

Memoria de Proyecto Final
de Desarrollo de Aplicaciones
Multiplataforma

APLICACIÓN

Fo*o*dBye

“El chico de los recados”



Llamas Álvarez, José Antonio
Miranda Alcántara, Víctor Ernesto

ÍNDICE

	<u>Página</u>
1. INTRODUCCIÓN.....	3
2. PLAN DE EMPRESA	4
2.1. Justificación.....	4
2.2. Nombre y logo.....	4
2.3. Producto.....	4
2.4. Modelo de negocio	4
2.5. Consumidores y posibles clientes.	5
2.6. Competencia.	5
2.7. DAFO	5
2.8. Publicidad y promoción	5
3. DESCRIPCIÓN DETALLADA DEL SISTEMA Y LISTADO DE HISTORIAS DE USUARIO.....	6
4. MODELADO	7
4.1. Bocetos, sketching y storyboards de las diferentes aplicaciones (móvil y web).....	7
ANGULAR	7
ANDROID.....	14
4.2. Diagrama de clases del modelo	15
5. DISEÑO	16
5.1. Esquema de clases diseñado para la base de datos.....	16
5.2. Diseño de los servicios web	16
6. IMPLEMENTACIÓN	18
6.1. Descripción de los diferentes paquetes y clases de cada aplicación	18
API	18
ANGULAR	18
ANDROID.....	20

1. INTRODUCCIÓN.

FoodBye es un proyecto que trata de dar una solución informática a una empresa que se dedique a realizar el reparto de los pedidos de distintos establecimientos desde el local hasta los clientes.

La aplicación consta de un api desarrollada en NodeJs+express y Mongoose. El api se encuentra desplegada en Heroku.com mientras que la base de datos está alojada en Mongo Atlas.

Hay una aplicación web desarrollada con Angular que sirve de panel de administración a la que sólo tendrán acceso los usuarios administradores. Desde ella podrán realizar la gestión de usuarios y dar de alta y gestionar los pedidos.

También se ha implementado una app Android a la que tendrán acceso todos los repartidores (bikers) para poder consultar qué pedidos hay disponibles y autoasignárselos para hacerse cargo de ellos. En ella podrá consultar los pedidos que ya hayan realizado.

Los nuevos administradores que se quieran introducir se crearán por medio del panel de Angular mientras que los “bikers” tienen también la posibilidad de registrarse ellos mismo en la app Android, aunque ese registro tendrá que validarlo el administrador del sistema.

2. PLAN DE EMPRESA

4.1. JUSTIFICACIÓN.

¿Por qué este negocio?

El negocio de la venta a domicilio ha sufrido una transformación profunda en los últimos años y cada vez más negocios necesitan de un reparto inmediato, que no puede esperar a las grandes compañías de mensajería.

¿Existe competencia?

Sí. Fundamentalmente Glovo.

¿Por qué motivos creo que el producto se venderá?

Hay multitud de negocios que no tienen un servicio de reparto propio y que se embarcarían en servir a domicilio si se les ofreciera este servicio de manera asequible y profesional.

¿Qué me diferencia de los demás?

Una imagen moderna y un panel que permite tener el control sobre las diferentes etapas del reparto (Emisión del pedido, asignación del repartidor, recogida y entrega).

4.2. NOMBRE y LOGO

Nombre.

FoodBye hace referencia, con un juego de palabras que resulte pegadizo, al movimiento y a la comida.

Logo.



4.3. PRODUCTO.

Ofrecemos a los establecimientos asumir el reparto de sus pedidos. Los pedidos se darán de alta en una aplicación que servirá para que los repartidores puedan saber que hay pedidos disponibles y realizarlos.

4.4. MODELO DE NEGOCIO.

Habrán dos posibilidades:

1. Los establecimientos abonarán un canon para que nuestro sistema se haga cargo de asumir su reparto de pedidos. El canon podrá ser pedido o por tramos (hasta 1000 pedidos X euros, hasta 5000 pedidos X euros, etc.).
2. A cambio de una cuota, el establecimiento contará con una instalación de nuestro sistema para gestionar ellos con sus repartidores.

4.5. CONSUMIDORES Y POSIBLES CLIENTES.

Los consumidores de nuestro producto serán empresas que necesiten un reparto. No necesariamente tiene que ser de comida a domicilio, podría ser un envío a otra empresa o a una oficina de la misma, etc.

4.6. COMPETENCIA.

Fundamentalmente Glovo y JustEat.

4.7. DAFO



4.8. PUBLICIDAD Y PROMOCIÓN

La forma que usaríamos para promover nuestro producto y que se haga conocer, es decir la publicidad, sería hacer promociones en las distintas ciudades de España en las que a una hora del día todos los envíos serían gratuitos, con esto llamaríamos la atención y el nombre de la marca se haría más conocido.

Otra forma de promocionarnos podría ser de manera contraria a la anterior de tal forma que contactaríamos con algunas empresas para ponernos de acuerdo con la promoción que nos beneficiaría a ambos. De este modo solo le cobraríamos el envío, quedando el producto enviado 100% gratuito.

También se usarán las redes sociales como medio para hacernos conocer, podemos hacer que entre nuestros seguidores se sortee 5 envíos gratuitos en sus siguientes compras, de tal forma que nuestra cantidad de usuarios crecería.

El coste que puede llevarnos estos planes de promoción serían muy bajos, teniendo en cuenta que nuestra influencia en el mercado crecería y por tanto nuestras ventas a la par.

3. DESCRIPCIÓN DETALLADA DEL SISTEMA Y LISTADO DE HISTORIAS DE USUARIO

Aplicación en la que desde un panel de gestión se registre un pedido que hay que llevar a cabo y todos los “bikers” que estén registrados puedan ver los pedidos sin asignar. En cuanto tomen un pedido les saldrá en un mapa la ruta a seguir.

Desde una aplicación web hecha en Angular, el usuario administrador podrá acceder a varias secciones:

- Sección de Usuarios: Habrá dos roles (Administrador y usuario repartidor), el usuario administrador podrá crear, editar y eliminar usuarios normales y también crear otros administradores.

Tendrá a su disposición un listado con los usuarios pendientes de validación y otro con los usuarios validados

También podrá validar el registro de los usuarios bikers que se hayan registrado en la app de Android.

De cada usuario se registrará el nombre, los apellidos, el rol (será usuario repartidor de forma predeterminada), el email y la contraseña. La foto de perfil habrá de subirla el propio usuario.

- Sección de pedidos: En ella, un administrador, podrá crear nuevos pedidos, editarlos, eliminarlos y consultar el estado de los mismos.

Habrá un listado de los pedidos “activos” y otro con un histórico de todos los pedidos realizados de modo que se pueda consultar cualquiera de ellos.

De cada pedido se registrará un número de pedido, un título, la descripción, la dirección de destino y un enlace para ver la localización de destino en el mapa. Asimismo, se podrá registrar el usuario repartidor al que se le haya asignado y si está realizado o está pendiente.

Desde una aplicación Android los usuarios repartidores podrán:

- Registrarse con su nombre, los apellidos, el email, la contraseña y la foto de perfil
- Loguearse para acceder con su email y contraseña.
- Tendrán un menú inferior con tres pestañas. La primera con los pedidos asignados a ese usuario para poder consultarlos y marcarlos como realizados cuando así sea. Asimismo, podrán ver en un mapa la ruta hasta el destino del pedido.
- La segunda pestaña mostrará un listado de los pedidos sin repartidor asignado con la opción de poder ver los detalles de un pedido y de poder asignárselo uno mismo.
- La tercera pestaña mostrará los datos de perfil del usuario y permitirá modificarlos.

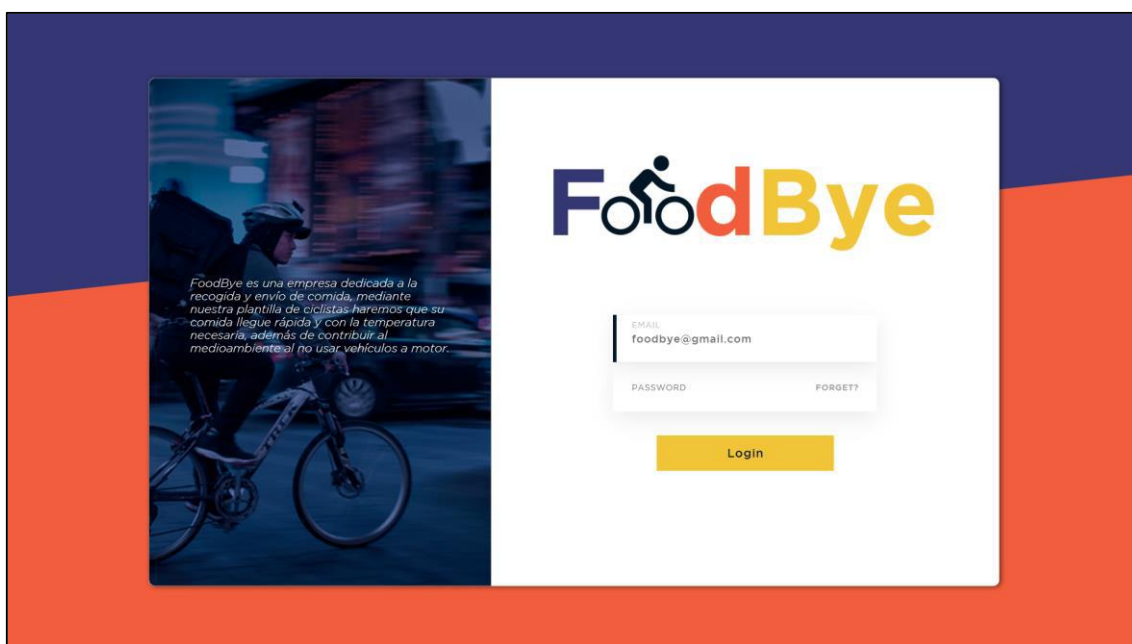
Todo ello funcionará por debajo con una Api REST hecha con Nodejs, Express, Mongoose y MongoDB.

4. MODELADO.

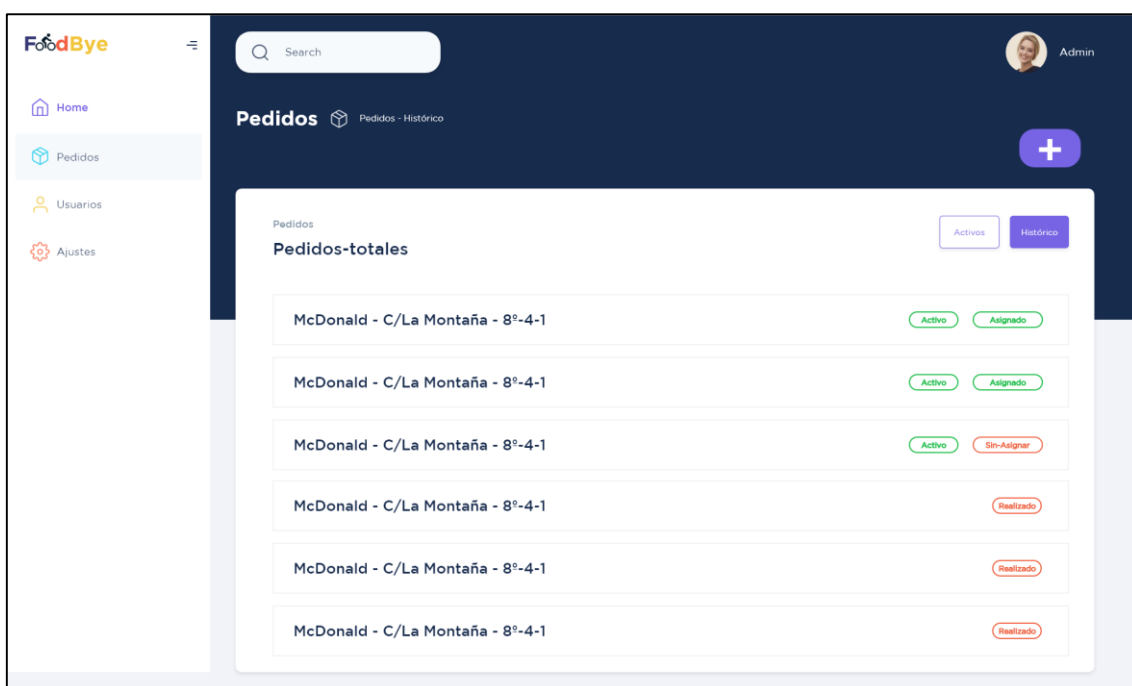
4.1. BOCETOS, SKETCHING Y STORYBOARDS DE LAS DIFERENTES APLICACIONES (MÓVIL Y WEB).

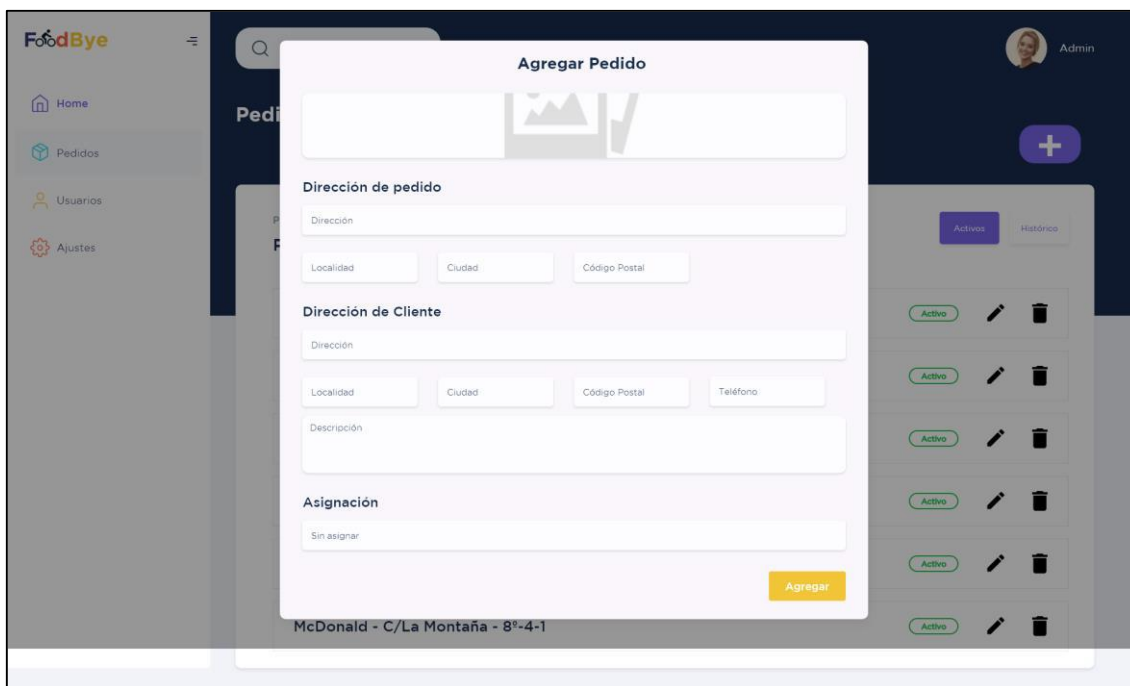
Sketching Aplicación Web Angular.

LOGIN




PEDIDOS





Agregar Pedido



Dirección de pedido

Dirección

Localidad Ciudad Código Postal

Dirección de Cliente

Dirección

Localidad Ciudad Código Postal Teléfono

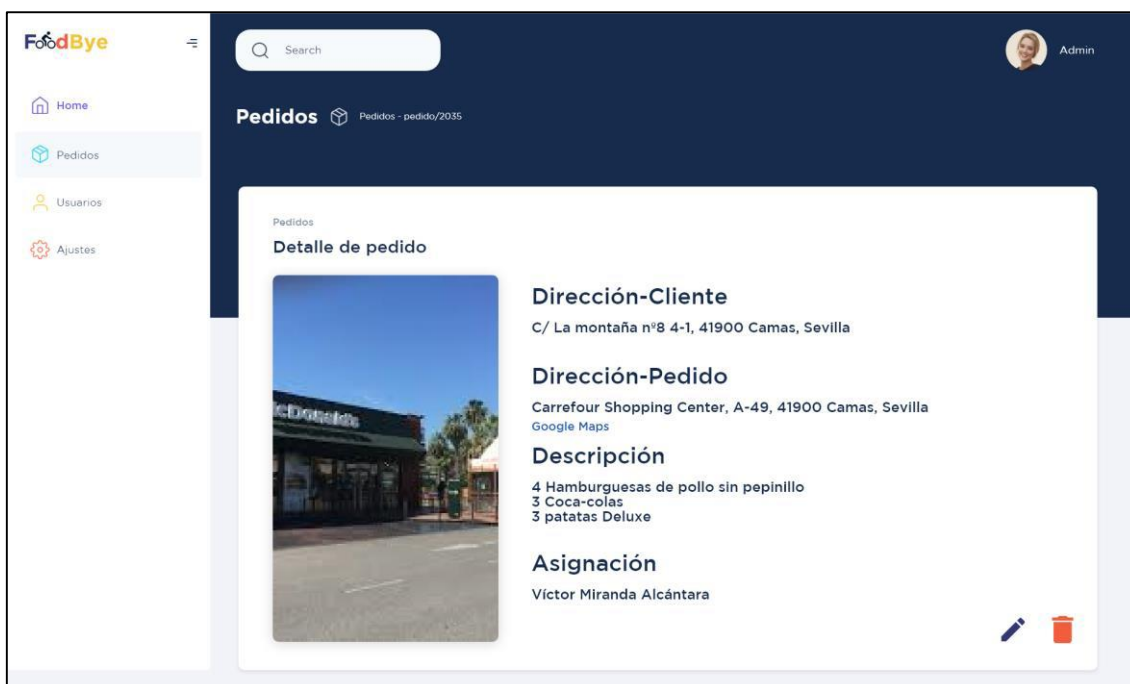
Descripción

Asignación

Sin asignar

Agregar

McDonald - C/La Montaña - 8°-4-1




FoodBye

Search

Pedidos Pedidos - pedido/2035

Detalle de pedido

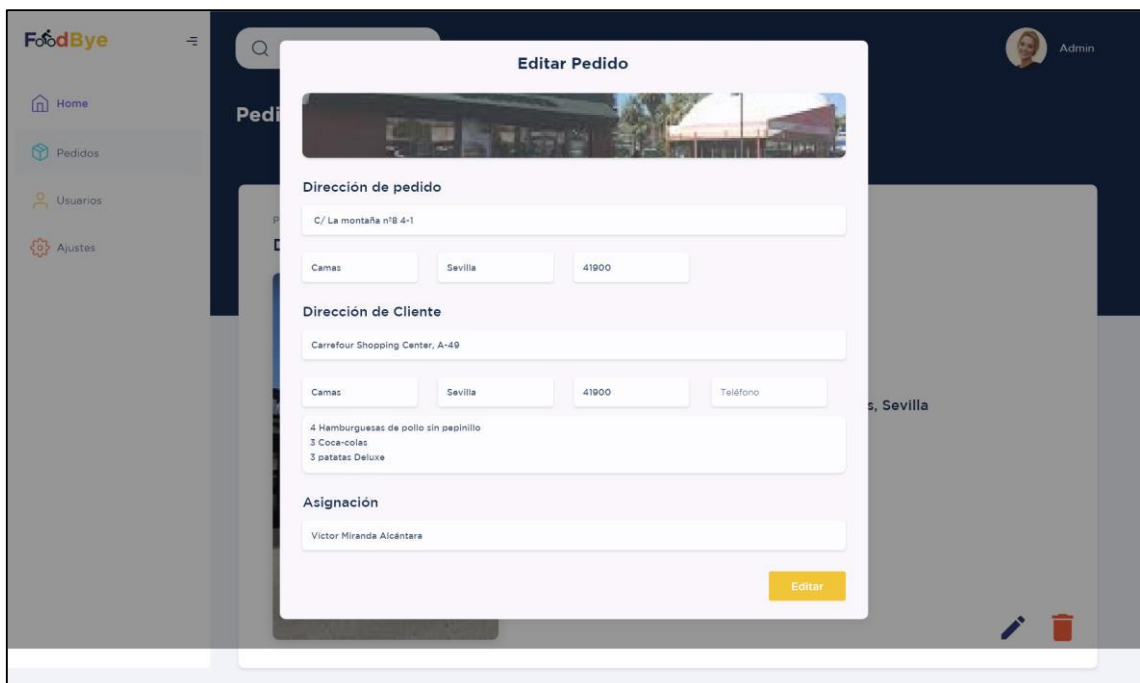
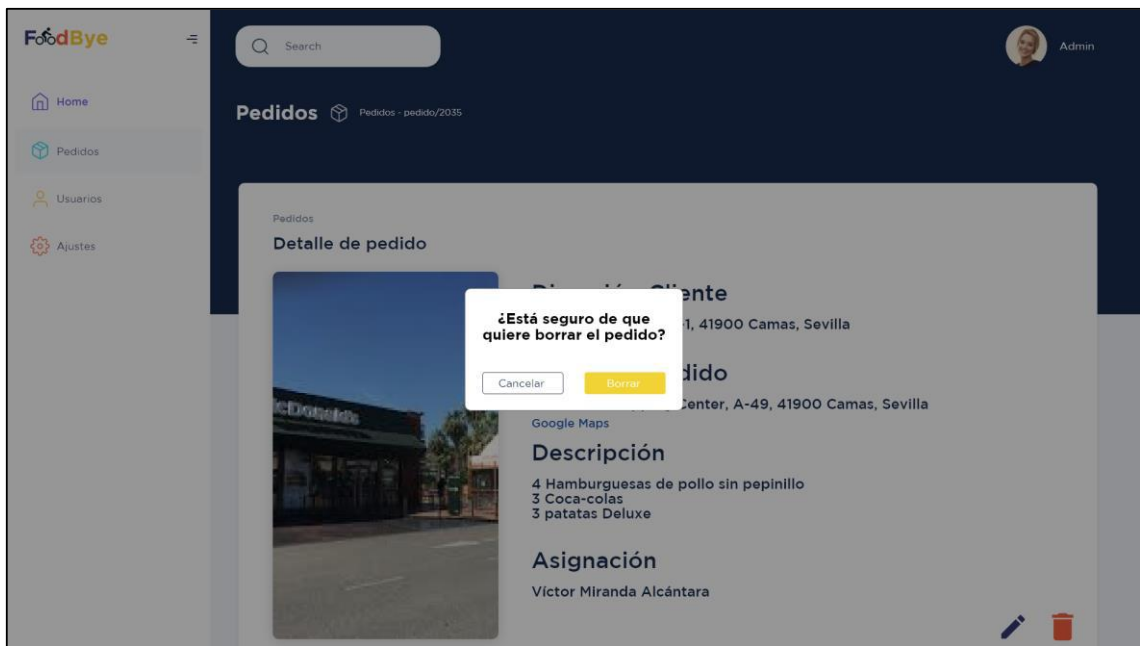


Dirección-Cliente
C/ La montaña nº8 4-1, 41900 Camas, Sevilla

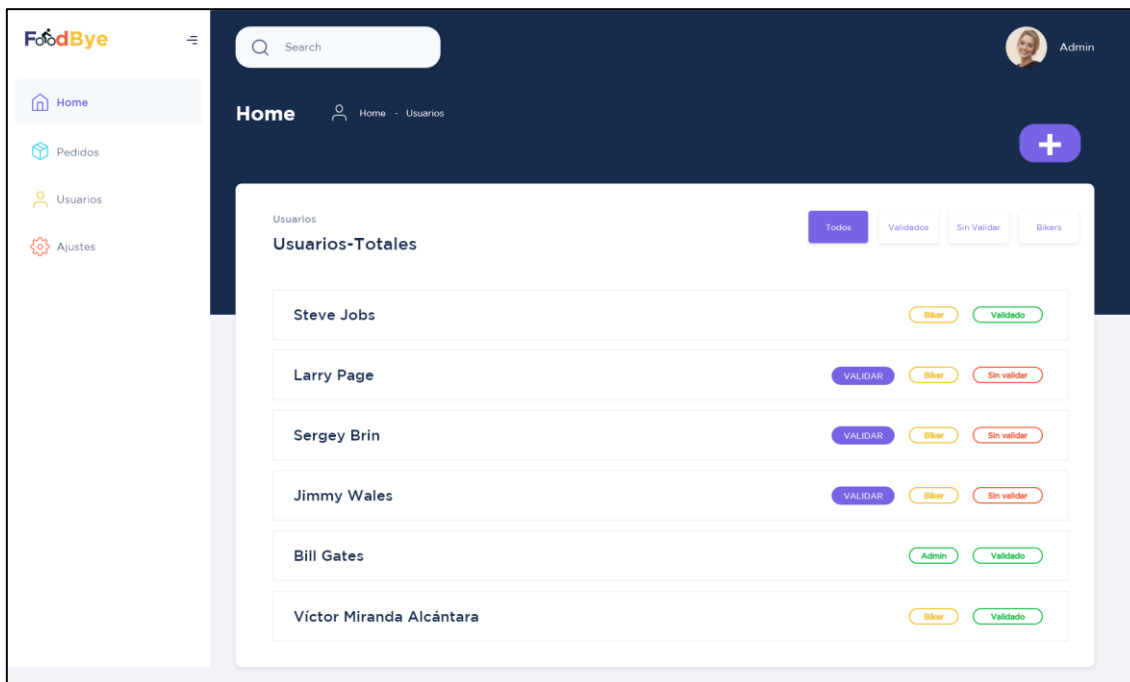
Dirección-Pedido
Carrefour Shopping Center, A-49, 41900 Camas, Sevilla
[Google Maps](#)

Descripción
4 Hamburguesas de pollo sin pepinillo
3 Coca-colas
3 patatas Deluxe

Asignación
Víctor Miranda Alcántara



USUARIOS



FoodBye

Search

Home

Pedidos

Usuarios

Ajustes

Home

Home - Usuarios

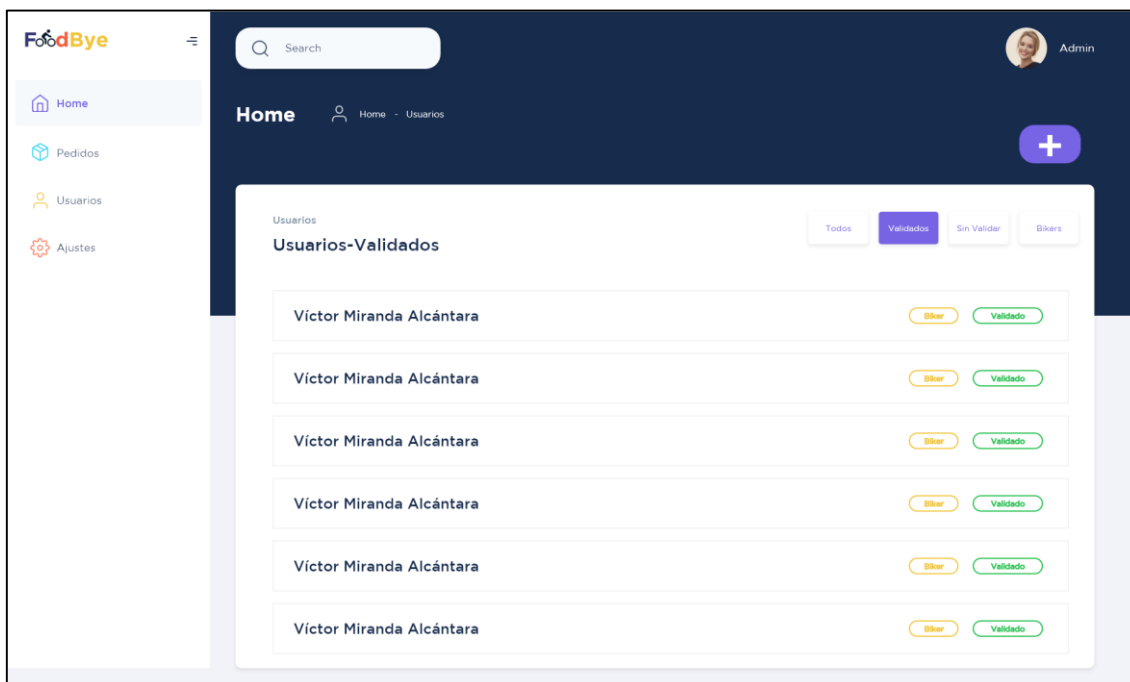
Admin

Usuarios

Usuarios-Totales

Todos Validados Sin Validar Bikers

Steve Jobs	Biker	Validado
Larry Page	VALIDAR	Biker Sin validar
Sergey Brin	VALIDAR	Biker Sin validar
Jimmy Wales	VALIDAR	Biker Sin validar
Bill Gates	Admin	Validado
Víctor Miranda Alcántara	Biker	Validado



FoodBye

Search

Home

Pedidos

Usuarios

Ajustes

Home

Home - Usuarios

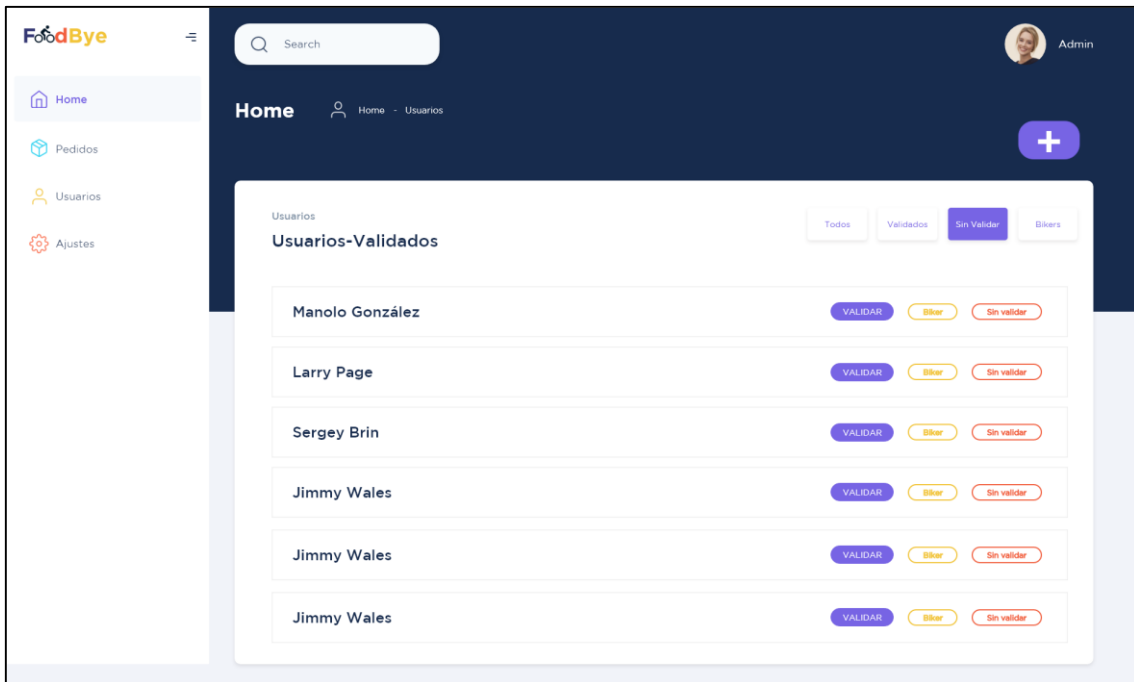
Admin

Usuarios

Usuarios-Validados

Todos Validados Sin Validar Bikers

Víctor Miranda Alcántara	Biker	Validado
Víctor Miranda Alcántara	Biker	Validado
Víctor Miranda Alcántara	Biker	Validado
Víctor Miranda Alcántara	Biker	Validado
Víctor Miranda Alcántara	Biker	Validado
Víctor Miranda Alcántara	Biker	Validado



FoodBye Search Admin

Home Pedidos Usuarios Ajustes

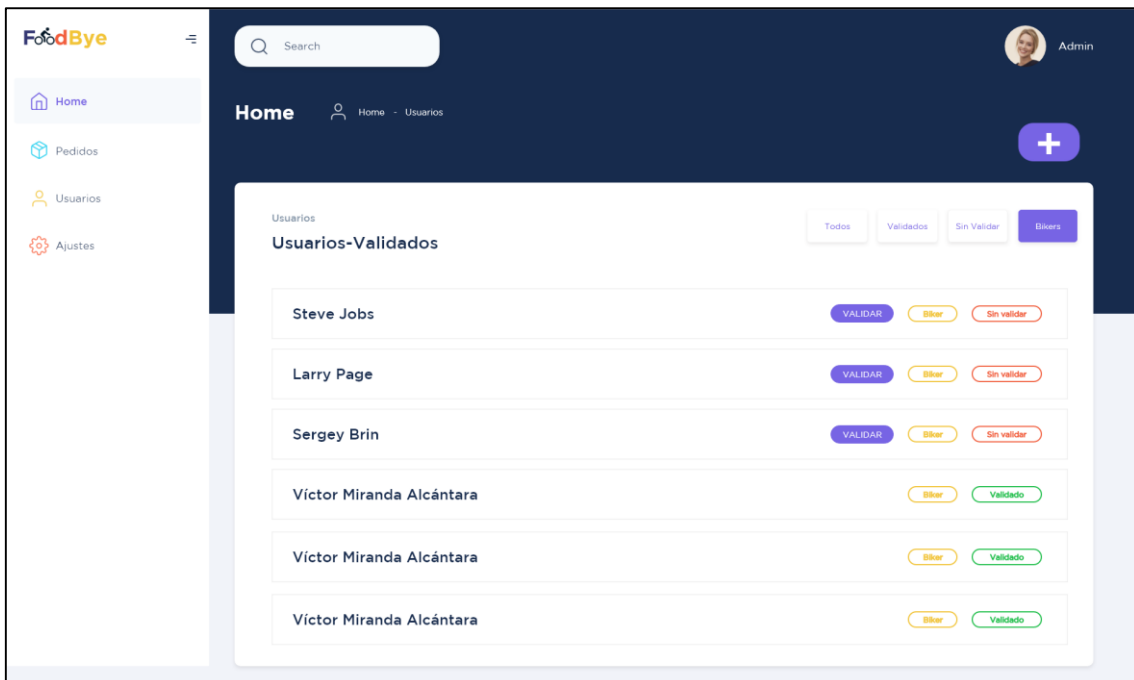
Home - Usuarios

Usuarios

Todos Validados Sin Validar Bikers

Usuarios-Validados

Manolo González	VALIDAR	Biker	Sin validar
Larry Page	VALIDAR	Biker	Sin validar
Sergey Brin	VALIDAR	Biker	Sin validar
Jimmy Wales	VALIDAR	Biker	Sin validar
Jimmy Wales	VALIDAR	Biker	Sin validar
Jimmy Wales	VALIDAR	Biker	Sin validar



FoodBye Search Admin

Home Pedidos Usuarios Ajustes

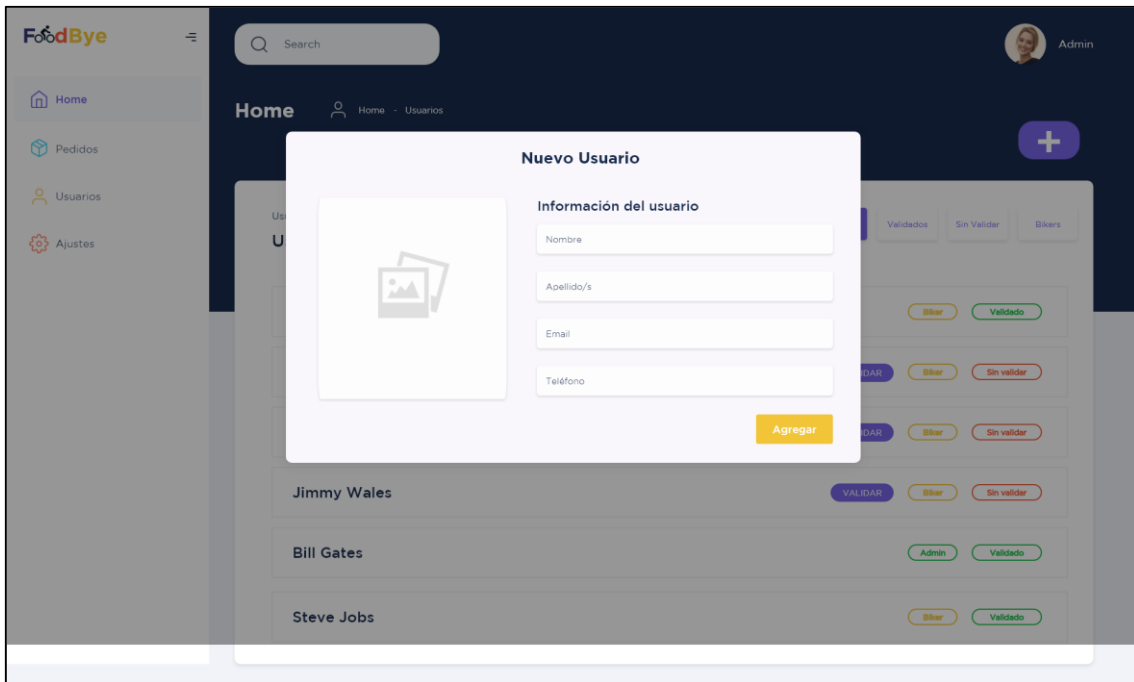
Home - Usuarios

Usuarios

Todos Validados Sin Validar Bikers

Usuarios-Validados

Steve Jobs	VALIDAR	Biker	Sin validar
Larry Page	VALIDAR	Biker	Sin validar
Sergey Brin	VALIDAR	Biker	Sin validar
Víctor Miranda Alcántara	Biker	Validado	
Víctor Miranda Alcántara	Biker	Validado	
Víctor Miranda Alcántara	Biker	Validado	



Nuevo Usuario

Información del usuario


Nombre

Apellido/s

Email

Teléfono

Agregar



Usuarios

Usuarios - usuario/323

Detalle de usuario

Steven Paul Jobs

Email

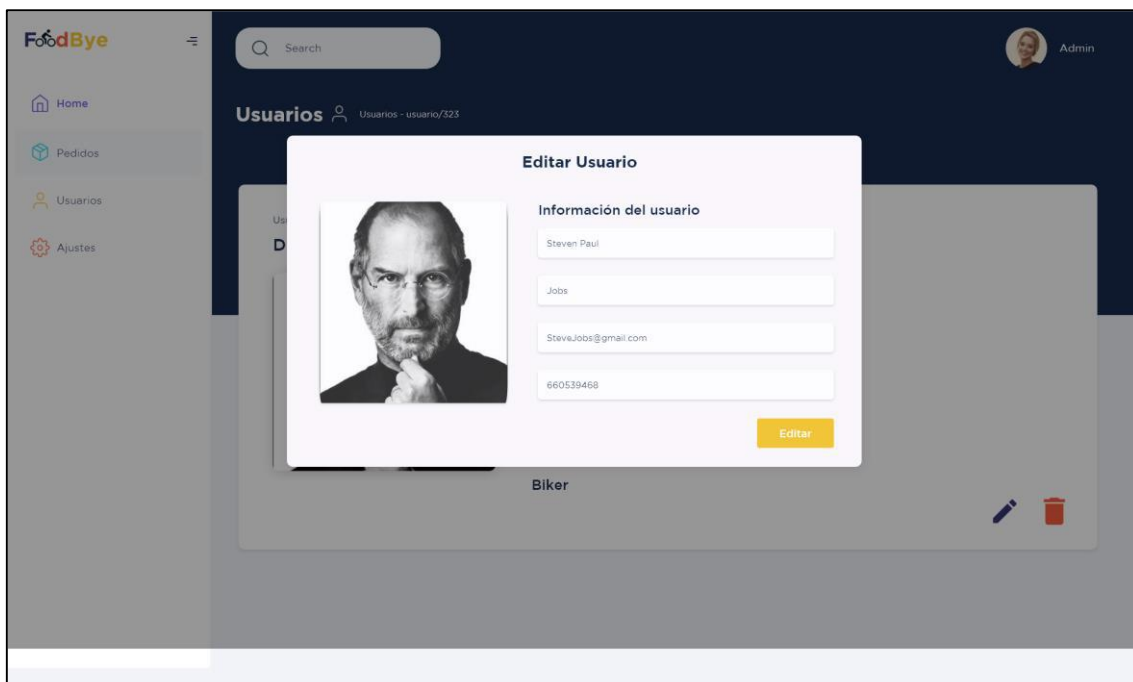
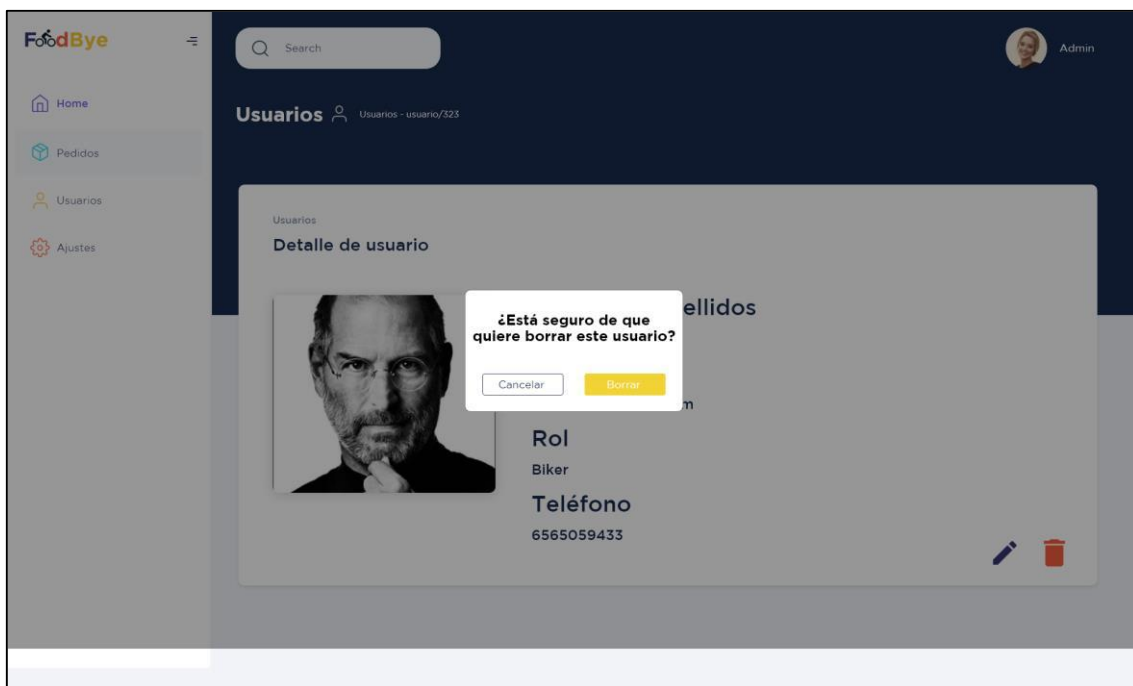
SteveJobs@gmail.com

Rol

Biker

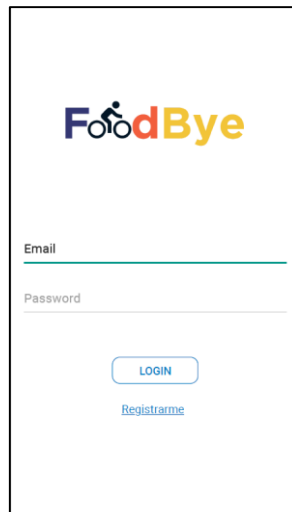
Teléfono

6565059433



Sketching Aplicación Android.

LOGIN



FoodBye

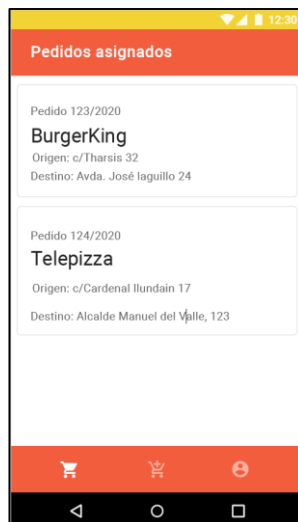
Email

Password

LOGIN

[Regístrate](#)

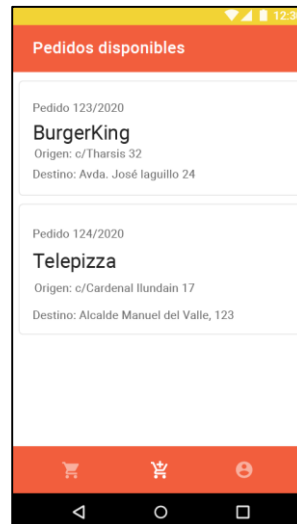
PEDIDOS



Pedidos asignados

Pedido 123/2020
BurgerKing
Origen: c/Tharsis 32
Destino: Avda. José Laguillo 24

Pedido 124/2020
Telepizza
Origen: c/Cardenal Ilundain 17
Destino: Alcalde Manuel del Valle, 123

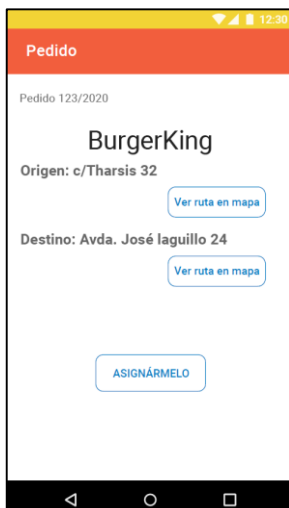


Pedidos disponibles

Pedido 123/2020
BurgerKing
Origen: c/Tharsis 32
Destino: Avda. José Laguillo 24

Pedido 124/2020
Telepizza
Origen: c/Cardenal Ilundain 17
Destino: Alcalde Manuel del Valle, 123

DETALLE DE PEDIDO



Pedido

Pedido 123/2020

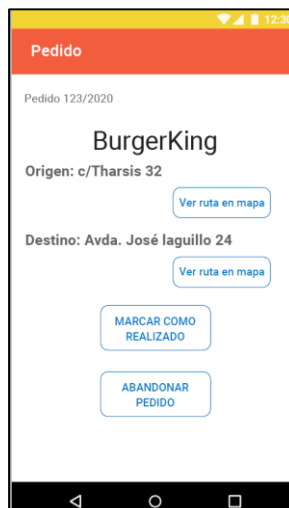
BurgerKing
Origen: c/Tharsis 32

[Ver ruta en mapa](#)

Destino: Avda. José Laguillo 24

[Ver ruta en mapa](#)

[ASIGNÁRMELO](#)



Pedido

Pedido 123/2020

BurgerKing
Origen: c/Tharsis 32

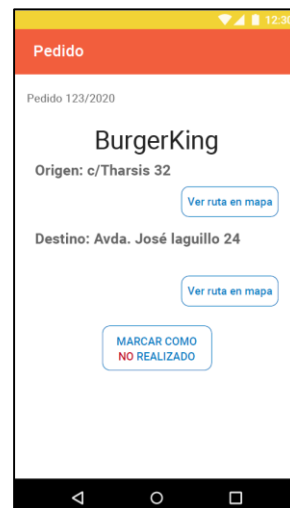
[Ver ruta en mapa](#)

Destino: Avda. José Laguillo 24

[Ver ruta en mapa](#)

[MARCAR COMO REALIZADO](#)

[ABANDONAR PEDIDO](#)



Pedido

Pedido 123/2020

BurgerKing
Origen: c/Tharsis 32

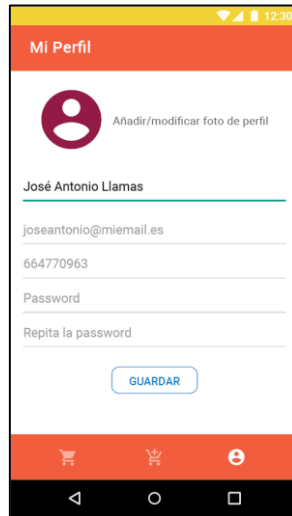
[Ver ruta en mapa](#)

Destino: Avda. José Laguillo 24


[Ver ruta en mapa](#)

[MARCAR COMO NO REALIZADO](#)

REGISTRO / EDITAR PERFIL



Mi Perfil

 Añadir/modificar foto de perfil

José Antonio Llamas

joseantonio@miemail.es

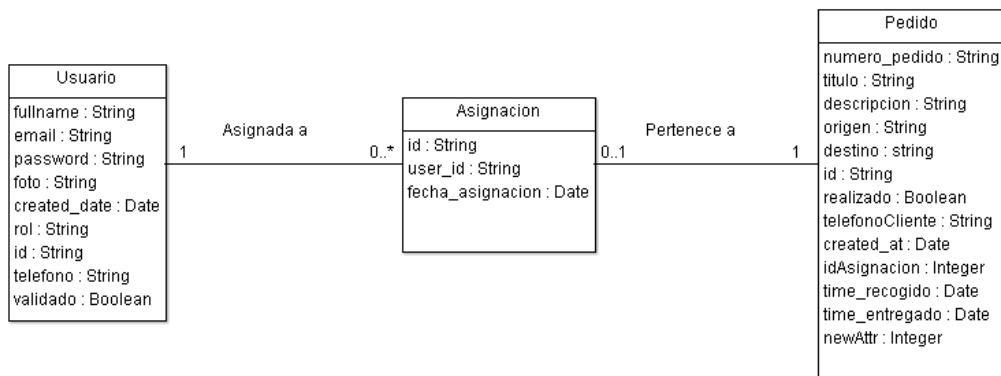
664770963

Password

Repita la password

GUARDAR

4.2. DIAGRAMA DE CLASES DEL MODELO.



5. DISEÑO.

5.1. ESQUEMA DE CLASES DISEÑADO PARA LA BASE DE DATOS.

La base de datos consta de una colección PEDIDOS que incluye toda la información de los mismos (Incluido un objeto de la clase Asignación, si es el caso), una colección USUARIOS con toda la información de los mismos y una Secuencia para asignar de manera automática los números de pedido.

5.2. DISEÑO DE LOS SERVICIOS WEB.

post('/api/login')

Petición para loguearse en el sistema.

post('/api/register')

Petición para registrarse en el sistema. Necesitará la validación del administrador.

get('/api/users')

Petición para obtener una lista de todos los usuarios.

get('/api/users/unvalidated')

Petición para obtener una lista de todos los usuarios sin validar.

get('/api/users/validated')

Petición para obtener una lista de todos los usuarios validados.

get('/api/users/bikers')

Petición para obtener una lista de todos los usuarios con rol BIKER.

put('/api/user/password/:id')

Petición para cambiar la contraseña.

get('/api/avatar/:id')

Petición para obtener el avatar de un usuario.

get('/api/user/:id')

Petición para obtener un usuario.

put('/api/user/:id')

Petición para editar un usuario.

put('/api/users/validar/:id')

Petición para validar un usuario.

put('/api /users/inhabilitar/:id')

Petición para inhabilitar un usuario validado.

delete('/api /users/:id')

Petición para eliminar un usuario.

put('/api /avatar/:id')

Petición para modificar la foto de perfil de un usuario.

post('/pedido/nuevo')

Petición para añadir un nuevo pedido.

get('/pedido /todos')

Petición para obtener todos los pedidos.

get('/pedido /sinasignar')

Petición para obtener una lista de pedidos sin asignar a ningún biker.

get('/pedido /usuario/:id')

Petición para obtener los pedidos asignados a un biker concreto.

get('/pedido /:id')

Petición para obtener un pedido.

put('/pedido /:id')

Petición para modificar un pedido.

delete('/pedido /:id')

Petición para eliminar un pedido.

put('/pedido /asignar/:id')

Petición para asignar un pedido a un biker.

put('/pedido /recogido/:id')

Petición para registrar la hora de recogida de un pedido.

put('/pedido /entregado/:id')

Petición para registrar la hora de entrega de un pedido y establecerlo como realizado.

put('/pedido /abandonar/:id')

Petición para desasignar un pedido y liberarlo.

6. IMPLEMENTACIÓN.

6.1. DESCRIPCIÓN DE LOS DIFERENTES PAQUETES Y CLASES DE CADA APLICACIÓN.

API EN NODEJS:

Paquete controllers:

Incluye las clases con los controladores para las peticiones.

- error_types.js: Definición de las respuestas de error del api.
- pedido.js: Definición de las peticiones que realizan operaciones sobre los pedidos.
- user.js: Definición de las peticiones que realizan operaciones sobre los usuarios.

Paquete middleware:

Incluye las clases con los middleware que se usarán en las rutas de las peticiones.

- index.js: Definición de los middleware necesarios.

Paquete models:

Incluye las clases con la definición de los modelos que vamos a usar en el api.

- pedido.js: Definición del modelo de pedido.
- user.js: Definición del modelo de usuario.
- sequence.js: Definición de la secuencia que usaremos para asignar el número de pedido.

Paquete routes:

Incluye las rutas para realizar las peticiones a la api.

- pedido.js: Definición de las rutas para usar los métodos en los controladores de pedidos, indicando si debe usar algún middleware.
- user.js: Definición de las rutas para usar los métodos en los controladores de usuarios, indicando si debe usar algún middleware.

APLICACIÓN ANGULAR:

Paquete dto:

Incluye los DTO de los modelos que necesitamos en las diferentes funcionalidades que hemos implementado. En nuestro caso:

- edit-dto.ts: Dto para editar un usuario.
- login-dto.ts: Dto para el login.
- pedido-dto.ts: Dto para de pedido para diversas funcionalidades.
- pedido-edit-dto.ts: Dto para editar un pedido.
- pedido-nuevo-dto.ts: Dto para crear un pedido.
- register-dto.ts: Dto para crear un usuario.
- usuario-dto.ts: Dto para mostrar un usuario.
- usuario-edit-password-dto.ts: Dto para editar la contraseña de un usuario.

Paquete guards:

Incluye los métodos para hacer comprobaciones en las rutas:

- auth.guard.ts: Contiene el método que redirige al login si no hay una sesión iniciada.

Paquete models:

Incluye los modelos de los objetos que usamos:

- avatar.interface.ts
- login-response.interface.ts
- pedido-nuevo-response.interface.ts
- pedido.interface.ts
- pedidoasignacion.interface.ts
- register-response.interface.ts
- usuario-edit-response.interface.ts
- usuario.interface.ts

Paquete service:

Incluye los servicios que contienen los métodos con las peticiones a la api:

- auth.service.ts: Peticiones para la autenticación.
- config.service.ts: Configuración de los recursos.
- pedidos.service.ts: Peticiones para manejar los pedidos.
- upload.service.ts: Peticiones para el registro de usuarios.
- usuarios.service.ts: Peticiones para el manejo de los usuarios.

Paquete session:

Incluye los componentes para el login y el registro:

- signin: Login de los usuarios.
- signup: Registro de los usuarios.

PAQUETE DASHBOARD:

En el incluimos todos los componentes principales de la aplicación.

- Detalle-pedido: Detalle de un pedido.
- Detalle-usuario: Detalle de un usuario.
- Dialog-add-pedido: Diálogo para crear un nuevo pedido.
- Dialog-add-usuario: Diálogo para crear un nuevo usuario.
- Dialog-borrar-pedido: Diálogo para borrar un pedido.
- Dialog-borrar-usuario: Diálogo para borrar un usuario.
- Dialog-editar pedido: Diálogo para editar un pedido.
- Dialog-editar-usuario: Diálogo para editar un usuario.
- Lista-pedidos: Muestra un listado de todos los pedidos.
- Lista-pedidos-usuario: Muestra un listado de los pedidos de un usuario.
- Lista-usuarios: Muestra los listados de todos los usuarios, sólo los bikers, sólo los activados, sólo los pendientes de activar.
- Snack-bar-usuario-agregado: Aviso de usuario creado.

- Snack-bar-usuario-borrado: Aviso de usuario eliminado.
- Snack-bar-usuario-editado: Aviso de usuario editado.

También se incluyen los dos ficheros principales de configuración:

- Dashboard.module.ts: Incluye los imports de los módulos usados, las declaraciones de los componentes y la declaración de los componentes usados como entryComponents.
- Dashboard.routing.ts: Incluye la definición de las rutas de angular necesarias para navegar entre componentes.

APLICACIÓN ANDROID:

Paquete common:

En este paquete incluimos todos los recursos que vamos a usar a la hora de programar.

- Constantes: Clase en la que se definen strings que no van a variar.
- MyApp: Clase que nos sirve para llamar al contexto si no nos encontramos en un activity o en fragmentos.
- SharedPreferencesManager: Clase que usamos para rescatar el token del usuario que ha iniciado sesión en la aplicación.

Paquete data:

En este paquete incluimos dos paquetes que a su vez incluyen los repositorios y los viewModels.

- Pedido: paquete que incluye el repositorio de Pedido y sus viewModels
 - PedidoRepository: Métodos para llamar a las peticiones del api.
 - PedidoDetallesViewModel: Incluye los métodos que llaman al repositorio de pedido para obtener las respuestas de las peticiones de un pedido.
 - PedidoViewModel: Incluye los métodos que llaman al repositorio de pedido para obtener las respuestas de las peticiones relacionadas con las listas.
- User: Paquete que incluye el repositorio de Usuario y su ViewModel.
 - UserRepository: Clase en la que están los métodos para llamar a las peticiones del API.
 - UserViewModel: Incluye todos los métodos que llaman al repositorio de usuario para obtener las respuestas de las peticiones.

Paquete models:

Incluye los modelos de los objetos que usamos:

- Request
 - RequestAsignarPedido: Modelo que necesita el cuerpo de la petición para asignar un pedido.
 - RequestEditPassword: Modelo que necesita el cuerpo de la petición para editar la contraseña de un usuario.

- RequestEditUser: Modelo que necesita el cuerpo de la petición para editar un usuario.
- RequestLogin: Modelo que necesita el cuerpo de la petición para loguearte en la aplicación.
- Response
 - Asignacion: Modelo de la respuesta de la petición asignar pedido.
 - PedidoResponse: Modelo de la respuesta de las peticiones de pedido.
 - ResponseLogin: Modelo de la respuesta de la petición de iniciar sesión.
 - UserResponse: Modelo de la respuesta de las peticiones de usuario.

Paquete retrofit:

En este paquete incluimos los archivos relacionados en ponerse en contacto con el API.

- IService: Interfaz en las que se define las rutas de las peticiones, sus requisitos y sus repuestas.
- ServiceGenerator: Clase encargada de modificar las peticiones de manera que se puedan en poner en contacto con el api.

Paquete ui:

Paquete en los que se definen los fragmentos de la página principal de la aplicación.

- MisPedidos: Fragmento que pinta la lista de pedidos del usuario iniciado.
- Pedidos: Fragmento que pinta la lista de todos los pedidos sin asignar.
- Profile
 - EditUserFragment: Fragmento que pinta los datos del usuario iniciado.
 - EditPasswordActivity: Activity que pinta el formulario para editar la contraseña.
 - EditAvatarActivity: Activity que pinta el formulario para editar el avatar.

DetalleActivity: Activity que pinta el detalle de un pedido.

LoginActivity: Activity que pinta el formulario de login.

MainActivity Activity principal en el que se pintan los 3 fragmentos del paquete ui.

MapRutaActivity Activity en el que se pinta un mapa donde se ve la ruta que debe hacer el usuario.

RegisterActivity Activity en el que se pinta el formulario de registro.