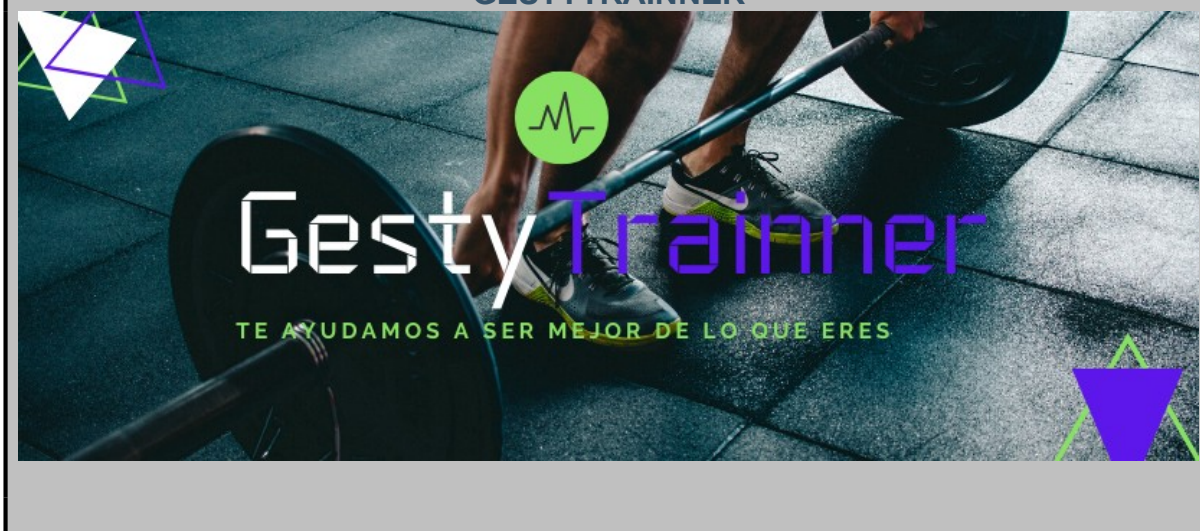


MEMORIA FINAL DE PROYECTO

WEB PARA GYMNASIOS GESTYTRAINER



CICLO FORMATIVO DE GRADO SUPERIOR

DESARROLLO DE APLICACIONES WEB

AUTORES

JAVIER RIQUELME CERDÁ

TUTOR

JESÚS VIVES CÉSPEDES

AGRADECIMIENTOS

Ha sido un período de aprendizaje intenso, no solo en el ámbito académico, también a nivel personal. Realizar este proyecto ha tenido un gran impacto en mí y es por eso que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Primero de todo, me gustaría agradecer a todos los profesores que me han impartido clase durante estos dos años por su interés y dedicación. Me habéis apoyado enormemente y siempre habéis estado ahí para ayudarme cuando lo necesitaba. Particularmente me gustaría nombrar a mi tutor de proyecto, Jesús Vives Céspedes. Me gustaría agradecerle tu cooperación y darte las gracias por todos los consejos que me has dado durante la realización de este proyecto.

Definitivamente me habéis brindado todas las herramientas necesarias para completar mi trabajo de fin de grado satisfactoriamente.

INDICE

1	INTRODUCCIÓN.....	4
2	INTRODUCCIÓN EN INGLÉS.....	5
3	ALCANCE DEL PROYECTO.....	6
4	ESTUDIO DE VIABILIDAD.....	7
4.1	Estado actual del sistema.....	7
4.2	Requisitos del cliente.....	7
4.3	Posibles soluciones.....	7
4.4	Solución elegida.....	7
4.5	Planificación temporal de las tareas del proyecto [nuevo proyecto].....	8
4.6	Planificación de los recursos a utilizar.....	8
5	ANÁLISIS.....	9
5.1	Requisitos funcionales.....	9
5.2	Requisitos no funcionales.....	9
6	DISEÑO.....	10
6.1	Estructura de la aplicación.....	10
6.2	Componentes del sistema.....	11
6.3	Arquitectura de la red.....	11
6.4	Herramientas.....	11
7	IMPLEMENTACIÓN.....	12
7.1	Entorno de implementación.....	12
7.2	Tablas creadas.....	13
7.3	Carga de datos.....	14
7.4	Ficheros de configuración actualizados.....	14
7.5	Configuraciones realizadas en la aplicación web.....	15
7.6	Implentaciones de código realizadas.....	17
8	PRUEBAS.....	19
8.1	Casos de pruebas.....	19
9	EXPLOTACIÓN.....	21
9.1	Planificación.....	21
9.2	Preparación para el cambio.....	21
9.3	Plan de formación.....	21
9.4	Pruebas de implantación.....	22
10	DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN.....	23
11	CONCLUSIONES.....	25
12	FUENTES.....	26

1 INTRODUCCIÓN

El principal objetivo de este proyecto es la implantación de la aplicación web “GestyTrainer” que consiste por una parte, en la gestión, organización y publicación de clases dirigidas para los monitores de los gimnasios, así como la gestión de los alumnos. Y por la parte de los usuarios facilitar la posibilidad de apuntarse a las clases dirigidas de su gimnasio, así como la posibilidad de comentar en un blog su opinión sobre la clase y de esta manera compartir sus experiencias con los demás usuarios. Además de intentar establecer una comunicación más directa entre alumnos y monitores a través de la aplicación web. Para su realización ha sido necesario conseguir los siguientes objetivos secundarios: análisis de las necesidades de los usuarios y los monitores, el estudio de las plataformas existentes en el mercado, la propuesta de una solución y el estudio de su viabilidad y la evaluación de la solución propuesta.

La intención de la aplicación “GestyTrainer” es ayudar a los clubes deportivos, gimnasios y otras instalaciones deportivas a gestionar de una forma más eficaz, fácil y rápida todas sus clases dirigidas independientemente de la modalidad o disciplina deportiva de la clase dirigida. Ya que la aplicación web dispone de un apartado de categorías, donde puedes crear y gestionar todo tipo de categorías en función de las necesidades de cada gimnasio.

Para la implantación de la aplicación web no es necesario la instalación de ningún software, solo una conexión a internet para poder acceder a ella. También se ha tenido en cuenta el uso de software “open source” que consta de un soporte empresarial y que está en constante evolución, por si en un futuro surge la necesidad de implementar alguna mejora en la aplicación web. De esta manera se ha conseguido abaratar los costes de producción.

2 INTRODUCCIÓN EN INGLÉS

The main objective of this project is the implementation of the “GestyTrainer” web application, which consists, on the one hand, of the management, organization and publication of classes for monitors of the gyms, as well as the management of students. And on the part of the users, facilitate the possibility of signing up for the directed classes of their gym, as well as the possibility of commenting on a blog about their opinion about the class and in this way share their experiences with other users. In addition to trying to establish more direct communication between students and monitors through the web application. For its realization it has been necessary to achieve the following secondary objectives: analysis of the needs of users and monitors, the study of existing platforms in the market, the proposal of a solution and the study of its feasibility and the evaluation of the solution proposal.

The intention of the “GestyTrainer” application is to help sports clubs, gyms and other sports facilities to manage all their directed classes more efficiently, easily and quickly, regardless of the sport modality or discipline of the directed class. Since the web application has a category section, where you can create and manage all kinds of categories depending on the needs of each gym.

For the implementation of the web application it is not necessary to install any software, just an internet connection to access it. The use of “open source” software that consists of business support and that is constantly evolving has also been taken into account, in case the need to implement any improvement in the web application arises in the future. In this way, production costs have been reduced.

3 ALCANCE DEL PROYECTO

El objetivo principal del proyecto web es que cada vez más gimnasios y usuarios utilicen la aplicación para apuntarse y gestionar sus clases dirigidas. Además podemos tener un registro informático de nuestras clases dirigidas y así poder saber que clases pueden ir mejor en función del tipo de usuarios que asisten a nuestro gimnasio.

El desarrollo de éste proyecto se llevará a cabo en varias fases: estudio de viabilidad, análisis, diseño, implementación y pruebas, y explotación o ejecución. A continuación se detallan las actividades/tareas/procedimientos de cada una de estas fases.

En el estudio de viabilidad se va a explicar como la aplicación da solución a la forma de apuntarse a las clases dirigidas. Así como la planificación que se ha llevado a cabo para implementar la aplicación.

En el análisis se explicarán algunas de las partes funcionales y no funcionales de la aplicación, que tienen más relevancia dentro de la aplicación.

En el diseño se explicará el motivo de la elección del diseño de la aplicación tanto en la parte administrativa, como en la parte cliente.

En el estudio de la implementación explicaré y pondré algún ejemplo de las partes más importantes de la implementación de la aplicación, como son la creación de modelos, controladores y vistas en Laravel. También expondré algunos casos de prueba.

En la fase de explotación explicaré como voy a implantar la aplicación “GestyTrainer” en un gimnasio, así como la idea de extender el uso de la aplicación progresivamente a otros gimnasios.

4 ESTUDIO DE VIABILIDAD

4.1 Estado actual del sistema

Actualmente para apuntarte a una clase dirigida, debes ir al gimnasio o centro deportivo y sacar un ticket en recepción para la clase dirigida que quieres realizar. Esto quiere decir que no se lleva un registro informático de la gente que asiste a cada una de las clases dirigidas que se imparten en los gimnasios o centros deportivos.

4.2 Requisitos del cliente

El cliente quiere una aplicación web para que los monitores puedan publicar sus clases y que los usuarios tengan mayor facilidad para apuntarse a las clases dirigidas de su gimnasio o centro deportivo. Además de poder ver y gestionar las clases y los usuarios que asisten a las mismas.

4.3 Posibles soluciones

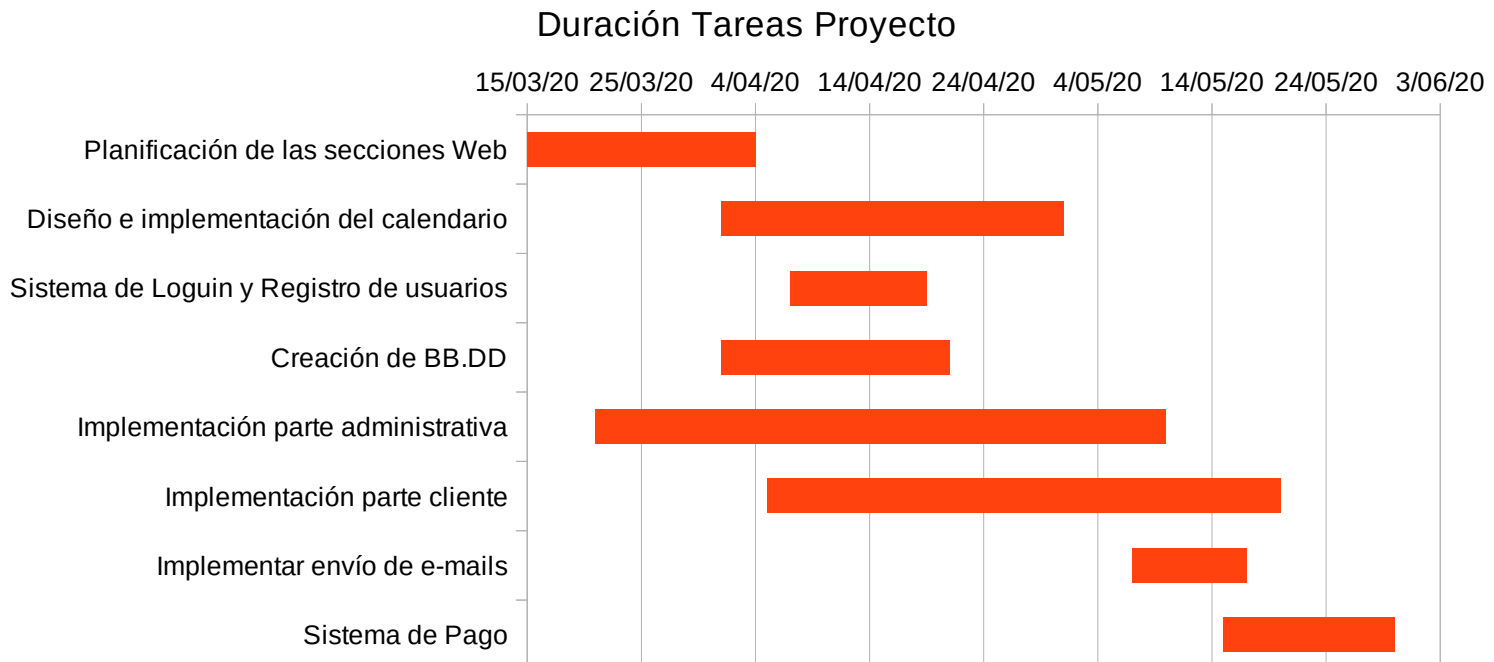
Tras una rigurosa investigación puedo confirmar, que a día de hoy son muy pocas o casi inexistentes las aplicaciones web en el mercado con este tipo de características en el sector de las clases dirigidas.

Por esa razón la aplicación web “GestyTrainer” debe ofrecer la posibilidad a los usuarios de apuntarse a las clases dirigidas, desde cualquier lugar, realizando el pago desde la misma aplicación, sin necesidad de tener que ir expresamente al gimnasio para sacar un ticket.

4.4 Solución elegida

Ante la escasa existencia de aplicaciones web con este tipo de características en el sector de las clases dirigidas, se ha tomado como solución la implantación de la aplicación “GestyTrainer” para la gestión de los usuarios y de las clases dirigidas.

4.5 Planificación temporal de las tareas del proyecto [nuevo proyecto]



A continuación se muestra un diagrama de Gantt con los tramos de tiempo dedicados a la realización de las diferentes partes más importantes del proyecto. Cabe destacar que este proyecto ha sido realizado en su integridad por una única persona.

4.6 Planificación de los recursos a utilizar

Al ser una aplicación con características novedosas para los monitores del gimnasio o centro deportivo, es necesario realizar unos cursos para formar a los monitores. De esta manera la implantación de la aplicación será más sencilla.

5 ANÁLISIS

5.1 Requisitos funcionales

En la parte de administración la aplicación cuenta con un calendario donde los monitores pueden organizar sus clases dirigidas. El calendario es el mismo para todos los monitores, debido a que deben coordinarse todas las clases dirigidas de un mismo gimnasio. De esta forma evitamos que dos monitores diferentes puedan poner a la misma hora y en la misma sala dos clases diferentes.

Además en la parte de cliente cuenta con una plataforma de pago donde los usuarios pueden apuntarse y realizar el pago de sus clases dirigidas. Lo que se pretende conseguir con la plataforma de pago es proteger los datos bancarios introducidos por el usuario. Como medida de seguridad, si el usuario ya está apuntado a una clase e intenta volver a apuntarse, la aplicación no se lo permitirá y le avisará mediante un mensaje que ya está apuntado a dicha clase.

5.2 Requisitos no funcionales

El diseño de la aplicación es bastante sencillo e intuitivo, ya que cada una de los apartados de la aplicación hay una breve introducción del funcionamiento o intención de dicho apartado. Pero si en alguno de los casos la explicación fuese insuficiente, la aplicación cuenta con unas validaciones mínimas en cada uno de los campos que tiene que cumplir el usuario para rellenarlos. Así como en las acciones que puede realizar dentro de la aplicación, como apuntarse a clases que ya esta apuntado. En estos casos el usuario siempre recibirá un feedback por parte de la aplicación, en forma de mensaje indicándole donde ha sido el error y como debería proceder para su validación.

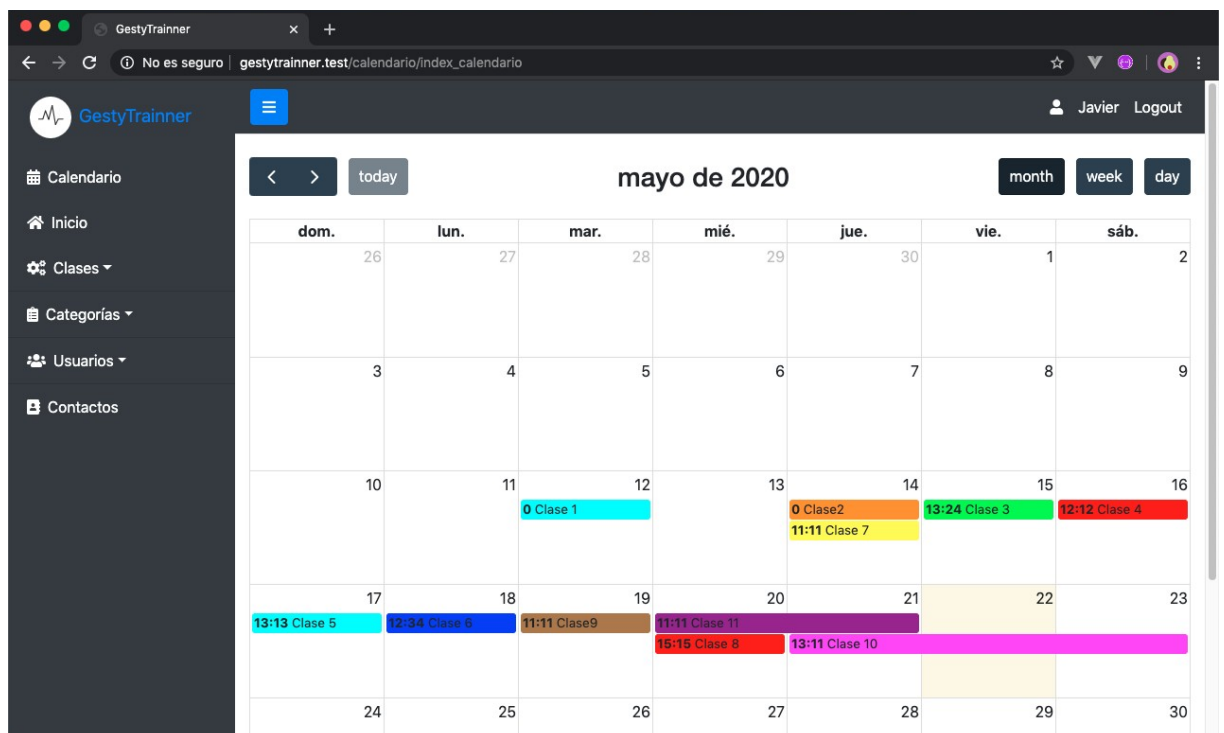
Además la aplicación cuenta con un diseño atractivo para el usuario centrado sobre todo en la visualización de las clases, porque al final lo que se pretende en la aplicación es presentar la clase dirigida de forma atractiva, para que el usuario quiera apuntarse.

6 DISEÑO

6.1 Estructura de la aplicación

La aplicaicón web “GestyTrainer” se divide en dos apartados principales, por un lado la parte administrativa que es donde los monitores llevan a cabo toda la gestión de sus clases dirigidas mediante un calendario y sus usuarios.

El calendario está diseñado para ver las clases dirigidas por mes, semana o día. Además tiene la posibilidad de crear, modificar o eliminar las clases dirigidas en el mismo calendario, ya que cuando haces click en uno de los días sale una ventana con un formulario donde puedes introducir todos los datos necesarios para crear la clase. O bien si haces click en una clase en la misma ventana salen los datos de la clase que se ha seleccionado y puedes modificarla.



Por otro lado está la parte del cliente, que es donde los usuarios ven los posts de las clases y dentro de cada post pueden realizar comentarios sobre la clase. También pueden apuntarse a cualquier clase dirigida y ver a que clases están apuntados. Además hay un apartado de “contactos”, donde el usuario puede enviar mensajes a los monitores de las clases dirigidas.

6.2 Componentes del sistema

Para la realización de la aplicación se han utilizado los siguientes componentes:

- MariaDB para la Base de Datos.
- Valet como entorno de desarrollo.
- Nginx como servidor.
- Laravel como framework.
- Stripe como pasarela de pago.
- FullCalendar como librería para el calendario.
- MailTrap para el envío de e-mails.

6.3 Arquitectura de la red

La aplicación estará alojada en un servidor web el cual se accede mediante una conexión a internet.

6.4 Herramientas

MariaDB, Valet, Nginx, Larabel, Stripe, FullCalendar.

7 IMPLEMENTACIÓN

7.1 Entorno de implementación

Para implementar la aplicación web “GestyTrainer” se han utilizado diferentes lenguajes de programación, que pasaré a explicar a continuación.

Para definir el contenido de las páginas (DOM) he utilizado HTML, porque es un lenguaje de marcado que se compone de etiquetas o tags con las cuales se puede expresar las partes de un documento (header, body, footer...).

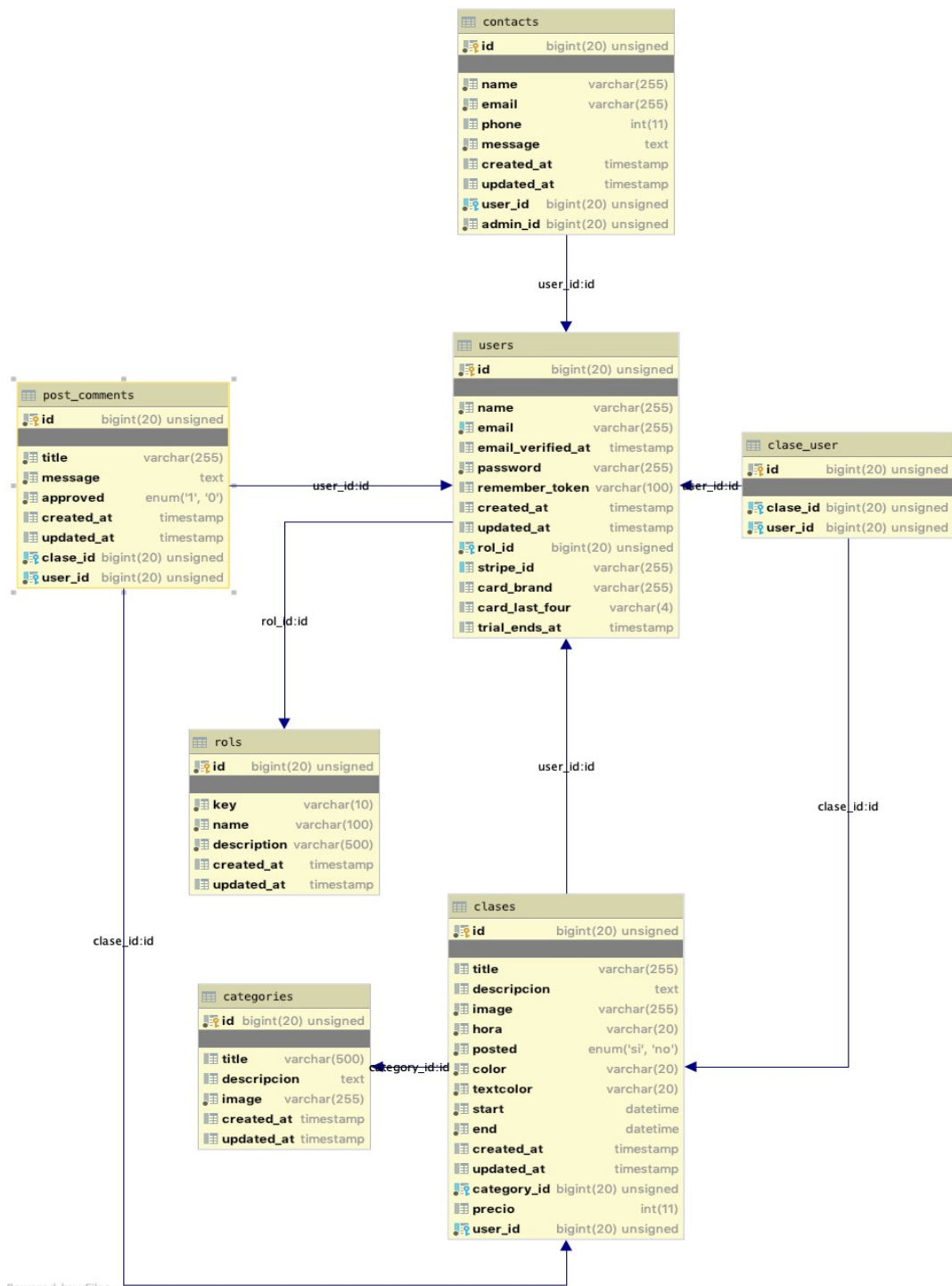
He utilizado CSS y Bootstrap para darle el diseño visual de las páginas, porque son lenguajes creados para controlar el aspecto de los documentos HTML. Y me permiten separar el contenido de la presentación, haciendo más fácil modificar el estilo de las páginas en un futuro.

Con el fin de trabajar con los datos, es decir enviar y recoger información de las páginas, he utilizado PHP. Y para la creación de eventos, como la creación de las clases en el calendario y la creación del método de pago al apuntarse en las clases, he utilizado JavaScript. También he empleado AJAX para realizar peticiones asincronas y evitar que se recargue la página entera de nuevo.

Lo que me ha permitido unir todo este tipo de lenguajes y ha hecho posible que trabajen como uno sólo ha sido el uso del framework Laravel. Que me ha facilitado la organización de los archivos a través de su Modelo Vista Controlador (MVC).

7.2 Tablas creadas

Como base de datos se ha usado MariaDB usando el tipo InnoDB. A continuación se muestra el modelo entidad relación de la base de datos usado:



Powered by yFiles

7.3 Carga de datos

Los datos mínimos necesarios cargar para que la aplicación funcione son un usuario administrador y otro usuario regular, el resto de datos de la aplicación depende de cada gimnasio. Porque son los usuarios los que introducen los datos de las clases, crean las categorías, hacen comentarios y se apuntan a las clases.

7.4 Ficheros de configuración actualizados

Al realizar la aplicación con el framework Laravel, este tiene un archivo llamado “.env” en el cual se especifican todas las variables de entorno de la aplicación. A continuación voy a explicar las más importantes.

Las opciones “APP_” sirven para poner el nombre de tu aplicación, configurar el entorno del proyecto que por defecto esta como local y puedes cambiarlo a producción cuando sea necesario. Cuando creas un proyecto por la consola de comandos, Laravel te asigna una “APP_KEY”. En esta opción, la “KEY” se usa para proteger tu proyecto y es muy importante para la seguridad del mismo. Para saber porque esta dando error el proyecto, hay que tener la opción “APP_DEBUG” como true, ya que sirve para mostrar errores. Y en la opción “APP_URL” se debe de poner la URL del servidor o donde estás trabajando con Laravel.

```
1 APP_NAME=GestyTrainer
2 APP_ENV=local
3 APP_KEY=base64:e/gVJ00cZtVwNYgJoPDdCxLTEjTxrakxoUzby0ND4bw=
4 APP_DEBUG=true
5 APP_URL=http://localhost
```

Las opciones “DB_” sirven para poner el tipo de base de datos con la que vas a trabajar para tu proyecto, en mi caso es mysql. Poner el servidor en el que esta alojado tu proyecto, puedes colocar localhost o 127.0.0.1. El puerto en el cual se comunica la base de datos y para poner el nombre de tu base de datos.

```
9 DB_CONNECTION=mysql
10 DB_HOST=localhost
11 DB_PORT=3306
12 DB_DATABASE=gestytrainer
13 DB_USERNAME="root"
14 DB_PASSWORD="secretroot"
```

7.5 Configuraciones realizadas en la aplicación web

Para subir la aplicación "GestyTrainer" al servidor donde se va a alojar fue necesario realizar algunas configuraciones para pasar de un entorno de desarrollo a uno de producción.

Para comenzar, hay que dirigirse al archivo .env y modificar estas dos variables:

```
APP_ENV=production
APP_DEBUG=false
```

Al estar utilizando webpack mix, abro la consola, en la raíz de nuestro proyecto y ejecuto el siguiente comando:

```
npm run production
```

una vez hecho esto también hay que ejecutar el siguiente comando:

```
composer dumpautoload
```

Tras esto, ya está todo listo para comenzar con la subida de nuestra aplicación.

Hay que "dividir" las carpetas del proyecto en dos partes y comprimir las en archivos .zip. La primera esta compuesta por todas las carpetas del proyecto a excepción de la carpeta public. Y la segunda, una vez comprimidos todos estos archivos, hay que entrar en la carpeta public y comprimirlos también.

En el administrador de archivos del cpanel, hay que crear una carpeta para subir el archivo zip creado anteriormente sin la carpeta public, en este caso se la carpeta se llama gestytrainer y dentro de ésta, hay que hacer click a "cargar" donde se arrastra el archivo zip. Una vez subido nuestro zip, hay que hacer click derecho sobre el archivo y seleccionaremos la opción "extract" para extraer el archivo comprimido.

Ahora hay que hacer el mismo proceso con el archivo zip de la carpeta public, pero esta vez no hay que crear ninguna carpeta, sino que hay que ir a la carpeta "public_html" del panel de administración, y ahí se sube el archivo zip para posteriormente descomprimirlo.

El siguiente paso es modificar nuestro archivo index.php para indicarle donde se encuentra la aplicación gestytrainer, donde están las rutas, vistas, controladores, etc.

Hay que hacer click derecho en el archivo index.php, luego pulsar en "edit" y modificar las siguientes líneas de código:

```
require __DIR__.'/../gestytrainer/vendor/autoload.php';
```

```
$app = require_once __DIR__.'/../gestytrainer/bootstrap/app.php';
```

Una vez modificadas esas rutas, el archivo index.php ya está apuntando hacia la carpeta de la aplicación.

Ahora nos hay que ir a la carpeta gestytrainer, a la siguiente ruta dentro de la carpeta "app/providers" y editar el archivo "AppServiceProvider.php" para editar su función register, añadiendo el siguiente código:

```
public function register(){  
    $this->app->bind('path.public',function(){  
        return'/home/rickzere/public_html';  
    });  
}
```


7.6 Implementaciones de código realizadas

en la implementación de la aplicación web “GestyTrainer” se ha utilizado el framework Laravel. Este framework propone a la hora de desarrollar usar “routes with Closures”, en lugar del MVC tradicional con el objetivo de hacer el código más claro. Aún así permite el uso de MVC tradicional. A continuación voy a explicar como se crea un modelo, una vista y un controlador en Laravel.

Laravel incluye un sistema de mapeo de datos relacional llamado Eloquent ORM(mapeo de objeto relacional) que facilita la creación de modelos. Es opcional el uso de Eloquent, porque también dispone de otros recursos que facilitan la interacción con los datos, como la creación de modelos.

La forma de crear Modelos en Laravel usando Eloquent ORM, es:

```
<?php
namespace App;

use App\Category;
use App\PostComment;
use App\User;
use Illuminate\Database\Eloquent\Model;

class Clase extends Model
{
    protected $fillable=['title', 'category_id', 'descripcion', 'color', 'textcolor', 'start', 'end', 'hora', 'posted', 'image', 'precio', 'user_id'];

    public function category(){
        return $this->belongsTo(Category::class);
    }

    public function comments(){
        return $this->hasMany(PostComment::class);
    }

    public function user(){
        return $this->belongsTo(User::class);
    }

    public function users(){
        return $this->belongsToMany(User::class);
    }
}
```

También puedes crearlo con el siguiente comando de Artisan:

```
php artisan make:model Clase
```

Laravel incluye un sistema de procesamiento de plantillas llamado Blade. Este sistema de plantillas favorece un código mucho más limpio en las vistas, además de incluir un sistema de caché que lo hace mucho más rápido. El sistema Blade de Laravel, permite una sintaxis mucho más reducida en su escritura.

Un ejemplo de una Vista con plantilla Blade, sería:

```
@csrf

<div class="form-group">
    <label for="title">Título</label>
    <input class="form-control" type="text" name="title" id="title" value="{{old('title', $clase->title)}}">
    @error('title')
    <small class="text-danger">{{ $message }}</small>
    @enderror
</div>
<div class="form-row">
    <div class="form-group col-3">
        <label for="category_id">Categoría</label>
        <select class="form-control" name="category_id" id="category_id">
            @foreach ($categories as $title => $id)
                <option {{ $clase->category_id == $id ? 'selected="selected"' : '' }} value="{{ $id }}">{{ $title }}</option>
            @endforeach
        </select>
    </div>
```

Los controladores contienen la lógica de la aplicación y permiten organizar el código en clases sin tener que escribirlo todo en las rutas. Todos los controladores deben extenderse de la clase Controller.

Un ejemplo de un Controlador en Laravel, sería:

```
class ClaseController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware(['auth', 'rol.admin']);
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $clases = Clase::with('category')->orderBy('start', request('start', 'DESC'));

        if ($request->has('search')) {
            $clases = $clases->where('title', 'like', '%' . request('search') . '%');
        }

        $clases = $clases->paginate(5);
        return view('dashboard.clase.index', ['clases' => $clases]);
    }
}
```

Estos pueden ser llamados en las rutas usando la siguiente sintaxis:

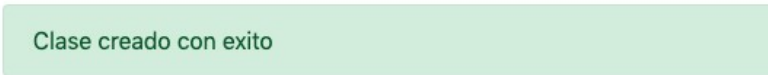
```
Route::get('dashboard/clase/inicio', 'dashboard\ClaseController@inicio')->name('inicio');
```

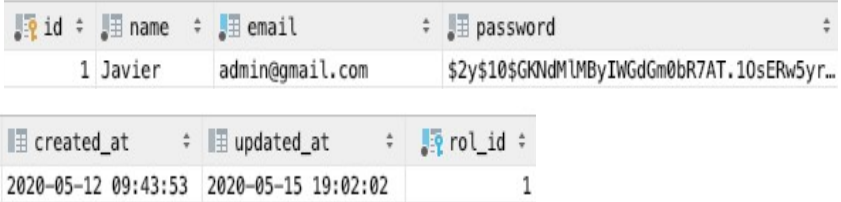
8 PRUEBAS

Son muchas las pruebas que pueden realizarse en un proyecto, para eliminar los posibles errores y garantizar su correcto funcionamiento. Los casos de prueba establecen las condiciones/variables que permitirán determinar si los requisitos establecidos se cumplen o no.

A continuación se detallan algunos de los casos de prueba que se ejecutarán para comprobar la correcta construcción de este proyecto.

8.1 Casos de pruebas

Creación de una clase nueva	
Fecha/autor/versión	12-05-20/Javier/versión 2
Descripción	Crear una clase nueva desde el módulo administrativo de la aplicación.
Condiciones de ejecución	-Rellenar correctamente los campos del formulario "Nueva Clase". -Al hacer click en el botón enviar, debe crearse la nueva clase.
Entrada	Título, categoría, precio, hora, comienzo, final, color, color de texto, posteo, descripción, imagen.
Resultado esperado	Tiene que salir un mensaje con el texto "Clase creada con éxito".
Resultado obtenido	
Evaluación	Como el resultado esperado y obtenido han sido satisfactorios, pasamos a la siguiente fase de la implementación. Pero en caso de no haber coincidido ambos resultados, debería revisar la vista "master.-blade.php". Porque es la vista que se encarga de renderizar dicho mensaje.

Insertar en la BB.DD un usuario	
Fecha/autor/versión	25-05-20/Javier/versión 4
Descripción	Cuando el monitor crea un nuevo usuario la aplicación debe insertar en la BB.DD los datos de dicho usuario.
Condiciones de ejecución	-Recoger los datos del usuario. -Insertar los datos del usuario en la BB.DD.
Entrada	Id, name, email, rol, password, created_at, updated_at.
Resultado esperado	Los datos del usuario insertados en el formulario deben aparecer en tabla “users” de la BB.DD
Resultado obtenido	
Evaluación	Como el resultado esperado y obtenido han sido satisfactorios, pasamos a la siguiente fase de la implementación. En caso de no haber coincidido ambos resultados, debería revisar el modelo “User” y el controlador “UserController” que son los encargados de insertar en la BB.DD los datos del nuevo usuario creado.

9 EXPLOTACIÓN

La implantación es la fase más crítica del proyecto ya que el sistema entra en producción, es decir opera en un entorno real, con usuarios reales.

9.1 Planificación

Primero se implantaría en un gimnasio de la provincia para probar la aplicación en un ámbito real. Se haría una demostración de la aplicación con usuarios ficticios, para posteriormente dar de alta a los usuarios reales en la aplicación, tanto a monitores como alumnos. Además se realizaría un seguimiento del uso de la aplicación para poder aclarar dudas sobre la misma a los monitores. Y también les preguntaría que otras necesidades podría cubrir la aplicación, con la intención de mejorar y personalizar la aplicación.

9.2 Preparación para el cambio

Con el fin de preparar a los monitores se les proporcionaría una serie de cursos sobre el uso de la parte administrativa de la aplicación y se les entregaría un manual de usuario. También sería conveniente que los monitores promocionaran la aplicación entre sus alumnos y dieran de alta a alguno de los alumnos. De esta se daría a conocer la aplicación antes entre los usuarios del gimnasio.

9.3 Plan de formación

En función del tipo de suscripción de la aplicación que elija el gimnasio, se ofertará más cursillos, mayor seguimiento y la posibilidad de personalizar o agregar mejores en función de las necesidades del gimnasio.

En el anexo explico detalladamente tanto la parte administrativa, como la parte cliente de la aplicación en el apartado del “Manual del usuario”.

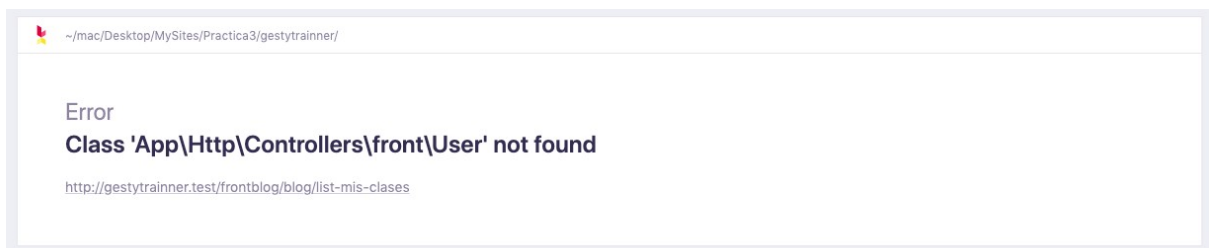
9.4 Pruebas de implantación

Para realizar las pruebas de usabilidad y navegabilidad, primero se implantó en un gimnasio, con la idea de expandirlo a otros gimnasios una vez realizadas las pruebas. Las pruebas de usabilidad sirven para asegurar que la interfaz soporta cada tipo de usuario. Las pruebas de navegabilidad sirven para asegurar que toda la sintaxis y la semántica de navegación se ejecutan para descubrir cualquier error de navegación (por ejemplo, vínculos muertos, inadecuados y erróneos).

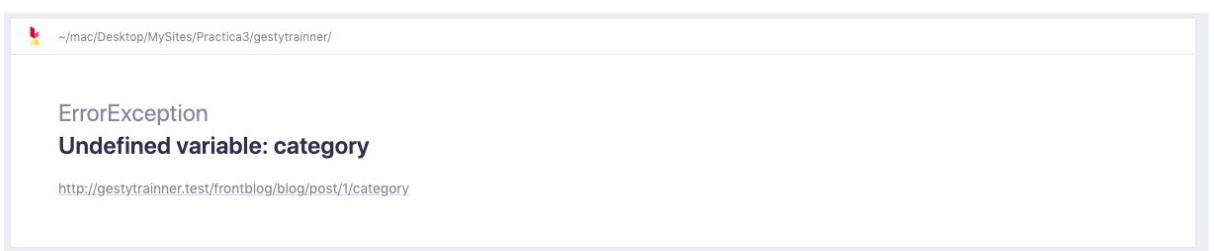
10 DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN

A lo largo del ciclo de vida del proyecto se producirán cambios e incidencias que deberán controlarse y registrarse. Para realizar la corrección de errores he utilizado el debugger que tiene Laravel por defecto, como he explicado en el apartado “7.4 Ficheros de configuración actualizados”. A continuación mostraré algunos de los errores más comunes durante el desarrollo de la aplicación.

Este tipo de error puede ocurrir por varios motivos, como por ejemplo cuando no importas el modelo en el controlador. Cuando la ruta del modelo está mal escrita y el controlador no la encuentra. O simplemente no instancias correctamente el modelo dentro de la función del controlador.



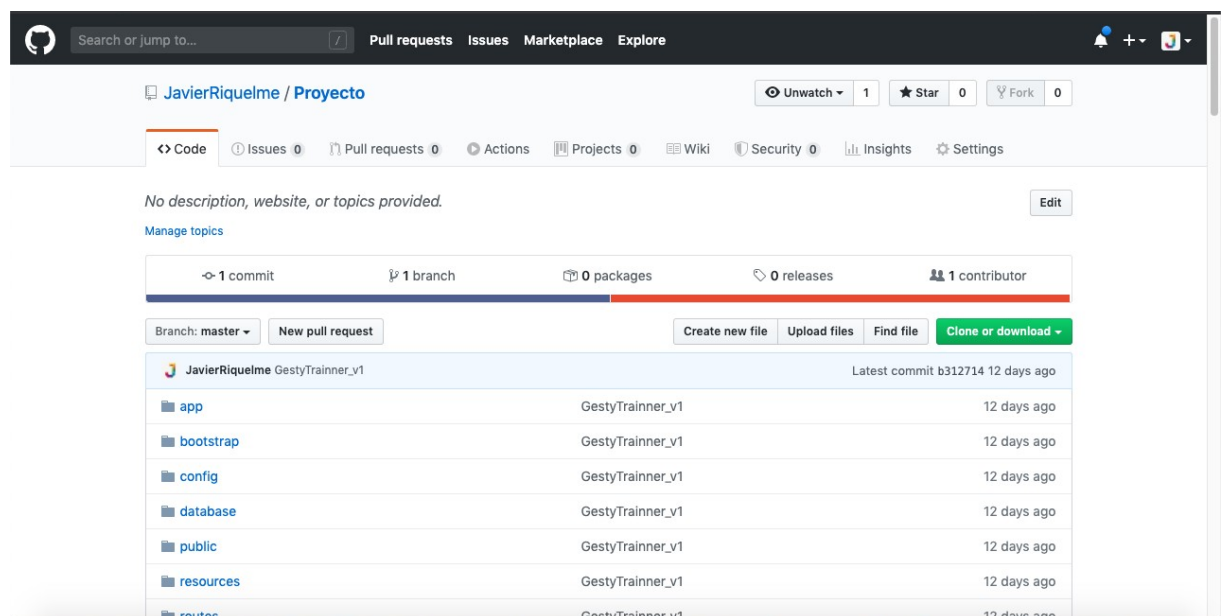
También se produce el siguiente error cuando no le pasamos los parámetros requeridos a la función del controlador. O falta por pasar a la función del controlador algún parámetro que no se le ha pasado correctamente.



Además de este debugger que trae Laravel por defecto, existen otros como “Laravel Debugbar” que son muy interesantes. Este debugger nos permite conocer rápidamente información de todo lo que se está ejecutando Laravel como, por ejemplo el nombre de la ruta asociada, las consultas SQL ejecutadas, memoria utilizada, tiempo de respuesta, etc. Toda esta información es útil para tener control total de la aplicación y a su vez corregir posibles errores e inconvenientes de rendimiento.

Como sistema de control de versiones he utilizado GitHub, porque me permite crear repositorios donde puedo subir el código de la aplicación y disponer de él en cualquier sitio. Además si por alguna casualidad se perdieran o dañaran los archivos del proyecto, tienes la tranquilidad de tener la última versión de tu código en el repositorio de GitHub.

En este caso en particular solo existe la rama master, porque el proyecto ha sido realizado por una única persona, pero GitHub te permite crear diferentes ramas y subir el código realizado por los diferentes desarrolladores, en caso de trabajar en un equipo.



11 CONCLUSIONES

Tras proponer el proyecto “GestyTrainer” y elegir las herramientas necesarias para su posterior desarrollo. Comencé por implementar los módulos de login y registro de usuarios, porque son los dos apartados principales con los que se inicia la aplicación.

Seguidamente realicé la implementación de la parte de administración, más concretamente el módulo del calendario utilizando “fullCalendar”. A continuación realicé los módulos de clases, categorías, usuarios y contactos. El módulo de inicio lo dejé para el final porque necesita de los módulos anteriores para poder implementarse.

Una vez terminado la parte de administración, comencé a implementar la parte cliente, en particular los módulos blog, categorías y contacto. Además dentro del módulo blog también implementé el módulo de comentarios. Una vez implementados los módulos anteriores, realicé la implementación de la pasarela de pago para que los alumnos se pudieran apuntar a las clases.

Por último, implementé los módulos sobre mí y contacto, ya que son los más independientes de la aplicación. También implementé el envío de e-mails en los módulos de pago y registro.

Personalmente, ha sido una experiencia muy buena, donde he podido aplicar todos los conocimientos adquiridos durante estos años. También me ha dado la oportunidad de realizar un gran reto, como es el desarrollo de una aplicación web. Pasando por todas sus fases como he explicado en esta memoria de proyecto, desde que es solo una simple idea, hasta que se sube a internet.

12 FUENTES

DAW

- <https://laravel.com/docs/7.x>
- <https://fullcalendar.io/docs>
- <https://stripe.com/docs>
- <https://mailtrap.docs.apiary.io/#reference>
- <https://es.stackoverflow.com/>