



Gestión de pedidos y entregas por:

Damià Martínez Grau

Jesús Salvador Giménez

Miguel Navarro Travé

Vicent Sargues Blanch

Wei Luo

ÍNDICE

A. Aplicación Móvil en React Native

1. LoginScreen
2. MainScreen
3. ItemsListScreen
4. ClientsListScreen
5. AddClientScreen
6. VisitSalesScreen
7. OrderItemsListScreen
8. OrderAddItemScreen
9. OrderConfirmScreen
10. VisitDeliverScreen
11. DeliverCheckScreen
12. DeliverConfirmScreen

B. Backoffice con React

C. Backend + WebApi

D. Despliegue del servidor

Webgrafia

1. Material Design Icons: <https://kutt.it/RP1egG>
2. Claves primarias compuestas: <https://kutt.it/SGYVd8>
3. Renderización reiterada en ReactNative: <https://kutt.it/3PCFuC>
4. Comunicación padre-hijo mediante estados: <https://kutt.it/aQZx5T>
5. Guía de flexbox: <https://kutt.it/xCCMyG>
6. Configurar CORS Api Rest: <https://kutt.it/Vv3Jgx>
7. Eliminar SSL de la Api Rest: <https://kutt.it/b9WesV>
8. Publicar Api Rest .Net en IIS: <https://kutt.it/lIhrVZ>
9. Montar servidor NGINX: <https://kutt.it/GHVJwA>
10. Permitir a WebApi en IIS recibir modificaciones de datos: <https://kutt.it/zMT6Qm>
11. Compilar la APK: <https://kutt.it/tq4Pk3>
12. Librería AsyncStorage: <https://kutt.it/45ZBpG>
13. Librería PrimeReact: <https://kutt.it/50Q9LK>
14. Librería ReactNative: <https://kutt.it/GeZxnV>
15. Librería ReactNativeElements: <https://kutt.it/VBYfAK>

Brief explanation of the app

GPE is an application designed and developed for the solution of our fictitious company dedicated to the sale of products for bars and restaurants.

We allow our company's employees to connect with their own employee and depending on the work they do in the company the application will behave in one way or another, allowing to place orders to employees listed as salesman and allowing deliveries to employees listed as deliverer.

With the app we can have real-time information about the stocks of the warehouse and the lots that we have to make the sales.

Employees also have a feature to ennsing new customers in case they make a new acquisition.

All information they generate will be sent to the servers instantly after completing the required insertion of data requested by the application itself.

Apartado A. Aplicación móvil con React Native

La aplicación ha sido desarrollada en su totalidad con React Native utilizando el emulador de Android Studio para comprobar el desarrollo de la misma.

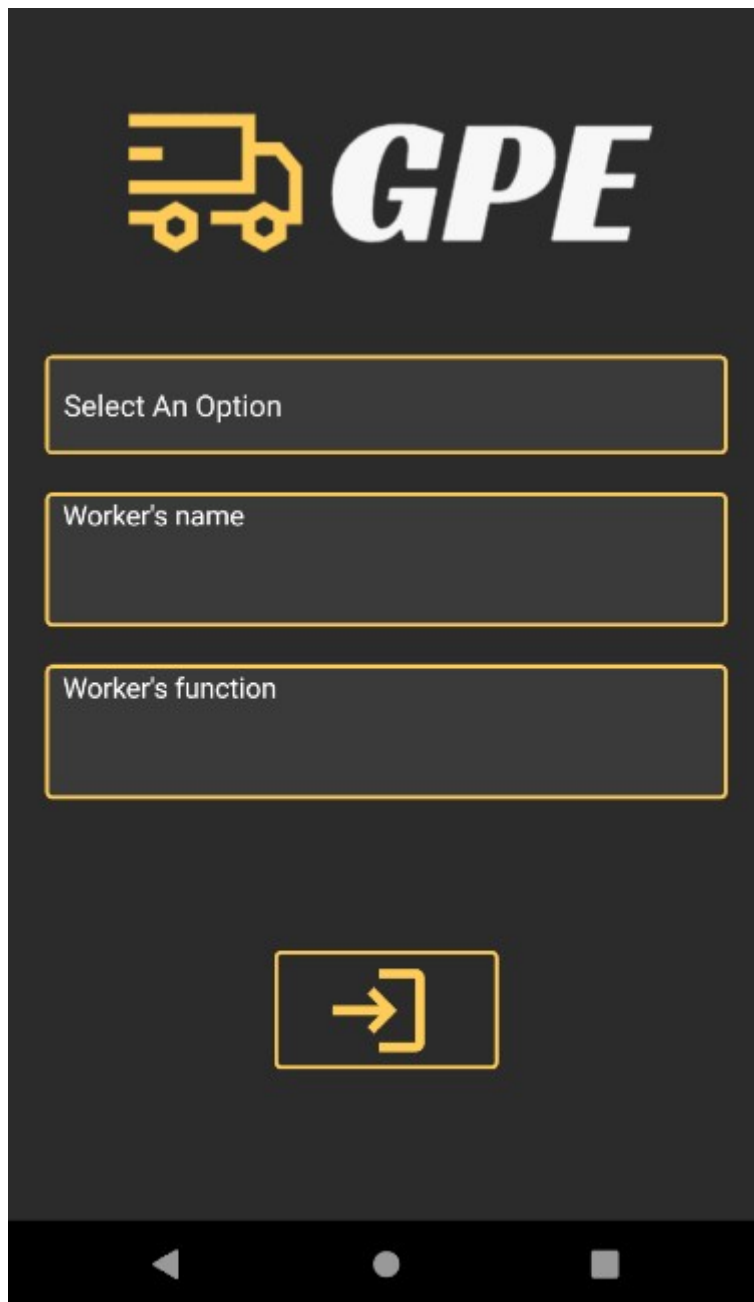
Las librerías utilizadas han sido las de React, ReactNativeElements y ReactNativeVectorIcons. Para la persistencia de datos en la aplicación se ha hecho uso de la librería de AsyncStorage y la navegación mediante la navegación “*Stack*” de react-navigation.

Se esperaba desarrollar la aplicación tal y como la hemos desarrollado, se han incluido cosas que no se contemplaron desde un inicio ya que nos dimos cuenta de que la aplicación requería de más cosas para ser mínimamente funcional y entre otros que afrontase una solución a un problema real. Gestión de pedidos con numeración automática, control de lotes, venta basada en stocks, creación de clientes nuevos desde la app, entrega de pedidos y creación de los mismos...

Hemos tenido que dejar muchas otras funcionalidades en el tintero debido a la escasez de tiempo, almacenamiento de datos en una BBDD en el dispositivo para evitar hacer las mínimas llamadas al servidor, creación de series de pedidos, un sistema de autoventa mediante el cual el propio empleado pueda generar sus propios pedidos y entregarlos él mismo, un sistema de regulación de stock del vehículo del transportista e inventariado, impresiones mediante impresoras portátiles con formularios desde el dispositivo y la posibilidad de acceder a una lista de cobros pendientes para aquellas entregas que se realizaron pero no se cobró la totalidad del pedido.

1. LoginScreen

Pantalla de inicio de la aplicación, mediante el uso de un componente propio “*GPEPicker*” de datos que trabaja con una “promesa” para recuperar todos los empleados marcados como “activos” en la BBDD del servidor podemos escoger que empleado va a hacer el *login*, nos mostrará en pantalla cual es el nombre del empleado que hemos seleccionado y cuál es su función. Una vez se pulse sobre el botón de *login* guardará en el “*LocalStorage*” el objeto empleado que hemos escogido, este objeto será utilizado en 2 pantallas más para mandar información o dar funcionalidad a la aplicación. Si nos hemos deslogueado, al acceder de nuevo a esta pantalla seguiremos teniendo el *login* anterior. En caso de no se escoja ningún empleado avisaremos de que no se ha escogido y no podremos continuar.

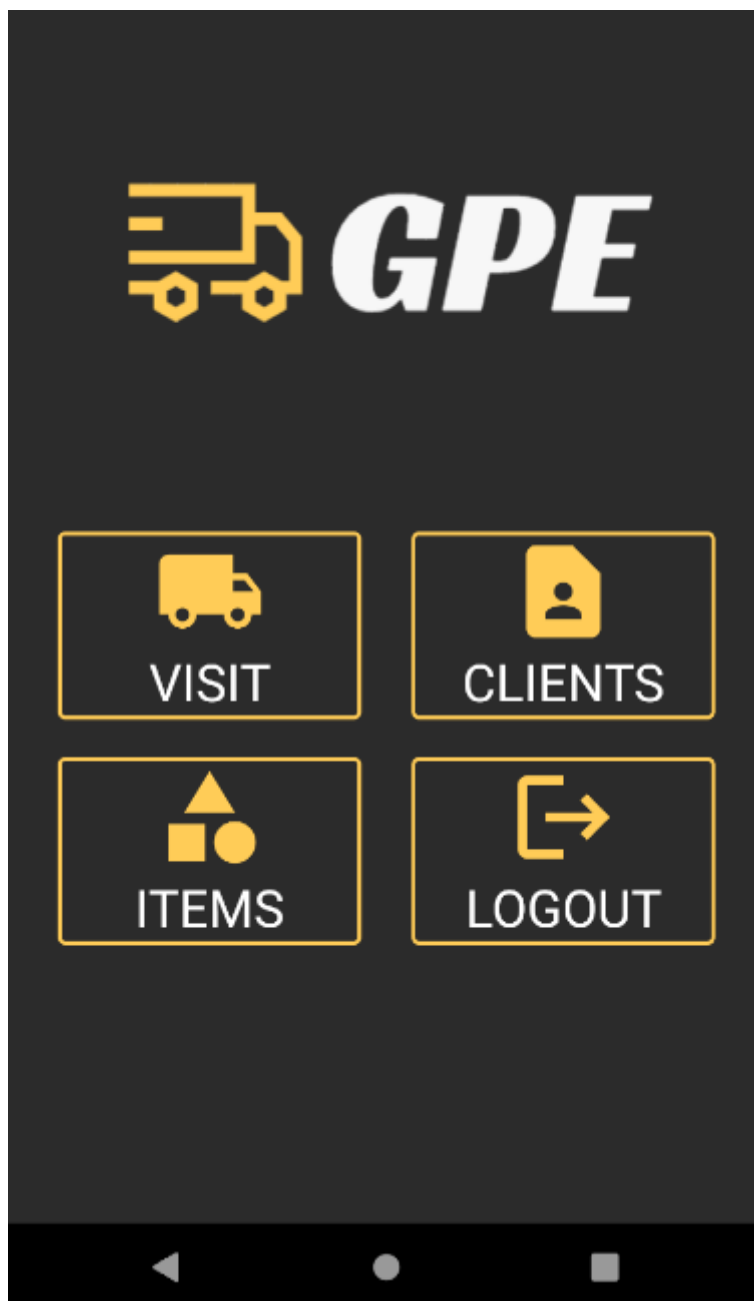


The screenshot displays the GPE application's login screen. At the top, there is a logo consisting of a yellow truck icon and the text "GPE" in a large, white, bold, italicized font. Below the logo, there are three input fields with yellow borders. The first field is labeled "Select An Option" and contains a dropdown menu. The second field is labeled "Worker's name" and is empty. The third field is labeled "Worker's function" and is also empty. At the bottom of the form, there is a yellow button with a right-pointing arrow and a bracket symbol. The entire interface is set against a dark gray background. At the very bottom, there is a black bar with three white icons: a triangle, a circle, and a square, which are standard Android navigation icons.

2. MainScreen

Pantalla principal de la aplicación, aquí disponemos de las 4 opciones básicas de la aplicación, podemos acceder a visitar clientes, esta opción está sujeta al tipo de empleado que se ha *loggeado*, en caso de ser un “*Deliverer*” tendrá acceso a la pantalla de “*VisitDeliverScreen*” y en caso de ser un “*Salesman*” tendrá acceso a la pantalla de “*VisitSalesScreen*” las cuales explicaremos más adelante. Podremos acceder a los apartados de clientes y artículos, ambos independientes de la función del trabajador. También podemos desloguearnos y acceder de nuevo a la pantalla de *login* para cambiar de empleado.

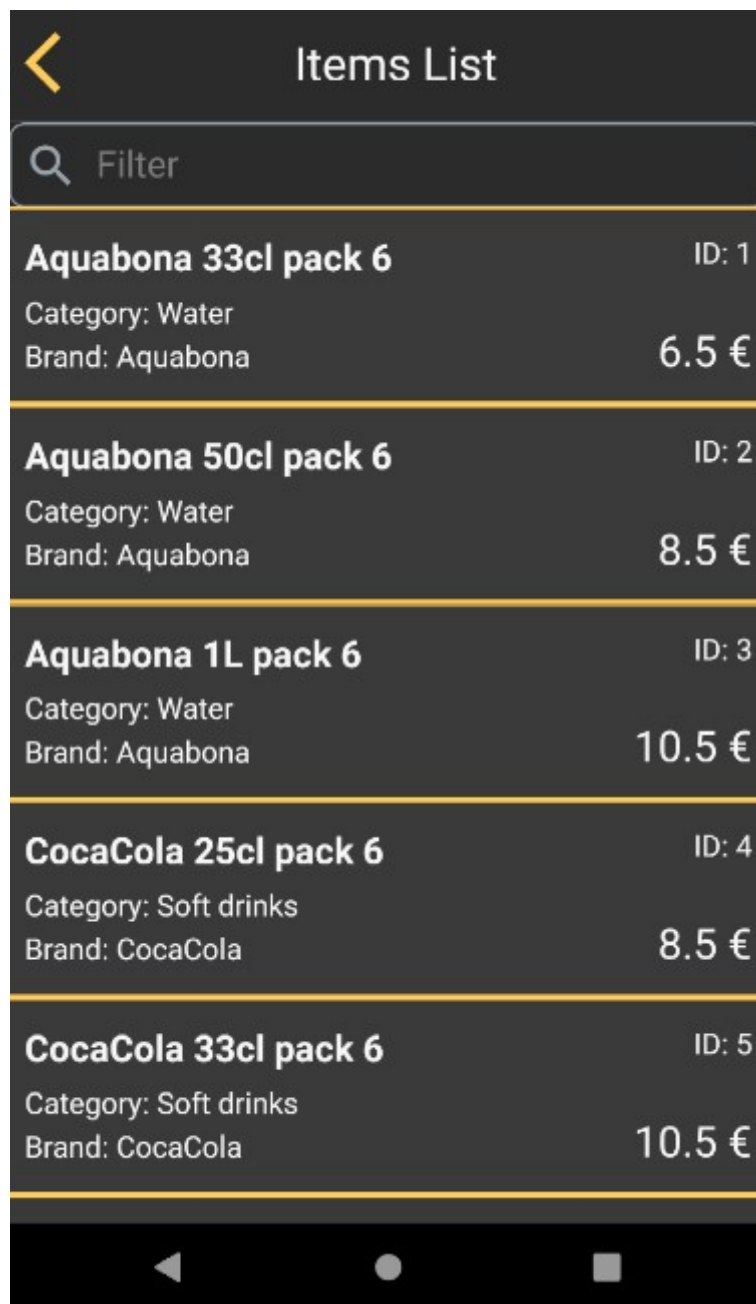
En esta pantalla se recupera desde el “*LocalStorage*” el objeto empleado para la condición de navegación que existe en el botón de visita.



3. ItemsListScreen

Pantalla donde tenemos acceso a una vista rápida de los artículos, utilizamos un componente propio como barra de navegación, otro como input de filtrado de información de la propia pantalla, se puede escribir cualquier campo de los que tenemos a la vista y poder filtrar por él, a excepción del precio. Por último, se utiliza un “Flatlist” para mostrar el componente “ItemCard” hecho por nosotros mismos tantas veces como resultados nos haya devuelto la “promesa” desde el servidor.

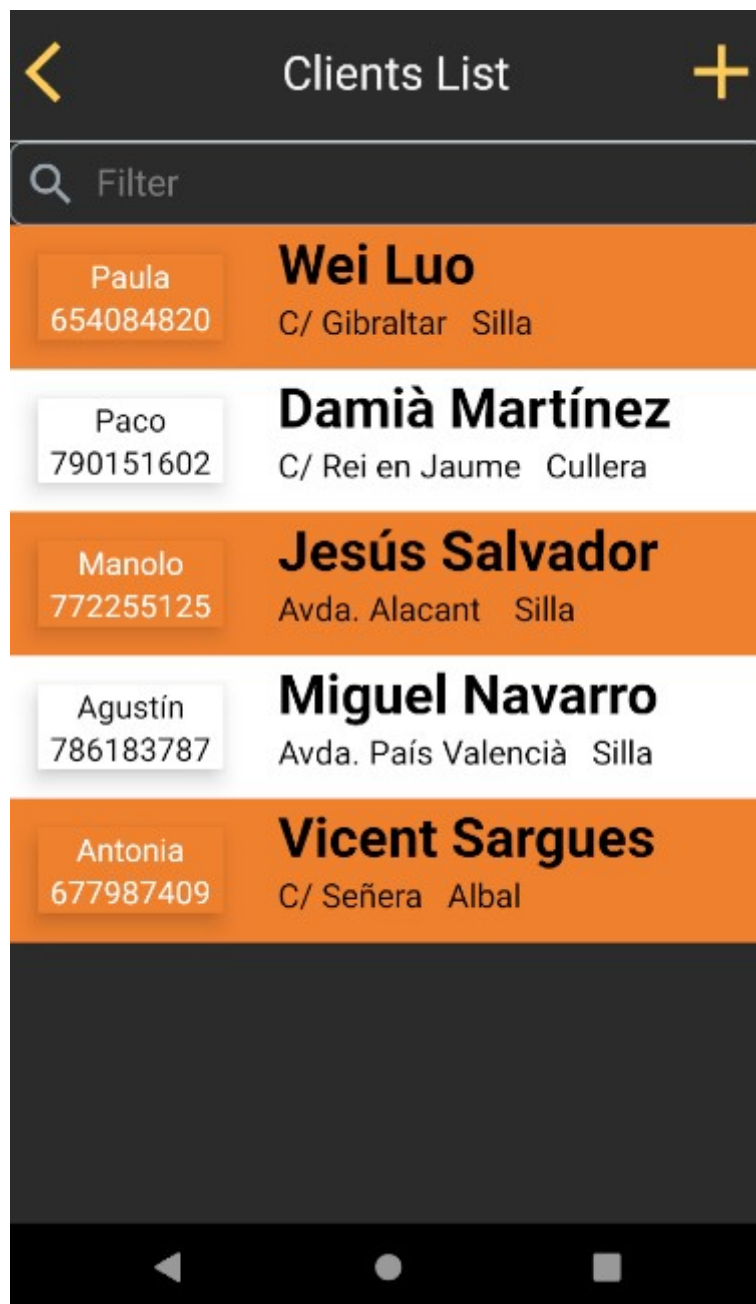
La navegación de esta pantalla solo permite volver a la pantalla anterior, la pantalla principal.



4. ClientsListScreen

Pantalla donde tenemos acceso a la lista de clientes que nos proporciona el servidor a través de una “promesa”, podemos utilizar el filtrado para todos los datos que vemos en pantalla, mostramos una “*FlatList*” que renderiza el componente “*ClientCard*” realizado por nosotros, mostrando una pequeña tarjeta con la persona de contacto y el teléfono junto a los datos de la empresa y su ubicación, dependiendo del “*index*” de la “*FlatList*” cada tarjeta sale de un color u otro para facilitar la diferenciación de tarjetas.

Desde esta pantalla podemos navegar a la pantalla de “*ClientAddScreen*”, asimismo, podemos volver a la pantalla principal.



5. ClientAddScreen

Pantalla en la que mediante una “promesa” podemos realizar la inserción de un cliente nuevo, Controlamos que se inserte el NIF correctamente, si no coincide la letra con el número se avisa, utilizamos el componente “*GPEInput*” para realizar la inserción de los datos, asimismo, en caso de que alguno de los campos no esté relleno avisaremos de ello.

Una vez le damos al botón del “*check*” nos aparecerá un el componente “*GPEModal*”, el cual nos mostrará un mensaje indicándonos si deseamos continuar o si queremos seguir en la pantalla, si le damos a aceptar enviará la información al servidor.

Ambos botones nos permitirán navegar a la pantalla anterior.

The screenshot shows the 'Add Client' screen with the following fields:

- Name (placeholder: Name)
- Email (placeholder: Email)
- Contact Name (placeholder: Contact name)
- NIF (placeholder: NIF)
- Phone number (placeholder: Phone number)
- City (placeholder: City)

6. VisitSalesScreen

Pantalla de visitas de los comerciales, en caso de que el trabajador seleccionado sea un “*Salesman*”, como hemos comentado en la pantalla principal, nos permitirá tener acceso a la lista de clientes que recuperamos con una “promesa”, tenemos la opción de utilizar el filtrado que hemos comentado en pantalla anteriores o podemos crear un cliente nuevo en el caso de que no lo tengamos disponible con el botón “+” de la barra de navegación, nos derivará a la pantalla de “*ClientAddScreen*”.

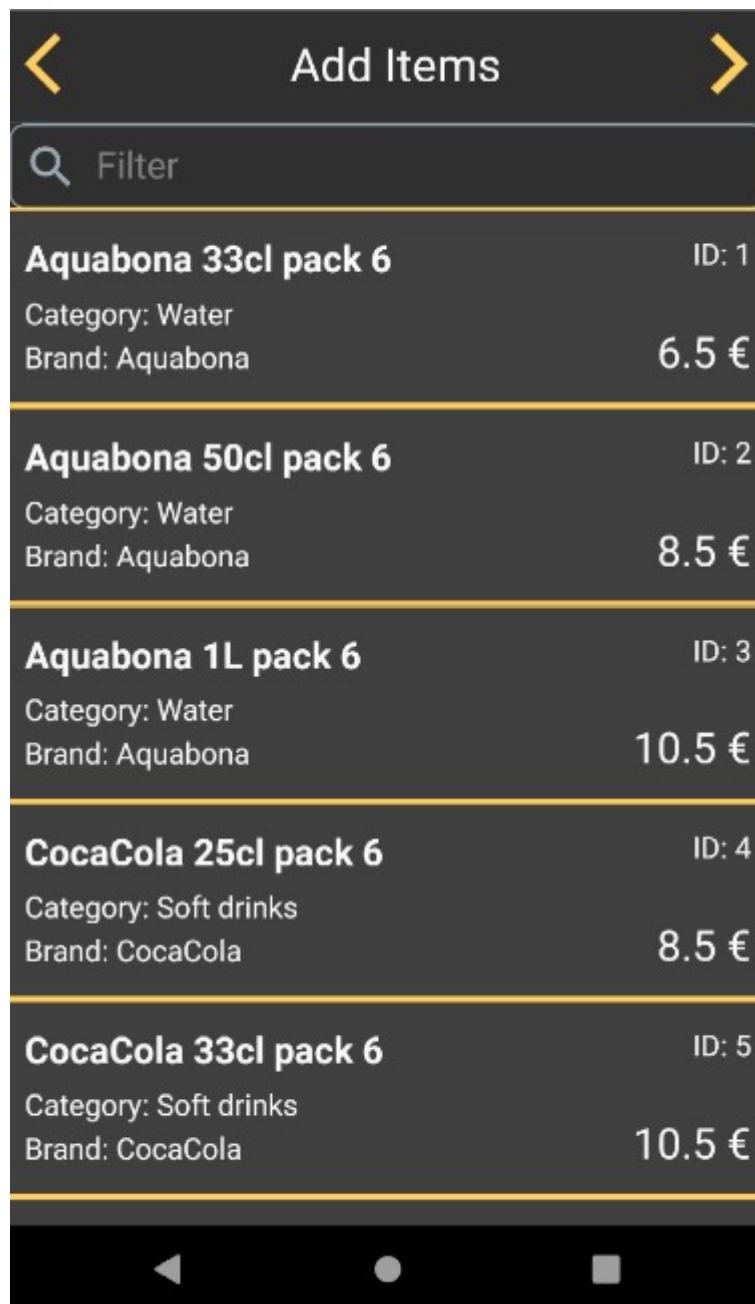
Para visitar un cliente y coger toda su información para pasarla por parámetros a la próxima pantalla solamente debemos pulsar sobre una tarjeta configurada como “*Pressable*” y nos navegará a la pantalla de “*OrderArticlesScreen*” donde iniciaremos el pedido. También podemos navegar atrás a la pantalla principal.



7. OrderArticlesScreen

Pantalla de venta de artículos, aquí disponemos de una “*FlatList*” donde mostramos la información recuperada a través de una “promesa”, podemos utilizar el filtrado para buscar cualquier información de la que disponemos visualmente. Pulsando sobre un elemento de la “*FlatList*” navegaremos a la pantalla de “OrderAddItemScreen” o si pulsamos sobre la flecha superior derecha navegaremos a la pantalla de “OrderConfirmScreen” para finalizar el pedido. Cada vez que pulsemos sobre un elemento de la “*FlatList*” estaremos realizando la acción de insertar nuevas líneas de pedido.

Al navegar a la siguiente pantalla le pasamos por parámetros el cliente, el pedido y las líneas de pedido. En caso de navegar a la inserción de artículos pasamos por parámetros los artículos, lotes y líneas de pedido.



8. OrderAddItemsScreen

Pantalla de inserción de artículos en las líneas del pedido, al tratar con artículos alimenticios debemos realizar un control de lotes, disponemos de un componente “*GPEPicker*” al cual le pasamos la información de los lotes del artículo que hemos escogido. Mostramos con nuestros “*GPELabel*” la información de precio del artículo y el total de la línea que hace unos cálculos internos para mostrar el precio final de venta con descuentos e impuestos aplicados, informando en todo momento que % de impuestos se aplican.

Esta pantalla no permite vender por encima del stock disponible que nos indica el servidor que tenemos, nos avisará en caso de hacerlo y en caso de no rellenar los campos de lote y unidades.

Si le damos al botón superior derecho confirmará los campos y los introducirá en un “*array*” de objetos donde almacenamos las líneas del pedido y navegaremos a la pantalla anterior pasando por parámetros la información recibida (pedido y líneas de pedido).

< Add Item +

Turia 33cl pack 6

Select An Option

Units 0 X

Unit price 10.5

Discount 0 X

Total (IVA applied: 21%) 0

9. OrderConfirmScreen

Pantalla de finalización del pedido, aquí recibimos por parámetros el los objetos pedido, líneas de pedido y cargamos desde el “*LocalStorage*” la información del objeto empleado para cuando enviemos la promesa con el pedido y sus líneas añadir que empleado ha realizado el pedido y que empleado lo repartirá, ya que cada “*Salesman*” tiene asignado en la BBDD su “*Deliverer*”.

Disponemos de un componente que permite modificar el cliente al que le estamos haciendo el pedido en caso de habernos equivocado para no tener que borrar el pedido entero y volver a hacerlo. También disponemos de un componente propio “*ModifyQuantity*” que permite modificar las unidades vendidas sobre líneas del pedido ya introducidas e incluso eliminarlo. Siempre que haya una modificación el “*GPELabel*” del final se actualizará con el total del pedido. Cuando pulsemos sobre el “*check*” mostrará el “*GPEModal*” y en caso de que le demos a aceptar al mensaje que nos muestra mandaremos la “promesa” con el pedido y sus líneas al servidor.

< Confirm ✓

Wei Luo
DNI: 23192424Y **CHANGE**

CocaCola 33cl pack 6
ID: 5 - 324 +
Price: 10.5€
Remove Total: 4116.42€

CocaCola 75cl pack 6
ID: 7 - 123 +
Price: 14.5€
Remove Total: 2158.03€

CocaCola 50cl pack 6
ID: 6 - 32 +
Price: 12.5€

Total: €
6758.45 €

10. VisitDeliverScreen

Pantalla de visitas a la que accedemos con el tipo de empleado “*Deliverer*”, aquí mostraremos únicamente un filtro y una “*FlatList*” que recibe la “promesa” del servidor con todos los pedidos marcados como “*Delivered*”=false y que pertenecen al empleado que se ha loggeado. También mostramos, a diferencia de otras “*FlatList*” de clientes, el nº de pedido para ayudar al repartidor a saber que pedido debe entregar en caso de tener más pedidos del mismo cliente ese día, esta condición está hecha sobre el componente “*ClientCard*” solamente pasando por propiedades la pantalla en la que nos encontramos mostrará esta información.

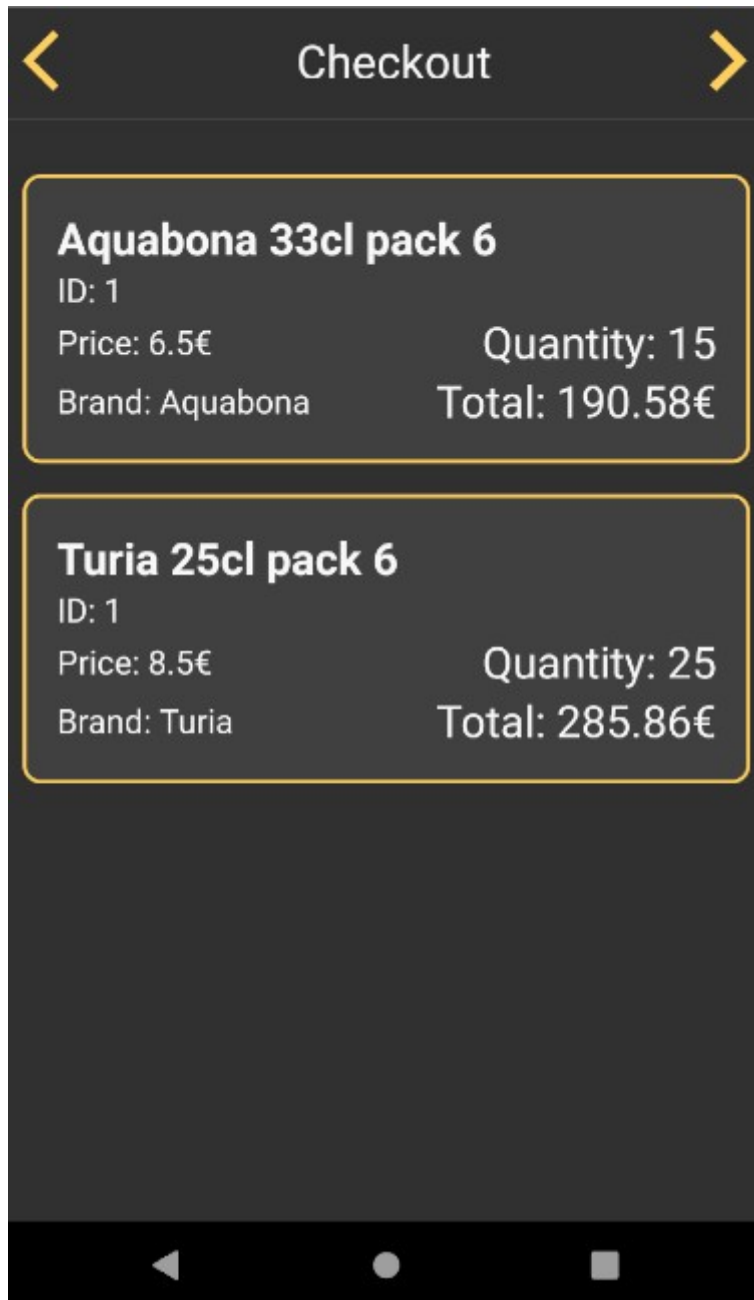
Si pulsamos sobre uno de los pedidos nos navegará a la siguiente pantalla pasando por parámetros toda la información del pedido.



11. DeliverCheckScreen

Pantalla de revisión de artículos a entregar, recuperamos por parámetros la información del pedido seleccionado en la pantalla anterior y mostramos en una “*FlatList*” todos los artículos que hay en las líneas del pedido para que sepa que tiene que entregar al cliente.

Al navegar a la siguiente pantalla pasará por parámetros todo el objeto del pedido.



12. DeliverConfirmScreen

Pantalla de confirmación de entrega del pedido, disponemos de la información básica de que cliente es al que le estamos entregando el pedido con la cantidad a cobrar con los “*GPELabel*”, un “*GPEPicker*” al que le pasamos por propiedades la lista de opciones de pago. Aquí el empleado puede marcar el pedido como pagado con “*Cash*”, “*Credit Card*” o “*Pending*”, en caso de escoger alguna de las dos primeras le obliga al usuario a marcar en el “*GPEInput*” la cantidad que cobrará, no tiene porque ser el total, si no lo hace y termina la entrega tras saltar el “*GPEModal*” con el que verificaremos que quiera finalizar la entrega le avisará de que no está todo completo y no enviará la “promesa”. Si escoge la forma de pago “*Pending*” la forma de pago será siempre 0 aunque el empleado ponga una cantidad.

Payment

Select An Option

Paid
0.0€

Total
1938.98

Client
Wei Luo

Contact Name
Paula

Apartado B. Backoffice con React

La web que actúa como backoffice para dar acceso web a los administradores de las bases de datos ha sido desarrollada en React junto a la librería de Prime-React. Se trata de una web SPA (Single Page Application) en la que damos una vista de las tablas de la BBDD sin necesidad de acceder directamente desde una app de terceros y mediante la cual dotamos de ciertos mecanismos de inserción/consulta/modificación de los datos de la BBDD de forma que evitamos errores en consultas SQL.

No requiere tener conocimientos de ningún lenguaje para que el usuario pueda trabajar sobre el mismo.

En la web utilizamos JavaScript (JSX), hacemos uso de la librería Axios para hacer llamadas al servidor para hacer todas las consultas sobre la BBDD y también se hace uso de MomentJs para mostrar la información de las fechas de forma que el usuario pueda entenderlas debidamente para la región en la que nos encontramos en lugar de la vista americana que devuelve la webApi.

La aplicación es escalable, es más que probable que se amplíe la misma con una mejora visual de la información, así como más opciones de uso y manejabilidad.

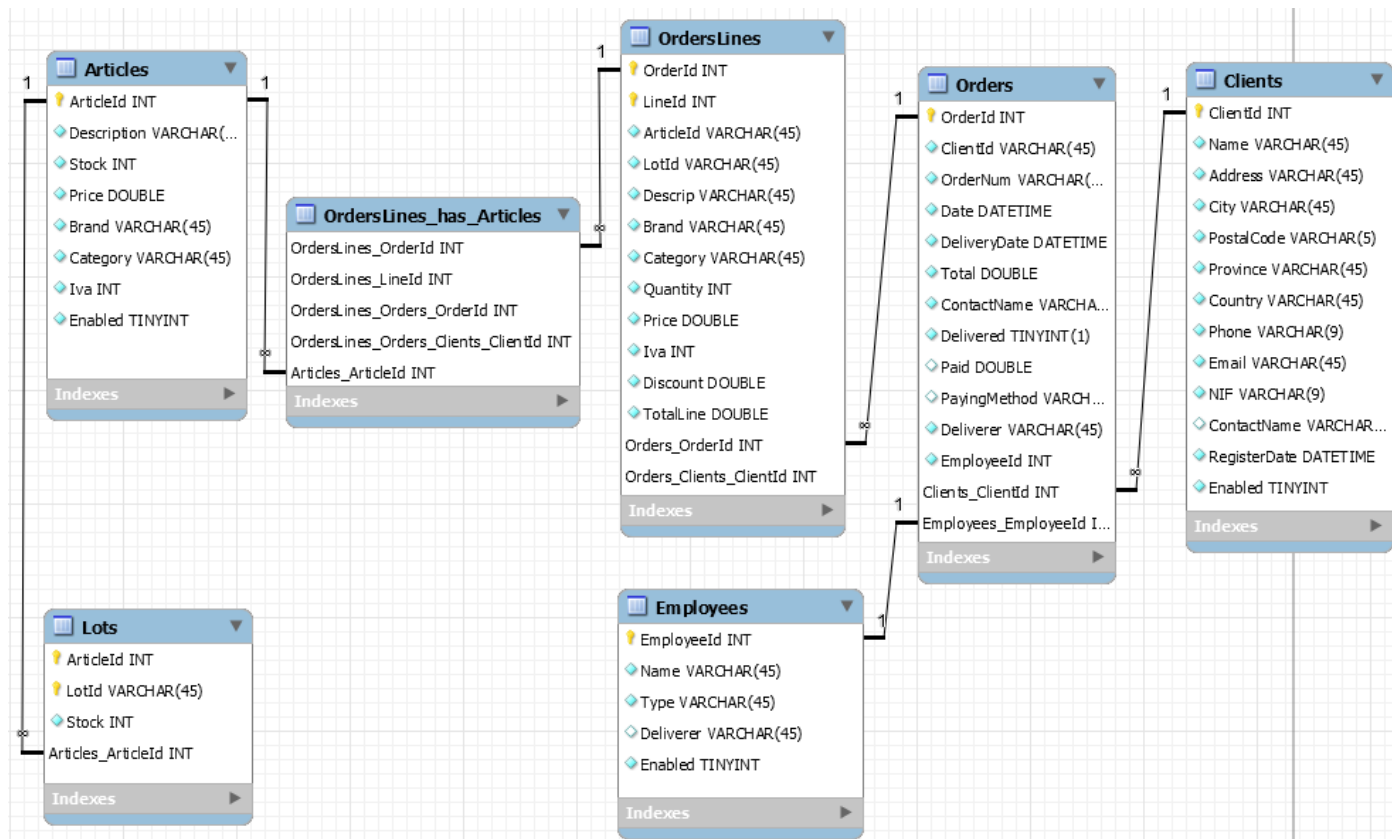
A futuro se podría implementar un sistema de login con usuario y contraseña de forma que también podríamos generar categorías de usuarios con ciertos rangos de permisos, permitiendo a usuarios de alto nivel tener acceso a todas las funcionalidades y a usuarios de bajo nivel solamente lanzar consultas de lectura sobre la BBDD para evitar problemas de pérdida de información indebidas.

Para realizar el deploy del backoffice hemos seguido el punto 9 de la webgrafía.

Apartado C. Backend + WebApi

El backend ha sido desarrollado completamente sobre la plantilla (vacía) de WebApi de Microsoft que nos proporciona Visual Studio Enterprise 2019. Se ha realizado toda la lógica de BBDD orientada en “CodeFirst” mediante “LinQ” con “EntityFramework” desarrollado en C#.

Primero generamos las relaciones en Workbench para tener una ligera idea de que queremos hacer:



Finalmente realizamos las relaciones mediante “CodeFirst” con las clases de cada tabla y sus constructores enlazados con las referencias que debían tener cada tabla.

Configuramos el backend para que fuese compatible con Cors para las “promesas” mediante Axios.

Seguimos la documentación de Microsoft, imprescindible:

Webgrafia punto 6

Instalamos todos los “Nuggets” desde Visual Studio:

- Microsoft.EntityFrameworkCore (versión 3.1.9)
- Microsoft.EntityFrameworkCore.Tools (versión 3.1.9)
- Pomelo.EntityFrameworkCore.MySql (versión 3.2.3)

El “deploy” del backend con WebApi se realizó siguiendo la guía que proporcionamos en la webgrafía puntos 7, 8 y 10.

Apartado D. Despliegue del servidor

El servidor fue desplegado sobre una máquina virtual en AWS con Windows Server 2019, instalamos Xampp para mantener la BBDD MySQL, VisualStudio 2019, HeidiSQL y Git. Configuramos el servidor para que funcionase con IIS siguiendo una guía de la webgrafía punto 8.

Configuramos una IP fija sobre la máquina virtual, permitimos los puertos 80 para la WebApi y el puerto 81 para la web del backoffice.

Se configuró una clave privada con AWS para que solo podamos recuperar la contraseña con la clave privada que solamente tenemos nosotros.