DOCUMENTACIÓN

RETO 3 - GESTOR DE INCIDENCIAS
ASCENSORES IGOBIDE

GRUPO 3

Alaitz Candela
Txaber Gardeazabal
Daniel Tamargo

ÍNDICE

ESTRUCTURA DEL PROYECTO

URLs de la aplicación:

Aplicación desplegada en la web: https://igobide.herokuapp.com/
Repositorio GitHub: https://github.com/DanielTamargo/grupo3-reto3

Diagramas:

Gantt: https://sharing.clickup.com/g/h/4-54652996-7/f79c69294e57923
Base de Datos: https://dbdiagram.io/d/61ee76fe7cf3fc0e7c5b7334
MER: https://i.gyazo.com/9a019dc5abce0ea7b15f782261bad331.png

Otros diagramas:
Casos de uso.
Diagrama de secuencia creación de tarea.
Diagrama de secuencia trabajo técnico.

Entorno:

Vagrant box: **Laravel Homestead v12.0.0 (Laravel 8)**
Enlace al vagrant box.
Documentación oficial Laravel 8.

Motor de renderizado de plantillas: **Blade**

Maquetación:

Framework: **Bootstrap v5.1.3**
Documentación oficial de Bootstrap.

Estilos modificados y trabajados con: **Sass** y **CSS**
Sass versión: 1.48.0
Documentación Dart Sass Cli.

Despliegue:

Aplicación Web desplegada en: **Heroku**
Página oficial de Heroku.

DISEÑO

RWD

Responsive Web Design es una parte fundamental para el desarrollo web ya que nos permite que nuestra aplicación se adapte correctamente a los distintos dispositivos sin importar el tamaño de pantalla. En este reto hemos tenido en cuenta las necesidades del cliente a la hora de decidir el planteamiento y disposición de cada vista. Teniendo en cuenta dos puntos importantes:

1. El cliente nos dijo que los técnicos solo iban a utilizar dispositivos móviles a la hora de usar nuestra aplicación, así que si hiciéramos la parte del técnico viéndose también desde el ordenador, sería una pérdida de tiempo y a la hora de programar el tiempo es muy valioso.

2. Teniendo lo anterior en cuenta, el resto de usuarios (operarios, jefes de equipo y administradores) utilizarán dispositivos donde las medidas no se ceñirán a unas medidas fijas, por lo que debemos adaptarnos a distintas dimensiones y pantallas pudiendo utilizar la aplicación desde cualquier dispositivo disponible y, si en un futuro cambian de dispositivos, no tengan ningún problema.

Layouts

Hemos diseñado nuestros layouts **pensando siempre en el cliente y sus necesidades**. Una de sus prioridades era que la aplicación tuviese una funcionalidad clara e intuitiva.

Por esto mismo nuestros diseños están enfocados a la funcionalidad. Hemos tratado de combinar esta funcionalidad con un aspecto simple pero elegante, respetando los colores de la empresa y una máxima homogeneidad posible.

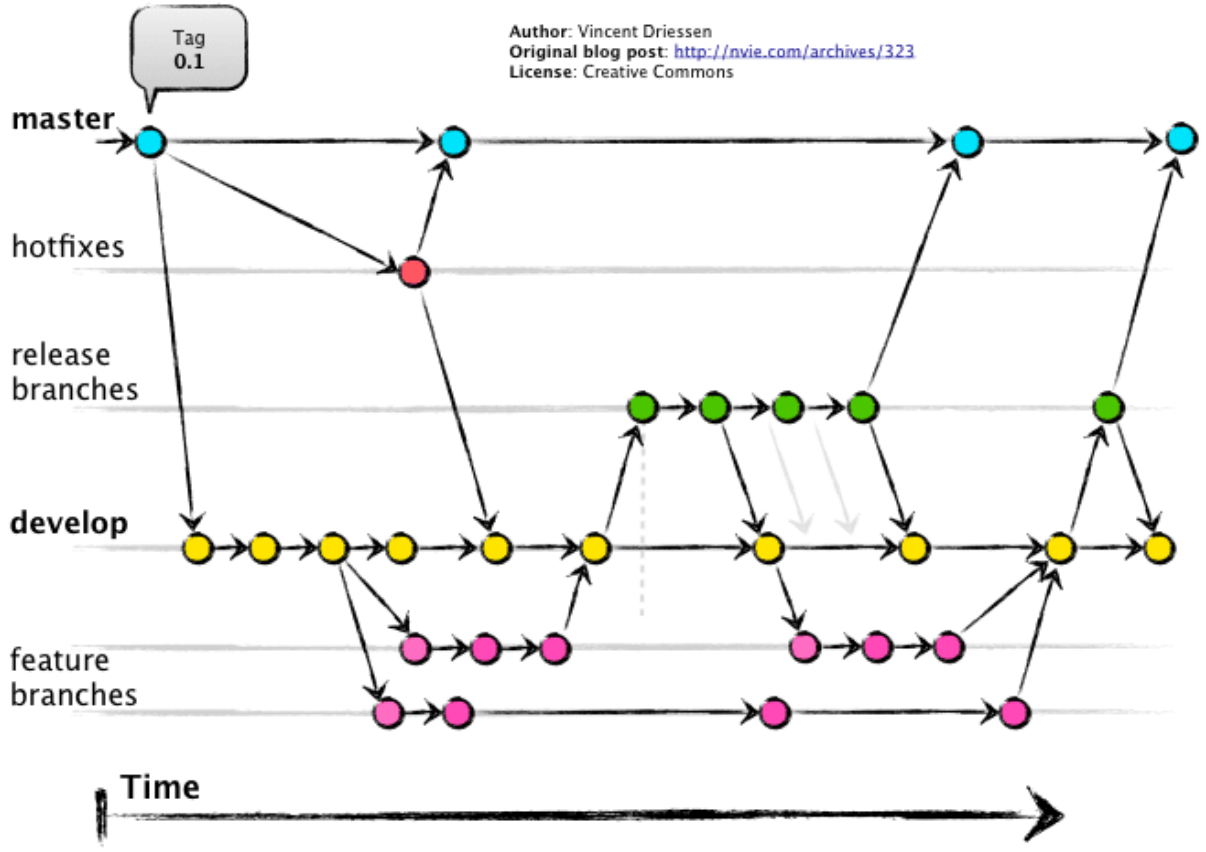
Dentro de la página podemos encontrar dos estilos distintos, puesto que el enfoque que se ha dado con el servicio que ofrece la aplicación a los técnicos está enfocado a un uso sencillo y claro desde un dispositivo móvil, mientras que el servicio para los demás roles está más enfocado para dispositivos como tablets y ordenadores (tal y como comentó el propio cliente).

METODOLOGÍA GIT

Como metodología git hemos utilizado **Git Flow**, una metodología enfocada a disponer de una rama principal de desarrollo (develop), una rama de producción (main), una rama donde convergen versiones estables antes de subir a producción (release) y múltiples ramas de desarrollo de funcionalidades a implementar después en develop (feature-.....).

También existe una rama auxiliar para ejecutar cambios rápidos y puntuales sobre posibles errores que puedan surgir en la rama de producción (hotfix).

Un esquema ejemplo de la metodología git flow es el siguiente:



Importante destacar que debido al movimiento **black lives matter**, en numerosos proyectos y estándares la rama **master** ha pasado a llamarse **main** o **mucho**.

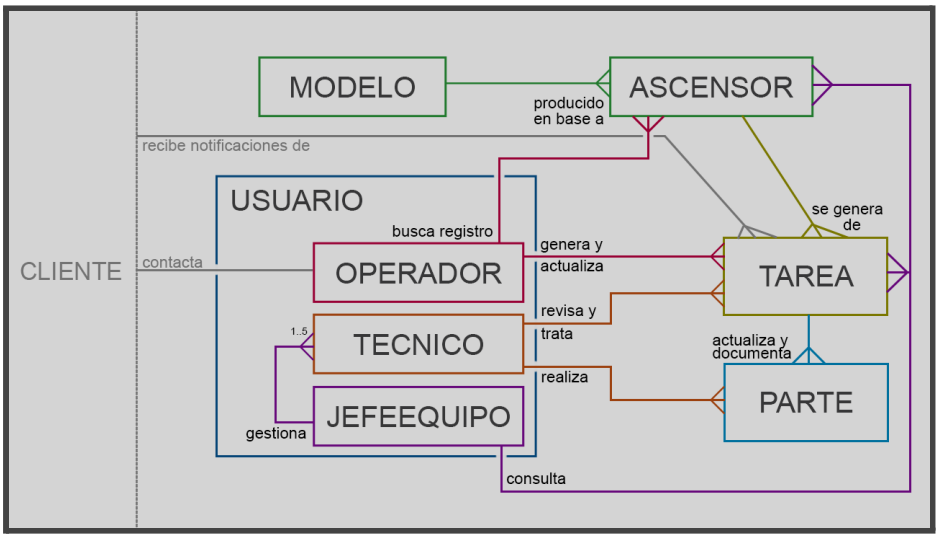
Fuente:
https://platzi.com/blog/cambios-en-github-master-main/

DIAGRAMAS

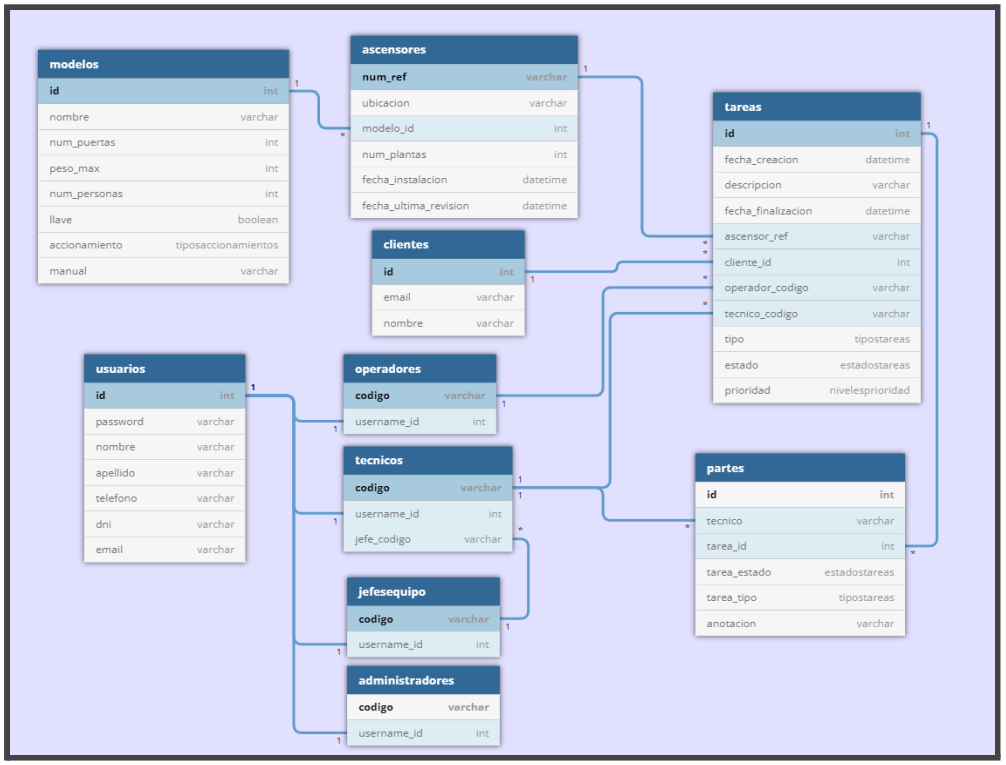
Para diseñar y preparar una **base de datos** completa, robusta y estable, que contemplase los datos y necesidades aportados por el cliente hemos diseñado dos diagramas:

- Diagrama Modelo Entidad Relación
- Diagrama de Base de Datos

MER



DBDiagram



Para realizar el diagrama de base de datos hemos utilizado una herramienta online llamada DBDiagram.io, se puede consultar el diagrama en el siguiente enlace: https://dbdiagram.io/d/61ee76fe7cf3fc0e7c5b7334

HEROKU

Como servicio de despliegue de aplicaciones web en la nube hemos utilizado **Heroku**.

Para ello hemos tenido que registrarnos y crear nuestra aplicación, la cual está vinculada al repositorio GitHub donde subimos nuestro trabajo.

Una herramienta muy útil y conveniente es el **despliegue automático**, haciendo que cada vez que subamos nuevos cambios a una rama en concreto (podemos indicar cualquier rama existente en nuestro repositorio), Heroku procesará y desplegará dichos cambios, haciendo todo el proceso automático y mucho más eficiente.

Heroku despliega nuestra página web en:
https://igobide.herokuapp.com/

Nota:
Hemos sopesado la posibilidad de comprar un dominio u obtener un dominio gratis como por ejemplo **igobide.tk**, pero finalmente llegamos a la conclusión de que el dominio es mucho más atractivo si finaliza en **.com** y comprar un dominio .com llevaría costos innecesarios para el proyecto, por lo que dejamos el por defecto que proporciona heroku, herokuapp.com

Cómo hemos configurado nuestro proyecto en Heroku:

- Para la base de datos hemos instalado un add-on llamado **JawsDB MySQL** (también existe JawsDB MariaDB). Este add-on nos crea un espacio en su servicio proporcionándonos una base de datos y unas credenciales para acceder a ella y manipularla.
- En Laravel, tanto las variables de las credenciales de la BBDD como otras variables de entorno son configuradas en el fichero .env, este fichero de ninguna manera debe subirse a repositorios públicos, por lo que en el repositorio no estarán. Para que Heroku disponga de dichas variables de entorno tendremos que acceder a la pestaña configuración de la aplicación y añadirlas. En nuestro caso hemos añadido las credenciales necesarias para que funcione correctamente la aplicación, la BBDD y el servicio SMTP (envío de e-mails).

Extra: Datos auto generados en la nube
El cliente pedía que el proyecto funcionase como un piloto, por lo que no disponíamos de datos reales que inculcar en nuestro proyecto. Por lo que para realizar una generación masiva de datos hemos aprovechado la librería fakerphp y la herramienta seeding de Laravel. Una vez preparados, para ejecutarlo también en Heroku y disponer de los datos en la BBDD de la nube, hemos tenido que acceder a las opciones, abrir una consola y ejecutar los comandos:

```
composer install
composer dump-autoload
php artisan migrate:fresh --seed
```

DOCUMENTACIÓN FRONT-END

JavaScript es el lenguaje de programación interpretado por el navegador que se encargará de la lógica en la parte front-end de la aplicación, es decir, en la parte expuesta de cara al usuario.
Para apoyarnos y desarrollar JavaScript con más funcionalidades y de una forma más productiva hemos utilizado tanto librerías como TypeScript.

TypeScript

En algunas vistas, para desarrollar JavaScript hemos utilizado **TypeScript**.
Como bien indica este post, TypeScript (TS) es un lenguaje de programación construido a un nivel superior de JavaScript (JS). Esto quiere decir que TypeScript dota al lenguaje de varias características adicionales que hacen que podamos escribir código con menos errores, más sencillo, coherente y fácil de probar, en definitiva, más limpio y sólido.

Mientras desarrollamos en TS, tenemos que ir compilando para que nos genere el fichero JS ya que este será el fichero a interpretar por el navegador.

Véase aquí la diferencia entre un fichero desarrollado en TS y su resultado en JS.

TypeScript basa su configuración en un fichero llamado **tsconfig.json**, donde a través de unos parámetros podremos personalizar su configuración al completo. Por ejemplo, podemos indicarle qué archivos incluir para buscar la existencia de funciones y clases, qué archivos excluir para evitar duplicados, y otras configuraciones como que elimine los comentarios al compilar (entre otras muchas). Un ejemplo de una función en TS y en JS:

```ts
/**
 * Obtiene un string aleatorio y lo pone en el input como passwo
 * @param {int} length define la longitud de la contraseña
 */
function passwordAleatoria(length: number=10): void {
    let password: string = cadenaAleatoria(length);
    $("#registro-password").val(password);
}
```

```js
function passwordAleatoria(length) {
    if (length === void 0) { length = 10; }
    var password = cadenaAleatoria(length);
    $("#registro-password").val(password);
}
```

Aquí se puede consultar toda la configuración disponible para personalizar en el fichero tsconfig.json.

Web Components

Para no complicar el proyecto intentando abarcar más de lo que pudiéramos contemplar en las 3 semanas que disponemos para desarrollarlo, descartamos la opción de utilizar Vue.js.

Por lo que optamos por **generar los web components en JavaScript plano**.
Como **Laravel** ya nos proporciona herramientas como uso de layouts y renderizado de plantillas (blade), hemos empleado los web components para generar determinados botones en cada panel de los distintos roles de usuarios.

Hemos creado el component **boton-panel**, que será nuestro custom tag donde **cargaremos nuestro web component**.

```
<boton-panel
texto="Ver lista de empleados"
rol="administrador"
id_p="1"
ruta="{{ route('empleados.index') }}"></boton-panel>
```

A través de los atributos del componente podremos enviarle datos para que los utilice al ser generado o ejecutar acciones. Por ejemplo, **combinamos blade con el web component** para enviarle la ruta a la que tendrá que acceder una vez es clicado.
Se puede consultar todo el código del web component aquí.

Comprobaciones

Pese a que la mayoría de validaciones y comprobaciones de datos y privilegios se realizan de parte del servidor, en la parte front-end también tenemos algunas comprobaciones, como la validación de formato y código del DNI.

```
dni = dni.toUpperCase();
let regex_dni: RegExp = /^\d{8}[a-zA-Z]$/;

// Comprobamos la expresión regular
if (!regex_dni.test(dni)) {
    // Error: ¡Formato no válido!
    //console.log('Error: formato DNI no válido');
    return false;
}

// Comprobamos el valor de la letra del DNI
let lista_letras: string = 'TRWAGMYFPDXBNJZSQVHLCKET';
let numero: number = Number(dni.substring(0, dni.length - 1));
let letra_dni: string = dni.substring(dni.length - 1);
if (letra_dni != lista_letras.charAt(numero % 23)) {
    // Error: ¡DNI no válido!
    //console.log('Error: DNI no válido');
    return false;
}

// ¡DNI Válido!
//console.log('¡DNI válido!');
return true;
```

DOCUMENTACIÓN PHP

Para la parte de back-end hemos utilizado Laravel. Laravel es un marco de aplicación web con una sintaxis expresiva y elegante. Las bases están sentadas, para no preocuparnos de la estructura del proyecto. Nos ayuda a mantener una sintaxis limpia y simple, para que sea entendible por cualquiera que coja nuestro código y pueda entenderla fácilmente. Además, nos proporciona varias funcionalidades:

- El acceso a una base de datos MySql, Postgres, SQLite y SQL Server para poder crear tablas, insertar/borrar o modificar datos e incluso hacer consultas a esta.

```php
Schema::create('tecnicos', function (Blueprint $table) {
    $table->string('codigo', 20)->primary();
    $table->unsignedBigInteger('user_id');
    $table->string('jefe_codigo', 20);
    $table->timestamps();

    // Relación con el usuario
    $table->foreign('user_id')
        ->references('id')
        ->on('users')
        ->onDelete('cascade');

    // Relación con el jefe
    $table->foreign('jefe_codigo')
        ->references('codigo')
        ->on('jefes_equipos')
        ->onDelete('cascade');
});
```

```php
$alaitz = \App\Models\User::create([
    'nombre' => 'Alaitz',
    'apellidos' => 'Candela Murelaga',
    'email' => 'alaitz.candela@igobide.com',
    'email_verified_at' => now(),
    'telefono' => '693248546',
    'rol' => Roles::ADMINISTRADOR, // administrador, tecnico, operador, jefeequipo
    'dni' => '81939760Y',
    'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
    'remember_token' => 'abcdefghij',
]);
$alaitz->save();

$alaitz = \App\Models\Administrador::create([
    'codigo' => "adm_" . str_pad($alaitz->id, 5, "0", STR_PAD_LEFT),
    'user_id' => $alaitz->id
]);
$alaitz->save();
```

- El uso de colas para enviar trabajos más lentos en segundo plano como enviar correos electrónicos, generar informes… mientras mantiene tiempos de respuesta rápidos.

```php
<body>
    <h1 class="titulo">Ascensores Igobide</h1>
    <p>Estimado/a {{ Str::ucfirst($detalles['nombre']) }}, <br>
        Le comunicamos que ya hemos tramitado el registro de la incidencia. <br>
        Se ha asignado a uno de nuestros técnicos disponibles y lo resolverá lo antes posible.
    <p>Si tiene alguna duda o indicación adicional no dude en volver a contactarnos llamando de nuevo o respondiendo a este
    <p>Un saludo, <br>
        Ascensores Igobide</p>
</body>
</html>
```

- Al implementar la autenticación podemos acceder al usuario que ha iniciado sesión para obtener su información y cotejar si dispone de los privilegios necesarios para ejecutar cada acción o navegar por la página.

```php
// Comprobamos que tiene permisos
$user = Auth::user();
if ($user->rol != "administrador" && $user->rol != "jefeequipo") {
    return response()->json([
        'ok' => false,
        'message' => 'No dispones de los suficientes permisos para solicitar estos datos',
        'rol' => $user->rol
    ]);
}
```

- El uso de API creando una interfaz para juntar JavaScript con PHP y poder hacer peticiones desde Javascript.

```
}
if(Auth::user()->rol == 'jefeequipo'){
    //$tareas = Tarea::where('ascensor_ref', 'like', "%$filtro_numref%")->where('tipo','like',"%$filtro_tipo%")-
    $tareas = array_filter(Tarea::where('ascensor_ref', 'like', "%$filtro_numref%")->where('tipo','like',"%$filt
        // dd(array_column(Auth::user()->puesto->tecnicos->toArray(), "codigo"));
        return in_array($tar['tecnico_codigo'], array_column(Auth::user()->puesto->tecnicos->toArray(), "codigo"
    });
    // dd($tareas);
    $tareas = array_values($tareas);
}

return response()->json([
    'ok' => true,
    'tareas' => $tareas,
    'filtro' => $filtro_estado,
    'ascensores' => $ascensores,
    'tecnicos' => $tecnicos,
], 200);
```

```
$.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': jQuery('meta[name="csrf-token"]').attr('content')
    }
});
$.ajax({
    type:'GET',
    url:"/api/v1/tareas",
    data:{
        'filtro_numref': filtro_numref,
        'filtro_estado': filtro_estado,
        'filtro_tipo': filtro_tipo
    },
    success: function(json){
        console.log(json);
        //cogo todas las tareas e inicializo las primeras 10
        todas_tareas = json['tareas'];
        tecnicos = json["tecnicos"];
        ascensores = json["ascensores"];
        mostrarTareas(null);
    }
})
```

- Laravel utiliza Eloquent como ORM, está construido y mapeado de tal forma que, si hemos configurado bien las migraciones y los modelos, a través de las relaciones podemos ir obteniendo los datos que necesitemos. También podemos ejecutar queries fáciles de construir, rápidas y efectivas.

```
$tareas = Tarea::where('ascensor_ref', 'like', "%$filtro_numref%")
        ->where('tipo','like',"%$filtro_tipo%")
        ->where('estado','like',"%$filtro_estado%")
        ->orderBy('id', 'desc')
        ->get();
```

También podemos unir las funciones de php para trabajar con las colecciones de Eloquent (¡ojo! hay que transformar esas colecciones en arrays con el método toArray).

```
$tareas = array_filter(Tarea::where('ascensor_ref', 'like', "%$filtro_numref%")
                        ->where('tipo','like',"%$filtro_tipo%")
                        ->where('estado','like',"%$filtro_estado%")
                        ->orderBy('id', 'desc')
                        ->get()
                        ->toArray(), function($tar) {
    return in_array($tar['tecnico_codigo'], array_column(Auth::user()->puesto->tecnicos->toArray(), "codigo"));
});
```

Faker

Faker es una librería para crear datos falsos, ya sea para hacer pruebas en tu proyecto, iniciar la base de datos, crear documentos XML… En nuestro caso lo hemos usado para crear unos datos falsos y poder tener en cuenta cómo se vería y funcionaria la página web en un caso real.

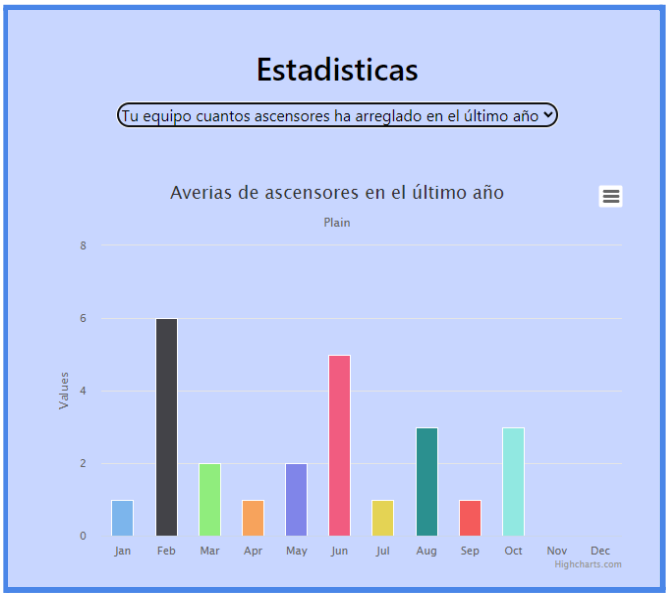
```php
 */
public function definition()
{
    // Se generan fechas de instalación aleatorias desde 1 de Enero de 1990 hasta hace 7 días
    $fecha_instalacion = new DateTime();
    $fecha_instalacion->setTimestamp($this->faker->numberBetween(946688461, time() - (7 * 24 * 60 * 60)));

    return [
        'num_ref' => Str::random(10),
        'ubicacion' => $this->faker->address(),
        'num_plantas' => $this->faker->numberBetween(2, 8),
        'fecha_instalacion' => $fecha_instalacion,
        'fecha_ultima_revision' => $fecha_instalacion // <- se actualizarán más adelante
    ];
}
```

```php
 */
public function run()
{
    // VARIABLES QUE VARIARÁN EL NÚMERO DE DATOS A GENERAR
    $v_num_jefes_equipos = 5; // mínimo 1, default 5
    $v_num_min_tecnicos_equipo = 1; // mínimo 1, default 1
    $v_num_max_tecnicos_equipo = 5; // default 5
    $v_num_min_operadores = 4; // mínimo 1, default 4
    $v_num_max_operadores = 8; // default 8
    $v_num_min_ascensores_modelo = 1; // default 1
    $v_num_max_ascensores_modelo = 8; // default 8
    $v_num_min_tareas = 0; // POR ASCENSOR, default 0
    $v_num_max_tareas = 15; // POR ASCENSOR, default 15
    $v_num_min_tareas_pendientes = 15; // default 15
    $v_num_max_tareas_pendientes = 30; // default 30

    // COMIENZO DEL SEEDING
    $this->command->info("Starting Seeding. ");
    // ------------------------------------------------------------------------
```

Highcharts

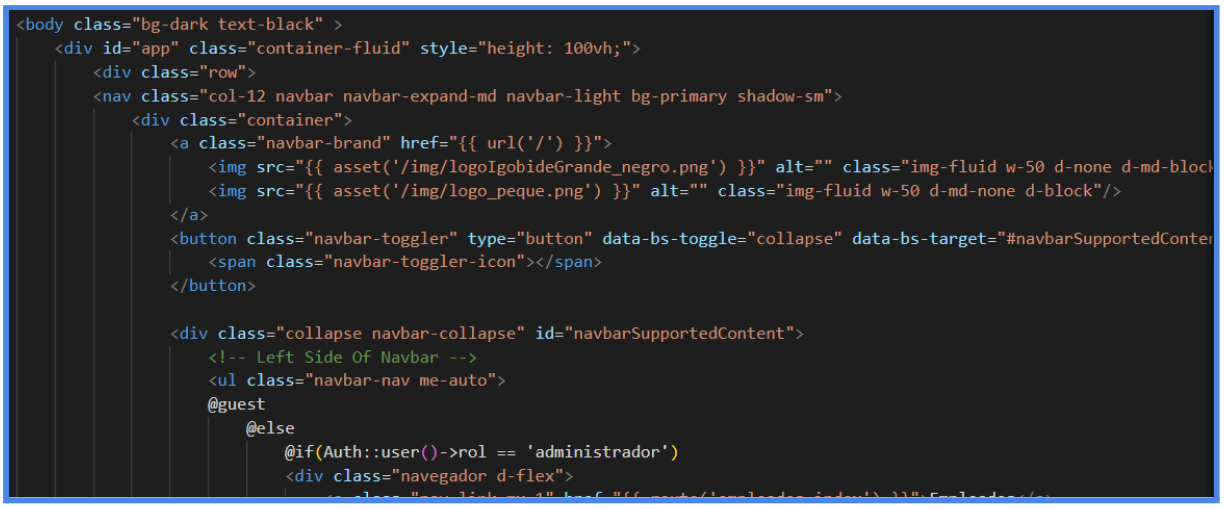
Highcharts es una librería utilizada para crear visualizaciones de datos confiables y seguras(estadísticas). Esta librería usa JavaScript y TypeScript y funciona con cualquier base de datos de back-end ofreciendo envoltorios para los lenguajes de programación más populares: PHP, Python , Java.. así como para Android e IOS y marcos como angular, Vue y React. La librería a la hora de crear nuestras estadísticas nos ofrece la posibilidad de poder descargarlas (en formato PDF, PNG, SVG, JPEG), verlo en pantalla completa,imprimirla y ver los datos en diferentes formatos (CSV, XLS o tabla de datos, con los dos primeros es descargable).

Ejemplos de estadísticas que se pueden hacer:

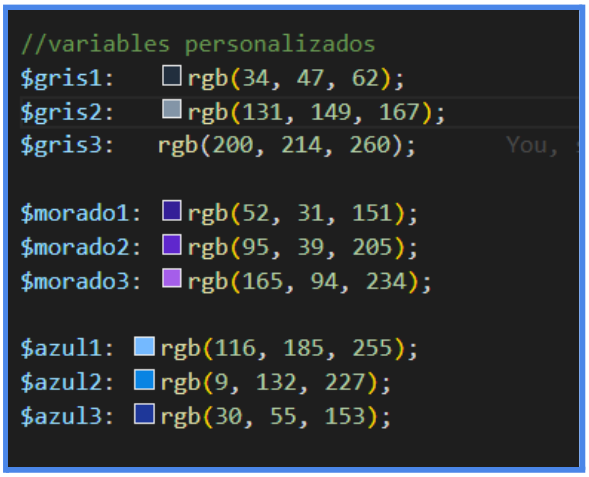


Bootstrap

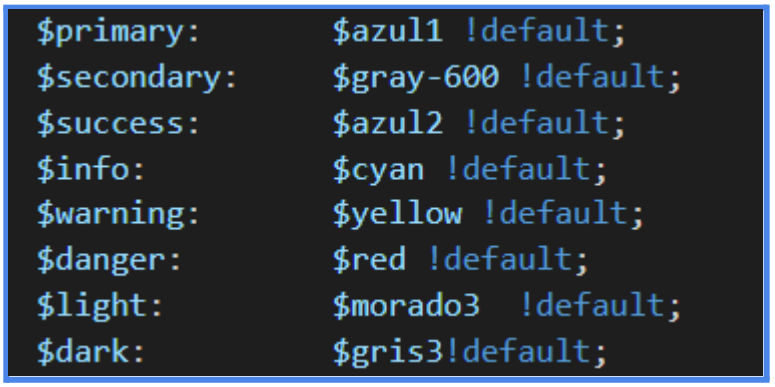
Bootstrap es una librería que nos ayuda a diseñar y personalizar rápidamente nuestras páginas web. Es el conjunto de herramientas de código abierto de front-end más popular en el mundo, que incluye variables y mixins de Sass, un sistema de cuadrícula receptivo, amplios componentes preconstruidos y potentes complementos de JavaScript. En el proyecto hemos utilizado bootstrap para la maquetación de la página y la personalización de esta mediante las variables de Sass.



```
<body class="bg-dark text-black" >
    <div id="app" class="container-fluid" style="height: 100vh;">
        <div class="row">
        <nav class="col-12 navbar navbar-expand-md navbar-light bg-primary shadow-sm">
            <div class="container">
                <a class="navbar-brand" href="{{ url('/') }}">
                    <img src="{{ asset('/img/logoIgobideGrande_negro.png') }}" alt="" class="img-fluid w-50 d-none d-md-block
                    <img src="{{ asset('/img/logo_peque.png') }}" alt="" class="img-fluid w-50 d-md-none d-block"/>
                </a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedConte
                    <span class="navbar-toggler-icon"></span>
                </button>

                <div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <!-- Left Side Of Navbar -->
                    <ul class="navbar-nav me-auto">
                    @guest
                        @else
                            @if(Auth::user()->rol == 'administrador')
                            <div class="navegador d-flex">
```



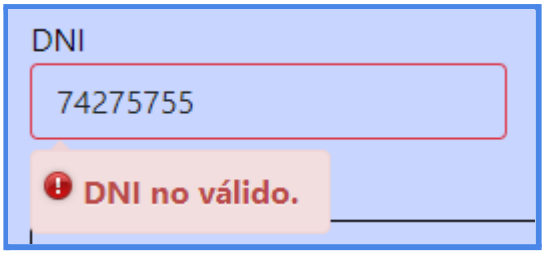
```
//variables personalizados
$gris1:    rgb(34, 47, 62);
$gris2:    rgb(131, 149, 167);
$gris3:    rgb(200, 214, 260);

$morado1:  rgb(52, 31, 151);
$morado2:  rgb(95, 39, 205);
$morado3:  rgb(165, 94, 234);

$azul1:  rgb(116, 185, 255);
$azul2:  rgb(9, 132, 227);
$azul3:  rgb(30, 55, 153);
```



```
$primary:       $azul1 !default;
$secondary:     $gray-600 !default;
$success:       $azul2 !default;
$info:          $cyan !default;
$warning:       $yellow !default;
$danger:        $red !default;
$light:         $morado3  !default;
$dark:          $gris3!default;
```

Notify

Notify.js es un complemento jQuery para proporcionar notificaciones simples pero totalmente personalizables. Esta librería la hemos utilizado para mostrar notificaciones a la hora de crear nuevas tareas.



JQuery

jQuery es una biblioteca de JavaScript rápida, pequeña y rica en funciones. Hace que cosas como el recorrido y la manipulación de documentos HTML, el manejo de eventos, la animación y Ajax sean mucho más simples con una API fácil de usar que funciona en una multitud de navegadores. Con una combinación de versatilidad y extensibilidad, jQuery ha cambiado la forma en que millones de personas escriben JavaScript.

Por ejemplo, añadir un evento a un elemento es tan sencillo como:

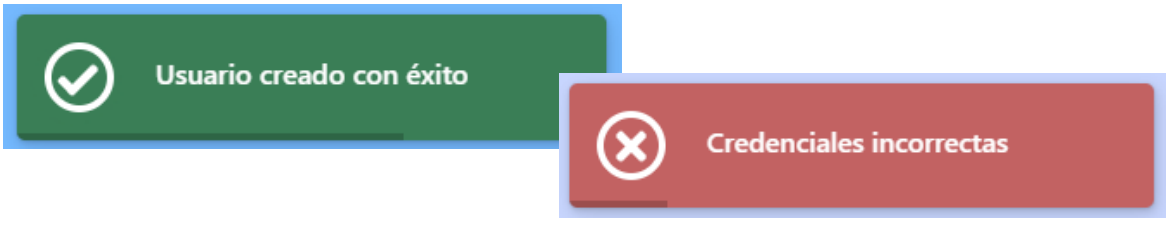
```
// Listeneres modificaciones filtros
$('#filtro-num_ref').on('keyup', evt => {
    filtro_numref = evt.target.value;

    clearTimeout(obtenerDatosAsincrono);
    obtenerDatosAsincrono = setTimeout(obtenerDatos, ms_asincronia);
    obtenerDatosAsincrono;
});
```

Al trabajar con Laravel y querer ejecutar peticiones a la API, cuando queremos proporcionar las credenciales de usuario para que la petición sea una petición válida debemos proporcionar el token.

```
$.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': jQuery('meta[name="csrf-token"]').attr('content')
    }
});
```

SweetAlert2

SweetAlert2 es una librería de JavaScript para hacer alerts más bonitos y de manera responsiva. En nuestro caso hemos utilizado principalmente la función toast, pudiendo sacar alertas pequeñas en la parte superior derecha de la pantalla que duran entre 2 y 3 segundos, dando información al usuario sin ser molestas o pesadas.



BIBLIOGRAFÍA

Librerías / Paquetes utilizados

- FakerPHP
  - Enlace packagist
  - Documentación oficial

- Laravel UI Bootstrap + Auth

Documentación que resultó muy útil

- Laravel 8 Docs
  - Primary Keys personalizadas (tipo string y que no se llamen id)
  - Eloquent: Relaciones
  - Redirecciones

- Ficheros
  - Laravel 8 Descargar ficheros de la carpeta public
  - Laravel 8 Subir ficheros al almacenamiento

- Exportar Excel y CSV (Maatwebsite)

- Laravel 8 enviar emails (gmail)

- Seguridad
  - Laravel 8 HTTPS en Producción (al cargar una ruta devuelve enlace HTTPS)
  - Laravel 8 redirigir HTTP a HTTPS en Producción (siempre que entra en HTTP redirige a HTTPS) (en Heroku produce bucles infinitos, no se ha implementado)

- Web Components en ficheros JS (guía en español y completa)

- Librerías y herramientas
  - JQuery
  - Notify
  - SweetAlert2
  - Highcharts
  - Bootstrap
  - TypeScript

Metodologías y decisiones

- ¿Por qué no hemos usado enums en las migraciones de Laravel?
- Metodología de trabajo Git: Git Flow

También puedes consultar la bibliografía en nuestro repositorio GitHub accediendo a este fichero con formato MarkDown.