

Sappia Badra, Francisco

Rojas, Juan Pablo

Sistema de gestión de talleres automovilísticos

**Tesis para la obtención del título de
grado de Ingeniero en Sistemas**

Directores:

Ignacio Luciano Carreño Federico

Eduardo Porrini

Documento disponible para su consulta y descarga en Biblioteca Digital - Producción Académica, repositorio institucional de la Universidad Católica de Córdoba, gestionado por el Sistema de Bibliotecas de la UCC.



[Esta obra está bajo una licencia de Creative Commons Reconocimiento- No comercial- Compartir igual 4.0 Internacional.](https://creativecommons.org/licenses/by-nc-sa/4.0/)



**UNIVERSIDAD
CATÓLICA
DE CÓRDOBA**
JESUITAS

Sistema de Gestión de talleres automovilísticos

Francisco Sappia Badra - 1911857

Juan Pablo Rojas - 1906022

Índice

RESUMEN	4
ABSTRACT	4
PRESENTACIÓN DEL TEMA	4
GLOSARIO	4
DIAGNÓSTICO (PROBLEMÁTICA)	5
OBJETIVOS	7
MARCO TEÓRICO	8
Investigación sobre la problemática	8
Diálogo con los talleres	9
Información del dominio	10
Análisis de opciones existentes	11
Tecnologías consideradas	12
Plataformas de implementación	13
Backend:	13
Motores de Bases de Datos	17
Frontend	19
Arquitecturas	21
Despliegue	22
Patrones	23
Chatbot	24
Pruebas	25
PROPUESTA DE SOLUCIÓN	28
Alcance Funcional	28
Etapas	28
Planificación y análisis del proyecto	28
Definición Ante Proyecto	29
Confección Libro de Tesis	29
Etapas de diseño	29
Etapas de desarrollo	29
Etapas de prueba	30
Etapas de despliegue	30
Modelado de Requerimientos	31
Cliente	31
Talleres	32
Administrador	33
ChatBot	34
Historias de Usuario	34
Cliente	34
Talleres	36
Administrador	38
ChatBot	39
Casos de Uso	39
Diseño - Vistas	55

Módulo Login	55
Módulo Registro	56
Módulo Cliente	56
Módulo Taller	66
Módulo Administrador	76
Diagramas	83
Diagrama de Casos de Uso	83
Cliente	83
Taller	85
Administrador	86
Tecnologías elegidas	87
Arquitectura	89
Arquitectura del sistema	94
Chatbot	94
Implementación	98
BENEFICIOS POST-IMPLEMENTACIÓN	98
IMPACTOS	100
IMPACTO ECONÓMICO (estudio de costos)	100
IMPACTO SOCIAL	101
IMPACTO MEDIOAMBIENTAL	101
CONCLUSIONES	103
FUENTES	104

ABSTRACT

Nowadays, the use of a car has become almost essential for many people, but with constant use, it is common for vehicles to break down or need maintenance every few months. In these situations, having a reliable mechanical workshop becomes essential to solve these problems and provide peace of mind and confidence to vehicle owners.

The objective of this project is to benefit both workshops and clients, strengthening the link between both parties and improving the overall experience. To achieve this, it is proposed to develop a web application called TuneUp that facilitates the organization and management of mechanical workshops, while improving the user experience by avoiding delays and waste of time.

The system will allow customers to book appointments easily and quickly, avoiding the need for long waits or uncertainty about workshop availability. On the other hand, it will help workshops optimize their internal operations, efficiently manage shifts and resources, and maintain clear and transparent communication with their customers.

RESUMEN

En la actualidad, el uso del automóvil se ha vuelto casi indispensable para muchas personas, pero con el constante uso, es común que los vehículos se rompan o necesiten mantenimiento cada pocos meses. Ante estas situaciones, contar con un taller mecánico confiable se vuelve fundamental para resolver estos problemas y proporcionar tranquilidad y confianza a los propietarios de los vehículos.

El objetivo de este proyecto es beneficiar tanto a los talleres como a los clientes, fortaleciendo el vínculo entre ambas partes y mejorando la experiencia global. Para lograr esto, se propone desarrollar una aplicación web llamada TuneUp que facilite la organización y gestión de los talleres mecánicos, al mismo tiempo que mejore la experiencia del usuario al evitar demoras y pérdidas de tiempo.

El sistema permitirá a los clientes reservar turnos de manera fácil y rápida, evitando la necesidad de largas esperas o la incertidumbre sobre la disponibilidad del taller. Por otro lado, ayudará a los talleres a optimizar sus operaciones internas, gestionar eficientemente los turnos y recursos, y mantener una comunicación clara y transparente con sus clientes.

PRESENTACIÓN DEL TEMA

Este proyecto consiste en desarrollar un sistema web para la gestión de turnos de talleres mecánicos. El mismo surge porque notamos que tanto los clientes como los talleres de autos tienen problemas para organizar los turnos de manera eficiente en el día a día. La solución propuesta se enfoca en facilitar el proceso de reserva de turnos, ayudando a organizarse y reducir tiempos a ambas partes mencionadas.

La implementación de esta plataforma tiene como objetivo fundamental optimizar la experiencia de todos los involucrados en la gestión de turnos en talleres mecánicos.

GLOSARIO

- API (Application Programming Interface): Es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones. Permiten que diferentes aplicaciones se comuniquen entre sí. Actúa como una capa intermedia que procesa las transferencias de datos entre sistemas, permitiendo a las empresas abrir sus datos y funcionalidades de aplicaciones a desarrolladores externos, socios comerciales y departamentos internos dentro de sus empresas.
- Chatbot: programa informático diseñado para interactuar con los usuarios a través de conversaciones en lenguaje natural. Utiliza diferentes técnicas para comprender y responder a las preguntas y solicitudes de los usuarios de manera automatizada.
- DAO (Data Access Object): El patrón DAO es una abstracción que se utiliza para separar la lógica de acceso a datos de la lógica de negocio en una aplicación. Un objeto DAO proporciona una interfaz para acceder a una base de datos o cualquier otra fuente de datos, permitiendo que la lógica de negocio se comuniquen con la fuente de datos de manera independiente.
- DTO (Data Transfer Object): Un DTO es un patrón de diseño utilizado en programación que se utiliza para transferir datos entre subsistemas de una aplicación. Generalmente, un DTO es una clase simple que contiene campos de datos y métodos para acceder a ellos. Se utiliza para encapsular datos que deben transmitirse entre diferentes componentes del software.
- ORM (Object-Relational Mapping): El mapeo objeto-relacional es una técnica que permite a las aplicaciones interactuar con bases de datos relacionales utilizando objetos en lugar de consultas SQL directas. Un ORM mapea objetos de una aplicación directamente a tablas en una base de datos, lo que facilita el acceso y la manipulación de datos de manera más orientada a objetos en lugar de usar SQL tradicional.

DIAGNÓSTICO (PROBLEMÁTICA)

El proyecto surge como respuesta a la necesidad de mejorar la gestión de talleres de autos, con el objetivo de brindar una experiencia más eficiente y transparente tanto para los clientes como para los talleres. La problemática que dio origen a este trabajo se basa en las dificultades y desafíos que enfrentan tanto los clientes como los talleres mecánicos al momento de organizar y gestionar los turnos.

En la vida real, es común encontrarse con situaciones en las que los clientes tienen dificultades para encontrar un taller mecánico disponible en el momento que necesitan realizar una reparación o mantenimiento en su vehículo. Por otro lado, los talleres también enfrentan desafíos al intentar organizar y asignar eficientemente los turnos de trabajo, evitando demoras y conflictos.

La idea de este proyecto surge como una contribución a la sociedad, con el objetivo de mejorar esta problemática y facilitar la interacción entre los clientes y los talleres mecánicos. A través del desarrollo de un sistema web de gestión de turnos, buscamos optimizar el proceso de reserva de turnos por parte de los clientes, brindándoles una interfaz intuitiva y accesible donde puedan solicitar y confirmar sus turnos de manera rápida y sencilla.

Para los talleres, nuestro sistema les permitirá gestionar de manera eficiente los turnos recibidos, llevando un registro completo de todas las citas y trabajos realizados. Esto les brindará una visión clara de su carga de trabajo, les ayudará a asignar los recursos adecuados y a minimizar las demoras.

Sin ser redundantes, la propuesta de este sistema surge puntualmente de las problemáticas que se presentan a continuación:

- **Falta de transparencia en los procesos de reparación:** muchas veces los clientes no están seguros de lo que están haciendo a sus vehículos mientras están en reparación.
- **Problemas para programar turnos:** los clientes tienen problemas a la hora de coordinar turnos con los talleres debido a una mala gestión de fechas y horarios. El sistema permitirá a los clientes solicitar turnos con mayor facilidad y eficiencia.

Reduciría la interacción entre el cliente y personal del taller, lo que le permitirá que no se produzcan interrupciones en el trabajo del turno.

- **Falta de organización en los talleres:** los talleres suelen presentar problemas de organización, que lleva a los mismos a generar retrasos y una menor eficiencia en los procesos de reparación y también por apuro de no llegar con las fechas estipuladas, pueden cometer errores en dicha reparación. Este sistema permitirá a los talleres organizar mejor sus turnos y mejorar la eficiencia en la reparación de vehículos.

El impacto de nuestro sistema en la sociedad es significativo, ya que contribuye a optimizar la operatividad de los talleres de autos, ahorrando tiempo y recursos tanto para los profesionales como para los clientes. Esto se traduce en una mejor experiencia de servicio, reducción de tiempos de espera y una mayor satisfacción en general. Además, al fomentar la transparencia y el contacto directo, se fortalece la relación entre el taller y el cliente, generando un vínculo duradero y beneficiando a toda la comunidad automotriz.

OBJETIVOS

A la hora de implementar nuestro sistema nos enfocamos en cumplir algunas metas. Estos objetivos se dividen en dos categorías: globales y específicos. A continuación, detallaremos cada uno de ellos para brindar una visión clara de nuestra dirección y propósito.

- **Globales**

- Fortalecer la transparencia en el vínculo entre los talleres y los clientes: Nuestro sistema se enfoca en mejorar la comunicación y la información compartida entre los talleres y los clientes. Buscamos proporcionar a los clientes acceso transparente a la disponibilidad de turnos, políticas del taller y actualizaciones sobre el progreso de sus reparaciones.
- Mejorar la eficiencia en la gestión de turnos: Nos proponemos optimizar los procesos de programación y asignación de turnos en los talleres. Nuestro sistema ayudará a reducir los tiempos de espera, evitará conflictos en la programación y facilitará la administración de los turnos, lo que permitirá una atención más rápida y efectiva.

- **Específicos**

- Organizar y optimizar los procesos de reparación en los talleres: Nuestro sistema buscará mejorar la planificación y gestión de las reparaciones en los talleres. Implementaremos herramientas para la asignación eficiente de recursos y el control de tiempos, lo que reducirá retrasos y errores en los procesos y garantizará una mayor calidad en los servicios brindados.

MARCO TEÓRICO

En el marco teórico de nuestro proyecto, nos enfocaremos en investigar y analizar diversos aspectos relacionados con la problemática y el dominio del problema que pretendemos resolver.

Investigación sobre la problemática

Comenzamos realizando entrevistas a nuestros conocidos y familiares para comprender cómo se manejaban al tener que llevar su auto al mecánico. Identificamos que muchos de ellos experimentaban conflictos y dificultades en la gestión de los tiempos y comunicación con los talleres.

Entrando más en detalle, estos eran los problemas que tenían ellos:

- Ir al taller y que no haya lugar, es decir, los clientes iban hacia un taller a querer realizarle una revisión o reparación a su auto, y la gente del taller les decía que no podía atenderlos porque ya estaba colmada la capacidad de vehículos. Eso lleva a la gente a una pérdida de tiempo y dinero (nafta) que se transforma en una experiencia desfavorable para ellos.
- Relacionado a problemas con el tiempo, muchas veces también los clientes tenían conflicto con el pacto de fechas de entrega con los talleres. Un taller les decía que, por ejemplo, su auto iba a demorar 3 días en ser arreglado y entregado. Pasado los tres días el auto seguía sin estar arreglado, lo que lleva a una demora de entrega con fecha desconocida por parte del cliente.
Esto es un tema grave, porque el auto para mucha gente es su método de transporte para el trabajo y no saber cuando te lo van a entregar, genera mucha incertidumbre y estrés.
- Falta de comunicación: este también es un conflicto que nos nombraron, donde en muchos casos, la mala comunicación entre clientes y talleres lleva a tener problemas. Por ejemplo, si un cliente fue a llevar el auto para una reparación de un espejo roto, luego le podría llegar una factura con servicios adicionales que no estuvieron pactados verbalmente pero el taller consideró que eran necesarios hacerlos, como un cambio de aceite.

- Costos altos: algunos clientes se quejan de los distintos costos que puede llegar a haber en comparación de un taller con otro. En diferentes casos, los clientes quedan sorprendidos con el precio que el taller les brinda luego de haber realizado la reparación de los autos, sin una explicación lo suficientemente clara como para entender de dónde salen esos precios. Luego consultan en otros lugares, y haciendo la misma tarea, les cobran menos. Esto hace pensar a los clientes que fueron estafados y que hay falta de transparencia en el rubro.

Diálogo con los talleres

Nos acercamos a los propios talleres para obtener su perspectiva y comprender cómo un sistema de gestión podría ayudarles en su trabajo diario. Para garantizar que el sistema se adapte a las necesidades reales de los talleres, llevamos a cabo diálogos directos con distintos talleres mecánicos de la ciudad de Córdoba, incluyendo tanto talleres más grandes con múltiples empleados como aquellos de menor tamaño.

Durante nuestras conversaciones, encontramos un interés común en la búsqueda de soluciones que pudieran mejorar la organización y la eficiencia en las operaciones diarias. En particular, un taller más grande expresó su necesidad de optimizar tiempos y recursos, lo que les llevó a la conclusión de que un sistema de gestión de turnos sería una herramienta valiosa.

En el caso de otro de los talleres visitados, observamos que aún utilizaban métodos tradicionales, anotando los turnos y sus fechas con papel y lapicera. Este enfoque, aunque funcional, generaba cierta complejidad y pérdida de tiempo. También se habló de otro conflicto, el cual era que, en muchos casos, los clientes reservaban un turno y luego no cumplían con la cita programada. Esta situación generaba inconvenientes considerables, ya que los mecánicos y recursos estaban preparados para atender a clientes que finalmente no se presentaban.

En los distintos diálogos, se nos brindaron ciertos requerimientos que los talleres solicitaban para ayudarlos a crecer y a organizarse mejor. Los requerimientos de todos los talleres con los que hablamos englobaban que el sistema los ayude a mejorar su organización interna y la gestión de turnos. Querían poder gestionar de manera correcta sus recursos, tanto materiales como de personal de trabajo. Solicitaban también evitar pérdidas de tiempo y efficientizar su flujo.

Asimismo, pidieron no tener que estar contestando mensajes de clientes más tiempo del que utilizan para reparar vehículos, y aplicar un sistema de notificaciones automatizado para evitar esos mensajes.

Creemos que la implementación de nuestro sistema les ayudaría a resolver sus problemas, eliminando la necesidad de gestionar manualmente los turnos y permitiéndoles realizar un seguimiento más preciso y automatizado de su carga de trabajo.

Información del dominio

La construcción de un sistema que permita sistematizar la interacción entre el cliente y el taller, permitiendo la trazabilidad de los turnos, optimización de tiempos de los talleres y alcanzar el mejor nivel de eficiencia en los talleres. En el sistema existen flujos de estados que brindan información sobre cómo está el turno que el cliente sacó, permitiendo en todo momento saber cómo viene el turno. Permite tener un historial de todos los turnos a los que fueron sometidos los autos de cada cliente, lo cual fomenta la transparencia y ante cualquier cosa, se acude a la información centralizada que se encuentra alojada en el sistema.

Entonces el análisis de campo que hicimos fue, cómo nombramos anteriormente, acercarnos a 2 talleres, donde conversamos con los dueños y gente que trabaja en el taller, identificando las debilidades y cuestiones a mejorar, las cuales son las siguientes:

- Mala gestión operativa interna: haciendo referencia a la falta de eficiencia en los procesos internos de los talleres, como falta de coordinación entre empleados o problemas con la gestión de inventario.
- Conflictos de comunicación: malentendidos entre talleres y clientes, cuestiones que no quedan claras, ausencia de canales de comunicación.
- Pérdidas de tiempo: debido a malas planificaciones, retrasos por mala organización. Esto afecta a la rentabilidad del taller.
- Gestión deficiente de recursos: mala gestión del personal de trabajo, máquinas y espacio físico que puede provocar tiempos de espera prolongados y retrasos.

Nos comentaron cómo eran sus operaciones diarias y como era su día a día, el cual comenzaban abriendo a las 8 de la mañana y arrancando a revisar y reparar los vehículos que se encontraban en el taller. El límite de capacidad de vehículos a atender en un mismo momento en el taller más grande con el que dialogamos es de 5 autos.

A medida que están reparando los autos pendientes, otros clientes van llegando al lugar para que les revisen el suyo. Aquí surge otro problema, y es que, en muchas ocasiones se les tiene que comentar a los clientes que la capacidad está colmada y que deben volver más tarde o dejar el auto en espera hasta que terminen de revisar los demás pendientes.

Luego de resolver el problema del vehículo, se llama al cliente por teléfono y se le comunica que el auto está listo para ser retirado y abonado. Si todo está en orden, el cliente se retira con su vehículo y deja su lugar para otro posible cliente.

En cuanto al tema de los turnos, en el taller Suspension Martin utilizan una computadora pero con un sistema básico el cual utilizan para fijar fechas en el calendario acompañado de un sistema de facturación. Las solicitudes de turnos las reciben por teléfono o por orden de llegada al taller.

En el taller Bustos Fierro, las solicitudes de turnos se hacen de la misma manera que el taller anterior pero no tienen un sistema de gestión de turnos. Esto lo hacen anotando los turnos en una hoja de papel con una lapicera, lo cual consideramos que se podía sistematizar. Además, en este caso las solicitudes vía telefónica las recibe la misma persona que repara los vehículos, lo cual lo hace ineficiente.

Básicamente, el flujo a simple vista parece sencillo, recibir y reparar autos, gestionar turnos y el pago de los mismos, pero dialogando con los talleres nos dimos cuenta que esto se podría mejorar y optimizar para lograr mejores resultados. Tomamos nota de todos los problemas que se podían abordar con el sistema que luego de una retroalimentación, le propusimos desarrollar para mejorar potencialmente esos puntos específicos. A través de esto, se nos permitió analizar y profundizar más en el dominio del problema, dando pie al nacimiento de nuevos requerimientos y establecimientos de reglas de negocios generales.

Análisis de opciones existentes

Investigamos las opciones similares disponibles en el mercado para la gestión de talleres y turnos. Evaluamos sus características, funcionalidades y alcance. Sin embargo,

consideramos que nuestro sistema trata de apuntar a un enfoque de transparencia y trazabilidad que en otros sistemas no se observó.

Algunos de los sistemas existentes en el mercado son TallerOn, TallerGP y MotorEasy 3.0. Por motivos de que estos sistemas son pagos y especialmente para talleres mecánicos, no tenemos acceso a ellos pero podemos analizar los servicios que ofrecen y su página web.

Aspecto \ Taller	TallerOn	TallerGP	MotorEasy 3.0
PROS	+ Permite la gestión de turnos online para clientes y talleres. + Gestión de Pagos + Calendario integrado para gestionar las fechas de los turnos. + Historial de reparaciones + Gestión de inventarios + Gestión de recursos + Permite generar reportes	+ Gestión de Cobros y sueldos + Historial de reparaciones + Protocolos de seguridad + Gestión de inventarios + Soporte completo + Permite generar reportes	+ Sistema de notificaciones + Gestión de clientes + Control de caja + Gestión de inventarios + Registro de compras y ventas
CONTRAS	- No se pueden ver los presupuestos desde la página	- No nombra nada puntual acerca de la gestión de turnos. - Usuarios limitados	- No nombra nada puntual acerca de la gestión de turnos. - Interfaz poco moderna

Tecnologías consideradas

Cuando hablamos de elegir las tecnologías para nuestro sistema de gestión de turnos en talleres mecánicos, queremos fundamentar esas decisiones de una manera sencilla y que tenga sentido.

Nuestro enfoque y objetivo a la hora de desarrollar el sistema está centrado en cuatro grandes características:

- **Escalabilidad:** es esencial garantizar que la aplicación tenga la capacidad de crecer sin inconvenientes a medida que la demanda y la carga de trabajo aumenten con el tiempo. Este aspecto es fundamental para garantizar que el sistema pueda manejar eficientemente un mayor volumen de datos y usuarios a medida que más talleres y clientes utilicen la plataforma.

Nuestra intención es diseñar una arquitectura que permita la expansión fluida de la capacidad, garantizando que, a medida que más talleres se integren al sistema y más clientes soliciten turnos, no se produzcan obstáculos técnicos. Queremos evitar posibles cuellos de botella en el rendimiento y asegurar que la experiencia del usuario se mantenga rápida y eficiente incluso en momentos de alta demanda.

- **Facilidad de uso:** En cuanto a la facilidad de uso, nos enfocaremos en desarrollar un sistema que sea intuitivo y amigable para los usuarios, tanto para los talleres mecánicos como para los clientes que reservan turnos. La interfaz de usuario se diseñará de manera clara y sencilla, con el objetivo de minimizar la curva de aprendizaje y permitir que los usuarios naveguen por la plataforma de manera eficiente.
- **Seguridad:** La seguridad es una prioridad fundamental en el diseño de nuestro sistema. Nos esforzamos por implementar medidas robustas para proteger tanto la información de los talleres como la de los clientes. Esto incluye la adopción de prácticas de cifrado para datos sensibles.

- **Compatibilidad con dispositivos y plataformas existentes:** Buscamos que nuestro sistema sea versátil y compatible con una variedad de dispositivos y plataformas. Esto implica garantizar que la plataforma sea accesible desde dispositivos móviles, tabletas y computadoras de escritorio, permitiendo a los usuarios acceder y gestionar turnos de manera conveniente desde cualquier lugar.

A partir de las características nombradas, elegimos distintas tecnologías que creemos que cumplen con lo que buscamos.

A continuación se encuentran definidas todas las posibles tecnologías y paradigmas que hemos considerado para nuestro sistema.

Plataformas de implementación

- **Aplicación web:** Las aplicaciones web, son aplicaciones que tienen lugar en el navegador web. Estas se cargan en el servidor web y se ejecuta en un navegador, la misma no requiere de ninguna instalación.
Cabe aclarar que una aplicación web no es lo mismo que una página web. La diferencia está en que la aplicación web contiene un modelo de negocio por detrás y la página web puede ser un sitio estático informativo y de contacto.
- **Aplicación móvil:** Es un tipo de aplicación diseñada para ejecutarse en un dispositivo móvil, que puede ser un celular inteligente, tablets. Estas proporcionan funcionalidades aisladas y limitadas. Sería de mucha utilidad que los usuarios puedan utilizar la aplicación desde su celular.

Backend:

- **C#:** Es un lenguaje de programación compilado desarrollado por Microsoft que se utiliza principalmente para el desarrollo de aplicaciones en el entorno de .NET. Es un lenguaje orientado a objetos y está diseñado para ser simple, moderno y seguro. Su sintaxis clara y su amplio conjunto de características lo hacen adecuado para el desarrollo de aplicaciones web robustas y escalables.

Consideramos utilizar este lenguaje por varias razones. En primer lugar, la experiencia previa que teníamos con este lenguaje de programación fue un factor clave. Ya hemos trabajado con C# en proyectos anteriores y estábamos familiarizados con su sintaxis, sus características y su ecosistema de desarrollo. Esta familiaridad nos proporciona una ventaja inicial al comenzar el proyecto, permitiéndonos avanzar rápidamente y con mayor confianza en la implementación. Investigando, notamos que cuenta con una comunidad activa de desarrolladores y una amplia gama de recursos de aprendizaje disponibles, lo cual nos será de gran ayuda a la hora de enfrentarse a problemas.

Asimismo, la forma en que C# maneja la programación orientada a objetos es clave para estructurar nuestro proyecto de manera clara y escalable, ya que nos va a permitir organizar nuestro código de forma ordenada.

- **ASP.NET Core:** Es un framework de desarrollo de software de código abierto desarrollado por Microsoft que se utiliza para crear aplicaciones web y servicios. Proporciona un conjunto de herramientas y bibliotecas que permiten desarrollar aplicaciones web de alto rendimiento y escalables. Su arquitectura modular y su integración con .NET Core lo hacen ideal para el desarrollo de servicios web eficientes y seguros.

Además, es multiplataforma lo cual nos es super útil ya que buscamos que el sistema pueda usarse en todos los dispositivos, y es de código abierto lo cual nos permite obtener respuestas a problemas por parte de la comunidad.

- **.NET Framework:** es un entorno de desarrollo de software desarrollado por Microsoft que proporciona un amplio conjunto de bibliotecas y herramientas para crear aplicaciones y servicios web, de escritorio y móviles. En el último tiempo, hubo mucho crecimiento de .NET 5.0 a 6.0. Esta actualización fue clave y un indicio de que .NET está creciendo cada vez más.

Una vez que consideramos que C# podría ser un lenguaje a utilizar, pensamos en .NET para aplicarlo como framework, ya que anteriormente hemos trabajado con estas herramientas en otros proyectos.

- **ASP.NET Web API:** es un framework de desarrollo de servicios web RESTful basado en .NET Framework. Permite a los desarrolladores construir servicios web que se pueden consumir mediante HTTP utilizando una variedad de formatos, como JSON

y XML. Consideramos esta opción porque te proporciona una estructura con las configuraciones listas para poder generar APIS y poder migrar hacia la arquitectura que nosotros deseemos.

- **Java:** Tuvimos en cuenta Java principalmente debido a su gran popularidad y la abundancia de recursos de aprendizaje disponibles. Al ser uno de los lenguajes más utilizados en la industria, cuenta con una comunidad activa de desarrolladores que comparten su conocimiento y experiencia en línea. Esto facilita la resolución de problemas y el aprendizaje continuo durante el desarrollo de nuestras aplicaciones.

Además, Java es conocido por su rapidez, seguridad y confiabilidad, lo que lo hace adecuado para una amplia gama de proyectos. Es un lenguaje multiplataforma, orientado a objetos y centrado en la red que se puede utilizar como una plataforma en sí mismo.

- **Spring-Boot:** Spring es una herramienta que viene a solucionar el problema y complejidad de crear proyectos en Java, es una tecnología para crear aplicaciones que son autocontenidas. Que de este modo, el equipo se puede enfocar directamente en el desarrollo a medida, sin tener mucho en cuenta la arquitectura, Es decir, se delega en spring boot las tareas de configuración de las dependencias y el despliegue del servicio de las aplicaciones. Spring Boot agiliza el proceso utilizando internamente un servidor embebido, que por lo visto el más común es Tomcat. También siguiendo con la investigación, contiene un gestor robusto de dependencias y facilita la creación de aplicaciones independientes.

Consideramos estas opciones para el Backend porque son los lenguajes con los que tenemos una mayor afinidad y experiencia. Conocemos estas tecnologías más a fondo que otras existentes y contamos con cierta experiencia trabajando con ellas. Esto nos permite desarrollar nuestro proyecto de manera más eficiente y sin tener que aprender otros lenguajes desde cero.

- **ORM:** Es una técnica de programación que mapea objetos de un lenguaje de programación orientado a objetos a estructuras de datos relacionales en una base de datos relacional. Esto permite interactuar con la base de datos utilizando objetos y métodos familiares en lugar de consultas SQL directas.

La velocidad en un ORM se basa en la eficiencia de la generación de consultas SQL, la optimización de consultas, el manejo de la caché y cómo gestiona los cambios. También depende de cómo esté estructurada la base de datos y las relaciones entre los datos.

Los ORMs que consideramos utilizar son:

- Entity Framework (.NET)
- Dapper (.NET)

Entity Framework proporciona una capa de abstracción más completa y permite el desarrollo rápido de aplicaciones, mientras que Dapper ofrece un rendimiento más rápido y un control más granular sobre las consultas SQL generadas.

De estas tecnologías tenemos experiencia previa, por eso la consideración principal por ellas. Además, ambos ORMs están integrados de manera nativa en el ecosistema de .NET, lo que facilita su implementación, y cuentan con una extensa documentación y una comunidad activa de desarrolladores que brindan soporte.

- **DAO:** es un patrón de diseño utilizado en el desarrollo de software para abstraer y encapsular el acceso a la base de datos. Su objetivo principal es proporcionar una capa de abstracción entre el código de la aplicación y la base de datos, lo que facilita la gestión y el mantenimiento de la lógica de acceso a los datos.

La relación entre DAO y ORM es que ambos están relacionados con el acceso a los datos en una aplicación, pero tienen diferentes responsabilidades y enfoques:

- El ORM se encarga de mapear objetos de la aplicación a la base de datos y viceversa, mientras que el DAO se encarga de encapsular las operaciones de acceso a los datos.
- El ORM proporciona una abstracción de más alto nivel al mapear objetos de la aplicación a la base de datos, mientras que el DAO proporciona una abstracción más baja al encapsular operaciones específicas de acceso a los datos.

Motores de Bases de Datos

- **SQL Server:** es un sistema de gestión de bases de datos relacional desarrollado por Microsoft. Ofrece una plataforma robusta y escalable para almacenar y administrar datos de manera eficiente, lo cual es muy útil para lo que estamos buscando.

Decidimos considerar esta opción porque esta herramienta es muy popular y hay mucha documentación en internet. Cuenta con un gran número de usuarios y una amplia comunidad de desarrolladores, lo que significa que hay una gran cantidad de recursos disponibles para ayudarnos a resolver problemas y hacer mejoras en la base de datos.

- **MySQL:** es un sistema de administración de bases de datos relacionales que permite almacenar datos en forma de tablas con filas y columnas. Se trata de un sistema ampliamente conocido en el que se basan numerosas aplicaciones web, sitios web dinámicos y sistemas integrados.

Investigando sobre esta herramienta, notamos algunas razones para trabajar con MySQL. En primer lugar, es conocido por ser fácil de usar, lo cual nos viene bien, ya que queremos una solución que sea fácil de implementar. Además, MySQL es muy escalable, lo que significa que podremos adaptarlo fácilmente a medida que nuestro proyecto crezca y necesite manejar más datos y usuarios, lo cual tenemos pensado que ocurra.

También tenemos en cuenta que MySQL es muy popular, lo que significa que hay un montón de recursos, documentación y ayuda disponibles en caso de que nos topemos con algún problema o desafío técnico.

- **PostgreSQL:** es un sistema de administración de bases de datos relacionales de objetos que aporta más flexibilidad en cuanto a tipos de datos, escalabilidad, simultaneidad e integridad de los datos.

A pesar de que no contábamos con tanta experiencia previa utilizando PostgreSQL como con SQL Server y MySQL, nos sentimos intrigados por las capacidades que ofrecía y decidimos considerarlo. La gran ventaja de esta herramienta es que es Open Source y multiplataforma, donde no tendríamos que renegar por el sistema operativo. También investigamos sobre su administrador de bases de datos, PgAdmin, el cual es sencillo de manejar para personas con poca experiencia en ella.

- **Database-First:** este enfoque consiste en armar la base de datos dentro del gestor y una vez ya el entorno de desarrollo está configurado y conectado a la base de datos, puede migrar la base de datos de SQL Server, a la aplicación. Donde genera la capa de modelado con las clases y relaciones equivalentes a la base de datos.
- **Code-First:** consiste en lo contrario a lo anterior, donde se desarrolla la capa de modelado de código en el entorno de desarrollo, estableciendo las relaciones y demás. Se migra todo el modelado a la base de datos relacional SQL Server.

La ventaja de estos dos últimos métodos es que permiten agilizar el vínculo entre la base de datos y el backend, además de que si el sistema escala, permiten seguir migrando los cambios en ambas direcciones y trazar las diferentes migraciones en el tiempo.

En cuanto a las diferencias entre bases de datos relacionales y de documentos, las primeras, como SQL Server, MySQL y PostgreSQL, ofrecen una estructura más rígida con tablas y relaciones definidas. Son ideales para datos estructurados y complejos. Por otro lado, las bases de datos de documentos, como MongoDB, son más flexibles y pueden almacenar datos no estructurados en documentos JSON.

En el caso de los motores de bases de datos, tuvimos en cuenta estos motores de bases de datos porque ya tenemos experiencia previa con bases de datos relacionales. Nos sentimos cómodos trabajando con ellas y tenemos un conocimiento más sólido de cómo funcionan en comparación con otros tipos de bases de datos.

Además, consideramos que las bases de datos relacionales son más adecuadas para nuestro proyecto debido a su capacidad para manejar relaciones complejas entre los datos y garantizar la integridad de los mismos.

Documentación API

- **Swagger:** es una herramienta de código abierto que permite documentar la API y las ventajas sobre la utilización son:
 - **Interfaz interactiva:** Proporciona una interfaz interactiva para la documentación de tu API. Esto significa que los usuarios pueden explorar toda la API y probar sus endpoints desde la documentación, viendo qué

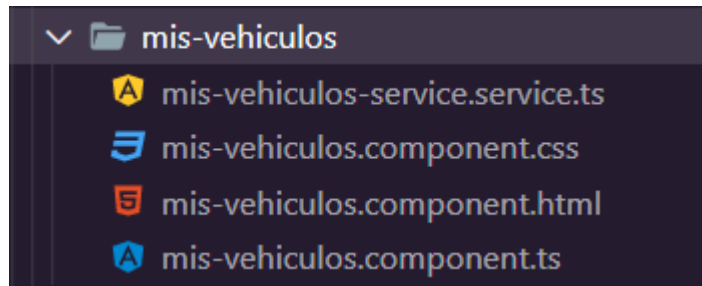
objeto va a devolver la respuesta, entre otros detalles. Es muy útil para los desarrolladores como para las personas que quieren entender el funcionamiento de los endpoints.

- **Generación automática de documentación:** Swagger genera automáticamente la documentación de la API a partir del código de la aplicación. Esto nos ahorra tiempo en comparación a la escritura manual de la documentación de cada endpoint. Swagger nos brinda qué necesita la request del endpoint, qué va a devolver la response, los tipo de datos de ambos, la URL, entre otros detalles.

Frontend

- **Angular:** Angular es un framework de desarrollo de aplicaciones web que nos permite construir interfaces de usuario robustas y dinámicas. Su enfoque en la modularidad y el rendimiento nos permite crear una experiencia de usuario fluida. Permite desarrollar en una arquitectura de módulos y componentes en la cual permite un desarrollo modular y atómico en la cual permite separar las diferentes responsabilidades en los diferentes módulos y componentes. Donde existen diferentes niveles de componentes, un nivel más alto general que renderiza los componentes de niveles más bajos a medida que demanda la renderización.

Es un framework muy robusto y de fácil escalamiento. Algunas de las características que fortalecen los atributos anteriores son la vinculación bidireccional de datos, que permite una sincronización entre el modelo y la vista. También el concepto de inyección de dependencias que es un concepto también utilizado en .NET (C#), que permite una mayor modularidad y reutilización de código. Por ejemplo: Un método de un servicio del componente x que conecta con un endpoint para recuperar algún dato, es utilizado por el componente x para poder luego renderizar en el HTML de dicho componente X.



Angular fue una de las opciones que consideramos desde un principio, ya que trabajamos con ella en proyectos previos y tenemos experiencia con la herramienta. Además, investigando un poco más en profundidad, notamos que tiene soporte de Google por lo que encontraremos un gran respaldo frente a problemas o dudas que aparezcan. Promete también realizar aplicaciones en menor tiempo de codificación, ya que TypeScript ayuda a Angular a identificar eficazmente los errores y a eliminarlos en una fase más temprana del ciclo de desarrollo que muchos otros frameworks.

Esto nos es de gran ayuda, ya que queremos realizar la aplicación de manera rápida y eficaz.

- **React:** También se nos ocurrió hacer el Frontend con React, ya que es más ligero y fácil de aprender a comparación con otras herramientas, pero creemos que Angular es más completa para el uso que le queremos dar al trabajar con un sistema grande.

React es una biblioteca de JavaScript que se utiliza para construir interfaces de usuarios, toda la aplicación web React se compone de componentes reutilizables que conforman partes de la interfaz de usuario. Podemos tener un componente distinto para nuestra barra de navegación y otro para el pie de página, etc. Tener estos componentes reutilizables facilita el desarrollo porque no debemos repetir el código reiterativo. Solo tendríamos que crear su lógica e importar el componente en la parte del código donde se necesite.

- **Angular Material:** Una vez que consideramos Angular como una opción sólida para nuestro proyecto, la elección de Angular Material vino de la mano. Cuando trabajamos previamente con Angular, utilizamos Angular Material, la cual es una herramienta que aporta gran valor al diseño de interfaces y nos ha dado grandes resultados.

Cuenta con una amplia variedad de componentes y estilos predefinidos, lo que nos permite diseñar interfaces atractivas y coherentes de manera rápida y eficiente.

Utilizamos las siguientes librerías:

- MapBox: Utilizaremos MapBox para la personalización y visualización del mapa de talleres. Nos permite mostrar ubicaciones geográficas, rutas y personalizar la apariencia del mapa de acuerdo a nuestras necesidades.
- FullCalendar: es una biblioteca de JavaScript que proporciona una interfaz de usuario para la visualización y gestión de eventos y calendarios en aplicaciones web.
- NgCharts: es una biblioteca de gráficos para Angular que permite crear visualizaciones de datos interactivas y personalizables en aplicaciones web.

Estas librerías ya las conocíamos de haber trabajado con ellas y nos resultaron un éxito, así que decidimos considerarlas para este proyecto también.

Arquitecturas

- **Arquitectura en capas:** La arquitectura en capas es un enfoque de diseño de software que organiza una aplicación en una serie de capas o niveles lógicos, donde cada capa tiene una responsabilidad específica y está separada de las demás. Este enfoque promueve la modularidad, la reutilización y la mantenibilidad del código al establecer límites claros entre las diferentes responsabilidades de la aplicación.

Muchas aplicaciones por lo que se vio adoptan este tipo de arquitectura ya que permite también a largo plazo, migrar a otra arquitectura de forma sencilla, como microservicios. Pero desde un primer momento, microservicios no fue una alternativa para el desarrollo.

- **Arquitectura monolítica:** La arquitectura monolítica es un estilo de arquitectura de software en el que una aplicación se desarrolla como una sola unidad o componente indivisible, tienen como característica el uso de una base de código única para sus servicios o funcionalidades. En este enfoque, todos los componentes y

funcionalidades de la aplicación están interconectados y desplegados juntos como una sola entidad en un entorno de ejecución. La aplicación se despliega como un único archivo o un conjunto de archivos interdependientes. Todos los aspectos funcionales, como la lógica de negocio, interfaz de usuario, acceso a datos y demás servicios, están empaquetados y ejecutados dentro del mismo proceso.

La desventaja que identificamos en la investigación de la misma es que a largo plazo, genera mucho acoplamiento y es difícil de mantener, puede volverse muy compleja.

- **MVC (Modelo-Vista-Controlador):** El patrón de diseño Modelo-Vista-Controlador es una forma de estructurar una aplicación de software que separa la lógica de negocio (Modelo), la presentación de la interfaz de usuario (Vista) y el control de la interacción entre el modelo y la vista (Controlador). En este patrón, el modelo representa los datos y la lógica de negocio de la aplicación, la vista representa la interfaz de usuario que muestra los datos al usuario y el controlador actúa como intermediario entre el modelo y la vista, gestionando las solicitudes del usuario y actualizando el modelo en consecuencia. Esta separación de responsabilidades facilita la escalabilidad, la mantenibilidad y la prueba de la aplicación.

Esta arquitectura por lo que investigamos, se puede aplicar a partir de un cierto marco en diferentes tecnologías. Si bien sigue un hilo de implementación, el mismo varía según la tecnología.

Despliegue

- **Vercel:** es una plataforma en la nube que ofrece servicios de alojamiento y despliegue para aplicaciones web y sitios estáticos. Vercel se especializa en proporcionar un entorno de desarrollo rápido y eficiente para desarrolladores que trabajan con tecnologías web modernas.
Vercel es una de las pocas plataformas gratuitas que nos permiten alojar nuestro sistema, por esa razón es una fuerte opción a utilizar en nuestro proyecto.
- **Somee:** es un proveedor de servicios de alojamiento web que ofrece opciones de alojamiento compartido y dedicado para sitios web. Somee proporciona una

plataforma para que los usuarios puedan alojar sus sitios web y aplicaciones web en servidores gestionados por la empresa.

Patrones

- **El patrón repositorio:** Se encarga de abstraer la lógica que viene de la mano de la arquitectura implementada (Capas), donde accede a la capa de negocio y permite que sea más agnóstica o que no tenga conocimiento directo o dependencia a una implementación concreta.
- **Inversión de dependencias:** Permite que los módulos de alto nivel no deben depender de los módulos de bajo nivel. Ambos dependen de abstracciones.
- **Segregación de interfaces:** Los clientes no deben ser forzados a depender de interfaces que no utilizan, por lo que tanto para realizar tests y disminuir el acoplamiento, la idea es segregar las interfaces por significado y lógica de negocio que cumplen.
- **Inyección de dependencias:** Es un patrón de diseño donde las dependencias de las clases no se crean, sino que se proporcionan externamente. Reduce notablemente el acoplamiento, brinda más flexibilidad , ayuda a la gestión de las dependencias y a la realización de pruebas unitarias. Para dar un ejemplo, supongamos que creamos múltiples implementaciones de el metodo nombrado en una interface ICalculadorImpuestos, pero tenemos un cálculo para impuestos basados en un país y otra para estados. La clase que lo utilice, no debe saber que implementación utilizará, simplemente usará la interfaz. Esto permite que si se modifica la implementación concreta, no haya que modificar la clase que los utiliza.
- **Principio de responsabilidad simple:** En cada una de las clases o módulos de las capas debe tener solo una razón para cambiar, es decir cumplir una única función de negocio o lógica de negocio.

Chatbot

A raíz de los problemas por falta de comunicación entre talleres y clientes, investigamos herramientas que permitan mejorar la interacción entre ambas partes, y optimizar la atención al cliente. Queremos que las dos entidades se vean beneficiadas, tanto el cliente ayudando a despejar sus dudas en todo momento, como los talleres, evitándoles perder tiempo respondiendo mensajes por celular y utilizando su tiempo para revisar y reparar vehículos.

En primer lugar, hay que tener en claro el concepto de Chatbot. Estas son aplicaciones software que simulan mantener una conversación con una persona al proveer respuestas automáticas, las cuales son previamente establecidas por un conjunto de expertos a entradas realizadas por el usuario.

Hay varias opciones para crear un chatbot. Algunas plataformas ofrecen bots ya hechos por otras personas que puedes usar directamente para tus necesidades. Estos bots ya están programados y listos para funcionar, y cabe destacar que son pagos. Por ejemplo, DialogFlow o ManyChat.

Por otro lado, también hay librerías gratuitas y herramientas que te permiten programar tu propio chatbot desde cero. Esto te da más libertad para personalizar el chatbot según tus necesidades específicas. Por ejemplo, las librerías de Telegram y Whatsapp.

Whatsapp te brinda una API que permite a las empresas integrar funciones de chat en sus aplicaciones y sistemas existentes. Esto les permite interactuar con los clientes a través de la plataforma de WhatsApp. Esta API es únicamente utilizable por empresas que cumplan con ciertos requisitos, como ser una entidad verificada.

Por otra parte, Telegram es una aplicación que está adaptada a la fácil creación y gestión de Chatbots a través de BotFather, que permite a los desarrolladores crear nuevos bots, configurar sus opciones y obtener tokens de autenticación para su uso en la API. Adicionalmente a esto, existen librerías en Github que ayudan a complementar el bot para programarlo a tus necesidades.

Pruebas

En primer lugar, investigamos y nos adentramos informándonos sobre el testing en el desarrollo de una aplicación de software. Lo que nos topamos fue que existen distintos tipos de testing o pruebas de software que permiten validar que nuestro sistema funciona correctamente luego de generar alguna implementación o al correr cambios nuevos sobre nuestro código.

Cabe destacar que cuando uno trabaja en proyectos pequeños e individuales, muchas veces no existe dicha necesidad de escribir estas pruebas. Pero en proyectos grandes y colaborativos, debemos escribir tests.

Para tener en cuenta, los equipos se organizan para ejecutar conjuntos de pruebas. Estos conjuntos los llamamos “Test Suite”, engloban el contexto que va a incluirse en la prueba que se va a llevar a cabo.

Existen diferentes tipos de tests:

- **Test Unitarios:** son a bajo nivel, es decir están cercanas al código fuente de nuestra aplicación y consiste en probar de forma individual las funciones y/o métodos de las diferentes clases o componentes que sean utilizados por nuestro sistema

Algunos detalles que encontramos en nuestra investigación sobre las pruebas unitarias son:

- Cuando las planeamos, debemos aislar la funcionalidad hasta un punto que no se pueda desglosar más.
- Estas pruebas verifican que el nombre de la función o el método sea adecuado y descriptivo, que los parámetros sean correctos, lo mismo para el valor de retorno.

TDD: sus siglas Test Driven Development, hace referencia a unit test. Es decir se usan pruebas de este tipo como especificaciones de lo que nuestro código debe hacer.

Pruebas End-to-End: Estas pruebas replican el comportamiento de los usuarios con el software, en un entorno de aplicación completo. Estas pruebas verifican que los flujos que sigue un usuario trabajen como se espera, y pueden ser tan simple como esperar cargar el módulo de inicio de sesión o tan complejas como verificar notificaciones vía mail.

Estas son útiles, pero pueden ser difíciles de mantener si es que se quiere automatizar. En nuestro caso no planeamos automatizar y creemos que es un tipo de prueba que puede servirnos y aportarnos valor.

Pruebas de Regresión: Las pruebas de regresión verifican que todos los escenarios de la aplicación funcione correctamente. Es decir, a medida que van pasando etapas del desarrollo, es importante que en cada incremento, se realice una prueba de regresión para ver que lo desarrollado anteriormente con lo nuevo funcione.

Algunos detalles que rescatamos en la investigación de este tipo de prueba:

- Si las pruebas de regresión actuales no pasan, no hay que agregar nuevas características.
- Una falla en una prueba de regresión, significa que una nueva funcionalidad afecto a otra que era correcta en el pasado. Esto también es una clara referencia de que hubo malas prácticas en la implementación del código y su arquitectura. Por lo que es una característica que vamos a tener en cuenta.
- Una falla en un test de regresión indica que hemos vuelto a producir un bug que ya había sido resuelto en el pasado.

Pruebas Exploratorias: El objetivo es descubrir errores que no son obvios. Una sesión de pruebas exploratorias no debería llevar más de 2 horas y es necesario tener bien definido el alcance, para poder ayudar a la gente encargada a centrarse en un área específica del software. Entonces, la idea es segmentar las personas a que realicen pruebas sobre ciertos módulos de la aplicación para verificar el correcto funcionamiento del sistema.

Resulta útil principalmente cuando se agrega una nueva característica importante a nuestra aplicación, para ayudar a comprender cómo se comporta en casos extremos.

Luego de esta investigación teórica adentrándonos un poco en el concepto de testing en lo que es el desarrollo de productos de software, nos inclinamos a analizar algunas herramientas con las cuales podremos realizar los unit test en nuestro futuro desarrollo, en base a la tecnología que decidamos utilizar para desarrollar el backend.

Para el backend en:

- **.NET:**

- **NUnit:** Es un marco de prueba openSource que es transportado por Junit. Esta última versión cuenta con soporte para una amplia gama de plataformas .NET, es una gran herramienta para llevar a cabo TDD. Esta herramienta permite la ejecución de pruebas tanto en serie como paralelas (hasta 4).
- **XUnit.Net:** es un marco de pruebas también openSource basado en el marco .NET. Significa que los creadores de NUnit crearon xUnit. Ya que querían segmentar las características asociadas al lenguaje de .Net y no incrementar NUnit. XUnit está creado con más enfoque en la comunidad.
- **Moq:** Es una de las bibliotecas más populares dentro del entorno de desarrollo en .NET, que permite crear objetos o “mocks” ficticios que usamos para escribir código de test. Es útil ya que muchas veces nuestras funcionalidades trabajan con objetos de terceros o acciones con acceso a base de datos, lo que queremos es simular el comportamiento del objeto real mediante la creación de un objeto ficticio.

- **Java:**

- **JUnit 4:** Framework Java que permite implementar Test unitario en Java. Se basa en anotaciones. No se escapa mucho del concepto de testing que lleva a cabo .NET.
- **Mockito:** Es una librería Java que permite simular el comportamiento de una clase de manera dinámica. La idea es aislar de las dependencias con otras clases y testear únicamente la funcionalidad concreta que se requiera.

PROPUESTA DE SOLUCIÓN

Basándonos en el análisis que realizamos de la problemática y los desafíos identificados en la gestión de turnos de talleres mecánicos, hemos desarrollado una sólida propuesta de solución que busca transformar la manera en que los clientes interactúan con los talleres y optimizar la organización de los servicios de reparación y mantenimiento.

Nuestra solución se centra en la creación de un sistema web de gestión de turnos, con el objetivo de mejorar la eficiencia, la transparencia y la comunicación entre clientes y talleres. Además de poder permitir a los talleres mejorar su organización y eficientizar su tiempo.

Alcance Funcional

El alcance funcional del proyecto principalmente viene dado desde que el cliente solicita un turno en el taller que seleccione hasta que el cliente retira el vehículo del taller con el procedimiento realizado, registrando este evento en el sistema. Dentro de este proceso hay que pasar por distintas tomas de decisiones de ambas entidades, decidiendo aceptar o rechazar ciertas cuestiones.

Básicamente el alcance está basado en el estado del turno del cliente, desde que se solicita hasta que se encuentra en “Finalizado”, pasando por distintos estados a lo largo del turno.

Etapas

Planificación y análisis del proyecto

- Planteo de la idea.
- Construcción de documento Ante Proyecto.
- Análisis construcción y definición de requerimientos de potenciales clientes.
- Investigación de tecnologías para construcción de ChatBot. Análisis de posible módulo estadístico.
- Definición de alcance. Construcción de Gantt.

Definición Ante Proyecto

- Planificación y análisis del proyecto.

Confección Libro de Tesis

- Desarrollo del libro de tesis.
- Glosario.
- Objetivos globales y específicos.
- Desarrollo del marco teórico.
- Desarrollo de propuesta de solución.
- Investigación de impactos.

Etapas de diseño

- Diseño de modelo de entidad relación.
- Definir arquitectura de backend.
- Definir los prototipos de las vistas.
- Confección de libro documentación.

Etapas de desarrollo

- Configurar script de build y startup de la API (backend).
- Construcción de la capa de modelado de datos.
- Desarrollo de capa de modelos.
- Construcción de la capa de acceso a datos.
- Desarrollo del contexto de base de datos.
- Desarrollo de interfaces e implementación.
- Construcción de la capa de lógica de negocio.
- Desarrollo de interfaces e implementación.
- Desarrollo de excepciones personalizadas.
- Construcción de la capa de controlador.
- Desarrollo de controladores y definición de endpoints.
- Desarrollo de los unit test.

- Creación de proyecto del frontend.
- Creación de módulos y componentes.
- Creación de módulo de autenticación.
- Desarrollo de componentes de registro e inicio de sesión.
- Creación de módulo Cliente.
- Desarrollo de componente o módulo de sidebar y navbar.
- Desarrollo de componente home.
- Desarrollo de componentes de turnos y mis vehículos (solicitar, estado de turnos, calendario de turno, centro de ayuda, mis vehículos).
- Desarrollo de módulo de Talleres.
- Desarrollo de componente de turnos (homepage, gestión de turnos, calendario, historial de turnos, gestión de inventario, centro de ayuda).
- Desarrollo de componente de Estadística.
- Desarrollo de componente Calendarios.
- Desarrollar componente de Centro de ayuda.
- Creación de módulo de Administrador.
- Creación de componentes de administrador (homepage, usuarios, centro de ayuda).
- Confección de flujos de conversación ChatBot.

Etapas de prueba

- Probar endpoints por separado.
- Detectar fallos.
- Corregir fallos.
- Volver a probar el código corregido.

Etapas de despliegue

- Investigar la herramienta de despliegue de la API.
- Investigar la herramienta de despliegue del front.
- Investigar la herramienta de despliegue de la DB.
- Desplegar software.
- Corroborar los 3 entornos desplegados.

Modelado de Requerimientos

Hemos definido los requerimientos que se contemplaban para el desarrollo del sistema de las tres entidades que lo abarcan: Cliente, Taller, Administrador. Además planteamos los requerimientos necesarios para el Chatbot integrado al sistema.

Cliente

1. Consultar el listado de talleres asociados con el sistema:

- El sistema debe mostrar una lista de talleres asociados al cliente.

2. Consultar en mapa los talleres asociados por el sistema:

- El sistema debe mostrar un mapa interactivo con la ubicación de los talleres asociados al cliente.
- El cliente debe poder observar la ubicación de los talleres en el mapa.
- El cliente debe poder acceder a la ubicación de cada taller.

3. Buscar por nombre de taller:

- El sistema debe permitir al cliente buscar talleres por su nombre.
- El sistema debe mostrar una lista de resultados de búsqueda y permitir al cliente seleccionar el taller deseado.

4. Solicitar un turno(-):

- El sistema debe permitir al cliente solicitar un turno en un taller determinado.
- El cliente debe llenar un formulario con los datos correspondientes al turno.
- El sistema debe confirmar al cliente que la solicitud de turno ha sido recibida y mostrarle la información del mismo (fecha, hora, taller, etc.).

5. Consultar estado de turnos(-):

- El sistema debe permitir al cliente consultar el estado de sus turnos.
- El sistema debe mostrar al cliente información actualizada sobre la fecha, hora y estado de su turno.
- El sistema debe notificarle al cliente en caso de rechazar el turno adjuntando un motivo.

6. Cancelar un turno(-):

- El sistema debe permitir al cliente cancelar un turno previamente solicitado.
- El sistema debe confirmar al cliente que el turno ha sido cancelado.

7. Agregar múltiples vehículos a su cuenta y administrar turnos para cada uno:

- El sistema debe permitir a los clientes agregar múltiples vehículos a su cuenta.

- El cliente debe poder administrar turnos para cada vehículo individualmente.

8. Calendario

- El sistema debe permitir al cliente visualizar en el calendario, la fecha y datos de el turno solicitado.

Talleres

1. Consultar turnos pendientes

- El sistema debe permitir a los talleres ver los turnos pendientes que han sido solicitados.
- Los talleres deben poder acceder a esta información desde su panel de control.

2. Tomar turno(-).

- El sistema debe permitir a los talleres aceptar un turno solicitado por un cliente.
- El sistema debe agendar ese turno en una sección de turnos aceptados por el taller.
- El sistema debe tener la opción de seleccionar la fecha en la que se va a atender el turno.
- El sistema debe permitir a los talleres determinar el estado en el que se encuentra el turno.
- El sistema debe mostrar el motivo del estado de cancelación de un turno.

3. Calendario:

- El taller debe poder crear registros y establecer fechas de entrega de los turnos atendidos

4. Ver historial de turnos resueltos:

- El sistema debe permitir al taller ver el historial de los turnos resueltos.
- El listado de turnos debe mostrar información relevante como el nombre, apellido, estado del turno, la marca del auto, el modelo, número de turno, fecha de emisión, fecha estimada de entrega y tipo de servicio.

5. Gestión de inventario:

- El sistema debe permitir al taller llevar un registro del inventario de repuestos y herramientas.

6. Ver estadísticas de servicios:

- El sistema debe permitir al taller ver estadísticas de los servicios que atendió.
- El sistema debe mostrar información relevante como la cantidad de servicios realizados, el tiempo promedio de reparación, los ingresos generados por cada servicio, la cantidad de turnos por tipo, la cantidad de turnos por estado, el número de empleados activos, la cantidad de piezas en inventario.

7. Notificaciones en tiempo real:

- El sistema debe permitir al taller notificar a los clientes cuando se actualiza el estado de su turno.
- Las notificaciones deben ser entregadas en tiempo real mediante correo electrónico.

Administrador

1. Dar de alta a usuarios:

- El sistema debe permitir a los administradores dar de Alta usuarios con el rol de taller y empleados de un taller determinado..
- El sistema debe mostrar registros de todos los usuarios tanto cliente como talleres.
- El sistema debe permitir modificar, eliminar datos de los usuarios.
- El sistema debe permitir ver detalles de los usuarios.
- El sistema debe permitir buscar usuarios por nombre y apellido.
- El sistema debe permitir filtrar por rol los usuarios, tanto tipo cliente como taller
- El sistema debe permitir añadir empleados a un taller determinado.
- El sistema debe permitir eliminar y modificar datos de un empleado.
- El sistema debe permitir determinar si ese empleado está activo o no.
 - En caso de vacaciones se setea inactivo para que el ICT indique la proporción de carga de trabajo correcta.

ChatBot

1. Asistencia automatizada

- El sistema debe ofrecer al cliente asistencia de manera estructurada para la gestión de turnos.
- El sistema debe brindar al cliente una serie de opciones con la finalidad de filtrar información.
- El sistema debe ser capaz de responder en un tiempo no mayor a 3 segundos a las solicitudes para evitar demoras con la información solicitada por el cliente.

Historias de Usuario

En base a los requerimientos, definimos las historias de usuario. Cada historia de usuario captura una necesidad específica de los clientes y los talleres, y se traduce en una funcionalidad concreta en nuestro sistema. Estas historias de usuario han sido desarrolladas en base a los requerimientos identificados, con el objetivo de ofrecer una experiencia satisfactoria para todas las partes involucradas.

Cliente

1. Consultar el listado de talleres asociados con el sistema:

- Como cliente quiero ver una lista de los talleres a los cuales estoy asociado para ver determinada información.
- Como cliente quiero poder ordenar la lista de los talleres a los que estoy asociado por diferentes criterios (nombre, calificación, etc.) para facilitar mi búsqueda.

2. Consultar en mapa los talleres asociados por el sistema:

- Como cliente quiero poder ver un mapa interactivo con la ubicación de los talleres a los cuales estoy asociado para corroborar cual se encuentra más cerca de donde estoy.
- Como cliente quiero poder buscar talleres en un área determinada del mapa para poder encontrar fácilmente talleres cercanos a mi ubicación actual.
- Como cliente quiero poder acceder a la ubicación de cada taller para saber donde se encuentran.

3. Buscar por nombre de taller:

- Como cliente quiero poder buscar talleres por su nombre para poder encontrar rápidamente el taller en el que estoy interesado.
- Como cliente quiero poder ver una lista de talleres que coincidan con el nombre que ingresé en el sistema para poder seleccionar el taller deseado.

4. Solicitar un turno(-):

- Como cliente quiero poder solicitar un turno en un taller determinado para que revisen mi vehículo.
- Como cliente quiero poder llenar un formulario con los datos correspondientes al turno para poder agendar una cita para mi vehículo de manera precisa.
- Como cliente quiero recibir una confirmación de que mi solicitud de turno ha sido recibida y ver la información relevante del mismo para poder estar seguro de que mi solicitud ha sido registrada correctamente.

5. Consultar estado de turnos(-):

- Como cliente quiero poder consultar el estado de mis turnos para saber en qué estado se encuentran.
- Como cliente quiero poder ver información actualizada sobre la fecha, hora y estado de mi turno en el sistema para poder planificar mi tiempo y asegurarme de que mi turno ha sido registrado correctamente.

6. Cancelar un turno(-):

- Como cliente quiero poder cancelar un turno previamente solicitado en el sistema para poder liberar el horario y permitir que otro cliente reserve ese mismo turno.
- Como cliente quiero recibir una notificación en caso de que mi turno sea rechazado para conocer el motivo del mismo.

7. Agregar múltiples vehículos a su cuenta y administrar turnos para cada uno:

- Como cliente quiero poder agregar múltiples vehículos a mi cuenta para
Como cliente quiero poder administrar los turnos para cada vehículo
administrar cada uno de ellos con sus respectivos turnos.
- individualmente para mantener organizado mis vehículos con sus respectivos turnos.

8. Calendario

- Como cliente quiero poder visualizar un calendario donde se encuentren mis turnos para saber las fechas y horarios de los mismos.

Talleres

1. Consultar turnos pendientes(-)

- Como taller quiero poder acceder a una lista de turnos pendientes desde mi panel de control en el sistema para poder verlos de manera rápida y gestionar los servicios que aún deben ser atendidos.

2. Tomar turno(-).

- Como taller quiero poder aceptar un turno solicitado por un cliente en el sistema para poder confirmar mi disponibilidad para brindar el servicio.
- Como taller quiero poder agendar el turno aceptado en una sección de turnos aceptados en el sistema para poder mantener un registro organizado de los turnos pendientes de atender.
- Como taller quiero poder seleccionar la fecha en la que se va a atender el turno aceptado en el sistema para poder programar mi trabajo y cumplir con los plazos acordados con el cliente.
- Como taller quiero poder establecer el estado de un turno como "pendiente", para indicar que aún no se ha confirmado una fecha y hora para el servicio solicitado.
- Como taller quiero poder establecer el estado de un turno como "cancelado" y enviar el motivo de la cancelación, para indicar que el servicio solicitado no se llevará a cabo y proporcionar una explicación a mi cliente.
- Como taller quiero poder establecer el estado de un turno como "aceptado", para indicar que el servicio solicitado ha sido confirmado y se llevará a cabo en la fecha y hora acordadas.
- Como taller quiero poder establecer el estado de un turno como "en proceso", para indicar que el servicio solicitado está en progreso y aún no se ha completado.

- Como taller quiero poder establecer el estado de un turno como "demorado", para indicar que el servicio solicitado se está llevando a cabo pero hay un retraso en su finalización.

3. Calendario:

- Como taller quiero poder crear registros y establecer fechas de entrega de los turnos atendidos para poder mantener una organización coherente de mi negocio.

4. Ver historial de turnos resueltos(-):

- Como taller quiero poder ver el historial de los turnos resueltos para tener acceso a información relevante sobre los servicios realizados.
- Como taller quiero que el detalle de cada turno resuelto muestre información relevante como el problema detectado, la solución aplicada, los repuestos utilizados, los gastos y el costo de reparación para tener una visión detallada del servicio realizado.

5. Gestión de inventario(-):

- Como taller, quiero que el sistema permita llevar un registro del inventario de repuestos y herramientas, para poder saber qué recursos están disponibles en cualquier momento y realizar un seguimiento de los mismos.

6. Ver estadísticas de servicios:

- Como taller quiero poder acceder a estadísticas de los servicios que atendí para tener en cuenta esa información para futuras solicitudes de turnos.
- Como taller quiero que el sistema muestre información relevante sobre los servicios más comunes, como la cantidad de servicios realizados, el tiempo promedio de reparación y los ingresos generados por cada servicio , para poder mejorar los servicios en base a estadísticas.

7. Notificaciones en tiempo real(-):

- Como taller quiero poder notificar a los clientes cuando se actualiza el estado de su turno, para mantenerlos informados sobre cualquier cambio y evitar confusiones.
- Como taller quiero que las notificaciones se entreguen en tiempo real mediante correo electrónico o mensaje de texto, para garantizar que los clientes reciban la información lo antes posible.

Administrador

1. Dar de alta a usuarios:

- Como administrador quiero poder dar de alta usuarios con el rol de cliente, para que puedan acceder a los servicios ofrecidos por el sistema.
- Como administrador quiero poder dar de alta usuarios con el rol de taller, para que puedan acceder a las herramientas y recursos necesarios para realizar su trabajo.
- Como administrador quiero poder modificar los datos de los usuarios de tipo cliente para mantener actualizada su información en el sistema.
- Como administrador quiero poder modificar los datos de los usuarios de tipo taller para mantener actualizada su información en el sistema.

ChatBot

1. Asistencia automatizada

- Como cliente quiero recibir la información que solicité, para poder tener una experiencia más fluida y sin inconvenientes al utilizar el sistema.
- Como taller quiero que el sistema brinde opciones de filtrado al cliente para obtener información precisa y relevante sobre su problema con el fin de ofrecer un servicio más eficiente y efectivo.
- Como cliente quiero que el sistema de gestión de turnos responda con una velocidad no mayor a 3 segundos a mis solicitudes para poder realizar mis gestiones de manera ágil y sin demoras.

Casos de Uso

Cliente

1)

Caso de Uso		Consultar el listado de talleres asociados con el sistema
Descripción		El cliente registrado consulta por los talleres asociados al sistema presionando el botón "Ver talleres".
Precondiciones		- El cliente debe estar registrado. - Debe haber talleres asociados al sistema.
Secuencia de Pasos		Alternativas
1	El cliente ingresa al sistema.	El sistema no encuentra talleres asociados al sistema. El sistema genera un error con el mensaje "No hay talleres asociados al sistema" El sistema le indica al usuario que vuelva a intentarlo cuando haya talleres asociados al sistema. Continúa en el paso(5)
2	El cliente se encuentra en la vista inicial.	
3	El cliente presiona el botón "Ver talleres".	
4	El sistema verifica que haya talleres asociados al sistema.	
4.1		
4.2		
4.3		
4.4		
5	El sistema muestra una lista de talleres asociados al sistema ordenados por cercanía del usuario.	
Postcondiciones		- El cliente puede buscar taller por nombre - El cliente puede seleccionar un taller para ver su ubicación en el mapa

2)

Caso de Uso		Buscar taller por nombre	
Descripción		El cliente registrado puede buscar talleres asociados al sistema por su nombre	
Precondiciones		- El cliente debe estar registrado. - Debe haber talleres asociados al sistema.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa al sistema.	El sistema no encuentra talleres asociados al sistema. El sistema genera un error con el mensaje “No hay talleres asociados al sistema” El sistema le indica al usuario que vuelva a intentarlo cuando haya talleres asociados al sistema. Continua en el paso(5)	
2	El cliente se encuentra en la vista inicial.		
3	El cliente presiona el botón “Ver talleres”.		
4	El sistema verifica que haya talleres asociados al sistema.		
4.1			
4.2			
4.3			
4.4			
5	El sistema muestra una lista de talleres asociados al sistema ordenados por cercanía del usuario.		
Postcondiciones		- El cliente puede buscar taller por nombre - El cliente puede seleccionar un taller para ver su ubicación en el mapa	

3)

Caso de Uso		Revisar estado de turnos	
Descripción		El cliente registrado visualiza y es capaz de gestionar sus turnos al apretar la sección "Estado de turnos".	
Precondiciones		• El cliente debe estar registrado. • Debe haber talleres asociados al sistema. • El cliente debe tener turnos en algún estado.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa al sistema.	El sistema no encuentra turnos asociados al cliente. El cliente no puede visualizar sus turnos. El cliente debe solicitar un turno y ser aceptado por un taller. Continúa en el paso (4)	
2	El cliente ingresa a la sección "Estado de turnos".		
3	El sistema verifica que el cliente tenga turnos.		
3.1			
3.2			
3.3			
3.4			
4	El sistema muestra una lista de los turnos asociados al cliente.		
5	El cliente revisa el estado de sus turnos.		
Postcondiciones		• El cliente puede visualizar el estado de sus turnos. • El cliente puede ver los cambios de estado de sus turnos.	

4)

Caso de Uso		Solicitar un turno	
Descripción		El cliente registrado solicita un turno a un taller asociado al sistema.	
Precondiciones		• El cliente debe estar registrado. • Debe haber talleres asociados al sistema. • El cliente debe tener por lo menos un auto asociado a su cuenta.	
Secuencia de Pasos			Alternativas
1	El cliente ingresa al sistema.	El sistema verifica que el cliente no tiene vehículos asociados a su cuenta. El cliente debe añadir un vehículo a su cuenta. Continúa en el paso (6)	
2	El cliente ingresa a la sección “Solicitar un turno”.		
3	El cliente visualiza una serie de pasos a completar, como primer paso seleccionar un taller.		
4	El cliente selecciona un taller para solicitar un turno.		
5	El cliente selecciona un vehículo asociado a su cuenta.		
5.1			
5.2			
5.3			
6	El cliente selecciona el tipo de turno adecuado a su problema.		
7	El cliente visualiza información final del turno antes de confirmar la solicitud.		
8	El cliente confirma la solicitud de turno.		
9	El sistema verifica la solicitud y la da como confirmada.		
Postcondiciones		• El cliente puede visualizar el estado de su solicitud de turno. • El estado del turno pasa a estar “Pendiente”. • El sistema recibe la solicitud de turno, y decide si aprobarla o rechazarla.	

5)

Caso de Uso		Añadir vehículo	
Descripción		El cliente registrado agrega un vehículo a su cuenta al entrar a la sección “Mis vehículos”.	
Precondiciones		- El cliente debe estar registrado. - Debe haber talleres asociados al sistema.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa al sistema.	El sistema verifica que la información agregada por el cliente no cumple con las reglas de validación. El cliente escribe correctamente la información. Continúa en paso (7)	
2	El cliente ingresa a la sección “Mis vehículos”.		
3	El cliente aprieta el botón “Añadir vehículo”.		
4	El sistema solicita patente, marca, modelo y año del vehículo a añadir.		
5	El sistema verifica los datos ingresados		
6			
6.1			
6.2			
6.3			
7	El sistema añade con éxito el vehículo a la cuenta del cliente.		
Postcondiciones		- El cliente puede visualizar los vehículos añadidos a su cuenta. - El cliente puede visualizar el historial de turnos de sus vehículos.	

6)

Caso de Uso		Ver historial de turnos de vehículo	
Descripción		El cliente registrado visualiza el historial de turnos asociados a un vehículo.	
Precondiciones		• El cliente debe estar registrado. • Debe haber talleres asociados al sistema. • El cliente debe tener, al menos, un vehículo asociado a su cuenta.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa al sistema.	El sistema verifica que el cliente no tiene vehículos cargados a su cuenta. El cliente añade un vehículo invocando el caso de uso "Añadir vehículo". Continúa en paso (4)	
2	El cliente ingresa a la sección "Mis vehículos".		
3	El sistema verifica que el cliente tiene cargado en su cuenta un vehículo.		
3.1			
3.2			
3.3			
4	El cliente es capaz de visualizar el historial de turnos de sus vehículos.		
Postcondiciones		• El cliente puede visualizar el historial de turnos de sus vehículos.	

7)

Caso de Uso		Configurar perfil	
Descripción		El cliente registrado configura el perfil al apretar el ícono de perfil registrado.	
Precondiciones		El cliente debe estar registrado.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa al sistema.		
2	El cliente ingresa a la sección de “Mi perfil” apretando el ícono de perfil registrado.		
3	El cliente puede modificar su contraseña, DNI y email.		
Postcondiciones		El sistema actualiza la información modificada del cliente.	

8)

Caso de Uso		Comunicarse con Chatbot	
Descripción		El cliente se comunica con el Chatbot de Telegram	
Precondiciones		El cliente debe haberse registrado en el sistema anteriormente.	
Secuencia de Pasos		Alternativas	
1	El cliente ingresa a la app Telegram		
2	El cliente le escribe al Chatbot		
3	Si el cliente le habla por primera vez, el bot le solicita el mail para corroborar de que el cliente está registrado.		
4			
Postcondiciones		El Chatbot muestra la información solicitada por el cliente.	

Taller

1)

Caso de Uso		Aceptar turno	
Descripción		El taller registrado acepta la solicitud de turno por parte del cliente.	
Precondiciones		- El taller debe estar registrado. - El taller debe tener solicitudes de turnos pendientes	
Secuencia de Pasos		Alternativas	
1	El taller ingresa al sistema.	El sistema verifica que el taller no tiene solicitudes de turnos pendientes. El taller debe recibir nuevas solicitudes de turnos. Continúa en paso (4)	
2	El taller ingresa a la sección “Gestión de turnos”.		
3	El sistema muestra las solicitudes de turnos pendientes del taller.		
3.1			
3.2			
3.3			
4	El taller selecciona un turno y aprieta el botón “Tomar turno” para aceptar la solicitud.		
5	El sistema genera una alerta para confirmar la aceptación de la solicitud de turno.		
6	El taller confirma la aceptación de la solicitud.		
Postcondiciones		- El sistema actualiza la información modificada del cliente. - El turno pasa a estar en estado “En revisión”.	

2)

Caso de Uso		Rechazar turno	
Descripción		El taller registrado rechaza la solicitud de turno por parte del cliente.	
Precondiciones		• El taller debe estar registrado. • El taller debe tener solicitudes de turnos pendientes	
Secuencia de Pasos		Alternativas	
1	El taller ingresa al sistema.	El sistema verifica que el taller no tiene solicitudes de turnos pendientes. El taller debe recibir nuevas solicitudes de turnos. Continúa en paso (4)	
2	El taller ingresa a la sección “Gestión de turnos”.		
3	El sistema muestra las solicitudes de turnos pendientes del taller.		
3.1			
3.2			
3.3			
4	El taller selecciona un turno y aprieta el botón “Rechazar turno” para rechazar la solicitud del turno.		
5	El sistema genera una alerta para confirmar el rechazo de la solicitud de turno.		
6	El sistema le solicita al taller un motivo de rechazo para enviar al cliente		
7	El sistema verifica que el taller completó el campo obligatorio de motivo de rechazo.		
7.1		El sistema verifica que el taller no completó el campo obligatorio de motivo de rechazo.	
7.2		El sistema solicita nuevamente completar el campo de motivo de rechazo	
7.3		El taller completa el campo de datos solicitado	

7.4	El taller confirma el rechazo de la solicitud adjuntando un motivo.	Continúa en paso (8)
8		
Postcondiciones		El sistema notifica al cliente que la solicitud de turno fue rechazada.

3)

Caso de Uso	Añadir nueva pieza	
Descripción	El taller registrado añade una nueva pieza a su inventario.	
Precondiciones	El taller debe estar registrado.	
Secuencia de Pasos		Alternativas

1	El taller ingresa al sistema.	
2	El taller se encuentra en la vista inicial.	
3	El taller presiona el botón “Gestión de inventario”.	
4	El taller visualiza su stock de piezas.	
5	El taller añade una nueva pieza al inventario apretando el botón “Añadir nueva pieza”.	
6	El sistema solicita al taller añadir nombre, descripción, cantidad y precio por unidad de la pieza a añadir.	
7	El taller completa los datos correspondientes.	
8	El sistema verifica que se completaron los campos obligatorios solicitados al taller.	
8.1		El sistema verifica que no se completaron los campos obligatorios solicitados al taller.
8.2		El sistema solicita nuevamente completar los campos obligatorios faltantes.
8.3		El taller completa el campo de datos solicitado
8.4		Continúa en paso (8)
9	El sistema añade la nueva pieza al inventario del taller.	
Postcondiciones		El sistema se actualiza mostrando la nueva pieza añadida al inventario.

4)

Caso de Uso		Modificar pieza
Descripción		El taller registrado modifica una pieza de su inventario.
Precondiciones		• El taller debe estar registrado. • El taller debe contar con, al menos, una pieza en su inventario.
Secuencia de Pasos		Alternativas
1	El taller ingresa al sistema.	
2	El taller se encuentra en la vista inicial.	
3	El taller presiona el botón “Gestión de inventario”.	
4	El taller visualiza su stock de piezas.	
5	El taller modifica una pieza del inventario apretando el botón con forma de lápiz al lado “Modificar pieza”.	
6	El sistema ofrece al taller modificar nombre, descripción, cantidad y precio por unidad de la pieza.	
7	El taller modifica los datos correspondientes.	
8	El sistema verifica que se completaron los campos obligatorios solicitados al taller.	
8.1		El sistema verifica que no se completaron los campos obligatorios solicitados al taller.
8.2		El sistema solicita nuevamente completar los campos obligatorios faltantes.
8.3		El taller completa el campo de datos solicitado
8.4		Continúa en paso (8)
9	El sistema modifica los datos de la pieza del inventario del taller.	

Postcondiciones	El sistema se actualiza mostrando la pieza modificada en el inventario.	

5)

Caso de Uso		Determinar diagnóstico
Descripción		El taller determina un diagnóstico a los turnos que se encuentran “En revisión”.
Precondiciones		- El taller debe estar registrado. - El taller debe contar con, al menos, un turno en estado “En revisión”.
Secuencia de Pasos		Alternativas
1	El taller ingresa al sistema.	
2	El taller ingresa a la sección “Gestión de turnos”.	
3	El taller aprieta el botón “Turnos tomados” para visualizar las solicitudes de turnos aceptadas.	
4	El sistema muestra por defecto los turnos en estado “En revisión”.	
5	El taller aprieta el botón “Determinar diagnóstico”.	
6	El sistema solicita al taller un diagnóstico escrito.	
7	El taller completa el campo escribiendo un diagnóstico.	
8	El sistema verifica que el taller completó el campo obligatorio de diagnóstico.	
8.1		El sistema verifica que el taller no completó el campo obligatorio de diagnóstico.
8.2		El sistema solicita nuevamente completar el campo obligatorio faltante.

8.3	El sistema envía el diagnóstico a la cuenta del cliente.	El taller completa el campo de datos solicitado.
8.4		Continúa en paso (9)
9		
Postcondiciones		- El sistema actualiza la información modificada del cliente. - El estado del turno cambia a “Aceptado”.

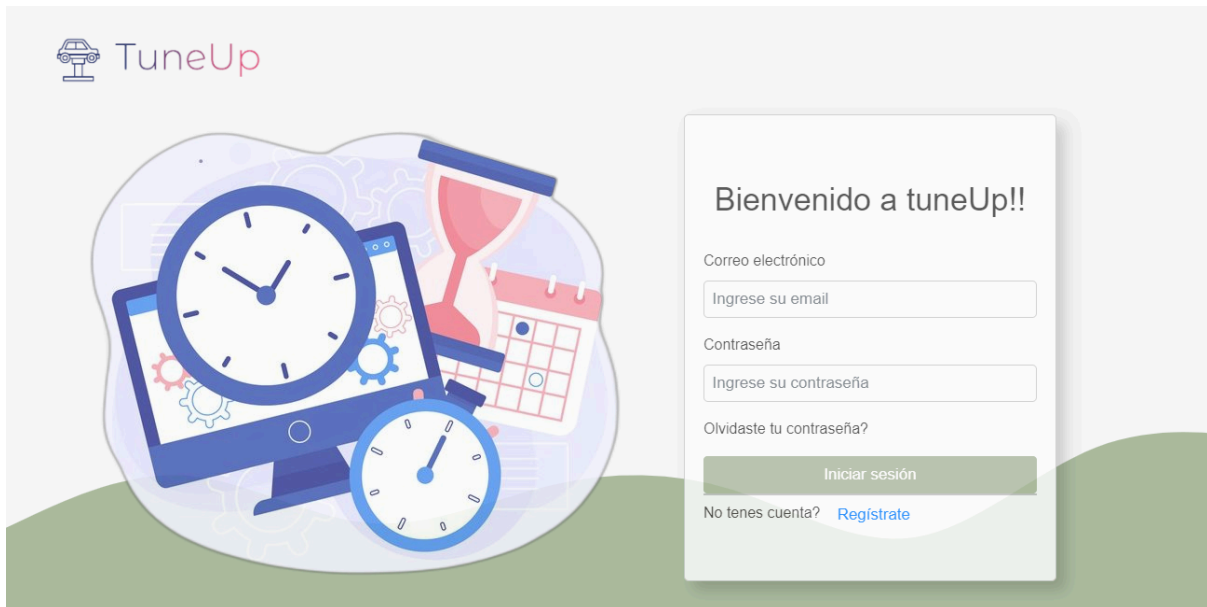
6)

Caso de Uso		Configurar perfil
Descripción		El taller registrado configura el perfil al apretar el ícono de perfil registrado.
Precondiciones		El taller debe estar registrado.
Secuencia de Pasos		Alternativas
1	El taller ingresa al sistema.	
2	El taller ingresa a la sección de “Mi perfil” apretando el ícono de perfil registrado.	
3	El cliente puede modificar su contraseña.	
Postcondiciones		El sistema actualiza la información modificada del taller.

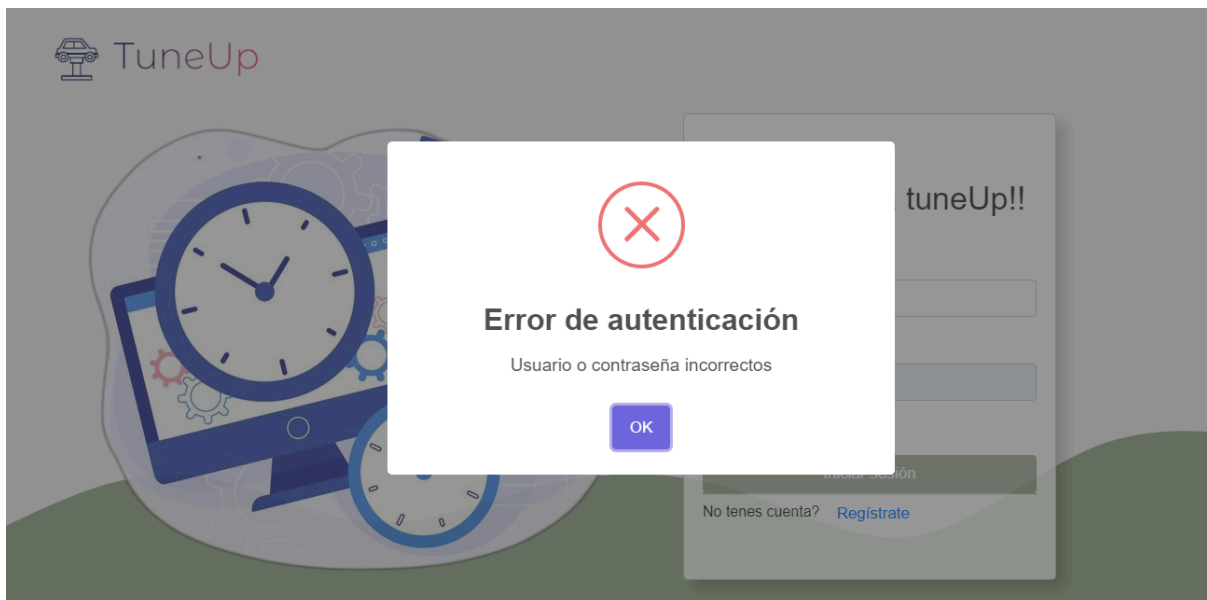
Diseño - Vistas

Módulo Login

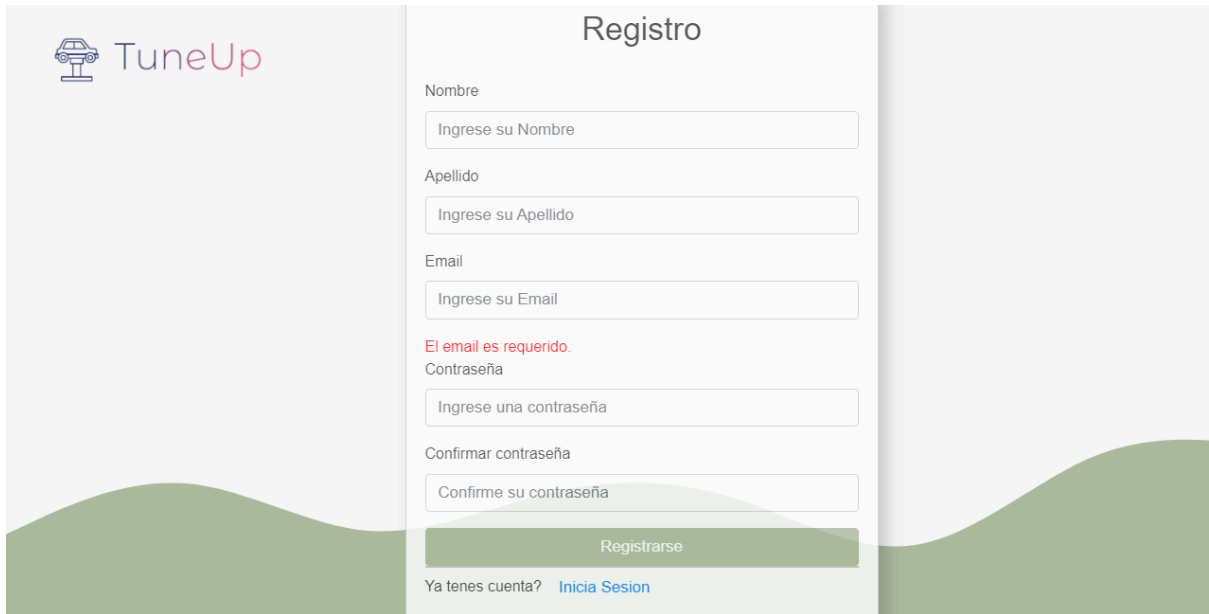
- Vista Login - Home



- Vista Login con información incorrecta



Módulo Registro



The registration form is titled "Registro" and features the TuneUp logo on the left. It includes input fields for "Nombre", "Apellido", "Email", "Contraseña", and "Confirmar contraseña". A red error message "El email es requerido." is displayed below the email field. A "Registrarse" button is at the bottom, along with a link "Ya tenes cuenta? Inicia Sesión".

Registro

Nombre
Ingrese su Nombre

Apellido
Ingrese su Apellido

Email
Ingrese su Email

El email es requerido.

Contraseña
Ingrese una contraseña

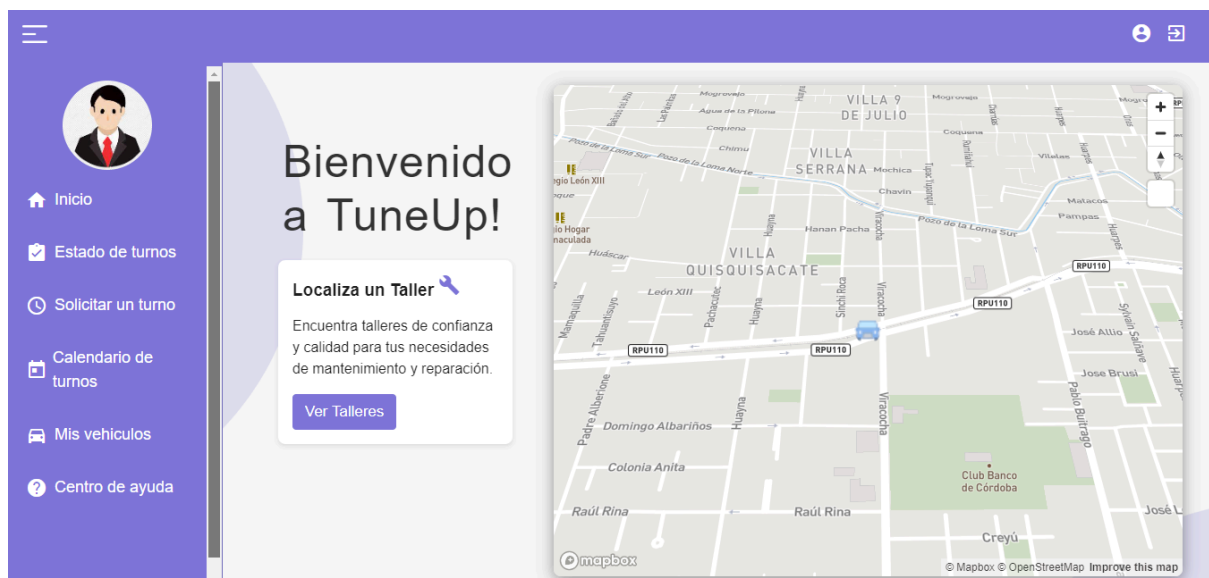
Confirmar contraseña
Confirme su contraseña

Registrarse

Ya tenes cuenta? [Inicia Sesión](#)

Módulo Cliente

- Vista Principal - Home



- **Vista Estado de Turnos - Mis Turnos**

Mis Turnos

Aquí puedes gestionar y visualizar todos tus turnos.

Fecha Estimada	Nombre de Taller	Tipo	Auto	Estado	Número de Cita	Diagnostico
2023-08-15T13:47:00	Bustos Fierro Taller	Revision	K	Finalizado	fOm3bCPe	Ver Diagnostico
2023-08-14	Bustos Fierro Taller	Revision	K	Finalizado	uVymdTDv	Ver Diagnostico
2023-08-10	Bustos Fierro Taller	Mecanica Liviana	K	Finalizado	2aYiHHgl	Ver Diagnostico

Items per page: 5 1 – 3 of 3

- **Vista Solicitar Turno**

Solicitar Turno

Bienvenido a nuestro sistema de solicitud de turnos. Aquí podrás programar un turno para el servicio de tu vehículo de manera fácil y rápida.

1 2 3 4

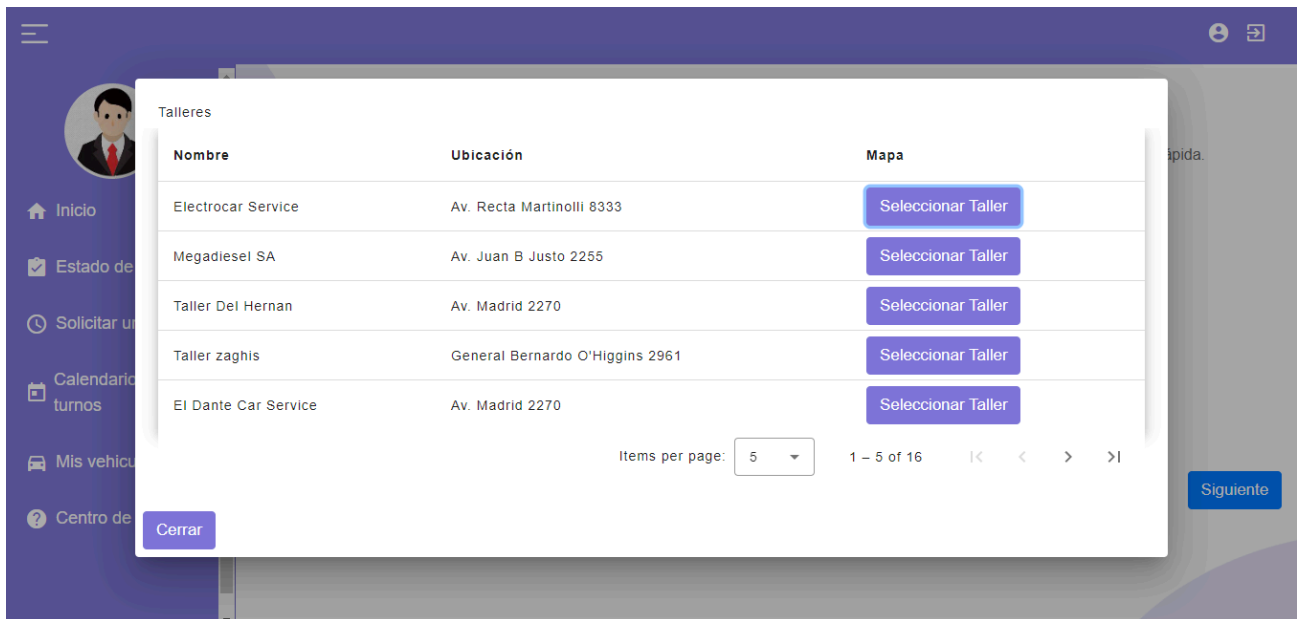
Selecciona un taller

Selecciona uno de los talleres disponibles a continuación para programar tu turno. Nuestros expertos estarán listos para brindarte un excelente servicio y resolver cualquier problema con tu vehículo.

[Ver Talleres](#)

[Siguiente](#)

- **Vista Ver Talleres**



- **Vista Selecciona un vehículo**



- **Vista Selecciona tipo de turno**

Solicitar Turno

Bienvenido a nuestro sistema de solicitud de turnos. Aquí podrás programar un turno para el servicio de tu vehículo de manera fácil y rápida.

1 2 3 4

Selecciona el tipo de turno

Revision

Tipo de turno seleccionado:
Revision

Anterior Siguiete

- **Vista Información de turno**

Solicitar Turno

Bienvenido a nuestro sistema de solicitud de turnos. Aquí podrás programar un turno para el servicio de tu vehículo de manera fácil y rápida.

1 2 3 4

Información de turno

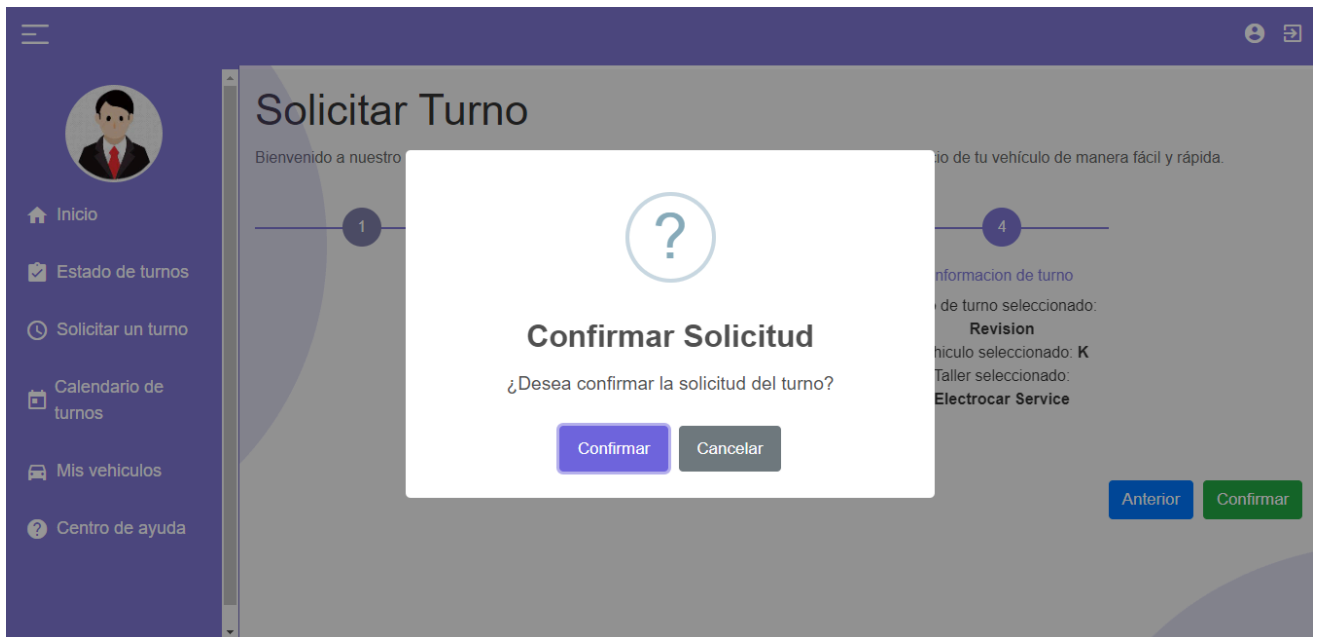
Tipo de turno seleccionado:
Revision

Vehiculo seleccionado: **K**

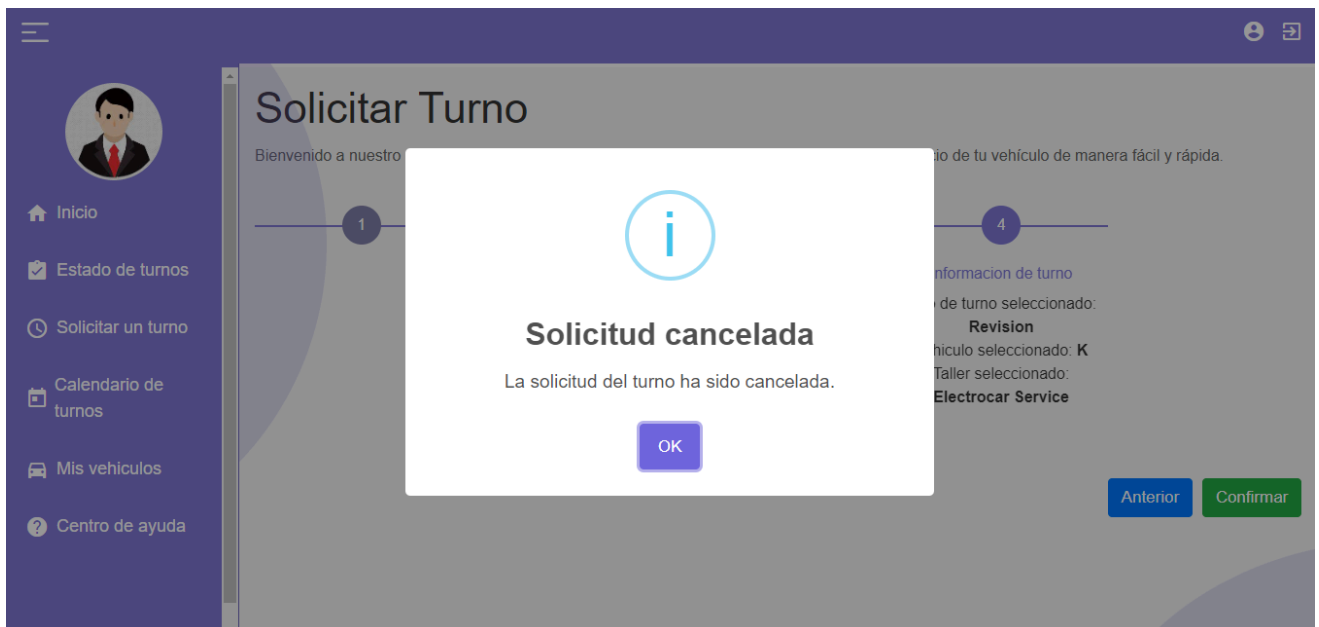
Taller seleccionado:
Electrocar Service

Anterior Confirmar

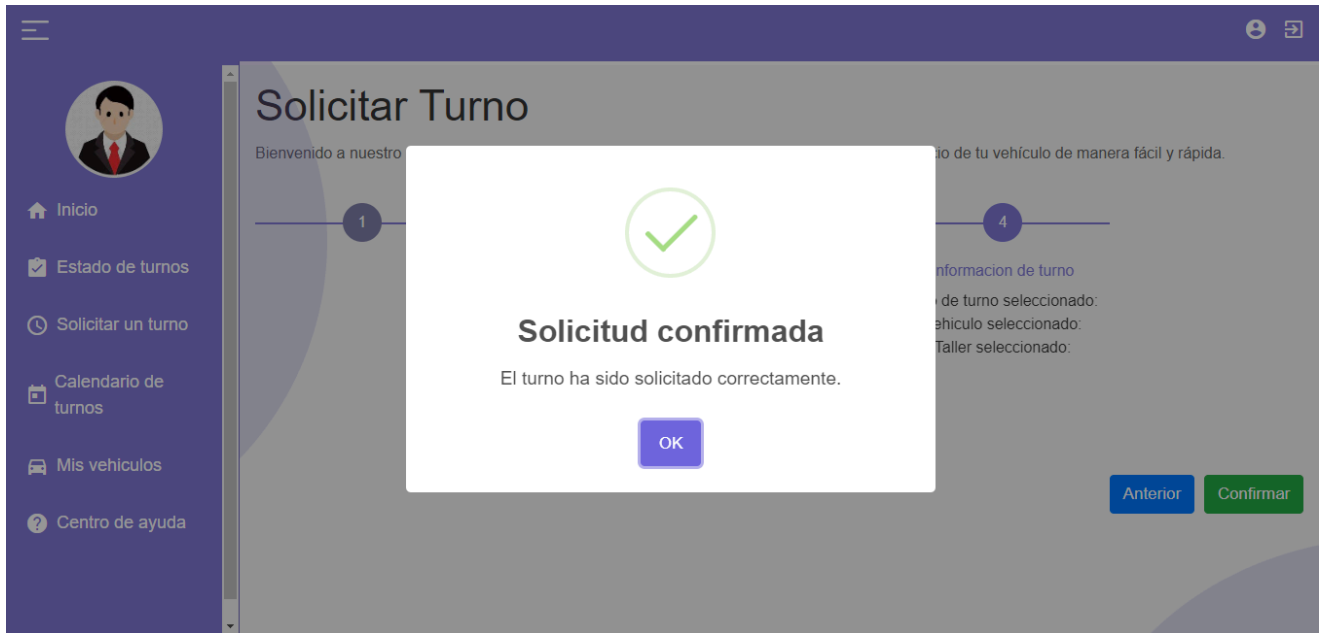
- **Vista Confirmación de solicitud de Turno**



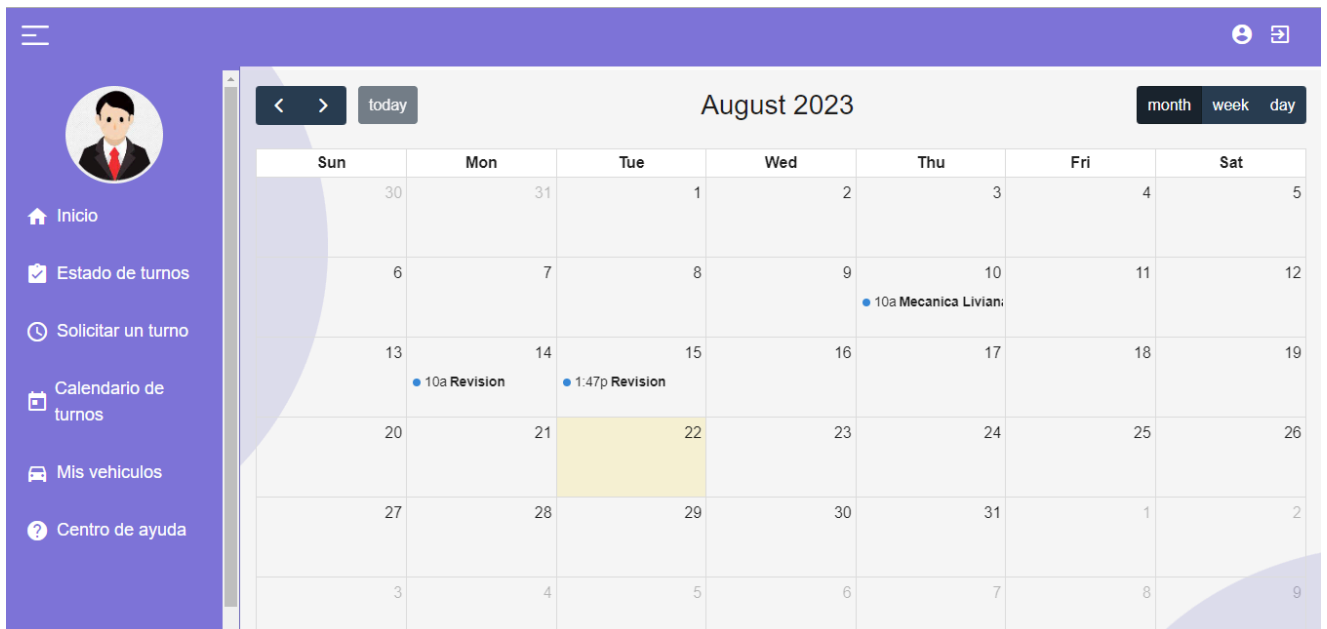
- **Vista Solicitud de Turno Cancelada**



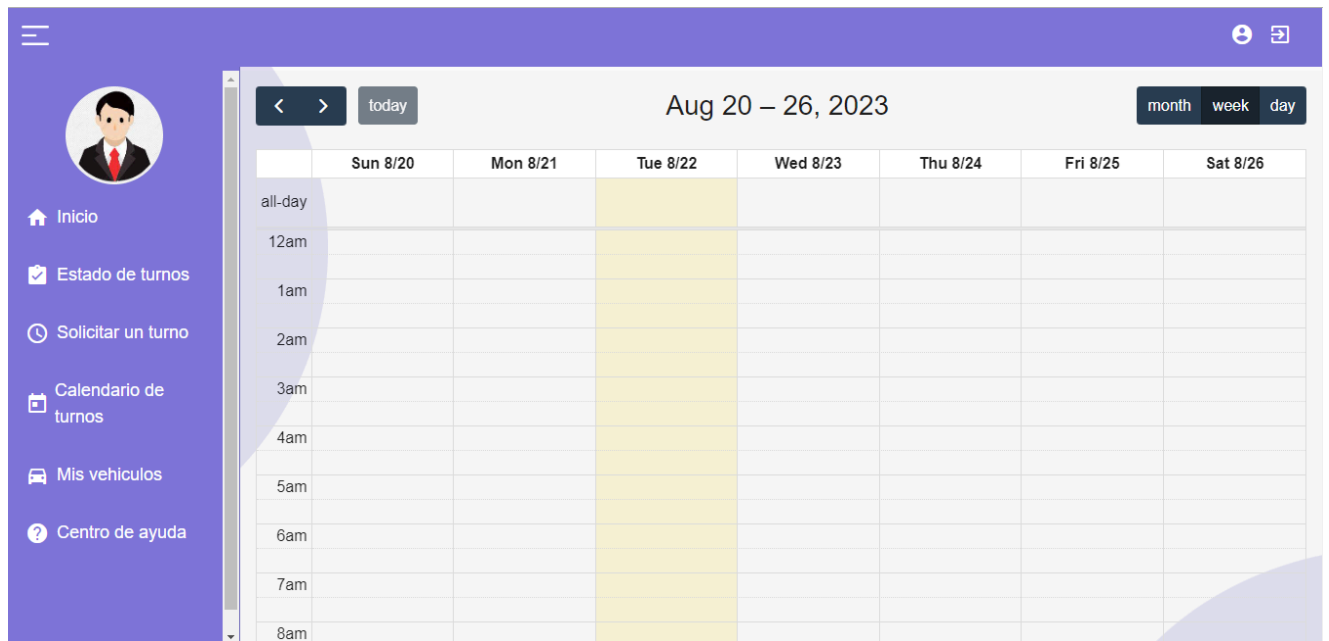
- **Vista Solicitud de Turno Confirmado**



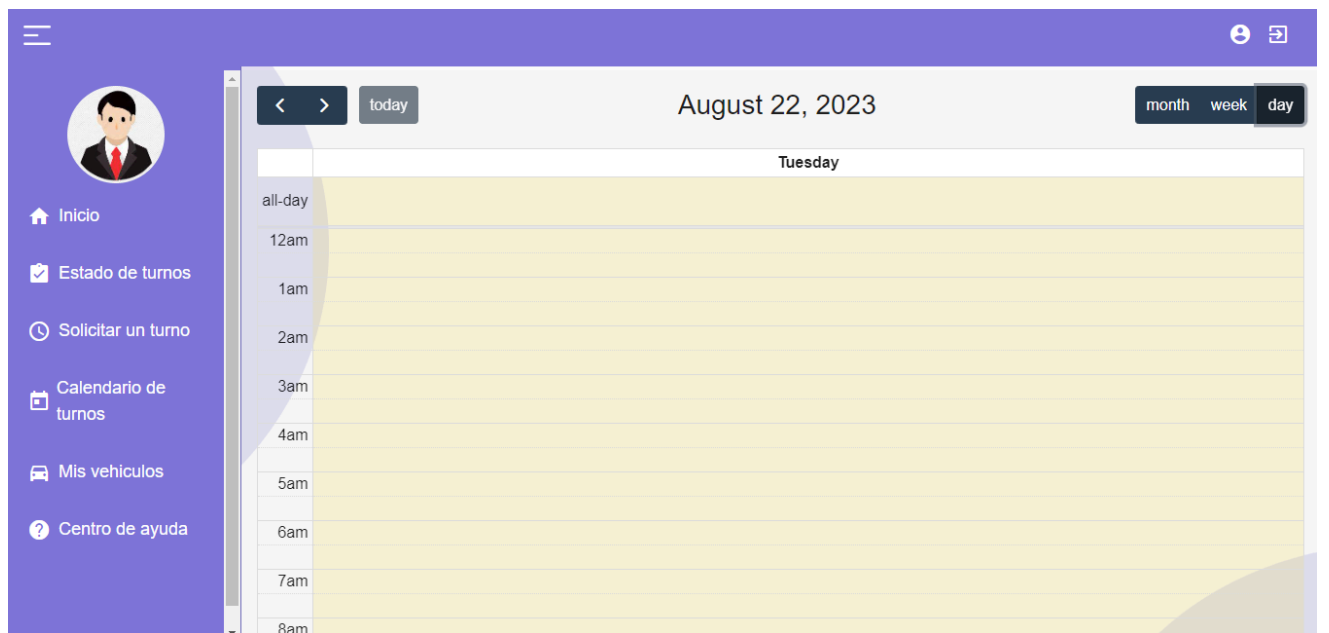
- **Vista Calendario de Turnos - Por mes**



- **Vista Calendario de Turnos - Por semana**



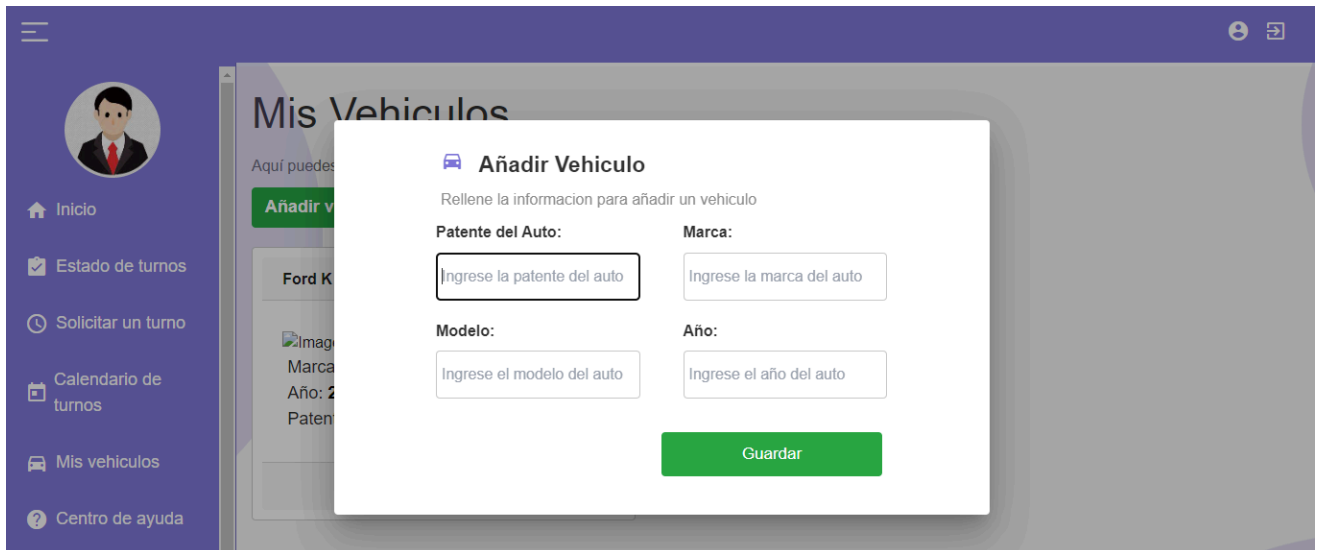
- **Vista Calendario de Turnos - Por día**



- **Vista Mis vehículos**



- **Vista Añadir vehículo**



- **Vista Ver historial de turnos**

Historial de turnos

Descripción del turno	Fecha del servicio	Costo de servicio	Duración de servicio
se parcho la pinchadura	2023-08-09T09:39:24.93	1500 \$	1 Dia/s
Se cambio cubierta de la rueda derecha delantera de Ford Ka	2023-08-13T13:53:37.55	30000 \$	1 Dia/s
se cambio optica derecha	2023-08-14T15:43:47.727	5000 \$	1 Dia/s

Items per page: 5 1 - 3 of 3

Cerrar

- **Vista Centro de ayuda**

Centro de Ayuda

¿Cómo puedo agendar una cita para mi vehículo?

Agendar una cita en nuestro sistema es un proceso sencillo y fácil de seguir. Primero, debes tener registrado al menos un vehículo en nuestra plataforma. Una vez hecho esto, estás listo para programar tu cita. Dirígete a la sección "Solicitud de Turno". Allí, podrás seleccionar el taller que más te convenga según tu ubicación o preferencia. Después de seleccionar el taller, elige el vehículo para el cual deseas programar la cita. Posteriormente, tendrás que seleccionar el tipo de servicio que necesitas: Mecánica Ligera, Mecánica Pesada o Revisión. Este es un campo obligatorio, ya que proporciona una guía inicial al taller sobre la naturaleza del trabajo requerido. Si no estás seguro de la gravedad del problema de tu vehículo, no te preocupes. Simplemente selecciona "Revisión" y nuestros profesionales en el taller determinarán y te informarán sobre la mejor solución para tu vehículo.

¿Qué tipos de servicios proporcionamos en nuestros talleres asociados?

Nuestros talleres asociados se especializan en una amplia gama de servicios de reparación y mantenimiento de vehículos, para asegurar que tu vehículo siempre esté en las mejores condiciones. Con la capacidad de manejar una diversidad de problemas automotrices, nuestros talleres son confiables y altamente competentes. Además, cada taller ha sido certificado previamente por nuestro riguroso equipo de inspección, garantizando así su alta calidad y fiabilidad. Por lo tanto, tu única tarea es elegir el taller que mejor se ajuste a tu ubicación y necesidades, dejando el cuidado y reparación de tu vehículo en nuestras expertas manos.

¿Cómo puedo cancelar o modificar una cita existente?

Para cancelar o modificar una cita, hay ciertas reglas a tener en cuenta. Una vez que has solicitado una cita, esta queda en un estado pendiente. Durante esta fase, tienes la capacidad de cancelar la cita según lo necesites. Sin embargo, una vez que el taller ha aceptado tu cita, ya no se puede cancelar. En esta etapa, el taller revisará tu

- **Vista Configuración de Perfil**

Configuración de perfil

Cambiar contraseña
Realiza el cambio de contraseña.

Contraseña actual
Contraseña actual

Nueva contraseña
Nueva contraseña

Confirmar contraseña
Confirmar contraseña

Cambiar contraseña

- **Vista Cerrar Sesión**

Configuración de perfil

¿Estás seguro?
¿Quieres cerrar sesión?

Sí **Cancelar**

Cambiar contraseña

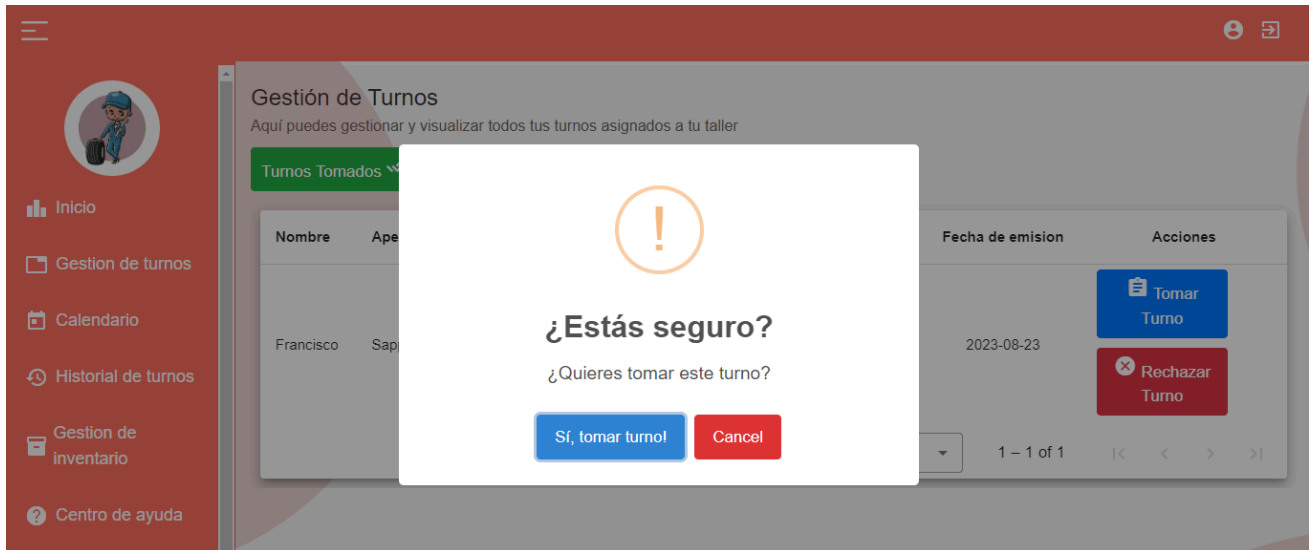
Módulo Taller

- Vista Estadísticas y resultados - Home

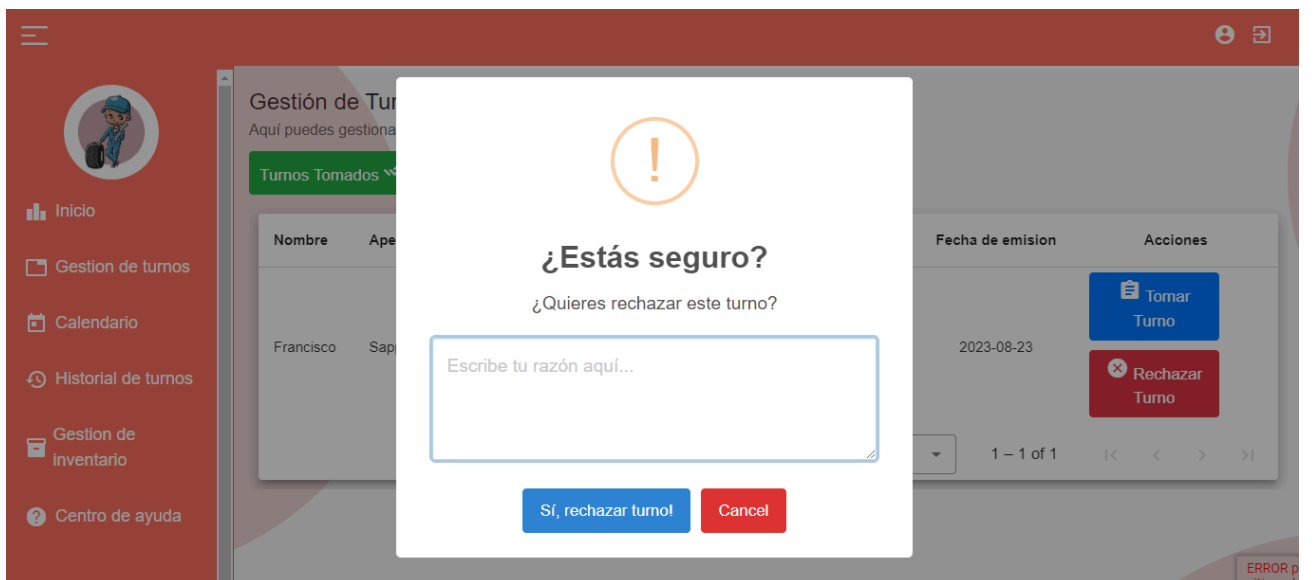


- Vista Gestión de turnos

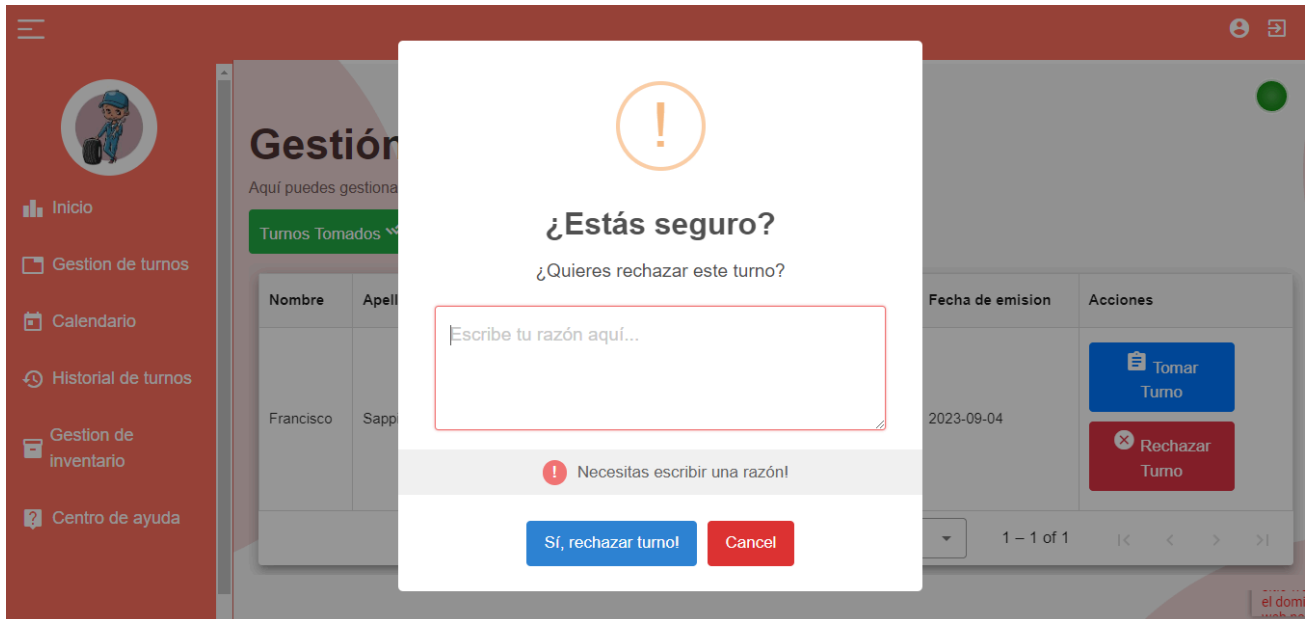
- **Vista Tomar turno**



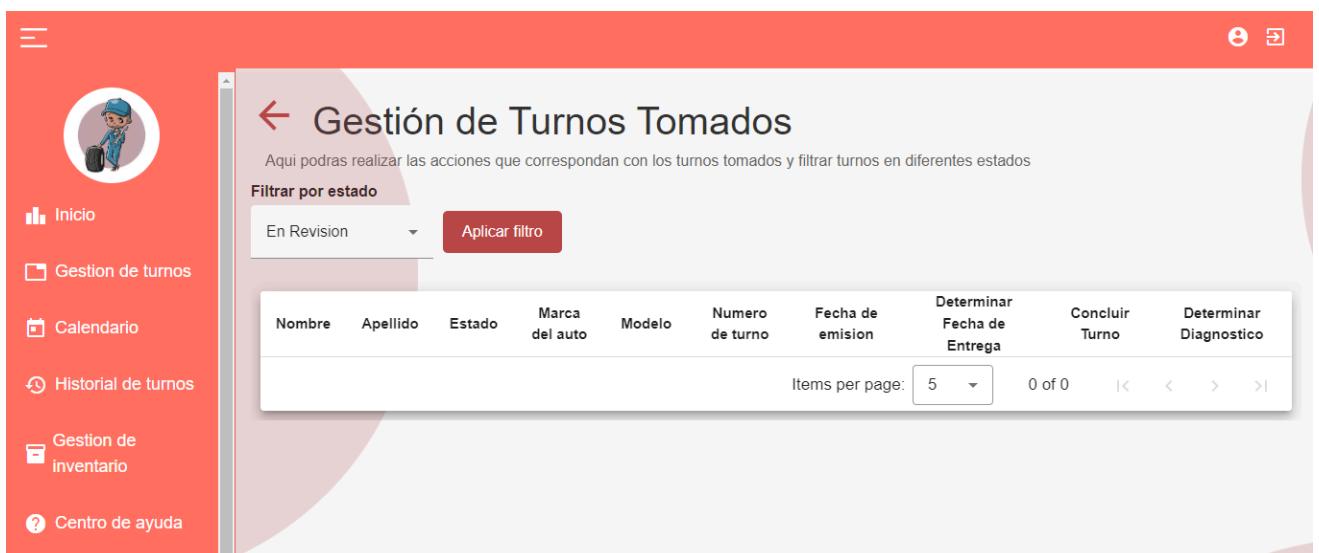
- **Vista Rechazar turno**



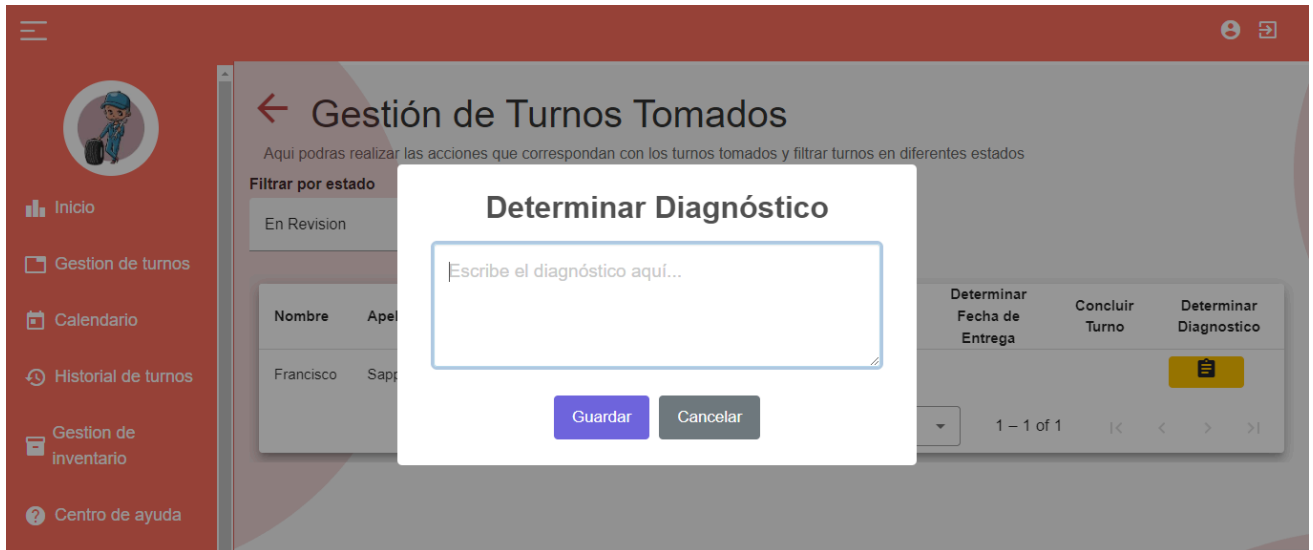
- **Vista Rechazar Turno - Solicitud obligatoria de motivo de rechazo**



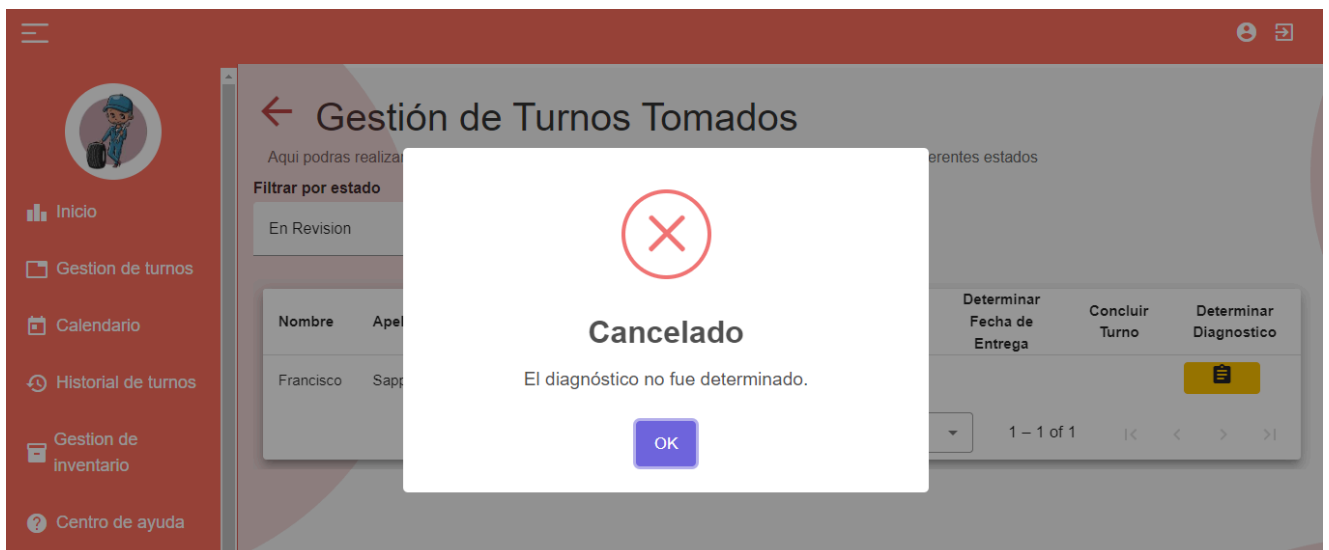
- **Vista Turnos Tomados - Gestión de turnos tomados**



- **Vista Determinar diagnóstico**



- **Vista diagnóstico cancelado**



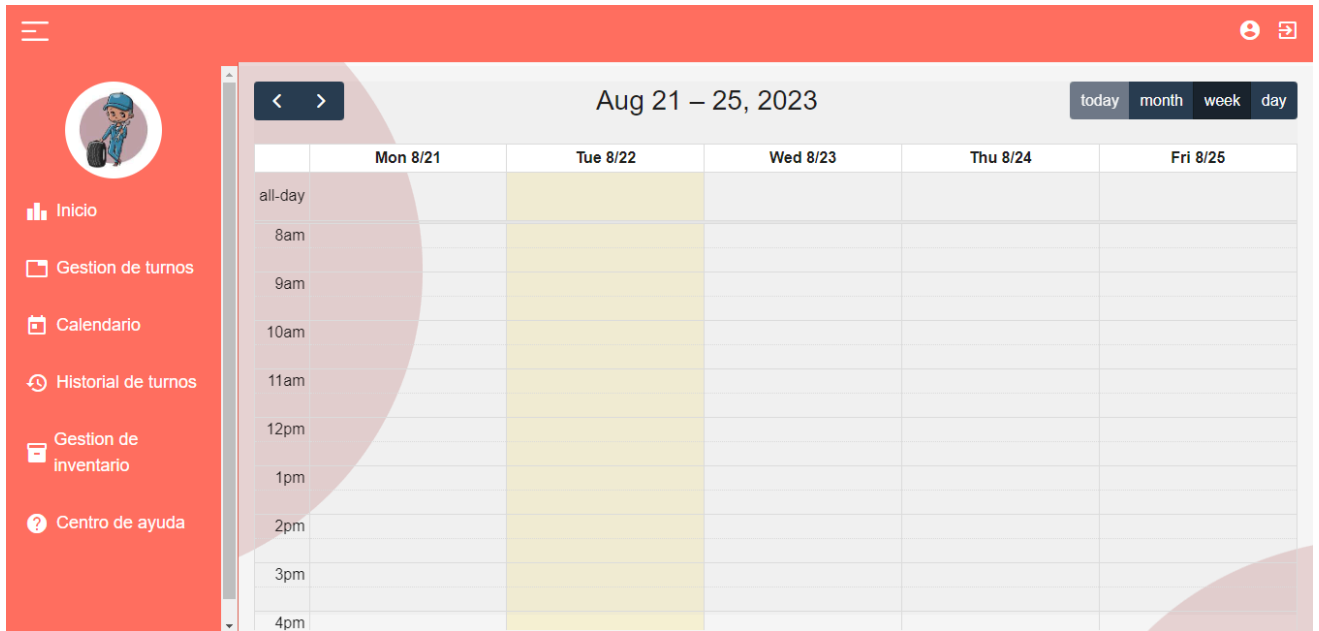
- **Vista Filtrar por estado**

The screenshot shows the 'Gestión de Turnos Tomados' page. On the left is a sidebar with navigation links: Inicio, Gestion de turnos, Calendario, Historial de turnos, Gestion de inventario, and Centro de ayuda. The main content area has a title 'Gestión de Turnos Tomados' and a subtitle 'Aqui podras realizar las acciones que correspondan con los turnos tomados y filtrar turnos en diferentes estados'. Below the subtitle is a 'Filtrar por estado' section with a dropdown menu currently showing 'Aceptado'. Other options in the dropdown include 'En Revision', 'Aceptado' (with a checkmark), 'Diagnostico Listo', 'Rechazado', 'Cancelado', and 'Finalizado'. A red 'Aplicar filtro' button is next to the dropdown. Below the filter section is a table with columns: Estado, Marca del auto, Modelo, Numero de turno, Fecha de emision, Determinar Fecha de Entrega, Concluir Turno, and Determinar Diagnostico. At the bottom of the table area, it says 'Items per page: 5' and '0 of 0'.

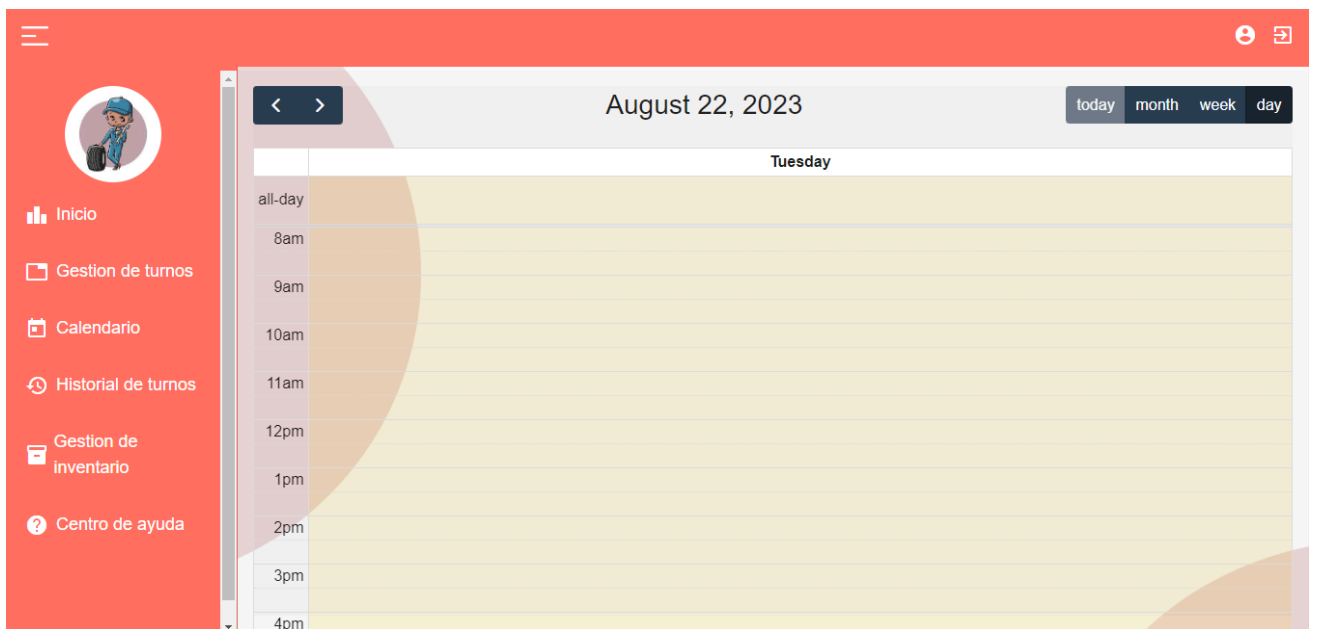
- **Vista Calendario de turnos - Por mes**

The screenshot shows the 'Vista Calendario de turnos - Por mes' page. The sidebar is the same as in the previous screenshot. The main content area has a title 'August 2023' and a subtitle 'today month week day'. Below the subtitle is a calendar grid for August 2023. The grid has columns for Mon, Tue, Wed, Thu, and Fri. The dates are: Mon (31, 7, 14, 21, 28, 4), Tue (1, 8, 15, 22, 29, 5), Wed (2, 9, 16, 23, 30, 6), Thu (3, 10, 17, 24, 31, 7), and Fri (4, 11, 18, 25, 1, 8). The date 22 is highlighted in yellow.

- **Vista Calendario de turnos - Por semana**



- **Vista Calendario de turnos - Por día**



- Vista Historial de turnos

Historial de turnos
Aqui podras ver el historial de turnos asignados al taller.

Nombre	Apellido	Estado	Marca del auto	Modelo	Numero de turno	Fecha de emision	Fecha estimada de entrega	Tipo de servicio
w	rojas	Finalizado	Peugeot	308	sEF6Ska7	2023-06-28	2023-07-29	Mecanica Pesada
w	rojas	Finalizado	Citroen	c3	n2duj5AQ	2023-07-28	2023-07-29	Revision
juan	rojas	Finalizado	AUDI	TT	IPQ844Q8	2023-06-26	2023-07-28	Mecanica Liviana
w	rojas	Finalizado	Peugeot	308	79LmL1xw	2023-06-28	2023-07-28	Mecanica Liviana
w	rojas	Finalizado	AUDI	TT	nIrgRlCu	2023-07-17	2023-07-28	Mecanica Pesada

Items per page: 5 1 - 5 of 38

- Vista Gestión de inventario

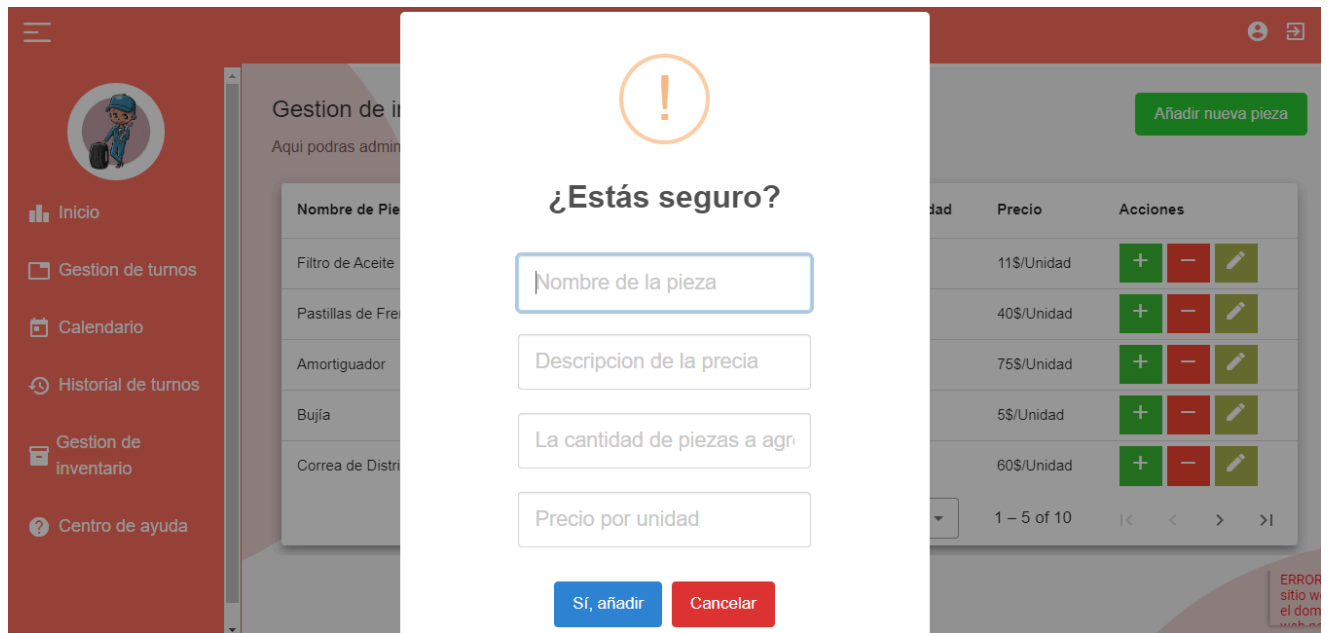
Gestion de inventarios
Aqui podras administrar el stock de piezas con la que cuentas en tu taller.

Añadir nueva pieza

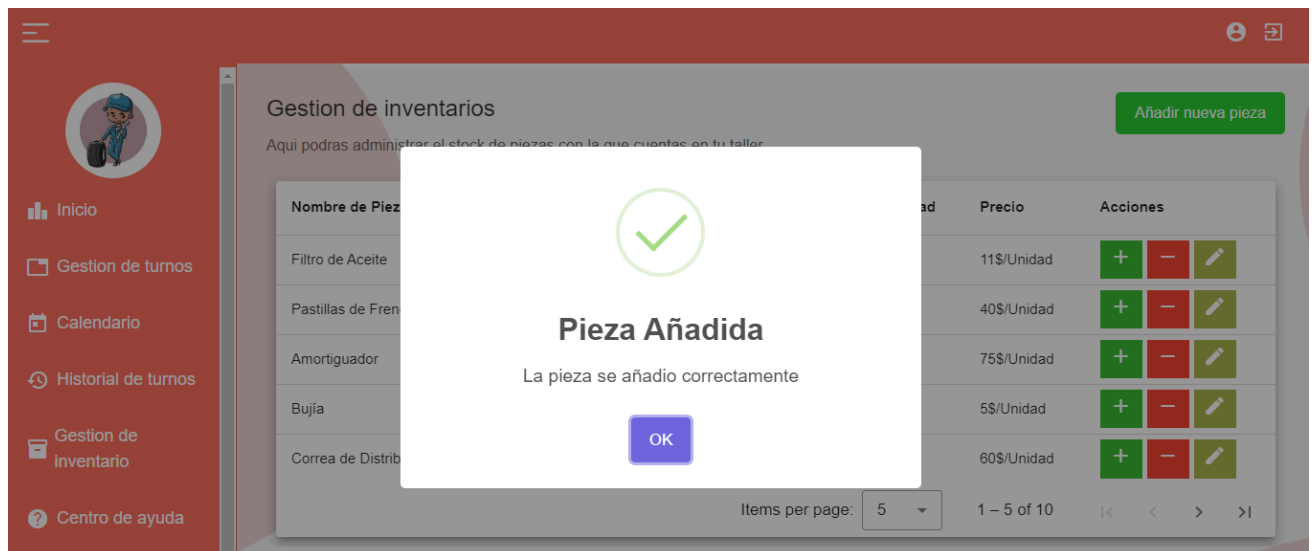
Nombre de Pieza	Descripcion	Cantidad	Precio	Acciones
Filtro de Aceite	Filtro de aceite para vehículos de gasolina	23	11\$/Unidad	+ - ✎
Pastillas de Freno	Pastillas de freno para automóviles pequeños	30	40\$/Unidad	+ - ✎
Amortiguador	Amortiguador delantero para sedanes	15	75\$/Unidad	+ - ✎
Bujia	Bujia de encendido para motores de 4 cilindros	50	5\$/Unidad	+ - ✎
Correa de Distribución	Correa de distribución para motores de 2.0L	10	60\$/Unidad	+ - ✎

Items per page: 5 1 - 5 of 10

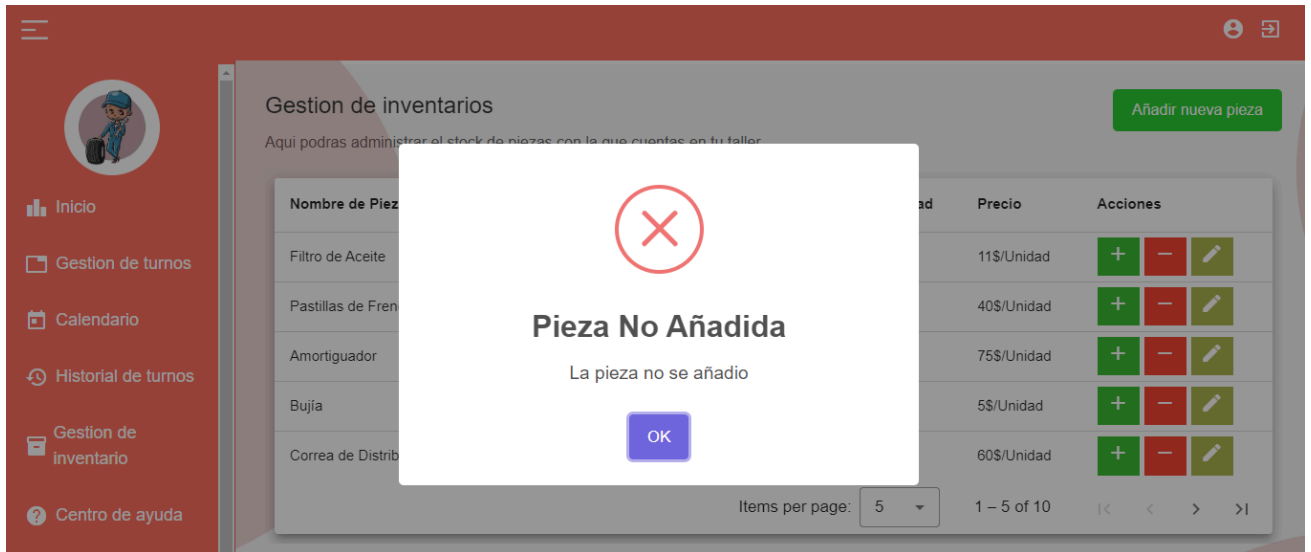
- **Vista añadir nueva pieza**



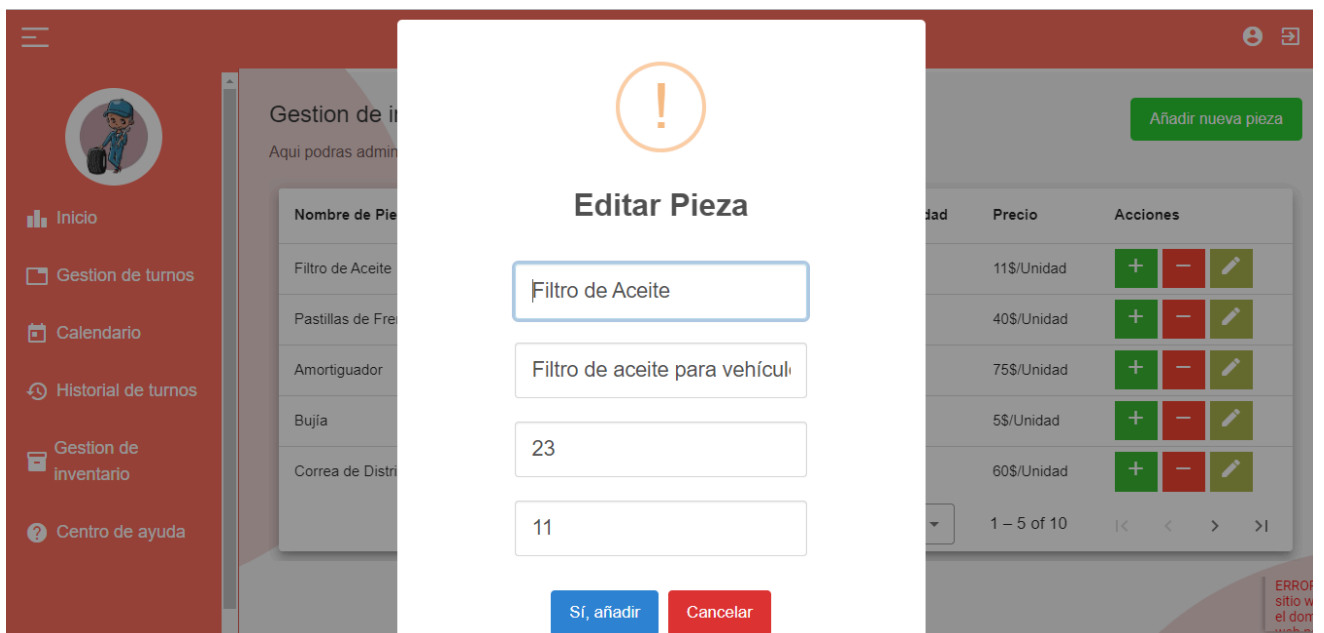
- **Vista Pieza añadida**



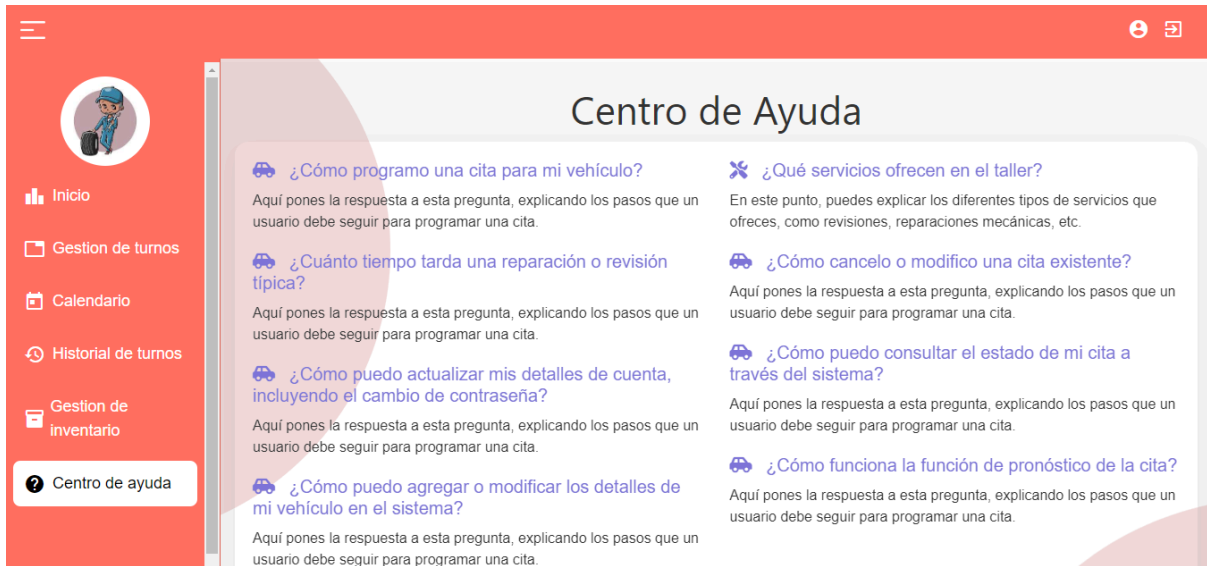
- **Vista Pieza no añadida**



- **Vista Editar Pieza**



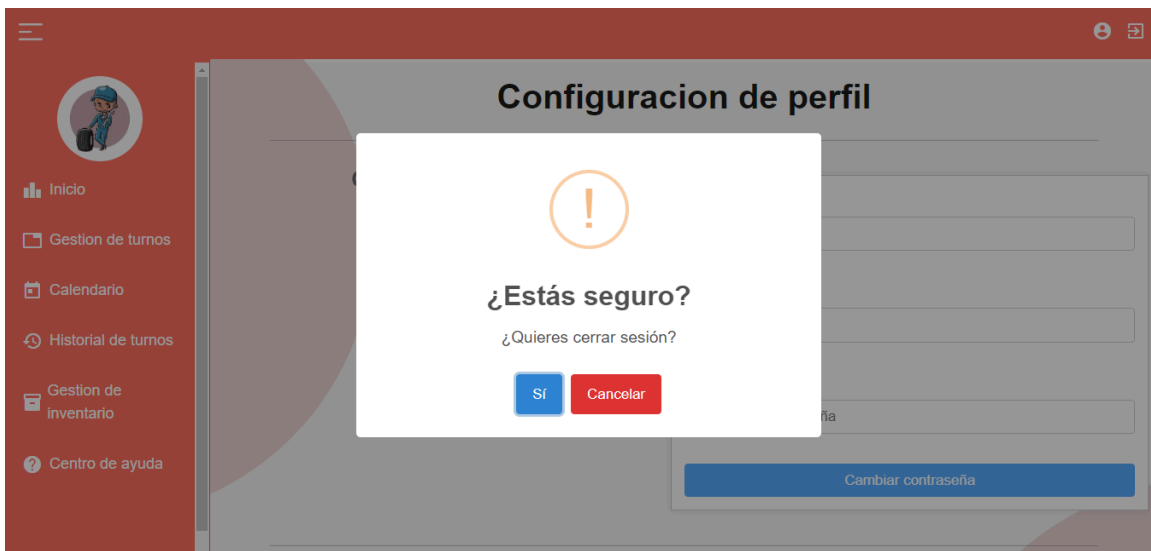
- **Vista Centro de ayuda**



- **Vista Configurar Perfil**

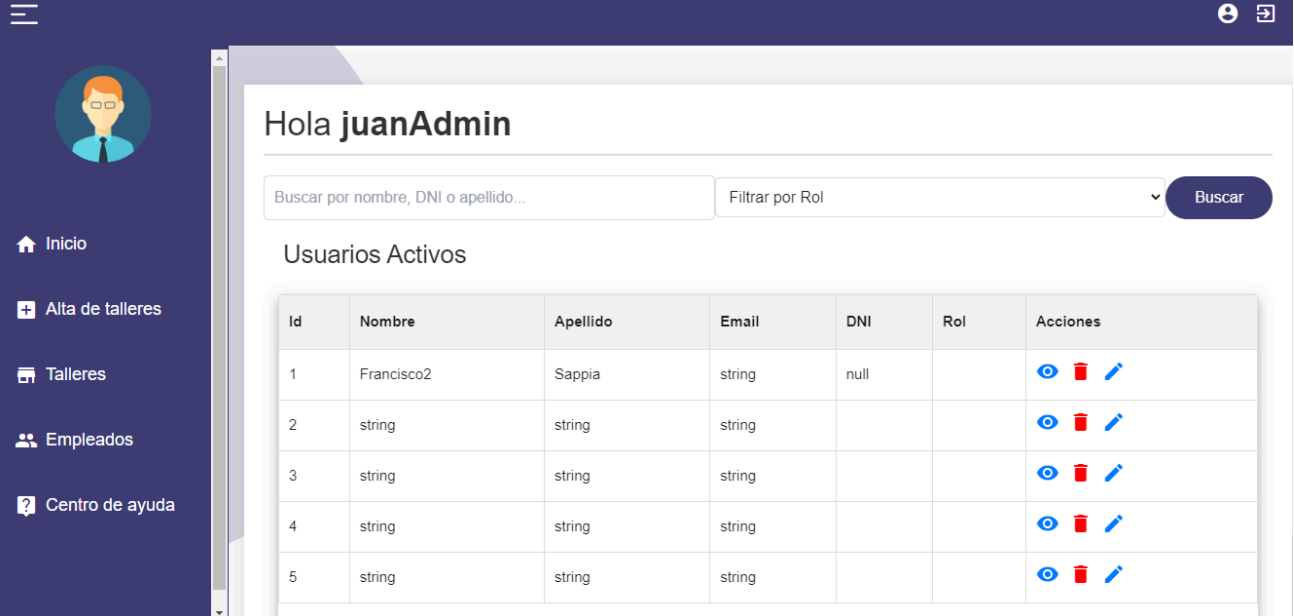


- **Vista Cerrar Sesión**



Módulo Administrador

- Vista Home - Admin



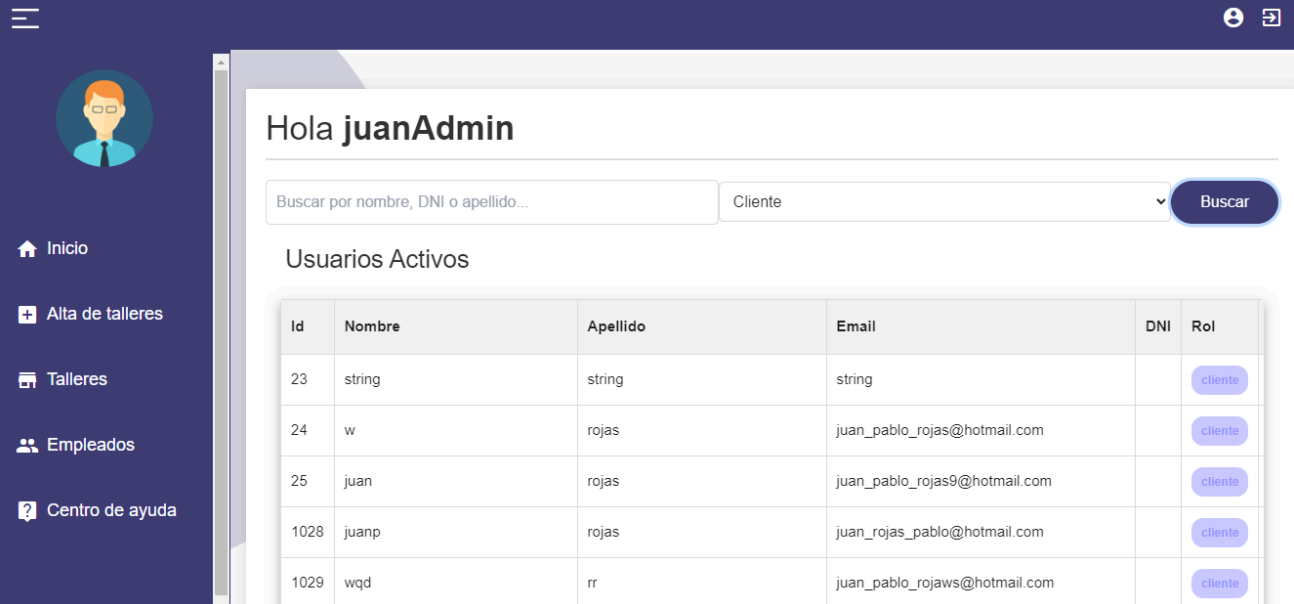
Hola **juanAdmin**

Buscar por nombre, DNI o apellido... Filtrar por Rol **Buscar**

Usuarios Activos

Id	Nombre	Apellido	Email	DNI	Rol	Acciones
1	Francisco2	Sappia	string	null		  
2	string	string	string			  
3	string	string	string			  
4	string	string	string			  
5	string	string	string			  

- Vista Filtrar por Rol Cliente



Hola **juanAdmin**

Buscar por nombre, DNI o apellido... Cliente **Buscar**

Usuarios Activos

Id	Nombre	Apellido	Email	DNI	Rol
23	string	string	string		cliente
24	w	rojas	juan_pablo_rojas@hotmail.com		cliente
25	juan	rojas	juan_pablo_rojas9@hotmail.com		cliente
1028	juanp	rojas	juan_rojas_pablo@hotmail.com		cliente
1029	wqd	rr	juan_pablo_rojaws@hotmail.com		cliente

- **Vista Filtrar por Rol Taller**

Hola **juanAdmin**

Buscar por nombre, DNI o apellido... Taller **Buscar**

Usuarios Activos

Id	Nombre	Apellido	Email	DNI	Rol	Acciones
1027	Rafael	Bustos Fierro	bustos_fierro@gmail.com	documento_identificacion	Taller	

Items per page: 5 1 - 5 of 109 |< < > >|

- **Vista Acciones - Visualizar usuarios activos**

← **Regresar al inicio**

Datos del usuario **Rafael Bustos Fierro**

Informacion del perfil

Id: 1027

Nombre: Rafael

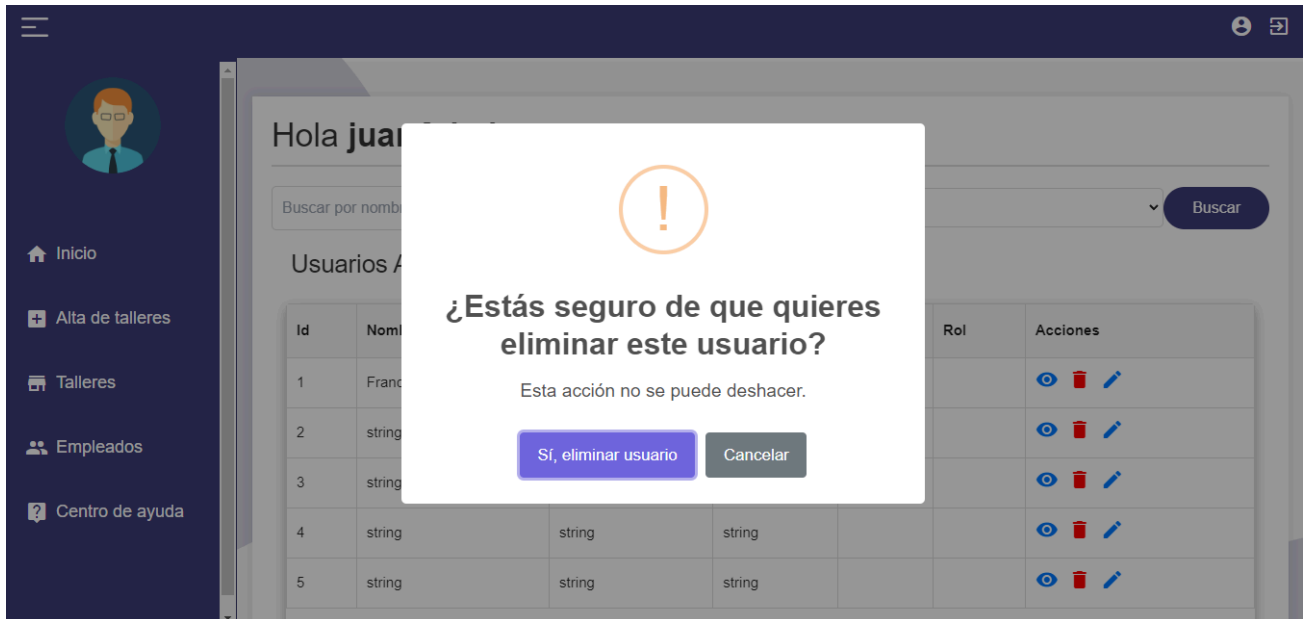
Apellido: Bustos Fierro

DNI: documento_identificacion

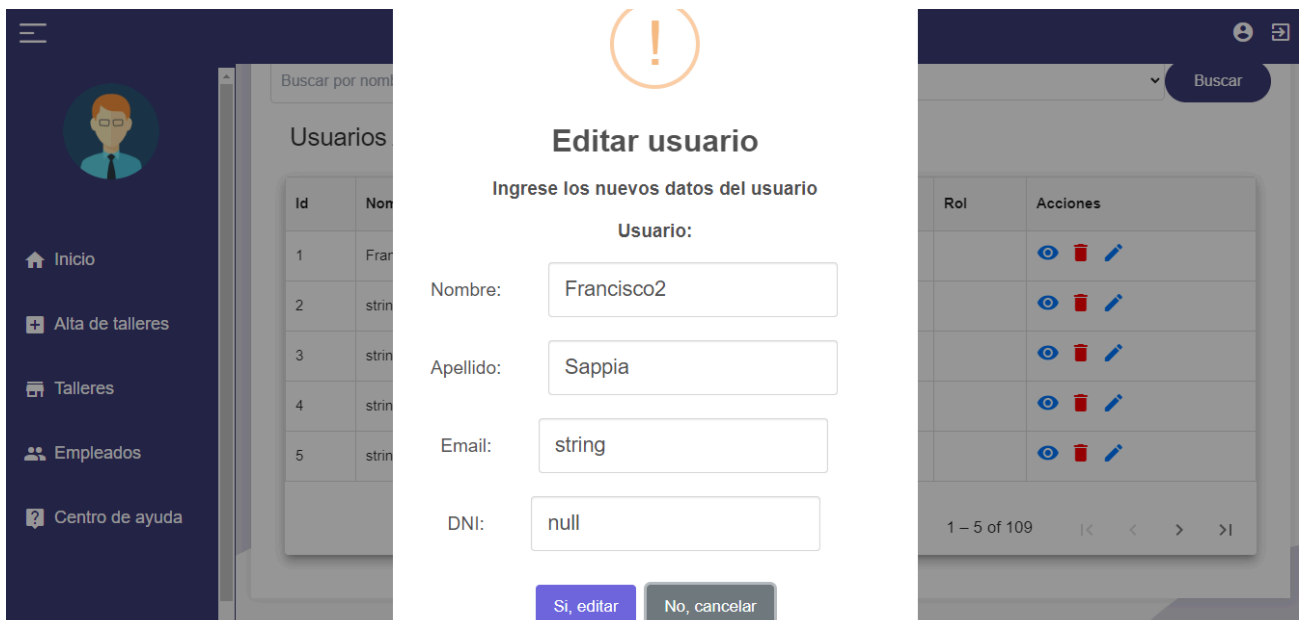
Email: bustos_fierro@gmail.com

Contraseña: 0t1W9WWOG1QOMTrkE8sBjsXtHxx73lq5j0vMIVXoXiSxkMzt

- **Vista Eliminar Usuarios activos**



- **Vista Editar Usuarios Activos**



- Vista Dar de alta talleres - Añadir taller

The screenshot shows a web application interface with a dark blue sidebar on the left and a main content area. The sidebar contains a user profile icon and a menu with the following items: Inicio, Alta de talleres, Talleres, Empleados, and Centro de ayuda. The main content area has a header with a user profile icon and a title 'Alta de Usuario taller'. Below the title is a subtitle: 'A continuación, registra a un usuario que estará vinculado al taller.' The form contains four input fields: 'Nombre de usuario:' with the value 'fransappia01@gmail.com', 'Apellido del usuario:' with the placeholder 'Ingrese el apellido del usuario', 'Email del usuario:' with the placeholder 'Ingrese el email del usuario', and 'Contraseña del usuario:' with the placeholder '....'. A '→ Siguiente' button is located at the bottom right of the form.

- Vista Dar de alta talleres - campos de validación

The screenshot shows the same web application interface as the previous one, but with validation errors. The 'Apellido del usuario:' field has a red error message: 'El apellido es requerido.' The 'Email del usuario:' field has a red error message: 'El email es requerido.' The 'Nombre de usuario:' field still contains the value 'fransappia01@gmail.com'. The 'Contraseña del usuario:' field still has the placeholder '....'. The '→ Siguiente' button is still present at the bottom right of the form.

- **Vista Gestión de talleres - Talleres Activos**

Talleres Activos

Aqui se listan todos los talleres activos en el sistema, para añadir un empleado en un taller, busque el taller y seleccione "Añadir empleado"

Filtrar talleres

Buscar talleres por nombre,direccion,cuit...

Añadir taller

Id	Nombre	Direccion	Latitud	Longitud	CUIT	Acciones	Acciones
1	Bustos Fierro Taller	Antigua 1956	-31.4829	-64.2451	CUIT123456789	  	Añadir empleado
2	Taller Tito	José Javier Díaz 1815	-31.4467	-64.1696	CUIT123456789	  	Añadir empleado

- **Vista Gestión de Empleados**

Empleados

Aqui figura el listado de todos los empleados asociados a los talleres

Añadir Empleado

Id	Nombre	Apellido	Horas de trabajo	Estado	Taller Asociado	Acciones
1	Empleado	Ejemplo	8	Activo	Bustos Fierro Taller	  

Items per page: 5 1 - 1 of 1

- Vista Gestión de Empleados - Empleado inactivo

The screenshot shows a web application interface for managing employees. On the left is a dark blue sidebar with a menu containing: 'Inicio', 'Alta de talleres', 'Talleres', 'Empleados', and 'Centro de ayuda'. The main content area has a header 'Empleados' with a subtitle 'Aqui figura el listado de todos los empleados asociados a los talleres'. A blue button 'Añadir Empleado' is in the top right. Below is a table with columns: 'Id', 'Nombre', 'Apellido', 'Horas de trabajo', 'Estado', 'Taller Asociado', and 'Acciones'. The table contains one row with data: '1', 'Empleado', 'Ejemplo', '8', 'Activo' (in a blue pill), and 'Bustos Fierro Taller'. The 'Acciones' column has icons for delete, edit, and toggle. At the bottom of the table, there is a pagination control showing 'Items per page: 5' and '1 - 1 of 1'.

- Vista Gestión de Empleados - Editar empleado

This screenshot shows the same 'Gestión de Empleados' page as before, but with a modal dialog box open in the center. The modal has a title 'Confirmación' and a question '¿Seguro desea editar el empleado?'. It lists the employee details: 'Empleado: Nombre: Empleado', 'Apellido: Ejemplo', and 'Horas: 8'. At the bottom of the modal are two buttons: 'Si, editar' (blue) and 'No, cancelar' (grey). The background page is dimmed, showing the same table and sidebar.

- **Vista Dar de alta Empleados**

Alta de Empleados

Rellene la información para añadir un Empleado

Nombre de Empleado:

Apellido de Empleado:

Horas laborables:

Taller Asociado:

[→ Siguiendo](#)

- **Vista Centro de Ayuda**

Centro de Ayuda

¿Cómo programo una cita para mi vehículo? ▾

¿Cuánto tiempo tarda una reparación o revisión típica? ▾

¿Cómo puedo actualizar mis detalles de cuenta, incluyendo el cambio de contraseña? ▾

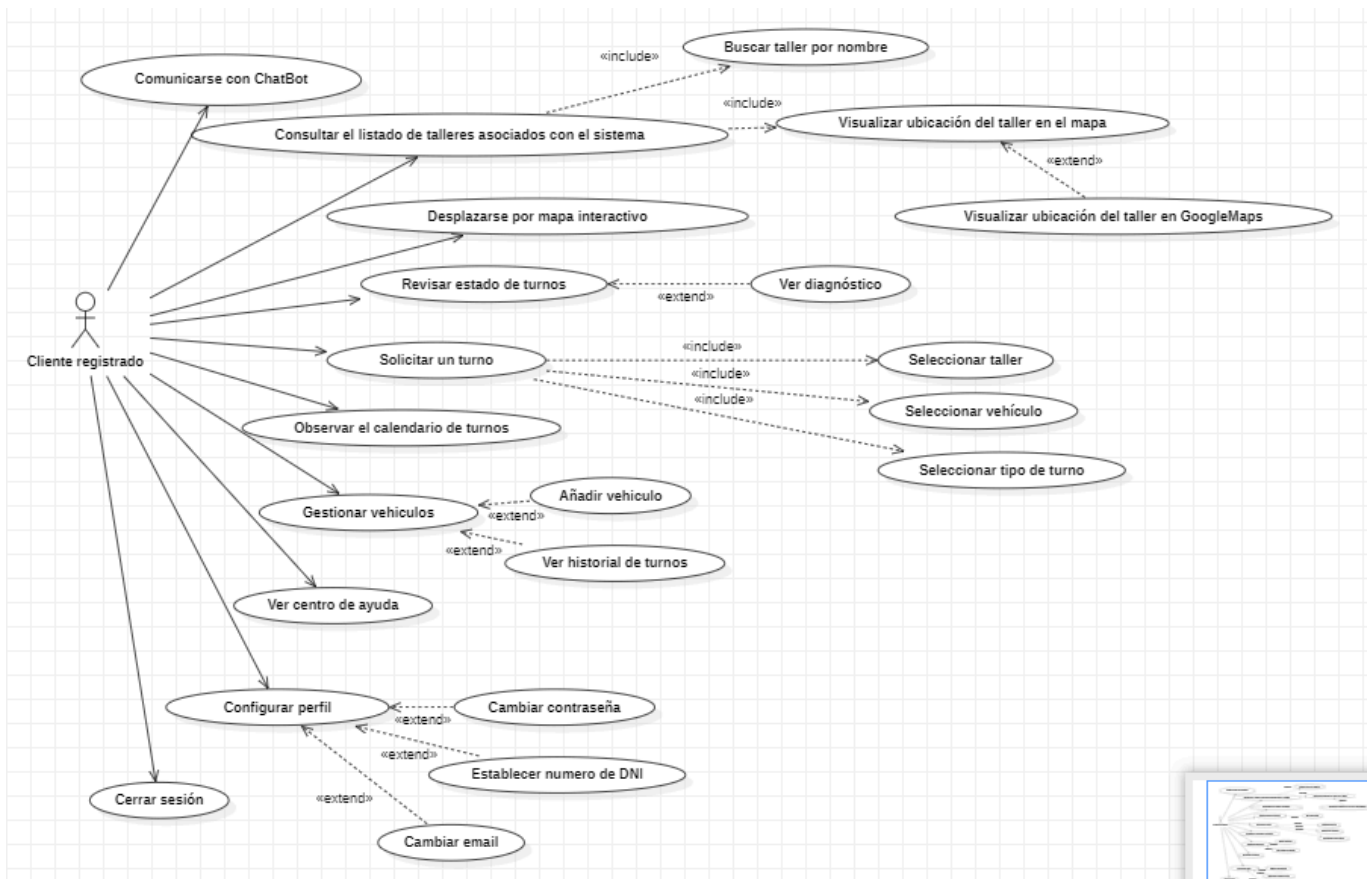
Aquí pones la respuesta a esta pregunta, explicando los pasos que un usuario debe seguir para programar una cita.

¿Cómo puedo agregar o modificar los detalles de mi vehículo en el sistema? ▾

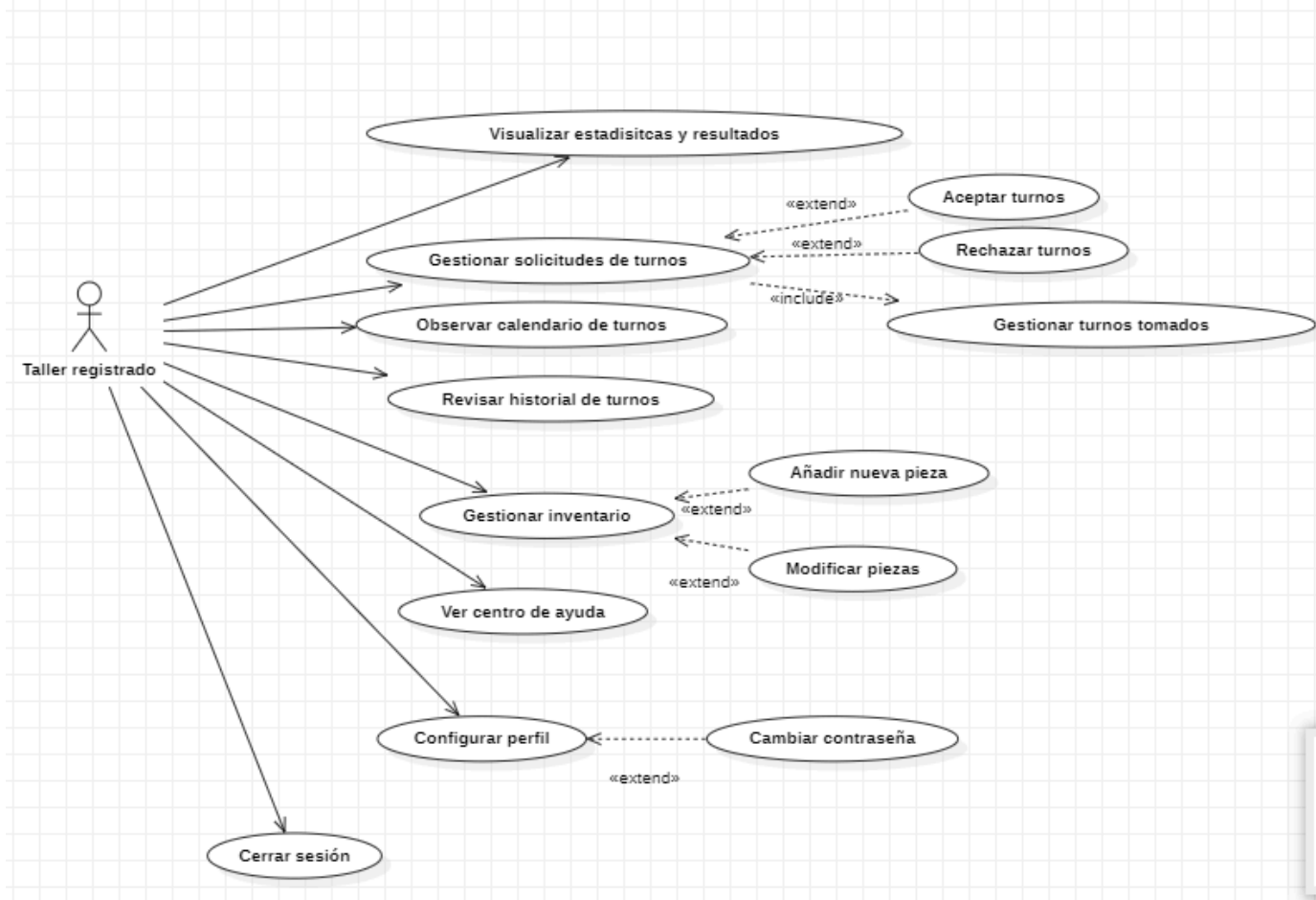
Diagramas

Diagrama de Casos de Uso

Cliente



Taller



Administrador

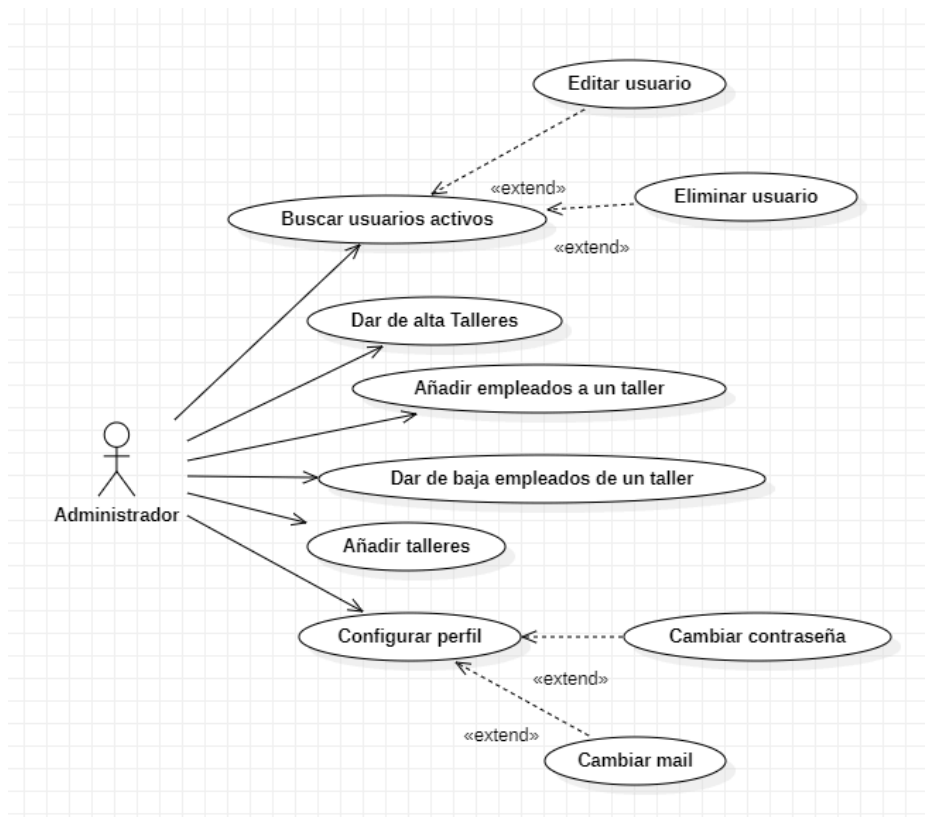
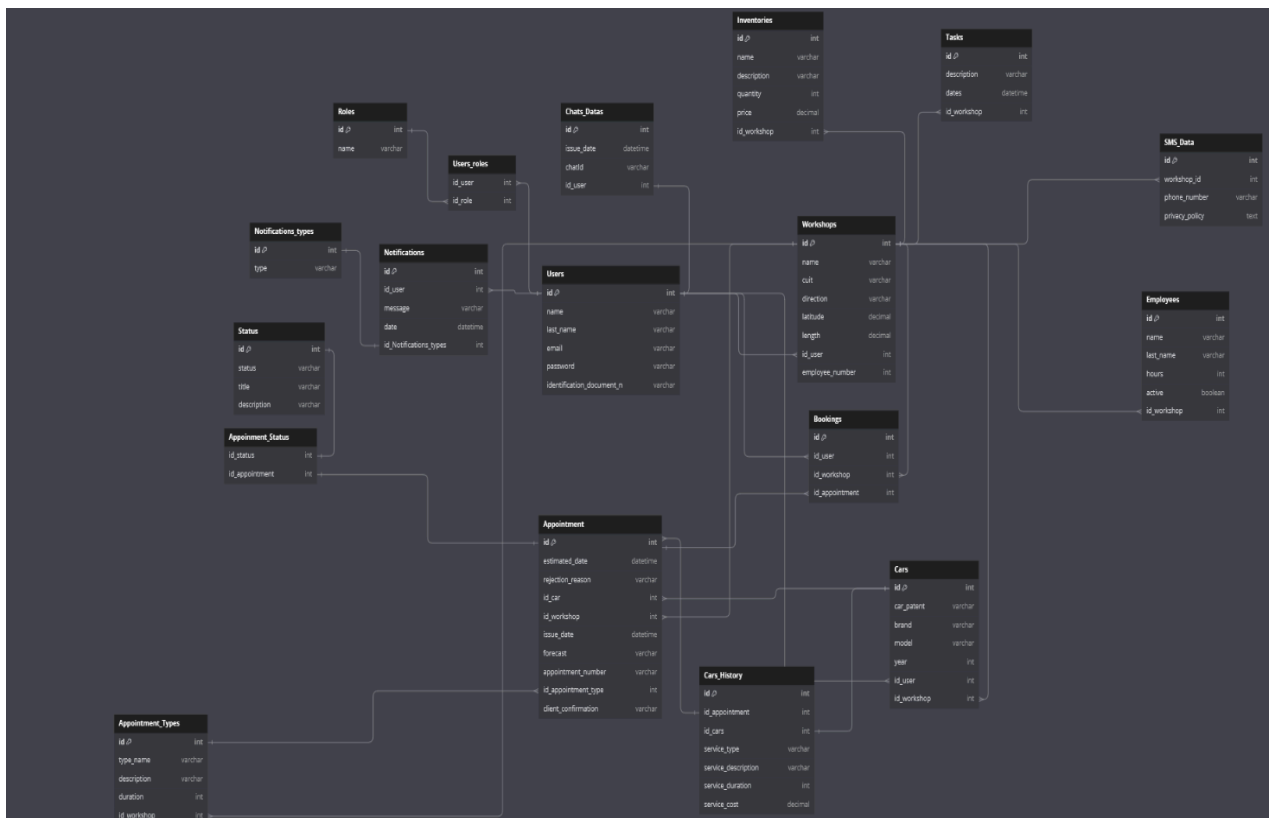


Diagrama de Clases



Tecnologías elegidas

A partir de un análisis de las tecnologías que podrían utilizarse en el desarrollo del sistema realizado anteriormente, evaluamos las ventajas y desventajas de cada una considerando aspectos como la escalabilidad, la facilidad de uso, la seguridad y la compatibilidad con los dispositivos y plataformas existentes. En base a esto, seleccionamos las tecnologías más adecuadas que nos permitirán alcanzar nuestros objetivos de manera eficiente y efectiva, las cuales nombraremos a continuación:

Backend

- C#, ASP.NET Core, .NET Framework, ASP.Net Web API
- ORM: Entity Framework
- Base de datos: SQL Server

Frontend

- Angular
- Angular Material

En cuanto al lado del servidor, elegimos estas tecnologías porque son muy versátiles, potentes y útiles para el desarrollo de aplicaciones web. La misma tecnología está compuesta por el lenguaje C# con sus frameworks de .NET CORE. El framework que usaremos para desarrollar las API's correspondientes en el sistema es ASP.Net Web API, el cual proporciona las herramientas y bibliotecas necesarias para la construcción de aplicaciones web y servicios de API que son robustos y permiten una escalabilidad. Estas tecnologías tienen origen en Microsoft y tienen constante soporte y actualizaciones de sus herramientas.

Decidimos elegir Entity Framework como ORM, que simplifica mucho la interacción entre el backend y la base de datos. Permite la manipulación rápida, segura de los datos, permite trabajar con los datos utilizando objetos de .NET, lo que puede hacer que el código se pueda escribir más fácil y entender mejor. También la gran ventaja que permite de generar migraciones de la base de datos bidireccionalmente.

La elección de Angular sobre React se basó en varios factores clave. Aunque React es conocido por su comodidad y flexibilidad, requiere un conjunto avanzado de conocimientos en diversas herramientas y marcos de trabajo en JavaScript. En contraste, Angular, aunque puede parecer inicialmente más complejo, ofrece una integración más sólida con ciertas bibliotecas y tecnologías en el entorno de .NET. Además React no es un framework como lo es Angular, es una librería.

La estructura y las convenciones predefinidas de Angular facilitan la organización y el desarrollo de aplicaciones complejas, lo que puede resultar beneficioso a largo plazo.

Haciendo referencia a la base de datos, decidimos utilizar SQL Server por el hecho que es la base de datos que más vínculo tiene con la tecnología utilizada para el desarrollo de la aplicación. Además, es una base de datos relacional que tiene un rendimiento robusto y escalable, y soporta muchas funcionalidades. Una de las funcionalidades principales que destacamos es la versatilidad de crear y almacenar procedimientos almacenados donde podemos ejecutar procesos complejos que es beneficioso tenerlo almacenado directamente en la base de datos. SQL Server se integra bien con las tecnologías de Microsoft, entonces es otra de las ventajas por la que seleccionamos esta opción.

Pruebas

Dada la decisión de utilizar .NET, en lo que respecta a las pruebas unitarias de código, implementando TDD, decidimos que la herramienta más adecuada para realizar los unit test fue XUnit en lugar de NUnit. Las razones son las siguientes:

- Proporciona un mejor aislamiento de las pruebas en comparación a los otros marcos de trabajo.
- Por cada caso de prueba, la clase de prueba se ejecuta y se descarta después de su ejecución, lo que permite que las pruebas se puedan ejecutar en cualquier orden y genera desacoplamiento entre las pruebas ya que no hay dependencias.
- **Extensibilidad:** esta característica es crucial, ya que implica la capacidad de agregar nuevas funcionalidades, modificar el comportamiento existente o integrar el marco de prueba con otras herramientas o bibliotecas de manera sencilla y eficiente. Xunit permite el uso de atributos [Fact] y [Theory] que permiten escribir pruebas de manera más flexible y parametrizada. También elimina algunos atributos presentes

en los marcos de Nunit, permitiendo que la implementación de funcionalidades sea más accesible.

- **Ejecución concurrente:** Permite la ejecución de pruebas en paralelas, lo que podría ayudar en un futuro si se quieren implementar pruebas automatizadas y este tipo de ejecución mejoraría notablemente el rendimiento.
- **Mecanismo de Aserción:** se refiere a cómo se realizan las afirmaciones sobre el comportamiento esperado del código en las pruebas. Xunit utiliza "Assert.Throws", que ofrece una forma más robusta de realizar pruebas de manejo de excepciones en comparación de [ExpectedException] en NUnit.

Moq: La utilización de esta herramienta que nos permite crear "mocks" limpios y mantenibles en el entorno de pruebas. Decidimos usarla porque su sintaxis es clara y permite crear código "Fluent" es decir podemos utilizar operaciones lambda. Lo único necesario es crear el mock a partir de la interfaz o la clase que queramos y describir el comportamiento para después analizar los resultados. Luego se llama al mock y devolvemos la instancia del objeto simulado. Entonces, al ser tan útil y sencillo de utilizar, fue nuestra primera y única alternativa para poder hacer los mocks de las entidades del modelado de la base de datos para las diferentes pruebas.

Arquitectura

Decidimos realizar una implementación de arquitectura en capas. Esto nos permitirá separar las responsabilidades y tener una estructura organizada en nuestro código. Cada capa tendrá un propósito específico y contribuirá a la modularidad, mantenibilidad y escalabilidad del sistema y permitirá comunicarse con otras capas mediante interfaces bien definidas. Esta arquitectura se puede aplicar de diferentes formas. En nuestro caso utilizamos lo que es el Repository Pattern. En nuestra arquitectura de capas tenemos las siguientes capas:

- **Capa de controlador:** en esta capa manejamos las peticiones del lado del cliente, es decir las request que llegan del front end hacia nuestro endpoint.

- **Capa de Business (negocio):** en esta capa por lo general se realiza toda la lógica de negocio, pero en nuestro caso decidimos en dejar esta capa para realizar y validar las excepciones.
- **Capa de modelado:** En esta capa tendremos las clases correspondientes a las entidades del modelo de datos, como también nuestros DTOs.
- **Capa de Acceso a datos DAO:** En esta capa tendremos el contexto de la base de datos que es el intermediario entre el backend y la base de datos, se mockean las entidades y se establece el contexto. Esto nos va a permitir que en esta capa desarrollemos la lógica de negocio que habíamos dijimos que íbamos a desarrollar aquí. Utilizando nuestro ORM Entity Framework y consumiendo nuestros Procedimientos almacenados en la base de datos. Con sus respectivas validaciones.
- **Las diferentes capas se sirven una de otras a partir de la inyección de dependencias,** el cual es un patrón de diseño de software que se utiliza para implementar el principio de inversión de control. Este mismo establece que un objeto no debe crear sus propias dependencias sino que debe recibirlas de un objeto externo. Es decir si un objeto necesita acceder a un servicio o otro objeto, no debería ser responsable de crearlo o buscarlo. En lugar de eso, este servicio “la dependencia” se inyecta en el objeto que lo demanda. La inyección de dependencias permite:
 - **El desacoplamiento:** ayuda al desacoplamiento del código, ya que un objeto no debe saber cómo se crea o obtiene la dependencia. Permite que el código sea modular para su respectiva funcionalidad y permite que sea reusable y que se pueda cambiar sin afectar a las clases que utilizan ese objeto.
 - **Pruebas más fáciles:** ayuda a los test porque cuando se inyectan las dependencias, es más fácil reemplazarlas por mocks para probar el código en aislamiento.

```
public class Tests
{
    private Mock<IAumnoDao> dataAccess;
```

```

[Test]
public void GetAll_Is_Not_Null()
{

    //arrange
    List<Alumno> alumnos = new List<Alumno>();
    Alumno alumno = new Alumno()
    {
        Id = 1,
        Nombre = "Juan",
        Apellido = "Perez",
        Dni = "12345678",
        Email = "juanpe@ucc.edu.ar"
    };
    alumnos.Add(alumno);

    dataAccess = new Mock<IALumnoDao>();

    dataAccess.Setup(a => a.GetAlumno()).Returns(alumnos);

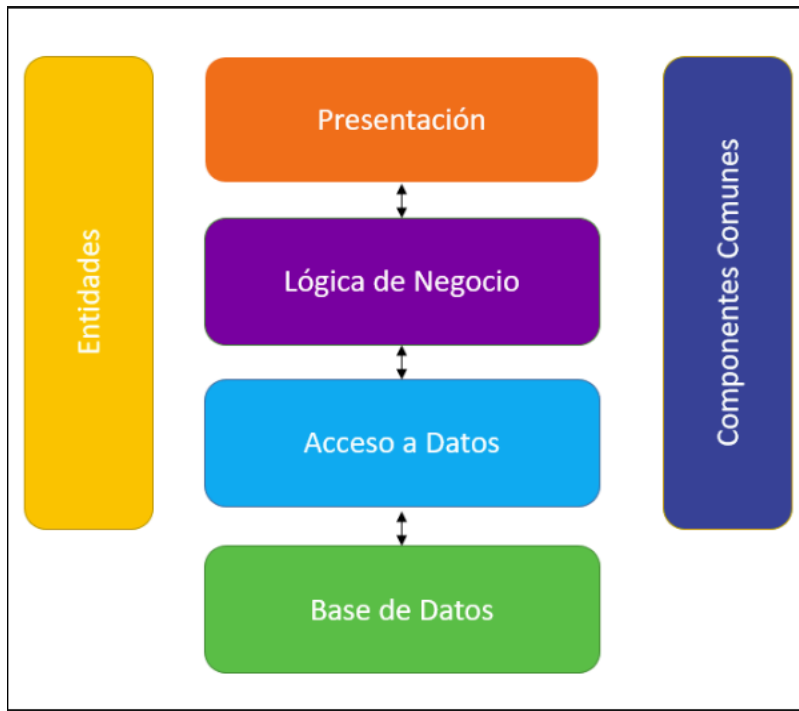
    //act
    var result = dataAccess.Object.GetAlumno();

    //assert

    Assert.IsNotEmpty(result);
    Assert.AreEqual(alumno, result[0]);
}

```

En conclusión, la arquitectura de capas es una arquitectura que se puede implementar de diferentes maneras, siempre y cuando cumpla con sus principios de separar responsabilidades, que sea modular, que sea flexible y permita escalar. Después, como te guste implementarla, queda a criterio del equipo o desarrollador. En nuestro caso nos pareció ésta una buena forma de implementarla.



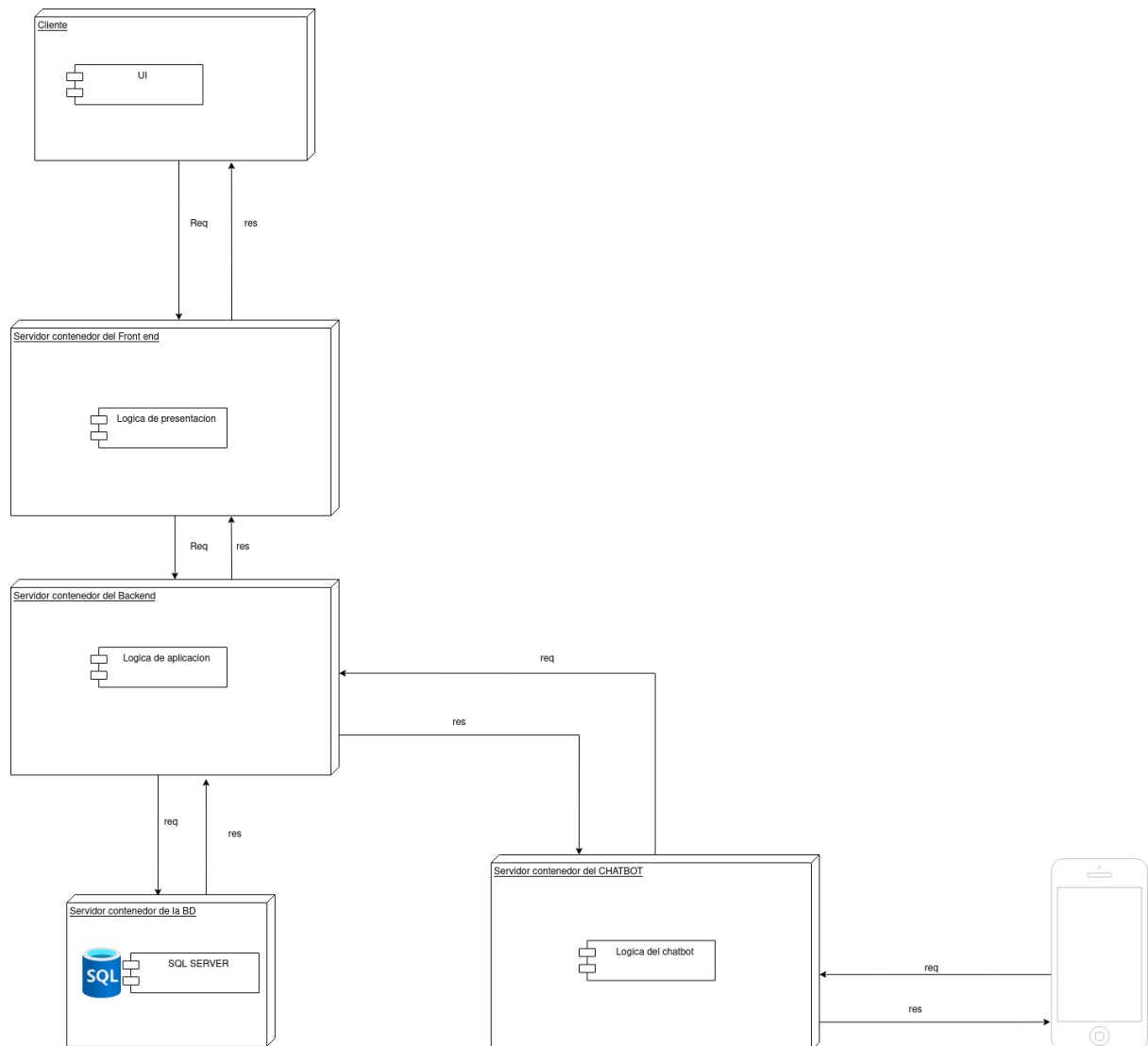
Las ventajas de esta arquitectura son:

- **Separación de responsabilidades:** A partir de la arquitectura que se eligió para el proyecto, en este caso una arquitectura de capas, lo que permite separar las responsabilidades en diferentes partes de la aplicación, cada capa tiene una responsabilidad clara y bien definida.
- **Modularidad:** La modularidad en el desarrollo de software permite dividir el sistema en partes independientes y cohesivas. Esto facilita la reutilización del código, la corrección de errores, la escalabilidad y el entendimiento del sistema. En resumen, hace que el desarrollo sea más eficiente y mantenible.
- **Flexibilidad:** Permite flexibilidad en la manipulación y desarrollo gracias a la arquitectura, permite identificar rápidamente los errores debido a que existe un nivel de granularidad y abstracción que contribuye a la flexibilidad. Además es adaptable a diferentes entornos.
- **Seguridad:** En este modelo, la seguridad se relaciona con la protección de los datos y la información del sistema contra posibles amenazas externas e internas. Algunas

de las formas en que la seguridad se relaciona con la arquitectura son las nombradas a continuación:

- **Separación de responsabilidades**
 - **Control de acceso.**
- **Control de acceso:** Se pueden incorporar medidas de control de acceso a ciertas capas, para proteger los datos y la información del sistema. Por ejemplo: solo algunos usuarios pueden acceder a ciertos datos sensibles de la capa de acceso a datos.
- **Escalabilidad:** Permite que el sistema pueda escalar de forma sencilla, agregar capas sin problemas. Al separar también las responsabilidades, permite que no se generen dependencias a la hora de escalar y permite que no haya interferencias. Entonces permite tanto añadir o quitar capas de forma sencilla y segura.

Arquitectura del sistema



Chatbot

A la hora de decidir implementar un chatbot, pensamos en un sistema que sea capaz de facilitarle al cliente la comunicación con el taller a través de respuestas rápidas y simples. La idea es que este chatbot sea capaz de ofrecer opciones estáticas sobre cierta información de un taller determinado que elija el usuario y pueda contestar, en un breve período de tiempo, con la información solicitada por el cliente.

Hay que tener en cuenta que hay muchos talleres mecánicos medianos y pequeños que cuentan con poco personal y el mismo mecánico es quien repara los autos y, a su vez, contesta los mensajes. Es por esta razón, que buscamos eficientizar el proceso de trabajo

de los talleres evitando que los trabajadores deban estar respondiendo constantemente los mensajes de los clientes y que puedan utilizar ese tiempo para enfocarse en su trabajo.

En el momento que surgió la idea de implementar este tipo de sistema, tuvimos que realizar una profunda investigación sobre el tema, donde analizamos las distintas ventajas y desventajas que esto podía llegar a tener y el impacto que generaría en los usuarios. Comprendimos que la automatización de la comunicación con los clientes a través de un chatbot no solo aumentaría la eficiencia de los talleres, sino que también mejoraría significativamente la experiencia del cliente. Las razones clave que respaldan nuestra decisión son las siguientes:

- Disponibilidad 24/7: Un chatbot puede estar disponible las 24 horas del día, los 7 días de la semana, lo que permite a los clientes obtener respuestas a sus preguntas en cualquier momento, incluso fuera del horario comercial del taller.
- Respuestas consistentes: Al tener respuestas previamente definidas, el chatbot garantiza que la información proporcionada a los clientes sea coherente y precisa en todo momento, lo que ayuda a construir la confianza de los clientes.
- Reducción de la carga de trabajo: Los talleres suelen tener personal limitado, y los mecánicos a menudo tienen que dividir su tiempo entre la reparación de vehículos y la atención al cliente. Al automatizar las respuestas a preguntas comunes, se libera a los trabajadores de estas tareas y les permite centrarse en su labor principal.
- Ahorro de tiempo y recursos: La implementación de un chatbot puede ayudar a reducir los costos operativos al reducir la necesidad de contratar personal adicional para la atención al cliente. Además, ahorra tiempo tanto para el taller como para los clientes, ya que las respuestas son instantáneas.

En un comienzo habíamos planeado implementar el chatbot en Whatsapp, donde hicimos un análisis de los distintos planes y librerías que ofrecía la plataforma para desarrollarlo. Aquí se observa un FODA de dos posibles soluciones que habíamos definido: una librería no oficial y la API oficial de Whatsapp.

FORTALEZAS

FODA - whatsapp-web.js (librería no oficial)

DEBILIDADES

- 1. Flexibilidad:** WhatsApp Web es una biblioteca de código abierto que permite interactuar con WhatsApp a través de una interfaz web. Proporciona una gran flexibilidad para desarrollar chatbots personalizados con funcionalidades específicas.
- 2. Control total:** Al utilizar whatsapp-web.js, tenemos un control total sobre el flujo de mensajes y las interacciones con los usuarios. Podemos personalizar y adaptar el chatbot según nuestras necesidades.
- 3. Comunidad activa:** Existe una comunidad activa de desarrolladores que utilizan whatsapp-web.js, por lo cual podemos encontrar recursos, ejemplos de código y soporte en línea fácilmente.
- 4. Código abierto y gratuito:** Cualquier persona puede utilizar esta librería sin costo alguno.

- 1. Mantenimiento y actualizaciones:** WhatsApp puede realizar cambios en su plataforma que pueden afectar la compatibilidad o funcionalidad de whatsapp-web.js. Esto puede requerir actualizaciones y mantenimiento constantes por parte de los desarrolladores para garantizar un funcionamiento correcto.
- 2. Limitaciones de funcionalidad:** Aunque whatsapp-web.js ofrece muchas características y funcionalidades, puede haber ciertas limitaciones en comparación con la API oficial de WhatsApp. Esto puede restringir algunas acciones o capacidades avanzadas del chatbot.

OPORTUNIDADES

- 1. Integración con otros servicios:** Al utilizar whatsapp-web.js, podemos aprovechar esta biblioteca para integrar el chatbot de WhatsApp con otros servicios y sistemas, como bases de datos, APIs externas o herramientas de automatización.
- 2. Escalabilidad:** Si bien whatsapp-web.js es una biblioteca popular y bien mantenida, también es posible escalar y adaptar el chatbot según nuestras necesidades a medida que nuestra base de usuarios crece.

AMENAZAS

- 1. Cambios en las políticas de WhatsApp:** WhatsApp puede realizar cambios en sus políticas y términos de servicio que pueden afectar la viabilidad de utilizar whatsapp-web.js para desarrollar chatbots. Es importante estar al tanto de las actualizaciones y cumplir con las políticas de WhatsApp para evitar problemas de bloqueo de cuentas.
- 2. Competencia de soluciones oficiales:** WhatsApp también ofrece una API oficial a través de su plataforma de negocios. Pueden surgir soluciones alternativas o competidores que utilicen la API oficial de WhatsApp, lo que puede ofrecer ventajas adicionales en términos de funcionalidad, soporte y confiabilidad.

FORTALEZAS

FODA - whatsapp API oficial

DEBILIDADES

- 1. Funcionalidad completa:** La API oficial de WhatsApp brinda acceso a todas las características y funcionalidades ofrecidas por WhatsApp. Esto incluye el envío de mensajes multimedia, la gestión avanzada de interacciones y la integración con otras herramientas empresariales.
- 2. Soporte y confiabilidad:** Al utilizar una plataforma paga que ofrece acceso a la API oficial, podemos beneficiarnos de un soporte técnico dedicado y una mayor confiabilidad en comparación con soluciones de terceros.
- 3. Escalabilidad y crecimiento:** La API oficial y las plataformas pagas están diseñadas para manejar altos volúmenes de mensajes y usuarios, lo que te permite escalar tu chatbot a medida que tu base de usuarios crece.

- 1. Costo:** Utilizar la API oficial de WhatsApp a través de una plataforma paga implica un costo adicional en comparación con soluciones gratuitas o de código abierto.
- 2. Dependencia externa:** Al utilizar una plataforma paga, estás confiando en un proveedor externo para el acceso y la gestión de la API oficial. Esto implica una dependencia adicional y puede haber limitaciones en cuanto a personalización y control total sobre el funcionamiento del chatbot.

OPORTUNIDADES

- 1. Integración empresarial:** Al utilizar la API oficial a través de una plataforma paga, puedes aprovechar las integraciones con otras herramientas y servicios empresariales, como CRM, sistemas de atención al cliente y plataformas de automatización.
- 2. Mejora continua:** La API oficial de WhatsApp está sujeta a actualizaciones y mejoras constantes. Al utilizar una plataforma paga, es más probable que recibas actualizaciones y nuevas funcionalidades de manera oportuna, lo que te permite mantener tu chatbot actualizado y relevante.

AMENAZAS

- 1. Competencia y alternativas:** Existen soluciones alternativas y competidores que ofrecen acceso a la API oficial de WhatsApp a través de plataformas pagas. La aparición de nuevas opciones puede generar una mayor competencia en términos de precios, funcionalidades y calidad de servicio.
- 2. Cambios en las políticas de WhatsApp:** WhatsApp puede realizar cambios en sus políticas y términos de servicio que pueden afectar la viabilidad y el funcionamiento de la API oficial. Es importante estar al tanto de las actualizaciones.

A partir del análisis realizado en base a los FODA, optamos por utilizar Telegram para el sistema del ChatBot, con posibilidad de implementarlo en Whatsapp en un futuro.

Implementación

En el proceso de implementación de nuestro sistema en producción, evaluamos varias opciones para su despliegue y funcionamiento continuo. Una posibilidad era alojar nuestro sistema en un servidor existente, lo que nos iba a permitir mantenerlo operativo 24/7. Sin embargo, esta opción presentaba desafíos en términos de configuración compleja, mantenimiento (dado que el servidor era antiguo y no se había mantenido recientemente), costos y escalabilidad.

Posteriormente, exploramos dos plataformas que facilitan el despliegue de los diferentes componentes de nuestra aplicación:

- Para el componente del front-end: Elegimos Vercel, una plataforma unificada que facilita los despliegues, la gestión y la escalabilidad de las aplicaciones. Ofrece integración con un pipeline para implementar CI/CD, lo que permite la automatización de estos procesos. Esta robusta solución resultó ser la más adecuada para nuestro componente de cliente.
- Para el componente de backend y base de datos: Optamos por una plataforma llamada Somee. Esta plataforma es ideal para desplegar proyectos nativos de ASP.NET y bases de datos SQL Server. Ofrece la posibilidad de escalar los planes según aumenta el tráfico, y proporciona un plan gratuito que permite probar e interactuar con el sistema desplegado. El despliegue se realiza a través de FTP utilizando la herramienta de publicación nativa de Visual Studio 2022.

BENEFICIOS POST-IMPLEMENTACIÓN

Tras la puesta en producción del sistema, los beneficios emergen tanto para los clientes (usuarios) como para los talleres, reflejando el impacto positivo del sistema en ambos grupos.

Para los Clientes:

Los clientes disfrutarán de una aplicación accesible desde cualquier navegador, diseñada para facilitar la programación de turnos en talleres asociados al sistema. La

interfaz, intuitiva y sencilla, permite a los usuarios sumergirse en un proceso eficiente de solicitud y consulta de turnos. Este enfoque centrado en el usuario ofrece una experiencia gratificante, gracias a la constante retroalimentación proporcionada por el sistema, que mantiene a los usuarios informados sobre el progreso y los estados de sus citas.

Para los Talleres:

Una vez asociados al sistema, los talleres podrán comenzar a recibir y gestionar turnos de manera eficiente, adaptando su carga de trabajo a los recursos disponibles. Esta gestión se ve reforzada por un indicador de carga de trabajo proporcionado por el sistema, que ayuda en la toma de decisiones. Los talleres también podrán manejar datos relevantes como ingresos, administración de personal, métricas de turnos, y el control de inventario de stock. Un calendario integrado mostrará las fechas de entrega programadas para cada vehículo.

Estas herramientas permitirán a los talleres incrementar su productividad, optimizar las operaciones y minimizar las interrupciones y pérdidas de tiempo. Además, la organización eficaz de turnos según los recursos disponibles y la comunicación constante con los clientes, tanto a través del sistema como por email, facilitará la gestión del tiempo y mejorará la organización general del taller.

IMPACTOS

IMPACTO ECONÓMICO (estudio de costos)

Para la realización del sistema, el impacto de los costos del mismo es mínimo, ya que el único costo que se incurre es el alojamiento del backend y base de datos en la plataforma Somee, donde se pagaron \$59,40 USD anuales.

Impacto Económico en los Usuarios de la Aplicación: El uso de nuestra aplicación no solo contribuye a un impacto medioambiental positivo, sino que también ofrece beneficios económicos significativos a los usuarios. Anteriormente, sin un sistema de programación eficiente, los usuarios podrían enfrentarse a situaciones en las que, después de llegar al taller, se les informaba que debían regresar otro día o que su vehículo no podía ser atendido inmediatamente. Nuestra aplicación elimina esta incertidumbre, asegurando que los clientes reciban confirmaciones claras y fiables sobre la disponibilidad del taller y el momento exacto en que deben presentarse.

Consideremos que un usuario realiza un viaje innecesario al taller, cubriendo un total de 16 kilómetros en total (8 kilómetros de ida y 8 de vuelta). Si el vehículo tiene un consumo de 8 litros de Nafta Súper cada 100 kilómetros, este viaje consumiría aproximadamente 1.28 litros de combustible ($16 \text{ km} / 100 \text{ km} * 8 \text{ litros}/100 \text{ km}$). Con un precio de 639 pesos por litro de Nafta Súper, el costo innecesario de este viaje sería de aproximadamente 817.92 pesos ($1.28 \text{ litros} * 639 \text{ pesos/litro}$). Este gasto, que se puede evitar utilizando nuestra aplicación, representa un ahorro considerable para los usuarios, especialmente si se considera en un contexto de uso frecuente del taller. Sin embargo, creemos que el mayor impacto de nuestro sistema se vería reflejado en el ámbito medioambiental.

Por el lado de los talleres, estos también se verán beneficiados. Al automatizar la gestión de turnos, los talleres pueden aprovechar al máximo sus recursos y aumentar la eficiencia de su personal. Esto significa menos tiempo perdido y más trabajo realizado, lo que se traduce directamente en ahorro de dinero.

Al mejorar la forma en que los talleres manejan sus operaciones diarias, nuestra aplicación evita errores y retrasos que podrían resultar bastante costosos. Por ejemplo, al

tener un registro claro de todos los turnos programados y los servicios solicitados, los talleres pueden evitar confusiones y malentendidos que podrían generar reparaciones incorrectas o insatisfactorias. Esto no solo protege la imagen del taller, sino que también evita posibles reclamos de los clientes que podrían resultar en costos adicionales.

IMPACTO SOCIAL

La implementación de nuestra aplicación también tiene un impacto social positivo al abordar varios aspectos relacionados con la experiencia de los usuarios y la interacción entre talleres y clientes.

Mejora en la Experiencia del Cliente: Nuestra aplicación proporciona a los clientes una experiencia más eficiente y conveniente al programar y gestionar sus citas de manera efectiva. Al eliminar la incertidumbre y la necesidad de realizar viajes innecesarios, los usuarios experimentan una mayor satisfacción y comodidad en la gestión de los servicios para sus vehículos. Esto contribuye a una relación más positiva entre los clientes y los talleres mecánicos, mejorando la calidad de servicio percibida.

Fomento de la Transparencia y Confianza: La aplicación busca fortalecer la transparencia en la relación entre los talleres y los clientes. Al proporcionar información clara sobre la disponibilidad de turnos, políticas del taller y progreso de las reparaciones, se establece una comunicación más abierta y transparente. Esto puede contribuir a construir la confianza entre ambas partes, ya que los clientes se sienten informados y comprenden mejor el proceso de servicio.

Reducción de estrés y mejora en la planificación: Al evitar situaciones de incertidumbre y esperas innecesarias, los usuarios experimentan menos estrés al utilizar nuestros servicios. La aplicación permite una mejor planificación de los servicios para el vehículo, facilitando la organización de la rutina diaria de los usuarios.

IMPACTO MEDIOAMBIENTAL

La implementación de nuestra aplicación tiene un impacto medioambiental significativamente positivo, principalmente en la reducción de la huella de carbono asociada con los viajes de los vehículos. Al optimizar la gestión de citas para talleres de vehículos, la aplicación ayuda a disminuir los viajes innecesarios, lo que conduce a una reducción en las emisiones de gases de efecto invernadero.

Reducción de Emisiones por Menos Viajes Innecesarios: Al evitar que los usuarios realicen viajes innecesarios al taller, se reduce la cantidad total de kilómetros recorridos. Esto disminuye directamente las emisiones de CO₂ y otros contaminantes emitidos por los vehículos. Esta reducción es especialmente significativa en áreas urbanas, donde la densidad del tráfico y los problemas de congestión pueden exacerbar la emisión de gases nocivos.

CONCLUSIONES

En resumen, esperamos que este proyecto logre sus objetivos y pueda ponerse a prueba en el mundo real beneficiando a las dos entidades relacionadas, talleres y clientes. Si bien el sistema se centra principalmente en mejorar las operaciones de los talleres, también hemos integrado características valiosas para los clientes. Estas características no solo fomentan el uso del sistema por parte de los clientes, sino que esperamos faciliten una comunicación fluida y beneficiosa entre ambas partes.

El sistema no solo está enfocado en la eficiencia operativa, sino que también apunta a generar un cambio positivo en la sociedad. A través de la transparencia y trazabilidad que ofrece, el sistema busca elevar el estándar de servicio en los talleres, proporcionándoles una ventaja competitiva y una imagen destacada en el mercado. Esta diferenciación es vital para aquellos talleres que se comprometen con nuestra cultura y valores, fortaleciendo la confianza con sus clientes.

Como conclusión personal de ambos, podemos decir que el proyecto fue todo un logro gracias al trabajo en equipo que realizamos. Nos organizamos de gran manera, dividiendo las tareas de acuerdo a nuestras fortalezas y habilidades. Cada uno pudo aportar sus ideas y perspectivas, lo que logró darle creatividad al proyecto.

Asimismo, supimos enfrentar los desafíos que surgieron en el camino, trabajando juntos para encontrar soluciones efectivas. La comunicación y confianza entre ambos fue algo clave para realizar este gran proyecto.

FUENTES

Competencias:

- <https://www.tallergp.com/ar/>
- <https://www.ts-sistemas.com.ar/motoreasy/>
- <https://talleron.com.ar/index.html>

Conceptos:

- Aplicación Web:
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-una-web-app-y-que-clases-hay/>
- Aplicación Móvil: <https://anincubator.com/que-es-una-aplicacion-movil/>
- Arquitectura en capas:
<https://learn.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/com-mon-web-application-architectures#what-are-layers>
- Arquitectura monolítica y Microservicios:
<https://www.chakray.com/es/devops-arquitectura-monolitica-vs-microservicios/>
- ORM: <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>
- API:
<https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>

Tecnologías

- .NET: <https://learn.microsoft.com/es-es/dotnet/>
- Java: <https://aws.amazon.com/es/what-is/java/>
- Python: <https://aws.amazon.com/es/what-is/python/>
- React: <https://kinsta.com/es/base-de-conocimiento/que-es-react-js/#qu-es-react>
- Angular: <https://desarrolloweb.com/home/angular>
- SQL Server:
<https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver16>
- MySQL y PostgreSQL:
<https://aws.amazon.com/es/compare/the-difference-between-mysql-vs-postgresql/>