IES POLITÉCNICO
HERMENEGILDO LANZ
G R A N A D A

Documentación proyecto final:

Arturo García García

Índice:

1. Análisis del problema:

1.1. Introducción:

Se nos presenta el caso de un trabajador del sector de la albañilería el cual quisiera tener un mayor control sobre los presupuestos y demás documentos que se generan a raíz de su trabajo y de las citas que tiene con los clientes, además de tener más contacto con los clientes que recurran a su servicio.
Con esto en mente, se hará una aplicación web con la cual pueda tener control de los proyectos que tiene activos cuando un cliente ingrese en la página para que también ellos puedan tener conocimiento del estado de su reforma, también que se pueda mandar presupuestos a los clientes y un método de control de citas para concretar detalles de la reforma.

1.2. Objetivos:

El interesado debe poder mantener un control sobre los proyectos, presupuestos y citas que se generen a partir de la efectuación de su trabajo y poder mantener al tanto a los clientes del estado de su reforma.

1.3. Qué se pretende hacer:

- Creación de proyectos por parte de los clientes.
- Control de citas: de su fecha, lugar y estado por parte de ambas partes.
- Creación de presupuestos y presentación a los clientes por parte del interesado.
- Control de los presupuestos creados por parte de ambas partes.
- Creación y control de actualizaciones a lo largo del proyecto para guardar un control del transcurso del mismo.
- Control de los usuarios que entran en la página.

1.4. Planteamiento y evaluación de diversas soluciones:

En el caso de la creación de proyectos se podría hacer que al momento en el que entre un cliente a la página se le presente un formulario sobre el tipo de reforma qué quiere realizar y más datos relevantes del proyecto o plantearles el formulario antes del registro, hacerlo de esta forma puede ser más invasiva debido a que se le tiene que hacer un cuestionario más extenso directamente, de la primera manera se dilata más en el tiempo el cuestionario dejando una sensación más tranquila, pero da pie a que se creen usuarios en la página que luego no continúen con el cuestionario sobre el proyecto.

En el caso de las citas sería viable crear un calendario con días marcados o un apartado de tarjetas en cada proyecto donde cada tarjeta sea una cita a realizar, la primera manera podría llegar a ser confusa si hay demasiadas citas un mismo día pero expondría de manera rápida todas las próximas citas, la segunda por contraparte es menos confusa si hay varias el mismo día aunque tomaría más espacio para poder ver las citas venideras.

En el caso de las actualizaciones quizá estaría bien una barra de progreso que muestre el porcentaje de proyecto que se lleva realizado y por consiguiente el restante, o una sería de tarjetas donde se exponga qué se ha hecho y en qué día. La primera forma es mucho más visual para el cliente pero carece de información importante para el mismo, mientras que las tarjetas, aún siendo menos visuales, exponen más información relevante para el cliente.

En el caso de los presupuestos tal vez sería capaz de tomar presupuestos hechos ya y añadidos a la página mediante distinto tipos de ficheros o tener incorporado un creador de presupuestos propio, este último ayudaría a tener la información directamente en el sistema y evitaría problemas que se puedan dar al sacar la información de otro tipo de ficheros, pero no dejaría tomar presupuestos ya creados por otros medios, personas o instituciones.

1.5. Justificación de la solución elegida:

Tratando el tema de los proyectos se ha optado por realizar el formulario después de registrarse en la página para no hacer tan invasiva la experiencia del usuario.

Al hablar de las citas, se ha decido crear tarjetas que muestren los datos de la cita en sí ya que da pie también a crear un apartado de citas donde se presenten en forma de tablas con las que están pendientes de confirmación y próximas a su realización. El calendario un punto negativo es el uso obligatorio de dos tipos de tablas distintas (calendario y tabla) lo cual podría confundir aún más cuando se añada la sección de citas.

En el caso de las actualizaciones, se ha elegido crear también tarjetas aunque sea reiterativo respecto a las citas pero así la estética de la página mantiene una coherencia y un orden, además de proporcionar al usuario información más relevante.

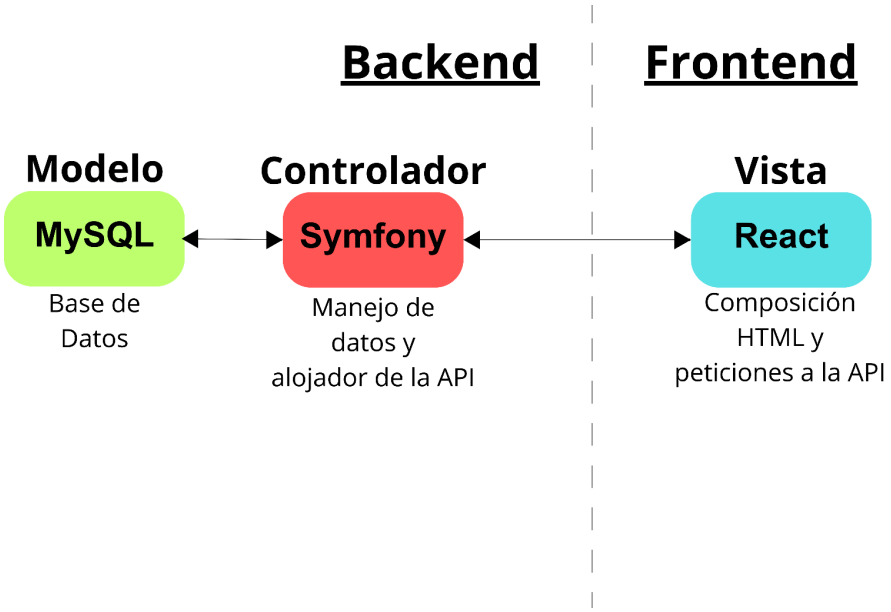
Por último, tratando los presupuesto, se ha preferido desarrollar un creador de presupuesto propio que permita al usuario añadir los capítulos que necesite, así como la unidad en la que se referirá ese capítulo, las cantidades de esa unidad y el precio por cada unidad, que también calculará de manera automática el precio total con el IVA correspondiente.

1.6. Modelado de la solución:

- Recursos humanos: será desarrollado en su integridad por el alumno del grado superior de desarrollo de aplicaciones web Arturo García.
- Recursos de hardware: Un servidor para las consultas de la API que se creará, otro para las consultas a la base de datos y otro para alojar el frontend.
- Recursos de software:
    - Herramientas de desarrollo: Docker, VSCode, Git, Postman.
    - Tecnologías usadas: React, Symfony, MySQL.

2. Diseño e implementación del proyecto:

2.1. Arquitectura de la aplicación:

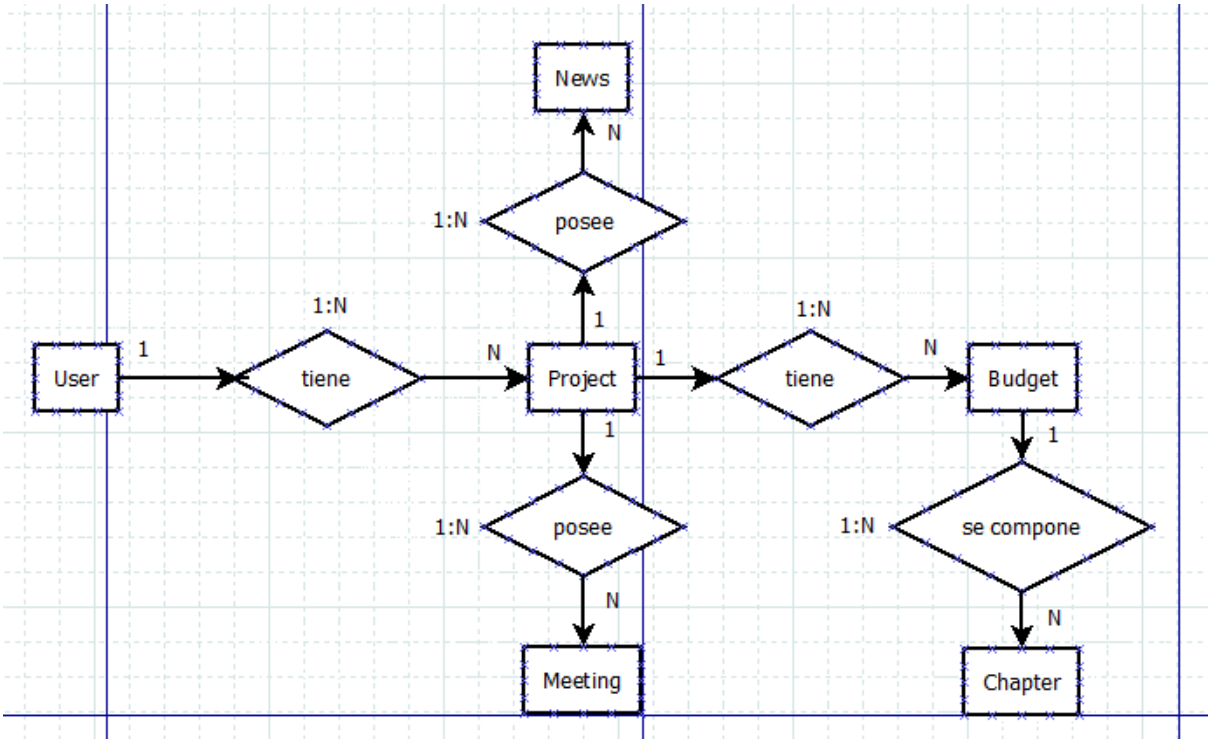


El frontend se creará usando React, como su propio nombre indica, por la reactividad que tiene al cambio de datos y la facilidad de estructuración y modulación del proyecto.

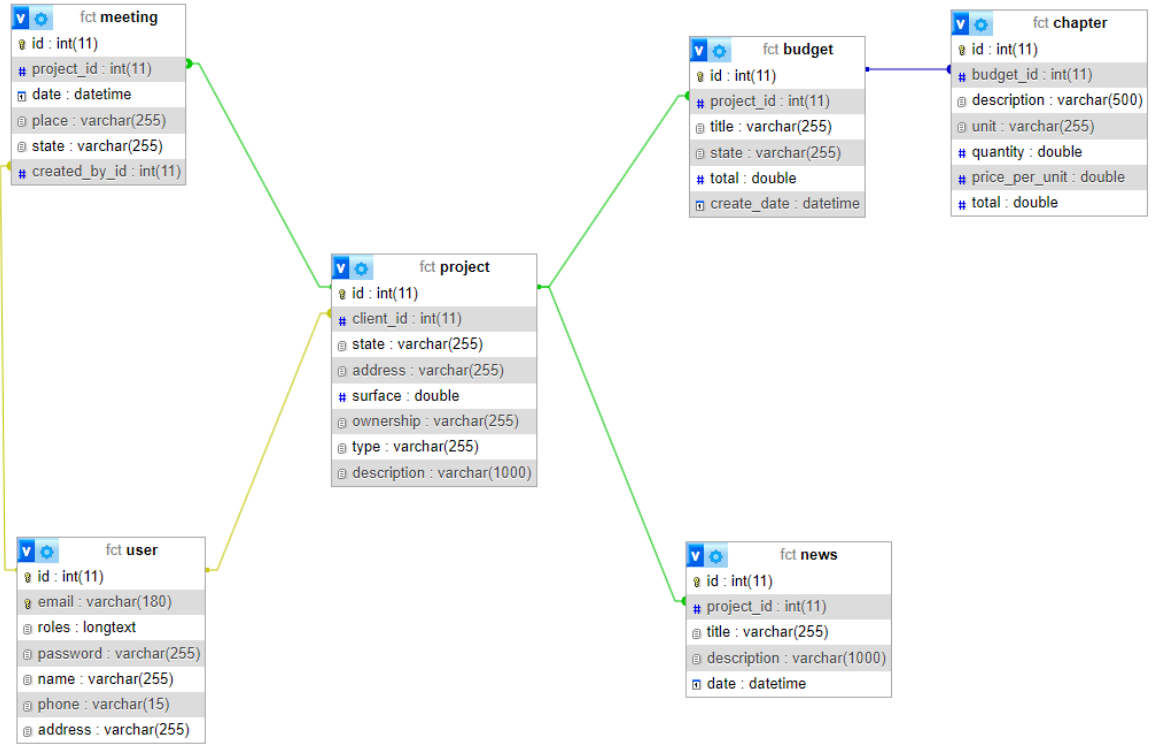
La mayor parte del backend se realizará en Symfony debido a que estructura por defecto los controladores de los objetos y campos que usaremos además de tener la implementación Api Platform para crear una API de manera sencilla y rápida y LexikJWT para la autenticación mediante tokens de los usuarios.

La base de datos se hará en MySQL por la experiencia que se tiene en ella y por ser una base de datos relacional, lo cual no servirá para mantener una cohesión en los datos que se vayan a guardar en ella. Contará con: React Router Dom para el mapa de navegación que se describirá más adelante en este documento, Sweet Alert 2 para avisar a los usuarios de las acciones que van a realizar y pequeños formularios y MUI Icons para los iconos de la aplicación.

El Diagrama Entidad Relación del modelo que se usará será el siguiente:



Y los datos que se guardará en cada entidad serán los siguientes:



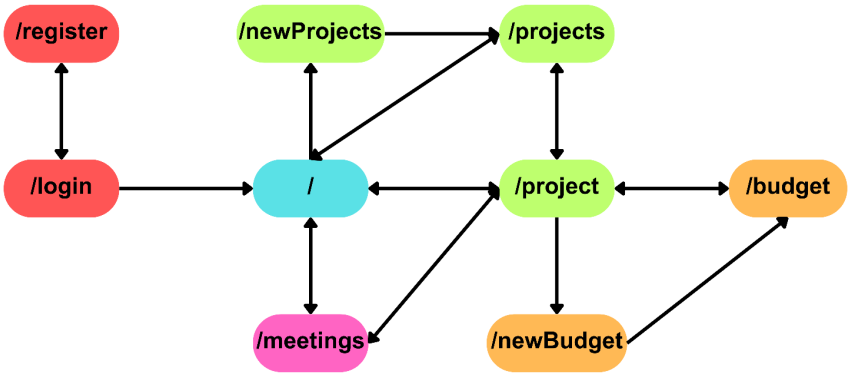
Aquí podemos ver los campos que relacionan qué entidad y con cuál.

2.2. Tipos de usuario:

A continuación se describirán los tipos de usuarios y las acciones que pueden ejercer.

- Cliente: registrarse en la página, crear un proyecto y acceder a él, crear citas para su proyecto, aceptar o rechazar las citas que le pida el trabajador y aceptar o rechazar presupuestos enviados por el trabajador.
- Trabajador: registrarse en la página, modificar el estado de los proyectos, acceder a los proyectos de los clientes, crear citas, aceptar o rechazar las citas enviadas por los clientes, crear actualizaciones sobre el proyecto, crear y enviar presupuestos.
- Administrador: lo mismo que el trabajador, manejar los datos de los usuarios de la página.

2.3. Mapa de navegación:



2.4. Funciones:

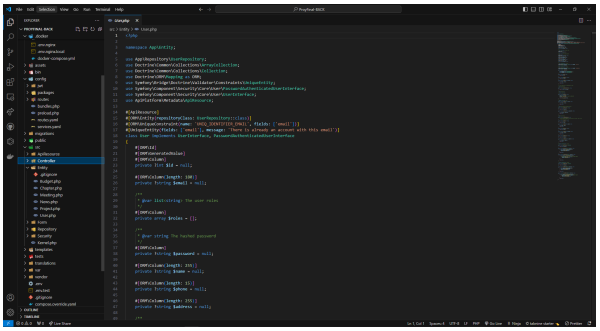
- Backend: esta sección se divide en: Controller, donde se encuentran las funciones que manejan la creación, exposición de datos, modificación de datos y eliminación de las entidades de la base de datos; Entity, donde se define la estructura de los datos de las entidades; Form, aunque no utilizado casi en su totalidad, define los formularios para la creación y modificación de datos; Repository, donde se definen queries de SQL para comunicarse con la base de datos y manejarla.

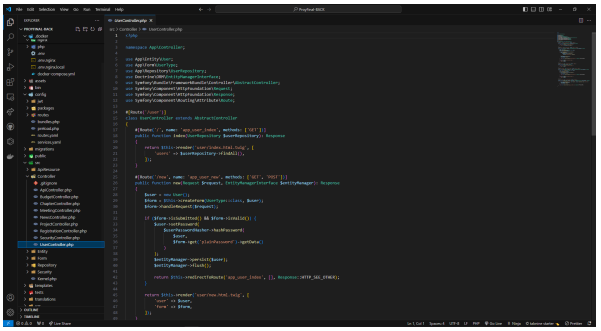
En el controlador podemos encontrar funciones como el método 'new()'

```
    #[Route('/new', name: 'app_project_new', methods: ['GET', 'POST'])]
    public function new(Request $request, EntityManagerInterface
$entityManager): Response
    {
        $project = new Project();
        $form = $this->createForm(ProjectType::class, $project);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager->persist($project);
            $entityManager->flush();

            return $this->redirectToRoute('app_project_index', [],
Response::HTTP_SEE_OTHER);
        }

        return $this->render('project/new.html.twig', [
            'project' => $project,
            'form' => $form,
        ]);
    }
```

El cual define la creación y la persistencia de los datos de la entidad Project en este ejemplo.

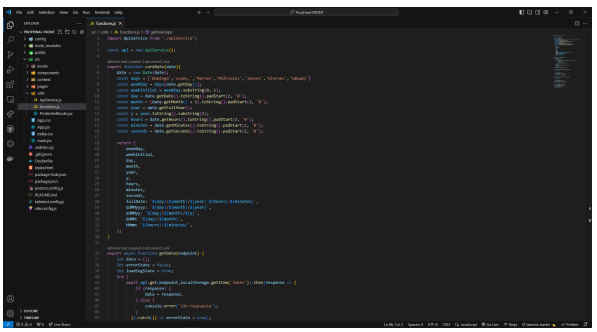
Entidad usuario y su controlador:

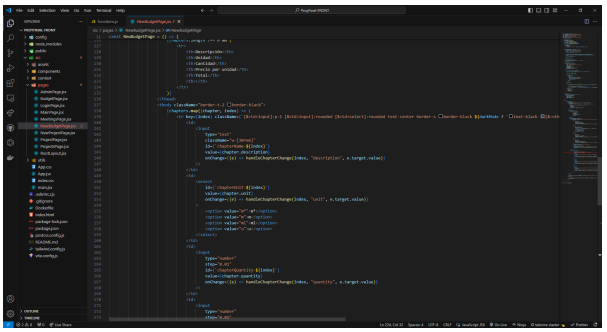




- Frontend: esto se divide en:
  - Pages: donde se estructuran las páginas como tal de la aplicación.
  - Components: donde se definen componentes que usan las páginas y así modularizar la aplicación.
  - Context: genera un contexto general para mantener datos entre las distintas páginas de la aplicación como los datos del usuario o la elección sobre el modo oscuro.
  - Utils: aquí se definen ficheros con funciones que serán útiles por toda la aplicación tales como: ApiService, el cual gestiona las peticiones que se realizarán a la API; ProtectedRoute, donde se validarán ciertos datos en la aplicación para permitir al usuario la libre navegación y functions, donde se definen funciones de utilidad más general como los siguiente métodos:
    - *cureDate()*: para transformar la fecha en una más legible.
    - *allowRoleAbove()*: para validar que el usuario tenga el rol necesario o uno superior.
    - *allowRoleEqual()*: para validar que el usuario tenga exactamente el rol necesario.

Método cureDate y la página de creación de presupuestos:





Las páginas de la sección *pages* cuentan con los métodos:

- *loadData()*: realiza una petición a la API para recibir los datos de la página que se esté mostrando en ese caso.
- *handleChange()*: no todas se llaman igual porque especifican qué hacen en el nombre, modifican los datos de cada entidad mediante peticiones a la API.
- *handleDelete()*: borran los datos mediante la API de las entidades.

Verbigracia de un *loadData()*:

```
    const loadData = async () => {
        try {
            const response = await api.get('projects',
localStorage.getItem('token'));
            if (response['@context'] === '/api/contexts/Error') {
                setError(response);
            } else if (response['hydra:member']) {
                setActiveProjects(response['hydra:member'].filter(p =>
p.state === 'created' || p.state === 'accepted' || p.state === 'wip'));
                setEndedProjects(response['hydra:member'].filter(p =>
p.state === 'abandoned' || p.state === 'finished' || p.state ===
'rejected' || p.state === 'paused'));
            } else {
                setError('Unexpected response format');
            }
        } catch (error) {
            setError(error.message);
        } finally {
            setLoading(false);
        }
    };
```

3. Fase de pruebas:

3.1. Pruebas durante el transcurso del desarrollo:

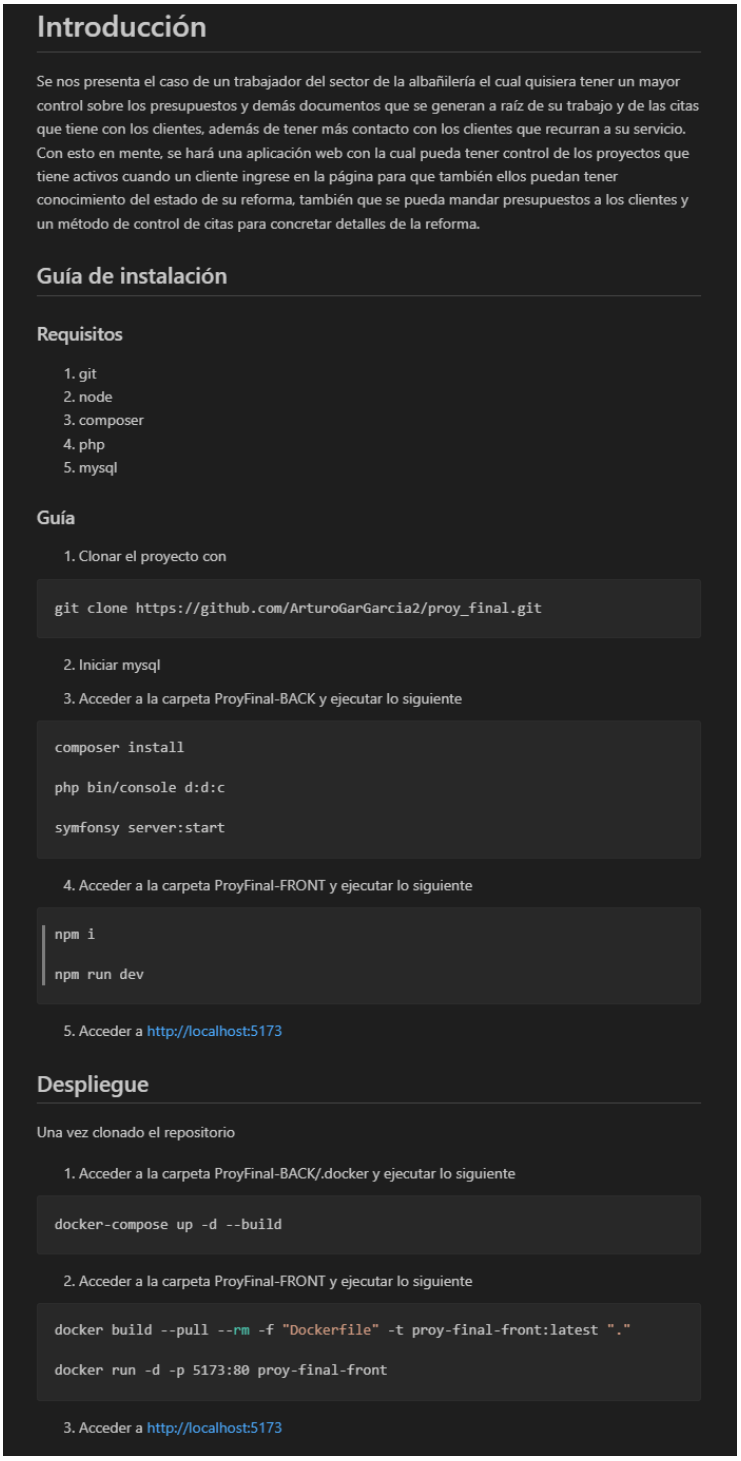
Mientras se estaba desarrollando la aplicación, a cada nueva funcionalidad se le atacó de todas las maneras posibles, más o menos rebuscadas, para ver dónde podía fallar y fallaba, así cuando se encontraba un error, se solventaba en el momento o se esperaba a que estuviese la implementación completa para así poder arreglarlo con mayor seguridad. Un ejemplo de esto puede ser al crear el sistema de generación de presupuestos, se ha probado el generar muchos capítulos, al poner carácteres raros en la creación, al borrar capítulo antes de generar el presupuestos y por ejemplo, se ha descubierto que al poner punto (.) en los inputs de cantidad por unidad y precio por unidad se borra el contenido ya que el sistema de numeración que usa es el español ya que usa la coma para los decimales en vez del punto, entonces esto no se puede considerar como fallo en sí, ya que está utilizando un sistema diferente pero también útil.

3.2. Pruebas posteriores al desarrollo:

Una vez finalizado el desarrollo de la aplicación, se le pidió a varios usuarios que no conocían nada del desarrollo de la misma para que hicieran el transcurso de un usuario corriente. La primera vez que se hizo se encontraron varios errores tales como una redirección errónea o falta de datos al pasar de una vista a otra, una vez solucionado se le pidió a otro usuario que realizara la misma acción hasta que se logró encontrar un punto en el que la coherencia de la página se mantuviese.

4. Documentación de la aplicación:

4.1. Introducción a la aplicación y manual de instalación.



Introducción

Se nos presenta el caso de un trabajador del sector de la albañilería el cual quisiera tener un mayor control sobre los presupuestos y demás documentos que se generan a raíz de su trabajo y de las citas que tiene con los clientes, además de tener más contacto con los clientes que recurran a su servicio. Con esto en mente, se hará una aplicación web con la cual pueda tener control de los proyectos que tiene activos cuando un cliente ingrese en la página para que también ellos puedan tener conocimiento del estado de su reforma, también que se pueda mandar presupuestos a los clientes y un método de control de citas para concretar detalles de la reforma.

Guía de instalación

Requisitos

1. git
2. node
3. composer
4. php
5. mysql

Guía

1. Clonar el proyecto con

```
git clone https://github.com/ArturoGarGarcia2/proy_final.git
```

2. Iniciar mysql
3. Acceder a la carpeta ProyFinal-BACK y ejecutar lo siguiente

```
composer install

php bin/console d:d:c

symfonsy server:start
```

4. Acceder a la carpeta ProyFinal-FRONT y ejecutar lo siguiente

```
npm i

npm run dev
```

5. Acceder a http://localhost:5173

Despliegue

Una vez clonado el repositorio

1. Acceder a la carpeta ProyFinal-BACK/.docker y ejecutar lo siguiente

```
docker-compose up -d --build
```

2. Acceder a la carpeta ProyFinal-FRONT y ejecutar lo siguiente

```
docker build --pull --rm -f "Dockerfile" -t proy-final-front:latest "."

docker run -d -p 5173:80 proy-final-front
```

3. Acceder a http://localhost:5173

4.2. Acciones de usuario:

Acciones de un usuario corriente:

1. Registrarse en la aplicación.
2. Hacer login en la aplicación.
3. Crear su proyecto en la ruta /newProject.
4. Acceder el proyecto y desde ahí puede:
    1. Pedir citas con el trabajador.
    2. Cancelar las citas que haya pedido.
    3. Aceptar o rechazar las citas que le mande el trabajador.
    4. Revisar las actualizaciones que reciba del trabajador.
    5. Aceptar o rechazar presupuestos que reciba del trabajador.

Acciones de un usuario trabajador:

1. Registrarse en la aplicación.
2. Hacer login en la aplicación.
3. Acceder a los proyectos de los clientes.
4. Pedirle citas a los clientes.
5. Cancelar las citas mandadas a los clientes.
6. Aceptar o rechazar las citas recibidas.
7. Crear actualizaciones en los proyectos de los clientes
8. Crear presupuestos en los proyectos de los clientes
9. Crear los presupuestos ocultados al cliente para su posterior entrega.
10. Cancelar los presupuestos enviados al cliente.
11. Eliminar presupuestos.

Acciones de un usuario administrador:

1. Registrarse en la aplicación.
2. Hacer login en la aplicación.
3. Acceder a los proyectos de los clientes.
4. Pedirle citas a los clientes.
5. Cancelar las citas mandadas a los clientes.
6. Aceptar o rechazar las citas recibidas.
7. Crear actualizaciones en los proyectos de los clientes

8. Crear presupuestos en los proyectos de los clientes
9. Crear los presupuestos ocultados al cliente para su posterior entrega.
10. Cancelar los presupuestos enviados al cliente.
11. Eliminar presupuestos.
12. Acceso al panel de administración de usuarios con el que cambiar los roles.

5. Conclusiones finales:

5.1. Objetivos cumplidos.

Se ha conseguido prácticamente todos los objetivos esperados, se ha conseguido tener un control sobre los presupuestos y su creación de manera intuitiva, una manera de comunicación con los clientes mediante las actualizaciones y un sistema de gestión de citas.
Aunque se podría haber añadido también una manera de leer documentos externos para poder generar los presupuestos y así tomarlos creados ya por otros medio, mostrar una cita nueva o una actualización nueva cuando aún no se ha creado ninguna pero no se han podido hacer estas funciones.
En general se han conseguido realizar todos los puntos que se habían planeado, no con todas las funcionalidades que se le podrían haber añadido, pero se puede decir que el 100% de los planeado se ha conseguido.

5.2. Modificaciones o ampliaciones

Cosas que se podrían añadir en un futuro sería un sistema de generación de presupuestos a partir de archivos externos, una barra de progreso del proyecto y también una gestión de usuarios más completa, también implementar una mejor vista en formato móvil de la gestión de usuarios y de creación de presupuestos.

6. Bibliografía:

- https://stackoverflow.com/
- https://github.com/lexik/LexikJWTAuthenticationBundle
- https://api-platform.com/docs/distribution/
- Implementación de autenticación con JWT: https://www.youtube.com/watch?v=XT4oy1d1j-g&t=963s