UNIVERSIDAD NACIONAL de MAR DEL PLATA

*UNMDP*
*Facultad de Ingeniería*
*Departamento de Informática*

# Sistema de búsqueda y notificación de servicios o productos con alta demanda

**Autores:**

Fioriti, Joaquin.
joaquinfioriti9@gmail.com
Matarazzo, Tomás.
tmatarazzo03@gmail.com

**Director:**

MBA Genin, Fernando.

**Fecha:**

15 de Junio del 2024.

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

**Mar del Plata, Argentina.**

UNIVERSIDAD NACIONAL de MAR DEL PLATA

*UNMDP*
*Facultad de Ingeniería*
*Departamento de Informática*

# Sistema de búsqueda y notificación de servicios o productos con alta demanda

**Autores:**

Fioriti, Joaquin.
joaquinfioriti9@gmail.com
Matarazzo, Tomás.
tmatarazzo03@gmail.com

**Director:**

MBA Genin, Fernando.

**Fecha:**

15 de Junio del 2024.

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

**Mar del Plata, Argentina.**

# Agradecimientos

Si bien el proceso de una carrera universitaria depende mucho del mérito propio de cada uno, consideramos igual de importante las relaciones y el esfuerzo de las personas que acompañaron este proceso.

En primer lugar queremos agradecer a nuestras familias, quienes nos dieron la posibilidad de estudiar mediante apoyo emocional y económico.

A todos los compañeros y futuros colegas que transitaron el camino con nosotros, creemos profundamente que el compañerismo es la clave para el progreso en conjunto de los estudiantes.

A todos los profesores que nos brindaron su conocimiento para transformarnos en los profesionales que somos hoy en día. En especial a Fernando Genin, director de este trabajo quien fue crucial para el desarrollo del mismo.

Es un honor para nosotros haber tenido la posibilidad de finalizar la carrera con un proyecto propio y emocionante, con un impacto social que tuvimos la oportunidad de presenciar.

Fue un placer comenzar, transitar y finalizar la carrera juntos.

# Resumen

El presente Proyecto Final fue desarrollado por Joaquín Fioriti y Tomás Matarazzo, estudiantes de la carrera de Ingeniería en Informática en la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata.

Esta iniciativa implica la creación de una aplicación web que adopte una arquitectura basada en microservicios. El objetivo primordial radica en la creación de un sistema que facilite la obtención de servicios y/o productos de alta demanda y que brinde notificaciones a los usuarios. Para lograr esto, se implementó una suite de componentes reutilizables que permiten escalar en funcionalidad, mantenibilidad y usabilidad.

El documento detalla el análisis, diseño e implementación llevados a cabo por los estudiantes. Asimismo, se exponen las estimaciones de tiempo planificadas y se comparan con la duración real del proyecto. Se realizó el análisis de la búsqueda de pasajes de la compañía Trenes Argentinos como caso de uso del sistema, lo cual implicó la toma de decisiones a partir de consideraciones comerciales durante el desarrollo del proyecto.

Actualmente, la plataforma está implementada y se utiliza a diario para la búsqueda y compra de pasajes de Trenes Argentinos. Gestiona una gran demanda de pasajes y satisface cientos de reservas mensuales.

# Capítulo 1: Introducción

En la actualidad, existe un contexto donde la demanda de servicios y productos en línea muestra una tendencia constante al alza. Ocurre muy a menudo una situación en la que se cumplen las siguientes condiciones: alta demanda y muy baja disponibilidad. Estos factores pueden generar ocasionalmente una oportunidad para facilitar la adquisición de un producto o servicio al cliente.

A partir de esta situación, se desarrolló un sistema que permite aprovechar estas circunstancias para facilitar el proceso de adquisición de servicios/productos. El software se basa en una arquitectura de microservicios reutilizables que incluye una plataforma para que los usuarios realicen sus pedidos, sean notificados sobre la adquisición del producto y cuenten con un proceso de compra integrado.

Como caso de estudio principal, se seleccionó la venta de pasajes de Trenes Argentinos. Estos trenes realizan numerosos viajes a diario a lo largo de todo el país, se conectan diferentes ciudades y regiones. Algunos de los servicios de trenes a larga distancia más populares en Argentina incluyen el Tren Sarmiento, el Tren Mitre, el Tren Roca y el Tren San Martín, entre otros.

El servicio del Tren Roca cubre la ruta entre Mar del Plata y Buenos Aires, con paradas intermedias. Debido a la atracción turística de ambas ciudades, este servicio es uno de los más demandados y con muy poca disponibilidad. Esto lo convierte en un caso ideal para la implementación del sistema, ya que cuenta con viajes diarios y un volumen de aproximadamente 2000 personas a diario.

Una característica particular de este proyecto es que los demandantes son los propios estudiantes, quienes actúan tanto como emprendedores y desarrolladores del sistema. Cuando se hace referencia a los estudiantes en su rol de emprendedores y demandantes, se utilizarán estos términos. Mientras que en su rol de desarrolladores, se utilizará la denominación "desarrolladores" y "estudiantes".

# Capítulo 2: Dominio del problema

En el contexto de la era digital y la creciente demanda de servicios y productos, la simplificación y optimización a la hora de obtener los mismos se convirtieron en objetivos clave para mejorar la experiencia del usuario y aumentar la eficiencia operativa. Este desafío se extiende a una variedad de sectores, desde la industria del transporte hasta la atención al cliente y la gestión de compras en línea.

La adquisición de servicios o productos puede ser un proceso complejo y a menudo frustrante para los usuarios. Los sistemas existentes pueden presentar una variedad de problemas, desde largos tiempos de espera hasta dificultades para completar transacciones. Estos problemas pueden tener un impacto significativo en la experiencia del usuario.

Algunos productos o servicios generan tal demanda que su obtención a través de canales en línea se vuelve prácticamente imposible. Ejemplos claros de esto son los pasajes de Trenes Argentinos , los turnos para trámites de ciudadanía italiana o entradas a conciertos. En estos casos, la alta demanda genera competencia por los recursos limitados disponibles, y resulta en dificultades significativas para los usuarios que intentan asegurar estos productos o servicios en línea.

Este proyecto pretende brindar una solución al usuario mediante el desarrollo de un sistema innovador basado en un conjunto de microservicios reutilizables que integran diversas tecnologías para el sector de compras en línea.

# Capítulo 3: Problemática a resolver

## 3.1 Objetivo general

Desarrollar e implementar un sistema informático distribuido que simplifique la obtención de servicios y productos en línea.

## 3.2 Objetivos específicos

- Implementar una arquitectura de microservicios, que garantice la escalabilidad del sistema y que disponga de capacidad de adaptación a nuevos productos y servicios, basado en la reutilización de componentes.

- Implementar la solución en la empresa Trenes Argentinos:

    - Desarrollar un microservicio de búsqueda de pasajes que se adapte fácilmente al sistema.

    - Realizar iteraciones del producto que satisfagan la demanda del mercado y posicionarse como principal opción en el mismo.

# Capítulo 4: Gestión de Proyecto

## 4.1 Análisis FODA

El análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas) es una herramienta de estudio estratégica que se utiliza para evaluar los factores internos y externos que pueden influir en el éxito de un proyecto. Este análisis se realizó a priori, con el objetivo de identificar las ventajas competitivas y los desafíos potenciales del equipo y del proyecto, en pos de maximizar los beneficios y mitigar los riesgos.

**Fortalezas:**

**Experiencia colaborativa y sinergia en equipo:** El equipo posee una amplia experiencia trabajando juntos, respaldada por la ejecución exitosa de numerosos proyectos en conjunto.

**Dominio tecnológico:** El equipo está familiarizado con diversas tecnologías relevantes para sus funciones.

**Motivación por cumplimiento de plazos:** La motivación de los estudiantes se deriva de la necesidad cumplir con los rigurosos plazos impuestos por ellos mismos en su rol de demandantes , lo que impulsa a mejorar continuamente su desempeño.

**Reducción de tiempo para los usuarios:** El sistema simplifica la obtención de servicios y productos para el usuario. La automatización integral del mismo busca mejorar significativamente la experiencia del usuario. El tiempo de trabajo manual por parte del usuario disminuye.

**Componentes reutilizables:** La suite de componentes reutilizables aporta modularidad y eficiencia al desarrollo del proyecto. Es una fortaleza técnica que permite la fácil adaptación a nuevos productos emergentes que cumplan con una alta demanda y poca oferta.

**Oportunidades:**

**Demandantes propios del proyecto:** Brinda la posibilidad de ganar experiencia directa y participar activamente en la definición y desarrollo del sistema según necesidades y

preferencias de los estudiantes. Esto permite tomar decisiones clave sobre el diseño y la implementación, lo que contribuye significativamente al aprendizaje y desarrollo profesional.

**Unicidad en el mercado:** Pioneros en el ámbito, no se encuentran soluciones a esta problemática en el mercado actualmente. Esto otorga una posición privilegiada para capitalizar el crecimiento continuo dentro del mercado.

**Mercado en crecimiento:** En un contexto de crecimiento del mercado, la demanda de servicios como el que se ofrece está en constante aumento. Esto se debe a que las personas buscan cada vez más soluciones confiables y seguras de gestión de pagos. En este escenario, se abre una mayor oportunidad para satisfacer estas necesidades de los consumidores.

**Expansión a otros servicios:** La tecnología propuesta, consiste en una suite de componentes reutilizables, tiene potencial para ser aplicada en una variedad de servicios de búsqueda adicionales en el futuro. Esta flexibilidad permite explorar nuevas oportunidades de mercado y diversificar las ofertas.

**Colaboración con proveedores de servicios:** Existe la oportunidad de colaborar con proveedores de servicios lo que podría conducir a una demanda más amplia del servicio del equipo.

**Debilidades:**

**Primer proyecto con impacto real:** Los alumnos carecen de experiencia en la implementación de proyectos propios con salida al mercado. Además, requiere que los alumnos realicen la puesta en producción, una práctica nunca antes abordada por ellos.

**Disponibilidad:** Disponibilidad horaria acotada de los integrantes.

**Competencia:** Puede haber competencia futura de otras aplicaciones y sistemas de reserva de servicios y productos que podría afectar la viabilidad del proyecto y motivación de los alumnos.

**Costo de mantenimiento:** A medida que el proyecto crezca, el costo de mantenimiento aumentará.

**Posibles errores técnicos:** La complejidad técnica puede aumentar la probabilidad de errores en la aplicación

**Amenazas:**

**Amenaza para la planificación inicial:** Los tiempos del mercado podrían afectar negativamente a las decisiones iniciales en cuanto al diseño y análisis del proyecto. Puede llegar a comprometer la calidad y eficacia de las soluciones propuestas.

**Cambios regulatorios**: Cambios en las regulaciones gubernamentales o políticas pueden impactar en la operatividad del sistema y requerir ajustes adicionales para cumplir con los nuevos requisitos legales. Estos cambios podrían generar incertidumbre y desmotivación entre los estudiantes participantes.

**Cambios técnicos:** Cambios en el proceso de la adquisición del producto o de la industria podrían afectar los tiempos de operación del proyecto.

**Desaceleración económica:** Una desaceleración económica podría reducir la demanda de servicios y productos, lo que afectaría la viabilidad del proyecto.

## 4.2 Conclusión de Análisis FODA previo al desarrollo

**Aprovechar las fortalezas y oportunidades:** Utilizar el conocimiento del equipo y la experiencia previa para desarrollar e implementar el proyecto de manera eficiente. Aprovechar la motivación por el cumplimiento de plazos, donde se priorice mantener un enfoque ágil y orientado a resultados. Capitalizar la unicidad en el mercado y la expansión a otros servicios en pos de diversificar las ofertas y aumentar la ventaja competitiva.

**Abordar las debilidades:** Aplicar medidas para mejorar la disponibilidad horaria de los integrantes, como la flexibilidad en los horarios de trabajo o la asignación de roles y

responsabilidades claras. Establecer un plan de control de calidad para reducir la probabilidad de errores técnicos y minimizar el costo de mantenimiento.

**Mitigar las amenazas:** Monitorear los tiempos del mercado y realizar ajustes en la planificación según sea necesario para que no sea comprometida la calidad y eficacia del proyecto. Mantenerse actualizado sobre los cambios regulatorios y técnicos que puedan afectar los plazos del mismo.
Al implementar esta estrategia, el equipo está mejor posicionado para aprovechar las oportunidades del mercado, superar los desafíos identificados y alcanzar el éxito con el proyecto.

## 4.3 Conclusión del Análisis FODA posterior al desarrollo

**Fortalezas y oportunidades**

Tal como se anticipó, la experiencia previa trabajando en equipo y el dominio de las tecnologías se convirtieron en una fortaleza clave, permitiendo una implementación eficiente del proyecto. Cada integrante logró maximizar sus habilidades, lo que resultó en una productividad notable.

Otra de las oportunidades previstas y acertadas fue la unicidad del producto en el mercado. El hecho de ser la única solución disponible en ese momento contribuyó notoriamente a la cantidad de clientes, y fue un acierto por parte del equipo que supo aprovechar esta ventaja competitiva.

Un acierto fue la significativa reducción del tiempo para los usuarios al simplificar la obtención de servicios y productos. La automatización integral del sistema mejoró la experiencia del usuario, disminuyendo el tiempo de trabajo manual. En comparación con la complejidad de la interfaz y metodología de compra de los Trenes Argentinos, la solución del equipo se mostró mucho más accesible y eficiente, lo que se tradujo en una mayor satisfacción del cliente y una adopción más rápida del producto en el mercado.

Es importante destacar que se consideró correctamente una amenaza relacionada con los tiempos del mercado, es decir, la posibilidad de que algún evento externo afectará la planificación del proyecto. Los plazos impuestos por el mercado, que en un principio fueron percibidos como una amenaza, resultaron ser una fortaleza oculta. Esta situación llevó al equipo a desarrollar una **capacidad de gestión** que no se había descrito previamente. A lo largo del proyecto, surgió una competencia clave: la habilidad del equipo para adaptarse a los desafíos del mercado. Los estudiantes no sólo reorganizaron tareas y se adaptaron a los cambios, sino que lograron avanzar mientras el producto ya se encontraba en uso, mostrando una habilidad de adaptación ágil y eficaz, algo fundamental para el éxito del proyecto.

Este enfoque permitió al equipo no solo cumplir con las expectativas, sino también fortalecer habilidades esenciales para futuros desafíos.

**Debilidades y amenazas**

Además de identificar correctamente las fortalezas y oportunidades del proyecto, se lograron reconocer con éxito las amenazas y debilidades. Afortunadamente, las debilidades no impactaron la planificación ni generaron costos adicionales. En cuanto a las amenazas, el equipo identificó correctamente los cambios regulatorios como una amenaza, que se materializó luego de la temporada de verano de 2023-2024. Este cambio impactó directamente en el precio de los boletos de tren, alterando la oferta y la demanda de este servicio. Como resultado, la solución propuesta por el equipo se vio afectada debido a la paridad de precios entre el tren y otros medios de transporte, lo que generó una amplia disponibilidad de boletos de tren. Sin embargo, el equipo pudo aprovechar la demanda existente antes del aumento del precio del boleto, lo que permitió capitalizar en el corto plazo y maximizar el impacto de su propuesta antes de que el mercado cambie drásticamente.

## 4.4 Gestión de Riesgos

Se examinaron posibles factores que podrían poner en riesgo su viabilidad o afectar de manera considerable, además de los aspectos considerados en el análisis FODA. Cada riesgo potencial se evaluó en función de:

- Su probabilidad de ocurrencia (Po)
- Su impacto en el proyecto (I)
    - Se utiliza una escala del 1 al 3, donde 3 indica el mayor grado de riesgo.
- El **peso** de cada riesgo se determina al multiplicar su probabilidad de ocurrencia por su impacto.
    - Si este valor es igual o superior a 6, es necesario establecer un plan de contingencia para contrarrestar su impacto.

| ***Riesgo*** | ***Consecuencia*** | ***Po*** | ***I*** | ***Peso*** |
|---|---|---|---|---|
| Cambios en la obtención del producto o servicio | Implementación y actualización de features. Ausencia en el mercado hasta solucionar los cambios | 2 | 3 | **6** |
| Ausencia de los integrantes | Pérdida de capacidad productiva | 3 | 2 | **6** |
| Desconocimiento en el procedimiento de puesta en producción | Posibles demoras en la puesta en producción. | 2 | 3 | **6** |
| Estimaciones erróneas de tiempo y trabajo | Mayor cantidad de tiempo necesario para el cumplimiento de las tareas | 1 | 3 | 3 |
| Cambios en los requerimientos iniciales | Retrasos en el cronograma | 1 | 3 | 3 |

| | | | | |
|---|---|---|---|---|
| Cierre de Trenes Argentinos. | Desmotivación de los estudiantes. Imposibilidad de desarrollar software específico para el servicio de Trenes Argentinos | 2 | 3 | **6** |

**Planes de Contingencia**

**Ausencia de los integrantes:**
Establecer un plan de comunicación claro y reorganizar las tareas que contemple las ausencias de los estudiantes.

**Desconocimiento en el procedimiento de puesta en producción:**
Estimar las tareas de producción con un alto margen de tiempo adicional. Cuando se desconoce las tecnologías y los procedimientos, es incierto el tiempo que se deberá invertir en el aprendizaje.

**Cambios en la obtención del producto o servicio:**
Utilizar buenas prácticas de programación, patrones de diseño y principios SOLID. Su aplicación disminuye notablemente los tiempos de desarrollo ante posibles cambios. Además, minimiza el tiempo de inoperabilidad dentro del mercado.

**Cierre de Trenes Argentinos:**
Desarrollar una arquitectura de microservicios modularizada que permita la migración entre distintos servicios o productos. Esto permitirá disminuir los tiempos de desarrollo y mantenimiento a partir de la reutilización de componentes.

## 4.5 Estimación inicial

En la primera estimación realizada , visualizada en la planificación con Diagramas de Gantt, se proyectó comenzar la gestión de un proyecto basado en uno previamente iniciado por los estudiantes. Este proyecto consistía en una solución básica para la intermediación de pasajes de Trenes Argentinos. Sin embargo, esta aplicación resultó requerir un considerable trabajo manual por parte de los estudiantes. La solución implementada no era escalable y tampoco permitía la adaptación del software para otros tipos de servicios o productos.

Para agosto de 2023, el software presentaba componentes precarios con varios puntos de falla que dejaban el sistema inutilizable. Constaba de un formulario de Google para recopilar la información del usuario, un servicio para buscar pasajes, y dentro de este, la opción de enviar correos electrónicos. Los pagos se realizaban únicamente por transferencias bancarias a los emprendedores. En este contexto, el usuario debía ingresar los datos cada vez que deseaba realizar una búsqueda, sin que se llevará a cabo ninguna validación, lo que generaba numerosas fallas en el sistema de búsqueda. Como consecuencia, los demandantes debían realizar una validación manual de los datos posteriormente para evitar fallos en los procesos de búsqueda siguientes.

En esta etapa, el software se utilizaba exclusivamente entre un grupo de allegados a los demandantes. Esta limitación en el uso del software permitió recopilar información de alta calidad y obtener retroalimentación directa de los usuarios. La retroalimentación proporcionada, junto con la comprensión de las áreas de mejora potencial, contribuyó a llevar a cabo una estimación y planificación inicial más precisa. Este contexto sentó las bases a partir de las cuales se desarrolló el diagrama de Gantt y se realizaron las estimaciones pertinentes para el proyecto.

La estimación se llevó a cabo con la idea de que durante los meses de septiembre, octubre y noviembre, el tiempo dedicado al proyecto se limitaría exclusivamente a la definición del problema y la corrección de errores. Esto se debía a la suposición de que los estudiantes tendrían menos tiempo disponible para dedicar al proyecto debido a sus compromisos académicos en la facultad.

Los alumnos eran conscientes de que la definición de tareas y el mantenimiento serían constantes a lo largo del tiempo debido a que el uso por parte de los usuarios revelaría nuevas tareas y errores que necesitan ser abordados continuamente.

Para diciembre y enero, se planificó una etapa dedicada al análisis y diseño del proyecto. Se anticipaba que durante este período se contaría con la información necesaria para iniciar el análisis del proyecto y reflejar en el nuevo diseño del sistema.

Durante los meses restantes, se planeaba arrancar con la implementación. Se establecieron plazos a partir de la experiencia previa del equipo en desarrollo de software y el tiempo disponible de los estudiantes. Se reconoció que los meses de verano suelen presentar desafíos adicionales, ya que puede interrumpirse con vacaciones o compromisos personales.

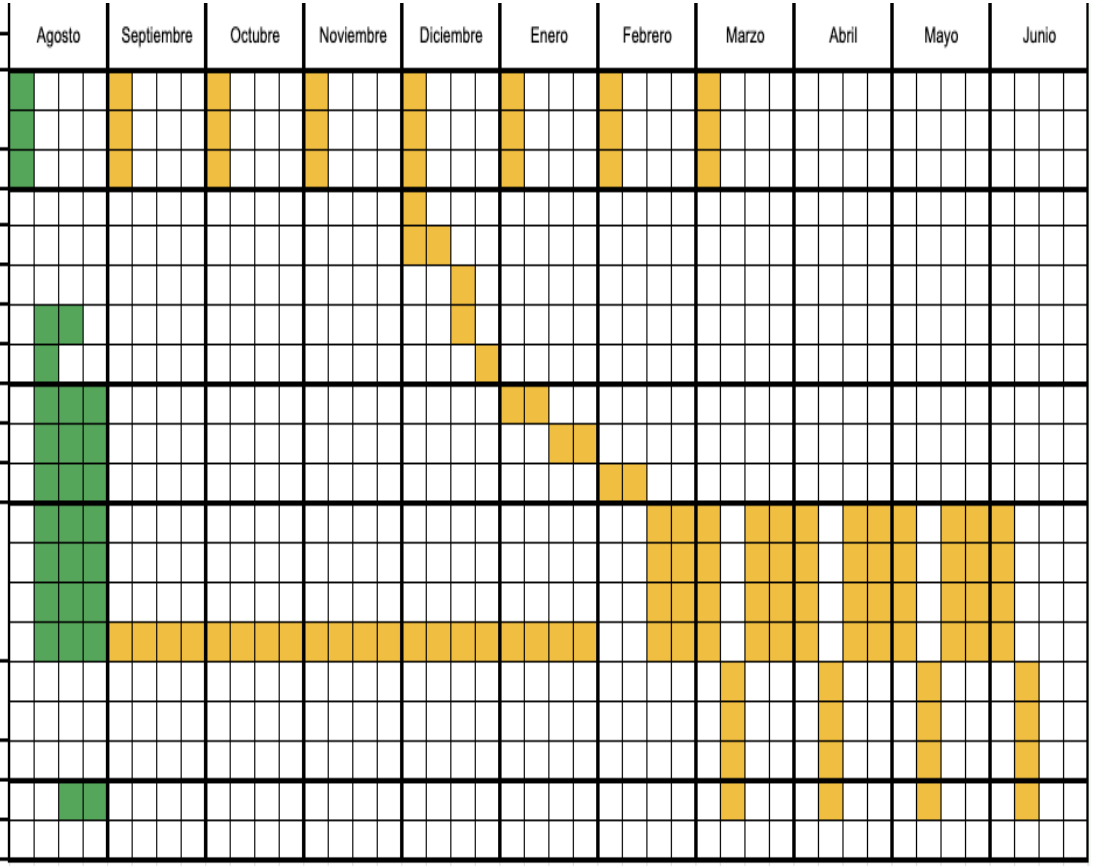
Por último, la puesta en producción no se realizaría una única vez al final del desarrollo. Al depender de los tiempos del mercado, fue necesario considerar la actividad al finalizar cada mes para poder actualizar las distintas versiones con mejoras dentro de las funcionalidades y arreglo de errores. Además, se consideró que la poca experiencia que poseen los estudiantes en esta etapa podrían afectar los plazos del proyecto.

### 4.5.1 Diagrama de Gantt

A continuación, se presenta el Diagrama de Gantt elaborado al inicio del proyecto. Este diagrama no estuvo condicionado por plazos específicos del mercado; el principal objetivo fue ir satisfaciendo la demanda de los clientes mes a mes hasta lograr un producto final. Además esta planificación enfocó el mayor tiempo de desarrollo en verano, cuando los estudiantes tendrían una alta disponibilidad horaria para no interferir en los exámenes de las materias. Es decir, fue una planificación ideal con el objetivo de amoldarse a los tiempos de los alumnos.

Las tareas completadas al comienzo del proyecto se muestran en color verde, mientras que las tareas pendientes se representan en color amarillo.

| Tareas | Subtareas | Cantidad de horas |
|---|---|---|
| Identificacion del problema | Definición de los objetivos y requisitos del sistema. | 30 |
| | Identificación de problemas y desafíos actuales en la gestión de solicitudes. | |
| | Recopilación de datos relevantes sobre los procesos manuales existentes. | 40 |
| Analisis | Creacion de SRS | 10 |
| | Identificación y modelado de los flujos | 30 |
| | Definición de los casos de uso | 10 |
| | Diseño de la arquitectura general del sistema | 10 |
| | Diseño de la base de datos y definición de las relaciones entre las entidades. | 10 |
| Disenio | Diseño detallado de la interfaz de usuario (UI/UX) | 40 |
| | Especificación de la arquitectura de microservicios | 40 |
| | Diseño de la lógica de negocio de cada microservicio | 40 |
| Implementacion | Desarrollo de la aplicación web | 400 |
| | Desarrollo de los microservicios | |
| | Integracion de los microservicios | |
| | Mantenimiento y solucion de bugs | |
| | Pruebas unitarias y de integración inter-microservicio | 80 |
| | Pruebas unitarias y de integración intra-microservicio | |
| | Implementacion de la seguridad (intra e inter) | 80 |
| Despliegue | Despliegue de la aplicación y los microservicios en entornos de producción. | 40 |
| | Monitoreo y gestión de errores en producción. | 40 |
| | **Horas totales** | **900** |



## 4.6 Metodologías de Trabajo

En un principio, previo al inicio del proyecto como Trabajo Final, este comenzó con la reunión de los demandantes sobre la potencial oportunidad en el mercado de crear un software para facilitar la obtención de pasajes de Trenes Argentinos. Se inició sin un plan preestablecido, a partir de la identificación de una necesidad y la ejecución de una solución. Las necesidades se respondían mediante soluciones identificadas de manera ad hoc, lo que resultó en un proceso caracterizado por la improvisación y la aplicación de parches. Finalmente, al comenzar a desarrollarse como Trabajo Final y el deseo de lograr una escalabilidad, se plantearon distintas metodologías de trabajo.

### 4.6.1 MVP, Cascada, Kanban:

Una estrategia de producto mínimo viable (MVP) implica la creación de una versión simplificada de un producto o servicio con las características esenciales para satisfacer las necesidades del cliente y validar la viabilidad de la idea.

En los inicios del proyecto, se partió de un MVP que cumplía con la obtención del pasaje. Este producto brindó información de valor tanto para la gestión del proyecto como para la viabilidad del mismo. Sin embargo, la solución encontrada presentaba muchos problemas y procesos manuales tediosos.

A partir de la información recopilada gracias al MVP, se propuso una visión a largo plazo del proyecto que prioriza el análisis y diseño inicial para establecer una base sólida. Se reconoció la importancia de estos procesos para la expansión del servicio, tanto dentro de Trenes Argentinos como hacia nuevos productos y servicios.

El desarrollo de software en cascada implica un enfoque secuencial y lineal, donde las etapas del ciclo de vida del proyecto se realizan de manera secuencial. La desventaja de este modelo, es la pérdida de retroalimentación, por lo que solo se utilizó en las etapas tempranas del proyecto. En contraste, las metodologías ágiles permiten generar entregas pequeñas en períodos cortos de tiempo, donde se fomenta la retroalimentación continua.

Se empleó la metodología ágil Kanban para la gestión visual de las tareas, el seguimiento del progreso del proyecto y su desarrollo. Esto permitió una implementación iterativa, con lanzamientos que retroalimentaron la ejecución del proyecto ante nuevas necesidades o decisiones contradictorias. Kanban ofrece un sistema altamente visual que organiza las tareas en un tablero, donde cada tarea puede pasar por diferentes etapas, desde “pendiente” hasta “completada”. Esta estructura permite que todo el equipo tenga una visión clara del estado de cada tarea y del proyecto en su conjunto. La retroalimentación inmediata permitió al equipo priorizar y ajustar las tareas en base a las necesidades reales de los usuarios.

Se utilizó Azure DevOps como herramienta para la gestión del proyecto, ya que proporciona un entorno centralizado para supervisar y avanzar en el mismo. Se emplearon tickets para describir las tareas y su estado, así como para relacionarlas según corresponda, además de permitir su asignación a los estudiantes y definir prioridades. Esta plataforma también facilita la vinculación de los repositorios de trabajo con las tareas, lo que resulta en una visualización sencilla del progreso del proyecto.

# Capítulo 5: Análisis

## 5.1 Dominio

Dentro del análisis de un proyecto, es crucial profundizar en la investigación y comprensión del dominio, ya que esto repercute en mejores decisiones futuras. Se debe lograr una definición en detalle de las entidades que conforman el problema, sus relaciones y los procesos existentes.

La dificultad de obtener un producto o servicio altamente demandado de manera online es común en distintos rubros, por lo que el análisis general debe ser lo suficientemente abstracto para facilitar posteriormente el análisis de cada caso en particular.

Cada producto o servicio tendrá sus variantes para la obtención de sí mismo, se ve en detalle el caso particular de Trenes Argentinos. Sin embargo, los requerimientos que no estén directamente relacionados al proceso de obtención son comunes entre los diferentes casos

**Trenes Argentinos**

Trenes Argentinos es la empresa ferroviaria estatal de Argentina. Cuenta con destinos de corta y larga distancia. Como se mencionó antes, este trabajo se enfoca en el estudio de la venta de pasajes de larga distancia. Se unen 391 estaciones a través de 1,893 viajes diarios. El servicio tiene capacidad para transportar 1.3 millones de pasajeros al día. Se hace énfasis en el estudio del viaje entre Mar del Plata y Buenos Aires, y viceversa, el cual tiene una capacidad de 550 personas. A diario se realizan 2 viajes tanto de ida como de vuelta, lo que suma un volumen de 2,200 pasajeros, a veces con servicios adicionales los fines de semana. Es crucial señalar que ambos destinos figuran entre las principales ciudades y atractivos turísticos del país. Durante el verano, el servicio suele incrementar la cantidad de viajes diarios debido al aumento en la demanda.

Este servicio cumple con las condiciones de escasa disponibilidad y alta demanda. Esta situación se debe a la posibilidad de obtener pasajes de tren a importes más bajos en

comparación con otras opciones como viajar en auto o en colectivo. La brecha de precios entre los distintos medios de transporte y el tren es significativa.

La dificultad de conseguir viajes en tren originó una especie de comunidad y "*fomo*" por los trenes argentinos, ya que la incapacidad para obtener pasajes genera un atractivo hacia el servicio que lo hace más demandado. Con frecuencia, se forman largas filas o se realizan búsquedas constantes en internet para conseguir el servicio deseado.

Se pueden adquirir pasajes de forma presencial o virtual. Trenes Argentinos cuenta con distintos locales en sus estaciones a lo largo del país, donde se ofrece la opción de adquirirlos. Sin embargo, la disponibilidad de pasajes en ventanilla suele ser escasa, ya que la mayoría se venden de manera online. De manera online, está disponible a través de la página oficial de Trenes Argentinos, para la cual se requiere la creación de un usuario. Una vez creado, se puede revisar la disponibilidad de pasajes en la página.

**Proceso de adquisición de boletos en Trenes Argentinos**

El proceso de venta funciona de la siguiente manera:

Trenes Argentinos libera los pasajes para cada mes con un mes de anticipación. Por ejemplo, si se desea viajar en el mes de abril, a principios de marzo se ponen a la venta los pasajes para todo el mes de abril. Este formato genera un día en el que la gente está expectante a que se encuentren disponibles los pasajes,los cuales suelen agotarse ese mismo día.

Es relevante destacar que los trenes permiten la posibilidad de devolución del boleto, donde se reintegra un porcentaje elevado del importe. Los pasajes devueltos se vuelven a poner disponibles en la web. El público está pendiente de las devoluciones, por lo que una vez que aparece uno, suele estar disponible por poco tiempo.

Hoy en día, entre 72 y 24 horas antes del viaje, es necesario efectuar una confirmación de la reserva. Si no se realiza, el boleto se pierde y vuelve a aparecer disponible en la página. Este mecanismo surgió porque antes los tickets solían tener precios tan bajos que la gente los compraba por si acaso y no viajaban, lo que llevaba a múltiples reservar sin utilizarse.

## 5.2 Aspectos relevantes del modelo de negocio

En el contexto actual, obtener un producto o servicio de alta demanda puede requerir una considerable cantidad de tiempo dedicado a la constante consulta de disponibilidad, algo que no todos pueden permitirse. Este problema se vuelve especialmente en gastos de tiempo en situaciones donde la oferta es limitada y la competencia por los recursos es alta, como en el caso de la compra de pasajes de tren en rutas populares.

No se definió un modelo de negocio formal, sino que se identificaron y destacaron aspectos relevantes que impactan directamente en el desarrollo del software. Se propone una solución centrada en la adquisición de productos o servicios en nombre de terceros, a cambio de una comisión. Este enfoque libera al cliente de la carga de invertir tiempo y esfuerzo en la búsqueda, y permite que obtenga el producto o servicio deseado de manera eficiente y sin estrés. De esta manera, el usuario consigue su objetivo sin necesidad de involucrarse en el proceso de búsqueda y adquisición, mientras la plataforma se encarga de todo el trabajo necesario. Los pilares del negocio son la baja disponibilidad de los pasajes, y la posibilidad de devolución.

Segmentos de mercado:

- Cualquier persona que desee trasladarse en tren. Se hace foco principal en el tramo Buenos Aires - Mar del Plata por su baja disponibilidad y alta demanda.

Propuestas de valor:

- Simplificación del proceso de obtención de pasajes de tren, especialmente en rutas de alta demanda.
- Notificaciones automatizadas para alertar a los usuarios sobre la disponibilidad de pasajes.
- Integración de un proceso de compra completo y fácil de usar.

Canales:

- Página web adaptable a dispositivos móviles.
- Notificaciones por correo electrónico y WhatsApp para mantener a los usuarios informados.

Relaciones con clientes:

- Soporte técnico y atención al cliente a través de la aplicación y redes sociales.
- Comunicación continua para recibir feedback y mejorar el servicio.

Fuentes de ingresos:

- Comisiones por la venta de pasajes.

Recursos clave:

- Infraestructura tecnológica. Se incluyen servidores y servicios en la nube.
- Base de datos con información de los usuarios y pasajes.

Actividades clave:

- Desarrollo y mantenimiento de la aplicación y microservicios.
- Integración continua y despliegue de nuevas funcionalidades.
- Atención al cliente y soporte técnico.

Asociaciones clave:

- Alianzas con Trenes Argentinos y otras compañías de transporte.

Estructura de costos:

- Gastos de infraestructura y servicios en la nube.
- Costes de marketing y promoción.

## 5.3 Requerimientos

### 5.3.1 Requerimientos funcionales

| Identificador | Requerimiento | Descripción |
|---|---|---|
| **Acceso** | | |
| RF1 | Alta de usuarios | El sistema debe permitir que los usuarios se registren mediante un correo electrónico y una contraseña a través de una página web. |
| RF1.1 | Login | El sistema debe utilizar un sistema de login para autenticar al usuario a través de un sistema propio y de terceros. Por ejemplo, Google Auth. |
| RF2 | Modificacion de usuarios | El sistema debe permitir que los usuarios modifiquen su contraseña. |
| RF3 | Alta de datos de usuario | El sistema debe permitir que los usuarios registrados completen los datos necesarios para la adquisición del producto o servicio. En la búsqueda de pasajes de trenes<br>- Nombre<br>- Apellido,<br>- Apodo,<br>- Dni,<br>- Nro de Trámite,<br>- Género,<br>- Cod Área, |

| | | |
|---|---|---|
| | | - Número,<br>- Correo,<br>- Correo Trenes,<br>- Contraseña Trenes |
| RF3.1 | Validación de datos de usuario | El sistema debe permitir la validación de los datos ingresados por el usuario. En caso de ser erróneos o incompletos no se permitirá que el usuario realice el pedido de la búsqueda. |
| RF4 | Alta de servicio | El sistema debe permitir que los usuarios registrados soliciten su servicio o producto a través de un formulario. Se detalla:<br>- Fecha del viaje<br>- Origen<br>- Destino<br>- Horario de viaje |
| RF5 | Visualización de pedido | El sistema debe permitir que los usuarios registrados visualicen su pedido dentro de la página web. |
| RF6 | Baja de servicio | El sistema debe permitir que los usuarios registrados den de baja su servicio o producto a través de un botón. Se debe consultar al usuario si está seguro de la acción a realizar. |
| RF7 | Seguridad en baja de servicio | El sistema debe permitir la baja de servicio solo al usuario solicitante del mismo. |
| RF3.4 | Seguridad en los endpoints | El sistema debe exigir que el usuario esté autenticado para cualquier acción, excepto para el registro de usuario y el login |
| **Notificación y** | | |

| | | |
|---|---|---|
| **Pagos** | | |
| RF9 | Notificación al usuario | El sistema debe notificar al usuario vía email y Whatsapp en caso de que el sistema adquiera el producto o servicio. |
| RF10 | Compra del servicio | El sistema debe permitir la compra del servicio una vez el usuario haya sido notificado de la adquisición del servicio. |
| RF11 | Comisión del servicio | El sistema debe cobrar una comisión del servicio o producto adquirido. El sistema debe integrarse con Mercado Pago para generar links de pago |
| **Control y Monitoreo** | | |
| RF12 | Enviar nuevas búsquedas | El sistema debe permitir al administrador enviar búsquedas una vez se encuentran en la BBDD |
| RF13 | Automatización del sistema | El sistema debe ser capaz de recibir un pedido, almacenarlo en la BBDD y comenzar con la busqueda, solo si el boleto solicitado ya está a la venta |
| RF14 | Visualización de búsquedas activas | El sistema debe permitir al administrador la visualización de las búsquedas activas. |
| RF15 | Estado de las búsquedas | El sistema es responsable de manejar el estado de las búsquedas (activa, descartada, finalizada) |

### 5.3.2 Requerimientos no funcionales

| Identificador | Requerimiento | Descripción |
|---|---|---|
| RNFD1 | Usabilidad | La aplicación web debe ser accesible en una variedad de dispositivos y tamaños de pantalla para garantizar una experiencia de usuario consistente |
| RNFD2 | Seguridad | El sistema debe garantizar la seguridad del sistema en su totalidad.<br><br>*La protección de la información es crucial en general. Este aspecto se aborda de manera integral, asegurando que todos los datos del sistema estén protegidos frente a posibles amenazas. |
| RNFD3 | Mantenibilidad | El sistema debe mantener registros detallados de las operaciones realizadas, incluidos los errores, para permitir la auditoría. |
| RNFD4 | Disponibilidad y Confiabilidad | El sistema debe ser monitoreado continuamente para garantizar la disponibilidad y confiabilidad |
| RFND5 | Escalabilidad | El sistema debe ser escalable para la búsqueda de distintos productos o servicios |

# Capítulo 6: Diseño

El diseño del sistema ocupa un lugar fundamental en la concepción de soluciones tecnológicas que sean robustas, escalables y eficientes. Se brinda una perspectiva detallada de la interacción entre los diferentes componentes y actores del sistema que facilite el proceso de diseño y desarrollo.

Se presenta el diagrama de arquitectura y microservicios, donde se examina cómo la estructura global del sistema se organiza en módulos interconectados por un servidor. Se aborda la arquitectura y su aplicabilidad en el diseño de sistemas distribuidos, los cuales ofrecen escalabilidad, flexibilidad y alta mantenibilidad. Además, se realiza un análisis detallado para seleccionar las tecnologías más adecuadas dentro de cada componente, a partir de las necesidades específicas de cada parte del sistema.

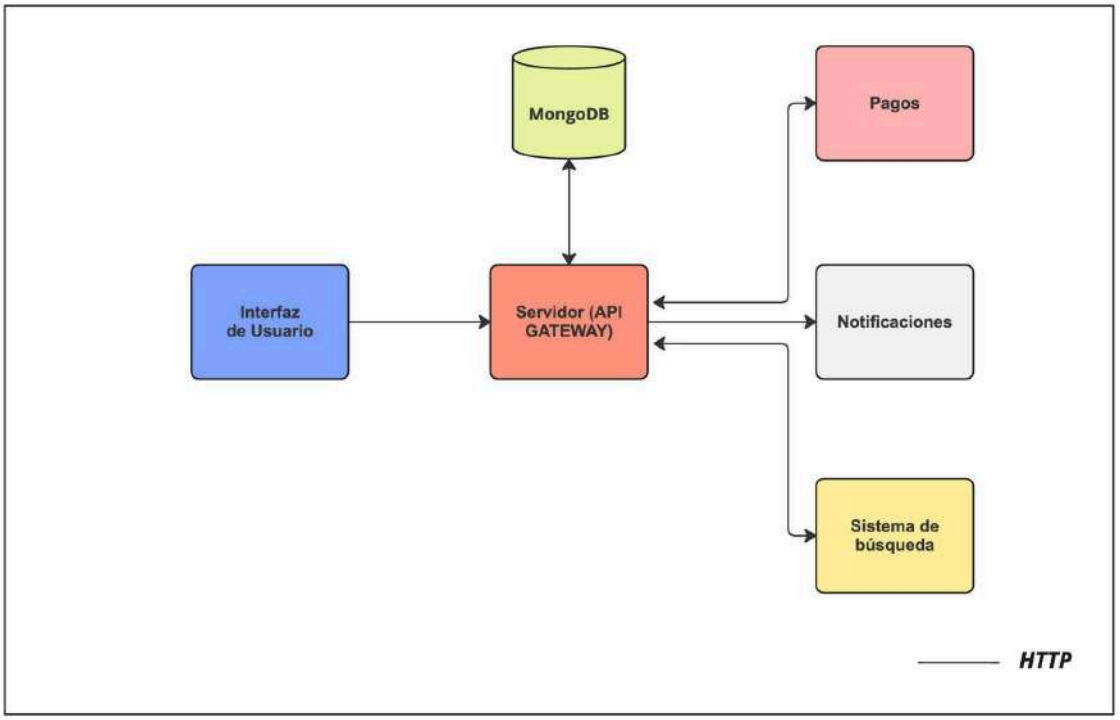
También se aborda el diseño de la base de datos, un aspecto elemental que afecta en la eficiencia y la integridad de la gestión de información dentro del sistema.

Por último, se detalla el uso de estrategias para maximizar la disponibilidad del sistema como Blue Green Deployment, basada en la implementación de servidores secundarios para el desarrollo de la aplicación y permitir un cambio gradual desde la etapa de desarrollo hasta el despliegue en producción.

## 6.1 Diagrama de arquitectura

El objetivo de la arquitectura es permitir que el usuario satisfaga sus necesidades de manera eficiente y asegurar una escalabilidad sencilla en el futuro. A continuación, se detalla la arquitectura planteada para el caso de Trenes Argentinos.

La arquitectura comprende una interfaz de usuario que se comunica directamente con el servidor API Gateway. Este último es responsable de la comunicación con el servicio de búsqueda, el de notificaciones, el de pagos y la base de datos. Los flujos de proceso propuestos se analizan detalladamente en un diagrama de secuencia más adelante.




### 6.1.1 Arquitectura basada en microservicios

La estrategia de desarrollar software con microservicios nace a partir de los problemas inherentes de las arquitecturas monolíticas, donde las funcionalidades están estrechamente acopladas y cualquier cambio puede tener repercusiones imprevistas en todo el sistema. Los

microservicios, por otra parte, proponen facilidades para implementar y realizar cambios en el software de forma rápida.

El modelo de negocio que abarca este proyecto, al depender de terceros, puede verse afectado por modificaciones en el proceso de venta u obtención del servicio, todo cambio derivará en una modificación que deberá sufrir también el software. Si bien, cada producto tendrá un proceso distinto, se pueden encontrar funcionalidades comunes en cada uno de ellos.

Es por ello que se propuso una arquitectura basada en microservicios.

### 6.1.2 Ventajas

- La modularidad de los microservicios se alinea perfectamente con el modelo de negocio del proyecto, lo que facilita la replicabilidad y la integración de componentes. Esto permite una escalabilidad más fácil y eficiente a medida que el proyecto evoluciona y crece.

- Cada microservicio es independiente, permite flexibilidad a la hora de utilizar la tecnología más adecuada para abordar los desafíos específicos de cada componente. Esto optimiza el rendimiento y la eficiencia de cada servicio, al tiempo que reduce la complejidad técnica del sistema en su conjunto.

- El desarrollo puede llevarse a cabo de forma simultánea que minimice el tiempo de desarrollo y, en consecuencia, los costos asociados.

### 6.1.3 Desventajas

- Al descentralizar la lógica del negocio, el monitoreo y la administración del sistema se vuelven más complejos. Se requiere una infraestructura robusta de monitoreo y gestión para garantizar la disponibilidad y el rendimiento del sistema en su conjunto.

- Las pruebas también son más complicadas en un entorno de microservicios. Además de las pruebas individuales de cada microservicio, se deben realizar pruebas exhaustivas de la comunicación entre los diferentes componentes para garantizar la integridad y la coherencia del sistema en su conjunto.

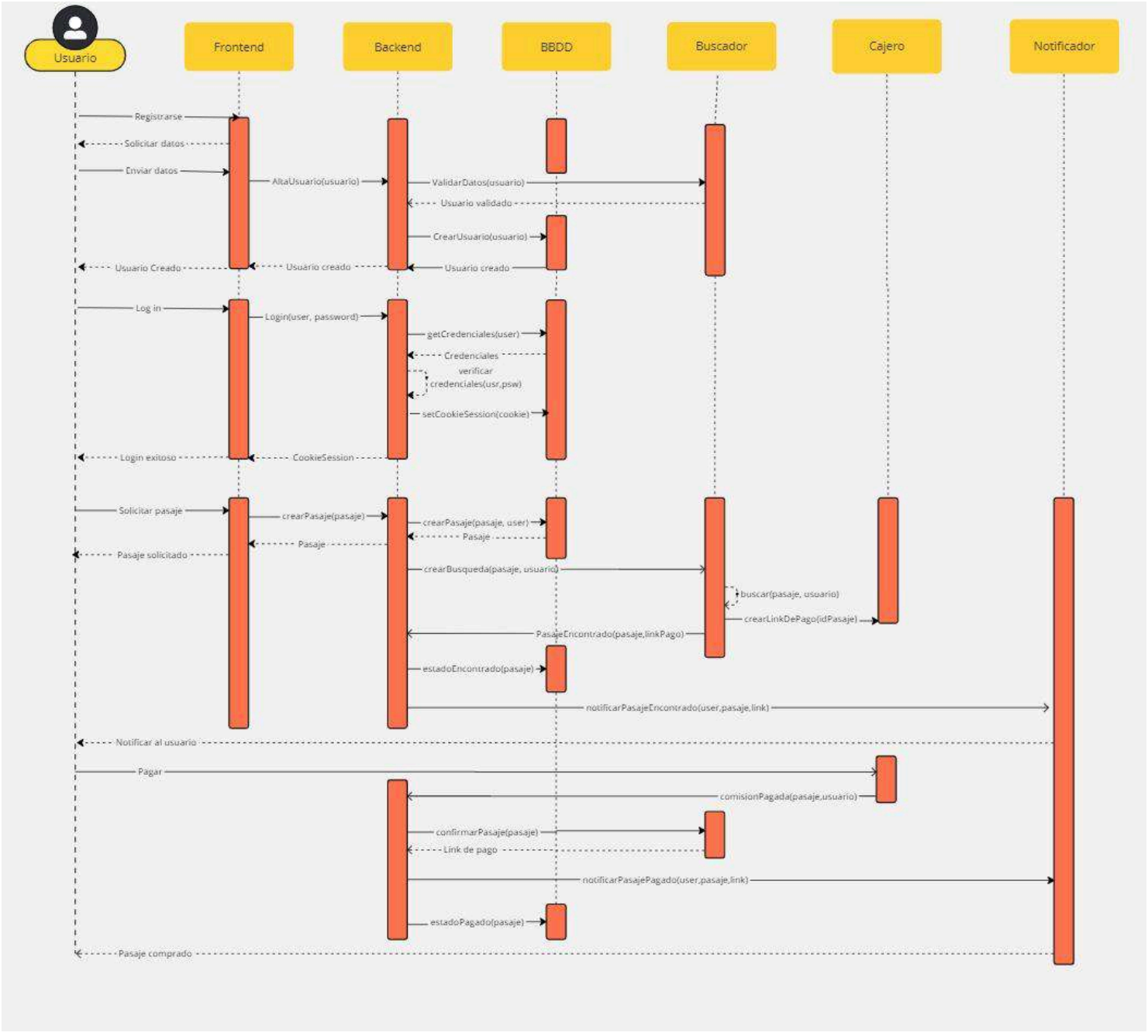
- El API Gateway, como punto centralizado de entrada a la arquitectura de microservicios, representa un solo punto de falla potencial. Un fallo en el API Gateway puede resultar en la indisponibilidad de todo el sistema, lo que subraya la importancia de implementar medidas de mitigación de riesgos y redundancia.

### 6.1.4 Comunicación

Para el proyecto, se utilizó una arquitectura del tipo REST. La arquitectura propuesta se basa en un modelo cliente-servidor, alojado en una página web, lo que permite la comunicación con el usuario a través de los protocolos HTTP / HTTPS. Se seleccionó esta arquitectura porque el equipo tenía experiencia previa en ella. De modo tal de no invertir tiempo en aprendizaje,ya que el proyecto estaba limitado por los plazos del mercado. Además, HTTP es el protocolo más utilizado hoy en día.

### 6.1.5 Diagrama de secuencia

A continuación se visualiza el diagrama de secuencia realizado con el fin de comprender cómo se relacionan los diferentes módulos del sistema. Para más detalles sobre la interacción de los módulos revisar el manual de administrador.



## 6.2 Front-end

La Interfaz de Usuario desempeña un papel central en la interacción entre los usuarios y el sistema, al facilitar la gestión de pedidos a través de una interfaz web intuitiva y eficiente. Para su desarrollo, se propone el uso de la tecnología React junto a la librería de estilizado de componentes, Styled Components y la herramienta de desarrollo ESLINT.

El sistema web asumirá diversas responsabilidades esenciales, se incluye el proceso de inicio de sesión de los usuarios, la validación de los datos proporcionados, así como la gestión de altas y bajas de pasajes.

Es importante señalar que la comunicación de la interfaz de usuario es únicamente con el servidor. Esta medida se tomó para garantizar la coherencia y la seguridad en la interacción con los microservicios subyacentes. La limitación del acceso directo a los endpoints del servidor contribuye a preservar la integridad del sistema y proteger la privacidad de los usuarios.

### 6.2.1 Diseño web

Durante el diseño (UI-UX) de la página web, se tomaron decisiones con el propósito de asegurar una experiencia satisfactoria para el usuario. El objetivo del sistema es simplificar la adquisición del servicio mediante un proceso de solicitud de pasaje sencillo y sin dificultades. En consecuencia, se planteó la creación de una página web que, a primera vista, resultara atractiva para el usuario y que revelará cuál es su propósito: simplificar la reserva de pasajes de tren. Además, es esencial que el usuario comprenda el proceso de compra, por lo que se ideó una guía paso a paso que explicara detalladamente cómo funciona dicho proceso.

Al final del proceso de diseño, se decidió que la página web sólo permitiría a los usuarios reservar pasajes para sí mismos. Si una persona desea viajar con alguien más, esa otra persona deberá crear su propia cuenta en el sistema. Esta decisión se basó en la premisa de que siempre se requieren los datos del usuario para solicitar un pasaje. Esto garantiza un servicio simple y directo: para obtener un pasaje, simplemente bastaría con crear una cuenta en la página web y realizar la solicitud. Esto simplifica el proceso, ya que elimina la

necesidad de gestionar reservas para múltiples personas desde una sola cuenta, y proporciona una experiencia más clara y fácil de usar para los usuarios.

El diseño debe cumplir con la cobertura de los requerimientos abordados en el análisis.

## 6.3 Servidor

El servidor cumple un papel fundamental dentro de esta arquitectura de microservicios , ya que actúa como  API Gateway, es decir es el punto de comunicación entre los distintos microservicios. Se comunica con el resto de los microservicios a partir de una API REST. Para la construcción del servidor se decidió utilizar Node JS , junto a la librería Express. El servidor además de tener la responsabilidad de encargarse de la comunicación y el acceso a la base de datos, también se encarga de manejar las sesiones de los distintos usuarios dentro de la página y asegurarse de la autenticación del usuario.

### 6.3.1 API Gateway en una arquitectura de microservicios

**Ventajas**:

Seguridad mejorada:

Centraliza la aplicación de reglas de seguridad y filtros para proteger los microservicios.  Gestiona la autenticación y autorización de manera centralizada.

Simplificación para los clientes:

Ofrece una única URL de entrada para los clientes. Simplifica la integración y permite al API Gateway adaptar las solicitudes y respuestas según necesidades específicas.

Monitoreo:
Facilita la supervisión del tráfico y la recolección de estadísticas detalladas en un punto común en el sistema.

Desacoplamiento:
Permite que los microservicios evolucionen de forma independiente.

**Desventajas**

Punto único de fallo:

Si el API Gateway falla, puede interrumpir el acceso a todos los servicios.

Latencia adicional:

Forzar a que cada solicitud de intercomunicación entre los microservicios pase por una entidad central puede aumentar la latencia de las solicitudes

Cuello de botella potencial:
De no gestionarse de manera correcta, puede convertirse en un punto de congestión que limite el rendimiento general del sistema.

## 6.4 Base de Datos

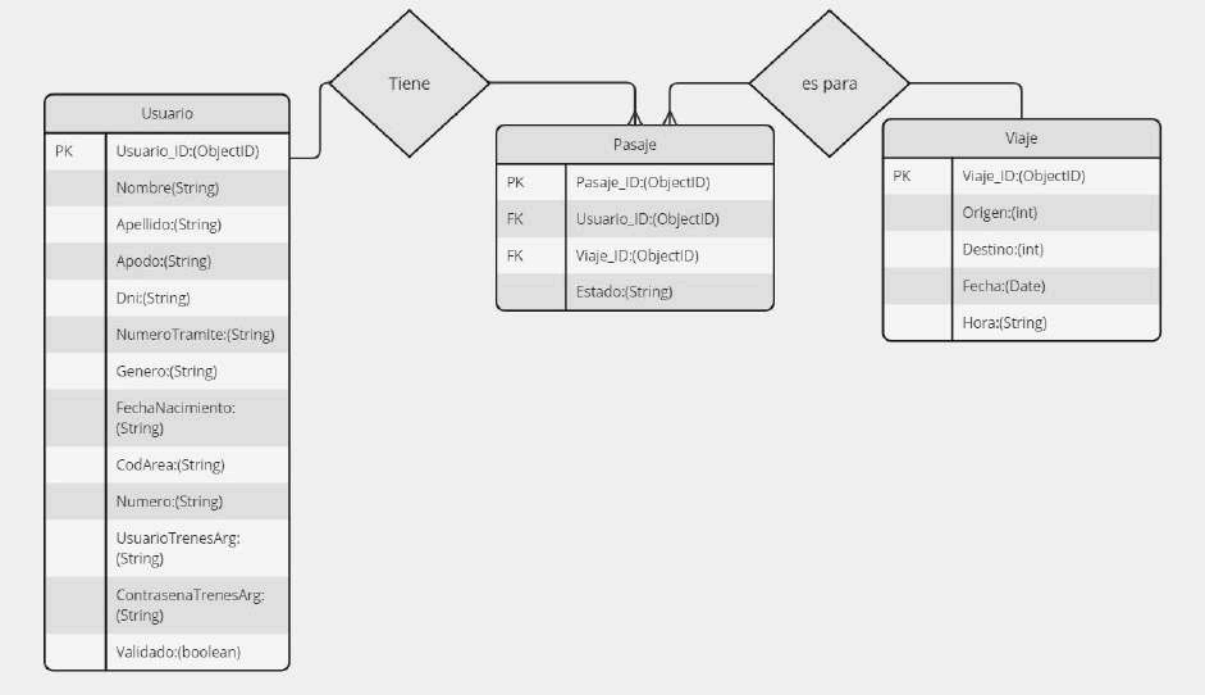
En este apartado, se aborda el proceso de diseño de una base de datos que establece una comunicación únicamente con el servidor. Para este fin, se optó por adoptar el paradigma NoSQL, con la elección específica de MongoDB como sistema de gestión de bases de datos.

Se muestra un diagrama de colecciones que sirve como guía para la estructura y organización de los datos. Permite no solo establecer una sólida arquitectura de base de datos, sino también facilitar la futura implementación dentro del código. Al tener un diseño claro y bien definido, se agiliza el desarrollo y se garantiza una integración efectiva con aplicaciones y servicios del equipo.

### 6.4.1 Caso de uso

Las principales razones para la elección del paradigma NoSQL fueron la experiencia del equipo al trabajar anteriormente con este paradigma y la flexibilidad que otorga en los esquemas de datos. El paradigma NoSQL, y específicamente MongoDB, permite almacenar la información de manera flexible. En lugar de seguir un esquema rígido como en las bases de datos relacionales, se puede almacenar los datos de usuarios, pasajes y viajes en documentos de MongoDB de manera que se adapten perfectamente a necesidades específicas. Esto es especialmente útil para el caso del usuario, donde se necesita almacenar una variedad de datos que pueden variar de un usuario a otro, como detalles de contacto, preferencias de viaje, etc. Asimismo, es útil en el contexto en el que existe una gran dependencia de terceros y la dependencia de la situación del mercado es muy alta. La base de datos permite adaptarse de forma rápida a cambios en los requisitos del sistema o a nuevas características del mercado, sin necesidad de realizar modificaciones complejas en la estructura de la base de datos.

### 6.4.2 Modelo de datos




El modelado está diseñado para la búsqueda de pasajes de Trenes Argentinos.

## Usuario

La entidad Usuario tiene como utilidad almacenar toda la información que Trenes Argentinos solicita con el fin de emitir un pasaje. Para mejorar la experiencia de usuario, se propuso almacenar todos esos datos de modo de que el usuario solo debe brindar la información una sola vez, durante la creación del usuario.

Además, se decidió agregar un campo extra “validado” que permite determinar si los datos del usuario son correctos.

Este campo permitirá únicamente que soliciten pasajes aquellos usuarios que se encuentran con los datos validados por el sistema.

## Pasaje

La entidad Pasaje simboliza el pedido de búsqueda por parte del usuario. Cada pasaje pertenece a un usuario, un usuario va a poder solicitar más de un pasaje. Se asocia a un viaje en particular. Asimismo, se agregó un atributo estado el cual simboliza si el pasaje está actualmente en búsqueda, fue comprado, o fue cancelado. Su implementación es esencial para permitir una gestión autónoma del sistema de búsqueda.

## Viaje

La entidad Viaje es utilizada para almacenar el origen, destino, fecha y horario en que el usuario desea viajar. Se utiliza con el objetivo de normalizar la base de datos. Los datos de un viaje son información que podría haber estado dentro de la entidad pasaje, pero se decidió crear una entidad nueva para que el pasaje haga referencia al id y evitar información repetida.

## 6.5 Servicio de búsqueda

El servicio busca optimizar la obtención y compra automatizada de pasajes en Trenes Argentinos. Inicialmente, se utilizaron pruebas con Selenium, una herramienta de código abierto diseñada para la automatización de navegadores web, se encontró que manejar múltiples instancias de navegador no era escalable debido a la alta potencia computacional requerida.

### 6.5.1 Decisiones de diseño

Luego de analizar como es el proceso de compra en la plataforma oficial de Trenes Argentinos, se detectaron dos obstáculos grandes principales que había de resolver en la etapa de diseño del mismo:

- La resolución de captcha: Este captcha aparece cada vez que se desea consultar la disponibilidad para un determinado viaje. Se supone que fue una decisión del equipo de trenes argentinos para disminuir el tráfico al servidor principal con el fin de evitar que los usuarios puedan refrescar constantemente en busca de disponibilidad

- Manejar una gran cantidad de solicitudes de consultas: Al considerar que cada mes tiene treinta días, con un foco únicamente del ramal Mar del Plata - Buenos Aires, se tienen dos sentidos de viaje diarios, y en cada sentido se tienen dos posibles horarios. Es decir, 30*2*2 = 120 diferentes viajes posibles dentro de un mismo mes. El sistema debe estar preparado para soportar la concurrencia de sesiones para cada viaje deseado. En cuanto a decisiones de diseño no sería óptimo crear una sesión por cada usuario, donde podría ocurrir el caso en el que varias sesiones consulten la misma información lo que generaría competencia entre los usuarios. Esto podría llevar a que dos o más usuarios consulten por la misma información a la hora de encontrar un pasaje que ambos quieran obtener el mismo pasaje. Además generaría un gasto innecesario de recursos.

Python fue elegido por su capacidad de multithreading e integración con IA para la resolución de CAPTCHA.

## 6.6 Servicio de pagos

El objetivo de este servicio es recibir pagos de los usuarios, generar enlaces de pago y que, cuando se realice el pago, se pueda ejecutar una acción que desencadene otra. El equipo conocía la existencia de la API de MercadoPago, que permite la integración en proyectos web, por lo que se decidió utilizarla como pasarela de pago dentro del proyecto. Se analizaron las distintas posibilidades que ofrecía MercadoPago y la solución que se adecuaba al caso del equipo fue la de MercadoPago Checkout Pro.

## 6.7 Servicio de comunicación

Para implementar este microservicio, utilizamos un servidor en Node.js para reutilizar funcionalidades del servidor.

La comunicación y automatización de mensajes se gestiona a través de herramientas como Selenium y APIs. Selenium permitirá la automatización de interacciones web, y asegura que los mensajes se envíen de manera efectiva. Al mismo tiempo se estudió opciones de proveedores de APIs, que proporcionan una interfaz robusta para la integración con servicios de terceros, donde se garantiza la fiabilidad y eficacia del sistema.

# Capítulo 7: Producto

En esta sección se detalla cómo fue llevada a cabo la implementación del producto y las decisiones tomadas en cada componente del sistema. A lo largo del desarrollo del proyecto, se realizaron varias iteraciones para garantizar que el producto final cumpliera con los requisitos establecidos y proporcionar una experiencia de usuario óptima. Se explica cómo funciona cada componente, las tecnologías utilizadas y la manera en que se llevaron a cabo las distintas iteraciones.

El sistema se compone de varios componentes clave, cada uno de los cuales desempeña un papel crucial en la funcionalidad global del producto. A continuación, se describen los componentes principales y sus detalles técnicos.

| **Componente** | **Tecnologías y Herramientas** | **Frameworks y Librerías Más Utilizadas** | **Despliegue** |
|---|---|---|---|
| **Frontend** | React | Redux | Netlify |
| **Backend** | NodeJS | Express, Passport.js | Google Cloud |
| **Microservicio de Pagos** | NodeJS | MercadoPago API | DigitalOcean |
| **Microservicio de Notificaciones** | NodeJS | WhatsApp Web JS | DigitalOcean |
| **Microservicio de Búsqueda** | Python | TensorFlow, Flask | Local |

- Frontend:
    - Tecnologías: React para la construcción de la interfaz de usuario, desplegado en Netlify para una entrega continua y manejo de la CDN.
    - Frameworks y Librerías: Se utiliza Redux para la gestión del estado de la aplicación.
    - Despliegue: Netlify, una plataforma de alojamiento estático con integración continua.
- Backend:
    - Tecnologías: NodeJS para la lógica del servidor, desplegado en Google Cloud para aprovechar su infraestructura escalable.
    - Frameworks y Librerías: Express para la creación de la API y Passport.js se utiliza para la autenticación y manejo de sesiones.
    - Despliegue: Google Cloud, para asegurar escalabilidad y confiabilidad.
- Microservicio de Pagos:
    - Tecnologías: NodeJS para la implementación del servicio.
    - Frameworks y Librerías: MercadoPago API Checkout Pro para las integraciones de pago.
    - Despliegue: DigitalOcean
- Microservicio de Notificaciones:
    - Tecnologías: NodeJS para el servicio de notificaciones.
    - Frameworks y Librerías: WhatsApp Web JS para enviar notificaciones a través de WhatsApp.
    - Despliegue: DigitalOcean.
- Microservicio de Búsqueda:
    - Tecnologías: Python para implementar la lógica de búsqueda.
    - Frameworks y Librerías: TensorFlow para el procesamiento de datos y aprendizaje automático. Flask para la creación de la API
    - Despliegue: Local, con posibilidad de evaluar otras opciones en el futuro.

## 7.1 Frontend

Para la implementación de la interfaz gráfica, seguimos el plan establecido durante las etapas de diseño. El equipo inició el desarrollo con React. La prioridad inicial fue lanzar el sitio web lo más rápido posible debido a la alta demanda de pasajes esperada durante el verano. Con el objetivo de lanzar una web funcional y aplicar los distintos cambios a medida que se implementen.

Para el diseño de la página,  se contrató a un diseñador gráfico que se encargó de darle una identidad visual a la plataforma, logrando un atractivo comercial más viable. Es importante mencionar que el diseñador gráfico proporcionó plantillas en formato PDF, y los estudiantes se encargaron de plasmar esas ideas en código.

Se realizaron entrevistas para definir la dirección de la marca y sus objetivos. El proceso de trabajo fue el siguiente: primero, se creó un boceto de la página web en React según el diseño establecido. Posteriormente, tres entrevistas para definir la marca de la aplicación. Una vez establecida la identidad de la marca, incorporamos esos elementos, como logotipos, eslogan y colores, a la aplicación.

Finalmente, una vez definida la identidad de la marca, el diseñador colaboró en el diseño de la página para mejorar su aspecto estético.

## 7.2 Servidor

El microservicio se creó en un servidor Node.js con Express. Este servidor es responsable de comunicarse con la base de datos y funciona como API Gateway dentro de la aplicación, es decir, todas las solicitudes pasan a través del servidor. La implementación de este microservicio es crucial debido a su papel en asegurar la escalabilidad, la eficiencia y la mantenibilidad del sistema.

## 7.3 Servicio de pagos

El microservicio de notificaciones se creó en un servidor en Node.js con Express. Este microservicio se encarga de la creación y recepción de pagos. Al implementar pasarelas de pago, se puede optar por crear un formulario propio o delegar todo a un tercero. Para la solución, se decidió utilizar MercadoPago debido a su rapidez, comodidad y alta fiabilidad en transacciones.

Al ofrecer un servicio que brinda una opción alternativa para la obtención de un producto, pareció primordial utilizar MercadoPago para asegurar la confianza del usuario. La gente reconocería el método de pago y no sentiría que sus datos estaban comprometidos al no ingresar información en una página desconocida.

Una vez decidida la implementación con MercadoPago, se analizó cómo integrarlo, dado que el equipo no tenía experiencia previa con pasarelas de pago. MercadoPago ofrece distintas APIs según el tipo de solución requerida. En el caso de estudio, se necesitaba generar un link de pago que indique el monto a pagar de la comisión y tercerizar el proceso de compra hacia MercadoPago. Para esto, MercadoPago ofrece la solución llamada Checkout Pro.

## 7.4 Servicio de comunicación

El microservicio de comunicación es esencial en esta arquitectura, se encarga de notificar a los distintos usuarios cuando se encuentra un pasaje. Su principal característica debe ser la capacidad de manejar una gran cantidad de mensajes concurrentes. Debe ser lo suficientemente robusto para soportar un alto volumen de transacciones. Es común que el microservicio de búsqueda obtenga pasajes en lotes, es decir, que encuentre 20 pasajes a la vez, por lo que el microservicio de comunicación debe ser capaz de notificar a estos 20 usuarios simultáneamente.

Además, después de notificar a estos usuarios, el microservicio debe poder informarles rápidamente cuando se haya recibido el pago de la comisión y poder proporcionar a los usuarios el enlace de pago del pasaje. Cualquier falla en este microservicio se consideraría una vulnerabilidad para el usuario, ya que es una parte crucial del proceso de compra. Por lo tanto, su relevancia es muy alta.

En un principio, la primera implementación de este microservicio consistía en un sistema de notificación por correo electrónico. Aunque este método era robusto, presentaba una desventaja significativa: las personas no suelen revisar su correo electrónico con frecuencia, lo que podría resultar en una baja tasa de ventas.

El objetivo era que este microservicio notificará a los usuarios a través de WhatsApp. Al integrarlo con una aplicación móvil ampliamente utilizada en Argentina, se esperaba llegar a un público más amplio, que otorgara una mejor experiencia al usuario. La probabilidad de que los usuarios vean la notificación de que se les ha encontrado un pasaje sería mucho mayor, para mejorar la efectividad del servicio.

Para la implementación se realizó un servidor en Node.js y Express que interactúa con WhatsApp Web a través de Selenium para enviar mensajes automatizados.

## 7.5 Servicio de búsqueda

Para el microservicio de búsqueda se utilizó Python como lenguaje principal en el desarrollo del microservicio. Permite la programación con hilos (Threads), que es útil para la compleja administración de muchas sesiones. El equipo utilizó TensorFlow (una libreria de Python dedicada a Machine Learning) para el desarrollo de una inteligencia artificial capaz de resolver el captcha de Trenes Argentinos. Para recibir solicitudes HTTP se utilizó Flask.

El producto obtenido es un web service diseñado para interactuar con Trenes Argentinos.

Este servicio tiene varias funcionalidades esenciales en el funcionamiento del sistema:

- Procesamiento de las búsquedas
    - Recibe, procesa y obtiene las búsquedas que deben realizarse
    - Consulta de horarios
        - Los horarios que se muestran como opciones en el frontend son obtenidos mediante la interacción de este servicio con Trenes Argentinos
- Validación de datos
    - La información del pasajero debe ser correcta para que no impida la obtención del pasaje
- Actualiza la BBDD a través del backend
    - Estado de las búsquedas:
        - Comprados/Descartados
    - Estado de los usuarios:
        - Si los datos del usuario no son correctos, invalida al usuario

La decisión de diseño de utilizar Python fue un acierto por parte de los estudiantes. La capacidad de multithreading permitió al equipo enfocarse en un flujo único para la obtención de los pasajes, y utilizarlo por cada búsqueda que se solicite, variando únicamente los datos del usuario y del viaje.
Además, el equipo aprovechó las facilidades que Tensorflow ofrece para el desarrollo de inteligencia artificial y reducir el costo de la integración de la búsqueda con la resolución del captcha. El hecho de estar todo en el mismo servidor facilitó la integración.

## 7.6 Comparación entre iteraciones de productos

El desarrollo de este trabajo tiene la característica de estar influenciado por los tiempos del mercado, lo que limita la planificación debido a las demandas del mismo. Al comienzo del trabajo, solo se tenían versiones muy precarias del producto que, si bien podían gestionar la compra de pasajes, carecían de una arquitectura que permitiera su uso a gran escala. Por lo tanto, se proyectó obtener un MVP que satisficiera las necesidades básicas del cliente, y a medida que avanzara el proyecto, se agregarían más funcionalidades y mejoras.

Se decidió que, a lo largo del trabajo final, se implementarían mejoras basadas en el desarrollo de estas funcionalidades y en las demandas del mercado. En esta sección, se estudia cómo fue el avance de los distintos microservicios durante el desarrollo del mismo y qué mejoras se implementan en los diferentes lanzamientos.

### 7.6.1 Frontend

***Desarrollo del frontend previo al trabajo final***

Inicialmente, se utilizaba un formulario de Google Forms para que los usuarios pudieran cargar sus respectivos pasajes. Este fue el punto de partida, pero presentaba varios errores y desventajas. Para el cliente, resultaba incómodo tener que cargar sus datos personales cada vez que necesitaba un pasaje, lo que comprometía la experiencia del usuario. En cuanto a la gestión de los pasajes, también se veía afectada, ya que no existía ningún tipo de validación de los datos. Si había algún dato incorrecto, era necesario contactar al cliente, lo que demostraba que el modelo no era escalable.

**Aspectos positivos:**

- Implementación rápida con Google Forms.
- Permite recopilar información básica de los usuarios y sus pasajes.
- Solución inicial que permitió iniciar operaciones y aparecer en el mercado.

**Aspectos negativos:**

- Incómodo para los usuarios, ya que tenían que llenar el formulario completo cada vez.
- No había validación de datos, lo que generaba errores y la necesidad de validar los datos manualmente con la necesidad de contactar a los clientes para correcciones.
- No escalable por lo que dificulta la gestión de un número creciente de usuarios.

***Primer lanzamiento del frontend***

El primer lanzamiento del frontend tenía como objetivo permitir que los usuarios pudieran tener su cuenta dentro de la aplicación para mejorar su experiencia. De esta manera, cuando deseaban un pasaje, únicamente tenían que completar la información del viaje deseado sin necesidad de volver a completar con sus datos personales. Además, la validación de datos, que se lograba mediante el uso de una página web, era fundamental. Con estos dos factores era suficiente para hacer el primer lanzamiento de la página web. Sin embargo, todavía faltaban algunos aspectos por terminar: no se había contactado a un diseñador para las vistas, no se mostraban los textos del paso a paso ni las preguntas frecuentes, lo que hacía que la página fuera poco informativa, y no permitía cancelar las solicitudes. A pesar de esto, la página cumplía con el objetivo de poder solicitar pasajes.

**Aspectos positivos:**

- Mejora de la experiencia del usuario al permitir la creación de cuentas.
- Simplificación del proceso de solicitud de pasajes mediante un formulario de búsqueda.
- Implementación de la validación de datos para reducir errores.

**Aspectos negativos:**

- Falta de diseño profesional en las vistas, lo que afectaba la apariencia y usabilidad.
- Ausencia de textos informativos y preguntas frecuentes, lo que podía causar confusión.
- No permitía cancelar solicitudes lo que limita la funcionalidad.

**Desarrollo continuo y lanzamiento final**

A medida que se avanzaba en el desarrollo, se hizo un nuevo lanzamiento en el cual se implementó el diseño trabajado junto al diseñador. Se agregaron las secciones de paso a paso y preguntas frecuentes, así como las funcionalidades restantes. De esta manera, se obtuvo la versión final de la página.

**Aspectos positivos:**

- Diseño profesional y atractivo. Mejora en la experiencia visual y de usabilidad.
- Inclusión de textos informativos y preguntas frecuentes. Mayor claridad al usuario.
- Implementación de todas las funcionalidades necesarias,se incluye la opción de cancelar solicitudes.

### 7.6.2 Backend

***Primer lanzamiento del servidor***

El primer lanzamiento de la página web necesitó el soporte de un servidor que pudiera gestionar a los distintos usuarios y comunicarse con la base de datos y el resto de los microservicios. Este release se centró principalmente en acompañar la página web. Si bien esto permitió resolver varios problemas ya mencionados, principalmente en cuanto a la validación de datos, este lanzamiento no se encargaba de la gestión de los pasajes ni de la automatización de todo el proceso. Todas las tareas de gestión, como anotar qué pasajes se seleccionaron, cuáles se vendieron, cuáles las personas ya no necesitaban, entre otros casos, se realizaban manualmente, lo que representaba una gran desventaja para los alumnos.

Es importante destacar que, al principio, no se utilizaba una base de datos como soporte para los pasajes. Como consecuencia de utilizar Google Forms en conjunto con Google Sheets, la implementación del servidor comenzó con la gestión de los datos mediante Google Sheets. El servidor se encargaba de registrar los pasajes en la base de datos de MongoDB, pero esta no se utilizaba activamente; en su lugar, se realizaba una réplica en Google Sheets, desde donde se llevaban a cabo las distintas tareas de gestión de los pasajes.

En este release, ya se habían considerado cuestiones de seguridad y la posibilidad de identificarse por medio de una cuenta propia, o a través de la cuenta de Google.
La identificación era necesaria para saber con quien vincular la nueva búsqueda, y al usuario le brindaba una mejor experiencia de usuario, ya que ingresaba los datos personales una sola vez

**Aspectos positivos:**

- Resolución de problemas relacionados con la validación de datos.
- Soporte para la página web y gestión de usuarios.
- Permite iniciar operaciones básicas rápidamente.

**Aspectos negativos:**

- Gestión de pasajes y tareas administrativas realizadas manualmente.
- Alta carga de trabajo manual, lo que es ineficiente y propenso a errores.
- Dependencia de Google Forms y Google Sheets para la gestión de datos.

***Lanzamientos graduales del servidor***

En el segundo release del servidor, las decisiones fueron altamente influenciadas por los tiempos del mercado y la demanda del servicio. En lugar de enfocarse en la implementación de una base de datos como soporte e iniciar la automatización del proceso, se decidió continuar con el proceso existente junto a Google Sheet. Este release constaba de lanzamientos constantes y graduales que agregan funcionalidades poco a poco, con el objetivo de mejorar la gestión de pasajes de manera incremental. Se optó por esta opción debido al temor de no cumplir con los tiempos del mercado al implementar la automatización completa. Por lo tanto, se continuó con la semi-automatización de tareas con Google Sheets junto con App Script, lo que permitió realizar tareas automáticamente y con código, para facilitar la gestión y reducir la cantidad de tareas manuales. Este enfoque fue esencial, ya que permitió la disminución de tareas manuales y una adaptación progresiva a las necesidades del mercado.

**Aspectos positivos:**

- Continuidad en el uso de herramientas familiares (Google Sheets y App Script).
- Semi-automatización de tareas para reducir parcialmente la carga de trabajo manual.
- Adaptación rápida a las demandas del mercado sin retrasos significativos.
- Lanzamientos constantes y graduales con el fin de añadir funcionalidades de manera incremental.

**Aspectos negativos:**

- Falta de una base de datos robusta y soporte completo para la automatización.
- Persistencia de algunos procesos manuales y semi-automatizados, lo que puede ser ineficiente a largo plazo.
- No se aprovechó el potencial completo de la automatización debido a las restricciones de tiempo del mercado.

**Automatización del servidor**

Finalmente, el último lanzamiento del servidor se llevó a cabo con objetivos claros: poder automatizar el proceso y dejar de depender de Google Sheets. Se buscó que los distintos procesos de gestión utilicen la base de datos de MongoDB. Esta tarea parecía ser compleja en un principio, pero al tener ya definidos todos los casos de uso dentro del servidor y la base de datos que actuaba como registro de los pasajes, se pudo realizar de manera efectiva y sin complejidad adicional. Además, se realizó en un momento en el cual la demanda había disminuido, lo que permitió probar los distintos casos sin presiones adicionales.

En este release fue necesario tomar medidas de seguridad más rigurosas, ya que la existencia de endpoints que modifican el estado de la base de datos son peligrosos. Para los endpoints de comunicación entre los microservicios, se utilizó un Basic Auth.

**Aspectos positivos:**

- Automatización del proceso: eliminación de la dependencia de Google Sheets permitió automatizar las tareas de gestión, lo que mejoró la eficiencia y redujo errores.
- Independencia de plataforma: al utilizar la base de datos de MongoDB, se logró una mayor estabilidad y escalabilidad del sistema.
- Claridad en los casos de uso: haber definido previamente todos los casos de uso dentro del servidor facilitó la implementación de la automatización.
- Testeo en momento oportuno: el lanzamiento se realizó en un período de baja demanda, lo que permitió probar los distintos casos y realizar ajustes sin presiones adicionales.

**Aspectos negativos:**

- Complejidad inicial: Aunque la tarea parecía compleja al principio, la implementación resultó ser efectiva y sin complicaciones adicionales.
- Dependencia previa de Google Sheets: La transición de la dependencia de Google Sheets a MongoDB pudo haber requerido ajustes significativos en el código y procesos existentes.
- Potenciales errores durante la transición: La migración de datos y la implementación de la automatización podrían haber introducido errores en el sistema, lo que requeriría tiempo adicional para corregirlos y estabilizar el sistema.

### 7.6.3 Servicio de Búsqueda

***Prototipo inicial***

Las ideas concebidas durante el desarrollo del primer prototipo han servido como pilares fundamentales que perduran en el microservicio hasta la fecha. Estas ideas incluyen el diseño de entidades, los flujos de trabajo y la utilización de hilos. Inicialmente desarrollado en Java (Spring Boot) con el uso de Selenium, el software enfrentaba limitaciones

significativas en cuanto a escalabilidad debido a la necesidad de abrir una nueva instancia de navegador para cada búsqueda, lo cual resultaba en un alto costo computacional y falta de capacidad para escalar el sistema.

Las búsquedas se enviaban manualmente en formato JSON, a partir de Google Sheets y App Scripts. Cada búsqueda requería la creación de nuevas instancias de navegadores que iniciaban el proceso de búsqueda. Para la resolución de captchas, se desarrolló un script en Python invocado por cada instancia del navegador, lo cual ejercía una carga extrema sobre la capacidad de la computadora anfitriona. Además, este primer software adopta un enfoque monolítico, con la combinación del servidor receptor de peticiones, la integración con MercadoPago y el módulo notificador en una única aplicación.

A pesar de las limitaciones inherentes a la dependencia de navegadores, se implementaron estrategias para mejorar el producto existente, como la optimización de flujos de adquisición, diseño de estructuras de datos eficientes y gestión avanzada de hilos. Estas estrategias sentaron las bases para el diseño final del servicio de búsqueda.

**Aspectos positivos:**

- Producto funcional desde el inicio.
- Diseño eficiente de clases y estructuras de datos.
- Recopilación de información valiosa para identificar flujos potenciales y desarrollar estrategias futuras.

**Aspectos negativos:**

- Arquitectura monolítica que limita la escalabilidad.
- Dependencia crítica de instancias de navegadores para operaciones cruciales.
- Rendimiento y eficiencia.

### *Transición a Python*

La transición a Python fue más compleja de lo que el equipo estimaba. Aunque el sistema iba a abarcar únicamente las acciones de comunicación con Trenes Argentinos y el diseño ya estaba planeado, Python carece de herramientas que faciliten la detección de errores en comparación con Java. Los errores en Python aparecen en tiempo de ejecución, a diferencia de Java, que permite detectar errores antes de la compilación.

Además, la transición a Python se realizó bajo presión debido a los plazos del mercado. Mar del Plata es un destino turístico en temporada de verano, y los pasajes para diciembre, enero y febrero de la temporada 2023-2024 estaban próximos a salir a la venta. Era necesario tener un sistema que soportara esta nueva gran demanda.

Para fines de noviembre de 2023, la transición se había concretado con una notable mejora en efectividad, rapidez y volumen de búsquedas concurrentes. El sistema pasó de soportar un máximo aproximado de 20 usuarios concurrentes a un máximo aproximado de 600. La velocidad y la eficacia se incrementaron notablemente, ya que gracias a la ingeniería inversa se logró la comunicación directa con el servidor de Trenes Argentinos.

Debido al rápido desarrollo para cumplir con los plazos, no se desarrollaron pruebas unitarias ni de integración. Aunque el sistema comenzó a ser utilizado y satisfacía la demanda, presentaba muchos errores y debía reiniciarse reiteradamente. Además, no se contaba con un sistema de logging, lo que dificultaba la identificación y resolución de errores. Algunos errores eran silenciosos, es decir, no se detectaban, pero impedían encontrar muchos de los pasajes solicitados.

Las búsquedas aún debían enviarse manualmente, ya que aún se dependía de Google Sheets.

**Aspectos positivos:**

- Software escalable, más rápido y eficaz
- Mayor volumen de búsquedas
- Utilización de correctos lenguajes y librerías para la resolución de los problemas

**Aspectos negativos:**

- Sin comunicación con el backend
- Las búsquedas aún debían enviarse de forma manual
- Existencia de errores, y necesidad de reiniciar el sistema frecuentemente
- Deploy no concretado

***Corrección de fallas y automatización***

En la versión más reciente del microservicio, se implementó un sistema de registro detallado (logging) y se realizaron pruebas exhaustivas para identificar y resolver problemas potenciales. Paralelamente, se trabajó en la automatización del servidor para eliminar la dependencia de Google Sheets. Además, el microservicio de búsqueda fue ajustado para este propósito específico. Al iniciar, el sistema de búsqueda solicita al backend todas las búsquedas correspondientes y se encarga de actualizar su estado, ya sea cuando se completan o se cancelan. Además, las búsquedas son procesadas de manera inmediata tan pronto como son recibidas por el microservicio.

Sin embargo, en esta fase aún no se logró realizar el despliegue adecuado del microservicio en producción. El software presenta altos requisitos computacionales y la integración con TensorFlow representa un desafío significativo, dado que hay limitada información disponible en línea sobre cómo implementar un servicio web junto con esta librería.

**Aspectos positivos:**

- Registro detallado para la detección temprana de errores.
- Automatización completa del servidor.
- Procesamiento inmediato de las solicitudes de búsqueda.
- Actualización automatizada de la base de datos cuando las búsquedas se completan o cancelan.

**Aspectos negativos:**

- Dificultades en la puesta en producción del microservicio.
- Altos requisitos computacionales y complejidad debido a la dependencia de TensorFlow.

***Intervención durante suba de precios***

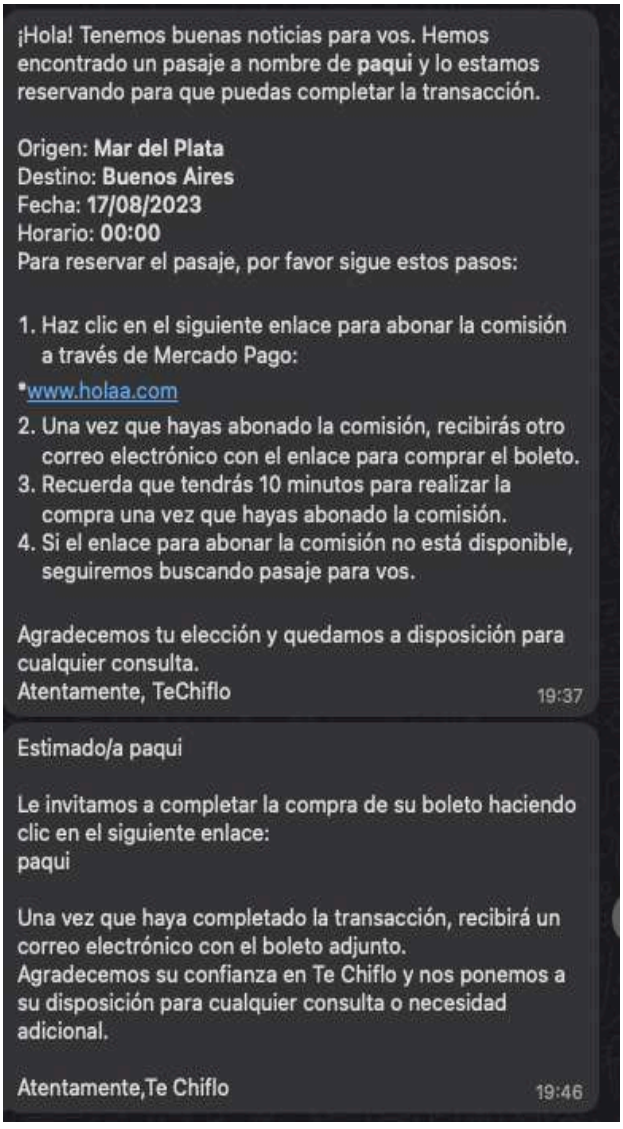
En un release posterior, la suba de precios generó una reducción en la demanda. Para incentivar las compras, se implementó la selección automática de la opción más barata, es decir entre primera categoría y pullman siempre se eligió el boleto menos costoso.

### 7.6.4 Servicio de Notificaciones

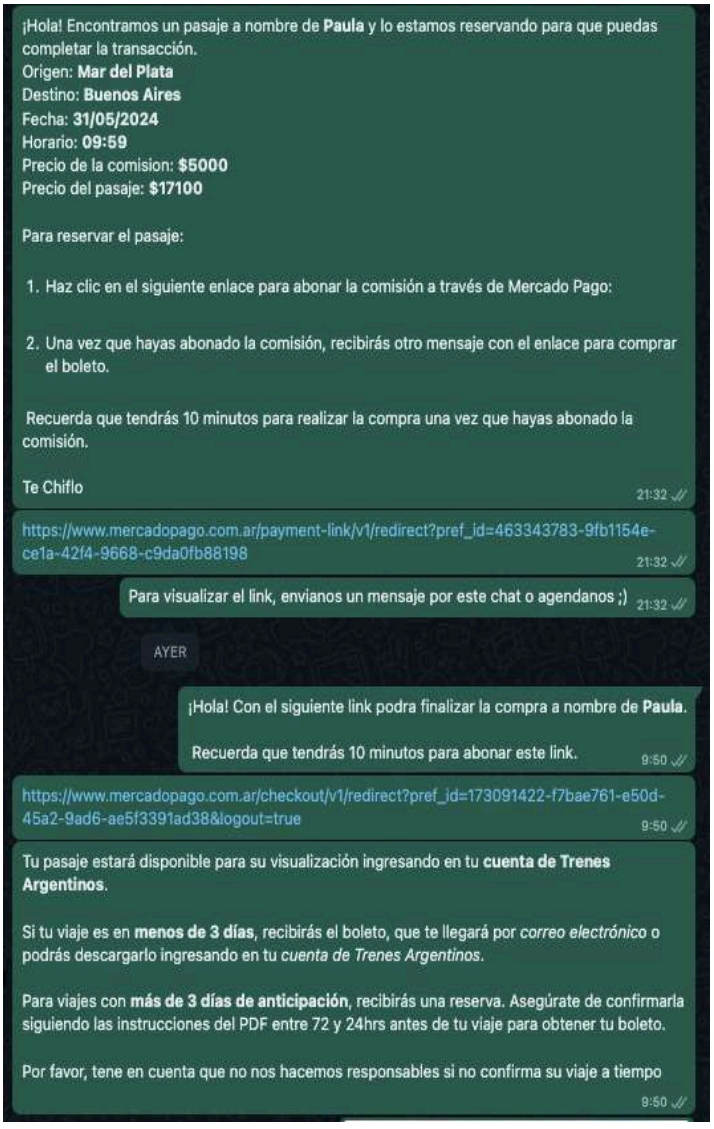
El microservicio de notificaciones se planeó desde el principio con el objetivo de utilizar WhatsApp como medio de comunicación principal. Esto se logró desde el primer lanzamiento, aunque el microservicio no estaba completamente optimizado. Se trabajó en la implementación de parches sobre la versión actual para que pudiera manejar una mayor cantidad de mensajes concurrentes.

Un aspecto muy importante fue la redacción de los textos enviados por este medio. Era esencial que la gente entendiera cómo finalizar los pagos. Con el tiempo, y en función de las distintas preguntas y necesidades de los usuarios, se modificaron los textos que enviaba WhatsApp con el fin de eliminar información redundante y mejorar la claridad de los mensajes.

**Mensaje inicial:**




**Mensaje actual**



El contenido del mensaje comienza con el nombre de la persona que está próxima a recibir el boleto. En reiteradas ocasiones, los usuarios viajaban en grupo y una sola persona se encargaba de la gestión de la compra, por lo que era necesario informar a quien pertenecía el pasaje.

También era crucial informar los datos del viaje,  como el origen, destino, fecha y horario. Es común que los usuarios necesiten viajar ida y vuelta, o que les sea indistinto diferentes horarios. En este último caso, el sistema podría encontrar diferentes horarios, y el usuario elegiría el más conveniente.

Informar los costos de antemano fue una decisión importante. Inicialmente los usuarios pagaban la comisión sin tener noción del costo que tendría el pasaje. Luego de la suba de los precios, el dato pasó a tener más relevancia. Los usuarios pagaban la comisión y luego les parecía caro el precio del boleto. Los reintegros se realizan de forma manual por parte de los emprendedores, entonces se eliminó este caso mediante la comunicación de los costos.

Luego se informa que debe abonar el enlace donde se especifica que el link corresponde a la comisión del servicio y no al pasaje. Además, se explica que es importante que una vez abonado este link, le llegara el link final correspondiente al boleto, con un tiempo de expiración de diez minutos.

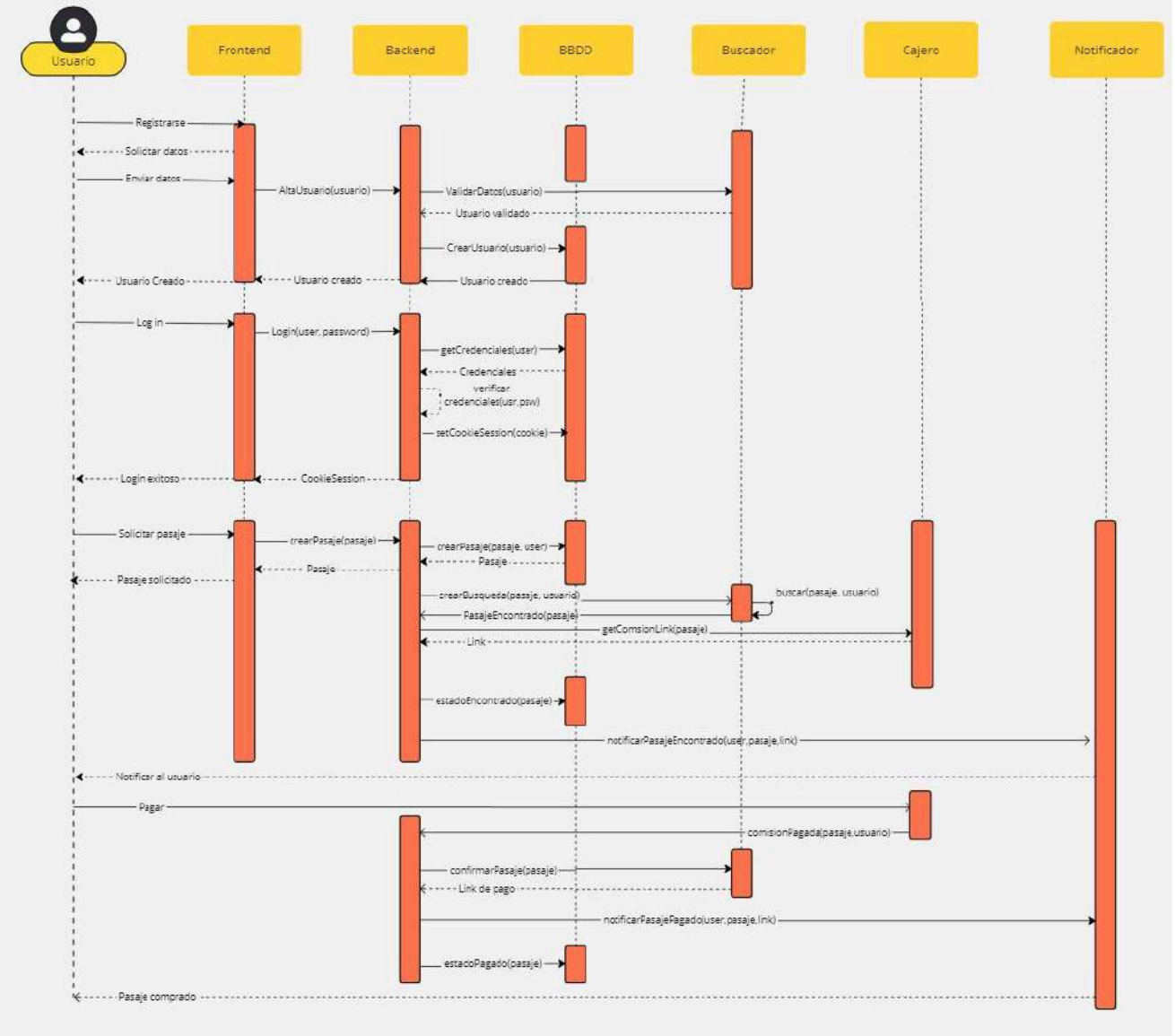
Por último, se envía el link final e información adicional de cómo funciona el proceso que corresponde realizar en Trenes Argentinos. Se deja en claro que el servicio prestado por el equipo fue satisfactorio, y que el cumplimiento de los pasos siguientes son responsabilidad del usuario.

Los mensajes variaron con el tiempo con el objetivo de comunicar de la forma más eficaz el proceso para evitar confusiones. Lamentablemente no era conveniente ser extremadamente detallista con los mensajes, ya que los usuarios no leen el contenido. El mensaje final obtenido es la evolución de diferentes versiones donde se aprecia que no hay información redundante y está bien aclarado que hacer una vez recibas tu boleto.

### 7.6.5 Servicio de Pagos

El microservicio de pagos de todos los servidores es el menos complejo algorítmicamente y el que realiza menos operaciones. Sin embargo, durante la implementación del producto aparecieron distintos inconvenientes para la gestión de los pagos debido a que la documentación no explicaba con claridad cómo abordarlos. Así, se realizó un primer lanzamiento de MercadoPago, y posteriormente se hicieron cambios para contemplar casos extremos que surgieron en la práctica diaria, como rechazar los pagos con saldo insuficiente.

## 7.7 Funcionamiento en conjunto de los servicios




A continuación se detalla el funcionamiento de todo el sistema en su conjunto. Abarca las interacciones entre los diferentes servicios, se contempla únicamente el caso de éxito. Desde que el usuario solicita el pasaje por la web, hasta que obtiene su pasaje.

El proceso comienza cuando el usuario necesita viajar en tren, y no encuentra disponibilidad en la página de Trenes Argentinos. Para poder utilizar el servicio, este necesita crearse una cuenta en la web.

A través de un formulario, el usuario envía sus datos personales al servidor. Como primer paso, el servidor necesita validar los datos contra Trenes Argentinos, por lo que se comunica con el microservicio de búsqueda para que este lleve a cabo la tarea.
Una vez validados los datos, el servidor crea el usuario y lo guarda en la base de datos.

Posteriormente, el usuario necesita identificarse ante el sistema a través de un login. El usuario ingresa usuario y contraseña a través de un formulario (o con Google) y el servidor resuelva la autenticación través de:

- Login propio:
    - Coincidencia de datos con la base de datos
- Google Auth
    - Transparente al desarrollo

Luego de la exitosa autenticación, el servidor crea una cookie de sesión y la almacena en la base de datos, junto a otros datos del usuario. Esta cookie es el ticket de entrada que el usuario utiliza para realizar diferentes acciones.

Una vez identificado el usuario, ya está habilitado para poder solicitar el pasaje deseado. A través de un formulario, el usuario envía los datos del viaje que desee. El servidor recibe la información y crea un nuevo pasaje a nombre del usuario solicitante:

- Lo guarda en la base de datos, y se obtiene un ID
    - Devuelve el pasaje creado al frontend
- Envía la búsqueda al microservicio de búsqueda para dar comienzo real a la búsqueda del pasaje

Ahora, el usuario queda en espera hasta ser notificado. En la práctica, esta espera es muy variada, pueden ser unos minutos, horas, o días. Ya que la disponibilidad depende exclusivamente de que otra persona devuelva el pasaje.

Cuando el microservicio de búsqueda obtiene el pasaje, este se comunica con el sistema de Pagos a través del backend para obtener el link de la comisión. El servidor actualiza el estado del pasaje en la base de datos y se comunica con el micro servicio de notificaciones. Se envian los datos del viaje, del pasajero, y el link de pago.

El microservicio de notificaciones notifica al usuario por Whatsapp y email. El usuario abona el link de pago. MercadoPago notifica al microservicio de pago, y se realizan validaciones para corroborar que el pago fue realizado con éxito.

Posteriormente, el microservicio de pagos se comunica con el backend para notificarle que el usuario abonó la comisión correctamente. El servidor se comunica con el servicio de búsqueda para que este finalice la compra y obtenga el link de pago final a través de Trenes Argentinos.

El servidor obtiene el link final para la compra del boleto, actualiza la base de datos, y nuevamente se comunica con el notificador para que éste comunique al usuario final.

### 7.7.1 Caching bajo demanda

Un aspecto crucial en el proceso de solicitud de boletos es la precisión del horario. El microservicio de búsqueda depende de esta información precisa para realizar búsquedas efectivas que reflejen la disponibilidad real de Trenes Argentinos.

El sistema de búsqueda cuenta con un endpoint dedicado a obtener los horarios disponibles para un viaje específico. Sin embargo, este proceso involucra resolver un captcha, lo cual implica un costo computacional significativo.

Para mitigar este impacto, se implementó una estrategia de cacheo bajo demanda. Esta estrategia implica consultar inicialmente la base de datos para los horarios disponibles de un viaje en particular. En caso de ser la primera consulta, se obtiene la información a través del microservicio de búsqueda y se almacena en la base de datos para consultas futuras. Las siguientes consultas a ese mismo viaje no requieren interactuar nuevamente con Trenes Argentinos, lo que mejora el rendimiento.

Esta optimización no solo reduce la carga sobre el microservicio de búsqueda, sino que también mejora la experiencia del usuario al acelerar la respuesta del frontend y reducir los tiempos de espera.

## 7.8 Puesta en producción

La puesta en producción de un producto de software es crucial en su ciclo de vida. Este proceso implica la implementación y el lanzamiento oficial del sistema tras su desarrollo y pruebas. Es el momento en el que el sistema comienza a ser utilizado por los usuarios finales. Esta etapa es crítica y desafiante, ya que supone la transición de un entorno de desarrollo, donde todo parece funcionar correctamente, a un entorno de producción con recursos limitados, donde pueden surgir nuevos errores que no se habían previsto.

Durante el desarrollo, los sistemas operativos de los desarrolladores contienen dependencias que son útiles para el sistema y que se dan por sentadas. Sin embargo, en los ambientes de producción, estas dependencias no están presentes por defecto y su ausencia provocaría errores.

Docker es una herramienta de virtualización de contenedores que simplifica el proceso de desarrollo, distribución y despliegue de aplicaciones. Docker proporciona una capa de abstracción sobre el sistema operativo subyacente que permite a los desarrolladores empaquetar aplicaciones junto con todas sus dependencias en contenedores autónomos y autosuficientes. Estos contenedores son independientes del entorno en el que se ejecutan con el fin de garantizar que las aplicaciones funcionen de manera predecible y consistente, independientemente de la infraestructura subyacente.

El uso de Docker durante las etapas de desarrollo facilita el proceso futuro de puesta en producción, por lo que se decidió complementar el desarrollo con Docker.

Actualmente, existen diversas plataformas como servicio (PaaS) que facilitan el proceso de puesta en producción para eliminar preocupaciones relacionadas con la configuración del servidor, las redes y el mantenimiento del sistema operativo. Esto permite a los desarrolladores enfocarse principalmente en el desarrollo. Algunos ejemplos de estas plataformas son Netlify y Google Cloud Run.

El equipo decidió utilizar estas PaaS para la puesta en producción de los diferentes microservicios. La tarea de puesta en producción resultó ser una de las más complejas para el equipo, debido a la falta de experiencia en el campo y el desconocimiento de la amplia

variedad de soluciones disponibles en el mercado. Además, cada proveedor ofrece distintas opciones, lo que incrementó la dificultad.

Cada microservicio se puso en producción varias veces, luego de una nueva feature significativa. Se buscó siempre la solución más práctica y económica posible. Debido a la falta de experiencia en la configuración de servidores como Infraestructura como Servicio (IaaS), inicialmente se optó por soluciones de PaaS exclusivamente.

### 7.8.1 Frontend

Netlify es una plataforma de alojamiento y servicios diseñada para aplicaciones web modernas. Ofrece una solución integral para el despliegue continuo, la gestión y el mantenimiento de sitios web y aplicaciones.

Una de las características más destacadas es el despliegue automático. La plataforma permite una integración directa con repositorios de código fuente como GitHub, GitLab y Bitbucket. Cada vez que se realizan cambios en el código y se actualiza el repositorio, se despliega automáticamente la nueva versión del sitio web. Esto simplifica enormemente el flujo de trabajo de los desarrolladores con el objetivo de eliminar la necesidad de gestionar manualmente el proceso de despliegue.

Para el despliegue de la página web, se optó por el uso de Netlify porque permite una integración muy simple con el código fuente desde GitHub, que facilita así el proceso de despliegue continuo. Una de las principales ventajas es su capacidad de redeploy rápidamente la aplicación y revertir a versiones anteriores en caso de ser necesario. Esto fue especialmente útil para el desarrollo y mantenimiento de la aplicación, ya que permitió manejar de manera eficiente los cambios y errores que pudieran surgir. Además, Netlify ofrece la administración de DNS del dominio de la página, lo que permitió realizar toda la configuración en la misma plataforma.

### 7.8.2 Backend

#### Google Cloud

Google Cloud es una plataforma de servicios en la nube ofrecida por Google, que proporciona una gama completa de productos y servicios para computación, almacenamiento, análisis de datos, inteligencia artificial y más.

#### Google Cloud Run

Google Cloud Run es un servicio de Platform as a Service (PaaS) que permite ejecutar aplicaciones basadas en contenedores sin necesidad de gestionar servidores. Se pueden conectar las aplicaciones en contenedores directamente desde GitHub.

La elección fue motivada por la necesidad de poner el servidor en producción lo más rápido posible y evitar perder tiempo en la configuración inicial. Con su capacidad de despliegue automático, escalabilidad y modelo de pago por uso, se presenta como una opción eficiente y efectiva para manejar el backend de aplicaciones modernas. Fue necesario variar los recursos computacionales asignados al servidor, con el fin de lograr un alto rendimiento con el menor costo posible.

### 7.8.3 Microservicio de Notificaciones y Microservicio de Pagos

#### Digital Ocean

DigitalOcean es una plataforma de nube diseñada para simplificar la infraestructura en la nube para desarrolladores y pequeñas empresas. Proporciona una variedad de servicios que incluyen servidores virtuales, almacenamiento, bases de datos gestionadas, y herramientas de desarrollo. Permite desplegar aplicaciones directamente desde repositorios como GitHub o GitLab, y gestionar el aprovisionamiento, escalado y la infraestructura de manera automática.

Los microservicios de notificaciones y pagos se alojaron en DigitalOcean. Es un servicio que proporciona infraestructura bajo demanda, con servidores virtuales conocidos como Droplets. Sin embargo, para simplificar el despliegue y la gestión de las aplicaciones, se

eligió utilizar App Platform, la opción de PaaS. Aunque este servicio es de pago, se aprovecharon los descuentos ofrecidos por GitHub Students, que proporciona créditos gratuitos durante un año para estudiantes. Esta oferta fue determinante en la elección de DigitalOcean, ya que permitía acceder a sus servicios de manera gratuita durante el primer año, lo que resultó muy beneficioso para el proyecto.

Para estos servicios, se utilizó Docker para la puesta en producción, a diferencia del frontend y el backend, que se utilizó github con *continuos deployment.*

### 7.8.4 Microservicio de Búsqueda

Inicialmente, el microservicio se ejecutó localmente en las computadoras de uno de los estudiantes. Se configuraron los puertos del router para facilitar la comunicación con los diferentes microservicios desplegados.

En un intento inicial, se exploró desplegar el servicio en Heroku, plataforma bien valorada para proyectos Python. Sin embargo, este esfuerzo no alcanzó los objetivos esperados debido a las numerosas dependencias del microservicio, como TensorFlow para aprendizaje automático, que complicaron considerablemente su implementación en producción.

Posteriormente, se optó por utilizar contenedores en el servicio para utilizar DigitalOcean y así aprovechar la experiencia previa con otros microservicios. A pesar de lograr la virtualización con éxito, la eficiencia no fue óptima y la velocidad de procesamiento en producción resultó notablemente inferior a la experiencia local del equipo.

### 7.8.5 Blue Green Deployment

Blue Green Deployment es una técnica de implementación de software que busca minimizar el tiempo de inactividad y el riesgo al desplegar nuevas versiones de aplicaciones. Esta estrategia se utiliza principalmente en entornos de producción para asegurarse de que las actualizaciones se realizan de manera fluida y con la menor interrupción posible para los usuarios finales.

En esta estrategia, el entorno azul es el entorno de producción actual que está en funcionamiento y que los usuarios utilizan. Esta es la versión estable de la aplicación que se encuentra en uso. El entorno verde es un entorno idéntico al azul donde se despliega la nueva versión de la aplicación, pero este entorno no está visible para los usuarios finales al inicio.

Una vez que la nueva versión de la aplicación se desplegó en el entorno verde, se realizan pruebas exhaustivas para asegurarse de que todo funciona correctamente y que no hay errores ni problemas. Si las pruebas en el entorno verde son satisfactorias, el tráfico de los usuarios se redirige del entorno azul al verde. Durante un período de tiempo, se monitorea el entorno verde para asegurar que la nueva versión esté en correcto funcionamiento en producción. Si se detecta algún problema crítico, se puede revertir rápidamente el tráfico de nuevo al entorno azul, lo que minimiza el tiempo de inactividad y los posibles impactos negativos para los usuarios.

Una vez que se ha confirmado que el entorno verde está en ejecución correctamente, el entorno azul puede ser desactivado o reutilizado para la próxima actualización.

Esta técnica permite que el equipo ponga en producción los servicios y teste su integridad en la nube sin afectar al producto en producción actual. En reiteradas ocasiones el comportamiento en producción era distinto al comportamiento del sistema en local. Esta estrategia fue el pilar para detectar las fallas antes de transitar a la siguiente versión del sistema.

La adopción de plataformas como Netlify, Google Cloud Run y DigitalOcean App Platform permitió un acceso ágil y rápido a la página web y sus microservicios. Estas plataformas proporcionaron integraciones sencillas, herramientas de automatización y gestión de contenedores, y créditos gratuitos para estudiantes, lo que facilitó un despliegue eficiente y económico. Esta estrategia permitió al equipo centrarse en el desarrollo y la mejora continua de la aplicación, con el propósito de optimizar recursos y simplificar la puesta en producción.

# Capítulo 8: Memorias

## 8.1 Cumplimiento de los objetivos

En esta sección, se revisan los objetivos planteados al inicio del proyecto y se evalúan si se han cumplido uno por uno. Se comienza con el objetivo global y luego se analiza los objetivos secundarios.

### 8.1.1 Objetivo principal:

***Desarrollar e implementar un sistema informático distribuido que simplifique la obtención de servicios y productos en línea.***

Se desarrolló e implementó un sistema distribuido que simplifica significativamente el proceso de obtención de servicios y productos. Se validó el sistema con los pasajes de Trenes Argentinos. Se facilitó a los usuarios la interacción con la plataforma y se aumentó la eficacia de obtención. Los usuarios destacaron la eficacia del sistema a la hora de conseguir pasaje.

### 8.1.2 Objetivos específicos:

**Implementar una arquitectura de microservicios, que permita garantizar la escalabilidad del sistema y disponga de capacidad de adaptación a nuevos productos y servicios, basado en la reutilización de componentes.**

Se implementó con éxito un conjunto de microservicios que garantizan la escalabilidad del sistema y su capacidad de adaptación a nuevos productos y servicios. Se basa en la reutilización de componentes.

Los diferentes upgrades mencionados anteriormente como las mejoras de validación en el frontend, y la transición de Java a Python son ejemplos concretos que permitieron escalar el

sistema. La incorporación de WhatsApp como método de comunicación es un ejemplo de adaptación que mejora notablemente la experiencia de usuario.

**Implementar la solución en la Empresa trenes Argentinos:**

Se logró un sistema automático aplicado al caso concreto de Trenes Argentinos. El usuario ingresa sus datos, estos se validan automáticamente y el pedido se procesa de manera eficiente. Los usuarios son notificados y reciben un plan de acción claro para la obtención del producto final. Permite que las personas que utilicen el producto se desliguen completamente del proceso de gestión de compra del boleto.

**Desarrollar un microservicio de búsqueda de pasajes que se adapte fácilmente al sistema.**

Se desarrolló, implementó con éxito un microservicio de búsqueda especializado en la comunicación con Trenes Argentinos. Se integra sin problemas en el sistema general y se aprovecha la arquitectura de microservicios previamente implementada.

**Realizar iteraciones del producto que satisfagan la demanda del mercado y posicionarse como principal opción en el mercado**

Se cumplieron los plazos establecidos. El equipo logró entregar el proyecto dentro del tiempo requerido propuesto por los estudiantes. Además, se logró satisfacer los tiempos del mercado y posicionar el sistema como una solución confiable a partir del lanzamiento de distintas iteraciones. Se satisficieron las expectativas de los demandantes de poder difundir el sistema entre la gente. Las ventas en los meses del verano se incrementaron al igual que la demanda, aunque no se pudo satisfacer toda la demanda de las personas que solicitaban pasajes. Desde diciembre a febrero solicitaron más de dos mil pasajes de los cuales se pudieron conseguir pasajes por lo menos a la mitad de las personas.

El equipo logró adaptarse a la demanda del mercado mientras mejoraba el sistema con clientes reales.
El sistema se posicionó exitosamente como una opción principal en el mercado para la obtención de pasajes. Se destacó por su eficiencia y confiabilidad. Es importante mencionar que, actualmente, no existe competencia significativa, lo que refuerza el posicionamiento del equipo como la opción preferida.

## 8.2 Planificación en relación a los tiempos reales

Inicialmente, la planificación consideraba que el inicio de la etapa de desarrollo iniciaría en Febrero. Sin embargo, debido a razones del mercado, se proyectó que el verano sería la época con mayor cantidad de ventas. Por lo tanto, se decidió iniciar el desarrollo antes de lo planificado. Además, el modelo de negocios de Trenes Argentinos solía liberar los pasajes con un mes de anticipación, pero para la temporada de verano, todos los pasajes fueron liberados el 1 de diciembre, lo que presentaba una mayor oportunidad de ventas. La idea principal fue planificar el desarrollo para que el sistema de búsqueda pudiera soportar una gran cantidad de consultas, anticipándose al aumento de la demanda. Una vez finalizado el mes de diciembre, con el objetivo de enfocarse en las ventas y el mantenimiento del servicio, se detuvo el desarrollo del sistema hasta marzo, cuando uno de los estudiantes regresó de su viaje.

Es de suma relevancia considerar que los cambios entre la planificación y el resultado real se debe a los plazos impuestos por el mercado. Cuando los demandantes advirtieron la importancia que significaba el aumento de la demanda que iba a haber en la temporada, la gestión por parte de los estudiantes cambió drásticamente, con el fin de conseguir un MVP que satisfaga esta necesidad.

En las siguientes páginas se muestra el diagrama de gantt planificado y el real, realizado una vez finalizado el proyecto.

### 8.2.1 Plazos

El principal plazo impuesto en este Gantt era cumplir con la alta demanda esperada para diciembre, ya que la mayoría de los pasajes se liberarían en esa fecha. Por ello, era esencial contar con una página web para tomar pedidos, un servidor funcional que soportara las operaciones básicas, y un microservicio de búsqueda optimizado para manejar múltiples búsquedas simultáneas. Además, aunque el microservicio de comunicación no fuese óptimo, debía estar operativo para esa fecha.

**Septiembre y Octubre**

- Definir análisis y diseño de la aplicación para mediados de octubre.
- Comienzo de desarrollo de página web y servidor.

**Noviembre**

- Desarrollo y despliegue del MVP de la página web.
- Desarrollo y despliegue de la primera iteración del servidor con el fin de realizar requerimientos esenciales junto a la página web.

- A mitades de noviembre iniciar con el desarrollo del microservicio de búsqueda con el fin de poder optimizar este microservicio.

**Diciembre**

- Desarrollo y despliegue de un servicio de búsqueda optimizado.
- Desarrollo y despliegue del microservicio de comunicación a través de WhatsApp.

**Enero**

- Mantenimiento del microservicio de comunicación para evitar caídas y soportar una mayor cantidad de mensajes.
- Mantenimiento general de la aplicación, incluyendo solución de bugs con el objetivo de incrementar ventas.

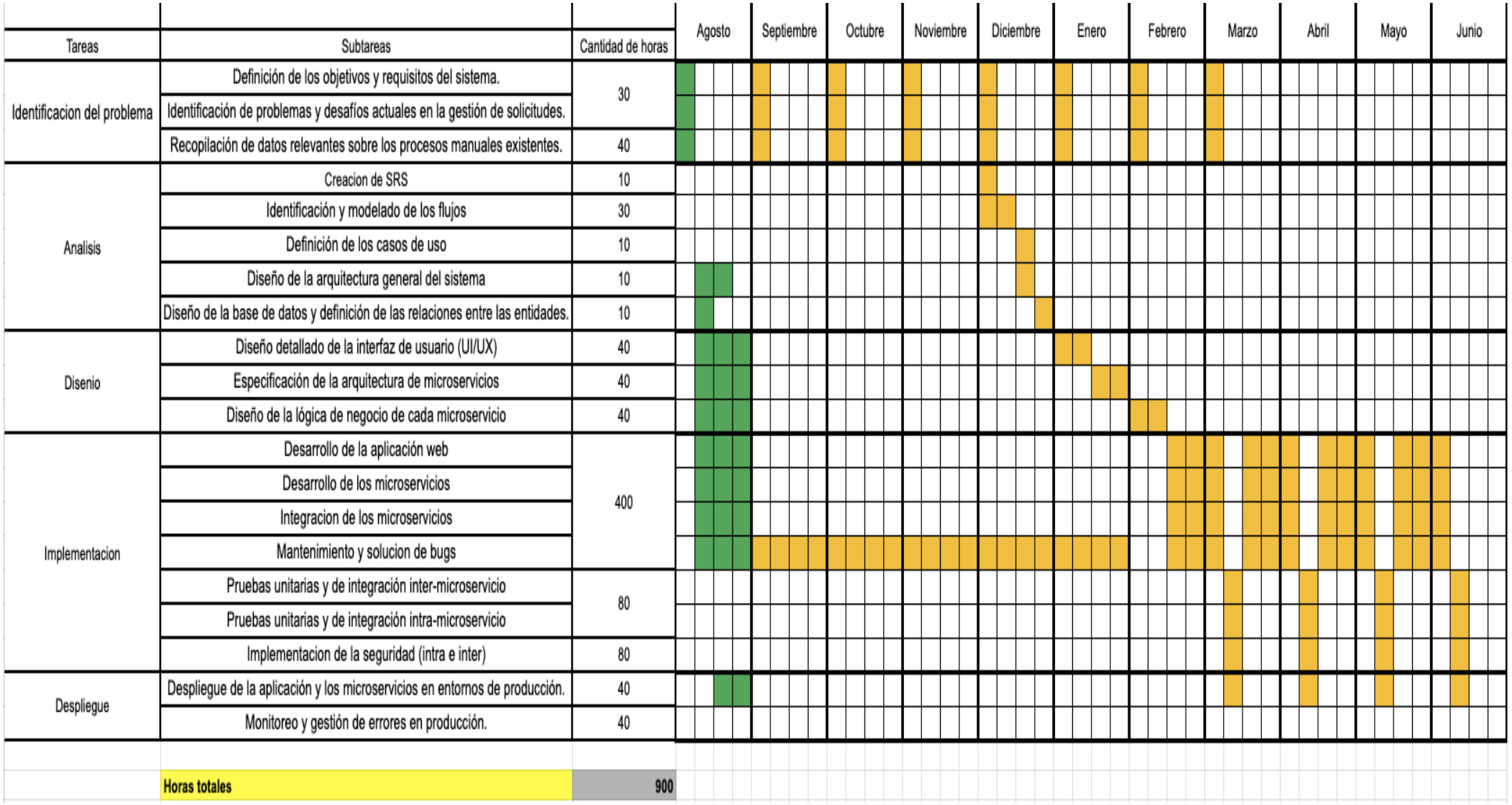
**Febrero - Marzo**

- Mantenimiento general de la aplicación, incluyendo solución de bugs con el objetivo de incrementar ventas.

**Abril - Junio**

- Finalización del servidor para automatizar todo el sistema, eliminando la necesidad de tareas manuales. Se aprovechó la disminución de la demanda para completar estas tareas.

***Diagrama de gantt planificado.***



| Tareas | Subtareas | Cantidad de horas |
|---|---|---|
| Identificacion del problema | Definición de los objetivos y requisitos del sistema. | 30 |
| | Identificación de problemas y desafíos actuales en la gestión de solicitudes. | |
| | Recopilación de datos relevantes sobre los procesos manuales existentes. | 40 |
| Analisis | Creacion de SRS | 10 |
| | Identificación y modelado de los flujos | 30 |
| | Definición de los casos de uso | 10 |
| | Diseño de la arquitectura general del sistema | 10 |
| | Diseño de la base de datos y definición de las relaciones entre las entidades. | 10 |
| Disenio | Diseño detallado de la interfaz de usuario (UI/UX) | 40 |
| | Especificación de la arquitectura de microservicios | 40 |
| | Diseño de la lógica de negocio de cada microservicio | 40 |
| Implementacion | Desarrollo de la aplicación web | 400 |
| | Desarrollo de los microservicios | |
| | Integracion de los microservicios | |
| | Mantenimiento y solucion de bugs | |
| | Pruebas unitarias y de integración inter-microservicio | 80 |
| | Pruebas unitarias y de integración intra-microservicio | |
| | Implementacion de la seguridad (intra e inter) | 80 |
| Despliegue | Despliegue de la aplicación y los microservicios en entornos de producción. | 40 |
| | Monitoreo y gestión de errores en producción. | 40 |
| | **Horas totales** | **900** |

***Diagrama de gantt real.***

| Tareas | Subtareas | Cantidad de horas |
|---|---|---|
| Identificacion del problema | Definición de los objetivos y requisitos del sistema. | 6,6 |
| | Identificación de problemas y desafíos actuales en la gestión de solicitudes. | |
| | Recopilación de datos relevantes sobre los procesos manuales existentes. | 3,3 |
| Analisis | Creacion de SRS | 6 |
| | Identificación y modelado de los flujos | 40 |
| | Definición de los casos de uso | 6 |
| | Diseño de la arquitectura general del sistema | 15 |
| | Diseño de la base de datos y definición de las relaciones entre las entidades. | 10 |
| Disenio | Diseño detallado de la interfaz de usuario (UI/UX) | 40 |
| | Especificación de la arquitectura de microservicios | 45 |
| | Diseño de la lógica de negocio de cada microservicio | 50 |
| Implementacion | Desarrollo de la aplicación web | 480 |
| | Desarrollo de los microservicios | |
| | Integracion de los microservicios | |
| | Mantenimiento y solucion de bugs | |
| | Pruebas unitarias y de integración inter-microservicio | 80 |
| | Pruebas unitarias y de integración intra-microservicio | |
| | Implementacion de la seguridad (intra e inter) | 20 |
| Despliegue | Despliegue de la aplicación y los microservicios en entornos de producción. | 45 |
| | Monitoreo y gestión de errores en producción. | 30 |
| | **Horas totales** | **876,9** |

### 8.2.2 Comparativa general

En este apartado se estudia la planificación inicial y las distintas dificultades que surgieron durante su ejecución. En un principio, se planteó realizar una planificación que tomaría 900 horas totales. En la práctica, se realizaron 877 horas totales, lo que significa que no se cumplió con exactitud el plazo. A pesar de ello, la diferencia es mínima, se estimó un 2,6% más del tiempo realizado. Por lo tanto, se puede concluir que el equipo cumplió en gran medida con los plazos de tiempo establecidos para el trabajo.

Es importante aclarar que las horas de gestión no fueron estimadas en la planificación inicial. Esto ocurrió debido a la inexperiencia del equipo. Si bien no afectó a los plazos del proyecto, no contabilizar las horas reales se traduce en incertidumbre en los costos del desarrollo del proyecto.

En el siguiente gráfico se muestran las horas planificadas vs. horas reales y su diferencia porcentual. Luego, se estudia con más detalle dónde hubo falencias a la hora de estimar los tiempos.

La diferencia porcentual negativa simboliza una sobre-estimación, mientras que un porcentaje positivo es una subestimación.

| ***Tareas*** | ***Horas planificadas (hrs)*** | ***Horas reales (hrs)*** | ***Diferencia porcentual (%)*** |
|---|---|---|---|
| ***Identificación del problema*** | *70* | *10* | *-85,71* |
| ***Análisis*** | *70* | *77* | *10* |
| ***Diseño*** | *120* | *135* | *12,5* |

| | | | |
|---|---|---|---|
| ***Implementación*** | *560* | *580* | *3,57* |
| ***Despliegue*** | *80* | *75* | *-6,25* |
| ***Total*** | ***900*** | ***877*** | ***-2,56*** |

#### 8.2.2.1 Identificación del problema

La identificación del problema se sobreestimó. Esta sobreestimación fue una medida de precaución, adoptada para garantizar que todos los posibles escenarios y riesgos estuvieran cubiertos. Sin embargo, la definición de los objetivos fue clara y concisa desde el comienzo del proyecto ya que el equipo poseía una idea clara del problema a resolver. El desafío central es recibir pedidos, ejecutar las búsquedas, conseguir el pasaje, notificar al usuario y que este siga una secuencia de acciones para finalizar el proceso. Los detalles específicos, como los requerimientos particulares, se desarrollaron durante la fase de análisis.

#### 8.2.2.2 Análisis

El análisis es una etapa del proyecto que se subestimó, aunque fue levemente acertada. En este espacio se determinó dominio y el alcance del trabajo. La definición del modelo de negocio y la especificación de requerimientos fueron las actividades que más tiempo requirieron. El tiempo real resultó ser un 10% mayor al estimado, con un desfase de 7 horas. La estimación no tuvo en cuenta que algunos requerimientos cambian a través del tiempo y necesitan de revisiones reiterativas. Por otra parte, los aspectos relevantes del modelo de negocio se propusieron en una etapa inicial y se mantuvieron.

#### 8.2.2.3 Diseño

La planificación del diseño se estimó en 120 horas totales. Durante su ejecución, fuimos certeros con los plazos, ya que al conocer de antemano lo que se quería lograr, se logró una estimación precisa. La ejecución llevó 135 horas, es decir, 15 horas adicionales. Los estudiantes fueron precisos a la hora de estimar ya que se tenía en cuenta el trabajo de

ingeniería inversa, necesario para entender el diseño de Trenes Argentinos. Además, se invirtió tiempo en revisar aspectos del diseño en consecuencia de las reuniones con la diseñadora, la revisión de las arquitecturas y la retroalimentación proporcionada por el usuario.

#### 8.2.2.4 Implementación

La implementación fue la etapa con más tiempo asignado en la estimación, y la planificación resultó ser precisa, con un desfase de 20 horas, equivalente al 3,57%. Los estudiantes, con experiencia trabajando juntos en equipo, conocían bien sus tiempos de desarrollo. Desde el principio, los estudiantes fueron muy precavidos al estimar la duración, se intentó sobreestimar la duración del mismo para poder llegar bien con los plazos establecidos. Aunque se tuvieron en cuenta los microservicios planificados, aún no contaban con un análisis y diseño concretos para estimar con mayor certeza el tiempo de implementación de cada uno. Sin embargo, a pesar de intentar sobreestimar inicialmente la cantidad de horas en esta etapa, se requirió más tiempo del planificado.

#### 8.2.2.5 Despliegue

La estimación inicial para las tareas de despliegue se fijó en 80 horas. Abarca tanto el despliegue de la aplicación en producción como el monitoreo de posibles errores posteriores. Desde el principio, el equipo fue consciente de su inexperiencia en el despliegue de aplicaciones, lo cual motivó una estimación generosa de tiempo. Esta previsión resultó ser una decisión prudente, ya que permitió completar el despliegue dentro del tiempo planificado. Afortunadamente, no ocurrieron errores significativos en la producción, lo que redujo considerablemente las horas necesarias para el monitoreo. Finalmente, el tiempo dedicado al despliegue se limitó a 75 horas.

#### 8.2.2.6 Documentación

No se tuvo en cuenta el tiempo de documentación. La documentación consta únicamente del desarrollo del informe una vez finalizado el proyecto. El informe se realizó como última instancia desde la finalización del proyecto, el cual se inició a fines de abril y se finalizó a principios de julio. Un total de 160 horas.

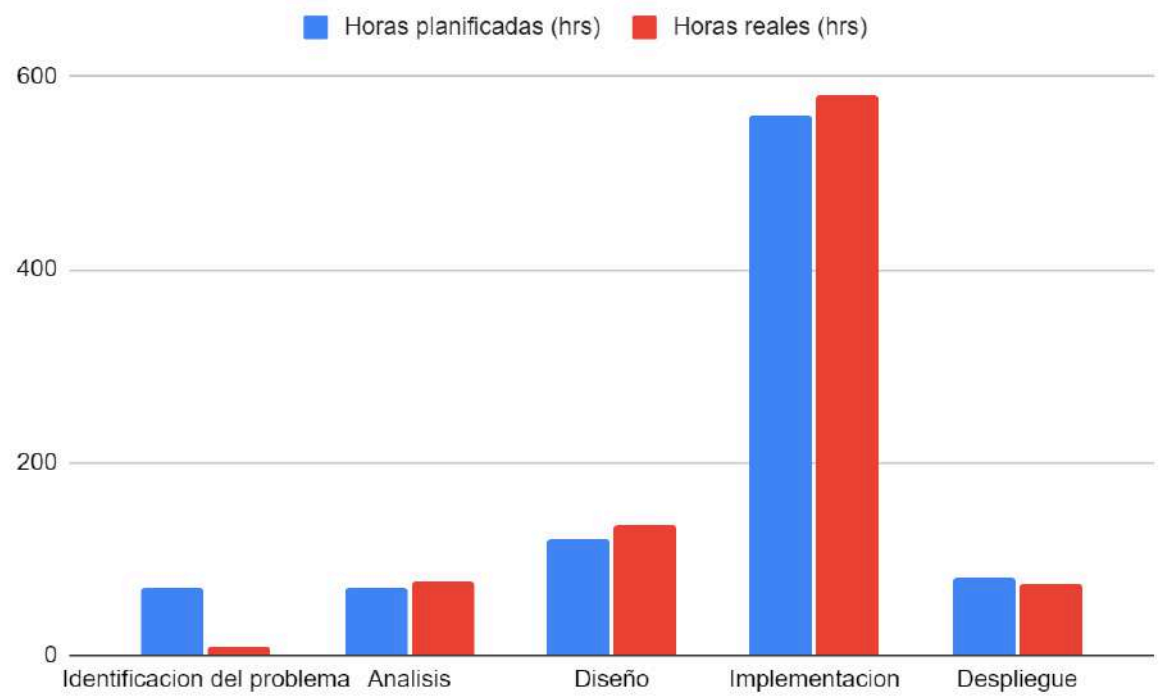
#### 8.2.2.7 Consideraciones finales

En general , las estimaciones fueron correctas y precisas debido a la experiencia que tenía el equipo trabajando en conjunto y la información previa que se disponía al haber partido de un MVP. El equipo contó con la disciplina necesaria para cumplir con los plazos establecidos al estar desarrollando un producto limitado por los tiempos del mercado.

Por otra parte, las horas de gestión no fueron contabilizadas porque eran decisiones tomadas en reuniones informales de los estudiantes. Por ejemplo, el permanente contacto vía whatsapp, o simples reuniones sociales sin foco principal en el proyecto.

Si este tiempo se hubiera contemplado, posiblemente el tiempo final habría sobrepasado las 900 horas.

#### 8.2.2.8 Gráfico de barras



| ***Tareas*** | ***Subtareas*** | ***Horas planificadas*** | ***Horas reales*** | ***Diferencia porcentual*** |
|---|---|---|---|---|
| *Identificación del problema* | *Definición de los objetivos y requisitos del sistema.* | *15* | *3,3* | *-77,78* |
| | *Identificación de problemas y desafíos actuales en la gestión de solicitudes.* | *15* | *3,3* | *-77,78* |
| | *Recopilación de datos relevantes sobre los procesos manuales existentes.* | *40* | *3,3* | *-91,67* |
| *Análisis* | *Creación de SRS* | *10* | *6* | *-40,00* |
| | *Identificación y modelado de los flujos* | *30* | *40* | *33,33* |
| | *Definición de los casos de uso* | *10* | *6* | *-40,00* |
| | *Diseño de la arquitectura general del sistema* | *10* | *15* | *50,00* |
| | *Diseño de la base de datos y definición de las relaciones entre las entidades.* | *10* | *10* | *0,00* |
| ***Diseño*** | *Diseño detallado de la interfaz de usuario (UI/UX)* | *40* | *40* | *0,00* |
| | *Especificación de la arquitectura de microservicios* | *40* | *45* | *12,50* |
| | *Diseño de la lógica de negocio de cada microservicio* | *40* | *50* | *25,00* |
| *Implementación* | *Desarrollo de la aplicación web* | *100* | *120* | *20,00* |
| | *Desarrollo de los microservicios* | *100* | *120* | *20,00* |
| | *Integración de los microservicios* | *100* | *120* | *20,00* |
| | *Mantenimiento y solución de bugs* | *100* | *120* | *20,00* |
| | *Pruebas unitarias y de integración inter-microservicio* | *40* | *40* | *0,00* |
| | *Pruebas unitarias y de integración* | *40* | *40* | *0,00* |

| | | | | |
|---|---|---|---|---|
| | *intra-microservicio* | | | |
| | *Implementación de la seguridad (intra e inter)* | *80* | *20* | *-75,00* |
| *Despliegue* | *Despliegue de la aplicación y los microservicios en entornos de producción.* | *40* | *45* | *12,50* |
| | *Monitoreo y gestión de errores en producción.* | *40* | *30* | *-25,00* |
| ***Total*** | | ***900*** | ***877*** | *-2,63* |

### 8.2.3 Desaciertos

Durante el desarrollo del proyecto y tras su finalización, el equipo planteó cuáles fueron los desafíos principales y dónde hubo problemas que se podrían haber abordado de otra manera. Esta reflexión permitió identificar dos desaciertos importantes que merecen ser mencionados y analizados con mayor profundidad.

Es muy importante destacar que, si bien todas las funcionalidades previstas se implementaron en algún momento, el equipo se vio obligado a priorizar qué elementos se debían desarrollar primero para poder lanzar el producto dentro del plazo requerido. Esto significó que algunas implementaciones, aunque necesarias y previstas desde el inicio, tuvieron que ser pospuestas porque no había tiempo suficiente o porque era necesario priorizar otras etapas del desarrollo que eran más críticas para el lanzamiento.

Este proceso también permitió descubrir una fortaleza clave del equipo: la capacidad de gestionar eficazmente el proyecto con un enfoque centrado en las necesidades del cliente. La habilidad de reorganizar el plan de desarrollo de manera flexible para garantizar que el negocio del cliente prospere o tenga éxito es una competencia fundamental que un ingeniero en informática debe poseer. Este tipo de gestión no solo aseguró que el producto llegara a tiempo al mercado, sino que también demostró la importancia de adaptar las soluciones

tecnológicas a las demandas reales del cliente, un aspecto crucial en cualquier proyecto exitoso.

#### 8.2.3.1 Testing

El primero de estos desaciertos se relaciona con las pruebas de integración entre los distintos microservicios. A pesar de que se implementaron pruebas unitarias y se realizaron tests en diversas etapas del desarrollo, la urgencia por implementar los cambios lo antes posible para que el usuario pudiera beneficiarse de las nuevas funcionalidades llevó a quitar de las pruebas de integración del alcance del proyecto. Esta decisión, aunque comprensible en el contexto de querer entregar valor rápidamente, resultó en fallas de consistencia en la comunicación entre los microservicios. Se manifestaron en problemas de sincronización y errores de funcionalidad que podrían haberse evitado con una integración continua y pruebas más rigurosas a lo largo del desarrollo.

Esta fue una decisión de gestión consciente por parte del equipo. Aunque los estudiantes eran plenamente conscientes del riesgo que suponía la puesta en producción sin realizar todas las pruebas necesarias, como emprendedores sabían que debían priorizar la rapidez para satisfacer la demanda inmediata del mercado. Decidieron que entregar una solución funcional, aunque imperfecta, era crucial para mantener la relevancia del producto y aprovechar la oportunidad comercial a tiempo. Esta elección estratégica fue clave para el éxito del negocio, ya que permitió que la solución llegará al mercado en el momento oportuno.

#### 8.2.3.2 Google Sheet

Al iniciar el desarrollo del proyecto, se optó por utilizar Google Sheets como consecuencia de emplear Google Forms. Google Sheets brindaba cierta comodidad durante el desarrollo, especialmente en la visualización de la información, lo que llevó a que inicialmente no se considerará un cambio inmediato. Es importante destacar que el uso de Google Sheets en producción aumentaba los tiempos de gestión del sistema, dado que el sistema no era automático. A través de AppScripts se lograron algunas mejoras, como ordenar las búsquedas por estado, y obtener el JSON necesario para enviar los pedidos al microservicio de búsqueda.

En un principio, se prioriza el desarrollo de nuevas funcionalidades sobre la implementación de una base de datos robusta y un sistema automático. Esta decisión llevó a construir una arquitectura alrededor de Google Sheets, que funcionaba y era eficaz en ese momento. Esta solución, aunque temporalmente adecuada, no era sostenible a largo plazo. Con el paso de las iteraciones y la incorporación de más usuarios, se hizo evidente la necesidad de migrar a una base de datos todo el sistema, con el fin de automatizar y disminuir los tiempos de gestión del sistema.

Sin embargo, debido a la alta demanda de consultas de pasajes y la constante actividad del sistema, el equipo decidió postergar la migración. Realizar la migración en ese momento podría haber causado problemas en la producción. Finalmente, cuando la demanda disminuyó, se llevó a cabo la migración, lo que permitió que el sistema funcionara de manera completamente autónoma.

Lo ideal hubiera sido iniciar el desarrollo con una base de datos desde el principio, a fin de asegurar la automatización del sistema y evitar la dependencia de Google Sheets. Aunque la transición no fue tan compleja, el equipo estaba influenciado por la falta de tiempo y el temor a automatizar todo desde el inicio. Esta experiencia dejó como enseñanza la importancia de planificar adecuadamente la arquitectura del sistema desde el principio. La eficiencia inmediata como la escalabilidad futura deben ser consideradas en una etapa temprana.

## 8.3 Proyecto comercial

El proyecto tiene como particularidad que se utilizó de manera real durante su desarrollo. Salir al mercado en una etapa temprana de vida del proyecto tuvo ventajas y desventajas.

Ventajas

- Retroalimentación continua y temprana por parte de los usuarios. Permite realizar mejoras basadas en experiencias y necesidades reales de los usuarios.

- Lanzar una idea innovadora permitió establecer una base sólida de usuarios leales, y propuso una ventaja competitiva.

- Permitió validar la viabilidad del proyecto. Se verificó la existencia de una demanda real para el servicio

Desventajas

- La utilización de un servicio con desarrollo incompleto tuvo problemas de calidad. Las fallas estuvieron presentes durante todo el ciclo de vida del proyecto.

- Una mala experiencia de un cliente condiciona la reputación de la empresa. Un usuario cuando utiliza un servicio o producto, espera que funcione bien. No tiene el mismo peso cumplir, que no hacerlo. En etapas tempranas donde existían muchas fallas, algunos usuarios se llevaron una mala impresión.

- El proyecto se convirtió en un negocio, y comenzó a requerir tiempo que no estaba previsto. Por ejemplo, la administración y la comunicación con los clientes.

### 8.3.1 Te Chiflo

"Te Chiflo" es el nombre comercial del producto enfocado en Trenes Argentinos. Los primeros usuarios fueron principalmente estudiantes universitarios residentes en Buenos Aires y La Plata.

Durante el desarrollo del sistema, se distinguieron dos etapas clave: antes y después de las vacaciones de verano. El primer objetivo del equipo era maximizar las ventas durante la época de verano con el fin de anticipar el aumento significativo en la demanda debido al atractivo turístico de Mar del Plata.

Para alcanzar este objetivo, era esencial cumplir con dos metas primordiales: primero, tener un sistema funcional y escalable que pudiera soportar un alto volumen de usuarios. Las decisiones se orientan a evitar puntos de falla debido a la gran cantidad de usuarios. La

prioridad era la capacidad del sistema para manejar este volumen. Como consecuencia otras tareas fueron pospuestas, como la automatización de gestiones o la autonomía del sistema, a un segundo plano.

#### 8.3.1.1 Inicios de Te Chiflo

Es importante recordar cómo se inició en un principio, los pedidos eran recibidos a través de los Whatsapp personales de los demandantes. Luego la información era escrita manualmente a una hoja de cálculo. Con el paso del tiempo, la difusión de los números personales y el tiempo de trabajo manual derivaron en:

- Utilización de Google Forms para tomar los pedidos que eran insertados directamente en la hoja de cálculo
- Creación de canales oficiales de comunicación de Te Chiflo, principalmente Whatsapp Business. Se utilizaron respuestas automáticas explicativas del servicio, y un link al formulario de Google Form. A pesar de implementar WhatsApp como medio de soporte al cliente, la comunicación vía Email era el medio de notificación del sistema.

La implementación de estas herramientas permitió que los estudiantes inviertan tiempo en el desarrollo. Sin embargo, la validación de datos era aún un proceso manual.
Si alguno de los datos era incorrecto, era necesario comunicarse manualmente con la persona y comentarle al respecto. La validación y posterior comunicación con el cliente era la tarea que más tiempo consumía.

#### 8.3.1.2 Etapa previa a temporada de verano

La primera decisión que tomó el equipo fue el desarrollo de una página web y el servidor este desarrollo fue con el objetivo de poder validar los datos que ingresaba el usuario (detalles adicionales en la iteración 1 de la página web). Esta plataforma no solo redujo la carga administrativa, sino que también aseguró que los pedidos fueran válidos para el sistema y estar preparado para el aumento previsto en la demanda durante el verano.

#### 8.3.1.3 Desarrollo de front y servicio de comunicación

El siguiente paso fue mejorar el microservicio de comunicación. El equipo sabía que implementar WhatsApp como medio principal de comunicación aumentaría las ventas, ya que es el medio más utilizado en Argentina. Aunque la tarea de análisis y diseño de WhatsApp fue ardua, finalmente se logró incorporar Whatsapp como otro medio de notificación. Este cambio se reflejó en un aumento notable de las ventas durante noviembre.

Anteriormente, los usuarios eran notificados vía email para finalizar la compra de sus pasajes, pero muchos no revisaban sus correos. Con WhatsApp, los usuarios recibían sus pasajes al instante y eliminó la necesidad de notificaciones adicionales. Las ventas aumentaron significativamente. Sin embargo, el primer lanzamiento del WhatsApp no soportaba un gran número de usuarios simultáneamente, lo que podía afectar la estabilidad de la aplicación. A pesar de esto, se implementaron mejoras continuas para soportar una mayor cantidad de usuarios y se obtuvo un WhatsApp escalable recién a finales de febrero.

#### 8.3.1.4 Desarrollo del servicio de búsqueda

Paralelamente, se trabajó en el servicio de búsqueda. El MVP inicial no permitía muchas búsquedas consecutivas, lo que limitaba su capacidad. A partir de la información recolectada, se creó un microservicio de búsqueda basado en Python que aumentó la eficacia y permitió una mayor cantidad de búsquedas concurrentes. En enero y con una gran demanda, el sistema alcanzó a buscar más de 600 pasajes simultáneamente, aunque la tasa de éxito disminuyó debido a la alta demanda. Aun así, la cantidad de pasajes vendidos fue récord.

#### 8.3.1.5 Gestión y ampliación del equipo

Durante este periodo, uno de los estudiantes emprendió un viaje de cuatro meses al exterior. Como se anticipó, las consultas y pedidos aumentaron exponencialmente en verano. Dado que el sistema aún no era autónomo, y se requería tareas administrativas diarias, y por la futura ausencia de un miembro del equipo, se decidió incorporar un empleado para ayudar con las tareas administrativas y la comunicación con los clientes. Las tareas de desarrollo se detuvieron hasta el retorno del estudiante, pero el producto logró captar la alta demanda de la temporada y cumplió con los objetivos establecidos.

La siguiente imagen permite visualizar cómo a partir de diciembre las ventas incrementaron notablemente. A partir de febrero las métricas comenzaron a bajar debido al fin de la

temporada de verano. Hasta abril el producto fue utilizado con normalidad.



**Etapa post verano**

El producto se encontraba en el mejor momento en cuanto a desarrollo. El equipo había mitigado las deudas técnicas y se logró la automatización del proceso. El trabajo administrativo se vio reducido drásticamente, al punto de únicamente responder las consultas de los usuarios. Aunque el producto funcionara mejor que nunca, los usuarios dejaron de pedir pasaje. Debido a las políticas del gobierno de recorte del gasto público Trenes Argentinos sufrió cambios estructurales, entre ellos, una suba significativa en el precio de los pasajes.

El cambio de precio generó un cambio en la demanda de los pasajes y en la disponibilidad de los mismos. Ya no existía necesidad de encargar la búsqueda a través del servicio y pagar una comisión a cambio. Los usuarios podían sacar el pasaje directamente por la web de Trenes Argentinos, ya que había una constante disponibilidad para casi todos los días. Te Chiflo comenzó a ser utilizado únicamente para los días de mucha demanda, como los fines de semana largos.

### 8.3.2 Análisis

Análisis de los pasajes vendidos en *Te Chiflo* hasta julio

**Usuarios Totales**

- *Te Chiflo* cuenta con un total de **6000 usuarios** hasta julio de 2024.

**Pasajes Vendidos**

- Se encargaron un total de **10,000 pasajes**.
- Se vendieron un total de **2892 pasajes.**

**Eficacia del Sistema**

- Con 2892 pasajes vendidos de un total de , la eficacia del sistema es del **29%.** Esto es una eficacia sumamente alta considerando la dificultad que existía a la hora de conseguir pasaje de tren. Además, los pasajes deben ser devueltos para que salieran nuevamente a la venta. El adecuado uso de *Te Chiflo* y con anticipación a la hora de de encargar los pasajes garantiza una efectividad casi del 100% a la hora de conseguir pasaje.

### 8.3.3 Economía

El equipo tuvo que involucrarse en asuntos económicos y fiscales para gestionar adecuadamente los cobros y optimizar las ganancias. Utilizaron la plataforma de pago MercadoPago, que aplicaba una comisión del 8% a cada transacción. Para mantener los costos operativos bajo control y estar exentos de impuestos dentro de MercadoPago, los integrantes del equipo se registraron como monotributistas. De no haberlo hecho, los impuestos habrían sido del 20%.

### 8.3.4 Impacto social

El sistema aborda una problemática común en la sociedad al ofrecer una solución económica para viajar en tren. La diferencia de costos entre el tren y otros medios de transporte era altamente significativa. El pasaje en tren costaba hasta cuatro veces menos, o incluso más, en comparación con otros transportes. Inicialmente, el viaje en tren costaba

$2500, mientras que el pasaje en colectivo costaba $15000. Muchas personas necesitan viajar reiteradas veces al mes, y el servicio que los emprendedores brindaron permitió que pudieran hacerlo a un costo mucho menor.

El rango etario de los usuarios era variado. Los jóvenes estudiantes que no eran de Buenos Aires utilizaron el servicio principalmente para volver a ver a sus familiares y allegados. Los adultos lo usaban generalmente para realizar trámites, pero también para visitar a sus seres queridos. La gente de la tercera edad lo utilizaba tanto por temas de salud como para visitar a familiares y amigos, y también con fines turísticos.

En el verano grupos numerosos se contactaron con el equipo con una anticipación hasta de dos meses para obtener los pasajes para enero o febrero.

El hecho de que mucha gente de edad avanzada utilizara el servicio habla muy bien del producto logrado. A pesar de que suelen tener problemas para el uso de la tecnología, conformaron aproximadamente el 35% de los usuarios activos.

Los usuarios se mostraron realmente agradecidos con el sistema. Muchas personas necesitan viajar varias veces al mes y Te Chiflo es una solución para ellos durante mucho tiempo. En un contexto donde la economía no es estable y a las personas les cuesta llegar a fin de mes, poder contribuir a la sociedad a través de esta solución es algo que llena de orgullo al equipo.

### 8.3.5 Difusión, Marketing y comunicación

La principal técnica de publicidad del producto fue el “boca a boca”. Los usuarios, por sí solos, difundieron el producto entre su círculo social. Al equipo le llamó mucho la atención que sin hacer publicidad hasta diciembre de 2023 el producto se difundiera entre tantas personas y cómo llegó a alcanzar a cientos. Esto hizo evidente que continuar la mejora del servicio sería la mejor manera de llegar a más personas. Debido a la creciente demanda y a que los usuarios se comunicaban de manera personal con el equipo, se hizo necesario centralizar las consultas en un único WhatsApp.

Los estudiantes utilizaron herramientas de marketing para visualizar aún más el producto en el mercado. Se realizaron campañas publicitarias a través de las redes sociales.

A través de herramientas como Instagram Reels y Posts, los estudiantes y la diseñadora crearon contenido publicitario y lo impulsaron con campanas de Facebook Ads.

Estas campañas permiten describir el perfil de cliente que se busca. Los resultados dependen de qué tan acertados sean los filtros descriptivos que utilicen. Los filtros utilizados fueron de edad y ubicación.

La comunicación es otro aspecto importante en un proyecto comercial. El equipo recibía consultas de nuevos clientes a diario por Whatsapp, y se implementaron estrategias de respuestas automáticas. Los textos fueron pensados con detenimiento, con el fin de explicar todo el servicio en detalle, y minimizar las dudas en el lector.

El aprendizaje sobre la comunicación fue clave durante todo el desarrollo del proyecto. Esta competencia resultó esencial tanto en el rol de emprendedores como en el de desarrolladores. Como emprendedores, el éxito radicó en la capacidad de explicar un proceso de compra relativamente complejo, que generaba muchas preguntas. La implementación de respuestas automáticas, que fue identificada como una necesidad desde el rol de emprendedores para mejorar la gestión del negocio, se llevó a cabo gracias a las herramientas técnicas desarrolladas por los estudiantes en su rol de desarrolladores. Esto no solo redujo significativamente las consultas, sino que también optimizó la experiencia del cliente y mejoró la eficiencia operativa. Es muy importante saber comunicarle al cliente de la manera correcta para que comprenda cómo es que funciona el producto. Además también es importante conocer qué es lo quiere y que capta la atención del cliente con el fin de poder tener una mayor repercusión dentro del mercado. Razón por la cual se hizo énfasis en la imagen del producto para poder transmitir una solución que sea confiable y eficiente.

Asimismo, la comunicación interna entre los estudiantes en su rol de desarrolladores fue vital. Lograr que el equipo estuviera alineado y tomara decisiones en función de los plazos del mercado fue un desafío. Una buena comunicación no solo facilitó la planificación, sino que también impulsó el desarrollo eficiente del proyecto, permitiendo cumplir con los objetivos establecidos.

### 8.3.6 Situación actual de Trenes Argentinos

El sistema funciona en servicios de alta demanda y poca oferta, como ocurría con los Trenes Argentinos. El contexto político a fines de 2023 estaba afectado por las elecciones, y tras la victoria de Javier Milei, su campaña se centró en recortar el gasto público y reformar las empresas estatales. Esto impactó fuertemente a la empresa ferroviaria. La primera medida fue un aumento gradual en el precio de los boletos, que pasó de $6.000 a $30.000 entre marzo y mayo de 2024. El tren dejó de ser tan barato en comparación con el pasaje de colectivo. Actualmente, el pasaje de colectivo entre Buenos Aires y Mar del Plata cuesta alrededor de $30.000.

Este incremento de precios provocó un cambio significativo: la gente dejó de usar el tren como medio de transporte y los trenes que antes siempre viajaban completos, ahora lo hacen con menor ocupación. En la página web propia de Trenes Argentinos se pueden obtener boletos rápidamente, excepto en fechas concurridas como los fines de semana largos. En esas ocasiones, el sistema es util para obtener boletos, aunque debido al aumento de la oferta, su uso diario ha disminuido notablemente.

## 8.4 Trabajo a futuro

Uno de los principales trabajos a futuro es la integración de funcionalidades avanzadas para los administradores directamente desde la interfaz web. Actualmente, la gestión administrativa, incluida la supervisión del producto que se otorga, se realiza a través de herramientas como Postman o el administrador de la base de datos NoSQLBooster. Si bien estas herramientas son eficaces, trasladar estas funcionalidades a la interfaz web proporcionará mayor eficiencia operativa y facilidad de uso.

La creación de un perfil para administradores podría incluir una serie de herramientas y paneles de control que permitan una gestión completa y eficiente del sistema. Entre las funcionalidades propuestas se encuentran:

- **Vista General de Pasajes**: Un panel donde los administradores pueden ver las solicitudes de los usuarios, con opciones de filtrado y búsqueda avanzada.
- **Gestión de Pasajes**: Funcionalidades para dar de baja, modificar o actualizar el estado de los pasajes directamente desde la web.

- **Estadísticas y Reportes**: Herramientas para generar informes y estadísticas sobre la venta de pasajes que permita identificar tendencias y áreas de mejora.

La integración de funcionalidades avanzadas para administradores directamente en la interfaz web representa un paso fundamental en la evolución y mejora del producto para poder optimizar las operaciones internas.

Desde el punto de vista comercial, uno de los objetivos es que *Te Chiflo* se mantenga como la opción principal para la compra de pasajes de tren, especialmente durante los períodos de alta demanda, como el verano. Para ello, será necesario llevar a cabo un mantenimiento constante de la plataforma que asegure su fiabilidad y rendimiento.

Este mantenimiento incluirá:

- **Actualización y optimización continua** de los microservicios, especialmente en la infraestructura de notificaciones y en el servicio de búsqueda, para manejar eficazmente un mayor volumen de usuarios y solicitudes.
- **Monitoreo del rendimiento** para identificar y resolver rápidamente cualquier posible fallo en la escalabilidad, asegurando que el sistema funcione sin interrupciones incluso durante los picos de demanda.

Si *Te Chiflo* se sigue optimizando y adaptando a las necesidades del mercado, podrá seguir siendo la opción preferida cuando las personas necesiten conseguir pasajes de tren, asegurando una experiencia confiable y eficiente.

## 8.5 Conclusión

El objetivo principal de este trabajo, a pesar de centrarse en el caso específico de Trenes Argentinos, era la construcción de un sistema que facilitara la obtención de un servicio o producto, con la flexibilidad necesaria para adaptarse a diferentes contextos. Este objetivo se cumplió con creces, evidenciado por el uso del sistema en un emprendimiento real por parte de los propios estudiantes, lo que demuestra su éxito y abre la posibilidad de mejorarlo continuamente para alcanzar su máximo potencial. Además, esta experiencia emociona a los estudiantes por la perspectiva de aplicar el sistema en otros productos con alta demanda y poca oferta.

Durante el desarrollo del proyecto, se generó un profundo sentido de pertenencia y compromiso entre los estudiantes. Más allá de cumplir con los requisitos técnicos, se involucraron emocionalmente en cada fase del proceso. Este compromiso resultó fundamental para afrontar los desafíos y perseguir constantes mejoras en cada iteración del proyecto. Esta dinámica creó una experiencia enriquecedora, equiparable a la de construir una empresa o formar un equipo. El sentido de pertenencia no se logra de la noche a la mañana, sino que es fruto del compromiso sostenido del equipo, del trabajo en conjunto durante meses con el objetivo de ofrecer lo mejor en cada etapa del proyecto.

Este trabajo tiene una doble faceta, ya que no solo fuimos los desarrolladores del sistema, sino también sus administradores. Nuestra tarea no terminó con las horas de desarrollo contabilizables, sino que se extendió mucho más allá. Existe un trabajo invisible, que va más allá de las horas dedicadas al código: estar disponibles para las personas desde temprano, participar en reuniones diarias, gestionar imprevistos con los clientes, discutir sobre posibles mejoras y ayudar a los usuarios a comprender el sistema. Este conjunto de cosas sobresalen del desarrollo de software, que se consiguen al enfrentar el problema en carne propia y vivirlo en el día. El hecho del que producto funcionara y tener clientes que utilizaban el servicio constantemente género el compromiso de querer siempre dar el mejor servicio posible, esto tuvo como consecuencia el aprendizaje continuo en cuanto a la gestión de un producto hacia el público.

# Glosario

**REST**

*REST* (Transferencia de Estado Representacional) es un estilo arquitectónico utilizado en el diseño de sistemas de software distribuidos. Se basa en un conjunto de principios que buscan crear servicios web escalables, flexibles y de fácil mantenimiento.

En *REST*, todo se considera un recurso, que puede ser tanto una entidad física como abstracta, y cada recurso se identifica mediante un *URI* (Identificador de Recurso Uniforme). Las operaciones básicas sobre estos recursos siguen el modelo *CRUD* (Crear, Leer, Actualizar, Eliminar), Se asocian a los métodos *HTTP: POST* para crear, *GET* para leer, *PUT* o *PATCH* para actualizar, y *DELETE* para eliminar.

**AppScript**

Entorno de desarrollo en la nube proporcionado por Google, permite a los usuarios escribir scripts en JavaScript para automatizar tareas a través de los productos de Google, como Google Sheets, Google Docs y Google Drive. Es muy útil para crear aplicaciones personalizadas que interactúan con los servicios de Google.

**Basic Auth**

Método simple para implementar autenticación en una aplicación web. Se basa en el uso de un nombre de usuario y una contraseña codificados en Base64, que se envían en el encabezado de una solicitud HTTP. Aunque fácil de implementar, no es seguro a menos que se use sobre HTTPS, ya que la información de autenticación puede ser fácilmente interceptada.

**CAPTCHA**

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) es una herramienta de seguridad utilizada para distinguir entre humanos y bots en internet.

**DAO**

DAO (Data Access Object) es un patrón de diseño que proporciona una abstracción para acceder a una base de datos. Separa la lógica de negocio de la lógica de acceso a datos,facilita el mantenimiento y la evolución del código. Los DAO permiten a los desarrolladores cambiar la base de datos subyacente sin modificar el resto de la aplicación.

**Lenguaje Interpretado**

Un lenguaje interpretado es aquel en el que el código fuente se ejecuta directamente por un intérprete sin necesidad de ser compilado previamente en código máquina. Ejemplos de lenguajes interpretados incluyen Python, JavaScript y Ruby. La interpretación puede hacer que la ejecución sea más lenta en comparación con los lenguajes compilados.

**Middleware**

Software que actúa como intermediario entre diferentes aplicaciones o servicios para facilitar la comunicación y gestión de datos. Se usa comúnmente en sistemas distribuidos y aplicaciones web para realizar tareas como autenticación, registro de actividades, y gestión de sesiones.
Ejemplos de middleware: servidores web, servidores de aplicaciones, y sistemas de mensajería.

**Mutex**

Primitiva de sincronización usada en programación concurrente para evitar que múltiples threads accedan a un recurso compartido al mismo tiempo. Solo un thread puede poseer el mutex a la vez  para garantizar la exclusión mutua.

**NoSQLBooster**

Herramienta de desarrollo y administración para bases de datos MongoDB. Proporciona un entorno gráfico que facilita la creación de consultas, visualización de datos, y gestión de la base de datos. Es compatible con JavaScript y ofrece características avanzadas como la validación de consultas, autocompletado, y depuración de código.

**ORM**

ORM (Object-Relational Mapping) es una técnica de programación que permite a los desarrolladores interactuar con una base de datos relacional mediante un modelo orientado a objetos. Los ORMs convierten datos entre sistemas incompatibles (objetos en lenguajes de programación y tablas en bases de datos) y facilitan operaciones CRUD (Crear, Leer, Actualizar, Borrar).

**Singleton**

Patrón de diseño que restringe la instanciación de una clase a un único objeto. Es útil cuando se necesita exactamente una instancia de una clase para coordinar acciones en todo el sistema.

**SOLID**

Acrónimo que representa cinco principios de diseño orientado a objetos que promueven la escritura de código más claro, flexible y mantenible. Los principios son:

S: Single Responsibility Principle (Principio de responsabilidad única)
O: Open/Closed Principle (Principio de abierto/cerrado)
L: Liskov Substitution Principle (Principio de sustitución de Liskov)
I: Interface Segregation Principle (Principio de segregación de la interfaz)
D: Dependency Inversion Principle (Principio de inversión de dependencias)

**Stack de Tecnologias**

Conjunto de tecnologías usadas juntas para crear una aplicación completa. Un ejemplo es el stack MEAN (MongoDB, Express.js, Angular, Node.js) para desarrollo web.

**Servicio Web( WebService):**

Aplicación que interactúa con otras aplicaciones a través de una red, se utilizan estándares y protocolos como HTTP, XML, SOAP, y REST. Los servicios web permiten a diferentes sistemas interactuar entre sí, intercambiar datos y realizar operaciones remotas sin importar las plataformas o lenguajes de programación subyacentes.

**URL**

URL (Uniform Resource Locator) es una dirección que especifica la ubicación de un recurso en Internet y el mecanismo para recuperarlo.

***FOMO***

Sus siglas en inglés: *Fear of missing out*

Sentimiento de preocupación por la posibilidad de perderse eventos interesantes.

# Manual de administrador

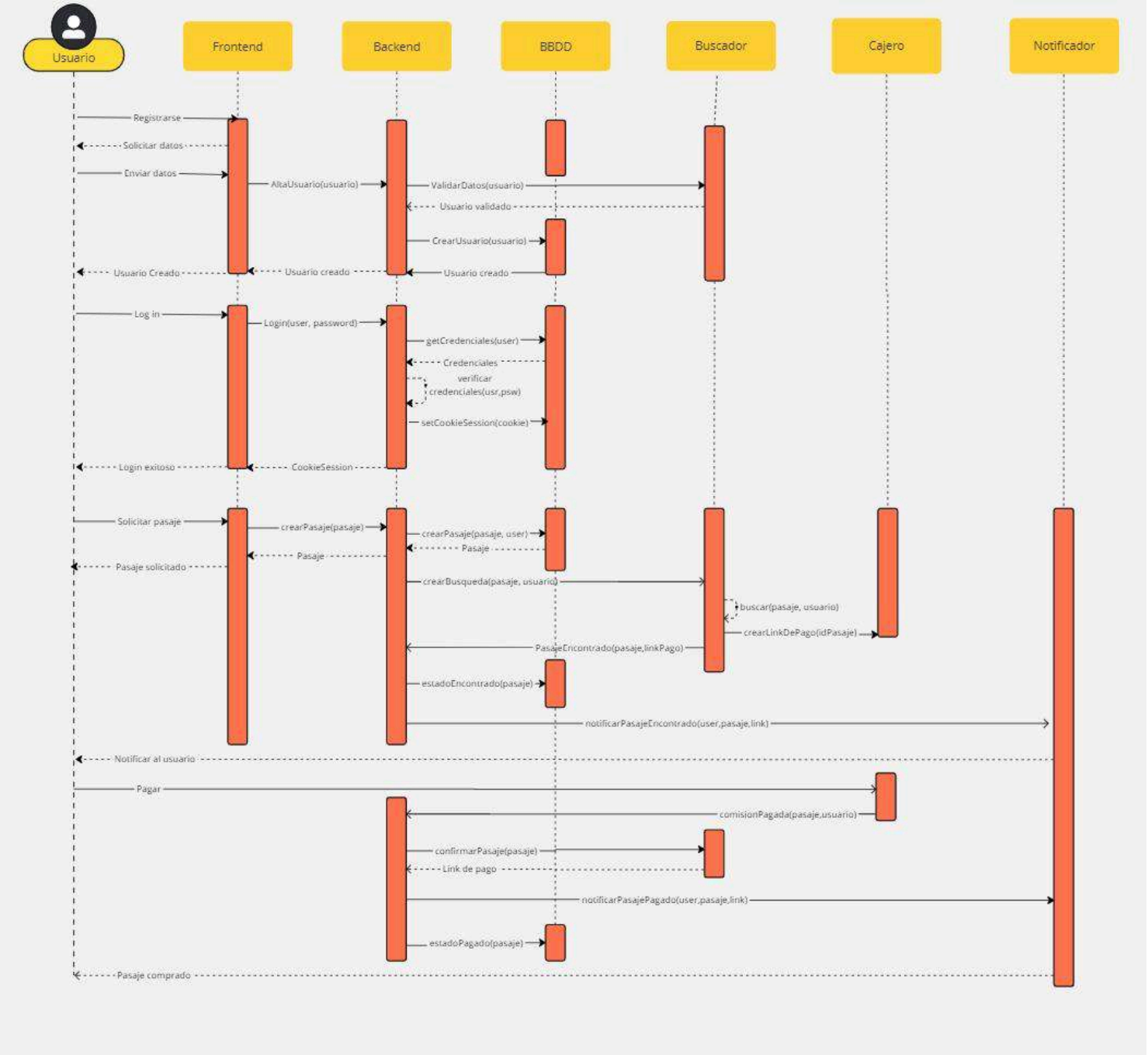
Este manual fue creado para proporcionar una guía completa sobre el uso y la administración del sistema. Dirigido a administradores y personal técnico, su propósito es ofrecer una referencia detallada y práctica sobre los distintos procesos, funcionalidades y microservicios integrados en la plataforma.

El manual abarca desde el registro de usuarios, procesos de autenticación, hasta la solicitud y gestión de pasajes, brindando una visión clara de cómo interactúan los diversos componentes del sistema.

Este documento es fundamental para asegurar que los administradores puedan mantener y optimizar el funcionamiento del sistema, resolver incidencias y garantizar una experiencia de usuario fluida y eficiente.

# 1. Diagrama de Secuencia

A continuación se detalla, desde un punto de vista técnico, el flujo de trabajo del sistema y la interacción entre los diferentes microservicios a través de un diagrama de secuencia.



**Registro del usuario**

El primer paso es el registro del usuario en la página web. Este proceso incluye los siguientes pasos:

1. El usuario ingresa su correo electrónico y contraseña en la interfaz web.
2. Estos datos se envían al servidor a través de una solicitud HTTP.
3. El servidor valida los datos del usuario requeridos por Trenes Argentinos. Lo hace a través del sistema de búsqueda.
4. Si los datos son correctos, se crea un nuevo registro de usuario en la base de datos.
5. Finalmente, el usuario recibe una notificación de confirmación en la web de registro exitoso.

**Inicio de sesión del usuario**

El siguiente paso es el inicio de sesión del usuario:

1. El usuario ingresa su correo electrónico y contraseña.
2. Estos datos se envían al servidor.
3. El servidor verifica que las credenciales sean correctas.
4. Si son correctas, se genera un ID de sesión (sessionID) que se guarda en la base de datos.
5. Este sessionID se envía al usuario y se almacena en una cookie en su navegador.
6. La cookie se utilizará para identificar al usuario y mantener su sesión activa durante sus acciones en la web.

**Solicitud de pasaje**

Una vez que el usuario haya iniciado sesión, puede solicitar un pasaje:

1. El usuario especifica el día, recorrido y fecha del pasaje deseado.
2. El servidor registra la solicitud del pasaje en la base de datos.
3. El servidor solicita al sistema de búsqueda que encuentre el pasaje solicitado.
4. Comienza la búsqueda. Cuando encuentra disponibilidad, el sistema de búsqueda informa al cajero para que genere un enlace de pago de la comisión.
5. El enlace de pago se envía al servidor.
6. El servidor delega al servicio de notificaciones la tarea de informar al usuario sobre la disponibilidad del pasaje y le proporciona el enlace de pago

**Pago de la comisión**

Una vez se consigue pasaje se notifica al usuario que se consiguió el mismo. Para adquirirlo el usuario debe realizar el pago de la comisión:

1. El usuario accede al enlace de pago y completa el pago.
2. El cajero verifica que el pago se haya realizado correctamente.
3. Una vez verificado, el cajero notifica al servidor.
4. El servidor informa al sistema de búsqueda que la compra del pasaje puede finalizar.
5. El sistema de búsqueda proporciona el enlace de pago final del pasaje.
6. El servidor recibe el enlace, y lo transmite al notificador, quien notifica al usuario y le brinda el enlace.
7. Finalmente, el servidor actualiza la base de datos para reflejar que el pasaje ha sido pagado.

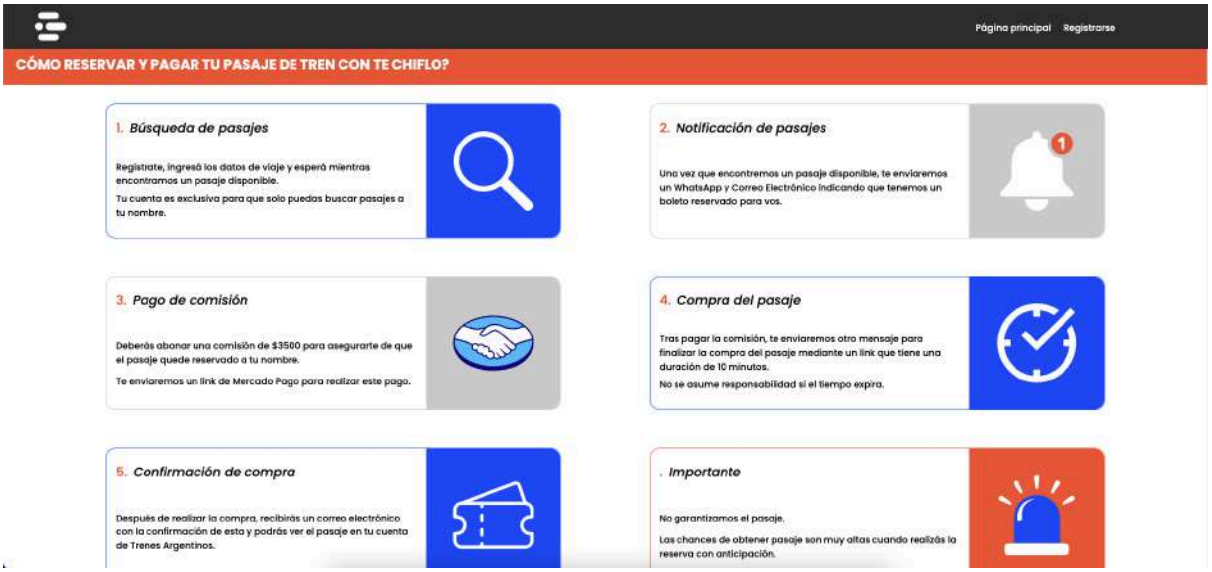
# 2. Detalles del producto

En esta sección se presenta una descripción detallada de aspectos técnicos del producto, junto con las tecnologías y los aspectos clave que sustentan cada uno de los microservicios que lo componen. Cada microservicio se analiza individualmente, explicando las herramientas, marcos y aspectos relevantes de cada módulo.

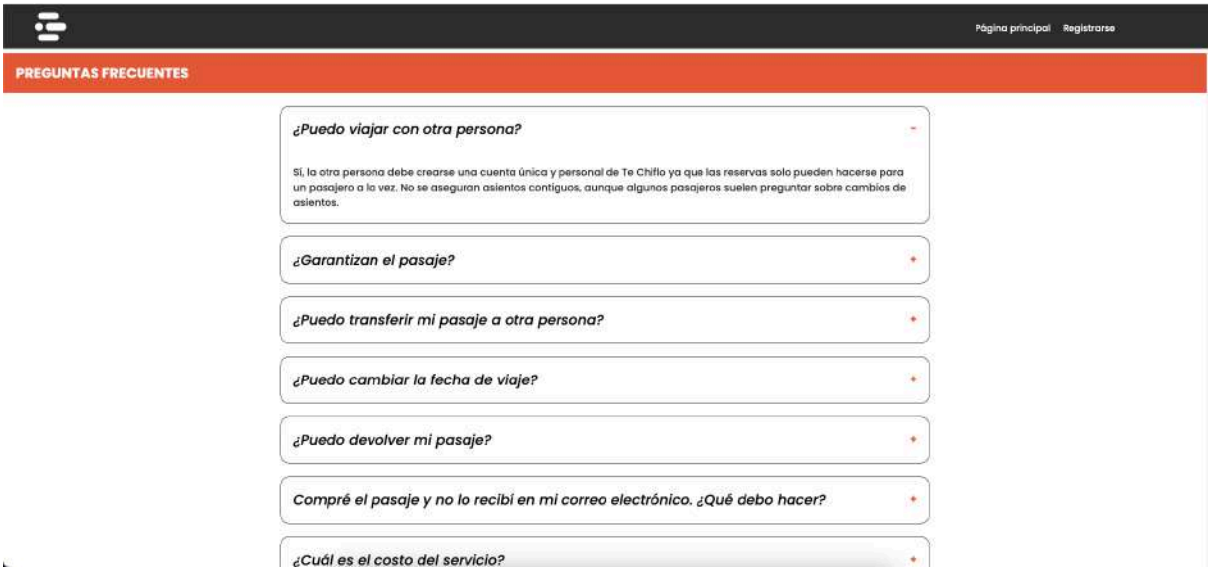
## 2.1 Front-End

### 2.1.1 Página principal

Para la página de bienvenida, se buscaba algo que atraiga al usuario a la primera y cumpla con el objetivo de mostrarle al usuario que es lo que hacemos: conseguir pasajes de tren. Se buscó implementar un diseño minimalista en la página web, con el objetivo de evitar distracciones para el usuario y asegurar que comprenda su propósito.

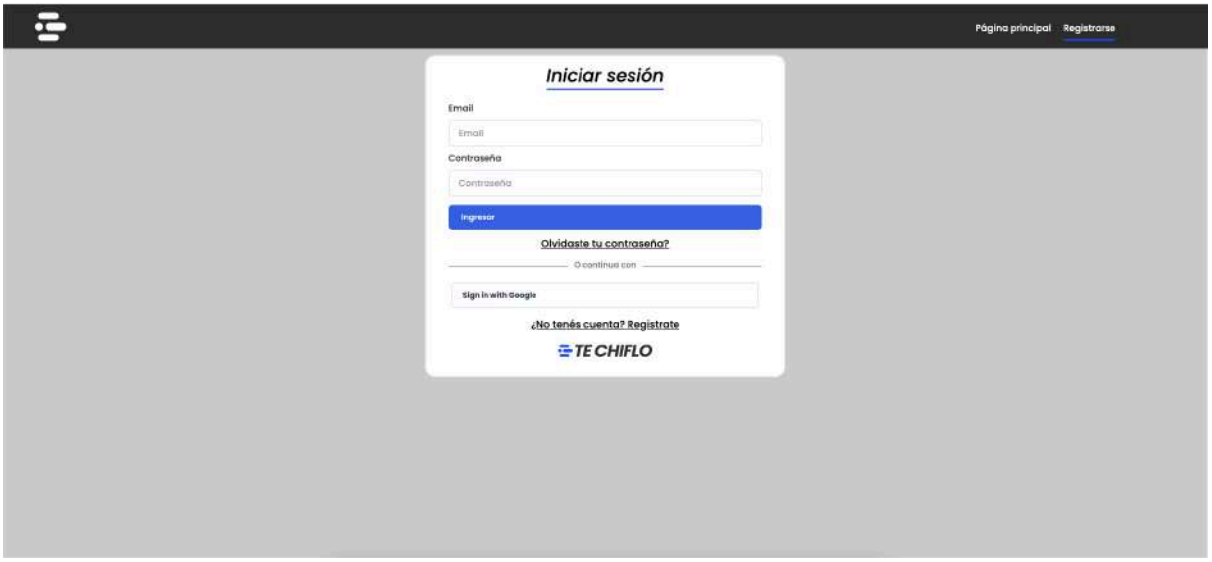


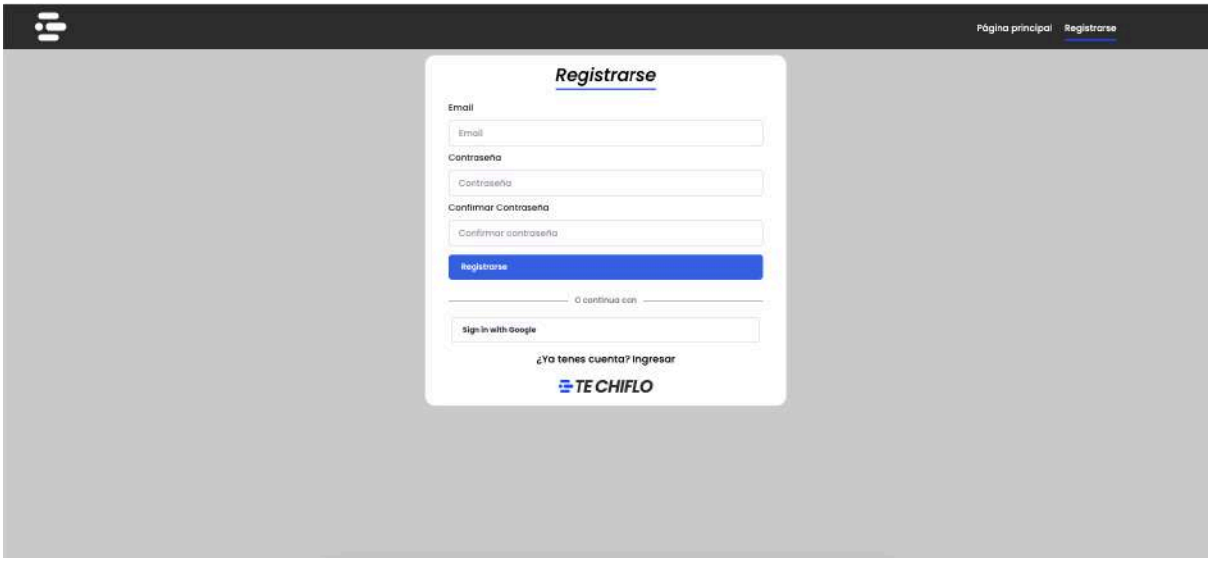

Además, consideramos crucial que el usuario comprendiera completamente el proceso para que pudiera utilizarlo sin ninguna duda. Por esta razón, creamos una sección de preguntas y respuestas al final de la página de bienvenida que aborda las consultas el equipo había recibido anteriormente. Aunque esta sección pueda parecer de poca importancia, es extremadamente relevante, ya que aborda todas las preguntas que se habian planteado antes de lanzar la página web al público.



### 2.1.2 Login usuario

Dos secciones importantes del sistema son las de ingreso y registro. A través de estas, los usuarios pueden acceder al sistema para encargar un pasaje o, en caso de no tener cuenta, registrarse. Ambas secciones comparten un diseño similar y sencillo. Se diferencian únicamente en algunos campos del formulario. Se añadió la opción de ingresar o registrarse tanto de manera local como a través de terceros, como Google. Esta funcionalidad busca ofrecer a los usuarios diversas opciones según sus preferencias, que permitan facilitar el acceso a la plataforma de manera conveniente.



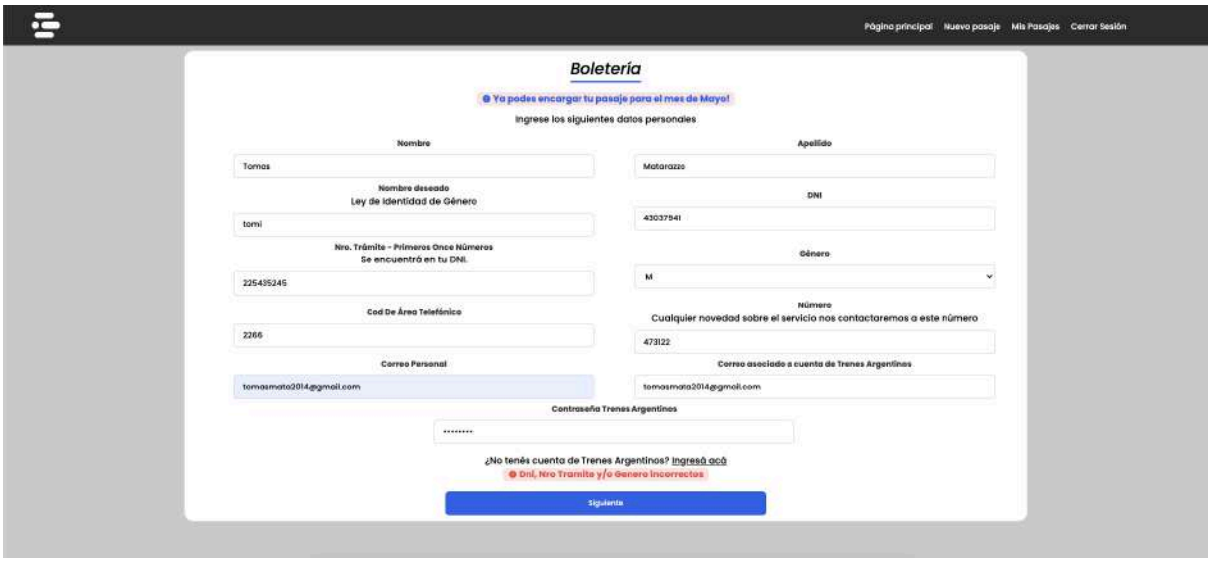



### 2.1.3 Validación de datos

Una vez que el usuario se encuentra logueado en el sistema, es fundamental que, al realizar la búsqueda de un pasaje, sus datos sean correctos. Debido a las condiciones del microservicio de búsqueda y de Trenes Argentinos, cualquier búsqueda con datos erróneos no es procesada lo que impide que la búsqueda sea efectiva. Por este motivo, es crucial asegurar que estos datos sean precisos.

Después del registro del usuario, no se permite realizar ninguna acción dentro del sistema hasta que el usuario valide sus datos. Este proceso de validación consiste en un formulario que, una vez completado, verifica la exactitud de los datos con el servidor de Trenes Argentinos, a través del microservicio de búsqueda. Si los datos son correctos, se actualiza el estado del usuario en la base de datos, y se marca como validado.

Este mecanismo de validación tiene como objetivo garantizar que todos los usuarios proporcionen información correcta desde el principio, para evitar problemas durante la búsqueda y compra de pasajes. La validación de datos no solo asegura la integridad del proceso, sino que también mejora la experiencia del usuario al prevenir errores y rechazos en etapas posteriores.

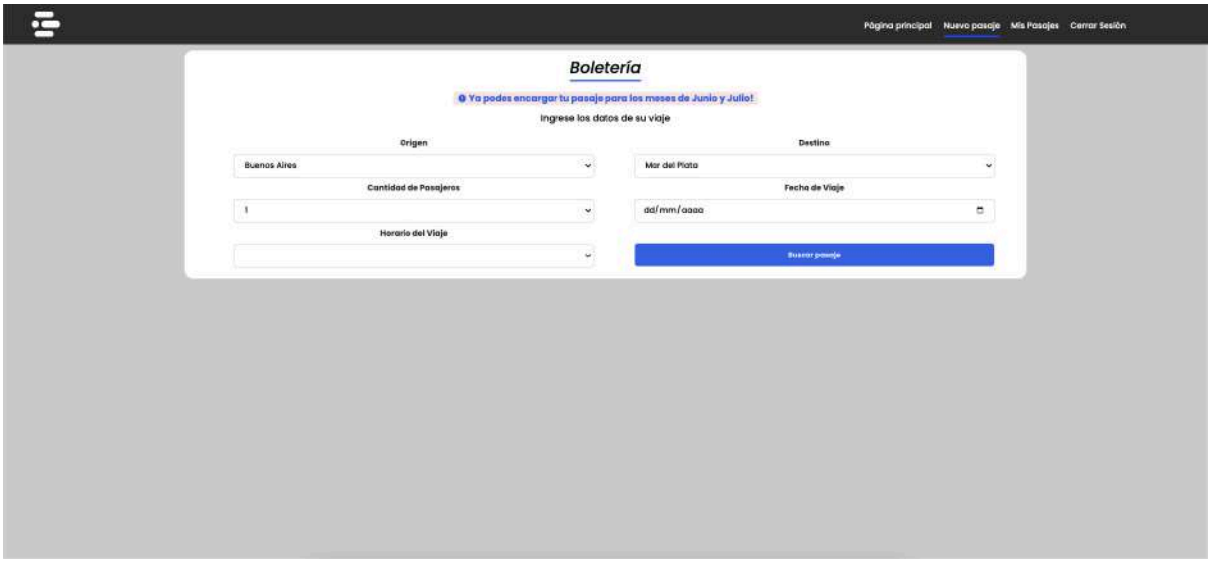


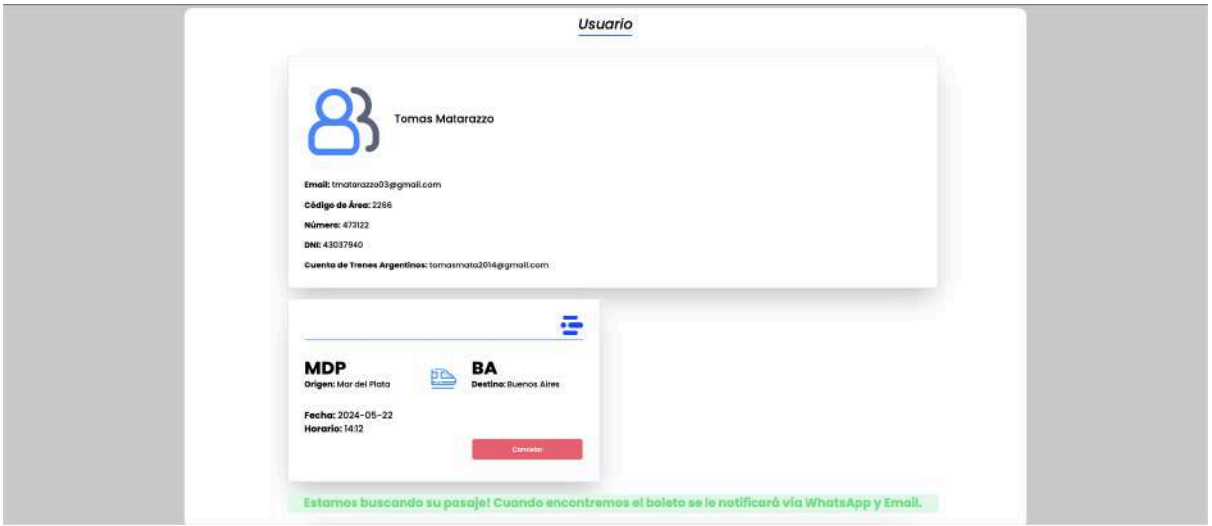

Una vez validados los datos del sistema, el usuario podrá solicitar un pasaje a través del formulario en la sección "Nuevo Pasaje". Este formulario incluye campos para el origen,

destino, fecha y horario del viaje. El usuario podrá realizar la solicitud de pasaje una única vez. Tras enviar la solicitud, se le notifica vía WhatsApp y email, si se consiguió el pasaje.

Además, el usuario tendrá acceso a la sección "Mi Pasaje", donde podrá ver todos los pasajes pendientes a partir de la fecha actual. En esta sección, el usuario también tendrá la opción de cancelar una reserva si así lo desea. Esta funcionalidad permite al usuario gestionar sus pasajes de manera eficiente y realizar cancelaciones directamente a través del sistema.

Esta estructura asegura que el proceso de solicitud y gestión de pasajes sea claro y sencillo. Mejora la experiencia del usuario al proporcionar una interfaz intuitiva y opciones de notificación y cancelación convenientes.



## 2.2 Servidor

El microservicio se creó en un servidor Node.js con Express. Este servidor es responsable de comunicarse con la base de datos y funciona como API Gateway dentro de la aplicación, es decir, todas las solicitudes pasan a través del servidor.

Fue necesario utilizar los distintos esquemas diseñados para la base de datos de MongoDB dentro del servidor. Para esto, se utilizó la librería Mongoose, un ORM (Object-Relational Mapping) muy popular hoy en día. Facilita la interacción con MongoDB al proporcionar una capa de abstracción que permite definir esquemas y modelos de datos de manera sencilla y estructurada.

Una vez definidos los distintos esquemas, se creó un esquema Data Access Object (DAO) para cada esquema general y, posteriormente, un DAO personalizable para cada entidad distinta. Esta práctica de código fue de gran importancia, ya que seguir buenas prácticas de programación no solo mejora la legibilidad y mantenibilidad del código, sino que también facilita el desarrollo a largo plazo. Implementar patrones como el DAO ayudó a desacoplar la lógica de negocio de la lógica de acceso a datos, esto permitió que el sistema sea más modular y fácil de extender o modificar.

Además, el uso de Mongoose y la implementación del patrón DAO aportaron varios beneficios adicionales:

- Validación de datos: Mongoose permite definir reglas de validación directamente en los esquemas con el objetivo de garantizar que los datos almacenados en la base de datos cumplan con ciertos criterios de calidad y formato.

- Consultas y población de datos: Mongoose proporciona métodos para realizar consultas complejas y la población de documentos relacionados. Así se facilita la recuperación eficiente de datos relacionados entre diferentes colecciones de MongoDB.

La integración de estas herramientas y prácticas resultó en un microservicio mantenible y adaptable, capaz de manejar un crecimiento en la carga de trabajo sin comprometer el rendimiento. La capacidad de escalabilidad del sistema se vio fortalecida y permite que la aplicación maneje un número creciente de usuarios y operaciones sin degradar su desempeño.

Otro rol de suma importancia del servidor es el manejo del inicio de sesión de la página web. Se encarga de la identificación y autenticación de los usuarios. Para esta tarea, se optó por utilizar una librería conocida como Passport.

### 2.2.1 Passport

Passport es una biblioteca de autenticación flexible para Node.js. Su principal objetivo es facilitar la autenticación de usuarios en una aplicación web. Lo hace al propocionar un marco de autenticación que es fácil de implementar y que puede adaptarse a las necesidades específicas de una aplicación.

La implementación con Passport permitió simplificar el registro de usuarios , se puede hacerlo de dos maneras: login propio y a través de Google. Se implementó ambas estrategias. Además, Passport facilita el manejo de la autenticación de forma transparente, lo cual simplifica su desarrollo. También proporcionaba middlewares para la autenticación, lo cual es crucial para asegurar la seguridad dentro de la aplicación. Esto garantiza que solo usuarios autenticados puedan acceder a ciertos endpoints para poder reducir las vulnerabilidades y mejorar la integridad del sistema.

## 2.3 Servicio de pagos

### 2.3.1 Descripción de endpoints

Se generaron distintos endpoints para la implementación del servidor. Es importante destacar que este microservicio se puede utilizar tanto para Trenes Argentinos como para otros servicios.

***Generar Link de Pago***

Este endpoint genera un enlace de pago a partir de los datos proporcionados en la solicitud.

***Recibir pago***

Este endpoint recibe notificaciones sobre transacciones provenientes de Mercado Pago y realiza acciones correspondientes, como cancelar un enlace de pago una vez fue pagado (para evitar pagos duplicados) y notificar sobre el pago completado. Se implementa la lógica para verificar si el pago se realizó correctamente. Una vez confirmado el pago se notifica al servidor de que el pago fue realizado con éxito, entonces el servidor se encarga de notificar al microservicio de búsqueda que finalice le otorgue el link para pagar el pasaje el usuario.

***Cancelar link***

Tras la confirmación del pago, era necesario cancelar el link para evitar pagos duplicados. Esta lógica también se implementó, ya que a veces ocurría que las personas pagaban dos veces el mismo link.

## 2.4 Servicio de comunicación

### 2.4.1 Descripción del marco legal en Argentina y su impacto en las redes sociales

En Argentina, el marco legal que regula las comunicaciones electrónicas y el uso de datos personales establece pautas claras para proteger a los usuarios de prácticas invasivas como el spam. Este marco legal afecta directamente a las redes sociales, ya que muchas de estas plataformas no ofrecen APIs públicas para el envío masivo de mensajes, como es el caso de WhatsApp. Esta restricción se debe, en parte, a la necesidad de cumplir con las normativas de protección de datos y privacidad vigentes en el país.

Debido a estas regulaciones, el acceso a las distintas APIs se ve comprometido y, además, por razones empresariales, WhatsApp no permite su uso a todo el mundo. En la actualidad, para hacer uso de estas funcionalidades, es necesario inscribirse como negocio dentro de la plataforma de WhatsApp, lo que se conoce como WhatsApp Business. Para el uso de la API, es necesario estar verificado en Meta, es decir, que la cuenta sea auténtica y esté validada a fin de ganar la confianza de los clientes. Para obtener la verificación de Meta, existen dos maneras:

1. **Verificación de la empresa por parte de WhatsApp:** Este proceso implica que WhatsApp realice una revisión directa de tu empresa. Para ello, se debe proporcionar documentos que demuestren la existencia legal y operativa de la empresa. Una vez presentada la solicitud, WhatsApp revisará la información proporcionada y, si todo está en orden, aprobará la verificación.

2. **Verificación de la empresa a través de proveedores oficiales de WhatsApp:** Otra forma de obtener la verificación es a través de un Business Solution Provider (BSP) oficial de WhatsApp. Estos proveedores actúan como intermediarios y facilitan el proceso de verificación.

Se intentó verificar la cuenta de WhatsApp de las dos maneras. Primero, se intentó directamente con formularios en línea, pero debido a la complejidad y selectividad del

proceso, se descartó esta opción. Luego, se exploró la verificación a través de terceros, a través de reuniones con Meta para asignar un BSP adecuado. Sin embargo, las soluciones ofrecidas eran demasiado avanzadas y costosas para las necesidades básicas del proyecto, que solo requería acceso a la API para enviar mensajes personalizados.

Paralelamente, mientras se estudiaba cómo obtener la verificación en Meta, se exploraban soluciones a través de automatizaciones con herramientas como Selenium. Con el paso del tiempo, se decidió que esta era una mejor opción y se optó por utilizar esta solución. El principal desafío era la necesidad de optimizarla para manejar grandes volúmenes de datos.

Inicialmente, en la versión inicial de este microservicio de búsqueda basado en automatizaciones con Selenium, la cantidad de mensajes simultáneos que se podían manejar era de aproximadamente diez. Aunque esta capacidad era muy limitada, cumplía con el objetivo de posibilitar los mensajes por WhatsApp. Se decidió implementar esta solución, ya que la opción a través del BSP resultaba demasiado costosa. Aunque tenía una desventaja significativa en términos de capacidad, se tomó la decisión de mejorarla paralelamente con el desarrollo del proyecto.

La decisión de utilizar Selenium para automatizar el envío de mensajes por WhatsApp, a pesar de sus desafíos iniciales, proporcionó una solución viable y económica. Con un enfoque en la optimización y escalabilidad, se pudo satisfacer la necesidad de comunicación eficiente y efectiva con los usuarios que asegure el crecimiento y éxito continuado del proyecto.

### 2.4.2 Descripción de endpoints

**Enviar mensaje**

Este endpoint maneja la lógica principal para enviar mensajes de WhatsApp basados en los datos recibidos en el cuerpo de la solicitud, para este servicio se necesitan únicamente dos mensajes. El mensaje de se encontró un pasaje y el de una vez se pagó la comisión. Este endpoint utiliza un mutex para asegurar que sólo una instancia del proceso de envío de mensajes se ejecute a la vez para evitar conflictos en el acceso a los recursos de WhatsApp Web.

- Entrada: Información del usuario y mensaje que se desea enviar.
- Salida: Un mensaje de éxito o error basado en el resultado de enviar los mensajes.

**Mensaje múltiple**

Este endpoint está diseñado para enviar un mensaje de recordatorio a múltiples destinatarios. Cada entrada en el array se procesa de manera asíncrona y se envía un mensaje personalizado.

- Entrada: Números de teléfono que se desea enviar recordatorio.
- Salida: Un mensaje de éxito o error basado en el resultado de enviar los mensajes

## 2.5 Servicio de búsqueda

Para el microservicio de búsqueda se utilizó Python como lenguaje principal en el desarrollo del microservicio.El equipo utilizó TensorFlow (una libreria de Python dedicada a Machine Learning) para el desarrollo de una inteligencia artificial capaz de resolver el captcha de Trenes Argentinos. Para recibir solicitudes HTTP se utilizó Flask.

### 2.5.1 Implementación

La arquitectura obtenida para este microservicio consta de tres niveles. Una capa de Controlador, encargada de recibir solicitudes HTTP.

- La capa de Orquestador, responsable de los datos y la organización de los mismos.
- La capa de Thread Running, que es la etapa en la que se realiza la búsqueda de los pasajes concretamente.
- La capa de Controlador, resuelve las solicitudes HTTP entrantes al utilizar las herramientas de Flask. Se definen endpoints representativos a las acciones que se necesitan realizar.
- Algunos ejemplos son:
    - Crear nueva búsqueda
    - Ver búsquedas activas
    - Eliminar búsqueda

La arquitectura del sistema se estructura en varias capas cruciales para la gestión eficiente de las solicitudes de pasajes en Trenes Argentinos:

**Capa de controlador:**

Esta capa recibe las solicitudes de los usuarios y delega el trabajo a la siguiente capa del sistema.

**Capa de orquestador:**

El Orquestador agrupa a los usuarios por viaje y horarios específicos. Utiliza una robusta estructura de objetos para gestionar y mantener toda la información relevante de los viajes, pasajeros y sus interrelaciones en memoria.

Esta capa prepara el contexto de ejecución para cada hilo antes de iniciar su trabajo. Administra los hilos a lo largo del proceso y se asegura de que solo haya un hilo monitor por cada viaje para minimizar así las solicitudes al servidor de Trenes Argentinos.

Antes de agregar a un pasajero en la estructura de datos, se valida la corrección de sus datos para prevenir errores en la obtención del pasaje. En caso de datos incorrectos, se notifica al backend para actualizar la base de datos y solicitar al usuario que actualice su información (por ejemplo, en caso de cambio de contraseña en Trenes Argentinos).

**Capa de hilos y sesión monitor:**

Mediante sesiones monitor en la capa de hilos, se minimiza la cantidad de consultas de disponibilidad para un viaje específico. Cada consulta a un viaje devuelve la disponibilidad de todos los horarios correspondientes, esto permite agrupar los viajes incluso cuando los horarios varían. Esta estrategia no solo optimiza el rendimiento, sino que también reduce el uso del Captcha Solver, que es costoso computacionalmente y en tiempo.

Esta arquitectura está diseñada para asegurar una gestión eficiente y escalable de las operaciones relacionadas con la obtención de pasajes en Trenes Argentinos. Optimiza recursos y mejora la experiencia del usuario al minimizar tiempos de espera y errores potenciales.

Esta es la información que debe enviarse para obtener la disponibilidad:


Y este es el display de la disponibilidad:


En las primeras etapas del proyecto, el proceso de obtención de pasajes se realizaba con Selenium, con la instancia de un navegador por cada viaje. Sin embargo, se adoptó posteriormente una estrategia más eficiente basada en sesiones y peticiones HTTP, que almacena la información relevante en cookies.

Cuando el monitor detecta disponibilidad, activa un nuevo hilo para el viaje correspondiente con los datos del primer pasajero en la lista para el horario disponible. Este nuevo hilo maneja todo el proceso hasta la obtención del pasaje.

Una vez obtenido el pasaje, el sistema se comunica con el microservicio de pagos a través del backend para obtener el enlace de pago de la comisión. Se utilizan los datos del viaje y

del pasajero para contactarse entonces con el micro servicio de notificaciones para enviar avisos por correo electrónico y WhatsApp.

Tras el pago de la comisión por parte del usuario, se repite el proceso de notificación con el enlace de pago del boleto de tren. La sesión finaliza con éxito una vez completado este último paso.