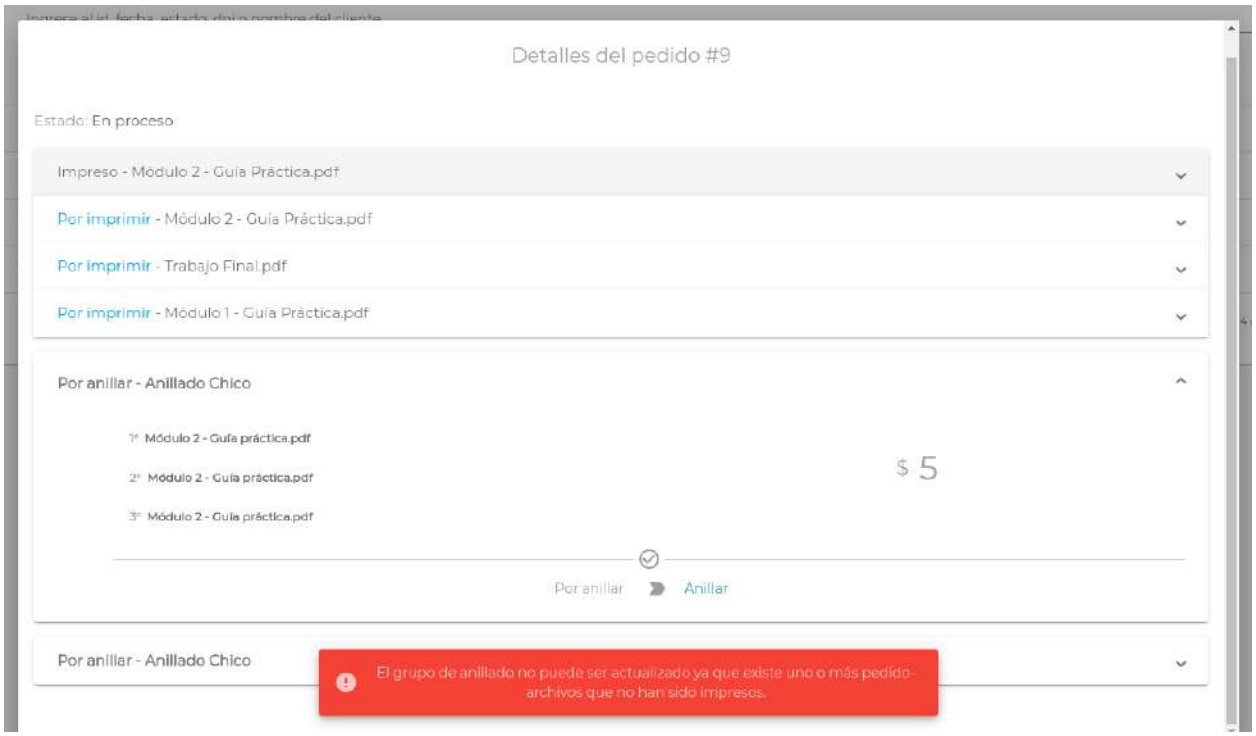



**Figura 54**. Error asociado a anillar un grupo de archivos que no han sido impresos

En caso de que todos los archivos asociados al grupo de anillado se encuentren impresos y luego de que el usuario presione el botón “Anillar”, el sistema modificará el estado del ítem a “Anillado”.


**Figura 55**. Visualización de un ítem en estado “Anillado”

***Nota:*** *una vez que se finalicen todas las acciones asociadas al pedido en cuestión, el sistema cambiará automáticamente el estado global del pedido a “Listo para retirar”. En el ejemplo asociado a la anterior captura de pantalla, esto se produce luego de imprimir el archivo “Módulo 1 - Guía Práctica.pdf” y anillar el último ítem “Anillado Chico”.*


**Figura 56**. Visualización del cambio automático del estado global de un pedido

*Cada cambio de estado, como se mencionó anteriormente, se verá reflejado también en la tabla principal.*

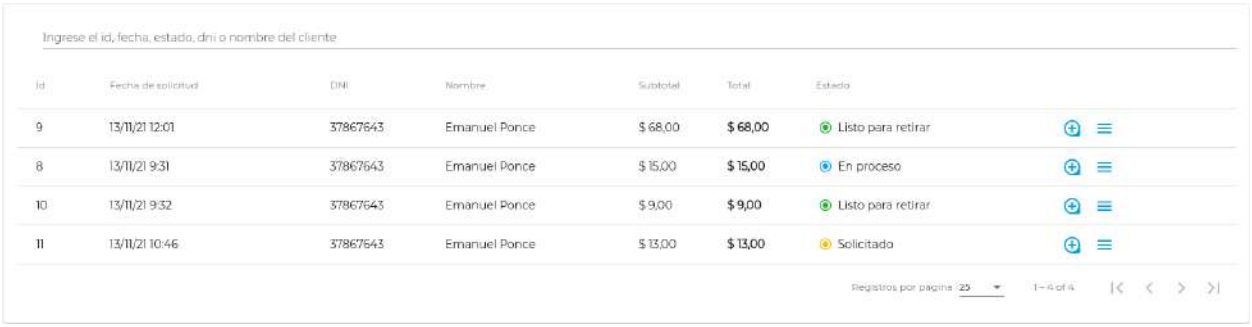



**Figura 57**. Listado de archivos cuyo estado se actualizó automáticamente

### Gestión de entrega de un pedido (Pedidos activos)

El usuario sede tiene la facultad de modificar manualmente un pedido a uno de los siguientes estados finales: “Entregado”, “No entregado” y “Cancelado”. Para realizar estas acciones el usuario debe presionar sobre el botón que se encuentra en la sección derecha asociada al pedido que se desea modificar, lo que mostrará un submenú con las acciones disponibles según el estado actual del pedido.

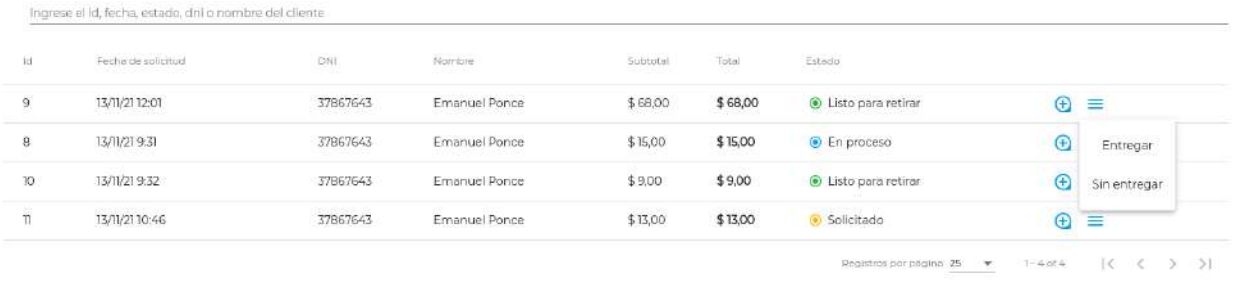



**Figura 58**. Acciones posibles de un pedido “Listo para retirar”

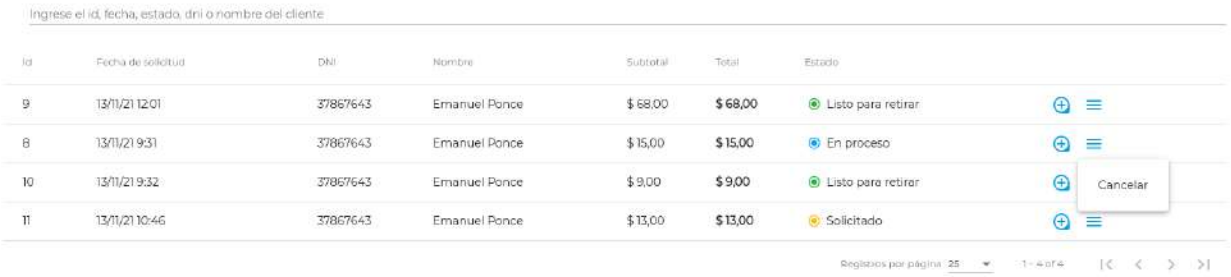



**Figura 59**. Acciones posibles de un pedido “En proceso”

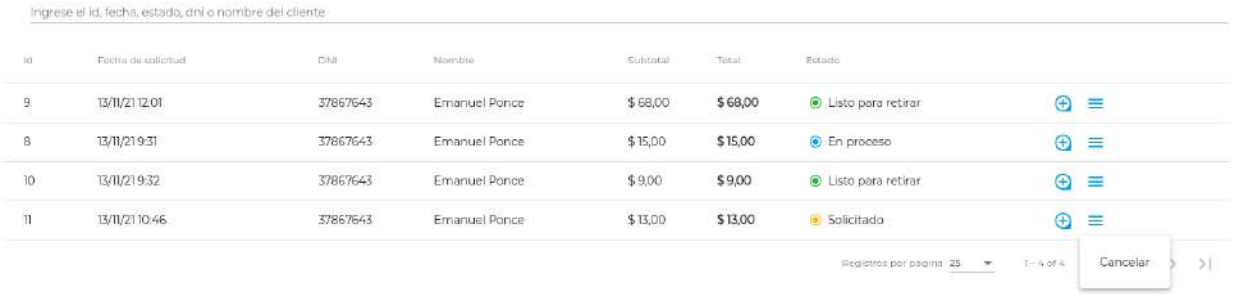



**Figura 60**. Acciones posibles de un pedido “Solicitado”

Como se observa en las 3 imágenes anteriores, un pedido en estado “Listo para retirar” puede pasar al estado “Entregado” (para ello el usuario debe seleccionar la opción “Entregar”) o “No entregado” (si el usuario selecciona la opción “No entregar”). Por otro lado, para los estados “En proceso” y “Solicitado” el usuario puede modificar el estado del pedido a “Cancelado” si presiona sobre el botón “Cancelar”. Es importante aclarar que el monto abonado será reincorporado al usuario solicitante únicamente si se cancela un pedido en estado “Solicitado” (si se cancela un pedido en estado “En proceso” el monto no será restablecido al estudiante).

Al seleccionar una de las tres acciones, el pedido automáticamente se oculta de este listado y se incorpora en el listado asociado a la vista “Pedidos históricos”.

**Gestión de pedidos finalizados (Pedidos históricos)**

El sistema otorga una vista dedicada a visualizar aquellos pedidos que han sido finalizados, es decir aquellos que se encuentren en estado “Entregado”, “No entregado” o “Cancelado”. Para ello, el usuario de tipo sede debe dirigirse a la página “Pedidos históricos”.

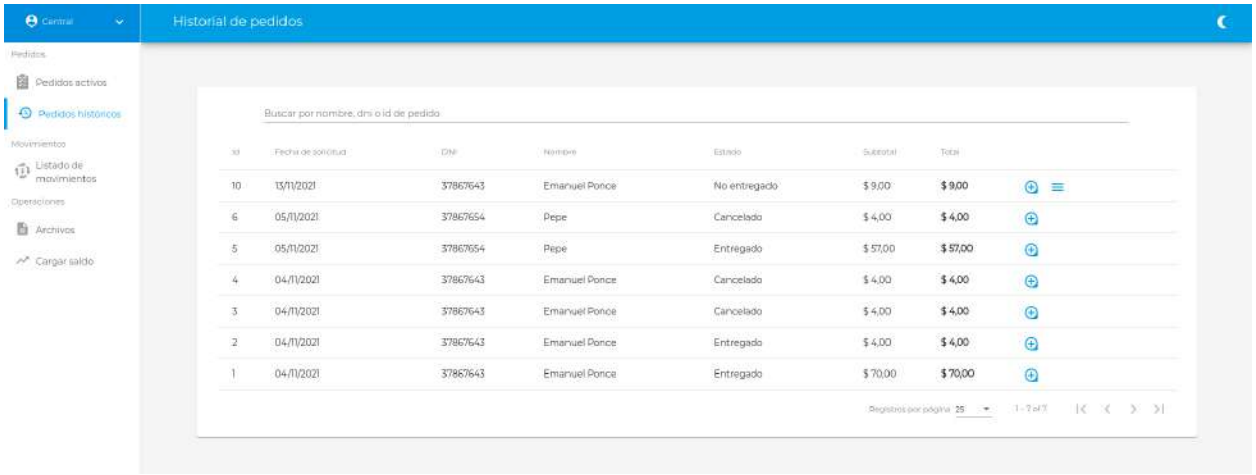



**Figura 61**. Listado de pedidos históricos visualizados por el usuario sede

Esta pantalla muestra una tabla con los pedidos finalizados asociados al usuario sede que se encuentra en la sesión.

En la sección superior se muestra un campo para que el usuario filtre los pedidos históricos por el número identificatorio del pedido o bien por nombre, apellido o DNI del estudiante solicitante.

La tabla de pedidos muestra las siguientes columnas:

- **Id**: valor numérico que identifica unívocamente al pedido.
- **Fecha de solicitud**: fecha y hora de solicitud del pedido.
- **DNI**: número del Documento Nacional de Identidad del estudiante que solicitó el pedido.
- **Nombre**: nombre completo del usuario estudiante.
- **Estado**: estado de finalización del pedido.
- **Subtotal**: precio del pedido antes de realizar los descuentos otorgados por la condición de becado del estudiante (en caso de que así sea).
- **Total**: precio final del pedido abonado por el usuario luego de haber realizado los descuentos correspondientes.

Además, cada fila en la sección derecha contiene un botón con un ícono “+" para acceder al detalle del pedido (redirigirá a la misma pantalla que la vinculada a los pedidos activos). Por último, aquellos pedidos que se encuentren en estado “No entregado” poseen un botón adicional (con tres líneas horizontales), cuya acción abrirá un submenú con la opción “Entregar”.

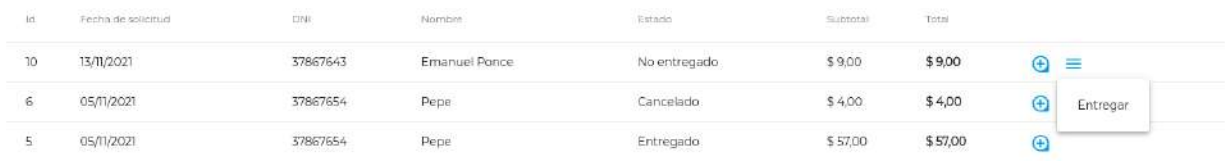


**Figura 62**. Acción posible de un pedido “No entregado”

En caso de que el usuario efectivamente presione sobre dicha opción, el sistema lanzará un modal de confirmación que el usuario deberá aceptar (presionando sobre el botón “Continuar”) para luego modificar el pedido, pasando del estado “No entregado” a “Entregado”.


**Figura 63**. Modal de confirmación de cambio de estado de un pedido


**Figura 64**. Visualización del cambio de estado del pedido asociado

# Movimientos

## Listado de movimientos del estudiante

Todo estudiante podrá acceder al listado de sus movimientos en la plataforma a través del menú lateral, haciendo click en el apartado “Mis movimientos”.

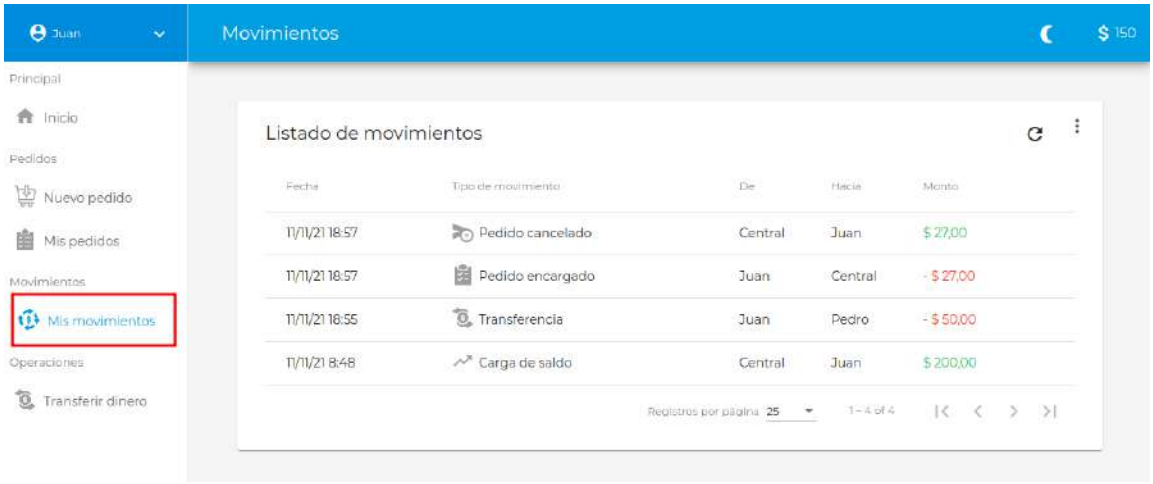



**Figura 65**. Listado de movimientos del estudiante

En esta pantalla el estudiante podrá acceder a todo el historial de movimientos, ordenados de más reciente a más antiguo. A fin de facilitar la búsqueda se provee un filtro accesible a través del botón de la esquina superior derecha del listado que permite acceder a movimientos específicos.

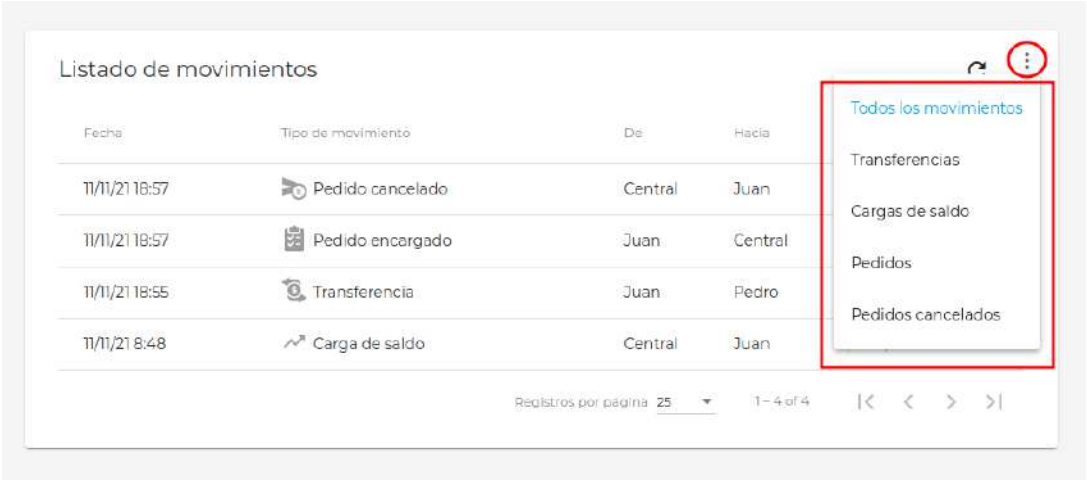



**Figura 66**. Filtrado de movimientos

A través de la selección de cualquiera de sus opciones se accede a una vista refrescada del listado exclusiva para dicha categoría. A modo de ejemplo, para visualizar todos los pedidos encargados se puede elegir “Pedidos” y obtener los mismos:

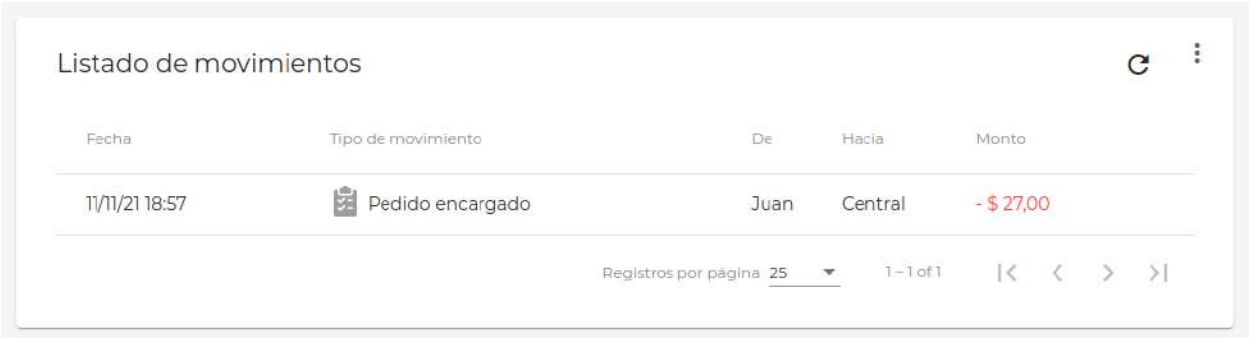



**Figura 67.** Listado de movimientos relacionados con encargos de pedidos

Estos pedidos representan un débito en la cuenta del alumno y por lo tanto son tenidos en cuenta como un movimiento. En resumen, las alternativas disponibles son:

- Pedidos encargados (débitos)
- Transferencias (entrantes o salientes)
- Cargas de saldos (créditos)
- Cancelaciones de pedidos (créditos)

## Listado de movimientos de la sede

Para el caso del usuario “sede” se tendrá un listado muy similar al del estudiante, accesible ahora desde el botón “Listado de movimientos” del menú lateral.

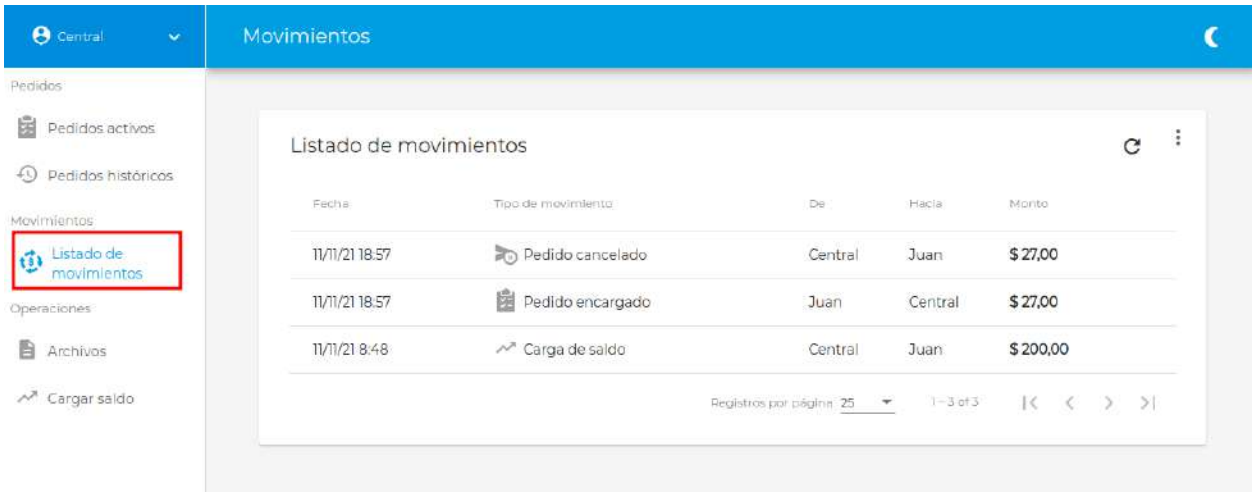



**Figura 68**. Listado de movimientos de la sede

El listado al cual accederá permitirá conocer los movimientos producidos por el usuario “sede” que haya iniciado sesión. Nuevamente, haciendo click sobre el botón de la esquina superior derecha de la tabla podrá acceder al filtro de movimientos.

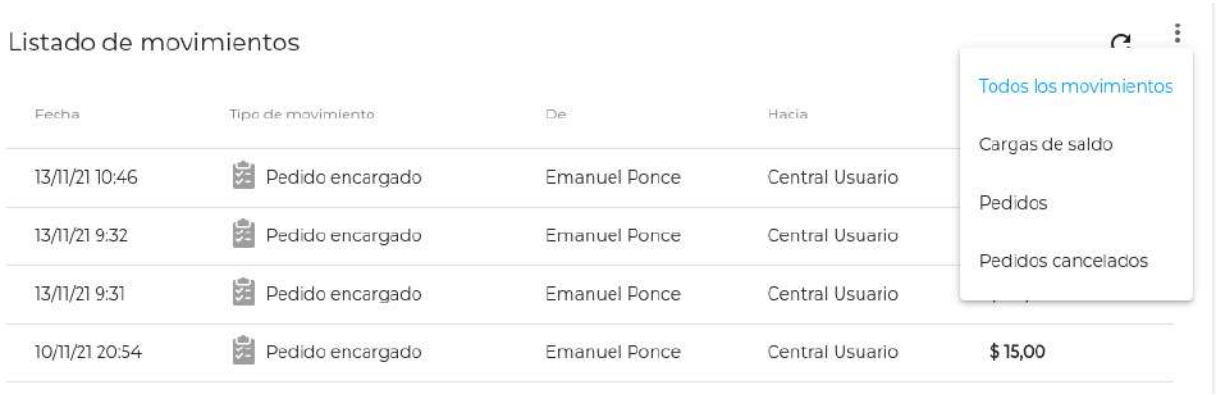



**Figura 69.** Filtrado de movimientos

De la misma forma que sucede para el estudiante, una vez seleccionada una opción, el listado será refrescado mostrando solo los movimientos pertinentes.

A diferencia del estudiante, la sede podrá visualizar los siguientes tipos de movimientos:

- Pedidos encargados
- Cargas de saldos
- Cancelaciones de pedidos

# Transferencia de saldo

La transferencia de saldo es una función disponible para estudiantes y becados. Estos usuarios podrán acceder al apartado de “Operaciones” en el menú lateral en la opción “Transferir dinero”.

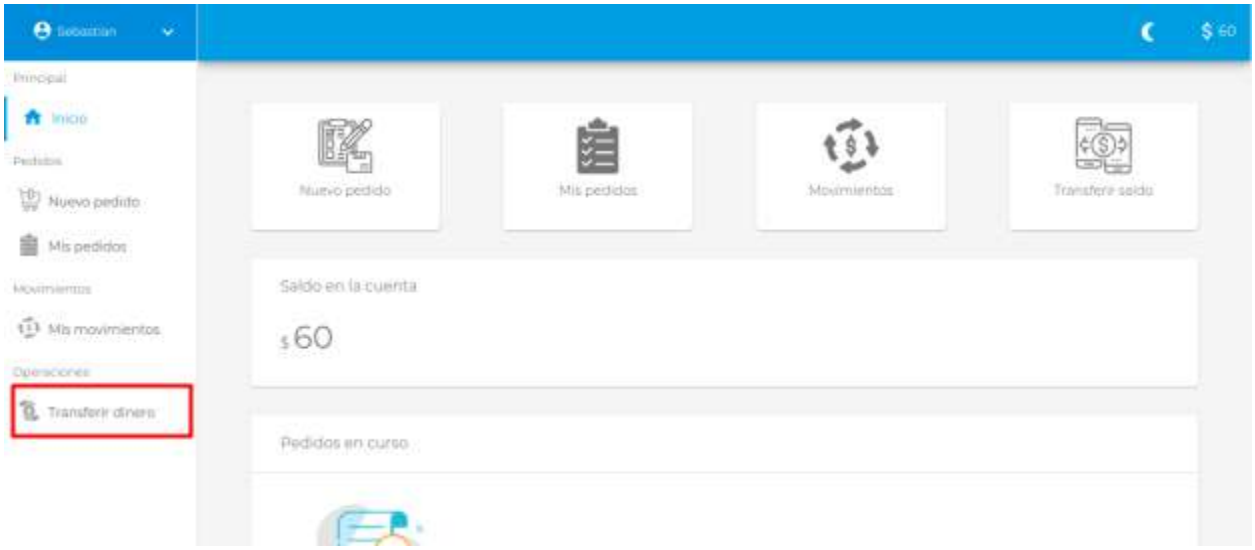



**Figura 70**. Opción de Transferir dinero en menú lateral

En esta pantalla, el usuario encontrará un listado de todos los estudiantes (con sus datos) que pertenecen al sistema, además de una barra de búsqueda para buscar un usuario en particular ya sea por su DNI, nombre o email.

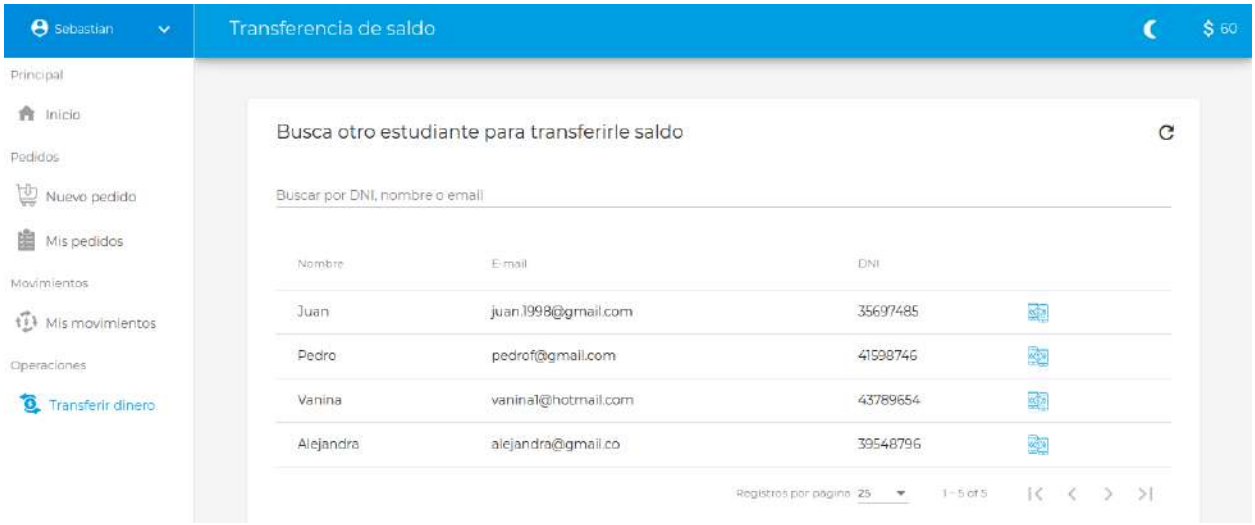



**Figura 71**. Pantalla de transferencia de saldo, con listado de estudiantes y barra de búsqueda

***Nota**: la búsqueda por nombre y apellido estará dada por cómo el usuario se haya denominado a sí mismo en la plataforma.*

Una vez que el estudiante logra identificar al usuario al cual le realizará la transferencia del dinero, deberá hacer click sobre el botón que se encuentra al final de la fila perteneciente al usuario en cuestión.

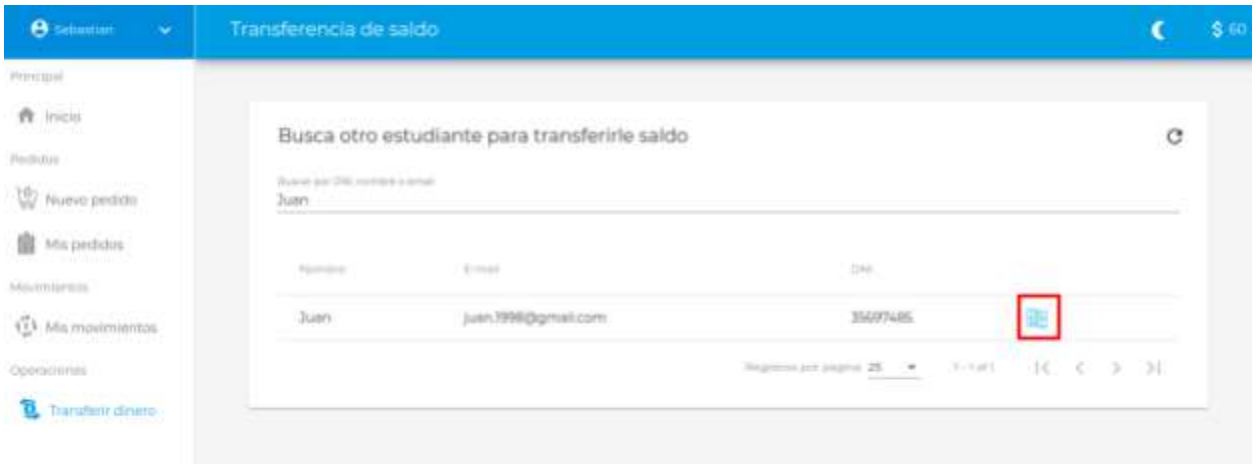



**Figura 72.** Botón para accionar la pantalla de transferencia de saldo al usuario deseado

Una vez realizada esta acción, se desplegará una segunda pantalla, en la cual deberá ingresar el monto total a transferir. Luego, presionará el botón “Transferir”.

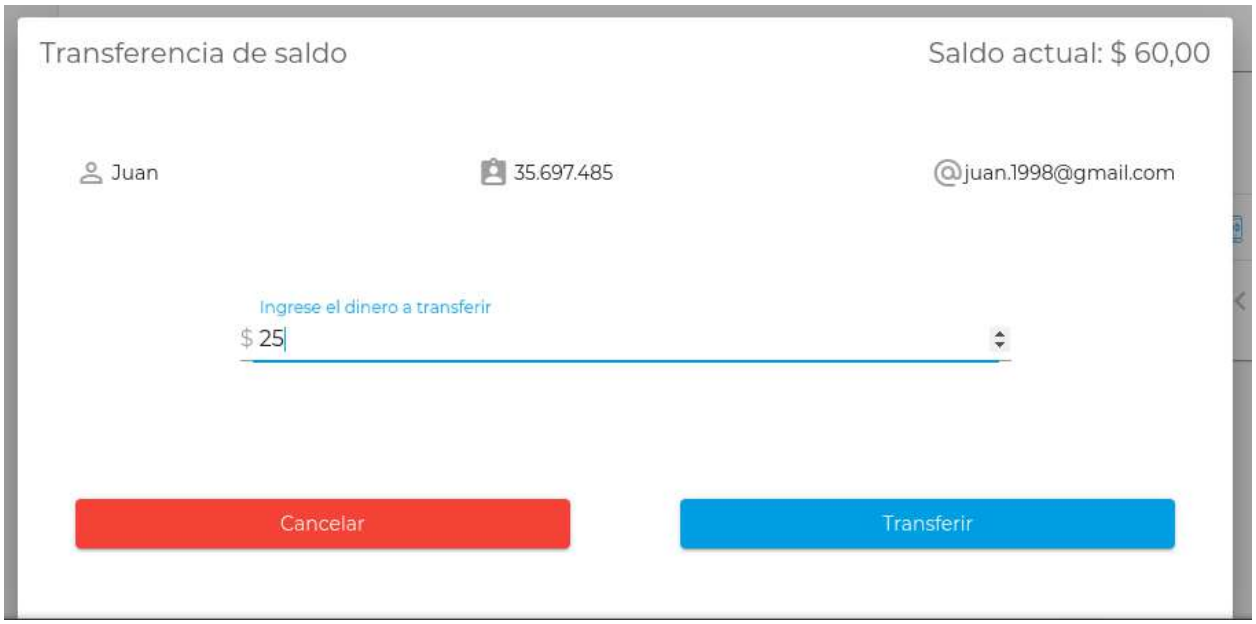



**Figura 73**. Modal de transferencia de saldo

***Nota**: Es importante recalcar que para realizar transferencias de saldo se debe disponer del dinero en cuenta que se desee transferir, caso contrario se desplegará un error indicando que no es posible transferir más dinero del que actualmente se posee en la billetera:*

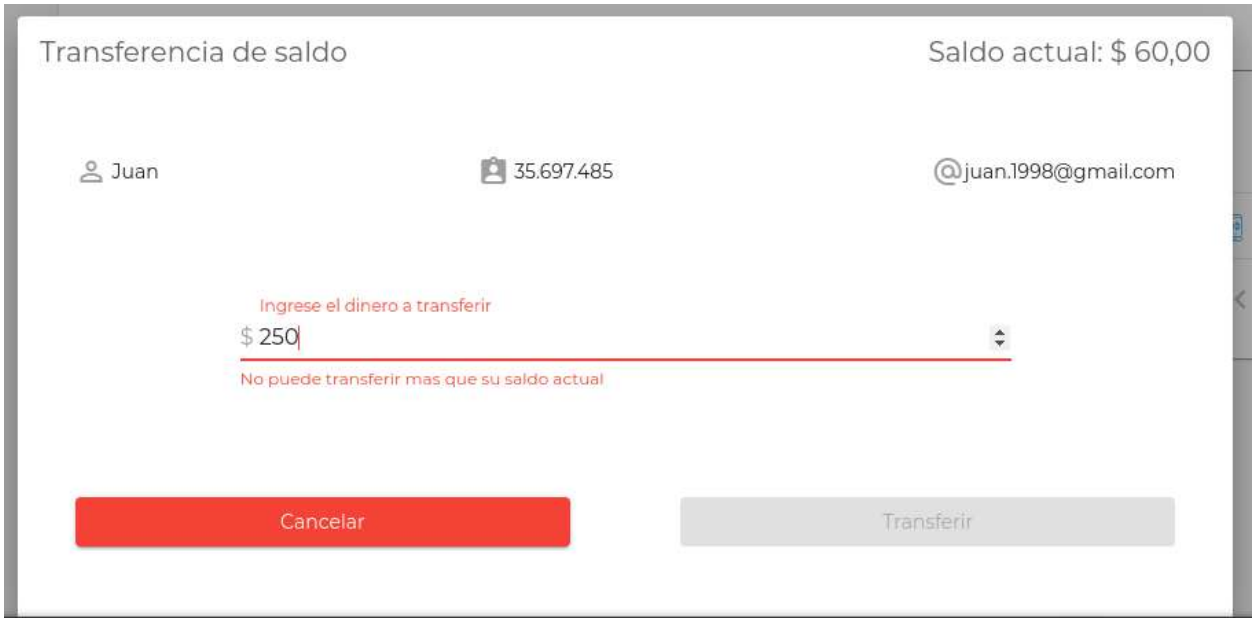



**Figura 74.** Error al intentar transferir una cantidad mayor de dinero del que posee el estudiante

En el flujo idóneo, al presionar “Transferir”, se abre una ventana de confirmación, en donde se desplegarán los datos de la transacción, tanto usuario destino como el saldo que se transfiere. Si se rechaza la operación se volverá a la pantalla anterior, y si se confirma se ejecuta el movimiento.


**Figura 75.** Ventana de confirmación de movimiento de transferencia de saldo

Una vez confirmada la transacción y luego de su ejecución, hará aparición una ventana que indica que el movimiento se realizó exitosamente.


**Figura 76.** Ventana de confirmación exitosa de transferencia

# Carga de saldo

La funcionalidad de carga de saldo estará habilitada para el rol “sede”. Los usuarios de este tipo podrán acceder a la pantalla pertinente desde el menú lateral clickeando en la opción “Cargar saldo”.

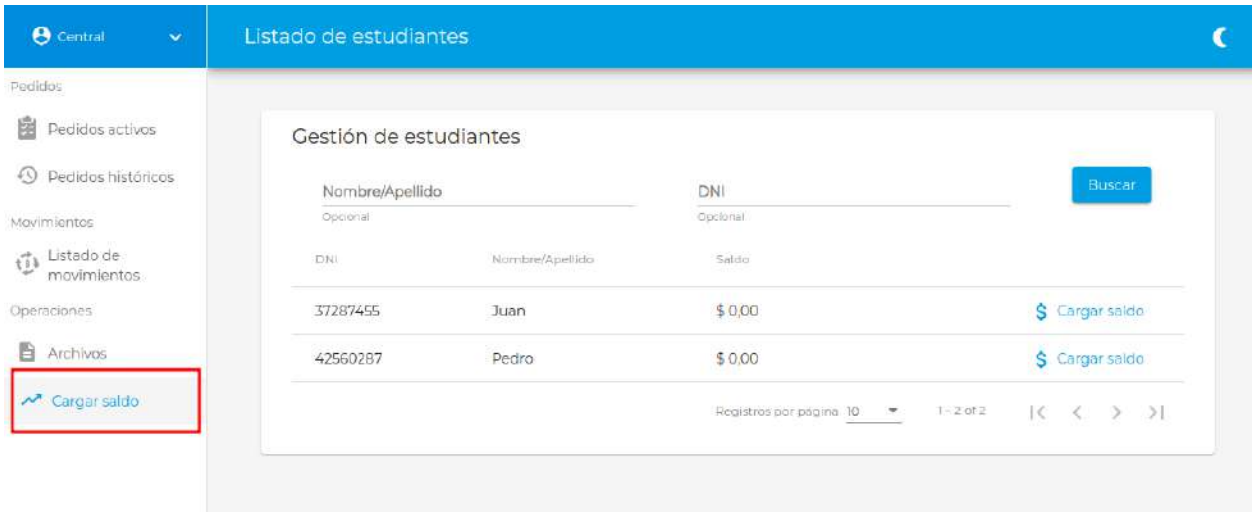



**Figura 77**. Pantalla para carga de saldo

Una vez ingresada a la misma, el usuario se encontrará con un listado de estudiantes a los cuales se les puede otorgar el crédito de dinero. A fin de facilitar la búsqueda del alumno deseado, se provee un

filtro en la parte superior que podrá combinar una búsqueda por nombre y apellido (como un único campo) y DNI. De esta manera se facilita en gran medida su identificación.

***Nota**: la búsqueda por nombre y apellido estará dada por cómo el usuario se haya denominado a sí mismo en la plataforma.*


**Figura 78**. Búsqueda de estudiante por nombre y DNI

Una vez encontrado y pactado con el estudiante cuál será el dinero a acreditar se debe ingresar al botón “Cargar saldo” y elegir el monto adecuado.


**Figura 79**. Carga de saldo a estudiante

Una vez chequeada la operación a realizar y ejecutada la misma, el estudiante gozará del dinero acreditado y el resultado podrá visualizarse desde el propio listado dado que el saldo se verá incrementado.

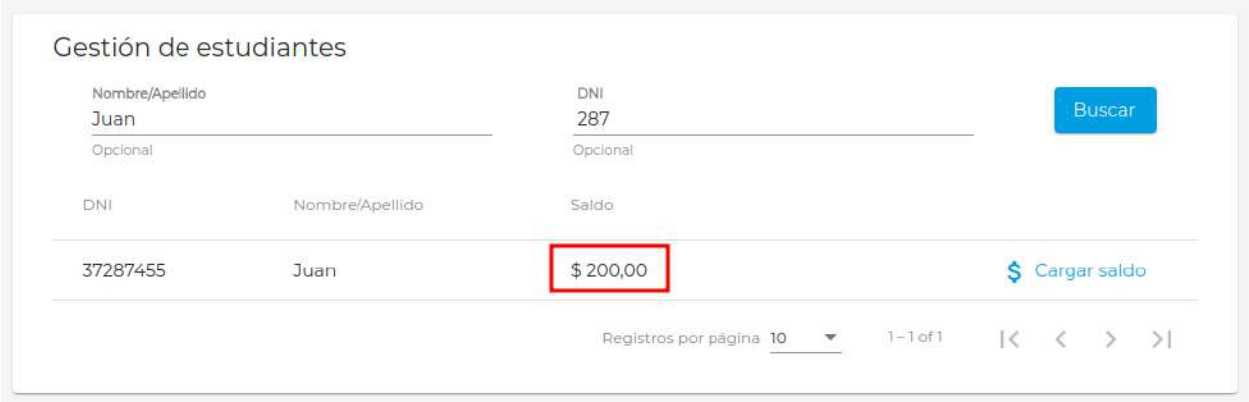



**Figura 80**. Saldo acreditado para el estudiante

# Gestión de archivos

La administración de los archivos que estarán disponibles a los estudiantes se realiza a través de los roles de administrador, sede o cátedra. Cada uno de ellos tiene un menú que les permite acceder a la pantalla de gestión mencionada.

El sistema reconoce sólo dos tipos de archivos:

- Archivos pertenecientes al staff administrativo (admin o sede)
- Archivos subidos por las cátedras

En este esquema, tanto los administradores como los usuarios de tipo sede tienen **control total** sobre **todos** los archivos del sistema.

## Admin/Sede

Dado que las pantallas para estos dos roles son similares, el funcionamiento a describir aplica para ambos. Cuando el usuario ingresa al apartado pertinente se encontrará con un listado de todas las materias existentes en el sistema.

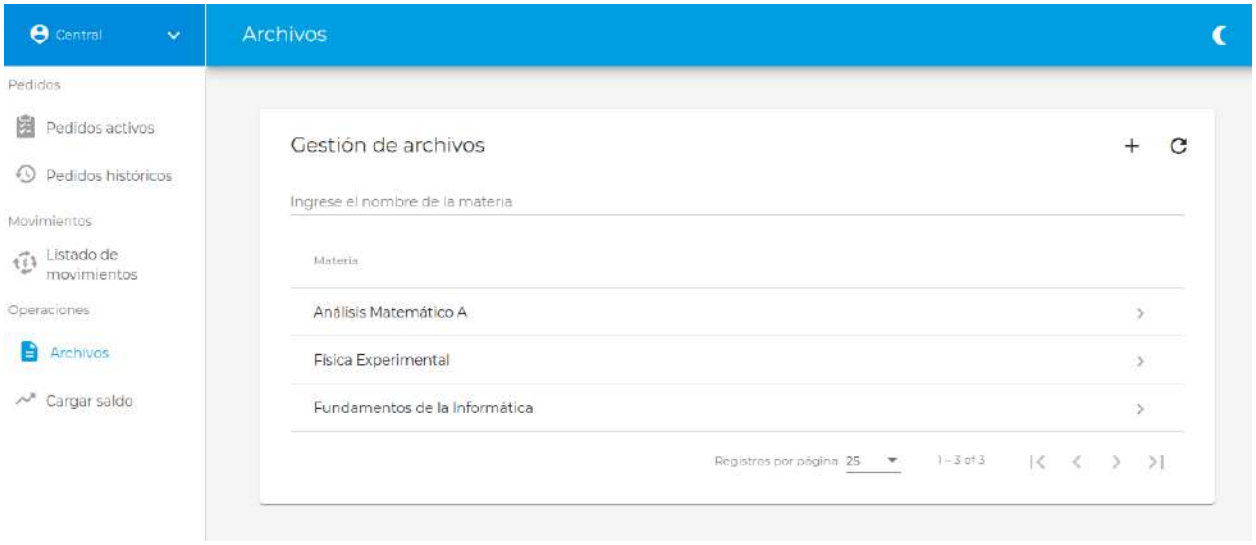



**Figura 81**. Listado de materias disponibles para gestionar sus archivos

El usuario tendrá dos opciones aquí: proceder con la creación de un nuevo archivo o editar alguno de los ya existentes.

La creación de un nuevo archivo será a través del botón “+” en la esquina superior derecha del listado. Una vez ingresada a la pantalla a la cual es redirigido, el usuario deberá seleccionar todas aquellas materias a las cuales desea vincular con él o los archivos por subir. Una vez definidas las mismas y cargados los archivos pertinentes se podrá proceder a la creación de los mismos.

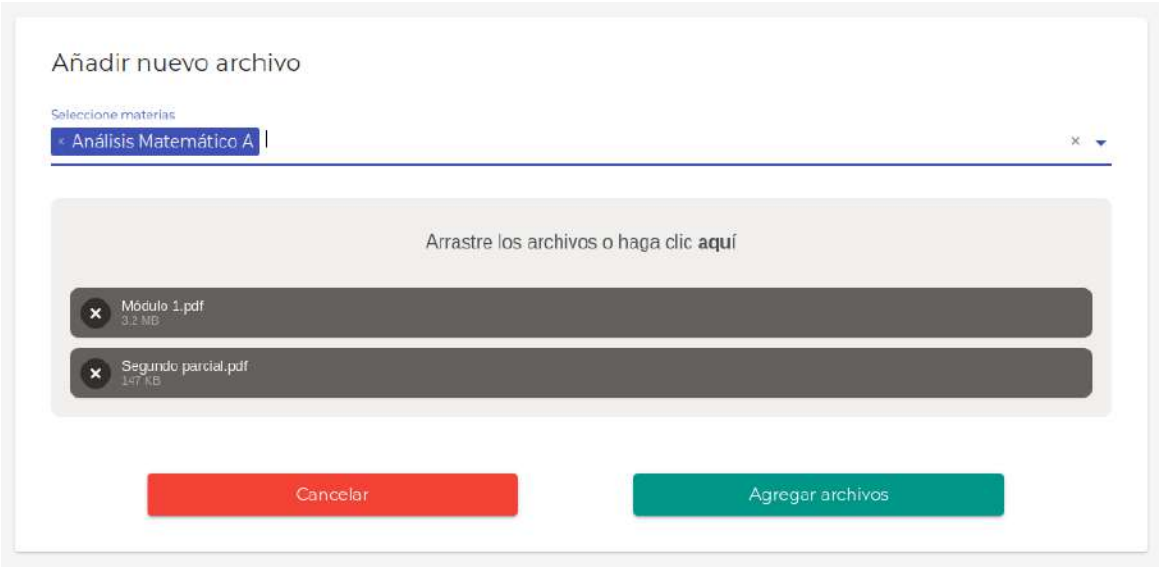



**Figura 82**. Creación de archivos para la materia Análisis Matemático A

En caso que se hubiera deseado vincular ambos archivos con una segunda materia, la misma podría haber sido también seleccionada desde el primer campo del formulario.

La gestión de archivos existentes se realiza una vez que se ingresa a una de las materias presentadas en la figura 81. Una vez dentro de la misma es posible observar los archivos que están asociados a la asignatura y quién es el dueño de cada uno de ellos.

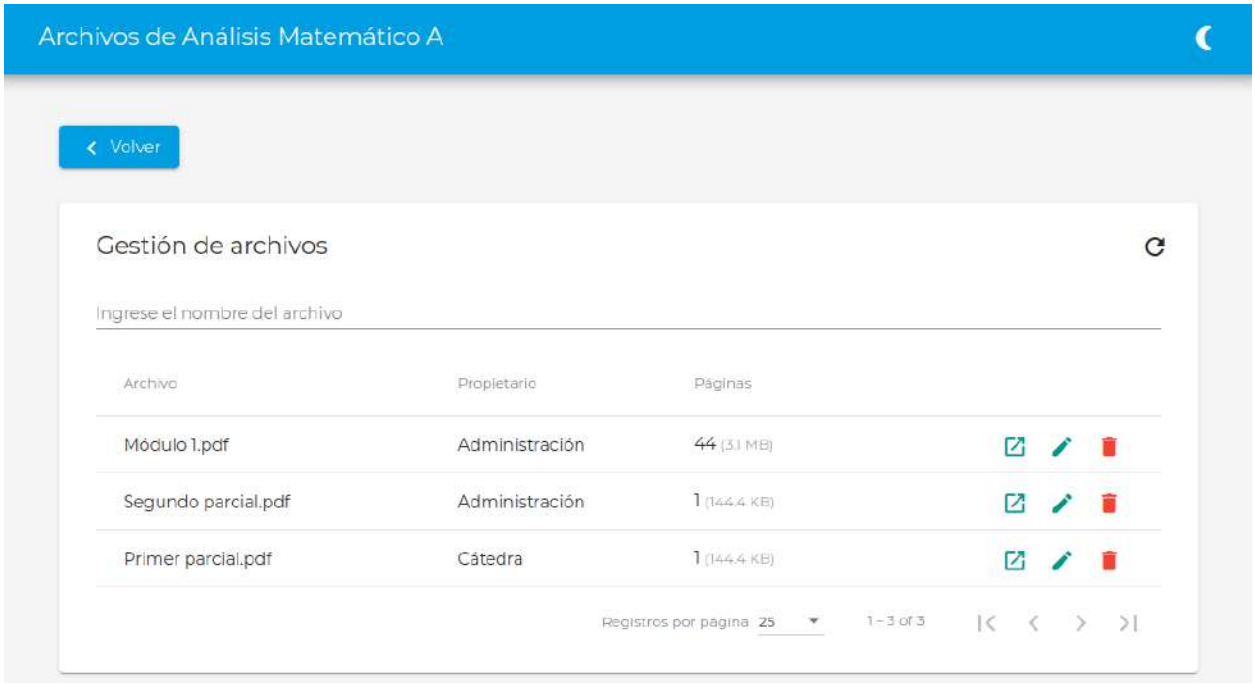



**Figura 83**. Listado de archivos disponibles para la asignatura Análisis Matemático A

Dado que en la figura anterior se encuentra logueado como un usuario “sede”, se tiene total control sobre los archivos, pudiendo editar para cada uno de ellos su nombre o incluso eliminarlos del sistema.

Respecto a la eliminación se debe aclarar que una vez confirmada la eliminación de un archivo, el mismo será borrado lógicamente para todas las materias con que se encuentra vinculado. De esta manera, no estará más disponible para ser seleccionado desde la sección de “Nuevo pedido”. Sin embargo, el contenido del mismo seguirá existiendo en el servidor hasta tanto se detecte que no exista ningún pedido más en proceso que lo utilice. Esto incluye tanto pedidos solicitados como en curso, es decir, que no han sido entregados.

## Cátedra

Para el caso del usuario “cátedra” se tiene una dinámica muy similar a lo descrito anteriormente con la excepción que la misma no es presentada con el listado de materias completo. En lugar de ello se muestran directamente los archivos que la misma tiene cargados en el sistema asociados a la materia a la cual está vinculado su usuario.

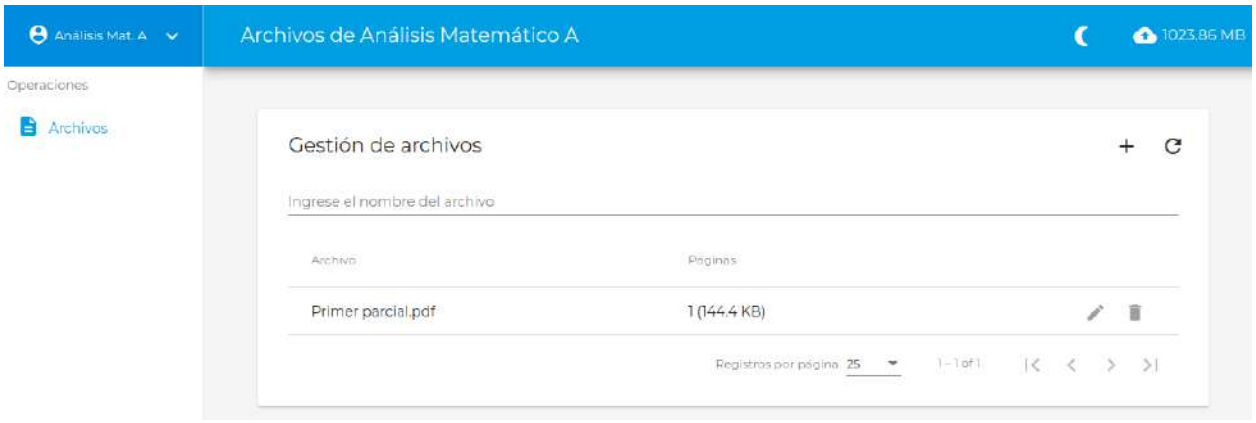



**Figura 84**. Listado de archivos de la cátedra de Análisis Matemático A

La carga de un nuevo archivo sólo permite seleccionar los recursos a cargar, quedando los mismos vinculados con la asignatura que esté vinculada al usuario actual.


**Figura 85**. Carga de archivos para el usuario “cátedra”

Finalmente, de forma análoga a lo expuesto para los otros roles, aquí también es posible modificar el nombre del archivo o eliminarlo del sistema, originando el mismo comportamiento de disponibilidad del recurso que se mencionó en la sección anterior (es decir, que si se borra seguirá disponible físicamente hasta tanto no haya pedidos en proceso).

# Portal de administración

Los administradores del sistema podrán acceder a todo un portal exclusivo para su rol que les permitirá gestionar usuarios, materias, carreras y otras entidades de la plataforma. Cada una de ellas tendrá un apartado exclusivo accesible a través del menú lateral.

## Gestión de usuarios

La pantalla más importante de este portal es la que permite conocer y manipular los diferentes usuarios que tendrán acceso a la plataforma. Dado que se mencionó al comienzo de este manual que existen varios roles que participan en el sistema, se ha decidido segmentar la gestión de los mismos para algunos casos en específico.

### Visualización de usuarios

Al acceder al menú correspondiente, el administrador es presentado con un listado genérico de usuarios registrados en el sistema.

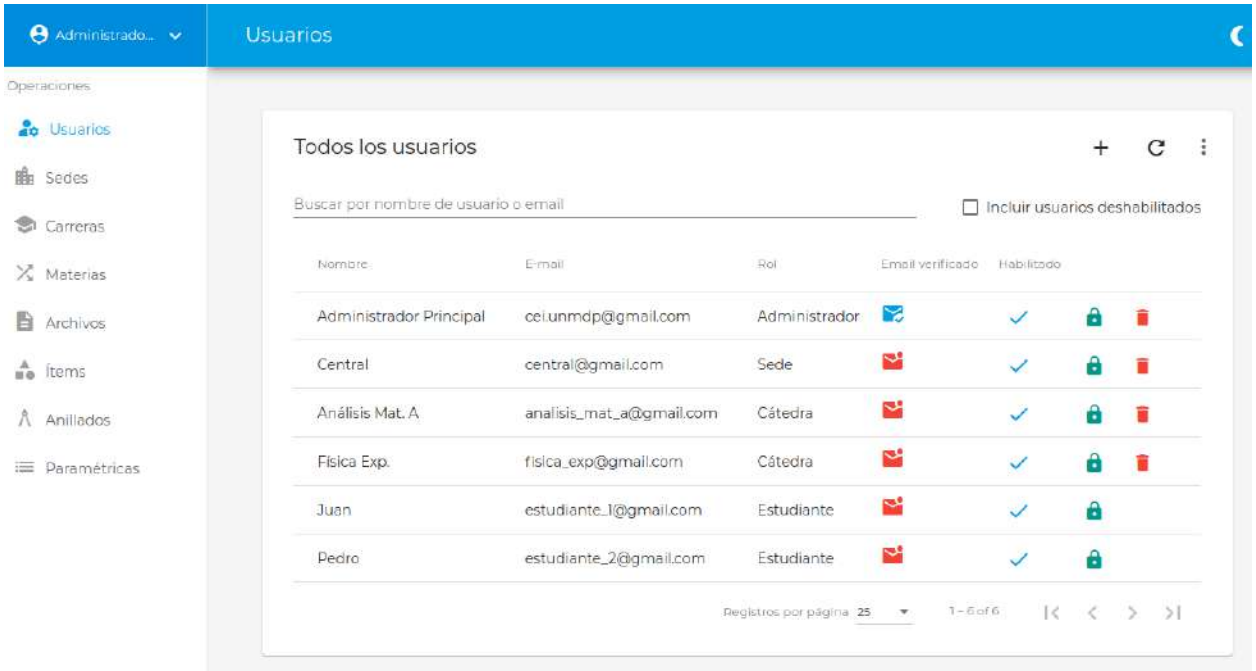



**Figura 86**. Listado de usuarios genérico

Tal como puede observarse en la figura anterior, existen en esta lista tanto usuarios estudiantes como cátedras, sede o incluso administradores. A través del mismo listado es posible conocer si han o no verificado su email, deshabilitar o habilitar usuarios para que puedan acceder al sistema, resetear su contraseña en caso de extravío o incluso eliminar algunos tipos de usuarios en específico.

Todas estas acciones son posibles a través de los botones que se encuentran en el mismo listado.


**Figura 87**. Botones para ejecutar algunas funciones básicas sobre los usuarios

Cabe mencionar también que los usuarios visibles tanto en esta vista como en las filtradas son aquellos habilitados en la plataforma. En caso de desear ver también los usuarios deshabilitados se debe tildar el checkbox superior al listado.

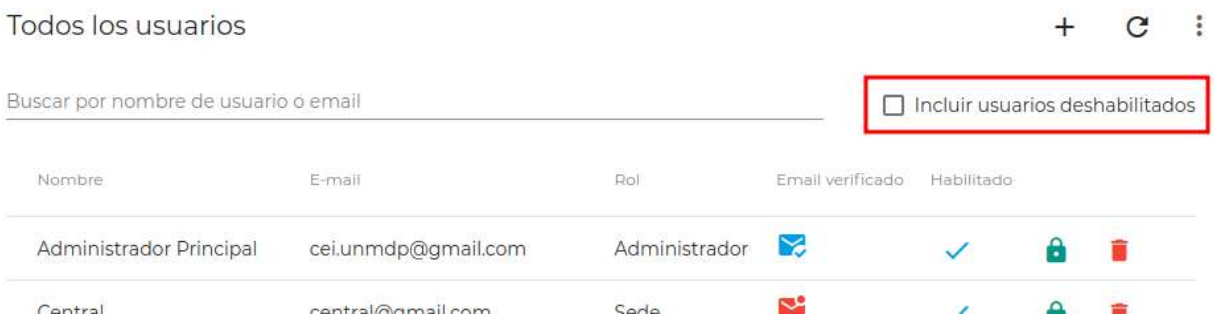



**Figura 88**. Inclusión de usuarios deshabilitados

Dado que cada rol de usuario tiene atributos particulares, se decidió filtrar por dicho rol antes de acceder a las funcionalidades específicas que posee cada uno de ellos. Esto es posible a través del desplegable que se encuentra en la esquina superior derecha del listado.

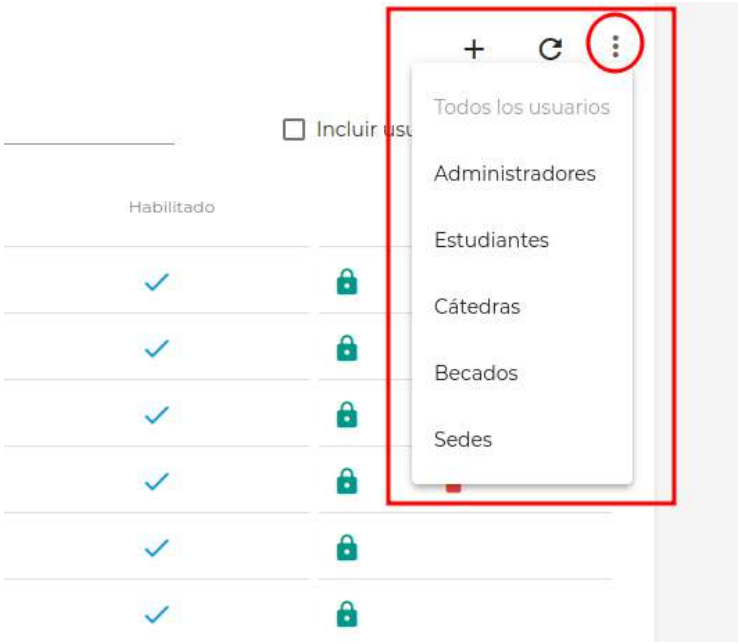



**Figura 89**. Filtrado de usuarios por rol

Una vez aplicado el filtro, el listado es refrescado de forma tal de sólo mostrar los usuarios correspondientes. En esta nueva vista aparece ahora la posibilidad de editar al usuario deseado, a través del botón correspondiente. Esto se cumple para todos los roles.

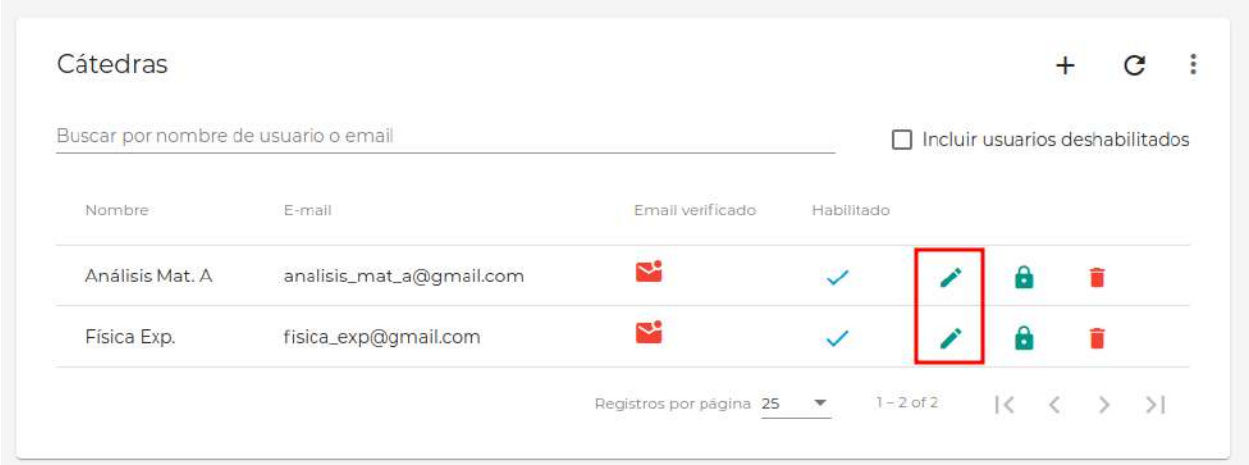



**Figura 90**. Filtrado de usuarios por el rol “cátedra” con capacidad de edición ahora

### Creación de usuarios

La creación de usuarios se realiza accediendo al botón con el signo “+”, el cual será una constante para las demás funcionalidades accesibles del menú. Al hacer click sobre el botón será desplegado un pequeño menú sobre el cual indicar qué tipo de usuario se desea crear en el sistema. Las posibilidades incluyen todos los roles a excepción del estudiante y becado, los cuales tienen un ciclo de vida diferente (su registración se realiza desde la pantalla de acceso al sistema).

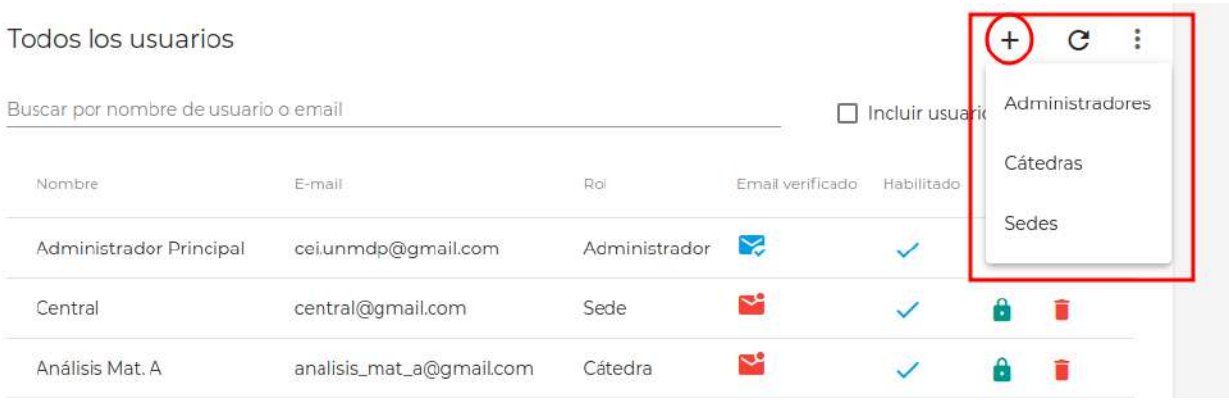



**Figura 91**. Submenú para creación de usuarios

A través de la selección de alguno de los roles habilitados será redirigido a la pantalla pertinente en la cual serán solicitados los datos específicos que se necesiten. Una vez proporcionados se podrá proseguir con la creación del usuario y, en caso de no generarse ningún error, el administrador será redirigido nuevamente al listado de usuarios.

Para el caso de un nuevo usuario de tipo sede, por ejemplo, la pantalla se ve de la siguiente forma:

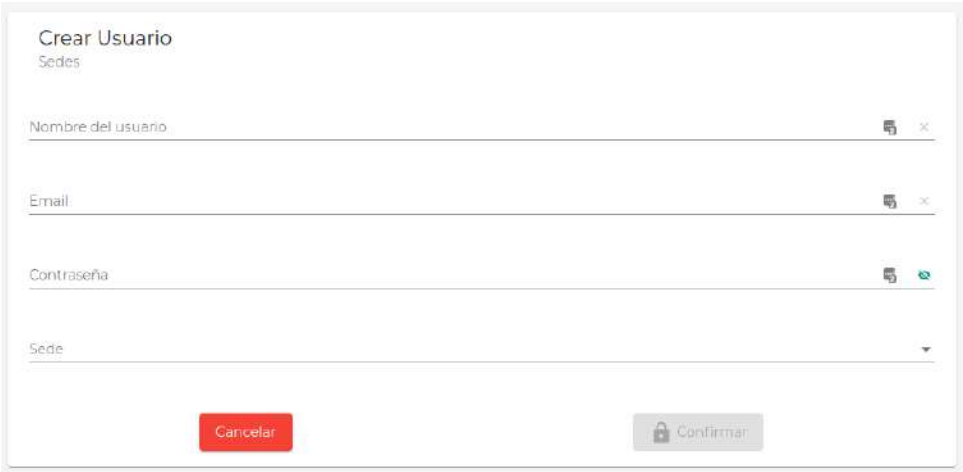



**Figura 92**. Creación de usuario “sede”

### Edición de usuarios

Como bien se mencionó antes, cada listado prefiltrado permite acceder a la edición de los usuarios del rol seleccionado. Para el caso de un usuario cátedra, dicha pantalla se despliega de la siguiente manera:

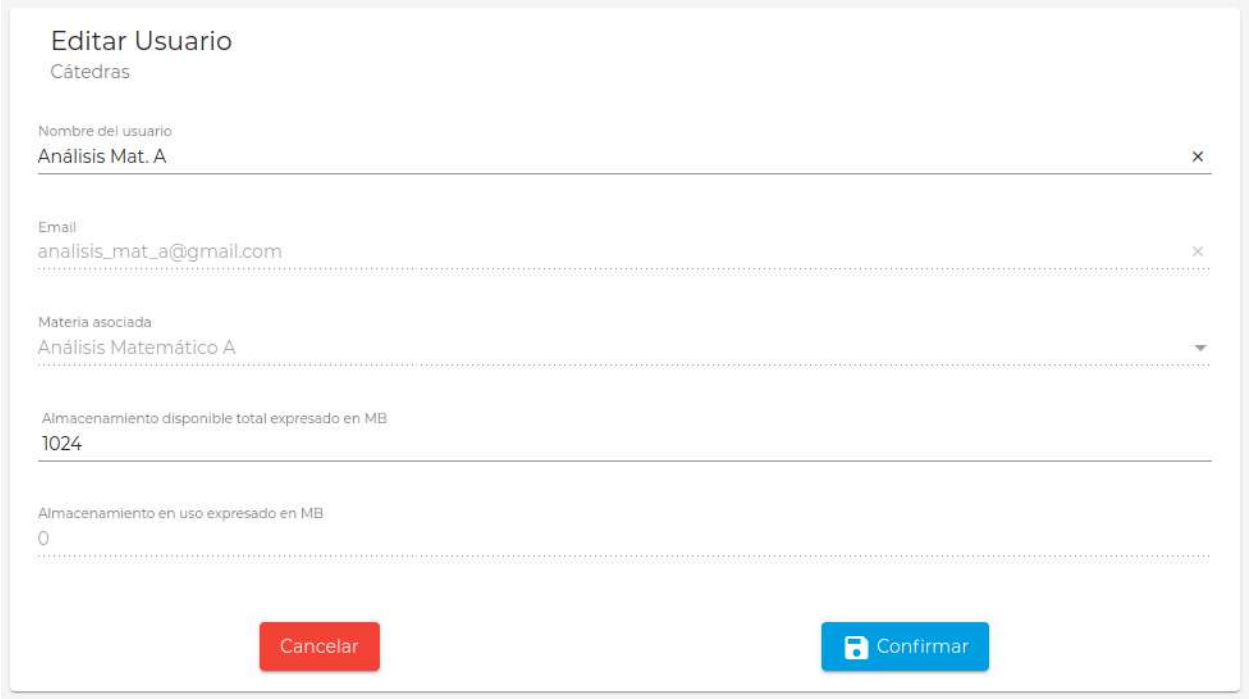



**Figura 93**. Edición de usuario "cátedra"

Para demás roles la pantalla de edición es similar, variando simplemente los atributos solicitados.

Especial atención merece la gestión de becados en esta sección. Al aplicar el filtro para visualizar estudiantes (que involucra también a usuarios becados), el listado pertinente es desplegado y se puede ver fácilmente en qué categoría está asignado cada estudiante a través de la columna señalada en la siguiente figura:

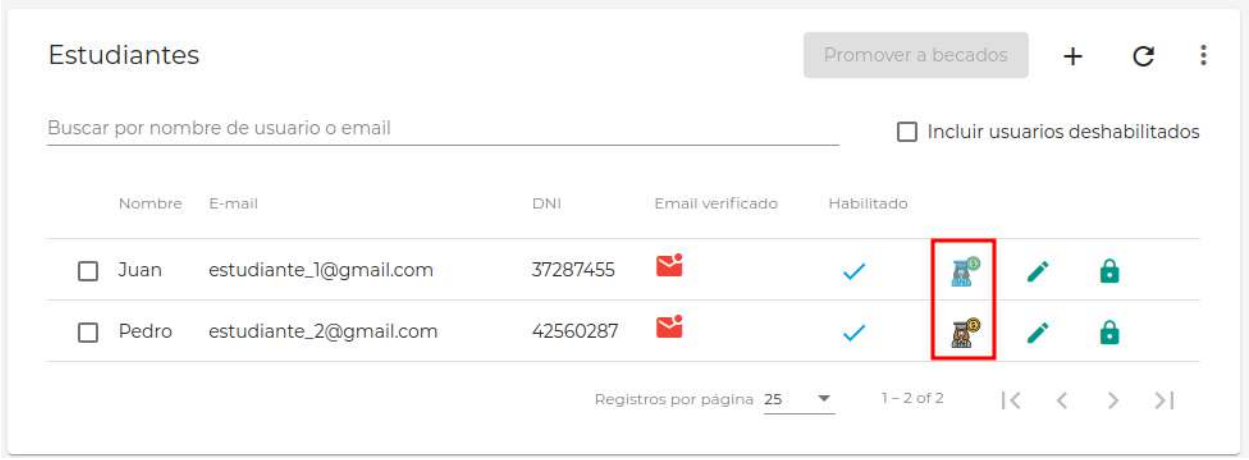



**Figura 94**. Visualización de estudiantes y becados

En el listado se puede ver que el primer usuario es un estudiante mientras que el segundo es un becado. La columna cumple una segunda función, permite a través del botón en ella promover o degradar un estudiante de la categoría que tiene asignada, teniendo un comportamiento similar a un interruptor de encendido/apagado.

Para simplificar la promoción de varios estudiantes a becados también es posible seleccionar todos aquellos estudiantes que se desea promover a través del checkbox de cada usuario en la primera columna del listado. Una vez seleccionados será posible promover a todos los alumnos a través del botón superior del listado.

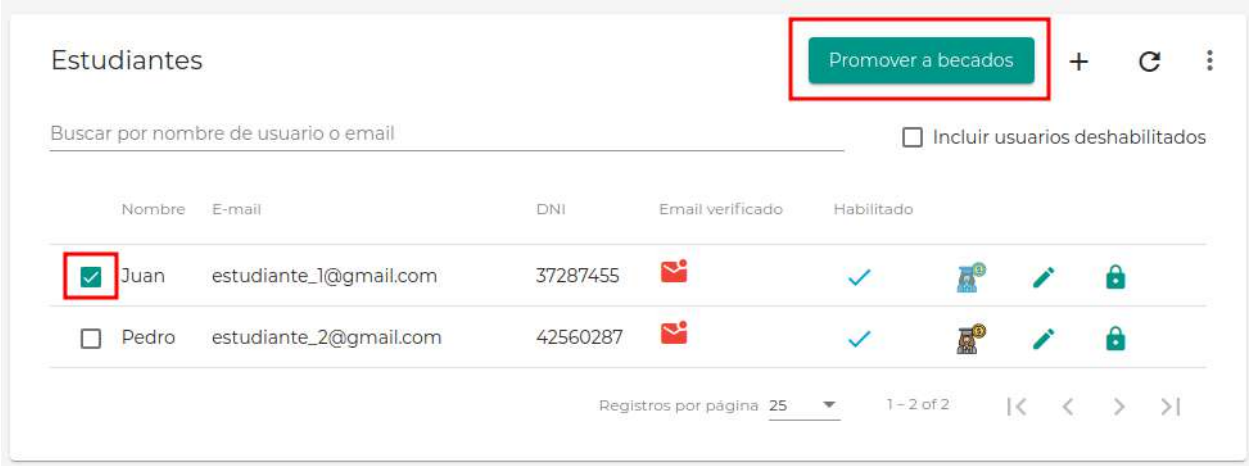



**Figura 95**. Segunda alternativa para promover estudiantes a becados

Del lado del becado (habiendo aplicado el filtro pertinente), es posible acceder a esta misma funcionalidad pero ahora la acción se enfocará en degradar becados a usuarios estudiantes.

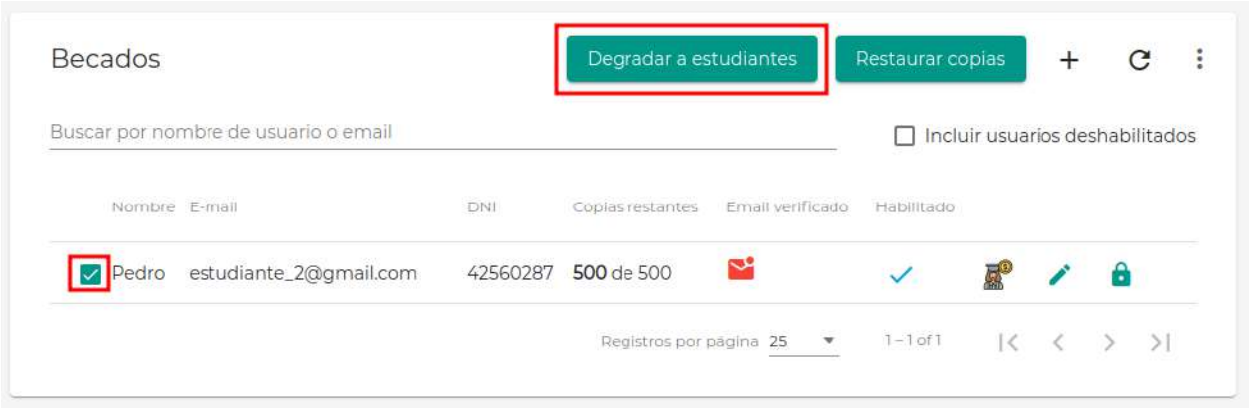



**Figura 96**. Degradación masiva de becados a estudiantes

Vale aclarar que la degradación específica de un usuario sigue siendo posible a través del botón pertinente que indica la categoría del usuario.

Por último, a fin de también facilitar la restauración de copias brindadas a cada estudiante por cuatrimestre, se dispone de un botón en la parte superior del listado de becados. A partir de su ejecución se vuelve a su valor máximo (el que tiene asignado cada becado en específico) las copias restantes de ***cada uno*** de ellos.

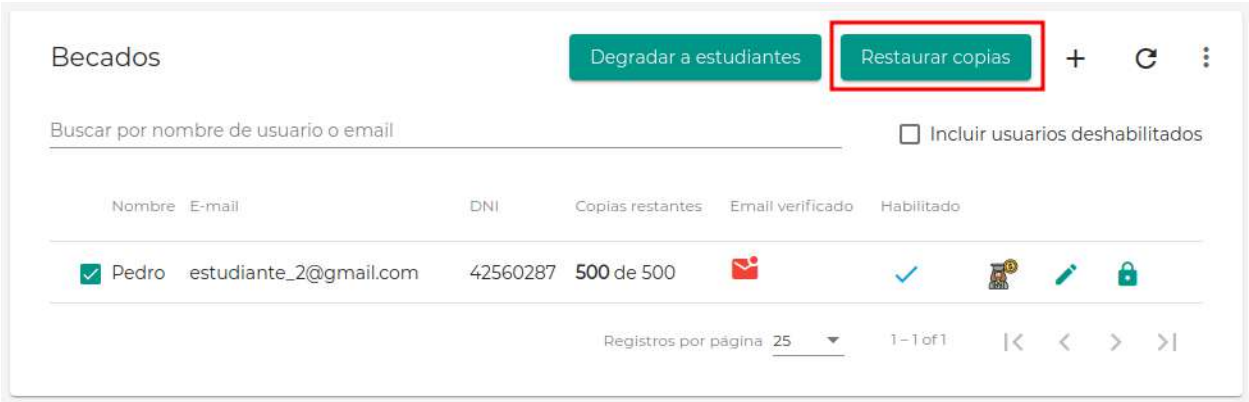



**Figura 97**. Restauración de copias restantes de todos los becados

### Eliminación de usuarios

La eliminación de usuarios se realiza a través del listado general o cualquiera de los listados filtrados, con algunas excepciones. Por decisiones de diseño se ha dispuesto que los usuarios estudiantes y becados no puedan ser eliminados del sistema, a fin de evitar inconsistencias en el modelo de datos.

Los demás roles del sistema pueden ser eliminados sin problemas. Una restricción impuesta por el sistema es que **siempre debe existir al menos un usuario administrador en el sistema**, por lo cual un administrador nunca podrá eliminarse a sí mismo.

Por otro lado, otra característica importante a mencionar es que la eliminación de un usuario "cátedra" **no elimina** los archivos que esta haya subido a la plataforma, a fin de evitar cualquier tipo de error que pueda cometer el usuario administrador inconscientemente. Estos archivos serán desvinculados del usuario a eliminar y quedarán estando disponibles en el sistema. Solo recién cuando la materia a la cual estaba asociado el usuario "cátedra" sea borrada del sistema serán borrados todos los archivos asociados a la misma.

## Gestión de sedes

El administrador contará con un ABM de sedes, las cuales permitirán luego ser asociadas con un usuario de tipo sede y poder realizar pedidos a la misma.

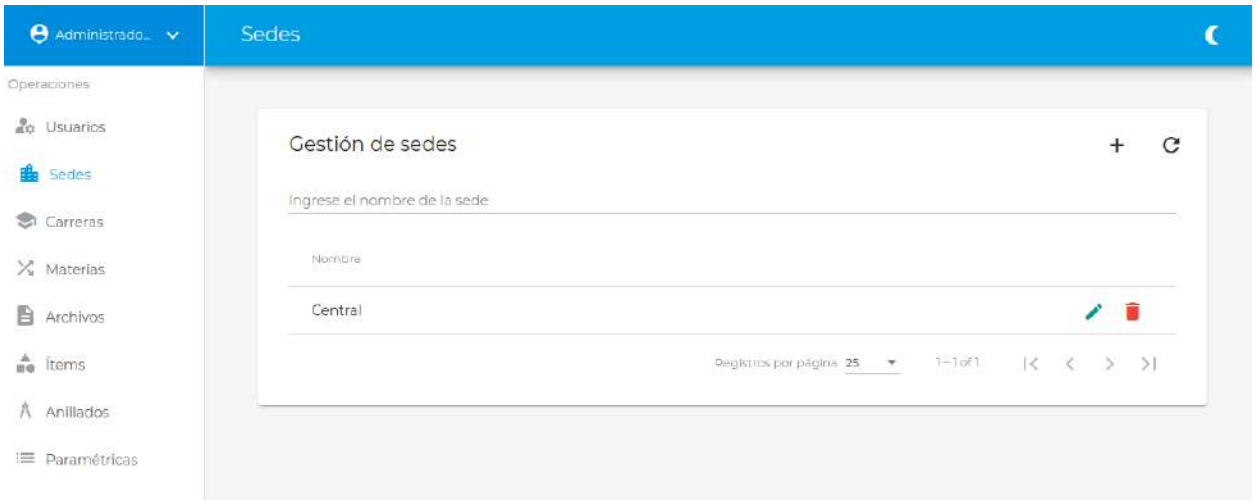



**Figura 98**. Pantalla de administración de sedes

El administrador podrá buscar las sedes creadas, alterar su nombre o incluso eliminarlas (siempre y cuando no estén asociadas a un usuario de tipo sede habilitado en la plataforma).

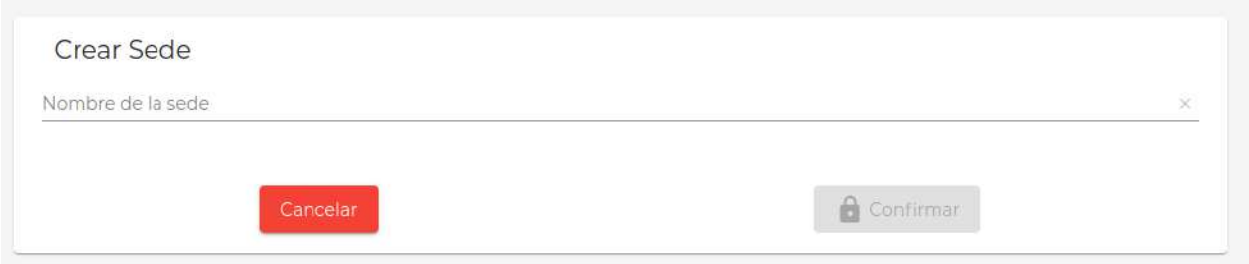



**Figura 99**. Creación de sede

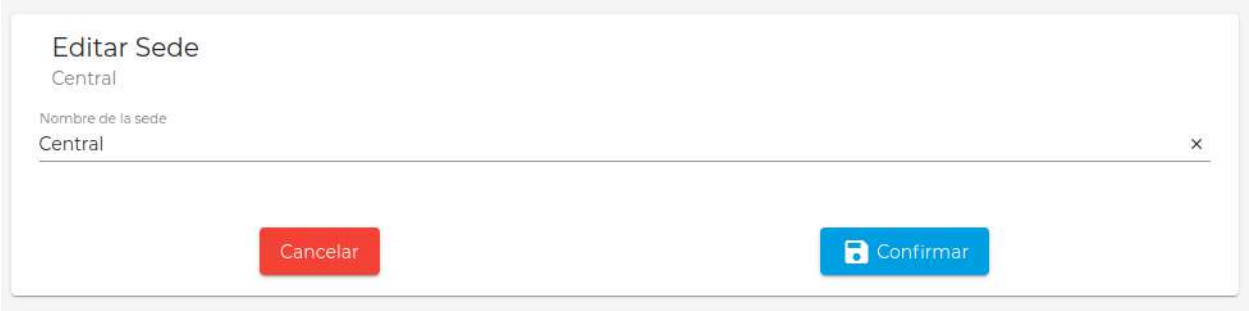



**Figura 100**. Edición de sede

En caso que la sede esté asociada a un usuario, un mensaje de error será retornado:

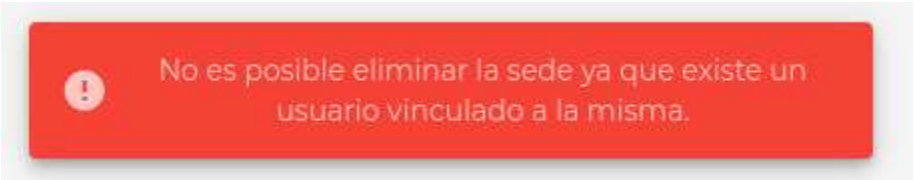



**Figura 101**. Mensaje de error durante eliminación de sede

## Gestión de carreras

Para el caso de las carreras su gestión es muy similar a la presentada para las sedes anteriormente. Se permite un ABM de las mismas y aquí la única diferencia es que la condición para que una carrera sea borrada consiste en que no esté asociada a ninguna materia existente.

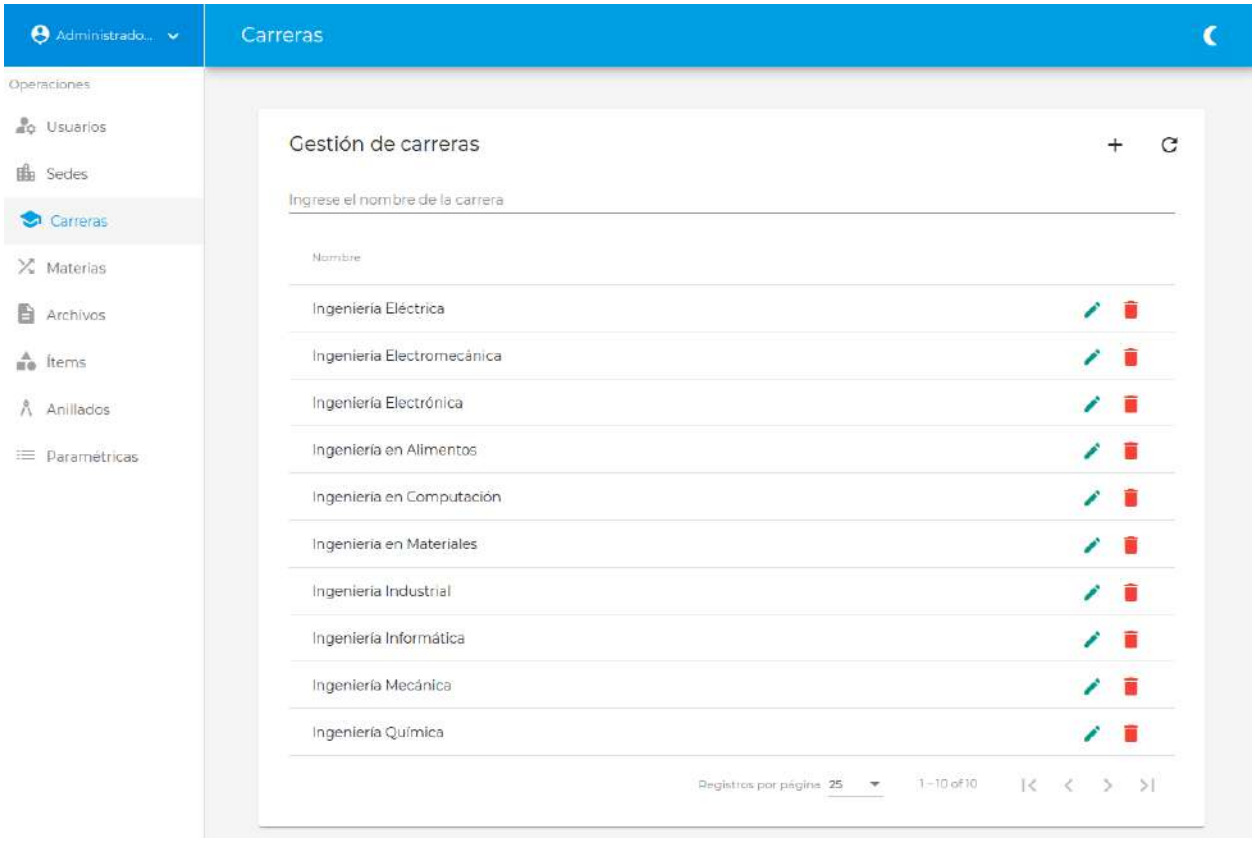



**Figura 102**. Administración de carreras

## Gestión de materias

La administración de materias se encuentra fuertemente influenciada por las carreras que hayan sido creadas en el sistema.

El administrador en primera instancia puede consultar las materias registradas en el sistema y las carreras que han sido asociadas a cada una de ellas.

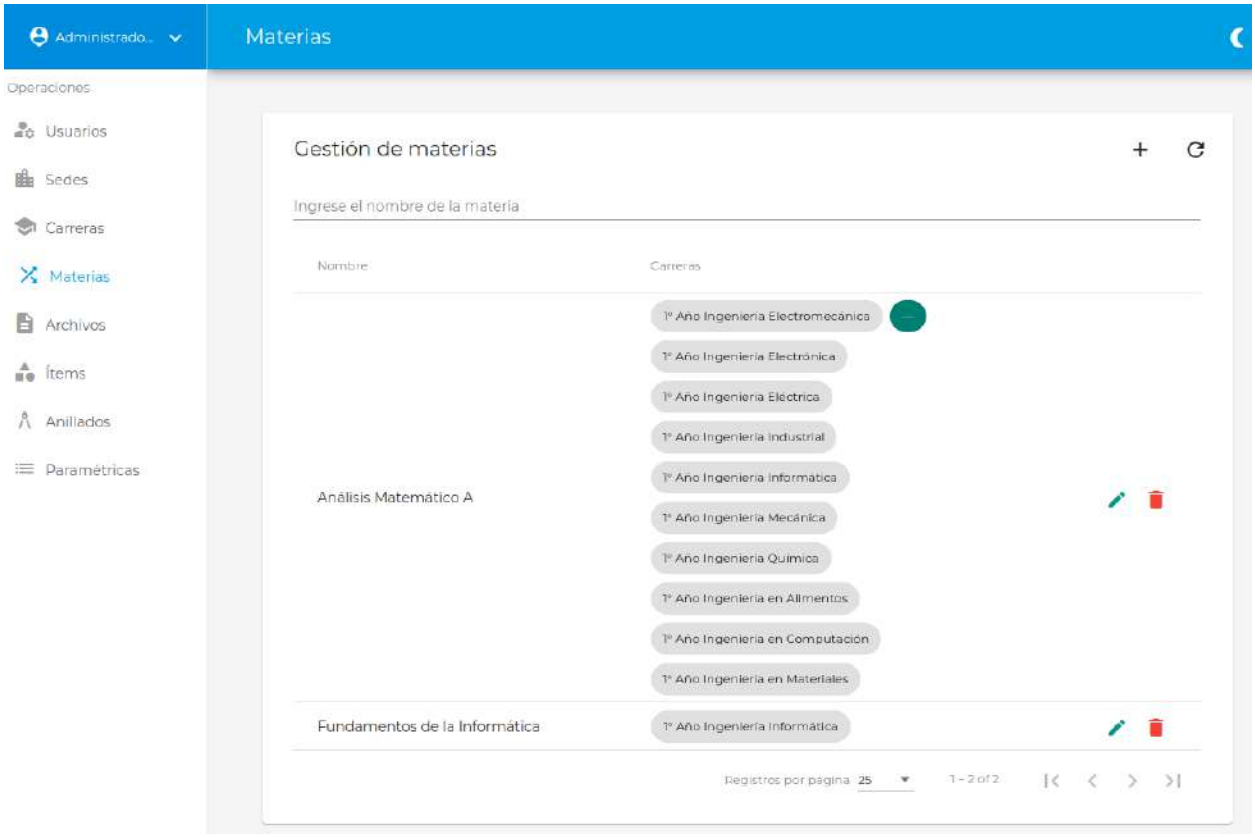



**Figura 103**. Listado de materias registradas en el sistema

La creación de una nueva materia requiere suministrar el nombre de la misma y establecer las relaciones que se deseen establecer con las diferentes carreras.


**Figura 104**. Creación de nueva materia

En la figura anterior puede observarse que la materia “Física Experimental” ha sido asociada con todas las carreras en 3er año, a excepción de Ingeniería Electrónica que la tiene planificada para el 4to año.

Una vez creada la materia, su edición es a través de una pantalla exactamente igual que la del alta.

La manipulación de materias y la gestión de sus relaciones con las distintas carreras impacta directamente en el árbol de archivos que visualiza el estudiante en la emisión de un nuevo pedido.

Respecto a la eliminación de materias se deben hacer dos aclaraciones importantes:

1. La acción de eliminar una materia está disponible solo cuando no existe un usuario cátedra asociado a la misma. En caso que exista esta situación se devuelve un mensaje de error.
2. La eliminación de una materia provoca que todos los archivos asociados ***exclusivamente*** a ella (y no para archivos que están vinculados a esta y otras materias) sean borrados del sistema, una vez que los pedidos en curso que los utilizaban hayan finalizado. Esto incluye archivos creados por el administrador, un usuario sede o la propia cátedra.

## Gestión de ítems

El sistema considera como ítems con un valor asociado a las copias color, simple y doble faz. Los mismos son posibles de ser actualizados a través del listado de esta sección del portal de administración.

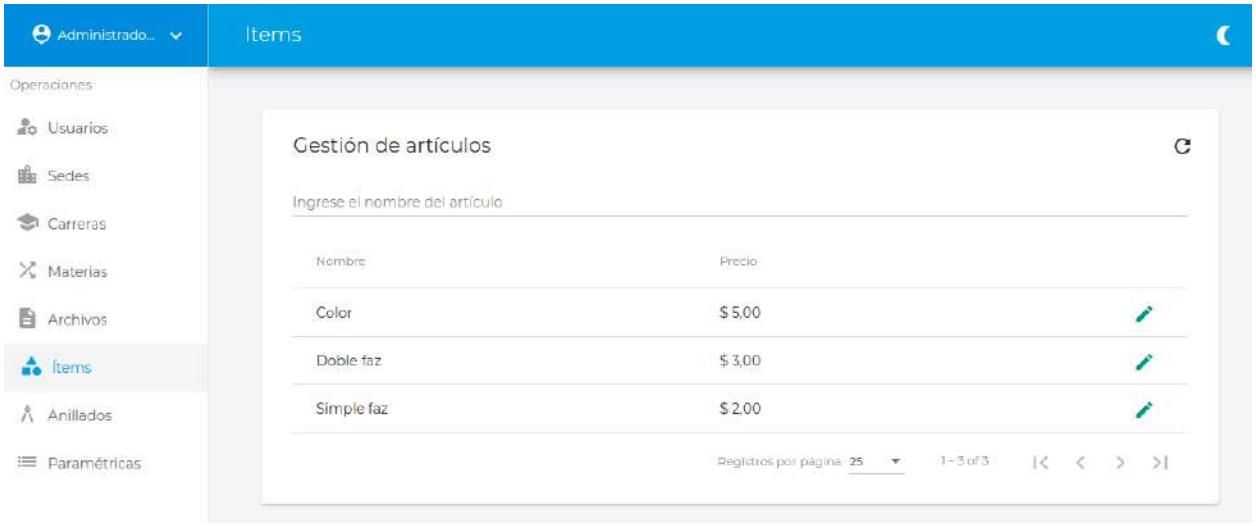



**Figura 105**. Listado de ítems existentes en el sistema

Dado que estos ítems son imprescindibles para el funcionamiento del sistema, solo se permite modificarlos y no crear nuevos o borrar los existentes.

A través de la pantalla de edición se podrá modificar el valor que poseen. Esto no afectará a los pedidos que ya hayan sido generados, sino que el valor empezará a regir desde ese instante en adelante.

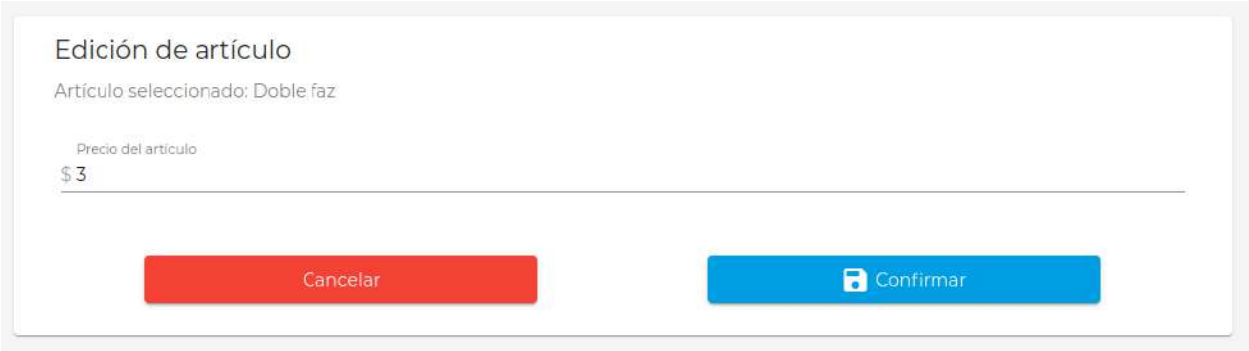



**Figura 106**. Edición del precio del ítem “Doble faz”

## Gestión de anillados

Similar al caso anterior presentado, la plataforma soporta la gestión de anillados a ser utilizados en los pedidos a generar. La principal diferencia radica en que ahora se podrán crear o borrar anillados, con la única condición que ***deberá existir siempre al menos un anillado en el sistema***, a fin de ser compatible el flujo de emisión del pedido.

De esta forma, el listado de anillados se puede ver de la siguiente manera:

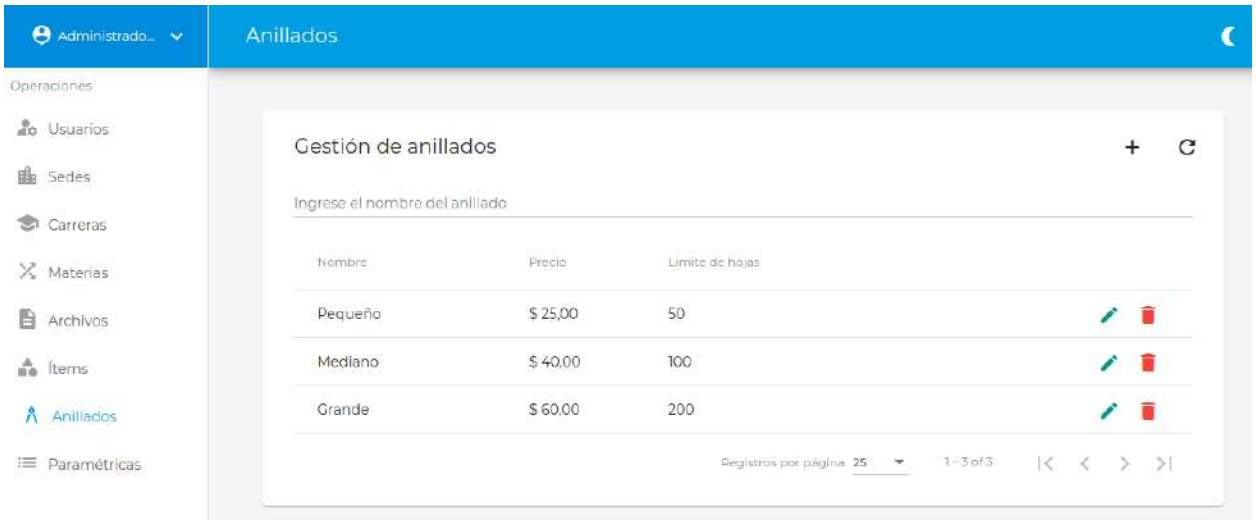



**Figura 107**. Listado de anillados

La creación de un nuevo anillado se realiza proporcionando su nombre, precio y cantidad de hojas que admite. Por otro lado, la edición permite modificar cualquiera de estos campos afectando a los pedidos por emitirse y no los actualmente ya creados.


**Figura 108**. Creación de nuevo anillado

Por último, la eliminación de un anillado se puede ejecutar sin problema, nuevamente sólo surtiendo efecto para los nuevos pedidos por emitirse.

***Nota**: originará un error en el caso que solo exista un único anillado en el sistema y desee borrarse.*

## Gestión de paramétricas

Con el objetivo de volver dinámico al sistema se decidió implementar un apartado que permita modificar algunos de los parámetros que rigen su comportamiento. A través de esta sección el administrador podrá alterar su funcionamiento sin necesidad de reiniciar toda la plataforma.

Nuevamente, todas estas variables pueden ser consultadas y modificadas desde el portal del administrador.

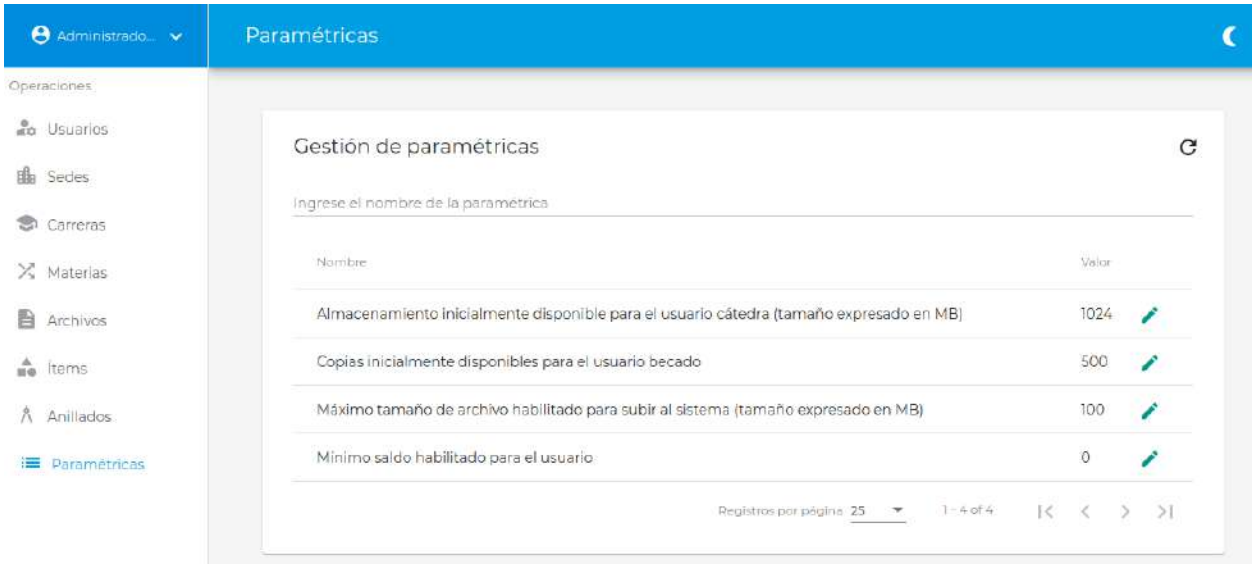



**Figura 109**. Listado de parámetros del sistema

A fin de uniformizar su tratamiento del lado del servidor, todas las variables han sido normalizadas para trabajar con números dentro de un orden de notación fácil de manipular por el administrador. Particularmente para aquellas variables que referencian espacio de almacenamiento, se ha seleccionado como unidad MB (Megabytes).

En esta sección no es posible crear nuevos parámetros o eliminar los existentes, sólo está permitido su edición.

En caso que el sistema experimente modificaciones o nuevas funcionalidades y sean necesarios algunos parámetros, los mismos podrán ser agregados del lado del servidor sin mucha complejidad.

La pantalla de edición de un parámetro puede verse de la siguiente manera:


**Figura 110**. Edición de copias inicialmente disponibles para un usuario becado


dbdiagram.io


dbdiagram.io

```
Enum users_type{
  admin
  campus_user
  professorship
  scholarship
  student
}


Enum files_type{
  system_staff
  system_professorship
  temporary
}


Enum parameters_code{
  users_minimum_balance_allowed
  users_professorships_initial_available_storage
  users_scholarships_initial_available_copies
  orders_minimum_number_of_sheets_for_deposit
  orders_percentage_of_deposit
  files_max_size_allowed
}
```

```
Enum items_code{
  simple_sided
  double_sided
  colour
}


Enum items_type{
  item
  binding
}
```

```
Enum functionalities_code{
  menu
  home
  orders
  movements
  operations
  new_order
  my_orders
  my_movements
  top_up
  transfer_money
}


Enum roles_code{
  admin
  campus_user
  professorship
  scholarship
  student
}
```

```
Enum notification_types_code {
  top_up
  incoming_transfer
  promotion_to_scholarship
  degradation_to_student
  available_copies_restored
  order_in_process
  order_ready
  order_cancelled
  order_undelivered
  order_delivered
}


Enum movement_types_code {
  requested_order
  cancelled_order
  top_up
  transfer
}
```

```
Enum order_states_code{
  requested
  in_process
  ready
  cancelled
  undelivered
  delivered
}


Enum file_states_code{
  to_print
  printing
  printed
}


Enum binding_group_state{
  to_ring
  ringed
}
```