



MEMORIA FINAL DEL PROYECTO

AutoSelect: Soluciones de Alquiler Premium

CICLO FORMATIVO DE GRADO SUPERIOR

DESARROLLO DE APICACIONES

MULTIPLATAFORMA

CURSO 2022-2023

AUTOR:

Jorge Sánchez Berrocoso

TUTOR:

José Manuel Pérez Lobato

DEPARTAMENTO DE INFORMÁTICA Y COMUNICACIONES

I.E.S. LUIS VIVES



Resumen

En este proyecto me enfoco en crear una tienda moderna y centrada en el cliente para el alquiler de automóviles premium. Esto responde a la necesidad creciente en la población de personas que no se pueden pagar automóviles de alta gama.

La documentación abarca todo el proceso de investigación, diseño y desarrollo de una plataforma de alquiler de vehículos.

El objetivo de este proyecto es no solo cumplir con los requisitos académicos, sino también establecer las bases para una posible implementación real en el mercado en el futuro. El proyecto ofrece una visión completa y práctica de una tienda de alquiler de automóviles en línea.



Contenido

Resumen	2
1. Introducción	4
1.1 Justificación	5
1.2 Objetivos.....	6
1.3 Alcance.....	7
2 Requisitos.....	8
2.1 Requisitos Funcionales	8
2.2 Requisitos No Funcionales	9
3. Diagramas	10
3.1 Diagrama de Clases.....	10
3.2 Diagrama de Casos de Uso	11
4. Metodologías Agiles	13
5. Competencia.....	14
6 Posibilidades Tecnológicas.....	16
6.1 Backend	16
6.2 Bases de datos	18
6.3 Frontend.....	20
7 Frontend	22
7.1 Herramientas y Tecnologías Utilizadas.....	23
7.2 Implementación.....	27
8 Backend.....	37
8.1 Herramientas Utilizadas	39
9 Base de Datos.....	48
10 Despliegue.....	50
10.1 Contratación de Servicios de Hosting y Dominio:	50
10.2 Preparación del Servidor y Despliegue de la Aplicación:	53
10.3 Configuración de Nginx para Redirección de Tráfico:	54
10.4 Implementación de Certificados SSL con Let's Encrypt:.....	56
10.5 Dockerización y Orquestación con Docker Compose:	57
10.6 Modelo de Servicio en la Nube	58
11 Coste del proyecto	59
12 Conclusiones y Trabajo Futuro.....	60
13 Desafíos y Resoluciones a lo Largo del Proyecto.....	61
14 Enlaces	62
15 Bibliografía	63

1. Introducción

Este proyecto se centra en el desarrollo de una plataforma en línea especializada en el alquiler de automóviles de alta gama. La aplicación consta de dos roles principales: el cliente y el administrador.

Administrador

El administrador desempeña un papel crucial en la gestión de la plataforma. A través de una interfaz de usuario intuitiva, tiene la capacidad de administrar los puntos de localización de las tiendas, incorporando coordenadas de latitud y longitud en el mapa. Además, tiene el control sobre la gestión del catálogo de automóviles, que incluye cuatro tipos distintos: "Furgoneta", "Coche", "Camión" y "Motocicleta". El administrador puede añadir, modificar o eliminar vehículos, así como gestionar los usuarios, modificar reservas y anularlas, y monitorizar incidencias.

Cliente

El usuario cliente cuenta con funcionalidades específicas diseñadas para hacer su experiencia de alquiler más fluida. Puede realizar reservas de automóviles, gestionarlas mediante la posibilidad de anularlas o descargarlas, y expresar sus valoraciones sobre los vehículos utilizados.

El propósito fundamental de esta aplicación es simplificar y agilizar el proceso de alquiler de automóviles de alta gama, ofreciendo una experiencia rápida y eficiente.

En las secciones siguientes, se abordarán en detalle los aspectos clave del proyecto, desde la conceptualización hasta la implementación tecnológica.



1.1 Justificación

La creación de esta aplicación de vehículos de alta gama en línea se debe a las siguientes razones:

- **Demanda del Mercado:** Existe una creciente demanda de clientes que buscan alquilar automóviles exclusivos y cómodos para ciertos eventos. Esta aplicación busca satisfacer esa necesidad ofreciendo una forma fácil de acceder a vehículos de prestigio.
- **Rentabilidad:** La industria de alquiler de automóviles de alta gama suele ser mas rentable que otras industrias debido a que se suelen pagar tarifas mas altas por los automóviles.
- **Experiencia de vehículos de alta gama:** Dar accesibilidad a personas a conducir vehículos que en condiciones normales estarían muy lejos de su presupuesto dándoles una experiencia única.

Y por estas razones me inclino más a crear una aplicación web de alquiler de automóviles de alta gama con el fin de crear una aplicación que nos permita simplificar y agilizar todo el proceso.



1.2 Objetivos

Los objetivos de este proyecto son:

- **Autenticación y Registro de Usuarios:** Implementar un mecanismo de registro que permita a los usuarios establecer una cuenta personalizada y acceder a la plataforma a través de un sistema de autenticación seguro y eficiente.
- **Reservas en Línea en Vivo:** Crear un sistema interactivo de reservas que habilite a los usuarios para escoger y reservar fechas específicas, verificar la disponibilidad de los vehículos en tiempo real y concretar la reserva con facilidad.
- **Valoraciones:** Incorporar una funcionalidad de comentarios, posibilitando a los clientes compartir sus opiniones y experiencias sobre los vehículos y el servicio, lo que a su vez, servirá para futuros usuarios.
- **Catálogo Digital de Vehículos:** Desarrollar una interfaz de usuario intuitiva y visualmente atractiva que exhiba un inventario exhaustivo de vehículos disponibles, con detalles como imágenes, características de los vehículos y tarifas, mientras se ofrece a los administradores herramientas para la gestión del catálogo, incluyendo la adición, eliminación y actualización de los datos de los coches.
- **Geolocalización y Direcciones:** Añadir un componente de mapa interactivo que muestre la ubicación de las tiendas de alquiler.



1.3 Alcance

El alcance del proyecto "AutoSelect" incluye objetivos como:

- **Desarrollo de una Plataforma Integral:** Elaborar una página web intuitiva que agilice el proceso de búsqueda, selección y reserva de vehículos a través de una interfaz en línea.
- **Sistema de Reservas Avanzado:** Establecer un sistema de reservas ágil y en tiempo real, proporcionando a los usuarios un método eficaz y sencillo para asegurar sus vehículos preferidos.
- **Interfaz de Usuario Innovadora:** Construir una interfaz visualmente atractiva y fácil de navegar que ofrezca un catálogo detallado de los coches disponibles, incluyendo imágenes, características técnicas y precios.
- **Comunicación Mejorada:** Mejorar la interacción directa entre los clientes y la empresa de alquiler mediante un sistema de valoraciones en cada ficha de vehículo.
- **Desarrollo Profesional:** Aprovechar la oportunidad para ganar experiencia valiosa en el desarrollo de soluciones web y móviles, lo que contribuirá al crecimiento de habilidades técnicas en el ámbito tecnológico.



2 Requisitos

2.1 Requisitos Funcionales

Autenticación de Usuarios:

- Implementación de una funcionalidad de registro que permita la creación de cuentas personales en la plataforma.
- Proceso de inicio de sesión robusto y ágil para garantizar la seguridad y eficiencia del acceso a la plataforma.

Gestión de Reservas Dinámica:

- Capacidad para que los usuarios elijan y programen las fechas de recogida y devolución de los vehículos.
- Visualización en tiempo real de la disponibilidad de vehículos, permitiendo a los usuarios reservar de manera práctica y rápida.

Sistema de Feedback Integrado:

- Posibilidad para los usuarios de publicar opiniones y valoraciones sobre los vehículos y la experiencia de alquiler.
- Accesibilidad a las reseñas por parte de otros usuarios.

Catálogo Interactivo:

- Exhibición de un catálogo completo y actualizado de vehículos disponibles, con detalles como imágenes, características técnicas y precios.
- Herramientas administrativas para la gestión eficiente del catálogo, permitiendo añadir, eliminar y actualizar la información de los vehículos y usuarios.



2.2 Requisitos No Funcionales

Desarrollo Web:

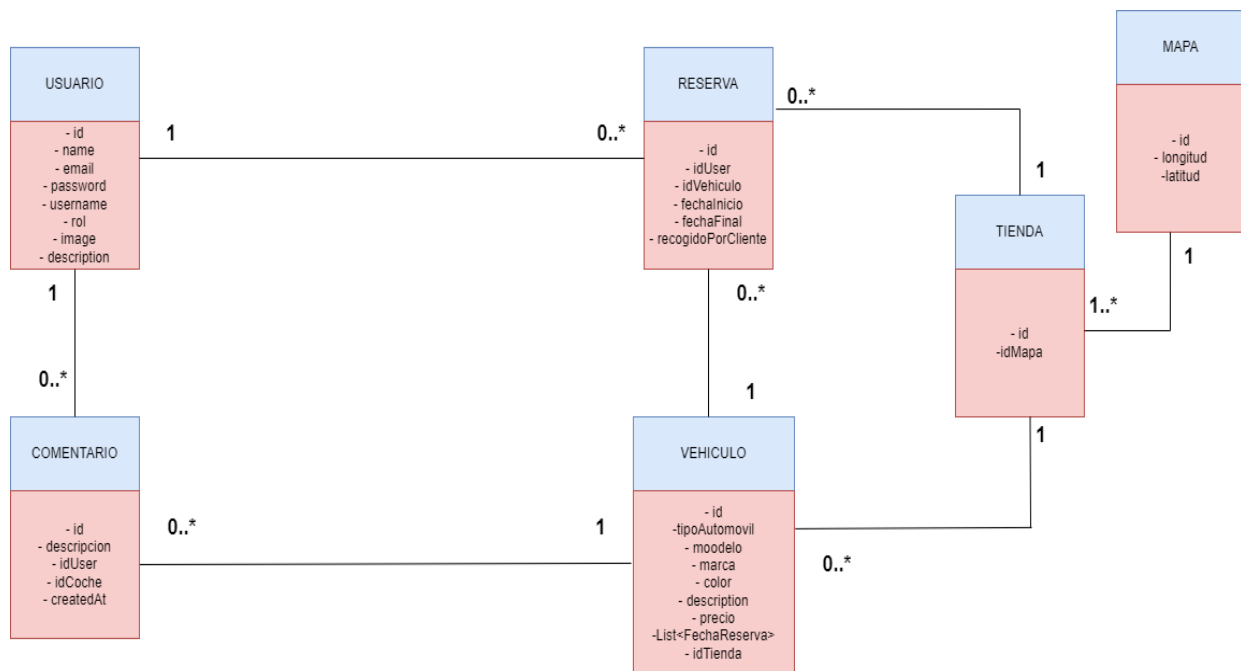
- Construcción del front-end utilizando el framework Angular para una experiencia de usuario óptima.
- Programación del back-end con Kotlin, asegurando un rendimiento eficiente y mantenible.

Seguridad Informática Avanzada:

- Seguridad del back-end reforzada con la implementación de tokens JWT Bearer.
- Seguridad del servidor mediante SSL cifrando la comunicación entre el servidor y el cliente.

3. Diagramas

3.1 Diagrama de Clases

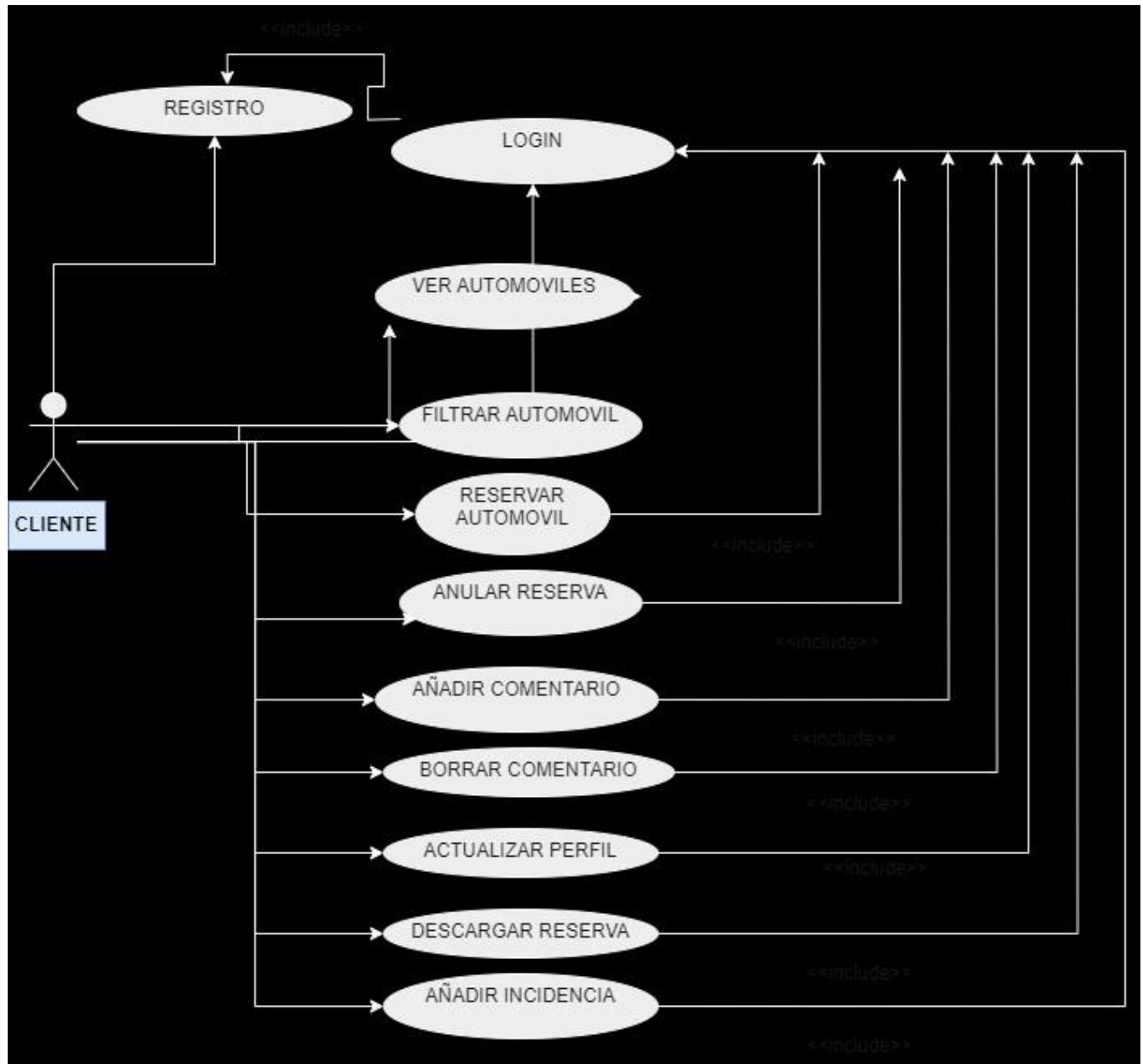


Entidades y Sus Atributos

- **Usuario:** Representa a las personas que utilizan la aplicación. Cada usuario tiene un ID único, nombre, email, contraseña, nombre de usuario, rol en la aplicación, imagen y una descripción personal.
- **Comentario:** Contiene opiniones de los usuarios sobre los vehículos. Cada comentario tiene su propio ID, una descripción, el ID del usuario que lo creó, el ID del vehículo al que hace referencia y una marca de tiempo de su creación.
- **Reserva:** Gestiona las reservas que los usuarios hacen de los vehículos. Incluye un ID único, el ID del usuario, el ID del vehículo reservado, fechas de inicio y final de la reserva, y un indicador de si la recogida es por parte del cliente.
- **Vehículo:** Detalla los vehículos disponibles para alquilar. Cada uno tiene un ID, un tipo de automóvil, marca, modelo, color, descripción, precio, una lista de fechas de reserva y el ID de la tienda donde se encuentra.
- **Tienda:** Corresponde a los establecimientos físicos donde se alquilan los vehículos. Cada tienda tiene un ID único y está asociada con uno o más mapas que indican su ubicación.
- **Mapa:** Proporciona las coordenadas geográficas de las tiendas con un ID, longitud y latitud.

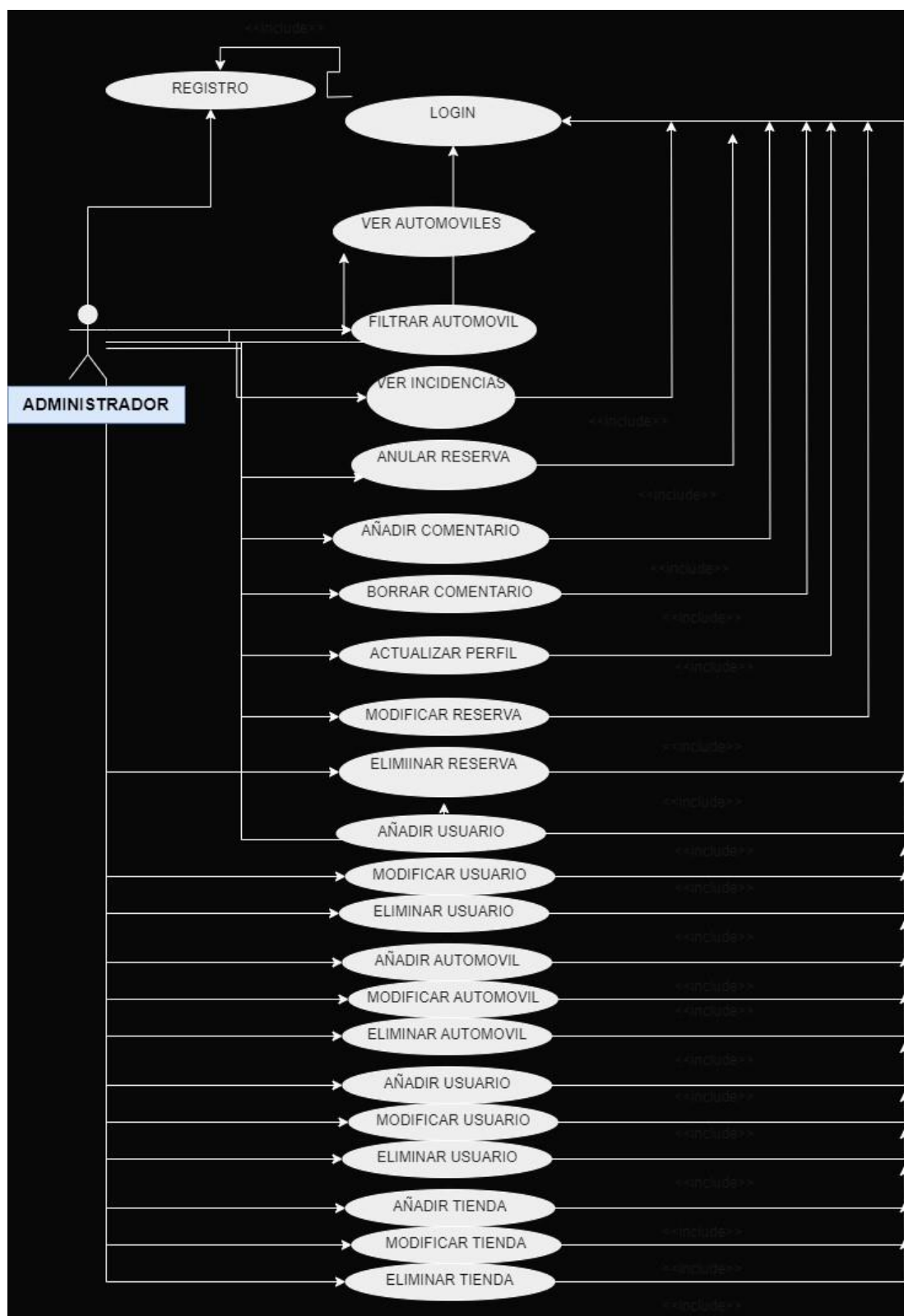
3.2 Diagrama de Casos de Uso

Cliente





Administrador



4. Metodologías Ágiles

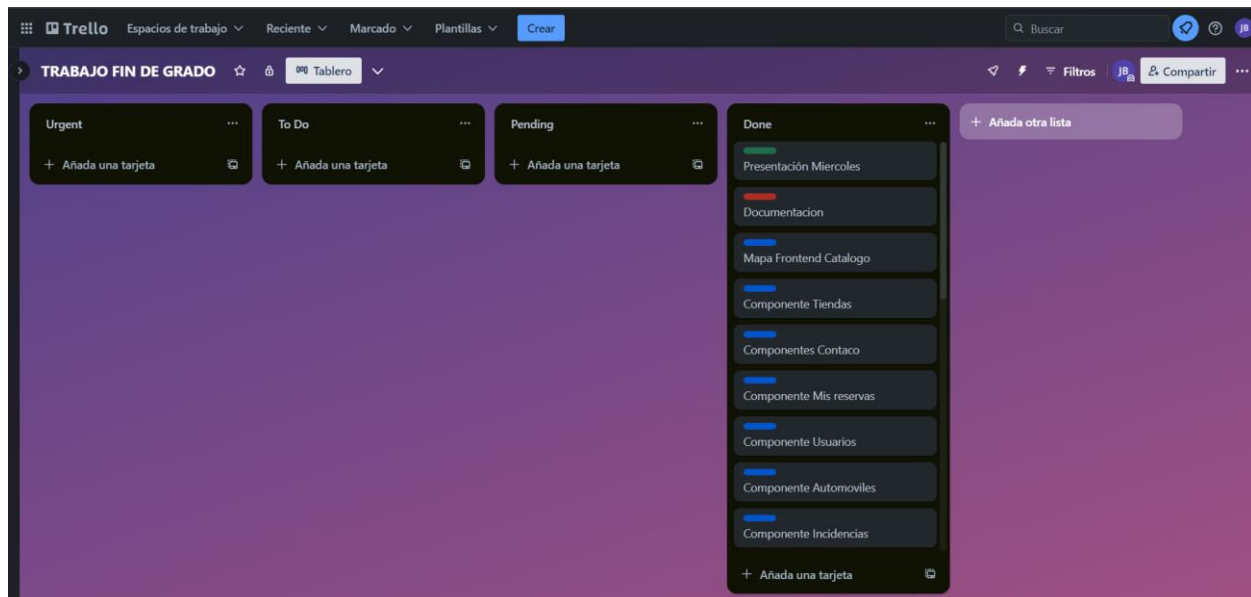
En este proyecto, he adoptado la metodología Scrum, una forma ágil de gestión y planificación de proyectos. Esta metodología se caracteriza por su enfoque incremental, con un ciclo de desarrollo dividido en sprints de dos semanas.

Organización del Sprint

1. Planificación del Sprint: Al inicio de cada sprint, seleccionaba tareas del trello, basándome en la prioridad.
2. Objetivos del Sprint: Establecía objetivos claros y alcanzables para cada sprint, alineados con las metas del proyecto.

Ejecución y Monitoreo

Tablero de Trello: Utilicé un tablero de Trello, para visualizar el avance de las tareas durante el sprint, con columnas como "Urgent", "To Do", "Pending" y "Done".



5. Competencia

Hertz:

Hertz es una de las empresas de alquiler de automóviles más conocidas a nivel mundial. Su aplicación móvil permite a los usuarios reservar coches en una amplia variedad de ubicaciones y ofrece opciones de alquiler a corto y largo plazo. Proporciona un amplio catálogo de vehículos, desde económicos hasta de lujo.



Enterprise Rent-A-Car:

Enterprise es otra empresa líder en el alquiler de coches. Su aplicación permite a los usuarios buscar vehículos por tipo, marca y ubicación. También ofrecen opciones de alquiler a largo plazo y programas de fidelidad para clientes frecuentes.





Turo:

Turo se diferencia al permitir que los propietarios de automóviles alquilen sus vehículos a otros usuarios. Su plataforma conecta a arrendadores y arrendatarios, lo que brinda una amplia variedad de opciones de automóviles, desde automóviles económicos hasta vehículos exóticos.





6 Posibilidades Tecnológicas

6.1 Backend

Spring

Ventajas:

- Utiliza el patrón de inyección de dependencias.
- Está dividido en módulos.
- Se centra en la programación basada en interfaces, lo que mejora la testabilidad y la calidad del código.

Desventajas:

- Es bastante complejo para un iniciado en el desarrollo de software.
- En las actualizaciones del framework es verdaderamente complicado migrarlo.





Ktor

Ventajas:

- Como proyecto de código abierto, fomenta la colaboración y la innovación.
- Es fácil y rápido de aprender.
- Ofrece una sintaxis simple y potente
- Destaca por su naturaleza ligera, aumentando la eficiencia del rendimiento.
- Multiplataforma
- Incorpora las corrutinas de Kotlin, optimizando las operaciones asíncronas.
- Cuenta con documentación completa y de fácil comprensión.

Desventajas:

- Su uso está limitado exclusivamente a proyectos en Kotlin.
- Al ser una herramienta más reciente, no tiene una extensa comunidad al contrario que spring.



6.2 Bases de datos

MariaDB

Ventajas:

- Mejorado Rendimiento con Grandes Datos: Optimizado para manejar grandes volúmenes de información.
- Mayor Seguridad: Ofrece características de seguridad mejoradas.
- Código Abierto: Disponible para uso y modificación.

Desventajas:

- Compatibilidad Limitada.
- Soporte Limitado.



MySQL

Ventajas:

- MySQL es de **distribución libre** y gratuita.
- MySQL es **Open Source**.
- Es **veloz** al realizar operaciones, y garantiza un buen rendimiento de las aplicaciones.
- Es **multiplataforma**.

Desventajas:

- No es **intuitivo**, en comparación con otros programas.
- No maneja de manera tan **eficiente** una base de datos con un tamaño muy grande.



MongoDB

Ventajas:

- Eficiencia de Recursos: Funciona bien con recursos limitados.
- Código Abierto y Buena Documentación: Ampliamente accesible y bien documentado.
- Esquema Dinámico: Permite un esquema de datos flexible.
- Escalabilidad y Flexibilidad: Excelente para adaptarse y escalar según las necesidades.
- Optimización en Grandes Volúmenes de Datos: Maneja eficazmente grandes cantidades de datos.
- Basado en JavaScript: Integración natural con tecnologías web modernas.

Desventajas:

- Ausencia de Joins: No soporta operaciones de join tradicionales.
- Limitaciones en Consultas SQL.
- Transacciones Complejas Limitadas.



6.3 Frontend

Flutter

Ventajas:

- Desarrollo ágil con recarga en caliente.
- Compatible con múltiples plataformas.
- Bibliotecas extensas de widgets para UI.
- Rendimiento cercano al nativo.
- Código abierto.

Desventajas:

- Utiliza Dart, un lenguaje menos común.
- Mayor tamaño de archivo comparado con otros frameworks.
- Limitaciones en la integración con hardware.



React

Ventajas:

- Agiliza el desarrollo de aplicaciones.
- Basado en JavaScript, ampliamente utilizado.
- Recarga de cambios en caliente.
- Buen rendimiento y apariencia nativa.

Desventajas:

- Desafíos en la depuración.
- Dependencia de desarrolladores para funciones nativas.
- Gestión subóptima de la memoria.
- Actualizaciones lentas de funcionalidades nuevas.
- Rendimiento reducido con funciones complejas.



Angular

Ventajas:

- Usa TypeScript, mejorando la calidad del código.
- Desarrollo eficiente para aplicaciones web y móviles.
- Amplio soporte de Google.
- Facilidad en la escritura de pruebas.
- Alta productividad y flexibilidad en lenguajes.

Desventajas:

- Complejidad en ciertos aspectos del framework.
- Curva de aprendizaje desafiante.
- Tamaño relativamente grande de las aplicaciones.



ANGULAR



7 Frontend

Para el desarrollo de mi proyecto, he optado por utilizar Angular como el framework principal. Esta decisión se basó en varias características clave y ventajas que Angular ofrece, adecuándose perfectamente a las necesidades y objetivos de mi aplicación. Las características que ofrece son:

- **Arquitectura Basada en Componentes:** Angular promueve una estructura modular y reutilizable, lo cual es ideal para mi proyecto, que requiere una organización clara y la posibilidad de reutilizar componentes en diferentes partes de la aplicación. Esta arquitectura mejora la mantenibilidad y facilita la escalabilidad del proyecto.
- **TypeScript:** La integración de TypeScript en Angular es otro factor crucial en mi elección. TypeScript ofrece un entorno de desarrollo más estructurado y seguro, con características de tipado estático y orientación a objetos. Esto contribuye a un código más limpio, menos propenso a errores y más fácil de mantener y escalar.
- **Adecuado para Aplicaciones Empresariales:** Dada la naturaleza y el alcance de mi proyecto, que se alinea con aplicaciones empresariales complejas y de gran escala, Angular emerge como la opción ideal debido a su capacidad para manejar aplicaciones robustas y su enfoque en la escalabilidad y mantenibilidad.





7.1 Herramientas y Tecnologías Utilizadas

Angular Material

Elección: Lo he elegido por su conjunto de componentes de interfaz de usuario que están adaptados a los principios de Material Design de Google. Ofrece una consistencia estética y una amplia gama de elementos interactivos que se integran a la perfección con el ecosistema de Angular.

Contribución al Proyecto: Angular Material ha sido la base sólida para la interfaz de usuario, proporcionando coherencia visual y una experiencia de usuario intuitiva.

```
login.component.html
src > app > components > login > login.component.html > ...
Go to component
1 <div id="loginContainer" class="container-fluid">
2   <div class="col-md-8"></div>
3   <div class="col-md-4">
4     <form class="ng-pristine ng-valid" [formGroup]="loginForm" (ngSubmit)="login()">
5       <div class="jumbotron vertical-center" style="margin-bottom: -40px;">
6         <div class="col-md-12 login-box" style="background-color: #ffffffec;padding: 24px; border-radius: 20px;">
7           <div style="color: #000000">
8             <h1>Alquiler de Automóviles</h1>
9           </div>
10          <div class="alert alert-danger" *ngIf="error" style="text-align: left;">
11            {{'Error al iniciar sesión'}}
12          </div>
13
14          <div class="col-lg-6 col-lg-offset-3 text-center">
15            <mat-spinner mode="indeterminate" *ngIf="isLoading" class="colorLogin" diameter="40"></mat-spinner>
16          </div>
17          <div>
18            <mat-form-field style="border-radius: 10px;">
19              <input mdinput type="text" autofocus class="form-control form-line" id="username" [(ngModel)]="username" name="username" placeholder=" Usuario" formControlName="username" />
20            </mat-form-field>
21          </div>
22          <div [attr.disabled]="showSpinner">
23            <mat-form-field>
24              <input mdinput type="password" placeholder=" Contraseña" class="form-control form-line" id="password" [(ngModel)]="password" name="password" formControlName="password" />
25            </mat-form-field>
26          </div>
27          <button type="submit" id="submit" class="btn btn-custom pull-right fs-12">Iniciar Sesión</button>
28          <button type="button" id="register" class="btn btn-custom pull-right fs-12" [routerLink]="'/register'" style="margin-left: 10px;">Registrarse</button>
29
30          <p>&nbsp;</p>
31        </div>
32      </div>
33    </form>
34  </div>
35</div>
```





Bootstrap

Elección: Lo he seleccionado por su eficiencia en la creación de diseños responsivos y su sistema de grid flexible. Es ampliamente conocido por su capacidad para adaptar rápidamente una interfaz de usuario a diferentes tamaños de pantalla, lo que lo hace ideal para garantizar la accesibilidad y la usabilidad en dispositivos móviles y de escritorio.

Contribución al Proyecto: La implementado con el objetivo de crear una interfaz intuitiva y responsiva debido a su sistema de grid flexible.





PrimeNG

Elección: He elegido PrimeNG para complementar Angular Material, proporcionando una gama aún más amplia de componentes de UI avanzados y específicos.

Contribución al Proyecto: Lo he utilizado en la sección de incidencias del administrador utilizando las capacidades de ordenamiento de las tablas de primeng.

```
Go to component
1 <p-table [value]="comentarios" [tableStyle]="{'min-width': '60rem'}">
2   <ng-template pTemplate="header">
3     <tr>
4       <th pSortableColumn="id">ID <p-sortIcon field="id"></p-sortIcon></th>
5       <th pSortableColumn="descripcion">Descripción <p-sortIcon field="descripcion"></p-sortIcon></th>
6       <th pSortableColumn="idUser">ID Cliente <p-sortIcon field="idCliente"></p-sortIcon></th>
7     </tr>
8   </ng-template>
9   <ng-template pTemplate="body" let-comentario>
10    <tr>
11      <td>{{comentario.id}}</td>
12      <td>{{comentario.descripcion}}</td>
13      <td>{{comentario.idCliente}}</td>
14    </tr>
15  </ng-template>
16 </p-table>
17
```



PRIMENG

Leaflet

Elección: La elección de Leaflet se debe a su eficiencia y facilidad para integrar mapas interactivos, esenciales para la visualización de datos geográficos.

Contribución al Proyecto: Leaflet ha permitido incorporar un mapa dinámico y personalizable en la aplicación, mejorando la representación y manipulación de datos geoespaciales de manera eficiente y efectiva.

```
const map = new Map('map').setView([40.42517, -3.68718], 13);
tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
  maxZoom: 19,
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

this.ubicaciones.forEach(location =>{
  marker([Number(location.latitud), Number(location.longitud)]).addTo(map)
});
```





7.2 Implementación

Componentes

Los componentes en Angular son elementos reutilizables, con los que se crea la aplicación.

Los componentes están compuestos por:

Un archivo HTML que es el que se va a mostrar en la interfaz de usuario.

Un archivo CSS donde crearemos el estilo de nuestro HTML.

Un archivo TS (Typescript) donde está la lógica del componente (propiedades, métodos).

```
@Component({
  selector: 'app-usuario',
  templateUrl: './usuario.component.html',
  styleUrls: ['./usuario.component.css']
})
export class UsuarioComponent implements OnInit {

  public usuarios : UserDto[]
  public user : UserDto;
  panelOpenState = false;
  filterUsername = '';
  public userFilter : UserFilter = new UserFilter();
  public roles : string[]

  constructor(private httpClient: HttpClient, private utilsService : UtilsService, public dialog: MatDialog){
    this.usuarios = [];
    this.user = new UserDto();
    this.roles = ['ADMINISTRADOR', 'CLIENTE']
  }
}
```

Para la generación de componentes solamente debemos indicar en terminal: `ng generate component /components/nombreCarpeta`

Servicios

Los servicios son objetos reutilizables que realizan tareas específicas y proporcionan funcionalidades compartidas en toda la aplicación. Los servicios en Angular se utilizan para encapsular la lógica de negocio, la manipulación de datos, la comunicación con servidores y otras operaciones que no están directamente relacionadas con la interfaz de usuario.

```
@Injectable({
  providedIn: 'root'
})
export class NewUsuarioPropertyService {

  private usuarioProperty: BehaviorSubject<UserDto> = new BehaviorSubject<UserDto>(new UserDto());
  private fileProperty : BehaviorSubject<File | null> = new BehaviorSubject<File | null>(null);

  constructor() { }

  public getUsuarioPropertyObservable():Observable<UserDto> {
    return this.usuarioProperty.asObservable();
  }

  public emitUsuarioProperty(usuario:UserDto):void {
    this.usuarioProperty.next(usuario);
  }

  public getFilePropertyObservable():Observable<File | null> {
    return this.fileProperty.asObservable();
  }

  public emitFileProperty(file:File | null):void {
    this.fileProperty.next(file);
  }
}
```



Clases

Typescript nos ofrece el uso de tipados en variables, tenemos las clases para definir tipos más complejos.

Para la generación de componentes solamente debemos indicar en terminal: `ng generate class /carpeta/nombreClase`.

```
export class UserDto {  
  public id: string;  
  public nombre: string;  
  public email: string;  
  public username: string;  
  public password : string;  
  public descripcion : string;  
  public image : string | null;  
  public rol: string;  
  
  constructor() {  
    this.id = '';  
    this.nombre = '';  
    this.email = '';  
    this.username = '';  
    this.password = '';  
    this.descripcion = '';  
    this.image = null;  
    this.rol = '';  
  }  
}
```



Navegación

Para facilitar la navegación entre las distintas vistas de la aplicación, he implementado un sistema de enrutamiento utilizando el módulo RouterModule de Angular. Este módulo me permite definir una colección de rutas donde cada ruta está asociada con un componente específico. Además, la seguridad de las rutas se maneja mediante la implementación de estrategias de autenticación y autorización, que permiten restringir el acceso a ciertas partes de la aplicación en función del estado de la sesión del usuario.

La interfaz de usuario aprovecha un sidebar o barra lateral para la navegación, que organiza los enlaces a las diferentes rutas de manera intuitiva y accesible. Este sidebar se diseñó utilizando el componente MatSidenav de Angular Material, que ofrece una integración estrecha con el sistema de enrutamiento de Angular.

Cada ítem del sidebar está vinculado a una ruta específica definida en la configuración del enrutador. Por ejemplo, un ítem puede llevar al usuario a la vista de perfil, mientras que otro puede redirigir a la lista de reservas. La visibilidad de ciertos ítems del sidebar se controla mediante directivas condicionales como `*ngIf`, que verifican el rol del usuario para mostrar elementos adecuados en función del rol, como opciones solo para administradores o para clientes.

```
const routes: Routes = [
  {path: '',redirectTo: 'catalogo',pathMatch: 'full'},
  {path:'login',component: LoginComponent},
  {path:'register',component: RegisterComponent},
  {path:'automovil',component: AutomovilesComponent},
  {path:'usuarios',component: UsuarioComponent},
  {path:'comentarios',component: ComentarioComponent},
  {path:'reservas',component: ReservasComponent},
  {path:'perfil',component: PerfilComponent},
  {path:'misreservas',component: MisreservasComponent},
  {path:'misvaloraciones',component: MisvaloracionesComponent},
  {path:'mapas',component: MapaComponent},
  {path:'contacto',component: ContactoComponent},
  {path:'catalogo',component: CatalogoComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```



```
SO to component
<div class="example-container" [class.example-is-mobile]="mobileQuery.matches" >
  <mat-toolbar class="example-toolbar" *ngIf="isNavbarVisible">
    <button mat-icon-button (click)="snave.toggle()" ><mat-icon [style.color]='white'>menu</mat-icon></button>
    <a class="navbar-brand-icon nb-logo" [routerLink]="['login']" routerLinkActive="active" style="cursor:auto;">
      
    </a>
    <!-- Este div vacío actúa como un espacio para centrar el logo. -->
    <div></div>
    <button mat-button style="position: absolute; right: 10px; color: white;" [routerLink]="login" *ngIf="iniciarSesion">
      <mat-icon [style.color]='white' class="icono-grande" >exit_to_app</mat-icon>
      Iniciar Sesión
    </button>

    <button mat-button style="position: absolute; right: 10px; color: white;" (click)="sesion()" [routerLink]="login" *ngIf="cerrarSesion">
      <mat-icon [style.color]='white' class="icono-grande" >exit_to_app</mat-icon>
      Cerrar Sesión
    </button>
  </mat-toolbar>

  <mat-sidenav-container class="example-sidenav-container" [style.marginTop.px]="mobileQuery.matches ? 56 : 0">
    <mat-sidenav #snave [mode]="mobileQuery.matches ? 'over' : 'side'" [fixedInViewport]="mobileQuery.matches" fixedTopGap="56">
      <mat-nav-list>

        <a mat-list-item [routerLink]="catalogo" *ngIf="isClienteVisible">
          <mat-icon>home</mat-icon>
          Catalogo
        </a>

        <a mat-list-item [routerLink]="perfil" *ngIf="perfil">
          <mat-icon>person</mat-icon>
          Mi Perfil
        </a>

        <a mat-list-item [routerLink]="misreservas" *ngIf="isClienteVisible">
          <mat-icon>event</mat-icon>
          Mis Reservas
        </a>

        <a mat-list-item [routerLink]="misvaloraciones" *ngIf="isClienteVisible">
          <mat-icon>star</mat-icon>
          Mis Valoraciones
        </a>

        <a mat-list-item [routerLink]="usuarios" *ngIf="isAdminVisible">
          <mat-icon>group</mat-icon>
          Usuarios
        </a>

        <a mat-list-item [routerLink]="automovil" *ngIf="isAdminVisible">
          <mat-icon>directions_car</mat-icon>
          Automoviles
        </a>

        <a mat-list-item [routerLink]="comentarios" *ngIf="isAdminVisible">
          <mat-icon>comment</mat-icon>
          Incidencias
        </a>

        <a mat-list-item [routerLink]="reservas" *ngIf="isAdminVisible">
          <mat-icon>calendar_today</mat-icon>
          Reservas
        </a>

        <a mat-list-item [routerLink]="['mapas']" *ngIf="isAdminVisible">
          <mat-icon>place</mat-icon>
          Tiendas
        </a>

      </mat-nav-list>
    </mat-sidenav>
    <mat-sidenav-content >
      <router-outlet></router-outlet>
    </mat-sidenav-content>
  </mat-sidenav-container>
</div>
```

```
    <a mat-list-item [routerLink]="['mapas']" *ngIf="isAdminVisible">
      <mat-icon>place</mat-icon>
      Tiendas
    </a>

    <a mat-list-item [routerLink]="contacto" *ngIf="isClienteVisible">
      <mat-icon>perm_contact_calendar</mat-icon>
      Contacto
    </a>
  </mat-nav-list>
</mat-sidenav>
<mat-sidenav-content >
  <router-outlet></router-outlet>
</mat-sidenav-content>
</mat-sidenav-container>
</div>
```



alquilaenmadrid.com/catalogo

AUTO SELECT CAR Cerrar Sesión

Selecciona un... Elige una fecha Elige una fecha Search

Resultados totales: 3 Automoviles

Filtros

Mapa de Madrid

Mercedes
Modelo -> GTR33
Color -> Rojo
Precio Por Día -> 25000\$
Reservar Comentarios

Audi
Modelo -> R8
Color -> Negro
Precio Por Día -> 15000\$
Reservar Comentarios

Llamadas a la API

En Angular, para realizar llamadas a una API, se utiliza el módulo HttpClient. Este módulo proporciona servicios para realizar operaciones HTTP, como GET, POST, PUT y DELETE.

```
private getUsersAll(){
  const url : string = 'https://alquilaenmadrid.com/api/users/listaUsuarios'
  //const url : string = 'http://localhost:6969/api/users/listaUsuarios'

  const token = localStorage.getItem('access_token');

  if (token) {
    const headers = new HttpHeaders({
      Authorization: `Bearer ${token}`
    });

    this.httpService.get(url, { headers }).toPromise().then((value: any) => {
      this.usuarios = value as UserDto[];
      console.log(this.usuarios)
    }).catch((error) => {
      console.log('Se ha producido un error al obtener los usuarios');
    });
  }
}
```



Llamadas a la API con JWT

Para realizar consultas en las que se necesita un Token, vamos a crear una cabecera(Header) el tipo de header *Bearer* y el token.

```
if (token) {  
  const headers = new HttpHeaders({  
    Authorization: `Bearer ${token}`  
  });  
  
  this.httpService.get(url, { headers }).toPromise().then((value: any) => {  
    this.usuarios = value as UserDto[];  
  });  
}
```

Notificaciones

Este servicio proporciona métodos simples para mostrar notificaciones en la aplicación utilizando la biblioteca ngx-toastr. Las notificaciones se pueden clasificar en cuatro tipos: éxito, error, advertencia e información.

Este método se encarga de mostrar notificaciones según el estado proporcionado. Puedes utilizar este método para mostrar notificaciones de éxito, error, advertencia o información.

```
@Injectable({  
  providedIn: 'root'  
)  
export class UtilsService {  
  
  constructor(private toastr:ToastrService) { }  
  
  public alert(state:string,msg:string,title:string="",timeout=4000):void{  
    state = state.toLowerCase()  
    if(state == "success"){  
      this.toastr.success(msg,title,{timeOut:timeout});  
    }  
    else if(state == "error"){  
      this.toastr.error(msg,title,{timeOut:timeout});  
    }  
    else if(state == "warning"){  
      this.toastr.warning(msg,title,{timeOut:timeout});  
    }  
    else if(state == "info"){  
      this.toastr.info(msg,title,{timeOut:timeout});  
    }  
    else{  
      this.toastr.info("Tipo de alerta invalida msg: " + msg, title,{timeOut:timeout});  
    }  
  }  
}
```

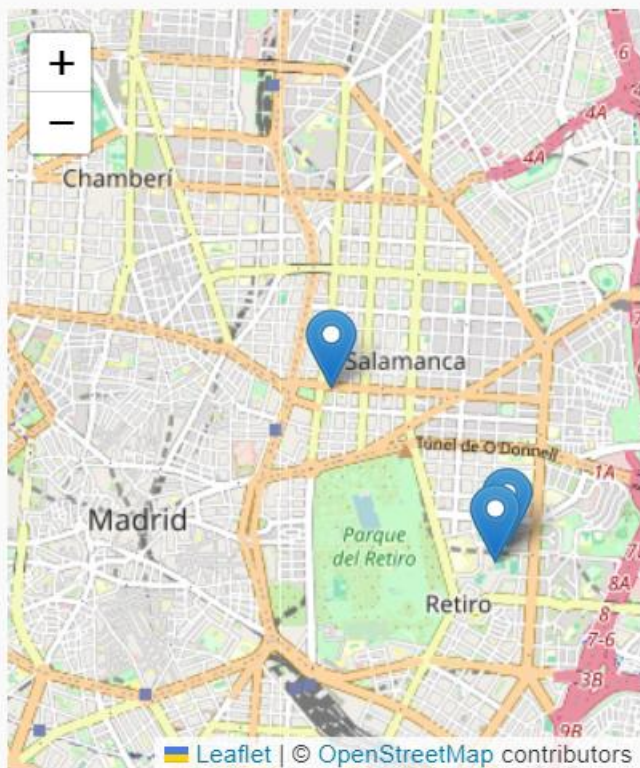

Vistas

Mapa

Para la creación del mapa he elegido la biblioteca Leaflet debido su eficiencia a la hora de integrar mapas interactivos.

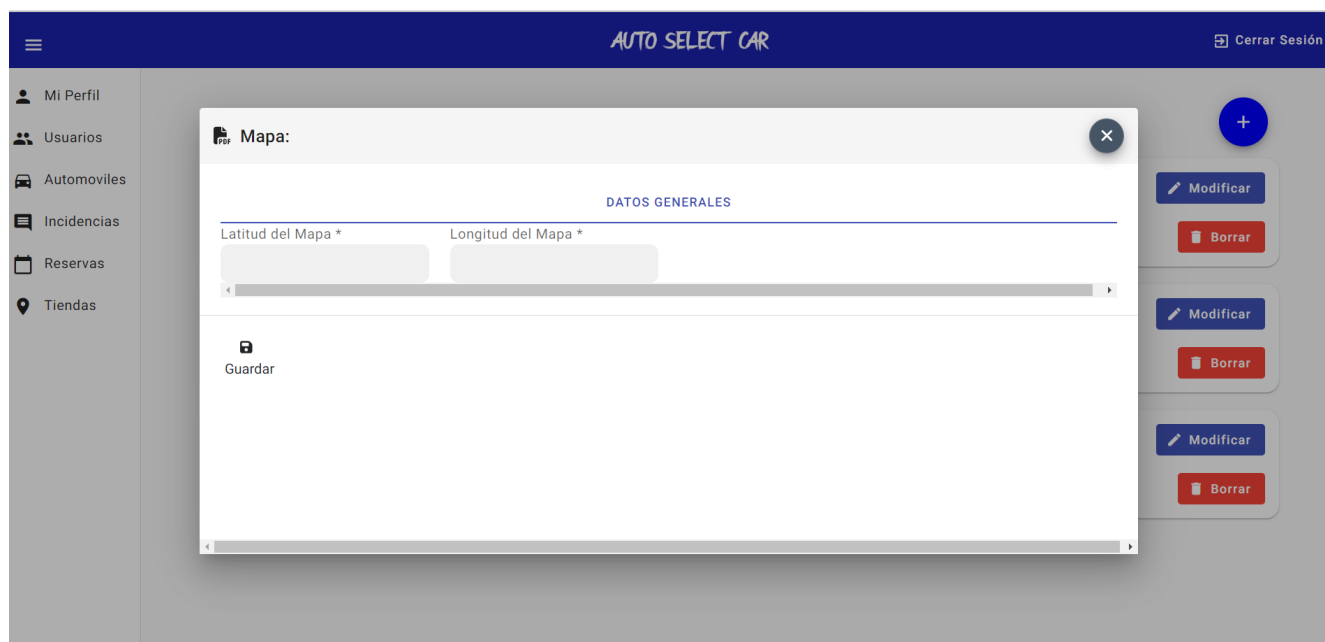
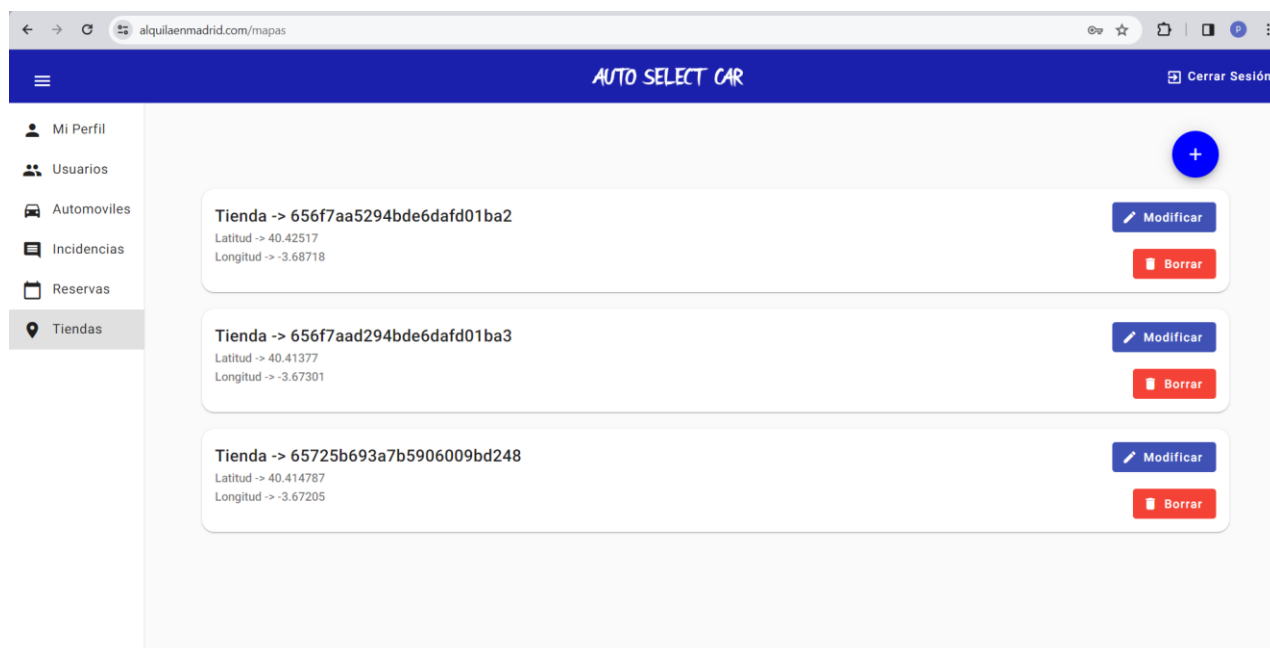
En Leaflet se puede personalizar, añadir acciones, marcadores.

```
const map = new Map('map').setView([40.42517,-3.68718], 13);  
tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {  
  maxZoom: 19,  
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'  
}).addTo(map);  
  
this.ubicaciones.forEach(location =>{  
  marker([Number(location.latitud), Number(location.longitud)]).addTo(map)  
});
```





El mapa se muestra en la zona del catálogo pero la acción de añadir una tienda al mapa la tiene el administrador.



Como podemos ver el administrador puede añadir, borrar y modificar la localización de una tienda.



Calendario

Dentro de la interfaz del catálogo de la plataforma de alquiler de vehículos, he integrado una herramienta esencial para la gestión de reservas: el calendario. Para proporcionar una experiencia de usuario óptima y evitar la selección de fechas no válidas, se implementó un sistema de selección de fecha utilizando Angular Material Datpicker.

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

```
<mat-form-field class="example-full-width">
  <mat-label>Elige una fecha</mat-label>
  <input matInput [matDatepickerFilter]="myFilter" [matDatepicker]="pickerInicio" [(ngModel)]="reservaFilterInicio" >
  <mat-datepicker-toggle matIconSuffix [for]="pickerInicio"></mat-datepicker-toggle>
  <mat-datepicker #pickerInicio></mat-datepicker>
</mat-form-field>

<mat-form-field class="example-full-width">
  <mat-label>Elige una fecha</mat-label>
  <input matInput [matDatepickerFilter]="myFilter" [matDatepicker]="pickerFin" [(ngModel)]="reservaFilterFin">
  <mat-datepicker-toggle matIconSuffix [for]="pickerFin"></mat-datepicker-toggle>
  <mat-datepicker #pickerFin></mat-datepicker>
</mat-form-field>
```



El componente DatePicker se implementó con un filtro personalizado, denominado myFilter, que garantiza que los usuarios no puedan seleccionar fechas anteriores al día actual. Esta restricción es crucial para el proceso de reserva, asegurando que todas las transacciones se realicen con fechas presentes o futuras.

```
myFilter = (d: Date | null): boolean => {  
  const today = new Date();  
  today.setHours(0, 0, 0, 0);  
  
  const selectedDate = (d || new Date());  
  selectedDate.setHours(0, 0, 0, 0);  
  
  return selectedDate >= today;  
};
```



8 Backend

Para el desarrollo de mi proyecto en el backend, he optado por utilizar Spring Boot con el lenguaje Kotlin. Esta decisión se basó en varias características clave y ventajas que Spring con Kotlin me ofrece, adecuándose perfectamente a las necesidades y objetivos de mi aplicación. Las características que ofrece son:

Kotlin

- **Sintaxis Concisa y Clara:** Kotlin se caracteriza por su sintaxis limpia y directa, lo que hace que el código sea más legible y mantenible.
- **Seguridad en el Manejo de Nulos y Funciones de Extensión:** Su tratamiento de los nulos y las funciones de extensión aportan seguridad y flexibilidad al desarrollo.
- **Interoperabilidad con Java:** La capacidad de interoperar con Java permite aprovechar el rico ecosistema de Java.
- **Corrutinas para Programación Asíncrona:** Además de ser más ligeras que los hilos, las corrutinas en Kotlin permiten escribir código asíncrono de manera secuencial, mejorando la legibilidad y reduciendo la complejidad del manejo de operaciones asíncronas.
- **Funciones de Orden Superior y Lambdas:** Kotlin soporta funciones de orden superior y lambdas, lo que facilita la programación funcional y permite escribir código más expresivo y conciso.
- **Extensiones de Tipo Nulo:** Kotlin permite extender clases con funciones adicionales sin tener que heredar de la clase, lo que facilita añadir funcionalidades específicas a clases existentes de una forma limpia y mantenible.
- **Null Safety:** El sistema de tipos de Kotlin está diseñado para eliminar el peligro de referencias nulas, uno de los errores más comunes en Java.





Spring Boot

Spring Boot es un proyecto dentro del ecosistema de Spring que simplifica el proceso de configuración y desarrollo de aplicaciones basadas en Spring. Está diseñado para minimizar la cantidad de configuración necesaria para arrancar una aplicación Spring, facilitando el desarrollo, las pruebas y el despliegue de aplicaciones robustas y eficientes.

- **Configuración y Arranque Simplificados:** Spring Boot se caracteriza por su capacidad para simplificar la configuración inicial y el proceso de arranque de aplicaciones Spring. La anotación `@SpringBootApplication` y el método `SpringApplication.run` en la clase principal son indicativos de un proyecto basado en Spring Boot.
- **Autoconfiguración:** Spring Boot proporciona autoconfiguración para la mayoría de los proyectos Spring, reduciendo la necesidad de configuración manual y permitiendo un enfoque más rápido y eficiente en el desarrollo de características específicas de la aplicación.
- **Independencia del Contenedor de Servlets:** Las aplicaciones Spring Boot son autocontenidas y no requieren un servidor de aplicaciones externo, ya que pueden incorporar un contenedor de servlets como Tomcat o Jetty.
- **Spring Boot Starters:** Los starters de Spring Boot son una serie de dependencias pre-configuradas que simplifican la adición de componentes a la aplicación, como Spring Security, Spring Data JPA, entre otros.
- **Gestión Eficiente de Dependencias:** Spring Boot facilita la gestión de dependencias y garantiza la compatibilidad de versiones entre diferentes módulos de Spring y bibliotecas de terceros.



Spring Boot

REST API



8.1 Herramientas Utilizadas

Integración de Corrutinas de Kotlin en Spring: Optimizando la Concurrencia y Asincronía

En el desarrollo de aplicaciones modernas, especialmente aquellas que requieren manejo eficiente de operaciones asíncronas y concurrentes, la elección de herramientas y frameworks adecuados es crucial. En este proyecto, he optado por integrar las corrutinas de Kotlin con el framework Spring.

¿Qué son las Corrutinas de Kotlin?

Las corrutinas son una de las características más poderosas y distintivas de Kotlin, proporcionando una forma más ligera y menos costosa que los hilos tradicionales para manejar la concurrencia y la asincronía. Son especialmente útiles para tareas que involucran operaciones de entrada/salida, llamadas a API, operaciones de base de datos y cualquier procesamiento que requiera esperar a que algo más termine.

Ventajas de Usar Corrutinas en el Backend

- **Manejo Eficiente de Operaciones Asíncronas:** Las corrutinas permiten escribir código asíncrono de una manera más directa y legible, similar a las operaciones síncronas normales. Esto mejora la legibilidad y mantenibilidad del código.
- **Menor Uso de Recursos:** A diferencia de los hilos, las corrutinas son más ligeras en términos de memoria y CPU. Esto significa que puedes lanzar muchas corrutinas en una sola aplicación sin preocuparte por el consumo excesivo de recursos.
- **Simplificación de Código Complejo:** Las corrutinas facilitan el manejo de operaciones complicadas y dependencias de datos, lo que es especialmente valioso en flujos de trabajo complejos y operaciones de E/S.



Conclusión:

La decisión de integrar las corrutinas de Kotlin con Spring en este proyecto no ha sido aleatoria, sino una elección bien pensada. He buscado un equilibrio entre eficiencia y calidad en el desarrollo del software. Trabajar con corrutinas ha significado un cambio sustancial en la manera de manejar procesos asíncronos y concurrentes, simplificando tareas complejas y mejorando significativamente la experiencia de desarrollo.

Ejemplo Repositorio:

```
interface ReservasRepository : CoroutineCrudRepository<Reserva, String> {  
    fun findByUuid(uuid: String): Flow<Reserva>  
    fun findAllByFechaInicioAfter(fechaInicio: LocalDate) : Flow<Reserva>  
    fun findAllByFechaFinalBefore(fechaFin: LocalDate) : Flow<Reserva>  
    fun findAllByClienteId(clienteId : String): Flow<Reserva>  
  
    fun findAllByFechaInicioBetween(fechaInicio : LocalDate, fechaFin : LocalDate): Flow<Reserva>  
  
    @Query("{ 'fechaInicio' : { \\\$gte : ?0 }, 'fechaFinal' : { \\\$lte : ?1 } }")  
    fun findAllByFechaInicioBetweenAndFechaFinalAfter(  
        @Param("fechaInicio") fechaInicio: LocalDate,  
        @Param("fechaFin") fechaFin: LocalDate  
    ): Flow<Reserva>  
}
```

Ejemplo Método Servicio:

```
suspend fun findAllByCliente(id: String) = withContext(Dispatchers.IO) { this: CoroutineScope  
    return@withContext repository.findAllByClienteId(id)  
}
```

SPRING
KOTLIN
COROUTINES





Implementación de Seguridad con Tokens y Bcrypt en Spring

En el desarrollo de aplicaciones web modernas, la seguridad es una preocupación primordial. Para abordar eficazmente este aspecto crucial, he utilizado Spring Security, tokens y Bcrypt.

Autenticación y Autorización con Tokens en SpringBoot

- **Uso de Tokens para la Seguridad:** He implementado un sistema de autenticación y autorización basado en tokens, un método probado y seguro para manejar la identidad y los permisos de los usuarios. Los tokens ofrecen una manera flexible y eficiente de controlar el acceso a los recursos de la aplicación sin comprometer la seguridad.
- **Spring Security para la Gestión de Tokens:** Spring Security ha sido fundamental en la implementación de este sistema. Proporciona un marco robusto para la gestión de la autenticación y la autorización, permitiéndonos definir políticas de seguridad detalladas y adaptadas a nuestras necesidades.
- **@PreAuthorize para Control de Acceso:** Utilizamos la anotación **@PreAuthorize** de Spring para controlar el acceso a diferentes partes de la aplicación. Esto nos permite especificar con gran detalle quién puede hacer qué en la aplicación, asegurando que solo los usuarios autorizados puedan acceder a recursos sensibles.





```
@PreAuthorize("hasRole('ADMINISTRADOR')")
@PostMapping("/delete/{numeroChasis}")
suspend fun delete(@PathVariable numeroChasis : String): ResponseEntity<String> {
    logger.info { "Borrar automovil con numeroChasis: $numeroChasis" }
    service.delete(numeroChasis)
    return ResponseEntity.ok( body: "Borrado")
}
```

JwtAuthenticationFilter

```
class JwtAuthenticationFilter(
    private val jwtTokenUtil: JwtTokenUtils,
    private val authenticationManager: AuthenticationManager
) : UsernamePasswordAuthenticationFilter() {

    override fun attemptAuthentication(req: HttpServletRequest, response: HttpServletResponse): Authentication {
        logger.info { "Intentando autenticar" }

        val credentials = ObjectMapper().readValue(req.inputStream, UsuarioLoginDto::class.java)
        val auth = UsernamePasswordAuthenticationToken(
            credentials.username,
            credentials.password,
        )
        return authenticationManager.authenticate(auth)
    }

    override fun successfulAuthentication(
        req: HttpServletRequest?, res: HttpServletResponse, chain: FilterChain?,
        auth: Authentication
    ) {...}

    override fun unsuccessfulAuthentication(
        request: HttpServletRequest,
        response: HttpServletResponse,
        failed: AuthenticationException
    ) {...}

    private data class BadCredentialsError(
        val timestamp: Long = Date().time,
        val status: Int = 401,
        val message: String = "Usuario o password incorrectos",
    ) {
        override fun toString(): String {
            return ObjectMapper().writeValueAsString( value: this)
        }
    }
}
```



JwtAuthorizationFilter

```
class JwtAuthorizationFilter(  
    private val jwtTokenUtil: JwtTokenUtils,  
    private val service: UsuarioService,  
    authManager: AuthenticationManager,  
    ) : BasicAuthenticationFilter(authManager) {  
  
    @Throws(IOException::class, ServletException::class)  
    override fun doFilterInternal(  
        req: HttpServletRequest,  
        res: HttpServletResponse,  
        chain: FilterChain  
    ) {  
        logger.info { "Filtrando" }  
        val header = req.getHeader(HttpHeaderNames.AUTHORIZATION.toString())  
        if (header == null || !header.startsWith(JwtTokenUtils.TOKEN_PREFIX)) {  
            chain.doFilter(req, res)  
            return  
        }  
        getAuthentication(header.substring(7)).also { it: UsernamePasswordAuthenticationToken }  
        SecurityContextHolder.getContext().authentication = it  
    }  
    chain.doFilter(req, res)  
}  
  
private fun getAuthentication(token: String): UsernamePasswordAuthenticationToken? = runBlocking { this: CoroutineScope }  
    logger.info { "Obteniendo autenticación" }  
  
    if (!jwtTokenUtil.isTokenValid(token)) return@runBlocking null  
    val userId = jwtTokenUtil.getUserIdFromJwt(token)  
    val user = service.loadUserByUuid(userId.toUUID().toString())  
    return@runBlocking UsernamePasswordAuthenticationToken(  
        user,  
        credentials: null,  
        user?.authorities  
    )  
}
```



JwtTokenUtils

```
@Component
class JwtTokenUtils {

    @Value("${TrabajoFinDeGrado2023}")
    private val jwtSecreto: String? = null

    @Value("${3600}")
    private val jwtDuracionTokenEnSegundos = 0

    fun generateToken(user: Usuario): String {
        logger.info { "Generando token para el usuario: ${user.username}" }
        val tokenExpirationDate = Date( date: System.currentTimeMillis() + jwtDuracionTokenEnSegundos * 1000)

        return JWT.create()
            .withSubject(user.uid.toString())
            .withHeader(mapOf("typ" to TOKEN_TYPE))
            .withIssuedAt(Date())
            .withExpiresAt(tokenExpirationDate)
            .withClaim( name: "username", user.username)
            .withClaim( name: "nombre", user.nombre)
            .withClaim( name: "roles", user.rol.split( ...delimiters: ",").toSet().toString())
            .sign(Algorithm.HMAC512(jwtSecreto))
    }

    fun getUserIdFromJwt(token: String?): String {
        logger.info { "Obteniendo el ID del usuario: $token" }
        return validateToken(token!!)?.subject
    }

    fun validateToken(authToken: String): DecodedJWT? {
        logger.info { "Validando el token: ${authToken}" }

        try {
            return JWT.require(Algorithm.HMAC512(jwtSecreto)).build().verify(authToken)
        } catch (e: Exception) {
            throw TokenInvalidException("Token no válido o expirado")
        }
    }
}
```

```
private fun getClaimsFromJwt(token: String) =
    validateToken(token)?.claims

fun getUsernameFromJwt(token: String): String {
    logger.info { "Obteniendo el nombre de usuario del token: ${token}" }

    val claims = getClaimsFromJwt(token)
    return claims!!["username"]!!.asString()
}

fun getRolesFromJwt(token: String): String {
    logger.info { "Obteniendo los roles del token: ${token}" }

    val claims = getClaimsFromJwt(token)
    return claims!!["roles"]!!.asString()
}

fun isTokenValid(token: String): Boolean {
    logger.info { "Comprobando si el token es válido: ${token}" }

    val claims = getClaimsFromJwt(token)!!
    val expirationDate = claims["exp"]!!.asDate()
    val now = Date(System.currentTimeMillis())
    return now.before(expirationDate)
}

companion object {
    const val TOKEN_HEADER = "Authorization"
    const val TOKEN_PREFIX = "Bearer "
    const val TOKEN_TYPE = "JWT"
}
```



Seguridad de Contraseñas con Bcrypt

- **Bcrypt para el Hashing de Contraseñas:** La seguridad de las contraseñas es otro componente esencial de nuestro enfoque de seguridad. He elegido Bcrypt para el hashing de contraseñas debido a su robustez y resistencia a los ataques de fuerza bruta. Bcrypt nos permite almacenar contraseñas de manera segura en nuestra base de datos, protegiéndolas incluso en el caso de una violación de datos.
- **Ventajas de Bcrypt sobre Otros Algoritmos de Hashing:** Bcrypt es preferible a algoritmos de hashing más simples debido a su capacidad para incorporar un 'salt' y su naturaleza adaptativa, lo que significa que podemos ajustar la dificultad del hashing para mantenernos al día con el aumento de la capacidad de procesamiento de los atacantes.

```
@Configuration
class EncoderConfig {
    @Bean
    fun passwordEncoder(): PasswordEncoder {
        return BCryptPasswordEncoder()
    }
}
```



Validadores para Garantizar la Integridad de los Datos

- **Implementación de Validadores Personalizados:** Además de la seguridad en la autenticación y el almacenamiento de contraseñas, también he puesto un énfasis considerable en la validación de la entrada de datos. He desarrollado validadores personalizados para asegurarnos de que solo los datos correctos y seguros sean procesados y almacenados.
- **Prevención de Entradas Maliciosas o Incorrectas:** Esta práctica es esencial para prevenir problemas comunes como inyecciones SQL, XSS y otros vectores de ataque que podrían explotar datos mal formados o maliciosos.

```
fun UsuarioCreateDto.validate(): UsuarioCreateDto {  
    if (this.nombre.isBlank()) {  
        throw UsuariosBadRequestException("El nombre no puede estar vacío")  
    } else if (this.email.isBlank() || !this.email.matches(Regex( pattern: "[A-Za-z0-9+_.-]+@(.+)\$")))  
        throw UsuariosBadRequestException("El email no puede estar vacío o no tiene el formato correcto")  
    else if (this.username.isBlank())  
        throw UsuariosBadRequestException("El username no puede estar vacío")  
    else if (this.password.isBlank() || this.password.length < 4)  
        throw UsuariosBadRequestException("El password no puede estar vacío o ser menor de 4 caracteres")  
  
    return this  
}  
  
fun UsuarioUpdateDto.validate(): UsuarioUpdateDto {  
    if (this.nombre.isBlank()) {  
        throw UsuariosBadRequestException("El nombre no puede estar vacío")  
    } else if (this.email.isBlank() || !this.email.matches(Regex( pattern: "[A-Za-z0-9+_.-]+@(.+)\$")))  
        throw UsuariosBadRequestException("El email no puede estar vacío o no tiene el formato correcto")  
    else if (this.username.isBlank())  
        throw UsuariosBadRequestException("El username no puede estar vacío")  
  
    return this  
}
```



Cache

Una de las estrategias clave para mejorar la eficiencia y el rendimiento en mi proyecto ha sido la implementación de la caché, utilizando las capacidades que ofrece Spring a través de la anotación **@Cacheable**. La gestión efectiva de la caché es crucial para optimizar la carga de trabajo en el servidor y acelerar la respuesta a los usuarios.

- **Mejora del Rendimiento y Reducción de la Carga en el Servidor:** Al almacenar en caché respuestas a consultas comunes o cálculos intensivos, reducimos significativamente la carga en nuestros servidores y bases de datos. Esto no solo mejora la velocidad de respuesta de nuestra aplicación, sino que también aumenta la escalabilidad al reducir la cantidad de recursos necesarios para atender a un gran número de usuarios simultáneamente.

Beneficios de la Caché en el Proyecto

- **Respuestas Más Rápidas para los Usuarios:** Una de las ventajas más notables del uso de **@Cacheable** es la mejora en el tiempo de respuesta de la aplicación. Los usuarios experimentan tiempos de carga más rápidos, lo que mejora significativamente la experiencia del usuario.
- **Optimización de Recursos:** Al disminuir la frecuencia de operaciones costosas, como las consultas a la base de datos, optimizamos el uso de los recursos del sistema, lo que es especialmente importante en momentos de alta demanda.
- **Consistencia y Control de Datos:** A través de una gestión cuidadosa de la caché, nos aseguramos de que los datos críticos se actualicen de manera oportuna, manteniendo un equilibrio entre rendimiento y consistencia de datos.

```
@Cacheable("reservas")
suspend fun findAll() = withContext(Dispatchers.IO) { this: CoroutineScope
    return@withContext repository.findAll()
}
```




9 Base de Datos

En el desarrollo de la aplicación web de alquiler de automóviles en línea, he optado por utilizar MongoDB como mi sistema de gestión de base de datos. Esta decisión se fundamenta en varias características y ventajas que MongoDB ofrece, las cuales se alinean perfectamente con las necesidades y exigencias de mi aplicación.

Características de MongoDB

- **Base de Datos NoSQL Orientada a Documentos:** A diferencia de las bases de datos relacionales tradicionales, MongoDB es una base de datos NoSQL orientada a documentos. Esto significa que es altamente flexible en términos de estructura de datos, lo cual es ideal para manejar la variedad y la complejidad de los datos asociados con el alquiler de automóviles, como información de vehículos, datos de clientes, y reservas.
- **Escalabilidad y Rendimiento:** MongoDB está diseñado para ofrecer una alta escalabilidad y rendimiento. Su capacidad para manejar grandes volúmenes de lecturas y escrituras lo hace adecuado para nuestra aplicación, que espera gestionar un número significativo de consultas de usuarios.
- **Esquemas Flexibles:** La flexibilidad en los esquemas de MongoDB me permite evolucionar fácilmente la estructura de la base de datos a medida que se añaden nuevas características a la aplicación o cambian los requisitos del negocio. Esto es especialmente valioso en el contexto de una plataforma de alquiler de automóviles en línea, donde las necesidades y expectativas de los usuarios pueden cambiar rápidamente.



Beneficios de MongoDB

- **Manejo Eficiente de Datos Diversos:** En una plataforma de alquiler de automóviles, los datos pueden variar rápidamente, desde especificaciones detalladas de vehículos hasta historiales de alquiler de los usuarios. MongoDB maneja esta diversidad de manera eficiente
- **Adaptabilidad a Cambios:** El mercado de alquiler de vehículos en línea es dinámico y requiere que nuestra aplicación se adapte rápidamente a las nuevas tendencias y demandas. La capacidad de MongoDB para adaptarse a cambios en la estructura de datos sin requerir una reestructuración completa es una gran ventaja.
- **Soporte para Consultas Complejas:** MongoDB ofrece un potente motor de consultas que puede manejar búsquedas complejas, lo cual es esencial para proporcionar a los usuarios una experiencia de búsqueda y filtrado eficiente en la plataforma.





10 Despliegue

10.1 Contratación de Servicios de Hosting y Dominio:

Hosting Elegido:

En el desarrollo de nuestra plataforma de alquiler de vehículos en línea, la elección de un proveedor de hosting confiable y eficiente era una prioridad. Después de considerar varias opciones, seleccioné Hetzner para el hosting de nuestro sitio web por varias razones clave:

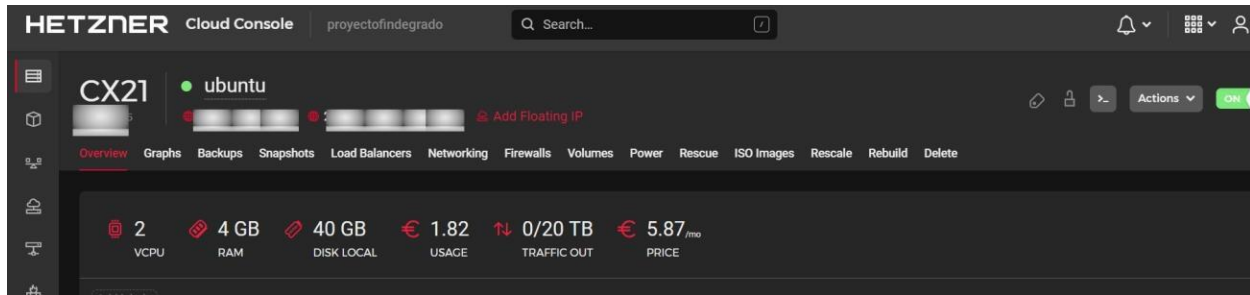
- **Rendimiento y Fiabilidad:** Hetzner ha demostrado un alto nivel de rendimiento y estabilidad, características cruciales para asegurar una experiencia de usuario continua y confiable.
- **Costo-Efectividad:** Hetzner ofrece un equilibrio ideal entre costo y calidad, proporcionando un servicio de hosting robusto sin un gasto excesivo, lo cual es fundamental para optimizar nuestros recursos financieros.
- **Escalabilidad:** La flexibilidad de Hetzner para escalar los recursos en respuesta a nuestras necesidades nos asegura que podemos ajustar nuestra infraestructura de hosting conforme crece la demanda en nuestra plataforma.

Características del Servidor Elegido

- **Sistema Operativo Ubuntu:** Opté por Ubuntu debido a su estabilidad, seguridad y facilidad de uso. Ubuntu es conocido por su excelente soporte y amplia comunidad, lo que facilita la gestión y resolución de problemas. Además, su compatibilidad con una amplia gama de software de servidor lo hace ideal para un entorno de hosting web.
- **Capacidad de Procesamiento y Memoria:** Un servidor con 2 núcleos y 4 GB de RAM ofrece un equilibrio adecuado entre rendimiento y costo. Esta configuración es suficiente para manejar el tráfico y las cargas de trabajo iniciales esperadas en nuestra plataforma, proporcionando una respuesta rápida a las solicitudes de los usuarios sin incurrir en recursos innecesarios.



- **Ubicación en Nuremberg:** La elección de un centro de datos en Nuremberg se debe a su ubicación estratégica en Alemania. Esto asegura una conectividad rápida y fiable, no solo a nivel local sino también para los usuarios en Europa. La ubicación geográfica es un factor clave para garantizar tiempos de respuesta rápidos y una experiencia de usuario óptima.



HETZNER



Dominio en Namecheap:


Una parte esencial en el lanzamiento de la plataforma de alquiler de vehículos en línea es la selección y el registro de un dominio web que reflejara claramente nuestra marca y propósito. Tras una cuidadosa consideración, he elegido el dominio **alquilaenmadrid.com**, registrado a través de **Namecheap**.

Elección del Dominio **alquilaenmadrid.com**

- **Relevancia y Facilidad de Recordación:** El nombre **alquilaenmadrid.com** fue seleccionado por su relevancia directa y su facilidad de recordación para mi público objetivo. Este dominio intuitivo es fácilmente asociable con servicios de alquiler de vehículos en la ciudad de Madrid, facilitando así a los usuarios locales y turistas encontrar nuestra plataforma.
- **Beneficios en Marketing y SEO:** Un dominio descriptivo y centrado en la ubicación como **alquilaenmadrid.com** es también una herramienta poderosa para nuestras estrategias de marketing digital y optimización para motores de búsqueda (SEO). Ayuda a mejorar la visibilidad en línea para búsquedas relacionadas con el alquiler de automóviles en Madrid, atrayendo tráfico relevante a nuestro sitio.

Elección de Namecheap para el Registro

- **Confiabilidad y Costo-Efectividad:** Decidimos registrar nuestro dominio a través de Namecheap debido a su reputación como proveedor confiable y económico. Namecheap ofrece una plataforma de registro sencilla y segura, con precios competitivos, lo cual es importante para mantener los costos operativos bajo control.

<input type="checkbox"/> Domains	Status	Auto-Renew	Expiration	
<input type="checkbox"/>  alquilaenmadrid.com <small>Domain Privacy protection is ON</small>	✓ ACTIVE	<input type="checkbox"/>	Nov 18, 2024	<button>MANAGE</button>





10.2 Preparación del Servidor y Despliegue de la Aplicación:

La fase de preparación del servidor y el despliegue de la aplicación de alquiler de vehículos en línea fue un proceso meticuloso y esencial para asegurar un lanzamiento exitoso y una operación sin interrupciones. La estrategia de despliegue se centró en la eficiencia y la fiabilidad, utilizando un artefacto de aplicación autocontenida.

Creación del Artefacto de Aplicación .jar

- **Artefacto Autocontenido:** El archivo ejecutable .jar, creado como un artefacto de nuestra aplicación, encapsula tanto el front-end como el back-end. Esta estructura autocontenida asegura que todas las dependencias y recursos necesarios estén incluidos, simplificando el proceso de despliegue y ejecución.
- **Despliegue de Recursos del Front-end:** Al ejecutar el archivo .jar, los recursos del front-end se despliegan automáticamente en la carpeta **resources/static**. Esta configuración sigue la estructura de proyecto estándar de Spring Boot, permitiendo que la aplicación se ejecute de manera autónoma y eficiente.
- **Facilidad de Mantenimiento y Actualización:** Esta metodología de empaquetado y despliegue facilita la gestión y actualización de la aplicación. Los cambios, tanto en el front-end como en el back-end, pueden implementarse y desplegarse rápidamente, asegurando que nuestra plataforma permanezca actualizada con las últimas mejoras y correcciones.



10.3 Configuración de Nginx para Redirección de Tráfico:

Para la aplicación de alquiler de vehículos, la infraestructura web debe ser tan ágil y segura. Con este objetivo, he configurado Nginx, un servidor web de alto rendimiento, para dirigir y optimizar el flujo de tráfico a nuestra plataforma.

Configuración de Nginx para Redirección HTTPS

- **Redirección de Puerto Estándar:** Se ha establecido una redirección en Nginx para que todo el tráfico que llega al puerto HTTP estándar (80) sea redirigido al puerto HTTPS (443), garantizando que todas las conexiones se realicen de forma segura a través del protocolo TLS/SSL.
- **Configuración de Certificados SSL:** Nginx ha sido configurado con la ubicación precisa de los certificados SSL, asegurando una encriptación confiable para la comunicación entre el cliente y el servidor. Esto no solo protege la privacidad de nuestros usuarios, sino que también mejora nuestra clasificación en motores de búsqueda, un factor crítico para el marketing online.
- **Directivas de Seguridad Mejoradas:** Se han implementado cabeceras HTTP estrictos y políticas de seguridad mejoradas para mitigar vulnerabilidades y ataques, manteniendo la integridad de la comunicación de usuario.

Configuración del Proxy Inverso

- **Conexión con la Aplicación:** Utilizando Nginx como proxy inverso, hemos configurado una pasarela para las solicitudes al back-end de nuestra aplicación, que corre en un servidor interno en el puerto 8080. Esta capa intermedia refuerza la seguridad y permite un manejo más eficiente del tráfico.
- **Manejo de Caché:** Aprovechamos las capacidades de Nginx para la caché de contenido estático, lo que disminuye la carga en nuestros servidores de aplicaciones y acelera la entrega de contenido a los usuarios.



Configuración de Nginx

```
GNU nano 6.2
events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    upstream app {
        server automoviles:8080;
    }

    server {
        listen 80;
        server_name aquilaenmadrid.com;

        # Redirigir todo el tráfico HTTP a HTTPS
        return 301 https://$server_name$request_uri;
    }

    server {
        listen 443 ssl;
        server_name aquilaenmadrid.com;

        ssl_certificate /etc/ssl/certs/alquilaenmadrid.com.pem;
        ssl_certificate_key /etc/ssl/private/alquilaenmadrid.com.key;

        location / {
            proxy_pass http://app;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }
}
```





10.4 Implementación de Certificados SSL con Let's Encrypt:

Para fortalecer la seguridad, se obtuvieron certificados SSL a través de Let's Encrypt, una autoridad de certificación que proporciona certificados gratuitos para la habilitación de HTTPS. El proceso de obtención y renovación de certificados se automatizó utilizando Certbot, una herramienta que simplifica este proceso y garantiza que la comunicación entre el usuario y el servidor sea cifrada.

Implementación de Certificados SSL Gratuitos

- **Let's Encrypt como Autoridad de Certificación:** Hemos elegido Let's Encrypt por su reputación como una autoridad de certificación confiable y respetada que proporciona certificados SSL de forma gratuita. Su misión de promover conexiones web cifradas en todo el mundo alinea con nuestra visión de una experiencia de usuario segura y protegida.
- **Automatización con Certbot:** Para manejar la obtención y renovación de los certificados, implementamos Certbot, una herramienta eficiente que automatiza estos procesos. Certbot no solo nos ayuda a configurar inicialmente SSL/TLS, sino que también monitorea la validez de los certificados y gestiona la renovación antes de que expiren, manteniendo así la seguridad sin interrupción.

Beneficios de la Encriptación SSL/TLS

- **Encriptación de Datos:** Los certificados SSL/TLS aseguran que todos los datos transmitidos entre el navegador del usuario y nuestro servidor estén completamente cifrados. Esto es crítico al manejar información sensible como datos personales y detalles de transacciones.
- **Confianza del Usuario y Cumplimiento de Normativas:** La utilización de SSL/TLS no solo incrementa la confianza del usuario en nuestra plataforma, sino que también cumple con las normativas de seguridad de datos, como el GDPR. Los usuarios pueden estar tranquilos sabiendo que sus interacciones con nuestra plataforma son privadas y protegidas.





10.5 Dockerización y Orquestación con Docker

Compose:

Para facilitar el despliegue y la escalabilidad, la aplicación y el servidor web Nginx se contenerizaron utilizando Docker. Se creó un archivo docker-compose.yml que define los servicios necesarios para ejecutar la aplicación, incluyendo la imagen de Nginx y el contenedor que ejecuta el archivo .jar de la aplicación. Docker Compose gestiona la creación, configuración y ejecución de estos contenedores, proporcionando un entorno de despliegue coherente y replicable.

Configuración:

- **Imagen de Nginx Actualizada:** Utilizamos la última imagen oficial de Nginx para asegurarnos de tener las características y parches de seguridad más recientes.
- **Mapeo de Puertos:** Los puertos 80 y 443 del host se mapean a los puertos correspondientes en el contenedor Nginx, permitiendo manejar el tráfico HTTP y HTTPS.
- **Volúmenes:** Montamos el archivo de configuración de Nginx y los certificados SSL directamente en el contenedor. Esto asegura que nuestro servidor Nginx esté configurado para dirigir el tráfico correctamente y asegurar las comunicaciones con SSL.

```
GNU nano 6.2 docker-compose.yml
version: '3.9'

services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/live/alquilaenmadrid.com/fullchain.pem:/etc/ssl/certs/alquilaenmadrid.com.pem
      - /etc/letsencrypt/live/alquilaenmadrid.com/privkey.pem:/etc/ssl/private/alquilaenmadrid.com.key
    depends_on:
      - automoviles
    networks:
      - default

  automoviles:
    image: openjdk:17
    restart: unless-stopped
    volumes:
      - /docker/automoviles/data:/app
      - /docker/automoviles/data/logs:/opt/aplicaciones/automoviles/application/web/logs
    ports:
      - "8080:8080"
    command: java -jar /app/Backend-Automoviles-0.0.1-SNAPSHOT.jar
    networks:
      - default

networks:
  default:
    driver: bridge
```



10.6 Modelo de Servicio en la Nube

Para el desarrollo y despliegue de la plataforma de alquiler de vehículos en línea, he seleccionado un modelo de servicio en la nube que combina la **flexibilidad de la Infraestructura como Servicio (IaaS) con las capacidades de una Plataforma como Servicio (PaaS)**, proporcionando así una base sólida y flexible para la operación de la aplicación.

Infraestructura como Servicio (IaaS)

He optado por un servidor provisto por el proveedor de la nube, equipado con especificaciones determinadas (dos núcleos, 4 GB de RAM, y 40 GB de almacenamiento) y un sistema operativo ya instalado. Este entorno IaaS me ha permitido tener el control total sobre el sistema operativo y la libertad de instalar y configurar el software necesario para el proyecto.

Autogestión de la Plataforma

Con el objetivo de simplificar el proceso de despliegue y gestión de las aplicaciones, tomé la iniciativa de instalar Docker, un sistema de contenedorización que me permitió encapsular las aplicaciones y servicios, incluyendo la base de datos MongoDB, dentro de contenedores.

Esta estrategia me proporcionó una abstracción similar a la que ofrece un entorno PaaS, donde pude:

Desarrollar y desplegar aplicaciones de manera eficiente y aislada.

Gestionar la base de datos y otros servicios de manera independiente y escalable.

Aprovechar las ventajas de los contenedores para asegurar la consistencia del entorno a través de diferentes etapas de desarrollo y producción.





11 Coste del proyecto

En el proceso de llevar la plataforma de alquiler de vehículos en línea desde la concepción hasta su lanzamiento, se han considerado detenidamente los costes operativos y de desarrollo para asegurar una inversión eficiente y justificada.

- **Gastos de Infraestructura**

La infraestructura del servidor es un componente esencial para la operatividad de nuestra plataforma. Hemos seleccionado un servidor que equilibra adecuadamente el rendimiento y el precio, resultando en un coste mensual de 6,50 euros. Esta inversión ha sido crucial para garantizar una experiencia de usuario fluida y confiable, así como para proporcionar una base sólida para el funcionamiento de la plataforma.

- **Adquisición del Dominio**

El dominio alquilermadrid.com se ha registrado con un costo de 10 euros. Esta inversión inicial es fundamental para establecer la presencia en línea de la plataforma y asegurar una identificación directa y profesional para nuestros usuarios, contribuyendo significativamente a la facilidad de acceso y al reconocimiento de la marca.

- **Inversión de Tiempo**

La creación y el desarrollo de la plataforma han requerido un periodo significativo de trabajo dedicado.



12 Conclusiones y Trabajo Futuro

Conclusiones

El desarrollo de la plataforma de alquiler de vehículos en línea ha sido una experiencia de aprendizaje y crecimiento técnico. A lo largo de este proyecto, he adquirido conocimientos desde cero sobre tecnologías clave del front-end, tales como Angular, y he profundizado en el uso de herramientas de diseño de interfaz como Angular Material, Bootstrap y PrimeNG.

La curva de aprendizaje para estas tecnologías al principio fue difícil; sin embargo, la experiencia ha sido enormemente gratificante.

Este proyecto ha implicado una inmersión en documentaciones técnicas, tutoriales y foros, lo que me ha permitido no solo construir una plataforma funcional sino también establecer una base sólida para mi desarrollo profesional en tecnologías de software modernas.

Trabajo Futuro

Mirando hacia el futuro, planeo expandir la funcionalidad de la plataforma para incluir características como:

- Un sistema de recomendaciones personalizadas basado en IA.
- Una aplicación móvil para facilitar el acceso en dispositivos móviles.
- Integración con sistemas de gestión de flotas para optimizar la disponibilidad de vehículos.
- Una aplicación multimodular dividida en microservicios para disminuir los tiempos de espera y hacer más accesible la aplicación.



13 Desafíos y Resoluciones a lo Largo del Proyecto

Incidente de Seguridad (29 de Noviembre):

Durante el curso del proyecto, he enfrentado un desafío el 29 de noviembre, cuando la base de datos alojada en el puerto 27017 fue hackeada. Los atacantes eliminaron todos los datos y me dejaron un mensaje, exigiendo un pago de 0.01 BTC para evitar la eliminación de los datos.

Mensaje:

All your data is backed up. You must pay 0.01 BTC to 14PYVptPexgRpHRm7SSr....mkRA55 In 48 hours, your data will be publicly disclosed and deleted. (more information: go to "...")After paying send mail to us: rambler+...@onionmail.org and we will provide a link for you to download your data. Your DBCODE is: 13N5P

Problema de Acceso (Error 403):

Durante la implementación de la seguridad en Spring Boot, me enfrenté a un desafío significativo que resultó en la denegación de entrada con un código de error 403. La raíz de este problema estaba relacionada con la forma incorrecta en que se estaba enviando el token desde el frontend al backend.

Detalles del Problema:

Al analizar el problema, identifiqué que la transmisión incorrecta desde el frontend del token estaba generando el error 403, impidiendo el acceso a recursos protegidos.

Desafíos al Implementar SSL: Problemas de CORS entre Frontend y Backend

Implementación de SSL y Problemas Emergentes:

Durante la implementación de SSL en nuestro servidor, me enfrenté a desafíos inesperados que afectaron la comunicación entre el frontend y el backend. Estos desafíos estaban relacionados con los problemas de Cross-Origin Resource Sharing (CORS), lo que resultó en la incapacidad del frontend para mostrar correctamente los recursos seguros.

Detalles del Problema:

El problema principal residía en la configuración incorrecta de CORS en el backend después de la implementación de SSL. Las solicitudes del frontend al backend eran bloqueadas debido a las políticas de seguridad CORS, generando errores de acceso desde el navegador.



14 Enlaces

Enlace a Repositorio de Github(Código Proyecto)

<https://github.com/jorgesanchez3212/Trabajo-Fin-De-Grado-JorgeSanchez>

Enlace a Video del Proyecto

<https://youtu.be/ffyvIDlcTtl>



15 Bibliografía

- 15.1 Documentación Oficial de Spring Boot:**
Spring Boot. (Año). *Spring Boot Reference Documentation*.
<https://spring.io/projects/spring-boot>
- 15.2 Documentación Oficial de Angular:**
Angular. (Año). *Angular Documentation*.
<https://angular.io/docs>
- 15.3 Documentación Oficial de MongoDB:**
MongoDB, Inc. (Año). *MongoDB Documentation*.
<https://docs.mongodb.com>
- 15.4 Guía Oficial de Docker:**
Docker, Inc. (Año). *Docker Documentation*.
<https://docs.docker.com>
- 15.5 Let's Encrypt para Certificados SSL:**
Let's Encrypt. (Año). *Let's Encrypt Documentation*.
<https://letsencrypt.org/docs/>
- 15.6 Nginx como Servidor Web y Proxy Inverso:**
Nginx, Inc. (Año). *Nginx Documentation*. <https://nginx.org/en/docs/>
- 15.7 Tutoriales y Guías de Namecheap:**
<https://www.namecheap.com/>
- 15.8 Hetzner como Proveedor de Hosting:**
Hetzner Online GmbH. (Año). *Hetzner Documentation*.
<https://wiki.hetzner.de/>
- 15.9 Curso de Angular Udemy Fernando Herrera:**
Udemy Academy – Fernando Herrera
<https://www.udemy.com/course/angular-fernando-herrera/>

