

Departamento de Informática
Instituto Caparrella
Lérida

Proyecto de desarrollo de aplicaciones web

[Flourish & Blotts]

Alumno: Brandon Sotomayor Jesus

PROYECTO DE DESARROLLO DE APLICACIONES WEB

ENTREGADO AL DEPARTAMENTO DE INFORMÁTICA DE INS CAPARRELLA
PARA OPTAR AL TÍTULO DE
TÉCNICO DE GRADO SUPERIOR DE DESARROLLO DE APLICACIONES WEB

CFGs D.A.W.

CURS 2021-2022

Resumen

Esta aplicación se hizo con el fin de facilitar y agilizar las gestiones de una biblioteca.

Todos podrán mirar la misma parte publica, datos de la biblioteca, el catálogo y los horarios.

En la parte privada, una vez han iniciado sesión, dependiendo de su rol, tendrán las siguientes opciones:

- El administrador es el encargado de dar de alta (activar) o dar de baja (desactivar) a uno o varios responsables de la biblioteca, también puedo agregarlos.
- El responsable de la biblioteca se encarga de agregar a los 3 tipos de usuarios, profesor, estudiante, pas. Puede hacerlo de forma individual o conjunta. También se encargará de agregar libros de manera individual o conjunta, de la misma manera que podrá aceptar las reservas que hacen los usuarios de la biblioteca
- Los usuarios profesor, estudiante, pas, podrán reservar libros si hay ejemplares del mismo. Podrán opinar.

Todos los usuarios tendrán un tablero de notificaciones

Agradecimientos

Estoy muy agradecido a cada uno de los profesores que tuve, no fue fácil llegar hasta este punto, pero sin duda sin vuestra ayuda no hubiera sido posible, gracias por estar disponibles cuando hacía falta y sobre todo por vuestra buena voluntad.

- Maria Àngels Cervero
- Marc Iborra
- Josep Maria Flix
- Salvador Angelet
- Ferran Eloy

Índice

Resumen	ii
Agradecimientos	iii
1.Propósito del documento.	2
2.Objetivos	2
2.1. Funcionalidad.....	3
2.2.Objetivos del autor/a	3
3.Estructura del documento	3
3.1. Arquitectura de la aplicación web	3
3.2.Arquitectura del MVC	4
3.2.1Modelo.....	4
3.2.2.Vista	4
3.2.3.Controlador	4
3.3.Herramientas utilizadas	5
3.3.1. Framework utilizado y características.....	5
3.3.2.Lenguajes y entorno de desarrollo	5
4.La aplicación de la biblioteca.	6
4.1. Creación de una aplicación web con codeigniter	6
4.2.Funcionamiento de una aplicación con codeigniter	6
4.3.Gestión de los datos con codeigniter	7
5.2.Planteamiento de la arquitectura del MVC	8
6.2.Análisis de los requisitos.....	8
3.1.2Requisitos de seguridad.....	9
6.2.2.Diagrama de casos d'ús.....	9
6.3.Disseny	9
1.1.1.Arquitectura MVC	10
6.3.2.Apis i crides	10
6.3.3.Ejemplos rutas, método, controlador y función	13
6.3.4.Funcionalidades de los usuarios	18

3.3.4.1.1Parte pública	18
3.2.4.3.1Parte pública	21
6.4.Implementación de la aplicación	23
6.4.2.VISTAS	23
8.2.2.Controlador	23
8.2.3.Model	23
8.2.4.Diagrames de seqüència	23
9.2.Plan de trabajo	23
9.3.Diagrama de Gantt.....	24
11.2.Lliurament del treball.....	30
https://github.com/BrandonSotomayor/DAW2-Sintesi-Sotomayor-Brandon	30
Descripció de fitxers i carpetes adjuntades	30

Introducción

En esta aplicación se pretende hacer más fácil, el uso de una biblioteca con todo lo que conlleva, gestionar usuarios, responsables, libros, ejemplares, etc.

1. *Propósito del documento.*

Aclarar la forma de trabajo, el reparto de tiempo dedicado a cada apartado, cada implementación, los conocimientos aprendidos, etc.

2. *Objetivos*

El objetivo de este proyecto es el desarrollo de una aplicación web, para ayudar a la gestión de libros de las bibliotecas.

2.1. *Funcionalidad*

Para que cada usuario pueda tener acceso a sus funcionalidades, dependiendo del rol, debe estar registrado previamente, con los datos en común, y con el correo electrónico y contraseña.

2.2. *Objetivos del autor/a*

Tener la capacidad de implementar o representar los conocimientos aprendidos, durante este curso. Poder presentar un trabajo donde todas las funciones y gestiones de la aplicación funcionen correctamente. Tanto en la parte web como en la de móvil

3. *Estructura del documento*

- Explicación del problema
- Planteamiento de solución

- Solución

En los siguientes puntos se explicará el método u organización que he seguido para desarrollar la aplicación

3.1. Arquitectura de la aplicación web

- Lo primero que hice fue diseñar la base de datos, con las respectivas entidades
- Hacer el estudio de como implementar los datos en las tablas
- Diseñar la página web con las diversas rutas
- Insertar datos en las tablas para hacer pruebas
- Pruebas con las funcionalidades de cada usuario
- Arrancar la aplicación con Docker
- Hacer las pruebas **con API**
- **Crear App**
- **Pruebas funcionalidades con el móvil**

3.2. Arquitectura del MVC

3.2.1 Modelo

- He creado un modelo específico para cada tabla
- Todas las peticiones que hace el controlador se le pide al respectivo modelo, el cual devuelve la información pedida de una tabla en concreto

3.2.2. Vista

- He creado 3 layouts para los diferentes usuarios:
 - privado_administrador

- privado_responsable
- privado_usuario
- Cada tipo de usuario tiene una subcarpeta, dentro están las vistas que corresponde al rol
- Todo lo que muestra la vista y tiene alguna funcionalidad, va dirigida al controlador pasando por un filtro, quien procesa la información recibida

3.2.3. Controlador

- El cual gestiona todas las peticiones del usuario, quien redirige para ver la información y enviar información
- He creado un controlador para las funciones en común y en las apis, para la parte pública y la parte privada de cada rol

3.3. Herramientas utilizadas

3.3.1. Framework utilizado y características

- **Codeigniter**
 - Trabaja con la estructura de MVC
 - Admite desarrollar muchas funcionalidades con **MySQL**, SQLite y PostgreSQL
 - Plantillas que facilitan la velocidad y rendimiento de la aplicación web
 - Tiene una serie grande de validaciones para formularios y datos
 - Contiene un sistema de seguridad **CSRF**
- **Angular**
 - Es una plataforma de desarrollo, construida sobre TypeScript
 - Basado en componentes para crear aplicaciones web escalables
 - El `ng-model` es un proceso que permite a los usuarios manipular elementos de la página web

- El **AppModule** proporciona el mecanismo de arranque para iniciar la aplicación
 - **Componentes**, la forma de organización de la página
 - Los **servicios** sirven para compartir información entre componentes o incluso hacer peticiones http a apis
- **Ionic**
- Permite desarrollar y desplegar aplicaciones híbridas, que funcionan en múltiples plataformas, como iOS nativo, Android, escritorio y la web
 - Emplea **Capacitor** (o Cordova) para implementar de forma nativa o se ejecuta en el navegador como una aplicación web progresiva.
 - Está construido sobre tecnologías web: **HTML**, **CSS** y **JavaScript**.
 - Se puede usar con los frameworks frontend más populares, como **Angular**

3.3.2. Lenguajes y entorno de desarrollo

- **Php**

Antes de empezar a programar, tenemos que instalar y modificar el entorno de desarrollo, descargar **XAMP** activamos los componentes de **Apache** y **MySQL**

- **Bootstrap**

- Permite crear interfaces
- Se adapta a los diferentes navegadores, ya sea web de escritorio o móviles, tablets a cualquier medida
- Permite utilizar un diseño de LESS y estándares CSS3/HTML5
- Es ligero
- En la parte del contenido, te permite una variedad de colores, tipo de fuente, tablas, lista, etc

- **Postman**

- Creación de peticiones para consultarlas con un **servicio REST**, se pueden guardar y llamarlas cuando se necesite
- Permite la creación de colecciones, la que guarda todas las peticiones que deseemos
- **Gestionar la Documentación**, genera documentación basada en las API

4. *La aplicación de la biblioteca.*

4.1. *Creación de una aplicación web con codeigniter*

Estudio previo de la base de datos

Diseñe toda la base de datos, la estructura correcta y estudie las diferentes maneras para hacer las inserciones de datos. Para cuando empiece a programar, diseñar la interfaz no tenga problemas con los datos que quiero mostrar o enseñar, la cual cosa llevaría a una reestructuración de la base de datos y posiblemente de la interfaz.

4.2. *Funcionamiento de una aplicación con codeigniter*

El funcionamiento se resume al **MVC**, el controlador renderiza al usuario a la vista que necesita ver, ya sea en la parte pública o privada.

El usuario cuando quiere hacer servir alguna funcionalidad en la parte pública, lanzará los datos al controlador, que a su vez dependiendo del pedido, el controlador se pondrá en contacto con el modelo quien atacará a la base de datos para devolver datos o el controlador solo renderizará a otra vista.

En caso que sea en la parte privada, cuando el usuario hace una petición, antes de pasar al controlador pasará por un filtro

4.3. *Gestión de los datos con codeigniter*

Para gestionar los datos CI4 tiene la característica de conectarse con MySQL, para las consultas de las tablas de una base de datos.

Dependiendo del rol de un usuario, el controlador hará ciertas peticiones a modelos comunes o individuales para una determinada funcionalidad.

5. Planteamiento del problema

Para el back-end usaré el framework de **CodeIgniter** versión 4, como lenguaje principal para el desarrollo de aplicación web y para la aplicación móvil **ionic**.

Para el front-end, dar estilos, estructura a la aplicación posicionamiento utilizaré el framework **Bootstrap-web**, para la app móvil **ionic**.

Para la creación de Api Rest, utilizaré la herramienta de **POSTMAN**

Para la visualización y revisión de datos coherentes en la base de datos utilizaré el software libre y código abierto **HeidiSQL**

Para arrancar la aplicación web sin la ayuda de la herramienta de XAMPP, utilizaré **Docker**, que tendrá instalados contenedores que cumplirán las misma funcion que el MySQL, Apache y Codeigniter.

5.1. Planteamiento de la arquitectura del MVC

- El usuario hará una reserva, pasará por el filtro de seguridad, una vez dentro el controlador gestionará su reserva y llamará al modelo para insertar su reserva, en la tabla. Una vez hecho todo este proceso será redireccionado a la página donde estaba el usuario y el estado de la reserva será pendiente
- El servidor recibirá esta reserva en su apartado de reservas, donde el podrá aceptar la reserva y posar una fecha de búsqueda. Una vez aceptada la reserva, el controlador llama al modelo, donde actualiza la información de la reserva 'fecha de busqueda'

6. Solución

6.1. *Análisis de los requisitos*

- Análisis de requisitos
 - Tener una base de datos correcta, que la relación entre campos de tablas sea correcta
 - Tener una base de **PHP** + el framework de **CodeIgniter**
 - Para el diseño visual, agregar estilos con el framework de **Bootstrap**
- Diseño de la aplicación
 - Tener los formularios + métodos correctos, con las rutas correctas, para que pueda entrar correctamente al controlador que a su vez el modelo hará peticiones a las tablas, para obtener los datos (MVC)
- Estudio previo de la base de datos, relaciones, claves primarias/foráneas
- Hacer las migraciones con el helper de javascript. Estas migraciones ayudan a borrar y crear tablas de la base de datos.
- Crear las plantillas
- Implementar la parte pública, después la privada
- Diseñar la interfaz de la aplicación. Iré por orden de administración, de superior a inferior. Comenzaré por el administrador y terminaré por los usuarios de la biblioteca
- Crear los controladores y los modelos necesarios, para la inserción, actualización y eliminación de datos

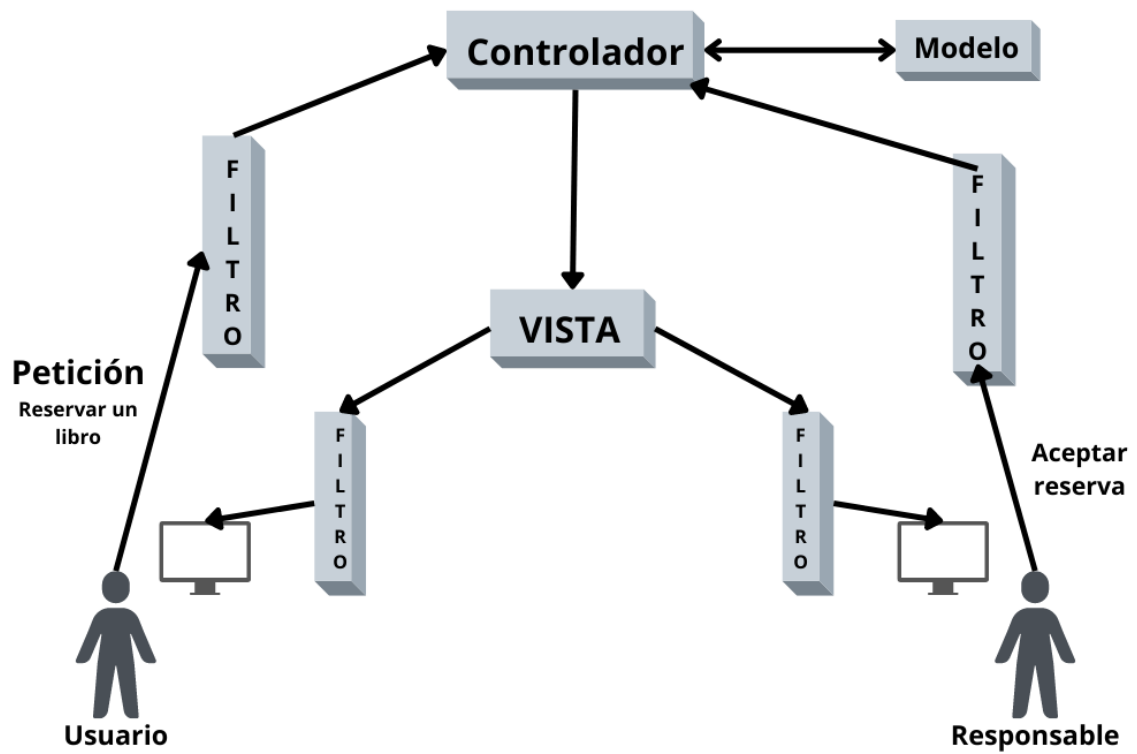
6.2. *Requisitos de seguridad*

Cada usuario que es registrado individualmente, recibirá una contraseña por parte del responsable, si es una inserción masiva, cada usuario tendrá una contraseña aleatoria y recibirá el correo electrónico, contraseña y dirección URL de la aplicación.

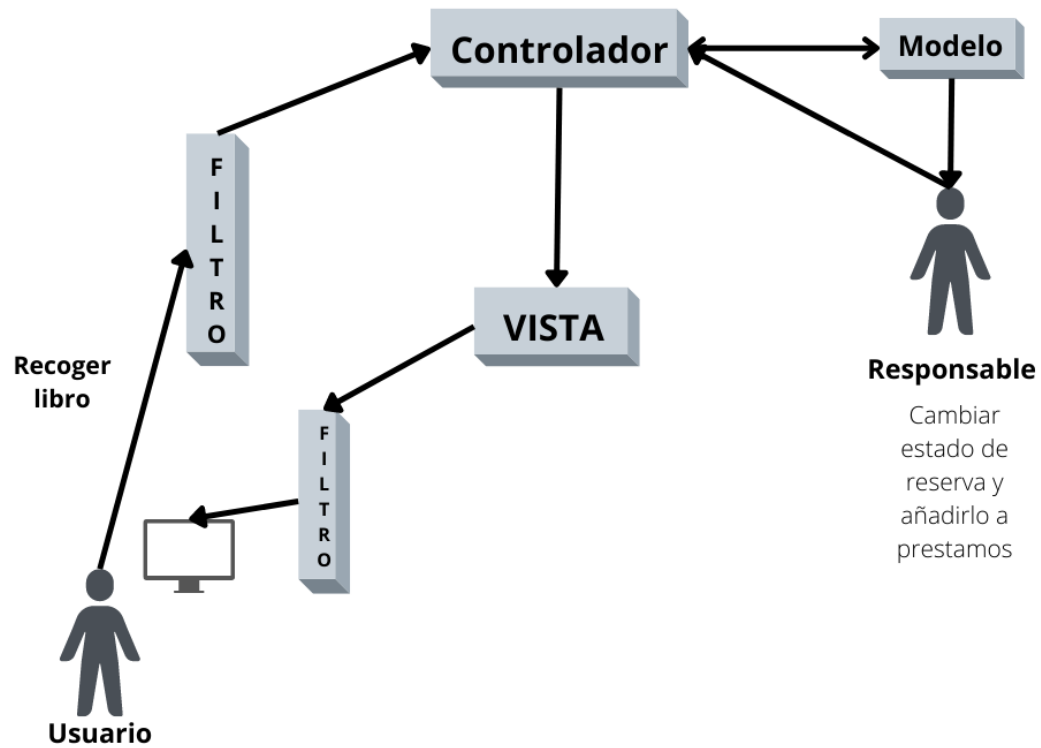
En cuanto a la seguridad, de la aplicación web, utilizamos un filtro, que valida los datos del usuario y si son correctos envía los datos a la sesión actual, en caso que los datos no sean correctos se volverá a la página de inicio de sesión

6.3. Diagrama de casos de uso

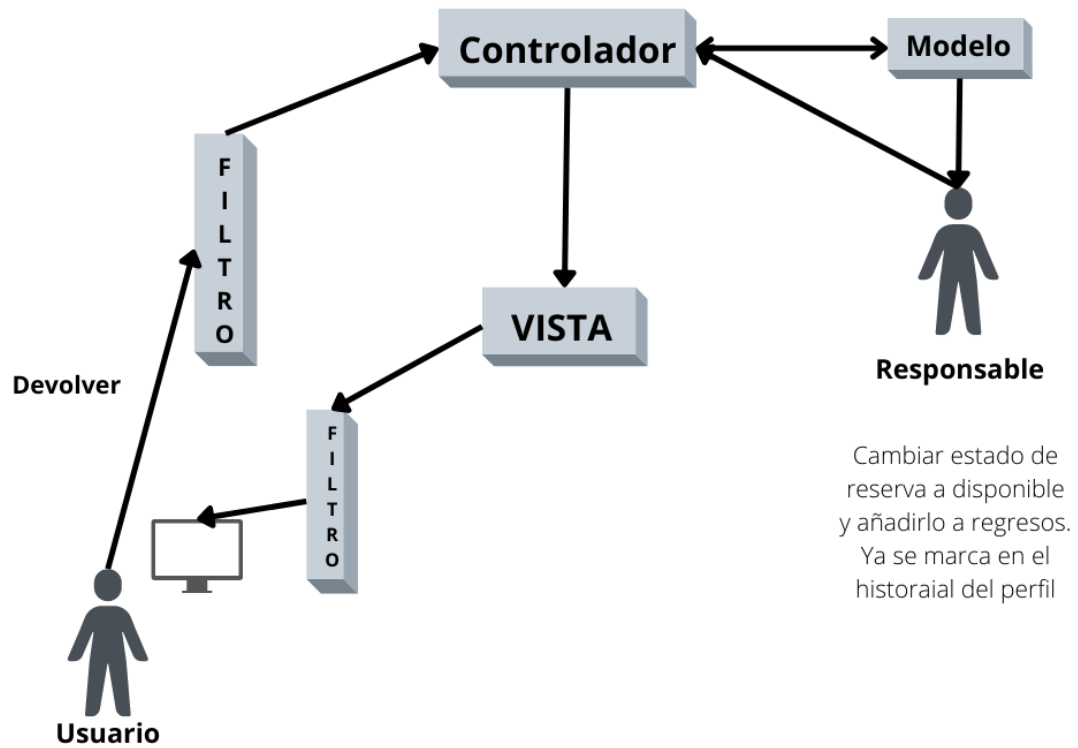
6.3.1. Reservar libro



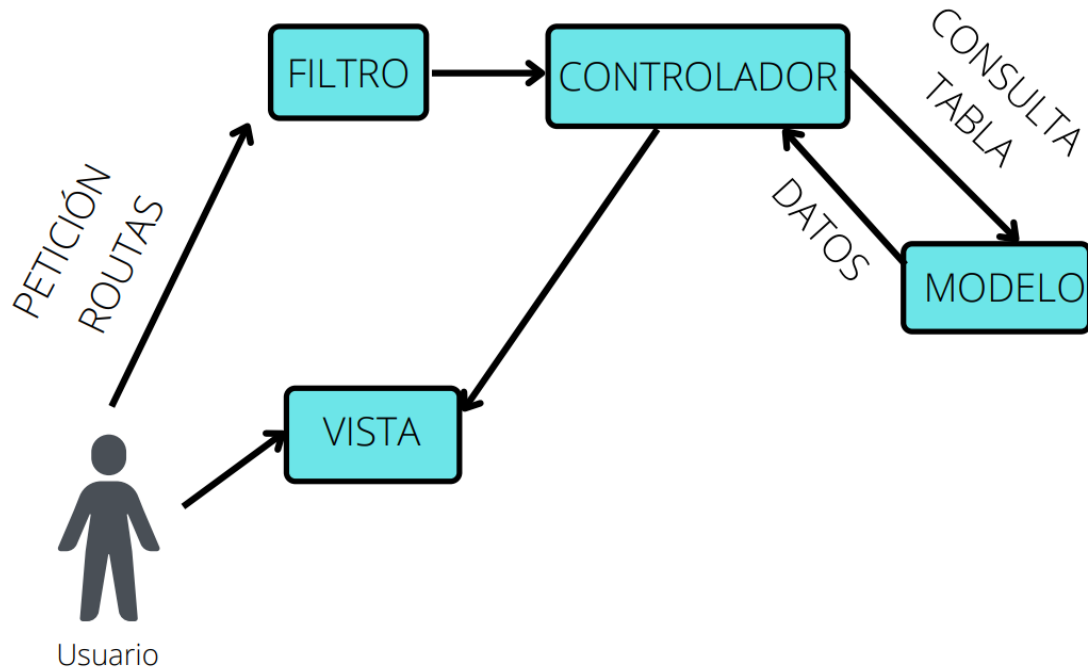
6.3.2. Recoger libro



6.3.3. Devolver libro



6.4. Arquitectura MVC



6.5. Apis i crides

Rutas	Método	Controlador
Parte pública		
localhost/api/publica/inicio	GET	ApiPublicaController/nicio
localhost/api/publica/catalogo		ApiPublicaController/catalogo
localhost/api/publica/horarios		ApiPublicaController/horarios
localhost/api/publica/busqueda_simple		ApiPublicaController/busqueda_simple
localhost/api/publica/catalogo/busqueda_avanzada	POST	ApiPublicaController/busqueda_avanzada
localhost/api/iniciar_sesion		ApiPublicaController/niciar_sesion

Administrador		
localhost/api/usuarios/privado/1/gestion_responsable	GET	ApiUsuarioAdministradorController/ gestion_responsable
localhost/api/usuarios/privado/1/agregar_responsable		ApiUsuarioAdministradorController/ agregar_responsable_post
localhost/api/usuarios/privado/1/activar_desactivar		ApiUsuarioAdministradorController/ activar_desactivar
localhost/api/usuarios/privado/1/mi_cuenta_administrador		ApiUsuarioAdministradorController/ mi_cuenta_administrador
localhost/api/usuarios/privado/1/mi_cuenta_administrador	POST	ApiUsuarioAdministradorController/ mi_cuenta_administrador_post
Responsable		
localhost/api/usuarios/privado/2/editar_biblioteca	GET	ApiUsuarioResponsableController/ editar_biblioteca
localhost/api/test/usuarios/privado/2/editar_biblioteca	POST	ApiUsuarioResponsableController/ editar_biblioteca_post
localhost/api/usuarios/privado/2/gestion_libros	GET	ApiUsuarioResponsableController/ gestion_libros
localhost/api/usuarios/privado/2/agregar_libro	POST	ApiUsuarioResponsableController/ agregar_libro_post
localhost/api/usuarios/privado/2/borrar_libro		ApiUsuarioResponsableController/ borrar_libro

localhost/api/usuarios/privado/2/gestion_ejemplares	GET	ApiUsuarioResponsableController/ gestion_ejemplares
localhost/api/usuarios/privado/2/agregar_ejemplar		ApiUsuarioResponsableController/ agregar_ejemplar
localhost/api/usuarios/privado/2/borrar_ejemplar		ApiUsuarioResponsableController/ borrar_ejemplar
localhost/api/usuarios/privado/2/gestion_usuarios		ApiUsuarioResponsableController/ gestion_usuarios
localhost/api/usuarios/privado/2/agregar_profesor	POST	ApiUsuarioResponsableController/ agregar_usuario_post
localhost/api/usuarios/privado/2/agregar_estudiante		ApiUsuarioResponsableController/ agregar_usuario_post
localhost/api/usuarios/privado/2/agregar_pas		ApiUsuarioResponsableController/ agregar_usuario_post
localhost/api/usuarios/privado/2/activar_desactivar	GET	ApiUsuarioResponsableController/ activar_desactivar
localhost/api/usuarios/privado/2/mi_cuenta		ApiUsuarioResponsableController/ mi_cuenta
localhost/api/usuarios/privado/2/mi_cuenta	POST	ApiUsuarioResponsableController/ mi_cuenta_usuario_post
localhost/api/usuarios/privado/2/reservas	GET	ApiUsuarioResponsableController/ reservas

localhost/api/usuarios/privado/2/reserva_aceptada	POST	ApiUsuarioResponsableController/ reserva_aceptada
Usuarios		
localhost/api/usuarios/privado/3/mi_cuenta	GET	ApiUsuarioController/mi_cuenta
localhost/api/usuarios/privado/3/mi_cuenta	POST	ApiUsuarioController/ mi_cuenta_profesor_post
localhost/api/usuarios/privado/3/catalogo	GET	ApiUsuarioController/catalogo
localhost/api/usuarios/privado/3/reservar		ApiUsuarioController/reservar
localhost/api/usuarios/privado/3/recogido		ApiUsuarioController/recogido
localhost/api/usuarios/privado/3/historial_reservas		ApiUsuarioController/historial_reservas
localhost/api/usuarios/privado/3/formulario_opinion		ApiUsuarioController/formulario_opinion
localhost/api/usuarios/privado/3/opinar	POST	<u>ApiUsuarioController/opinar</u>
localhost/api/usuarios/privado/3/opiniones	GET	ApiUsuarioController/opiniones
localhost/api/usuarios/privado/3/devolver	GET	ApiUsuarioController/devolver

6.6. Funcionalidades de los usuarios

6.6.1. Web

6.6.1.1. Parte pública

Todos los usuarios tendrán las mismas funcionalidades de en la parte pública

- Ver los datos de la biblioteca

- Ver el catálogo, que tiene dos formas de búsqueda, una por el título del libro y otra búsqueda avanzada, que busca por el autor, nombre libro y categoría
- Ver los horarios de la biblioteca y ver los responsables de cada franja horaria
- Iniciar sesión

6.6.1.2. ***Parte privada***

Administrador

Gestiona los responsables de la biblioteca

- Registrar individualmente a los responsables de la biblioteca
- Activar o desactivar a cualquier responsable. Un responsable activo tiene todas las funcionalidades que se mencionarán en el siguiente punto, si esta desactivado no podrá gestionar nada y mucho menos iniciar sesión
- Cambiar sus datos personales

Responsable

El usuario que tiene más funciones activas, es el intermediario entre la biblioteca y los usuarios.

- Una de sus funciones es aceptar una reserva pedida. De manera que pondrá la fecha a la que el usuario tiene que venir a buscar el libro y aceptar la reserva
- Cambiar sus datos personales

- Modificar los datos de la biblioteca actual
- Libros:
 - Eliminar los libros que vea necesario, la cual cosa también sus ejemplares activos se desactivarán
 - Añadir libros individualmente, con todos los datos obligatorios
 - Añadir libros masivamente, mediante un archivo csv
- Ejemplares:
 - Generar un PDF con los códigos QR del ejemplar seleccionado
 - Añadir nuevos ejemplares de un libro
 - Borrar ejemplares, solo desactiva el último ejemplar de la lista, pero no lo elimina por cuestiones de estadísticas y historial
 - Ver el historial de todos los ejemplares, activos y desactivados
- Usuarios:
 - Activar o desactivar los usuarios: profesor, estudiante y pas
 - Añadir masivamente a: profesores, estudiantes y pas
 - Añadir individualmente un: profesor, estudiante y pas

Usuario general (profesores, estudiante, pas)

- Reservar ejemplar de un libro
- Aceptar el prestamos de un ejemplar
- Devolver el ejemplar

- Cambiar sus datos personales

6.6.1.3. Ejemplos archivos CSV

- Carga masiva profesores -> [Mockaroo - CàrregaMassivaProfessorat](#)
- Carga masiva estudiantes-> [Mockaroo - CàrregaMassivaEstudiants](#)
- Carga masiva pas-> [Mockaroo - pas](#)
- Carga masiva libros -> [Mockaroo - libros](#)

6.6.2. Aplicación móvil

6.6.2.1. Parte pública

Todos los usuarios tendrán las mismas funcionalidades de en la parte pública

- Ver los datos de la biblioteca
- Ver el catalogo, que tiene dos formas de búsqueda, una por el título del libro y otra búsqueda avanzada, que busca por el autor, nombre libro y categoría
- Ver los horarios de la biblioteca y ver los responsables de cada franja horaria
- Iniciar sesión

6.6.2.2. Parte privada

Administrador

- Cambiar sus datos personales

Responsable

- Una de sus funciones es aceptar una reserva pedida. De manera que escaneará el código QR, en específico los 2 últimos números que hacen referencia a el número del ejemplar. Esta información vendrá un tablero de anuncios cuando inicie sesión
- Regresar el ejemplar en curso, de la misma manera que reservar, escaneará el número de ejemplar y lo volverá a poner en disponible
- Cambiar sus datos personales
- Libros:
 - Añadir libros individualmente, escaneará el ISBN-13 del libro y si es válido añadirá el libro al catálogo
- Ejemplares:
 - Añadir nuevos ejemplares de un libro. Se escaneará el ISBN-13 de un libro y si existe se añadirá otro ejemplar de ese libro

Usuario general (profesores, estudiante, pas)

- Reservar ejemplar de un libro. Desde el catálogo de la parte privada, los usuarios verán los ejemplares disponibles, si lo selecciona directamente se pasa al tablero de anuncios del responsable, quien debe aprobar la reserva
- Cambiar sus datos personales

6.7. Uso de Docker

El profesor nos va facilitar el funcionamiento de Docker. Solo teníamos que modificar lo siguiente:

- **Archivo:** docker-compose.yml

```
version: "3.4"

services:

  codeigniter4:
    build:
      context: ./docker/php
      container_name: 'codeigniter41'
      hostname: codeigniter4
      ports:
        - 80:80
      links:
        - mysql
      volumes:
        - ./www:/var/www/html

  mysql:
    build:
      context: .\docker\mysql
      container_name: docker-mysql1
      hostname: dockermysql
      environment:
        MYSQL_DATABASE: flourish_blotts
        MYSQL_ROOT_PASSWORD: 1234
      ports:
        - "3307:3306"
      restart: always
```

- asignar el nombre de la base de datos del proyecto

- Una contraseña
- En mi caso cambiar el puerto al 3307

- **Archivo:** .env

```
database.default.hostname = dockermysql
database.default.port = 3307
database.default.database = flourish_blotts
database.default.username = root
database.default.password = 1234
database.default.DBDriver = MySQLi
```

- En este archivo se tenía que asignar el nombre del contenedor de mysql, que cumple la función de MySQL de Xampp
 - En mi caso añadir el puerto 3307
 - Nombre de la base de datos
 - Asignar contraseña
- El siguiente paso ejecutar el comando **docker-compose up**, que creará los contenedores que cumplirán las funciones de PHP y MySQL
- Una vez finalizada la comanda, arrancamos el contenedor de MySQL y entramos

```

docker exec -it 21caac34312dd284ed89d49c37cd471d7fdb98e931f47dff205ee8b36f252e6 /bin/sh
# su
root@dockermysql:/# ls
bin dev                                entrypoint.sh home lib64 mnt proc run srv tmp var
boot docker-entrypoint-initdb.d etc    lib media opt root sbin sys usr
root@dockermysql:/# docker etc
bash: docker: command not found
root@dockermysql:/# cd etc
root@dockermysql:/etc# ls
X11                                deluser.conf host.conf ld.so.conf.d mtab perl rcS.d ssl
adduser.conf dpkg hostname ldap mysql profile resolv.conf subgid
alternatives environment hosts libaudit.conf nanorc profile.d rmt subuid
apt fstab init.d localtime nsswitch.conf rc0.d securetty systemd
bash.bashrc gai.conf inputrc logcheck opt rc1.d security terminfo
bindresvport.blacklist group issue login.defs os-release rc2.d selinux timezone
cron.daily group- issue.net logrotate.d pam.conf rc3.d shadow update-motd.d
debconf.conf gshadow kernel mecabrc pam.d rc4.d shadow- xattr.conf
debian_version gshadow- ld.so.cache mke2fs.conf passwd rc5.d shells
default gcc ld.so.conf motd passwd- rc6.d skel
root@dockermysql:/etc# cd mysql
root@dockermysql:/etc/mysql# ls
conf.d my.cnf my.cnf.fallback
root@dockermysql:/etc/mysql# nano my.cnf
root@dockermysql:/etc/mysql#

```

Añado la siguiente línea al final del archivo

```

docker exec -it 0f200227d5df0f0ecede4a2cfcc7d2a930b54dc4d9b995d5e4b485cd745e248d /bin/sh
# su
root@dockermysql:/#
root@dockermysql:/etc/mysql# nano my.cnf
GNU nano 3.2 my.cnf
root@dockermysql:/etc/mysql#
root@dockermysql:/etc/mysql# # Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
root@dockermysql:/etc/mysql#
root@dockermysql:/etc/mysql# # The MySQL Server configuration file.
root@dockermysql:/etc/mysql#
root@dockermysql:/etc/mysql# # For explanations see
root@dockermysql:/etc/mysql# # http://dev.mysql.com/doc/mysql/en/server-system-variables.html
root@dockermysql:/etc/mysql#
root@dockermysql:/etc/mysql# [mysqld]
root@dockermysql:/etc/mysql# pid-file = /var/run/mysqld/mysqld.pid
root@dockermysql:/etc/mysql# socket = /var/run/mysqld/mysqld.sock
root@dockermysql:/etc/mysql# datadir = /var/lib/mysql
root@dockermysql:/etc/mysql# secure-file-priv= NULL
root@dockermysql:/etc/mysql#
root@dockermysql:/etc/mysql# # Custom config should go here
root@dockermysql:/etc/mysql# !includedir /etc/mysql/conf.d/
root@dockermysql:/etc/mysql# port=3307
root@dockermysql:/etc/mysql#

```

Contenedor mysql

- Arrancamos el contenedor de php y ejecutamos la siguiente comanda, para obtener datos cuando nos conectemos a la web

```
# su
root@codeigniter4:/var/www/html# cd
root@codeigniter4:~# ls
root@codeigniter4:~# ^C
root@codeigniter4:~# cd /var/www/html
root@codeigniter4:/var/www/html# cd Flourish_and_Blotts/
root@codeigniter4:/var/www/html/Flourish_and_Blotts# php spark migrate

CodeIgniter v4.1.9 Command Line Tool - Server Time: 2022-05-19 12:52:49 UTC-05:00

Running all new migrations...
Running: (App) 2022-05-15-151955_App\Database\Migrations\LibrosMigration
Running: (App) 2022-05-15-152313_App\Database\Migrations\AutoresMigration
Running: (App) 2022-05-15-152428_App\Database\Migrations\LibroAutorMigration
Running: (App) 2022-05-15-152534_App\Database\Migrations\CategoriasMigration
Running: (App) 2022-05-15-152719_App\Database\Migrations\LibroCategoriaMigration
Running: (App) 2022-05-15-152815_App\Database\Migrations\EjemplaresMigration
Running: (App) 2022-05-15-152928_App\Database\Migrations\UsuariosMigration
Running: (App) 2022-05-15-153108_App\Database\Migrations\RolesMigration
Running: (App) 2022-05-15-153150_App\Database\Migrations\ProfesoresMigration
Running: (App) 2022-05-15-154416_App\Database\Migrations\EstudiantesMigration
Running: (App) 2022-05-15-154633_App\Database\Migrations\ReservasMigration
Running: (App) 2022-05-15-154753_App\Database\Migrations\RegresosMigration
Running: (App) 2022-05-15-154908_App\Database\Migrations\PrestamosMigration
Running: (App) 2022-05-15-155036_App\Database\Migrations\OpinionesMigration
Running: (App) 2022-05-15-155236_App\Database\Migrations\PenalizacionesMigration
Running: (App) 2022-05-15-155343_App\Database\Migrations\AddRevokeTokensTable
Running: (App) 2022-05-19-152659_App\Database\Migrations\BibliotecasMigration
Done migrations.
```

Para actualizar el proyecto si hacemos alguna modificación, ejecutamos la comanda **composer update --ignore-platform-reqs**

7. Gestión del proyecto

7.1. *Plan de trabajo*

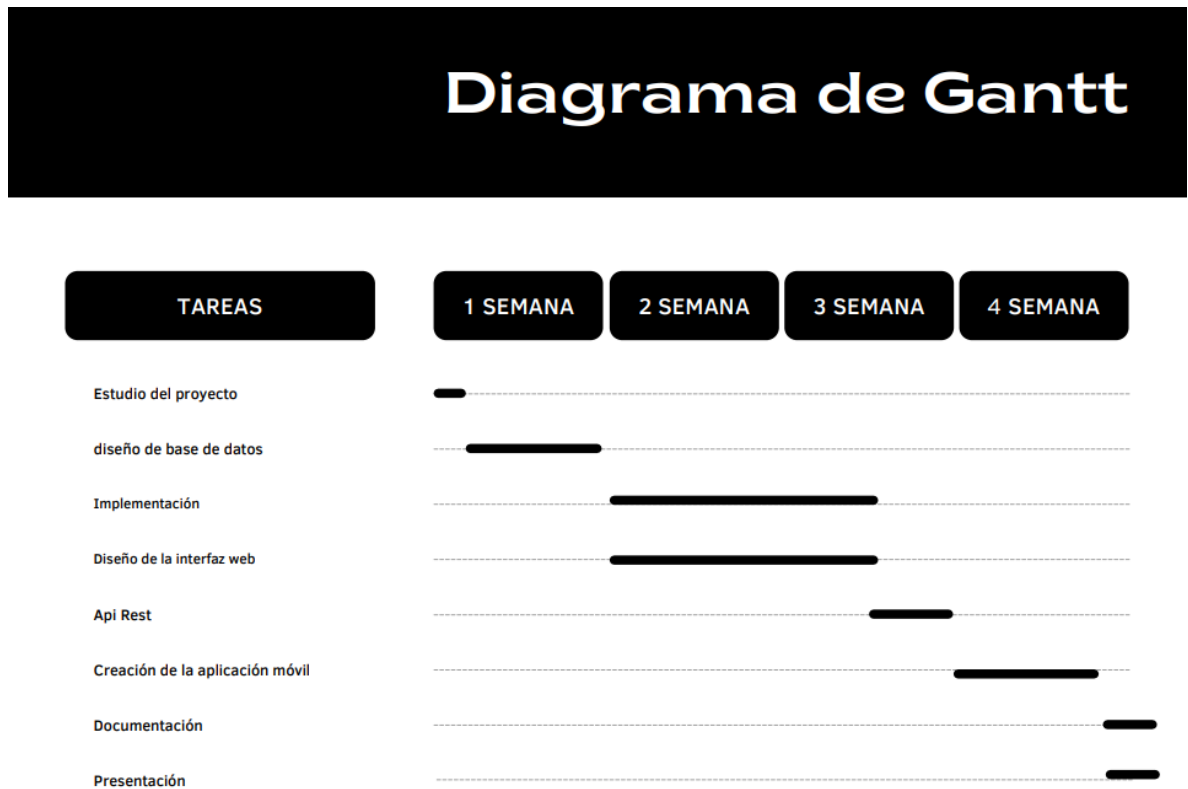
-1ra semana: estudio del proyecto + diseño de base de datos

-2da y 3ra semana: implementación + diseño de la interfaz web + Api Rest

-4ra semana: creación de la aplicación móvil + documentación + presentación

- Los primeros días me dediqué a leer el proyecto, sobre todo las funcionalidades que tenía cada usuario. Las entidades que deben estar y las que no son necesarias, pero se pueden crear. Esto se alargó hasta 1 semana y media, y la primera semana de proyecto me dediqué al estudio y diseño de bases de datos, puede que hasta se haya alargado a 1 semana y 3 días

7.2. Diagrama de Gantt



7.3. Gestión de la aplicación móvil

En la aplicación móvil, en la parte privada del responsable, hay 2 maneras de gestionar las reservas y regresos:

- Reservar un libro -> previamente el usuario 'profesores, estudiante, PAS' ha tenido que reservarlo. Paso siguiente la reserva le aparece al responsable en su respectivo tablero, con el estado de **espera**. Una vez el responsable lo ha reservado para ese

usuario marcará el estado del ejemplar como **reservado**

- Regresar ejemplar -> Esta función marcará el estado actual del ejemplar como disponible y desaparecerá del tablero del responsable y la opción reservar desde el catálogo volverá a estar disponible

Porque no hice la gestión de préstamos, porque esto conllevaría una respuesta del usuario, según está diseñada la API. Como la única funcionalidad del usuario 'profesor, estudiante, pas' es modificar sus datos, mostrar su tablero y reservar un libro

8. Conclusiones.

8.1. Más importante

Sin duda el perfil del responsable es el que tenía más funcionalidades, aunque ahora no parezca difícil en el momento de implementación habían du La gestión de ejemplares, reservar, aceptar reserva, devolver, etc. Era algo que veía la forma de hacerlo, aunque con dudas cuando pasaba a la acción, porque me venían nuevas ideas de cómo hacerlo y si, si un día lo tenía claro cómo hacerlo (y esto lo digo en general) pero antes de finalizar el día lo tenía operativo al día siguiente ya había ideas diferentes de como se tenía que gestionar.

8.2. Propósito y objetivos

8.2.1. Propósito

En cuanto a la forma de trabajo, en un principio parecía que teníamos tiempo y me relajé,

pero me di cuenta que en un proyecto, hay que controlar muchas cosas y hay que llevarlo al día, con sus respectivos apuntes y lista de tareas para no relajarse.

El reparto de mi tiempo, el que me llevo más horas de trabajo fue la parte web, porque hay que controlar muchas cosas, más el estilo que se tiene que añadir y sea fácil o deducible para el consumidor. Después la aplicación móvil, con las API diseñadas es más ligero de gestionar, aunque también se tiene que dedicar su tiempo y atención

Los conocimientos que adquirí haciendo el proyecto son aun mayores a los que pude adquirir en su día, cuando eran prácticas relacionadas. Encarar el problema, buscar soluciones, me llevo una gran enseñanza. En el diseño de **bases de datos**, relaciones, claves primarias, **PHP**, **ANGULAR**, **GitHub**.

8.2.2. Objetivos

Mi objetivo principal era desarrollar la aplicación, que sea funcional, que sea fácil de usar, que al menos se vea bien, aunque se puede mejorar visualmente, veía que me consumiría tiempo y decidí tenerlo funcional, pero sin descuidar la parte visual.

8.3. Mejoras

Todos los datos de la biblioteca son editables, cada vez que se haga algún cambio se mostrará lo modificado, pero el nombre de la biblioteca es estático.

El catálogo de la aplicación móvil, la parte pública para ser más concreto la búsqueda simple, muestra todos los libros, pero no pude hacer el control de letras mayúsculas, por

consiguiente, si la letra coincide, pero es mayúscula o minúscula no habrá resultado.

8.4. Nivel personal

Este proyecto me ayudo a sumar más conocimientos, que antes de iniciar el proyecto no lo sabía o si lo sabía podía hacerlo mejor.





A nivel de organización, muy bueno porque me ayudo a tener una lista con todo lo que tengo pendiente o cosas a mejorar. Apuntar las ideas que venían y sobre todo estar atento en clase ahorraría estrés y tiempo.

9. ANNEXOS:

9.1. Lliurament del treball

<https://github.com/BrandonSotomayor/DAW2-Sintesi-Sotomayor-Brandon>

Descripció de fitxers i carpetes adjuntades¹

 <Docker_capa_main>	<La carpeta donde está la configuración para hacer arrancar el proyecto con Docker>
 <Flourish_and_Blotts>	<Carpeta donde se encuentra el proyecto de la aplicación web con codeigniter>
 <Flourish_and_Blotts_App>	<Carpeta donde se encuentra la aplicación móvil de la biblioteca>
 <Memoria>	<Memoria del proyecto>

¹