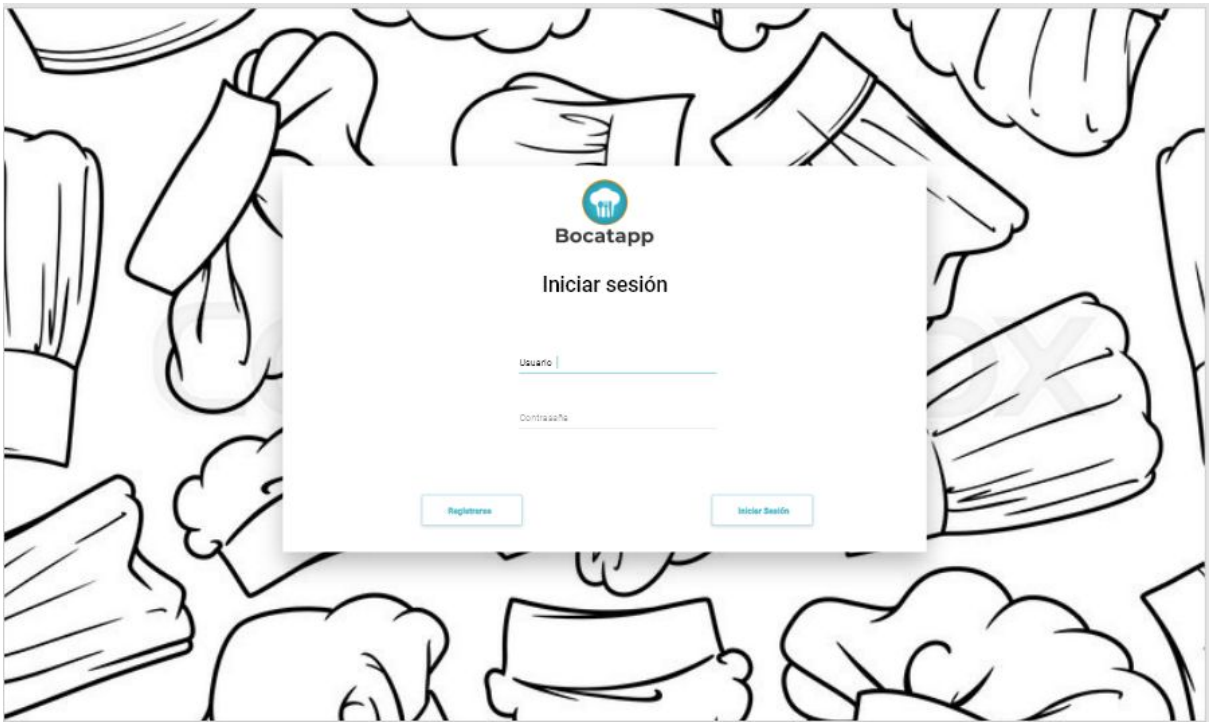
Bocatapp

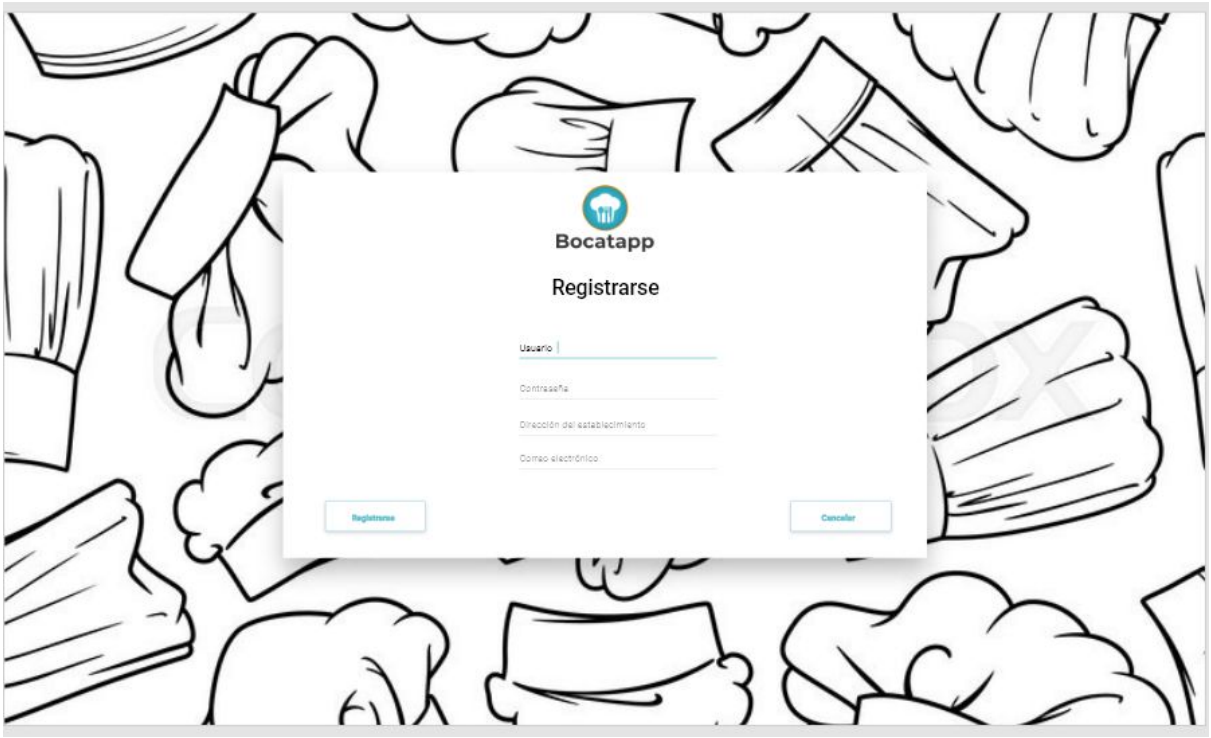
Alejandro Díaz Santos, Pablo Rodríguez Roldán.

2° DAM (Salesianos de Triana, Sevilla).

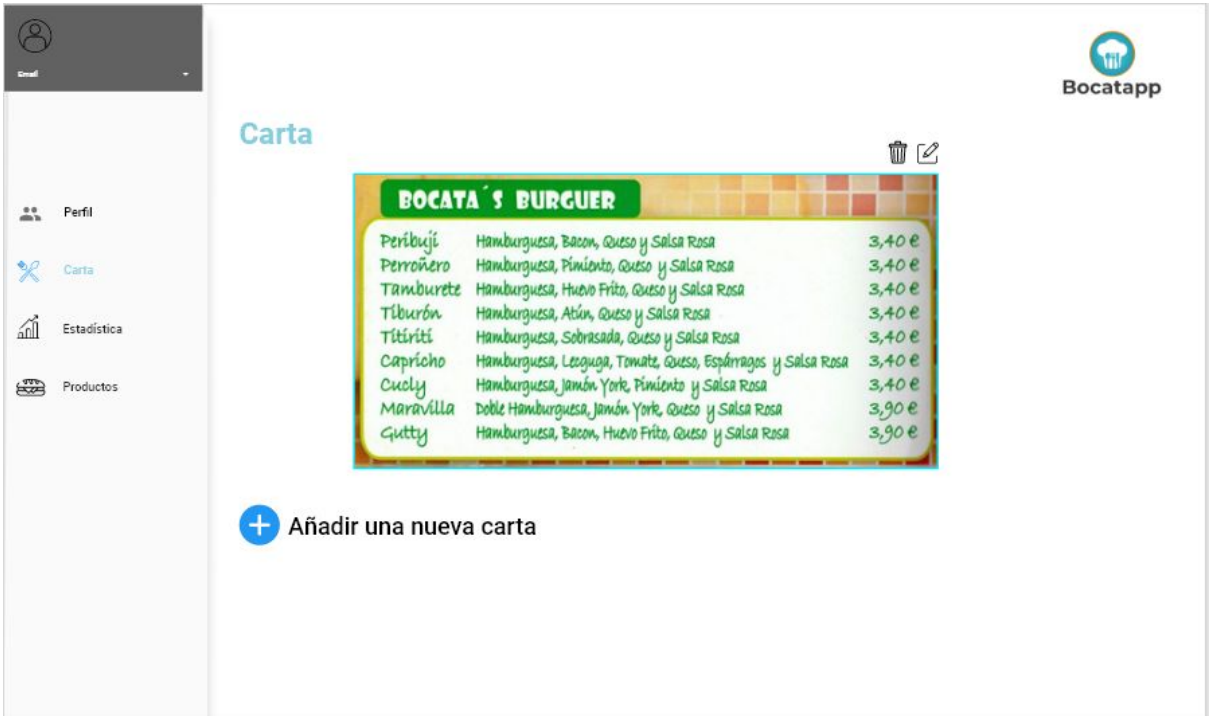
MODELADO(SKETCHING DE LA APLICACIÓN WEB Y DE LA APLICACIÓN DE ANDROID)

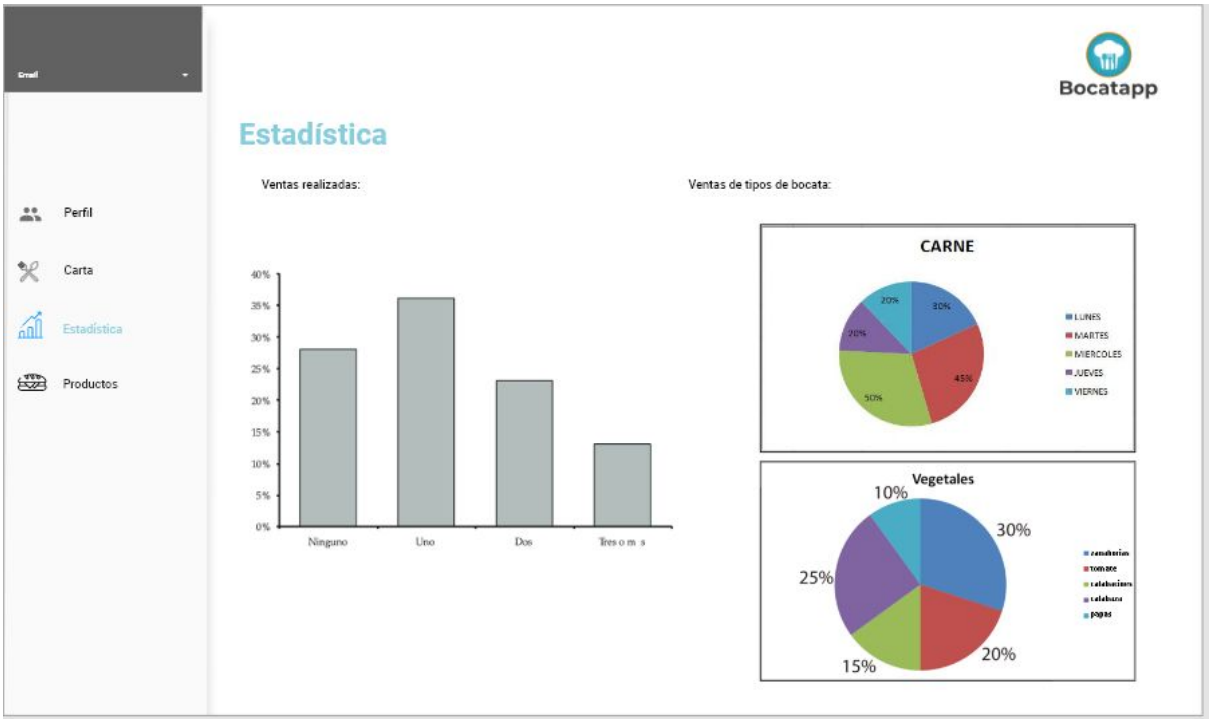
SKETCHING WEB

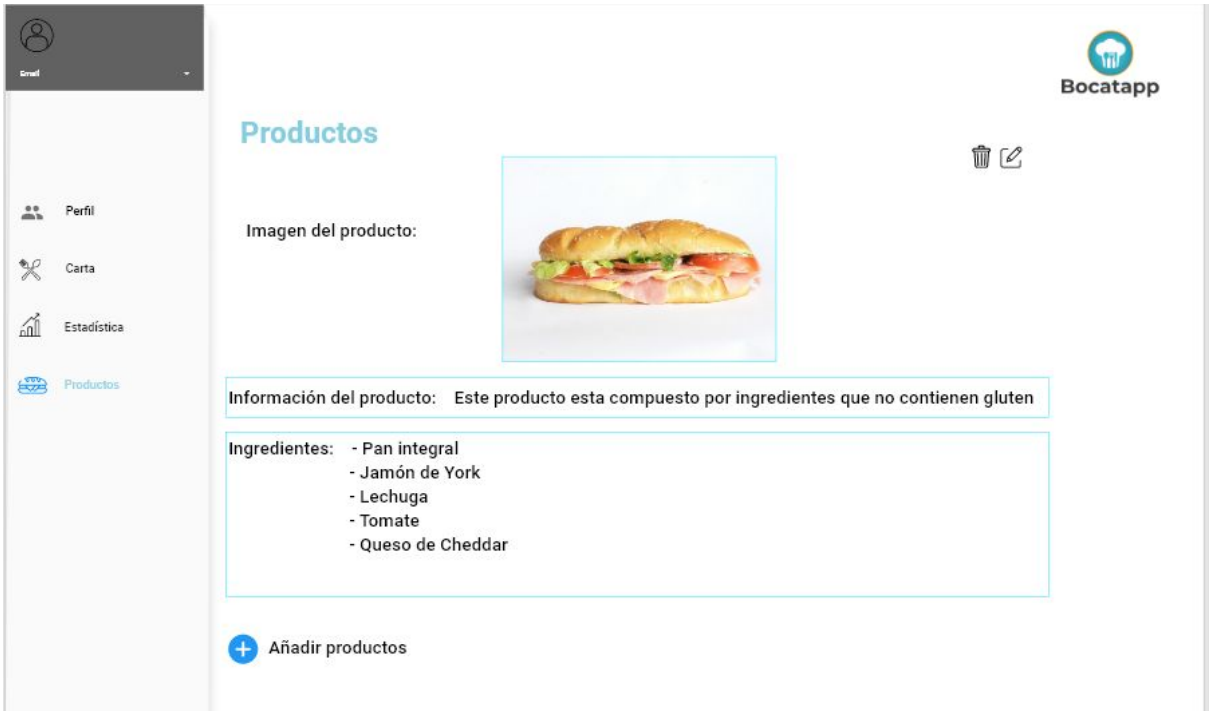

Bocatapp
Iniciar sesión
Usuario
Contraseña
Registrarse
Iniciar Sesión


Bocatapp
Registrarse
Usuario
Contraseña
Dirección del establecimiento
Correo electrónico
Registrarse
Cancelar


Email
Perfil
Carta
Estadística
Productos
Bocatapp
Perfil
PORTADA DEL ESTABLECIMIENTO:
UBICACIÓN: Calle Condes de Bustillo número 21
HORARIOS: Lunes a Viernes de 11:00 PM a 22:30 PM


Email
Perfil
Carta
Estadística
Productos
Bocatapp
Carta
BOCATA´S BURGUER
Peribuji Hamburguesa, Bacon, Queso y Salsa Rosa 3,40 €
Perroñero Hamburguesa, Pimiento, Queso y Salsa Rosa 3,40 €
Tamburete Hamburguesa, Huevo Frito, Queso y Salsa Rosa 3,40 €
Tiburón Hamburguesa, Atún, Queso y Salsa Rosa 3,40 €
Titiriti Hamburguesa, Sobrasada, Queso y Salsa Rosa 3,40 €
Capricho Hamburguesa, Lecguga, Tomate, Queso, Espárragos y Salsa Rosa 3,40 €
Cucly Hamburguesa, Jamón York, Pimiento y Salsa Rosa 3,40 €
Maravilla Doble Hamburguesa, Jamón York, Queso y Salsa Rosa 3,90 €
Gutty Hamburguesa, Bacon, Huevo Frito, Queso y Salsa Rosa 3,90 €
Añadir una nueva carta


Gmail
Perfil
Carta
Estadística
Productos
Bocatapp
Estadística
Ventas realizadas:
40%
35%
30%
25%
20%
15%
10%
5%
0%
Ninguno
Uno
Dos
Tres o m s
Ventas de tipos de bocata:
CARNE
LUNES
MARTES
MIERCOLES
JUEVES
VIERNES
Vegetales
10%
30%
25%
20%
15%

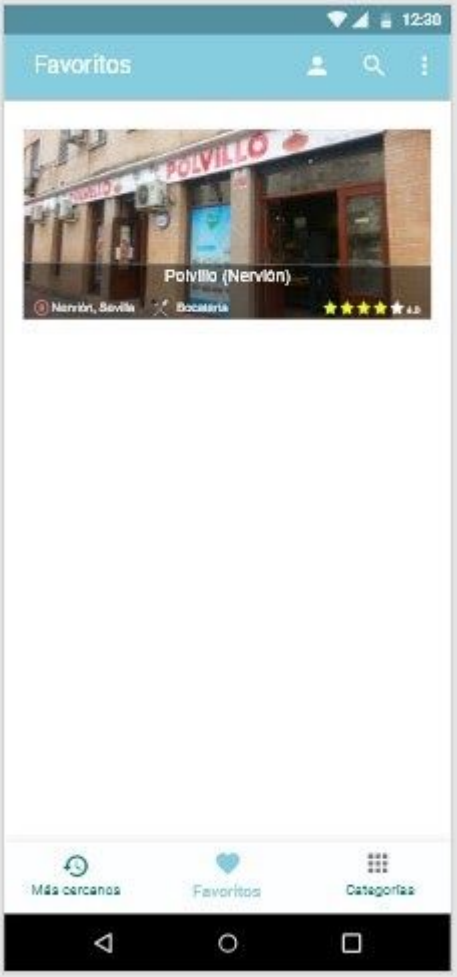

Gmail
Perfil
Carta
Estadística
Productos
Bocatapp
Productos
Imagen del producto:
Información del producto: Este producto esta compuesto por ingredientes que no contienen gluten
Ingredientes: - Pan integral
- Jamón de York
- Lechuga
- Tomate
- Queso de Cheddar
Añadir productos


Email
Perfil
Carta
Estadística
Productos
Almacén
Bocatapp
Almacén
Ingredientes:
Cantidad
- Pan integral 20
- Jamón de York 3kg
- Lechuga 5
- Tomate 30
- Queso de Cheddar 15
- Jamón Serrano 1kg
- Tortilla de patatas 2kg
- Bacon 4kg
- Lomo 5kg
- Pollo 6kg

SKETCHING APLICACIÓN

Más Cercanos

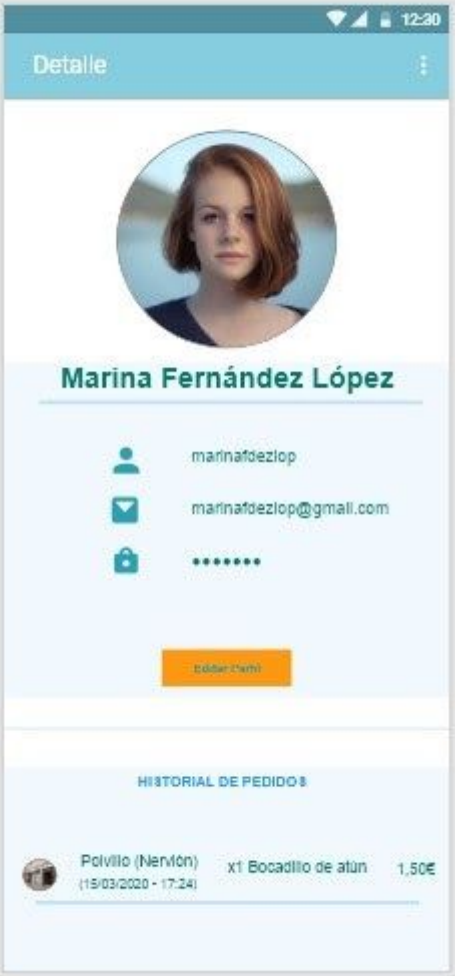
Favoritos



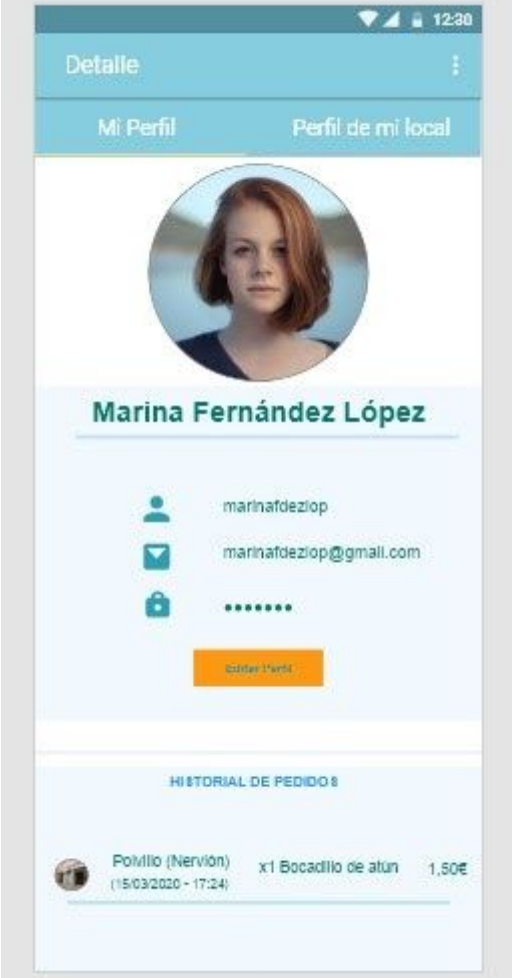
Categorías

Lista (Filtro categorías)

Perfil Cliente



Perfil Gerente



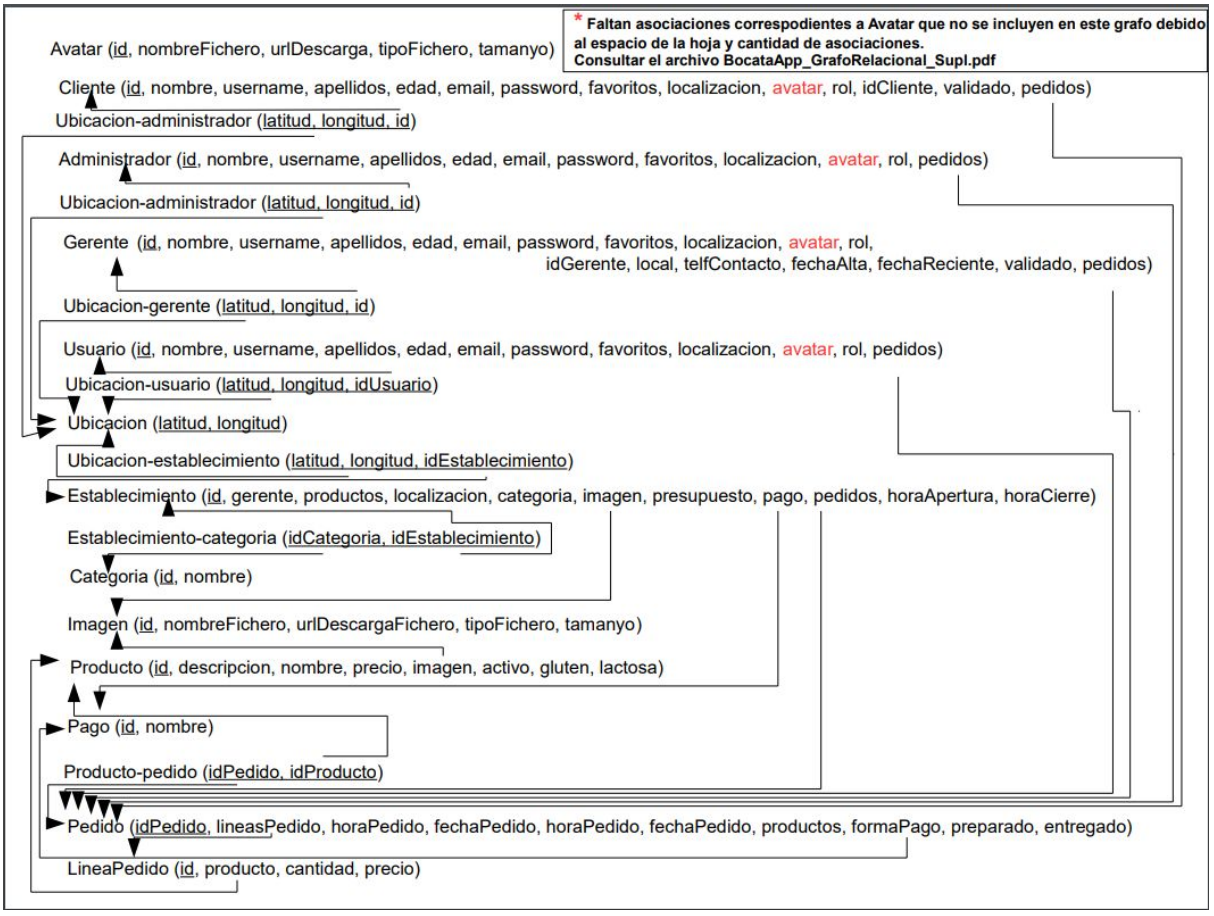
Perfil (Local)

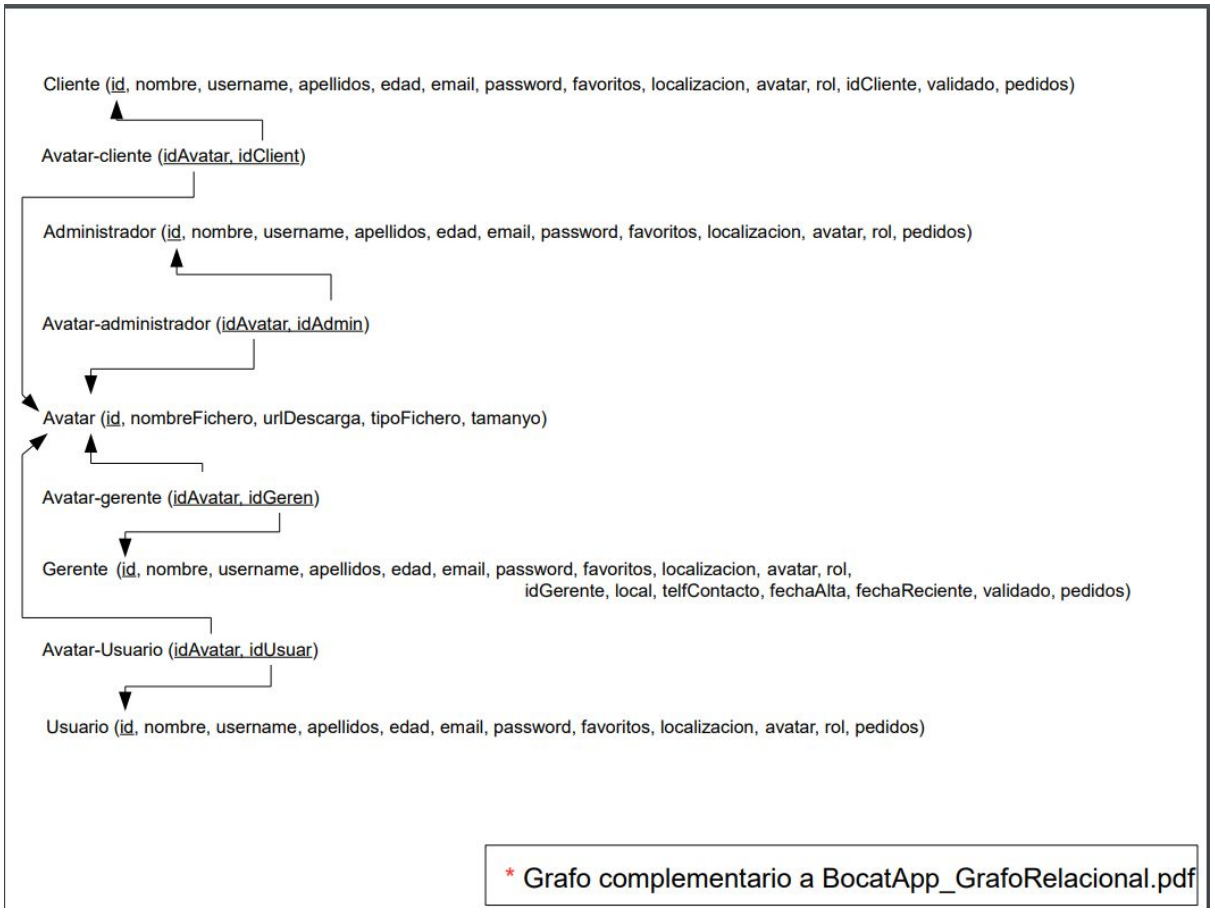
Detalle Local (Cerrado)

Detalle Local

Final Pedido

DISEÑO WEB BOCATAPP





* Grafo complementario a BocatApp_GrafoRelacional.pdf

Administrador:

- **(POST) register: “{{base_url}}/register/”:**
    - Crea un usuario administrador
    - Se le pasará por parámetros las distintas variables, siendo una petición multiparte.
    - Devolverá el administrador creado

- **(GET) getRole: “{{base_url}}/username/”:**
    - Devuelve el rol del usuario indicado
    - Se le pasa el nombre de usuario para obtener el rol
    - Devolverá el rol del usuario indicado

Categoría:

- **(POST) nuevaCategoria: “{{base_url}}/categoria/”:**
    - Crea una nueva categoría
    - Se le pasa como cuerpo la categoría creada
    - Devolverá la categoría registrada

- **(GET) getCategoriasName: “{{base_url}}/categorias/name”:**
    - Devuelvetodos los nombres de las categorías existentes
    - Devuelve una lista con todos los nombres de las categorías

- **(GET) unaCategoria: “{{base_url}}/categoria/{id}”:**
    - Busca una categoría por su id
    - Se le pasa el id de la categoría seleccionada
    - Devuelve la categoría seleccionada
- **(PUT) editarCategoria: “{{base_url}}/categoria/{id}”:**
    - Edita una categoría
    - Se le pasa el id de la categoría que vamos a editar
    - Devuelve la categoría editada

- **(DELETE) deleteCategoria: “{{base_url}}/categoria/{id}”:**
    - Elimina una categoría
    - Se le pasa el id de la categoría que vamos a eliminar

Cliente:

- **(POST) register: “{{base_url}}/register”:**
    - Crea un usuario cliente
    - Se le pasará por parámetros las distintas variables, siendo una petición multiparte.
    - Devolverá el cliente creado
- **(GET) findByEmail: “{{base_url}}/findByEmail”:**
    - Busca un cliente por su email
    - Se le pasará el email del usuario que queremos buscar
    - Devolverá el usuario que coincida con el email

- **(GET) listarClientes: “{{base_url}}/list”:**
    - Lista todos los clientes registrados
    - Devolverá la lista con todos los clientes registrados

- **(PUT) editarFotoCliente: “{{base_url}}/client/{id}/editPhoto”:**
    - Editará la foto del usuario cliente registrado
    - Se le pasará el id del cliente seleccionado y el cuerpo del archivo que se usará para editar.
    - Devolverá el usuario cliente con el archivo editado.

Establecimiento:

- **(GET) getEstablecimientoGerente:”{{base_url}}/local/me”:**
    - Devuelve el establecimiento del gerente loggeado
    - Devuelve el establecimiento

- **(GET) buscarLocal:”{{base_url}}/local/”:**
    - Crea una lista de todos los establecimientos registrados

    - Devuelve la lista con todos los establecimientos registrados

- **(GET) unEstablecimiento:"{{base_url}}/local/{id}":**
    - Devuelve un establecimiento buscado por id
    - Se le pasa el id del establecimiento seleccionado
    - Devuelve el establecimiento seleccionado

- **(POST) añadirLocalFavs:"{{base_url}}/local/{id}":**
    - Añade un local a la lista de favoritos
    - Se le pasa el id del establecimiento seleccionado para añadir a la lista de favoritos

- **(GET) listarLocalesFavs:"{{base_url}}/local/fav/":**
    - Lista todos los locales de la lista de favoritos
    - Devuelve la lista de todos los locales favoritos

- **(POST) nuevoEstablecimiento:"{{base_url}}/local/":**
    - Crea un nuevo establecimiento
    - Se le pasa el archivo y los parámetros que componen el objeto de establecimiento, ya que es una petición multiparte
    - Devolverá el establecimiento registrado

- **(PUT) editarEstablecimiento: "{{base_url}}/local/{id}":**
    - Edita un establecimiento seleccionado
    - Se le pasa el id del establecimiento seleccionado

    - Devuelve el establecimiento seleccionado editado

- **(PUT)editarfotoEstablecimiento:"{{base_url}}/local/{id}/editPhoto":**
    - Edita una foto de un establecimiento seleccionado
    - Se le pasa el id del establecimiento seleccionado y el cuerpo del archivo que se utilizará para editar
    - Devolverá el establecimiento con la imagen editada

- **(DELETE) deleteEstablecimiento:"{{base_url}}/establecimiento{id}":**
    - Elimina un establecimiento
    - Se le pasa el id del establecimiento que vamos a eliminar

File:

- **(GET) downloadFile: "{{base_url}}/downloadFile/{filename}":**
    - Descarga un archivo
    - Se le pasa el nombre del fichero
    - Devuelve el fichero descargado

- **(GET) getImage: "{{base_url}}/image/{filename}":**
    - Trata el archivo descargado para convertirlo en una imagen
    - Se le pasa el nombre del fichero
    - Devuelve el path del fichero para ser utilizado

Gerente:

- **(POST) register: "{{base_url}}/register":**

    - Crea un usuario gerente
    - Se le pasará por parámetros las distintas variables, siendo una petición multiparte.
    - Devolverá el gerente creado

- **(GET) listarGerentes: “{{base_url}}/list”:**
    - Lista todos los gerentes registrados
    - Devolverá la lista con todos los gerentes registrados

Pedido:

Producto:

- **(GET) getProductosGerente: “{{base_url}}/local/me/productos/”:**
    - Devuelve todos los productos de un gerente
    - Devuelve una lista con todos los productos del gerente
- **(POST) newProducto: “{{base_url}}/producto/”:**
    - Crea un nuevo producto
    - Se le pasa el archivo y los parámetros que componen el objeto de producto, ya que es una petición multiparte
    - Devuelve el producto creado

- **(DELETE) deleteProducto: “{{base_url}}/producto/{id}”:**
    - Borra un producto
    - Se le pasa el id del producto que vamos a borrar

- **(PUT) editProducto: “{{base_url}}/producto/{id}”:**
    - Edita un producto seleccionado
    - Se le pasa el id del producto que vamos a editar
    - Devuelve el producto editado

- **(PUT)editarFotoProducto :"{{base_url}}/local/{id}/editProducto":**
    - Edita una foto de un producto seleccionado
    - Se le pasa el id del producto seleccionado y el cuerpo del archivo que se utilizará para editar
    - Devolverá el producto con la imagen editada

Role:

- **(GET) getRole: "{{base_url}}/{username}":**
    - Devuelve el rol del usuario indicado
    - Se le pasa el nombre de usuario para obtener el rol
    - Devolverá el rol del usuario indicado

IMPLEMENTACIÓN

API

Aquí se describen las clases y paquetes de la API

CONTROLLERS
Este paquete recoge las clases que implementa las peticiones de los modelos

- **AdminController:** Clase que implementa los servicios del modelo Administrador
- **CategoriaController:** Clase que implementa los servicios del modelo Categoria

- **ClienteController:** Clase que implementa los servicios del modelo Cliente
- **EstablecimientoController:** Clase que implementa los servicios del modelo Establecimiento
- **FileController:** Clase que implementa los servicios del modelo File
- **GerenteController:** Clase que implementa los servicios del modelo Gerente
- **LineaPedidoController:** Clase que implementa los servicios del modelo LineaPedido
- **PedidoController:** Clase que implementa los servicios del modelo Pedido
- **ProductoController:** Clase que implementa los servicios del modelo Producto
- **RoleController:** Clase que implementa los servicios del modelo Role

CONVERSORES

Este paquete recoge las clases que implementan los métodos para convertir modelos a dto y viceversa

- **ConversorCategoria** clase que implementa los métodos para convertir dtos y modelos de Categoria
- **ConversorEstablecimiento** clase que implementa los métodos para convertir dtos y modelos de Establecimiento
- **ConversorGerente** clase que implementa los métodos para convertir dtos y modelos de Gerente
- **ConversorImagen** clase que implementa los métodos para convertir dtos y modelos de Imagen
- **ConversorProducto** clase que implementa los métodos para convertir dtos y modelos de Producto
- **ConversorUbicacion** clase que implementa los métodos para convertir dtos y modelos de Ubicacion
- **ConversorUsuario** clase que implementa los métodos para convertir dtos y modelos de Usuario

DTOS
Este paquete recoge todos los dtos utilizado para la conversión de los datos de modelo

- **CategoriaDto** clase que filtra la clase modelo Categoria
- **CategoriaDto2** clase que filtra la clase modelo Categoria
- **CategoriaDtoName** clase que filtra la clase modelo Categoria
- **ClienteDto** clase que filtra la clase modelo Cliente
- **CreateAdminDto** clase que filtra la clase modelo Admin
- **CreateCategoriaDto** clase que filtra la clase modelo Categoria
- **CreateClienteDto** clase que filtra la clase modelo Cliente
- **CreateEstablecimientoDto** clase que filtra la clase modelo Establecimiento
- **CreateGerenteDto** clase que filtra la clase modelo Gerente
- **CreateLocalizacionDto** clase que filtra la clase modelo Localizacion
- **CreateProductoDto** clase que filtra la clase modelo Producto
- **CreateUserDto** clase que filtra la clase modelo User
- **EditAvatarDto** clase que filtra la clase modelo Avatar
- **EditEstablecimientoDto** clase que filtra la clase modelo Establecimiento
- **EditImagenEstablecimientoDto** clase que filtra la clase modelo Establecimiento
- **EditImagenProductoDto** clase que filtra la clase modelo Producto
- **EditProductoDto** clase que filtra la clase modelo Producto
- **EstablecimientoDto** clase que filtra la clase modelo Establecimiento
- **GerenteDto2** clase que filtra la clase modelo Gerente
- **ImagenDto** clase que filtra la clase modelo Imagen
- **ListaEstablecimientoDto** clase que filtra la clase modelo Establecimiento
- **ListaProductosDto** clase que filtra la clase modelo Producto
- **ListUsersDto** clase que filtra la clase modelo Usuario
- **ProductoDto** clase que filtra la clase modelo Producto

- **ProductoDto2** clase que filtra la clase modelo Producto
- **UbicacionDto** clase que filtra la clase modelo Ubicacion
- **UserDto** clase que filtra la clase modelo Usuario

Dto.Converter
Paquete con la clase UserDtoConverter utilizada en la seguridad

- **UserDtoConverter** clase que es utilizada en la seguridad

Error
Este paquete recoge las clases que gestionan los distintos errores

- **ApiError** clase que gestiona los errores
- **CustomAccesDeniedHandler** clase que gestiona los errores
- **CustomAuthenticationEntryPoint** clase que gestiona los errores
- **EntityNotFoundException** clase que gestiona los errores
- **RestExceptionHandler** clase que gestiona los errores

Files
Este paquete recoge las clases que gestionan subida, descarga y almacenamiento de ficheros

- **FileStorageException** clase que gestiona el almacenamiento de archivos
- **FileStorageProperties** clase que gestiona las propiedades del almacenamiento
- **FileSystemStorageService** clase que implementa el servicio de almacenamiento
- **MyFileNotFoundException** clase que implementa las excepciones del almacenamiento

Model
Este paquete recoge las clases que implementan los modelos de la api

- **Admin** clase que implementa el modelo Administrador
- **Avatar** clase que implementa el modelo Avatar
- **Categoria** clase que implementa el modelo Categoria
- **Cliente** clase que implementa el modelo Cliente
- **Establecimiento** clase que implementa el modelo Establecimiento
- **Gerente** clase que implementa el modelo Gerente
- **Imagen** clase que implementa el modelo Imagen
- **LineaPedido** clase que implementa el modelo LineaPedido
- **Pago** clase que implementa el modelo Pago
- **Pedido** clase que implementa el modelo Pedido
- **Producto** clase que implementa el modelo Producto
- **Role** clase que implementa el modelo Role
- **Ubicacion** clase que implementa el modelo Ubicacion
- **Usuario** clase que implementa el modelo Usuario

Repository
Este paquete recoge las clases que implementan los repositorios de los modelos de la api

- **AdminRepository** clase que implementa el repositorio del modelo Administrador
- **AvatarRepository** clase que implementa el repositorio del modelo Avatar
- **CategoriaRepository** clase que implementa el repositorio del modelo Categoria
- **ClienteRepository** clase que implementa el repositorio del modelo Cliente
- **EstablecimientoRepository** clase que implementa el repositorio del modelo Establecimiento
- **GerenteRepository** clase que implementa el repositorio del modelo Gerente

- **ImagenRepository** clase que implementa el repositorio del modelo Imagen
- **ProductoRepository** clase que implementa el repositorio del modelo Producto
- **UbicacionRepository** clase que implementa el repositorio del modelo Ubicacion
- **UsuarioRepository** clase que implementa el repositorio del modelo Usuario

Security
Este paquete recoge las clases que implementan la seguridad OAuth2 + Spring Security

- **Cors** clase que implementa la seguridad
- **CorsConfigServer** clase que implementa la seguridad
- **CustomUserDetailsService** clase que implementa la seguridad
- **OAuthConfig** clase que implementa la seguridad
- **ResourceServerConfig** clase que implementa la seguridad
- **ServerSecurityConfig** clase que implementa la seguridad

Services
Este paquete recoge las clases que implementan los servicios de los modelos de la api

- **AdminService** clase que implementa los métodos del modelo Administrador
- **AvatarService** clase que implementa los métodos del modelo Avatar
- **BaseService** clase que implementa los métodos básicos
- **CategoriaService** clase que implementa los métodos del modelo Categoria

- **ClienteService** clase que implementa los métodos del modelo Cliente
- **EstablecimientoService** clase que implementa los métodos del modelo Establecimiento
- **GerenteService** clase que implementa los métodos del modelo Gerente
- **ImagenService** clase que implementa los métodos del modelo Imagen
- **ProductoService** clase que implementa los métodos del modelo Producto
- **UbicacionService** clase que implementa los métodos del modelo Ubicacion