

DOCUMENTACIÓN

FRANCISCO JOSÉ
SEIJO ORENES

OMNIBANK

ÍNDICE

INTRODUCCIÓN	2
PROBLEMAS	2
OBJETIVOS	2
MEDIOS TÉCNICOS Y RECURSOS NECESARIOS	3
TECNOLOGÍAS A UTILIZAR	3
TAREAS PROGRAMADAS INICIALMENTE	4
TEMPORALIZACIÓN	5
DISEÑO	6
ARQUITECTURA	11
ARQUITECTURA BASE DE DATOS	11
ARQUITECTURA APP	12
DESPLIEGUE EN HEROKU	14
EJEMPLO DE SESIÓN	25
APLICACIÓN WEB PROGRESIVA (PWA)	27
INSTALACIÓN PWA	28
PROBLEMAS SURGIDOS QUE LLEVAN A CAMBIOS EN LA PLANIFICACIÓN	33
VERSIONES	35
MEJORAS FUTURAS	38
CONCLUSIONES	38
BIBLIOGRAFÍA:	39
ANEXOS	40

INTRODUCCIÓN

Omnibank es una app que nos permitirá consultar nuestros movimientos y acciones bancarias de manera omnicanal desde la plataforma que más nos convenga en cada momento, para poder así ser utilizada como cada cliente quiera y le sea más sencillo. Recopilará las estadísticas de uso de cada usuario y las enviará a una base de datos de donde se podrán extraer perfiles detallados de cada usuario.

PROBLEMAS

Los clientes de bancos de una avanzada edad no suelen usar las aplicaciones de estos porque no entienden su funcionamiento, o por qué tienen que usar X o Y aplicación para diferentes operaciones.

En vez de aprender a utilizarlas, acaban acudiendo a su sucursal más cercana a consultar sus movimientos o a hacer traspasos sencillos que hubieran sido mucho más sencillos si hubieran optado por la alternativa digital.

Además, debido a la reciente pandemia que todos hemos vivido, agilizar el tráfico de las oficinas físicas y no tenerlas llenas de gente que va a realizar trámites cotidianos, se convierte en una prioridad.

También, todas las empresas necesitan hoy en día tener perfiles detallados de sus clientes para poder ofrecerles productos o servicios que les sean más interesantes.

OBJETIVOS

Omnibank busca dar solución a los problemas previamente planteados ofreciendo diferentes caminos para que todos los usuarios puedan utilizar el que consideren mejor.

Omnibank dispondría de una app para Android, que se adaptaría de una aplicación WEB, y también un chatbot que tendría las mismas funciones que las anteriores.

Desde cualquiera de las plataformas, Omnibank accedería a la base de datos del banco, y realizaría las consultas y movimientos pertinentes, y una vez finalizadas, le comunicará al cliente el éxito o los inconvenientes que hubieran surgido de la operación.

Además, Omnibank recopilará las estadísticas de uso del cliente, y las enviará a una CDP de las que se podrán extraer perfiles de clientes.

MEDIOS TÉCNICOS Y RECURSOS NECESARIOS

Para el desarrollo del proyecto, necesitaré un equipo lo suficientemente potente para realizar varias tareas pesadas simultáneamente con el fin de agilizar el proceso, como alojar un servidor, lanzar varias máquinas virtuales a la vez para hacer pruebas y para utilizar todos los programas actuales sin problema.

Una vez se desarrollara por completo el producto, tendré que buscar algún cliente que quiera comprarlo (bancos potencialmente), me facilitara el acceso a su base de datos y un alojamiento web donde desplegar la aplicación una vez se adapte a las credenciales otorgadas (no todas las bases de datos de todos los bancos van a ser iguales).

Como usuario de la aplicación, necesitaría un equipo que me permitiera ejecutar navegadores actuales que sean compatibles con el producto(en principio todos), una cuenta en el banco y una conexión a internet. Dado que la mayoría de operaciones se ejecutarán en el servidor, no hará falta que el equipo del usuario sea muy potente.

TECNOLOGÍAS A UTILIZAR

Para toda la parte del backend, que serán la base de datos y los scripts en PHP utilizaré el paquete de aplicaciones XAMPP, que hemos utilizado durante todo el curso, que viene con Apache, MySQL y PHP.

Para que Omnibank se convierta en una aplicación web de verdad, tendrá que ser accesible desde cualquier lugar y no solo en local, así que necesitaré un hosting que me permita mantener mi aplicación operativa mientras dure el proyecto.

Después de mucho trabajo de investigación, de mirar decenas de hostings gratuitos, elegí Heroku porque tiene una interfaz limpia, muy cuidada e intuitiva, además de obligarme a aprender a usar Git bien, lo cual durante el desarrollo del proyecto ha marcado un antes y un después en la calidad del mismo.

Heroku ofrece 500mb de espacio en la nube para poder subir tus aplicaciones, soporta la mayoría de los lenguajes de programación que se usan actualmente (Java, PHP, Python, Go), y dispone de herramientas muy útiles y fáciles de comprender para desarrolladores novatos, así que es el hosting que mejor se adapta a las necesidades del proyecto.

Para conectar la base de datos a la aplicación, Heroku dispone de infinidad de plugins para todas las bases de datos que se utilizan. Como quiero importar el trabajo en local que he hecho en MySQL, he añadido al hosting el plugin ClearDB MySQL, que me permite hacer justo lo que quiero.

Para la parte del frontend, que será la aplicación WEB, utilizaré JavaScript, HTML y CSS, que también hemos visto durante el curso.

Para la app android, convertiré la aplicación WEB en una aplicación WEB progresiva, utilizando chrome de android.

Para el envío de estadísticas de uso del cliente, utilizaré la API del CDP al que se lo enviaré, en mi caso, el CDP de mi empresa de las FCTs WhenWhyHow.

TAREAS PROGRAMADAS INICIALMENTE

Al inicio del proyecto, Omnibank estaba pensada para ofrecer una gran cantidad de funcionalidades, tales como permitir la consulta y realización de movimientos bancarios, ser accesible desde la web, un chatbot y una app móvil.

Estaban planeadas:

- La creación de las páginas tanto del login como de la aplicación.
- La transformación de una base de datos no relacional proporcionada por el cliente (en mi caso, mi empresa de las FCTs) a una relacional compatible con el backend de la aplicación.
- La conexión de la base de datos con las páginas de la aplicación.
- La creación de las páginas de estilos para dar un aspecto presentable a la aplicación.
- La exportación a Aplicación Web Progresiva para su instalación en dispositivos móviles.
- La creación de un chatbot que interactuara con la aplicación.
- El despliegue de la aplicación en un servicio de hosting.
- El envío de los datos de sesión de los usuarios a una plataforma CDP.

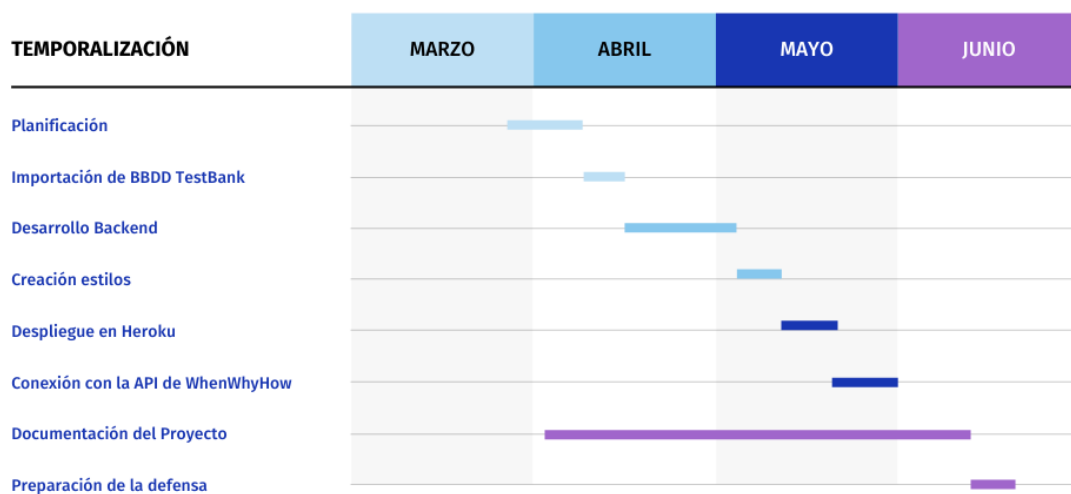
Es un proyecto ambicioso para la cantidad tan reducida de tiempo que se dispone para el mismo, motivo por el cual, se ha tenido que prescindir de algunas tareas más complicadas y simplificar otras.

TEMPORALIZACIÓN

Para hacer la temporalización del proyecto, he utilizado CANVA, una herramienta online gratuita que nos permite crear diagramas de manera muy sencilla, rápida e intuitiva.

Francisco Seijo

OMNIBANK



El proyecto comenzó exactamente el 28 de marzo y finalizará del 16 al 18 de junio, con la defensa del mismo.

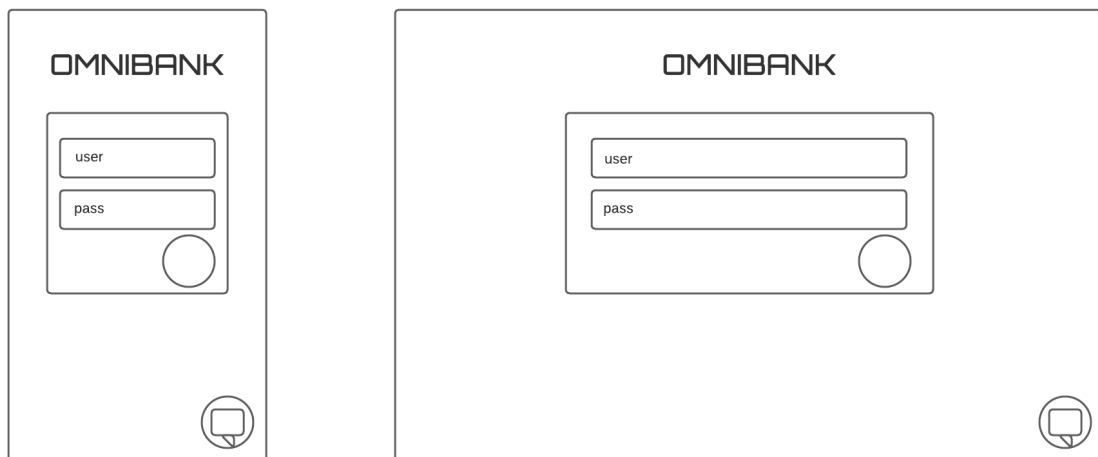
DISEÑO

Omnibank es una app orientada a la sencillez y accesibilidad de su utilización, así que he optado por una interfaz clara y fácil de entender con botones grandes y funciones útiles que los usuarios menos entendidos pueden considerar más imprescindibles.

El chatbot estará siempre disponible en todas y cada una de las pantallas en el mismo sitio, por si en cualquier momento el usuario lo necesita. Este podrá acceder a todas las funcionalidades de la aplicación desde el chatbot si lo prefiere.

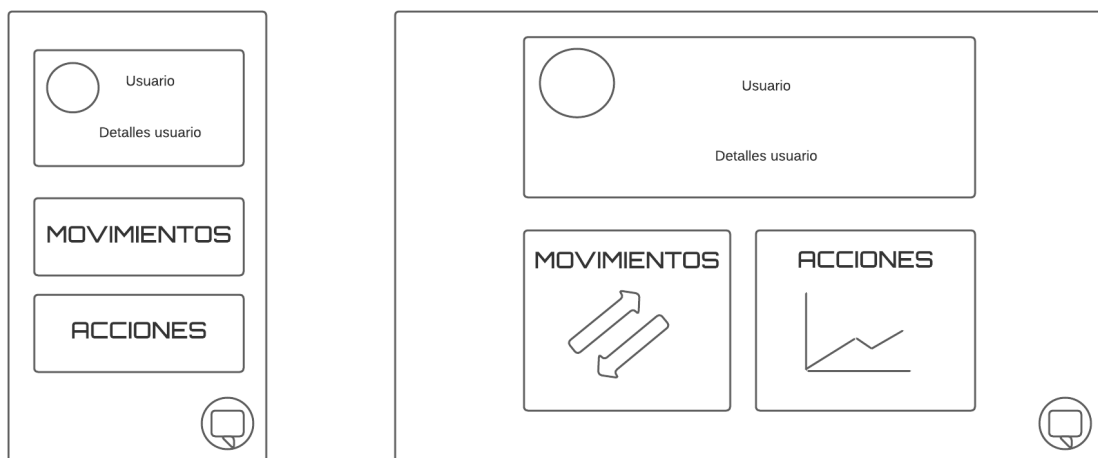
La pantalla de login tendrá el logo del banco que sea (OMNIBANK estará de placeholder hasta que algún cliente adquiera la aplicación), y un bloque donde poner el usuario y la contraseña junto con un botón de enviar. Cuando se introduzcan credenciales inválidas, saldrá un mensaje de error, y cuando se introduzcan credenciales válidas se accederá a la página principal de la aplicación.

PANTALLA DE LOGIN



La pantalla principal consistirá de tres bloques, uno en la parte superior de la pantalla con información del usuario que haya accedido, con su fotografía, su nombre, su número de cuenta, etc. y dos bloques más que serán dos botones que llevarán a la página de movimientos y a la página de acciones respectivamente. Ambos en su versión de escritorio contarán también con dos imágenes representativas de cada apartado.

PANTALLA PRINCIPAL




Finalmente, las dos pantallas restantes, de movimientos y acciones serán muy similares, contando con pestañas desplegables con la información del usuario sobre sus movimientos y sobre sus acciones, cuando las pestañas están sin desplegar mostrarán la información importante que identificará los trámites, como la fecha y el concepto de los movimientos o la empresa de la que se poseen acciones y la diferencia que se ha conseguido, y cuando se desplieguen, podremos ver el importe de los movimientos, el remitente/destinatario o datos como la inversión inicial y el porcentaje diferente en el caso de las acciones.

PANTALLA MOVIMIENTOS


MOVIMIENTOS

DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
Destinatario	Importe	▲
DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼



MOVIMIENTOS


DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
DD-MM-YY	Concepto	▼
Destinatario	Importe	▲
DD-MM-YY	Concepto	▼



PANTALLA ACCIONES


ACCIONES

Empresa	Diferencia	▼
Empresa	Diferencia	▼
Inversión	% Diferencia	▲
Empresa	Diferencia	▼
Empresa	Diferencia	▼

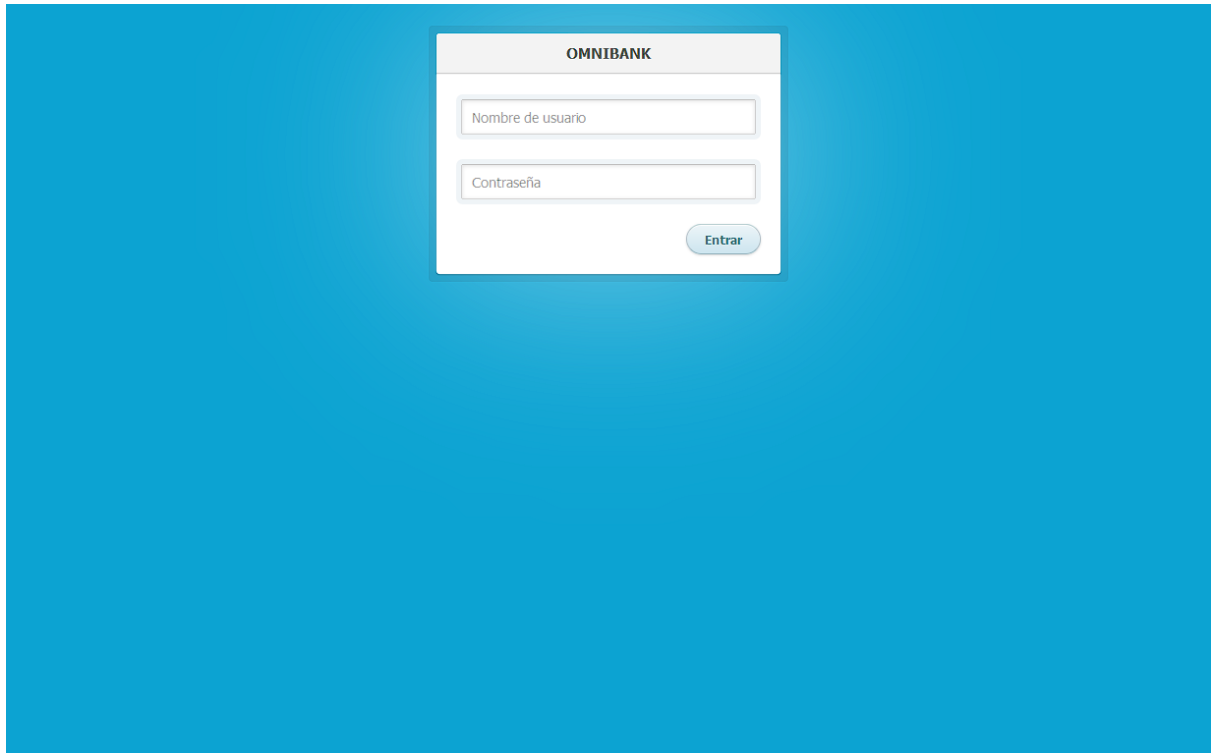


ACCIONES

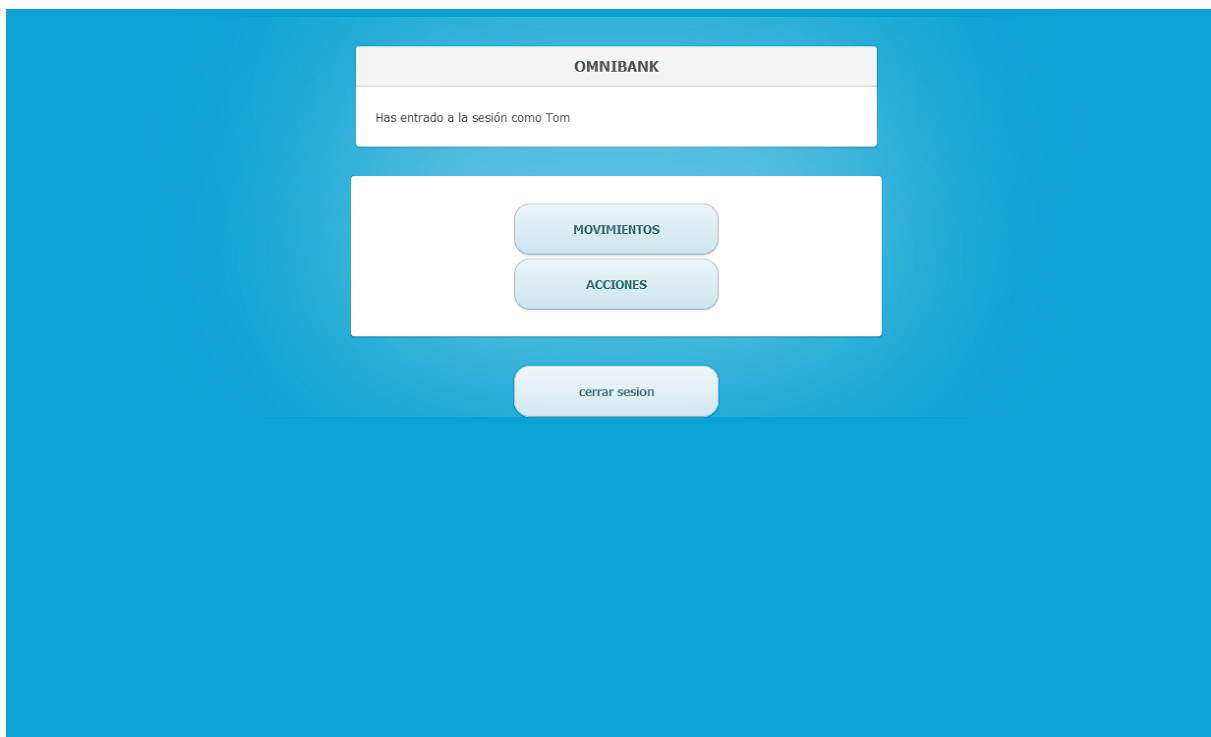
Empresa	Diferencia	▼
Empresa	Diferencia	▼
Empresa	Diferencia	▼
Empresa	Diferencia	▼
Inversión	% Diferencia	▲
Empresa	Diferencia	▼



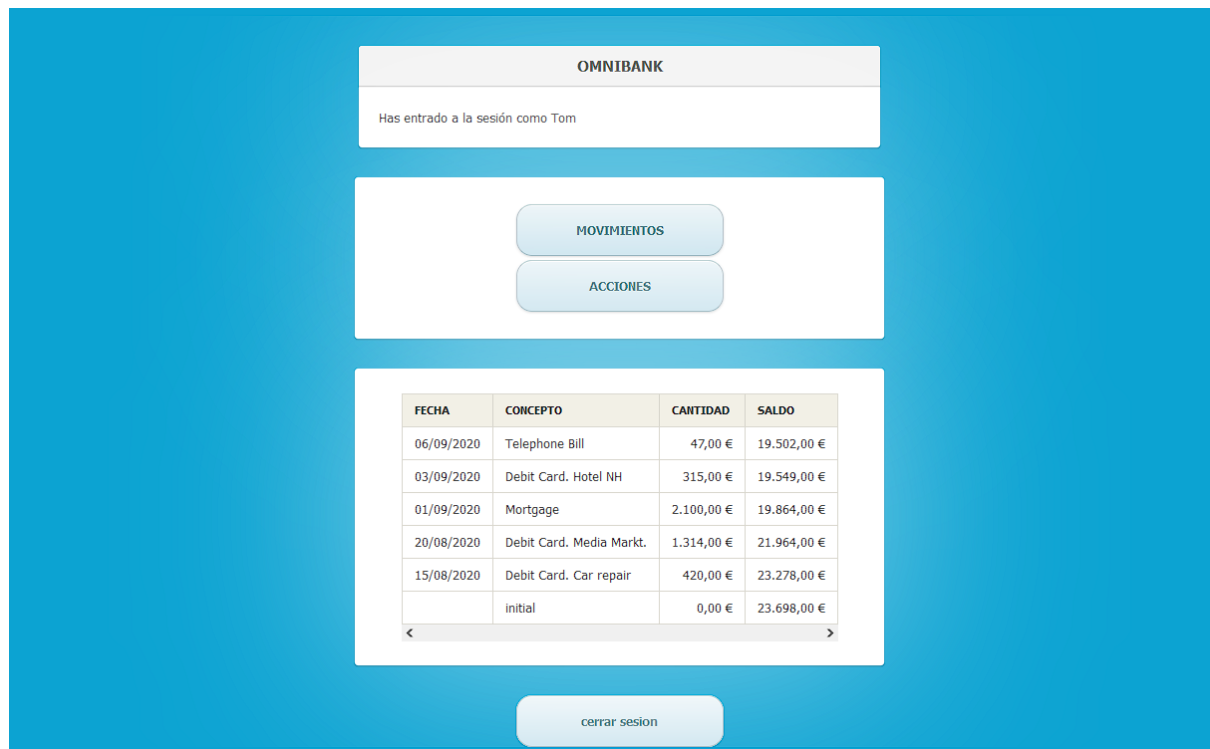
Aunque en un primer momento, Omnibank estaba diseñada para lucir así, debido al ajustado tiempo, ha terminado con esta apariencia:



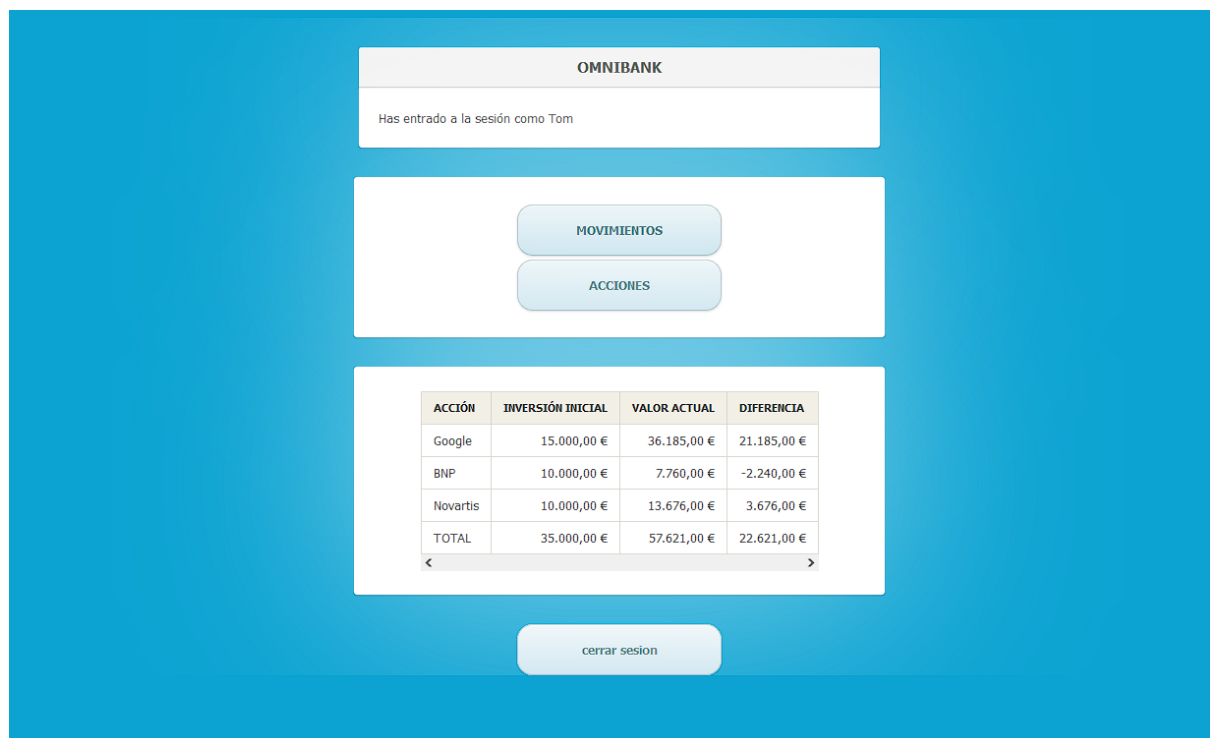
Pantalla de login



Pantalla principal



Pantalla principal - Movimientos



Pantalla principal - Acciones

ARQUITECTURA

ARQUITECTURA BASE DE DATOS

Para la base de datos me proporcionaron una en WhenWhyHow de un banco de prueba, para MongoDB y tuve que exportarla a MySQL y cambiarla un poco para adaptarla a la app.

La base de datos final consiste de tres tablas (customers, balance_info, stock_info)

customers (contiene la información de los usuarios)

customer_id (id de usuario)(varchar,4)**PK**

customer_name (nombre de usuario)(varchar,20)

customer_pin (contraseña de usuario)(int,11)

balance_info (contiene la información de los movimientos)

balance_info_id (id de movimiento)(int,11)**PK**

balance_info_customer (id de usuario que hizo el movimiento)(varchar,4)**FK**

balance_info_date (fecha del movimiento)(int,11)

balance_info_concept (concepto del movimiento)(varchar,50)

balance_info_amount (cantidad del movimiento)(float,15.2)

balance_info_remaining (cantidad restante tras el movimiento)(float,15.2)

stock_info (contiene la información de las acciones)

stock_info_id (id de la inversión)(int,11)**PK**

stock_info_customer (id de usuario que hizo la inversión)(varchar,4)**FK**

stock_info_share (empresa a la que se invierte)(varchar,30)

stock_info_investment (cantidad de la inversión)(float,15.2)

stock_info_current (valor actual de las acciones)(float,15.2)

stock_info_difference (diferencia de lo invertido con el valor actual)(float,15.2)

ARQUITECTURA APP

La arquitectura de mi proyecto está basada en el modelo MVC (Model, View, Controller) que divide el código según su función en la aplicación.

En el modelo estará el código que trabaja con los datos de la aplicación, tendremos mecanismos para acceder a la información, actualizar su estado, etc.

En la vista estará el código que va a producir la visualización de las interfaces de usuario, lo que renderiza los estados de nuestra app en HTML. Básicamente, el código que nos permite mostrar la salida.

El control será la capa que comunicará con el resto, donde se llamarán a los métodos del modelo y la vista para construir finalmente nuestra aplicación.

Adicionalmente a estas 3 carpetas, he añadido también una carpeta “styles” donde estarán localizados los archivos CSS de las páginas de la aplicación.

Mientras que en la raíz se encontrarán los archivos necesarios para el despliegue de la misma.

Esta arquitectura me ha permitido generar un código mucho más ordenado y accesible, y me ha facilitado en gran cantidad de ocasiones la resolución de problemas y la adición de nuevas funcionalidades.

En cuanto al código, he utilizado como base lo que hemos aprendido en la aplicación que hicimos en el módulo de Desarrollo en Entorno Servidor, tanto en el proyecto de GESVENTA que desarrollamos en el primer trimestre como en el proyecto de la API REST que desarrollamos en el segundo trimestre.

He utilizado también las expresiones regulares para transformar algunas salidas y he utilizado PHP_Curl para consumir la API de WhenWhyHow.

Adicionalmente, he preparado una página en local que realiza peticiones a la API para probar que efectivamente, se mandan los datos a la misma.

He generado una visualización en árbol con la terminal de Windows para mostrar cómo están organizados los archivos y carpetas. Voy a utilizarlo para comentar el contenido de cada archivo.

omnibank/

- | bank.png (icono de la app para la PWA)
- | composer.json (archivo necesario para el despliegue en Heroku)
- | index.php (página inicial de la app, te redirecciona al login)
- | manifest.json (archivo necesario para importar la app a PWA)
- |
- +---control
 - | login.php (página de login de la app, se inicia la sesión aquí y se piden un usuario y una contraseña, se comprueban con los parámetros de la base de datos y si son correctos se pasa a la siguiente pantalla y si no, se da la opción de reintentar y se suma 1 al contador de intentos)
 - | panel.php (página principal de la app, se muestra una tarjeta con el nombre del usuario que indica que se ha entrado correctamente y se muestra una tarjeta con dos botones que al ser pulsados muestran sus respectivas tablas del usuario que entró a la app, y finalmente se muestra también un botón para cerrar sesión y mandar toda la información de la sesión a la CDP de WhenWhyHow por medio de su API)
- |
- +---model
 - | conn.php (archivo donde se crea la conexión a la base de datos con los parámetros de "testbank.inc.php")
 - | counters.php (archivo donde se establecen los métodos de los contadores de la sesión)
 - | schemaDAO.inc.php (archivo donde se establecen los métodos que extraen datos de la base de datos y los devuelven para usarse en la app)
 - | testbank.inc.php (archivo donde se encuentran los parámetros de la base de datos)
 - | wwapi.php (archivo donde se establecen los métodos para consumir la API de WhenWhyHow)
- |
- +---styles
 - | login.css (estilos de login.php)
 - | panel.css (estilos de panel.css)
- |
- \---view
 - | vista.inc.php (archivo donde se establecen los métodos que generan las vistas de la aplicación)

DESPLIEGUE EN HEROKU

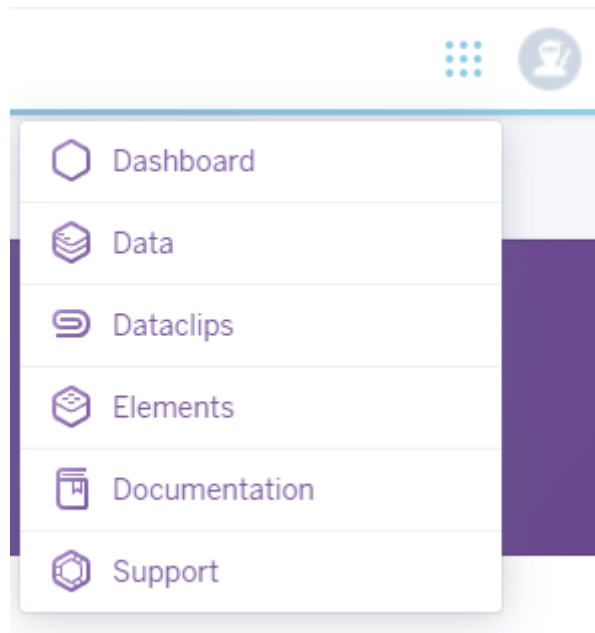
Para el despliegue de la aplicación en Heroku, busqué bastante documentación y al final encontré un videotutorial muy bueno donde explicaba a la perfección el procedimiento. El tutorial es de DoableDanny, un programador inglés que se dedica a subir tutoriales de diferentes tecnologías, herramientas y lenguajes de programación. Aparte del vídeo enseñando cómo se hacía, publicó un post en su página web con los pasos a seguir, así que voy a adaptarlo al castellano y poner capturas de como subí yo mi proyecto.

PASO1: Crear una cuenta en Heroku y descargar Heroku CLI.

Vamos a la página web de Heroku, pinchamos en registrarnos y nos saldrá una pantalla de registro en la que debemos introducir nuestros datos.

Nos enviará un correo de verificación a la dirección que hayamos introducido, abrimos el link que hay dentro y nos pedirá que introduzcamos una contraseña y finalmente podremos entrar a Heroku.

Una vez dentro, pincharemos en el desplegable la parte superior derecha y seleccionaremos la pestaña de Documentación.



Y en la página de documentación, pincharemos en “The Heroku CLI”.

Heroku Essentials

- ⚡ [Get Started](#)
- 📖 [The Heroku CLI](#)
- 📖 [Deploying with Git](#)
- 📖 [The Procfile](#)
- 📖 [Configuration and Config Vars](#)

Bajamos un poco y clicamos en el instalador que se adapte a nuestro equipo.:

Install with an Installer



The Windows installers display a warning titled “Windows protected your PC” to some users. To run the installation when this warning shows, click “More info”, verify the publisher as “salesforce.com, inc”, then click the “Run anyway” button.



Snap installs are no longer supported. Please use another install method below.

🍏 macOS

```
$ brew tap heroku/brew && brew install heroku
```

🪟 Windows

Download the appropriate installer for your Windows installation:

64-bit installer

32-bit installer

PASO1.5: Instalar Git

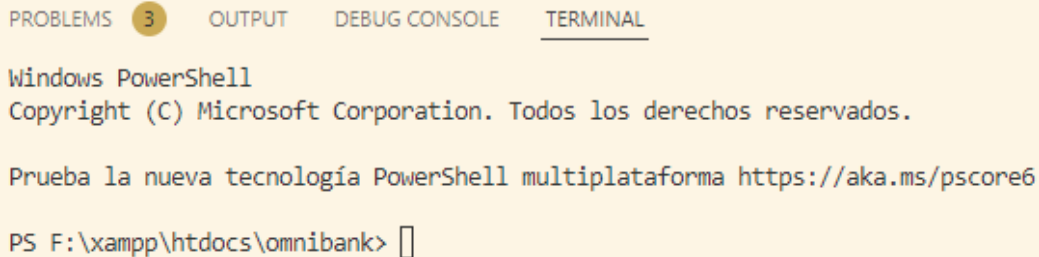
En la guía original no lo pone, pero hace falta tener instalado Git. Para ello simplemente vamos a la página oficial y nos lo descargamos como cualquier otro programa.



PASO2: Abrir la carpeta del proyecto en la terminal

Para utilizar tanto la CLI de Heroku como Git, necesitaremos una terminal, podemos usar la de Windows por ejemplo pero yo voy a usar la del VSCode porque uso este IDE para programar y quiero tenerlo todo junto.

En VSCode abrimos una nueva terminal desde la barra de herramientas o pulsando Ctrl+Shift+Ñ, y tendremos una terminal en la que podremos navegar hasta la carpeta del proyecto, en mi caso la tengo en la ruta F:\xampp\htdocs\omnibank/



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

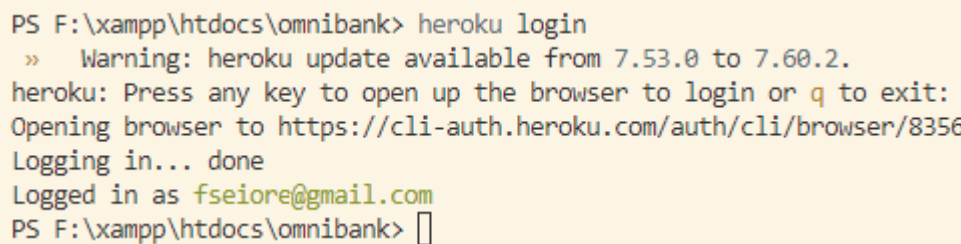
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS F:\xampp\htdocs\omnibank> 
```

PASO3: Entrar en Heroku desde la terminal

Simplemente utilizamos el comando *heroku login* y se abrirá una ventana del navegador para sincronizar nuestra cuenta.



```
PS F:\xampp\htdocs\omnibank> heroku login
» Warning: heroku update available from 7.53.0 to 7.60.2.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/8356
Logging in... done
Logged in as fseiore@gmail.com
PS F:\xampp\htdocs\omnibank> 
```

PASO4: Crear un archivo index.php

Para este proyecto, he decidido la página principal fuera el login de mi aplicación, así que quedaría así:

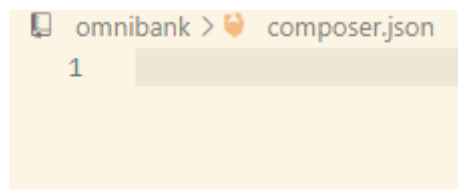
```
<!DOCTYPE html>
<html>
<head>
|   <link rel='manifest' href='manifest.json'>
</head>
<?php
|   if (!empty($_SERVER['HTTPS']) && ('on' == $_SERVER['HTTPS'])) {
|       $uri = 'https://';
|   } else {
|       $uri = 'http://';
|   }
|   $uri .= $_SERVER['HTTP_HOST'];
|   header('Location: '.$uri.'/control/login.php');
|   exit;
?>
```

El código lo genera automáticamente Heroku si no incluyes un index.php y puedes cambiar el header a la que quieras que sea la pantalla de inicio.

La parte de HTML es para la parte de la PWA de la que comentaré más tarde.

PASO5: Crear un archivo composer.json

Todos los proyectos de Heroku necesitan uno, así que creamos uno en la raíz del proyecto aunque lo dejemos vacío.



PASO6: Inicializar un repositorio Git, añadir todo y luego hacer commit.

Para utilizar Git, tendremos que inicializarlo en la carpeta de nuestro proyecto utilizando el comando *git init* en la raíz de la carpeta de nuestro proyecto, esto creará una carpeta oculta *.git* que contendrá todos los archivos de la herramienta.

```
PS F:\xampp\htdocs\omnibank> git init
Reinitialized existing Git repository in F:/xampp/htdocs/omnibank/.git/
PS F:\xampp\htdocs\omnibank> █
```

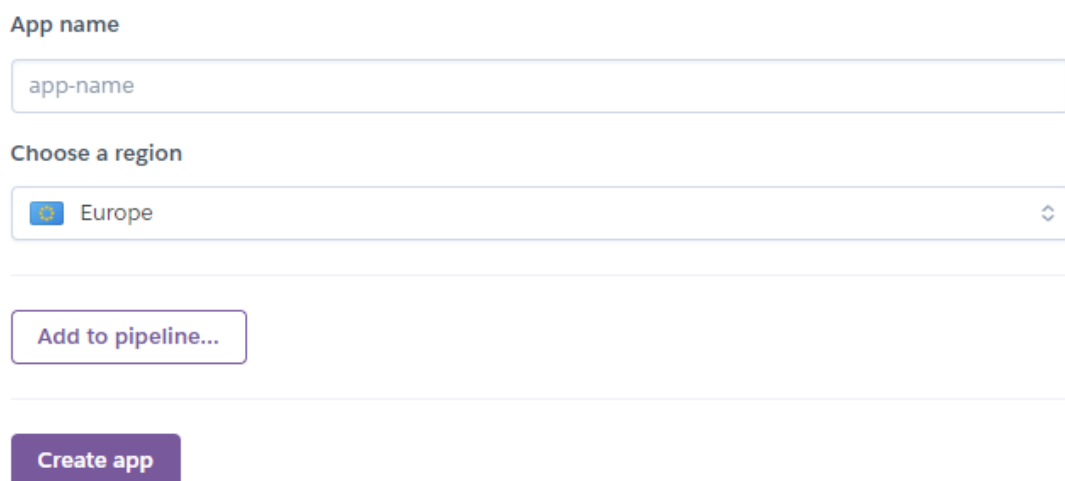
Para añadir todo, utilizaremos el comando `git add .` y para hacer commit a los cambios utilizaremos el comando `git commit -m "mensaje"` que nos permitirá hacer un commit y dejar un mensaje explicativo del mismo.

```
PS F:\xampp\htdocs\omnibank> git add .  
PS F:\xampp\htdocs\omnibank> git commit -m "commit tutorial"  
[master b385952] commit tutorial  
3 files changed, 5 insertions(+), 1 deletion(-)  
PS F:\xampp\htdocs\omnibank> █
```

Y ahora volvemos a Heroku.

PASO7: Crear una aplicación nueva en Heroku

En la pantalla principal pincharemos en “New” y seguidamente en “Create new app” para acceder a la pantalla de creación de apps, la cual nos pedirá el nombre de la app y la región donde queremos que se aloje.



App name

Choose a region

Europe

Add to pipeline...

Create app

Pinchamos en “Create app” y ya tendremos la aplicación creada en Heroku, ahora tenemos que subir los archivos.

PASO8: Subir los archivos de la aplicación

Para ello tendremos que volver a nuestra terminal y utilizar el comando:

heroku git:remote -a nombreapp donde nombreapp es el nombre que hayamos puesto en Heroku, en mi caso será omnibank2022:

```
PS F:\xampp\htdocs\omnibank> heroku git:remote -a omnibank2022
» Warning: heroku update available from 7.53.0 to 7.60.2.
set git remote heroku to https://git.heroku.com/omnibank2022.git
PS F:\xampp\htdocs\omnibank> █
```

Acto seguido solo tendremos que hacer un push a Heroku con el siguiente comando: *git push heroku branch* donde branch es la rama de git que deseamos subir, en mi caso subiré la rama master, que es donde están todos los cambios hechos:

```
PS F:\xampp\htdocs\omnibank> git push heroku master
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 8 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 15.27 KiB | 5.09 MiB/s, done.
Total 21 (delta 12), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
```

```
remote: -----> Compressing...
remote:      Done: 15.6M
remote: -----> Launching...
remote:      Released v16
remote:      https://omnibank2022.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/omnibank2022.git
4ff00a6..b385952 master -> master
PS F:\xampp\htdocs\omnibank> █
```

Y nos dará la URL de nuestra aplicación, que ya estará desplegada:

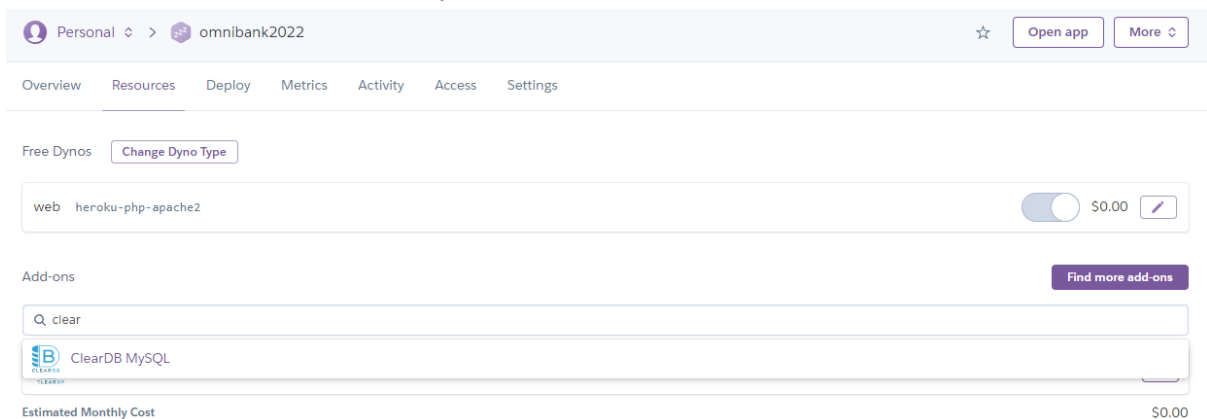


Si nuestra aplicación no se conectara a ninguna base de datos ya habríamos terminado pero como no es nuestro caso seguimos con el tutorial.

PASO9: Añadir una base de datos MySQL

Podríamos usar cualquier base de datos que estuviera operativa siempre, pero Heroku nos da la posibilidad de usar un add-on para alojar nuestra base de datos en los mismos servidores de Heroku.

Para añadirla, debemos irnos a la pantalla de nuestra aplicación, pinchar en “Resources” y en el buscador poner “ClearDB MySQL”:



Desgraciadamente, para añadir add-ons tendremos que vincular una tarjeta de crédito a nuestra cuenta, pero mientras elijamos tanto planes como add-ons gratuitos, no habrá problema. Cuando hayamos vinculado la tarjeta pinchamos en el add-on y en “Submit Order Form”.

Plan name

Ignite – Free

[View add-on details in Elements Marketplace](#)

By submitting this order form, you agree that the Add-on is governed by the applicable provider's terms of use, and the Heroku Services are governed by the [Salesforce Master Subscription Agreement](#), unless (except for free customers) you have entered into a written Master Subscription Agreement executed by SFDC for the Heroku Services as referenced in the Documentation.

Submit Order Form

Con esto ya tendremos añadido el add-on de MySQL.

PASO10: Conseguir las credenciales del servidor

En la pantalla de nuestra app en Heroku, vamos a la pestaña “Settings” y en el apartado “Config Vars” clicamos en “Reveal Config Vars”

Config Vars

[Reveal Config Vars](#)

Config vars change the way your app behaves.
In addition to creating your own, some add-ons come with their own.

Copiamos lo que nos aparece en la segunda entrada de texto:

Config Vars

[Hide Config Vars](#)

CLEARDB_DATABASE_URL

mysql://ba71271bd421e5:76694fab@eu-cdb-r-



KEY

VALUE

Add

Y lo pegamos en un bloc de notas para poder extraer las credenciales, las más son:

```
mysql://ba71271bd421e5:76694fab@eu-cdb-r-west-02.cleardb.net/heroku_20722431ee1d401?reconnect=true
```

De aquí podemos sacar:

USER: ba71271bd421e5

PASS: 76694fab

HOST: eu-cdb-r-west-02.cleardb.net/heroku_20722431ee1d401

Que son las credenciales de nuestro servidor.

PASO11: Conectarnos a la base de datos con PHPMyAdmin

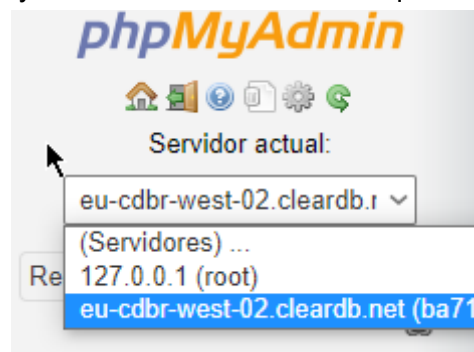
Una vez conseguidas las credenciales podremos configurar nuestro PHPMyAdmin para que se conecte a la base de datos y poder configurarla desde ahí. Para ello abriremos nuestro archivo de configuración de PHPMyAdmin "config.inc.php" y añadiremos las siguientes líneas al final del archivo:

```
/* Heroku remote server */  
$i++;  
$cfg["Servers"][$i]["host"] = "eu-cdbbr-west-02.cleardb.net"; //hostname del servidor  
$cfg["Servers"][$i]["user"] = "ba71271bd421e5"; //user del servidor  
$cfg["Servers"][$i]["password"] = "76694fab"; //contraseña del servidor  
$cfg["Servers"][$i]["auth_type"] = "config";
```

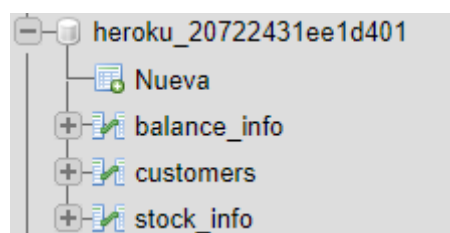
Lo cual añadirá a nuestro PHPMyAdmin el nuevo servidor de la base de datos.

PASO12: Configurar la base de datos

Accedemos a PHPMyAdmin y seleccionamos el servidor que hemos añadido:



Ahí estará creada una base de datos para que la utilicemos en nuestra aplicación, podemos añadirle las tablas y todo lo que necesitemos, en mi caso yo he importado la base de datos que estaba utilizando en local.



Es importante quedarnos con el nombre de la base de datos para luego poder ponerlo en las credenciales que utilice nuestra aplicación.

En mi caso el nombre de la base de datos es: *heroku_20722431ee1d401*

PASO13: Conectar a la base de datos de Heroku desde PHP

Este paso es realmente sencillo gracias a la utilización de un archivo de credenciales, simplemente cambiamos los valores de la base de datos en local por la nueva de heroku y ya estaría listo:

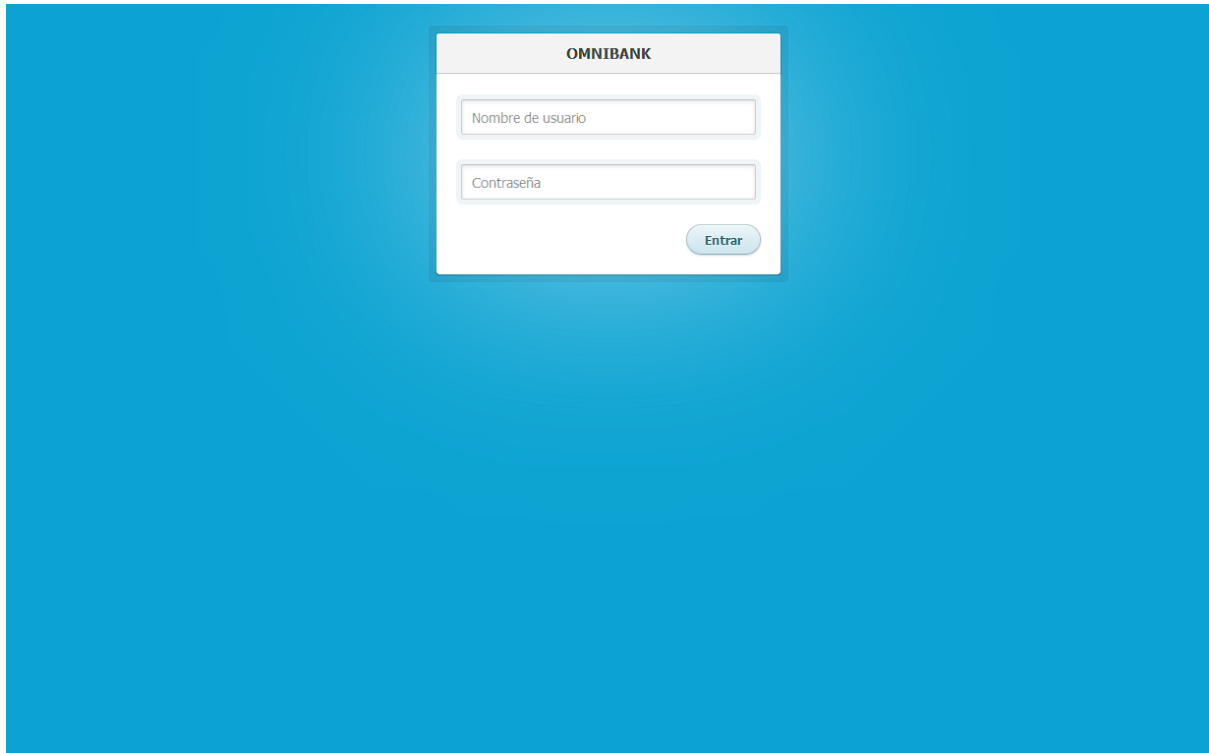
```
omnibank > model > testbank.inc.php > ...  
1  <?php  
2  //testbank.inc.php  
3  
4  
5  define('HOST', 'eu-cdbbr-west-02.cleardb.net');  
6  //define('HOST', 'localhost');  
7  define('PORT', '3306');  
8  define('BD', 'heroku_20722431ee1d401');  
9  //define('BD', 'testbank');  
10 define('USER', 'ba71271bd421e5');  
11 //define('USER', 'root');  
12 define('PWD', '76694fab');  
13 //define('PWD', '');  
14 define('CHRST', 'utf8');
```

EJEMPLO DE SESIÓN

Para utilizar la app, simplemente tenemos que ir a la dirección:

<http://omnibank2022.herokuapp.com>

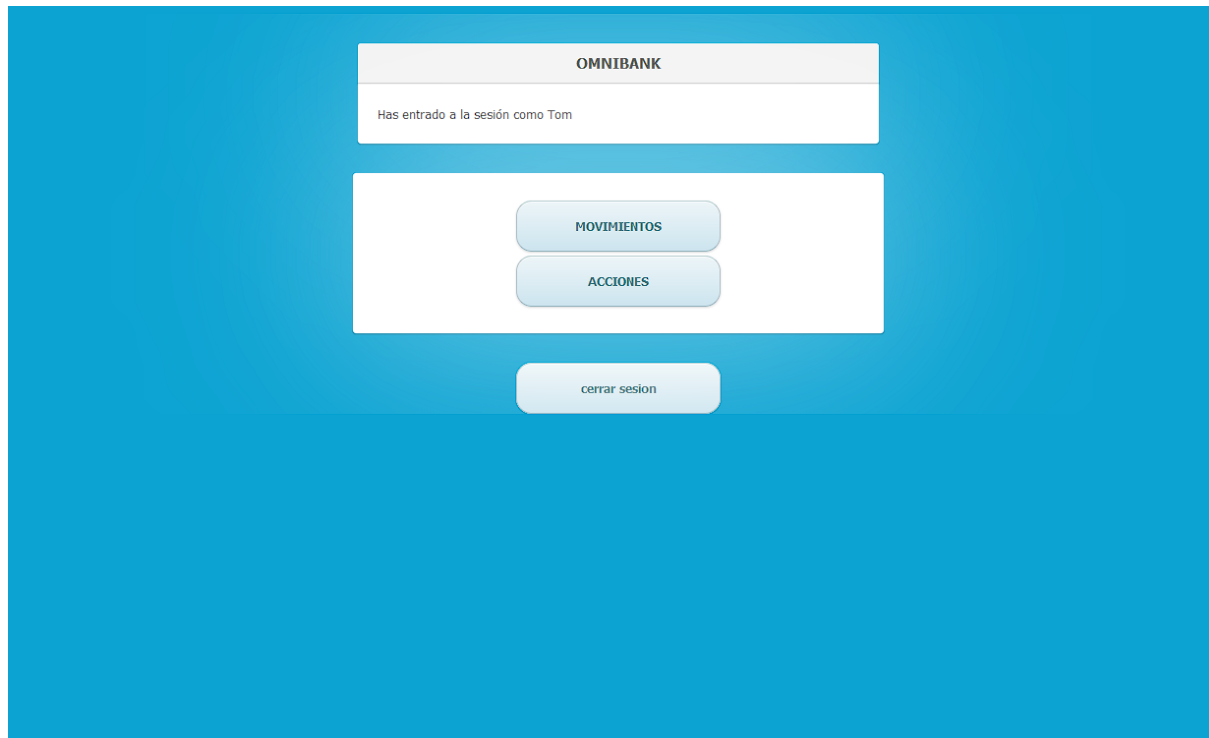
Y nos redireccionará a la pantalla de login:

The image shows a login interface for 'OMNIBANK'. It features a white rectangular form centered on a solid blue background. The form has a title bar at the top that says 'OMNIBANK'. Below the title bar, there are two input fields: the first is labeled 'Nombre de usuario' and the second is labeled 'Contraseña'. At the bottom right of the form, there is a blue button with the text 'Entrar' in white.

Una vez en esta pantalla, se iniciará la sesión y la variable de los reintentos y el tiempo. Si fallamos el proceso de autenticación, nos saldrá esta pantalla:

The image shows an error message screen. It features a white rectangular form centered on a solid blue background. The form has a title bar at the top that says 'ACCESO INCORRECTO'. Below the title bar, there is a blue button with the text 'Reintentar' in white.

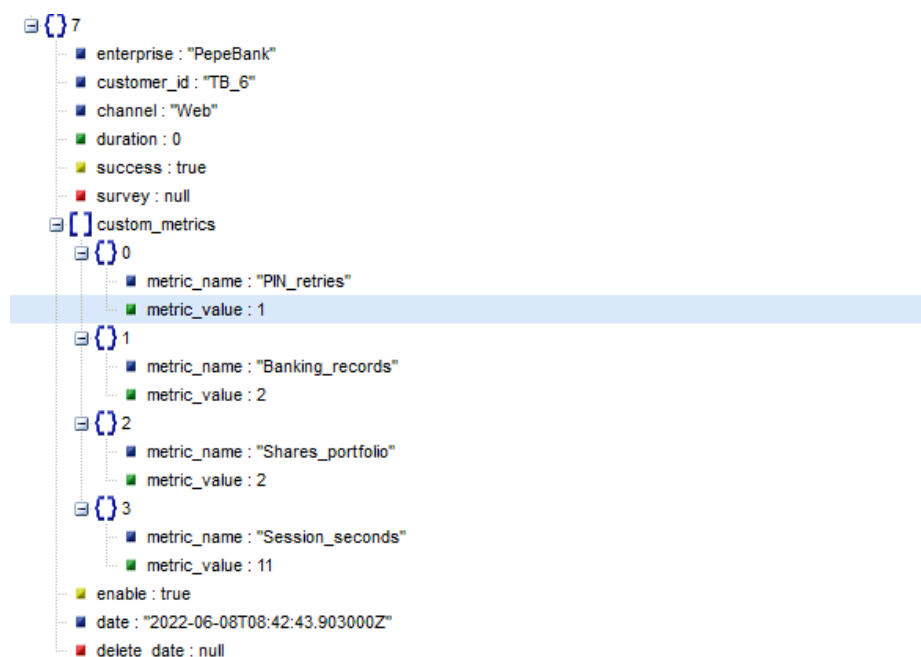
Y cuando le demos a reintentar volveremos al login y se sumará 1 al contador de reintentos. Vamos a logearnos con el usuario Tom por ejemplo, que tiene la ID TB_6 y su contraseña es 1234.



Accederemos a esta pantalla principal. Se inicializarán las variables de cada botón, y cada vez que pulsemos alguno de estos, se sumará 1 a sus respectivos contadores y se mostrará la tabla deseada.

Cuando hayamos terminado de consultar la aplicación, pulsaremos en el botón “cerrar sesión” para mandar las variables de los reintentos, los movimientos, las acciones y el tiempo a la CDP y volver a la pantalla de login.

Para ver si el envío ha sido satisfactorio, con una página adicional “checkdb.php” que he creado únicamente con el cometido de hacer una petición GET a la API de WhenWhyHow para que me devuelva todas las sesiones que tiene guardadas, vamos a comprobarlo.



Cabe destacar que debido a un problema de la propia API de WhenWhyHow, antes de enviar los valores, se les suma siempre 1, porque tiene que ser mayor que 0. Esto se corrige al visualizar en su aplicación los datos extraídos, en los que se restará 1 respectivamente. Por eso en el JSON se ve así, 1 reintento cuando hemos entrado a la primera y 2 en cada uno de los botones cuando solo los pulsamos 1 vez.

APLICACIÓN WEB PROGRESIVA (PWA)

Sin duda la tecnología más interesante y ambiciosa que he descubierto trabajando en el proyecto son las PWAs.

Las PWAs son aplicaciones web portadas a aplicación móvil que son capaces de funcionar incluso sin conexión (Omnibank necesita de una conexión a una base de datos en un servidor en la nube por lo que no es el caso).

Lo más interesante de esta tecnología es lo sencillo que es pasar de una aplicación web a una móvil sin necesidad de pasar por Android Studio o similares.

Simplemente añadiremos un manifest.json en la raíz de nuestra aplicación con el siguiente código:

```
omnibank > manifest.json > ...
1  {
2      "name": "OMNIBANK",
3      "short_name": "OmniBank",
4      "start_url": "/control/login.php",
5      "display": "standalone",
6      "background_color": "#fdfdfd",
7      "theme_color": "#0ca3d2",
8      "orientation": "portrait-primary",
9      "icons": [
10     {
11         "src": "bank.png",
12         "type": "image/png", "sizes": "72x72"
13     }
14 ]
15 }
```

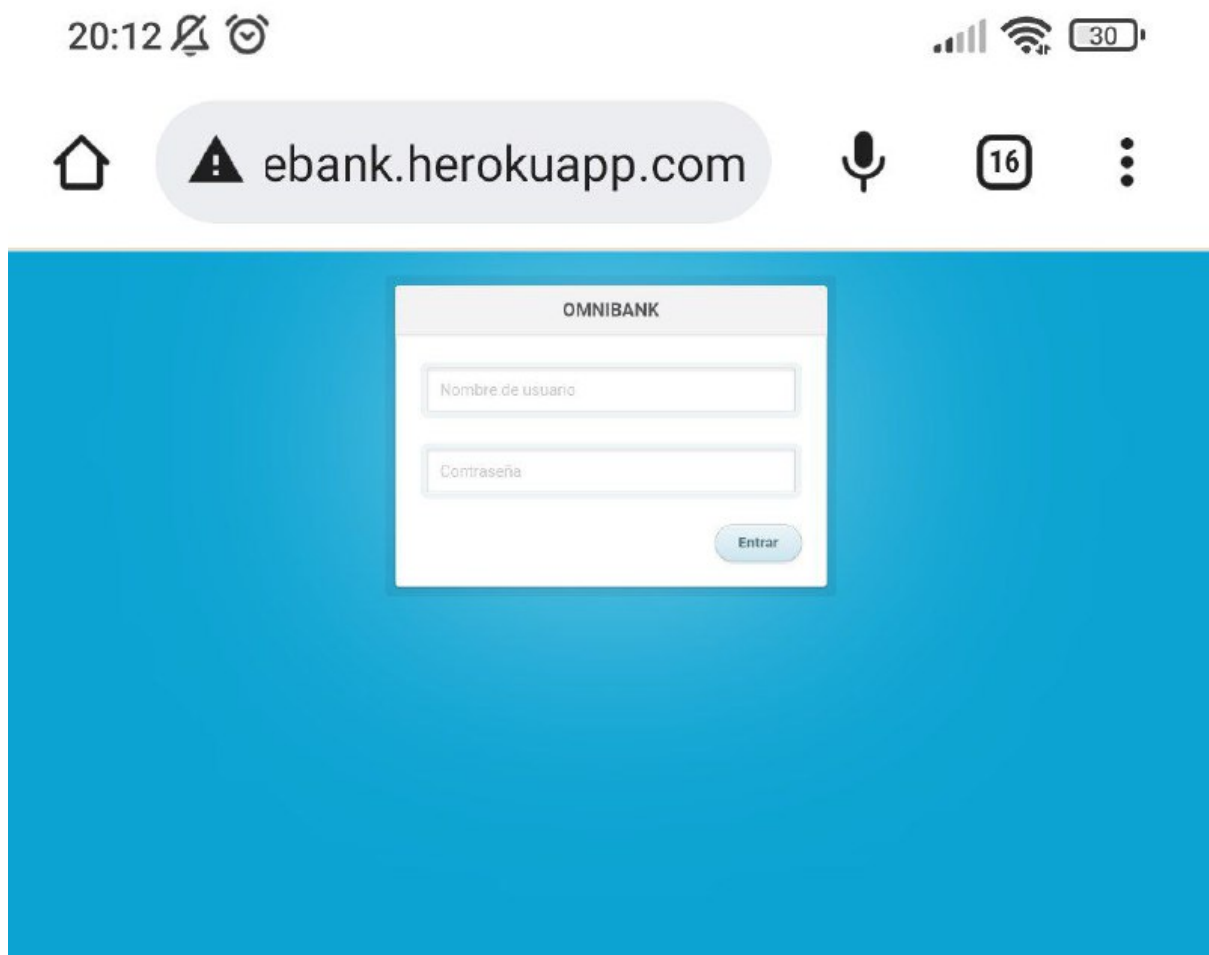
Con tan solo estos parámetros ya podremos pasar a PWA. (El parámetro icons te permite poner iconos para diferentes resoluciones pero no he conseguido que me funcione en ninguna).

Después de crear el manifest, nos dirigiremos a todas las páginas de nuestra app e incluiremos en el head un link al manifest, de esta manera:

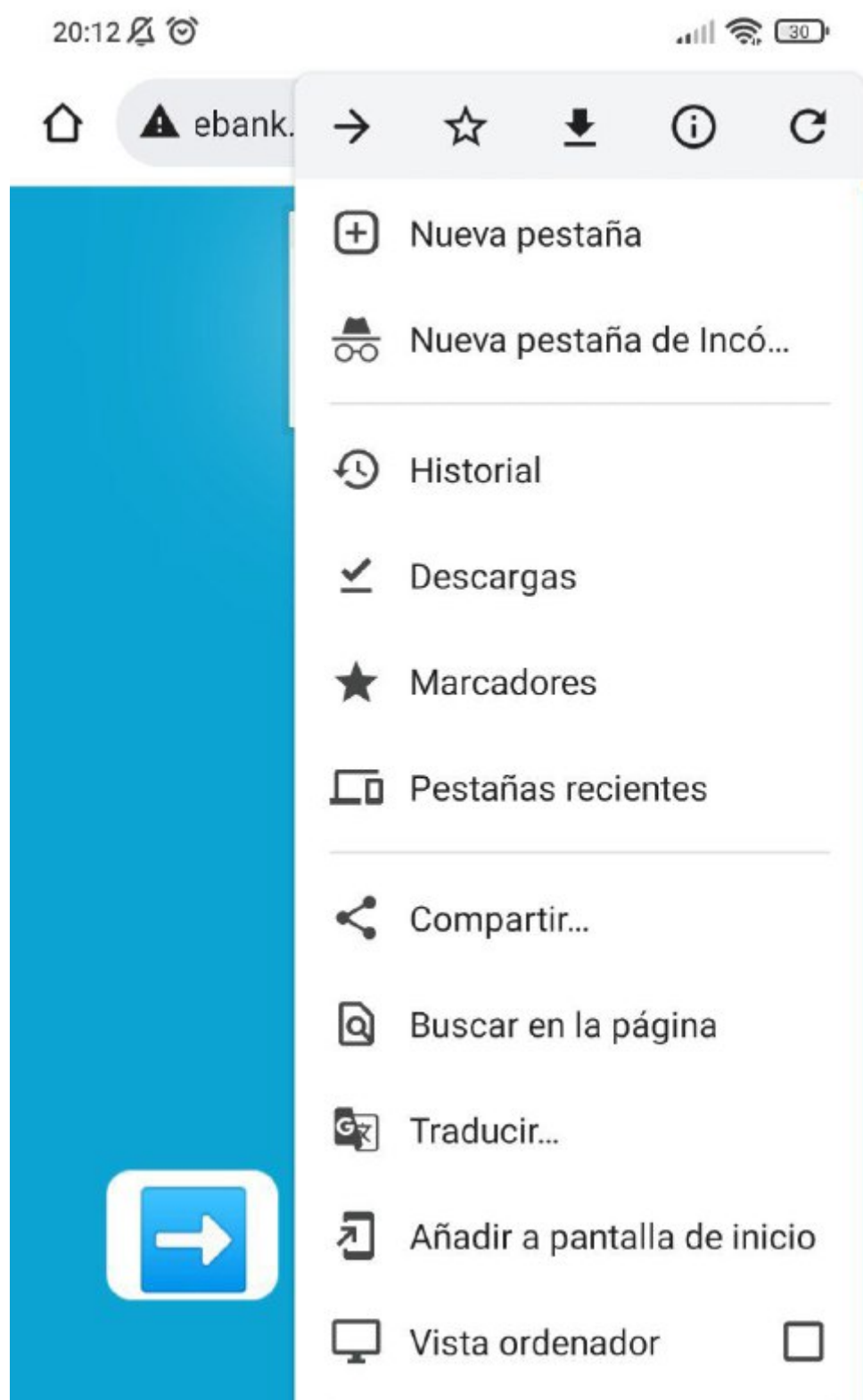
```
omnibank > control > login.php > html > body
1  <!DOCTYPE html>
2
3  <html>
4      <head>
5          <meta charset='UTF-8'>
6          <link rel="stylesheet" href="../styles/login.css">
7          <link rel='manifest' href='../manifest.json'>
8      </head>
9
```

INSTALACIÓN PWA

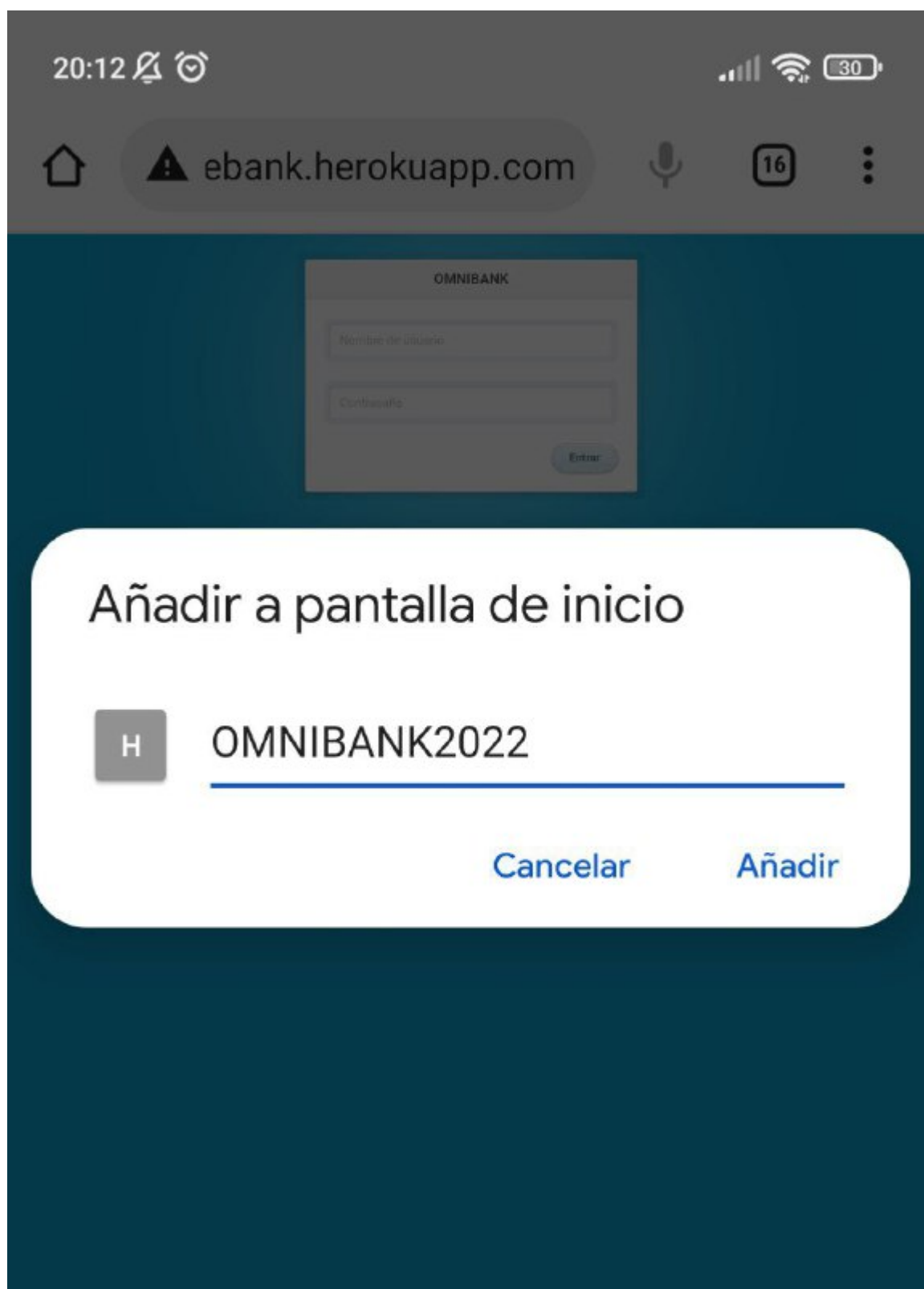
Primero, vamos a acceder a la página de omnibank.



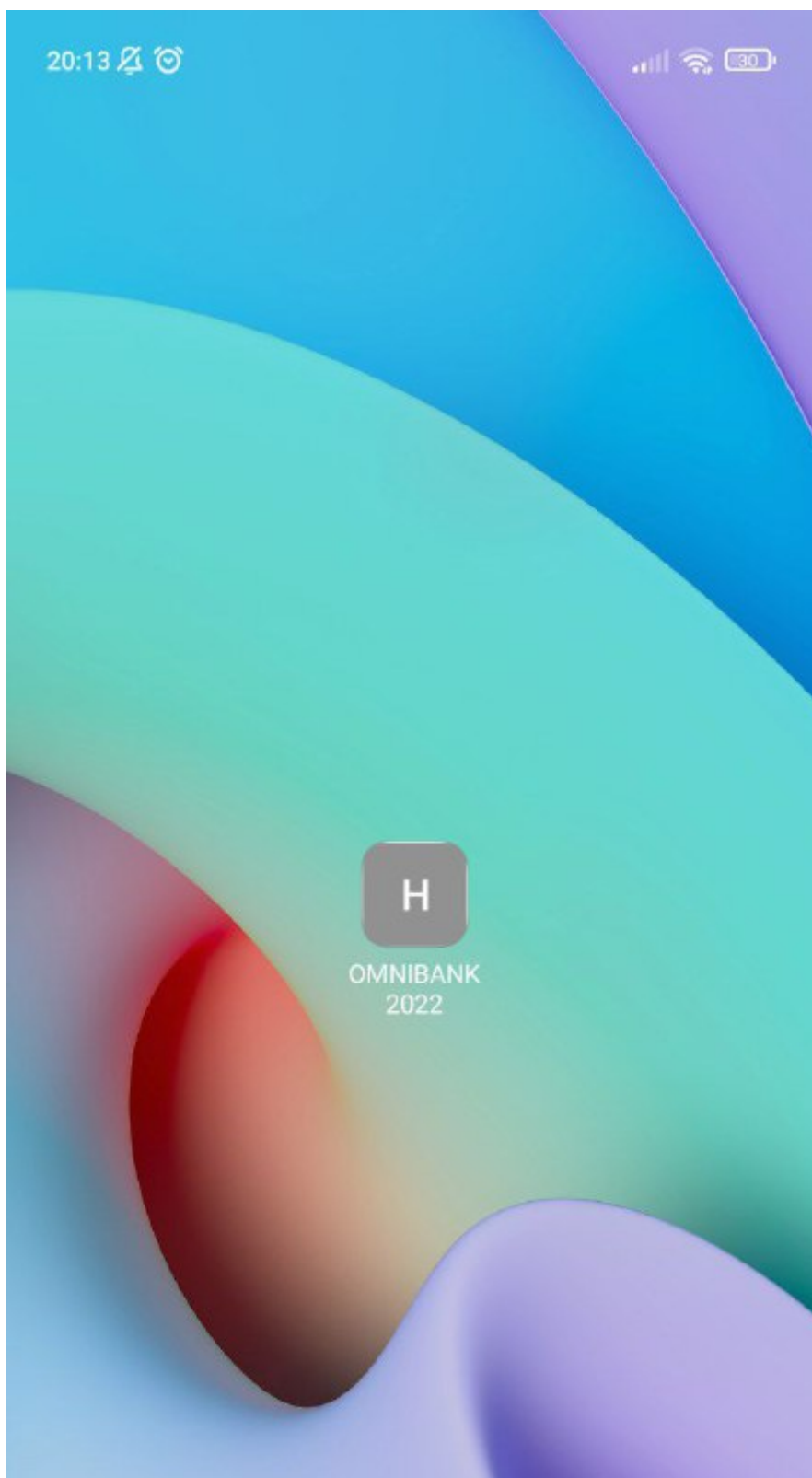
Una vez allí, pinchamos en las opciones:



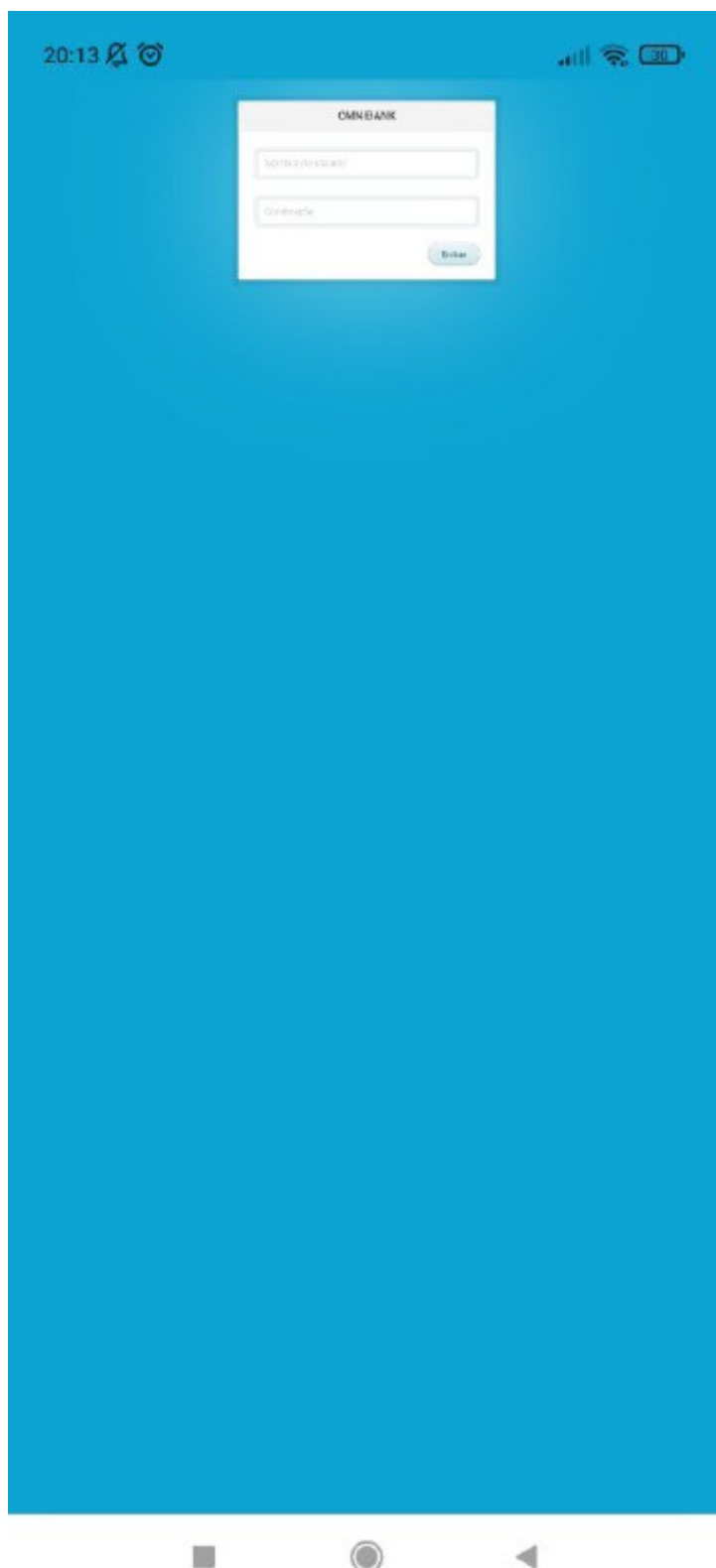
Y seleccionamos la opción “Añadir a pantalla de inicio”.



Nos pedirá el nombre con el que queremos que aparezca, le damos a añadir.



Y en nuestra pantalla de inicio aparecerá la app que se ejecutará como una app nativa.



Como se puede observar, no aparece el UI de Google Chrome y funciona igual que la página original de la web.

PROBLEMAS SURGIDOS QUE LLEVAN A CAMBIOS EN LA PLANIFICACIÓN

Durante el desarrollo del proyecto han ido surgiendo una gran cantidad de problemas a medida que iba avanzando.

Si bien, en teoría debería haber sido capaz de realizar el proyecto sin ningún inconveniente ya que durante los últimos dos años hemos sido más que formados para ello, la dura realidad es que a la hora de ponerte a trabajar y a desarrollar, surgen muchísimos problemas, que no siempre son muy complicados de solucionar, pero que en gran cantidad y combinándolo con la fatiga y el estrés que supone compaginar las FCTs con el proyecto, acaban acumulándose hasta el punto de tener estancamientos de horas e incluso días.

Al principio tenía una idea clara de cómo quería que quedara la aplicación pero al ir añadiendo funcionalidades de todas partes salen excepciones, avisos y errores que suman horas, a veces restan avances pero al final del día siempre te nutren como programador y profesional.

Pero existen diversas prácticas y enfoques de trabajo que logran minimizar dichos inconvenientes y facilitan el arreglo de los mismos:

Durante todo el proyecto, seguí el modelo MVC (Model, View, Controller) para desarrollar la aplicación. Este separa el código de la aplicación dependiendo de la funcionalidad del mismo. Cuando sigues esta organización, siempre puedes identificar rápidamente de dónde surgen los problemas y darle solución sin pisar el resto del código. Definitivamente seguir este modelo de trabajo me ha ahorrado una gran cantidad de quebraderos de cabeza, tiempo y estrés.

También ha sido especialmente duro darle estilos a la aplicación, ya que CSS es la tecnología que más se me atraganta, no por falta de lógica ni de esfuerzo, sino por escasez de imaginación. He intentado que fuera lo más presentable posible.

Otro de los problemas que me surgió fue que hasta muy avanzado el proyecto no había desplegado la aplicación en ningún servicio de hosting online, así que toda la aplicación corría en local. A primera vista esto no me parecía un inconveniente hasta que después de mucho esfuerzo en el despliegue (del que hablaré a continuación) empezaron a salir errores por doquier. Esto me ha servido para aprender a enfocar mi trabajo a la publicación en internet del mismo y prevenir algunos problemas que si hubiera tenido en mente en ese instante que mi aplicación iba a ser alojada en internet hubiera podido evitar.

El despliegue del proyecto me ocupó bastante tiempo pero también ha sido con lo que más realizado me he sentido, además de que me ha obligado a aprender a usar Git, herramienta que considero extremadamente útil y necesaria para cualquier proyecto (en mi opinión durante el desarrollo del grado debería dársele más importancia a esta herramienta, ya que su uso facilita casi todas las tareas).

Desplegué la aplicación en Heroku, un SaaS que ofrece hosting gratuito de aplicaciones WEB. Inicialmente era un poco complicada de usar, pero tiene una documentación excelente y una gran comunidad detrás que ha creado muchos videotutoriales y guías que me han hecho la vida mucho más fácil.

Cuando terminé de desplegar la aplicación, me dí cuenta de que sólo quedaba una semana de mayo, había invertido en el proyecto bastante más de 40 horas y todavía me quedaba implementar la conexión con la API de mi empresa, el chatbot y convertir la aplicación web en una aplicación móvil.

El tiempo se venía encima y he tenido que recortar funcionalidades, así que tanto el chatbot como adaptar los estilos a la versión de móvil se quedan para siguientes versiones.

Implementando la conexión con la API de WhenWhyHow surgieron algunos problemas porque en el grado aprendimos a crear APIs pero no le dimos mucho enfoque a consumirlas, además de que lo dimos en Java y mi proyecto está en PHP.

Pero al final resultó ser bastante sencillo captar la lógica gracias a que aprendimos a crear las APIs desde cero y teniendo esa base, aprender a consumirlas ha sido un paseo.

En general si bien ha sido un camino muy duro, ha enriquecido exponencialmente mis conocimientos y habilidades como programador, además de que gracias a pequeños detalles que aprendí tanto de mis profesores del grado como a mis compañeros en las FCTs, se me ha hecho bastante menos estresante y difícil de lo que en un principio me imaginaba.

En resumen, los objetivos que he logrado:

- La creación de las páginas tanto del login como de la aplicación.
- La transformación de una base de datos no relacional proporcionada por el cliente a una relacional compatible con el backend de la aplicación.
- La conexión de la base de datos con las páginas de la aplicación.
- La creación de las páginas de estilos para dar un aspecto presentable a la aplicación.
- El despliegue de la aplicación en un servicio de hosting.
- El envío de los datos de sesión de los usuarios a una plataforma CDP.
- La exportación a Aplicación Web Progresiva para su instalación en dispositivos móviles.

Y se ha quedado en el tintero:

- La creación de un chatbot que interactuara con la aplicación.
- Adaptación de los estilos a la app móvil,

VERSIONES

Para el control de versiones he usado la herramienta Git, que es la más popular para este cometido.

A lo largo del desarrollo del proyecto, cada pequeño cambio que he realizado ha sido documentado por la herramienta a base de commits que he ido haciendo periódicamente. Empecé a usar Git en una versión muy primitiva de la aplicación, la cual consistía en modificaciones de la aplicación GESVENTAS que desarrollamos durante el curso, y a partir de ahí, he ido desarrollando poco a poco lo que ha resultado ser a día de hoy Omnibank.

```
commit b38595202d2b293152dea0e2f489fca004e05b51 (HEAD -> master,
heroku/master)
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date:   Wed Jun 8 15:45:54 2022 +0200
```

```
commit tutorial
```

```
commit 4054b994ba0f15bf51c31453092998968e559e88
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date:   Tue Jun 7 18:54:14 2022 +0200
```

```
manifest bien app
```

```
commit 8de98160bc6eaeef98b291ce5f42194cf53835a51
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date:   Tue Jun 7 18:38:45 2022 +0200
```

```
canal web pwa
```

```
commit 95d1fb29c546d5fca04643c2b909383975c67c29
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date:   Tue Jun 7 18:27:17 2022 +0200
```

```
canal app pwa
```

```
commit 4ff00a6566e9ab6d19da57a65532b0555cacb925
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date:   Fri Jun 3 10:24:48 2022 +0200
```

```
añadida tarjeta del usuario
```

```
commit fd7ea4da43980707c936ffdf2c7f13df4f4bc95f
Author: fseiore <fseiore@gmail.com>
Date:   Wed Jun 1 19:02:46 2022 +0200
```

para entregar estado ejecucion

```
commit 5e54eeb8bbe56a046801d1a4c5a608579d36e1ae
Author: fseiore <fseiore@gmail.com>
Date:   Mon May 30 13:46:38 2022 +0200
```

añadida la conexion con la api, funciona v1

```
commit cd880ba3eb8d490632b49d48402cf3bb365f412e
Author: fseiore <fseiore@gmail.com>
Date:   Fri May 27 11:23:04 2022 +0200
```

todas las variables menos el tiempo (local)

```
commit 1d8f717839b042f8620b38ee1f37658e4f5ae21a
Author: fseiore <fseiore@gmail.com>
Date:   Thu May 26 13:07:37 2022 +0200
```

css terminado

```
commit c3d7cf2d7db07c351e422cb460e1226507a5b421
Author: fseiore <fseiore@gmail.com>
Date:   Tue May 24 13:55:15 2022 +0200
```

oops2

```
commit 3e49be5e4263d6975b80500accbfe6de8ad021f9
Author: fseiore <fseiore@gmail.com>
Date:   Tue May 24 13:54:04 2022 +0200
```

oops

```
commit 4e144bb438829a336a667bfc527bd46659825211
Author: fseiore <fseiore@gmail.com>
Date:   Tue May 24 13:47:49 2022 +0200
```

parametros hosting

```
commit 406c8104fd4806ba5d5e9f586c869e8220b13c17
Author: fseiore <fseiore@gmail.com>
Date:   Tue May 24 13:45:47 2022 +0200
```

cambiado al modelo mvc

```
commit bc1ed5e6ee2eef00359e710896c5e8cc3f3181ee
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Tue May 24 11:27:27 2022 +0200
```

nombres de columna user friendly, formato dinero

```
commit afd51fee37685909dcc28474a17ff51578ff2d51
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Tue May 24 11:01:15 2022 +0200
```

salida de datos con formato, login con css

```
commit 31cbc8e147cf56769ce335e6efb2b18eb5c40026
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Thu May 19 12:46:19 2022 +0200
```

arreglado boton de terminar del login2

```
commit f568db0260bd8afa2f304a3c7d8508dd9859a4f8
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Thu May 19 12:41:43 2022 +0200
```

arreglado boton de terminar del login

```
commit 90d843e38f09cdacc37ad05877ca3107c5f7a3a6
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Thu May 19 12:35:54 2022 +0200
```

bbdd en heroku

```
commit a900e9779aa4425a4f480153ac44a98c92803d60
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Thu May 19 10:21:13 2022 +0200
```

intento de arreglo de index

```
commit 2b67ff1ddfecf55f4fa43da29a54360bac30bf3e
```

```
Author: fseiore <fseiore@gmail.com>
```

```
Date: Wed May 18 13:43:54 2022 +0200
```

commit inicial

MEJORAS FUTURAS

Bien es cierto que con el tiempo que se supone que teníamos que haber invertido en el proyecto no era suficiente para cubrir la ambición de nuestras ideas, pero me gustaría incluir los avances y cambios que me hubiera gustado añadir si hubiera tenido más tiempo:

- Mejorar la seguridad de la app: utilizar métodos de encriptación para el login, controlar la sesión con el tiempo, añadir autenticación en dos pasos, etc.

- Mejorar los estilos de la app: pulir mis conocimientos sobre CSS, usar bootstrap, hablar con algún diseñador...

- Añadir la posibilidad de hacer movimientos bancarios simples, como transferencias, pagar facturas, etc.

- Añadir el chatbot, como tenía pensado en un principio.

- Añadir una página de registro para usuarios nuevos.

- Implementar JavaScript y hacerlo funcionar con la parte servidor, para por ejemplo, poder enviar a la CDP datos sobre el navegador que se está usando, el tamaño de la pantalla, etc.

CONCLUSIONES

Tras finalizar el proyecto y las prácticas, he cambiado mi visión completamente sobre la programación, y si bien me he sentido muy abrumado por el gran camino que tengo por delante, estoy a la vez motivado por desarrollar mis capacidades y hacerme un pequeño hueco en el inmenso océano que es esta profesión.

Antes del tercer trimestre, no estaba muy seguro de cómo iba a poder incorporarme a una empresa y hacer un buen trabajo, pero combinando los consejos que nos han dado los profesores, lo que hemos aprendido en las FCTs y en el proyecto y un poquito de esfuerzo, me veo con mucha más confianza para ser un programador hecho y derecho.

BIBLIOGRAFÍA:

Recursos Aula Virtual IES BARAJAS

Documentación API WhenWhyHow

<https://www.doabledanny.com/Deploy-PHP-And-MySQL-to-Heroku>

<https://www.sliderrevolution.com/resources/css-tables/>

<https://devcenter.heroku.com/categories/php-support>

<https://www.w3schools.com>

<https://www.php.net/manual/es/>

<https://rapidapi.com/blog/how-to-use-an-api-with-php/#prerequisites-to-start-using-api-with-php>

<https://stackoverflow.com>

<https://www.baulphp.com>

<https://desarrolloweb.com/manuales>

<https://web.dev/progressive-web-apps/#funciones-exclusivas-de-la-pwa>

<https://www.xatakandroid.com/aplicaciones-android/como-instalar-aplicaciones-web-progresivas-tu-android-para-ahorrar-espacio>

<https://git-scm.com/docs>

ANEXOS

URL OMNIBANK:

<http://omnibank2022.herokuapp.com>

CÓDIGO BBDD TESTBANK

```
DROP DATABASE IF EXISTS TESTBANK;
```

```
CREATE DATABASE TESTBANK;
```

```
USE TESTBANK;
```

```
CREATE TABLE IF NOT EXISTS customers (  
  `customer_id` VARCHAR(4) CHARACTER SET utf8,  
  `customer_name` VARCHAR(20) CHARACTER SET utf8,  
  `customer_pin` INT,  
  PRIMARY KEY (customer_id)  
);  
  
CREATE TABLE IF NOT EXISTS balance_info (  
  `balance_info_customer` VARCHAR(4) CHARACTER SET utf8,  
  `balance_info_date` INT,  
  `balance_info_concept` VARCHAR(50) CHARACTER SET utf8,  
  `balance_info_amount` FLOAT(15,2),  
  `balance_info_remaining` FLOAT(15,2),  
  `balance_info_id` INT(11) AUTO_INCREMENT,  
  PRIMARY KEY (balance_info_id),  
  FOREIGN KEY (balance_info_customer) REFERENCES customers(customer_id)  
);
```

```
CREATE TABLE IF NOT EXISTS stock_info (  
  `stock_info_customer` VARCHAR(4) CHARACTER SET utf8,  
  `stock_info_share` VARCHAR(30) CHARACTER SET utf8,  
  `stock_info_investment` FLOAT(15,2),  
  `stock_info_current` FLOAT(15,2),  
  `stock_info_difference` FLOAT(15,2),  
  `stock_info_id` INT(11) AUTO_INCREMENT,  
  PRIMARY KEY (stock_info_id),  
  FOREIGN KEY (stock_info_customer) REFERENCES customers(customer_id)  
);
```

```
INSERT INTO customers VALUES
```

```
  ('TB_1','Rose',1234),  
  ('TB_2','Alex',1234),  
  ('TB_3','Pam',1234),  
  ('TB_4','Mark',1234),  
  ('TB_5','Amanda',1234),  
  ('TB_6','Tom',1234);
```

```
INSERT INTO balance_info(  
balance_info_customer,  
balance_info_date,  
balance_info_concept,  
balance_info_amount,  
balance_info_remaining) VALUES
```

```
  ('TB_1',20200906,'Home Insurance',670.00,91733.00),  
  ('TB_1',20200903,'Debit Card. Restaurant High',97.00,92403.00),  
  ('TB_1',20200901,'Rent',2600.00,92500.00),  
  ('TB_1',20200820,'Car Insurance',1300.00,95100.00),  
  ('TB_1',20200815,'City Tax',600.00,96400.00),  
  ('TB_1',NULL,'initial',NULL,97000.00),  
  ('TB_2',20200906,'Energy',21.00,2235.00),  
  ('TB_2',20200903,'Debit Card. Alfredo's Bar',14.00,2256.00),  
  ('TB_2',20200901,'Rent',900.00,2270.00),  
  ('TB_2',20200820,'Car Insurance',280.00,3170.00),  
  ('TB_2',20200815,'City Tax',150.00,3450.00),  
  ('TB_2',NULL,'initial',NULL,3600.00),  
  ('TB_3',20200906,'Energy',78.00,11661.00),  
  ('TB_3',20200903,'Debit Card. Restaurant',32.00,11739.00),  
  ('TB_3',20200901,'Mortgage',775.00,11771.00),  
  ('TB_3',20200820,'Tennis Club annuality',800.00,12546.00),  
  ('TB_3',20200815,'Debit Card. Mark&Spencer',178.00,13346.00),  
  ('TB_3',NULL,'initial',NULL,13524.00),  
  ('TB_4',20200906,'Telephone Bill',18.00,84.00),  
  ('TB_4',20200903,'Debit Card. Alfredo's Bar',14.00,102.00),  
  ('TB_4',20200901,'Rent',250.00,116.00),  
  ('TB_4',20200820,'Debit Card. Media Markt.',1314.00,366.00),  
  ('TB_4',20200815,'Debit Card. Car repair',420.00,1680.00),  
  ('TB_4',NULL,'initial',NULL,2100.00),  
  ('TB_5',20200906,'Gas & Water',37.00,22318.00),  
  ('TB_5',20200903,'Debit Card. Hotel Berlin',230.00,22355.00),  
  ('TB_5',20200901,'Mortgage',680.00,22585.00),  
  ('TB_5',20200820,'Debit Card. Zara',118.00,23265.00),  
  ('TB_5',20200815,'Debit Card. Car repair',315.00,23383.00),  
  ('TB_5',NULL,'initial',NULL,23698.00),  
  ('TB_6',20200906,'Telephone Bill',47.00,19502.00),  
  ('TB_6',20200903,'Debit Card. Hotel NH',315.00,19549.00),  
  ('TB_6',20200901,'Mortgage',2100.00,19864.00),  
  ('TB_6',20200820,'Debit Card. Media Markt.',1314.00,21964.00),
```

```
('TB_6',20200815,'Debit Card. Car repair',420.00,23278.00),  
('TB_6',NULL,'initial',NULL,23698.00);
```

```
INSERT INTO stock_info(  
stock_info_customer,  
stock_info_share,  
stock_info_investment,  
stock_info_current,  
stock_info_difference) VALUES  
('TB_1','Apple',50000.00,77992.00,27992.00),  
('TB_1','Bayer',30000.00,26237.00,-3763.00),  
('TB_1','Ford Motor',30000.00,31786.00,1786.00),  
('TB_1','TOTAL',110000.00,136015.00,26015.00),  
('TB_2','No stock yet',0.00,0.00,0.00),  
('TB_3','Vonage',15000.00,19300.00,4300.00),  
('TB_3','TOTAL',15000.00,19300.00,4300.00),  
('TB_4','No stock yet',0.00,0.00,0.00),  
('TB_5','No stock yet',0.00,0.00,0.00),  
('TB_6','Google',15000.00,36185.00,21185.00),  
('TB_6','BNP',10000.00,7760.00,-2240.00),  
('TB_6','Novartis',10000.00,13676.00,3676.00),  
('TB_6','TOTAL',35000.00,57621.00,22621.00);
```

HERRAMIENTAS PRODUCCIÓN

- XAMPP v3.3.0
- Visual Studio Code v1.68
- Mozilla Firefox v101.0.1
- Notepad++ v8.3.3
- Terminal Linux (Ubuntu 20.04.4 LTS como aplicación de Windows)
- Online JSON Viewer <http://jsonviewer.stack.hu>
- Git v2.36.1