Proyecto FullStack

Pablo Ramallo Amador

Roberto Carlos Ascanio Falcón

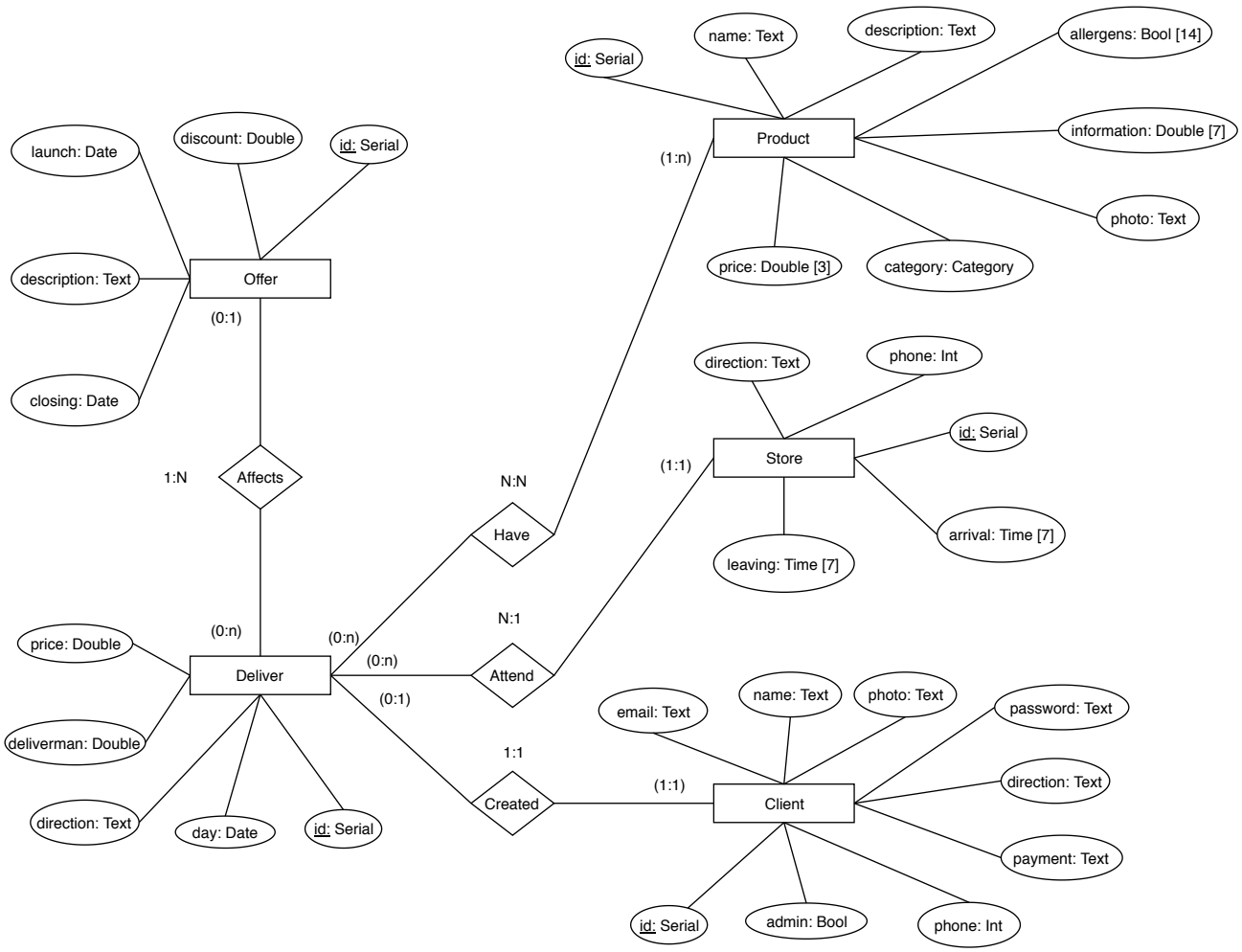
Índice

1.Especificación de requisitos

El cliente requiere una aplicación con la cual pueda realizar pedidos vía a domicilios; para una empresa llamada El Roque. Dicha aplicación existe y lo que desea nuestra empresa es una mejora de esta (nuevas tecnologias)

2. Descripción de la aplicación que se va desarrollar

E/R



Tecnologías

- Front-End: Ionic
- Back-End: Node.js
- ORM: Sequelize
- Base de datos: PostgreSQL

Relacional

Client (<u>id</u>, name, password, photo, direction, payment, phone, administrator)
Deliver (<u>id</u>, price, deliverman, day, direction, id-client*, id-store*, id-offer*)
Store (<u>id</u>, direction, phone, leaving, arrival, week)
Cart (<u>id</u>, sizing, extra, amount, id-product*, id-deliver*)
Offer (<u>id</u>, description, launch, closing, discount)

3. Descripción de cada tabla/entidad

Para lograr gestionar pedidos es necesario tener:

- Producto que contiene el Pedido
- Cliente al que va ser enviado el Pedido
- Tienda la cual gestionara el Pedido
- Oferta a la cual afecta el Pedido
- Carrito es necesario pues existe una relación N:N entre Pedido y Producto (Un Pedido contiene muchos Producto y un Producto puede estar en muchos Pedido)
- Catalogo es necesario pues existe una relación N:N entre Oferta y Pedido (Un Pedido es afectado por muchas Oferta y una Oferta puede afectar en muchos Pedido)

Deliver

- id: identificador del Pedido
- price: precio del Pedido
- deliverman: empleado encargado de trasportar el Pedido
- day: fecha de recogida del Pedido
- direction: dirección destinada del Pedido

Product

- id: identificador del Producto
- name: nombre del Producto
- allergens: Array que contiene la presencia o no de alergénicos del Producto
- price: precios en € dependiendo del tamaño (pequeño, normal, grande) del Producto
- information: información nutricional en kcal(energía) y g(grasas, grasas saturadas, carbohidratos, carbohidratos con azúcar, proteínas, sal) del Producto
- photo: ruta relativa que contiene la foto del Producto
- category: categoría a cual pertenece (bebida, postre, complemento, principal) el Producto
- description: descripción del Producto

Client

- id: identificador del Cliente
- name: nombre del Cliente
- password: contraseña cifrada del Cliente
- photo: ruta relativa que contiene la foto del Cliente
- direction: dirección por defecto la cual sera destinada el Pedido del Cliente
- payment: identificador de la pasarela de cobro del banco del Cliente
- phone: número de teléfono del Cliente
- administrator: tiene permiso o no de administrador el Cliente

Store

- id: identificador de la Tienda
- direction: dirección de la Tienda
- phone: número de teléfono de la Tienda
- arrival: horario de entrada de cada día de la semana de la Tienda
- leaving: horario de salida de cada día de la semana de la Tienda
- week: día de la semana que abre o no la Tienda

Offer

- id: identificador de la Oferta
- description: descripción del Oferta
- launch: fecha de lanzamiento de la Oferta
- closing: fecha de cierre de la Oferta
- discount: descuento en % de la Oferta

Cart

- id: identificador del Carrito
- sizing: tamaño del Producto metido en el Carrito
- extra: aclaración del Producto en el Carrito
- amount: cantidad del Producto en el Carrito

4. UML

Diagrama de Clases

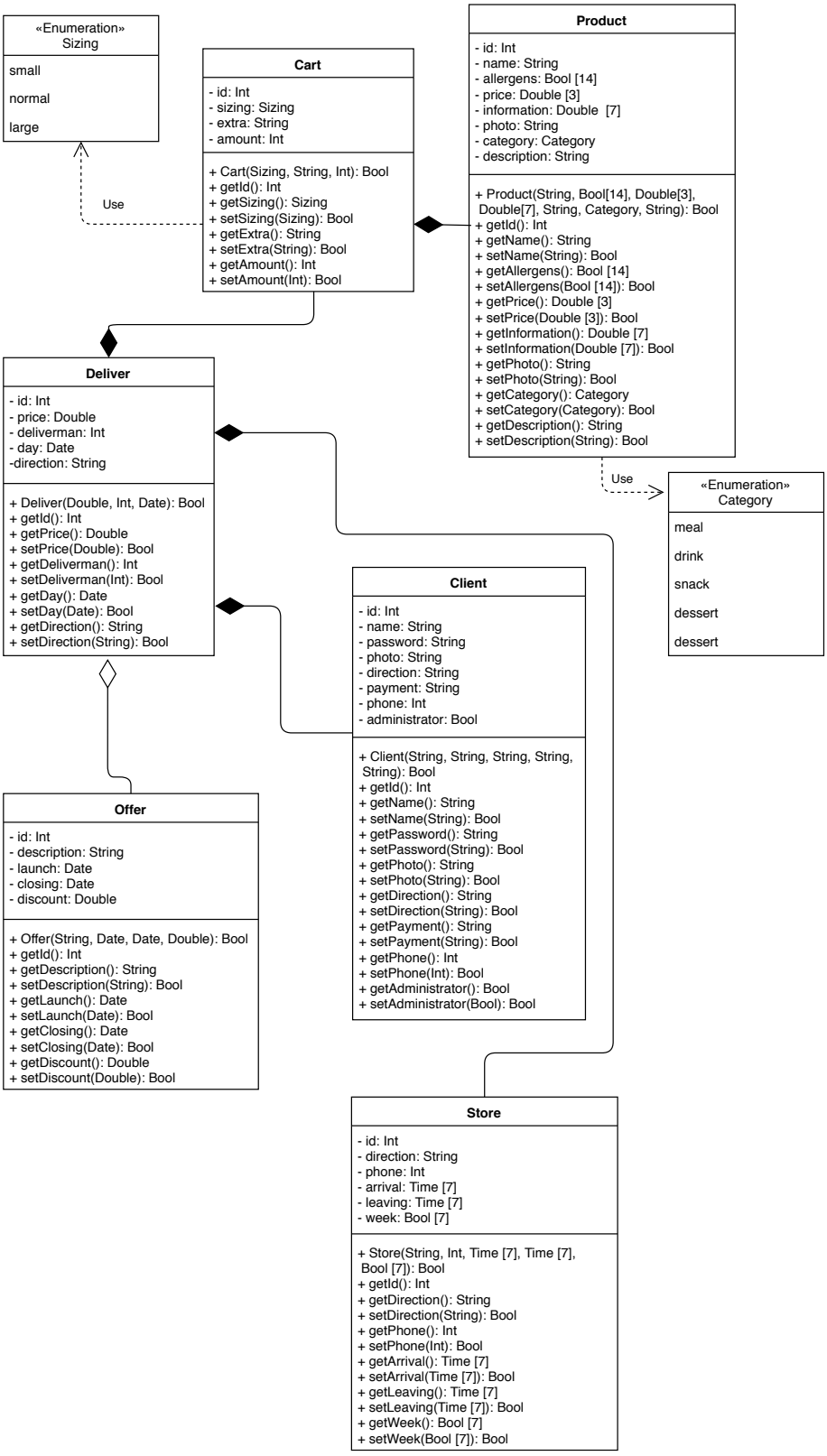
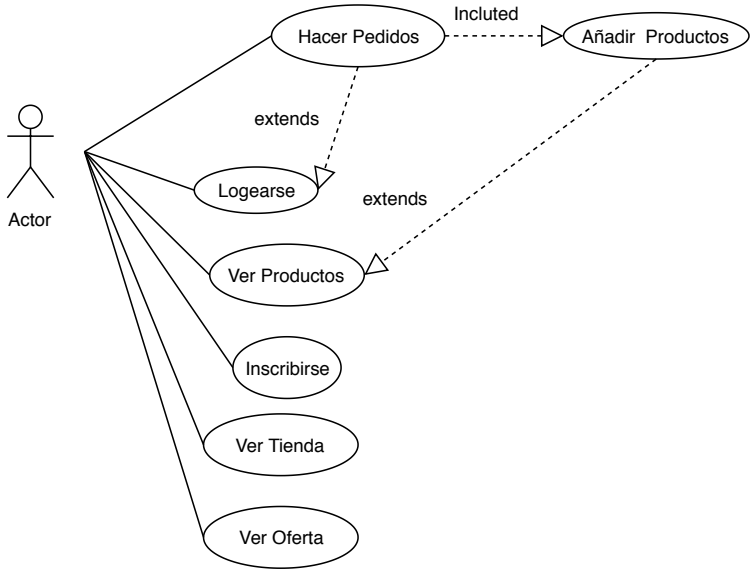


Diagrama de Casos de Usos



Requisitos de Usuario

R1.Hacer pedidos

- R1.1.Loguarse
  - R1.1.1.Registrarse
    - R1.1.1.1.Introducir los datos del usuario (email, nombre, contraseña, dirección por defecto, foto y teléfono)
  - R1.1.2.Introducir email y contraseña
- R1.2.Añadir productos al pedido
  - R1.2.1.Ver los productos
  - R1.2.2.Decidir que tamaño y cantidad deseada
  - R1.2.3.Añadir una nota para modificaciones del producto
- R1.3.Revisar su compra
  - R1.3.1.Modificar, eliminar productos de la compra
- R1.4.Añadir la dirección de entrega

R2.Ver la tiendas disponibles

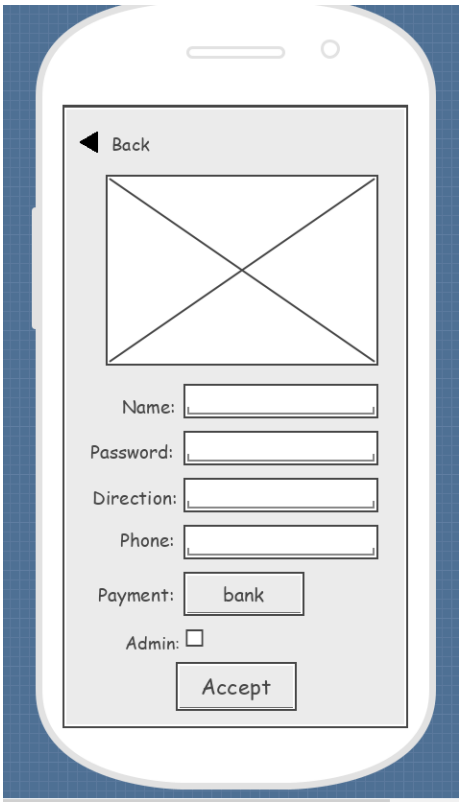
- R2.1.Ver la dirección, teléfono y horario de apertura y cierre.

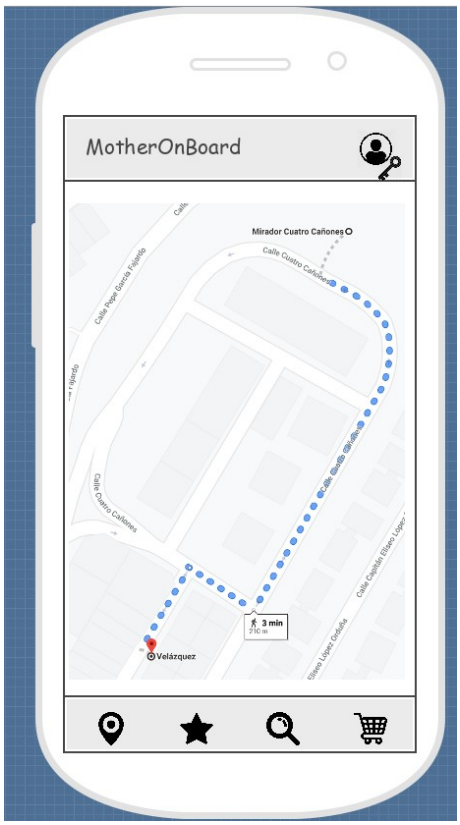
R3.Ver las ofertas disponibles

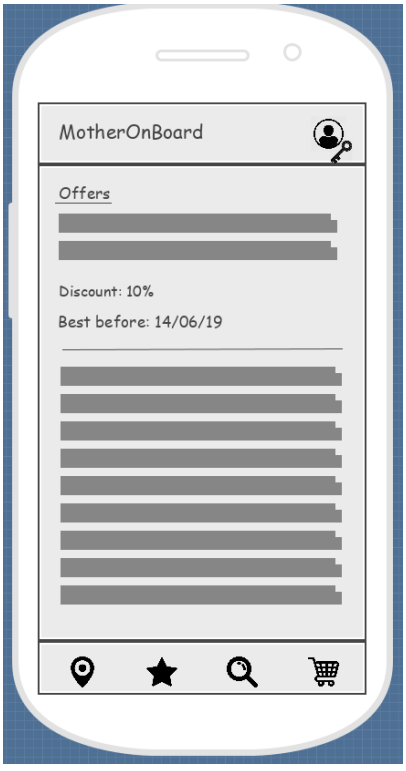
R4.Desloguarse, eliminar y modificar su cuenta junto a su información

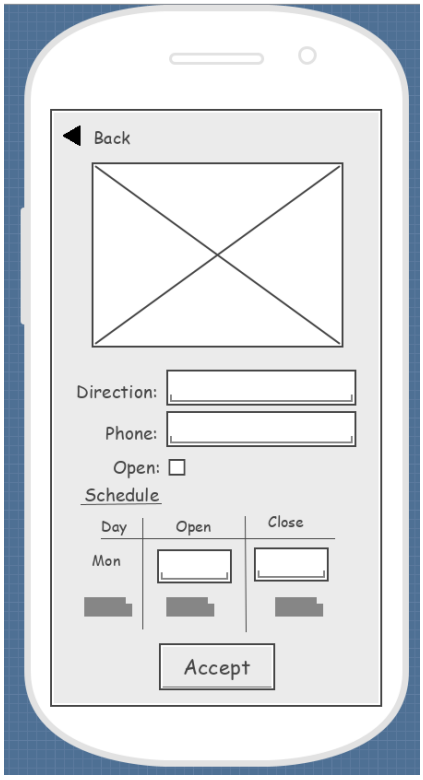
5. MockUp


MotherOnBoard
Cheese Burger
Meal
1
Normal
Cost: 3€
Note:
Add
100g meat covered with 10g cheese between two toasted bread
Nutricional Information
Energy: 3680kcal
Fat: 21.8g with saturates: 20.3
Allergens:


Back
Name:
Password:
Direction:
Phone:
Payment:
bank
Admin:
Accept


MotherOnBoard
Mirador Cuatro Cañones
3 min
Velázquez


MotherOnBoard
Offers
Discount: 10%
Best before: 14/06/19


Back
Direction:
Phone:
Open:
Schedule
Day
Open
Close
Mon
Accept

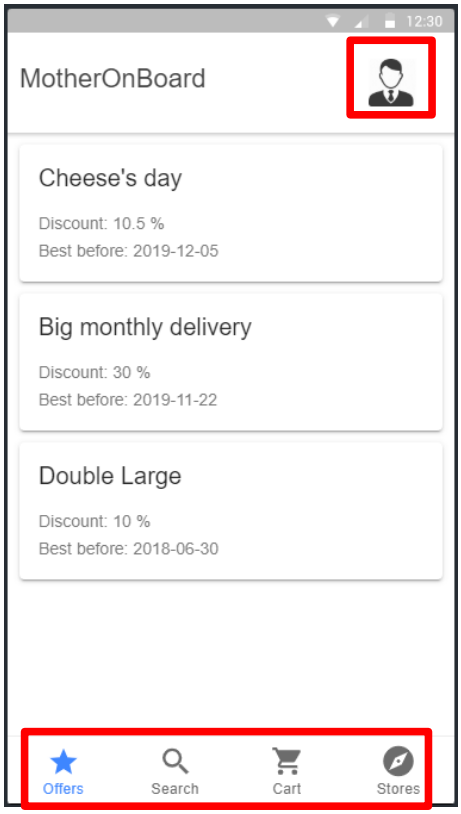

MotherOnBoard
Client:
Direction:
1
Cheese Burger
Normal
Cost: 3€
Note:

6. Manual de instalación y Manual de uso

Manual de instalación

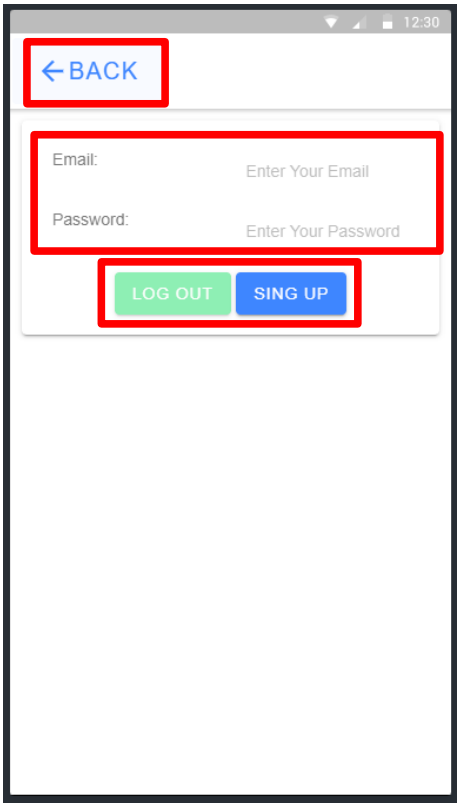
- Instalar la Base de datos: Seguir el video tutorial https://www.youtube.com/watch?v=z9QH14QJ1-k&t=472s
- Instalar el servidor (Node.js): Descargas el instalador correspondiente a tu sistema operativo https://nodejs.org/es/download/
- Instalar el cliente (Ionic Cordova): Accedes a tu consola de comandos e insertas ‘npm i ionic cordova’
- Crear un proyecto del cliente: Accedes a la consola de comandos de Node.js (Node.js command prompt) e insertas ‘ionic start’ el asistente te preguntara por el nombre del proyecto sera ‘Client’, el lenguaje ‘Angular’ y si deseamos empezar con una plantilla ‘tabs’.
- Importamos el proyecto: Copiamos el contenido de GitHub ‘Enlaza-Cliente’ en la dirección del proyecto.
- Instalamos las dependencias: En la consola de Node.js accedemos al proyecto ‘cd Client’ e instalamos las dependencias ‘npm i’
- Corremos la base de datos: En la el nombre de la base de datos clickamos botón derecho creamos un script. Una vez creamos abrimos el fichero ‘DDBB.sql’ del GitHub ’Enlaza’ y seleccionamos todo el documento excepto por la sección de ‘RESTART DATABASE’ y ‘DATA OBSERVATION’ y ejecutamos.
- Corremos el servidor: Para ello en la ruta ‘src\database\database.js’ del servidor y cambiamos los datos a su base de datos (usaurio, contraseña, nombre de la base de datos). Posteriormente, otra consola de Node.js accedemos a la carpeta del servidor (el contenido de GitHub ‘Enlaza’) y escribimos ‘npm run build’ (para convertirlo a una versión antigua sin problemas de compatibilidad) y lo iniciamos con ‘npm run start’ y esperamos a que aparezca en la consola ‘Hello World from the port 3000’.
- Corremos el Cliente: En la consola de Node.js que instalo las dependencias insertas ‘ionic serve -l’ y el asistente te preguntara si quieres instalar el ionic lab ‘yes’.

Manual de uso



Acceder a Login

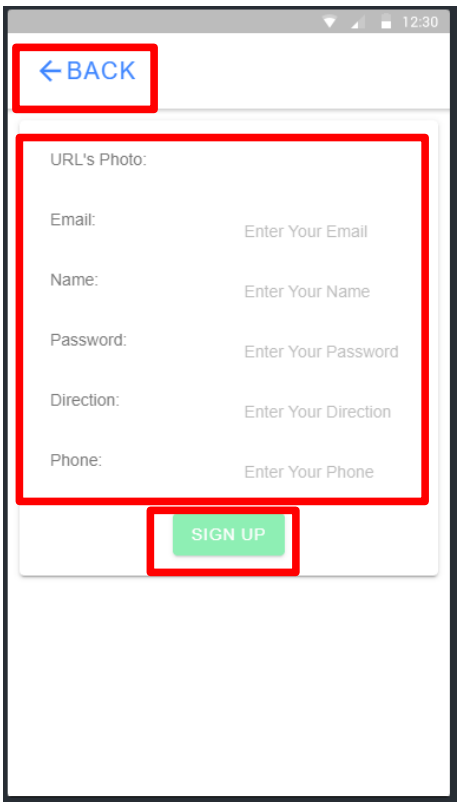
Acceder a otro Tab



Acceder al Tab anterior al acceder a Login

Introducir los datos para loguearse

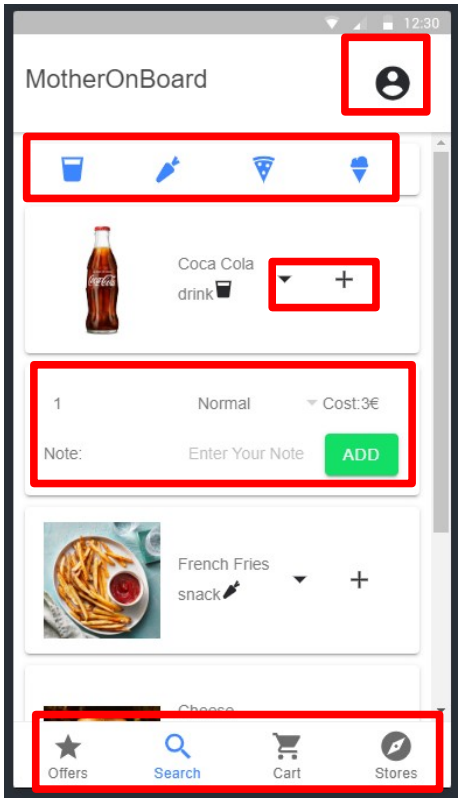
Sing Up para acceder a Sing Up y Sing In logearse.



Accedemos a Login

Introducir los datos para inscribirse

Inscribirse



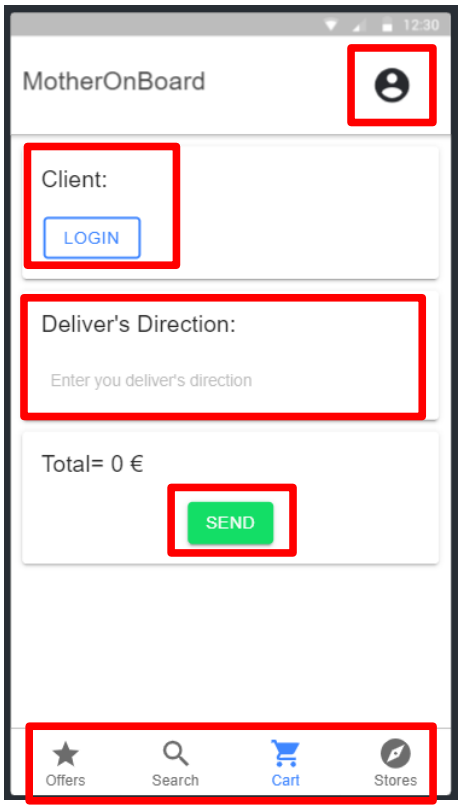
Accedemos a Login

Filtra por categorías los productos

Desplegar información del producto (Flecha) y desplegar la opción de compra (más)

Introducir las opciones de compra y ADD aceptar

Acceder a otro Tab



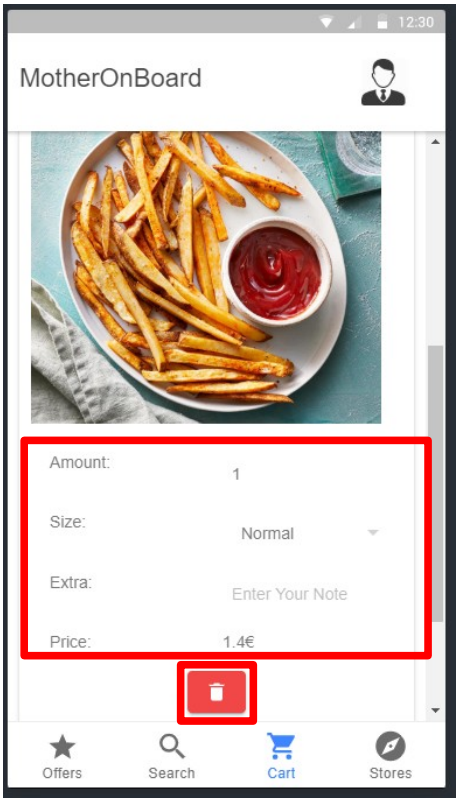
Accedemos a Login

Accedemos a Login

Introducir la dirección del pedido

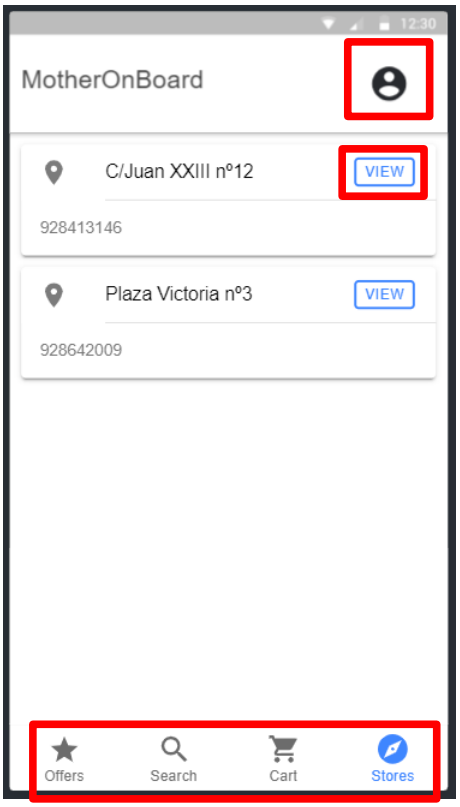
Enviar

Acceder a otro Tab



Modificar el pedido

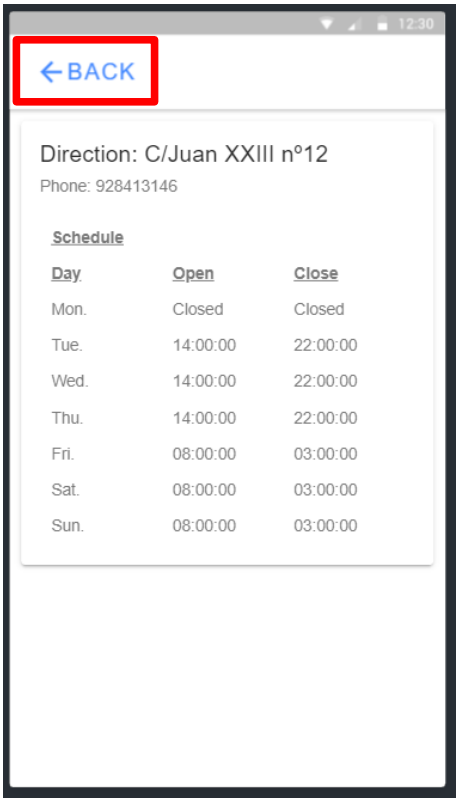
Eliminar Producto



Accedemos a Login

Accedemos Store

Acceder a otro Tab



Accedemos a Stores

7. Describir la implementación del código de la API REST

Postman: https://documenter.getpostman.com/view/8789945/SWE9ZwdM

8. Subir el código del proyecto completo por una repositorio

FrontEnd: https://github.com/SamaelIncubus/Enlaza-Client.git
BackEnd: https://github.com/SamaelIncubus/Enlaza.git

9.Usabilidad

Características

- Útil: La aplicación funciona según su objetivo; pudiendo crear pedidos a domicilio.
- Fácil de usar: La aplicación es fácil de usar pues es escueta en información y la poca que hay esta explicada y con pocas acciones es posible hacer un pedido.
- Fácil de Aprender: La información y la poca que hay esta explicada. Incluso en caso de error el usuario es avisado de lo que es necesario para realizar la acción que desea.
- Elegante en su diseño: El diseño aprovecha el espacio infinito digital. Teniendo limpieza, evitando la saturación. Utiliza colores tenues en el fondo para que resalte la interfaz y colores más vivos para la interfaz, con asociaciones familiares (rojo peligro, verde accesible). Se utiliza margenes y sombreados para diferenciar las partes de la interfaz.
- Eficiente: Solo con logearte y rellenar un formulario puedes crear un pedido. Acción que alternativamente se realiza llamando por teléfono, que en muchas ocasiones estando saturada la linea y después de la espera, tener que entenderte con el dependiente, normalmente con mucho ruido de fondo, para realizar el pedido.
- Beneficios: Ofrece beneficios económicos, pues no se requiere de un gasto de llamada constante, como también el tiempo de la llamada para realizar el pedido. Ofrece calidad pues, en la aplicación es fácil revisar tu pedido antes de confirmarlo, como modificarlo. Por ultimo da prestigio, pues la aplicación funciona correctamente y junto lo añadido anteriormente da prestigio.
- Previsión de errores: En la aplicación se bloquea todas las acciones que requieran de lorguarse anteriormente evitando posibles errores.
- Retroalimentación de la interfaz: La aplicación avisa al usuario de la necesita de loguarse cuando intente realizar una acción que requiera previamente loguarse
- Simplicidad en el diseño: La aplicación contiene escasa información pero la necesaria para el uso de esta

Usable

- El Usuario
  - Inicia acciones y controla tareas: Si puede crear pedidos, modificarlos, ...
  - Personaliza la interfaz: Puede personalizar su foto de perfil.
  - Interactuá con la interfaz: Si puede interactuar con la aplicación
  - Acceder a todo contenido: Puede acceder a la información que debería acceder y no más, pues es peligroso dejar que el usuario pueda acceder a toda la información
- Recuperación de errores: En términos generales se bloquea las acciones para evitar errores. Si el error es añadir productos al carrito de forma errónea o confundirse de cuenta esta puede corregirse (que es todo lo que se puede realizar en la aplicación)
- Estética de diseño: la interfaz clara, minimalista y colores están diseñado para que sea intuitivo la utilización de la interfaz

- Consistencia de la interfaz: Toda la interfaz sigue la misma “temática” las misma “normas” todas las secciones están divididas por margenes y sombras. Todo tiene los mismo colores. Los botones son estéticamente iguales, ...
- Simplicidad del diseño: El diseño es minimalista
- Retroalimentación: Se avisa al usuario si es necesario acciones previas para la acción que esta intentando acceder

Pautas resumen

- ¿La organización de los elementos de la interfaz es clara y consistente?: Si
- ¿Los elementos están economizados?: Si
- ¿Comunica, ajustando la presentación de contenidos a las capacidades del usuario?: Utilizamos colores y tamaños estándar para toda clase de públicos.

Pautas

- Diseño Visual: Las imágenes, iconos, mensajes ocultos y texto favorece la comunicación
- Color: El color utilizado es intuitivo usando el rojo para eliminar, el verde para añadir y el azul para obtener mas información.
- Matiz, contraste y resplandor: No utilizo ninguna personalización de colores. Uso los dado por defecto, lo cuales están diseñado para todo tipo de publico (como lo que tienen desorden visual)
- Disposición de las ventanas tipo formulario: Mis formulario están alineado su etiqueta del tipo de dato y la entrada del dato, siendo intuitivo.
- Redacción de texto en la interfaz: El texto que existe en la es el necesario para expresar información y que esta no sea confusa.

Principio básicos

- Familiaridad del usuario: No se innova en la interfaz, por tanto todo elemento es conocido por el usuario (tabs, butones, formularios, menu superiores)
- Consistencia: Cumple con la Consistencia de la interfaz
- Legibilidad: Todo texto es amplio y se encuentra junto al objeto de la interfaz relacionado
- Mínima sorpresa: Toda la interfaz es intuitiva, los tabs avisan de que contenido están mostrando, en la esquina superior el usuario que esta loqueado, …
- Recuperabilidad: Cumple con la Retroalimentación y la Recuperación de errores

Usuario final

- ¿El usuario está interesado en información precisa o en las relaciones entre los diferentes valores de los datos?: Da información precisa sobre los productos alimenticios.
- ¿Que rápido cambian los valores de la información?: Prácticamente no cambia toda la información presente siempre esta ahí solo que oculta, por lo que se desplaza pero nunca cambia.
- ¿Se informa al usuario cuando se cambia el valor?: Cuando este se loguea se le informa que ha sido logueado y se muestra su foto de perfil
- ¿Es necesario que el usuario lleve a cabo una acción por los cambios de la información?: No , a excepción que no este logueado o al loguarse el email o contraseña sean erroneos.
- ¿El usuario interactuá con la información de forma directa?: Si

Estructura

- Menús: Existe menús (los tabs y la parte superior de la aplicación)
- Ventanas: Existen ventanas el login, la información de la tienda y el inscribirse
- Cuadros de diálogos: Existen como los de errores, información del producto, ...
- Atajos de teclado: No existe para el móvil (que yo conozca)

Color

- ¿Es utilizado para comunicar ideas?: Si
- ¿Resalta los mensajes importantes?: Si, esta en rojo cuando existe un error
- ¿Crea un efecto psicológico en el ser humano?: Usamos código de colores (rojo: peligroso, verde: aceptar, ...)
- ¿Abusa del color?: No
- ¿Abusa del color en los mensajes?: No

Fuentes

- ¿Elige de forma adecuada la tipografía en la interfaz?: Si, no utilizo fuente difíciles de leer
- ¿Tamaño de la fuente es legible?: Si
- ¿No hay cuadros de texto extensos?: Si

Iconos

- ¿Usa iconos de forma equilibrada para la organización de la interfaz?: Si
- ¿Usa demasiados iconos?: No
- ¿El uso de iconos no entorpece la claridad de la interfaz?: No, entorpece
- ¿Hay combinación correcta entre texto e iconos?: Si

Interactivos

- ¿Todo elemento interactuable sabe el usuario conoce su interactuación?: Si
- ¿Se informa en procesos lentos que la aplicación esta trabajando y no colgada?: No hay acciones lentas
- ¿Se informa de errores, efectos y confirmaciones de las acciones?: Si

Datos

- ¿Están separados por funciones?: Si
- ¿Están repartidos de forma equilibrada?: Si
- ¿Existe simetría?: Si

Pantallas

- ¿Es conciso con la información?: Si
- ¿Es fácil la navegación?: Si
- ¿El contenido tiene estructura piramidal?: Si, la información más descriptiva es la primera que se ve
- ¿El lenguaje es cercano al usuario?: Si, no uso expresiones, ni dialectos
- ¿Los mensajes están separados, sin mezclarse?: Si

Seguridad

- El aseguramiento de la información que se maneja: Se asegura que toda la información salga de la base de datos online; por tanto siempre estarán correctos.
- Control de acceso: Si, solo los usuario pueden crear pedidos
- Criptografía: No
- Certificados y claves: Si, los usuarios tiene clave para poder loguarse

Elementos multimedia

- ¿Los elementos multimedia utilizados mejoran la retención de la información?: Si
- ¿Existe equilibrio entre los elementos multimedia y los que no?: Si

Principios

- ¿Cumple los principios básico de Organizar?: Si
- ¿Cumple los principios básico de Economizar?: Si
- ¿Cumple los principios básico de Comunicar?: Si
- ¿Es simple y estético?: Si
- ¿Cumple que sea accesible a todo usuario?: Si
- ¿Tiene un entorna agradable?: Si

Atributos

- Es fácil de aprender?: Si
- Eficiencia: Si
- Recuerdo de utilización en el tiempo: Si
- Tasas de error: Baja
- Satisfacción: Alta
- Beneficio de Usabilidad: Buena

10.Comparación de Tecnologías

Aplicación Híbrida

Ventajas

- Una buena parte de su desarrollo sirve para todas las plataformas, Android, Windows Phone, iOS, etc.
- Son accesibles desde un dispositivo móvil por medio de un icono en el escritorio.
- El desarrollo de aplicaciones híbridas es más sencillo, lleva menos tiempo y supone un menor coste.
- Gozan de mayor sostenibilidad. Las actualizaciones de la aplicación solo dependen de un desarrollo con pequeños cambios para cada plataforma.

Desventajas

- Se requiere su instalación en el dispositivo, ya que, aunque su diseño es simulado para parecer una aplicación nativa como lo son las cámaras, los sensores, etc., hay que descargarlas previamente.
- Si se trata de una aplicación muy compleja, es posible que la velocidad y la fluidez se vean mermadas.
- No aprovechan al 100 % las características del dispositivo, especialmente cuando se trata de la funcionalidad de videojuegos, la de 3D, etc.

- Consumen bastante espacio en el dispositivo del usuario.

Inonic

Ventajas

- Desde una única fuente podremos llegar a las plataformas que soporta este framework (Android e iOS).
- El desarrollo principal se realiza en HTML junto con CSS y JS, lenguajes muy extendidos por la comunidad de desarrolladores, con lo que la implantación de esta herramienta en la empresa, facilitará el desarrollo de proyectos de la forma más efectiva aun cuando la plantilla de desarrolladores contenga nuevas incorporaciones.
- Si ya contamos con una web app que queremos convertir en aplicación móvil, en la mayoría de los casos habremos hecho uso de JavaScript, por lo que el código es reutilizable.
- Para el caso de aplicaciones híbridas, tendremos con un único proceso de desarrollo e implementación, una app para Android, iOS y web.

Desventajas

- El rendimiento puede ser ligeramente menor que en aplicaciones desarrolladas de forma nativa, cosa que no debería ser un problema al menos que el proyecto sea para la creación de juegos con detallados gráficos u otras aplicaciones que hagan uso de grandes cantidades de recursos.

Angular

Ventajas

- Promovido por Google: Las aplicaciones de Google también utilizan este framework. Su estabilidad parece garantizada.
- TypeScript: Un frontend con Angular se crea utilizando el lenguaje TypeScript. Es un superscript para JavaScript, que garantiza una mayor seguridad. Este lenguaje facilita la depuración de errores al escribir el código o realizar tareas de mantenimiento.
- UI (User Interface) declarativa: Angular utiliza HTML para definir la UI de la aplicación. HTML, en comparación con JavaScript, es un lenguaje menos complicado. HTML es también un lenguaje declarativo e intuitivo. Con este método de trabajo no necesitamos dedicar tiempo a definir flujos de programa.
- POJO: Con Angular con frontend no se necesitan funciones adicionales de getter y setter. Dado que, cada objeto que utiliza es POJO (Plain Old Java Object). Esto implica que tenemos una definición de clases simples. Se permite la manipulación de objetos al proporcionar mediante JavaScript. Por supuesto podemos eliminar o agregar propiedades de los objetos. También es posible realizar bucles sobre estos objetos cuando sea necesario.
- Pruebas sencillas: Las pruebas son extremadamente simples. Los módulos soportan partes de la aplicación, que son fáciles de manipular. Con la separación de módulos se puede cargar los servicios necesarios, mientras se realizan las pruebas automáticas.
- Patrón MVC simplificado: Como ya se ha dicho, Angular está integrado con la configuración arquitectónica original del software MVC (Modelo-Vista-Controlador). Sin embargo, no cumple todas las normas establecidas. Angular requiere que los desarrolladores dividan una aplicación en diferentes componentes de MVC y construyan un código que pueda unirlos. Solamente se pide dividir la aplicación y Angular se encarga de todo lo demás
- Angular garantiza un desarrollo fácil, ya que elimina la necesidad de código innecesario. Tiene una arquitectura MVC simplificada.

Desventajas

- Es lento cuando se muestran grandes cantidades de datos.

Node.js

Ventajas

- OpenSource: sin duda, hoy el software open source ha tenido la evolución más rápida y con mejor aceptación de los usuarios, además que es totalmente gratis su uso, puede descargar el código fuente para analizarlo y crear tu propio proyecto de Open Source basado en este.
- Optimización de los recursos: Dada la naturaleza asíncrona de NodeJS nos permite tener una mucho mejor administración de los recursos.
- Desarrollo agiles: NodeJS permite crear aplicaciones de una forma simple y rápida, a diferencia de otros lenguajes de programación como Java o C#, que se requiere de una gran cantidad de clases para echar a volar un proyecto.
- FullStack: Dado que JavaScript se puede ejecutar del lado del servidor y del cliente, es posible crear aplicaciones de BackEnd y FrontEnd con una sola tecnología.
- Modularidad: Como ya lo hablamos, NodeJS es extremadamente modular, lo cual permite utilizar únicamente los módulos requeridos sin traernos nada de más.

Desventajas

- Tipos dinámicos: JavaScript permite enviar como parámetro un objeto que no tiene definición, solo abres y cierres llaves {} y dentro pones lo que sea, lo que hace muy complicado saber que valores puedes enviar. Si eres programador JavaScript seguramente me quedarás colgar vivo diciendo que eso es bueno.
- Callbacks: Si bien, como ya analizamos, esto hace que se utilicen mejor los recursos, la verdad es que las callbacks son un dolor de cabeza, ya que tener funciones para cada paso es incómodo y hace muy complicado seguir el código.
- Refactor: Si alguna vez has hecho un refactor importante en un proyecto de JavaScript seguro me entenderás, pues casi nunca es posible realizar uno de forma segura, pues el tipeado débil y tipos dinámicos hacen que sea complicado para el IDE determinar cómo refactorizar.
- Librerías estándar: Esta es a mi ver una de las más fuertes, es verdad que existen librerías para todos los problemas que se te puedan ocurrir, pero el problema es que existen muchas opciones para lo mismo, haciendo complicado elegir entre una u otra. Adicional que cada una avanza a un ritmo separado en vez de unir fuerzas para mejorar una versión estándar.
- Arquitectura: En mi experiencia las aplicaciones NodeJS tiene un nivel de arquitectura muy por debajo de los lenguajes de programación tradicional como Java o C# pues en NodeJS todo se basa en funciones que son enviadas y exportadas de clases. Y es raro ver que se implementan patrones arquitectónicos o patrones de diseño.

Sequelize

Ventajas

- Es necesario poco código para que funcione
- No necesitas conocer el lenguaje SQL para manejarlo
- Puede trabajar con cualquier motor de base de datos
- Es en su mayoría automatizado siendo innecesario ampliás implementaciones
- Se puede exportar sin complicaciones a todo tipo de ficheros

Desventajas

- Las sentencias complejas son lentas
- La documentación existente es muy pobre
- No tiene flexibilidad

PostgreSQL

Ventajas

- Instalación ilimitada y gratuita: Podemos instalarlo en todos los equipos que queramos. Independientemente de la plataforma y la arquitectura que usemos, PostgreSQL está disponible para los diferentes SO, Unix, Linux y Windows, en 32 y 64 bits. Ésto hace de PostgreSQL un sistema multiplataforma y también hace que sea más rentable con instalaciones a gran escala.
- Gran escalabilidad: Nos permite configurar PostgreSQL en cada equipo según el hardware. Por lo que es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima. Con ello logramos una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.
- Estabilidad y confiabilidad: No se han presentado nunca caídas de la base de datos. Ésto es debido a su capacidad de establecer un entorno de Alta disponibilidad y gracias a Hot-Standby, que nos permite que los clientes puedan realizar consultas de solo lectura mientras que los servidores están en modo de recuperación o espera. Así podemos hacer tareas de mantenimiento o recuperación sin bloquear completamente el sistema.
- pgAdmin: Se trata de una herramienta gráfica con la que podemos administrar nuestras bases de datos de forma fácil e intuitiva. Podemos ejecutar sentencias SQL, e incluso crear copias de seguridad o realizar tareas de mantenimiento.
- Estándar SQL: implementa casi todas las funcionalidades del estándar ISO/IEC 9075:2011, así pues, resulta sencillo realizar consultas e incluir scripts de otros Motores de Bases de Datos.
- Potencia y Robustez: PostgreSQL cumple en su totalidad con la característica ACID Compliant. ACID es un acrónimo de Atomicity, Consistency, Isolation y Durability (Atomicidad, Consistencia, Aislamiento y Durabilidad en español). Por ello permite que las transacciones no interfieran unas con otras. Con ello se garantiza la información de las Bases de Datos y que los datos perduren en el sistema.
- Extensibilidad: tenemos a nuestra disponibilidad una gran variedad de extensiones distribuidas por el grupo de desarrolladores de PostgreSQL. También por terceros o incluso nosotros mismos podemos crear nuestras propias extensiones. Éstas extensiones pueden ser lenguajes de programación, tales como, Perl, Java, Python, C++ y muchos más.

Desventajas

- Es relativamente lento en inserciones y actualizaciones en bases de datos pequeñas, PostgreSQL está diseñado para ambientes de alto volumen. Ésto hace que la velocidad de respuesta pueda parecer lenta en comparación con bases de datos de pequeño tamaño.
- Soporte oficial: No cuenta con un soporte en línea o telefónico. PostgreSQL cuenta con foros oficiales donde los usuarios pueden exponer sus dudas que responden otros usuarios de la comunidad. También, disponemos soporte empresarial como EnterpriseDB o TodoPostgreSQL. Cabe resaltar que la comunidad de usuarios PostgreSQL es una de las más activas en el mercado.
- La sintaxis de algunos de sus comando o sentencias puede llegar a no ser intuitiva si no tienes un nivel medio de conocimientos en lenguaje SQL.

11.Planificación

El Modelo de datos, lo realizamos en conjunto, debatiendo los puntos más difusos. Ya realizado, Nos asignamos por Front-End (Pablo) y Back-End (Roberto). Si teníamos dudas/estamos atascados nos ayudamos mutuamente, si persistia la situación esperábamos a Tiburcio para resolver el problema. Cuando Roberto termino de realizar el Back-End procedió a ayudar en el Front-End (mientras Pablo realizaba los Test y los Reports, Roberto se encargo de ampliar la interfaz y asociarla al Back-End).

12.Conclusión

El Proyecto Enlaza nos ayudo a formarnos y ponernos en contacto con en el mercado; hecho que se aprecia por la dificultada de encontrar experiencia laboral siendo tan jóvenes y es una formación extra y la más útil que existe; la cual no están obligados a dar y por tanto estamos muy agradecidos.

Pero por el otro lado, ha sido un caos total: la rubrica no esta bien narrada, había cosas repetidas, cosas sin sentido, muy especificas, … Lo peor, cada profesor interpretaba esta a su forma e incluso el mismo profesor cambiaba de parecer sobre el mismo punto. Hechos que no se informaban de forma adecuada. Añadido a que no vimos supervisión alguna por parte del profesorado (la cual se había establecido que pasarían semanalmente a revisar el trabajo realizado por el alumnado) y al escaso tiempo para realizar un proyecto tan grande junto a las otras unidades obligatorias que hemos de evaluar, el proyecto a resultado muy agobiante y estresante (hemos tenido un único descanso en navidades de 3 días, de resto ni un día libre y nos ha sido imposible llegar a los mínimos).

Por tanto, para evitar el agobio del alumnado y del profesorado (del cual también somo cocientes), deberían organizar la próxima promoción Enlaza mejor: Estableciendo un rubrica por el profesorado de dicho curso no reutilizar la anterior (o al menos reciclarla) y establecer plazos de tiempo realistas no pretender dar la teoría de un trimestre entero en un mes, por ejemplo.

Personalmente damos gracias a que el profesorado a sido consiente a la situación y han sido flexibles en todo lo que se les han propuestos y no hemos llegado a ninguna confrontación.

13.Enlaces y Referencias

Videos-Tutoriales de PostgreSQL: https://www.youtube.com/watch?v=jxIEDKzGrOs&list=PL8gxzfBmzgex2nuVanqvxoTXTPovVSwi2

Video-Tutorial del Back-End con ORM: https://www.youtube.com/watch?v=sA3t4d1v7OI

Video-Tutoriales de Ionic: https://www.youtube.com/watch?v=RWCb_ARrSxM&list=PLCKuOXG0bPi2EGYmUq7eidFV8A95xTjEx

Documentación oficial de Ionic: https://ionicframework.com/docs/components

Pagina oficial de JsReport: https://jsreport.net/on-prem

Video-Tutorial de JsReport: https://www.youtube.com/watch?v=11ZStDn87pk&t=

Criterios a seguir: http://www3.gobiernodecanarias.org/medusa/eforma/campus/course/view.php?