

08/02/2023

RECUPERACIO RETO 3



Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

BY: UNAI DÍEZ

TABLA DE CONTENIDO

Reto 3.....	2
Organización del proyecto	2
Tareas vistas para realizar el proyecto.....	2
Planificación temporal	2
Especificaciones de requisitos.....	3
Catalogo de requerimiento funcionales	3
Catalogo de requerimiento No-funcionales	4
Rangos de implementaciones	5
Diseño y organización de la página web	5
Prototipado / maquetación básica de la interfaz de las páginas web	5
Metodología del despliegue del entorno	7
Modelo de datos	8
Contenido del entorno tecnológico (como se controla la página web)	10
plantilla de las páginas web	11
CSS	12
control de imágenes/archivos	12
entorno cliente: javascript/typescript y vue.....	12
Sass	13
web.php.....	13
Consideraciones sobre la implementación	14
manual de usuario	14
Consideración del entorno cliente: vue.....	14
Consideración del entorno servidor: laravel 10.....	15
Consideración de envío de datos:	15
Consideración de control de estilos:	16
aplicaciones extras utilizadas en el reto	17
Incidencias surgidas	17
Conclusion	17

RETO 3

En este proyecto, mi aplicación será realizado a través de una petición al ayuntamiento de Vitoria a través de control de obras. El cliente que realice una obra deberá dar sus datos en caso de no tenerlos y especificar los datos principales para la creación de una obra,

Dado a las necesidades del empleado, este proyecto se centrará en aliviar el control de las obras en una aplicación desplegada con varios requisitos necesarios.

El objetivo con este proyecto es que cada usuario pueda ver y realizar todas las tareas que necesite hacer dicho usuario para cualquier administración con las obras, comentarlas y editarlas.

ORGANIZACIÓN DEL PROYECTO

TAREAS VISTAS PARA REALIZAR EL PROYECTO

Dado a que este proyecto es un mini-reto, no se incluye ninguna novedad a aprender en el reto exceptuando algunas cosas que se mencionarán posteriormente.

PLANIFICACIÓN TEMPORAL

Dado a que la planificación ha sido individual, no se ha realizado una planificación muy avanzada, pero el orden de las interpretaciones han sido lo siguiente.

- Control de los modelos de la base de datos
- Realización y control de las vistas
- Despliegue principal
- Control avanzado de base de datos y vistas
- Despliegue final

ESPECIFICACIONES DE REQUISITOS

Todo el contenido que hay que hacer en las tareas a realizar anteriormente mencionado, hay que compaginar con las peticiones que el cliente nos ha mencionado con los objetivos del proyecto.

Pare ello he realizado varios catálogos de requerimientos para concatenar todas las necesidades para que el proyecto funcione con normalidad.

El objetivo de realizar dichos catálogos es clasificar e identificar todas las necesidades, analizarlas y separarlas y catalogar dichas actividades.

CATALOGO DE REQUERIMIENTO FUNCIONALES

En este apartado detallo los requisitos funcionales para la implementación del proyecto.

Los requisitos funcionales, suelen ser funcionalidades especialmente pedidas por el cliente, aunque en algunos apartados puedan salir ciertas necesidades. ----En este caso no nos piden nada expresamente por el cliente, pero en algunos apartados se nos implementa----.

En este apartado detallo los requisitos funcionales para la implementación del proyecto.

nro.	Descripción	Prioridad	Exigencia
Módulo de inicio			
RI01	El navegador permitirá iniciar sesión con un correo y una contraseña	1	E
RI02	El navegador permitirá crear un nuevo usuario	3	D
Módulo de solicitante			
RS03	El usuario podrá crear un nuevo solicitante	1	E
RS04	El usuario podrá ver todos los solicitantes existentes	2	E
Módulo de Obra			
RO05	El usuario podrá ver todas las obras existentes	1	E
RO06	El usuario podrá crear una nueva obra	2	E
RO07	El usuario podrá editar una obra en concreto al dar "ver/editar obra"	1	E
RO08	El usuario podrá ir a ver todos los comentarios de una obra en concreto al dar "ver comentarios"	2	E

Módulo de comentario			
RC09	Al tocar el apartado “ver comentarios” podrá ver todos los comentarios de una propia obra	1	E
RC10	Mientras este dentro del apartado de comentarios podrá insertar un nuevo comentario	2	E

CATALOGO DE REQUERIMIENTO NO-FUNCIONALES

En este apartado detallo los requisitos no-funcionales para la implementación del proyecto.

Los requisitos no funcionales, suelen ser funcionalidades no escritas ni pedidas por el cliente, aunque en algunos apartados puedan salir ciertas necesidades. En este caso no nos piden nada expresamente por el cliente, pero en algunos apartados se nos implementa.

nro.	Descripción	Prioridad	Exigencia
RN01	El software se desarrollará para el uso general fuera del uso local	1	E
RN02	El lenguaje de programación de desarrollo de proyecto del lado cliente será de TS puro	1	E
RN03	El lenguaje de programación de desarrollo de proyecto del lado servidor php a través del framework de Laravel vite	1	E
RN04	El manejador de base de datos se usará el database de laravel	1	E
RN05	Los navegadores comprobados para el manejo del navegador web común actual (IE, Google Chrome, Mozilla Firefox, Opera)	2	D
RN06	El módulo del despliegue de la página será utilizado con la tecnología Docker, y exportándolo a Railway	1	E
RN07	El módulo del despliegue de la página será utilizado con la Desplegador dentro del	1	E
RN08	El control de la interfaz (UX) de la página web estará estructurada a través del sass, con bootstrap	2	E
RN09	Todo el sistema de archivos que se realicen en la página web se implementara dentro del propio GitLab creada para dicha página web.	1	E

RANGOS DE IMPLEMENTACIONES

Para realizar el catálogo de ambas partes, he sistematizado varios apartados para la fluidez correcta del sistema.

Aparte de las especificaciones de los apartados de requisitos, he sistematizado de forma sencilla todos los mínimos de la aplicación:

Cada uno de los apartados del catálogo requisitos he implementado diferentes maneras de enumerar los requisitos para poder sistematizar todo

Apartado	Formato de numero	Contenido variable
funcional	RAXX	A=>atributo de contenido XX=> 01 incrementado en 1
No-funcional	RNXX	XX=> 01 incrementado en 1

Prioridad		Exigencia	
Numero	Descripcion	Letra	Descripcion
1	Alta	E	Exigente
2	Media	D	Deseable
3	Baja	O	Opcional

DISEÑO Y ORGANIZACION DE LA PAGINA WEB

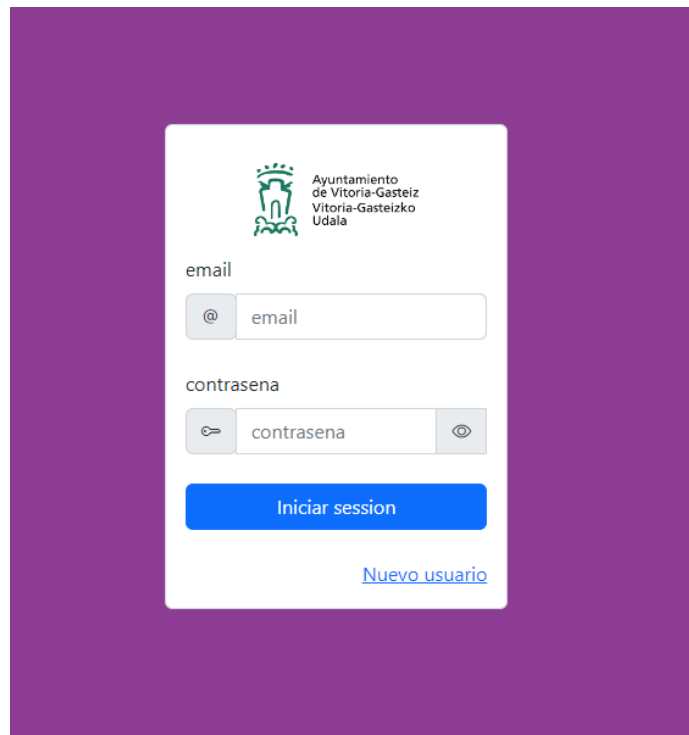
PROTOTIPADO / MAQUETACION BASICA LA INTERFAZ DE LAS PAGINAS WEB

Para este proyecto, dado a que era un minireto para la recuperación del reto 3, la mayoría de la base del prototipo de la aplicación ha sido similar .

Dichas plantillas se vieron para una tener una coherencia con las visualizaciones mínimas necesarias para la correcta fluidez del programa.

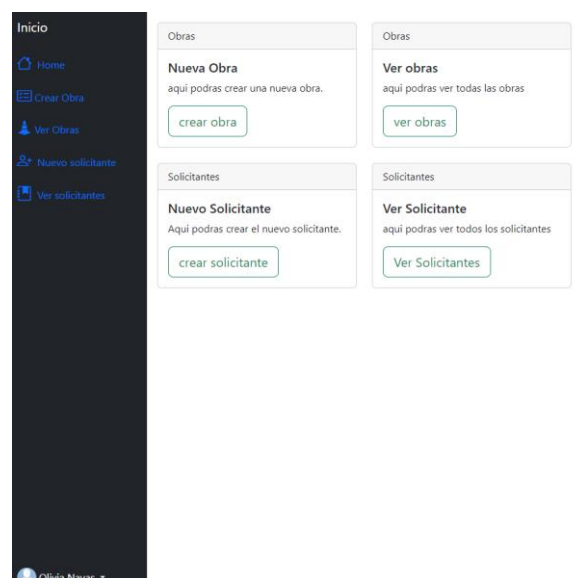
Los bocetos se han realizado conforme al inicio de la aplicación:

Login.php:



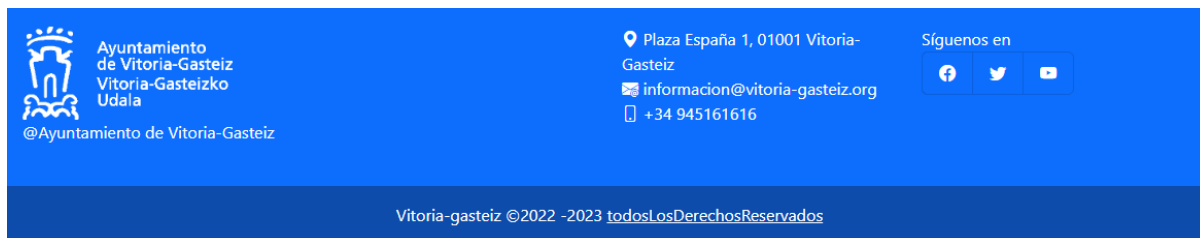
The login form is centered on a purple background. It features the logo of the Ayuntamiento de Vitoria-Gasteiz (Ayuntamiento de Vitoria-Gasteiz Vitoria-Gasteizko Udala) at the top. Below the logo, there are two input fields: one for 'email' with an '@' icon and one for 'contrasena' (password) with a key icon and a toggle eye icon. A blue button labeled 'Iniciar session' is positioned below the password field. At the bottom, there is a link labeled 'Nuevo usuario'.

Contenido principal base:



The main content area is divided into four sections arranged in a 2x2 grid. The top-left section is titled 'Obras' and contains a 'Nueva Obra' button. The top-right section is titled 'Obras' and contains a 'Ver obras' button. The bottom-left section is titled 'Solicitudes' and contains a 'Nuevo Solicitante' button. The bottom-right section is titled 'Solicitudes' and contains a 'Ver Solicitante' button. A sidebar on the left contains links to 'Inicio', 'Home', 'Crear Obra', 'Ver Obras', 'Nuevo solicitante', and 'Ver solicitantes'. The user's name 'Olivia Navas' is displayed at the bottom of the sidebar.

Pie de pagina



METODOLOGÍA DEL DESPLIEGUE DEL ENTORNO

Para realizar todas las peticiones (como mínimas, como algunas implementaciones extras), he tenido que organizar con varias tecnologías diferentes para el posible funcionamiento totalitario del proyecto.

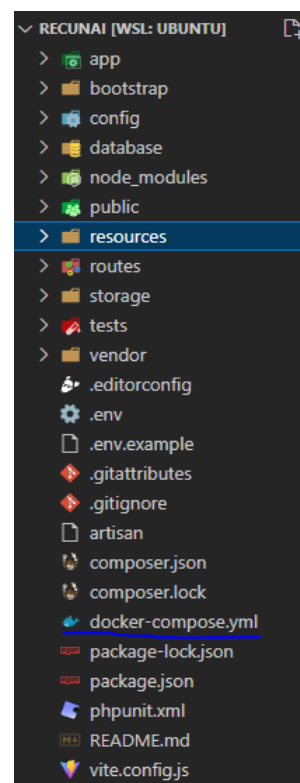
Para ello, he utilizado la tecnología de Docker para poder implementar el sistema de la aplicación.

Docker es un automatizador de despliegue de aplicaciones utilizando contenedores software. Permite crear, probar e implementar contenido a través de una nube.

Para poder tenerlo en uso, como voy a utilizar un framework y guardarlo en mi GitLab, solamente necesitaremos una con varios archivos y carpetas diferentes:

- `docker-compose.yml`: Configuración del contenido de los contenedores del Docker. Mas adelante hablaremos de las carpetas principales.

En el caso de este proyecto he tenido tres contenedores dentro de una carpeta



	recunai	-	Exited			
	laravel.test-1 75946a157df3	sail-8.2/app:l	Exited	5173:5173 80:80		
	mailpit-1 85a164704553	axllent/mailp	Exited	1025:1025 8025:8025		
	mysql-1 f6738ff44d13	mysql/mysql	Exited	3306:3306		
	selenium-1 6f2b0cef1c60	selenium/sta	Exited (143)			
	redis-1 e43ec326e807	redis:alpine	Exited	6379:6379		
	meilisearch-1 df0d9a02694d	getmeili/meil	Exited (143)	7700:7700		

Gracias al Docker, he podido implementar el previo del despliegue, per el objetivo final era ni mas ni menos que la exportación real a través del Railway.

Railway es una plataforma en la cual esta sustituyendo de forma rápida a heroku, dado a que esa plataforma ha decidido en dejar de ser gratuito los despliegues.

Railway te permite desplegar todo tipo de contenido(no solo paginas web, sino también bots de discord o telegram) y solamente con la preocupación de realizar el código, dicha interfaz te permite interactuar con todas las configuraciones de forma adecuada.

MySQL
 Created 2023-02-23

recunai
 recunai.up.railway.app
 41 minutes ago via GitHub

recunai
 Deployments Variables Metrics Settings

recunai.up.railway.app
 an hour ago via GitHub

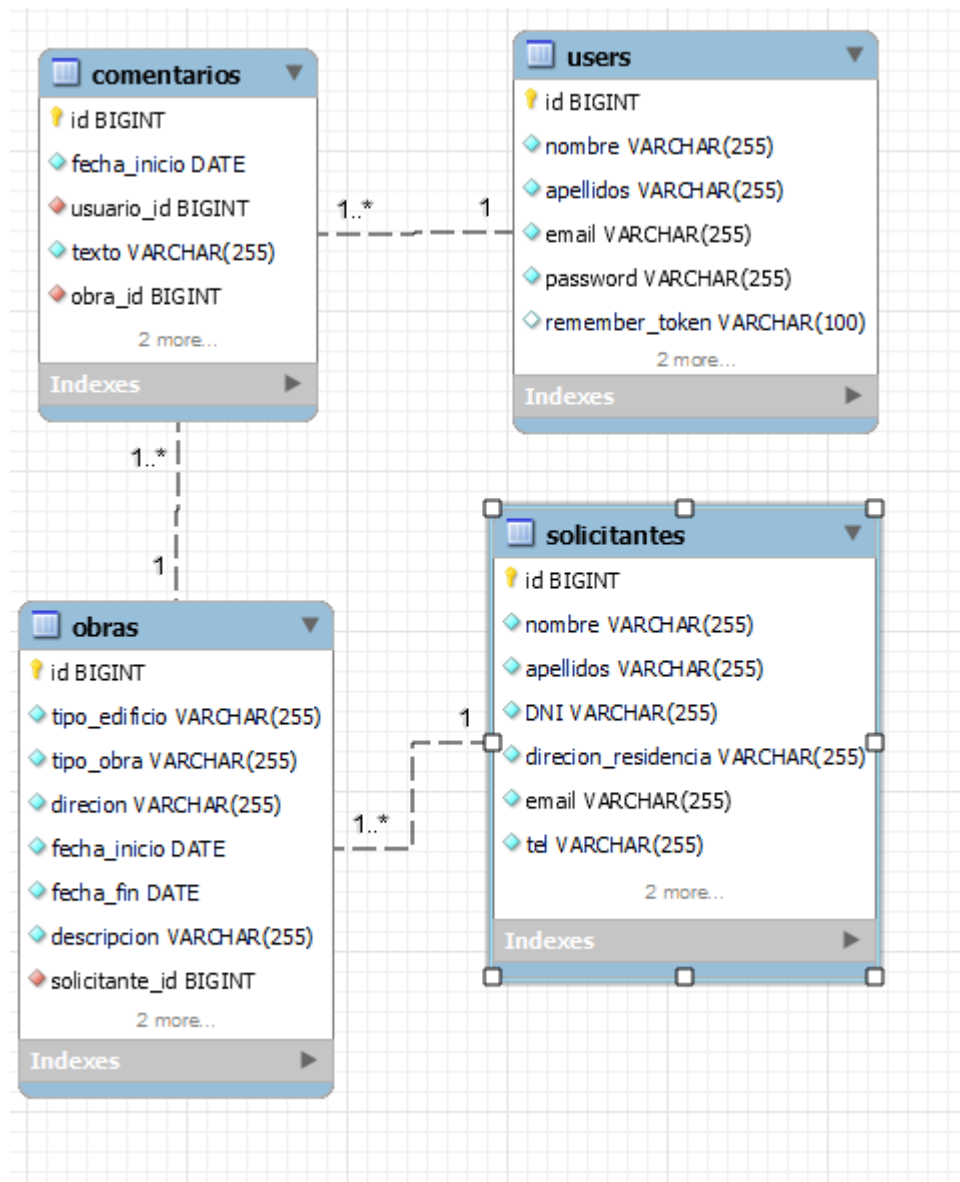
retoques 3
 main

HISTORY

MODELO DE DATOS

Todo el contenido de mi base de datos está organizada de esta manera:

En mi base de datos he implementado las siguientes tablas en mi proyecto:



En esta base de datos no hay mucho que detallar, así que voy a ir de uno en uno:

La tabla **solicitantes** aparecerán todos los datos que obtenemos del cliente, en la cual tiene estos valores:

- Id: identificador del solicitante
- nombre: nombre del solicitante
- apellidos: apellido del solicitante
- DNI: DNI del solicitante
- Dirección_residencia: dirección de la residencia del solicitante
- email: correo del solicitante
- tel: teléfono personal del solicitante

La tabla obras incluirán todas las obras que tiene un solicitante, que por cierto tiene estos atributos:

- Id: identificador de la obra
- Tipo_edificio: tipo de edificio de la obra (piso, casa, local, garaje, trastero o edificio)
- Tipo_obra: tipo de la obra que se realiza (nueva construcción o reforma)
- Dirección: dirección donde se realiza la obra
- Fecha_inicio: fecha de inicio de la obra
- Fecha_fin: fecha del final de la obra
- descripción: descripción de la obra

Esta tabla está relacionada con solicitantes dado a que una obra en concreto pertenece a un solicitante, pero un solicitante puede estar asignado a una obra o más. Por eso también tiene la id del solicitante como clave foránea;

La tabla user están los usuarios registrados para la aplicación. Dicha tabla tiene estos atributos:

- Id: identificador del usuario
- nombre: nombre del usuario
- apellidos: apellido del usuario
- email: correo del usuario
- password: contraseña del usuario

Ahora voy con la tabla comentarios, en la cual se guardan comentarios.

La tabla comentarios tiene estos atributos:

- Id: identificador del comentario
- Fecha_inicio: fecha del comentario
- texto: contenido del comentario

En cuanto a las relaciones de las tablas, la tabla comentarios tiene varias relaciones.

Un comentario pertenece a una única obra, pero una obra puede tener uno o varios comentarios o ninguno. Esto se puede ver de la misma manera con la tabla users, dado a que un usuario puede haber creado un comentario o más, pero un comentario ha sido creado de parte de un usuario.

CONTENIDO DEL ENTORNO TECNOLÓGICO (COMO SE CONTROLA LA PÁGINA WEB)

Anteriormente mencioné que íbamos a detallar más el framework, que íbamos a utilizar llamado laravel. Ahora es el momento.

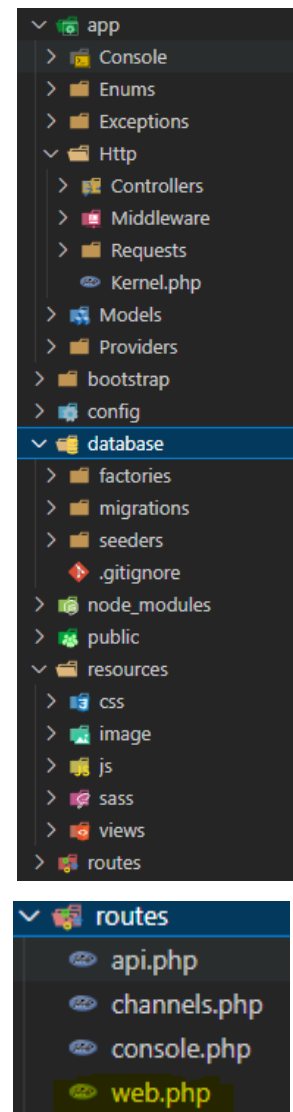
Laravel es un framework php para ayudar en un tipo de desarrollo sobre aplicaciones más completas, con la posibilidad de incluir otros lenguajes como js o css.

También hablaré de lo que es vite, dado a que es la herramienta frontend que utilizaremos para el proyecto:

Vite es un framework de compilación de código para el producción completa, también conocido como ViteJS.

El sistema de archivos esta separado en varias partes:

- App/http/controler: ahí se encuentran los controladores php
- App/Models: ahí se encuentran todos los modelos que se utilizan tanto para los controladores como para el uso de la base de datos
- Database: carpeta que contiene todo el contenido que realiza el proyecto. Tiene tres carpetas dentro:
 - Factories: carpeta que tiene el modelo para las inserciones principales o añadidos de forma automática a través de un comando específico, útil para los seeders dentro de la carpeta seeders.
 - Migrations: contenido de todas las tablas que se insertan en mi base de datos
 - Seeders: implementaciones principales que se realizan para las tablas que se crean por primera vez, o cuando se realiza una recarga completa de la base de datos.
- Resources: carpeta que contiene todas las carpetas/archivos necesarios para la interfaz de la pagina web.
 - Css
 - Js
 - Image: carpeta que contiene las imágenes comunes de la aplicación web
 - Sass
 - Views: vistas que se realizan en la pagina web.
- Routes/web.php: archivo donde se fluctuan todas las vistas de la pagina web
- .env: configuracion basica
- Vite.config.js: archivo donde se configuran todas las publicaciones que se implementan dentro de la pagina web.

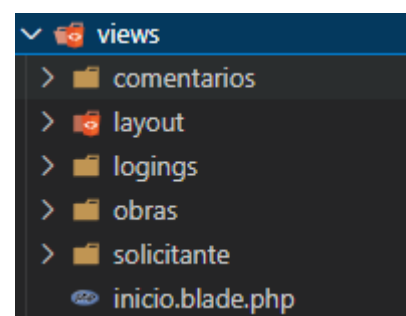


PLANTILLA DE LAS PAGINAS WEB

En el apartado de resources, se implementa todos los contenidos de la interfaz de la pagina, pero todas las vistas necesarias estan dentro de la carpeta views.

Dentro de ello, existen varias carpetas cuyo objetivos son los siguiente:

- Comentarios: contiene la vistas de los comentarios
- Layout: layout principal de la mayoría de las paginas web (la cabecera y el pie de la pagina)
- logings: carpeta de autenticacion de los usuarios que se implementan en el inicio de usuario
- obras: contiene la vista de la creacion de una obra, listado y edicion de la obra

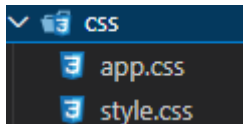


- solicitante: contiene la vista de creacion y listado de solicitantes.
- Inicio.blade.php:archivo de inicio tras realizar inicio de session

CSS

En el apartado de css están todas las implementaciones de estilos de todas las paginas web creados durante este proyecto.

Por defecto todos los estilos que se implementan dentro de la pagina estan con la modalidad de utilizar bootstrap



Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseños de sitios y aplicaciones web.

En mi caso la mayor parte del diseño se refleja en las clases de la aplicación web

COTROL DE IMÁGENES/ARCHIVOS

En el apartado de imagines están guardados todas las imágenes que contienen en la página de forma simple.

ENTORNO CLIENTE: JAVA SCRIPT/TYPESCRIPT Y VUE

En el apartado de js están todos los archivos de script de JavaScript implementadas en todas las páginas.

Dado a que he utilizado nuevas reglas dentro de la aplicación, voy a explicar paso por paso.

En el fichero app.js esta la raíz de utilización de todos los componentes que creamos a través de Vue, imágenes, etc, dado a necesidades de implementación con laravel vite.

Vue es un framework js de código abierto en la cual permite construir interfaces y componentes de forma completa y sencilla. Además, gracias a vite podemos implementar de forma sencilla dicho framework con laravel de forma sencilla, teniendo importado dichos archivos Vue dentro de un js que se utilice dentro del 'plugins[laravel ({input:""})]'.

También esta bootstrap.js para la importación base del Bootstrap en todas las páginas web.

También existen dos ficheros llamado contrasena.js y contrasena.ts . esto tiene una explicación sencilla.

En este reto todos los archivos que he querido utilizar y crear de forma manual dentro del entorno cliente son typescript y vue.

Typescript es un superconjunto de JavaScript que añade tipos estáticos y objetos basados en clase. Es decir es una mezcla de JavaScript y java pero comiendo las ventajas de cada una.

Con la utilidad de Typescript, te permite traspasar los archivos de typescript a JavaScript realizando el comando tsc (type script compiler) coma ha sucedido con el archivo de contraseña.js

El fichero contraseña.js tiene la posibilidad de visualizar/ocultar la contraseña que se ve a través de un ojo.

El fichero App.vue esta el componente base que se implementa a través de un componente el apartado del año actual del copyright dentro del pie de la página.

SASS

Para todo el css que he implementado en este reto, he querido tener un procesador de hojas de estilo css para poder tener los diseños de las hojas de estilos mas personalizables en las cuales no se podía hacer por php

WEB.PHP

Para que la aplicación funcione, tenemos que saber por dónde tiene que ir todas las vistas, datos y protecciones de autenticación en mi pagina. Por ello , en laravel existe este fichero.

Con ello podemos enlazar todas las url necesarias para el dominio , especialmente para controlar el contenido que se ingresa dentro de la barra de marcadores.

En mi caso he utilizado específicamente para encargar el flujo de solicitudes de respuesta desde y hacia el cliente con tres parámetros principales.

```
Route::get($uri, $callback)->name($name);
Route::post($uri, $callback )->name($name);
Route::delete($uri, $callback )->name($name);
```

- \$uri: dirección que aparece dentro de la URL
- \$callback: contenido que devuelve esa url
- \$name: alias que se asignan a formularios/enlaces/botones de envío/etc.

También existe controles de middleguard dentro de la aplicación, para el control de sesiones:

```
Route::middleguard('auth:') ->group(function(){
Route::get($uri, $callback)->name($name);
Route::post($uri, $callback )->name($name);
});
```

Aun teniendo el formato de enrutamiento, el \$callback se ha utilizado de manera mas avanzada, teniendo que llamar a un controlador con una función base.

```
Route::post($uri,['Controlador'::class,'función dentro del controlador']) ->name($name);
```

Todos los enrutamientos están comentados para el conocimiento de la aplicación web

CONSIDERACIONES SOBRE LA IMPLEMENTACION

Todo el contenido mencionado en el diseño de la página web está bien, pero hay cosas que nos gustaría mencionar de una manera más detallada.

MANUAL DE USUARIO

Para ver de forma mas detallada todas las funciones de mi pagina esta en el anexo que se implementa en mi apartado de anexos.

CONSIDERACION DEL ENTORNO CLIENTE: VUE

Todo el contenido de la estructura del vue es un poco lioso, asi que voy por partes.

Para la instalación de vue a través de laravel vite, utilizamos un plugin llamado @vitejs/plugin-vue

Y meterlo dentro del plugin la aplicación vue , importando en vite dicho plugin.

```
import vue from '@vitejs/plugin-vue';
```

```
plugins: [
  laravel({
    input: ['resources/css/app.scss', 'resources/js/app.js'],
    refresh: true,
  }),
  vue({
    template: {
      transformAssetUrls: {
        base: null,
        includeAbsolute: false,
      },
    },
  }),
],
```

Con ello, dentro del plugin “laravel”, metemos el js que importaremos el contenido del vue dentro de ese input.

Después, en el archivo donde se ha metido ese input, se inserta una serie de import dos cosas:

- el import de la creación de la app de vue
- el import del archivo vue donde esta el componente raíz

luego teniendo esos importes, se puede montar en donde nosotros queramos. teniendo una id seria lo más cómodo.

```
import { createApp } from 'vue';
//app absico
import App from './App.vue';
createApp(App).mount("#unapp");
```

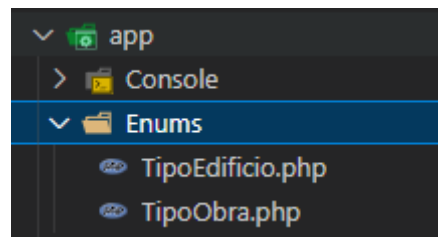
Si se desea montar mas de uno, no hace falta de importar la primera parte otra vez, solamente se tendrá que importar otra diferente, aunque no se porque puede dar problemas.

CONSIDERACION DEL ENTORNO SERVIDOR: LARAVEL 10

Desde el reto anterior, se ha realizado una nueva implementación de laravel a la versión 10, en la cual se han implementado varios apartados, y quería implementar algo con ello: las enumeraciones.

Laravel 10 y sobrepasa el php 8.1, con lo cual soporta las enumeraciones. En mi caso los he implementado en una carpeta:

- TipoEdificio: (piso, casa, local, garaje, trastero o edificio)
- TipoObra: (nueva construcción o reforma)



Dentro de ello, gracias a los casos que se implementan se pueden listar a través de las vistas de una manera más eficiente:

Ejemplo: sacar todos los casos de enumeración del TipoEdificio

```
@foreach(App\Enums\TipoEdificio::cases() as $edificio)
    <option value="{{ $edificio->value }}">{{ $edificio->value }}</option>
@endforeach
```

CONSIDERACION DE ENVIO DE DATOS:

Todos los datos que necesitemos enviar y recibir están en los controladores relacionados con cada modelo. Gracias a ello, antes de que enseñásemos las vistas de cualquier pagina, realizábamos todas las acciones de las paginas.

Ejemplo: inserción de un nuevo solicitante

```
public function store(SolicitanteRequest $request)
{
    $validated = $request->validated();

    $st= Solicitante::create([
        'nombre'=>$validated['nombre'],
        'apellidos'=>$validated['apellidos'],
        'DNI'=>$validated['DNI'],
        'direccion_residencia'=>$validated['direccion_residencia'],
        'email'=> $validated['email'],
        'tel'=> $validated['tel'],
    ]);

    return redirect()->route('inicio')->with('status', 'Solicitante creado!');
}
```

Ejemplo: mostrar todas las obras


```

public function show(obra $obra)
{
    $obras=[];
    $listaobras=Obra::all();
    //Vista de notas
    foreach($listaobras as $unaobra)
    {
        $solicitante = Solicitante::whereId($unaobra->solicitante_id)->first();

        array_push($obras, [
            "id"=>$unaobra->id,
            "solicitante" => $solicitante->nombre,
            "edificio" => $unaobra->tipo_edificio,
            "obra" => $unaobra->tipo_obra,
            "fecha_inicio" => $unaobra->fecha_inicio,
            "fecha_fin" => $unaobra->fecha_fin
        ]);
    }
    return view ('obras.listaobra',['obras'=>$obras]);
}

```

Además, al realizar cualquier acción de inserción y cambio, en la pantalla principal avisa con una alerta de aviso de suceso:

Ejemplo: suceso de nueva obra

CONSIDERACION DE CONTROL DE ESTILOS:

Todos los estilos, como ya está mencionado en la plantilla de las páginas web, tiene un estilo detallado de cada página. En mi caso he realizado varias visualizaciones básicas dependiendo del tamaño de la ventana de la página web.

Para comenzar, todo el contenido de la página está separado en dos partes:

- nav: 2/12 de la columna proporcionada
- Main_part: el resto del contenido
- Footer: debajo

Todas las partes y sus interiores están con los prefijos de las clases que se implementan gracias a los breakpoints que no proporciona Bootstrap. Estos son las medidas que nos facilitan:

	Extra pequeñas	Pequeñas	Medianas	Grandes	Extra Grandes	Extra extra grandes
Anchura máxima	576px	768px	992px	1200px	1400px	>1400px
Prefijo de la clase	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-	.col-xxl-
Número de columnas	12					

APLICACIONES EXTRAS UTILIZADAS EN EL RETO

Todas las aplicaciones mencionadas en los apartados, solamente mencionamos una aplicación de despliegue de la página web, un par de frameworks que utilizamos, un par de planificadores. Queda varios más por comentar que he utilizado de forma diaria en el proyecto:

Para el editor de código de mi página web, he utilizado Visual Studio Code.

Visual Studio Code es un editor de código fuente que comparte varios lenguajes de programación y un conjunto de características que puede o no estar en la interfaz del usuario.

En este caso he utilizado para la edición de los lenguajes de php, JavaScript, CSS, vue, typescript y YAML (solamente para el docker-compose.yml).

Para el control y compartimiento de archivos de este proyecto se ha utilizado dos aplicaciones para no tener dependencia de una: GitHub y Google Drive.

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones. Gracias a ello podemos controlar todos los cambios y separar las modificaciones sin la necesidad de alterar con otros compañeros al realizar cambios a los ficheros. También se utiliza para probar y desplegar el código

Google Drive es un servicio de alojamiento y sincronización de archivos con la posibilidad de compartimiento con multitudes personas. En mi caso lo he utilizado para realizar copias de seguridad por si no hay disponibilidad de subir a la nube a través del GitLab.

INCIDENCIAS SURGIDAS

Aun teniendo una versión mejorada de laravel, ha habido muchísimos problemas con el contenido del despliegue, incluyendo el contenido del retorno que se creaba en los nuevos modelos/controladores que se quería retornar(no se porque quería que retornase algo que ya daba el retorno)

También el control de las nuevas funcionalidades dio muchos problemas de creación de componentes, y se que la nueva versión de laravel no lleva ni 15 días.

CONCLUSION

Gracias a este proyecto, he aprendido con mejor profundidad la funcionalidad y la infraestructura de un framework de pagina web.

ANEXOS

Pagina web: <https://recunai.up.railway.app>