MEMORIA PROYECTO FINAL DE TRIMESTRE

Descripción breve

Este es el documento de memoria técnica para el proyecto final del primer trimestre correspondiente al módulo Desarrollo Web en entorno Servidor

Iván Hernández Fuentes

Profesor: Jesús Arribi Vilela

Resumen

Haces años la rama encargada de la ingeniería del software tenía como gran preocupación la creación de procesos que asegurasen la calidad de los programas realizados, esos procesos consistían en la estructuración de los programas y la reutilización del código para poder influir positivamente en la facilidad del desarrollo y mantenimiento del software.

Los ingenieros del software tienen como objetivo principal estudiar de qué manera se pueden mejorar los procesos de creación de software, y una de las soluciones o conclusiones a las que llegaron, es la arquitectura basada en capas, que separa el código en función de sus responsabilidades o conceptos.

Según Miguel Ángel Álvarez (2014, Desarrollo Web) *Por qué MVC*

Es ahí donde entran en juego los patrones de diseño comúnmente conocidos por sus siglas MVC (Modelo Vista Controlador) uno de los primeros patrones desarrollados en 1979, aunque desde entonces hasta ahora este modelo ha cambiado mucho adaptándose a varias variaciones como el **MVP (Modelo Vista Presentador)**, **PM (Modelo de presentación) y MVVM (Modelo-vista-modelo de vista).**

El presente documento pretende explicar y detallar el análisis del proyecto realizado, las implementaciones que se han hecho, así como los problemas encontrados y las futuras mejoras a las que puede optar la aplicación.

El proyecto ha sido desarrollado en PHP mediante ATOM como entorno de desarrollo.

Índice

Tabla de ilustraciones

Introducción

El Objetivo del proyecto es realizar una aplicación **CRUD** siguiendo el patrón de diseño **Modelo Vista Controlador (MVC)** ***(Véase figura 1.1)*** **y la Programación Orientada a Objetos (POO),** poniendo así en práctica los conocimientos adquiridos por el alumno a lo largo del trimestre.

Antes de seguir detallando en esta memoria el proyecto, explicaré brevemente en que consiste el **MVC**.

El MVC es un patrón de diseño muy **utilizado en la arquitectura del Software**, donde generalmente **existen 3 componentes** principales, por un lado, los **controladores**, por otro los **modelos** y por último las **Vistas**, el **objetivo** principal **del MVC es separar los datos y la lógica de negocio**, más conocido como la programación orientada a objetos **(POO).**

El MVC permite separar los diferentes componentes de nuestra aplicación dependiendo de la responsabilidad que tiene cada uno, es decir si realizamos un cambio en alguna parte del código, este no afecte a otra parte del mismo. Un ejemplo muy fácil de entender es si por ejemplo **modificamos nuestra Base de datos (BD),** sólo se debería de **modificar el modelo** que es quién se encarga de manejar los datos existentes en la BD, dejando así al resto de la **aplicación intacta**, de esta manera se respeta el **Single Responsibility Principle (SRP**) conocido comúnmente cómo **"El Principio de responsabilidad única"** en Ingeniería del Software.

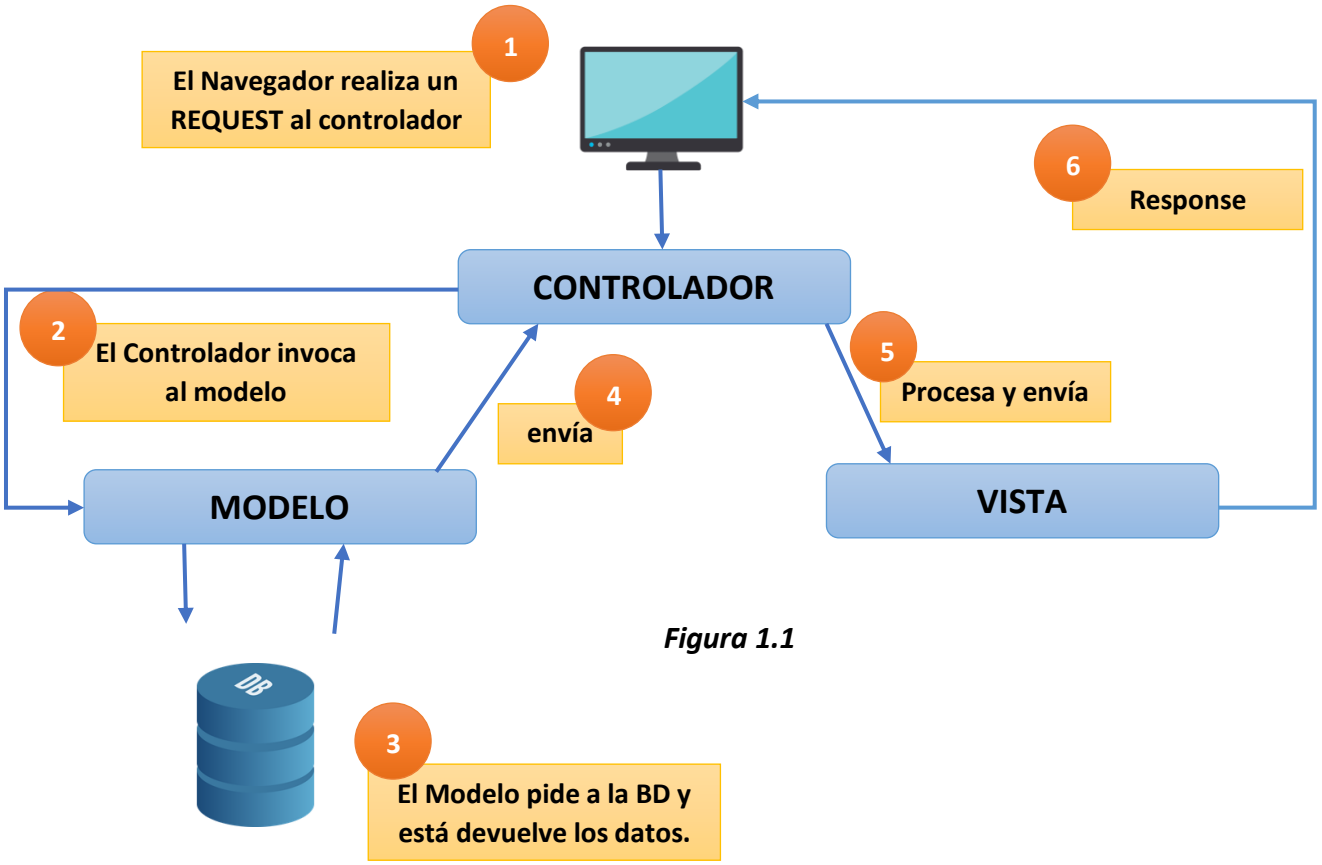


Figura 1.1

Descripción *Figura 1.1*

El controlador:

El controlador responde a los eventos o acciones del usuario, e interacciona si es necesario con la capa de persistencia(Modelo).

El Modelo:

Es donde se trabaja conjuntamente con la Base de Datos, se encarga de acceder y modificar la información. En él se realizan todas las operaciones **CRUD, Crear (Create), Leer(Read), Actualizar (Update) y eliminar(Delete).**

La Vista:

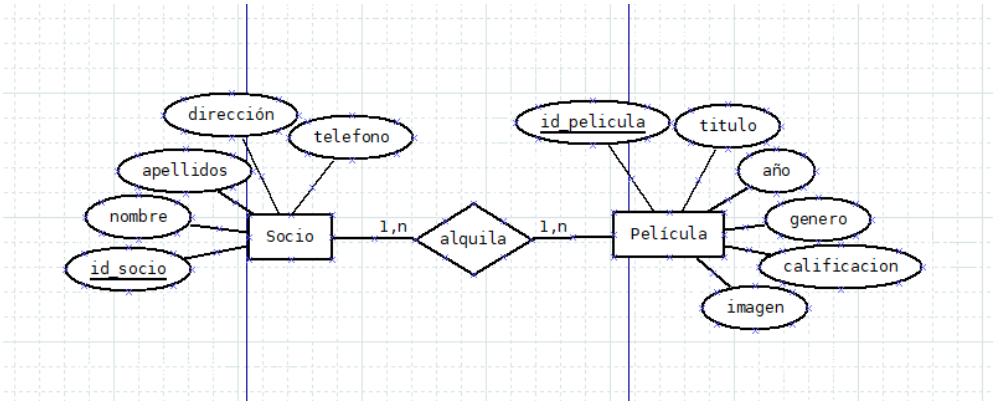
Es la capa de presentación(HTML), es la parte que el usuario final ve.

Análisis del proyecto

Modelo Entidad Relación

En el proyecto nos hemos basado en una aplicación para gestionar un videoclub, donde tenemos por un lado los socios y por otro las películas.

A continuación, se muestra el **Entidad Relación del videoclub**

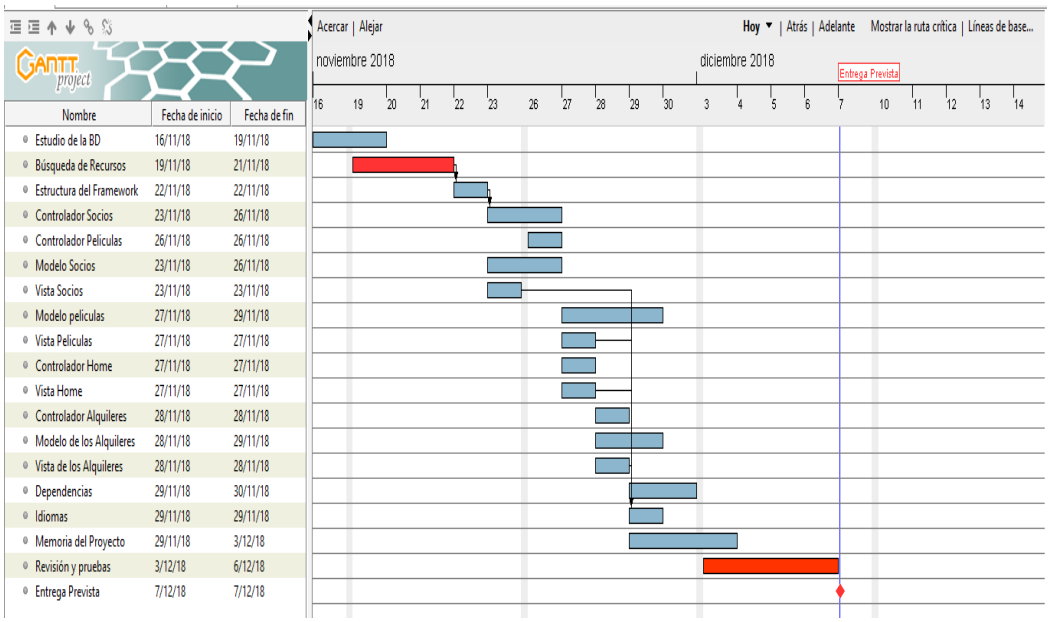


Cabe recordar que existe una tabla más (usuarios del sistema) *

En este caso un socio puede alquilar 1 o varias películas, y esa película puede ser alquilada por 1 o varios socios.

Respetando las relaciones existentes se generarían 3 tablas (***Socios, Alquileres, Películas***).

Diagrama de Gantt



Diseño e implementación

Estructura de carpetas

La estructura de nuestro proyecto está compuesta de las anteriores carpetas, se dividen en tres carpetas principales.

En primer lugar, encontramos **"app"** que contiene la estructura de carpetas del patrón de diseño MVC, tales como los **controladores, modelos y las vistas**.

Le sigue el directorio **"locale"** donde se almacena la configuración del **idioma de nuestra aplicación.**

En el directorio **"vendor"** sus subcarpetas son las **dependencias** que existen en nuestro proyecto (estás se detallan más adelante)

Por último, tenemos la carpeta "**public"**, dentro de ella esta la carpeta del "**css"**, cabe recordar que *en este proyecto se utiliza un **content delivery network (CDN)** conocido como red de distribución de contenidos.*

La carpeta **"img"** es donde se **almacenan las imágenes** que se suben al servidor, y la **carpeta "js"** donde tenemos los **scripts de nuestro proyecto**.

Dependencias

Se han utilizado varias dependencias en el proyecto utilizando el manejador de paquetes **composer**, las dependencias que hay en el proyecto son **PHPMailer y PHPDocumentor.**

PHPMailer: Es una librería que permite enviar emails.

PHPDocumentor: Librería que permite generar documentación automática.

Patrones de Diseño

Los patrones de diseño utilizados han sido principalmente **el MVC** y también el patrón **POST-Redirect-GET-FLASH.**

Aspectos Relevantes

La aplicación web consta de algunos elementos relevantes, descritos a continuación.

- **Login**: Existe un login, dónde el usuario puede tanto iniciar sesión como registrarse. A la hora de registrarse, la BD almacena tanto el email como la contraseña, de dicha contraseña se obtiene el hash, que es almacenado en la BD.

- **Internacionalización y localización:** La aplicación consta de diferentes idiomas entre ellos el español y el inglés.

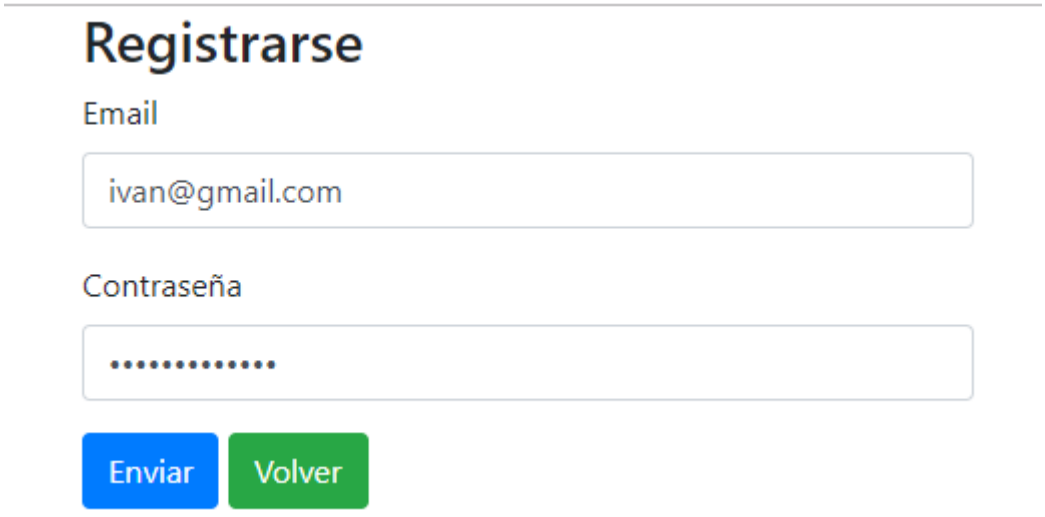
- **Uso de API:** El proyecto hace uso de la **Application Programming Interface,** comúnmente conocida por sus siglas **API**. El uso que se hace de la API es a la hora de **insertar una película**, puesto que existen dos formas para introducirla. La primera es mediante una **API externa (MOVIEBD),** el usuario introduce el título de la película ***(titulo original)*** se realiza la **consulta a la API** y se **muestra** al usuario la **información** devuelta por la misma. La segunda opción es rellenando el formulario.

Casos de Uso

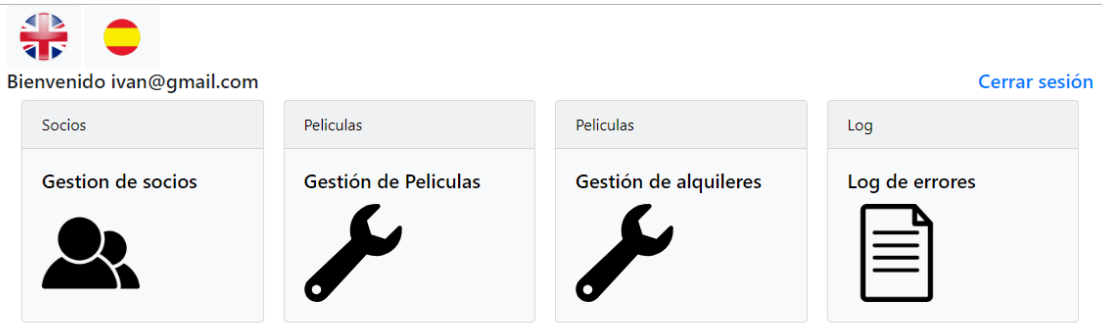
A continuación, detallaremos brevemente los “casos de uso” que tiene nuestra aplicación.

- Iniciar sesión.
- Registrarse en la Aplicación.
- Seleccionar el idioma.
- Gestionar Socios
    - Añadir Socios.
    - Modificar Socios.
    - Ver detalles del socio
- Gestionar Películas
    - Añadir Películas
    - Modificar Películas
    - Ver detalles de la película
- Gestionar los alquileres
    - Añadir alquileres.
    - Eliminar Alquileres.

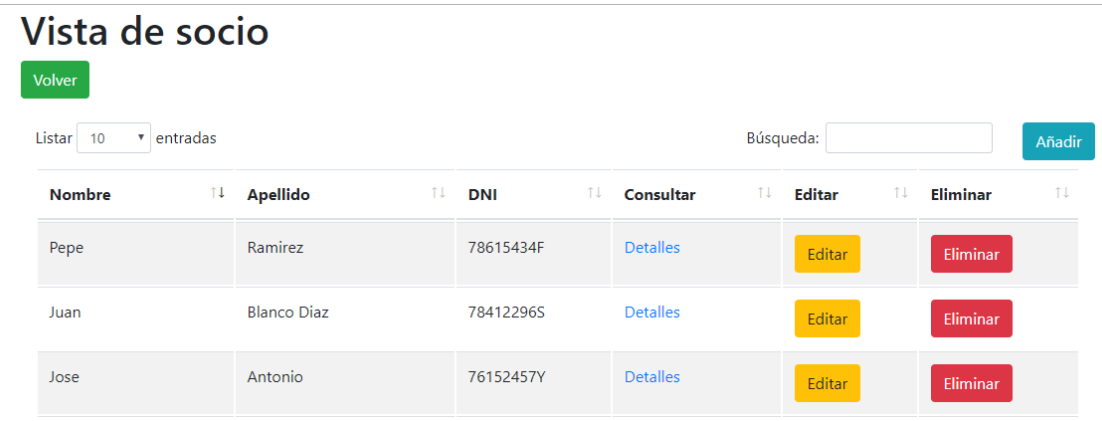
Ejemplos casos de uso



El empleado se registra en la aplicación, el sistema almacena en la BD el email y el hash de la contraseña.

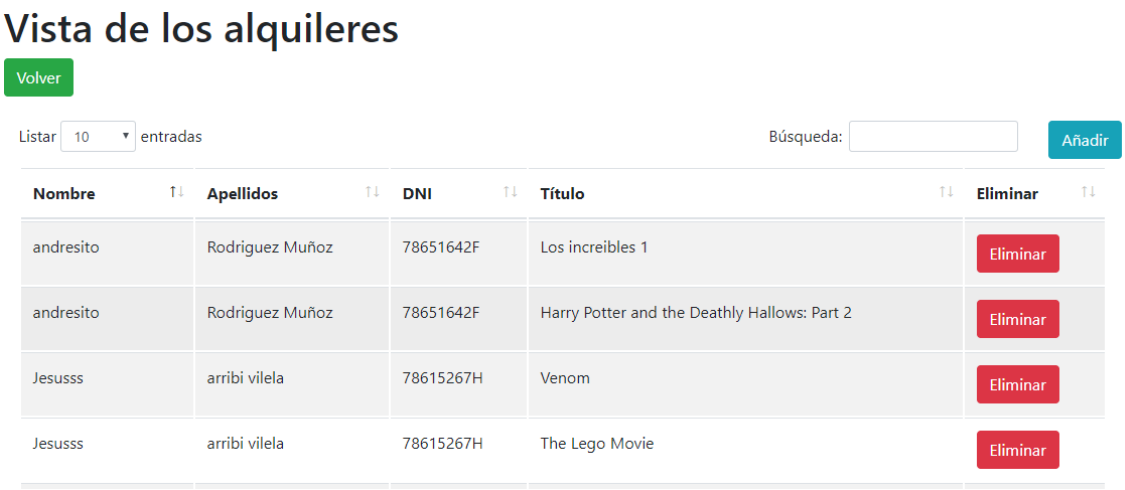


El usuario puede escoger el idioma que prefiera, inglés o español (por defecto está en español). Visualiza un panel con las gestiones que puede realizar.



*Si el empleado gestiona un socio le lleva a la vista general de los socios, dónde puede añadir, buscar y editar. Aunque también puede eliminar, pero es una opción que no estará disponible en el futuro (**Solo podrá eliminar el administrador).***

Si gestiona las películas el funcionamiento es el mismo que en los socios (añadir, editar y buscar).



Para la gestión de los alquileres en este caso el usuario normal si puede eliminar los alquileres.

Problemas encontrados

Como en toda aplicación en desarrollo, en esta se han tenido algunos problemas a la hora de desarrollarla.

El primer problema surgido fue en la parte de los alquileres, con los **Datalist**, mi idea era que en vez de que saliera un desplegable con los socios y otro con las películas se hiciera una especie de buscador ya que si es una base de datos con miles de registros sería muy complicado buscar uno por uno.

Intenté implementar algún cuadro de búsqueda y me encontré con **los "datalist" de HTML5** que permite realizar búsquedas. El problema surge al implementarlo ya que el resultado no era lo esperado. A la hora de buscar en el datalist, realizaba la búsqueda correctamente pero el resultado final que se mostraba al usuario no era de lo mas visual ya que se **representaba el id del socio y el id de la película**, busqué por internet como solucionar el problema, y encontré una posible solución, pero al implementarlo la propiedad "**value" del datalist pasaba hacer el titulo de la película o el nombre del socio**, cuando lo que yo **necesito es el id de ambos**. Al final lo deje como estaba para no complicarme mucho.

Otro problema encontrado fue cuando al cargar una **vista (socios),** en esta vista se **muestra** tanto el **número de películas alquiladas** por el socio, cómo **las imágenes de las mismas**. Al cargar la vista desde el **controlador Socios** se llamaban hasta 3 métodos o funciones diferentes desde el modelo películas y el modelo socio, esto hacía que por lo menos en ese momento yo no pudiera pasar mas argumentos a la función '**CargaVista'**, por lo que tuve que **modificar** esta **función** para que aceptara **3 argumentos.**

Conclusiones y futuros trabajos a desarrollar

Este proyecto me ha dado la oportunidad de adquirir multitud de conocimientos, además de aprender a como desarrollar una aplicación web de manera profesional, estructurada y segura utilizando los mecanismos correctos para ello.

El desarrollo del proyecto utilizando el patrón MVC ha sido el principal eje de todo, puesto que en la vida profesional **muchos frameworks** utilizan este patrón, por ejemplo **(Laravel, Symfony, etc..).** Aunque es verdad que ha sido tedioso al principio el uso y manejo del MVC, he podido adaptarme lo mayor posible a él para poder entender el funcionamiento que tiene su estructura **(Controladores, Vistas, Modelos).**

En cuánto al trabajo futuro la verdad que hay mucho por hacer, lo principal es **implementar un sistema de Roles**, como dije anteriormente el usuario normal puede añadir, editar y consultar, pero eliminar solo lo podrán hacerlo los administradores, al igual que a la hora de registrar los usuarios.

También se realizarán mejoras visuales y mejoras en el código.

Controlar de manera más detallada los fallos que comete el usuario y notificárselo.

Implementar los **mecanismos de protección** para evitar el uso indebido de la aplicación por personas no autorizadas, y a su vez mantener la seguridad dentro de la propia aplicación **evitando las inyecciones HTML o SQL.**

Acceso al Proyecto

Para poder acceder al proyecto desarrollado se proporcionan los siguientes enlaces:

- **Código en GitHub**
    - https://github.com/iherfue/DSW-2018/tree/master/proyecto_dsw_mvc
- **Servidor Web**
    - http://51.254.116.159/proyecto_dsw_mvc/
- **PHPDocumentor**
    - http://51.254.116.159/phpdocumentor_ivan/