PELUCAS SOLIDARIAS

Autor: Rubén Esteve Vicente

Ciclo: Desarrollo de aplicaciones web DAW

Familia profesional: Informática y comunicaciones

Curso: Segundo 21/22 **Año:** 2022

Centro: Ies la Vereda

Tutor: Carmen Martínez

Índice

1 Introducción.

1.1 Módulos a los que implica.

Para llevar a cabo este proyecto debo aplicar cada una de las **capacidades adquiridas durante los cursos de primero y segundo**.
Estas capacidades las puedo dividir en cada uno de los módulos realizados a lo largo de todo el grado superior de Desarrollo de Aplicaciones Web (DAW).

- ★ **Sistemas informáticos**
  - Este módulo me ha dado los conocimientos necesarios para saber configurar y conocer como funciona mi entorno de trabajo. Así como aprender el bash de linux, que hoy en día es el sistema operativo más usado en lo servidores.
- ★ **Bases de datos**
  - En esta asignatura he aprendido a diseñar (diagramas de E/R, diagramas ..etc), consultar y crear bases de datos. Aprendiendo principalmente bases de datos relacionales usando SQL con sus respectivas sentencias DDL,DML...
- ★ **Programación**
  - Gracias a comenzar con el lenguaje compilado y de tipado fuerte Java me ha hecho entender de forma mas robusta qué es y cómo funciona un lenguaje de programación. Aprendiendo a cómo programar con clases y objetos seguido de un código óptimo y legible.
- ★ **Lenguaje de marcas y DIseño de interfaces.**
  - Me forma de conocimientos básicos de cómo se diseña una página web. Aprendiendo el lenguaje de tipado HTML las plantillas de diseño CSS.
  - Aprendo metalenguajes para el uso mas sencillo y modular del css y js. Como por ejemplo Sass o JQuery. Me introduzco a la aplicación responsive de mis diseños con el uso de media queries y con ayuda frameworks de css como Bootstrap.

- ★ **Entornos de desarrollo**
  - El módulo mas aburrido pero el mas útil en mi opinión. En este he aprendido metodologías ágiles y aprender a hacer diagramas para el desarrollo progresivo de mi aplicación, como Scrum (Metodología usada en mi proyecto). Además de poner en práctica el sistema de control de versiones Git y a implementar testing en mi aplicación.
- ★ **Despliegue de Aplicaciones**
  - Asignatura muy importante para entender cómo funcionan los servidores, como desplegar tus proyectos, cómo crear tus propios servidores en Máquinas Virtuales... etc. Aprendo a usar tanto Docker como Docker-Compose para crear mi propio entorno de trabajo como desplegar mi aplicación.
- ★ **Entorno Cliente**
  - Paso de estar trabajando con páginas estáticas a diseñar páginas dinámicas. Aprendo las nociones básicas del lenguaje JavaScript, su DOM ... etc. Trabajo con npm que es el sistema de gestión de paquetes por defecto para Node.js y con esto aprendo el framework Angular que es con el que he realizado el frontend de mi aplicación.
- ★ **Entorno servidor**
  - Aprendo el lenguaje PHP que me sirve para conectarme a la base de datos. Descubro el uso de Sesiones y Cookies y por lo tanto a desarrollar un login y un register, adentrandome asi al mundo de las APIs. En mi caso no he usado PHP en mi aplicación, el backend lo he desarrollado con TypeScript.
- ★ **Empresa**
  - Esta asignatura no tiene nada que ver con los aspectos técnicos de la programación, pero como a lo largo de la asignatura hemos realizado la documentación de nuestra empresa ficticia me ha enseñado a documentar, diseñar y crear documentos de texto. Además de realizar exposiciones con algún Power Point y manera clara y concisa.

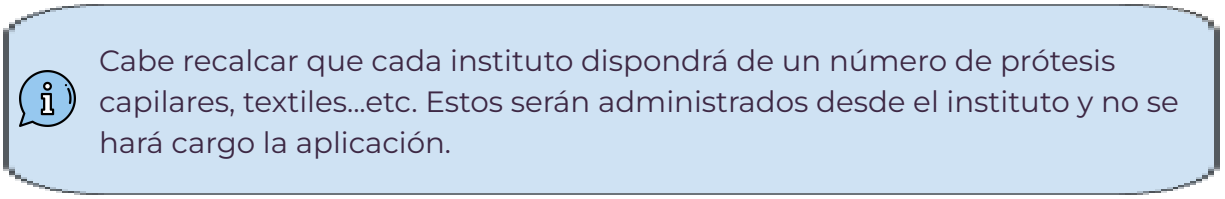
1.2 Breve descripción del proyecto.

Mi proyecto final **Pelucas Solidarias** está realizado en colaboración con el alumnado de 2º curso del GS de Estilismo y Dirección de Peluquería del centro IES la Vereda. Además de ser un proyecto candidato a su uso por parte de la Generalitat para ampliar su servicio en todos los centros afiliados a este.

1.2.1 Objetivos y requisitos del proyecto.

El proyecto nace para dar respuesta a las necesidades de personas que están sufriendo alguna enfermedad o **no disponen de recursos económicos** para costearse una peluca o pañuelo oncológico.

La idea consiste en solicitar un producto que pueda ser recogido en alguno de estos centros afiliados, y así **satisfacer la necesidad de la persona que lo necesite**.



1.2.2 Uso del proyecto.

Dentro de la web se debe **dar información** de que es lo que se ofrece, que centros están afiliados, en que consiste nuestro proyecto y ofrecer tanto un número de contacto como las redes sociales con información útil de como adaptar la prótesis, ponerse un pañuelo Ontológico entre otros.

Cualquier usuario registrado tiene la posibilidad de **diseñar y solicitar un producto**.

Además, al realizar la solicitud se le ofrece la opción de pedir un **servicio gratuito** que será ofrecido en el centro por el alumnado de 2º curso del GS de Estilismo y Dirección de Peluquería (servicios de peluquería o estética).
Tambien se indicará la **disponibilidad horaria** para la recogida de el/los producto/s y elegir el **instituto** donde se desea recoger, como tambien datos necesarios de los usuarios como sus **datos clínicos** o **medidas craneales** (necesarios para la correcta adaptación).

Claramente los empleados dispondrán de una interfaz de admin donde podrán **administrar** cualquiera de los datos. Estos tendrán la posibilidad de **informar a los usuarios** de cualquier imprevisto o información gracias a las notificaciones.

Alejandra se ha registrado en Pelucas Solidarias y ha diseñado una prótesis capilar y un pañuelo Ontológico añadiendo tambien un servicio de lavado facial. Indicando que lo quiere en el instituto IES la Vereda, pero solo puede ir a recogerla por la tarde porque por las mañanas trabaja. Además ha indicado que padece dermatitis.

Caso práctico:

El papel del administrador es obtener el diseño y los datos del usuario.
Con estos puede consultar el stok de pelucas del instituto y ver si tienen una prótesis y pañuelo que se asemeje. Si es así se buscará adaptar la prótesis y el pañuelo a las medidas craneales del usuario.
Si se cumple con los requisitos se le aceptará la solicitud y se le enviará un mensaje informando de la hora y el días de la recogida, que podría ser a las 5PM ya que sabemos que lo tiene disponible.

Además sabiendo que padece dermatitis podemos informarle del proceso que conlleva el lavado facial, adjuntando un enlace de un vídeo de nuestro canal de youtube, y si este le puede afectar a la piel.

Una vez llegado el día de la recogida puede optar en realizase el servicio ese mismo día si es posible, o llegar a un acuerdo (Esto ya no depende de la página web)

Las personas pueden donar productos en los institutos y el administrador puede añadir a la cuenta de ese usuario el rol de donante.

2 Estudio previo.

La idea de realizar este proyecto no ha sido idea mía. Fue una invitación por parte de una profesora a tomar la decisión de desarrollar una idea a raíz de un **proyecto a futuro presentado por la Generalitat**.

2.1 Motivación en la empresa y autoeficacia de Ilgawson

Es por ello que he estado en contacto con una profesora especializada en la familia

profesional de imagen personal que me ha explicado el proyecto que querían realizar y que idea tenía en mente que dispusiese la página web.

Tras **debatir las funciones** que debía cumplir la página web llegamos a la conclusión anteriormente descrita en el punto *1.2.2 Uso del proyecto*.

Se pedían excesivas funcionalidades que en el tiempo de 40h de proyecto eran costosas de desarrollar. Por ejemplo, de sepida un blog, una página de tips ...etc.

Estas no eran fundamentales y no han sido implementadas en la página Web.

2.1 Estudio de soluciones existentes.

Para satisfacer la idea de comunicar para que sirve la página web, es necesario crear **páginas básicas de información** para los visitantes. Donde se explique que se puede realizar en esta y que servio ofrece. Las más necesarias són la de los Centros afiliados, el Home y un About us.

Esta claro que el objetivo del proyecto es **ofrecer al cliente los formularios necesarios** para sacar la información que necesitamos saber para poderle ofrecer el artículo que desee adaptado a sus medidas y necesidades

Además de **datos personales** como su número de teléfono, correo electrónico, código postal para poder estar en contacto con este.

A todo esto se debe crear un **menú administrativo** para que los encargados de los centros tengan las herramientas necesarias para administrar las solicitudes recibidas y cualquier atributo que se quiera modificar (o bien porque se quiere añadir algo, o por si hay un error y se quiere modificar).
Este menú administrativo constará de muchas **tablas con filtros** para facilitar la búsqueda y agilidad al administrador. Y presentación de **gráficas** para hacerse a la idea de la situación de la aplicación

Para mantener informado de cualquier novedad he decidido crear un **sistema de notificaciones**, donde el administrador puede enviar un mensaje con una cabecera, un cuerpo y tipo de mensaje.

3 Plan de trabajo.

Adentrarse en este proyecto ha resultado un reto complejo ya que es la primera vez que desarrollo un aplicación Web sin tener claros los requisitos mínimos desde un principio.

Como he comentado en anteriores puntos, el desarrollo de **mi proyecto dependía de un cliente** el cual no tiene conocimientos de informática. Por lo tanto no tiene la capacidad de deducir el tiempo y la complejidad de la idea que está pidiendo.

Tras estudiar la situación deduje estos problemas:

- ➔ Me encuentro ante un **entorno complejo.**
- ➔ **Necesito obtener resultados pronto** para poder verificar si lo que estoy desarrollando se ciñe a las necesidades.
- ➔ Los requisitos son **cambiantes o poco definidos**.

Es por ello que me he decantado en usar una metodología ágil. Mas en concreto la metodología **Scrum**.
En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija. En mi caso he puesto una duración de 1 semana por ciclo.
Según la teoría de Scrum, existen 3 roles.

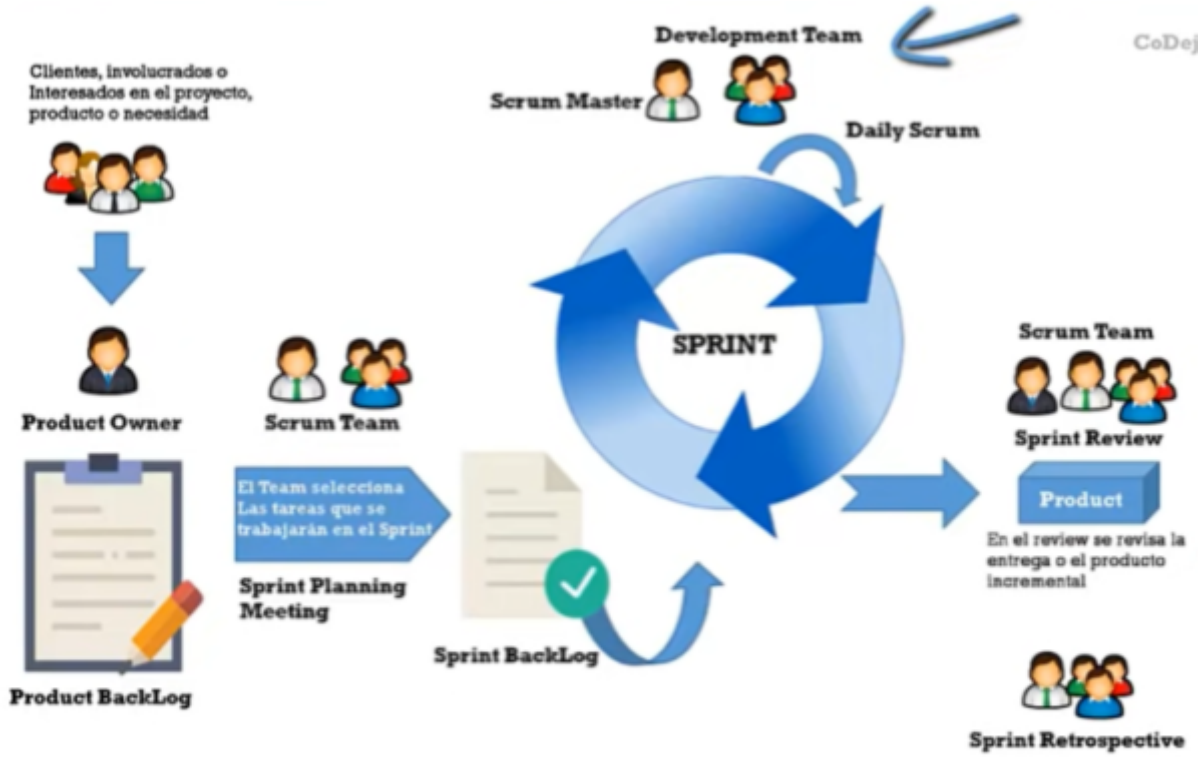
- ★ El product Owner (Dueño del producto)
- ★ El SCRUM Master
- ★ El Development Team (Equipo de Desarrollo)

El **product owner** es la representación del cliente dentro del equipo de trabajo.Y su principal responsabilidad es expresar claramente la necesidad del cliente.

El **scrum master** es el responsable de asegurar que el scrum es entendido y realizado al asegurarse de que el equipo trabaja ajustándose a la teoría,prácticas y reglas de Scrum.

El **equipo de desarrollo** que se compone de las personas responsables de dar cumplimiento a los ciclos o SPRINTS.

Como el proyecto solo lo he realizado yo sin ningún grupo de trabajo, he tomado todos los roles.

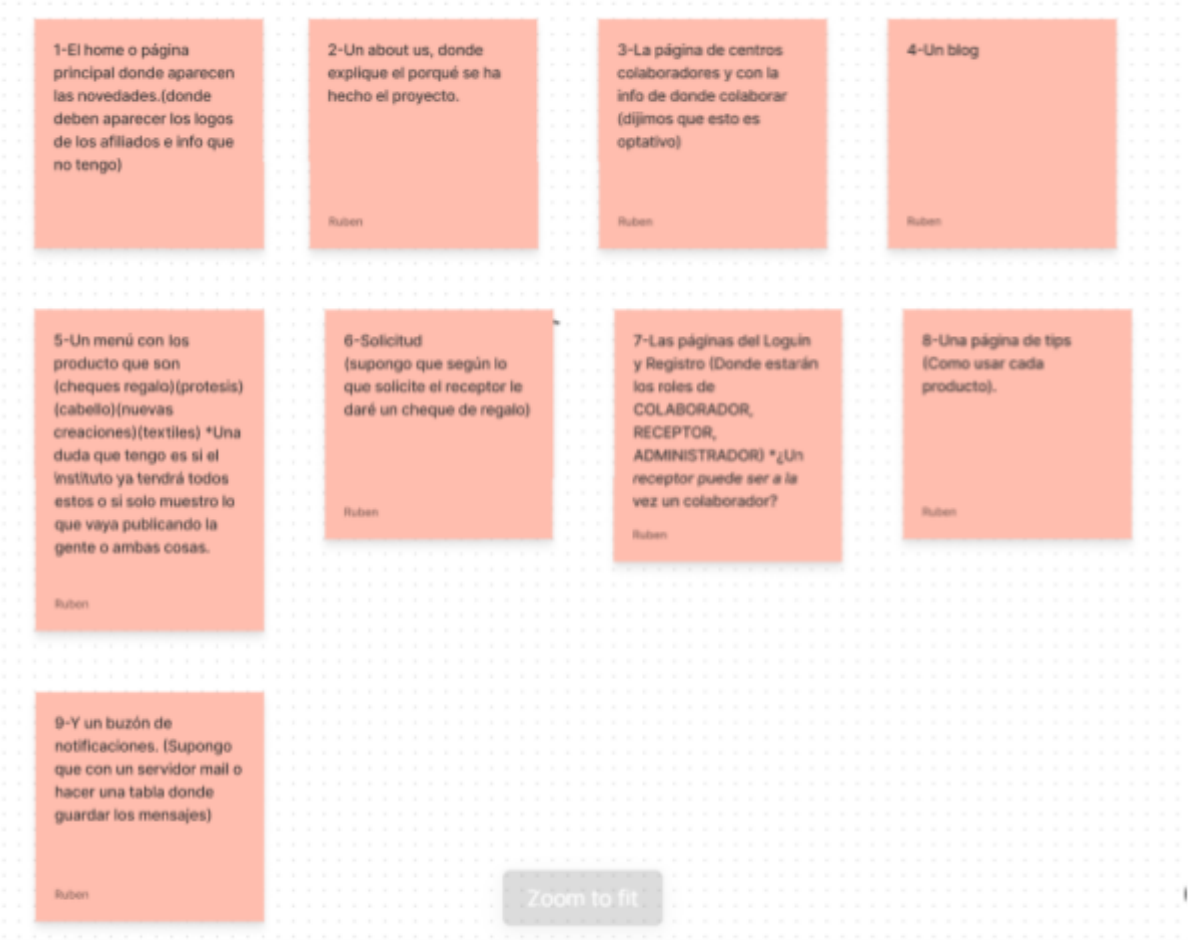


3.1 - Esquema metodología Scrum de Cristian Henao

Hay que entender el ciclo de vida de Scrum.

Yo siendo el **product owner** me encargo de desarrollar una lista o Product Backlog donde se encuentran las necesidades completas del cliente.

En el **product Backlog** se plasman todas las necesidades, ideas y requisitos que harán cumplimiento a la necesidad del cliente.



3.2 Product backlog creado desde figma

Posteriormente ese product backlog es entregado al equipo de desarrollo scrum master. Eso se hace en una reunión llamada **sprint planning meeting**.

A resultado de esa reunion se genera el **sprint backlog**, esto es una serie de funcionalidades o conjuntos de requisitos que se deben construir en un tipo de 1 o 2 semanas. Este tiempo es denominado el **sprint** (que en mi caso tiene una duración de 1 semana cada sprint)

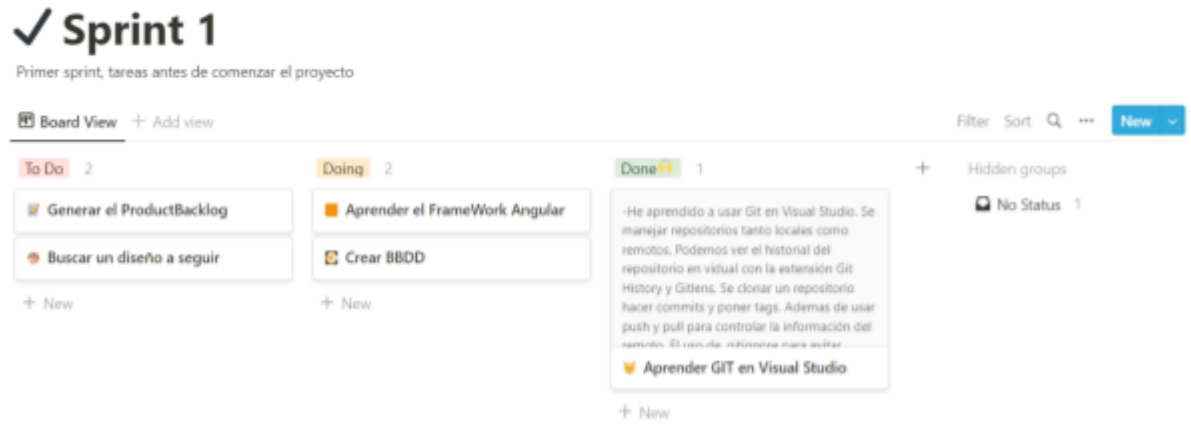
El sprint es el corazon del scrum, ya que corresponde al proceso de desarrollo o construcción de la necesidad del cliente pero divididas en un módulo funcional o producto incremental.

Dentro del sprint intervienen el scrum máster y el team development. El team developer son los encargados de desarrollar y construir esa necesidad que define el sprint.

Una de las actividades mas representativas del scrum son los **daily scrum**. Estas reuniones tienen como objetivo hacer seguimiento diariamente a todos los procesos que tengamos dentro del sprint.

Reuniones con preguntas muy puntuales preguntando que se hizo ayer, que se está haciendo hoy, que voy a hacer mañana y que problemas se encontró.

De normal estas reuniones se realizan alrededor de un tablero. **Sirve como sprint backlog**.



3.3 - Tablero creado desde una aplicación web llamada notion

Al finalizar el sprint se hace otra reunión llamada Scrum Review para verificar el cumplimiento de las metas o los objetivos del sprint en cuestión. Y así **garantizar la entrega del producto al cliente final**.

Una vez acabo del sprint se comienza otro nuevo tomando otra de las funcionalidades del product Backlog para sacar nuevamente el Sprint backlog.

Toda esta explicación ha sido la que he llevado a cabo con el proyecto Pelucas Solidarias, tomando el rol de Scrum Master y Team developer.

Tambien diseñé un diagrama de Gantt para organizarme en el tiempo.

3.4 Diagrama de grantt creado desde anasa

4 Diseño

El diseño en una **página dedicada al público** es un aspecto muy importante y el cual le he dado mucha prioridad.

4.1 Diseño general.

Como la página esta pensada para el uso principal de mujeres me he decantado en tomar una paleta de colores con colores primarios como el morado, rosa y azul.

4.1 Paleta de colores azul

4.2 Paleta de colores rosa

Estos son los colores generales que he usado en mi página web, que a parte de ser bonitos, combinan bien ya que estan juntos dentro del círculo cromático.

Para hacer más óptimo y rápido el desarrollo del proyecto he dedicado tiempo a aprender **Figma**. Es un editor de gráficos vectorial y una herramienta de generación de prototipos, principalmente basada en la web.

Los **beneficios** que me ha aportado figma es que he podido mostrar de manera gráfica el diseño de la página web y tener opinión directa del cliente para diseñarla a su gusto sin necesidad de picar código o realizar bocetos a mano.

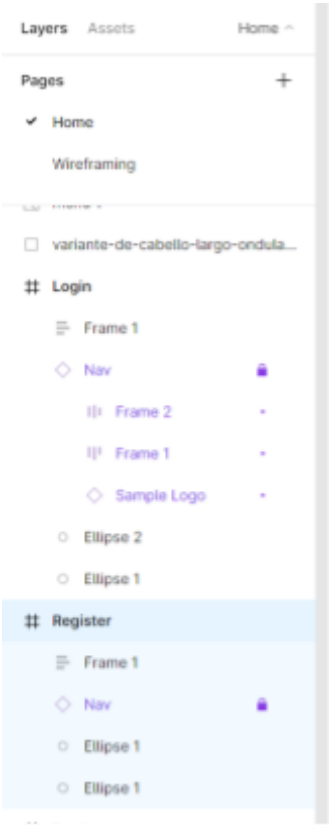
Aparte, en figma se diseña siguiendo la filosofía de la programación. ¿A qué me refiero con esto?

Como se puede ver en las imágenes, en figma se pueden diseñar páginas donde dentro puedes añadir **componentes** que tienen una gran similitud a las etiquetas del lenguaje de marcado HTML.

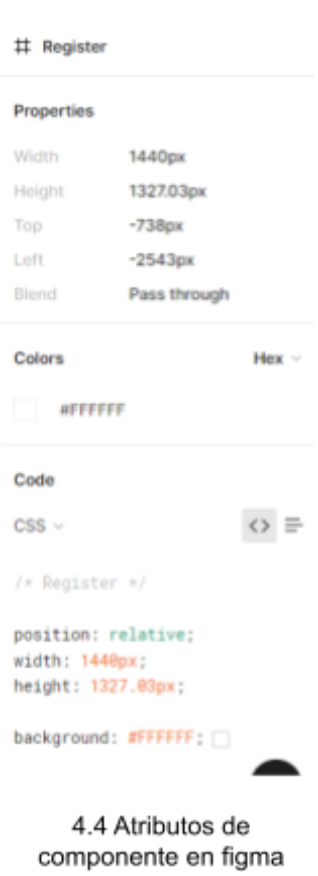
Además, a la hora de **añadir estilos a estos componentes**, ya sea de posicionamiento o como el estilo, este será generado (4.4).

Por lo tanto, el haber dedicado tiempo al diseño en esta aplicación hace que luego la implementación al código sea bastante sencilla e intuitiva.

Angular es un framework basado en componentes y esta herramienta resulta muy complementaria.

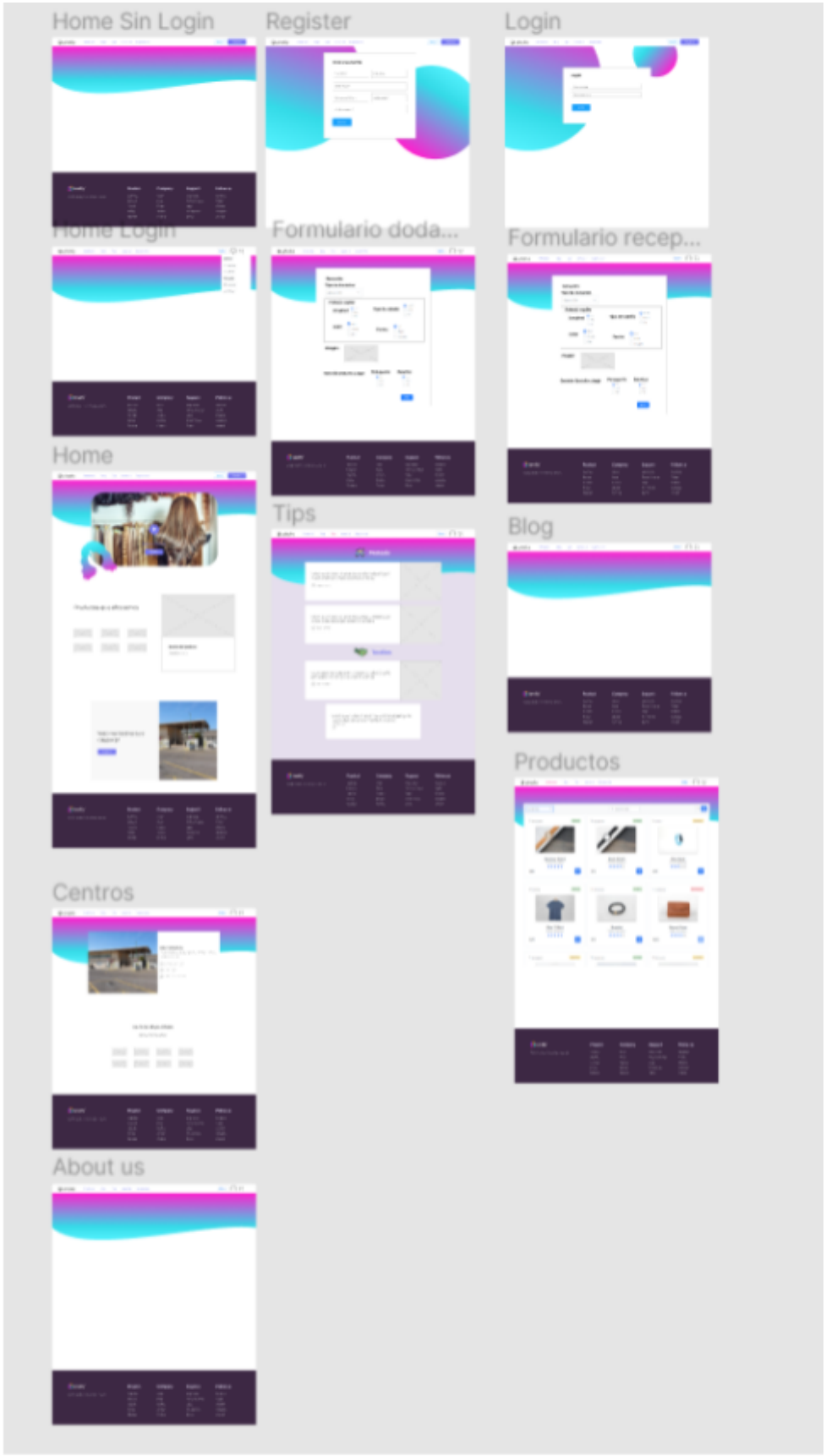


4.3 Menú de figma



4.4 Atributos de componente en figma

Este sería el resultado del diseño de mis páginas:



4.5 Diseño Pelucas Solidarias

4.2 Diseño detallado.

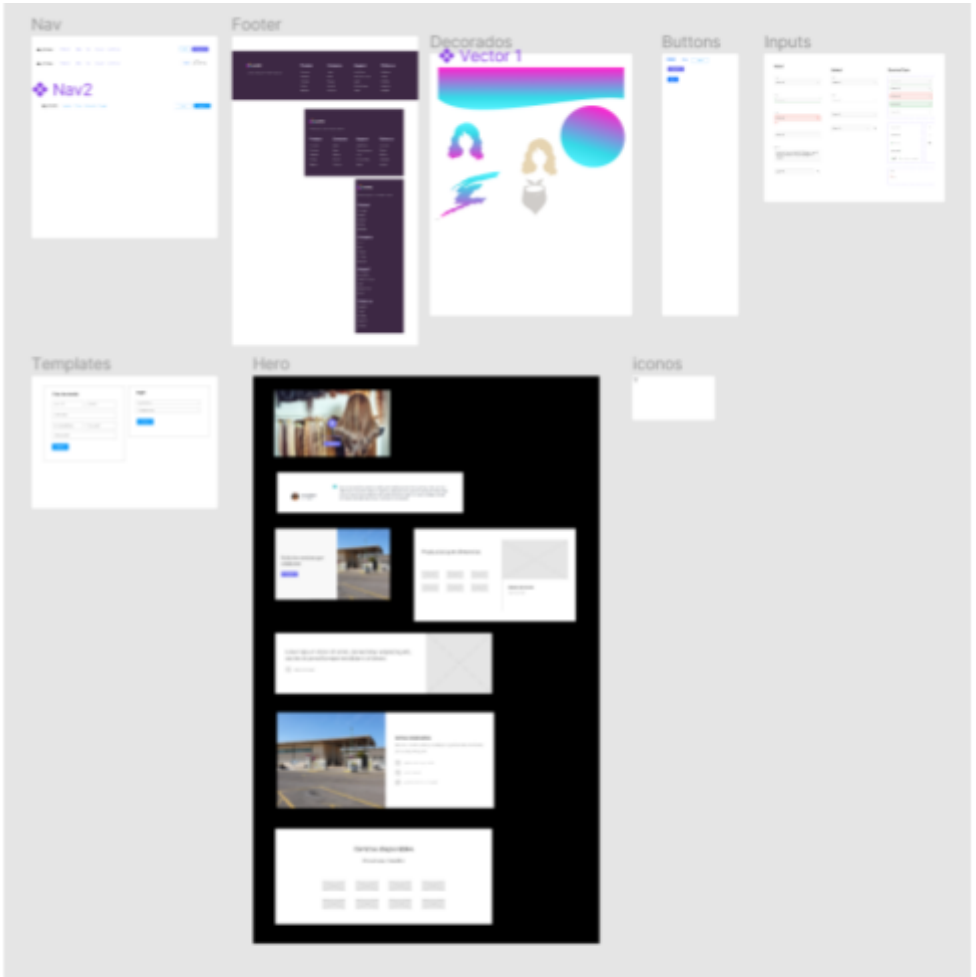
¿Conoces que es un **Wireframe**?

Un website wireframe, también conocido como un esquema de página o plano de pantalla, es una guía visual que representa el esqueleto o estructura visual de un sitio web.

Si nos ponemos a mirar con lupa el diseño de Pelucas Solidarias podemos observar que todo busca el mismo diseño.

El ejemplo más común son los **inputs/formularios** que todos buscan la misma apariencia. O mismamente los **botones** según su funcionalidad tienen un color u otro.

Para buscar esta armonia y no crear componentes muy diferentes entre ellos y así buscar una “imagen de marca” he creado un **Wireframe** en figma donde aparecen todos los componentes separados por funcionalidades o tipos.



4.6 Wireframing de Pelucas Solidarias

Como se puede observar, desde figma he creado diferentes páginas como en la imagen (4.5) pero con una funcionalidad diferente.

> En figma se pueden crear main componentes. Estos componentes podrán tener hijos los cuales no podrán ser modificados individualmente. Estos heredarán los cambios que se le aplican al padre

Las páginas **dividen los tipos de componentes padre que se van a guardar**. Para así, si en algún momento se desea reutilizar el mismo componente tenerlo a mano.

Las que más he usado es la de los decorados, botones y el de los hero.

Donde para los decorados he usado **gráficos vectoriales escalables svg** para mejorar la calidad de estos.

5 Implantación

En la implantación se van a hablar de temas más técnicos como el cómo y qué herramientas he usado para el desarrollo de la misma.

El desarrollo técnico de la aplicación está principalmente **pensado para un proyecto a futuro** y **fácil de escalar**.
Tanto las herramientas usadas como la jerarquía de carpetas y la forma en la que ha sido organizado ha sido pensado para un **posible desarrollo en equipo**.

La gran mayoría de las herramientas, por no decir casi todas no las he dado en el instituto, por lo que **he dedicado una gran parte a la investigación y aprender nuevos conceptos**.

Las dos aplicaciones han seguido un sistema de control de versiones **Git**.

La arquitectura que he usado en esta aplicación es la de una **aplicación frontend** que se comunique a una **API REST**.

5.1 En el servidor.

En la parte del servidor he usado como lenguaje de programación **JavaScript**, acompañado de un superconjunto de JavaScript llamado **TypeScript**, que esencialmente añade tipos estáticos y objetos basados en clases.

Js es un lenguaje muy flexible, TypeScript ayuda a transformar Js en un lenguaje de tipado fuerte y **hacer más robusta la aplicación**.

Para ello he usado la tecnología **Node.js**

¿Que es Node.js?

Es **un ambiente de ejecución** de JavaScript. Utiliza un **modelo de entrada y salida sin bloqueo controlado por eventos**, de esta manera lo hace un entorno ligero y eficiente.

Node.js cambió la forma en que programamos JavaScript ya que ahora todo nuestro código debe funcionar de manera **asíncrona a partir de eventos**.

¿Que beneficios trae Node.js?

- ★ La compilación de Node.js se realiza en tiempo de ejecución, **Just In Time (JIT)**, esto trae consigo una mayor optimización a las funciones que más veces sean llamadas.
- ★ Mediante clusters permite tener una **escalabilidad alta**.
- ★ Podemos **expandir nuestro código añadiendo módulos de forma fácil gracias al Node Package Manager (NPM)**.
- ★ Un **alto rendimiento** en proyectos donde necesitemos ejecución en tiempo real.
- ★ Podremos realizar front-end, back-end y hasta una aplicación móvil con un mismo lenguaje.

Dicho todo esto el **framework** que he usado para el Backend ha sido **Express.js**

Es un marco de trabajo de aplicación web Node.js mínimo, flexible y de código abierto. Se usa sobre Node.js para garantizar una mejor funcionalidad web. Express es el marco web más popular de Node.js.

Proporciona un **amplio conjunto de funciones para crear aplicaciones web** (de una sola página, de varias páginas e híbridas). Con Express, **puede estructurar una aplicación web que pueda manejar múltiples solicitudes HTTP** en una determinada URL.

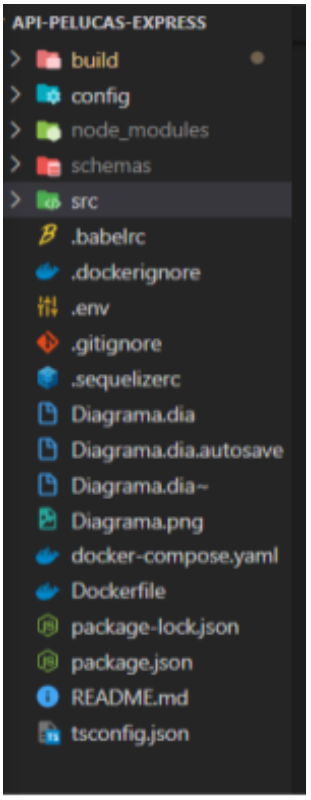
La flexibilidad es visible en numerosos componentes accesibles en un administrador de paquetes. Estos componentes perseveran automáticamente en Express.js.

La razón por la que Express es el marco web más popular es que facilita el desarrollo de aplicaciones web, sitios web y API.

5.1.1 Jerarquía de carpetas.

Antes de explicar el contenido y el uso de las herramientas usadas voy a explicar la jerarquía de carpetas del Backend para el framework Express Js.

Resumen general:



5.1 Jerarquía de carpetas Backend

build
Como esta programado con Typescript, este compila los archivos a JS y los introduce en build (esto se indica en el archivo de configuración de typescript tsconfig.json)

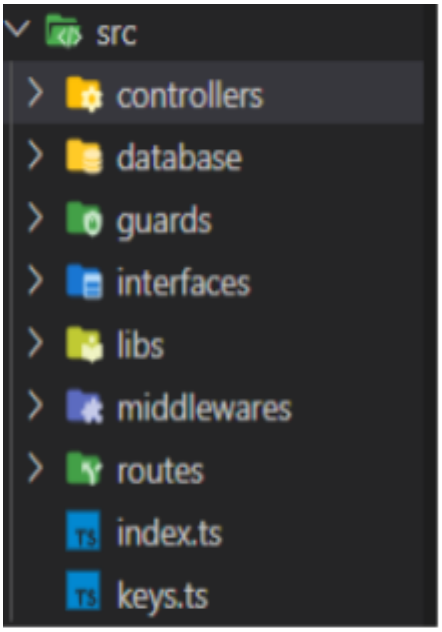
config
Como dice el nombre de la carpeta guardaré los archivos de configuración, tanto para el token como para la base de datos.

node_modules
Esta carpeta es generada por NodeJs tras realizar el comando npm init, en esta se encuentran tanto las herramientas necesarias como las que me instalo a parte.

schemas
Aquí guardo la base de datos, es un volumen creado con docker para ayudarme en el desarrollo.(Esto lo explicaré mas adelante)

src
Dentro se encuentran todos los archivos de la aplicación

Directorio src



5.2 Directorio src

controllers
Aquí se encuentra la lógica de las peticiones de cada uno de las tablas de la base de datos.

database
Se encuentran los modelos, migraciones y seeders del ORM Sequelize.

guards
Los guards sirven para proteger las peticiones.

interfaces
Guardo las interfaces creadas con TypeScript que me sirven para tener los atributos de los modelos.

middlewares
Sirven para filtrar o añadir una función extra a la lógica de los controladores en las rutas. Los guards son middlewares pero con finalidad diferente

routes
Lugar donde creo las rutas que llaman a un controlador. En estas indico el tipo de peticiones http.

El archivo **index.ts** es donde se encuentra el servidor, en este he creado una clase con métodos para su configuración, poner las rutas, conectar el ORM y finalmente crear un objeto de la clase del servidor y arrancarlo.

En **keys.ts** se encuentra un objeto json con claves para la base de datos, obtenidos desde las variables de entorno .env

Lo que he explicado digamos que es un resumen del “tronco” del backen. Con la finalidad de saber navegar por la aplicación cuando explique otras cosas.

5.1.2 Base de datos.

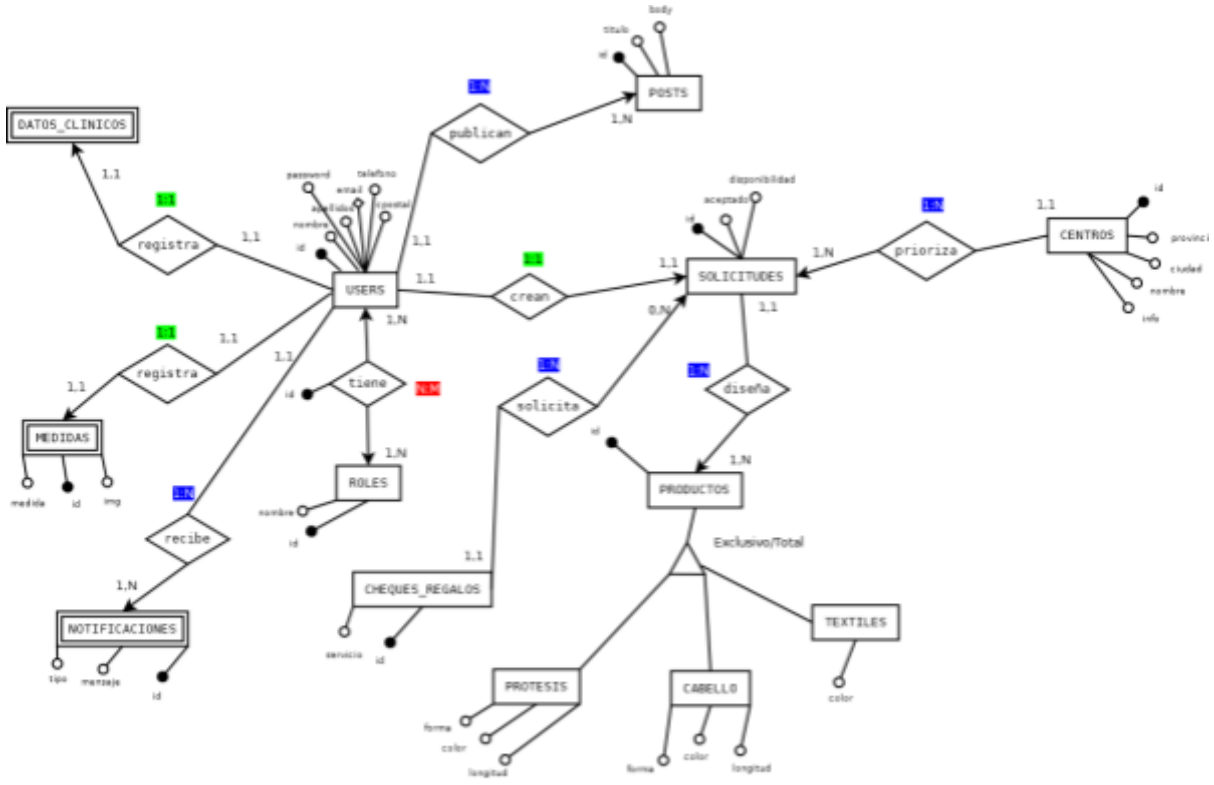
De normal las aplicaciones creadas en NodeJs suelen ser acompañadas por bases de datos no relacionales como MongoDB. Pero es una tecnología complicada de escalar.

Es por ello que me decantado por una hecha con **MySQL**.

La base de datos ha sido una de las **primeras cosas que he empezado a diseñar** cuando ya tenía una sólida idea del proyecto y las funcionalidades.

Para crear la base de datos he tenido que diseñar un **modelo E/R** donde se abarquen todas las necesidades planteadas en el product backlog.

Este seria el modelo:



5.3 Diagrama entidad/relación creado en Dia

Todos los diagramas y documentos de la aplicación estan publicados en el repositorio de GitHub. Repositorio en el punto 6.1

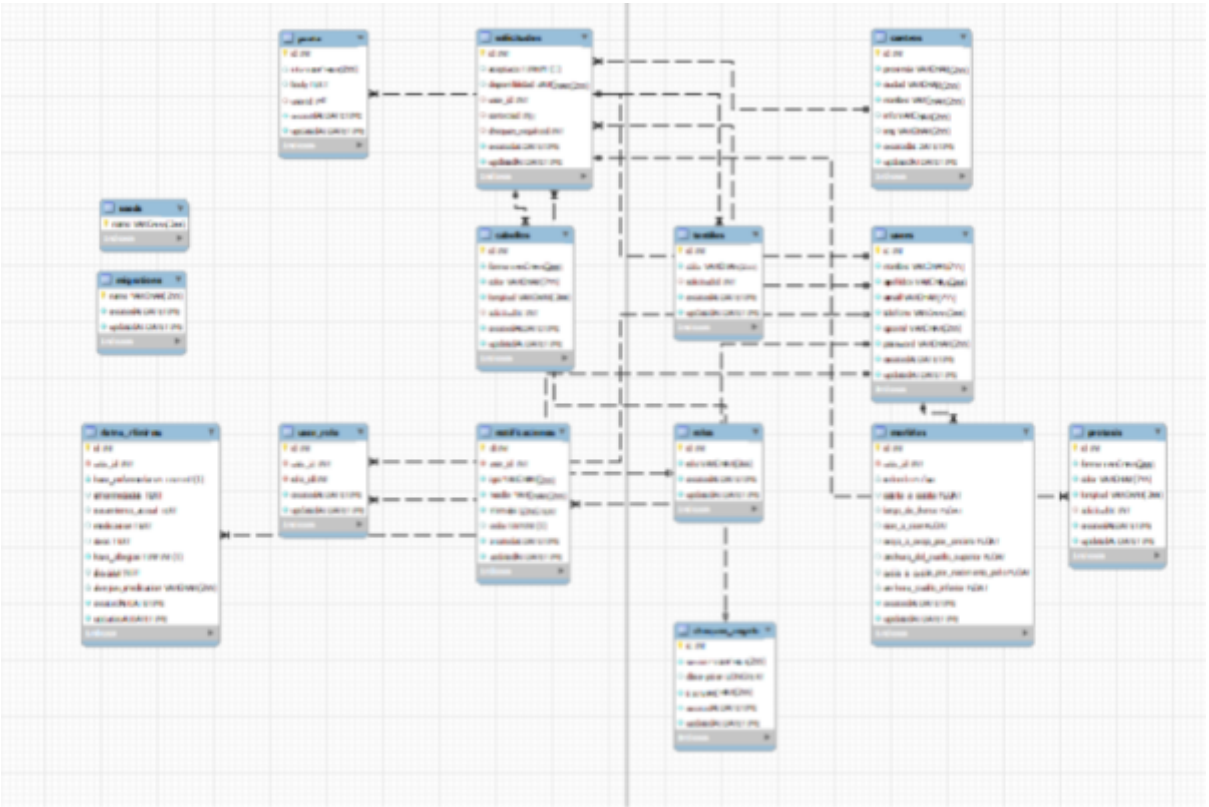
Es una base de datos algo extensa pero necesaria, la función principal de la web es administrar, por lo tanto se necesita el almacen de bastantes datos.

Las relaciones en color verde son 1:1, las que estan en rojo N:M y la de azul son 1:N

Contiene **entidades débiles** y al ser un modelo entidad relación extendido tambien contiene una **generalización**.

La entidad Post esta creada pero no tiene uso en la aplicación, ya que és se buscaba un blog en la web pero al final no se implementó.

También he generado un diagrama de la base de datos.



5.4 Diagrama creado con workbench

Aparte de las tablas mostradas en el modelo E/R existen dos tablas mas.

La de migraciones y los seeds que son creadas por el ORM Sequelize, explicado más adelante.

El programa que he usado para el manejo de base de datos ha sido **Workbench.**

5.1.3 ORM Sequelize

A la hora de buscar información directamente con consultas SQL puede llegar a resultar costoso y empalagoso.

Para agilizar el proceso, he decidido usar un **ORM** (Object Relational Mapping).

> ¿Que es un ORM?
> Un ORM es un modelo de programación que **permite mapear las estructuras de una base de datos relacional sobre una estructura lógica de entidades** con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones.
>
> Por lo tanto, las estructuras de **la base de datos relacional quedan vinculadas con las entidades lógicas** o base de datos virtual definida en el ORM, de tal modo que las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM.

Conociendo las ventajas de un ORM me puse a investigar entre todos los existentes cual es el mejor en proyectos con NodeJs y Express. Llegué a la conclusión de usar el ORM **Sequelize.**

Esta tecnología **no dispone de una gran comunidad**, por lo tanto aprenderla desde vídeos o foros ha resultado un reto.

En cambio, sequelize dispone de una documentación oficial en inglés que resulta útil pero es bastante técnica y se necesita conocimientos previos para comprenderla rápidamente.

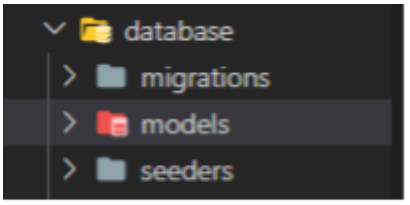
La creación de modelos u otros archivos se pueden crear a mano, pero para evitar esto he hecho uso de un **CLI que dispone Sequelize**.

El cli lo instalas en tu proyecto y te brinda un comando por consola para la generación automática de archivos.

Adentrándonos en mi proyecto ¿Cuáles són los archivos que pertenecen al ORM y como funciona este?

Sequelize necesita de un ***archivo de configuración***, un ***archivo donde se indique donde se generarán los archivos*** y los propios ***archivos de los modelos, migraciones y seeders.***

Estos tres últimos se encuentran en src>database.
Aquí se encuentra el corazón del ORM.
He implementado un **sistema de migraciones** y **seeders** que me ha resultado muy útil a la hora de actualizar la base de datos en distintos ordenadores.



5.5 Carpetas sequelize

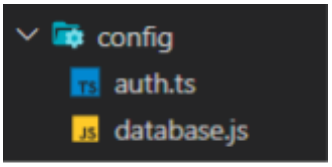
Las **migraciones** es un sistema de control de versiones pero para la base de datos. Y los **seeders** para bindear datos default en la base de datos.

En las **migraciones** se encuentran un archivo para cada tabla de la base de datos donde se indican los atributos sus relaciones. Pudiendo usar un comando del CLI y generarla en un solo paso.

Luego en **models**, se encuentran también distintos archivos para cada una de las tablas. Estos modelos nos sirven luego para darles uso en los controladores y hacer las consultas en la base de datos.

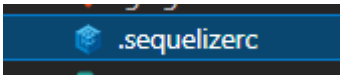
Estos modelos están **conectados entre si gracias al archivo index.js** que se encuentra dentro de la misma carpeta. Este destaca que gracias a la librería de node *fs* importa todos los modelos en el mismo archivo y a la hora de importar los modelos en los controladores solo llamo a index.js.
Además de **crear el objeto de conexión a la base de datos** usando la configuración. Este objeto llamado sequelize creará la conexión en el index.ts que es el main de la aplicación.

Antes he comentado que sequelize tiene un **archivo de configuración**, este se encuentra en *config>datadabase.js*



5.6 Configuración sequelize

Además el **cli de Sequelize** se le necesita marcar las rutas de donde quiere que se generen los archivos. Para ello se ha creado el archivo **.sequelizerc**. Se encuentra en el directorio raíz.



Las migraciones y los seeders generan una nueva tabla para cada uno donde se indica un historial de los archivos que se han aplicado

5.1.4 Sistema de autenticación basada en Token

Para el login de usuarios he usado una herramienta llamada **JWT** (Json Web Token).

¿Que es JWT?

JSON Web Token es un estándar abierto basado en JSON propuesto por IETF para la creación de tokens de acceso que permiten la propagación de identidad y privilegios.

El token es **un objeto JSON encriptado con una clave maestra**.

Cada vez que un usuario se registra o se loguea se genera un token con los datos del usuario y este es recibido por el frontend y se guarda como una sesión.

Gracias a un Middleware cada vez que se haga una petición a la API Rest se enviará el token como un header y este será comprobado en las rutas que tengan el middleware de auth (Es un middleware del backend que se encarga de verificar el token y dar acceso a la ruta que se desea pedir información, ya que para todas las rutas no se necesita estar autenticado).

El token necesita de una **clave maestra** para cifrar y descifrar. Además de indicarle el **tiempo de expiración** y **los rounds** que sería la dificultad de encriptación que quieres que tenga, cuanto mas grande sea el número mas encriptado estará.

Estos datos se encuentran en la carpeta config>auth.ts.

Todas las rutas que tengan el middleware auth tendrán siempre en el request el usuario logueado descodificado. Por lo tanto la creación de otros middlewares como por ejemplo de admin (para saber si el usuario logueado es admin y puede proceder a la petición) resulta mas facil de programar.

Cabe destacar que todo el tema del **registro y login** de usuario se lleva a cabo en el controlador auth. En este se han creado métodos relacionados con este ámbito. Por ejemplo existen herramientas para actualizar el token, obtener el usuario logueado... etc. Funciones que suenan sencillas pero que han ayudado mucho en el desarrollo.

Las contraseñas de los usuarios estan hasheadas para respetar la privacidad del usuario. Esto lo hago con la herramienta bcrypt

5.1.5 Estados de las peticiones.

El funcionamiento de **las peticiones consta de diferentes fases** para hacer la aplicación mas limpia y aseada.

Todas las rutas siguen este recorrido.

1. Primero **se crea la lógica de una tabla dentro de un controlador**, donde se encontrarán todos los métodos y los modelos de Sequelize para hacer las peticiones a la base de datos

2. Luego este controlador será importado en el archivo de rutas de la tabla que se encontrará en routes. Aqui se **creará un objeto router** de Express, donde **se añadirán las rutas hijas, el tipo de petición y el método al que llama del controlador**. Pudiendose añadir tambien un middleware.

3. Finalmente estas rutas serán importadas en el método routes del archivo **index.ts**. Aquí se añadirá la ruta padre y se importará el conjunto de rutas. Añadiendo tambien el middleware de auth en el paso que se solicite una autenticación.

```
this.router.post('/singin',authController.signIn)
```

5.7 Ejemplo ruta

Para el control más óptimo del resultado de las peticiones he tenido un **control de los estados**. Según el resultado que ofrezca el controlador se enviará la petición con el código de estado correspondiente. Por ejemplo si no encuentra nada el 404, si no tienes acceso el 401 ...etc.

5.1.6 Seguridad

Respecto a la seguridad del API Rest, como he explicado antes, he hecho **uso de los middlewares para el control de peticiones y de roles**.

Dentro de los controladores, en las peticiones CRUD he limitado el acceso. Por ejemplo, si soy un usuario cualquiera, no tener la capacidad de usar el método de buscar notificación por id y obtener una notificación que no pertenezca a él.

Los usuarios no tienen la posibilidad de realizar peticiones desde su ordenador con aplicaciones como Postman ya que he añadido la tecnología **CORS** a la aplicación. Que lo que hace es que simplemente bloqueé el acceso a las url que no esten registradas en el API. Solo se pueden hacer peticiones al API desde el dominio www.pelucassolidarias.tk.

Los **datos personales** del usuario en el JWT estan encriptados además de sus contraseñas.

5.2 En el cliente.

En la parte del frontend existe una gran variedad de frameworks con enfoques diferentes según el desempeño que tenga tu página web.

He seguido el mismo entorno que en el backend, es decir, el uso de NodeJs. Pero me he decantado en usar el framework **Angular**.

5.8 Logo del framework Angular

¿Porque Angular?

- ★ Lo que me ha hecho elegir Angular es por la capacidad de crear una página **SPA** o Single Page Application. El objetivo principal del desarrollo de aplicaciones de una sola página es una transición más rápida del sitio web. Un sitio web interactuará con el navegador web sustituyendo dinámicamente la página web actual con datos nuevos del servidor web.

- ★ Angular posee **un enlace de datos bidirecciona**l. Esto significa que cuando el marco se activa por un evento en el navegador web, puede hacer que el módulo y las acciones del usuario en la página web se actualicen y cambien los patrones necesarios y esenciales.

- ★ Como la arquitectura de angular se basa en módulos y componentes, **la escalabilidad, la coherencia y la reutilización de componentes** resulta muy sencilla.
- ★ Este framework tiene **una gran comunidad** y ha resultado sencillo resolver cualquier duda al respecto.

Para facilitar la creación de componentes, módulos, pipes ..etc He utilizado el **CLI de Angular**, este funciona en consola con el comando ng.

5.2.1 Jerarquía de carpetas.

Para entender como he trabajado con este framework y saber navegar dentro de este voy a explicar la jerarquía de carpetas.

Directorio raíz

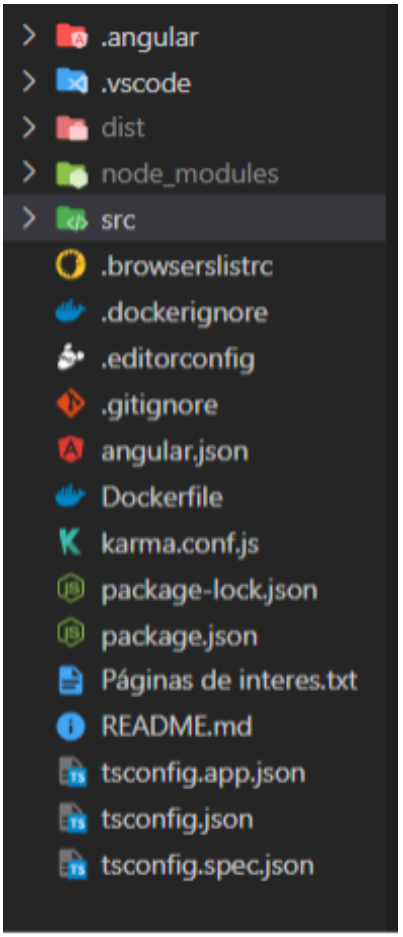
.angular, .vscode
Estas carpetas son creadas al hacer un npm init

dist
Aqui se encuentra la aplicación angular compilada cuando se usa el ng build.

node_modules
Se encuentran las herramientas utilizadas para desarrollar la aplicación.

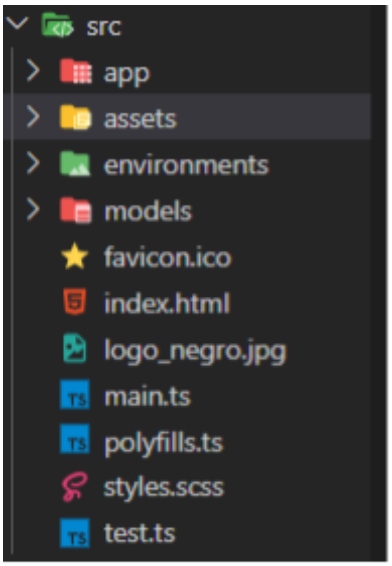
src
Se encuentra toda la aplicación.

angular.json
Es el archivo configuración de angular.



5.9 Jerarquía de carpetas angular

Directorio src



5.10 Carpeta src en proyecto angular

app
En esta carpeta se encuentra el módulo principal y los componentes.

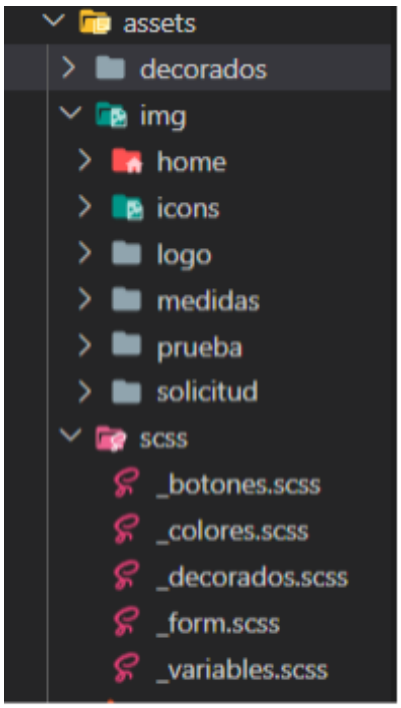
assets
Se encuentran las imágenes y hojas de estilos.

environments
Aqui se encuentran las variables de entornos, que se dividen en producción y desarrollo.

logo_negro.jpg
El logo del proyecto y el que uso para la pestaña del navegador.

index.html, styles.scss
Los archivos principales de la aplicación. Que al SPA el index.html irá cambiando según la ruta.

Directorio assets



5.11 Directorio assets Angular

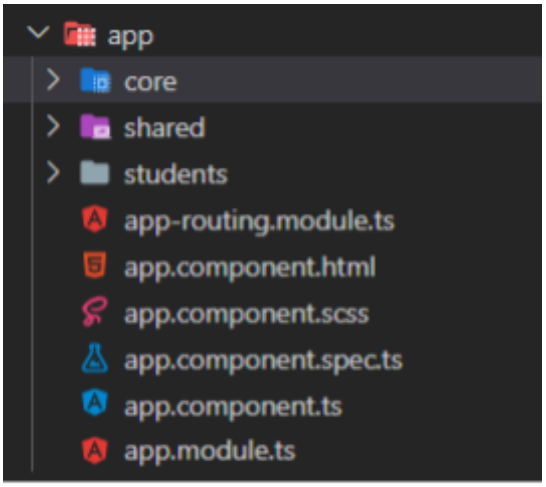
decorados
Guardo los svg o .png, .jpg de los decorados que he usado para realizar la página web.

img
Guardo las imágenes usadas en cada página de la aplicación.

scss
Como he usado SCSS, he creado distintos mixins, variables y estilos que luego importo en el styles.scss de la carpeta src.
Esto lo hago para tener el código mas organizado.

Directorio app

Para una mejor organización y para un posible trabajo en equipo he creado tres directorios para dividir el trabajo y la importancia de los componentes.



5.12 Directorio app de Angular

core

Aqui se encuentran todos los componentes padre y finales. Que son construidos como un lego, apartir de los componentes en shared.

shared

Aqui se encuentran componentes que funcionan como herramientas o heros que pueden ser usados en cualquier componente padre.

students

Se usan componentes pero para probarlos y testear.

Directorio app>core

components
Aqui separo por conceptos los módulos y sus respectivos componentes.
Dentro de cada una tendrá su propio módulo, su propias rutas, un template y una carpeta con los componentes.

constants
Aqui guardo objetos json con información que va a ser constante.

enums
Guardo objetos JSON y enums necesarios.

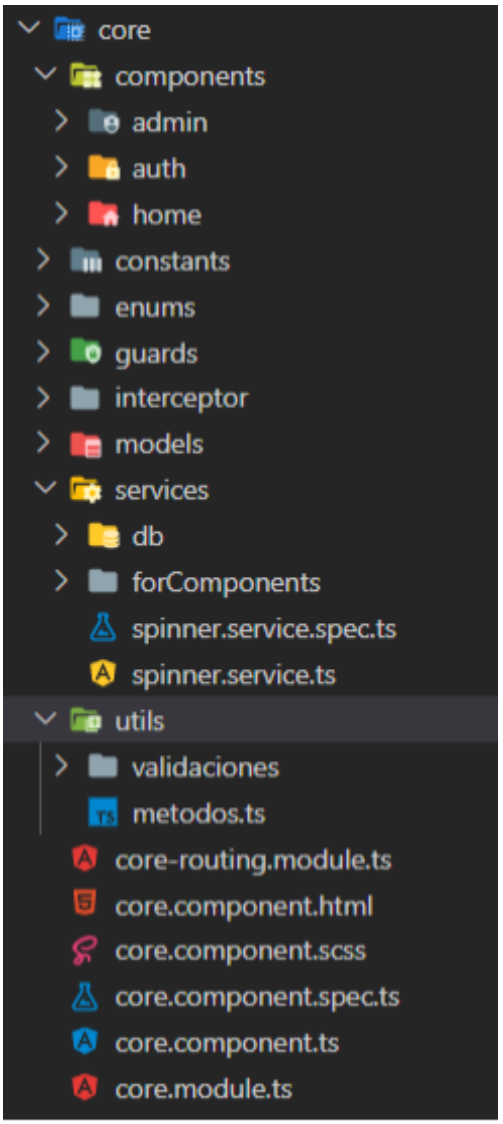
guards
Al igual que en el backend hago uso de guards para proteger algunas rutas.

interceptor
Sirven para añadir una función común a todas las peticiones. Por ejemplo añado siempre el token al header de la petición.

models
Almaceno las interfaces que me ayudan a conocer los atributos de los objetos recibidos por el backend.

services
Estan los servicios para la comunicación con la base de datos y ademas la comunicación entre componentes gracias a los subjects y eventEmitter.

utils
Guardo métodos, patterns, validaciones...etc.



5.13 Directorio core Angular

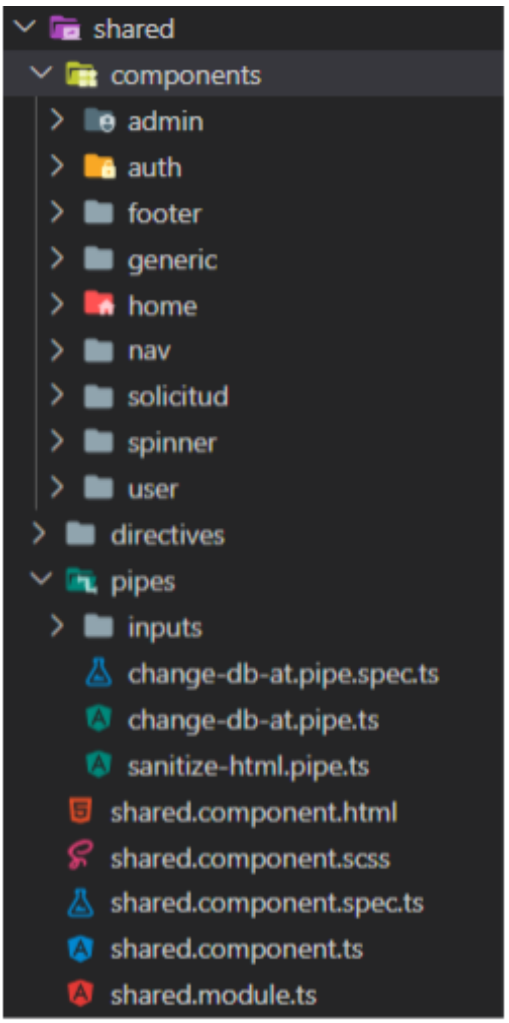
Directorio app>shared

components
Se encuentran los componentes que cumplen una función en concreto. Los separo por conceptos.

directives
No le he dado uso a esta carpeta

pipes
Creo mis propios pipe personalizados para modificar la salida de los datos que los usen.

Además en el módulo de shared importo las herramientas de primeng.



5.14 Directorio shared Angular

5.2.2 Routing

Recordemos que Angular es una aplicación SPA, esto significa que la página no se recarga nunca. Por lo tanto **cuando navegamos a una ruta lo que hace angular es cargar el componente que se le relaciona.**

Es por ello que hay crear el sistema de rutas. Las rutas en angular las creamos gracias a su módulo nativo de angular RouterModule.

Dentro de la carpeta app creo un módulo para las rutas padre llamado **app-routing.**

Este se comunica con el modulo de rutas dentro de core, llamado **core-routing**.

Y este último según como empieces la ruta cargará los componentes necesarios para esa ruta.

Ramificando se en los módulos de las rutas de **admin-routing ,auth-routing y home-routing**.

El hecho de cargar los componentes según la ruta en la que estes se denomina **Lazy Loading**. Esto ayuda a la carga mas rápida de estos y que la página sea mas fluida.

5.2.3 Colección PrimeNg

Para facilitar el desarrollo he utilizado una **colección de componentes de interfaz de usuario** para Angular llamado **PrimeNg**.

5.15 Logo PrimeNg

El proceso de aprender PrimeNg tambien ha sido un gran reto ya que he dedicado bastante tiempo en como manejar sus componentes. Ya que esta librería ofrece muchos componentes útiles pero hay que saber personalizarlos.

Ofrece herramientas como el carousel ,mensajes de alerta, tablas ...etc. Puede ser comparable al framework Bootstrap, pero ofrece el uso de JavaScript.

Gracias a usar PrimeNg me ha ayudado a comprender mejor como funciona la modularización de Angular y el uso de otras funciones.

Por ejemplo he creado mis **componentes de tablas personalizadas y muy generales** capaces de cambiar atributos gracias a los inputs que les envio.

5.2.4 Comunicación entre componentes.

Lo que voy a comentar son conceptos de Angular pero es necesario explicarlos si nunca has tocado este framework.

> En angular los componentes se tienen que importar y exportar para que sean usados por otros módulos. Y estos se llaman en el html con una etiqueta.

Los componentes que he creado en shared contienen atributos Inputs y Outputs.

Los **inputs** sirven para **asignar un valor a una variable del componente**.

Y el **output** es un evento EventEmitter que **esta a la espera de recibir un cambio del componente**.

Esto es bastante útil, pero cuando cuando surge el problema de comunicar muchos componentes no es una buena idea comunicarse de esta manera ya que se genera mucho código innecesario.

La solución a esto es el uso de los **observables** y **subjects.**

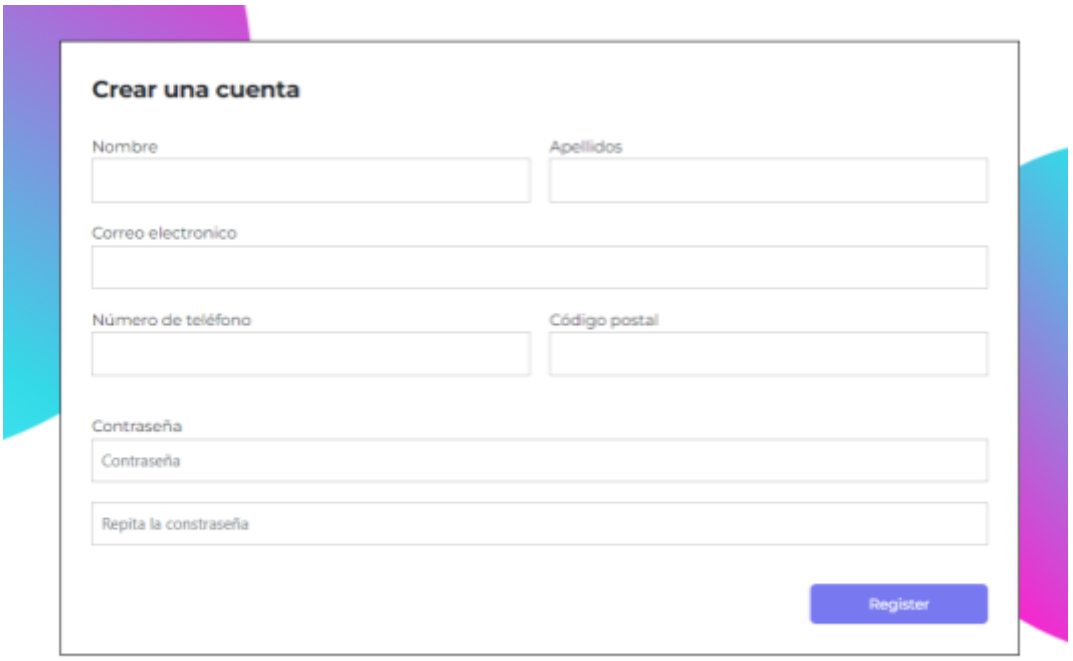
> ¿Que es un subject?
> Dado un subject, es posible suscribirse a él proporcionando un observer que comenzará a recibir valores. En su funcionamiento interno, el "subscribe" simplemente registra al observer en una lista de observers que "escuchan" valores del subject en cuestión.

Para usar esos observables o subjects **he creado servicios dentro de la carpeta services de la carpeta core.**

Me han ayudado mucho a la hora de desarrollar la solicitud donde he comunicado muchos componentes con distintos formularios.

5.2.5 Formularios

La naturaleza de Pelucas Solidarias es **ofrecer muchos formularios** para obtener la información necesaria para ofrecer nuestro servicio. Es por ello que los formularios cobran una gran importancia.



5.16 Formulario de registro Pelucas Solidarias

Para ello he hecho gran uso de las clases **FormGroup** y **FormBuilder.**

FormGroup me ha ayudado mucho para conocer el estado de los formulários, los errores que tiene y su validación.

Al crear un objeto de la clase **FormGroup** puedes añadir campos del formulario gracias al FormBuilder y añadir valores default con validaciones ya creadas con la clase Validators.

Aparte de poder **crear tus propias validaciones usando tanto patterns como métodos**.

Y **facilita mucho mostrar los mensajes de error personalizados** directamente en formulario y gracias al data binding al momento.

Podría explicar mas cosas pero se haría muy extenso.

5.2.6 Comunicación con el backend

Para conectarme al backend desde el frontend lo hago mediante el uso de **servicios.** Estos servicios se encuentran en la carpeta **db** de services.

La comunicación una vez creada el API REST es muy sencilla. **He creado un servicio para cada una de las tablas de la base de datos**. En estos únicamente cojo la url del backend y creo métodos llamando a las urls del backend con la clase nativa de angular **HttpClient.**

El servicio mas importante es el de auth. Ya que en este hago el control del token, almacenando lo en el localStorage, actualizando lo, obteniendo el usuario actual, deslogueartе .. ect.

5.2.7 Otras tecnologías

Aparte de los frameworks que he nombrado tambien he usado otras pero a menor medida.

En alguna ocasión he hecho uso de **JQuery** para **obtener el objeto del DOM** y modificarlo. No lo he usado mucho pero en alguna situación puntual me ha ayudado mucho.

5.17 Logo de bootstrap

5.18 Logo JQuery

Tambien he utilizado **bootstrap** para ayudarme a **estructurar el código html**.
Lo he usado en la mayoría de los componentes, utilizando el container, los row, cols .. etc.
A parte de **añadir algún pequeño detalle** rápido a alguna etiqueta HTML.

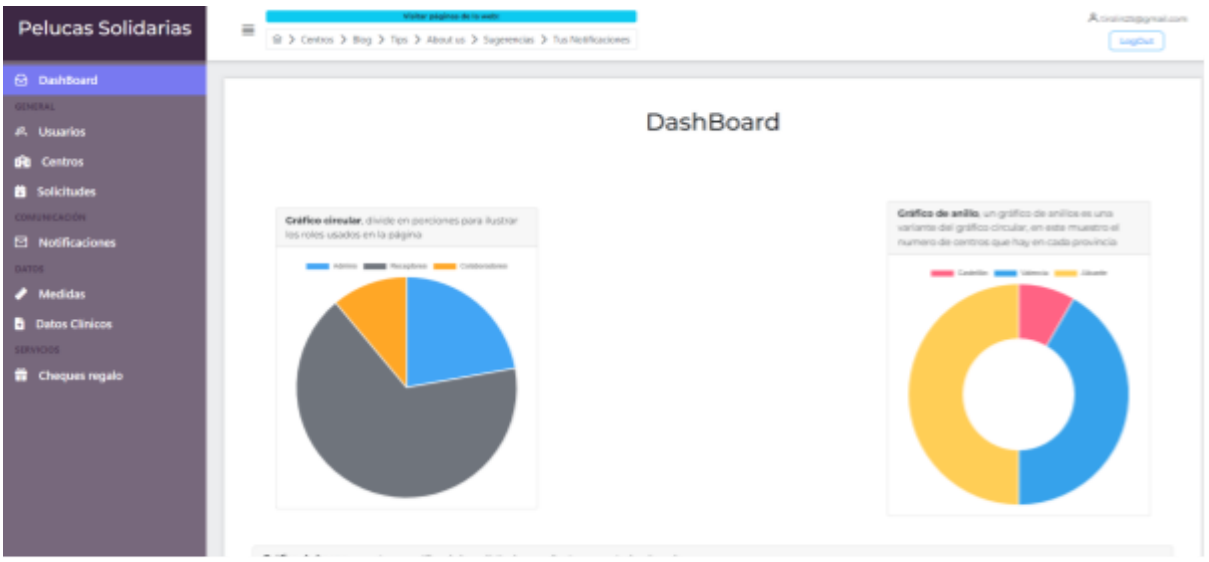
5.2.8 Responsive

Gracias al framework Bootstrap y a los media queries **he hecho que la página sea responsive**.

Toda la página es responsive, pero la página de administración, que son tablas, es mejor manejarlas desde el ordenador.

5.2.9 Páginas para los distintos roles.

La diferencia entre los distintos roles **admin**, **donante** y **receptor** es que el admin dispone de un panel admin donde podrá modificar toda la información que desee de los usuarios.



5.19 Panel administrativo Pelucas Solidarias

Luego los **donantes y el receptores** solo pueden navegar en la página principal pudiendo ver sus notificaciones,hacer una solicitud editar sus datos ..etc.

5.20 Formulario solicitud Pelucas Solidarias

Luego los usuarios no logueados no pueden solicitar, ni editar datos, ni ver sus notificaciones.

5.2.10 Uso de Snipper

Simplemente nombrar que he creado un sniper de carga que se activa siempre que esta en espera la petición. Esto lo he logrado con un interceptor.

5.2.11 Seguridad

Muy parecido al backend, las **rutas están protegidas por roles.**
Esto lo he logrado con los guards y añadiendo los en las rutas.

6 Recursos, Despliegue y Uso

En este apartado voy a hablar de las **herramientas software que usa cada aplicación.**

Los **requisitos hardware minimos** para su correcto funcionamiento.

Además del **entorno que he usado para desarrollar** en cada una y **como descargar, instalar y arrancar las aplicaciones**.

6.1 Repositorios GitHub y DockerHub

Respecto al sistema de control de versiones de GitHub **he creado dos repositorios públicos** donde en uno tengo el frontend y en el otro el backend. Esto lo he hecho para que se puedan apreciar bien los commits.

He usado ramas de desarrollo en cada uno de los repositorios y luego he hecho un merge a sus principales

Frontend

GitHub
https://github.com/urben88/cliente-pelucas-angular
Rama principal: master

DockerHub
https://hub.docker.com/repository/docker/urben88/frontend-pelucas

Backend

GitHub
https://github.com/urben88/api-pelucas-express
Rama principal: main

DockerHub
https://hub.docker.com/repository/docker/urben88/express-pelucas

6.2 Herramientas Hardware.

Al ser una aplicación web, **los requisitos del hardware no suelen ser muy elevados**.

Almacenamiento

El tamaño tanto del backend como del frontend varía considerablemente si tiene la carpeta **node_modules** instalada. Ya que ésta contiene todas las aplicaciones.

Al descargar las aplicaciones desde los repositorios github, esta carpeta no está subida ya que la debemos de instalar.

Pero esta carpeta en **IMPRESCINDIBLE** en el backend. Luego en el frontend solo es necesaria en su desarrollo (ya que luego el proyecto se compila)

Estos son los pesos de las aplicaciones:

- ★ Backend
  - Sin node_modules: 1MB
  - Con node_modules: **250 MB**

- ★ Frontend
  - Sin node_modules: 7MB
  - Con node_modules: **1,35GB**
  - Aplicación compilada en dist: **9,5MB**

Almacenamiento disponible mínimo: **2GB**

RAM

Como la aplicación no va a tener mucho tráfico la ram mínima son **2GB**

CPU

Al igual que la RAM cuanta más mejor pero como CPU mínimo recomiendo un **Intel BV80605001911APS LBLC**

6.3 Herramientas Software y su uso

El entorno de desarrollo que he usado para desarrollar tanto el frontend como el backend ha sido el **Visual Studio Code.** Ya que ofrece ciertas comodidades que otros IDEs no ofrecen. Como extensiones, terminal integrada, herramientas para usar Git, herramientas para usar docker, conectarse por ssh ...etc.

Por aquí dejo una lista de **extensiones** que he utilizado:

1. **Para el framework Angular**
    a. Angular 10 Snippets - TypeScript, Html, Angular Material, ngRx, RxJS & Flex Layout.
    b. Angular Language Service
    c. Angular Snippets (Version 13)
2. **Productividad**
    a. Auto Close Tag
    b. Auto Rename Tag
    c. Bootstrap 5 & Font Awesome Snippets
    d. Node.js Modules Intellisense
    e. Path Intellisense

3. **Diseño**
    a. Better Comments
    b. Figma
    c. Material Icon Theme
4. **Despliegue y conexión**
    a. Docker
    b. Remote - SSH
5. **Para GitHub**
    a. Git History
    b. GitHub Pull Requests and Issues
    c. GitLens — Git supercharged

6.3.1 Backend

Voy a **explicar las herramientas que he utilizado y que se necesita para su uso** (herramientas del package.json y otras)

Y luego que **pasos que debes seguir para instalar y arrancar la aplicación en local**.

Requisitos minimos

- Node Js
    - Debes tener instalado Node.js instalado en tu ordenador para poder usar el interprete de comandos npm
    - Versión utilizada en el desarrollo **v16.14.2**

- DB Mysql
    - He usado un contenedor docker con mysql instalado
    - Versión de mysql utilizada **mysql v5.7.25**

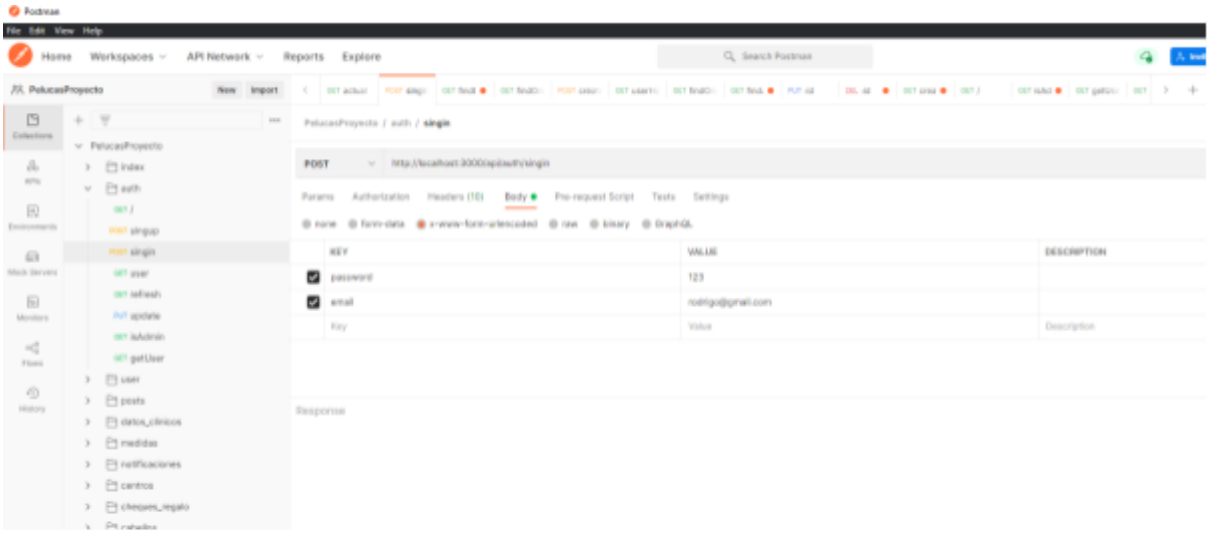
- Docker
    - A la hora de trabajar he usado un servidor de mysql en un contenedor docker por lo tanto para tener el mismo entorno es necesario tenerlo instalado (lo explico mas adelante)
    - En windows puedes descargar Docker Desktop

    - Versión docker utilizada **v20.10.13**

Éstas son las herramientas que he descargado desde npm en el desarrollo.

1. **bcryptjs**
    a. Esta herramienta me sirve para encriptar las contraseñas
2. **connect-timeout**
    a. Una herramienta que añado al servidor express añadir más tiempo de espera al backend. Ya que si te conectas a una base de datos lejana donde el tiempo de espera es alto, éste crashea con error timeout.
3. **cors**
    a. Como nombre en la seguridad, sirve para bloquear peticiones de dominios no deseados. Es un middleware para Express.
4. **dotenv**
    a. Me sirve para crear variables de entorno tanto de la aplicación como en el docker.
5. **express**
    a. Este es el framework que he utilizado para el desarrollo
6. **jsonwebtoken**
    a. Herramienta para la creación de los tokens y sus validaciones.
7. **morgan**
    a. Middleware de nivel HTTP, me sirve para ver en consola información de las peticiones recibidas al backend.
8. **mysql2**
    a. Necesario para conectarme a una base de datos mysql
9. **promise-mysql**
    a. Me sirve para crear promesas en mysql
10. **sequelize**
    a. El ORM utilizado.
11. **Nodemon**
    a. Herramienta de desarrollo que monitorea los cambios en el código fuente que se esta desarrollando y automáticamente reinicia el servidor.
12. **sequelize-cli**
    a. El cli de sequelize para generar los archivos automáticamente.

Otras herramientas que me han ayudado en el desarrollo del API Rest ha sido **Postman.**



6.1 Peticiones Pelucas Solidarias en Postman

Postman nos permite realizar peticiones de una manera simple para testear APIs de tipo REST propias o de terceros.

> He dejado en el repositorio de GitHub una carpeta llamada postman donde se encuentra un JSON con el workspace que he creado con todas las peticiones. Este se puede importar en postman y tener todas las peticiones de la aplicación.

Pasos para el despliegue del backend en local

1- Lo primero que tienes que hacer es clonar el repositorio de GitHub. Navegas en consola a la carpeta donde quieres que se cree la carpeta del proyecto y usas

6.2 Comando para clonar repositorio del backend

Comando: **git clone https://github.com/urben88/api-pelucas-express**

2- Teniendo Node Js instalado en el ordenador debes de instalar los node_modules. Para ello, dentro de la carpeta del proyecto usas:

6.3 Instalar los módulos

Comando: **npm install**

3- Ya está todo instalado ahora faltaría arrancar el servidor. Para ello debemos tener claras las variables de entorno de la aplicación.

```
#Docker variables

#!Para local
MYSQLDB_HOST=127.0.0.1
MYSQLDB_USER=root
MYSQLDB_ROOT_PASSWORD=urben
MYSQLDB_DATABASE=peliculasexpress

#!Para producción
MYSQLDB_HOST=pelucas-solidarias.corry4jqcr9s.eu-west-3.rds.amazonaws.com
MYSQLDB_USER=urben
MYSQLDB_ROOT_PASSWORD=pelucas88
MYSQLDB_DATABASE=pelucasexpress

# MYSQLDB_LOCAL_PORT=3306
# MYSQLDB_DOCKER_PORT=3306
# NODE_LOCAL_PORT=3000
# NODE_DOCKER_PORT=3000

#Auth Config
AUTH_SECRET=troloo      Urben, 2 months ago • Creo el auth (registrando usuarios
AUTH_SECRET=7d
AUTH_ROUNDS=10
```

6.4 .env de Pelucas Solidarias Backend

Se que en una falta grave de seguridad tener las contraseñas de la base de datos en el repositorio. Pero como este proyecto va a ser público y la base de datos de momento es de desarrollo podría restaurarla fácilmente.

Aquí tienes dos opciones:

1- **O creas un contenedor docker** con el servidor mysql y te conectas con la primera configuración en local (comentando las de producción)

```
#Docker variables

#!Para local
MYSQLDB_HOST=127.0.0.1
MYSQLDB_USER=root
MYSQLDB_ROOT_PASSWORD=urben
MYSQLDB_DATABASE=peliculasexpress

#!Para producción
# MYSQLDB_HOST=pelucas-solidarias.corry4jqcr9s.eu-west-3.rds.amazonaws.com
# MYSQLDB_USER=urben
# MYSQLDB_ROOT_PASSWORD=pelucas88
# MYSQLDB_DATABASE=pelucasexpress      You, 3 minutes ago • Uncommitted chan

MYSQLDB_LOCAL_PORT=3306
MYSQLDB_DOCKER_PORT=3306
# NODE_LOCAL_PORT=3000
# NODE_DOCKER_PORT=3000

#Auth Config
AUTH_SECRET=troloo
AUTH_SECRET=7d
AUTH_ROUNDS=10
```

6.5 .env para contenedor Docker

2- **O te conectas directamente a la de producción** (comentando las de local) Evitando tener que crear un contenedor docker porque la base de datos ya está creada y activa. **(estará esta configuración en default)**

```
#!Para local
# MYSQLDB_HOST=127.0.0.1
# MYSQLDB_USER=root
# MYSQLDB_ROOT_PASSWORD=urben
# MYSQLDB_DATABASE=peliculasexpress

#!Para producción
MYSQLDB_HOST=pelucas-solidarias.corry4jqcr9s.eu-west-3.rds.amazonaws
MYSQLDB_USER=urben
MYSQLDB_ROOT_PASSWORD=pelucas88
MYSQLDB_DATABASE=pelucasexpress
      You, 7 days ago • Preparo archivos para desplegar …
MYSQLDB_LOCAL_PORT=3306
MYSQLDB_DOCKER_PORT=3306
# NODE_LOCAL_PORT=3000
# NODE_DOCKER_PORT=3000

#Auth Config
AUTH_SECRET=troloo
AUTH_SECRET=7d
AUTH_ROUNDS=10
```

6.6 .env para conectarte a la db en prod

Si optas por crear un contenedor docker

De esta manera te conectarás al contenedor docker.

Pero **hay que crear el contenedor**. Para crearlo tienes que navegar al directorio raíz de la aplicación y ejecutar un docker-compose.

6.7 Ejecutar el docker-compose

De esta manera gracias al archivo de configuración docker-compose.yaml, lee las variables de entorno y crea un contenedor mysql.

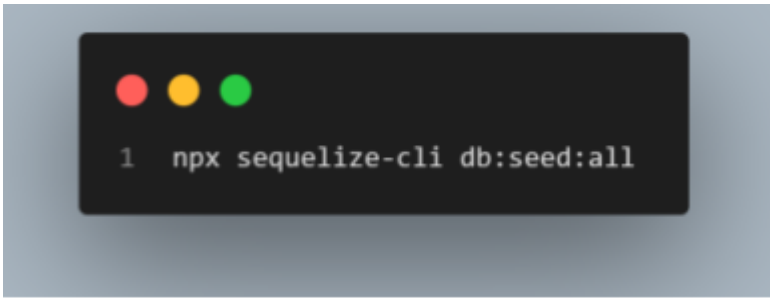
Este usa un volumen llamado **schemas,** que por defecto trae el repositorio y donde ya esta creada la base de datos con información en ellas (de las últimas pruebas que hize en local).

Si no existe esa carpeta, docker-compose se encarga de crearla pero dentro del mysql no habrá ninguna base de datos y tendremos que crearla gracias a las migraciones de sequelize.

Por lo tanto **hacemos estos pasos solo si no existe la carpeta schemas o deseas borrarla** para vaciar los datos. Primero usas el comando **npx sequelize-cli db:create** (esto crea la base de datos)

6.8 Crear las tablas a partir de las migraciones

1- Gracias a las **migraciones** creadas se puede crear la base de datos entera con este comando del cli de sequelize.



6.9 Añadir los seeds

2- Tambien tengo varios **seeds** para añadir información default.
(si usas los seed el usuario admin por default será Email: admin@gmail.com y contraseña: 12345)

Con esto tendrías la base de datos creada y con datos default.

El paso final que debes hacer **tanto si has creado el contenedor docker o como si te has conectado a la base de datos** ya creada es arrancar el servidor.

Para ello dentro del package.json ya existen varios comandos creados que puedes ver.

Pero para arrancar el servidor en desarrollo debes se usar este comando, y por defecto se arrancará por el puerto 3000 OJO! debes tener los puertos libres

6.10 Arrancar el servidor Backend

6.3.2 Frontend

Voy a **explicar las herramientas que he utilizado y que se necesita para su uso** (herramientas del package.json y otras)

Y luego que **pasos que debes seguir para instalar y arrancar la aplicación en local**.

Requisitos minimos

- Node Js
  - Debes tener instalado Node.js instalado en tu ordenador para poder usar el interprete de comandos npm
  - Versión utilizada en el desarrollo **v16.14.2**
- Docker (opcional)
  - Puedes usar docker para compilar y desplegar la aplicación en un servidor nginx en local.
  - En windows puedes descargar Docker Desktop

- CLI de Angular

- Debes tener el cli de angular para arrancar el servidor de pruebas que tiene.
- Para ello lo puedes instalar de manera global con **npm install -g @angular/cli.**

Estas son las herramientas que he descargado desde npm en el desarrollo.

1. youtube-player
   a. Me sirve para mostrar un video de youtube
2. animate.css
   a. Para añadir animaciones
3. bootstrap
   a. Descargar el framework de bootstrap
4. jquery
   a. Descargar Jquery
5. primeicons
   a. Tener los iconos de prime ng
6. primeng
   a. Tener los componentes de prime ng
7. quill
   a. Sirve para poder construir el editor de texto.
8. Typescript
   a. Para usar el lenguaje Typescript

Pasos para el despliegue del frontend en local

Tranquilo, los pasos para el frontend son mas sencillos.

He **hecho uso de docker pero se puede usar o no ya que Angular ya tiene su propio servidor de pruebas**. Docker he usado en desarrollo para ver si funcionaba bien la aplicación compilada.

1- Lo primero que tienes que hacer es clonar el repositorio de GitHub. Navegas en consola a la carpeta donde quieres que se cree la carpeta del proyecto y usas

6.11 Comando para clonar repositorio del frontend

Comando: **git clone https://github.com/urben88/cliente-pelucas-angular**

2- Teniendo Node Js instalado en el ordenador debes de instalar los node_modules. Para ello, dentro de la carpeta del proyecto usas:

6.3 Instalar los módulos

Comando: **npm install**

3- El último paso es arrancar el servidor de pruebas de angular.

Para ello usaremos el **comando del cli de angular ng serve**

Aqui tienes dos opciones:

1-Arrancas el servidor haciendo peticiones a **al backend en producción**

6.12 Arrancar servidor de desarrollo Angular conectado al backend de producción

2-Arrancas el servidor haciendo peticiones **al backend en local**

6.13 Arrancar servidor de desarrollo Angular conectado al backend de producción

6.4 Sistemas operativos empleados.

He trabajado desde un sistema operativo **Windows 10,** pero para el uso de docker he creado contenedores con sistemas operativos **linux con la distribución alpine**.

6.5 Despliegue de la aplicación.

Para el despliegue de Pelucas Solidarias he registrado un dominio gratuito llamado **pelucassolidarias.tk** y la he desplegado en **Amazon Web Services** que es una colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com.

6.5.1 Despliegue en AWS

He usado este servicio (Amazon web services) porque ofrece una gran cantidad de servicios y además te deja crearte una cuenta de pruebas con la capacidad se usar varios de sus servicios de forma gratuita durante 12 meses.

Me he creado una cuenta y **he indicado que los servicios que quiero usar los quiero crear en un servidor de paris**.

Luego he usado sus servicios **EC2** y **RDS**. El primero es para crear máquinas virtuales y el segundo para crear bases de datos.

(Podría explicar mucho mas de como lo he configurado pero no me quiero extender)

Dentro de la **instancia de máquina virtual Ubuntu** que he creado. Me he conectado a ella **por ssh y usado una clave pem**.

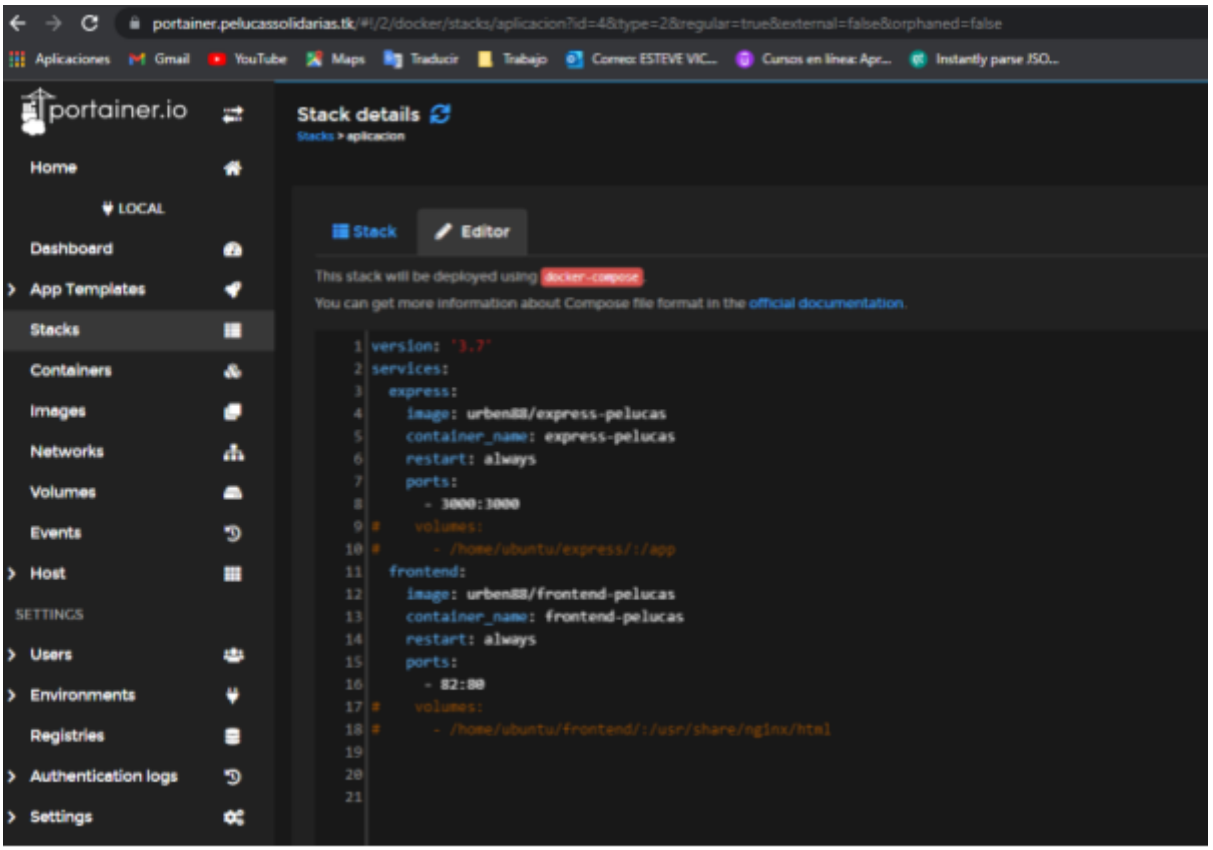
Dentro he instalado **Docker** y creado un contenedor de **portainer** para facilitar el uso de docker en el navegador.

Portainer es una interfaz grafica para docker.

Luego **he creado imágenes docker de mis aplicaciones** tanto del frontend como del backend y las he publicado en **docker hub.**

> Los repositorio de DockerHub los puedes encontrar en el punto 6.1

Y las he importado al portainer donde he creado un stack que es un **docker compose** y lo he desplegado.

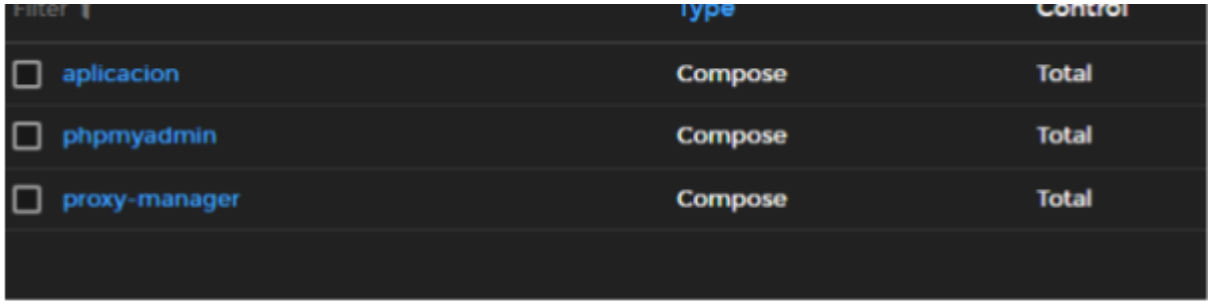


6.14 Aplicación portainer dentro de la instancia de máquina virtual AWS, docker-compose de la aplicación

Con esto tengo los dos contenedores en marcha funcionando por sus puertos.

Cabe destacar que **he tenido que abrir los puertos en AWS** para que funcionase portainer y las demas aplicaciones.

Además he creado dos docker-compose más para instalar las aplicaciones phpmyadmin y proxy-manager



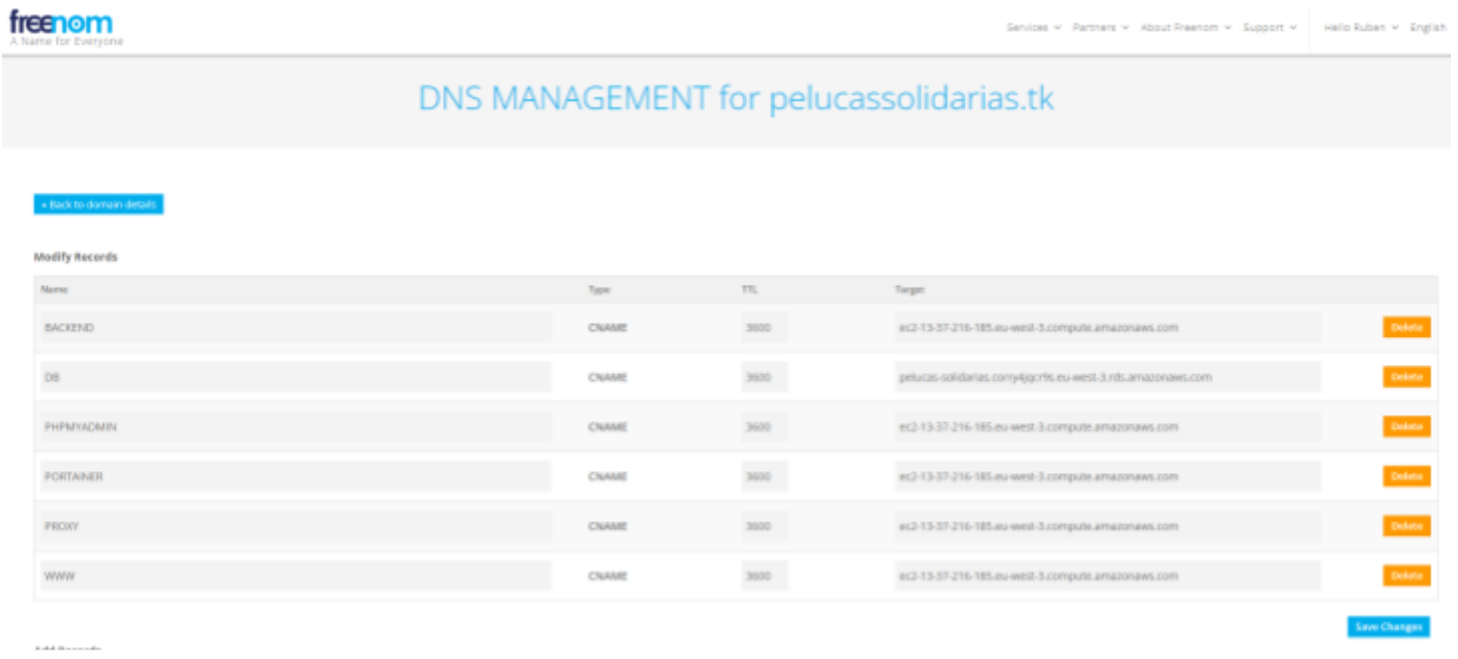
6.15 Docker-compose de las aplicaciones usadas en la máquina virtual

Phpmyadmin lo he instalado en la máquina virtual para tener un acceso remoto a la base de datos y poder modificar cualquier dato puntual.

Y **proxy-manager** es un proxy inverso que se encarga de dirigir el tráfico y que me ha servido para añadir certificados letsencrypt a todos los contenedores. En profundidad mas adelante.

6.5.2 Dominio pelucassolidarias.tk

Para ello me creado una cuenta en FreeNom y he registrado un dominio gratuito .tk llamado pelucassolidarias.tk.



6.16 Registros de pelucassolidarias.tk en freenom

FreeNom ofrece un servidor DNS donde puedo crear los registros CNAME de mi dominio para apuntar a mis distintos servidores y aplicaciones.

De esta manera puedo entrar a las aplicaciones usando mi domino y no el que me da AWS.

Estas serian las aplicaciones:

FrontEnd:
www.pelucassolidarias.tk

Backend:
backend.pelucassolidarias.tk

Base de datos
db.pelucassolidarias.tk

Portainer
portainer.pelucassolidarias.tk

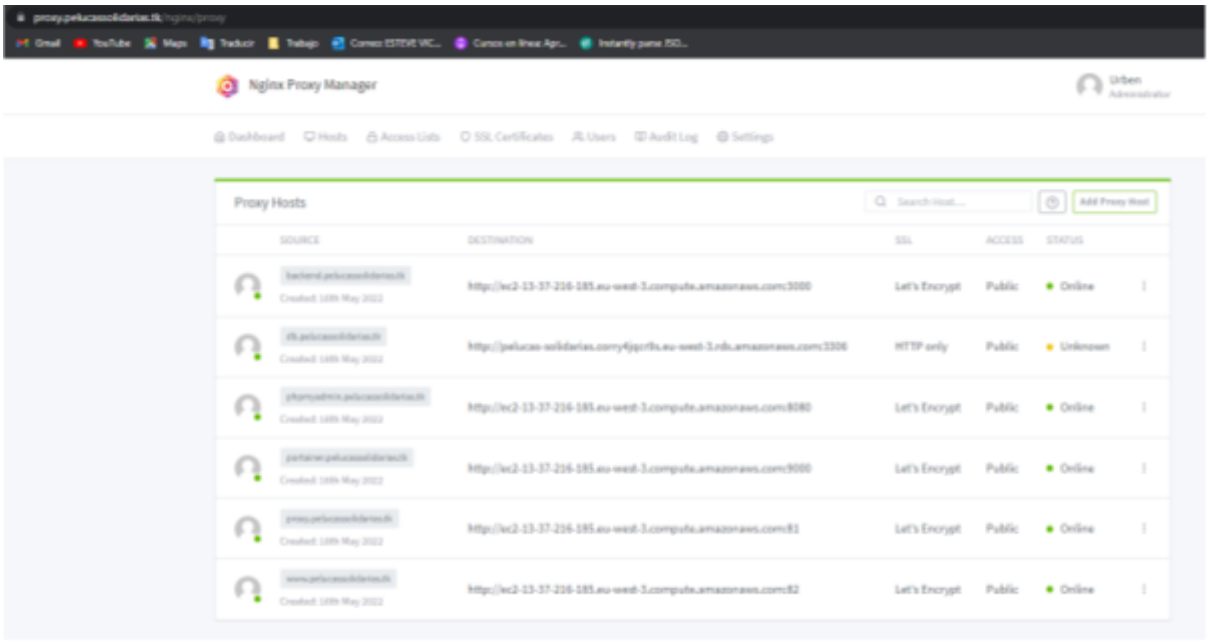
Phpmyadmin
phpmyadmin.pelucassolidarias.tk

ProxyManager
proxy.pelucassolidarias.tk

6.5.2 Como he añadido https

Todas la aplicaciones que he desplegado usan https. Le he añadido el certificado https para evitar problemas a la hora de la exposición en el instituto. Ya que el centro no permite las páginas web por http.

Aquí es donde entra en uso del proxy manager.



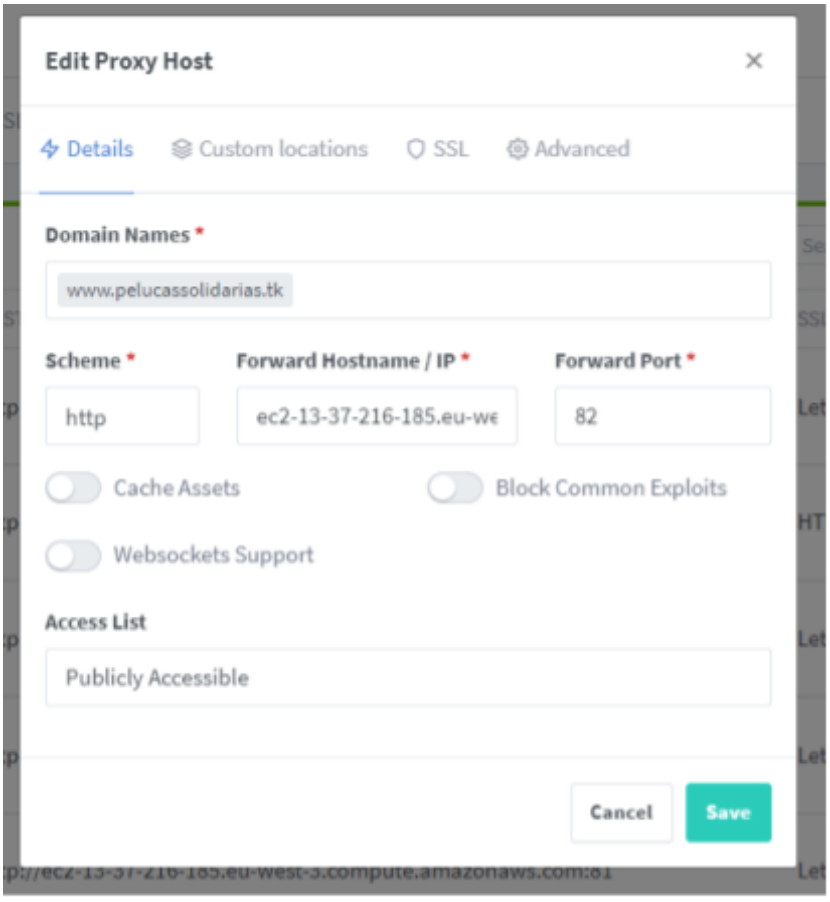
6.17 Menú de proxy hosts de proxy manager

Aqui se encuentran los proxy hosts.

Para configurar un proxy host debo **añadir el subdominio** al cual quiero hacer refencia.

Y luego pongo **la ip o el dominio con el puerto de la información que quiero mostrar** cuando entre a ese subdominio.

Finalmente puedo generar automáticamente un certificado ssl a este host.



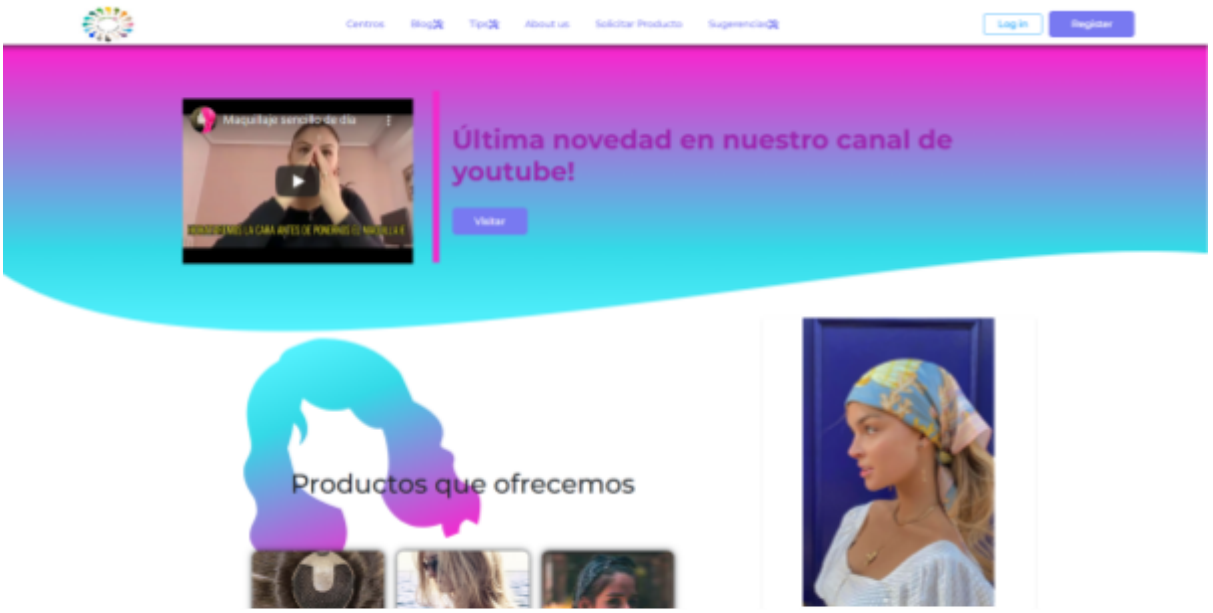
6.18 Ejemplo cómo se configura un proxy host

7 Manual de usuario

Los tres roles existente es la página son **colaborador**, **dotante**, y **administrador.**

La diferencia de navegación por la página entre estos roles es que el rol de **administrador** dispone de un menú de administrador el cual solo puede acceder el. Los **colaboradores** y **donantes** no pueden acceder a este men

Puedes visitar la web sin estar registrado, pero no tendrás notificaciones ni la capacidad de realizar una solicitud.



7.1 Página home de Pelucas Solidarias

La página que se cargará nada mas entrar en www.pelucassolidarias.tk es la página Home con información y novedades.

7.2 Nav de Pelucas Solidarias sin estar logueado

Desde el nav de navegación **sin estar logueado** puedes navegar a:

- ★ **Centros**
  - Puedes visitar los centros afiliados e informarte.
- ★ **About us**
  - Una pequeña introducción del proyecto.
- ★ **Solicitar producto**
  - Aparece el formulario para solicitar un producto (solo si estas logueado, si no es así te redirige al login)
- ★ **Botones de login y register**
  - Te envia a los formularios de login y registro.

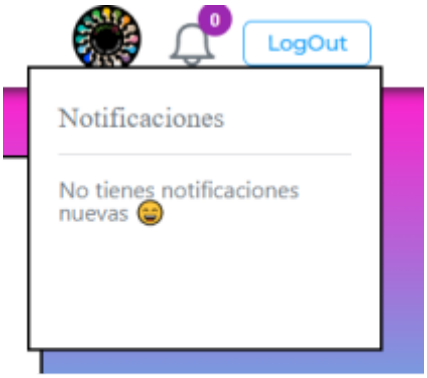
7.3 Nav de Pelucas Solidarias estando logueado

Este es el nav de un **usuario logueado sin el rol de admin**.
Los enlaces que se añaden son:

- ★ **Notificaciones**
  - Vas a un panel para ver tus notificaciones
- ★ **Logo de cuenta**
  - Si le das un click al logo puedes acceder a los formularios para editar datos de la cuenta.
- ★ **Campana**
  - Si pones el ratón en la campana se despliega una ventana con notificaciones no leídas
- ★ **Logout**
  - Te deslogueas

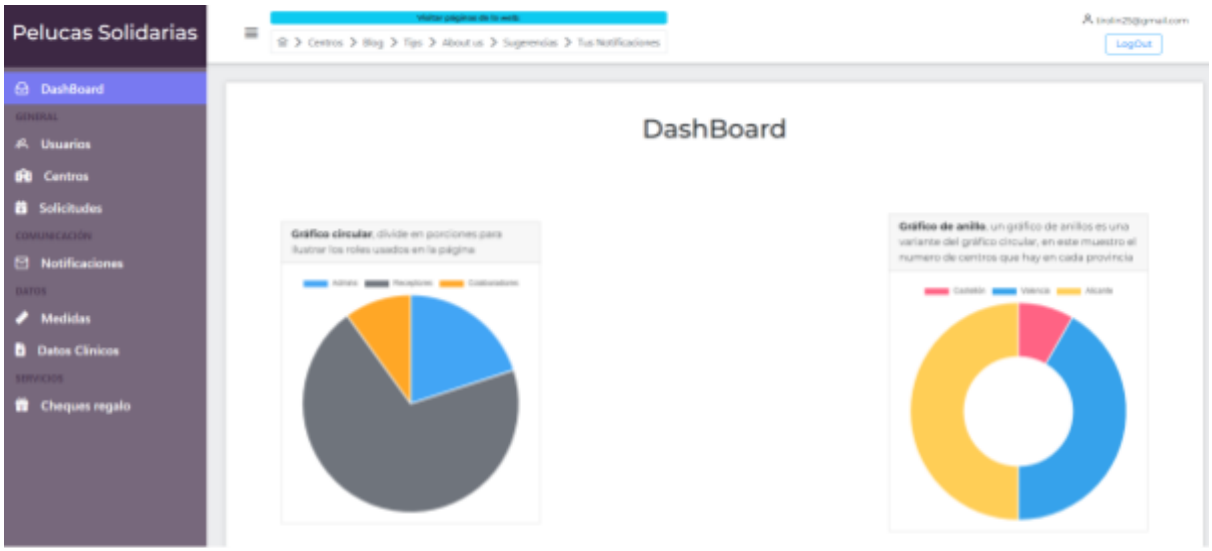
7.5 Solicitud en espera Pelucas Solidarias

7.4 Formulario solicitud Pelucas solidarias



7.6 Notificaciones sin leer Pelucas Solidarias

Si eres **administrador** dispones de un menu para administrar los datos.



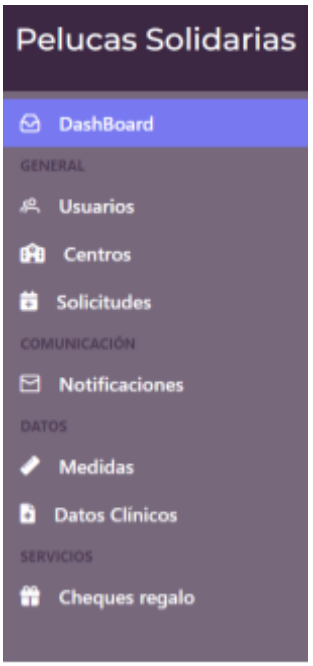
7.7 Menú administrativo Pelucas Solidarias

A simple vista de pueden ver dos menus, uno a la izquierda y otro arriba.

En el de la izquierda puedes navegar a otras páginas con tablas para modificar datos.

En el de arriba tienes un pequeño menu para navegar en la página principal. (Luego tienes un pequeño botón en el nav para volver al menú administrativo)

- ★ DashBoard
  - En esta página se encuentran gráficos de la página web.
- ★ Usuarios
  - Una tabla para buscar y modificar a los usuarios
- ★ Centros
  - Una tabla para buscar y modificar a los centros
- ★ Solicitudes
  - Puedes seleccionar las solicitudes existentes o seleccionar a un usuario para ver o crear una solicitud a un usuario.
  - Existen dos tablas, una con las solicitudes para poder aceptarlas o eliminarlas y otra para buscar a los usuarios.
- ★ Notificaciones



7.8 Menú admin

- Puedes enviar y modificar mensajes a los usuarios.

★ **Medidas y Datos clínicos.**

- Puedes buscar las medidas y datos clínicos de los usuarios para crearlos o modificarlos.

★ **Cheques regalos**

- Puedes crear y modificar los queques regalo y sus descripciones .

8 Conclusiones

8.1 Grado de consecución de objetivos

Se pedían bastantes objetivos pero no se adaptaban al tiempo y he realizado las funciones básicas que se pedían para poder administrar las solicitudes.

8.2 Problemas encontrados

Los grandes problemas encontrados han sido aprender todas las tecnologías ya que el 80% de las herramientas y frameworks que he usado no son las que he aprendido a lo largo del curso.

8.3 Mejora

El proyecto esta pensado a futuro pero me habria gustado haber creado una mejor comunicación entre el administrador y el receptor.

Había creado las tablas de post en el backend para hacer un blog pero no me ha dado tiempo desarrollar esa función.

Y añadir alguna funcionalidad al rol de donante.

9 Anexos y documentos complementarios

El **dominio** para entrar a la página web es www.pelucassolidarias.tk
El usuario admin es:
Correo: admin@gmail.com
passwrd: 12345

Los repositorios GitHub y Docker Hub:

Frontend

GitHub
https://github.com/urben88/cliente-pelucas-angular
Rama principal: master

Docker Hub
https://hub.docker.com/repository/docker/urben88/frontend-pelucas

Backend

GitHub
https://github.com/urben88/api-pelucas-express
Rama principal: main

Docker Hub
https://hub.docker.com/repository/docker/urben88/express-pelucas

El **diseño creado en figma**:
https://www.figma.com/community/file/1110567915060647229

10 Preferencias/Bibliografía

Iconos

Sacados desde flaticon https://www.flaticon.es/

Imágenes

2.1 https://fglawson.com/insights/motivacion-en-la-empresa/

3.1 https://www.youtube.com/watch?v=HhC75IonpOU&ab_channel=CristianHenao
3.2 https://www.figma.com/
3.3 https://www.notion.so
3.4 https://asana.com/

4.1 https://htmlcolorcodes.com/es/recursos/mejor-paleta-de-colores-generadores/
4.2 https://htmlcolorcodes.com/es/recursos/mejor-paleta-de-colores-generadores/
4.3 www.figma.com
4.4 www.figma.com
4.5 www.figma.com
4.6 www.figma.com

5.1 Captura de carpeta de visual studio
5.2 Captura de carpeta de visual studio
5.3 Diagrama hecho con dia http://dia-installer.de/index.html.es
5.4 Creado en workbench https://www.mysql.com/products/workbench/
5.5 Captura de carpeta de visual studio
5.6 Captura de carpeta de visual studio
5.7 Captura de carpeta de visual studio
5.8 https://es.wikipedia.org/wiki/Angular_(framework)

5.9 Captura de carpeta de visual studio
5.10 Captura de carpeta de visual studio
5.11 Captura de carpeta de visual studio
5.12 Captura de carpeta de visual studio
5.13 Captura de carpeta de visual studio
5.14 Captura de carpeta de visual studio
5.15 https://www.primefaces.org/primeng/
5.16 www.pelucassolidarias.tk
5.17 https://es.wikipedia.org/wiki/Bootstrap_(framework)
5.18 https://programarfacil.com/blog/introduccion-a-jquery-ii/
5.19 www.pelucassolidarias.tk
5.20 www.pelucassolidarias.tk

6.1 Del programa postman https://www.postman.com/
6.2 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.3 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.4 Captura de carpeta de visual studio
6.5 Captura de carpeta de visual studio
6.6 Captura de carpeta de visual studio
6.7 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.8 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.9 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.10 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.11 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.12 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.13 https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap
6.14 https://www.portainer.io/
6.15 https://www.portainer.io/
6.16 https://www.freenom.com/es/index.html?lang=es
6.17 https://nginxproxymanager.com/
6.18 https://nginxproxymanager.com/

Bibliografía

Estos son los enlaces de donde he sacado información para realizar la documentación

https://nhc.es/noticias/protesis-capilar-todo-lo-que-debes-saber/#:~:text=La%20pr%C3%B3tesis%20capilar%20es%20una,que%20parezca%20tu%20propio%20cabello.

https://conviveconelcancer.com/como-elegir-un-turbante-o-panuelo-oncologico/
https://proyectosagiles.org/que-es-scrum/

https://www.uifrommars.com/figma-primeros-pasos/

https://www.lucidchart.com/pages/es/que-es-un-wireframe-para-un-sitio-web

https://profile.es/blog/que-es-typescript-vs-javascript/

https://openwebinars.net/blog/que-es-nodejs/

https://www.ceac.es/blog/como-usar-workbench-de-mysql

https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

https://sequelize.org/docs/v6/core-concepts/model-querying-finders/

https://es.wikipedia.org/wiki/JSON_Web_Token

https://developer.mozilla.org/es/docs/Web/HTTP/Status

https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

https://www.acontracorrientech.com/routing-angular-guia-completa-parte-1/

https://rockcontent.com/es/blog/lazy-loading/#:~:text=El%20Lazy%20Loading%20es%20la,comportamiento%20de%20visita%20del%20usuario.

https://www.primefaces.org/primeng/

https://pablomagaz.com/blog/rxjs-subjects-que-son-como-funcionan

https://www.docker.com/

Pelucas Solidarias

Pelucas solidarias es un proyecto que **nace para dar respuesta a las necesidades de personas que están sufriendo alguna enfermedad y no disponen de recursos económicos** para costearse una peluca o pañuelo oncológico.

Esta realizado en colaboración con el alumnado de 2° curso del GS de Estilismo y Dirección de Peluquería del centro IES la Vereda.

El proyecto consiste en solicitar un producto que pueda ser recogido en alguno de nuestros centros afiliados, y así **satisfacer la necesidad de la persona que lo necesite**.

En este documento podrás encontrar toda la documentación técnica de cómo el proyecto Pelucas Solidarias ha sido desarrollado.

Desde que metodología se ha usado para el desarrollo hasta con que herramientas de software se ha construido.