

Proyecto Trabajo Fin de Ciclo: CarCare

Juan Manuel Gutiérrez Mellado
2º C.F.G.S. D.A.M.

Información previa.....	4
Aplicación Angular.....	5
Detalle de la aplicación.....	5
Login.....	5
Pantalla de Registro.....	6
Barra de Menú principal.....	6
Pantalla Inicio.....	7
Pantalla Vehículos.....	8
Pantalla Proveedores.....	9
Pantalla Sobre mí.....	10
Pantalla Administración.....	10
Aplicación Kotlin.....	12
Detalle de la aplicación.....	12
Pantallas OnBoarding.....	12
Pantallas Login y Registro.....	13
Login.....	13
Registro.....	13
Pantalla Vehículos.....	13
Pantalla listado de vehículos.....	14
Pantalla detalle de vehículo.....	14
Pantalla Proveedores.....	15
Pantalla listado de proveedores.....	15
Pantalla detalle de proveedor.....	15
Pantalla Gastos.....	15
Pantalla listado de gastos.....	16
Pantalla detalle de gasto.....	17
Menú de Opciones.....	17
Mensajes de aviso.....	19
Persistencia de datos.....	19
Room.....	19
SharedPreferences.....	20
Modelos y persistencia de datos.....	20
Modelo de datos en Firebase.....	21
Colección user.....	21
Colección vehicle.....	21
Colección provider.....	22
Colección log.....	22
Modelo de datos local en Room.....	23
Vehicles.....	23
Diagrama de flujo de la aplicación.....	23
Prototipo en Figma.....	23

API en Python.....	26
Documentación de la API.....	26
Ejemplos de respuestas por categoría:.....	26
Ejemplo de respuesta por marca.....	27
Respuesta errónea.....	28
Problemas encontrados:.....	28
Código completo de la aplicación Python.....	29
Gestión de datos del Log.....	33
Dashboard en Power BI.....	35
Transformaciones realizadas.....	35
Datos:.....	35
Dimensiones:.....	35
Órden:.....	36
Medidas:.....	36
Descripción del dashboard.....	36
Dashboard principal.....	36
Informe por tipo de operación.....	38
Informe por fecha.....	39
Relaciones entre las diferentes tablas.....	40
Exportación a PDF.....	40
Posibles mejoras y escalabilidad.....	41
Tecnologías utilizadas.....	42
Herramientas generales.....	42
Aplicación Angular.....	42
Aplicación Android.....	42
Aplicación Python.....	42
Videos del proyecto en funcionamiento.....	43
Webgrafía.....	44

Información previa

En este Trabajo Fin de Ciclo se presenta el desarrollo de una aplicación para la gestión de vehículos, gastos y proveedores correspondientes. Con el objetivo de ofrecer una solución integral y eficiente, se han desarrollado dos versiones de la aplicación utilizando dos tecnologías distintas: una versión en Angular y otra en Kotlin.

Angular es un framework desarrollado por Google, conocido por su robustez y capacidad para crear aplicaciones de una sola página (SPA) con una experiencia de usuario fluida y dinámica. Por otro lado, Kotlin es un lenguaje de programación moderno que permite desarrollar aplicaciones Android nativas.

Aplicación Angular

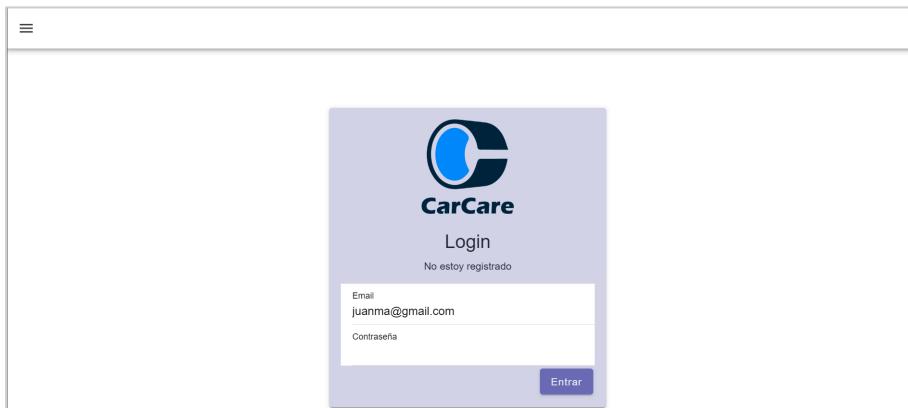
Disponible en este repositorio: <https://github.com/Juanma-Gutierrez/CarCare-Firebase>

La versión de la aplicación Angular es la presentada como proyecto de Acceso a datos en el segundo trimestre, con ciertas mejoras para cumplir los requisitos solicitados para el proyecto final TFC.

La aplicación está actualmente desplegada en Netlify: [Ionic App](#)

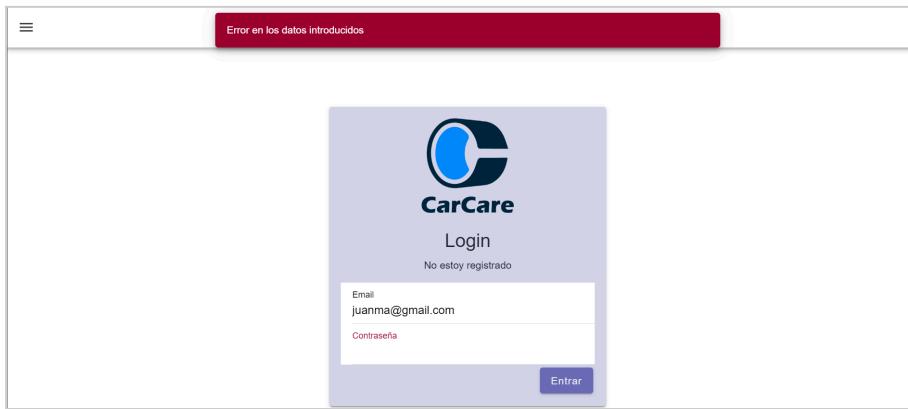
Detalle de la aplicación

Login



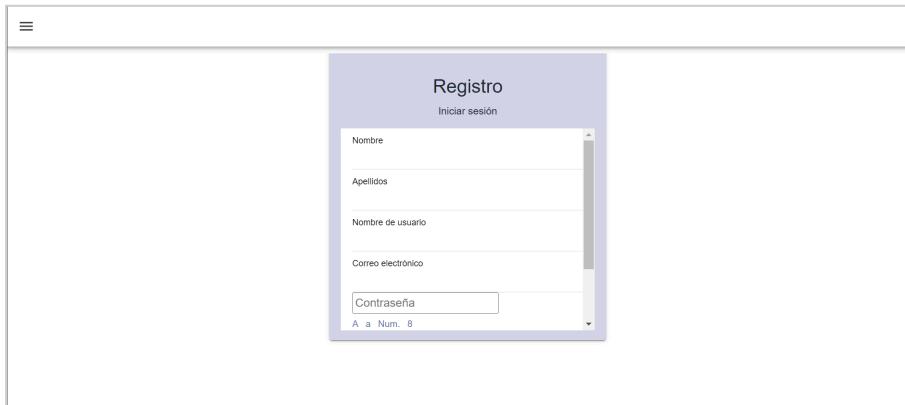
The screenshot shows the CarCare login interface. At the top center is the CarCare logo, which consists of a stylized blue 'C' inside a circle. Below the logo, the word 'CarCare' is written in a bold, sans-serif font. Underneath the logo, the word 'Login' is centered. To the right of 'Login', there is a link 'No estoy registrado'. Below these elements is a horizontal input field divided into two sections: 'Email' and 'Contraseña'. The 'Email' section contains the value 'juanma@gmail.com'. The 'Contraseña' section is partially visible. At the bottom right of the input field is a dark blue rectangular button labeled 'Entrar'.

- La aplicación cuenta con una pantalla de login para iniciar la sesión.
- El login se realiza con correo electrónico y contraseña.
- El correo electrónico queda almacenado en localstorage del navegador.
- Si se introducen mal los datos, se muestra un aviso de error.

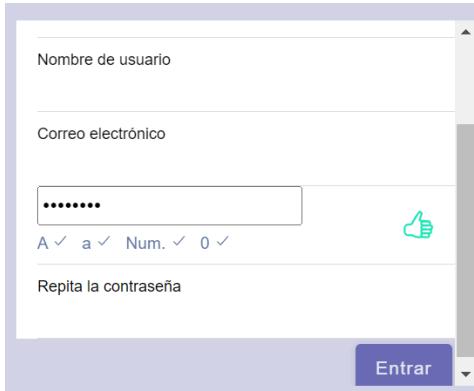


This screenshot is identical to the one above, showing the CarCare login page. However, it includes a red horizontal bar at the top with the text 'Error en los datos introducidos' in white. The rest of the interface, including the logo, 'Login' text, registration link, and input fields, is the same as the first screenshot.

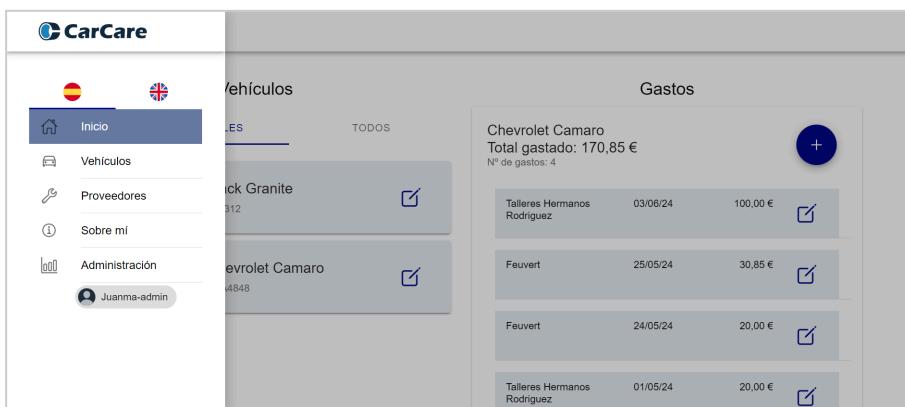
Pantalla de Registro



- La aplicación cuenta con una pantalla para registrar usuarios.
- Se controla que se introduzcan correctamente todos los datos.
- Hay control especial sobre la contraseña, solicita que tenga al menos una mayúscula, una minúscula, un número y 8 dígitos.
- Durante el proceso de inserción de la contraseña se va indicando si se cumplen los requisitos. Si cumple correctamente se muestra en pantalla.



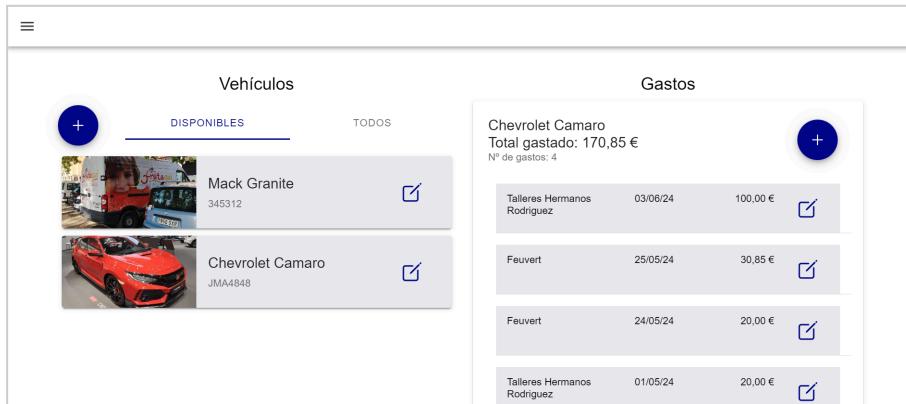
Barra de Menú principal



- Se muestra en toda la aplicación.

- Incluye botón para traducir la aplicación entre español e inglés.
- Si el usuario está logueado con el rol de Administrador, se mostrará la opción **Administración**.
- Permite el cierre de sesión del usuario.

Pantalla Inicio



- En la parte izquierda de la pantalla hay un listado con los vehículos registrados a nombre del usuario. Este listado se puede filtrar entre los vehículos disponibles o todos los registrados, incluyendo los no disponibles.
- Las imágenes de los vehículos se encuentran cargadas en Firebase Storage. Si no existe imagen asociada al vehículo, se muestra una imagen placeholder de la categoría del vehículo.
- Cuando se hace click en un vehículo, en la parte derecha se muestran su marca y modelo, así como el total gastado y el número de gastos. Debajo de la cabecera de los gastos se muestra el listado de gastos asociados.
- En esta pantalla hay acceso al formulario de alta, edición y borrado de vehículos y de gastos.

The top screenshot displays the 'Editar vehículo' (Edit vehicle) screen. It shows a form with the following fields:
 - Categoría: Coche
 - Marca: Chevrolet
 - Modelo: Camaro
 - Matrícula: JMA4848
 - A checked checkbox labeled 'Vehículo disponible' (Vehicle available)
 A blue 'GUARDAR' (Save) button is at the bottom.

The bottom screenshot displays the 'Editar gasto' (Edit expense) screen. It shows a form with the following fields:
 - Selecciona proveedor: Feuvvert
 - Importe: 30.85 €
 - Fecha del gasto: A calendar showing dates from 23 to 27, with the month set to mayo (May).
 A blue 'GUARDAR' (Save) button is at the bottom.

Pantalla Vehículos

The screen shows a list of vehicles with the following details:

- Kawasaki Ninja H2r**: Motocicleta, 3453RT, Fecha de registro: 16/05/24. Status: Vehículo no disponible (Vehicle not available).
- Mack Granite**: Camión, 345312, Fecha de registro: 01/05/24. Status: DISPONIBLE (Available).
- Chevrolet Camaro**: Coche, JMA4848, Fecha de registro: 07/02/20. Status: DISPONIBLE (Available).

- Se muestra un listado con todos los vehículos disponibles, así como información detallada de los mismos.
- Las categorías se muestran tanto con un campo de texto como con un ícono representativo.
- Los vehículos no disponibles se mostrarán con un texto indicativo y un efecto en la tarjeta para mostrar dicho estado.
- Existe un botón de compartir mediante el cual se puede enviar el listado de vehículos a cualquier aplicación admitida en el dispositivo.

Pantalla Proveedores

Listado de proveedores			
Nombre	Categoría	Teléfono	
Talleres Hermanos Rodriguez	Taller	952962094	<input checked="" type="checkbox"/>
Repsol	Gasolinera	951345588	<input checked="" type="checkbox"/>
Cepsa	Gasolinera		<input checked="" type="checkbox"/>
Feuvert	Taller	952131313	<input checked="" type="checkbox"/>
Línea Directa	Compañía De Seguros	44747512	<input checked="" type="checkbox"/>
Lavalava	Otros		<input checked="" type="checkbox"/>
Grúas Peláez	Grúa	951212135	<input checked="" type="checkbox"/> +

- Muestra un listado de los proveedores, así como la categoría a la que pertenece y su teléfono de contacto.

- En esta pantalla se puede dar de alta, editar y eliminar cualquier proveedor mediante el formulario dinámico correspondiente.

Nombre
Feuvert

Telefono
952131313

Categoría

Taller

GUARDAR

Pantalla Sobre mí

Autor: Juan Manuel Gutiérrez Mellado
Frontend Developer Jr.
Desarrollador de aplicaciones multiplataforma y de aplicaciones web. Estudios de HTML, CSS, JavaCript, Java, MySQL. Conocimientos C, Python, PHP, Kotlin y de las plataformas AWS y Docker.

GITHUB LINKEDIN

VOLVER

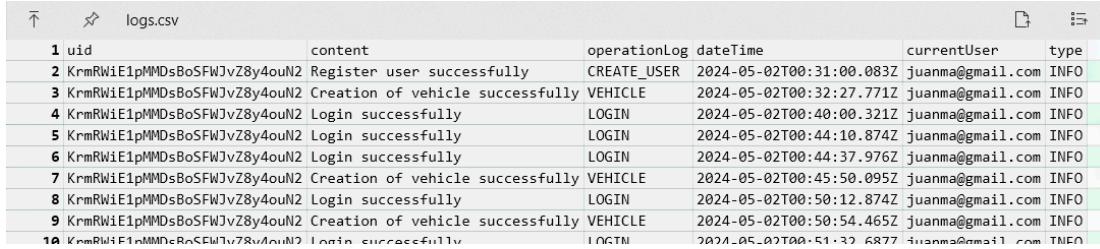
- Pantalla con información del autor de la aplicación.
- Incluye enlace a GitHub y a Linkedin del autor.

Pantalla Administración



- Pantalla de administración a la que se puede acceder únicamente si el usuario logueado tiene el rol de Administrador.

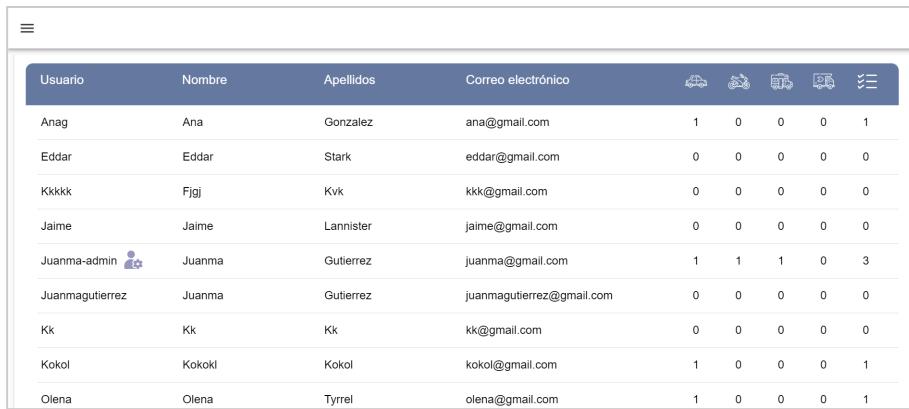
- Hay un botón de exportación de datos, que exporta un archivo en formato .CSV con el registro de la aplicación.
- El formato de exportación sería el siguiente:



The screenshot shows a CSV file titled 'logs.csv' with the following data:

uid	content	operationLog	dateTime	currentUser	type
2	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Register user successfully	CREATE_USER	2024-05-02T00:31:00.083Z	juanna@gmail.com	INFO
3	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:32:27.771Z	juanna@gmail.com	INFO
4	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:40:00.321Z	juanna@gmail.com	INFO
5	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:44:10.874Z	juanna@gmail.com	INFO
6	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:44:37.976Z	juanna@gmail.com	INFO
7	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:45:50.095Z	juanna@gmail.com	INFO
8	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:50:12.874Z	juanna@gmail.com	INFO
9	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:50:54.465Z	juanna@gmail.com	INFO
10	KrmRwiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:51:30.687Z	juanna@gmail.com	INFO

- Se incluyen tres gráficas, con las marcas de los vehículos más utilizados, número de vehículos por categoría y porcentaje de vehículos por categoría.
- Se muestra una tabla con el listado de usuarios, así como los vehículos que tiene registrados cada uno de ellos.
- En el caso del administrador, se muestra con un ícono indicativo junto al nombre.



The screenshot shows a table with the following data:

Usuario	Nombre	Apellidos	Correo electrónico	Carro	Bicicleta	Tren	Avión	Barco	Avión
Anag	Ana	Gonzalez	ana@gmail.com	1	0	0	0	0	1
Eddar	Eddar	Stark	eddar@gmail.com	0	0	0	0	0	0
Kkkkk	Fjgi	Kvk	kkk@gmail.com	0	0	0	0	0	0
Jaime	Jaime	Lannister	jaime@gmail.com	0	0	0	0	0	0
Juanma-admin	Juanma	Gutierrez	juanma@gmail.com	1	1	1	0	0	3
Juanmagutierrez	Juanma	Gutierrez	juanmagutierrez@gmail.com	0	0	0	0	0	0
Kk	Kk	Kk	kk@gmail.com	0	0	0	0	0	0
Kokol	Kokoli	Kokol	kokoli@gmail.com	1	0	0	0	0	1
Olena	Olena	Tyrrel	olena@gmail.com	1	0	0	0	0	1

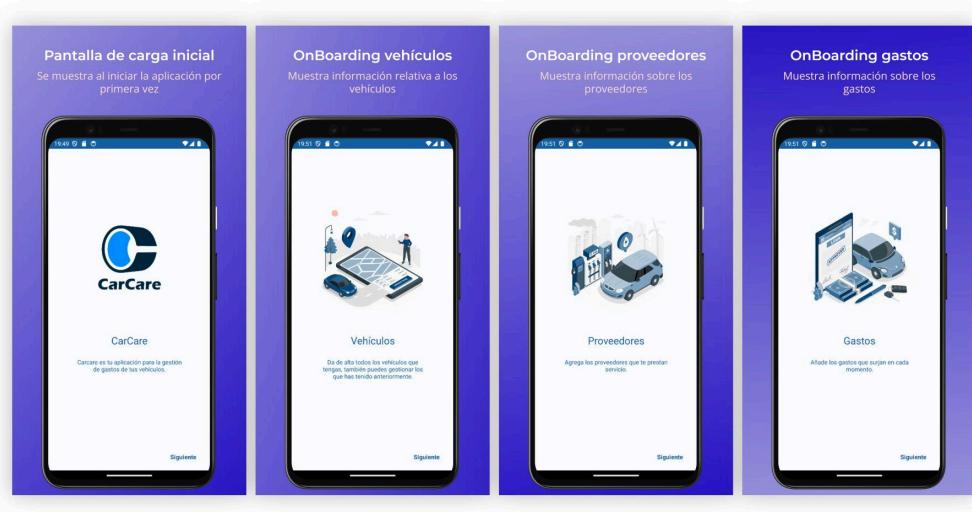
Aplicación Kotlin

Disponible en este repositorio: <https://github.com/Juanma-Gutierrez/CarCare-Kotlin>

Aplicación realizada desde cero durante el tercer trimestre como entrega final del TFC.

Detalle de la aplicación

Pantallas OnBoarding

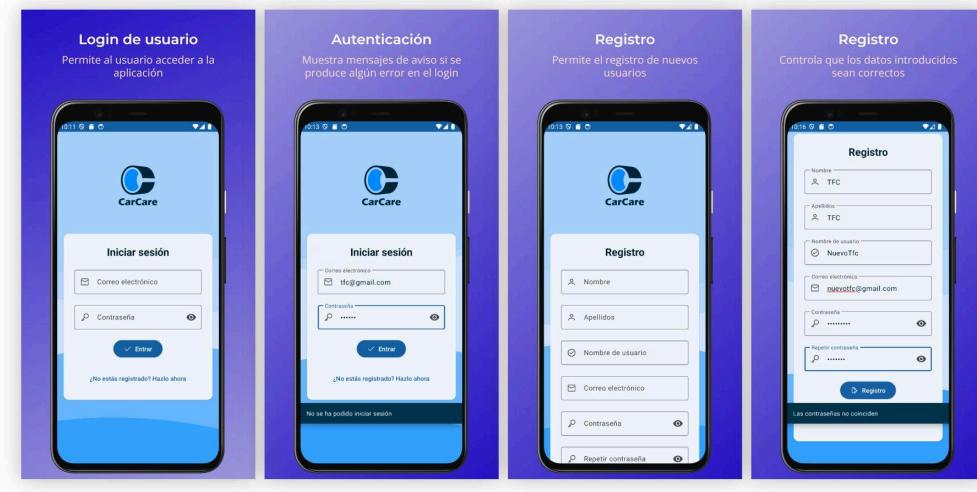


La aplicación cuenta con varias pantallas **OnBoarding**, que se muestran la primera vez que se accede a la aplicación. Estas pantallas muestran información sobre:

- Qué hace la aplicación.
- Cómo gestionar los vehículos.
- Cómo gestionar los proveedores.
- Cómo gestionar los gastos.

Estas pantallas OnBoarding se generan dinámicamente desde la aplicación.

Pantallas Login y Registro



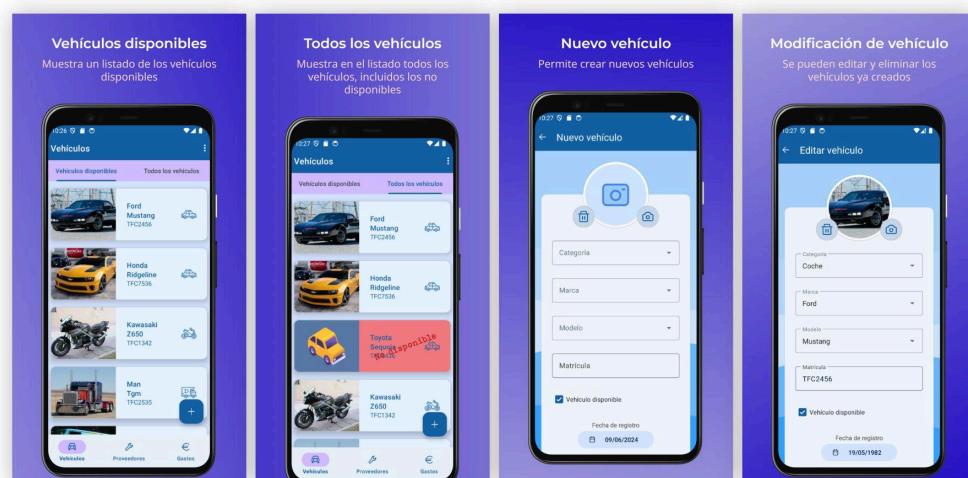
Login

- Solicita correo electrónico y contraseña para realizar el login.
- Si se produce algún error, se muestra un snack bar de aviso.

Registro

- Pide los datos del usuario.
- Controla si el correo electrónico ya está previamente en uso.
- Controla que la contraseña cumpla los requisitos de seguridad y que tanto la contraseña como la confirmación de contraseña coincidan.
- Una vez realizado el registro, automáticamente redirige al usuario a la pantalla inicial de la aplicación con el login hecho.

Pantalla Vehículos

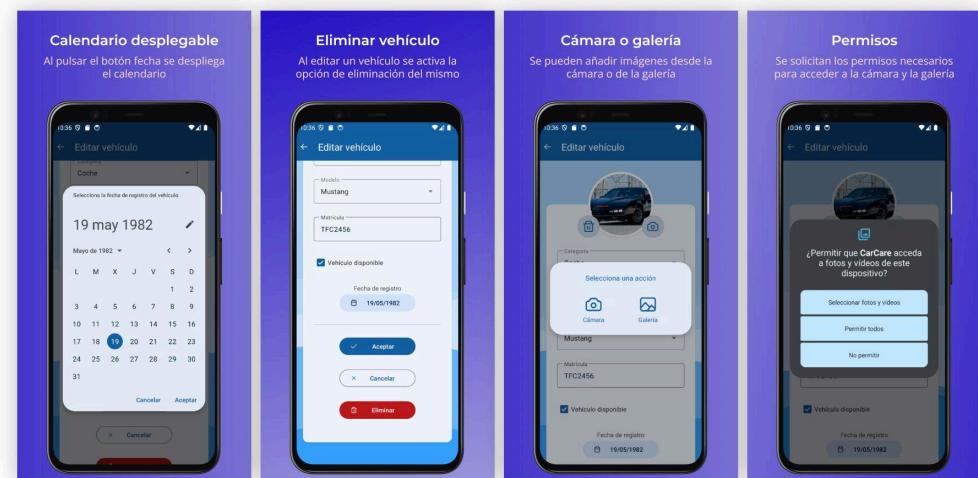


Pantalla listado de vehículos

- En esta pantalla se muestra un listado con los vehículos del usuario.
- Si no existe ningún vehículo creado, muestra un aviso.
- Se pueden filtrar por vehículos disponibles o todos los vehículos. Si el vehículo no está disponible, se muestra con filtro rojo y mensaje indicativo.
- Se muestra un ícono con la categoría correspondiente del vehículo.

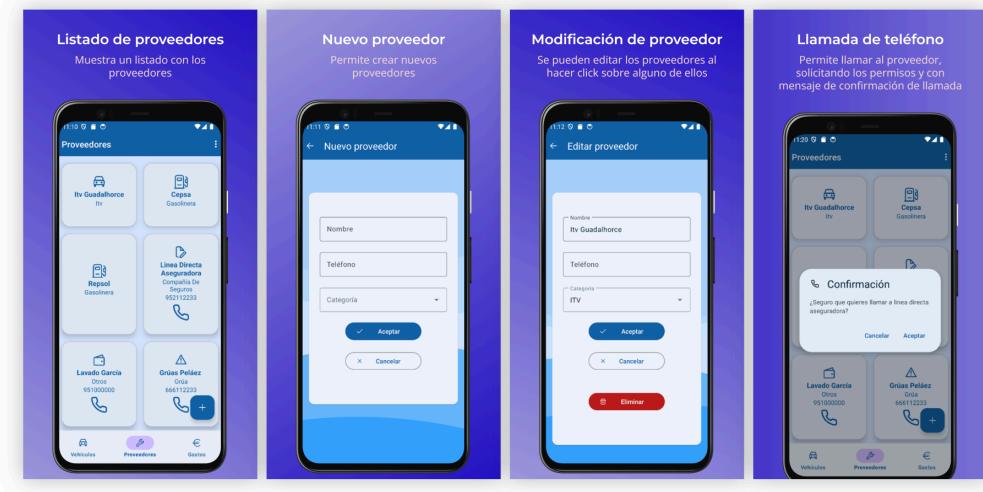
Pantalla detalle de vehículo

- Se pueden añadir nuevos vehículos, mostrando un formulario en blanco.
- Se pueden modificar vehículos del usuario, cargando todos los datos del vehículo.
- La carga de la categoría, marca y modelo se realizan de forma dinámica mediante consultas a la API desarrollada en Python para el proyecto.



- Al pulsar sobre el botón con la **fecha de registro**, se despliega el calendario interactivo.
- Si se le ha hecho click a un vehículo, mostrará la opción de eliminar. Esta opción permanece oculta en la creación de vehículo.
- Al pulsar en el botón de **cámara**, mostrará un diálogo solicitando al usuario de dónde se va a capturar la imagen, de la cámara o de la galería.
- Si no se han concedido previamente los permisos de acceso a cámara o galería, mostrará la solicitud de permisos correspondiente.
- La imagen capturada pasa por un proceso de compresión y redimensionado para minimizar el espacio en Firebase Storage.

Pantalla Proveedores



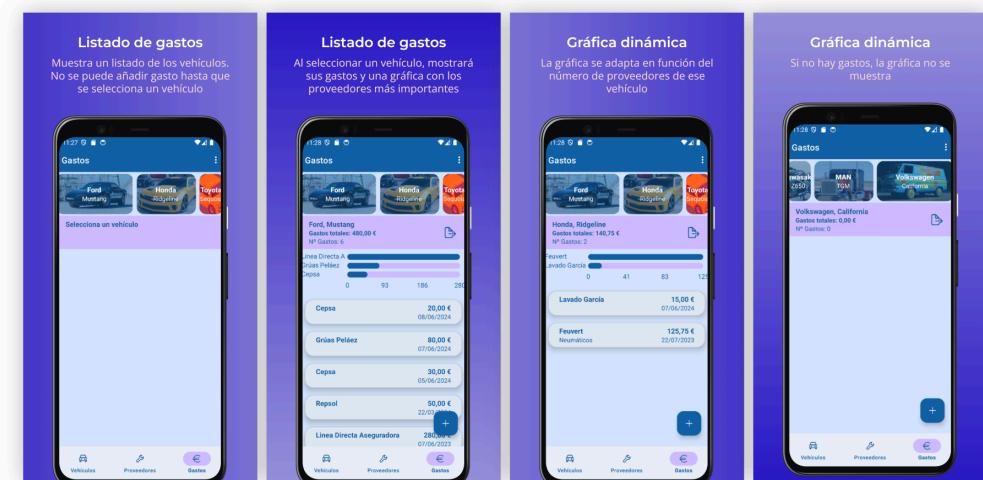
Pantalla listado de proveedores

- Muestra un listado con los proveedores.
- Si no existe ningún proveedor creado, muestra un aviso.
- Se muestra el nombre, la categoría, un ícono de la categoría correspondiente y el teléfono si lo tuviera.
- Si el proveedor tiene teléfono, al hacer click sobre el ícono del teléfono realiza una llamada.

Pantalla detalle de proveedor

- Permite crear un nuevo proveedor.
- Si se hace click sobre un proveedor, entra en la pantalla de edición.

Pantalla Gastos



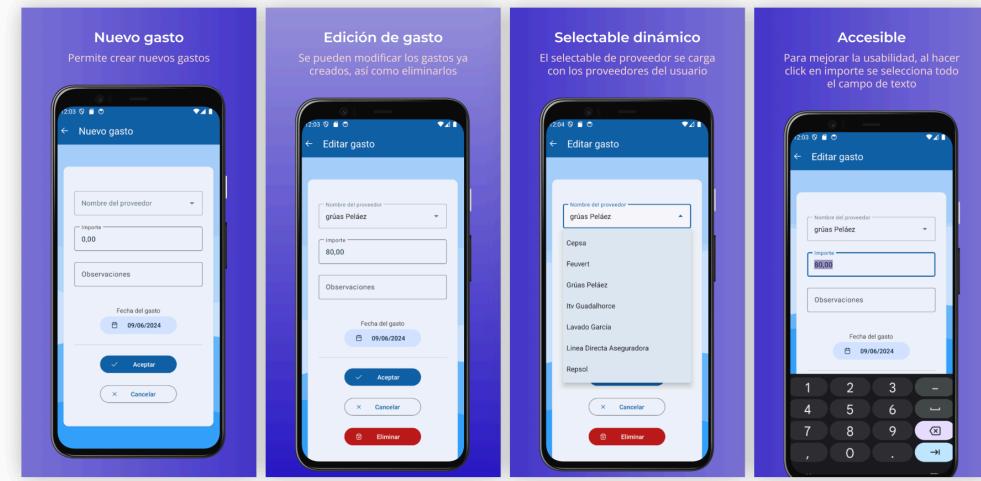
Pantalla listado de gastos

- Se muestra un carousel de Material Design en la parte superior de la pantalla.
- Los vehículos disponibles se muestran con un filtro azul y los no disponibles con un filtro rojo.
- Si no existe ningún proveedor creado, muestra un aviso.
- Mientras no se haga click en ningún vehículo, el **fab** de añadir gasto estará oculto, ya que no se puede asignar un gasto si no se ha indicado a qué vehículo asignarlo.
- Al hacer click en alguno de los vehículos, se muestra la siguiente información organizada en tres bloques separados:
 - Bloque totales:
 - Se muestran la marca y el modelo del vehículo.
 - Gastos totales de este vehículo.
 - Número de gastos del vehículo.
 - Botón de exportación de datos, con la fecha de exportación, vehículo, listado de gastos ordenado por fecha, con la fecha, proveedor e importe, número de gastos y gastos totales.
 - Ejemplo de exportación de datos:



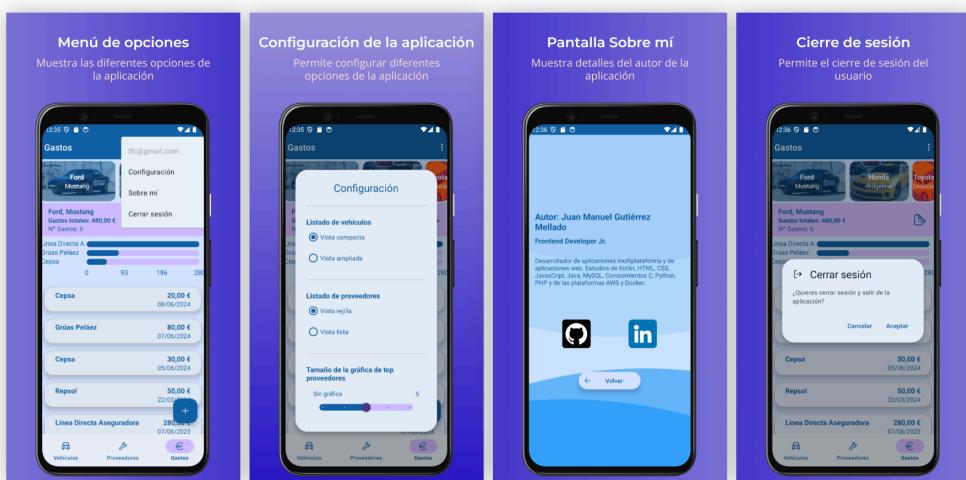
- Bloque gráfica:
 - Se muestra una gráfica animada con los proveedores que más importe han acumulado en este vehículo.
 - Si hay solamente un proveedor o menos, no se muestra este bloque.
 - El tamaño de la gráfica es adaptativa, hasta un máximo de 5 proveedores.
- Bloque listado de gastos:
 - Se muestra un recyclerView con un listado de los gastos asociados al vehículo seleccionado.
 - Al hacer click sobre alguno de estos gastos se accede a la pantalla de edición del gasto.

Pantalla detalle de gasto



- Permite la creación de un nuevo gasto.
- También permite la edición y eliminación de un gasto ya creado.
- El selectable se carga con los datos de proveedores del usuario.
- Para mejorar la accesibilidad, al hacer click en el importe, se selecciona todo el campo.
- Al hacer click sobre la fecha del gasto, se despliega el calendario dinámico.

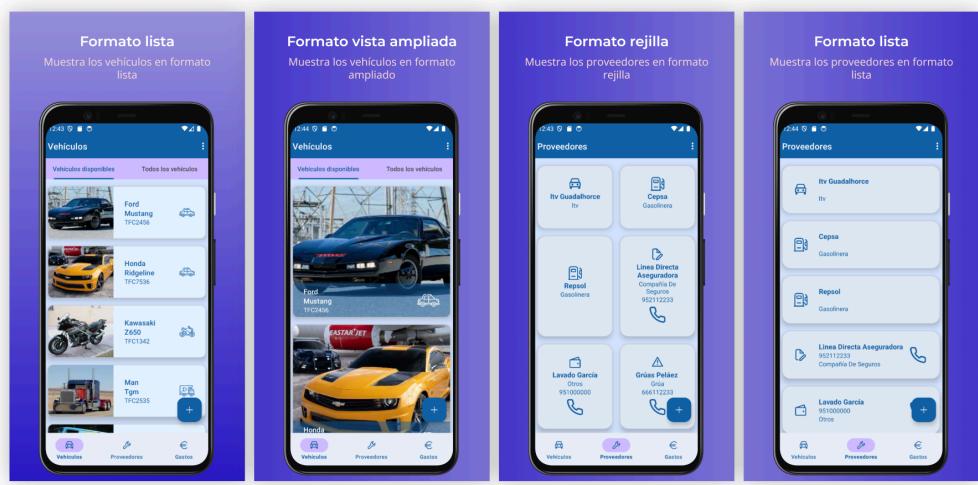
Menú de Opciones



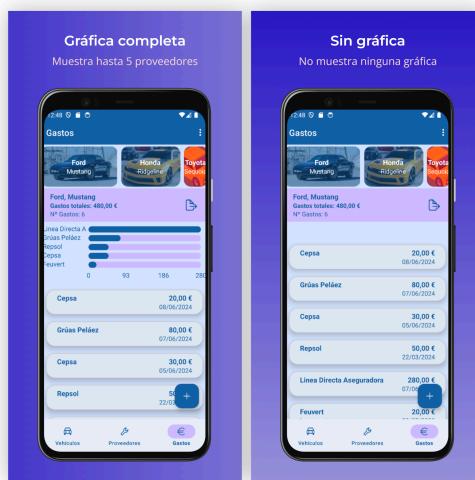
El menú de opciones incluye la siguiente información:

- Correo electrónico del usuario logueado.
- Opción de configuración.
 - La pantalla de configuración permite modificar diferentes opciones de la aplicación:
 - Listado de vehículos: En formato lista compacta o vista ampliada.

- Listado de proveedores: En formato vista rejilla o vista lista.

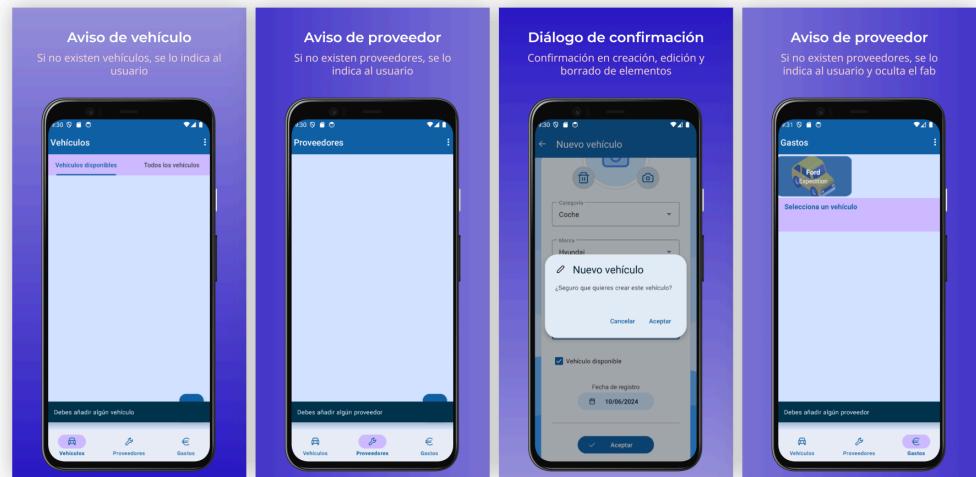


- Tamaño de la gráfica de proveedores: hasta un máximo de 5 proveedores o sin gráfica si así se desea.



- Pantalla Sobre mí. Muestra datos del autor de la aplicación, así como enlaces a GitHub y Linkedin del autor.
- Opción Cerrar sesión: Permite cerrar la sesión del usuario activo.

Mensajes de aviso



La aplicación tiene múltiples mensajes de aviso para informar al usuario de errores o consejos de uso. Estos son algunos ejemplos:

- Si no existe ningún vehículo, informa al usuario.
 - Si no existe ningún proveedor, informa al usuario.
 - Si entra en la pantalla de gastos, informa si no hay ningún vehículo. Si hay al menos un vehículo pero no hay ningún proveedor, informa y no muestra el botón de añadir gasto.
 - Para las operaciones de creación, edición y eliminación muestra un dialogo para confirmar la operación.
 - Para el cierre de sesión muestra un diálogo de confirmación.

Persistencia de datos

Room

En almacenamiento local se guardan los datos de los vehículos, mediante el uso de [Room](#). También existe persistencia de datos con Firebase, de manera que la aplicación puede ser utilizada parcialmente aunque no se disponga de conexión con la base de datos.

Database Inspector											
Pixel 3a API 34 extension level 7 x86 64 > com.juanmaGutierrez.carcare											
Database Inspector		Network Inspector		Background Task Inspector							
Databases											50
+	app_database	1	240607122011-atk	1	Ford	car	2024-06-07T12:20:11Z	vehicleImages/r/WI	Mustang	TFC4256	/vehicle/240607122011-atk
>	room_master_table	2	240607122151-rpu	1	Honda	car	2024-06-07T12:21:51Z	vehicleImages/r/WI	Ridgeline	TFC7536	/vehicle/240607122151-rpu
>	vehicles	3	240607122327-fd	0	Toyota	car	2024-06-07T12:23:27Z	null	Sequoia	TFC6436	/vehicle/240607122327-fd
		4	240607122625-ti	1	Kawasaki	motorcycle	2024-06-07T12:26:25Z	vehicleImages/r/WI	Z650	TFC1342	/vehicle/240607122625-ti
		5	240607122709-75	1	Volkswagen	van	2024-06-07T12:27:09Z	vehicleImages/r/WI	California	TFC9766	/vehicle/240607122709-75
		6	240607122803-dc	1	MAN	truck	2024-06-07T12:28:03Z	vehicleImages/r/WI	TGM	TFC2535	/vehicle/240607122803-dc

SharedPreferences

También existe persistencia de datos mediante el uso de SharedPreferences, para almacenar la configuración personalizada de la aplicación por parte del usuario.

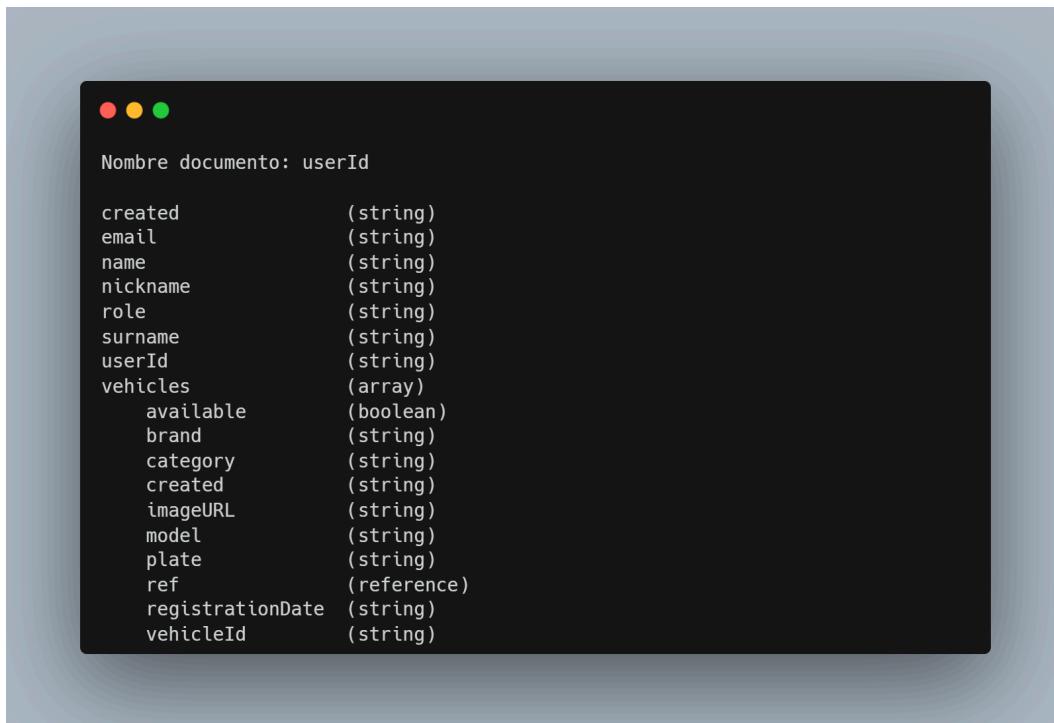
Modelos y persistencia de datos

El proyecto está realizado sobre una base de datos no relacional en Firebase. La estructura de datos y relación entre tablas tendría una estructura de este tipo:



Modelo de datos en Firebase

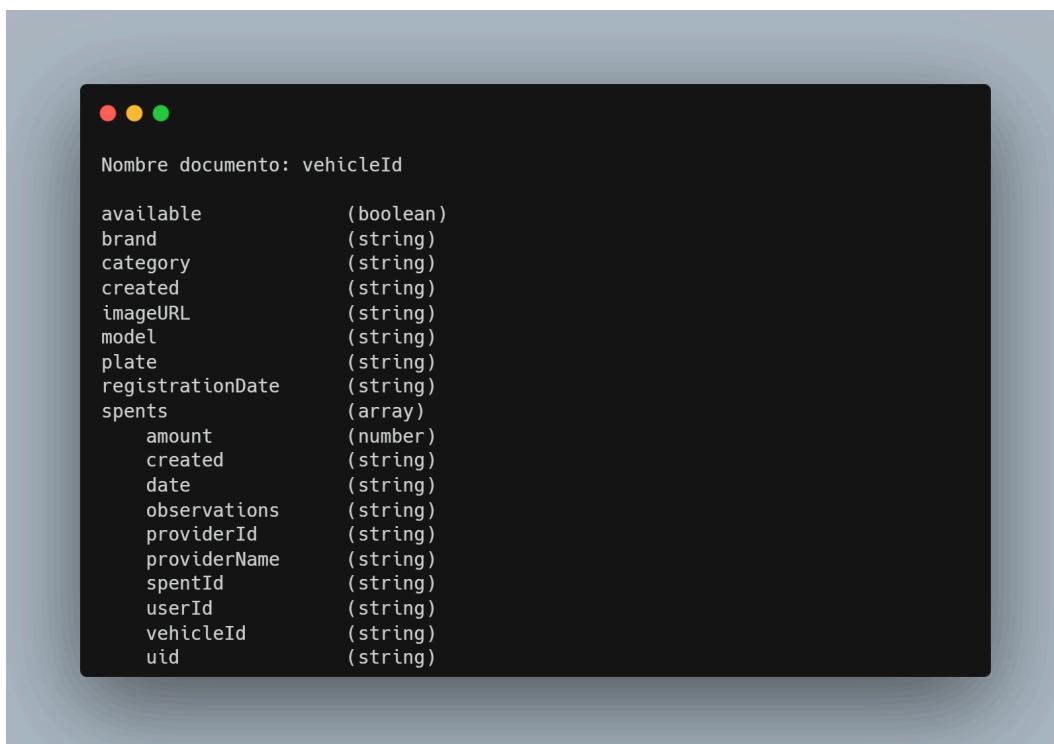
Colección user



Nombre documento: userId

```
created          (string)
email            (string)
name             (string)
nickname         (string)
role              (string)
surname          (string)
userId            (string)
vehicles          (array)
  available      (boolean)
  brand           (string)
  category        (string)
  created          (string)
  imageURL        (string)
  model            (string)
  plate             (string)
  ref              (reference)
  registrationDate (string)
  vehicleId       (string)
```

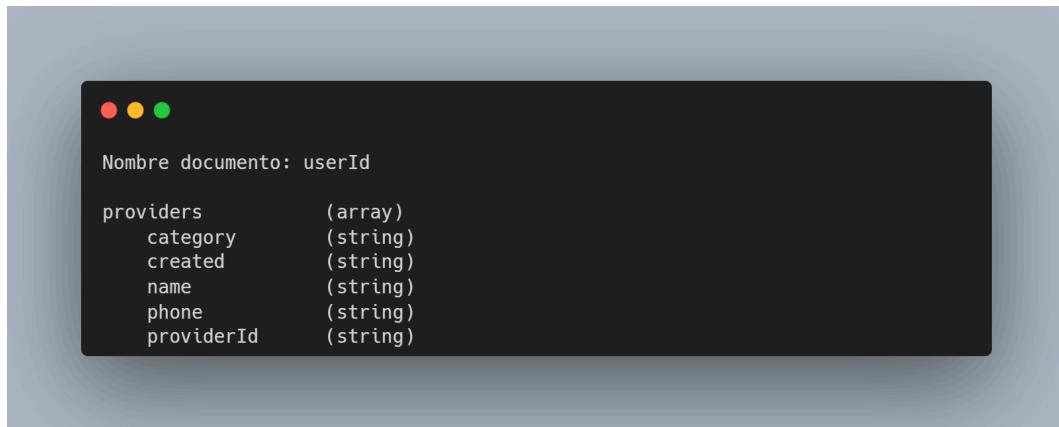
Colección vehicle



Nombre documento: vehicleId

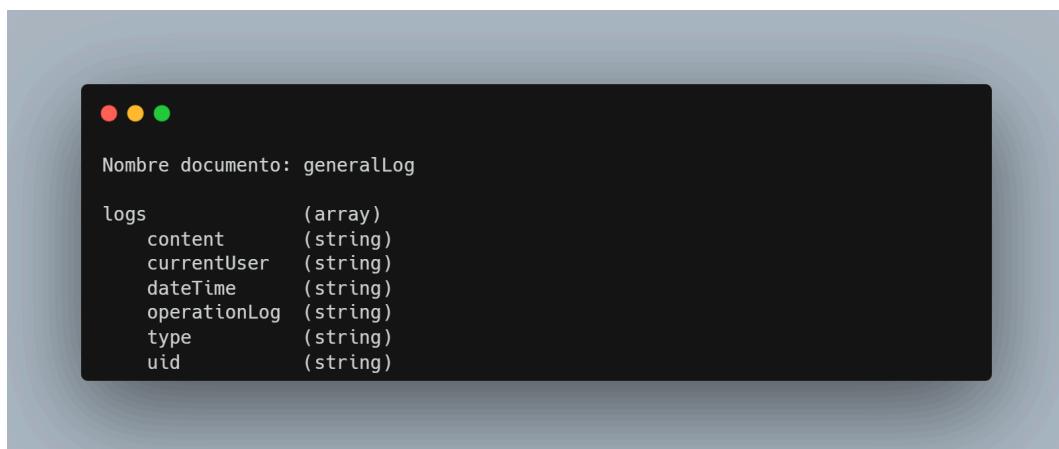
```
available        (boolean)
brand            (string)
category         (string)
created          (string)
imageURL        (string)
model            (string)
plate             (string)
registrationDate (string)
spents            (array)
  amount          (number)
  created          (string)
  date              (string)
  observations      (string)
  providerId        (string)
  providerName       (string)
  spentId            (string)
  userId              (string)
  vehicleId          (string)
  uid                 (string)
```

Colección provider



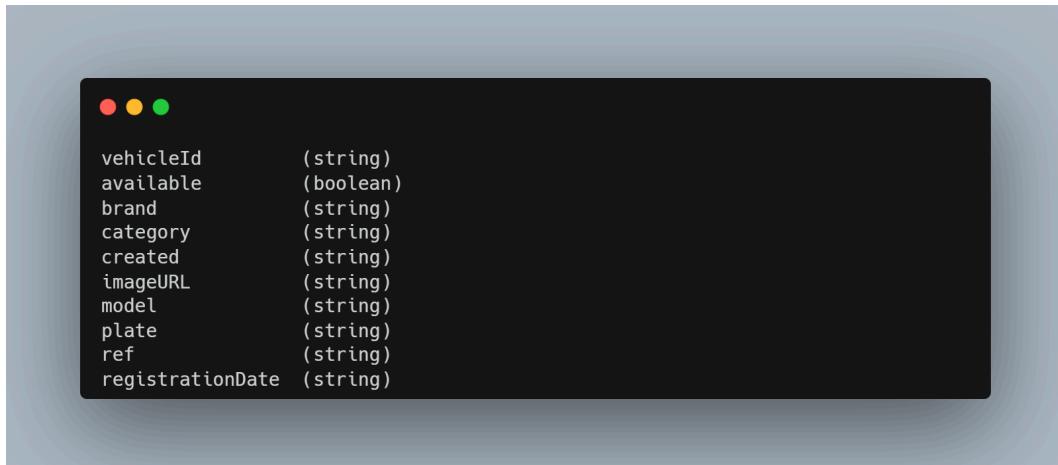
He incluido una colección que registra en un log los diferentes eventos realizados en la aplicación. Estos eventos podrán ser mostrados posteriormente en la exportación a PowerBI.

Colección log



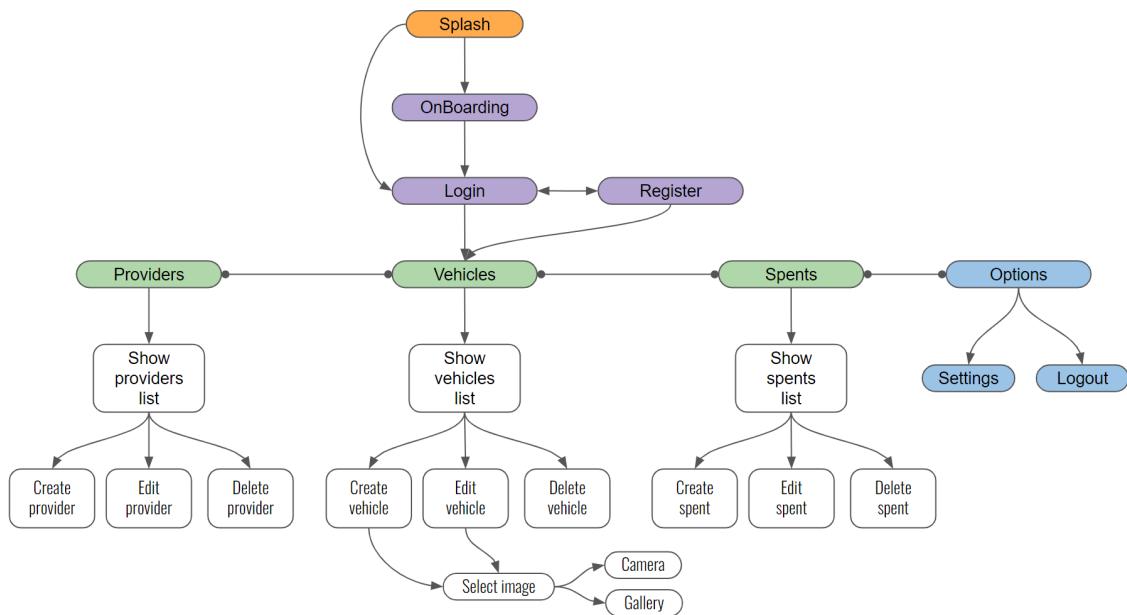
Modelo de datos local en Room

Vehicles:



```
vehicleId      (string)
available      (boolean)
brand          (string)
category        (string)
created        (string)
imageURL       (string)
model          (string)
plate          (string)
ref            (string)
registrationDate (string)
```

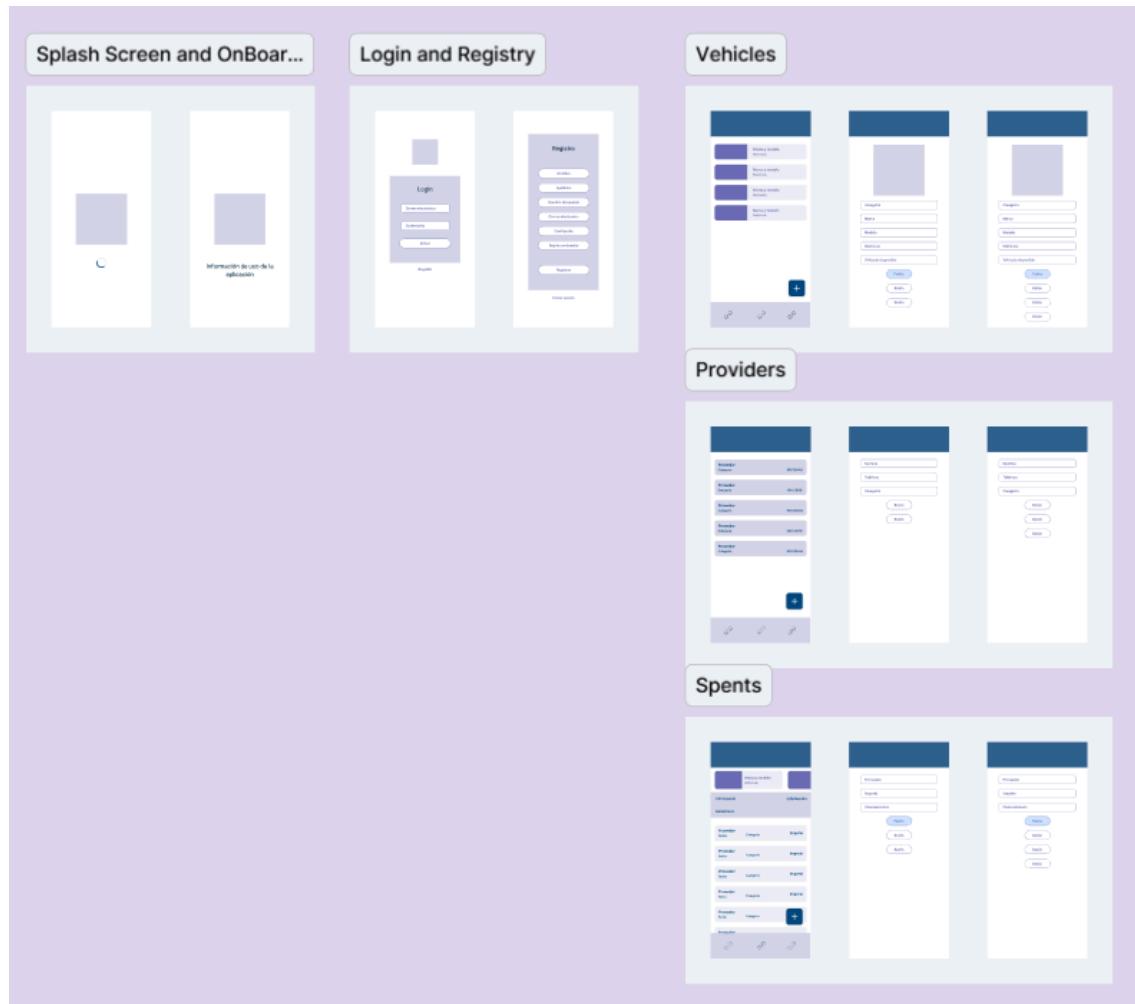
Diagrama de flujo de la aplicación



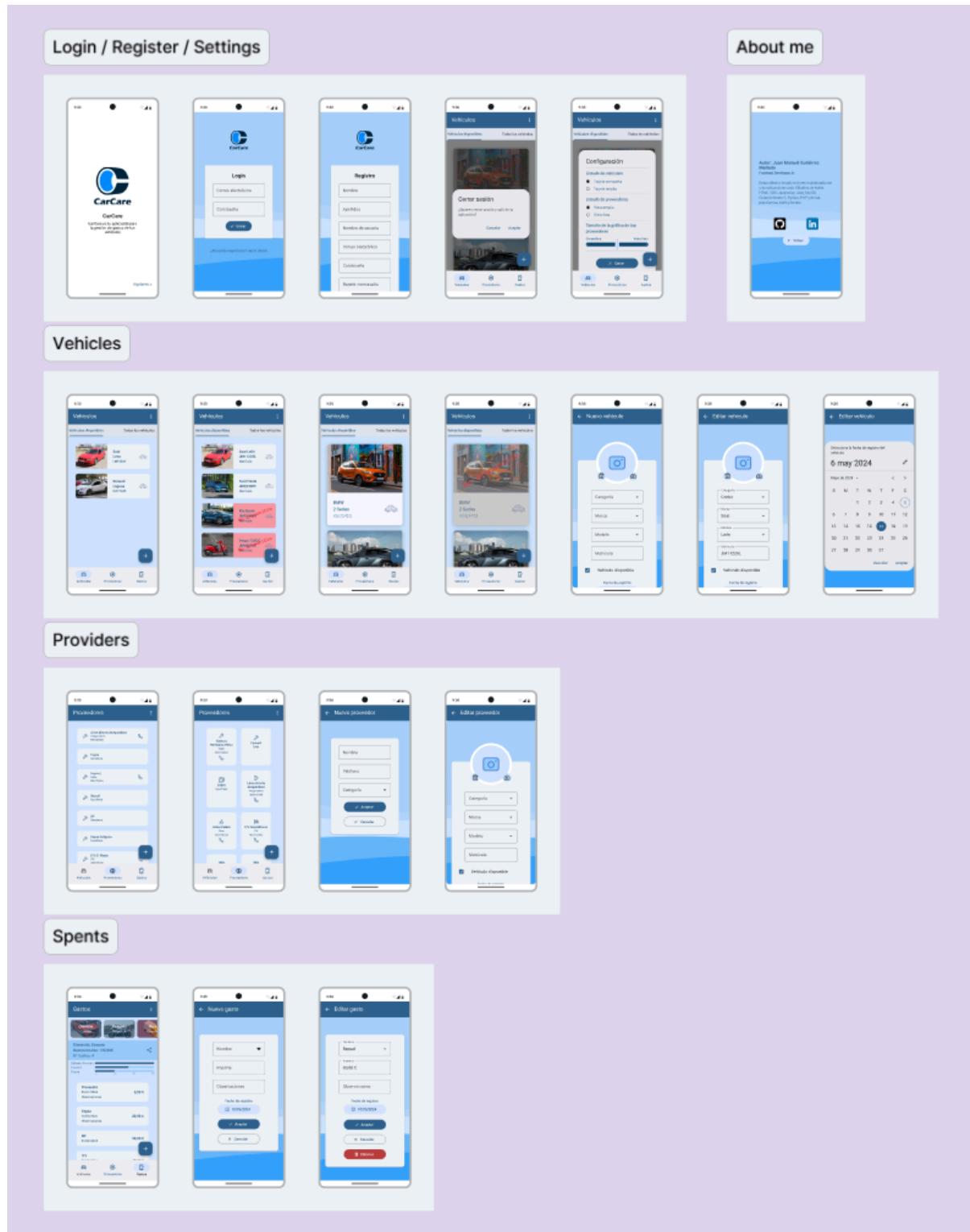
Prototipo en Figma

En Figma está realizado el prototipo de la aplicación, tanto en la versión wireframe como en la versión final.

Prototipo en modo wireframe: [enlace](#)



Prototipo en la versión final: [enlace](#)

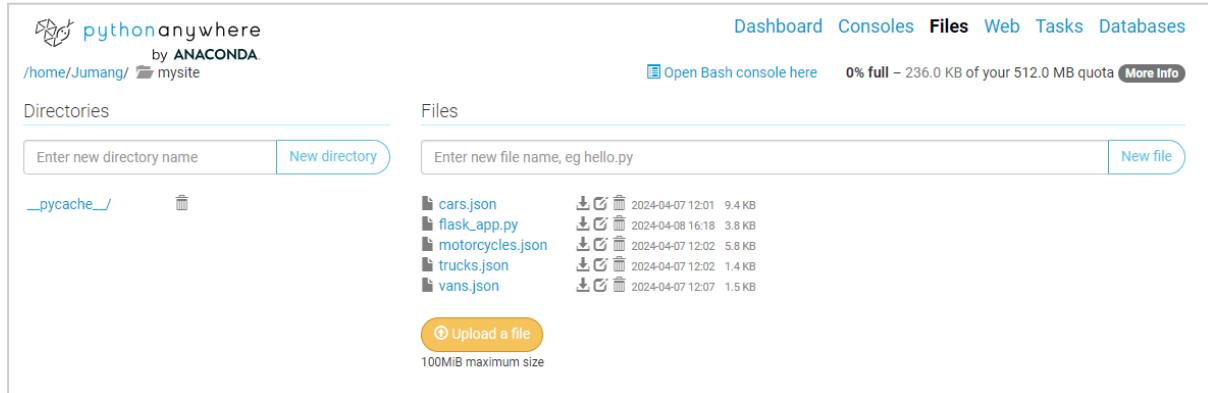


Modelo funcional desde Figma: [enlace](#)

API en Python

Para el proyecto, he desarrollado una API en Python que devolverá marcas y modelos de vehículos. En función de los diferentes endpoints recibirá diferentes respuestas.

La API de la aplicación está alojada en la web pythonanywhere.com, así como los archivos Json que proporcionan las respuestas, cargando el archivo `flask_app.py` que es donde está la lógica de programación de la API.



Captura de pantalla de la aplicación `flask_app.py` y de los json alojados en pythonanywhere.com

Documentación de la API

[Build, Collaborate & Integrate APIs | SwaggerHub](#)

Ejemplos de respuestas por categoría:

- Coches: <https://jumang.pythonanywhere.com/api/cars/brands>

The screenshot shows a terminal window with three colored dots (red, yellow, green) at the top. The window displays the following JSON output:

```
{  
  "brands": ["Toyota", "Ford", "Honda", ...]  
}
```

- Motocicletas: <https://jumang.pythonanywhere.com/api/motorcycles/brands>

```
●
{
  "brands": ["Honda", "Yamaha", "Kawasaki", ...]
}
```

- Furgonetas: <https://jumang.pythonanywhere.com/api/vans/brands>

```
●
{
  "brands": ["Mercedes-Benz", "Ford", "Volkswagen", ...]
}
```

- Camiones: <https://jumang.pythonanywhere.com/api/trucks/brands>

```
●
{
  "brands": ["Volvo", "Mercedes-Benz", "Scania", ...]
}
```

Ejemplo de respuesta por marca

- Todos los modelos de coche de la marca Toyota
<https://jumang.pythonanywhere.com/api/cars/models/Toyota>

```
●
{
  "models": ["Camry", "Corolla", "RAV4", ...]
}
```

- Todos los modelos de motocicleta de la marca Kawasaki
<https://jumang.pythonanywhere.com/api/motorcycles/models/Kawasaki>

```
{  
    "models": ["Ninja ZX-10R", "Ninja ZX-6R", "Ninja 400", ...]  
}
```

Respuesta errónea

- En caso de respuesta errónea, se devolverá la siguiente respuesta:
<https://jumang.pythonanywhere.com/api/motorcycles/models/ERROR>

```
{  
    "message": "Bad request: Invalid brand supply"  
}
```

Problemas encontrados:

Los problemas de CORS, se solucionan de esta manera:

- Abrir una terminal bash en pythonanywhere.

Instalar flask-cors.

```
pip install flask-cors
```

- Con este paquete instalado, se pueden gestionar las CORS para que estén habilitadas para toda la API.

Código completo de la aplicación Python

```
"""
@package my_flask_app
Documentation for the Flask application.

This module provides a Flask web application with several endpoints
to retrieve data about cars, motorcycles, vans, and trucks from JSON files.
"""

from flask import Flask, request, jsonify
from flask_cors import CORS

import json
import os

# Error message
errorMessage = {'message': 'Bad request: Invalid brand supply'}

# Get the path
localPath = os.path.dirname(os.path.abspath(__file__))

# Build paths for each JSON file
path_cars_json = os.path.join(localPath, 'cars.json')
path_motorcycles_json = os.path.join(localPath, 'motorcycles.json')
path_vans_json = os.path.join(localPath, 'vans.json')
path_trucks_json = os.path.join(localPath, 'trucks.json')

# Load cars from JSON file
with open(path_cars_json) as f:
    cars = json.load(f)

# Load motorcycles from JSON file
with open(path_motorcycles_json) as f:
    motorcycles = json.load(f)

# Load vans from JSON file
with open(path_vans_json) as f:
    vans = json.load(f)

# Load trucks from JSON file
with open(path_trucks_json) as f:
    trucks = json.load(f)

app = Flask(__name__)
CORS(app)

@app.route('/api/cars', methods=['GET'])
def getCars():
    """
    Endpoint to get all cars.

    @return: JSON response with all cars data.
    """
    return jsonify(cars)

@app.route('/api/cars/brands', methods=['GET'])
def getCarsBrands():
    """
    Endpoint to get all car brands.
    """
```

```
@return: JSON response with a list of car brands.  
"""  
brands = list(cars["data"].keys())  
return jsonify({"brands": brands})  
  
@app.route('/api/cars/models/<brand>', methods=['GET'])  
def getCarsModels(brand):  
    """  
    Endpoint to get models of a specific car brand.  
  
    @param brand: The brand of the car.  
    @return: JSON response with a list of models for the specified brand,  
    or an error message if the brand is not found.  
    """  
    if brand in cars["data"]:  
        models = cars["data"][brand]  
        return jsonify({"models":models})  
    else:  
        return jsonify(errorMessage), 400  
  
@app.route('/api/motorcycles', methods=['GET'])  
def getMotorcycles():  
    """  
    Endpoint to get all motorcycles.  
  
    @return: JSON response with all motorcycles data.  
    """  
    return jsonify(motorcycles)  
  
@app.route('/api/motorcycles/brands', methods=['GET'])  
def getMotorcyclesBrands():  
    """  
    Endpoint to get all motorcycle brands.  
  
    @return: JSON response with a list of motorcycle brands.  
    """  
    brands = list(motorcycles["data"].keys())  
    return jsonify({"brands": brands})  
  
@app.route('/api/motorcycles/models/<brand>', methods=['GET'])  
def getMotorcyclesModels(brand):  
    """  
    Endpoint to get models of a specific motorcycle brand.  
  
    @param brand: The brand of the motorcycle.  
    @return: JSON response with a list of models for the specified brand,  
    or an error message if the brand is not found.  
    """  
    if brand in motorcycles["data"]:  
        models = motorcycles["data"][brand]  
        return jsonify({"models":models})  
    else:  
        return jsonify(errorMessage), 400  
  
@app.route('/api/vans', methods=['GET'])  
def getVans():
```

```
"""
Endpoint to get all vans.

@return: JSON response with all vans data.
"""
return jsonify(vans)

@app.route('/api/vans/brands', methods=['GET'])
def get_vans_brands():
    """
    Endpoint to get all van brands.

    @return: JSON response with a list of van brands.
    """
    brands = list(vans["data"].keys())
    return jsonify({"brands": brands})

@app.route('/api/vans/models/<brand>', methods=['GET'])
def getVansModels(brand):
    """
    Endpoint to get models of a specific van brand.

    @param brand: The brand of the van.
    @return: JSON response with a list of models for the specified brand,
    or an error message if the brand is not found.
    """
    if brand in vans["data"]:
        models = vans["data"][brand]
        return jsonify({"models":models})
    else:
        return jsonify(errorMessage), 400

@app.route('/api/trucks', methods=['GET'])
def getTrucks():
    """
    Endpoint to get all trucks.

    @return: JSON response with all trucks data.
    """
    return jsonify(trucks)

@app.route('/api/trucks/brands', methods=['GET'])
def getTrucksBrands():
    """
    Endpoint to get all truck brands.

    @return: JSON response with a list of truck brands.
    """
    brands = list(trucks["data"].keys())
    return jsonify({"brands": brands})

@app.route('/api/trucks/models/<brand>', methods=['GET'])
def getTrucksModels(brand):
    """
    Endpoint to get models of a specific truck brand.

    @param brand: The brand of the truck.
    @return: JSON response with a list of models for the specified brand,
    or an error message if the brand is not found.
    """
```

```
or an error message if the brand is not found.
"""
if brand in trucks["data"]:
    models = trucks["data"][brand]
    return jsonify({"models":models})
else:
    return jsonify(errorMessage), 400

@app.route('/api/brands', methods=['GET'])
def getAllBrands():
    """
    Endpoint to get all brands of all vehicle categories.

    @return: JSON response with a dictionary containing lists of brands for
    cars, motorcycles, vans, and trucks.
    """
    all_brands = {
        "status": "success",
        "data": [
            "cars": list(cars["data"].keys()),
            "motorcycles": list(motorcycles["data"].keys()),
            "vans": list(vans["data"].keys()),
            "trucks": list(trucks["data"].keys())
        ]
    }
    return jsonify(all_brands)

@app.route('/')
def hello_world():
    """
    Test endpoint.

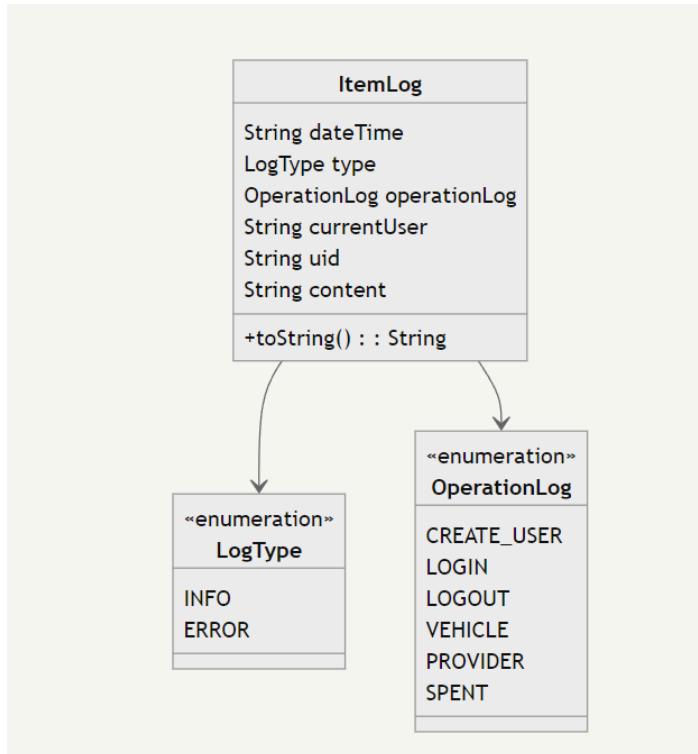
    @return: Simple 'Hello world!' message.
    """
    return 'Hello world!'

@app.route('/api')
def hello_world_api():
    """
    API test endpoint.

    @return: Simple 'Hello world!' message for the API.
    """
    return 'Hello world!'
```

Gestión de datos del Log

El log generado con las operaciones realizadas en nuestra aplicación, tanto web como Android, siguen este modelo de datos:



De este modo, tendremos para cada operación realizada sobre las colecciones Vehicle, Provider, Spent y además las operaciones de Login, Logout y Create_User.

La salida tendría esta estructura:

```

[{"content": "Edition of vehicle successfully", "currentUser": "juanma@gmail.com", "dateTime": "2024-06-01T20:34:17.410Z", "operationLog": "VEHICLE", "type": "INFO", "uid": "KrmRWiE1pMMDsBoSFWJvZ8y4ouN2"}, {"content": "Login successfully", "currentUser": "juanma@gmail.com", "dateTime": "2024-06-02T14:00:07.509Z", "operationLog": "LOGIN", "type": "INFO", "uid": "KrmRWiE1pMMDsBoSFWJvZ8y4ouN2"}]
  
```

The code block shows a JSON array representing log entries. Each entry contains fields for content, current user, date and time, operation log, type, and uid. The first entry is for editing a vehicle, and the second is for logging in.

Estos datos pueden ser extraídos a formato CSV para ser procesados posteriormente mediante Power BI para generar el dashboard de gestión de la aplicación para los usuarios con rol de administrador.

Formato del archivo .CSV:

uid	content	operationLog	dateTime	currentUser	type
2	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Register user successfully	CREATE_USER	2024-05-02T00:31:00.083Z	juanma@gmail.com	INFO
3	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:32:27.771Z	juanma@gmail.com	INFO
4	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:40:00.321Z	juanma@gmail.com	INFO
5	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:44:10.874Z	juanma@gmail.com	INFO
6	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:44:37.976Z	juanma@gmail.com	INFO
7	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:45:50.095Z	juanma@gmail.com	INFO
8	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:50:12.874Z	juanma@gmail.com	INFO
9	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Creation of vehicle successfully	VEHICLE	2024-05-02T00:50:54.465Z	juanma@gmail.com	INFO
10	KrmRWiE1pMMDsBoSFWJvZ8y4ouN2 Login successfully	LOGIN	2024-05-02T00:51:32.687Z	juanma@gmail.com	INFO

Dashboard en Power BI

Para el dashboard en Power BI, se utiliza el archivo .CSV exportado desde la aplicación web con permisos de administrador. Este archivo CSV es importado en Power BI para ser transformado y generar las tablas necesarias para mostrar el dashboard desarrollado.

Transformaciones realizadas

Sobre el archivo inicial, se realizan diferentes transformaciones para limpiar datos, agruparlos, generar tramos horarios.

Finalmente, se generan las siguientes tablas:

Datos:

- log: Datos originales del CSV sin modificación, no se habilita la carga.
- f_logs: Tabla de hechos

Dimensiones:

- d_Aux Usuario: Datos de usuarios
 - uid
 - Correo electrónico
- d_Aux Tipo: Tipo de línea de log
 - id_type
 - Tipo
- d_Aux Operación: Tipo de operación
 - id_operation
 - Operación
- d_Aux Contenido: Desglose del contenido del log
 - id_content
 - Contenido
 - id_typeOperation
 - id_typeLog
 - Correctos
- d_Aux TipoLog: Tipo de log
 - id_typeLog
 - TipoLog
- d_Aux TipoOperación
 - id_typeOperation
 - Tipo operación

Órden:

- d_Orden tramo horario: Tabla auxiliar para realizar ordenado personalizado en el tramo horario.
 - Tramo horario
 - Orden tramo horario

Medidas:

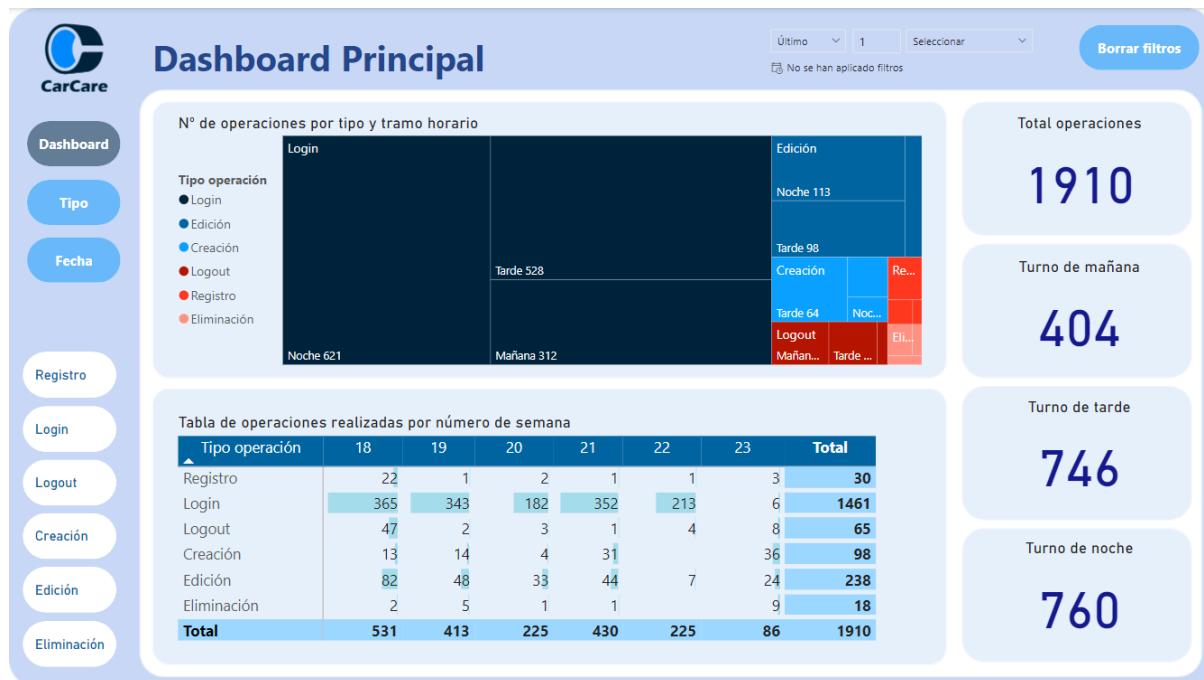
- Medidas: Registro de medidas calculadas.

A parte, se genera también la tabla **Dim Calendar** mediante un script en DAX.

Descripción del dashboard

El dashboard desarrollado dispone de tres pantallas de visualización de datos. Todas las pantallas disponen de elementos de segmentación comunes y sincronizados. Dichos elementos de segmentación son por fecha y por tipo de operación. También se han incluido botones de navegación entre pantallas y botón de limpiar segmentaciones.

Dashboard principal

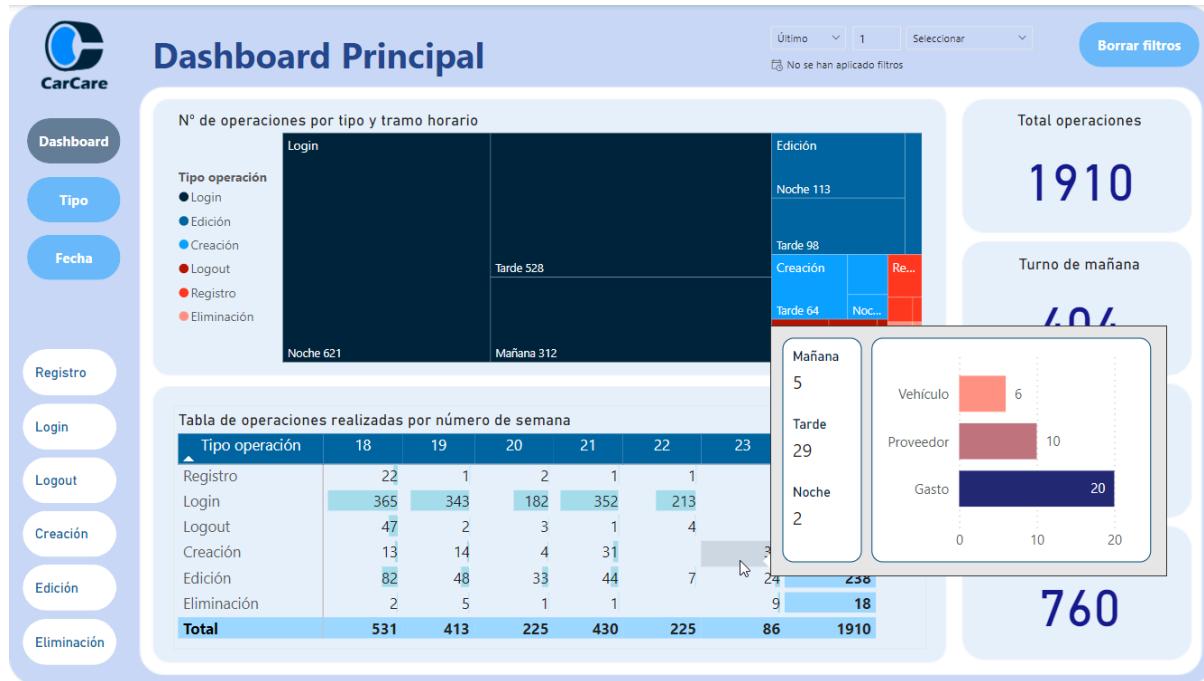


Dispone de los siguientes datos:

- Gráficas:
 - Número de operaciones por tipo y tramo horario en formato Treemap.
 - Tabla de operaciones realizadas por número de semana en formato Matriz.
- Tarjetas con medidas calculadas:

- Total de operaciones realizadas.
- Operaciones realizadas en turno de mañana.
- Operaciones realizadas en turno de tarde.
- Operaciones realizadas en turno de noche.

En esta pantalla, en la tabla de operaciones, se ha añadido una hover card que muestra datos de los turnos y categorías en las que se han realizado las operaciones indicadas en cada celda.



Informe por tipo de operación



Dispone de los siguientes datos:

- Gráficas:
 - Número de operaciones por tramo horario y tipo de operación en formato Columnas apiladas.
 - Número de operaciones por tipo de operación en formato Embudo.
 - Porcentaje de operaciones por tipo de operación en formato Gráfico de anillos. (No se incluyen las operaciones de registro, login y logout).
- Tarjetas con medidas calculadas:
 - Número de operaciones de vehículos.
 - Número de operaciones de proveedores.
 - Número de operaciones de gastos.
 - Porcentaje de error en operaciones realizadas sobre la base de datos.

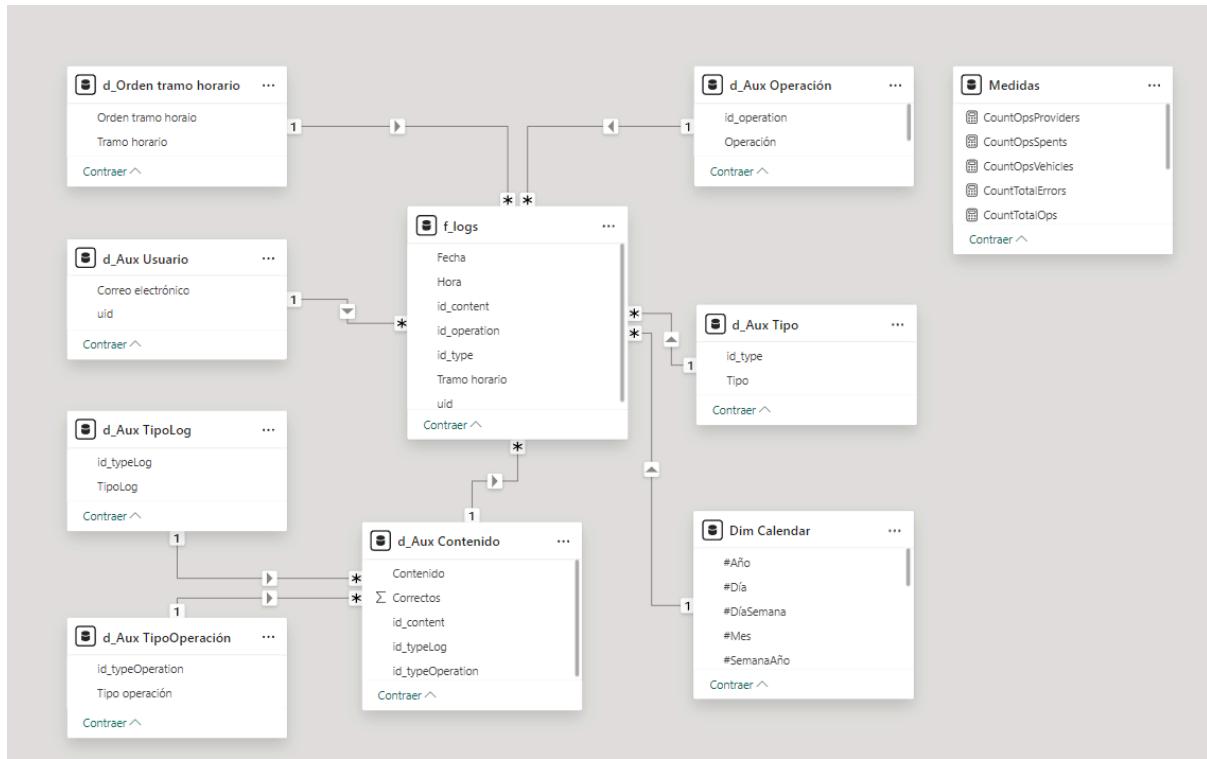
Informe por fecha



Dispone de los siguientes datos:

- Gráficas:
 - Número de operaciones por día de la semana en formato Columnas apiladas.
 - Número de operaciones por fecha.
 - Registro de nuevos usuarios por fecha.
 - Recuento de operaciones por tipo y número de semana en el año.

Relaciones entre las diferentes tablas



Exportación a PDF

Se incluye exportación del dashboard completo en formato PDF: [enlace](#)

Posibles mejoras y escalabilidad

Dentro de las posibilidades de ampliación de la aplicación, estas son algunas posibles mejoras a implementar:

- Gestionar la ubicación de los proveedores para su posterior localización mediante GPS.
- Añadir fecha de próxima revisión de ITV de los vehículos para que se muestre una notificación de aviso cuando se aproxima la fecha.
- Exportación de los gastos en formato PDF.
- Mayor posibilidades de filtrados de datos, como desglose de gastos por categoría, filtrado de vehículos por tipo de vehículo, etc.

Tecnologías utilizadas

Herramientas generales

- [Git](#) y [GitHub](#): Control de versiones y repositorio del proyecto.
- [Firebase](#): Sistema integrado backend de la aplicación.
 - Authentication: Autenticación del usuario.
 - Database: Base de datos no relacional.
 - Storage: Almacenamiento de las imágenes de la aplicación.
- [Figma](#): Diseño de prototipos.
- [Photopea](#): Edición de imágenes.

Aplicación Angular

- [VSCode](#): Entorno de desarrollo integrado (IDE).
- [Angular](#): Framework para el desarrollo de aplicaciones web.
- [Ionic](#): Framework para el desarrollo de aplicaciones móviles híbridas.
- [Capacitor](#): Herramienta para la implementación de funcionalidades nativas.
- [CompoDoc](#): Generación de documentación para Angular.
- [Render](#): Despliegue de aplicaciones y servicios web.
- [Netlify](#): Despliegue de aplicaciones.

Aplicación Android

- [Android Studio](#): Entorno de desarrollo integrado (IDE) para Android.
- [Dokka](#): Generación de documentación para Kotlin.
- [Coil](#): Librería de carga de imágenes para Android.
- [Lotties](#): Animaciones ligeras para tu aplicación.

Aplicación Python

- [VSCode](#): Entorno de desarrollo integrado (IDE).
- [Pythonanywhere](#): Hosting de aplicaciones Python.
- [Swagger](#): Herramienta para documentar y probar APIs.

Videos del proyecto en funcionamiento

- Enlace al video del proyecto final.

[Proyecto TFC 2º DAM CarCare](#)

- Enlace a video realizado para mostrar los avances realizados hasta el checkpoint.

[Checkpoint Proyecto TFC 2º DAM CarCare](#)

Webgrafía

- [YouTube](#)
- [Medium](#)
- [StackOverflow](#)
- [GitHub](#)
- [Angular](#)
- [Ionic Framework](#)
- [Capacitor by Ionic](#)
- [Android Developers](#)
- [Firebase](#)
- [Render](#)
- [Netlify](#)
- [Storyset](#)
- [AppMockUp Studio \(Beta\)](#)
- [Photopea](#)
- [ChatGPT](#)