UNMDP - Facultad de Ingeniería
Departamento de Informática

IlliApp: Sistema Informático de gestión académica para el Colegio Nacional Dr. Arturo Umberto Illia

Estudiantes

Cardelli, Ramiro

de Lellis, Lucas

Frasca Ponce, Josefina

González, Franco

Director

MBA Lic. Genin, Fernando

Codirector

Esp. Lic. Hinojal, Hernán

Proyecto final para optar por el grado de Ingeniero/a en Informática

Mar del Plata, 7 de mayo de 2026

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que, de distintas maneras, hicieron posible la realización de este trabajo.

En primer lugar, a nuestras familias, por su apoyo incondicional a lo largo de toda la carrera. Su acompañamiento, comprensión y confianza fueron fundamentales para poder transitar este camino y alcanzar este objetivo.

A nuestros directores de tesis, por su guía y acompañamiento durante el desarrollo de este proyecto. Sus aportes, sugerencias y orientación resultaron esenciales para llevar adelante este trabajo y enriquecer nuestra formación académica y profesional.

A nuestros compañeros y compañeras de la carrera, con quienes compartimos años de aprendizaje, desafíos y crecimiento.

Finalmente, a los docentes que formaron parte de nuestra formación a lo largo de la carrera, por transmitirnos no sólo conocimientos técnicos, sino también valores profesionales que hoy aplicamos en nuestro trabajo.

A todos ellos, nuestro más profundo agradecimiento.

Índice

Agradecimientos **2**
Índice **3**
Resumen 6
Introducción 7
Contexto y motivación del proyecto 7
Objetivo general 8
Alcance 8
Fuera del alcance 10
Impacto del proyecto 11
Estimación inicial 12
Análisis 12
Dominio 12
Problemas con el sistema actual 13
Problema a resolver 21
Conclusión del análisis 22
Análisis FODA 23
Requerimientos 24
Requerimientos funcionales 25
1.0 Generales 25
2.0 Asistencia 25
3.0 Actividades Evaluativas 26
4.0 Reportes 28
5.0 Roles 28
5.1 Administrador 29
5.2 Docente 29
5.3 Preceptor 30
5.4 Superpreceptor 30
5.5 Estudiante / Tutor 31
5.6 Directivo 31
5.7 División Estudiantes 32
6.0 Cambio de año 32
7.0 Seguridad 32
8.0 Cambios de curso/orientación 33
Requerimientos no funcionales 33
Benchmarking / análisis de la competencia 34
Análisis de riesgos 37
Análisis de costos 40
Diseño del sistema 41

Metodologías utilizadas....42
Metodologías de análisis.... 42
Metodologías de diseño....43
Metodologías de desarrollo....44
Metodologías de testing....46
Tecnologías y herramientas.... 48
Backend.... 49
Frontend....50
Base de datos.... 51
Otras herramientas.... 53
Arquitectura general del sistema....56
Diseño del modelo de datos....57
Módulos principales....58
Interfaz de usuario y experiencia de usuario....61
Seguridad y control de accesos.... 70
Componente innovador.... 74
Producto.... 76
Producto obtenido.... 76
Trabajos futuros....77
Memoria del proyecto.... 80
Problemáticas encontradas y soluciones....80
Dinámica de trabajo y planificación temporal.... 81
Organización inicial y forma de abordaje....81
Definición del modelo de datos y complejidad institucional....82
Validaciones funcionales y retrabajo....83
Entorno de ejecución y limitaciones técnicas.... 83
Interfaz de usuario y homogeneización del diseño....83
Síntesis.... 84
Cumplimiento de los objetivos....85
Trabajo en equipo....86
Comparación entre planificación esperada y ejecutada....87
Análisis por etapas.... 91
Análisis....91
Reflexión del análisis FODA.... 92
Retrospectiva del análisis de riesgos....94
Diseño....95
Desarrollo....96
Testing....97
Despliegue.... 99
Capacitación.... 101

Principales aprendizajes..... 102
Conclusiones..... 103
Glosario..... 105
Anexos..... 114
Anexo I: Diagramas Entidad-Relación.....114
Anexo II: Acuerdo de entrega, confidencialidad y desvinculación del sistema IlliApp.....120
Bibliografía.....123

Resumen

El presente informe describe el desarrollo del sistema de gestión académica destinado al Colegio Nacional Dr. Arturo Umberto Illia. El proyecto fue realizado por los estudiantes Ramiro Cardelli, Lucas de Lellis, Josefina Frasca Ponce y Franco González, en el marco del Trabajo Final de la carrera de Ingeniería en Informática de la Universidad Nacional de Mar del Plata.

El objetivo principal del trabajo fue diseñar e implementar una solución integral orientada a optimizar la administración escolar, facilitando la gestión de usuarios, el registro de asistencia de estudiantes y docentes, la carga y consulta de calificaciones, la generación automática de boletines y la organización de horarios de cursada. Asimismo, se buscó avanzar en la despapelización de los procesos administrativos y académicos del colegio, reduciendo la dependencia de registros manuales y documentación física, y promoviendo una gestión más eficiente, trazable y centralizada de la información.

El proyecto surge como un reemplazo del sistema preexistente, con el objetivo de acompañar nuevos requerimientos y mejorar determinados aspectos de su funcionamiento. A partir de este análisis, se planteó la ingeniería de una nueva aplicación orientada a optimizar la usabilidad y mantenimiento del sistema, incorporando tecnologías modernas en su arquitectura y diseño.

El desarrollo se abordó mediante una metodología iterativa e incremental, lo que permitió validar progresivamente el producto con el referente funcional del Colegio Illia. El equipo se responsabilizó de todas las etapas: análisis de requerimientos, diseño de la arquitectura, implementación del sistema, pruebas de validación e integración.

Los resultados obtenidos demuestran una mejora significativa en la digitalización de los procesos académicos y administrativos de la institución. Como evidencia cualitativa de los resultados obtenidos, el referente funcional del colegio manifestó su conformidad con el sistema desarrollado durante las instancias de validación, señalando que la solución responde a las necesidades operativas identificadas al inicio del proyecto. Asimismo, el proyecto permitió aplicar conocimientos teóricos de

ingeniería de software en un contexto real, consolidando competencias en gestión de proyectos, trabajo en equipo y desarrollo de sistemas complejos.

Introducción

La gestión eficiente de la información constituye un pilar fundamental para el funcionamiento óptimo de las instituciones educativas modernas. Las escuelas requieren sistemas confiables que permitan administrar de manera integral los procesos académicos, administrativos y comunicacionales, contribuyendo a una organización eficaz y a la calidad educativa.

El Colegio Nacional Dr. Arturo Umberto Illia, dependiente de la Universidad Nacional de Mar del Plata, se distingue por su compromiso con la innovación pedagógica y adopción tecnológica. No obstante, los sistemas disponibles hasta el momento de inicio de este proyecto presentaban limitaciones críticas en términos de estabilidad, escalabilidad y usabilidad, afectando significativamente la eficiencia del personal docente y administrativo.

En este contexto, el presente Trabajo Final de la carrera de Ingeniería en Informática tiene como propósito diseñar y desarrollar desde cero una plataforma de gestión académica orientada a centralizar la información institucional y digitalizar los principales procesos administrativos y académicos. El proyecto permite aplicar conocimientos teóricos en un entorno real mediante la implementación de una solución que mejora la trazabilidad de la información, reduce la carga operativa asociada a tareas manuales y facilita el acceso a los datos por parte de los distintos actores de la comunidad educativa del ámbito público.

Contexto y motivación del proyecto

El análisis del estado previo evidenció que las soluciones tecnológicas implementadas no satisfacían adecuadamente la complejidad operativa del modelo académico del Colegio Nacional Dr. Arturo Umberto Illia. Las carencias funcionales y los problemas de usabilidad generaban fricciones en tareas cotidianas como el

registro de asistencias, la carga de evaluaciones y la consulta de información académica, incrementando significativamente la carga administrativa y el margen de error en los procesos internos.

El proyecto aspira a modernizar y optimizar la gestión escolar mediante la aplicación de conocimientos de ingeniería de software en un entorno real, aportando valor tangible a la comunidad educativa. La motivación central es digitalizar y centralizar los procesos académicos y administrativos para mejorar la eficiencia operativa, fortalecer los canales de comunicación institucional y garantizar un control académico riguroso, contribuyendo así a la calidad educativa global de la institución.

Objetivo general

El proyecto tiene como objetivo diseñar e implementar un sistema informático para la gestión integral en tiempo real del Colegio Illia.

En este marco, se establecen como objetivos específicos apoyar a la dirección en su política de despapelización, acompañar la reingeniería de procesos institucionales y garantizar la disponibilidad de datos.

Alcance

El alcance del proyecto comprende el análisis, diseño, desarrollo e implementación de una plataforma web destinada a centralizar la gestión académica y administrativa del Colegio Nacional Dr. Arturo Umberto Illia. El sistema abarca las funcionalidades necesarias para acompañar los procesos institucionales vinculados al seguimiento académico, la organización escolar y la administración de la información, garantizando la disponibilidad de los datos en tiempo real y el acceso controlado según el rol del usuario.

El desarrollo contempla los siguientes módulos funcionales:

Gestión Académica
Administración de materias, talleres y cursos, así como su vinculación con docentes,

preceptores y estudiantes. Incluye la gestión de escalas de aprobación, respetando las normativas académicas propias de la institución.

Gestión de Usuarios

Alta, baja y modificación de usuarios, junto con la asignación y administración de roles institucionales. El sistema contempla la coexistencia de múltiples roles por usuario y el control de permisos asociado a cada uno.

Gestión de Asistencias

Registro y visualización de asistencias diarias de estudiantes y docentes, con posibilidad de carga retroactiva y futura para los docentes, incorporación de justificaciones y observaciones, y edición controlada de los registros. Este módulo es operado por docentes, preceptores, directivos y personal del establecimiento, y permite la consulta por parte de estudiantes y tutores.

Evaluaciones y Calificaciones

Incluye la creación y modificación de actividades evaluativas, carga y administración de calificaciones y validaciones específicas por tipo de actividad, contemplando restricciones y reglas particulares vinculadas al desempeño académico.

Anuncios

Publicación, edición y eliminación de anuncios institucionales, centralizando la comunicación oficial dentro de la plataforma.

Reportes Institucionales

Generación de boletines escolares, informes de rendimiento académico y visualización del historial de inasistencias y calificaciones, accesibles de acuerdo con los permisos definidos.

Seguridad y Autenticación

Gestión del acceso al sistema mediante autenticación obligatoria, administración de credenciales, controles de seguridad en primeros inicios de sesión y bloqueos ante intentos fallidos de acceso.

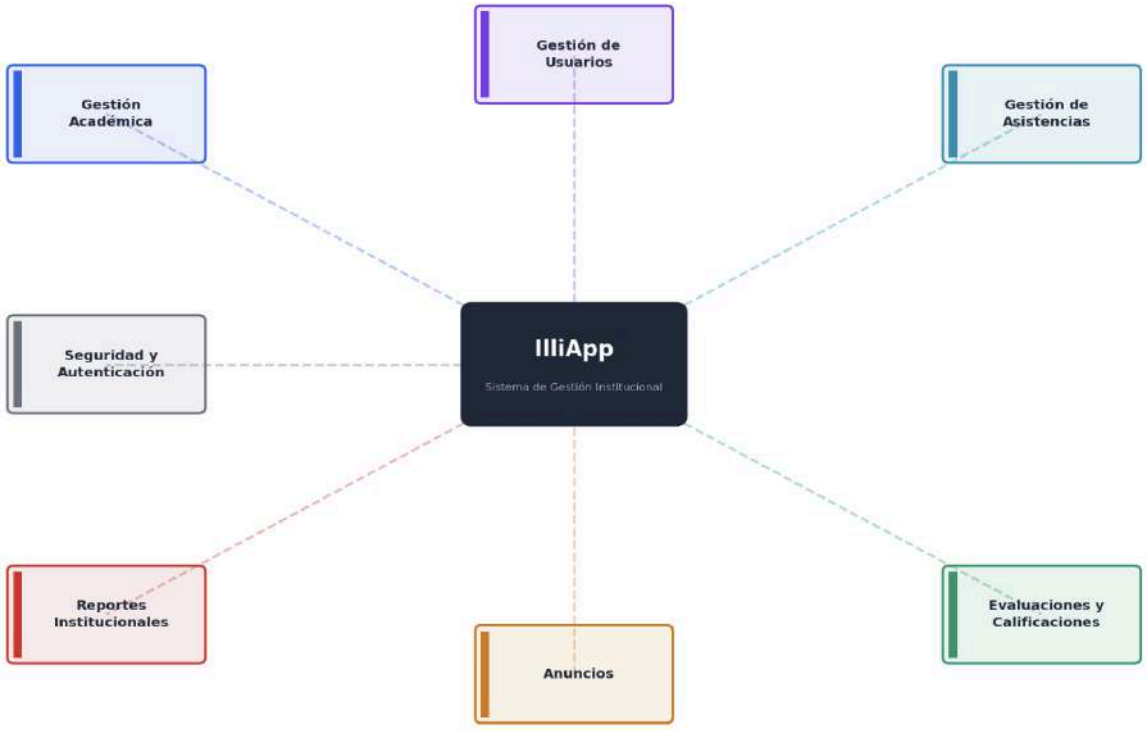


Figura 1. *Módulos funcionales del sistema IlliApp.*

Fuera del alcance

El presente proyecto no contempla el desarrollo de módulos vinculados a la gestión financiera, recursos humanos, mantenimiento de infraestructura ni a la administración técnica del sistema. Asimismo, quedan excluidas las integraciones con sistemas externos al ámbito académico, tales como plataformas gubernamentales de gestión educativa, sistemas contables u otras soluciones de terceros.

No se incluyen funcionalidades de comunicación sincrónica entre usuarios, como mensajería instantánea, chats o notificaciones en tiempo real. Tampoco forma parte del alcance la implementación de una aplicación móvil nativa, si bien la plataforma fue diseñada para ser utilizada desde dispositivos móviles a través del navegador web.

Además, se excluyen funcionalidades avanzadas de análisis de datos o tableros de control en tiempo real, limitándose el sistema a la generación de reportes

institucionales básicos relacionados con asistencia, calificaciones y estado académico.

Durante el desarrollo se identificaron requerimientos válidos que formaron parte del análisis inicial, pero que fueron excluidos del alcance de esta primera iteración para asegurar la estabilidad de las funcionalidades centrales y cumplir con los plazos del proyecto. Se decidió aplazar su implementación para futuras versiones a los siguientes requerimientos: inhabilitación automática de estudiantes egresados, historial visible de modificaciones en calificaciones, actividades evaluativas conjuntas entre materias, automatización de cambio de ciclo lectivo, promoción y cambios de orientación y cobertura de pruebas automatizadas extendida.

Finalmente, la implementación operativa inicial del sistema (carga o migración de datos reales) y las tareas de mantenimiento o soporte post-entrega quedan excluidas. Durante el desarrollo no se contó con acceso a bases de datos reales ni a información sensible de la institución (datos personales, registros de asistencia reales, calificaciones). Las validaciones del sistema se realizaron exclusivamente con datos ficticios para preservar la confidencialidad institucional.

El producto desarrollado quedará a disposición del colegio, permitiendo su puesta en marcha definitiva, evolución y la posterior incorporación de las funcionalidades aplazadas según sus propias necesidades y tiempos.

Impacto del proyecto

El sistema ofrece una solución tecnológica que supera las limitaciones del sistema preexistente, aumentando la eficiencia, estabilidad y calidad de los procesos educativos.

Los beneficiarios directos serán todos los miembros de la comunidad educativa: preceptores, docentes, estudiantes, padres o tutores y directivos. Podrán acceder a información actualizada en tiempo real, facilitando el trabajo diario, mejorando la comunicación y permitiendo un seguimiento más efectivo de las actividades académicas. La digitalización reducirá la carga administrativa y fomentará un

ambiente organizado y participativo, fortaleciendo la calidad educativa y la colaboración institucional.

Además, la implementación del sistema establecerá una base tecnológica para futuras ampliaciones que respondan a nuevas necesidades del colegio.

Estimación inicial

El proyecto fue planificado considerando un cronograma temporal definido y realista para la ejecución de las fases de análisis, diseño, desarrollo, pruebas e integración. Se adoptó una metodología iterativa e incremental que permitió validar la evolución del sistema en cada ciclo con la participación activa del referente funcional del colegio. Esta metodología aseguró una gestión eficiente del tiempo y recursos, permitiendo entregar un producto final que cumple con los objetivos establecidos dentro del marco académico.

Cronograma inicial planificado:

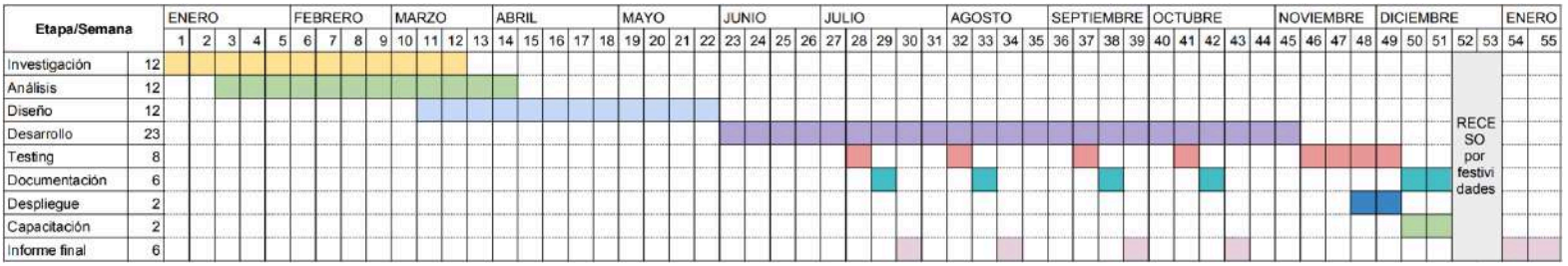


Figura 2. *Diagrama de Gantt presentado en el protocolo del proyecto.*

Análisis

Dominio

El proyecto se desarrolla en el contexto particular del Colegio Nacional Dr. Arturo Umberto Illia, institución con normativas, procedimientos y regulaciones propios que determinan su gestión académica y administrativa. Esto incluye la administración de

asistencia, calificaciones, horarios y comunicación, respetando los lineamientos específicos de la institución, que no se replican en otras escuelas.

Problemas con el sistema actual

Durante la etapa de análisis del sistema utilizado por el Colegio Nacional Arturo U. Illia, se identificaron múltiples deficiencias funcionales, estructurales y de usabilidad que motivaron la necesidad de desarrollar una nueva solución. Estas problemáticas fueron relevadas tanto a partir del uso cotidiano de la plataforma como mediante entrevistas y material provisto por el referente funcional de la institución.

A continuación se detallan las deficiencias críticas identificadas, organizadas por categoría:

Deficiencias en el modelo de datos

Modelado incorrecto de relaciones académicas

El sistema preexistente presentaba una modelación inadecuada de las entidades fundamentales. La vinculación entre docente, materia y curso no se encontraba definida explícitamente, estableciéndose de manera indirecta a través de la entidad ‘horario’. Esta decisión de diseño generaba inconsistencias estructurales que afectaban múltiples funcionalidades.

Como consecuencia, resultaba imposible identificar de forma directa qué materia dictaba un docente en un curso determinado sin consultar la tabla de horarios. Al ingresar al sistema, los profesores visualizaban sus horarios asignados pero no la materia asociada a cada bloque, dificultando la navegación y comprensión del contexto académico.

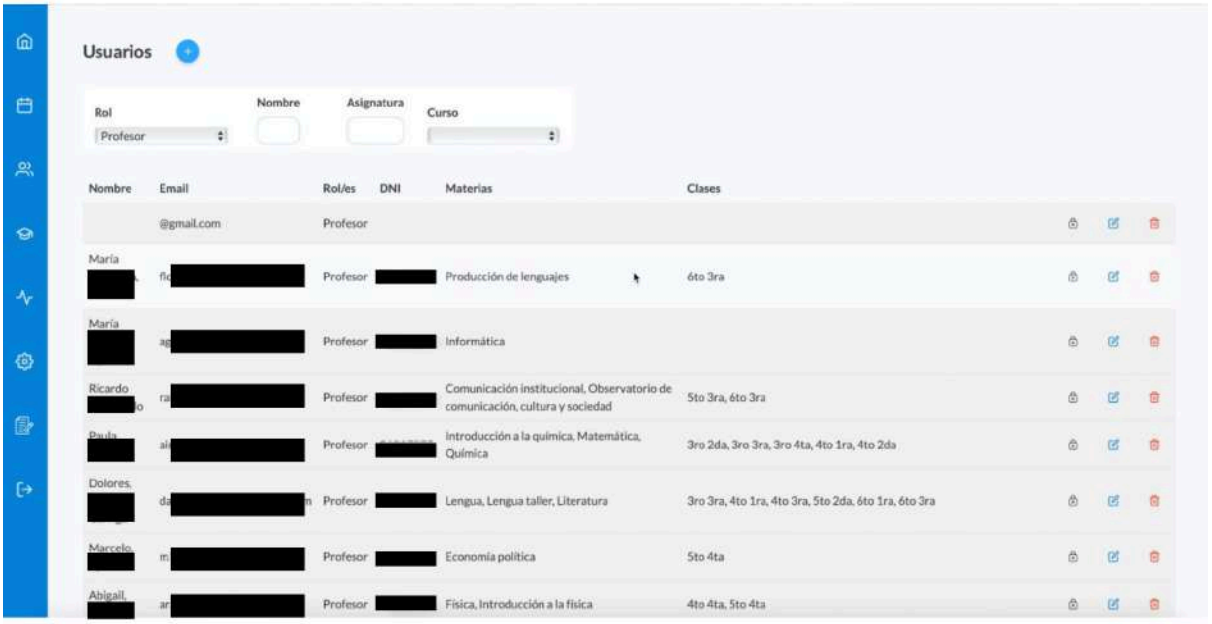


Figura 3. *Asociación ambigua entre clases y materias en la planilla de usuarios del sistema preexistente.*

Pérdida de información por dependencias incorrectas

Las calificaciones se encontraban asociadas directamente al docente que las registró. Esta decisión de modelado generaba una dependencia crítica: ante la desvinculación de un profesor del sistema, las notas cargadas por ese docente quedaban inaccesibles o se perdían completamente, afectando el historial académico de los estudiantes.

Falta de información temporal en calificaciones

Al consultar sus calificaciones, los estudiantes no podían distinguir a qué cuatrimestre correspondía cada una. La ausencia de contexto temporal generaba confusiones recurrentes, especialmente en materias anuales con evaluaciones parciales en ambos cuatrimestres.

Gestión inadecuada de roles y permisos

Coexistencia de roles sin control de contexto

El sistema asignaba simultáneamente todos los permisos correspondientes a los roles de un usuario, sin permitir seleccionar el rol activo en un momento determinado. Esta implementación provocaba confusión en la interfaz, habilitando funcionalidades inapropiadas para el contexto de uso actual.

Por ejemplo, un docente que también cumplía funciones administrativas visualizaba opciones de toma de asistencia de cursos ajenos o accedía a horarios que no le correspondían en su rol docente. La ausencia de separación contextual de permisos generaba interfaces sobrecargadas y aumentaba el riesgo de errores operativos.

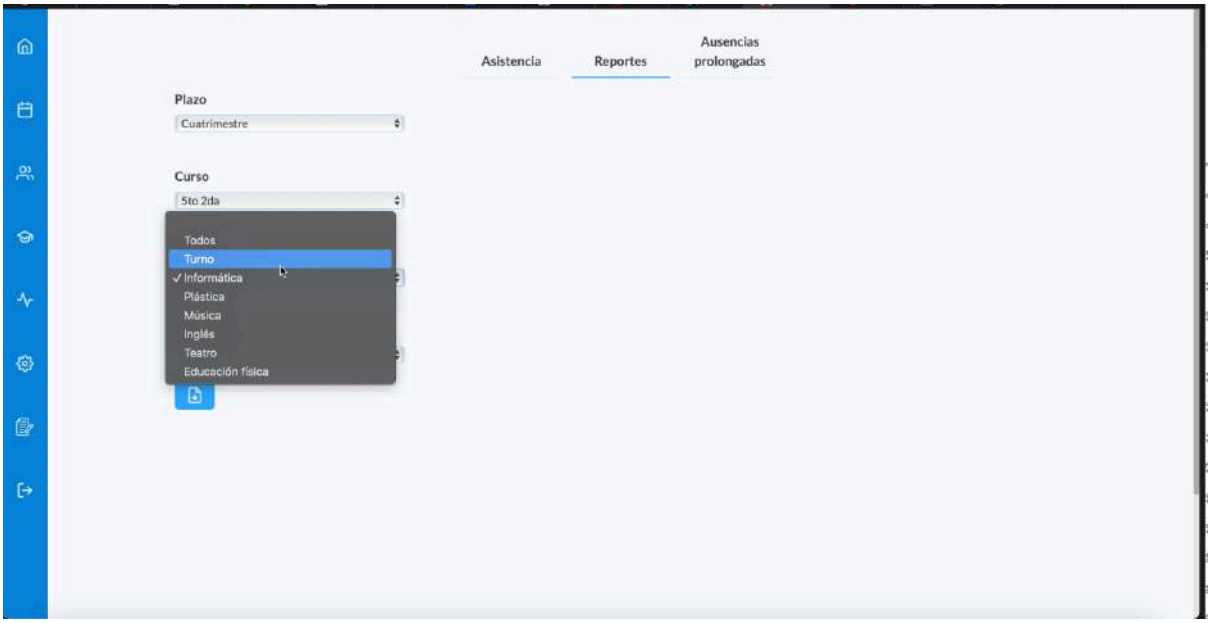


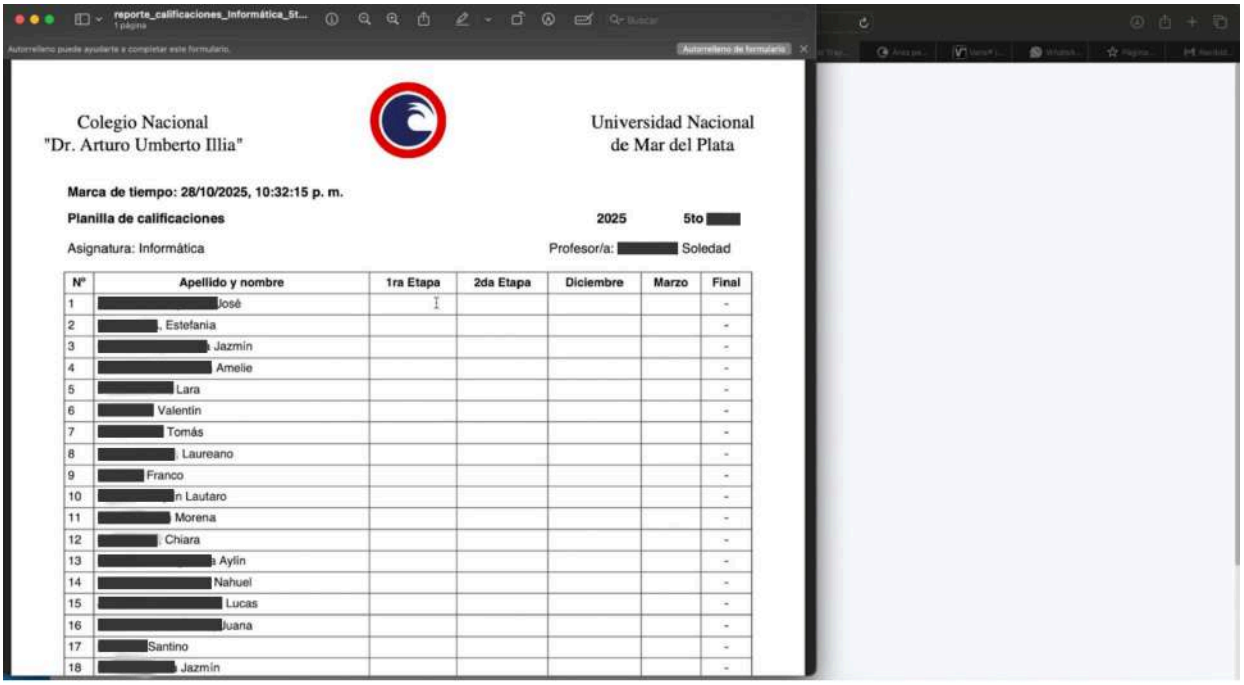
Figura 4. *Pantalla de toma de asistencia en la que un usuario con roles de profesor y administrador visualiza todos los cursos del sistema, evidenciando la falta de filtrado según el rol docente.*

Deficiencias en reportes institucionales

Generación inconsistente de reportes

La funcionalidad de generación de reportes presentaba fallas críticas en la visualización de calificaciones, con comportamientos diferentes según el perfil del

usuario que accedía. Esta inconsistencia nunca fue resuelta, llevando al abandono progresivo de esta funcionalidad.



Colegio Nacional
"Dr. Arturo Umberto Illia"

Universidad Nacional
de Mar del Plata

Marca de tiempo: 28/10/2025, 10:32:15 p. m.

Planilla de calificaciones **2025** **5to** [redacted]

Asignatura: Informática Profesor/a: [redacted] Soledad

Nº	Apellido y nombre	1ra Etapa	2da Etapa	Diciembre	Marzo	Final
1	[redacted] José					-
2	[redacted], Estefania					-
3	[redacted] Jazmín					-
4	[redacted] Amelie					-
5	[redacted] Lara					-
6	[redacted] Valentín					-
7	[redacted] Tomás					-
8	[redacted], Laureano					-
9	[redacted] Franco					-
10	[redacted]n Lautaro					-
11	[redacted] Morena					-
12	[redacted], Chiara					-
13	[redacted] Aylín					-
14	[redacted] Nahuel					-
15	[redacted] Lucas					-
16	[redacted] Juana					-
17	[redacted] Santino					-
18	[redacted], Jazmín					-

Figura 5. Plantilla de reporte de notas generada por el sistema, evidenciando la ausencia de datos para su visualización.

Uso limitado del sistema para reportes finales

Debido a los problemas mencionados, el sistema se utilizaba de forma parcial, principalmente para el registro de calificaciones conceptuales del espacio de Informes de Avance de las Trayectorias Educativas (IATE). Las calificaciones finales y la generación de boletines se gestionaban mediante procesos externos, en planillas de cálculo o documentos procesados manualmente.

Ausencia de generación automática

El sistema no generaba boletines ni analíticos de forma automática. Los documentos oficiales requeridos por normativa se elaboraban manualmente, incrementando significativamente la carga administrativa y el riesgo de errores de transcripción.

Deficiencias de seguridad y trazabilidad

Contraseñas por defecto sin cambio obligatorio

El sistema asignaba como contraseña inicial el número de DNI del usuario y no obligaba a su modificación en el primer inicio de sesión. Esta práctica exponía las cuentas a accesos no autorizados, especialmente considerando que los números de DNI son datos públicos fácilmente accesibles.

Ausencia de auditoría de operaciones

No existía un registro de auditoría que permitiera identificar quién realizó modificaciones sobre notas o asistencias, ni cuándo se efectuaron dichas modificaciones. Esta carencia resultaba especialmente problemática en un entorno institucional donde la trazabilidad de cambios en información académica sensible es fundamental para garantizar transparencia y resolver conflictos.

Eliminación de usuarios sin reversión

El mecanismo de eliminación de usuarios implementaba un borrado lógico que no podía revertirse desde la interfaz del sistema. Las reincorporaciones de estudiantes, situación frecuente en el ámbito educativo, requerían intervenciones directas sobre la base de datos, incrementando la dependencia del soporte técnico y los riesgos asociados a manipulaciones manuales.

Problemas de experiencia de usuario

Diseño no responsive

La interfaz no estaba completamente adaptada para dispositivos móviles. La visualización en tablets y smartphones presentaba elementos superpuestos, botones inaccesibles y secciones que requerían desplazamiento horizontal, dificultando su uso desde estos dispositivos.

Figura 6. *Pantalla de toma de asistencia.*

Figura 7. *Pantalla de selector de justificación de inasistencia.*

Errores sin retroalimentación adecuada

Se reportaron errores recurrentes sin mensajes informativos claros. Por ejemplo, la imposibilidad de guardar la asistencia cuando se cargaban justificaciones no

generaba ningún mensaje de error explícito, dejando al usuario sin certeza sobre si la operación se completó correctamente.

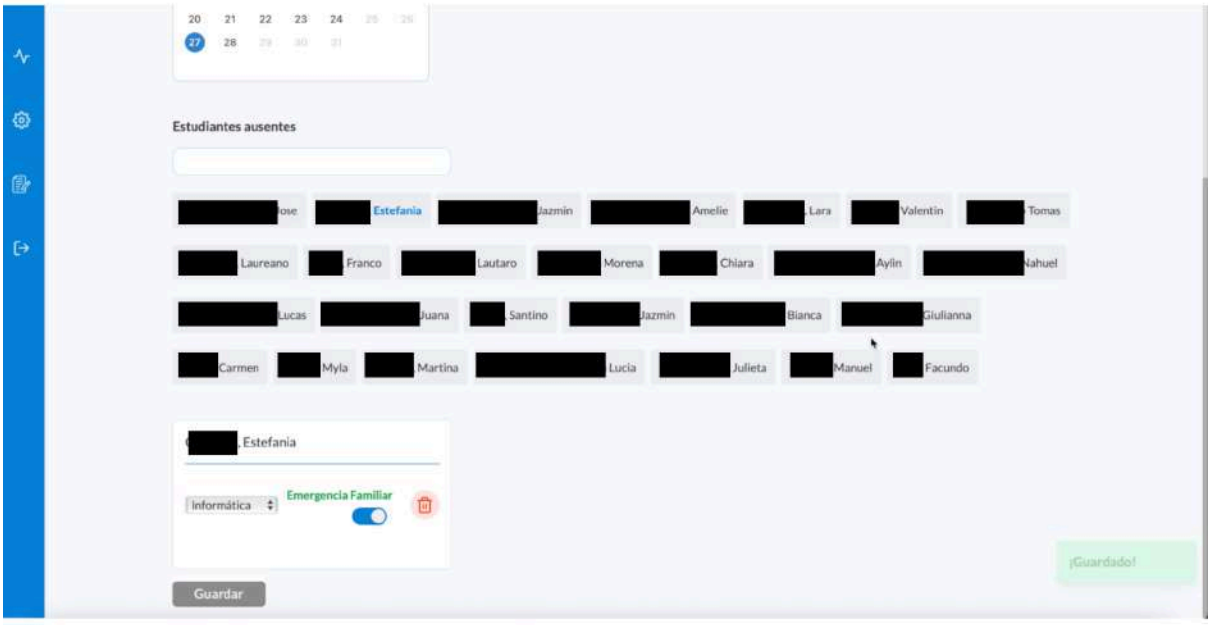


Figura 8. Ausencia de mensajes de error y confirmación en la carga de justificaciones de asistencia.

Funcionalidades no operativas

Diversos botones y enlaces no cumplían su función. El botón de "ver detalle de faltas" dentro de los reportes no ejecutaba ninguna acción al ser presionado. Estos elementos no funcionales incrementaban la frustración de los usuarios y la percepción de baja calidad del sistema.

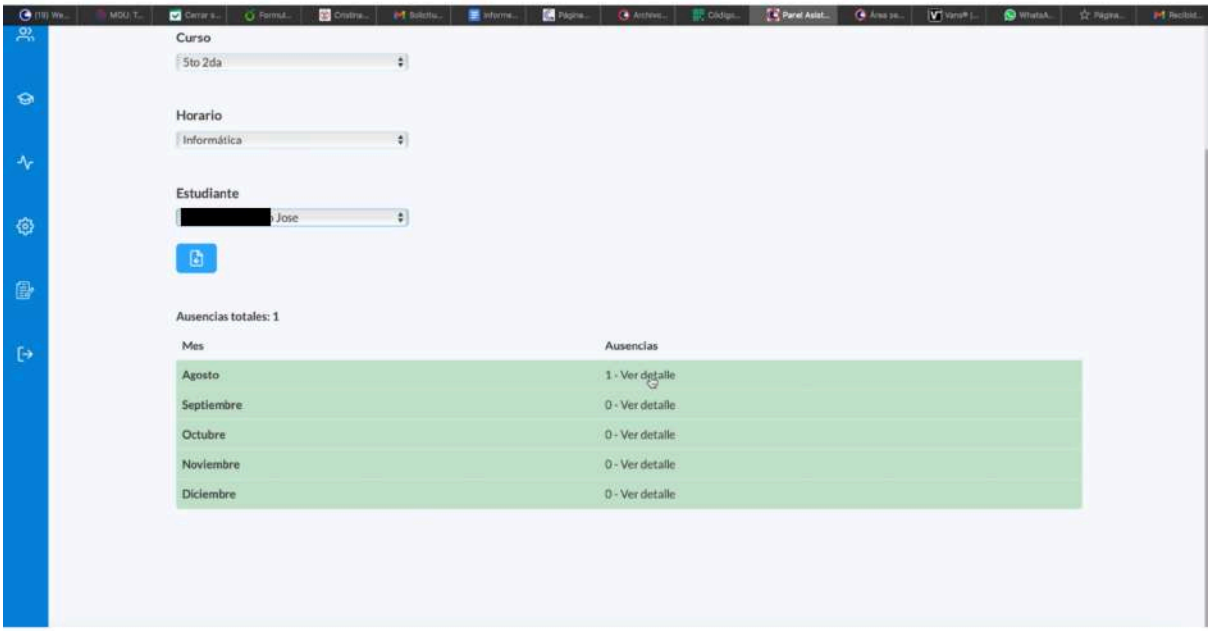


Figura 9. *Pantalla para ver detalle de inasistencias.*

Visualización de información irrelevante

El sistema mostraba información no aplicable al contexto del usuario, como horarios de cursos no asignados o materias de otros niveles educativos. Esta sobrecarga informativa incrementaba la carga cognitiva y dificultaba la localización de la información relevante.

Problemas de rendimiento y confiabilidad

Rendimiento deficiente

La aplicación presentaba tiempos de respuesta prolongados en operaciones rutinarias. La carga de páginas con listados de estudiantes, la consulta de asistencias históricas o la generación de reportes experimentaban demoras significativas, afectando la productividad de los usuarios.

Errores en notificaciones

El sistema de notificaciones presentaba fallas frecuentes, con mensajes que no se enviaban o llegaban con información incorrecta. Esta situación deterioraba la confianza en el sistema y obligaba a los usuarios a verificar información mediante canales alternativos.

Ausencia de restricciones temporales

No existían mecanismos para limitar acciones sensibles según el período académico. Por ejemplo, los docentes podían editar calificaciones de cuatrimestres finalizados sin restricciones, cuando según las normativas institucionales dichas modificaciones deberían estar bloqueadas o requerir autorización especial.

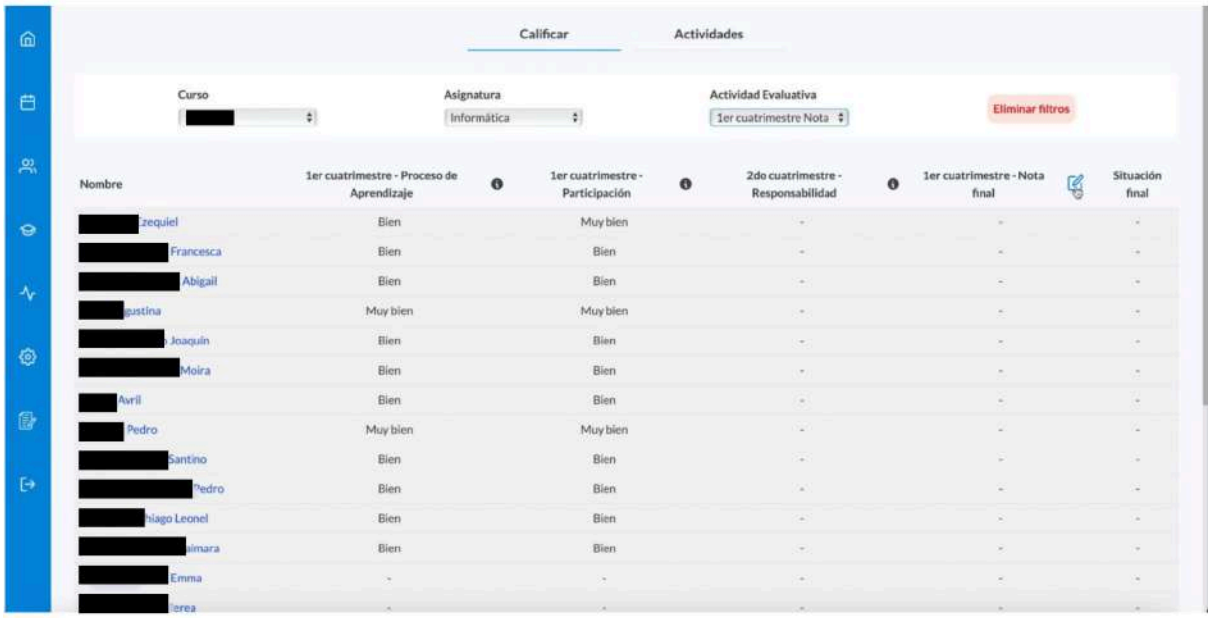


Figura 10. *Pantalla de edición de notas que permite modificar calificaciones del primer cuatrimestre durante el segundo cuatrimestre.*

Problema a resolver

El sistema informático utilizado por el Colegio Nacional Dr. Arturo Umberto Illia presentaba múltiples deficiencias que afectaban la gestión diaria de la información académica y administrativa. Las limitaciones identificadas abarcaban tres dimensiones críticas:

Limitaciones técnicas:

- Inestabilidad del sistema con errores frecuentes durante su operación

- Modelado inadecuado de las relaciones entre entidades académicas

- Rendimiento deficiente en operaciones rutinarias

- Ausencia de mecanismos de auditoría y trazabilidad

Cobertura funcional insuficiente:

- Funcionalidades incompletas para el modelo académico del Colegio Illia

- Imposibilidad de representar estructuras académicas específicas (grupos, talleres, IATE)

- Carencia de reportes institucionales confiables

- Generación manual de boletines y analíticos fuera del sistema

Problemas de usabilidad:

- Interfaz confusa y poco intuitiva

- Falta de diseño responsive para dispositivos móviles

- Flujos de trabajo complejos e innecesariamente extensos

- Ausencia de retroalimentación clara sobre el resultado de las operaciones

La ausencia de centralización generaba procesos fragmentados, incrementaba la probabilidad de errores por transcripción manual, y retrasaba el acceso a información crítica para la toma de decisiones académicas y administrativas. Esta situación motivó la necesidad de desarrollar una solución integral que superara estas limitaciones mediante una arquitectura moderna, un modelado correcto del dominio y una interfaz centrada en la experiencia del usuario.

Conclusión del análisis

Las problemáticas identificadas evidencian que el sistema preexistente no satisfacía adecuadamente las necesidades académicas, administrativas y de seguridad del Colegio Nacional Dr. Arturo Umberto Illia. Las deficiencias en el modelado de datos,

la gestión de roles, la experiencia de usuario y los mecanismos de seguridad justifican el desarrollo de una nueva plataforma que contemple:

- Modelado correcto de las entidades del dominio académico

- Gestión explícita de roles con contexto de operación

- Interfaz responsive centrada en la experiencia del usuario

- Mecanismos robustos de seguridad, autenticación y auditoría

- Trazabilidad completa de operaciones críticas

- Generación automática y confiable de reportes institucionales

El proyecto busca superar estas deficiencias mediante una plataforma digital integral, confiable y accesible, que consolide todas las funciones académicas y administrativas en un único entorno, disponible en tiempo real y alineado con las normativas específicas del Colegio Illia.

Análisis FODA

Para comprender mejor el contexto del proyecto y fundamentar las decisiones de diseño y desarrollo del sistema de gestión académica, se realizó un análisis FODA (fortalezas, oportunidades, debilidades y amenazas).

El FODA proporciona una visión estratégica que ayuda a priorizar objetivos, anticipar riesgos y aprovechar ventajas, asegurando que el proyecto responda de manera efectiva a las necesidades específicas del Colegio Nacional Dr. Arturo Umberto Illia.

Fortalezas

● El referente funcional es una persona con conocimientos técnicos y que posee un entendimiento pleno sobre los procesos del colegio.

● El demandante es un ente local, por lo que los trabajos de relevamiento y acceso al campo del usuario son factibles.

● La aplicación contará con licencia MIT para que distintas instituciones educativas, con requisitos similares al Illia, puedan utilizarla y mantenerla.

Oportunidades

● Inexistencia de una solución de esta índole que cumpla con todos los requisitos necesarios para el correcto funcionamiento del colegio.

● La preexistencia de una aplicación evidencia la necesidad de los usuarios de tener un sistema para la gestión académica.

● Insatisfacción de los usuarios con la app actual, lo que abre la posibilidad de posicionar una nueva aplicación con mejoras concretas, identificando los puntos de falla, los aspectos a mejorar y las funcionalidades deseadas por los usuarios.

● Apoyo de la dirección al cambio organizacional.

Debilidades

● Inexperiencia del equipo de desarrollo en un proyecto real de este tamaño.

● Durante la primera mitad del desarrollo del proyecto, la mayoría del equipo de trabajo se encuentra cursando varias materias por semana, lo que implica una disponibilidad horaria reducida.

● Dificultad para coordinar reuniones con la frecuencia deseada con el referente funcional.

● La solución es muy específica para el colegio Illia, por lo que solo será aplicable en aquellos casos con requerimientos similares.

Amenazas

● Cambios de legislación que puedan tener implicancias en el funcionamiento interno del colegio durante el desarrollo.

● Posible resistencia al cambio por parte de los usuarios debido a malas experiencias con la aplicación anterior.

Requerimientos

A continuación, se listan todos los requerimientos del sistema.

Requerimientos funcionales

1.0 Generales

- **RF-1:** El sistema debe permitir CRUD de los siguientes:
    - Materias
        - Deben tener: nombre, días y horarios.
    - Usuarios (administradores, tutores, estudiantes, preceptores, superpreceptores, docentes, directivos, departamento estudiantes)
        - Deben tener: Nombre, apellido, DNI, contraseña, y email.
    - Cursos
        - Deben tener: nombre
    - Talleres
        - Deben tener: nombre, aula(s), días y horarios.
    - Anuncios
        - Deben tener: título, descripción, fecha y hora inicio, fecha y hora fin.
- **RF-2:** El sistema debe permitir vincular una materia, un curso y uno o más profesores.
- **RF-3:** El sistema debe permitir vincular un preceptor y un curso.
- **RF-4:** Los estudiantes que hayan egresado y no adeuden materias, deben quedar inhabilitados del sistema.
- **RF-5:** El sistema debe permitir almacenar de forma permanente los datos de los usuarios.

2.0 Asistencia

- **RF-6:** El sistema debe permitir tomar y editar asistencia a los cursos que les correspondan.

- **RF-7:** El sistema debe permitir registrar una observación de los estudiantes que figuran como presentes pero no se encuentran en su materia.
- **RF-8:** El sistema debe permitir los siguientes tipos de inasistencia
    - Ausente
    - Presente
    - Llegada tarde
    - Retiro temprano
- **RF-9:** El sistema debe permitir asignar más de un tipo de inasistencia en un mismo día. Por ejemplo, un estudiante puede ausentarse en el turno *y* en una materia de contraturno.
- **RF-10:** El sistema debe permitir agregar una justificación a las faltas.
- **RF-11:** El sistema debe permitir registrar la asistencia de un día anterior al actual.
- **RF-12:** El sistema debe permitir visualizar el historial de asistencia de cada curso para el ciclo lectivo corriente.
- **RF-13:** El sistema debe permitir buscar la inasistencia de cada curso por día.

3.0 Actividades Evaluativas

- **RF-14:** El sistema debe permitir registrar actividades evaluativas para un curso y materia que les corresponda, indicando:
    - Fecha de finalización
    - Título
    - Descripción
- **RF-15:** El sistema debe permitir modificar los campos de una actividad evaluativa.
- **RF-16:** Las actividades evaluativas pueden tener nota tipo numérica o conceptual.

- **RF-17:** El sistema debe permitir crear y modificar una escala de calificación según corresponda, indicando un nombre para la escala y las opciones de calificación.
- **RF-18:** El sistema debe permitir cargar y modificar calificaciones de una actividad evaluativa.
- **RF-19:** El sistema debe llevar registro de los usuarios que carguen o modifiquen notas de una actividad evaluativa.
- **RF-20:** El sistema debe permitir que sólo los estudiantes que desaprobaron durante la cursada puedan rendir en las mesas que les correspondan (diciembre en caso de desaprobar un solo cuatrimestre, febrero en caso de desaprobar ambos o desaprobar en diciembre).
- **RF-21:** El sistema debe contemplar que la calificación obtenida en la instancia de diciembre sustituye la nota del cuatrimestre desaprobado. En caso de aprobación en dicha instancia, la calificación se promediará con la nota del cuatrimestre previamente aprobado. Si el estudiante no aprueba o se encuentra ausente, se habilitará la instancia de febrero. La calificación obtenida en febrero constituirá la nota final de la asignatura, sustituyendo las calificaciones de ambos cuatrimestres.
- **RF-22:** El sistema debe permitir cargar las notas tanto con coma, como con punto. (7.5 | 7,5)
- **RF-23:** El sistema debe permitir configurar la nota de aprobación, siendo la misma para todas las materias.
- **RF-24:** El sistema debe permitirle a un docente registrar una actividad evaluativa en conjunto con otra materia, es decir que una copia de la actividad evaluativa va a crearse en otra materia.
- **RF-25:** El sistema debe generar automáticamente actividades evaluativas obligatorias para todas las materias, según corresponda. Por ejemplo, que una materia tenga por defecto, nota final de primer cuatrimestre, de segundo cuatrimestre, diciembre y marzo.

4.0 Reportes

- **RF-26:** El sistema debe permitir generar reportes:
    - Boletines - Individuales y con las materias del plan de estudios del colegio
    - Notas de un curso en una materia
        - Estudiante
        - Notas para cada actividad evaluativa
        - Debe estar dividido por cuatrimestre
    - Analíticos en base al plan de estudios provincial
- **RF-27:** El sistema debe permitir poder visualizar las materias del plan de estudios nacional (el que utiliza el colegio) con su equivalencia del plan de estudios a nivel provincial.
- **RF-28:** El sistema debe generar boletines en base al plan de estudios del CNAI con:
    - Ciclo lectivo
    - Orientación y curso
    - Nombre y DNI del estudiante
    - Materias y calificaciones

5.0 Roles

- **RF-29:** El sistema debe permitir a todos los usuarios visualizar los anuncios.
- **RF-30:** El sistema debe permitir a todos los usuarios ver las ausencias de los profesores.
- **RF-31:** El sistema debe permitir que un usuario pueda cambiar de rol según los que tenga asignados, por ejemplo un docente que también es padre.
- **RF-32:** El sistema debe permitir que si un usuario cambia de rol, realice las funcionalidades en base al elegido.

5.1 Administrador

- **RF-33:** El sistema debe permitir al administrador asignar roles a los usuarios
- **RF-34:** El sistema debe permitir al administrador crear, visualizar, editar y eliminar anuncios.
- **RF-35:** El sistema debe permitir al administrador crear, visualizar, editar y eliminar usuarios.
- **RF-36:** El sistema debe permitir al administrador restablecer la contraseña de acceso de un usuario.
- **RF-37:** El sistema debe permitir al administrador realizar un CRUD de materias
- **RF-38:** El sistema debe permitir al administrador realizar un CRUD de cursos
- **RF-39:** El sistema debe permitir al administrador realizar un CRUD de talleres
- **RF-40:** El sistema debe permitir al administrador ejecutar el inicio de un nuevo ciclo lectivo.
- **RF-41:** El sistema debe permitir al administrador tomar y editar asistencias de un curso
- **RF-42:** El sistema debe permitir al administrador gestionar (crear y editar) los Informes de Avance de las Trayectorias Educativas (IATE), obligatorios para todas las materias.

5.2 Docente

- **RF-43:** El sistema debe permitir al docente visualizar sus horarios.
- **RF-44:** El sistema debe permitirle a un docente cargar notas de las actividades evaluativas obligatorias (Nota de cuatrimestre, IATE, diciembre, marzo)
- **RF-45:** El sistema debe permitirle a un docente generar reportes de una materia que tenga a cargo.

- **RF-46:** El sistema debe permitirle a un docente registrar una observación de los estudiantes que figuran como presentes pero no se encuentran en su materia.
- **RF-47:** El sistema debe permitir registrar cuál fue el profesor encargado de cargar o cambiar la nota de una actividad evaluativa.
- **RF-48:** El sistema debe permitirle a un docente ver las inasistencias de un estudiante en su materia.

5.3 Preceptor

- **RF-49:** El sistema debe permitir al preceptor visualizar los profesores ausentes.
- **RF-50:** El sistema debe permitir a un preceptor tomar y editar asistencia de los cursos que tenga a cargo.
- **RF-51:** El sistema debe permitir a un preceptor elegir una materia del curso que tenga a cargo y acceder a las notas y asistencias.
- **RF-52:** El sistema debe permitir a un preceptor tomar y editar asistencia de un día anterior de los cursos que tenga a cargo.
- **RF-53:** El sistema debe permitir a un preceptor visualizar el historial de asistencias de los cursos que tenga a cargo.
- **RF-54:** El sistema debe permitir registrar cuál fue el preceptor encargado de tomar o modificar la asistencia en un curso.

5.4 Superpreceptor

- **RF-55:** El sistema debe permitir al superpreceptor visualizar los profesores ausentes.
- **RF-56:** El sistema debe permitir a un superpreceptor tomar y editar asistencia de todos los cursos.
- **RF-57:** El sistema debe permitir a un superpreceptor seleccionar una materia de cualquier curso y acceder a las notas y asistencias.

- **RF-58:** El sistema debe permitir a un superpreceptor tomar y editar asistencia de un día anterior de cualquier curso.
- **RF-59:** El sistema debe permitir a un superpreceptor visualizar el historial de asistencias de todos los cursos.
- **RF-60:** El sistema debe permitir a los superpreceptores visualizar qué usuario tomó asistencia en cualquier curso.
- **RF-61:** El sistema debe permitir a un superpreceptor cargar, modificar y eliminar ausencias docentes.
- **RF-62:** El sistema debe permitirle a un superpreceptor cargar, modificar y eliminar anuncios.

5.5 Estudiante / Tutor

- **RF-63:** El sistema debe permitir al usuario visualizar las ausencias docentes registradas, tanto del día en curso como en fechas futuras.
- **RF-64:** El sistema debe permitir al usuario visualizar sus actividades evaluativas.
- **RF-65:** El sistema debe permitir al usuario visualizar sus horarios.
- **RF-66:** El sistema debe permitir al usuario visualizar su historial de inasistencias.
- **RF-67:** El sistema debe permitir a un tutor elegir la información de cuál de sus tutelados ver.

5.6 Directivo

- **RF-68** El sistema debe permitir a los directivos visualizar qué usuario tomó asistencia para cada curso.
- **RF-69:** El sistema debe permitir a un directivo visualizar el historial de asistencias de cada curso.
- **RF-70:** El sistema debe permitir a un directivo visualizar calificaciones de todos los estudiantes del sistema.

- **RF-71:** El sistema debe permitir a un directivo cargar, modificar y eliminar anuncios.
- **RF-72:** El sistema debe permitir a un directivo generar reportes y analíticos.

5.7 División Estudiantes

- **RF-73:** El sistema debe permitir a los usuarios de División Estudiantes generar reportes y analíticos de todos los cursos.
- **RF-74:** El sistema debe permitir a los usuarios de División Estudiantes visualizar el historial de asistencias de todos los cursos.
- **RF-75:** El sistema debe permitir a los usuarios de División Estudiantes visualizar para todos los cursos:
    - Materias
    - Estudiantes
    - Actividades evaluativas

6.0 Cambio de año

- **RF-76:** El sistema debe permitir cambiar de curso automáticamente a los estudiantes que adeuden menos de 2 materias y/o talleres al finalizar cada ciclo lectivo.
- **RF-77**: El sistema debe distinguir cuando un estudiante no tenga aprobadas 3 o más materias y/o talleres, pero aun así pasarlos de año.

7.0 Seguridad

- **RF-78:** El sistema debe permitir cambiar la contraseña.
- **RF-79:** El sistema debe obligar a los usuarios a cambiar su contraseña luego del primer inicio de sesión.
- **RF-80:** El sistema debe obligar a los usuarios que tengan mayores permisos a tener contraseñas más seguras.

- **RF-81:** Al cambiar la contraseña, no debe permitirse que la nueva sea igual a la anterior.

8.0 Cambios de curso/orientación

- **RF-82:** El sistema debe permitir cambiar de curso (u orientación) a los estudiantes. En caso de que un estudiante cambie de orientación, debe permitirle al administrador seleccionar qué materias son equivalentes con cuáles. Es decir, que debe seleccionar a qué orientación se va a pasar, y seleccionar qué materias “equivalen” para que el nuevo docente, pueda tener acceso a las notas anteriores del estudiante.
- **RF-83:** El sistema debe permitir cargar las notas de las equivalencias rendidas. En caso de que un estudiante haya deseado cambiar de orientación.

Requerimientos no funcionales

- **RNF-01:** La aplicación deberá ser una *webapp* desarrollada con Next.js, garantizando compatibilidad con los navegadores modernos.
- **RNF-02:** El sistema deberá ser responsive, asegurando un correcto funcionamiento y visualización en computadoras, tablets y dispositivos móviles.

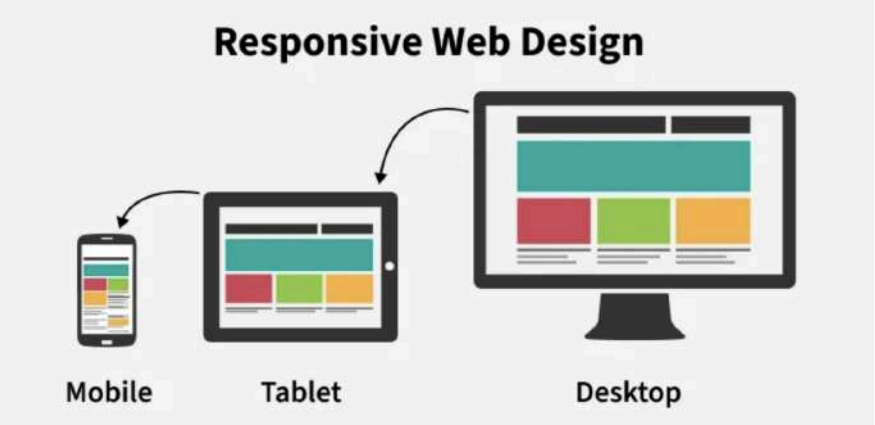


Figura 11. *Representación del concepto de diseño responsive, mostrando la adaptación de la interfaz a dispositivos móviles, tablets y escritorio.*

- **RNF-03:** La aplicación será *hosteada* en el servidor del CNAI, garantizando disponibilidad dentro del entorno institucional.
- **RNF-04:** La interfaz de usuario deberá ser amigable y atractiva, facilitando la navegación y minimizando la curva de aprendizaje.
- **RNF-05:** El sistema deberá implementar un control de acceso basado en roles, asegurando que cada usuario acceda únicamente a la información y funciones permitidas según su perfil.
- **RNF-06:** Los datos sensibles, como las contraseñas de los usuarios, deberán almacenarse en forma encriptada mediante un algoritmo seguro, garantizando la confidencialidad y protección de la información.

Benchmarking / análisis de la competencia

Para definir la estrategia más adecuada para el desarrollo del sistema de gestión académica del Colegio Nacional Dr. Arturo Umberto Illia, se analizaron distintas alternativas disponibles y su capacidad de adaptarse a las necesidades específicas de la institución.

1. Desarrollo a medida

El desarrollo de un sistema a medida se presentó como la opción más flexible y escalable para el Illia. Permite diseñar funcionalidades que respeten las normas y particularidades del colegio, así como adaptarse rápidamente a posibles cambios futuros en reglamentaciones o procesos internos. A largo plazo, esta opción garantiza independencia frente a terceros y control total sobre las actualizaciones y mejoras del sistema.

2. Plataformas adaptables (Open Source)

Se evaluaron varias plataformas de código abierto que permiten personalización parcial, aunque con limitaciones en ciertos procesos académicos no estándar:

- Moodle con *plugins*
    - Registro de notas, asistencia y gestión de usuarios con roles.

    - Adaptación a gestión de familias mediante cuentas relacionadas.
    - Limitada para cambios de curso u orientación sin configuraciones complejas.
    - Extensible mediante módulos personalizados (eventos, noticias, etc.).
- Fedena
    - Registro de asistencia, calificaciones y cambios de curso.
    - Incluye módulos de anuncios y cuentas para padres.
    - La versión gratuita tiene funcionalidades limitadas; la versión completa es de pago.
    - Requiere instalación y mantenimiento en un servidor propio.

3. Servicios privados: Acadeu

Acadeu es una plataforma privada de gestión escolar que ofrece funcionalidades estándar para la administración académica, incluyendo registro de calificaciones, asistencia, comunicación institucional y acceso para familias. Al tratarse de un servicio propietario, presenta limitaciones en cuanto a personalización y adaptación a procesos específicos del colegio. Asimismo, implica un costo de suscripción y una dependencia del proveedor para la implementación de cambios, integraciones o mejoras futuras. Estas características lo vuelven menos adecuado frente a una solución desarrollada a medida que pueda ajustarse completamente a las particularidades organizativas y académicas del Colegio Illia.

4. Ecosistema SIU

El Colegio Illia utiliza actualmente SIU Mapuche, sistema perteneciente al ecosistema del Consorcio SIU orientado a la gestión de recursos humanos en universidades nacionales. Durante el análisis inicial se consideró la posibilidad de utilizar alguna solución del consorcio para la gestión académica del colegio.

Sin embargo, estas herramientas están diseñadas principalmente para procesos administrativos universitarios y no contemplan funcionalidades específicas necesarias para el contexto de una institución de nivel secundario, como el registro

detallado de asistencias de estudiantes o mecanismos de comunicación con las familias.

Por este motivo, se descartó su adopción como solución principal para el presente proyecto. No obstante, a futuro podría evaluarse la posibilidad de complementar el sistema desarrollado con alguna solución del ecosistema SIU para aspectos puntuales, como el registro institucional de calificaciones finales, aunque esta alternativa se encuentra fuera del alcance del presente trabajo.

Figura 12. *Soluciones existentes evaluadas como alternativas al desarrollo a medida de IlliApp.*

El CNAI posee características académicas particulares que hacen que los sistemas genéricos resulten insuficientes. Por ejemplo, el registro de faltas contempla múltiples tipos de inasistencias, distintos de los utilizados en otros colegios, y la institución cuenta con instancias de exámenes particulares que requieren un manejo específico dentro del calendario académico. Estas particularidades dificultan la adaptación de plataformas estándar y refuerzan la necesidad de un sistema flexible y a medida.

Considerando la evaluación de las opciones disponibles y las particularidades del colegio, el desarrollo de un sistema a medida resulta la alternativa más adecuada. Esta solución permite reflejar fielmente las reglas internas del CNAI, garantizar la correcta administración de sus procesos académicos y mantener la flexibilidad para adaptarse a cambios futuros en normativas y procedimientos internos. Además,

evita la dependencia de proveedores externos y facilita la integración de mejoras o nuevas funcionalidades según las necesidades de la institución.

Análisis de riesgos

Todo proyecto de desarrollo de software conlleva una serie de riesgos que pueden afectar su planificación, ejecución o mantenimiento. Identificarlos de manera temprana permite establecer estrategias de mitigación adecuadas y minimizar su impacto en el resultado final.

Se consideraron tanto los riesgos técnicos como los organizacionales y operativos, teniendo en cuenta las particularidades institucionales, los recursos disponibles y los plazos establecidos.

Cada riesgo se valora según dos dimensiones:

- **Probabilidad (P):** mide la posibilidad de que el riesgo ocurra, en una escala de 1 (baja) a 3 (alta).
- **Impacto (I):** representa la magnitud de las consecuencias en caso de que el riesgo se materialice, también en una escala de 1 (baja) a 3 (alta).

El **peso (P × I)** permite priorizar los riesgos según su criticidad:

- Peso ≤ 3: Riesgo bajo (seguimiento general).
- Peso entre 4 y 5: Riesgo medio (monitoreo periódico).
- Peso ≥ 6: Riesgo alto (requiere plan de contingencia específico).

Este enfoque permite concentrar los esfuerzos de mitigación en aquellos factores con mayor potencial de afectar el proyecto, asegurando una gestión preventiva y eficiente.
En la siguiente tabla se detallan los principales riesgos identificados para el desarrollo del sistema de gestión académica del Colegio Nacional Dr. Arturo Umberto Illia, junto con su evaluación y las estrategias de mitigación propuestas.

Riesgo	Descripción	Consecuencia	Prob.	Impacto	Peso
R01. Complejidad del dominio institucional	Las particularidades del Illia (tipos de faltas, exámenes institucionales especiales, entre otros) pueden generar dificultades al modelar correctamente los procesos en el sistema.	Demoras en los tiempos de entrega.	2	3	**6**
R02. Cambios normativos o legislativos	Modificaciones en la normativa educativa o en las disposiciones internas del Illia podrían requerir ajustes en el funcionamiento del sistema.	Necesidad de retrabajo, demoras en los tiempos de entrega, frustración del equipo.	1	3	3
R03. Resistencia al cambio de los usuarios	Algunos usuarios pueden mostrarse reticentes a utilizar el nuevo sistema debido a experiencias negativas con la aplicación anterior.	Conflictos internos, poco uso del sistema.	2	2	4
R04. Dificultad de coordinación con el referente funcional	La imposibilidad o dificultad para coordinar reuniones con el referente funcional del colegio puede afectar la comunicación y la correcta validación de los módulos desarrollados, generando incertidumbre sobre los	Retrasos en la validación de funcionalidades, posibles malentendidos sobre los requerimientos, incremento del retrabajo y frustración en el	3	2	**6**

	requerimientos del proyecto.	equipo de desarrollo.			
R05. Cambio de las autoridades directivas de la UNMdP	La nueva gestión podría introducir cambios en las prioridades institucionales, solicitar modificaciones al sistema o generar demoras en validaciones y aprobaciones, así como posibles resistencias al cambio en la implementación del proyecto.	Realización de nuevas instancias de presentación y validación del proyecto ante las nuevas autoridades, así como la revisión o reafirmación de los acuerdos previamente establecidos.	2	2	**4**

Aquellos riesgos cuyo peso resultó igual o superior a 6 representan situaciones críticas que podrían afectar significativamente el desarrollo o la adopción del sistema.

Con el fin de mitigar sus efectos y garantizar la continuidad del proyecto, se elaboró un plan de contingencia específico para cada uno de estos riesgos. Este plan establece acciones preventivas para reducir la probabilidad de ocurrencia y medidas de contingencia para minimizar las consecuencias en caso de que el riesgo se materialice.

Plan de contingencia para R01: Complejidad del dominio institucional

Dado que las particularidades del CNAI, como los tipos de faltas y los exámenes especiales, pueden generar dificultades al modelar correctamente los procesos en el sistema, se implementará un plan de contingencia que combine la prevención y la acción correctiva. Para reducir la probabilidad de error, se mantendrán reuniones periódicas con el referente funcional para validar cada módulo desarrollado, se documentarán detalladamente todos los procesos y reglas específicas del colegio, y

se desarrollarán prototipos tempranos para asegurar la alineación con los requerimientos reales. En caso de que surjan problemas de modelado que afecten los tiempos de entrega, se ajustará el cronograma, se implementarán módulos piloto para validar los procesos antes de su despliegue final y se designará un responsable del dominio institucional para consultas rápidas y resolución de dudas.

Plan de contingencia para R04: Dificultad de coordinación con el referente funcional

La imposibilidad o dificultad para coordinar reuniones con el referente funcional del colegio puede afectar la comunicación y la validación de módulos, generando retrasos y malentendidos sobre los requerimientos. Como medidas preventivas, se establecerán horarios fijos de reunión, se realizará una planificación anticipada de los puntos a tratar y se hará uso de herramientas de comunicación asincrónica para seguimiento de pendientes. En caso de que surjan dificultades de coordinación, se reprogramarán reuniones priorizando las validaciones críticas, se mantendrá un registro detallado de acuerdos y decisiones para evitar malentendidos y se implementarán revisiones parciales de módulos aunque no sea posible realizar reuniones presenciales.

Análisis de costos

La viabilidad económica de IlliApp en su etapa productiva resulta altamente favorable, dado que los costos operativos y recurrentes se consideran marginales para el ecosistema tecnológico actual de la institución. Este escenario se fundamenta en una estrategia de aprovechamiento de la infraestructura y los recursos humanos preexistentes, la cual se detalla en los siguientes ejes:

- Infraestructura: El despliegue del sistema se realizará en un entorno de servidor compartido con la infraestructura que el colegio ya posee y administra. Previo a esta definición, se llevó a cabo un análisis de capacidad estructural, el cual concluyó que el servidor actual dispone de los recursos de cómputo suficientes para ejecutar la aplicación de manera óptima, sin degradar el rendimiento ni afectar la disponibilidad de los servicios preexistentes.

- Licenciamiento de Software: Toda la arquitectura del sistema ha sido desarrollada utilizando exclusivamente herramientas y tecnologías de código abierto. Esta decisión de diseño garantiza la total ausencia de costos ocultos o periódicos asociados al pago de licencias comerciales o de terceros.
- Esfuerzo Operativo y Mantenimiento: El soporte técnico y operativo será absorbido por el personal con el que ya cuenta el colegio. En términos de horas-hombre, la carga operativa no será constante: se concentrará de manera predecible y acotada al inicio de cada ciclo lectivo, momento en el cual es necesaria la carga de alumnos, cursos y talleres. Durante el resto del ciclo escolar, el tiempo de mantenimiento requerido dependerá estrictamente de la incidencia eventual de errores o de la solicitud de nuevos requerimientos funcionales, conformando una carga de trabajo asumible que no exige la contratación de personal externo.
- Gestión de Dominios y Certificados: La provisión del dominio de la aplicación, así como la gestión, configuración y renovación de los certificados de seguridad, son procesos administrados de forma centralizada por la Universidad. Por consiguiente, estos activos de red ya están cubiertos y no representan una erogación directa para este proyecto.

Diseño del sistema

El diseño del sistema constituye una etapa fundamental del desarrollo del proyecto, ya que define la estructura, los componentes y los procesos necesarios para cumplir los objetivos del Colegio Nacional Dr. Arturo Umberto Illia. En esta sección se describen las metodologías aplicadas en las fases de análisis, diseño, desarrollo, pruebas e implementación, así como la arquitectura general del sistema y las tecnologías empleadas en su construcción.

Se presenta además el diseño del modelo de datos, con la selección de tecnologías, la estructura de la base de datos y los diagramas entidad-relación (DER) que muestran las relaciones entre los distintos elementos del sistema. La integración de metodologías y herramientas garantiza un desarrollo ordenado y eficiente,

asegurando escalabilidad, mantenibilidad y la correcta implementación de la solución tecnológica propuesta.

Metodologías utilizadas

Metodologías de análisis

El análisis del sistema comenzó con reuniones iniciales con el Sr. Sebastián Salgueiro, referente funcional del colegio, para recopilar los requerimientos del Colegio Illia. Posteriormente, el equipo refinó estos requerimientos y les otorgó sentido técnico, identificando dudas y detalles adicionales. Cada iteración se revisaba nuevamente con Sebastián para validar el entendimiento y ajustar los elementos que lo requerían. Este proceso se repitió varias veces, hasta asegurar que los requerimientos reflejaran fielmente las necesidades reales del colegio.

Durante la fase de análisis se identificaron las entidades principales y sus interacciones, así como los procesos que era necesario digitalizar y despapelizar. Esta etapa también incluyó el estudio de asistencia y calificaciones, contemplando reglas particulares como el cálculo de inasistencias parciales, la toma de asistencia en turnos y contraturnos, la gestión de incidencias y la configuración de calificaciones finales según las mesas de diciembre y marzo. Se definieron casos especiales como cambios de orientación y la posibilidad de agregar actividades evaluativas fuera de calendario. Además, se documentó la estructura de grupos dentro de las materias, la visualización de anuncios, y la gestión de reportes y boletines, garantizando que cada funcionalidad respondiera a las necesidades del colegio.

Los requerimientos obtenidos sirvieron como base para el diseño de los módulos del sistema. Estos módulos incluyen gestión académica, gestión de usuarios y roles, asistencia, evaluaciones y calificaciones, anuncios, y reportes. Cada módulo fue analizado en detalle para determinar las entidades involucradas, sus interacciones y los flujos de información necesarios para garantizar un funcionamiento correcto y eficiente de la plataforma.

Metodologías de diseño

Durante la etapa de diseño se utilizaron herramientas de modelado para representar las entidades del dominio y las relaciones entre ellas. Estas herramientas permitieron definir de manera precisa el esquema de los datos y visualizar cómo interactúan los distintos componentes del sistema dentro de los procesos institucionales. El diseño de la base de datos se apoyó en estos modelos para garantizar consistencia, trazabilidad y escalabilidad a futuro, que luego fueron validados con el referente funcional.

En paralelo, se definió la arquitectura del sistema. Se optó por una arquitectura monolítica modular, integrando *frontend* y *backend* en una misma aplicación. Esta elección ofreció una estructura unificada del código y simplificó tanto el desarrollo como el despliegue.

La decisión de adoptar esta arquitectura se fundamentó en varios factores:

- Simplicidad en el desarrollo: un único repositorio y entorno de trabajo facilita la coordinación entre los integrantes del equipo.
- Despliegue directo: sin necesidad de manejar múltiples servicios ni infraestructura compleja.
- Reducción de sobrecarga técnica: evitando contenedores y orquestadores, lo que hace más eficiente el trabajo en proyectos con recursos limitados.

La arquitectura implementada se compone de tres partes principales:

- Interfaz de usuario (UI), responsable de gestionar todo lo que el usuario visualiza y con lo que interactúa.
- Aplicación del lado del servidor, que maneja la lógica de negocio y el acceso a los recursos del servidor.
- Base de datos relacional, encargada de almacenar las entidades del sistema y sus relaciones, organizadas en múltiples tablas según las necesidades del dominio.

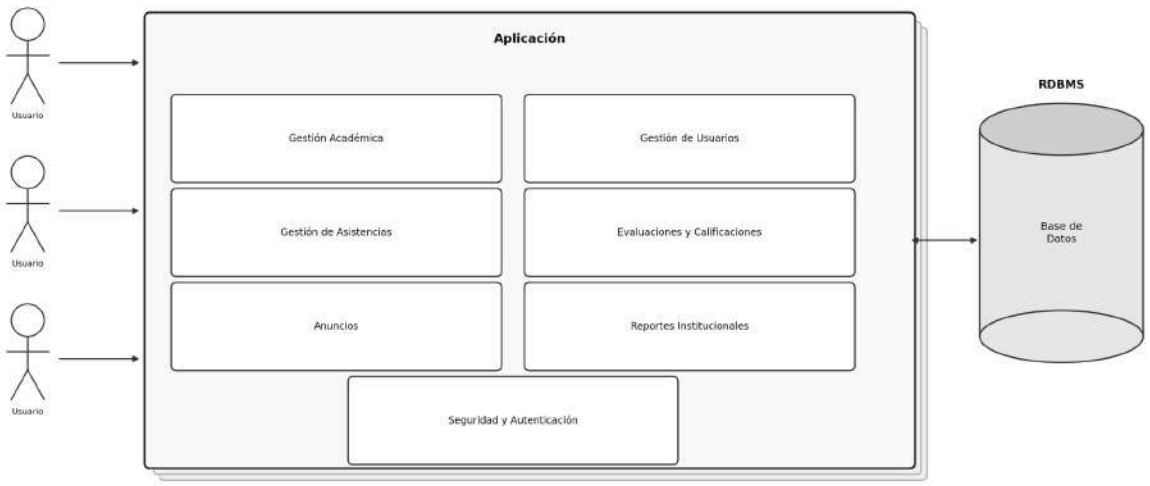


Figura 13. *Arquitectura monolítica modular del sistema IlliApp.*

Además, el diseño se abordó por módulos, de acuerdo con las funcionalidades principales del sistema: seguridad y autenticación, gestión académica, usuarios y roles, asistencia, evaluaciones y calificaciones, anuncios, y reportes.

La adopción de una arquitectura de microservicios fue descartada, ya que habría introducido una complejidad innecesaria en términos de infraestructura, despliegue y mantenimiento, sin aportar beneficios significativos para el alcance del sistema. La modularización dentro de un único monolito facilitó la organización del código, promovió la reutilización de componentes y mejoró la mantenibilidad general. Cada módulo posee responsabilidades claramente definidas y puede evolucionar de manera independiente sin afectar el funcionamiento del resto de la aplicación.

Metodologías de desarrollo

El desarrollo del sistema se llevó a cabo de manera iterativa e incremental, priorizando una construcción progresiva del producto y la validación continua con el referente funcional del colegio.

En una primera etapa se elaboró un prototipo mínimo que incluyó los módulos de ausencias docentes y anuncios. Esta versión inicial permitió validar los primeros

flujos del sistema, así como la disposición general de la interfaz. Luego de presentar el prototipo al referente funcional y recibir su aprobación respecto al diseño y funcionamiento, se continuó con el desarrollo del resto de los módulos, manteniendo la misma lógica de avance gradual y validación continua.

El equipo trabajó con una organización basada en iteraciones semanales. En cada reunión, realizada de forma online, se revisaban los avances, se discutían los inconvenientes detectados, se tomaban decisiones técnicas y se dividían las nuevas tareas. Paralelamente, se mantuvo una bitácora de seguimiento donde se registraban los temas tratados en cada encuentro, los progresos alcanzados, los problemas surgidos y las soluciones implementadas.

Además, la documentación del proyecto se elaboró de manera paralela al desarrollo, lo que permitió mantener la trazabilidad de las decisiones y garantizar la coherencia entre el diseño y la implementación.

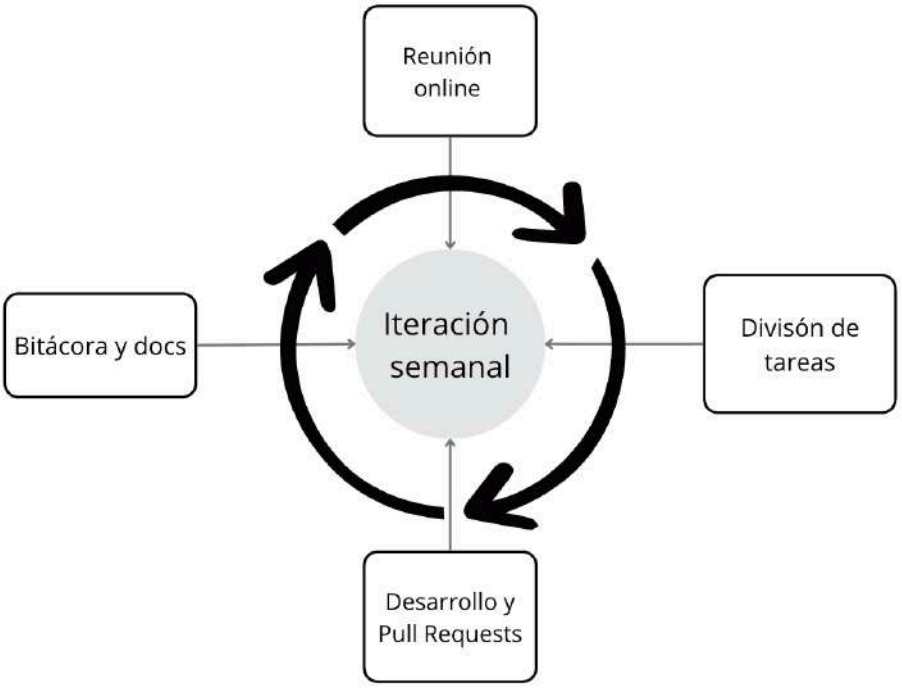


Figura 14. *Ciclo de desarrollo iterativo semanal aplicado durante el proyecto.*

Para la gestión del trabajo se utilizaron diversas herramientas de apoyo. Trello permitió organizar las tareas mediante tableros Kanban, brindando visibilidad sobre el estado de cada actividad y el responsable correspondiente. Git se empleó como sistema de control de versiones y GitHub como plataforma de alojamiento remoto

del repositorio. Cada integrante debía generar solicitudes de cambio (pull requests) revisadas por otro miembro del equipo antes de su aprobación, lo que favoreció la detección temprana de errores y el control de calidad. Mediante GitHub Actions se automatizaron tareas de verificación y formato, optimizando el proceso de revisión. Finalmente, Google Drive se utilizó como repositorio común para almacenar documentación, modelos, esquemas, registros de horas y otros recursos relevantes para el proyecto.

Metodologías de testing

La estrategia de testing adoptada en el proyecto se basó en un enfoque de múltiples niveles, combinando pruebas manuales y automatizadas con el objetivo de garantizar la calidad, confiabilidad y correcto funcionamiento del sistema desarrollado. Si bien se implementaron pruebas unitarias y end-to-end, el eje principal del proceso de validación estuvo puesto en el testing manual y en las pruebas cruzadas entre los integrantes del equipo.

Testing Manual y Pruebas Cruzadas

Las instancias de testing manual constituyeron el mecanismo de validación más utilizado durante el desarrollo. Estas pruebas estuvieron orientadas a la detección de errores de usabilidad, inconsistencias funcionales y comportamientos no previstos en escenarios reales de uso.

Se adoptó una metodología de pruebas cruzadas entre los integrantes del equipo, en la cual cada desarrollador validaba módulos implementados por otros miembros. Esta dinámica permitió incorporar una mirada externa sobre cada funcionalidad, facilitando la detección de problemas que podrían pasar inadvertidos para el autor original y mejorando la robustez general del sistema.

Testing Unitario

Para la validación de componentes individuales del sistema se implementaron pruebas unitarias utilizando el *framework* Vitest. Estas pruebas se enfocaron en verificar el comportamiento correcto de funciones y módulos críticos, particularmente

aquellos relacionados con la lógica de negocio, autenticación, validaciones y operaciones sobre la base de datos.

Se utilizaron mecanismos de *mockeo* de dependencias externas, como librerías de encriptación y acceso a datos, con el objetivo de aislar las unidades bajo prueba y asegurar que los resultados obtenidos dependieran únicamente de la lógica evaluada. Asimismo, se configuró un entorno de ejecución específico para *Node.js* y se incorporaron herramientas de medición de cobertura de código, permitiendo identificar áreas no testeadas y mejorar progresivamente la calidad del software.

Testing End-to-End

Para validar el comportamiento del sistema desde la perspectiva del usuario final se implementaron pruebas *end-to-end* mediante el *framework* Playwright. Estas pruebas simulan interacciones reales sobre la interfaz web, incluyendo procesos completos como autenticación de usuarios, navegación entre módulos y validaciones de formularios.

Se configuraron ejecuciones controladas para evitar conflictos sobre la base de datos compartida, incluyendo ejecución secuencial de pruebas y reintentos automáticos ante fallos transitorios. Además, se generaron reportes automáticos en formato HTML para facilitar el análisis de resultados.

Con el fin de facilitar la preparación de escenarios de prueba, se desarrollaron utilidades específicas para la gestión de datos de prueba, incluyendo funciones de limpieza y carga inicial de información, respetando las restricciones de integridad referencial definidas en la base de datos.

Figura 15. *Frameworks de testing utilizados en el proyecto.*

Tecnologías y herramientas

En las primeras etapas del proyecto se evaluaron distintas alternativas para definir la plataforma principal del sistema. El equipo analizó la posibilidad de desarrollar aplicaciones nativas para iOS y Android, así como la construcción de una aplicación web accesible desde cualquier dispositivo. Luego de considerar las ventajas y desventajas de cada enfoque, y tras su discusión con el referente funcional, se concluyó que una aplicación web resultaba la opción más adecuada.

La solución se implementó como una aplicación web progresiva (Progressive Web App, PWA), lo que permite que el sistema sea utilizado desde el navegador y, al mismo tiempo, pueda instalarse en dispositivos móviles y tablets, ofreciendo una experiencia similar a la de una aplicación nativa. Esta elección garantiza una mayor accesibilidad para toda la comunidad educativa, reduce la necesidad de mantenimiento de múltiples plataformas y mantiene coherencia con la plataforma preexistente del colegio, que también opera bajo un enfoque web.

A partir de esta definición, se inició el análisis de las tecnologías específicas para los distintos componentes del sistema. Se compararon opciones de *backend*, *frontend*, *frameworks*, sistemas de autenticación y motores de base de datos. Con esa información se seleccionó el conjunto tecnológico definitivo, teniendo en cuenta

criterios como la facilidad de mantenimiento, curva de aprendizaje, compatibilidad con los requerimientos del dominio y proyección futura del sistema.

Backend

Antes de iniciar el desarrollo se llevó a cabo un análisis detallado para definir las tecnologías que permitirían construir un sistema capaz de cubrir la complejidad institucional del Illia. La futura aplicación debía integrar múltiples módulos funcionales, sostener reglas internas particulares y ofrecer estabilidad en un entorno de uso cotidiano. Este escenario obligó a elegir herramientas con soporte sólido, buena documentación y una comunidad activa que facilitara resolver problemas sin retrasos innecesarios.

Desde el comienzo se consideró que el proyecto necesitaba un *framework* que aportara orden, estructura y buenas prácticas. El alcance del sistema superaba con claridad el de un desarrollo simple y resultaba evidente que trabajar sin un marco sólido incrementaría la probabilidad de errores, inconsistencias y retrabajo. Se investigaron varias opciones y, después de un análisis comparativo, se concluyó que la solución más adecuada era Next.js. Su modelo de trabajo unificado para interfaz y lógica del servidor ofrecía una base estable para un proyecto con módulos interrelacionados, además de un enfoque moderno que favorecía la organización interna del código.

Una vez tomada esta decisión, se adoptó como consecuencia el uso del ecosistema Typescript y Node.js. Ninguno de los integrantes del equipo tenía experiencia previa con este entorno, por lo que antes de comenzar el desarrollo se realizaron cursos breves y actividades de estudio destinadas a adquirir los fundamentos necesarios. Este proceso inicial permitió abordar el proyecto con un conocimiento mínimo común y evitó que la falta de familiaridad técnica afectara el avance durante las primeras iteraciones.

Para la interacción con la base de datos se seleccionó Prisma como ORM. Su sistema de migraciones facilitó mantener alineado el diseño de datos con la implementación real y su modelo tipado redujo errores comunes en el acceso a la

información. Esto permitió sostener un esquema de trabajo más seguro y ordenado a lo largo del ciclo de desarrollo.

La combinación de Next.js, Node.js y Prisma estableció un entorno tecnológico coherente y adecuado para afrontar un sistema institucional de esta magnitud, conservando un equilibrio entre facilidad de implementación, solidez técnica y capacidad de evolución futura.

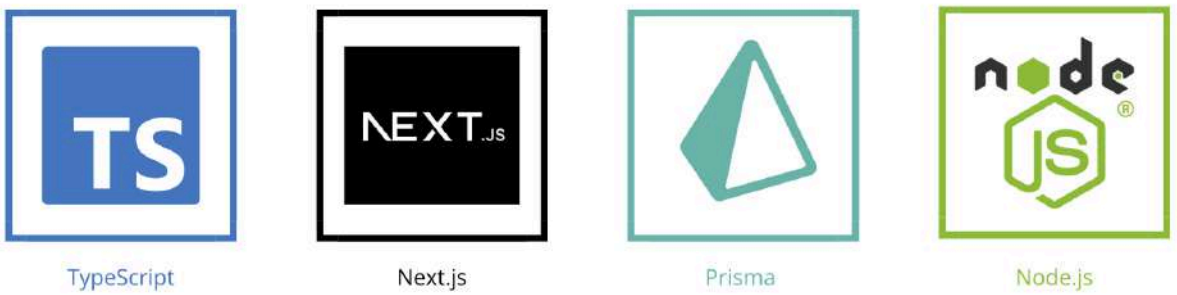


Figura 16. *Stack tecnológico utilizado para el desarrollo del backend de IlliApp.*

Frontend

Para el desarrollo del frontend se adoptó un enfoque basado en componentes mediante el uso de React, lo que permitió estructurar la interfaz de forma modular, reutilizable y coherente con la arquitectura general del sistema. Esta elección ofreció un modelo de trabajo predecible y facilitó la construcción de pantallas dinámicas manteniendo una separación clara entre la lógica de presentación y el resto de las capas de la aplicación.

Con el objetivo de garantizar una interfaz consistente y ordenada, se incorporó una biblioteca de componentes predefinidos. Se seleccionó *shadcn* por presentar un diseño contemporáneo, documentación extensa y una estructura adaptable a distintos contextos. El catálogo de componentes resultó adecuado para un sistema institucional, ofreciendo controles claros, accesibles y suficientemente neutros para evitar estilos visuales excesivamente particulares. Esta elección redujo el tiempo

destinado a construir elementos desde cero y permitió mantener un estándar visual uniforme en todas las secciones del proyecto.

Además del uso de React y *shadcn*, se trabajó con Tailwind CSS, un *framework* de estilos utilitario que permite aplicar clases directamente en los componentes para definir diseño, espaciados y tipografías de manera consistente y eficiente. El uso de utilidades de estilo facilitó el ajuste de detalles específicos de la interfaz cuando fue necesario, sin introducir hojas de estilo complejas ni dependencias innecesarias.

La combinación de una arquitectura basada en componentes, una biblioteca confiable y un estilo visual homogéneo posibilitó desarrollar una interfaz clara, mantenible y alineada con las necesidades del Illia, que demanda una presentación accesible, ordenada y apta para distintos dispositivos.

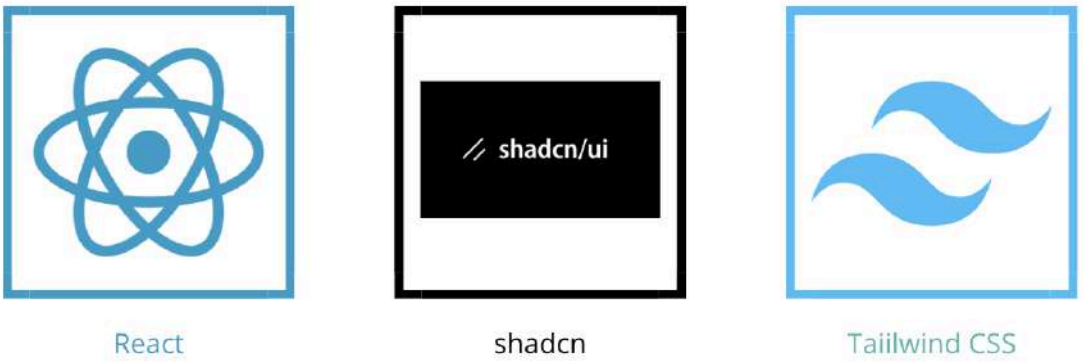


Figura 17. *Stack tecnológico utilizado para el desarrollo del frontend de IlliApp.*

Base de datos

Para la capa de persistencia se seleccionó un modelo de base de datos relacional, implementado mediante PostgreSQL como sistema gestor. La elección de un enfoque relacional frente a alternativas no relacionales se fundamentó en la naturaleza estructurada del dominio del sistema. La gestión académica involucra múltiples entidades con relaciones bien definidas (usuarios, cursos, materias, calificaciones y asistencias) que requieren consistencia, integridad referencial y

validaciones estrictas. En particular, procesos como la gestión de cursadas y el registro de asistencias presentan reglas complejas y dependencias entre múltiples entidades, lo que refuerza la necesidad de un modelo que garantice la coherencia de los datos.

Si bien las bases de datos no relacionales ofrecen ventajas en escenarios de alta escalabilidad o estructuras flexibles, no resultaban adecuadas para este caso, donde predominan operaciones transaccionales y reglas de negocio bien definidas.

Dentro de los motores relacionales, se seleccionó PostgreSQL por ser una solución open source, sin costos de licenciamiento, con amplia comunidad y documentación. En comparación con MySQL, ofrece un cumplimiento más estricto del estándar SQL, mayor robustez en la integridad referencial y mejor soporte para consultas complejas, lo que lo hace más adecuado para sistemas con alta exigencia en consistencia de datos. Estas características lo posicionan como una alternativa sólida para acompañar la complejidad y evolución del sistema.

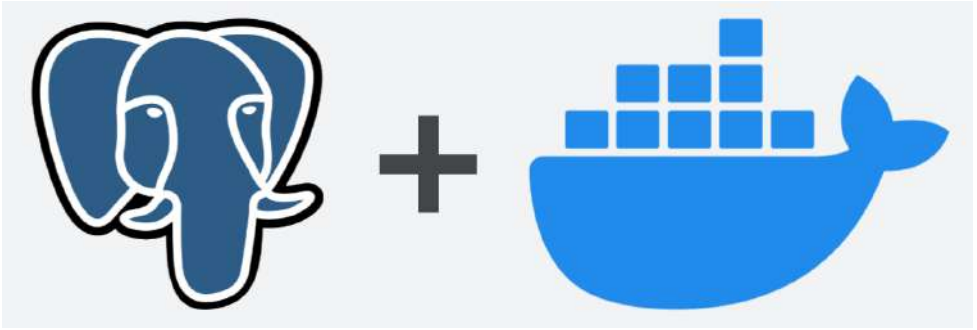


Figura 18. *Tecnologías utilizadas para la gestión y contenerización de la base de datos durante el desarrollo.*

Durante la etapa de desarrollo cada integrante del equipo dispuso de una instancia propia de PostgreSQL ejecutada mediante Docker. Esta estrategia uniformó el entorno de trabajo, evitando discrepancias en configuraciones o versiones instaladas localmente. También permitió generar entornos reproducibles con rapidez, lo que facilitó la experimentación, la validación de cambios estructurales y la resolución de incidentes sin comprometer el trabajo de otros miembros del equipo. Esta práctica contribuyó a mantener la estabilidad del proyecto y a reducir significativamente el tiempo dedicado a tareas de configuración.

Otras herramientas

Durante el desarrollo del sistema se emplearon diversas herramientas cuyo objetivo fue mejorar la organización del trabajo, asegurar la calidad del código y favorecer una comunicación clara entre los integrantes del equipo. Estas herramientas no forman parte del stack tecnológico central, pero resultaron esenciales para sostener el proceso de desarrollo de manera ordenada y consistente.

El control de versiones se gestionó mediante Git, y el repositorio se almacenó remotamente en Github. Esta plataforma permitió trabajar sobre un repositorio centralizado, registrar el historial de cambios y mantener un flujo de trabajo colaborativo. Se trabajó con un modelo basado en ramas, donde cada funcionalidad o corrección se desarrollaba de forma aislada antes de integrarse en la rama principal. La integración al repositorio requería la creación de un *Pull Request* y la revisión por parte de otro integrante del equipo, lo que colaboró a detectar errores tempranos y a elevar el estándar general del código.

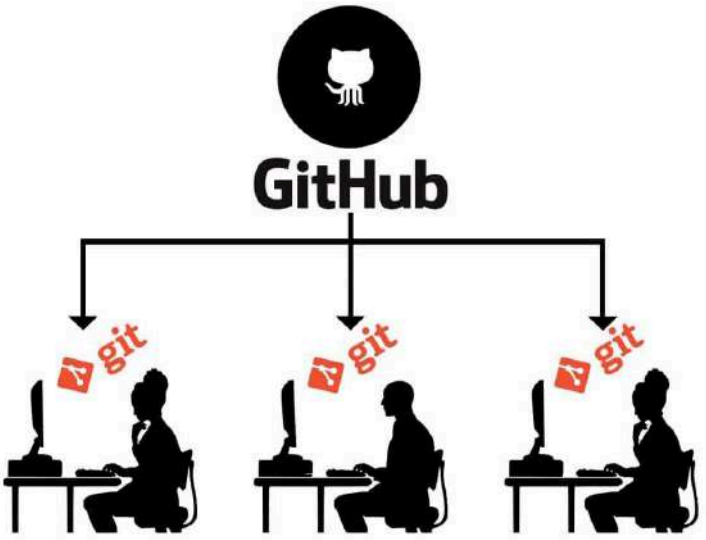


Figura 19. *Flujo de trabajo colaborativo mediante Git y GitHub, con integración centralizada a través de Pull Requests.*

Sobre esta dinámica se configuraron procesos de automatización mediante GitHub Actions. Cada *Pull Request* debía cumplir con una serie de validaciones previas al *merge*, entre ellas la ejecución de tareas de análisis estático y formateo de código y verificación de *build* del proyecto. Estas validaciones se configuraron para ejecutarse automáticamente, utilizando herramientas como *Prettier* y *ESLint*. La automatización evitó que errores de estilo o convenciones lleguen a la rama principal y redujo la carga manual de revisiones técnicas superficiales, dejando el foco del análisis en la funcionalidad y la arquitectura.

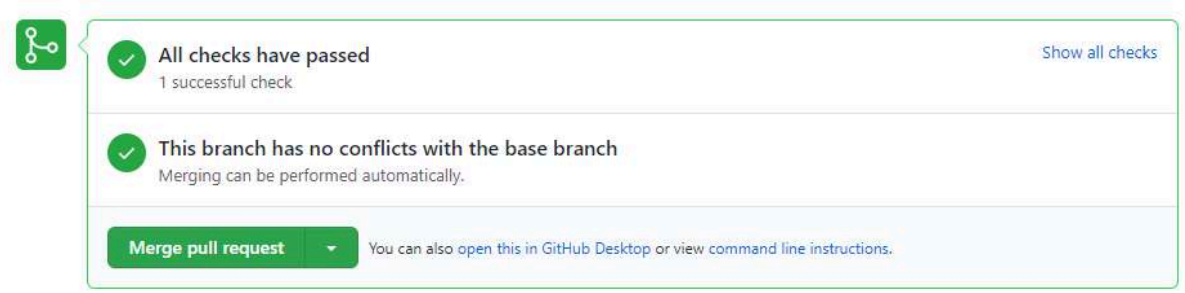


Figura 20. *Validaciones automáticas superadas en un Pull Request, requisito previo para su integración a la rama principal.*

Para el seguimiento de tareas se adoptó la metodología Kanban utilizando tableros en Trello. Cada integrante contaba con tarjetas asociadas a responsabilidades concretas y a su estado actual, lo que permitió visualizar el avance general del proyecto y detectar tareas bloqueadas o demandas emergentes. Esta dinámica facilitó la coordinación del equipo y la planificación semanal.

La documentación y el registro de decisiones se gestionaron en Google Drive. Allí se mantuvieron actas de reuniones, análisis de requerimientos, diagramas preliminares y archivos relevantes para el desarrollo. Este repositorio compartido permitió garantizar la trazabilidad a lo largo del proyecto y centralizó el acceso a materiales clave, evitando la dispersión de información y reduciendo la dependencia de comunicaciones informales.

En las etapas de diseño inicial se recurrió a herramientas de modelado. *Draw.io* se empleó para generar diagramas estructurales, en especial aquellos vinculados al modelado de datos y a la representación de entidades. De forma complementaria, se utilizó *Figma* para la elaboración de prototipos visuales de la interfaz, lo que

facilitó validar diseños con el referente funcional y ajustar criterios de usabilidad antes de implementar componentes en el sistema.

Durante el desarrollo del proyecto se incorporaron herramientas de inteligencia artificial como *ChatGPT* y *GitHub Copilot* como recursos de apoyo al proceso de desarrollo. Estas herramientas fueron utilizadas principalmente para optimizar tiempos de implementación, asistir en tareas de refactorización, mejorar la legibilidad y organización del código, y explorar alternativas de solución frente a problemas técnicos específicos.

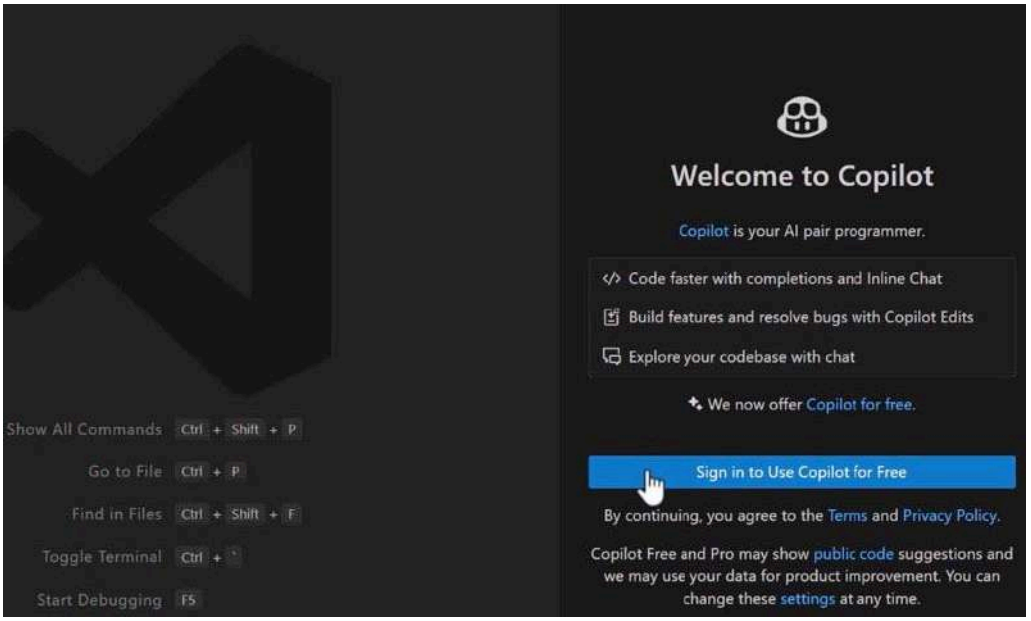


Figura 21. *Integración de GitHub Copilot en el entorno de desarrollo Visual Studio Code para asistencia en la generación de código.*

El uso de inteligencia artificial complementó, sin reemplazar, el análisis y las decisiones de diseño tomadas por el equipo. Estas herramientas funcionaron como apoyo en la investigación de buenas prácticas, comprensión de documentación técnica y evaluación de posibles *trade-offs* entre distintas implementaciones.

La incorporación de estas herramientas permitió acelerar iteraciones, elevar la calidad del código producido y enfocar el esfuerzo del equipo en aspectos de mayor valor, como el modelado del dominio, la definición de reglas institucionales y la coherencia general del sistema. Sin embargo, cabe aclarar que todo el código

generado y las decisiones propuestas siempre fueron validadas, primero por el encargado de desarrollar esa funcionalidad y luego mediante validación cruzada al hacer un *Pull Request.*

Finalmente, para la inspección temprana de datos y su estructura se trabajó con *Prisma Studio*. Esta herramienta proporcionó una vista clara y accesible de las tablas del sistema y de sus relaciones internas. Resultó especialmente útil durante las primeras etapas del proyecto, cuando el equipo aún no disponía de un gestor orientado a usuarios finales como *DBeaver*. Esto permitió validar rápidamente el diseño del modelo de datos y detectar inconsistencias antes de su propagación al resto del sistema.

Arquitectura general del sistema

El sistema se diseñó siguiendo una arquitectura monolítica modular *fullstack* basada en Next.js, un *framework* de React, donde el *frontend* y el *backend* se integran en una misma aplicación. El *frontend* se construyó con componentes React reutilizables que gestionan la interfaz de usuario y la interacción con los datos, mientras que el *backend* se implementó mediante *Server Actions*, responsables de la lógica de negocio, validación de reglas institucionales y manejo de solicitudes de los clientes.

Para la persistencia de la información se optó por una base de datos relacional PostgreSQL. Esta elección se fundamenta en la naturaleza altamente estructurada de los datos académicos que maneja el sistema, donde existen múltiples entidades fuertemente relacionadas entre sí, tales como estudiantes, cursos, materias, docentes, asistencias, evaluaciones y calificaciones. El uso de un modelo relacional permitió representar de manera consistente estas relaciones, garantizar la integridad referencial y aplicar restricciones que reflejan las normativas propias de la institución.

Asimismo, PostgreSQL ofrece mecanismos robustos para el manejo de transacciones, lo que resulta fundamental en operaciones críticas como la carga de notas, el registro de asistencias y la generación de reportes, donde es indispensable asegurar la coherencia de la información. La posibilidad de definir claves foráneas, índices y restricciones facilitó la implementación de validaciones a nivel de base de datos, reduciendo errores y mejorando la confiabilidad general del sistema.

Cada módulo del sistema (gestión académica, usuarios y roles, asistencia, evaluaciones y calificaciones, comunicaciones institucionales, reportes) se comunica con el *backend* a través de servicios internos que garantizan encapsulamiento, consistencia y seguridad de la información. Se implementaron mecanismos de control de acceso por roles y auditoría mediante registros de eventos para mantener trazabilidad de todas las operaciones críticas. La combinación de Next.js y React permite renderizado del lado del servidor y generación de contenido estático cuando es necesario, optimizando la performance y asegurando escalabilidad del sistema.

Diseño del modelo de datos

El diseño del modelo de datos fue una de las etapas más complejas del proyecto. La magnitud del dominio del Colegio Illia, sumada a la interacción entre múltiples actores y procesos académicos, implicó un esfuerzo considerable y un alto grado de iteración. Las primeras propuestas de modelado sufrieron modificaciones continuas; a medida que se discutían escenarios reales de uso, surgían nuevas formas de representar cursos, grupos, años académicos, evaluaciones y roles institucionales. Este proceso llevó al equipo a revisar varias veces las suposiciones iniciales.

El trabajo comenzó con un enfoque macro, identificando las entidades esenciales del sistema para comprender el panorama general. El primer esquema incluía componentes como materias, grupos, instancias evaluativas, cursos, usuarios, estudiantes, docentes e inasistencias, entre otros. A partir de esta base se generaron tres versiones sucesivas del modelo, ajustando relaciones, cardinalidades y niveles de normalización. La tercera versión logró un balance aceptable entre claridad conceptual y robustez estructural, y se convirtió en el punto de partida para avanzar con el resto de la arquitectura.

Sin embargo, con el desarrollo de los módulos resultó evidente que intentar modelar todo el sistema dentro de un único diagrama era poco práctico. La complejidad de las interacciones entre subsistemas hacía difícil mantener una visión clara y tomar decisiones acertadas. En consecuencia, se adoptó un enfoque modular: se elaboraron diagramas independientes para distintos sectores funcionales, como novedades, ausencias docentes, plan de estudios, asistencia y usuarios. Aunque todos estos esquemas forman parte del dominio global de Illia, separar el modelado

por áreas permitió al equipo discutir y refinar cada componente con mayor precisión. Esta estrategia redujo la carga cognitiva de las decisiones y facilitó la validación con escenarios reales, evitando cambios drásticos en etapas posteriores del desarrollo.

Los diagramas correspondientes a las distintas versiones del modelo, así como los esquemas específicos de cada módulo, se presentan en el Anexo I.

Módulos principales

El sistema se organizó mediante módulos funcionales independientes, cada uno orientado a resolver un subconjunto específico de requerimientos del Colegio Illia. Esta estrategia de modularización permitió abordar el desarrollo en etapas, realizar pruebas parciales y facilitar la integración gradual de componentes, evitando la acumulación de complejidad en fases tempranas. El orden de implementación se definió en función de la criticidad operativa y la dependencia funcional entre módulos.

Durante las primeras iteraciones se priorizó la construcción de los módulos de ausencias docentes, novedades, y gestión de usuarios. Se los seleccionó porque representaban un punto de partida manejable en términos de complejidad y al mismo tiempo ofrecían un marco mínimo indispensable para estructurar la aplicación. Su implementación temprana permitió validar el proceso de autenticación y autorización, definir patrones de diseño y navegación, consolidar la interfaz general y establecer criterios de persistencia de datos. Estas funcionalidades iniciales actuaron como cimiento sobre el cual se apoyó el resto del sistema.

A continuación se presentan los principales módulos desarrollados.

Gestión de usuarios

Este módulo permite la administración de las cuentas de la plataforma: alta, baja y modificación de usuarios, así como asignación y cambio de roles institucionales (docente, administrativo, preceptor, superpreceptor, directivo, tutor, alumno). Se incorpora además el restablecimiento de contraseñas por parte de un administrador, evitando la dependencia del equipo técnico y contribuyendo a la autonomía

operativa. Este módulo constituye el núcleo de control de acceso y determina los permisos vinculados al uso de los demás subsistemas.

Adicionalmente, el módulo contempla la vinculación entre preceptores y los cursos a su cargo, así como la asociación de estudiantes con uno o más tutores, reflejando la estructura organizativa real de la institución. De este modo, el módulo constituye el núcleo del control de acceso y de las relaciones institucionales, determinando los permisos y alcances vinculados al uso de los demás subsistemas.

Gestión académica

Este módulo abarca la administración de las entidades centrales del dominio: materias, cursos y talleres. Incluye operaciones CRUD y mecanismos para vincular docentes y estudiantes a las diferentes unidades académicas. Su diseño contempló particularidades propias del Illia, como la obligatoriedad de ciertos talleres o la coexistencia de grupos con estructuras heterogéneas. Estas decisiones permitieron consolidar una base académica coherente, sobre la cual se apoyan los módulos posteriores.

Gestión de asistencia estudiantil

Permite el registro diario de asistencia de estudiantes, incluyendo correcciones retroactivas, justificaciones de faltas y observaciones asociadas. El módulo contempla funcionalidades para registrar ausencias de manera retroactiva y generar un historial individual o por curso. Además, ofrece búsquedas por fecha o por recurrencia de inasistencias, funcionalidades necesarias para el seguimiento pedagógico y la detección temprana de situaciones problemáticas.

Asistencias docentes

Este módulo permite registrar las ausencias del personal docente, tanto de forma retroactiva como futura, y hacer disponible esta información a la comunidad educativa de manera clara y oportuna. Su objetivo principal es ofrecer a estudiantes

y familias un panorama confiable sobre las clases que no serán dictadas, evitando incertidumbre y fortaleciendo la comunicación institucional.

Anuncios institucionales

Este módulo provee un canal de comunicación interna para la publicación de avisos, novedades y eventos relevantes para la comunidad educativa. Las publicaciones pueden clasificarse según su naturaleza, distinguiendo entre anuncios, y cuentan con fecha de vigencia para asegurar que la información se mantenga actual y contextual. Su objetivo principal es evitar la dispersión de mensajes entre múltiples plataformas externas y consolidar un único entorno oficial donde estudiantes, familias y personal puedan mantenerse informados.

Actividades evaluativas

Este módulo reúne las funcionalidades relacionadas con evaluaciones y calificaciones. Permite la creación y modificación de calificaciones numéricas y conceptuales, la configuración de escalas de calificación y la generación automática de evaluaciones obligatorias. Incluye además la gestión de mesas de examen y el registro de trazabilidad sobre quién modifica cada calificación. Se trata de uno de los módulos de mayor complejidad al integrar criterios pedagógicos, reglas institucionales y datos académicos consolidados.

Reportes institucionales

Este módulo permite generar información de síntesis a partir de los datos registrados. Incluye la generación de boletines individuales y por curso, analíticos académicos según el plan de estudios provincial y reportes de notas por materia, curso o alumno. Además, contempla mecanismos de equivalencia entre el plan de estudio del colegio con el plan provincial. Estos reportes constituyen una herramienta clave para la comunicación con familias, autoridades educativas y para la administración interna del colegio.

Con estos componentes se dio por concluida la fase central de construcción del sistema, quedando preparados los cimientos funcionales para futuras extensiones y mejoras.

Interfaz de usuario y experiencia de usuario

La interfaz de usuario constituye el componente visible del sistema y actúa como puente entre las funcionalidades internas y las acciones ejecutadas por los distintos actores institucionales. Su diseño no se limita a aspectos gráficos; involucra la organización lógica de la información, la claridad de los flujos de navegación y la reducción de cargas cognitivas durante el uso. En paralelo, la experiencia de usuario comprende las percepciones resultantes de esa interacción: facilidad para completar tareas, consistencia visual, previsibilidad de acciones y bienestar general durante la utilización prolongada de la plataforma.

Durante el desarrollo se consideró una variable central: la heterogeneidad de los usuarios del Colegio Illia. La aplicación debía ser utilizada tanto por estudiantes, en su mayoría jóvenes familiarizados con entornos digitales, como por docentes, personal administrativo y tutores con diferentes niveles de alfabetización tecnológica. Este escenario obligó a priorizar un diseño accesible, intuitivo y poco ambiguo, evitando sobrecargas visuales o interacciones complejas. La máxima que guió las decisiones fue «menos es más»: interfaces limpias, consistentes y enfocadas en la tarea principal de cada pantalla.

En términos estéticos se optó por una paleta de colores inspirada en la identidad institucional del Colegio Illia, incorporando tonos de azul y rojo combinados con colores pastel. La elección buscó transmitir familiaridad con el entorno educativo y, al mismo tiempo, asegurar contraste suficiente para facilitar la lectura, evitando fatiga visual en uso prolongado. No se desarrolló un esquema de diseño recargado ni altamente ornamental, ya que el objetivo no era destacar aspectos artísticos sino reducir el potencial de errores y ofrecer un entorno visual estable y funcional.

Dado que el equipo no contaba con experiencia previa en diseño UI/UX, se priorizaron soluciones basadas en estándares consolidados y componentes

reutilizables. En este sentido, se emplearon componentes de *shadcn*, cuya documentación extensa y enfoque modular resultaron adecuados para estructurar interfaces coherentes sin necesidad de desarrollar patrones visuales desde cero. Esta decisión favoreció la consistencia entre vistas, la homogeneidad de interacción y una curva de aprendizaje más reducida para el equipo desarrollador.

La plataforma fue concebida como una aplicación web responsive, accesible desde computadoras de escritorio, notebooks, tablets y dispositivos móviles sin necesidad de versiones separadas. Esta decisión respondió al uso real que se anticipaba del sistema: consultas rápidas por parte de estudiantes, carga de datos desde dispositivos personales por parte de docentes y tareas administrativas desarrolladas en equipos propios del personal. Se adoptaron diseños adaptativos que priorizan la legibilidad, el escalado dinámico de componentes y la reorganización de elementos según el tamaño de pantalla.

Finalmente, se evitó incorporar funcionalidades superfluas o interacciones innecesariamente sofisticadas. La interfaz se construyó en capas incrementales, validando cada conjunto de pantallas con prototipos funcionales y evaluando su claridad antes de integrar nuevas acciones. El resultado obtenido no pretende innovar desde el diseño artístico, sino ofrecer una experiencia predecible, accesible y robusta, en línea con el carácter operativo del sistema y las necesidades concretas de la comunidad educativa.

A partir de las decisiones de diseño previamente mencionadas, la interfaz final del sistema presenta una estructura clara, fluida y visualmente coherente con la identidad institucional del Colegio Illia. Las pantallas priorizan la simplicidad operativa, la legibilidad y la accesibilidad para usuarios con distintos niveles de familiaridad tecnológica. A continuación, se muestran algunas vistas representativas de la plataforma, que ilustran la distribución general de los elementos, la organización de la información y la consistencia estética aplicada en todo el entorno.

Nota: Los datos mostrados en las siguientes pantallas son ficticios y se han generado con fines ilustrativos, para proteger la información confidencial de estudiantes, tutores y personal educativo.

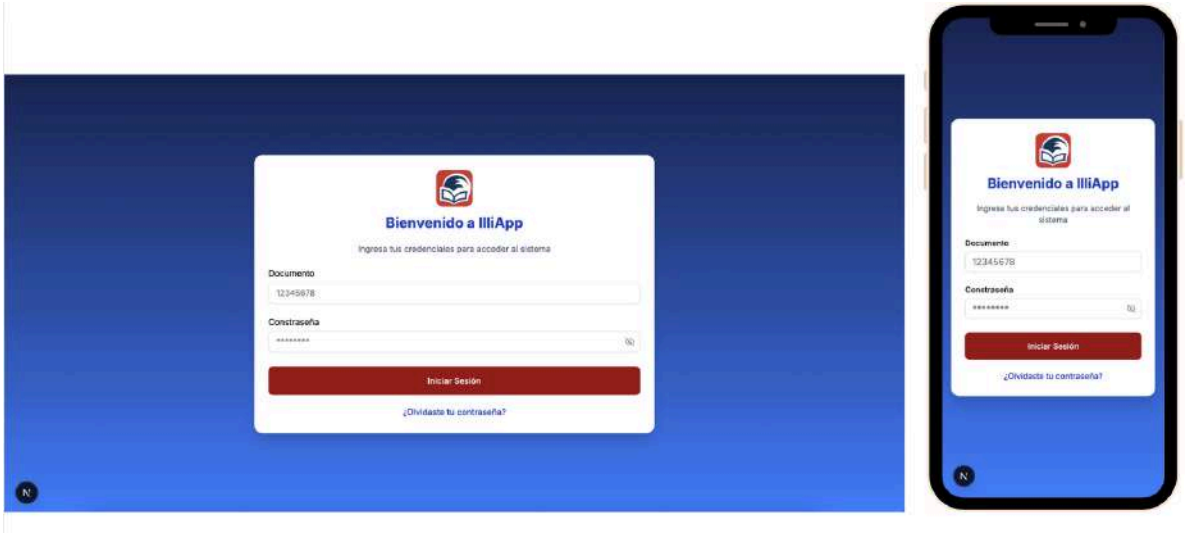


Figura 22. *Pantalla de inicio de sesión, versión desktop y mobile.*

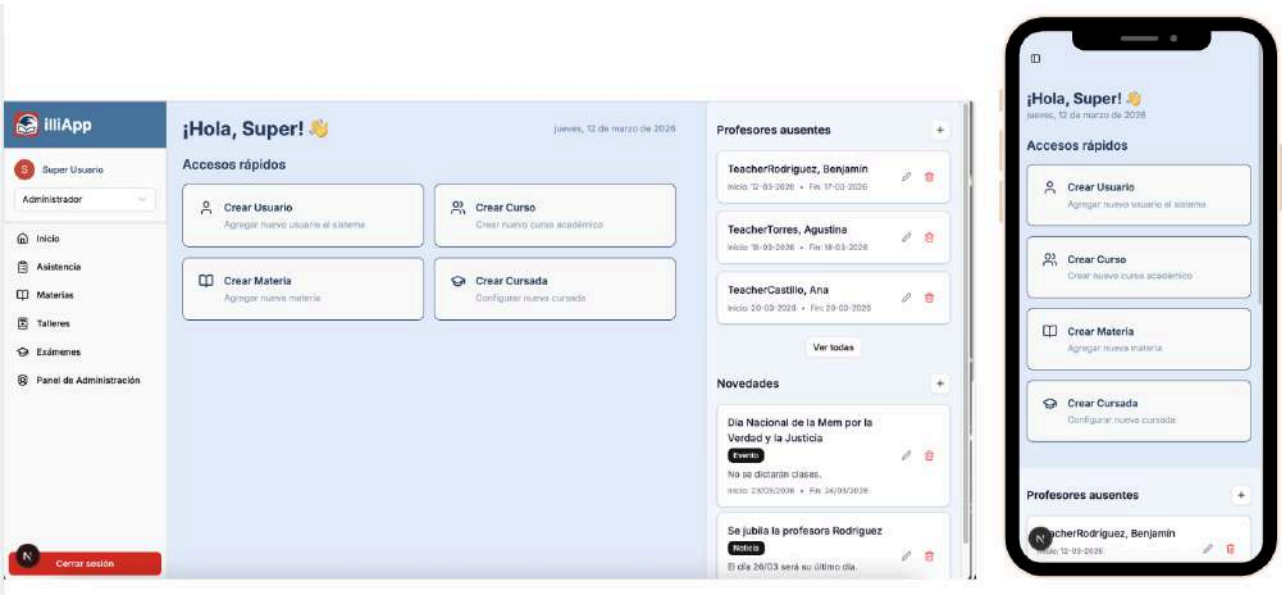


Figura 23. *Pantalla de inicio para el administrador, versión desktop y mobile.*

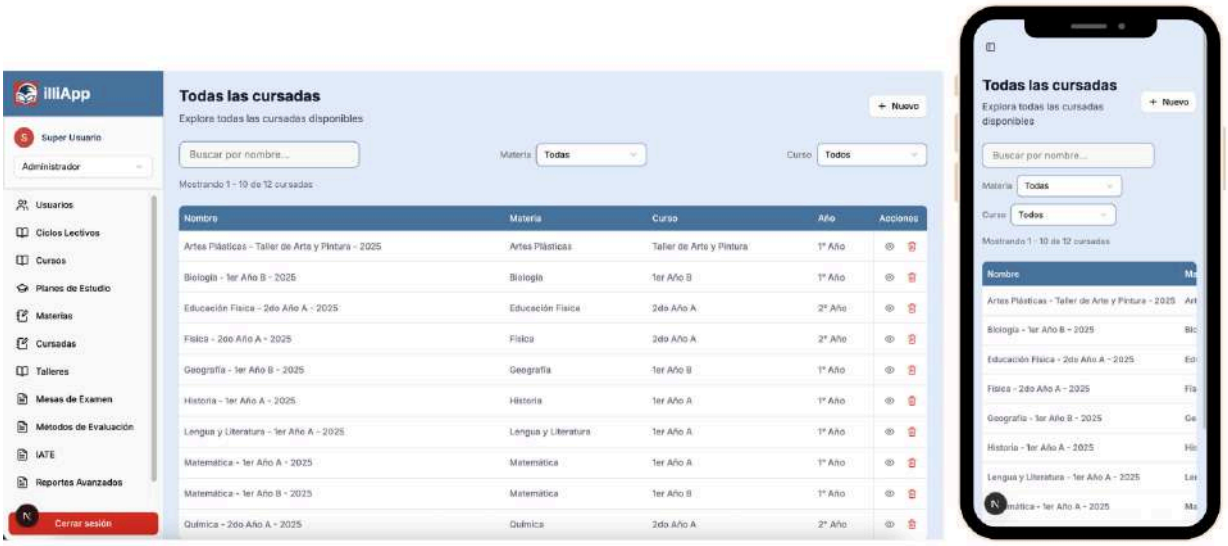


Figura 24 *Pantalla gestión de cursadas, dentro del panel de administrador. Versión desktop y mobile.*

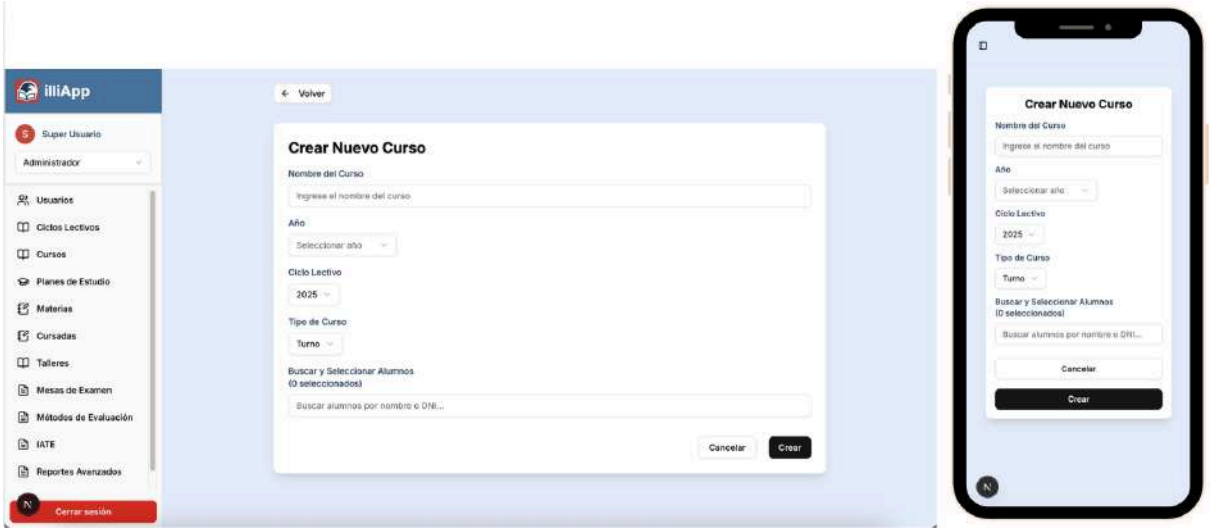


Figura 25. *Pantalla para crear un nuevo curso, dentro del panel de administrador. Versión desktop y mobile.*

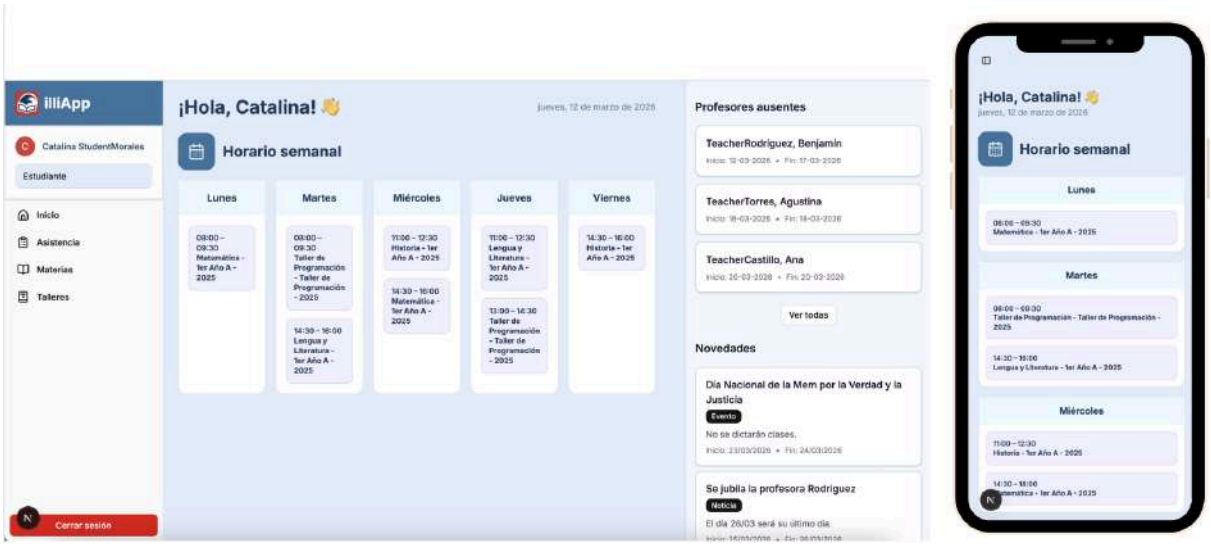


Figura 26. *Pantalla para inicio para un alumno Versión desktop y mobile.*

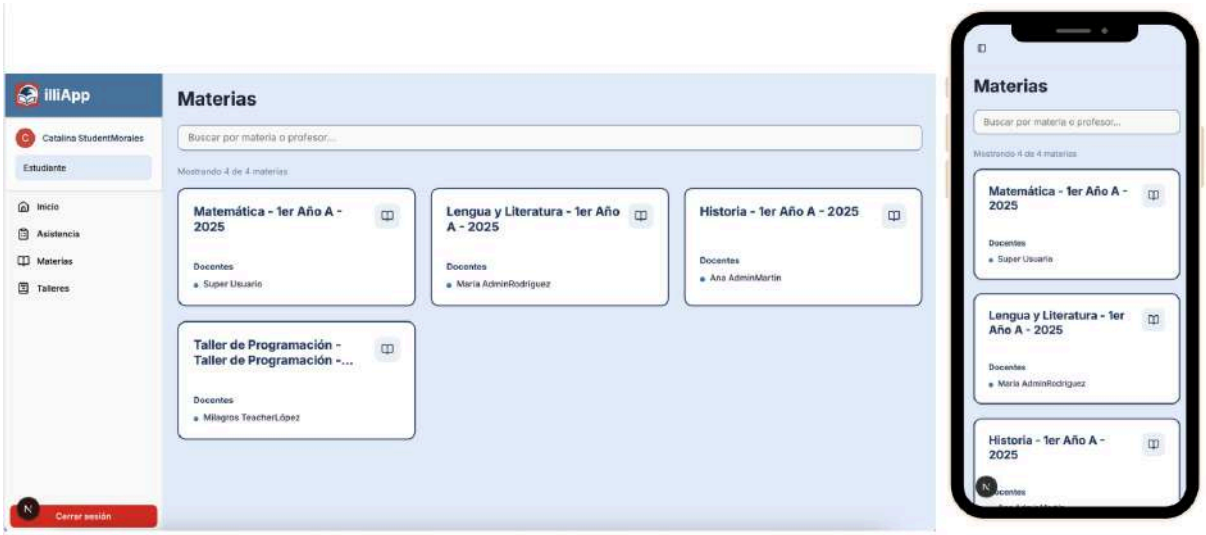


Figura 27. *Pantalla de materias de un alumno. Versión desktop y mobile.*

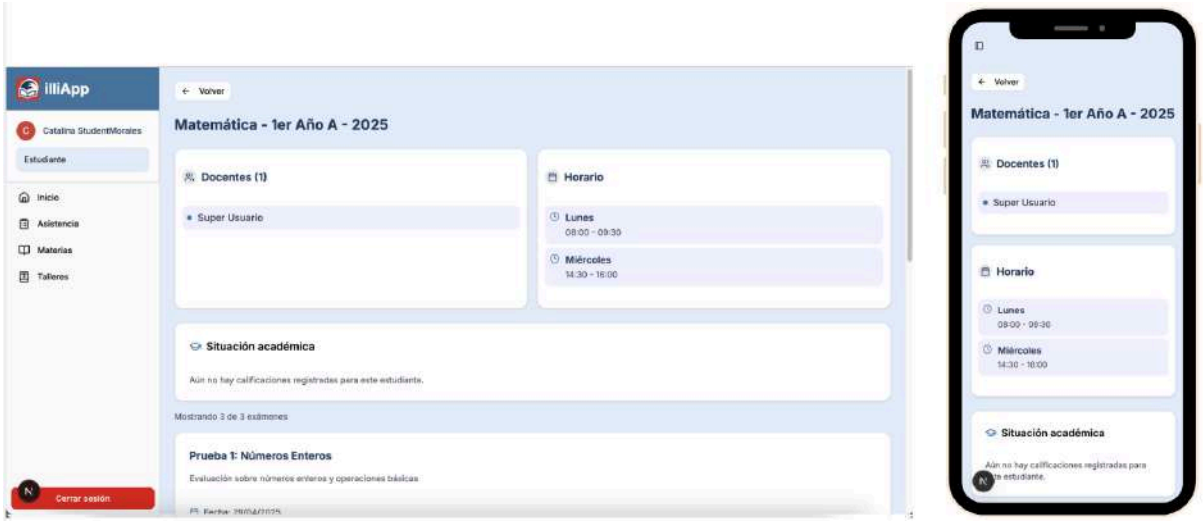


Figura 28. *Pantalla de una materia de un alumno. Versión desktop y mobile.*

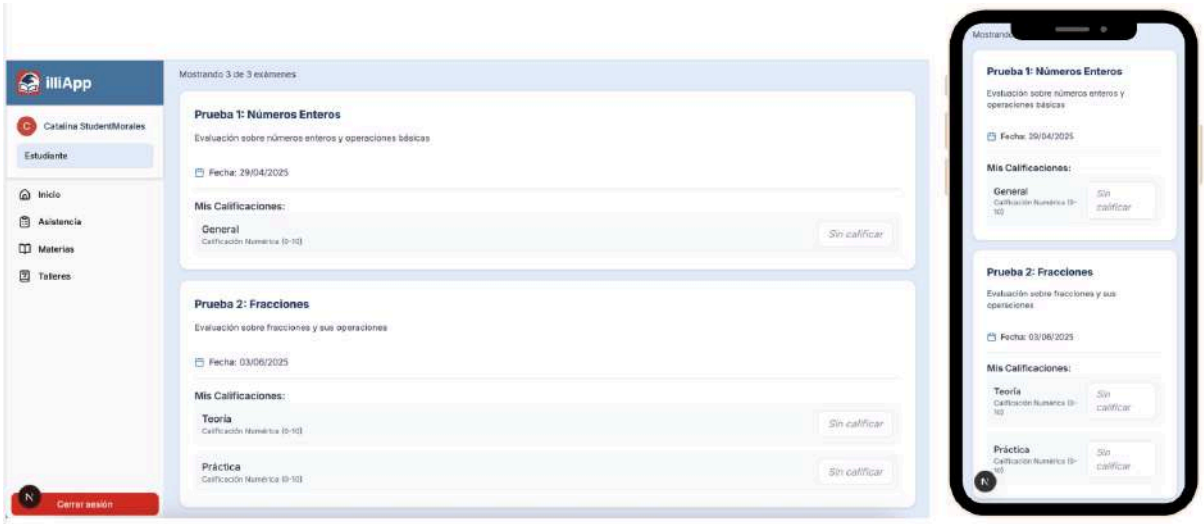


Figura 29. *Pantalla de una materia de un alumno. Detalle de las evaluaciones. Versión desktop y mobile.*

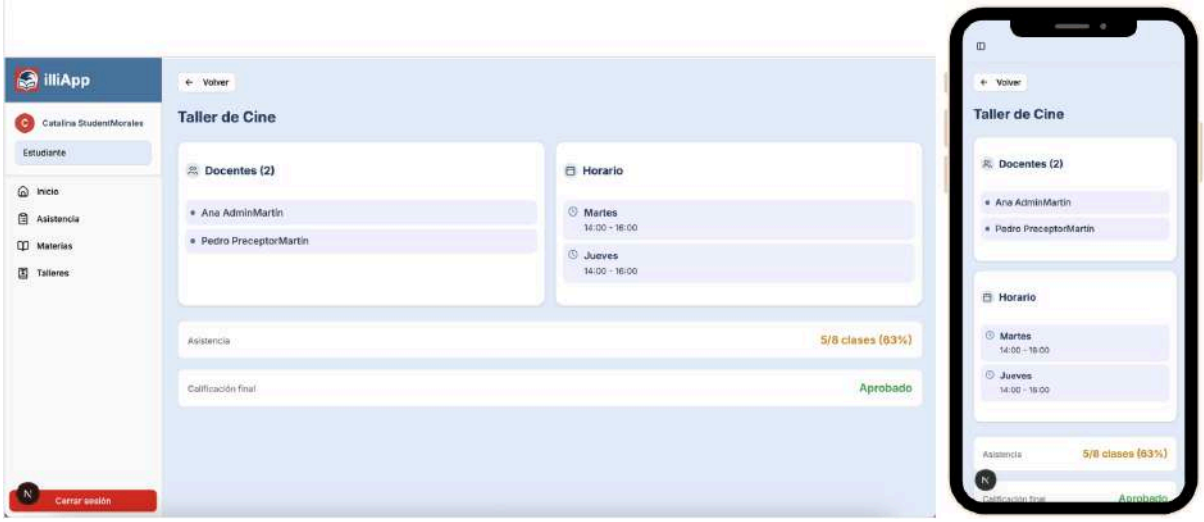


Figura 30. *Pantalla de detalle del taller de un alumno. Versión desktop y mobile.*

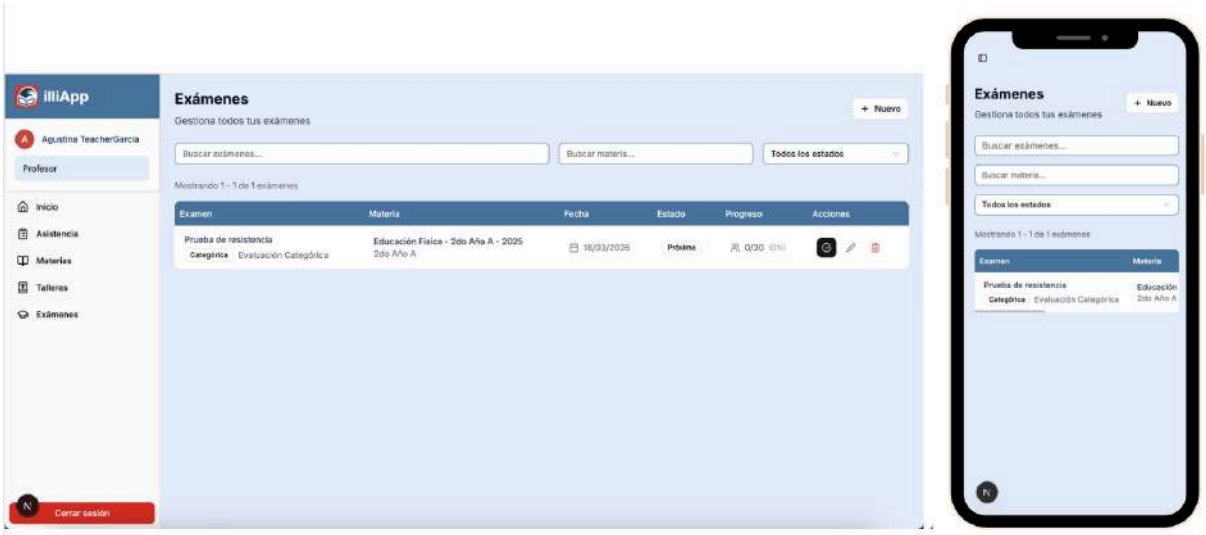


Figura 31. *Pantalla de exámenes de una materia de un profesor. Versión desktop y mobile.*

Figura 32. *Pantalla de calificar exámenes. Versión desktop y mobile.*

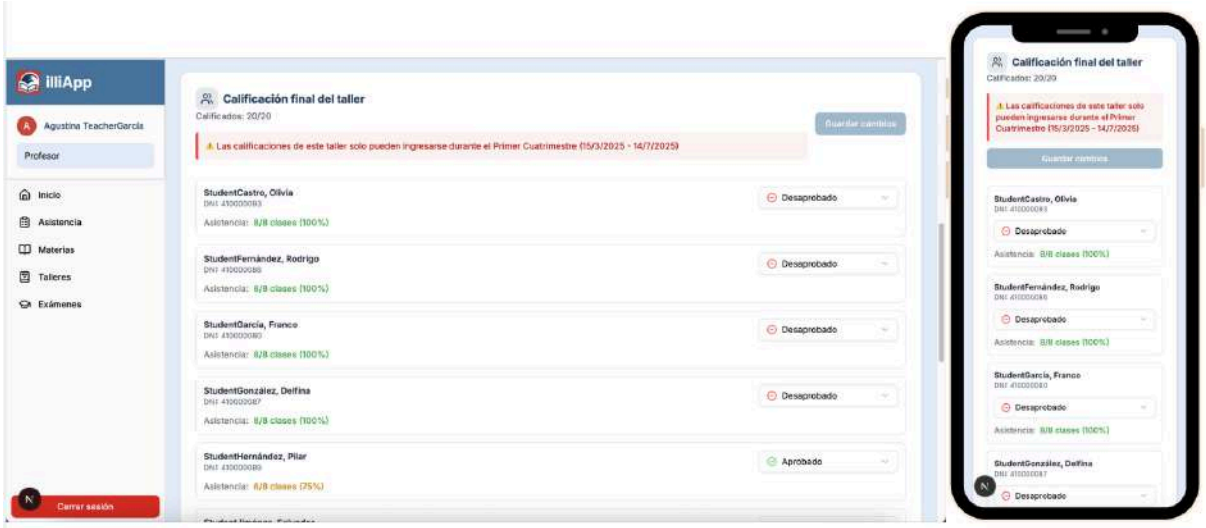


Figura 33. *Pantalla para que un profesor califique un taller*[1]*. Versión desktop y mobile.*

[1] En este ejemplo, se ilustra cómo un profesor no puede calificar fuera del ciclo lectivo vigente ni fuera de las fechas establecidas.

Figura 34. *Pantalla que muestra los cursos asignados a un preceptor para la toma de asistencia. Versión desktop y mobile.*

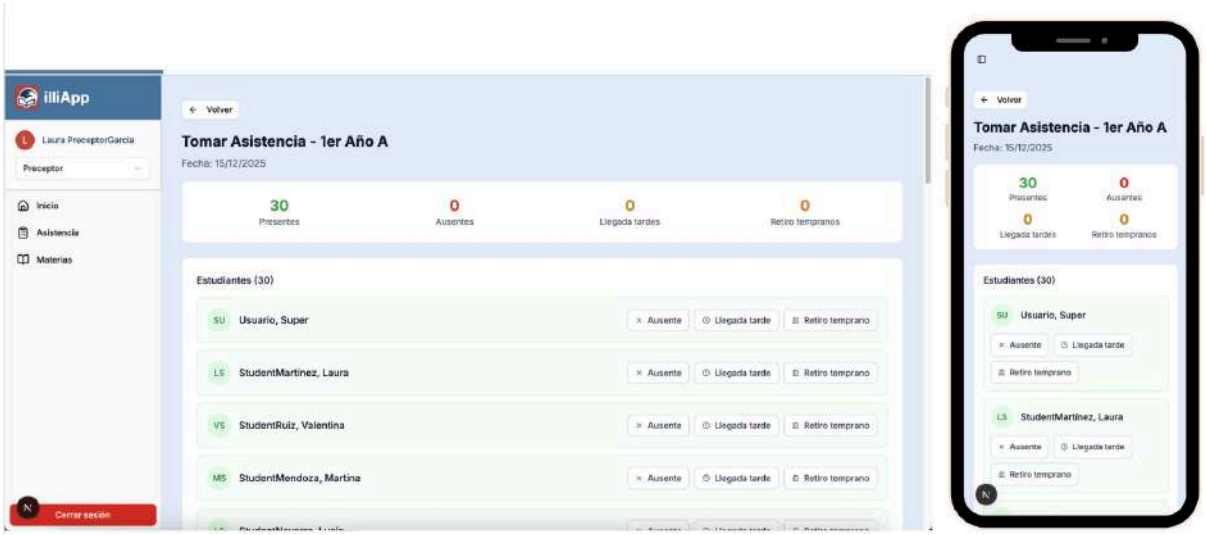


Figura 35. *Pantalla para la toma de asistencia de un curso en particular. Versión desktop y mobile.*

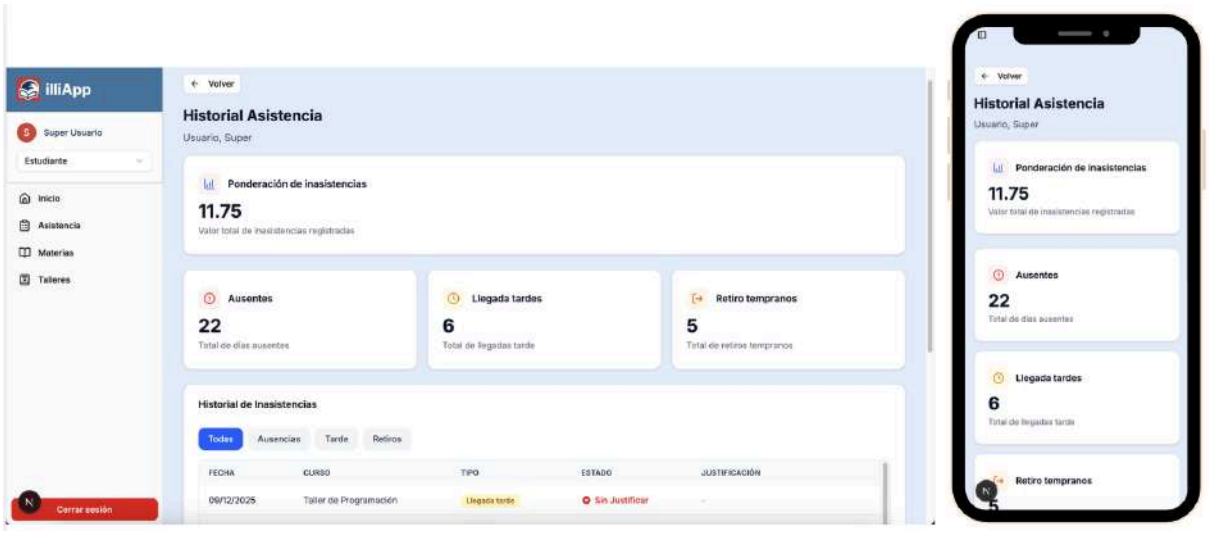


Figura 36. *Pantalla de historial de asistencia de un alumno. Versión desktop y mobile.*

Seguridad y control de accesos

La seguridad constituye un eje central en el desarrollo de la plataforma, dado que el sistema administra información personal y académica de estudiantes, docentes y personal administrativo. El diseño de la arquitectura contempló desde el inicio mecanismos de protección en cada capa: *backend*, *frontend* y procesos de almacenamiento. El objetivo fue minimizar el riesgo de accesos indebidos, prevenir manipulación de datos y garantizar la integridad de las operaciones realizadas por los actores del sistema. A continuación, se describen las medidas más relevantes implementadas.

Autenticación mediante JSON Web Token (JWT)

El sistema implementa un esquema de autenticación basado en JSON Web Token, mecanismo ampliamente utilizado en entornos web para gestionar sesiones sin la necesidad de mantener estado en el servidor. Cuando un usuario inicia sesión con credenciales válidas, el *backend* genera un token firmado digitalmente que contiene información mínima pero suficiente para validar la sesión: identificador del usuario, nombre y un conjunto de roles asociados. Este token se devuelve al cliente y es

almacenado localmente, siendo enviado en cada solicitud posterior a recursos protegidos mediante encabezados HTTP.

El uso de tokens firmados impide la modificación de su contenido sin invalidarlos, ya que la clave de firma se encuentra resguardada en el servidor mediante variables de entorno. Además, el sistema utiliza un esquema de expiración basado en dos tipos de tokens: un *access token* con una validez de 5 minutos y un *refresh token* con una duración de 30 minutos, lo que permite equilibrar seguridad y usabilidad al reducir la exposición de sesiones prolongadas sin requerir autenticaciones constantes.

A diferencia de los esquemas tradicionales basados en sesiones, donde el servidor almacena el estado de cada usuario, el uso de JWT permite un enfoque *stateless*, mejorando la escalabilidad y simplificando la distribución del sistema. Como contrapartida, la invalidación anticipada de sesiones no es directa, por lo que se optó por utilizar expiración temporal como mecanismo principal de control.

Control de acceso basado en roles

La plataforma implementa un esquema de autorización granular mediante roles asociados con usuarios, lo que permite restringir el acceso según responsabilidades institucionales reales. En lugar de depender únicamente de rutas o endpoints específicos, el sistema controla el acceso en múltiples capas mediante *middleware* y validaciones contextuales.

Los roles son definidos en el *backend* y se almacenan directamente en la base de datos. Durante la autenticación, los roles permitidos se incorporan como un arreglo dentro del JWT emitido. Esta decisión se adoptó para evitar escenarios de escalamiento de privilegios mediante modificaciones del lado del cliente, ya que cualquier cambio no autorizado invalida la firma criptográfica del token. Al momento de acceder a un recurso, el servidor decodifica el token y verifica que el usuario posea el rol apropiado para operar sobre dicho recurso. Si el rol no coincide, el acceso se rechaza sin ejecutar la lógica del endpoint.

Este mecanismo permite administrar permisos por funcionalidad (por ejemplo, carga de calificaciones, registro de ausencias, administración de usuarios) y evita que

usuarios con privilegios parciales accedan a información o acciones que excedan su función real en la institución.

Figura 37.Interfaz de cambio de rol que permite alternar entre distintos perfiles de usuario según los permisos definidos en el sistema.

Validación de rutas mediante middleware

El *backend* incorpora *middleware* de protección de rutas, un componente de software que intercepta peticiones antes de que alcancen la lógica principal del controlador. Este middleware verifica la presencia del token, su validez y los roles incluidos. De este modo, cualquier endpoint protegido puede ser bloqueado antes de ejecutar operaciones que involucren acceso a datos o modificaciones persistentes.

El uso de *middleware* aporta dos ventajas principales: por un lado, permite centralizar la lógica de seguridad y reducir la duplicación en el código; por el otro, asegura que la validación sea homogénea y no dependa de comportamientos aislados implementados en cada módulo. En caso de token inválido, expirado o insuficiente desde el punto de vista de permisos, el *middleware* detiene la ejecución y retorna un error estándar al cliente.

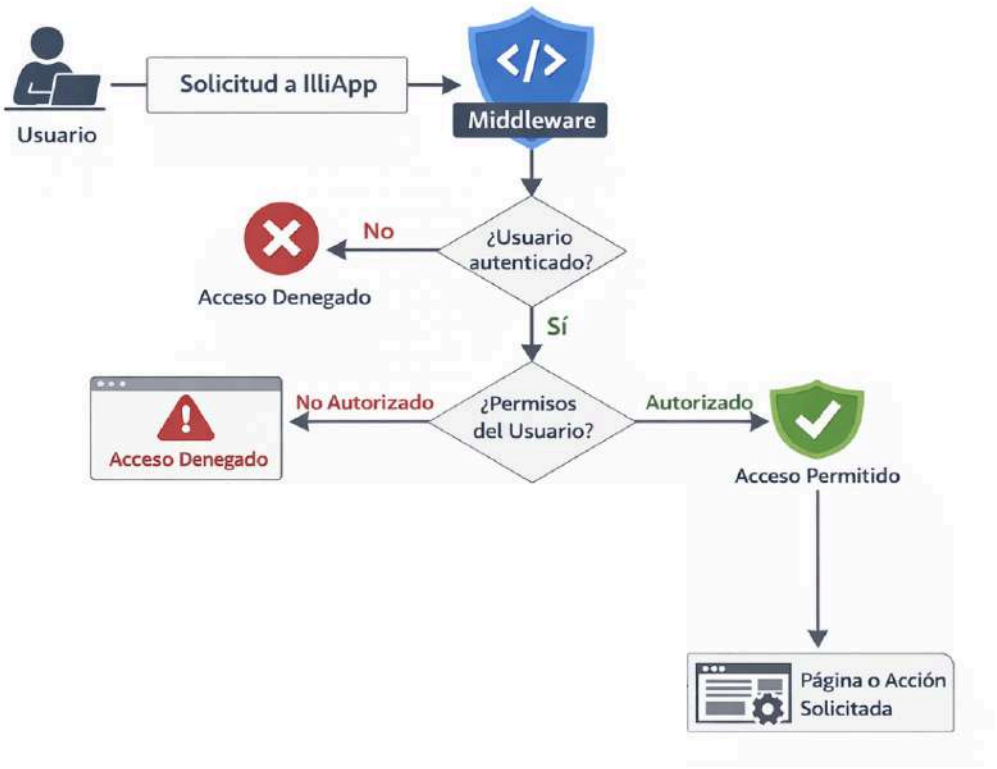


Figura 38. Flujo de funcionamiento del middleware del sistema.

Hashing y protección de credenciales

Para el almacenamiento de contraseñas, la plataforma utiliza bcrypt, un algoritmo de hashing adaptativo diseñado para mitigar ataques de fuerza bruta. Si bien existen alternativas más seguras, como Argon2, estas requieren un mayor tiempo de procesamiento, lo que no resulta del todo recomendable para la fluidez de una aplicación web. Bcrypt, en cambio, proporciona un balance ideal entre rendimiento y seguridad. Durante este proceso, se incorpora un fragmento de datos aleatorios denominado “salt”, que se combina con la contraseña antes de generar el hash. El uso de salt garantiza que contraseñas idénticas produzcan resultados cifrados distintos, lo que impide la detección de patrones y neutraliza el uso de tablas precalculadas (*rainbow tables*).

En el proceso de autenticación, la contraseña ingresada por el usuario se combina nuevamente con su salt correspondiente y se genera un nuevo hash, que se compara con el valor almacenado en la base de datos. No es necesario descifrar

información sensible en ningún punto del flujo, lo que reduce significativamente la superficie de ataque y refuerza la protección de las credenciales.

Protección mediante variables de entorno

Se emplean variables de entorno para almacenar información crítica, incluyendo claves de firma JWT, cadenas de conexión y valores sensibles de configuración. Esto impide que dichas claves formen parte del repositorio de código o sean visibles para terceros durante el desarrollo colaborativo. El uso de entornos diferenciados (desarrollo, test, producción) permite mantener configuraciones independientes, reduciendo el riesgo de exposición involuntaria o errores operativos.

Conclusión

La arquitectura de seguridad de la plataforma se concibe como un proceso dinámico y no como un estado estático. Las medidas implementadas para el resguardo de credenciales, el control de acceso y la protección de los datos durante su transmisión constituyen la base necesaria para preservar la confidencialidad y la integridad del sistema. Este enfoque resulta especialmente relevante en el contexto de una institución académica, donde se gestionan datos personales de directivos, docentes y estudiantes, incluidos menores de edad, así como información académica sensible. La protección de estos datos no solo responde a buenas prácticas técnicas, sino también a una responsabilidad ética y legal asociada al cuidado de la comunidad educativa.

Su efectividad depende de un mantenimiento continuo, revisiones periódicas de vulnerabilidades, actualización de dependencias y adopción constante de buenas prácticas. La seguridad se presenta así como un compromiso permanente que debe acompañar tanto el crecimiento funcional del proyecto como la evolución de su ecosistema tecnológico.

Componente innovador

Si bien el presente proyecto se enmarca principalmente en un proceso de transformación digital, IlliApp incorpora un componente innovador de carácter

contextual. La innovación del sistema no radica únicamente en el uso de tecnologías actuales, sino en su capacidad de adaptarse de manera específica a las particularidades organizativas, académicas y administrativas del Colegio Nacional Dr. Arturo Umberto Illia, las cuales no son abordadas de forma eficiente por los sistemas de gestión académica existentes en el mercado.

Durante la etapa de relevamiento se identificó que las soluciones existentes se encontraban orientadas a instituciones educativas con estructuras más estandarizadas y no contemplaban adecuadamente las particularidades propias del Colegio preuniversitario dependiente de la Universidad Nacional de Mar del Plata. Estas características, tales como la organización académica, los roles institucionales, las reglas de asistencia y evaluación, y los circuitos administrativos internos, requieren un alto grado de personalización que las plataformas genéricas no ofrecen sin incurrir en configuraciones complejas o adaptaciones poco sostenibles.

En este sentido, el proyecto se destaca por cubrir una necesidad actualmente no satisfecha en la ciudad de Mar del Plata: la ausencia de un software de gestión académica desarrollado a medida para un colegio nacional con las características del Illia. IlliApp fue concebido específicamente para este contexto, permitiendo reflejar fielmente su funcionamiento real y acompañar los procesos institucionales sin forzarlos a encajar en modelos predefinidos.

Asimismo, uno de los aspectos más relevantes del componente innovador del sistema es su potencial de expansión y reutilización. Si bien el desarrollo se orientó a resolver los requerimientos particulares del Colegio Illia, la arquitectura modular y la separación clara de responsabilidades permiten que el sistema pueda adaptarse a otras instituciones educativas con necesidades similares. De este modo, IlliApp se presenta como un modelo replicable dentro del ámbito universitario, capaz de ser extendido o ajustado sin necesidad de rediseñar su núcleo.

Finalmente, el desarrollo del sistema representa una contribución concreta y de alto valor agregado de la Facultad de Ingeniería a la comunidad educativa de la Universidad Nacional de Mar del Plata. No solo se trata de un producto funcional, sino de una solución que articula conocimientos académicos con una problemática

real del entorno, fortaleciendo el vínculo entre la universidad y sus instituciones dependientes.

Producto

Producto obtenido

El resultado final consiste en una aplicación web funcional que integra *frontend*, *backend* dentro de una misma solución *fullstack*, junto con una base de datos persistente. La plataforma opera bajo un esquema de autenticación obligatoria: ningún usuario puede acceder a las funcionalidades del sistema sin iniciar sesión previamente. Esta decisión responde a la naturaleza sensible de los datos administrados y a la necesidad de preservar la privacidad de los estudiantes, los tutores y el personal institucional.

La experiencia dentro del sistema está determinada por el rol del usuario. Cada perfil habilita un conjunto específico de funciones y restringe las acciones que exceden sus competencias. De esta manera, un preceptor, un docente o un tutor ven interfaces diferenciadas, con módulos y permisos adaptados a sus necesidades. Cuando un usuario posee múltiples roles, como puede ocurrir en el personal administrativo que también cumple tareas docentes, la plataforma incorpora un selector de rol que permite alternar entre perfiles sin necesidad de cerrar sesión. Este mecanismo evita confusiones y asegura que las acciones se realicen siempre bajo el rol correspondiente.

Desde el punto de vista funcional, la aplicación organiza sus operaciones en módulos claramente diferenciados. Cada módulo cuenta con pantallas propias, flujos de interacción coherentes y validaciones asociadas al contexto del rol que las utiliza. Esta estructura modular simplifica el mantenimiento, facilita la incorporación de nuevas funcionalidades y reduce la complejidad percibida por el usuario final.

En términos operativos, el *frontend* interactúa con la lógica del servidor mediante *Server Actions* provistas por Next.js, que permiten ejecutar código del lado del servidor a partir de interacciones iniciadas desde la interfaz. Estas acciones validan

la sesión activa, verifican los roles del usuario y aplican las reglas de acceso correspondientes antes de operar sobre los datos.

Desde el lado del cliente, la aplicación se estructura a partir de componentes reutilizables y patrones de navegación uniformes, lo que garantiza una experiencia de uso consistente en computadoras, tablets y dispositivos móviles, sin necesidad de exponer una API pública independiente.

El producto obtenido no es únicamente un conjunto de pantallas, sino que constituye un sistema integral pensado para el entorno educativo, en el que cada usuario interactúa exclusivamente con la información que le corresponde, dentro de límites bien definidos y preservando en todo momento la seguridad de los datos institucionales.

Trabajos futuros

Durante el desarrollo de IlliApp se identificaron y analizaron diversos requerimientos que, si bien resultaban válidos y coherentes con el dominio del sistema, debieron ser aplazados para una etapa posterior. Esta decisión estuvo motivada principalmente por la necesidad de priorizar el alcance del proyecto, asegurar la estabilidad de las funcionalidades centrales y cumplir con los plazos establecidos para la finalización del trabajo final de carrera.

En la etapa de análisis inicial se recopiló un conjunto amplio de requerimientos, algunos de los cuales, con el avance del desarrollo, demostraron implicar una complejidad técnica significativa en relación con su frecuencia de uso o valor inmediato para la institución. Como se menciona en secciones anteriores, la reevaluación de estos requerimientos permitió al equipo enfocar los esfuerzos en aquellas funcionalidades esenciales para el funcionamiento cotidiano del colegio, sin comprometer la posibilidad de incorporar los requerimientos aplazados en futuras versiones del sistema.

A continuación, se detallan los principales requerimientos que fueron postergados.

Inhabilitación automática de estudiantes egresados (RF-4)

Este requerimiento establecía que los estudiantes que hayan egresado y no adeuden materias debían quedar inhabilitados automáticamente del sistema. Si bien se trata de una regla coherente con el dominio académico y alineada con el funcionamiento institucional, su implementación implicaba incorporar lógica adicional vinculada al cierre del ciclo lectivo y a la verificación integral de la situación académica de cada alumno.

Tras evaluar su criticidad y prioridad dentro del contexto del proyecto, se determinó que su caso de uso efectivo se presenta únicamente al finalizar el año lectivo. En consecuencia, y considerando la necesidad de cumplir con los plazos establecidos para la entrega del trabajo final, se decidió postergar su desarrollo para una etapa posterior.

La funcionalidad no fue descartada, sino diferida estratégicamente, priorizando aquellas características con impacto directo en el uso cotidiano del sistema durante el período activo del ciclo lectivo.

Registro detallado de modificaciones en calificaciones (RF19)

Se consideró la posibilidad de llevar un registro exhaustivo de los usuarios que cargan o modifican notas, incluyendo un historial completo de cambios accesible desde la interfaz de usuario. Actualmente, el sistema registra estas acciones a través de archivos de log, permitiendo identificar quién realizó una modificación y en qué momento, aunque dicha información no se encuentra expuesta de manera directa en la UI. La incorporación de un historial visible y navegable implicaba un mayor esfuerzo en términos de diseño de datos y desarrollo, por lo que se decidió priorizar la correcta carga y visualización de calificaciones, dejando este aspecto como una mejora futura.

Creación de actividades evaluativas conjuntas entre materias (RF24)

Este requerimiento contemplaba la posibilidad de que un docente pudiera crear una actividad evaluativa compartida entre distintas materias. Sin embargo, durante su análisis se identificó una complejidad adicional: una misma actividad podía requerir calificaciones distintas según la materia (por ejemplo, una evaluación conjunta entre Historia e Informática donde un alumno obtenga calificaciones diferentes en cada

asignatura). Considerando este escenario, se optó por una solución más simple y flexible, en la cual cada docente crea su propia instancia de la actividad evaluativa, utilizando el mismo nombre y fecha, y registra la calificación correspondiente a su materia. Dado que la funcionalidad de creación conjunta aportaba principalmente una mejora de conveniencia en la interfaz y no resultaba crítica para el proceso académico, se decidió aplazar su implementación.

Cambio de ciclo lectivo y promoción de estudiantes (RF40, RF76, RF77)

Los requerimientos asociados al cambio automático de año lectivo y a las reglas de promoción de estudiantes implicaban la definición de una lógica compleja, estrechamente ligada a normativas institucionales y a situaciones académicas particulares. Además, su ejecución efectiva se produce al cierre del ciclo lectivo, fuera del período inmediato de uso del sistema durante el desarrollo del proyecto. Por estas razones, se optó por aplazar su implementación, priorizando la finalización y puesta en marcha del sistema dentro de los márgenes temporales establecidos.

Cambios de curso u orientación (RF82, RF83)

El cambio de orientación de un alumno conllevaba la gestión de equivalencias entre materias, la preservación de calificaciones previas y la reasignación de docentes y cursos. Si bien se trata de una funcionalidad relevante, es un evento de baja frecuencia dentro del Colegio Illia, lo que reduce considerablemente su impacto práctico. Su implementación habría requerido un desarrollo y un testing exhaustivos para garantizar la correcta gestión de los datos involucrados, lo cual no se justificaba en relación con su ocurrencia real ni con los tiempos de entrega del proyecto. Por estas razones, se decidió diferir su implementación, quedando como una posible mejora a evaluar en iteraciones futuras.

Ampliación de la cobertura de pruebas

Si bien se implementaron pruebas automatizadas a lo largo del desarrollo, un objetivo para iteraciones futuras es alcanzar el estándar del 80% de cobertura de código. La estrategia consistirá en abordar primero los módulos más críticos del sistema como el módulo de notas, asistencia o gestión de usuarios, hasta lograr la métrica deseada. Resultará conveniente hacer un mayor énfasis en esta fase de

aseguramiento de calidad una vez que los requerimientos de la aplicación alcancen un mayor grado de estabilidad.

Puesta en marcha con datos reales

Actualmente, el sistema se encuentra desplegado en el servidor del colegio. Sin embargo, por cuestiones de logística ajenas al desarrollo del proyecto, la puesta en marcha definitiva con datos y usuarios reales quedó a cargo del Colegio Illia.

Capacitación de usuarios finales

Si bien inicialmente la formación de los usuarios finales formaba parte de la planificación, al quedar la puesta en marcha masiva fuera del alcance inmediato del proyecto, también debió delegarse esta instancia. Como contingencia, la capacitación se dictó al referente funcional del colegio, quien asumirá el rol de instruir a los distintos actores del sistema de cara a la implementación final.

En conjunto, los requerimientos aplazados reflejan decisiones conscientes de gestión del alcance, orientadas a equilibrar la complejidad técnica, el valor funcional y los plazos del proyecto. Lejos de ser descartados, estos requerimientos constituyen una base clara para futuras extensiones del sistema, que podrán ser abordadas con una mayor disponibilidad de tiempo y recursos.

Memoria del proyecto

Problemáticas encontradas y soluciones

Durante el desarrollo de la plataforma se presentaron dificultades de diversa índole que impactaron tanto en la planificación inicial como en la ejecución técnica del proyecto. Estas situaciones estuvieron estrechamente vinculadas con la complejidad institucional del colegio Illia, con decisiones técnicas que exigieron iteraciones repetidas y con aspectos organizativos propios de un proyecto de esta magnitud.

Dinámica de trabajo y planificación temporal

El equipo trabajó con estimaciones de horas por funcionalidad que, en la mayoría de los casos, resultaron acertadas en cuanto al esfuerzo técnico requerido. No obstante, la dedicación efectiva no siempre se concentró de manera continua en los períodos previstos. En varias oportunidades, tareas estimadas en aproximadamente diez horas, por ejemplo, demandaron ese tiempo real de desarrollo, pero distribuidas en más días de los originalmente planificados.

Este fenómeno respondió principalmente a la naturaleza académica del proyecto y a la coexistencia con otras responsabilidades curriculares y personales. En consecuencia, se produjeron corrimientos en el cronograma general, no por subestimación técnica, sino por la fragmentación temporal del trabajo. El impacto se dio en el calendario de entregas, pero no en la complejidad ni en la carga real de implementación.

Esta experiencia permitió comprender con mayor claridad la diferencia entre estimación de esfuerzo y planificación temporal, y ajustar progresivamente la organización del trabajo en función de la disponibilidad real.

Organización inicial y forma de abordaje

Uno de los primeros desafíos fue definir cómo encarar un proyecto de esta escala. Ningún integrante del equipo había participado previamente en un desarrollo con este alcance, lo que generó cierta incertidumbre en las etapas iniciales. En diciembre de 2025 se comenzó a implementar funcionalidades, particularmente el módulo de asistencia, sin haber consolidado completamente los aspectos básicos del sistema, como la arquitectura general ni los roles de los usuarios. Esto derivó en retrabajo posterior, ya que varias decisiones iniciales no estaban suficientemente fundamentadas.

Si bien ese período no produjo entregables definitivos, resultó valioso como instancia de exploración tecnológica y adaptación al *stack* elegido (Next.js y React), así como para establecer dinámicas de trabajo colaborativo. A partir de los problemas detectados en esta etapa, el equipo adoptó una serie de mejoras en el flujo de trabajo: se migró de una instancia de base de datos compartida en Neon,

que generaba conflictos entre los integrantes, a instancias individuales levantadas mediante Docker; se estableció una convención de etiquetas para los *commits*, facilitando la trazabilidad de los cambios; y se definió el uso sistemático de ramas separadas por integrante y funcionalidad. Estas decisiones permitieron identificar errores de enfoque de forma temprana y sentar bases más sólidas para las etapas siguientes.

En cuanto a la asignación de tareas, inicialmente se adoptó una estructura en cascada, donde las responsabilidades estaban fuertemente encadenadas. Esto generaba bloqueos frecuentes, ya que el avance de un integrante dependía de la finalización de tareas previas por parte de otro. Además, al no existir aún una base implementada, resultaba complejo desacoplar módulos para distribuir el trabajo de forma verdaderamente independiente.

Esta situación comenzó a resolverse cuando, circunstancialmente, uno de los integrantes se ausentó temporalmente. La necesidad de reorganizar el trabajo con un miembro menos obligó a modularizar con mayor claridad los componentes existentes. A partir de esa experiencia, se adoptó una división más definida por módulos funcionales, donde cada integrante asumía la responsabilidad integral de diseñar, implementar y mantener un área específica del sistema. Al reincorporarse el equipo completo, esta modalidad se mantuvo, logrando un flujo de trabajo más estable y eficiente.

Definición del modelo de datos y complejidad institucional

El modelado de datos representó uno de los mayores desafíos técnicos. La organización académica del colegio Illia no responde a un esquema tradicional de educación secundaria. Existen materias compartidas entre cursos, proyectos interdisciplinarios evaluados por más de un docente, distintos tipos de inasistencias con pesos diferenciales y particularidades administrativas que no suelen contemplarse en sistemas educativos estándar.

Cada refinamiento del modelo implicó ajustes en esquemas, migraciones y servicios backend, aumentando la complejidad del desarrollo. Este proceso fue iterativo: en

varias ocasiones fue necesario modificar estructuras ya implementadas para adaptarlas a reglas o validaciones institucionales.

Validaciones funcionales y retrabajo

La coordinación con el referente funcional fue importante para validar procedimientos específicos de la institución. Si bien no constituyó el principal obstáculo del proyecto, en algunos momentos la información necesaria no estuvo disponible de manera inmediata. Para evitar frenar el avance, el equipo optó por asumir definiciones provisorias basadas en criterios razonables.

Un ejemplo concreto fue la definición de los campos asociados a los usuarios del sistema. Aunque idealmente estos datos deberían haberse cerrado en una etapa temprana, la confirmación definitiva se obtuvo recién avanzada la implementación. Como consecuencia, fue necesario agregar, eliminar o modificar atributos ya modelados, generando retrabajo tanto en la base de datos como en formularios y validaciones.

Este patrón se repitió en otros módulos: avanzar con supuestos permitió sostener el ritmo de desarrollo, pero implicó ajustes posteriores cuando se recibían validaciones formales.

Entorno de ejecución y limitaciones técnicas

En determinados momentos, el backend se ejecutó bajo Edge Runtime, entorno optimizado para baja latencia pero con restricciones respecto al acceso a recursos del sistema. Esto generó incompatibilidades con librerías que dependían de características propias de Node.js tradicional, particularmente en criptografía y *logging*. Los errores derivados de estas limitaciones no siempre eran explícitos, lo que exigió investigación adicional para diagnosticar su origen y redefinir configuraciones.

Interfaz de usuario y homogeneización del diseño

La interfaz de usuario fue otro punto de iteración constante. La plataforma está destinada a docentes, directivos y tutores, perfiles con distintos niveles de

experiencia digital, lo que implicó contemplar distintos niveles de familiaridad tecnológica desde el diseño.

Inicialmente, la interfaz fue abordada desde una perspectiva principalmente funcional, priorizando que las operaciones estuvieran disponibles antes que la consistencia visual. Los primeros prototipos cumplían con los requerimientos técnicos, pero no resultaban suficientemente intuitivos para usuarios no técnicos. A medida que el proyecto avanzó, se detectó que, si bien se habían definido criterios generales de diseño (colores, estructuras de formularios, componentes base), estos no se aplicaron de manera completamente homogénea en todos los módulos.

En consecuencia, se replantearon flujos de navegación, se redujo la complejidad de ciertas acciones y se reorganizaron pantallas para que las operaciones principales fueran más accesibles. Hacia la etapa final fue necesario realizar un proceso de unificación visual que incluyó la estandarización de formularios, la alineación de componentes y la reorganización de vistas ya implementadas. Este proceso implicó ciclos de diseño, pruebas internas y ajustes visuales sobre funcionalidades existentes, demandando tiempo adicional que podría haberse reducido con una definición más estricta desde el inicio. No obstante, permitió entregar una plataforma más coherente, clara y usable para todos los perfiles involucrados.

Síntesis

En conjunto, las principales dificultades no estuvieron relacionadas con la complejidad técnica aislada de cada funcionalidad, sino con la gestión inicial del proyecto, la distribución real del tiempo y el aprendizaje progresivo sobre cómo organizar un desarrollo de esta escala.

Cada obstáculo permitió mejorar prácticas internas: modularización más clara, validación temprana de supuestos, definición más precisa de responsabilidades y mayor atención a la coherencia de diseño. Estas mejoras se incorporaron durante el proceso y fortalecieron significativamente la calidad y estabilidad del producto final.

Cumplimiento de los objetivos

Se concluye que los objetivos planteados para el presente proyecto fueron cumplidos de manera satisfactoria. Se diseñó, desarrolló e implementó una plataforma web integral que permite gestionar de forma centralizada los procesos académicos y administrativos del Colegio Nacional Dr. Arturo Umberto Illia, proporcionando acceso a la información en tiempo real y promoviendo la despapelización de tareas institucionales.

El sistema incorpora los módulos necesarios para acompañar el funcionamiento cotidiano de la institución, incluyendo la gestión académica, usuarios, asistencias, evaluaciones, comunicación institucional y reportes, permitiendo mantener la trazabilidad de la información y mejorar la organización de los datos. Asimismo, la solución fue entregada desplegada en el entorno provisto por la institución, lo que posibilita su utilización efectiva por parte de la comunidad educativa.

No obstante, resulta importante señalar que la validación completa del impacto del sistema sólo podrá realizarse una vez finalizado un ciclo lectivo completo de utilización. Muchas de las funcionalidades desarrolladas, como la carga de calificaciones, el seguimiento de asistencias a lo largo del año o los procesos de cierre académico, dependen directamente del transcurso temporal del calendario escolar. Por este motivo, la medición objetiva de indicadores de eficiencia, adopción y utilidad requerirá un período de uso prolongado en condiciones reales.

A pesar de ello, desde un análisis cualitativo, se estima que la implementación del sistema producirá mejoras significativas en la centralización de la información, la reducción de tareas manuales basadas en papel y la disponibilidad de datos actualizados para la toma de decisiones institucionales. Asimismo, se espera una mejora en la experiencia de los usuarios beneficiados, al contar con una herramienta unificada y accesible según sus roles dentro de la comunidad educativa.

En relación con los requerimientos que fueron aplazados durante el desarrollo, se considera que esta decisión no afecta el cumplimiento de los objetivos generales del proyecto. La postergación respondió a criterios de priorización vinculados al esfuerzo de implementación en relación con su valor funcional inmediato,

manteniendo el foco en aquellas funcionalidades críticas para el funcionamiento del sistema. Los requerimientos diferidos constituyen oportunidades de evolución futura y no comprometen la operatividad ni la utilidad de la solución desarrollada.

En conclusión, el proyecto logró materializar una solución tecnológica alineada con las necesidades institucionales planteadas, cumpliendo con los objetivos definidos y estableciendo una base sólida para futuras ampliaciones y mejoras.

Trabajo en equipo

Desde el inicio del proyecto, el desarrollo del sistema se abordó de manera colaborativa, promoviendo un esquema de trabajo basado en el diálogo, la discusión técnica y la toma de decisiones consensuada. El equipo estuvo conformado por cuatro integrantes que ya habían coincidido previamente en distintas asignaturas de la carrera. Si bien el equipo como tal no había trabajado de forma conjunta en un mismo proyecto, la experiencia académica compartida y la colaboración previa entre varios de sus integrantes facilitaron la integración grupal y permitieron que la dinámica de trabajo se consolidara rápidamente.

A lo largo del desarrollo, se construyó un ambiente de trabajo positivo, con una comunicación fluida y eficaz. Las decisiones técnicas se debatían abiertamente, se evaluaban distintas alternativas de solución y se buscaba siempre arribar a acuerdos fundamentados. Esta dinámica favoreció la resolución de problemas y permitió sostener un ritmo de trabajo constante, incluso frente a situaciones imprevistas o ajustes en el alcance del proyecto.

Se realizaron reuniones periódicas, principalmente con una frecuencia semanal, que permitieron realizar un seguimiento del avance individual de cada integrante, coordinar tareas y redefinir prioridades cuando fue necesario. Estas instancias resultaron fundamentales para mantener una visión global del estado del proyecto y asegurar la coherencia entre los distintos componentes del sistema.

Tal como se mencionó en apartados anteriores, en las primeras etapas se presentaron dificultades en la división de tareas. Inicialmente, la asignación de

responsabilidades resultó poco modular y con baja cohesión, lo que generó solapamientos, dependencias innecesarias y ciertas trabas en el avance del desarrollo. Sin embargo, a partir de la experiencia adquirida y del análisis conjunto de estas problemáticas, el equipo logró ajustar progresivamente la organización del trabajo, adoptando una distribución de tareas más clara y alineada con la estructura modular del sistema.

Con el correr del proyecto, cada integrante pudo avanzar de manera más autónoma sobre sus responsabilidades, sin perder la integración con el trabajo del resto del equipo. Cuando surgían bloqueos o dificultades técnicas, estos eran comunicados a través de los canales establecidos incluso antes de las reuniones formales, lo que permitía poner la situación en común y buscar soluciones de manera colectiva. Este enfoque colaborativo contribuyó a reducir los tiempos de resolución y a fortalecer el aprendizaje compartido.

En conjunto, la experiencia de trabajo en equipo fue altamente positiva. La combinación de experiencia previa compartida, reuniones periódicas, comunicación fluida y apertura al debate técnico permitió no solo mejorar la calidad del producto final, sino también optimizar los procesos de desarrollo y fortalecer la coordinación entre los integrantes del equipo.

Comparación entre planificación esperada y ejecutada

El plan de trabajo original establecía una duración del proyecto comprendida entre el 1 de enero de 2025 y el 14 de enero de 2026, con una dedicación estimada de 5 horas semanales por integrante durante la primera mitad del año y 8 horas semanales durante la segunda mitad. Las etapas definidas (investigación, análisis, diseño, desarrollo, testing, documentación, despliegue, capacitación e informe final) fueron organizadas inicialmente de forma predominantemente secuencial, aunque con solapamientos previstos entre algunas de ellas.

La planificación se cumplió de manera adecuada en cuanto a la ejecución de las actividades previstas y al alcance técnico del proyecto. Las etapas de investigación

(semanas 1 a 12), análisis (semanas 3 a 14) y documentación (intercalándose entre julio, agosto, septiembre, octubre y diciembre) se desarrollaron dentro de los períodos estimados. Sin embargo, se produjeron desfases en otras etapas que en conjunto extendieron la finalización hasta febrero de 2026.

El desfase más significativo correspondió a la etapa de diseño: comenzó según lo planificado en la semana 11 (marzo de 2025), pero se extendió hasta la semana 49 (noviembre de 2025), cuando la estimación original contemplaba su cierre en mayo de 2025 (semana 22). El desarrollo se prolongó desde junio de 2025 hasta enero de 2026, superando en aproximadamente diez semanas la fecha de finalización prevista. El despliegue, planificado para noviembre de 2025, se concretó en febrero de 2026. El informe final comenzó a elaborarse recién en septiembre de 2025, nueve semanas después de lo planificado, lo que refleja la complejidad que implicó ese entregable. En cuanto a la capacitación, la etapa se encuentra actualmente en curso: el equipo está coordinando una reunión con el referente funcional del colegio para dar inicio formal a esta instancia.

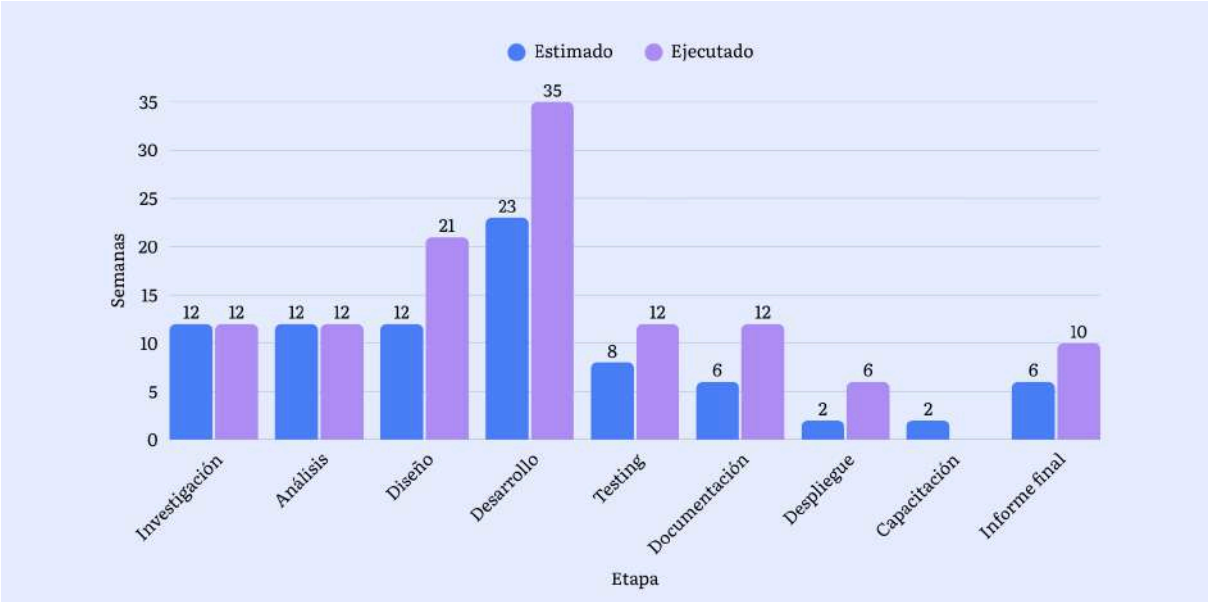


Figura 39. *Análisis comparativo de semanas estimadas y ejecutadas por etapa.*

Estos corrimientos respondieron a una combinación de factores. En primer lugar, el equipo no contaba con experiencia previa en Next.js, tecnología central del proyecto, lo que implicó una pequeña curva de aprendizaje que impactó en las etapas iniciales

del desarrollo. A esto se sumaron situaciones personales de los integrantes que en determinados momentos redujeron la disponibilidad efectiva para trabajar en el proyecto. Por otra parte, la coordinación con el referente funcional presentó dificultades propias: dado que resultaba complejo establecer reuniones periódicas, la comunicación se canalizó principalmente de forma asincrónica, lo que, si bien permitió mantener un intercambio fluido para evacuar dudas, validar funcionalidades y revisar interfaces, introdujo demoras que en ciertos momentos ralentizaron el avance. Esta situación fue particularmente notoria durante las etapas iniciales del análisis y el diseño, donde surgían interrogantes sobre el modelado de entidades del dominio académico que requerían respuestas precisas del referente, y durante el despliegue, donde los problemas con el servidor implicaron un ida y vuelta prolongado y discontinuo que dificultó la resolución ágil de los inconvenientes.

Un factor adicional, desarrollado con mayor detalle en el apartado de *Problemáticas encontradas y soluciones*, fue la fragmentación temporal del trabajo: en varias oportunidades, tareas cuyo esfuerzo real se correspondía con lo estimado demandaron más días de los previstos debido a la coexistencia con otras responsabilidades académicas y personales. Este fenómeno impactó principalmente en el calendario de entregas, sin afectar la complejidad ni la carga real de implementación. Finalmente, la complejidad inherente al desarrollo de un sistema integral y el tiempo necesario para alcanzar un nivel de estabilidad adecuado previo al despliegue completaron el conjunto de causas que explican la extensión del cronograma.

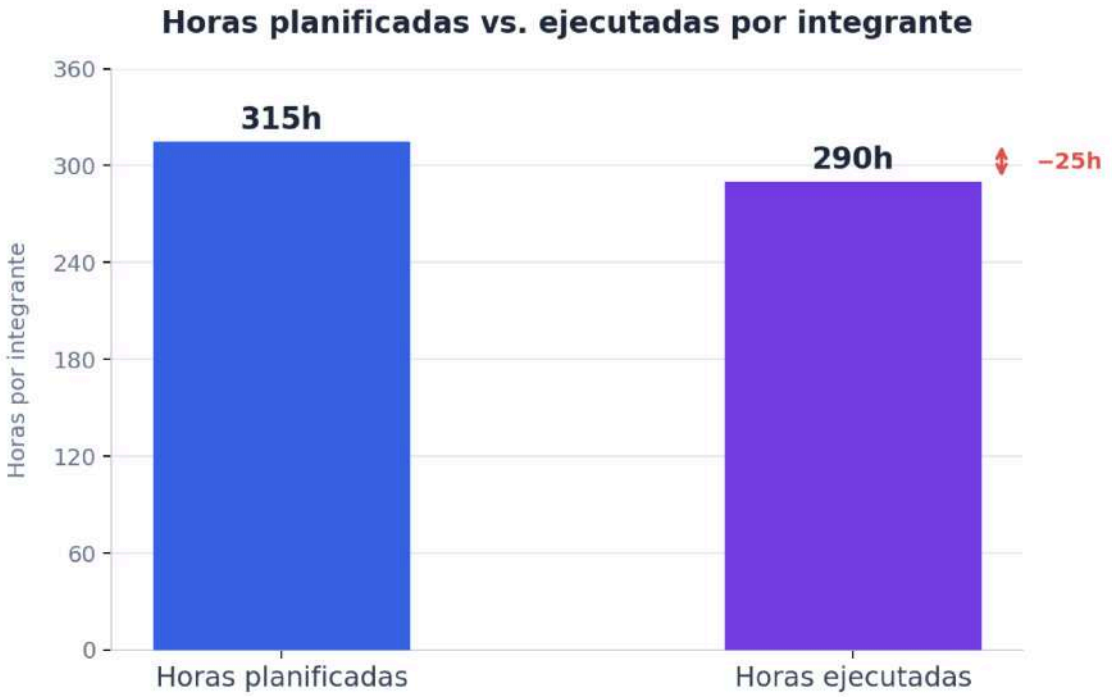


Figura 40. *Comparación de horas planificadas y ejecutadas por integrante del equipo de desarrollo.*

El enfoque adoptado por el equipo estuvo desde el inicio más cercano a una metodología ágil que a un modelo estrictamente secuencial, lo cual explica en gran medida la distribución real de las etapas observada en el Gantt ejecutado. Las etapas de diseño, desarrollo, testing y documentación no transcurrieron como fases cerradas y consecutivas, sino de forma iterativa y superpuesta a lo largo del proyecto. El diseño continuó activo con intervalos hasta la semana 49, solapándose con el desarrollo y el testing. Este último registró actividad intermitente desde la semana 28 hasta la 49, reflejo de los ciclos de validación que se realizaron a medida que se incorporaban nuevas funcionalidades. La documentación acompañó el desarrollo desde etapas tempranas y se extendió hasta enero de 2026. Este modo de trabajo permitió detectar errores de forma temprana, ajustar decisiones técnicas en función del feedback recibido y mejorar la calidad del producto de manera progresiva.

En conclusión, si bien existieron diferencias temporales entre la planificación original y la ejecución efectiva, particularmente en diseño, desarrollo y despliegue, el enfoque iterativo adoptado resultó apropiado para la naturaleza del proyecto y

permitió alcanzar una solución funcional, validada y desplegada en el entorno institucional, manteniendo el cumplimiento de los objetivos definidos.

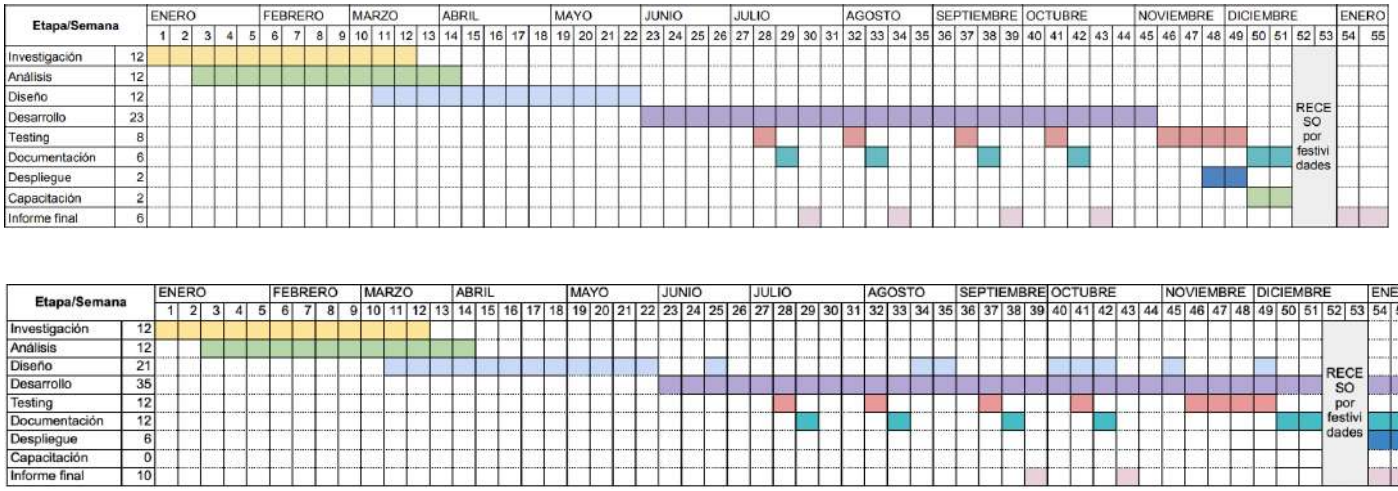


Figura 41. *Diagrama de Gantt estimado vs. Diagrama de Gantt ejecutado*

Análisis por etapas

Análisis

La etapa de análisis se desarrolló de manera ordenada y no se extendió más allá de lo planificado inicialmente. A lo largo de esta fase, el equipo logró recopilar y definir los requerimientos del sistema sin que se produjeran modificaciones significativas una vez consensuados los lineamientos principales. Este escenario permitió avanzar con claridad y reducir la necesidad de revisiones posteriores.

A partir de las reuniones mantenidas con el referente funcional, los requerimientos relevados fueron progresivamente formalizados en un documento de especificación de requerimientos del sistema (SRS). Este documento permitió traducir las necesidades planteadas desde el dominio educativo a definiciones técnicas concretas, estableciendo una base común y compartida para todo el equipo de desarrollo.

Si bien la cantidad de reuniones con el referente funcional fue acotada, las mismas resultaron eficientes y suficientes para relevar, ajustar y validar los requerimientos necesarios. El proceso se complementó con intercambios por correo electrónico,

que facilitaron la resolución de dudas puntuales y el refinamiento final de las definiciones, sin necesidad de reabrir el análisis de manera extensiva.

En conjunto, la etapa de análisis permitió consolidar un entendimiento claro del sistema a desarrollar y generar un SRS estable y validado, que sirvió como insumo principal para las etapas de diseño e implementación. La claridad alcanzada en esta fase contribuyó a un desarrollo posterior más predecible y a una disminución de ajustes estructurales durante el avance del proyecto.

Reflexión del análisis FODA

Fortalezas

Al analizar en retrospectiva el desarrollo de IlliApp, la fortaleza más sólida resultó ser el profundo conocimiento sobre los procesos internos del Colegio Illia. Esta ventaja se vio impulsada no solo por el referente funcional, cuya visión técnica nos brindó información invaluable sobre los aciertos y debilidades del sistema anterior, sino también por un factor clave que no habíamos contemplado inicialmente: dos integrantes del equipo eran exalumnos de la institución. Esta experiencia previa nos permitió avanzar ágilmente con el desarrollo sin la necesidad de depender de consultas constantes al referente. Por otro lado, la ventaja de contar con un demandante local mutó hacia una gran flexibilidad operativa, ya que el trabajo de campo presencial no fue necesario y pudimos realizar todo el relevamiento de forma eficiente y virtual. Finalmente, aunque logramos cumplir con la iniciativa de publicar la aplicación bajo una licencia MIT, su impacto como fortaleza disminuyó en la práctica, dado que la solución quedó tan estrechamente ligada a los requerimientos específicos del colegio que resulta difícilmente aplicable para otros establecimientos educativos.

Oportunidades

Evaluando las oportunidades en retrospectiva, confirmamos que la inexistencia de una solución de mercado adecuada era una realidad absoluta; durante el desarrollo nos percatamos de que las necesidades del colegio eran tan particulares que resultaría imposible cubrir todos los requisitos sin un desarrollo hecho completamente a medida. Por otra parte, la preexistencia de una aplicación y la

insatisfacción con la misma evidenciaban una clara necesidad de contar con un nuevo sistema de gestión académica. Sin embargo, como la puesta en marcha y el despliegue final del sistema terminaron quedando fuera del alcance del proyecto, no nos fue posible validar en la práctica la aceptación de la nueva plataforma ni de sus mejoras por parte de los distintos actores involucrados. Finalmente, en cuanto al respaldo institucional para el cambio organizacional, la oportunidad se materializó con ciertos matices: si bien no contamos con una presencia activa y un seguimiento constante por parte de la dirección durante el día a día, sí se mantuvo su apoyo general para la realización del proyecto.

Debilidades

Las debilidades vinculadas a la inexperiencia del equipo en proyectos de esta magnitud efectivamente tuvieron impacto durante las primeras etapas. Inicialmente resultó complejo definir una metodología de trabajo adecuada, desacoplar tareas y establecer una organización eficiente del desarrollo, lo que derivó en retrabajo y ajustes progresivos sobre la dinámica del equipo. Sin embargo, una vez finalizada la etapa inicial, donde el equipo logró tener los cimientos necesarios para trabajar, pudo establecerse una dinámica de trabajo que permitió llevar a cabo el proyecto de manera coordinada y eficaz.

Por otra parte, la dificultad para coordinar reuniones frecuentes con el referente funcional tuvo una incidencia mayor a la prevista en determinados momentos del proyecto. En varias ocasiones, la falta de validaciones inmediatas obligó al equipo a avanzar mediante supuestos basados en la experiencia de los integrantes egresados del Colegio Illia para no frenar el desarrollo, que posteriormente tuvieron que ser validados con el referente funcional, aumentando el riesgo de retrabajo.

Finalmente, la disponibilidad horaria reducida debido a responsabilidades académicas y personales simultáneas impactó principalmente en los tiempos efectivos de avance y en el corrimiento del cronograma inicialmente estimado.

Amenazas

Si bien durante el desarrollo no se produjeron cambios legislativos o institucionales que afectaran directamente al proyecto, posteriormente se informó al equipo que no

sería posible acceder a los datos del colegio debido a la sensibilidad de la información involucrada, tanto de menores de edad como del personal de la institución. Esta situación impactó en el alcance inicialmente previsto, ya que el objetivo original contemplaba una implementación completa del sistema. Sin embargo, el alcance final se limitó al despliegue de la aplicación, quedando la puesta en funcionamiento y carga de datos a cargo de la institución.

Asimismo, respecto a la posible resistencia al cambio por parte de los usuarios debido a malas experiencias con la aplicación anterior, no fue posible validar este aspecto en la práctica. Como consecuencia de las restricciones mencionadas, el sistema no llegó a implementarse operativamente dentro del colegio, por lo que no pudieron realizarse pruebas ni validaciones directas con los usuarios finales.

Retrospectiva del análisis de riesgos

Durante el desarrollo del proyecto se materializaron tanto el riesgo de complejidad del dominio institucional (R01) como la dificultad de coordinación con el referente funcional (R04). Esto representó una dificultad adicional, ya que gran parte del plan de contingencia definido para R01 dependía de realizar reuniones periódicas con el referente funcional para validar procesos y resolver dudas, justamente una de las actividades que resultó más difícil de coordinar durante el proyecto.

Frente a esta situación, el equipo debió reorganizar la dinámica de trabajo para maximizar el tiempo disponible con el referente funcional, priorizando las validaciones críticas y concentrando las reuniones en aquellos aspectos más complejos o sensibles del sistema. Además, la participación de dos integrantes del equipo egresados del Illia fue fundamental para poder continuar el desarrollo, ya que su conocimiento del funcionamiento institucional permitió avanzar sobre supuestos razonables y luego validarlos con el referente cuando era posible coordinar reuniones.

Gracias a estas medidas, ambos riesgos pudieron mitigarse sin comprometer la continuidad general del proyecto.

Diseño

En la planificación inicial del proyecto, la etapa de diseño se concentraba principalmente en las primeras semanas, con una finalización estimada hacia fines de mayo del año 2025. Sin embargo, en la práctica, el diseño no se constituyó como una fase completamente cerrada ni acotada a un único período temporal. Si bien la mayor parte de las definiciones estructurales y arquitectónicas se realizaron en las etapas tempranas del proyecto, el diseño acompañó al desarrollo a lo largo de todo el ciclo de vida del sistema.

Esta situación se debió, en gran medida, al enfoque de desarrollo por módulos adoptado por el equipo. A medida que cada módulo comenzaba a implementarse, surgía la necesidad de profundizar, ajustar o redefinir ciertos aspectos del diseño, tanto a nivel de modelo de datos como de flujos funcionales y responsabilidades internas. En este sentido, el diseño no fue un proceso estático, sino iterativo y evolutivo, adaptándose al avance concreto del desarrollo y a las particularidades que emergían en cada funcionalidad.

Durante el transcurso del proyecto se realizaron instancias puntuales de rediseño, motivadas principalmente por la detección de dependencias innecesarias, la necesidad de mejorar la cohesión entre componentes o la simplificación de ciertas estructuras inicialmente planteadas. Estas revisiones permitieron mejorar la claridad del diseño y evitar decisiones que, si bien resultaban válidas en una primera aproximación, no resultaban óptimas en un contexto de implementación real.

Si bien esta dinámica implicó que la etapa de diseño se extendiera más allá de lo originalmente previsto en el cronograma, no generó desvíos significativos en los tiempos generales del proyecto. Por el contrario, la incorporación progresiva de instancias de diseño y rediseño contribuyó a reducir retrabajos en etapas posteriores y facilitó una implementación más ordenada y coherente.

Durante esta etapa también se fortaleció el conocimiento técnico del equipo. A medida que avanzaba el proyecto, se incorporaron nuevas prácticas de diseño y desarrollo que permitieron mejorar la calidad general del sistema y consolidar una base técnica sólida para las fases posteriores de implementación y validación.

En conclusión, la etapa de diseño combinó una fase inicial de definición global del sistema con múltiples instancias posteriores de ajuste y refinamiento. Esta modalidad resultó adecuada para el tipo de proyecto desarrollado, permitiendo mantener una arquitectura consistente sin perder flexibilidad frente a las necesidades que surgieron durante el desarrollo.

Desarrollo

La etapa de desarrollo constituyó el núcleo del proyecto y concentró la mayor parte del esfuerzo total. Tal como suele ocurrir en proyectos de software de estas características, esta fase fue la más extensa y demandante, tanto en términos técnicos como organizativos. No obstante, el desarrollo se llevó a cabo de manera sostenida y sin desvíos críticos respecto a lo planificado inicialmente, aunque su finalización se concretó en el mes de febrero.

El enfoque iterativo e incremental adoptado resultó adecuado para el contexto del proyecto. La implementación progresiva por módulos permitió avanzar de forma ordenada, validar funcionalidades parciales y detectar tempranamente errores o ajustes necesarios, reduciendo así el riesgo de retrabajos significativos en etapas avanzadas. La construcción de un prototipo inicial fue especialmente valiosa, ya que permitió confirmar tempranamente decisiones de diseño y flujos principales del sistema junto al referente funcional.

Si bien el desarrollo fue planificado en bloques bien definidos, en la práctica se observó una superposición natural entre implementación, ajustes de diseño y validaciones funcionales. Esta dinámica respondió tanto a la complejidad inherente de algunos módulos como a la necesidad de adaptar ciertas decisiones técnicas a medida que el sistema adquiría mayor volumen y coherencia interna. Lejos de representar una dificultad, este proceso permitió mejorar la calidad final del producto y afinar aspectos que no habían sido completamente visibles en las etapas iniciales.

Durante esta etapa, el equipo se encontró en un proceso constante de aprendizaje y consolidación técnica. Si bien se contaba con experiencia previa en varias de las tecnologías utilizadas, el desarrollo del sistema implicó profundizar conocimientos, adoptar nuevas prácticas y mejorar progresivamente los criterios de implementación.

Este aprendizaje continuo tuvo un impacto positivo tanto en la calidad del código como en la organización general del trabajo.

La dinámica de reuniones semanales y el uso de herramientas de gestión y control de versiones favorecieron una comunicación fluida y una correcta coordinación entre los integrantes del equipo. La revisión cruzada mediante pull requests y la automatización de validaciones técnicas permitieron mantener un estándar de calidad homogéneo a lo largo del desarrollo, minimizando errores y facilitando la integración de los distintos aportes individuales.

En términos generales, la etapa de desarrollo se ejecutó de manera ordenada y consistente, logrando materializar los requerimientos definidos en etapas previas sin modificaciones sustanciales en el alcance del sistema. La combinación de validación temprana, trabajo iterativo y comunicación constante resultó clave para alcanzar un producto funcional, coherente y alineado con las necesidades del colegio.

Testing

Si bien durante el desarrollo del proyecto se implementaron pruebas automatizadas unitarias y end-to-end para determinados módulos del sistema, se reconoce que hubiera sido posible ampliar esta cobertura a la totalidad de las funcionalidades. No obstante, la implementación de pruebas automatizadas completas para todo el sistema habría implicado un esfuerzo considerable en términos de tiempo y complejidad, que no siempre resultaba proporcional al beneficio obtenido, especialmente para funcionalidades de menor criticidad o cambios frecuentes durante las etapas iniciales.

Por este motivo, y considerando las restricciones temporales del proyecto, se decidió priorizar la automatización de pruebas en un conjunto acotado de módulos. En una primera instancia, se trabajó sobre módulos de menor complejidad, como gestión de ausencias docentes y anuncios, con el objetivo de familiarizarse con las herramientas y metodologías de testing. A partir de esa experiencia, se avanzó hacia los módulos considerados más críticos, tales como autenticación, asistencia y actividades evaluativas.

No se buscó alcanzar una cobertura total del sistema, ya que implementar pruebas automatizadas sobre todos los módulos implicaba un esfuerzo considerable de desarrollo y mantenimiento. Además, pruebas incompletas, desactualizadas o incorrectamente diseñadas pueden generar una falsa percepción de confiabilidad y dificultar la detección real de errores. Por ello, se optó por priorizar pruebas de calidad sobre funcionalidades críticas y de mayor impacto funcional, complementándolas con testing manual y validaciones cruzadas entre integrantes del equipo.

Paralelamente, el testing manual tuvo un rol fundamental a lo largo del proyecto. Las instancias de validación permitieron detectar errores de manera temprana, identificar oportunidades de mejora y generar espacios de discusión dentro del equipo sobre decisiones de implementación y resolución de problemas. Este proceso resultó especialmente enriquecedor, ya que no solo contribuyó a mejorar la calidad del sistema, sino también a consolidar la comprensión de que las actividades de testing constituyen una parte esencial del ciclo de desarrollo de software, con un nivel de importancia comparable al de la propia implementación.

Adicionalmente, durante el proceso de desarrollo se utilizaron herramientas de revisión de código integradas en GitHub, particularmente mediante el análisis automático proporcionado por GitHub Copilot en las solicitudes de cambio (pull requests). Estas herramientas permitieron identificar posibles mejoras técnicas, problemas de calidad de código y oportunidades de optimización de manera temprana, lo que contribuyó a reducir el tiempo invertido en revisiones manuales. Sin embargo, su alcance se centró principalmente en aspectos estructurales y sintácticos del código, sin reemplazar las instancias de testing funcional o end-to-end necesarias para validar el comportamiento del sistema desde la perspectiva del usuario.

En retrospectiva, la experiencia adquirida permitió al equipo dimensionar con mayor claridad el valor del testing como práctica continua, tanto para la prevención de errores como para la mejora de la calidad del producto final.

Despliegue

El despliegue del sistema tuvo como objetivo instalar y dejar operativa la aplicación en la infraestructura del Colegio Illia, configurando dos entornos independientes, producción y *staging*, con sus respectivas bases de datos, un sistema de proxy inverso, un *pipeline* de integración y entrega continua, y una política de *backups* automáticos. A lo largo de todas las decisiones de arquitectura se priorizó la mantenibilidad del sistema, de modo que el personal técnico del colegio pueda operar y sostener la infraestructura de forma autónoma.

Sin embargo, esta fase presentó varios desafíos antes de poder concretarse. En enero se iniciaron las primeras pruebas para instalar y configurar la aplicación en el servidor del colegio, pero pronto surgieron problemas de acceso: la falta de permisos impedía la conexión remota, y el referente funcional demoró varios días en concurrir presencialmente para revisar la situación. Además, durante este período se produjeron cortes de luz y obras en el colegio, lo que dificultó la continuidad de las pruebas. Durante febrero se continuó intentando el acceso vía SSH sin éxito, y se confirmó que el contenedor de la aplicación se había apagado, aunque no se pudo determinar la causa exacta. Una vez restablecido el acceso a mediados de febrero, se inició formalmente la fase de despliegue. Con el correr de los días surgieron nuevos inconvenientes, incluyendo un corte de la VM en Proxmox, plataforma de virtualización que permite administrar máquinas virtuales y contenedores sobre un servidor físico y que el colegio utiliza para gestionar su infraestructura; y dificultades continuas de acceso, que requirieron la intervención del referente funcional y ajustes adicionales por parte del equipo de IT del colegio.

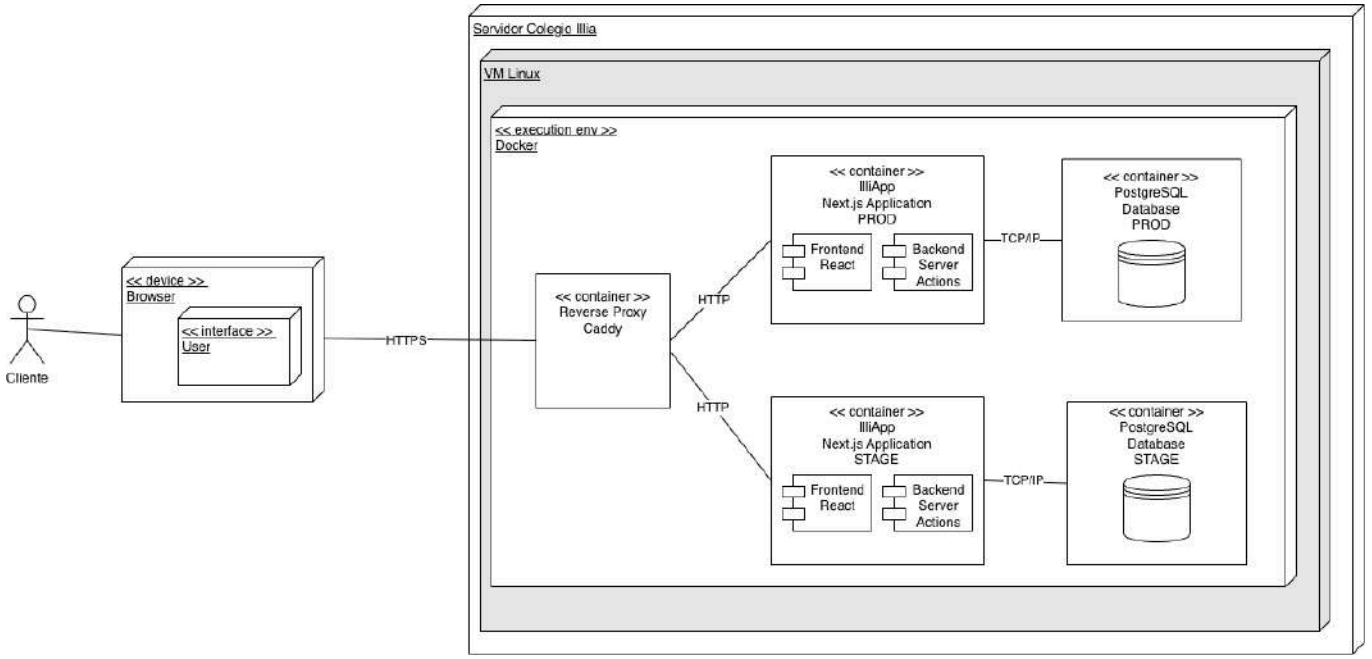


Figura 42. *Diagrama de despliegue del sistema con la distribución de los componentes de software en los nodos de infraestructura.*

La arquitectura de despliegue está basada en contenedores Docker sobre una VM Linux administrada mediante Proxmox. El uso de contenedores permite aislar cada componente del sistema en unidades independientes y reproducibles, simplificando tanto el despliegue como el mantenimiento y la actualización futura de cada parte. Se definieron dos entornos completamente independientes, producción y *staging*, cada uno compuesto por un contenedor con la aplicación Next.js y un contenedor con su propia base de datos PostgreSQL. Ambos entornos se configuraron mediante dos archivos Docker Compose prácticamente idénticos, diferenciados únicamente por sus variables de entorno. En cada uno de ellos, la base de datos se encuentra aislada en una red interna compartida únicamente con su instancia de Next.js correspondiente, de modo que no resulta accesible desde el exterior.

Para el enrutamiento del tráfico externo se utiliza Caddy como proxy inverso, el cual comparte una red con ambos contenedores de Next.js. Todas las solicitudes entrantes son recibidas por Caddy, que según el *host* de la petición las redirige al entorno de producción o al de *staging*, manejando además HTTPS de forma automática. Se eligió Caddy por su configuración simple y declarativa, lo que facilita su comprensión y modificación por parte del personal técnico del colegio.

En cuanto al *pipeline* de integración y entrega continua, se configuraron GitHub Actions de forma tal que cada *push* a la rama principal o a la rama de *staging* dispara automáticamente la construcción de una nueva imagen Docker, la cual se publica en el registro de imágenes de GitHub (GHCR). Luego, la acción accede a la VM por SSH, descarga la nueva imagen y levanta nuevamente el contenedor correspondiente, minimizando el tiempo de inactividad del servicio.

Respecto a la gestión de datos, se implementó una política de *backups* automáticos nocturnos de la base de datos de producción, con una retención de 30 días, tras los cuales cada copia es eliminada. Adicionalmente, cada vez que se realiza el *backup* nocturno, los datos de producción se transfieren a la base de datos de *staging*, manteniéndola actualizada con información real para facilitar las pruebas. Por último, al momento de levantar el entorno de producción, se ejecutan automáticamente todas las migraciones pendientes sobre la base de datos, garantizando que el esquema se encuentre siempre en su versión más reciente.

Cabe aclarar que el sistema fue desplegado en la infraestructura provista por la facultad, con el objetivo de validar la arquitectura de despliegue, el funcionamiento de los contenedores y el pipeline de integración y entrega continua. No obstante, el entorno actualmente instalado no se encuentra aún habilitado para su uso por parte de usuarios finales, ni contiene información institucional real.

La carga de datos iniciales, la configuración definitiva del entorno productivo y la eventual puesta en funcionamiento del sistema quedarán a cargo del personal técnico o administrativo que la institución designe, quienes podrán realizar dichas tareas una vez que consideren oportuno iniciar la operación del sistema.

Capacitación

En la planificación original del proyecto se contempló la elaboración de material audiovisual explicativo como soporte para la capacitación de los usuarios del sistema. No obstante, a medida que el proyecto avanzó, se acordó con la institución adoptar una modalidad diferente. Finalmente, se decidió que el proceso de capacitación y difusión del sistema quedara a cargo del propio personal del colegio,

quienes cuentan con un conocimiento directo de los procedimientos internos y de las necesidades de los distintos perfiles de usuario.

En este marco, el equipo desarrollador realizó una instancia inicial de presentación del sistema al referente funcional designado por la institución. Dicha demostración se llevó a cabo utilizando datos de prueba, con el fin de evitar el uso de información real durante esta etapa. A partir de esta instancia, el referente quedó a cargo de coordinar internamente las acciones de capacitación necesarias con el resto del personal según lo considerara pertinente.

Principales aprendizajes

La realización de IlliApp representó el primer proyecto de esta magnitud para todos los integrantes del equipo, abarcando de forma integral el ciclo de vida de un sistema de software. Desde el análisis inicial y el relevamiento de requerimientos hasta el despliegue del sistema y las instancias de capacitación al personal del Colegio Illia, el proyecto permitió aplicar y consolidar los conocimientos adquiridos a lo largo de la carrera en un contexto real y con usuarios concretos.

Uno de los principales aprendizajes estuvo vinculado a la comprensión de que el desarrollo de software no es un proceso lineal ni exento de dificultades. A lo largo del proyecto surgieron situaciones imprevistas y demoras en definiciones que obligaron al equipo a replantear decisiones técnicas y organizativas. Esta experiencia reforzó la importancia de la capacidad de adaptación y de resolución de problemas como competencias centrales del rol del ingeniero en informática, por encima de los conocimientos teóricos y la aplicación estricta de herramientas o metodologías.

Desde el punto de vista técnico, el proyecto implicó un proceso de aprendizaje continuo y profundo sobre tecnologías que, hasta el momento, no habían sido abordadas con este nivel de detalle. En particular, se adquirió experiencia práctica en el uso de React y Next.js para el desarrollo del *frontend*, incorporando conceptos como componentes reutilizables, manejo de estado y renderizado del lado del

servidor. En el *backend*, se utilizaron las funcionalidades *server-side* de Next.js para implementar la lógica de negocio, el control de acceso y la comunicación con la base de datos, así como con PostgreSQL para el modelado y la persistencia de la información. Asimismo, se fortaleció el uso de Git y GitHub como herramientas centrales para el control de versiones y el trabajo colaborativo, incluyendo el uso de *pull requests* y flujos de revisión de código.

El proyecto también permitió adquirir una visión más realista sobre la gestión del alcance y la toma de decisiones. Como se menciona en la sección de *Trabajos Futuros*, fue necesario reevaluar funcionalidades inicialmente previstas y priorizar aquellas que ofrecían un mayor valor en relación con su complejidad y el tiempo de implementación. Este proceso evidenció la importancia de analizar el *trade-off* entre esfuerzo técnico y beneficio funcional, constituyendo un aprendizaje clave para la gestión de proyectos de software en contextos reales.

Finalmente, el trabajo en equipo se consolidó como un aprendizaje transversal. La coordinación de tareas, la distribución de responsabilidades y la necesidad de sostener el avance del proyecto en un contexto académico exigente permitieron desarrollar habilidades de comunicación, organización y compromiso colectivo. Esta experiencia puso de manifiesto que el éxito de un proyecto de software no depende únicamente de las decisiones técnicas adoptadas, sino también de la capacidad del equipo para trabajar de manera coordinada, responsable y orientada a objetivos comunes.

Conclusiones

El desarrollo de IlliApp permitió materializar una solución concreta para una problemática real del ámbito educativo, vinculada a la gestión académica y administrativa del Colegio Nacional Dr. Arturo Umberto Illia. A lo largo del proyecto se logró diseñar e implementar un sistema que centraliza información clave, digitaliza procesos que anteriormente se realizaban de forma manual y mejora la disponibilidad y trazabilidad de los datos para los distintos actores de la institución.

Los objetivos planteados al inicio del trabajo fueron cumplidos satisfactoriamente. El sistema desarrollado permite gestionar usuarios y roles, registrar asistencias,

administrar evaluaciones y calificaciones, generar reportes y organizar horarios de cursada, todo dentro de una plataforma web accesible y alineada con el funcionamiento institucional del colegio. La despapelización de procesos administrativos representa un avance significativo en términos de eficiencia operativa y reducción de errores, sentando las bases para una gestión más ordenada y confiable.

Desde el punto de vista técnico, las decisiones adoptadas resultaron adecuadas para el alcance y la complejidad del proyecto. La elección de una arquitectura monolítica modular permitió mantener una estructura clara, reducir la sobrecarga de infraestructura y facilitar el desarrollo y mantenimiento del sistema. La modularización interna favoreció una evolución progresiva de la aplicación y dejó el sistema preparado para futuras ampliaciones, que podrán ser abordadas sin necesidad de reestructurar su núcleo.

En relación con el proceso de desarrollo, el trabajo en equipo fue un factor determinante. Si bien los cuatro integrantes no habían conformado un equipo en un proyecto previo, la experiencia académica compartida y la colaboración previa entre varios de ellos facilitaron la comunicación y permitieron que la dinámica grupal se consolidara rápidamente. Esto permitió sostener el avance del proyecto incluso en contextos de definiciones incompletas o validaciones externas no siempre inmediatas. Si bien la participación del referente funcional no fue constante a lo largo de todo el proyecto, las definiciones críticas del sistema fueron validadas con él en los casos en que resultaba necesario, garantizando la coherencia con las reglas y el funcionamiento institucional del colegio.

Desde una perspectiva crítica, se identifican aspectos susceptibles de mejora. Como suele ocurrir en proyectos de cierta complejidad y, particularmente, en las etapas finales de la carrera, hacia el cierre del trabajo se evidenciaron momentos de cansancio y desmotivación, acompañados por la necesidad de concluir el proyecto dentro de los plazos establecidos. Esta situación impactó principalmente en el ritmo de trabajo y en el esfuerzo requerido para las tareas finales, aunque no comprometió la funcionalidad ni los objetivos centrales del sistema.

Asimismo, a lo largo del desarrollo fue necesario revisar de manera crítica el alcance inicialmente definido. Como se mencionó en secciones anteriores, en las primeras etapas se incorporaron funcionalidades sin dimensionar completamente el costo técnico y temporal de su implementación en relación con el valor real que aportarían al sistema. Esta situación obligó a replantear prioridades y a enfocar los esfuerzos en aquellas funcionalidades esenciales y de mayor impacto, permitiendo asegurar la estabilidad del producto final y el cumplimiento del cronograma. Este proceso constituyó un aprendizaje relevante en relación con la gestión del alcance y la evaluación del balance entre complejidad, esfuerzo de desarrollo y utilidad efectiva en proyectos de software reales.

Por último, el desarrollo del proyecto dejó un aprendizaje que trasciende el aspecto puramente técnico: la escritura del informe final resulta tan importante como la implementación del sistema. La necesidad de documentar decisiones, justificar elecciones tecnológicas y comunicar de manera clara el trabajo realizado puso en evidencia que el producto final no se limita al software desarrollado, sino también a su correcta presentación y fundamentación.

A nivel personal y académico, la realización de este trabajo final constituyó una experiencia formativa integral. Permitió aplicar conocimientos teóricos en un contexto real, enfrentar decisiones técnicas con impacto directo en el producto final y adquirir una visión más completa del proceso de desarrollo de software, incluyendo sus dimensiones técnicas, organizativas y comunicacionales. En conjunto, IlliApp cumple con los objetivos del proyecto final de carrera y deja una base sólida tanto para su posible evolución futura como para el crecimiento profesional de los integrantes del equipo.

Glosario

Access Token

Token de autenticación de corta duración utilizado para autorizar el acceso a recursos protegidos en cada solicitud al servidor. Contiene información necesaria para validar la identidad del usuario y sus permisos, y se envía típicamente en los

encabezados HTTP. Su tiempo de vida reducido limita el impacto ante posibles filtraciones.

Backend

Conjunto de componentes del sistema responsables de la lógica de negocio, el procesamiento de datos, la validación de reglas y la comunicación con la base de datos. No es visible para el usuario final y se ejecuta del lado del servidor.

Backup

Copia de seguridad de los datos de un sistema, realizada con el objetivo de poder restaurarlos ante una pérdida, corrupción o fallo. Suelen ejecutarse de forma programada y almacenarse por un período determinado antes de ser eliminadas.

Caddy

Servidor web de código abierto que puede actuar como proxy inverso, es decir, como intermediario entre los usuarios y los servicios internos de una aplicación. Se destaca por su configuración simple y declarativa, y por gestionar de forma automática los certificados HTTPS, sin requerir configuración manual adicional.

CNAI

Acrónimo correspondiente a *Colegio Nacional Arturo Illia*.

CRUD

Acrónimo de Create, Read, Update y Delete. Define las cuatro operaciones básicas que pueden realizarse sobre los datos persistidos en un sistema de información.

Cursada

Entidad que representa la dictada de una materia específica para un curso determinado. Modela la relación entre una materia y un curso, permitiendo asociar docentes, horarios, evaluaciones, calificaciones y asistencia. Por ejemplo, “Matemática” dictada al curso “2.º 4.ª”.

Curso

Entidad que representa un conjunto de estudiantes que comparten una misma organización académica dentro del establecimiento. Un curso se identifica por su año, división y, cuando corresponde, orientación o grupo específico, por ejemplo “6.º 3.ª” o “2.º 4.ª Informática Grupo A”.

DER (Diagrama Entidad Relación)

Representación gráfica del modelo de datos de un sistema, donde se describen las entidades, sus atributos y las relaciones existentes entre ellas.

Docker

Plataforma de virtualización a nivel de contenedor que permite empaquetar una aplicación junto con sus dependencias en entornos aislados y portables, **garantizando consistencia entre desarrollo, pruebas y producción.**

Docker Compose

Herramienta que permite definir y ejecutar aplicaciones compuestas por múltiples contenedores Docker a través de un único archivo de configuración. En dicho archivo se especifican los servicios que conforman la aplicación, sus imágenes, variables de entorno, redes internas y volúmenes, lo que simplifica tanto el despliegue como la reproducibilidad del entorno en distintas máquinas.

End-to-End (E2E)

Tipo de prueba de software que valida el funcionamiento completo de una aplicación simulando el comportamiento real de un usuario, desde el inicio hasta el final de un flujo funcional. Su objetivo es verificar que todos los componentes del sistema (interfaz, lógica de negocio, base de datos, servicios externos, etc.) interactúan correctamente entre sí en un entorno lo más cercano posible al de producción.

Entorno staging/Stage

Entorno intermedio entre desarrollo y producción que replica las condiciones reales del sistema productivo, utilizado para realizar pruebas finales antes del despliegue definitivo.

Feedback

Retroalimentación provista por usuarios, referentes funcionales o miembros del equipo de desarrollo, utilizada para validar decisiones, corregir errores y mejorar el sistema de manera iterativa.

Framework

Conjunto de herramientas, librerías y convenciones que proveen una estructura base para el desarrollo de aplicaciones, facilitando la organización del código y acelerando el proceso de desarrollo.

Frontend

Parte del sistema con la que interactúan directamente los usuarios. Incluye la interfaz gráfica, la navegación y la visualización de la información, ejecutándose principalmente en el navegador.

Fullstack

Enfoque de desarrollo que abarca tanto el frontend como el backend de una aplicación, integrando la interfaz de usuario, la lógica de negocio y la persistencia de datos.

Hash

Resultado de aplicar una función criptográfica a un dato, generalmente una contraseña, para obtener una representación irreversible. Se utiliza para proteger información sensible y evitar el almacenamiento en texto plano.

Hipervisor

Software o capa de virtualización que permite crear y administrar máquinas virtuales, asignando recursos del hardware físico y garantizando su aislamiento.

Host

Nombre de dominio o dirección que identifica a un servidor o recurso dentro de una red. En el contexto del despliegue, el host de una solicitud permite determinar a qué aplicación o entorno debe ser dirigida.

HTTPS (HyperText Transfer Protocol Secure)

Versión segura del protocolo HTTP que utiliza cifrado (TLS/SSL) para proteger la comunicación entre cliente y servidor.

IATE

Acrónimo correspondiente a *Informe de Avance de las Trayectorias Educativas*. Sistema de evaluación institucional obligatorio del Colegio Nacional Illia. Consiste en evaluaciones particulares definidas por la institución, de carácter cualitativo y no numérico. Se estructura en cuatro dimensiones independientes: Alfabetización, Responsabilidad, Proceso de Aprendizaje y Participación, a las que se suma un campo de Observaciones de tipo descriptivo. Se realiza una evaluación IATE por cuatrimestre, permitiendo un seguimiento sistemático y continuo del desempeño académico de los estudiantes.

IT (Information Technology)

Conjunto de tecnologías, infraestructuras y servicios utilizados para el procesamiento, almacenamiento, transmisión y gestión de información digital.

Licencia GPL

Licencia de software libre que garantiza a los usuarios la libertad de ejecutar, estudiar, modificar y redistribuir el software, exigiendo que las versiones derivadas mantengan la misma licencia.

Logs

Registros generados por el sistema que almacenan información sobre eventos relevantes, como modificaciones de datos, accesos o errores, permitiendo auditoría, trazabilidad y diagnóstico de problemas.

Materia de contraturno

Materia que se dicta fuera del horario escolar regular. En este caso, la asistencia es tomada directamente por el docente responsable de la materia, quien registra la presencia de los estudiantes únicamente para su espacio curricular específico. Por ejemplo, un docente de informática de contraturno registra la asistencia sólo para esa materia. Estos registros son independientes de la asistencia general del turno.

Materia de turno

Materia que se dicta dentro del horario escolar regular del alumno. La asistencia general del turno es registrada por los preceptores, quienes marcan la presencia o ausencia diaria de los estudiantes. Adicionalmente, los docentes de turno pueden informar inconsistencias específicas, como la ausencia de un alumno que figura inicialmente como presente.

Merge

Operación en sistemas de control de versiones que consiste en integrar los cambios realizados en una rama dentro de otra, consolidando el trabajo desarrollado en paralelo.

Middleware

Componente intermedio que intercepta solicitudes entre el cliente y el servidor, permitiendo ejecutar lógica adicional como validaciones, autenticación, control de acceso o registro de eventos antes de llegar al destino final.

Mock / Mockeo

Técnica utilizada en testing que consiste en reemplazar dependencias reales de un sistema (por ejemplo, bases de datos, APIs externas o servicios) por versiones simuladas o controladas llamadas mocks. Esto permite probar de manera aislada una unidad de código, controlar los escenarios de prueba y evitar efectos secundarios, mejorando la velocidad y la confiabilidad de las pruebas.

Neon

Plataforma de base de datos PostgreSQL en la nube que ofrece una arquitectura serverless, permitiendo escalar automáticamente los recursos según la demanda.

ORM (Object Relational Mapping)

Técnica que permite mapear entidades del modelo de objetos de una aplicación a tablas de una base de datos relacional, facilitando la manipulación de datos sin necesidad de escribir consultas SQL de forma directa.

Pipeline

Secuencia automatizada de pasos que se ejecutan en orden ante un determinado evento. En el contexto de desarrollo de software, un pipeline de integración y entrega continua agrupa tareas como la construcción, las pruebas y el despliegue de una aplicación, permitiendo que estos procesos ocurran de forma automática y repetible ante cada cambio en el código.

Plugins

Extensiones o módulos adicionales que se integran a una herramienta o framework para agregar funcionalidades específicas sin modificar su núcleo.

Preceptor

Rol institucional responsable del seguimiento administrativo y académico de uno o más cursos asignados. El preceptor registra la asistencia diaria de los estudiantes pertenecientes a dichos cursos durante el turno escolar, pudiendo cargar ausencias, justificaciones y observaciones asociadas. Su accionar se encuentra limitado a los cursos a su cargo, quedando todas las operaciones registradas en el sistema para asegurar trazabilidad y control de modificaciones.

Progressive Web App (PWA)

Aplicación web que emplea tecnologías modernas del navegador para ofrecer una experiencia similar a la de una aplicación nativa, permitiendo su instalación en el dispositivo, funcionamiento sin conexión y carga optimizada, sin necesidad de distribuirse a través de tiendas de aplicaciones.

Proxmox

Plataforma de virtualización de código abierto que permite crear y administrar máquinas virtuales y contenedores sobre un servidor físico.

Proxy

Servidor intermediario que actúa entre un cliente y un servidor destino, gestionando, filtrando o redirigiendo las solicitudes.

Proxy inverso

Servidor intermediario que recibe las solicitudes de los clientes y las redirige a uno o varios servidores internos, ocultando su estructura y pudiendo realizar funciones como balanceo de carga o terminación SSL.

Pull Request

Mecanismo utilizado en sistemas de control de versiones para proponer la incorporación de cambios realizados en una rama al repositorio principal, permitiendo revisión de código y validación previa.

Rama (branch)

Línea de desarrollo independiente dentro de un repositorio de control de versiones, utilizada para trabajar en nuevas funcionalidades, correcciones o experimentos sin afectar la rama principal.

Refresh Token

Token de mayor duración utilizado para obtener nuevos access tokens sin requerir que el usuario vuelva a autenticarse. Se emplea como mecanismo para extender sesiones de forma segura, manteniendo un equilibrio entre usabilidad y control de acceso.

Responsive

Característica de una interfaz que le permite adaptarse automáticamente a distintos tamaños y resoluciones de pantalla, garantizando una correcta experiencia de uso en computadoras, tablets y dispositivos móviles.

Server Action

Mecanismo provisto por Next.js que permite definir funciones que se ejecutan exclusivamente del lado del servidor y que pueden ser invocadas desde componentes o formularios del frontend. Las *Server Actions* encapsulan lógica de negocio, validaciones y operaciones sobre la base de datos sin exponer endpoints explícitos, reduciendo la superficie de ataque y simplificando la comunicación cliente-servidor. Su ejecución ocurre siempre en el entorno del servidor, independientemente de dónde se origine la interacción del usuario.

Servidor

Sistema informático, físico o virtual, que proporciona servicios, recursos o funcionalidades a otros dispositivos (clientes) dentro de una red.

SSH (Secure Shell)

Protocolo de red que permite acceder y administrar de forma remota un servidor o dispositivo mediante una conexión cifrada.

Stateless (sin estado)

Modelo de arquitectura en el cual el servidor no almacena información de sesión entre solicitudes. Cada request contiene toda la información necesaria para ser procesada, lo que facilita la escalabilidad y la distribución del sistema, al no depender de almacenamiento compartido de estado.

Superpreceptor

Rol institucional con permisos ampliados dentro del sistema de gestión escolar. El superpreceptor puede registrar y modificar asistencias correspondientes a cursos y turnos que no le fueron asignados originalmente a otros preceptores, actuando como respaldo operativo. Todas las acciones realizadas por este rol quedan registradas mediante logs de auditoría.

Taller

Espacio curricular cuatrimestral propio del Colegio Nacional Illia. A diferencia de las materias, los talleres no poseen calificación numérica ni conceptual. Su resultado académico se registra en el analítico únicamente con la condición de aprobado o desaprobado, y no forman parte del esquema tradicional de notas del boletín. Su objetivo es complementar la formación del alumno mediante propuestas prácticas o interdisciplinarias definidas por la institución.

Trade-off

Decisión de diseño en la que se prioriza un aspecto del sistema a costa de otro, aceptando compromisos entre factores como complejidad, rendimiento, flexibilidad o tiempo de desarrollo.

Variables de entorno

Valores de configuración externos al código de una aplicación que condicionan su comportamiento en tiempo de ejecución. Se utilizan para definir parámetros que pueden variar según el entorno en el que se ejecuta la aplicación, como credenciales de acceso a la base de datos, claves de autenticación o URLs de servicios externos, evitando que esta información sensible quede expuesta dentro del código fuente.

Virtualización

Tecnología que permite crear versiones virtuales de recursos físicos, como servidores, sistemas operativos, almacenamiento o redes, para optimizar el uso del hardware y ejecutar múltiples entornos aislados en una misma infraestructura.

VM (Virtual Machine)

Entorno virtualizado que emula un sistema operativo completo sobre hardware físico mediante un hipervisor, permitiendo ejecutar múltiples sistemas operativos de forma aislada en una misma máquina.

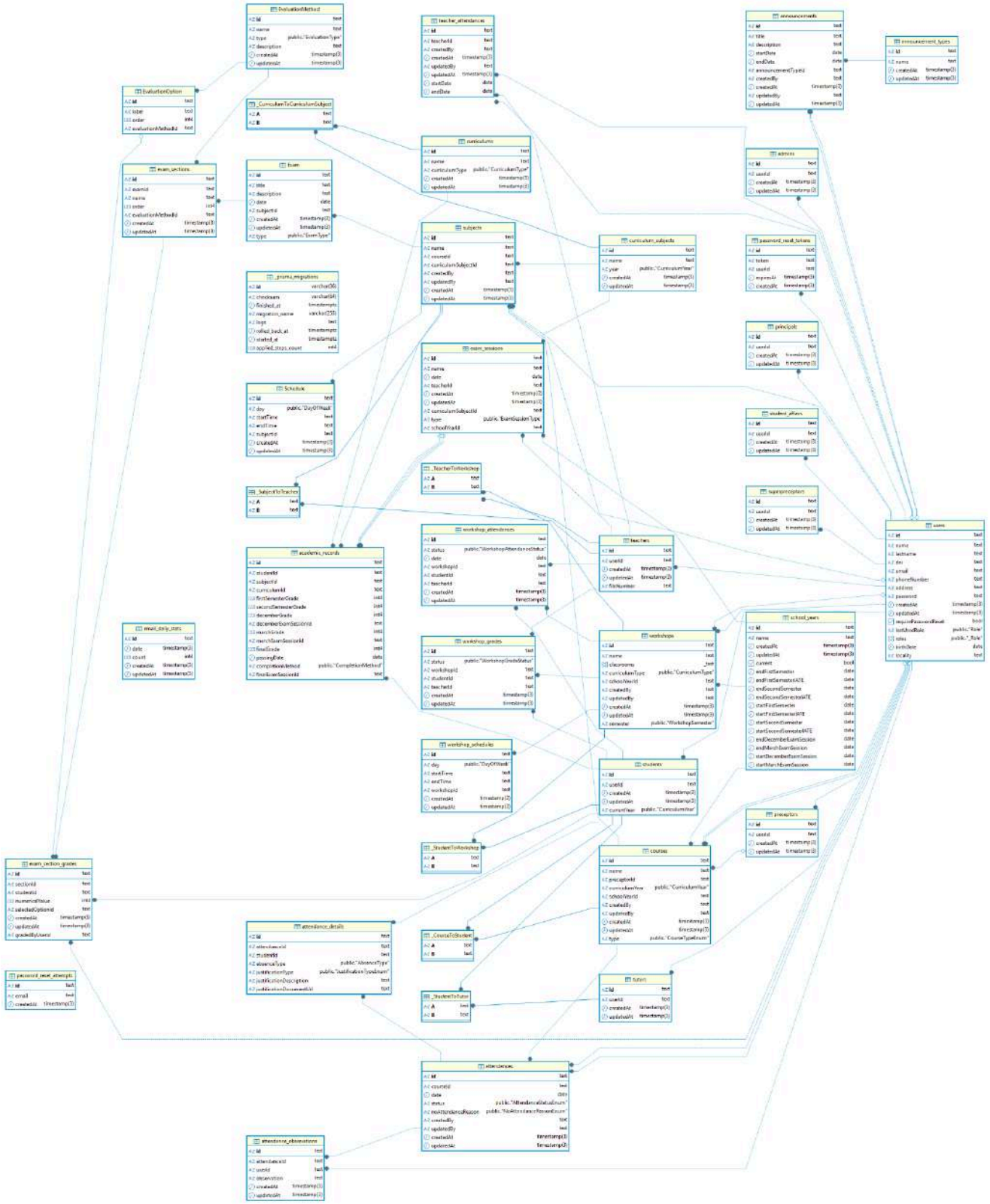
Webapp

Aplicación web accesible desde un navegador, que ofrece funcionalidades similares a una aplicación de escritorio o móvil sin requerir instalación local.

Anexos

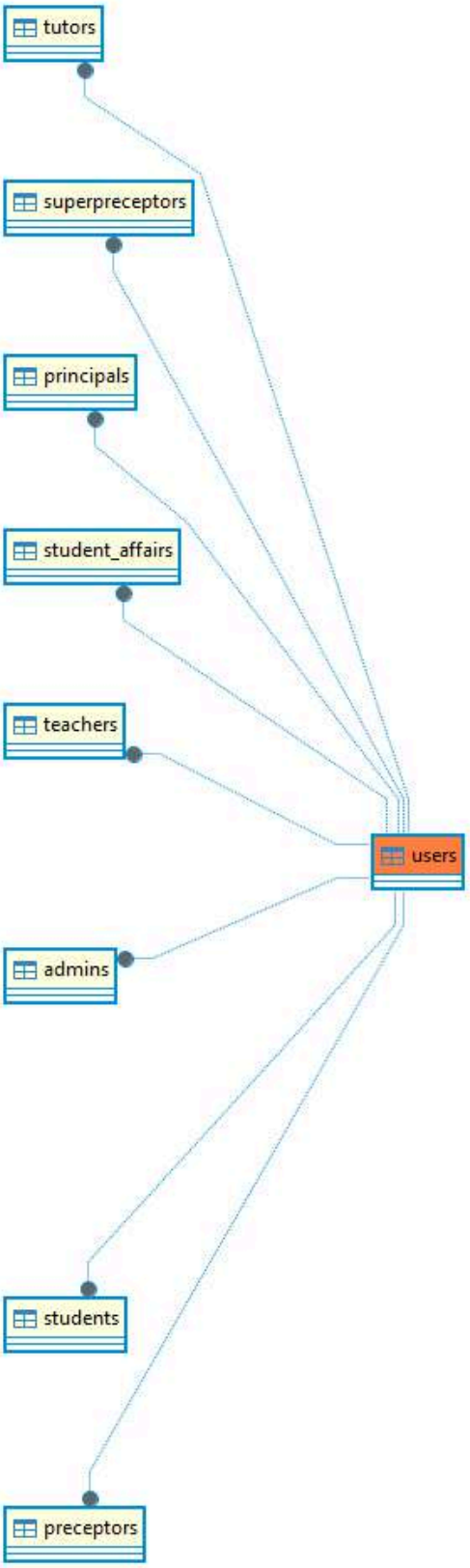
Anexo I: Diagramas Entidad-Relación

Diagrama Entidad-Relación completo



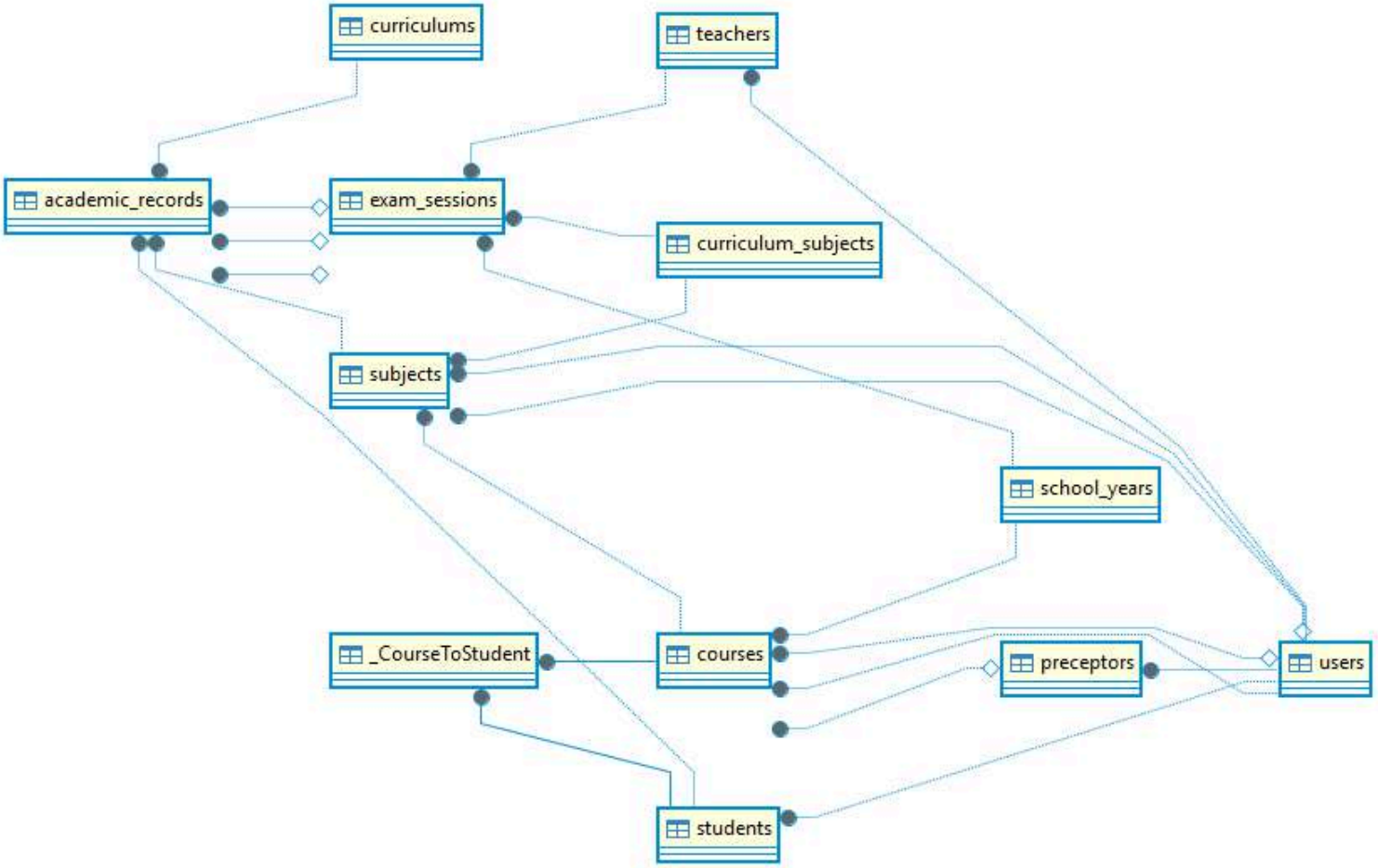
Diagramas Entidad-Relación por módulo

Módulo usuarios:

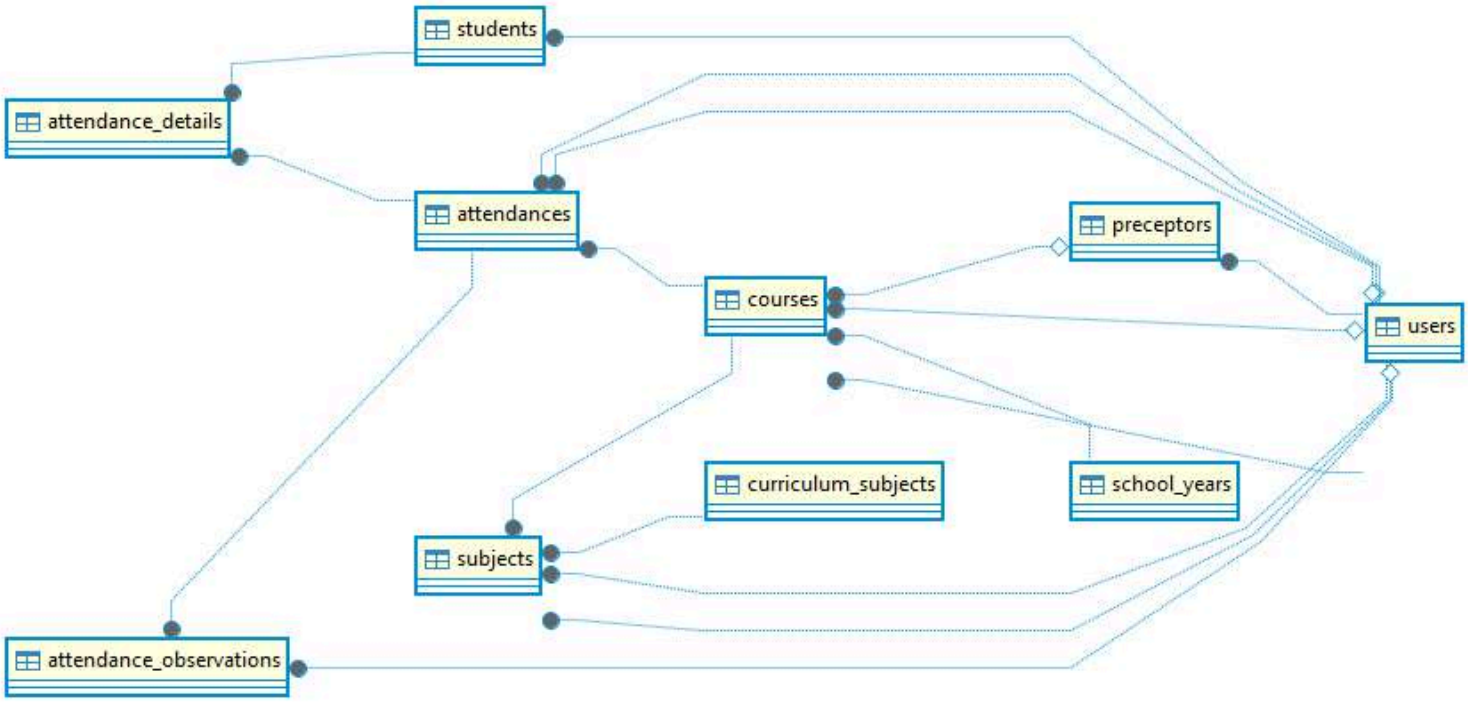


Módulo Materias - Cursadas

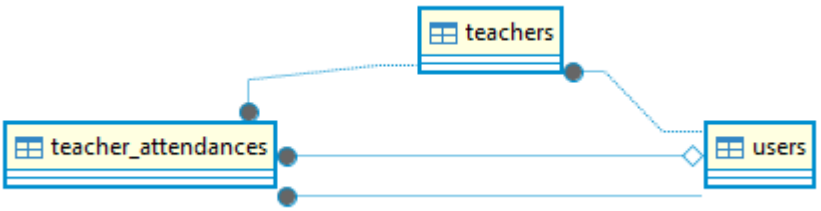
Nota:
Materia ~ *Curriculum Subject*
Cursada ~ *Subject*



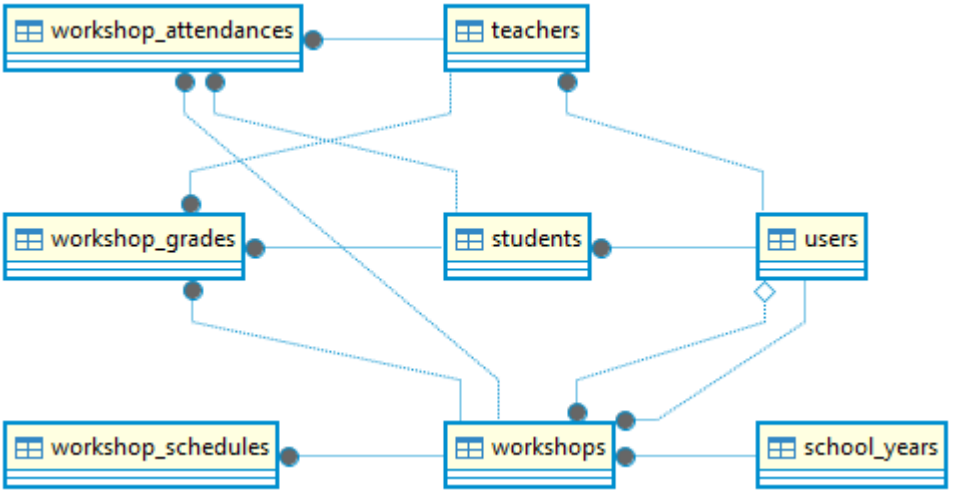
Módulo asistencia estudiantil



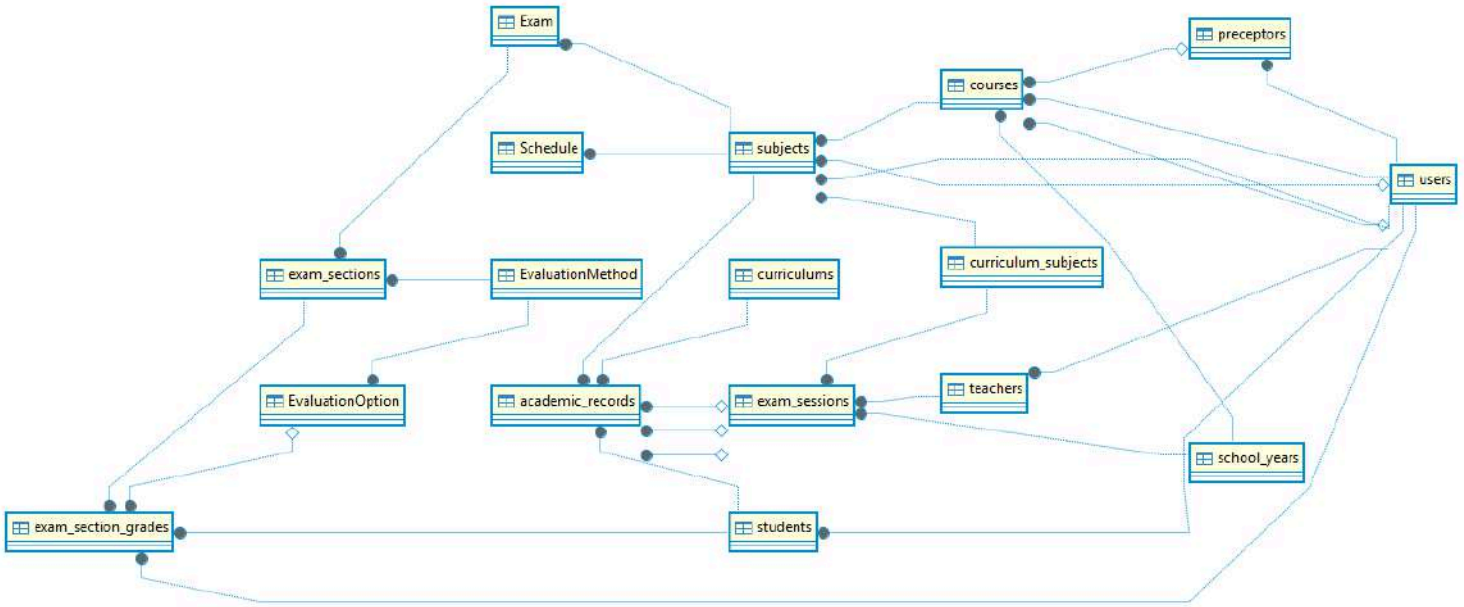
Módulo asistencia docente



Módulo talleres



Módulo gestión académica completo



Anexo II: Acuerdo de entrega, confidencialidad y desvinculación del sistema IlliApp

ACUERDO DE ENTREGA, CONFIDENCIALIDAD Y DESVINCULACIÓN DEL SISTEMA ILLIAPP

En la ciudad de Mar del Plata, a los ___ días del mes de _____ de 2026, se celebra el presente acuerdo entre:

Por una parte, el equipo desarrollador del sistema informático denominado IlliApp, integrado por:
• Cardelli, Ramiro – DNI 44.115.280
• de Lellis, Lucas – DNI 43.986.166
• Frasca Ponce, Josefina – DNI 44.145.157
• González, Franco – DNI 42.089.468

(en adelante, el Equipo Desarrollador)

y por la otra, Sebastian Salgueiro, en representación del Colegio Nacional Dr. Arturo Umberto Illia, dependiente de la Universidad Nacional de Mar del Plata (en adelante, la Institución).

Las partes acuerdan lo siguiente:

1. Entrega del sistema

El Equipo Desarrollador hace entrega a la Institución del sistema informático denominado IlliApp, desarrollado en el marco del Trabajo Final de Carrera de la carrera de Ingeniería en Informática.
El presente trabajo se entrega de manera totalmente gratuita, deslindando todo tipo de responsabilidad del Equipo Desarrollador a partir de su entrega al Colegio Nacional Dr. Arturo Umberto Illia.
La entrega comprende el software desarrollado, su código fuente y la documentación asociada necesaria para su utilización, administración y eventual evolución futura.
El sistema se entrega en el estado en que se encuentra al momento de la firma del presente acuerdo, correspondiente a la versión final desarrollada en el marco del proyecto académico.

2. Finalización de la participación del equipo desarrollador

A partir de la fecha de firma del presente acuerdo, el Equipo Desarrollador da por finalizada su participación en el desarrollo, implementación y soporte del sistema. En consecuencia, los integrantes del equipo no mantendrán acceso administrativo, técnico ni operativo a servidores, bases de datos, repositorios o cualquier infraestructura asociada al sistema. Si bien se usaron buenas prácticas de desarrollo de software respecto de la seguridad, el equipo de desarrollo no se hace

responsable por la gestión posterior de los datos almacenados en el sistema. Es responsabilidad de la Institución o de los profesionales que esta designe bloquear el acceso al servidor al Equipo Desarrollador.

3. Operación, mantenimiento e integridad de los datos

La administración, operación, mantenimiento, actualización y cualquier modificación futura del sistema quedarán exclusivamente bajo responsabilidad de la Institución o de los profesionales que ésta designe.
Una vez realizada la entrega del sistema, el Equipo Desarrollador no asume responsabilidad alguna respecto de la operación del sistema ni de la integridad, disponibilidad o consistencia de los datos que se registren, almacenen o procesen en el mismo.
Asimismo, el Equipo Desarrollador no será responsable por eventuales pérdidas de información, corrupción de datos, configuraciones incorrectas, intervenciones de terceros o cualquier otra situación que pudiera afectar el funcionamiento del sistema o la integridad de la información gestionada por la Institución con posterioridad a la entrega.

4. Confidencialidad

El Equipo Desarrollador se compromete a mantener la confidencialidad de toda la información técnica, institucional, académica o administrativa a la que hubiera tenido acceso durante el desarrollo del proyecto.
Dicha información no podrá ser divulgada, publicada ni cedida a terceros sin autorización expresa de la Institución.

5. Acceso a datos institucionales

Durante el desarrollo del proyecto, el Equipo Desarrollador no tuvo acceso a bases de datos reales ni a información sensible de estudiantes, docentes, tutores u otros miembros de la institución.
Las pruebas y validaciones del sistema se realizaron utilizando datos ficticios.

6. Licenciamiento del software

El sistema se entrega bajo Licencia MIT. En virtud de dicha licencia, la Institución podrá utilizar, copiar, modificar, fusionar, publicar, distribuir y adaptar el software conforme a sus necesidades institucionales.

7. Uso futuro del sistema

La Institución podrá utilizar, mantener, modificar o evolucionar el sistema conforme a sus necesidades institucionales, sin requerir la intervención del Equipo Desarrollador.

8. Conformidad

Mediante la firma del presente documento, las partes dejan constancia de:
• la entrega del sistema,
• la recepción del mismo por parte de la Institución, y
• la finalización del vínculo operativo del Equipo Desarrollador con el sistema.

En la ciudad de Mar del Plata, a los 7 días del mes de mayo de 2026, se firman dos ejemplares de un mismo tenor y a un solo efecto.

Por el Equipo Desarrollador

Firma: _____
Nombre: Ramiro Cardelli

Firma: _____
Nombre: Lucas de Lellis

Firma: _____
Nombre: Josefina Frasca Ponce

Firma: _____
Nombre: Franco González

Por el Colegio Nacional Dr. Arturo U. Illia

Firma: _____
Nombre: Sebastian Salgueiro
Cargo: _____

Bibliografía

1. Getting Started - Dashboard App [en línea]. Next.js. Consultado el 10 de enero de 2025 en https://nextjs.org/learn/dashboard-app/getting-started
2. Next.js Docs: App - Getting Started / Installation [en línea]. Next.js. Consultado el 17 de enero de 2025 en https://nextjs.org/docs/app/getting-started/installation
3. Next.js Docs: Project Structure [en línea]. Next.js. Consultado el 13 de febrero de 2025 en https://nextjs.org/docs/app/getting-started/project-structure#organizing-your-project
4. API Reference: Parallel Routes [en línea]. Next.js. Consultado el 12 de junio de 2025 en https://nextjs.org/docs/app/api-reference/file-conventions/parallel-routes
5. API Reference: Unauthorized [en línea]. Next.js. Consultado el 12 de junio de 2025 en https://nextjs.org/docs/app/api-reference/file-conventions/unauthorized
6. Guías: Prisma ORM con Next.js [en línea]. Prisma. Consultado el 27 de enero de 2025 en https://www.prisma.io/docs/guides/prisma-orm-with-nextjs
7. Prisma Schema: Relaciones Many-to-Many [en línea]. Prisma. Consultado el 13 de febrero de 2025 en https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/many-to-many-relations#implicit-many-to-many-relations
8. Prisma Client: Queries y Paginación [en línea]. Prisma. Consultado el 25 de junio de 2025 en https://www.prisma.io/docs/orm/prisma-client/queries/pagination
9. Documentación oficial [en línea]. NextAuth.js. Consultado el 27 de mayo de 2025 en https://next-auth.js.org/
10. Documentación oficial [en línea]. React Hot Toast. Consultado el 13 de junio de 2025 en https://react-hot-toast.com/
11. Spec v1.0.0 [en línea]. Conventional Commits. Consultado el 17 de enero de 2025 en https://www.conventionalcommits.org/en/v1.0.0/
12. CUID2 GitHub Repository [en línea]. ParallelDrive. Consultado el 24 de enero de 2025 en https://github.com/paralleldrive/cuid2
13. Building the Future of Scalable Applications in Next.js: A Modular Approach to Architecture [en línea]. Priyank Lad. Consultado el 14 de mayo de 2025 en

https://medium.com/@priyanklad52/building-the-future-of-scalable-applications-in-nextjs-a-modular-approach-to-architecture-89d811231f81
14. TypeScript Style Guide - Functions [en línea]. M. Kosir. Consultado el 28 de mayo de 2025 en https://mkosir.github.io/typescript-style-guide/#functions
15. Next.js + Prisma + Postgres Guide [en línea]. Vercel. Consultado el 13 de mayo de 2025 en https://vercel.com/kb/guide/nextjs-prisma-postgres
16. Introducing React Best Practices [en línea]. Vercel. Consultado el 30 de enero de 2026 en https://vercel.com/blog/introducing-react-best-practices
17. Discussion #50509 [en línea]. Next.js GitHub Discussions. Consultado el 12 de agosto de 2025 en https://github.com/vercel/next.js/discussions/50509