



TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

Desarrollo de una aplicación web para la gestión y  
administración de salas de cine

---

**Autor**

Sergio Azañón Cantero

**Tutor**

Juan Manuel Fernández Luna



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, marzo de 2021





# Desarrollo de una aplicación web para la gestión de salas de cine

---

**Autor**

Sergio Azañón Cantero

**Tutor**

Juan Manuel Fernández Luna

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL



# **Desarrollo de una aplicación web para la gestión de salas de cine**

Sergio Azañón Cantero

**Palabras clave:** cine, aplicación web, WEB, reserva, entradas, películas, eventos, salas de cine

## **Resumen**

En el presente Trabajo de Fin de Grado se expone el proyecto de desarrollo de una aplicación web cuyos objetivos principales son la administración de varios cines en una sola plataforma con sus respectivas salas de cine y reserva de entradas para la proyección de sesiones de películas o eventos, ya sean conciertos, conferencias o espectáculos variados. Además la plataforma contará con varias funcionalidades como son: información detallada de las películas o eventos pudiendo interactuar con ellos registrándose como usuario en la plataforma, búsqueda de dichas películas, zona para el perfil de usuario pudiendo visualizar sus datos, entradas compradas o cupones disponibles y formulario de contacto para realizar diferentes procedimientos.

Para la realización de este trabajo se han seguido una serie de fases, comenzando con desempeñar una planificación que incluye una estimación temporal del tiempo que se llevará para cada tarea, continuando con una fase de análisis de requisitos con sus respectivos diagramas asociados, siguiendo con la fase de diseño para proponer que arquitectura de software se llevará a cabo y qué diseño de la base de datos se usará y por último las fases de implementación y pruebas que explicarán el proceso para conseguir los objetivos propuestos y comprobar su funcionamiento respectivamente.

La parte visual de la aplicación web será fundamental para que el usuario tenga una experiencia positiva e intuitiva cuando utilice la plataforma. La reserva de entradas se hará de forma interactiva, pudiendo el usuario elegir sus asientos visualizando la estructura de la sala y sus asientos. Además la generación de entradas con sus respectivos datos y código QR asociado las tendrá disponibles en su perfil. Para el usuario administrador, la administración de nuevos cines, sesiones o películas se harán a través de formularios lo más simples posibles para que reducir la complejidad de este proceso, así como la asignación automática de las nuevas sesiones que se añadan a las salas disponibles.



# **Development of a web application for the management of movie theaters**

Sergio, Azañón Cantero

**Keywords:** cinema, web application, booking, tickets, films, events, movie theathers

## **Abstract**

In this Final Degree Project, the development project of a web application is exposed whose main objectives are the administration of several cinemas on a single platform with their respective movie theaters and reservation of tickets for the projection of movie sessions or events. , whether they are concerts, conferences or various shows. In addition, the platform will have several functionalities such as: detailed information on the films or events being able to interact with them by registering as a user on the platform, searching for said films, an area for the user profile being able to view their data, purchased tickets or available coupons and contact form to perform different procedures.

To carry out this work, a series of phases have been followed, starting with carrying out a planning that includes a temporary estimate of the time that will take for each task, continuing with a requirements analysis phase with its selected associated diagrams, continuing with the design phase to propose what software architecture will be carried out and what database design will be used and finally the implementation and testing phases that will explain the process to achieve the proposed objectives and check its operation respectively.

The visual part of the web application will be essential for the user to have a positive and intuitive experience when using the platform. The reservation of tickets will be done interactively, the user being able to choose their seats by viewing the structure of the hall and its seats. In addition, the generation of tickets with your own data and associated QR code will have them available in your profile. For the administrator user, the administration of new c, sessions or movies will be done through the simplest possible forms to reduce the complexity of this process, as well as the automatic assignment of new sessions that are added to the available rooms.

---

Yo, **Sergio Azañón Cantero**, alumno de la titulación INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76589409F, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Sergio Azañón Cantero

Granada a 23 de marzo de 2021 .

---

**D. Juan Manuel Fernández Luna**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado *Desarrollo de una aplicación web para la gestión de salas de cine*, ha sido realizado bajo su supervisión por **Sergio Azañón Cantero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 10 de noviembre de 2021 .

**Los directores:**

**Juan Manuel Fernández Luna**

# **Agradecimientos**

A mi familia y amigos por el apoyo en todo el camino recorrido.

# Índice general

<b>1. Introducción</b>	<b>14</b>
1.1. Motivación del trabajo . . . . .	14
1.2. Objetivos del proyecto . . . . .	15
1.3. Estado del arte . . . . .	15
1.3.1. Aplicaciones actuales . . . . .	16
1.3.2. Crítica al estado del arte . . . . .	16
<b>2. Planificación</b>	<b>18</b>
2.1. Desarrollo del software . . . . .	18
2.2. Planificación temporal . . . . .	20
2.3. Presupuesto . . . . .	21
<b>3. Análisis de requisitos</b>	<b>23</b>
3.1. Introducción . . . . .	23
3.2. Requisitos funcionales . . . . .	24
3.3. Requisitos no funcionales . . . . .	26
3.4. Requisitos de información . . . . .	27
3.5. Descripción de casos de uso . . . . .	27
3.6. Diagramas de casos de uso . . . . .	40
3.6.1. Identificación de usuario . . . . .	40
3.6.2. Comprar entrada/s online . . . . .	40
3.6.3. Comprar entrada/s en taquilla . . . . .	41
3.6.4. Gestionar entradas compradas . . . . .	42
3.6.5. Gestionar comentarios . . . . .	42
3.6.6. Funciones del usuario administrador . . . . .	43
3.7. Diagramas de secuencia . . . . .	44
3.7.1. Registro de un usuario . . . . .	44
3.7.2. Login de un usuario . . . . .	45
3.7.3. Olvidar la contraseña . . . . .	46
3.7.4. Eliminar usuario . . . . .	47
3.7.5. Añadir evento, cine y sesión . . . . .	48
3.7.6. Reserva de entradas . . . . .	51

<b>4. Diseño</b>	<b>52</b>
4.1. Base de datos . . . . .	52
4.1.1. Diagrama entidad-relación . . . . .	52
4.1.2. Paso a tablas y normalización de la base de datos . . . . .	56
4.2. Arquitectura software . . . . .	57
4.3. Interfaz de usuario (Mockups) . . . . .	58
4.3.1. Página principal . . . . .	58
4.3.2. Formulario para añadir evento, cine y sesión . . . . .	60
4.3.3. Cartelera . . . . .	61
4.3.4. Proceso de reserva . . . . .	62
4.3.5. Todos las películas/eventos . . . . .	63
4.3.6. Todos los usuarios . . . . .	64
4.3.7. Log del sistema . . . . .	64
4.3.8. Mi perfil y entradas . . . . .	65
4.3.9. Detalles de película/evento . . . . .	67
4.3.10. Detalles de película/evento . . . . .	68
<b>5. Implementación</b>	<b>69</b>
5.1. Entorno de desarrollo . . . . .	69
5.2. Diagrama de clases . . . . .	71
5.3. Base de datos . . . . .	74
5.3.1. Implementación de la base de datos . . . . .	74
5.3.2. Creación de entidades y sus relaciones en <i>Symfony</i> . . . . .	78
5.4. Desarrollo de aplicación web . . . . .	80
5.4.1. Gestión de usuarios . . . . .	80
5.4.2. Asignación de salas para sesiones . . . . .	80
5.4.3. Proceso de reserva de entradas . . . . .	82
5.4.4. The Movie Database API . . . . .	89
5.4.5. Paginación de resultados . . . . .	90
<b>6. Pruebas</b>	<b>91</b>
6.1. Caja blanca . . . . .	91
6.2. Caja negra . . . . .	91
6.3. Pruebas en <i>Symfony</i> . . . . .	91
<b>7. Conclusiones</b>	<b>97</b>
7.1. Cronología real de desarrollo . . . . .	97
7.2. Objetivos cumplidos . . . . .	98
7.3. Trabajos futuros . . . . .	99
<b>8. Anexos</b>	<b>100</b>
8.1. Manual de usuario . . . . .	100
8.2. Código fuente . . . . .	119
<b>Bibliografía</b>	<b>122</b>

# Índice de figuras

2.1. Modelo de cascada en el desarrollo de software . . . . .	19
2.2. Planificación inicial del proyecto . . . . .	20
3.1. Diagrama de identificación de usuarios . . . . .	40
3.2. Diagrama de compra de entradas online . . . . .	41
3.3. Diagrama de compra de entradas en taquilla . . . . .	41
3.4. Diagrama de gestión de entradas compradas . . . . .	42
3.5. Diagrama de gestión de comentarios . . . . .	42
3.6. Diagrama de funciones de usuario administrador . . . . .	43
3.7. Diagrama de secuencia de registro de usuario. . . . .	44
3.8. Diagrama de secuencia para logearse en el sistema. . . . .	45
3.9. Diagrama de secuencia de olvidar la contraseña. . . . .	46
3.10. Diagrama de secuencia de eliminación de usuario. . . . .	47
3.11. Diagrama de secuencia para añadir un evento. . . . .	48
3.12. Diagrama de secuencia para añadir un cine. . . . .	49
3.13. Diagrama de secuencia para añadir una sesión. . . . .	50
3.14. Diagrama de secuencia para reservar entradas. . . . .	51
4.1. Diagrama E-R . . . . .	55
4.2. Paso a tablas . . . . .	56
4.3. Diagrama del patrón MVC.[1] . . . . .	57
4.4. Mockup de la página principal . . . . .	59
4.5. Mockup del formulario para añadir un evento . . . . .	60
4.6. Mockup del formulario para añadir un cine . . . . .	60
4.7. Mockup del formulario para añadir una sesión . . . . .	61
4.8. Mockup de la cartelera . . . . .	61
4.9. Mockup del formulario para elegir los asientos . . . . .	62
4.10. Mockup de la confirmación de la reserva de entradas . . . . .	63
4.11. Mockup de todas las películas . . . . .	63
4.12. Mockup de todos los usuarios . . . . .	64
4.13. Mockup del log del sistema . . . . .	65
4.14. Mockup del perfil de un usuario . . . . .	65
4.15. Mockup de las entradas de un usuario . . . . .	66
4.16. Mockup de los detalles de un evento/película . . . . .	67
4.17. Mockup del formulario de contacto . . . . .	68

5.1. PHP, Symfony y MySQL . . . . .	70
5.2. HTML, CSS, Javascript y Bootstrap . . . . .	70
5.3. Diagrama UML de clases . . . . .	72
5.4. Ayuda para indicar de qué tipo es el atributo . . . . .	78
5.5. Ayuda para saber qué tipo de relación tiene . . . . .	79
6.1. Guía para crear un <i>test</i> en <i>Symfony</i> . . . . .	92
6.2. Ejecución correcta de ambos <i>tests</i> . . . . .	93
6.3. Ejecución fallida de uno de los <i>tests</i> . . . . .	94
7.1. Cronología real del proyecto . . . . .	98
8.1. Página de inicio . . . . .	100
8.2. <i>Header</i> sin identificar . . . . .	101
8.3. <i>Header</i> identificado como usuario . . . . .	101
8.4. <i>Header</i> identificado como admin . . . . .	101
8.5. Opciones de administrador . . . . .	101
8.6. Formulario para identificarse . . . . .	102
8.7. Formulario para recuperar la contraseña . . . . .	102
8.8. Detalles de la película o evento . . . . .	103
8.9. <i>Trailer</i> de película . . . . .	103
8.10. Sección de comentarios (no identificado) . . . . .	104
8.11. Trailer y sección de comentarios . . . . .	104
8.12. Cartelera . . . . .	105
8.13. Pantalla de selección de asientos . . . . .	106
8.14. Pantalla de confirmación de reserva . . . . .	107
8.15. Aplicación de cupón de descuento . . . . .	107
8.16. Ventana <i>modal</i> de todas las opciones de pago . . . . .	108
8.17. Listado de todos los cines . . . . .	108
8.18. Todas las películas . . . . .	109
8.19. Todos los eventos . . . . .	109
8.20. Películas que se estrenarán próximamente	110
8.21. Formulario de contacto . . . . .	110
8.22. Opciones del perfil . . . . .	111
8.23. Vista del perfil de un usuario . . . . .	111
8.24. Vista de todos los cupones de un usuario . . . . .	112
8.25. Vista de todas las entradas de un usuario . . . . .	113
8.26. Formulario para añadir un nuevo cine . . . . .	114
8.27. Formulario para añadir un nuevo evento o película . . . . .	115
8.28. Formulario para añadir una nueva sesión . . . . .	116
8.29. Usuarios del sistema . . . . .	116
8.30. <i>Log</i> del sistema . . . . .	117
8.31. Vista de la página principal desde las dimensiones de un móvil . . . . .	118
8.32. Paginación de un listado . . . . .	118

# Índice de cuadros

2.1. Desglose de coste mensual para empresario y salario del trabajador . . . . .	21
2.2. Desglose de gastos de ejecución . . . . .	22
3.1. Descripción caso de uso CU-1 . . . . .	28
3.2. Descripción caso de uso CU-2 . . . . .	28
3.3. Descripción caso de uso CU-3 . . . . .	29
3.4. Descripción caso de uso CU-4 . . . . .	29
3.5. Descripción caso de uso CU-5 . . . . .	30
3.6. Descripción caso de uso CU-6 . . . . .	30
3.7. Descripción caso de uso CU-7 . . . . .	31
3.8. Descripción caso de uso CU-8 . . . . .	31
3.9. Descripción caso de uso CU-9 . . . . .	32
3.10. Descripción caso de uso CU-10 . . . . .	32
3.11. Descripción caso de uso CU-11 . . . . .	33
3.12. Descripción caso de uso CU-12 . . . . .	33
3.13. Descripción caso de uso CU-13 . . . . .	34
3.14. Descripción caso de uso CU-14 . . . . .	34
3.15. Descripción caso de uso CU-15 . . . . .	35
3.16. Descripción caso de uso CU-16 . . . . .	35
3.17. Descripción caso de uso CU-17 . . . . .	36
3.18. Descripción caso de uso CU-18 . . . . .	36
3.19. Descripción caso de uso CU-19 . . . . .	37
3.20. Descripción caso de uso CU-20 . . . . .	37
3.21. Descripción caso de uso CU-21 . . . . .	38
3.22. Descripción caso de uso CU-22 . . . . .	38
3.23. Descripción caso de uso CU-23 . . . . .	39

# Capítulo 1

## Introducción

### 1.1. Motivación del trabajo

Las salas de cine llevan existiendo desde 1899, año el cual se creó el primer cine del mundo cerca de la ciudad de Marsella. Desde entonces los cines han ido evolucionando sus proyecciones, el uso de sus salas y su gestión para la venta de entradas. Antes de la llegada de Internet, para ver una película en casa a través de VHS o DVD recién estrenada en los cines se tenía que esperar una media de 2 años y medio, por lo que la afluencia de gente que asistía a las salas de cine era muy alta.

La llegada de Internet lo ha cambiado todo, para una sesión de cine pueden reservarse las entradas a través de la propia web del cine eligiendo de forma interactiva los asientos y visualizando cuáles han sido ya ocupados. También han surgido plataformas donde poder ver una gran cantidad de películas por un precio mensual, lo que ha hecho mucha competencia a los cines de hoy en día.

Por si lo anteriormente mencionado fuera poco, tras la pandemia vivida del COVID-19, muchas salas de cine se vieron obligadas a cerrar tras dos meses de cierre total y restricciones de aforo. La AIMC (Asociación para la Investigación de Medios de Comunicación) ha publicado la 23<sup>a</sup> edición de su Censo de Salas de Cine [2] en el cual indica que los locales de cine han pasado de ser 723 en España a 706. También han disminuido las salas de 3593 a 3563 y las butacas totales por sala de 786475 a 752291.

Tras los datos comentados anteriormente, las salas de cine han necesitado reinventarse y ofrecer otro tipo de eventos que no son proyecciones de películas como son conciertos, congresos, monólogos o presentaciones de productos para su venta.

Existen muchas plataformas para gestionar entradas y poder reservar asientos para ver una película o alguno de los eventos comentados anteriormente, cada sala de cine suele tener su propia web desde la que se pueden realizar este tipo de gestiones y dan información sobre ella.

También han surgido plataformas de valoraciones y críticas sobre películas en las que personas con conocimientos cinematográficos dan su opinión y los usuarios pueden comentar las *reviews*. Dichas plataformas te ofrecen todo tipo de detalles de las distintas películas y puedes acceder a detalles muy específicos de forma muy sencilla.

¿Por qué no juntar ambas ideas en una plataforma web común donde cada sala se añada con sus datos y el cliente pueda cambiar de cine y tener toda la información acerca

de las reservas, películas y *reviews*?

Ante los motivos comentados anteriormente, surge la motivación personal de desarrollar este proyecto y ser capaz de desarrollar una aplicación web totalmente capaz de realizar las ideas anteriormente descritas: reserva de entradas de cine, mostrar información detallada sobre las películas, administración de diferentes cines, sesiones, salas y gestión de los usuarios de la plataforma.

## 1.2. Objetivos del proyecto

El objetivo principal de este trabajo fin de grado es el desarrollo de una solución para la administración de diferentes cines y sus respectivas salas. De dicho objetivo general surgen varios objetivos específicos que deben ser realistas, medibles y alcanzables o realizables:

- Gestión de usuarios y eventos: El sistema deberá ser capaz de realizar una gestión adecuada para los usuarios y eventos almacenados en el sistema.
- Información accesible: El sistema deberá cumplir que todo usuario sea capaz de acceder a la información pública como son los detalles de cada evento o película, las sesiones, los distintos cines o la información de su perfil.
- Sostenibilidad de usuarios administradores: El sistema deberá garantizar que los usuarios administradores sean los únicos que puedan añadir, editar y eliminar cines, eventos o sesiones
- Asignación de salas a sesiones: El sistema tendrá la capacidad para asignar automáticamente una sala a una sesión añadida en una hora determinada siempre y cuando sea posible en base a la disponibilidad de dicha sala.
- Gestión de reserva de entradas: El sistema deberá ser capaz de ofrecer una reserva de entradas interactiva para una sesión determinada a todos los usuarios y evitar posibles concurrencias en dicho proceso.
- Gestión de cupones y entradas asociadas a un usuario: El sistema será capaz de ofrecer a cada usuario sus entradas compradas o sus cupones disponibles para su uso.
- Aprendizaje de nuevas tecnologías: Aprender ciertas nuevas tecnologías, lenguajes y *frameworks* como son *Symfony*, *Twig*, *PHPUnit* o *Javascript*.

## 1.3. Estado del arte

En esta sección se quiere hablar sobre el estado del arte, en este caso, es una compilación de investigaciones realizadas sobre otras aplicaciones webs actuales enfocadas al mismo tema del proyecto, es decir, dedicadas para la reserva de entradas de cine o la interacción (ya sea valoración, descripción o comentarios de usuarios) con eventos y películas a través de portales webs.

### 1.3.1. Aplicaciones actuales

Hay muchas aplicaciones webs para la reserva de entradas de cine, ya que cualquier cine actual que tenga una demanda mínimamente grande, necesita que los usuarios no necesiten desplazarse físicamente a comprar entradas y puedan visualizar el horario y fecha de las sesiones. Por lo que se ha investigado cines cercanos en la ciudad como son:

- **Kinepolis:** Es una cadena de cines muy conocida en toda España, tienen sede en Madrid, Valencia y Granada, un total de siete cines. No sólo ofrecen películas en sus salas, si no que también organizan eventos como son conciertos de ópera, monólogos o cenas de empresa. Tienen una aplicación web donde además de poder elegir el cine y día para reservar entradas de cine de forma interactiva, puedes buscar eventos, actores o películas para ver más detalles como pueden ser imágenes, trailers, actores que participaron o las sesiones activas de dicha película. [3]
- **IMDB:** Es uno de los portales webs más famosos del mundo, ya que es la base de datos en línea más grande que existe, sus siglas son Internet Movie Database. Lleva desarrollándose desde principio de los 90 y cuenta con toda la información detallada sobre películas, series, actores, productores, directores, etc. Además se pueden consultar noticias actuales relacionadas con el mundo del cine, valoraciones de expertos sobre las películas, sesiones de cines cercanos a tu localización y comentar películas. Una de las grandes funcionalidades para desarrolladores webs que tiene IMDB es la API que se puede utilizar por todos los usuarios.[4]
- **Cinesa:** Es otra de las cadenas de cines más conocida en España, cuenta con más de 30 cines en todo el país, de los cuales 6 de ellos son *deluxe* ya que cuentan con sillones especiales reclinables y una tecnología especial llamada *ScreenX* que consiste en unas pantallas expandidas, de doble cara y 270 grados proyectadas en las paredes del teatro. Al tener una cantidad de cines tan elevada, necesitan una web para centralizar las películas y dependiendo del cine elegido mostrar unas sesiones u otras. Para reservar entradas para una sesión de cine, necesitas estar autenticado en su web, además puedes comprar la comida y bebida junto con la entrada en la web. [5]

### 1.3.2. Crítica al estado del arte

Todas estas aplicaciones tienen apartados comunes como es la consulta de detalles de la película o evento, aunque en algunas webs los detalles sean mínimos o la posibilidad de reservar entradas para una sesión de cine de forma interactiva, pero la mezcla de ellas sería la aplicación ideal. A continuación comentaré las desventajas de cada una de las tres aplicaciones mencionadas anteriormente:

- **Kinepolis:** Uno de los mayores inconvenientes que tiene la aplicación web es su interfaz, cuenta con muchos anuncios a ambos lados que dificultan la interacción con la aplicación y navegar de un menú a otro. Los detalles que muestran de las películas son mínimos y tiene un error de visualización al deslizar las imágenes de las películas. Además el estilo de la web lleva sin actualizarse más de cinco años.

- **IMDB:** Esta es la aplicación web más completa y que menos inconvenientes tienen. Lleva desarrollándose desde principios de los 90 por lo que no han dejado de mejorar su aplicación web. El inconveniente más grande que tiene es que solo puedes consultar sesiones de películas de cines muy concretos y la visualización de dicha página es muy básica. Como es obvio, no se pueden reservar asientos de películas a través de IMDB.
- **Cinesa:** En el panorama nacional, quizá es la mejor aplicación web para reserva de entradas de cine ya que su interfaz para reservarlo es muy intuitiva y cómoda de usar pero aún así tiene varios inconvenientes. Uno de ellos es que no tiene buscador de películas o eventos por lo que no sirve para consultar detalles de películas. Tampoco se pueden reservar entradas si no estás identificado en el sistema, por lo que te obliga a crearte una cuenta para poder comprar entradas.

## Capítulo 2

# Planificación

### 2.1. Desarrollo del software

El proceso de desarrollo de un software (ó ciclo de vida del desarrollo de software), es una metodología que se creó para aplicarla en la realización del desarrollo de un determinado producto de software. Existen varios modelos para representar de forma abstracta el enfoque que se le quiere dar al proyecto y puesta en común de los equipos de desarrolladores con los clientes. Los modelos más conocidos son: modelo de cascada, modelo de espiral, desarrollo iterativo y desarrollo ágil. De entre todos los modelos, el modelo de cascada encaja con el proyecto ya que se ha seguido una estructura secuencial y se sabe desde el principio el objetivo final del proyecto. En este modelo no se puede empezar una etapa sin haber terminado la anterior, únicamente puede hacerse la comprobación de la etapa anterior. Las diferentes etapas que tendrá el modelo de cascada son[6]:

1. **Planificación:** es la etapa inicial cuando se empieza un proyecto. Se estiman los recursos que se van a utilizar , el coste estimado y la realización de una planificación temporal estimada de la duración de cada etapa del desarrollo. Se definen requisitos y reunión con los desarrolladores y clientes fundamentalmente.
2. **Análisis de requisitos:** esta etapa puede formar parte de la etapa anterior de planificación ya que se definen los requisitos que tendrá el producto software y el cliente deberá informar que funciones quiere que realice el producto.
3. **Diseño:** recoge todos los requisitos recogidos anteriormente y se encuentra una solución para ellos basándonos en la arquitectura del software y sus componentes.
4. **Implementación:** Es la ejecución de todo el diseño realizado en la etapa anterior, incluye programación, búsqueda de errores y pruebas unitarias. En la fase de implementación, el proyecto de software se traduce al correspondiente lenguaje de programación. Los diversos componentes se desarrollan por separado, se comprueban a través de las pruebas unitarias y se integran para el producto final [7]
5. **Verificación:** El producto se verifica en una versión beta con un entorno de desarrollo, si pasa las pruebas en dicho entorno, estará listo para su lanzamiento.

6. **Mantenimiento:** Es la última fase del modelo, se autoriza el lanzamiento del producto e incluye la entrega, el mantenimiento y la mejora del software.

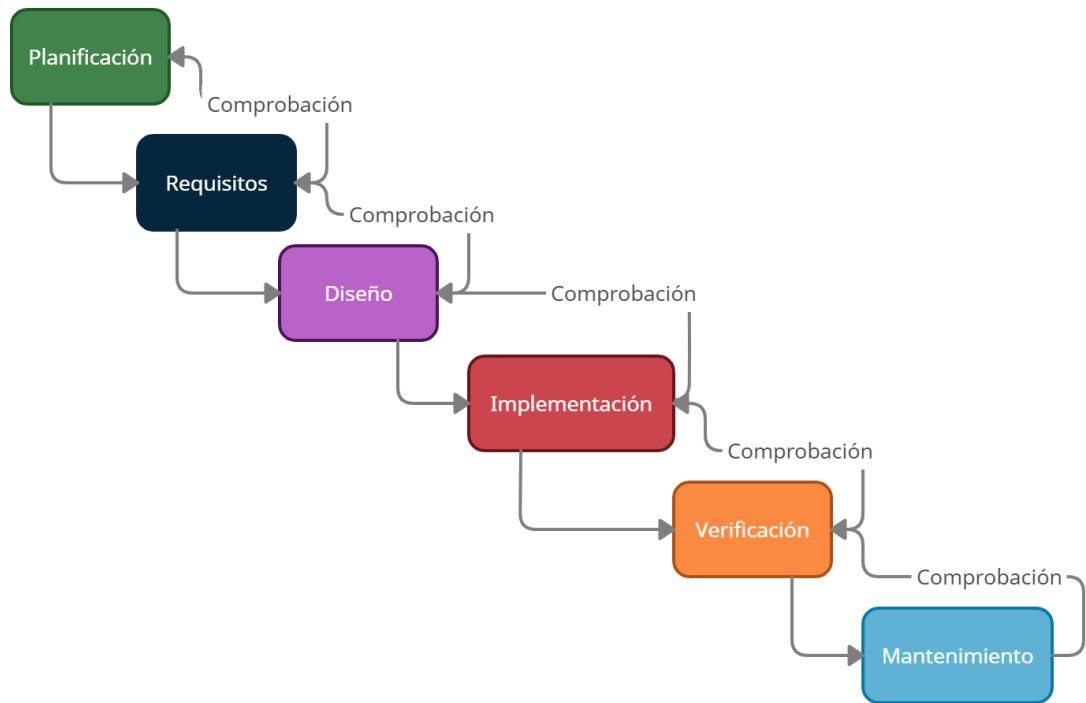


Figura 2.1: Modelo de cascada en el desarrollo de software

## 2.2. Planificación temporal

En esta sección se muestra la planificación inicial estimada para todas las fases del proyecto. Esto es muy útil tanto para tener una organización en el tiempo de lo que debería durar cada etapa como para la comparación con la planificación final y comparar las fases que han requerido menos y más tiempo.

Esta planificación se ha realizado mediante un diagrama de Gantt, este diagrama sirve para mostrar de forma gráfica la previsión del tiempo dedicado a las diferentes tareas (en este caso fases) a lo largo del tiempo, ya sean días, semanas o en este caso meses. Se ha dividido en siete grandes bloques correspondiendo principalmente así al modelo de cascada expuesto anteriormente. Como se puede ver, los dos bloques que mas tiempo requerirán como es lógico son la implementación de la aplicación y la redacción de la memoria.

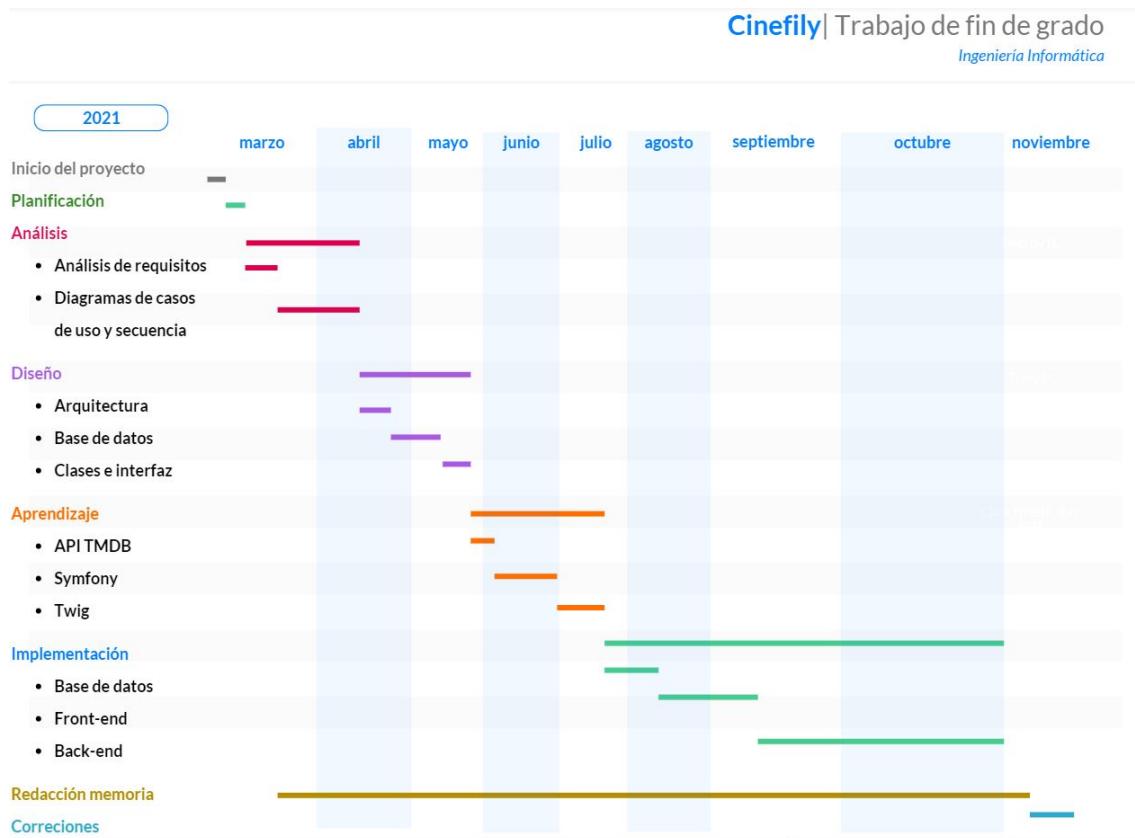


Figura 2.2: Planificación inicial del proyecto

## 2.3. Presupuesto

Cualquier proyecto tiene asociado un coste mayor o menor dependiendo del personal, el equipamiento, el tiempo requerido y la dificultad y los conocimientos técnicos que son necesarios. A continuación se hará un desglose de los gastos que implican realizar un proyecto como este.

Empezando por la contratación del personal, en este caso una sola persona, el salario mínimo para un programador web y de nivel junior tiene un rango entre 18000€ y 20000€ brutos anuales con jornada laboral completa. Si establecemos un salario de 19000€ brutos anuales, la cifra mensual es de 1600€ brutos, si el proyecto va a requerir 8 meses, serían 12700€ brutos. El salario neto sería de 1306.75€, a continuación se desglosarán todos los detalles en la siguiente tabla. [8] [9]

Detalles	Cantidad
<b>Coste mensual para el empresario</b>	
IRPF	3354.90€
Cotizaciones, enfermedad profesional y accidentes de trabajo	14 %
Coste seguridad social empresa	30 %
Coste seguridad social empleado	1257.90€
Salario a pagar tras retención	133.35€
Base de cotización	1672.65€
	2100.00€
<b>Salario bruto anual del contratado</b>	
Retenciones por IRPF	19000€
Cuotas a la Seguridad Social	2112.8€
Tipo de retención sobre la nómina	1206.5€
Categoría profesional	11.12 %
Edad	Ingenieros y licenciados
SALARIO NETO MENSUAL (12 PAGAS)	23 años
	1306.7€
<b>Coste total (8 pagas)</b>	<b>26832€</b>

Cuadro 2.1: Desglose de coste mensual para empresario y salario del trabajador

Tras calcular el coste mensual del personal que se tiene que contratar, se deben conocer los gastos de ejecución (alquiler, facturas, materiales). De los materiales que se incluyen en los gastos hay que diferenciar dos tipos de materiales: los materiales fungibles y los materiales inventariables. [10] Se considera inventariable todo bien perdurable en el tiempo, de carácter material e inmaterial, cuyo coste de adquisición sea igual o superior a 300,50 € (si es de rápido deterioro se podrá considerar fungible). Se considera material fungible a aquellos productos en los cuales no se puede hacer un uso adecuado a su naturaleza sin que se consuman, es decir, sin posibilidad de reutilizarse.

Descripción	Unidades	Precio	I.V.A (21 %)
<b>Material inventariable</b>			
Ordenador portátil	1	750€	157.5€
Móvil	1	300€	63€
Ratón	1	40€	8.4€
Monitor	1	200€	42€
TOTAL		1290€	
<b>Material fungible</b>			
Folios	1	8€	1.68€
Bolígrafos	1	4€	0.84€
Grapas	1	5€	1.05€
TOTAL		17€	
<b>Alquileres</b>			
Alquiler local	8	2400€	504€
Electricidad y agua	8	960€	201.6€
Internet	8	240€	50.4€
Dominio	1	20€	4.2€
Limpieza	8	250€	52.5€
TOTAL		3870€	
GASTO DE EJECUCIÓN TOTAL		5177€	
<b>GASTO DE TOTAL (EJECUCIÓN Y PERSONAL)</b>		<b>32009€</b>	

Cuadro 2.2: Desglose de gastos de ejecución

## Capítulo 3

# Análisis de requisitos

### 3.1. Introducción

El análisis de requisitos se define como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, de hardware o de software, así como el proceso de estudio y refinamiento de dichos requisitos. [11]

Es una parte importante a la hora de desarrollar un software desde cero ya que nos permite modelar y especificar los requisitos de un sistema. En esta parte, tanto el cliente como el desarrollador o desarrolladores (también pueden ser analistas de software) necesitan estar en contacto para que el cliente transmita lo que el software quiere que haga. Muchas de las veces suele haber ambigüedad ya que el cliente no posee los conocimientos de desarrollo de una aplicación y el desarrollador no entiende correctamente lo que el cliente quiere expresar, por eso, este es un proceso iterativo que implica más de una reunión ya que partimos de algo incompleto y con poca claridad que evoluciona en una especificación completa y clara.

Las siete propiedades principales de los requisitos son[12]:

- **Completos:** Todos los aspectos del sistema están representados en el modelo de requisitos.
- **Consistentes:** Los requisitos no se contradicen entre sí.
- **No ambiguos:** No es posible interpretar los requisitos de dos o más formas diferentes.
- **Correctos:** Representan exactamente el sistema que el cliente necesita y que el desarrollador construirá.
- **Realistas:** Los requisitos se pueden implementar con la tecnología y presupuesto disponible.
- **Verificables:** Se pueden diseñar pruebas para demostrar que el sistema satisface los requisitos.
- **Trazables:** Cada requisito puede rastrearse a través del desarrollo del software hasta su correspondiente funcionalidad del sistema.

En este proyecto, al no haber un cliente específico, cliente y desarrollador serán la misma persona. Aunque no se debe perder la perspectiva que tiene un cliente en este tipo de análisis.

### 3.2. Requisitos funcionales

Los requisitos funcionales se definen como cualquier actividad que el sistema debe realizar, es decir, el comportamiento de un software cuando se cumplen ciertas condiciones. Describe la interacción entre el sistema y su entorno, proporcionando servicios que proveerá el sistema o indicando la manera en que éste reaccionará ante determinados estímulos.

Este proyecto tendrá los siguientes requisitos funcionales:

- RF-1.** El sistema permitirá a cualquier usuario elegir un cine de los disponibles.
- RF-2.** El administrador podrá añadir o eliminar un cine/recinto al sistema
- RF-3.** El administrador podrá añadir o modificar nuevos horarios a las películas o eventos de cada cine.
- RF-4.** El administrador podrá añadir o modificar películas/eventos del sistema
- RF-5.** El sistema deberá mostrar todas las películas disponibles de la semana actual y la siguiente en el cine seleccionado.
- RF-6.** Un usuario registrado o sin registrar podrá reservar entradas para una película o evento del cine seleccionado.
  - RF-6.1** La reserva se hará de forma interactiva, así el usuario podrá visualizar que asientos están libres y cuáles no.
  - RF-6.2** Al elegir los asientos, el usuario deberá introducir sus datos si no está registrado o aparecerán sus datos ya llenos si lo está.
  - RF-6.3** Podrá utilizar un cupón de descuento si lo tiene.
- RF-7.** El usuario tendrá la opción de pagar las entradas mediante PayPal o tarjeta de crédito.
- RF-8.** El sistema permitirá que el usuario que ha comprado entradas, pueda modificarlas o cancelarlas si faltan 5 días o más para el inicio de la película o evento.
- RF-9.** Un usuario puede comprar un bono de 5, 10 o 15 entradas para películas, reduciendo así el precio por entrada.
- RF-10.** Un usuario podrá registrarse en el sistema si aún no lo ha hecho.
  - RF-10.1** El registro requerirá un nombre de usuario, una contraseña y un email.
  - RF-10.2** Opcionalmente podrá añadir su nombre y apellidos, una foto de perfil y un teléfono.

- RF-11.** El usuario registrado podrá identificarse en el sistema con sus datos de registro
- RF-12.** El usuario registrado podrá recuperar su contraseña introduciendo su email asociado a su usuario en el sistema y enviándole un email con la nueva contraseña
- RF-13.** El usuario registrado podrá modificar sus datos de perfil siempre que lo desee.
- RF-14.** El usuario registrado podrá visualizar sus últimas reservas de entradas y los cupones que tiene disponibles.
- RF-15.** El usuario registrado podrá añadir o eliminar comentarios a las películas o eventos del sistema.
- RF-16.** El usuario registrado podrá añadir películas o eventos a una lista personal de favoritas.
- RF-17.** El usuario registrado podrá solicitar el alquiler de una sala completa para un evento o congreso.
- RF-18.** El usuario registrado podrá eliminar su cuenta del sistema.
- RF-19.** Un usuario podrá consultar al sistema todos los datos relacionados de una película o evento disponible.
- RF-19.1** Consulta de una película: foto de la portada, tráiler, director, actores principales, género, duración, valoración, año de estreno, precio y horarios disponibles.
- RF-19.2** Consulta de un evento: descripción, protagonistas del evento (ya sea cantantes, empresarios, actores de teatro...etc), día/s que se impartirá dicho evento, precio, género, e imágenes.
- RF-20.** El sistema enviará un correo electrónico cuando se registre alguna de las siguientes acciones: registro de un usuario, consiga un cupón de descuento, recuperación de contraseña, emisión de una factura tras el pago de una/s entradas y entradas de la película/evento con su respectivo código QR.
- RF-21.** Se mostrará la ubicación del cine seleccionado con un mapa interactivo.
- RF-22.** El sistema generará un cupón de descuento a un usuario tras comprar entradas a cinco películas o eventos diferentes.
- RF-23.** El sistema tendrá cuatro tipos de usuarios diferentes según sus permisos para añadir, editar o eliminar: administradores y usuarios.
- RF-24.** El sistema contendrá widgets que informarán sobre las películas mejor valoradas del último mes, últimas películas o eventos visitados y películas más comentadas.
- RF-25.** El sistema contará con un apartado para que el usuario pueda consultar sus cupones de descuento para entradas con sus respectivos detalles sistema.

- RF-26.** El sistema tendrá un apartado de opiniones y solicitudes para que cualquier usuario pueda dar una opinión, sugerencia sobre la plataforma o solicitar el alquiler una sala de cine para algún evento.

### 3.3. Requisitos no funcionales

Describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo.

Este proyecto tendrá los siguientes requisitos no funcionales:

#### Arquitectura

- RNF-1.** La aplicación web debe poder utilizarse correctamente empleando cualquier navegador web.
- RNF-2.** Los datos de la aplicación deberán estar almacenados en un sistema gestor de bases de datos, los demás datos que no se almacenen se recuperarán de APIs externas.

#### Fiabilidad

- RNF-3.** Ante un posible fallo del sistema, no debe perderse la información previamente almacenada, por lo que se deberán hacer copias de seguridad periódicamente.

#### Rendimiento

- RNF-4.** El sistema deberá tener un tiempo máximo de respuesta de 5 segundos para cualquier operación de consulta.
- RNF-5.** Varios administradores podrán trabajar simultáneamente con el sistema, introduciendo información, así como varios usuarios podrán consultar al mismo tiempo datos almacenados en el mismo.

#### Seguridad

- RNF-6.** Es necesario controlar que solo las personas autorizadas (administradores) puedan acceder a las funciones de gestión del sistema e introducir información en él.

#### Físicos

- RNF-7.** El sistema necesita un servidor para almacenar la base de datos, imágenes, un back-end y un sistema de monitorización.

### 3.4. Requisitos de información

Describen necesidades de almacenamiento de información en el sistema.

#### **RI-1** Cuenta de usuario

Información sobre el usuario registrado

**Contenido:** nombre, primer apellido, segundo apellido, nombre de usuario, email, teléfono, foto de perfil, cupones.

**Requisitos asociados:** RF-5, RF-8, RF-9, RF-9.1, RF-9.2, RF-10, RF-11, RF-12, RF-13, RF-14, RF-15, RF-20.

#### **RI-2** Reserva de entrada

Información sobre la compra de una entrada para una película o evento.

**Contenido:** id, tipo de entrada, precio, usuario asociado (puede no ser ninguno), código QR, nombre de la película/evento, día, fecha, hora, precio, fila, asiento.

**Requisitos asociados:** RF-5, RF-5.1, RF-5.2, RF-5.3, RF-7, RF-8, RF-11, RF-17

#### **RI-3** Película/Evento

Información sobre una película/evento.

**Contenido:** id, nombre, sinopsis, portada, tráiler, director, actores principales, género, duración, valoración, comentarios y año de estreno.

**Requisitos asociados:** RF-4, RF-12, RF-16, RF-16.1, RF-16.2

#### **RI-4** Cine

Información sobre el cine

**Contenido:** id, nombre, ubicación, películas/eventos que tiene en ese momento, horarios de las películas/eventos, horario de apertura y cierre del cine.

**Requisitos asociados:** RF-1, RF-2, RF-3, RF-4, RF-18

### 3.5. Descripción de casos de uso

En esta sección se describirá cada caso de uso individualmente usando la plantilla básica donde se indica el caso de uso, el actor que interacciona, el tipo, requisitos funcionales implicados, precondiciones y poscondiciones (condiciones que se deben cumplir antes y después del requisito para que pueda realizarse), el propósito y resumen del caso de uso. En relación al tipo de caso de uso, dependiendo de diferentes factores pueden ser[13]:

- **Dependiendo de su importancia:**

- Primarios: Procedimientos comunes más importantes.
- Secundarios: procesos poco comunes, de error o que dan soporte a primarios.

- Opcionales: Pueden no llegar a ser implementados.
- **Dependiendo del procesamiento:**
  - Básico: describe de forma general el procesamiento.
  - Extendido: describe la secuencia de acción completa entre actores y sistema.
- **Dependiendo de su nivel de abstracción:**
  - Esencial: Expresado de forma abstracta, contiene poca tecnología y pocos detalles del diseño.
  - Real: Expresado en base al diseño actual, en el que aparecen relaciones con la Interfaz de Usuario.

<b>Caso de uso</b>		Iniciar sesión CU-1
<b>Actores</b>	Usuario/Administrador	
<b>Tipo</b>	Primario, básico y esencial	
<b>Referencias</b>	RF-11	
<b>Precondición</b>	Estar registrado en el sistema	
<b>Poscondición</b>	Estar identificado en el sistema	
<b>Propósito</b>		
Identificarse en el sistema con los datos del registro		
<b>Resumen</b>		
El usuario se identifica en el sistema con los datos del registro		

Cuadro 3.1: Descripción caso de uso CU-1

<b>Caso de uso</b>		Registrarse CU-2
<b>Actores</b>	Usuario/Administrador	
<b>Tipo</b>	Primario, básico y esencial	
<b>Referencias</b>	RF-10, RF-10.1 y RF-10.2	
<b>Precondición</b>	Rellenar los datos del registro	
<b>Poscondición</b>	Estar registrado en el sistema	
<b>Propósito</b>		
Registrarse en el sistema rellenando todos los datos		
<b>Resumen</b>		
El usuario se registra en el sistema rellenando los datos requeridos en el registro		

Cuadro 3.2: Descripción caso de uso CU-2

<b>Caso de uso</b>	Recuperar contraseña CU-3
<b>Actores</b>	Usuario/Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-12
<b>Precondición</b>	Estar registrado en el sistema
<b>Poscondición</b>	Contraseña actualizada
<b>Propósito</b>	
Recuperar la contraseña olvidada del usuario	
<b>Resumen</b>	
El usuario introducirá su email asociado a su cuenta en el sistema para que se le envie un email con su nueva contraseña	

Cuadro 3.3: Descripción caso de uso CU-3

<b>Caso de uso</b>	Validar email CU-4
<b>Actores</b>	Usuario/Administrador
<b>Tipo</b>	Secundario, extendido y esencial
<b>Referencias</b>	RF-12
<b>Precondición</b>	Haber introducido el email para recuperar la contraseña
<b>Poscondición</b>	Email validado
<b>Propósito</b>	
Validar el email introducido	
<b>Resumen</b>	
El sistema validará el email introducido por el usuario	

Cuadro 3.4: Descripción caso de uso CU-4

<b>Caso de uso</b>	Enviar email CU-5
<b>Actores</b>	Usuario/Administrador
<b>Tipo</b>	Secundario, básico y esencial
<b>Referencias</b>	RF-12
<b>Precondición</b>	Que el email esté validado
<b>Poscondición</b>	Email enviado
<b>Propósito</b>	
Enviar un email con la nueva contraseña	
<b>Resumen</b>	
El sistema validará enviará un email a la dirección de correo del usuario	

Cuadro 3.5: Descripción caso de uso CU-5

<b>Caso de uso</b>	Gestionar perfil CU-6
<b>Actores</b>	Usuario/Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-13
<b>Precondición</b>	Estar identificado en el sistema
<b>Poscondición</b>	Visualización y edición de datos del perfil
<b>Propósito</b>	
Gestionar los datos del perfil del usuario	
<b>Resumen</b>	
El usuario podrá visualizar, editar y eliminar datos de su perfil	

Cuadro 3.6: Descripción caso de uso CU-6

<b>Caso de uso</b>	Consultar mis reservas CU-7
<b>Actores</b>	Usuario
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-14
<b>Precondición</b>	Estar identificado en el sistema
<b>Poscondición</b>	Visualización de las entradas compradas
<b>Propósito</b>	
Consultar entradas compradas	
<b>Resumen</b>	
El usuario podrá visualizar todas las entradas que ha comprado	

Cuadro 3.7: Descripción caso de uso CU-7

<b>Caso de uso</b>	Descargar entradas CU-8
<b>Actores</b>	Usuario
<b>Tipo</b>	Opcional, extendido y esencial
<b>Referencias</b>	RF-14
<b>Precondición</b>	Tener entradas compradas
<b>Poscondición</b>	Descarga de entradas
<b>Propósito</b>	
Descargar una de las entradas compradas	
<b>Resumen</b>	
El usuario podrá descargar las entradas compradas	

Cuadro 3.8: Descripción caso de uso CU-8

<b>Caso de uso</b>	Modificar reserva CU-9
<b>Actores</b>	Usuario
<b>Tipo</b>	Opcional, extendido y esencial
<b>Referencias</b>	RF-8
<b>Precondición</b>	Tener entradas compradas y que falten 5 días para su inicio
<b>Poscondición</b>	Modificación de reserva
<b>Propósito</b>	
Modificar la reserva de entradas	
<b>Resumen</b>	
El usuario podrá modificar la reserva de las entradas compradas	

Cuadro 3.9: Descripción caso de uso CU-9

<b>Caso de uso</b>	Seleccionar película/evento CU-10
<b>Actores</b>	Usuario
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-19, RF-19.1 y RF-19.2
<b>Precondición</b>	Que haya alguna película almacenada
<b>Poscondición</b>	Visualizar los detalles de la película
<b>Propósito</b>	
Visualizar los detalles de la película	
<b>Resumen</b>	
El usuario podrá visualizar todos los detalles de la película seleccionada	

Cuadro 3.10: Descripción caso de uso CU-10

Caso de uso	Comentar CU-11
Actores	Usuario
Tipo	Primario, básico y esencial
Referencias	RF-15
Precondición	Que haya alguna película almacenada
Poscondición	Comentario añadido a la película
Propósito	Comentar la película
Resumen	El usuario podrá comentar la película seleccionada

Cuadro 3.11: Descripción caso de uso CU-11

Caso de uso	Borrar comentario CU-12
Actores	Usuario
Tipo	Primario, básico y esencial
Referencias	RF-16
Precondición	Que haya alguna comentario propio en la película
Poscondición	Comentario eliminado
Propósito	Eliminar un comentario añadido por el propio usuario
Resumen	El usuario podrá borrar un comentario que haya sido añadido por él mismo

Cuadro 3.12: Descripción caso de uso CU-12

Caso de uso	Añadir cine CU-13
Actores	Administrador
Tipo	Primario, básico y esencial
Referencias	RF-3
Precondición	Tener una cuenta de administrador
Poscondición	Cine añadido
Propósito	Añadir un nuevo cine al sistema
Resumen	El administrador podrá añadir un nuevo cine al sistema

Cuadro 3.13: Descripción caso de uso CU-13

Caso de uso	Eliminar cine CU-14
Actores	Administrador
Tipo	Primario, básico y esencial
Referencias	RF-3
Precondición	Que haya algún cine en el sistema
Poscondición	Cine eliminado
Propósito	Eliminar un cine del sistema
Resumen	El administrador podrá eliminar un cine al sistema

Cuadro 3.14: Descripción caso de uso CU-14

<b>Caso de uso</b>	Modificar información sobre películas CU-15
<b>Actores</b>	Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-4
<b>Precondición</b>	Que haya alguna película en el sistema
<b>Poscondición</b>	Información sobre película modificada
<b>Propósito</b>	
Modificar información sobre película	
<b>Resumen</b>	
El administrador podrá modificar información sobre una película del sistema	

Cuadro 3.15: Descripción caso de uso CU-15

<b>Caso de uso</b>	Añadir película CU-16
<b>Actores</b>	Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-4
<b>Precondición</b>	Que haya alguna película en el sistema
<b>Poscondición</b>	Película añadida
<b>Propósito</b>	
Añadir película al sistema	
<b>Resumen</b>	
El administrador podrá añadir una película al sistema	

Cuadro 3.16: Descripción caso de uso CU-16

<b>Caso de uso</b>	Procesar solicitudes de alquiler de sala CU-17
<b>Actores</b>	Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-26
<b>Precondición</b>	Tener una cuenta de administrador
<b>Poscondición</b>	Solicitud procesada
<b>Propósito</b>	
Procesar solicitud de alquiler de sala	
<b>Resumen</b>	
El administrador podrá procesar la solicitud de alquiler de una sala para algún evento	

Cuadro 3.17: Descripción caso de uso CU-17

<b>Caso de uso</b>	Eliminar película/evento CU-18
<b>Actores</b>	Administrador
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-4
<b>Precondición</b>	Que haya alguna película en el sistema
<b>Poscondición</b>	Película eliminada
<b>Propósito</b>	
Eliminar película/evento del sistema	
<b>Resumen</b>	
El administrador podrá eliminar una película o evento del sistema	

Cuadro 3.18: Descripción caso de uso CU-18

<b>Caso de uso</b>		Consultar películas CU-19
<b>Actores</b>		Usuario
<b>Tipo</b>		Primario, básico y esencial
<b>Referencias</b>		RF-5
<b>Precondición</b>		Que haya sesión para esa fecha
<b>Poscondición</b>		Sesiones disponibles
<b>Propósito</b>		
Consultar las sesiones de las películas para una fecha		
<b>Resumen</b>		
El usuario podrá consultar las sesiones disponibles de las películas en una fecha		

Cuadro 3.19: Descripción caso de uso CU-19

<b>Caso de uso</b>		Comprar entradas CU-20
<b>Actores</b>		Usuario
<b>Tipo</b>		Primario, básico y esencial
<b>Referencias</b>		RF-6, RF-6.1, RF-6.2 y RF-6.3
<b>Precondición</b>		Haber seleccionado una sesión para una película
<b>Poscondición</b>		Entradas compradas
<b>Propósito</b>		
Comprar entradas para una sesión de una película		
<b>Resumen</b>		
El usuario podrá comprar las entradas para una sesión de una película en una fecha determinada		

Cuadro 3.20: Descripción caso de uso CU-20

Caso de uso	Retirar del carrito CU-21
<b>Actores</b>	Usuario
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-8
<b>Precondición</b>	Haber elegido asientos para la sesión
<b>Poscondición</b>	Cancelar compra
Propósito	
<b>Resumen</b>	
El usuario podrá cancelar la compra de entradas para una sesión de una película en una fecha determinada	

Cuadro 3.21: Descripción caso de uso CU-21

Caso de uso	Pagar CU-22
<b>Actores</b>	Usuario
<b>Tipo</b>	Primario, básico y esencial
<b>Referencias</b>	RF-7
<b>Precondición</b>	Haber seleccionado asientos para la sesión
<b>Poscondición</b>	Entradas pagadas
Propósito	
<b>Resumen</b>	
El usuario podrá pagar las entradas para una sesión de una película en una fecha determinada mediante PayPal o tarjeta de crédito	

Cuadro 3.22: Descripción caso de uso CU-22

Caso de uso		Enviar entradas CU-23
Actores		Usuario
Tipo		Primario, básico y esencial
Referencias		RF-20
Precondición		Haber pagado y comprado las entradas
Poscondición		Entradas enviadas por email

Propósito
Enviar entradas compradas por email

Resumen
El usuario recibirá las entradas pagadas por email

Cuadro 3.23: Descripción caso de uso CU-23

### 3.6. Diagramas de casos de uso

Como explica la definición de diagrama, es un gráfico en el que se simplifica y esquematiza la información sobre un proceso o un sistema. Puede ser simple o complejo, con pocos o muchos elementos. Se trata de un resumen completo, que sirve para conocer e interpretar información de manera simple y visual.[14]

En este caso detallaremos los diferentes casos de uso que tiene la aplicación y se dará una breve explicación de cada uno de ellos.

#### 3.6.1. Identificación de usuario

En la figura 2.1 se muestra el caso de uso de un usuario sin identificar cuando entra a la plataforma, tiene tres opciones para identificarse:

- Iniciar sesión si ya está registrado, se comprueban sus datos y si son correctos se logeará en la plataforma donde puede gestionar su perfil ya creado, consultararlo, crearlo o eliminarlo.
- Registrarse en la plataforma, donde se comprobarán que los datos son correctos y que no haya un usuario ya registrado con ese email y podrá gestionar el perfil como anteriormente se ha descrito.
- Recuperar la contraseña si está registrado pero no la recuerda, se validará el email introducido y se enviará un enlace para cambiar la contraseña de la cuenta.

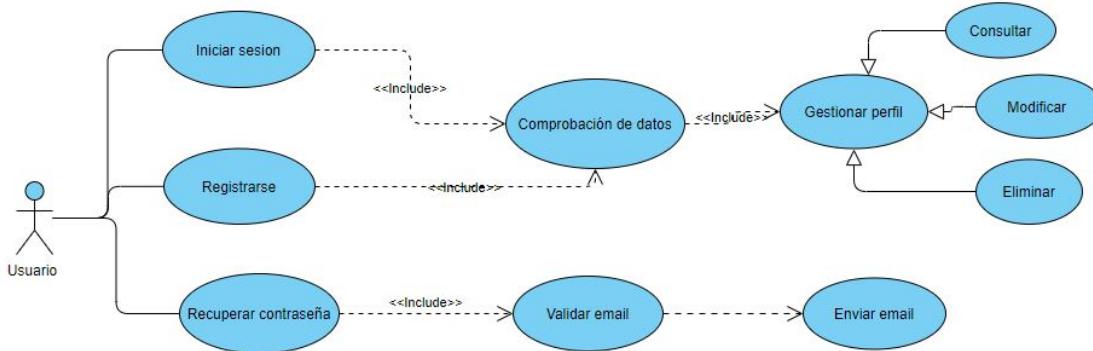


Figura 3.1: Diagrama de identificación de usuarios

#### 3.6.2. Comprar entrada/s online

En la figura 2.2 se muestra el caso de que un usuario quiera comprar una o varias entradas a través de la plataforma. Para comprar entradas, no hace falta estar registrado como usuario por lo que no se tiene en cuenta si el usuario está o no identificado por el sistema. Las opciones que tiene un usuario es consultar las películas y ver información sobre ella o elegir una película y comprar las entradas, se comprobarán que los datos introducidos son correctos y puedes pagar la entrada

con tarjeta de crédito, PayPal o un bono de entradas que tengas en tu cuenta, el paso final sería descargar o enviar al email las entradas.

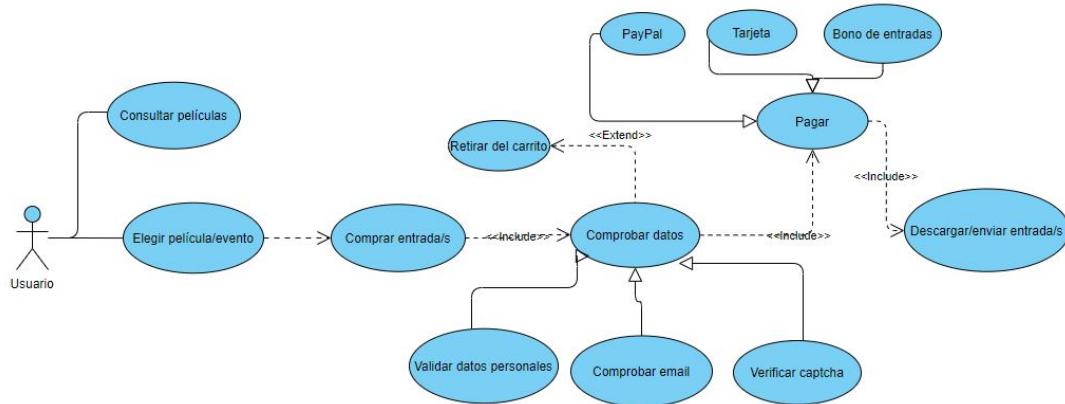


Figura 3.2: Diagrama de compra de entradas online

### 3.6.3. Comprar entrada/s en taquilla

En la figura 2.3 se muestra el caso de uso de un cliente que llega al cine y quiere comprar una entrada en la taquilla. Hay varios actores implicados además del cliente como es el empleado, el servicio de pago y el supervisor. El cliente compra un número de entradas determinado al empleado, que le cobra la entrada al precio de ese día y se la da. Al final de cada día habrá un supervisor que verificará que el dinero ingresado es el correcto y registrará el total de ventas.

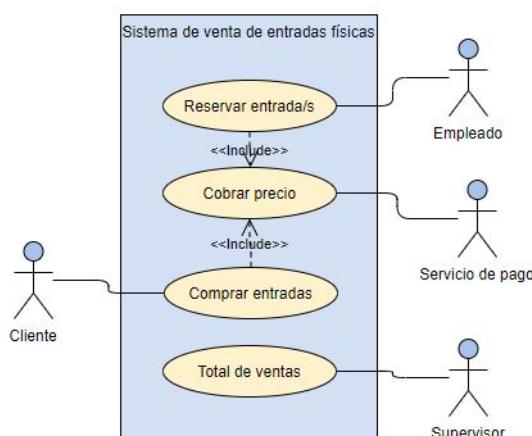


Figura 3.3: Diagrama de compra de entradas en taquilla

### 3.6.4. Gestionar entradas compradas

En la figura 2.4 se muestra como un usuario identificado en el sistema, puede consultar sus reservas de entradas y modificarlas, es decir, cancelar la reserva o modificar los asientos. La única condición para que esto se de, es que deben faltar 5 o más días para el inicio de la película o evento. También puedes descargar las entradas

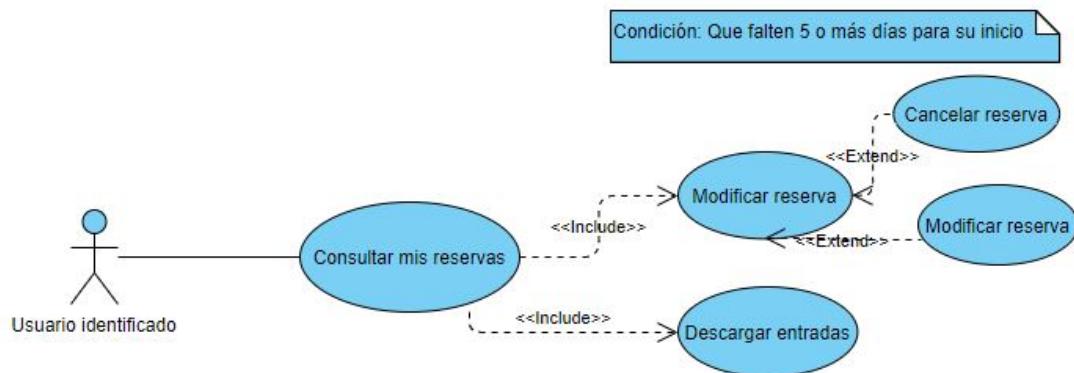


Figura 3.4: Diagrama de gestión de entradas compradas

### 3.6.5. Gestionar comentarios

En la figura 2.5 se muestra como un usuario identificado en el sistema puede seleccionar una película o evento para comentarla, valorarla o valorar comentarios de otros usuarios. Los usuarios moderadores son los únicos que pueden borrar comentarios de otros usuarios a excepción de que cada usuario puede borrar sus propios comentarios.

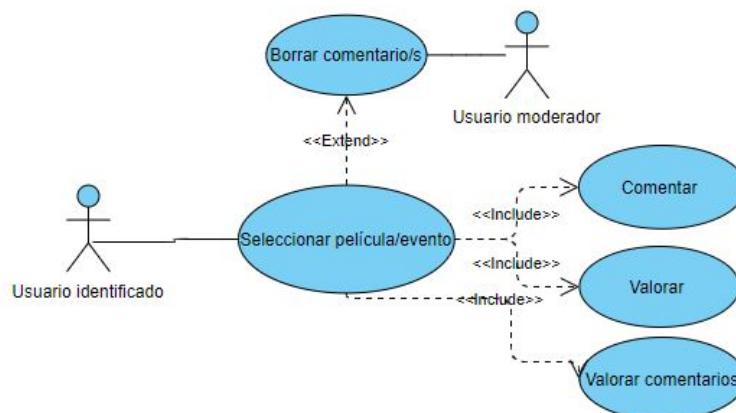


Figura 3.5: Diagrama de gestión de comentarios

### 3.6.6. Funciones del usuario administrador

En la figura 2.6 se muestra las funciones y privilegios que tiene un usuario administrador, puede añadir y eliminar un cine, además de añadir, modificar o eliminar una película/evento. Un usuario puede solicitar el alquiler de la sala de cine para un evento personal o empresarial, esa solicitud la procesará el administrador que la aceptará o denegará. También pueden banear a usuarios.

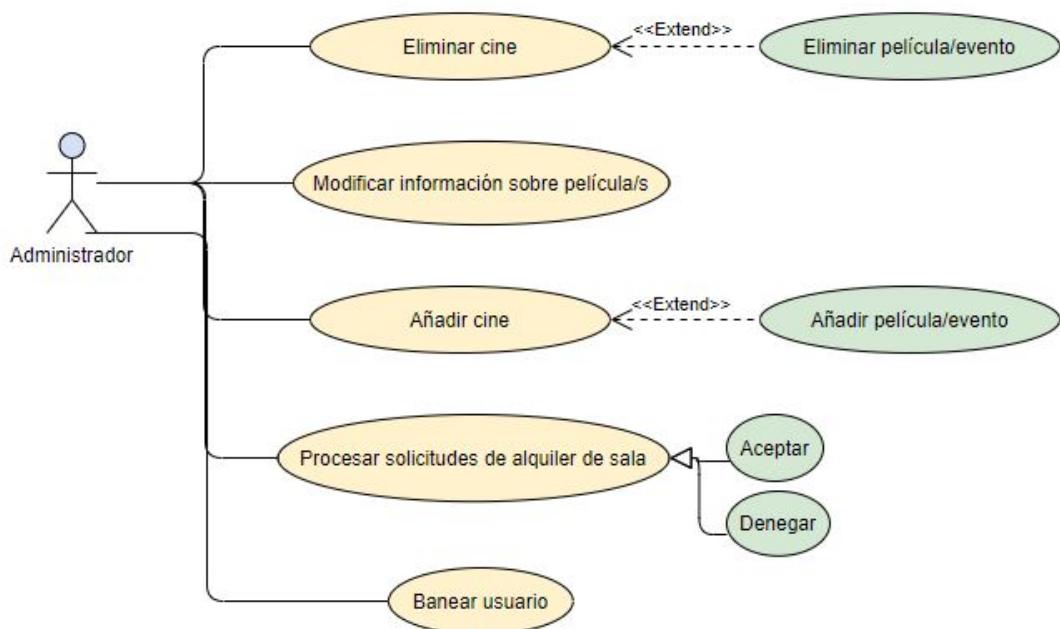


Figura 3.6: Diagrama de funciones de usuario administrador

### 3.7. Diagramas de secuencia

El diagrama de secuencia es un tipo de diagrama del lenguaje unificado de modelado (UML) que se trata de un lenguaje orientado a objetos y está compuesto por elementos gráficos. UML modela sistemas y procesos de la programación orientada a objetos así como procesos de negocio con el objetivo de presentar asuntos complejos de manera clara.[15]

Utiliza una serie de notaciones visuales para representar el flujo de una funcionalidad de nuestro sistema. Para diseñar dichos diagramas he utilizado la herramienta de Visual Paradigm Online.

#### 3.7.1. Registro de un usuario

Como podemos observar en la figura 3.7, cuando un usuario quiere registrarse en el sistema, primero envío los datos del formulario de registro al controlador de usuarios que valida los datos y si dichos datos son validados comprueba si el usuario está registrado en el sistema. Si no está registrado en el sistema, se crea el usuario en el sistema con los datos introducidos en el formulario, si no se crea un error de que el usuario ya existe.

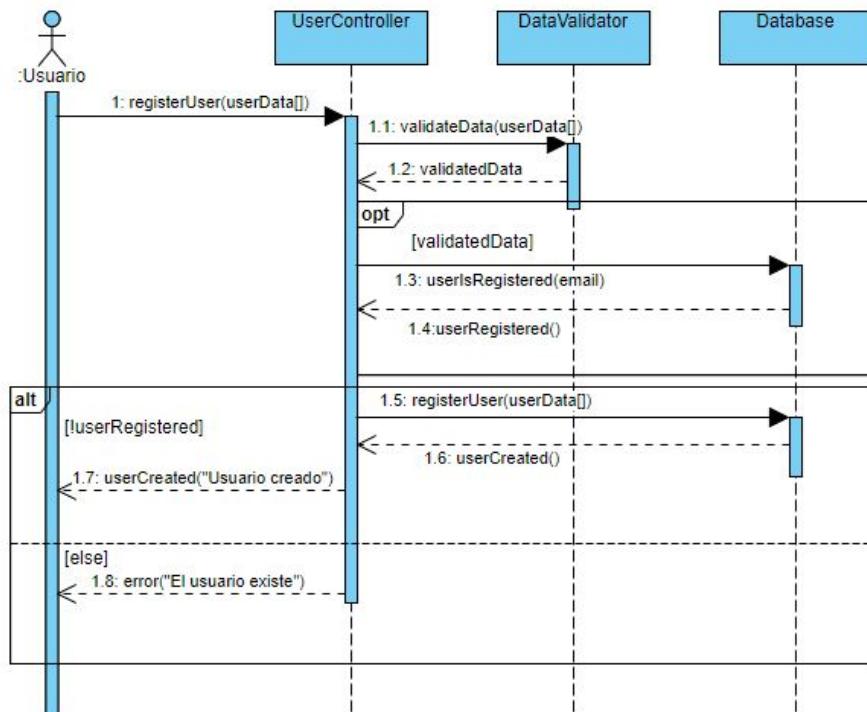


Figura 3.7: Diagrama de secuencia de registro de usuario.

### 3.7.2. Login de un usuario

Para que un usuario pueda logearse en el sistema, el controlador de usuarios hace una búsqueda del usuario con los datos que ha introducido (email y contraseña), si el usuario ha sido encontrado y la comprobación de los datos son correctos, se identificará en el sistema, de lo contrario mostrará un mensaje de información de que los datos introducidos no son válidos.

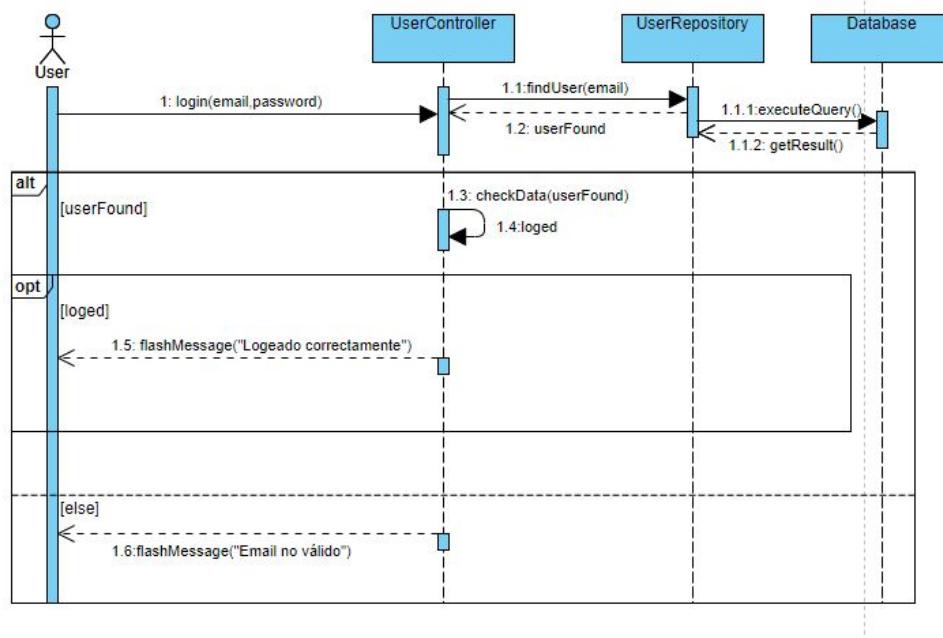


Figura 3.8: Diagrama de secuencia para logearse en el sistema.

### 3.7.3. Olvidar la contraseña

Cuando un usuario olvida la contraseña para identificarse con su cuenta, debe indicar su email y el sistema buscará que exista un usuario con ese email registrado. Si se ha encontrado el usuario, se genera una contraseña aleatoria y se envía el email con la contraseña generada. Si no ha ocurrido ningún error al enviarse el email, se cambia la contraseña que tenía el usuario por la generada aleatoriamente y se le avisa con un mensaje flash.

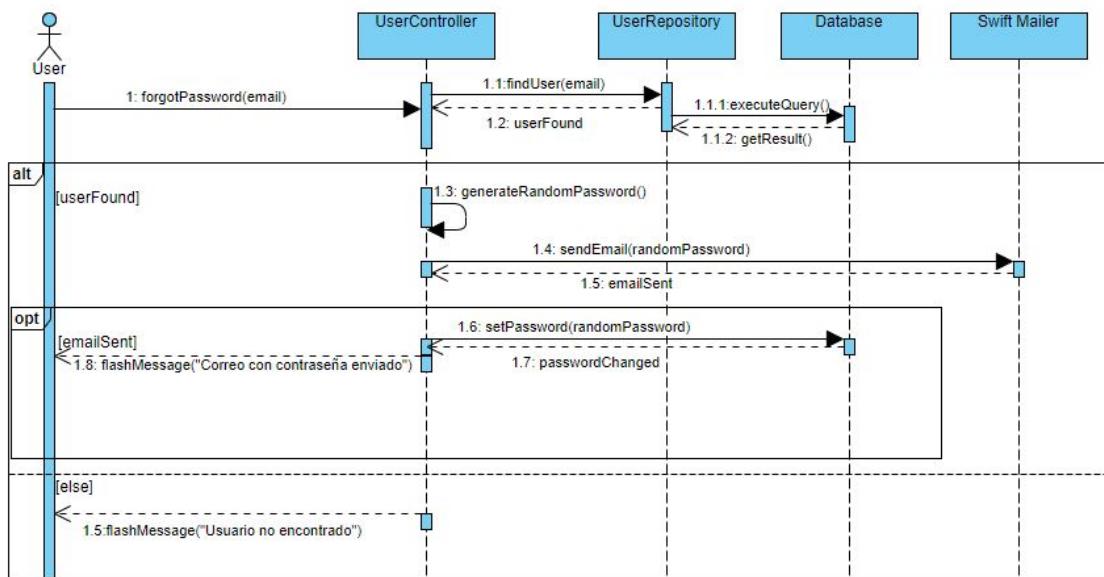


Figura 3.9: Diagrama de secuencia de olvidar la contraseña.

### 3.7.4. Eliminar usuario

Cuando un usuario con privilegios de administrador quiere eliminar un usuario del sistema, debe pasarle los datos del usuario a eliminar al controlador de usuarios, éste valida los datos y si los datos son válidos eliminar el usuario. Al eliminar el usuario se eliminará los tickets y comentarios asociados a él.

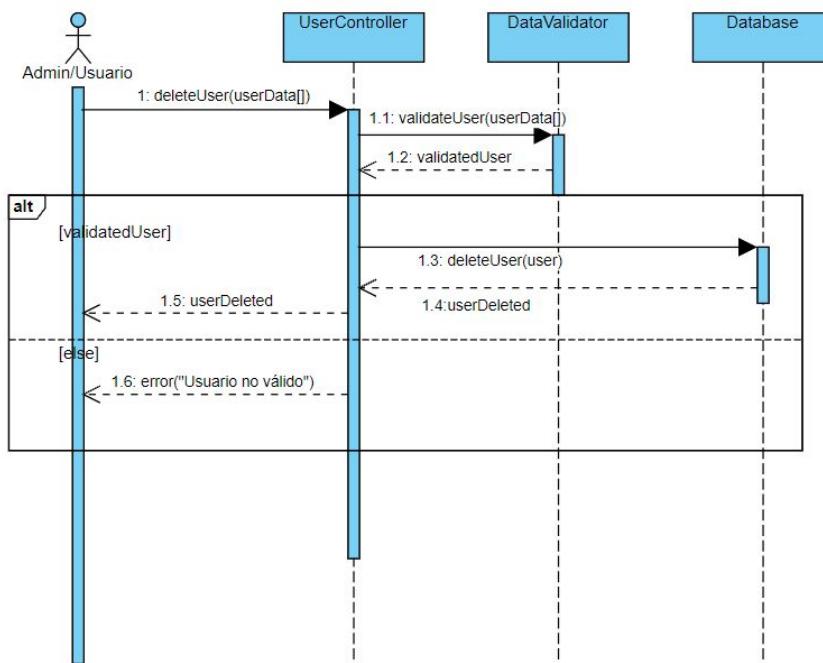


Figura 3.10: Diagrama de secuencia de eliminación de usuario.

### 3.7.5. Añadir evento, cine y sesión

Si un usuario con privilegios de administrador quiere añadir un evento al sistema, debe introducir todos los datos que tiene un evento. Tras la validación de los datos, el evento se añadirá.

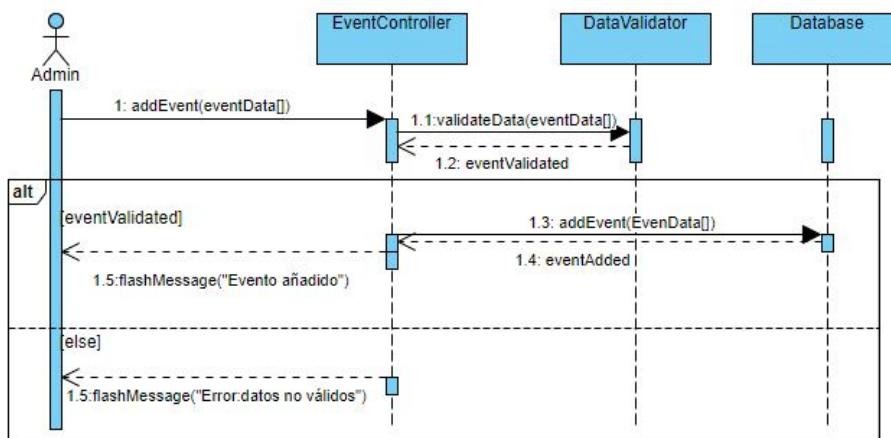


Figura 3.11: Diagrama de secuencia para añadir un evento.

Si un usuario administrador añade un nuevo cine al sistema, introducirá todos los datos de un cine. El controlador de cines validará el número de salas junto con sus filas y columnas. Si los datos son correctos, se crea el cine con sus datos, se crea una sala por cada una de las salas que el usuario ha introducido y se crea un asiento por cada fila y columna que se haya indicado.

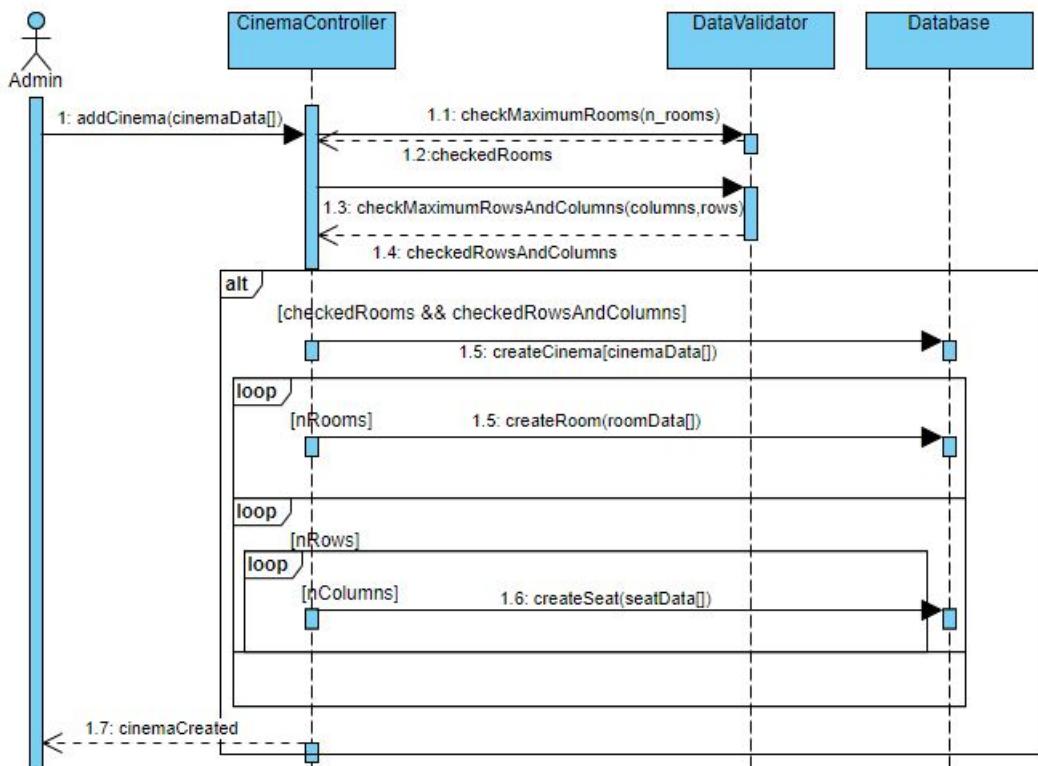


Figura 3.12: Diagrama de secuencia para añadir un cine.

Si un usuario administrador quiere añadir una nueva sesión, se introducen los datos de la sesión, el controlador de sesiones validará los datos (que la fecha sea superior a la actual). El controlador de sesiones obtiene todas las sesiones activas y las salas del cine que ha indicado el usuario. Si los datos son válidos, no hay sesiones activas y las salas existen significa que es la primera vez que se va a crear una sesión para esa fecha en ese cine. Si no es así, entonces hay sesiones para esa fecha por lo que hay que hacer una búsqueda para ver si está libre entre la fecha de inicio y la fecha de fin en cada sala, si está vacío entonces creamos la sesión con sus datos y salimos del bucle con *break*.

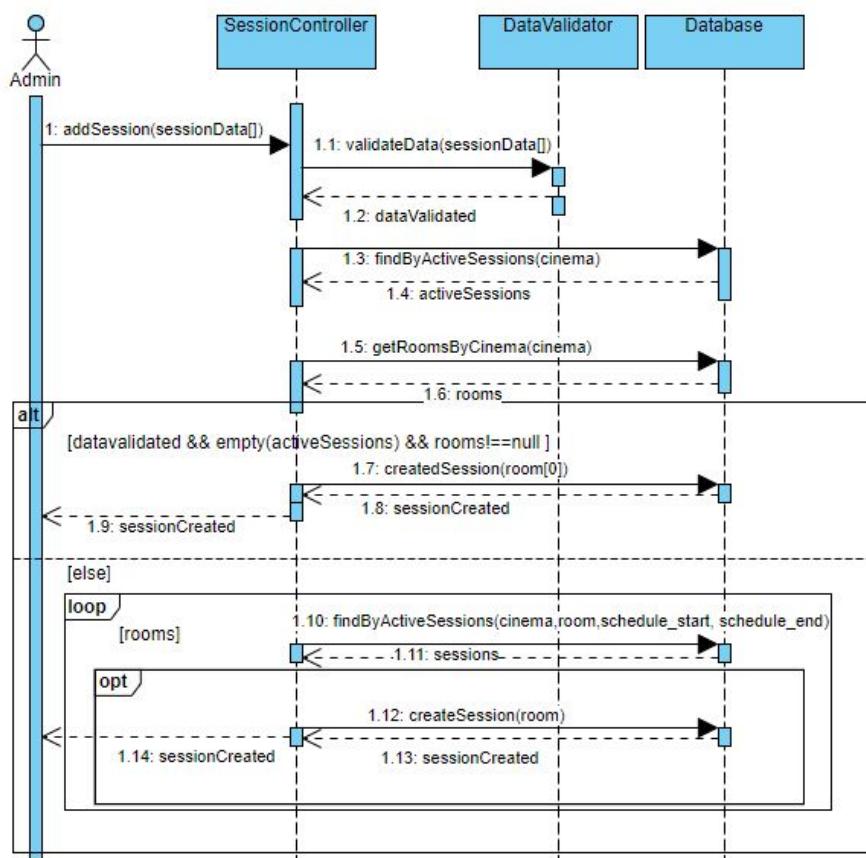


Figura 3.13: Diagrama de secuencia para añadir una sesión.

### 3.7.6. Reserva de entradas

Si un usuario quiere reservar entradas para película deberá introducir el cine y la fecha deseados. El controlador de eventos busca las sesiones de películas en ese cine y fecha concretos y se le devuelve al usuario en una vista. El usuario elige una sesión en concreto y si la sala existe, el controlador de reservas devuelve los asientos de esa sala y los asientos reservados para que el usuario puede elegir que asientos escoger. El usuario escoge los asientos, el email al que se le enviarán las entradas y el método de pago para que el controlador de reservas compruebe los datos y si son correctos crear un *ticket* y un asiento reservado por cada asiento que el usuario ha seleccionado. Al terminar el proceso se le enviarán al email las entradas que acaba de comprar.

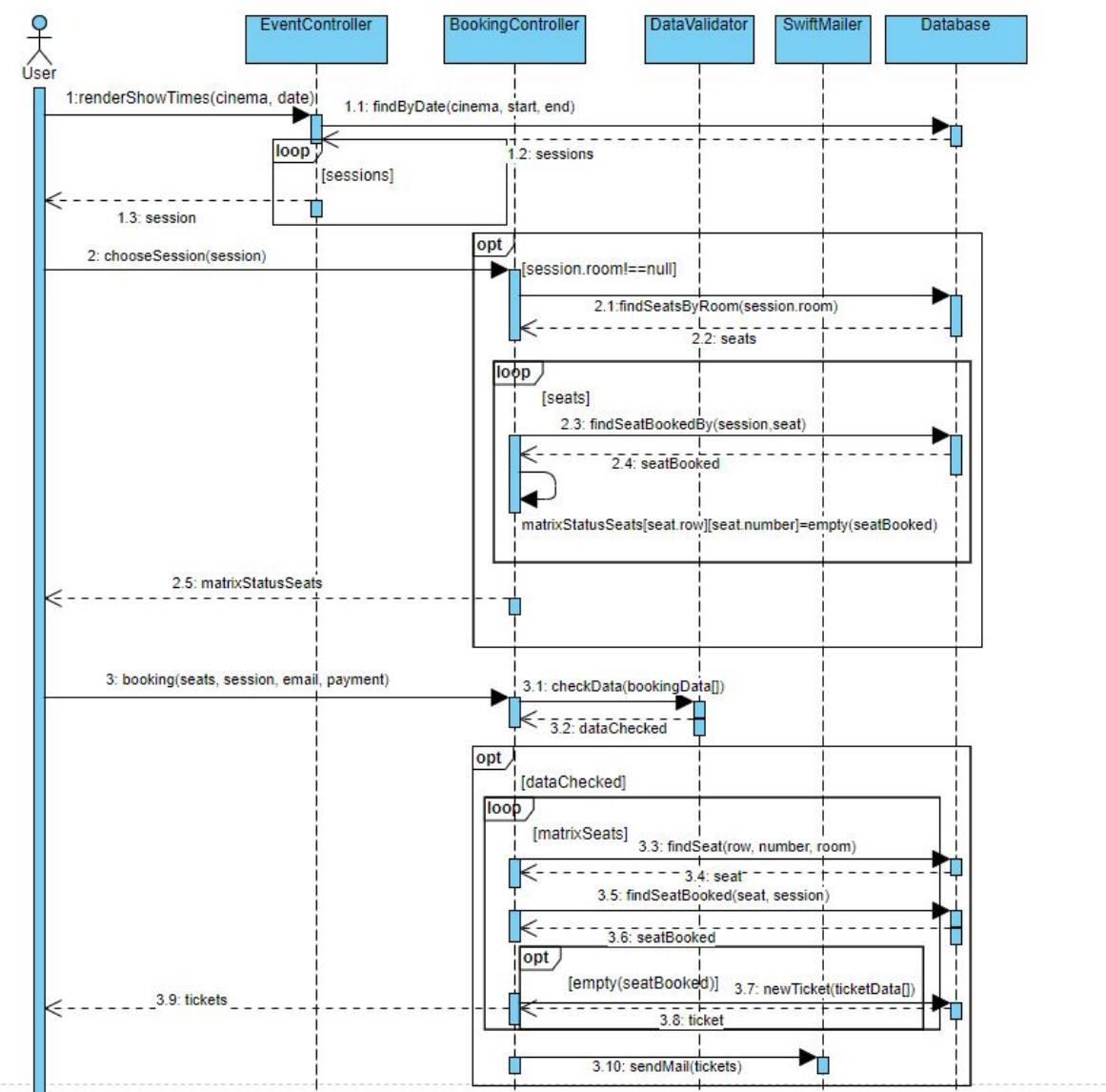


Figura 3.14: Diagrama de secuencia para reservar entradas.

# Capítulo 4

## Diseño

### 4.1. Base de datos

En esta sección se explicará todos los detalles del diseño de la base de datos de la aplicación web junto con las relaciones entre las diferentes entidades.

#### 4.1.1. Diagrama entidad-relación

En la figura 4.1 se muestra el diagrama entidad-relación de la base de datos del proyecto. Hay un total de diez entidades de las cuales dos son entidades débiles ya que su existencia depende de otras entidades. Vamos a explicar el flujo de todas las entidades y sus relaciones en este diagrama.

Primero empezaremos por explicar las **entidades**:

- La entidad **User** que almacenará toda la información relacionada con un usuario registrado en la aplicación, tiene como atributos:

Un ID como la clave primaria que identificará únicamente al usuario, nombre, apellidos, email con el que se identificará, la contraseña, privilegios (su valor dependerá del tipo de usuario que sea), una foto de perfil, su número de teléfono, su ciudad y cuándo ha sido creado.

- La entidad **EventData** que guardará toda la información necesario sobre un evento o película, tiene como atributos:

Un ID como la clave primaria que identificará únicamente al evento, el título, tipo de evento (película, concierto, congreso, charla..), género, descripción, duración, fecha de estreno, actores (también pueden ser los participantes o protagonistas del evento), director, la foto del póster, la foto de fondo, la valoración, el estado (actualmente en cartelera o no), el trailer de youtube, la frase de cabecera y la edad recomendada.

- La entidad **Comment** almacena el comentario de un usuario en una película. Tiene como atributos el ID del comentario para identificarlo, el texto, el ID de TMDB por si la película viene de la API de TMDB y cuando ha sido creado.

- La entidad **Session** almacena toda la información del horario de la película en el cine. Tiene como atributos el ID de sesión, la hora y fecha de cuando empieza y termina el evento y el idioma en el que se realiza (ya que puede haber versión original).

- La entidad **Cinema** almacena la información de cada cine. Tiene como atributos el ID del cine para identificarlo, el nombre, la localización y el precio de las entradas.
- La entidad **LogInfo** que almacena la información de todos los procesos que ocurren en la plataforma (inicio de sesión, compra de entradas, inserción de datos, edición de datos, borrado de datos...), por lo que sus atributos son el ID, fecha, información y tipo de información (ERROR, WARNING o SUCCESS)
- La entidad **Room** almacena la información de una sala de cine. Sus atributos son el ID de la sala para identificarla, el número de filas, el número de columnas y el número de la sala.
- La entidad **SeatBooked** almacena la información de un asiento que se ha reservado. Sus atributos son el ID del asiento reservado, su fila y su número de asiento.
- La entidad **Ticket** almacena la información de una entrada de cine para una sesión. Sus atributos son el ID del ticket para identificarlo, el precio, la fecha y hora a la que empieza el evento, el código QR y la fecha de venta.
- La entidad **Coupon** almacena la información de un cupón de descuento para aplicarlo en la compra de entradas. Sus atributos son el ID del cupón para identificarlo, el código, si está activo y la fecha de cuándo expira el cupón.

Una vez están explicadas las entidades con sus atributos, se explicarán las **relaciones** entre las distintas entidades:

- **User-Ticket**: Esta relación tiene una cardinalidad 1:N ya que un usuario puede tener varios tickets y varios tickets pueden pertenecer a un solo usuario.
- **User-Comment**: Esta relación tiene una cardinalidad 1:N ya que un usuario puede añadir varios comentarios y varios comentarios pueden pertenecer a un solo usuario.
- **User-Coupon**: Esta relación tiene una cardinalidad 1:N ya que un usuario puede tener varios cupones de descuento y varios cupones de descuento pueden pertenecer a un solo usuario.
- **EventData-Comment**: Esta relación tiene una cardinalidad 1:N ya que un evento puede tener varios comentarios y varios comentarios pueden pertenecer a un solo evento.
- **EventData-Session**: Esta relación tiene una cardinalidad 1:N ya que un evento puede tener varias sesiones y varias sesiones pueden pertenecer a un solo evento.
- **Session-Cinema**: Esta relación tiene una cardinalidad N:1 ya que varias sesiones pueden pertenecer a un solo cine y un cine le pueden pertenecer varias sesiones.
- **Cinema-Room**: Esta relación tiene una cardinalidad 1:N ya que un cine puede tener varias salas y varias salas pueden pertenecer a un cine.
- **Room-Session**: Esta relación tiene una cardinalidad 1:N ya que una sala puede tener varias sesiones y varias sesiones pueden pertenecer a una sala.

- **Room-SeatBooked:** Esta relación tiene una cardinalidad 1:N ya que una sala puede tener varios asientos resevadis y varios asientos reservados pueden pertenecer a una sala.
- **SeatBooked-Ticket:** Esta relación tiene una cardinalidad 1:1 ya que por cada asiento reservado corresponde un ticket y por cada ticket un asiento.
- **SeatBooked-Session:** Esta relación tiene una cardinalidad 1:N ya que un asiento reservado puede corresponder a diferentes sesiones y diferentes sesiones corresponden a un asiento reservado.
- **SeatBooked-Room:** Esta relación tiene una cardinalidad N:1 ya que muchos asientos reservados puede corresponder a una sola sala y una sala le pueden pertenecer varios asientos reservados.

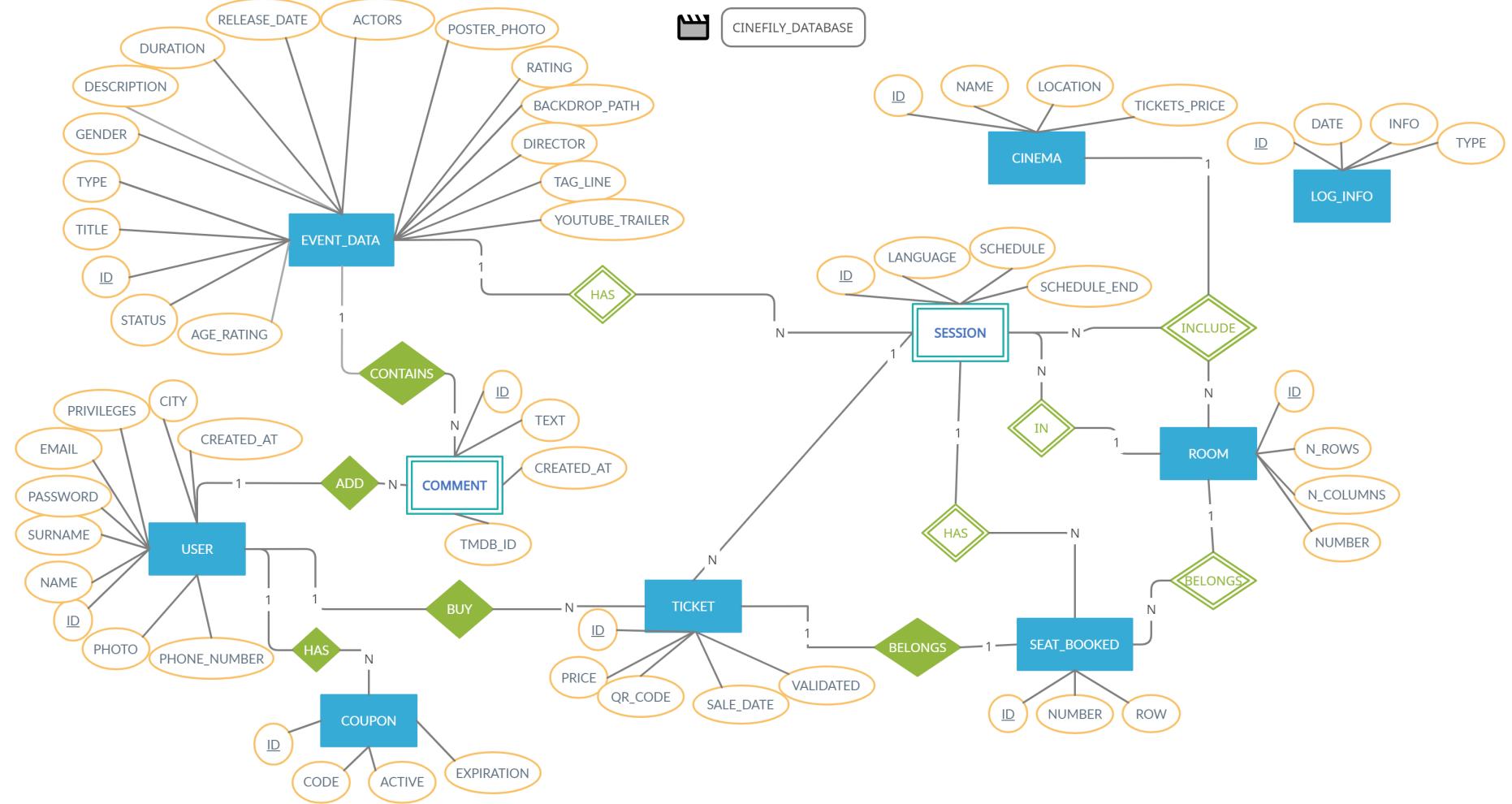


Figura 4.1: Diagrama E-R

#### 4.1.2. Paso a tablas y normalización de la base de datos

Cuando ya se ha realizado el diagrama entidad-relación de la base de datos, es necesario formalizar el paso a tablas de cada una de las entidades con sus respectivas relaciones que aparecen en el diagrama.

Como se puede observar en la figura 4.2, se han definido diez tablas con sus respectivas claves primarias, claves candidatas y claves externas si proceden. En este caso, no ha sido posible realizar ninguna fusión ya que se produciría pérdida de información al no tener ninguna tabla con información repetida, claves primarias iguales y que no procedan de una herencia.

```

1: user(id_, name, surname, password, email, privileges, created_at, photo, phone_number, city)
   CP           CC

2: comment(id_, text, , tmdb_id, user_id, created_at, event_data_id)
   CP           CE(1)        CE(3)

3: event_data(id_, title, type, gender, description, duration, release_date, actors, poster_photo, rating, backdrop_path, director, tag_line, age_rating, youtube_trailer)
   CP

4: session(id_, language, room_id, schedule, event_data_id, schedule_end, cinema_id_)
   CP           CE(6)        CE(3)        CE(5)

5: cinema(id_, name, location, tickets_price)
   CP

6: room(id_, n_rows, n_columns, number, cinema_id)
   CP           CE(5)

7: coupon(id_, code, active, expiration)
   CP

8: seat_booked(id_, session_id, room_id, ticket_id, row, number)
   CP           CE(4)        CE(6)        CE(9)
                  CC

9: ticket(id_, price, qr_code, sale_date, seat_booked_id)
   CP           CE(8)

10: log_info(id_, date, info, type)
    CP
  
```

Figura 4.2: Paso a tablas

La normalización de bases de datos es un proceso que consiste en designar y aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional con objeto de minimizar la redundancia de datos, facilitando su gestión posterior.[16]

La base de datos ya está en primera forma normal (1FN) ya que no hay repetición de grupos y los datos son atómicos. Además está en segunda forma normal (2FN) ya que está en 1FN y dado una clave primaria y sus atributos constituyentes que no forman parte de la clave primaria, el atributo no clave depende de toda la clave primaria en vez de solo de una parte de ella.

Por último, está en tercera forma normal (3FN) ya que además de estar en segunda forma normal, no hay ninguna dependencia transitiva problemática.

## 4.2. Arquitectura software

Una arquitectura software describe los patrones y técnicas que se utilizan para diseñar y desarrollar aplicaciones. Por lo que la arquitectura te proporciona unas pautas para estructurar la aplicación web en este caso. Usaremos la arquitectura **cliente-servidor** siguiendo un patrón **MVC** (Modelo, Vista, Controlador).

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.[17]

El patrón MVC separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.[18]

Cada uno de las capas tiene una función que desempeñar:

- La capa **Modelo** gestionará el almacenamiento y recuperación de datos y entidades del dominio. Principalmente constituye toda la información y operaciones con la base de datos, en Symfony cada entidad tiene asignado un repositorio, el repositorio está en la capa Modelo.
- La capa **Vista** genera la interfaz y representa un estado concreto del Modelo. En Symfony se utiliza el motor de plantillas Twig, todas las plantillas representan visualmente datos que les pasa el controlador.
- La capa **Controlador** es un coordinador general del sistema, actúa de *middleware* entre el usuario y el sistema (Vista y Modelo) y transforma los datos.

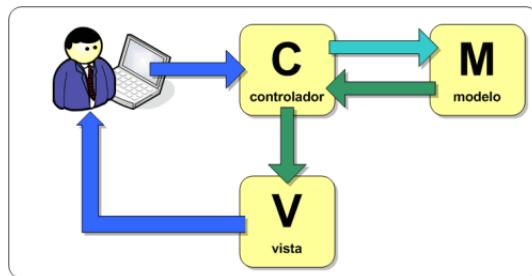


Figura 4.3: Diagrama del patrón MVC.[1]

### 4.3. Interfaz de usuario (Mockups)

El diseño de la interfaz de usuario en forma de prototipado es muy importante ya que la interfaz visual de una aplicación tanto web como móvil es lo primero que el usuario visualiza, la navegación entre las distintas páginas debe ser intuitiva y agradable para él. Además realizar un buen diseño ahorra mucho tiempo en el momento de la implementación del front-end de la aplicación web, ya que se tiene una idea clara de cómo es la disposición de los diferentes elementos de la web.

*Mockup* es un diseño digital de una aplicación o página web. Se utilizan en fases iniciales de proyectos para que tanto el cliente como el diseñador puedan previsualizar como sería el diseño de la aplicación. Mockup y prototipo no son lo mismo, ya que la diferencia principal es que el mockup es estático mientras que el prototipo es dinámico.

#### 4.3.1. Página principal

El diseño de la página principal busca ser lo más minimalista posible sin recargar la página de widgets y elementos innecesarios como se puede observar en la figura 4.4. Al entrar por primera vez el usuario verá unas cartas de películas con su información y donde si hace click en ellas puede ver todos los detalles. Las cartas están en un carrusel interactivo donde el usuario podrá desplazarse clickando en las flechas. En la barra superior, que es fija en todas las pantallas de la aplicación, se pueden observar cinco botones:

- Nuestros cines: el primero de ellos muestra toda la información sobre los cines de Cinefly (nombre, localización, número de salas, capacidad...etc). Si el usuario es administrador podrá eliminar cines.
- Cartelera: es donde el usuario elegirá un cine y una fecha para visualizar las sesiones y poder reservar tickets.
- Películas: para ver todas las películas de la aplicación paginadas. Si el usuario es administrador podrás editarlas y eliminarlas.
- Eventos: para ver todos los eventos de la aplicación paginados. Si el usuario es administrador podrás editarlos y eliminarlos.
- Contacto: es la zona donde el usuario podrá mandar una opinión o una solicitud para alquilar una sala de cine en una fecha concreta.

Si el usuario tiene privilegios de administrador tendrá un botón adicional con un submenú donde podrá añadir una sesión, evento o cine con su respectivo formulario, ver todos los usuarios y eliminarlos y visualizar el log del sistema.

Por último, la parte de mi perfil con un avatar como botón para un submenú donde se puede ver los datos del perfil, las entradas compradas o cerrar sesión.

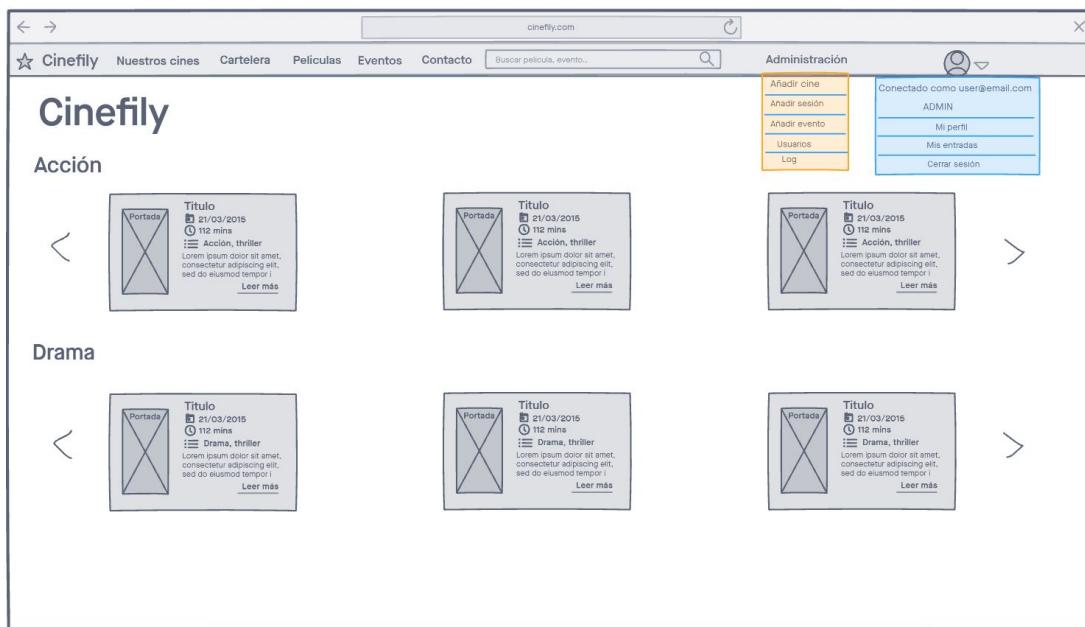


Figura 4.4: Mockup de la página principal

#### 4.3.2. Formulario para añadir evento, cine y sesión

Para poder utilizar estos formularios, el usuario debe tener privilegios de administrador ya que es para añadir un cine, un evento o una sesión con sus respectivos datos, propiedades y restricciones. Los tres tienen un diseño parecido ya que se busca la mayor facilidad para que el administrador pueda añadir a la aplicación nuevas demandas.

Este mockup muestra el formulario 'Añadir evento' en la aplicación Cinefly. El formulario incluye los siguientes campos:

- Título:** Un campo de texto para ingresar el título del evento.
- Tipo:** Un campo desplegable que actualmente muestra 'Película'.
- Género:** Una lista desplegable que incluye 'Acción X', 'Comedia X', 'Drama X', 'Terror X' y un botón para agregar más.
- Descripción:** Un campo de texto grande para describir el evento.
- Duración (minutos):** Dos campos numéricos para ingresar la duración.
- Valoración:** Un campo numérico para ingresar la puntuación.
- Fecha de estreno:** Un campo para seleccionar la fecha de estreno.
- Recomendada para:** Un campo desplegable que muestra 'Todos los públicos'.
- Director/es:** Un campo de texto para ingresar el nombre del director.
- Actores:** Un campo de texto para ingresar los nombres de los actores.
- Frase de película:** Un campo de texto para una cita de la película.
- Foto de portada:** Un campo para cargar una foto de portada con botones 'Buscar...' y 'Browse'.
- Foto de fondo:** Un campo para cargar una foto de fondo con botones 'Buscar...' y 'Browse'.
- Trailer (URL, YouTube):** Un campo de texto para la URL del trailer.
- En cartelera:** Un checkbox para indicar si el evento está en cartelera.
- Añadir:** Un botón para finalizar la creación del evento.

Figura 4.5: Mockup del formulario para añadir un evento

Este mockup muestra el formulario 'Añadir cine' en la aplicación Cinefly. El formulario incluye los siguientes campos:

- Nombre:** Un campo de texto para el nombre del cine.
- Localización:** Un campo de texto para la localización.
- Precio de las entradas (estándar):** Un campo de texto para el precio de las entradas.
- Salas (máximo 15):** Un campo numérico para el número de salas.
- Filas por sala (máximo 20):** Un campo numérico para el número de filas por sala.
- Asientos por fila (máximo 30):** Un campo numérico para el número de asientos por fila.
- Añadir:** Un botón para finalizar la creación del cine.

Figura 4.6: Mockup del formulario para añadir un cine

The mockup shows a modal window titled 'Añadir nueva sesión'. It contains four input fields: 'Evento' (Movie) dropdown set to 'Película 1', another 'Evento' dropdown set to 'Cine 1', 'Fecha' (Date) input field, and 'Idioma' (Language) dropdown set to 'Castellano'. A central button labeled 'Añadir' (Add) is at the bottom.

Figura 4.7: Mockup del formulario para añadir una sesión

#### 4.3.3. Cartelera

Para el diseño de la cartelera se ha escogido un selector donde aparecen todos los cines disponibles en la aplicación y eliges el cine y la fecha del día que se quieren visualizar sus sesiones. Al elegir, saldrán a la izquierda la foto de portada de la película o el evento junto con su título y a la derecha la sala y la hora de inicio de la proyección.

The mockup shows a search interface with 'Cine' (Cinema) dropdown ('Cine Kinefilly'), 'Fecha' (Date) input field, and a 'Buscar' (Search) button. Below, a table lists movies and their showtimes. The first row shows 'Película 1' with two showtimes: 'Sala 2' at 19:00 and 21:30. The second row shows 'Película 2' with one showtime: 'Sala 1' at 20:00. Placeholder images with an 'X' are shown for the movie posters.

Evento	Horario
Película 1	Sala 2 19:00 21:30
Película 2	Sala 1 20:00

Figura 4.8: Mockup de la cartelera

#### 4.3.4. Proceso de reserva

Este proceso está formado por dos pantallas, la primera de ella el usuario debe indicar el email al que se enviarán las entradas (si el usuario está identificado se autocompletará con su email) y el número de asientos que se quieren reservar (máximo 10 asientos). Tras llenar los campos anteriores, podrá proceder a elegir sus asientos de forma interactiva. Cada color tiene un significado diferente, verde es el asiento que se ha elegido, rojo el asiento ocupado y gris el asiento vacío.

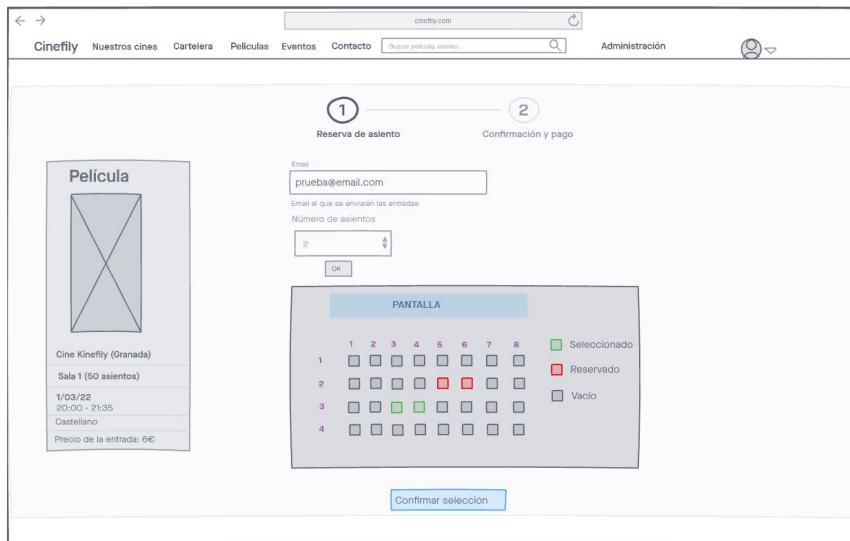


Figura 4.9: Mockup del formulario para elegir los asientos

Tras la elección de los asientos, se pasa a la segunda pantalla donde resume toda la información de la reserva: datos de la sesión, asientos elegidos, precio total de las entradas y el ticket generado. Cuando el usuario pulse el botón de comprar, aparecerá una ventana modal donde el usuario elegirá si quiere pagar las entradas a través de PayPal o pagarlas en taquilla.

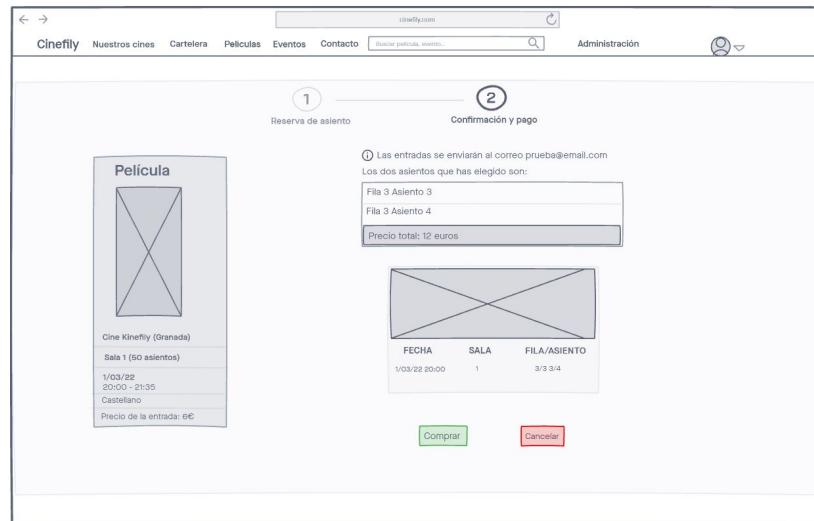


Figura 4.10: Mockup de la confirmación de la reserva de entradas

#### 4.3.5. Todos las películas/eventos

Cuando el usuario seleccione el botón de "Todas las películas" ó "Todos los eventos" podrá visualizar una tabla donde salen la portada de la película/evento junto con su descripción, sus sesiones activas (si tiene alguna) y las acciones, en este caso se pueden tanto editar como eliminar porque el usuario identificado tiene privilegios de administrador. Dicha tabla estará paginada por 10 eventos por página.

Todos las películas				
Evento	Descripción	Sesiones activas	Acciones	
 Película 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor i	Cine Kinefly 19:00      21:30		
 Película 2	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor i	No hay sesiones programadas		

Figura 4.11: Mockup de todas las películas

#### 4.3.6. Todos los usuarios

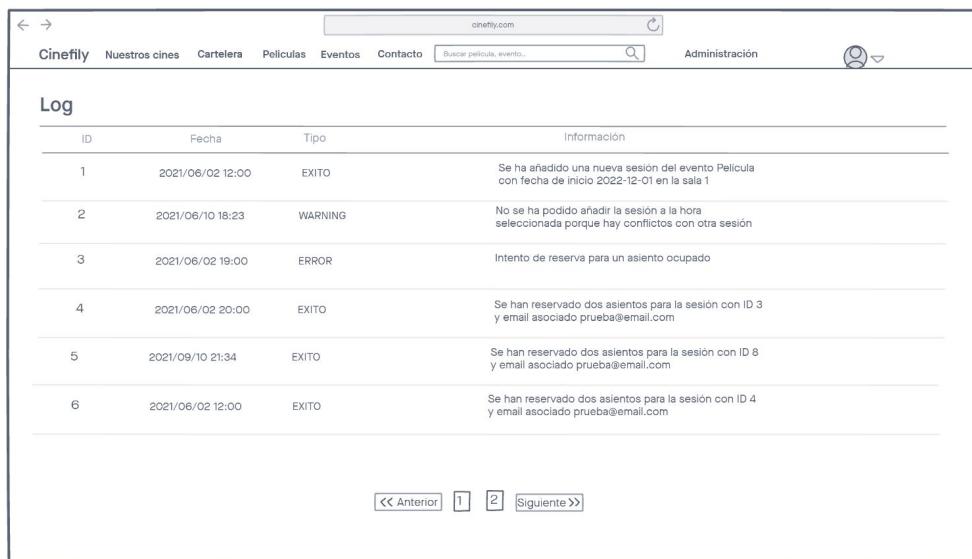
Cuando un usuario con privilegios de administrador seleccione en el menú de Administración el submenú de "Todos los usuarios" podrá ver una tabla paginada de todos los usuarios del sistema donde sale la foto de perfil, su nombre y apellidos, su email y desde cuando es miembro. Además podremos ver sus detalles y eliminar el usuario, la edición no está permitida por privacidad del usuario.

Foto	Usuario	Email	Miembro desde	Acciones
	Sergio Azañón	sergio@email.com	2021/06/02 12:00	
	Pepe González	pepe@email.com	2021/10/14 19:05	

Figura 4.12: Mockup de todos los usuarios

#### 4.3.7. Log del sistema

Cuando un usuario con privilegios de administrador seleccione en el menú de Administración el submenú de "Log" podrá ver una tabla paginada de todos los eventos que han sucedido en el sistema, tanto los que han dado error, los que han tenido éxito o los warnings.



The mockup shows a table titled "Log" with the following data:

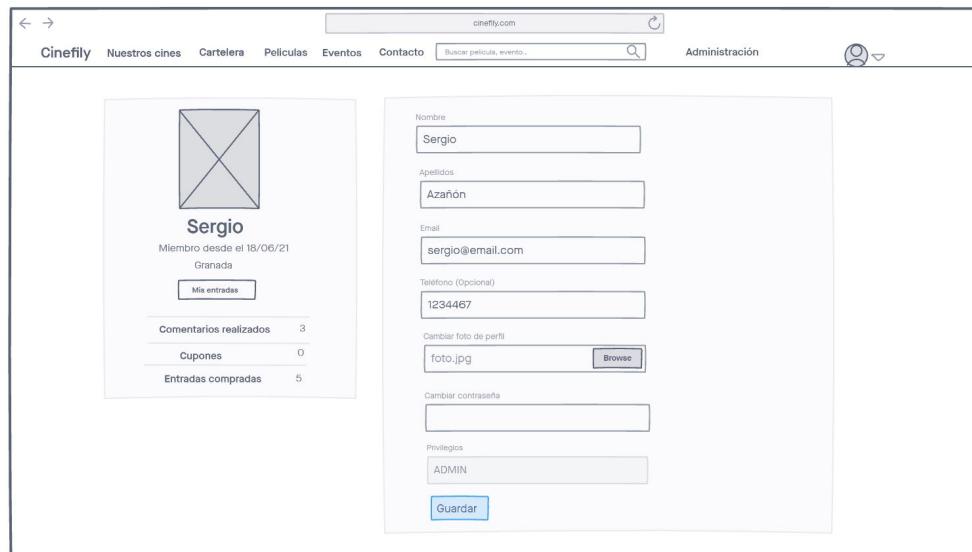
ID	Fecha	Tipo	Información
1	2021/06/02 12:00	EXITO	Se ha añadido una nueva sesión del evento Película con fecha de inicio 2022-12-01 en la sala 1
2	2021/06/10 18:23	WARNING	No se ha podido añadir la sesión a la hora seleccionada porque hay conflictos con otra sesión
3	2021/06/02 19:00	ERROR	Intento de reserva para un asiento ocupado
4	2021/06/02 20:00	EXITO	Se han reservado dos asientos para la sesión con ID 3 y email asociado prueba@email.com
5	2021/09/10 21:34	EXITO	Se han reservado dos asientos para la sesión con ID 8 y email asociado prueba@email.com
6	2021/06/02 12:00	EXITO	Se han reservado dos asientos para la sesión con ID 4 y email asociado prueba@email.com

Pagination controls at the bottom: << Anterior, 1, 2, Siguiente>>

Figura 4.13: Mockup del log del sistema

#### 4.3.8. Mi perfil y entradas

Si un usuario quiere ver su perfil, tendrá que clickar sobre su avatar en el header fijo de la aplicación web y en el menú dropdown seleccionará Mi perfil. Una vez seleccionado podrá visualizar sus datos personales con la posibilidad de cambiarlos y a la izquierda tendrá datos como el número de comentarios, cupones o entradas que ha comprado.



The mockup shows a user profile edit form with the following fields:

- Nombre:** Sergio
- Apellidos:** Azarón
- Email:** sergio@email.com
- Teléfono (Opcional):** 1234467
- Cambiar foto de perfil:** foto.jpg (Browse button)
- Cambiar contraseña:** (empty input field)
- Privilegios:** ADMIN

Buttons: Guardar

Figura 4.14: Mockup del perfil de un usuario

Si el usuario selecciona Mis entradas, podrá visualizar todas las entradas junto con sus detalles que ha comprado desde que es miembro de la aplicación web, cuenta con una paginación de 4 entradas por página.

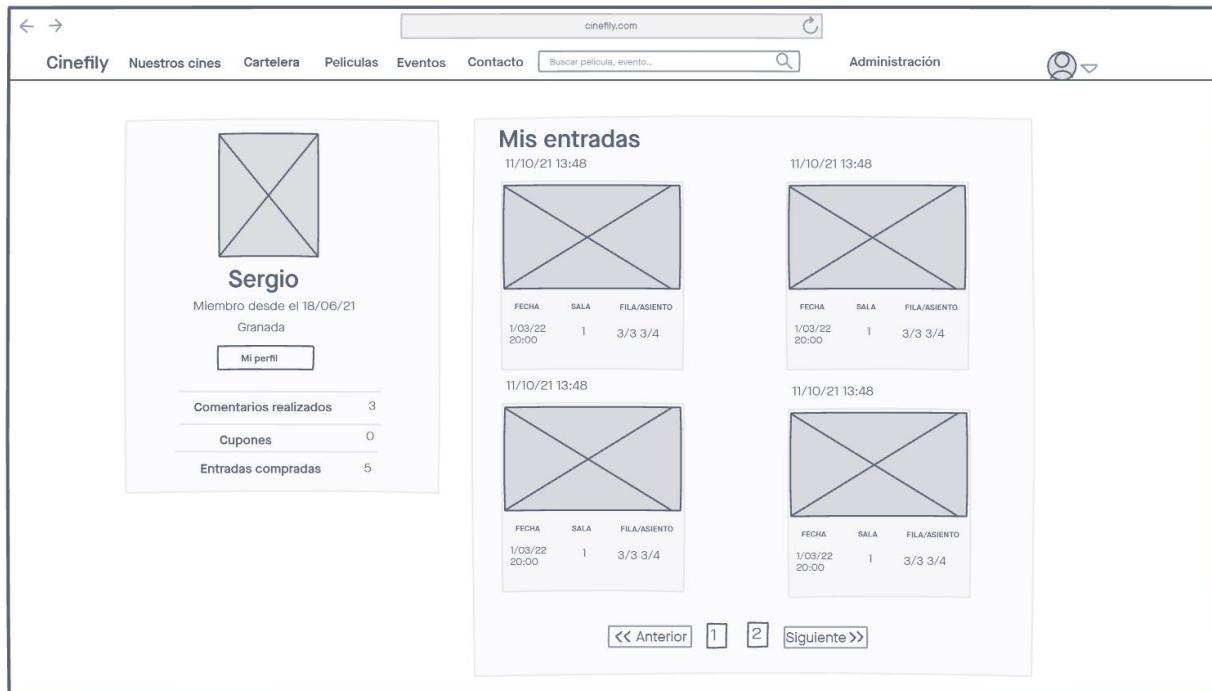


Figura 4.15: Mockup de las entradas de un usuario

#### 4.3.9. Detalles de película/evento

Si el usuario quiere ver los detalles de una película, debe clickar sobre el ícono del ojo en la tabla de película o sobre Leer más en la carta de la película. Una vez dentro podrá ver todos los detalles como el fondo de la película, la frase, el director, actores, trailer, valoraciones, géneros y comentarios de los usuarios. Si estás identificado, puede publicar un comentario o borrar comentarios que sean propios.

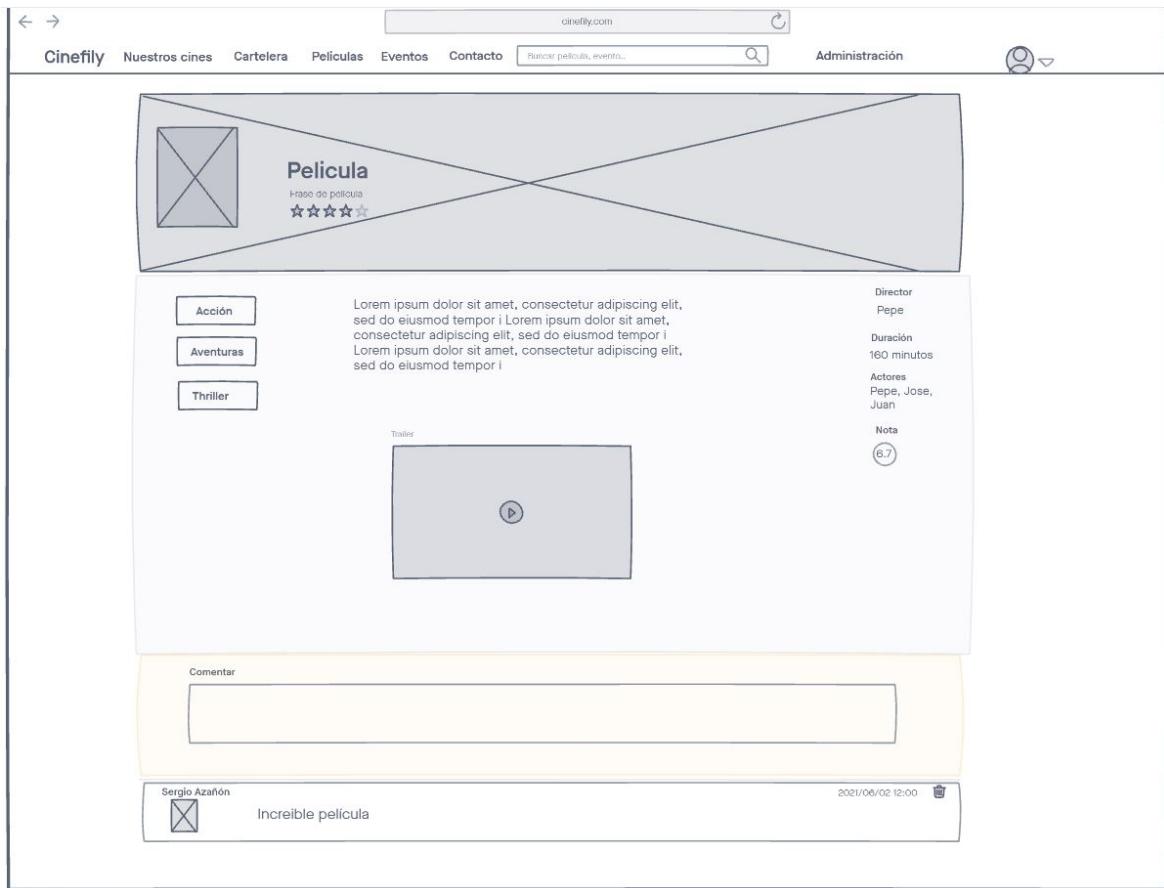


Figura 4.16: Mockup de los detalles de un evento/película

#### 4.3.10. Detalles de película/evento

Si el usuario quiere contactar con los propietarios de la página web puede hacerlo a través de este formulario de contacto donde indicará su email, el motivo del contacto (ya sea para opinar, alquilar la sala para un evento o mejoras) y la descripción de lo que quiere decir.

The mockup shows a website header for 'cinefily.com' with links for 'Cinefily', 'Nuestros cines', 'Cartelera', 'Películas', 'Eventos', 'Contacto', a search bar, and 'Administración'. Below the header is a large central form titled 'Contacto'. The form has three input fields: a dropdown menu labeled 'Motivo' with 'Alquiler de sala' selected, a text input field labeled 'Email', and a text input field labeled 'Descripción'. At the bottom right of the form is a blue 'Enviar' button.

Figura 4.17: Mockup del formulario de contacto

# Capítulo 5

# Implementación

En este capítulo se llevará a cabo la explicación de cómo se ha implementado el diseño anteriormente descrito, cómo se han resuelto ciertos problemas e inconvenientes que han surgido a lo largo de su desarrollo, decisiones técnicas sobre los frameworks o herramientas elegidos y la estructura del proyecto así como la distribución de sus archivos.

## 5.1. Entorno de desarrollo

Para alcanzar los objetivos propuestos del proyecto se necesitará un servidor web para desarrollar todo el *back-end* de la aplicación, esto es, toda la lógica de la aplicación que se ejecutará en dicho servidor. En el *back-end* se utilizará el lenguaje de programación PHP ya que es uno de los lenguajes más potentes para este objetivo, tanto su despliegue como el acceso a base de datos lo hace de forma sencilla y la comunidad es bastante grande por lo que es más fácil resolver dudas concretas. Las aplicaciones web y los sitios web programados en PHP son altamente seguros, ya que es un lenguaje encriptado, y se puede escalar fácilmente para su dinamismo y flexibilidad. Otra opción para el lenguaje era escoger Python, que tiene unas características parecidas al ser también de código abierto, orientado a objetos y fácil de utilizar pero es un lenguaje más orientado hacia ciencia de datos y proyectos científicos.

La elección de un framework es una parte muy importante ya que facilita el mantenimiento del código, es decir, una estructura común para que los desarrolladores no tengan que rehacerlo desde cero y puedan reutilizar el código proporcionado. De esta forma, los frameworks nos permiten recortar gran parte del trabajo y ahorrar mucho tiempo.[19] Los frameworks más populares para PHP son *Laravel*, *Laminas Project*, *CodeIgniter* y *Symfony*. Todoa utilizan el patrón Modelo-Vista-Controlador (MVC), que es un requisito indespensable para su uso. *Laravel* posee ciertas características que incluyen su elegante sintaxis, propensión a escalar y un diseño que favorece el crecimiento progresivo. *Laravel* [20] también tiene una comunidad grande y útil. Pero tiene falta de continuación entre versiones y algunas actualizaciones pueden ser problemáticas. *CodeIgniter* es bastante ligero, es customizable, admite enrutamiento explícito e implícito Pero requiere muchas bibliotecas, está impulsado por una empresa en vez de por una comunidad y no admite suficientemente la ruta HTTP. *Laminas Project (Zend framework)*[21] utiliza un patrón de software típico de modelo-vista-controlador (MVC), el marco viene equipado con componentes de PHP que incluyen inyección de dependencias, despachadores de eventos, validación de entrada, filtrado, paginación y

más. Tiene muchos recursos online y, aunque cuenta con el apoyo de una gran comunidad de usuarios, los desarrolladores a veces se confunden con la documentación excesiva. Por último *Symfony* tiene un gran parecido a Laravel, incluye un conjunto de bibliotecas y componentes PHP reutilizables. Los componentes reutilizables son siempre de gran conveniencia, ya que mitigan la repetición en el proceso de codificación pero es más complejo si no se tiene una experiencia previa con él. Por lo que teniendo en cuenta lo comentado anteriormente se utilizará el framework Symfony muy útil cuando quieras construir proyectos más complejos en función de las necesidades de los clientes y en cuanto a la desventaja de su complejidad, al tener experiencia previa, se mitiga dicha desventaja.

El servidor también tendrá acceso a una base de datos relacional SQL. Como gestor de base de datos se utilizará mySQL ya que es un gestor multiplataforma muy sencillo de implementar tanto para un proyecto más simple como uno complejo, ya que al ser escalable permite aumentar su funcionalidad conforme la aplicación lo requiera.

Para el *front-end*, es decir, la parte que muestra la información de forma visual para el usuario utilizaremos el lenguaje HTML, en concreto, HTML5. Como HTML representa información de forma estática, necesitaremos un lenguaje dinámico como es Javascript junto con Jquery que permite cambiar la información de la página según se necesite. Toda esta parte visual necesitará un estilo y una maquetación, por lo que utilizaremos el lenguaje CSS y el framework Bootstrap. Ya que Bootstrap ofrece mucho potencial con pocas líneas de código además de poder usarlo directamente en el HTML, aunque necesitaremos algo de CSS para estilos más concretos. Utilizaremos el motor de plantillas que forma parte de Symfony por defecto que es Twig. Esto nos permitirá programar los contenidos estáticos, imágenes, estilo...etc

Para el servidor web de desarrollo se utilizará el servidor llocalque ofrece Symfony. Para alojar la base de datos, utilizaremos la herramienta XAMPP, que es un paquete de software libre distribuido por Apache es una distribución de Apache que contiene MariaDB, PHP y Perl.

Otro reto es afianzar los conocimientos de programación en el lenguaje PHP, así como HTML, CSS y JavaScript. También aprender desde cero un framework tan potente como es Symfony, que está muy presente en el mundo laboral.



Figura 5.1: PHP, Symfony y MySQL



Figura 5.2: HTML, CSS, Javascript y Bootstrap

## 5.2. Diagrama de clases

El diagrama de clases es un diagrama orientado al modelo de programación orientado a objetos, ya que define las clases que se utilizarán cuando se pase a la fase de construcción y la manera en que se relacionan las mismas. Tiene una utilidad similar que el diagrama entidad-relación pero en Lenguaje Unificado de Modelado(UML), la representación de datos y su interacción. Ambos diagramas muestran el modelo lógico de los datos de un sistema.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. [22] UML proporciona mecanismos para representar los miembros de la clase, como atributos y métodos, así como información adicional sobre ellos. [23]

El proyecto sigue una arquitectura de tipo MVC, es decir, se distinguen tres tipos de clases: controladores, modelos y vistas. Los controladores actúan en toda la lógica de la aplicación y trasladan la información a la vista para que el usuario interactúe con ella. La información almacenada en la base de datos la gestionan los modelos, que se relacionan con los controladores para manipular dicha información.

A continuación se observa en la figura 5.3 la representación de los controladores, los modelos y sus respectivas relaciones. Las vistas no se recogen en el diagrama de clases ya que no pertenecen a ellas, al ser una representación de la información ofrecida por el controlador tras procesar los datos del modelo, no es necesario incluirlas.

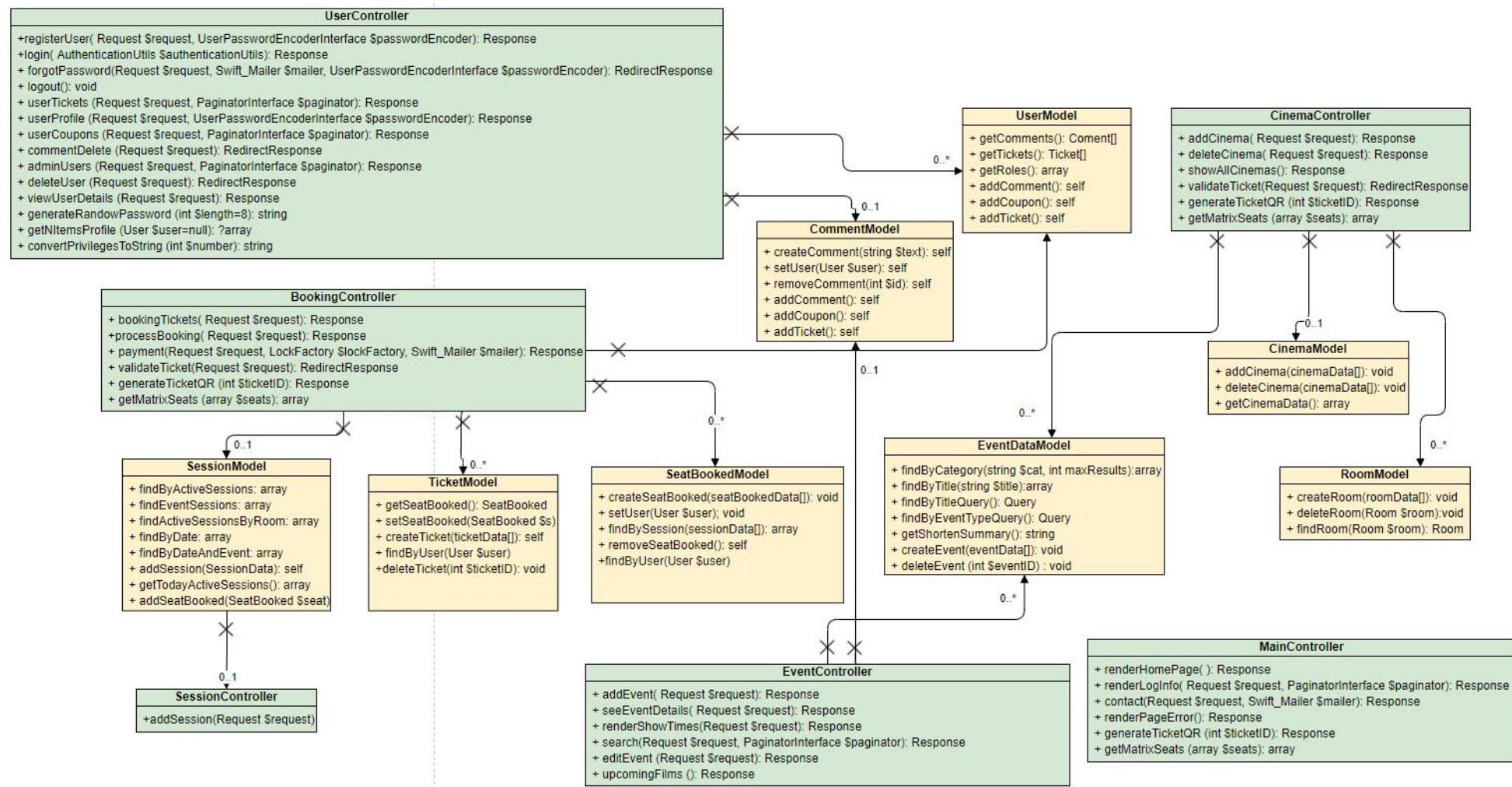


Figura 5.3: Diagrama UML de clases

Brevemente se explicarán cada uno de los controladores y modelos expuestos en el diagrama anterior:

### Controladores

- **UserController:** se encarga de todas las interacciones del usuario con la aplicación web: registro, identificación, olvidar la contraseña, entradas compradas por el usuario, comentar películas o eventos, cupones de descuento asociados y el cierre de sesión.
- **BookingController:** es el encargado de la gestión de reservas para la compra de entradas, es decir, la selección de asientos de forma interactiva, procesar dicha reserva con posibilidad de aplicar cupones de descuento y la pasarela de pago para la confirmación de dicha reserva.
- **EventController:** es el que se ocupa de las operaciones relacionadas con películas o eventos, es decir, añadir, editar y eliminar (CRUD). Además de renderiza toda su información y sus horarios, búsqueda de películas por título y conexión con la API de TMDB para consultar las películas que se estrenarán próximamente.
- **SessionController:** únicamente se ocupa de la creación, edición y eliminación de sesiones para películas y eventos.
- **MainController:** se ocupa de la renderización de la página principal con todos los eventos divididos por categorías, de la página de error cuando ha ocurrido algún fallo en la aplicación o se ha intentado buscar una página que no existe, renderiza la página de información del log para el administrador y de la página de contacto para enviar opiniones, mejoras o solicitud para alquilar una sala.
- **CinemaController:** es el responsable de las operaciones de añadir un nuevo cine o eliminarlo y la renderización de la página que muestra toda la información sobre los cines almacenados en la aplicación.

### Modelos

- **UserModel:** devuelve todos los datos relacionados con el usuario además de la creación de usuarios, comentarios, cupones y entradas asociadas al mismo. Está relacionado con *UserController* y *BookingController*
- **CommentModel:** crea comentarios en las películas o eventos asociados al usuario que lo ha escrito, además si el usuario es el propietario de dicho comentario lo podrá eliminar. Está relacionado con *UserController* y *EventController*
- **SessionModel:** tiene varios métodos muy útiles para la búsqueda de sesiones como son: encontrar las sesiones activas, sesiones activas de una película en concreto, sesiones activas por cine y sala, sesiones por fecha, sesiones por fecha y evento. Además crea las sesiones con sus respectivos datos. Está relacionado con *BookingController* y *SessionController*

- **TicketModel:** obtiene el asiento reservado asociado a ese ticket, encuentra los tickets asociados a un usuario, además de crear un ticket. Sólo se podría eliminar un ticket si se elimina una sesión que está asociada o el cine asociado. Está relacionado con *BookingController*.
- **SeatBookedModel:** devuelve el asiento reservado junto con sus datos. Crea y elimina asientos reservados y además puede buscar asientos reservados por sesión, para poder visualizar en la elección de asientos interactiva los que están ocupados. Está relacionado con *BookingController*
- **EventDataModel:** además de las operaciones habituales de CRUD, tiene métodos para buscar por categoría, por título, por tipo de evento y si está o no en cartelera. Está relacionado con *EventController* y *CinemaController*
- **RoomModel:** se usa para la creación y eliminación de salas asociadas a un cine concreto. Además tiene métodos para encontrar una sala por su ID o por su número de sala. Está relacionado con *CinemaController*.
- **CinemaModel:** se usa para la creación y eliminación de cines en el sistema además de obtener la información de cada cine. Está relacionado con *CinemaController*.

### 5.3. Base de datos

#### 5.3.1. Implementación de la base de datos

La implementación de la base de datos se realiza mediante Doctrine [24], una colección de proyectos creados para PHP que usan varios frameworks (entre ellos Symfony), enfocado principalmente en el almacenamiento de bases de datos y mapeo de objetos. Los proyectos centrales son Object Relational Mapper (ORM) y Database Abstraction Layer (DBAL) sobre el que se basa. Symfony proporciona todas las herramientas que necesitas para usar bases de datos en tus aplicaciones gracias a Doctrine , el mejor conjunto de bibliotecas PHP para trabajar con bases de datos. Estas herramientas admiten bases de datos relacionales como MySQL y PostgreSQL y también bases de datos NoSQL como MongoDB.

En cada alteración de la base de datos se guarda una versión de la migración, lo que permite tener un control de versiones por si en algunas de las alteraciones hay un fallo o se quiere volver a la anterior versión. Además Doctrine hace de “fachada” entre la base de datos y la aplicación. Un código de ejemplo de una de las migraciones es la creación de la tabla cupón:

codigo/Version20211020223130.php

```
1  public function up(Schema $schema) : void
2  {
3      // this up() migration is auto-generated, please modify it to your needs
4      $this->addSql('CREATE TABLE coupon (id INT AUTO_INCREMENT NOT NULL,
5          user_id INT DEFAULT NULL, code VARCHAR(255) NOT NULL, active TINYINT(1) NOT
6          NULL, INDEX IDX_64BF3F02A76ED395 (user_id), PRIMARY KEY(id)) DEFAULT
7          CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
8
9      $this->addSql('ALTER TABLE coupon ADD CONSTRAINT FK_64BF3F02A76ED395
10         FOREIGN KEY (user_id) REFERENCES user (id)');
```

```
6 }
```

Tras las migraciones, dichas sentencias se convierten en código SQL para crear las respectivas tablas, a continuación se muestra todo el código SQL necesario para crear las tablas de las entidades y sus relaciones entre ellas en la base de datos del proyecto:

```
codigo/cinelify.sql
```

```
1 CREATE TABLE `cinema` (
2     `id` int(11) NOT NULL,
3     `location` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
4     `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
5     `tickets_price` double NOT NULL
6 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
7
8 --
9 CREATE TABLE `comment` (
10    `id` int(11) NOT NULL,
11    `user_id` int(11) NOT NULL,
12    `text` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
13    `event_id` int(11) DEFAULT NULL,
14    `created_at` datetime NOT NULL,
15    `tmdb_id` int(11) DEFAULT NULL
16 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
17
18 CREATE TABLE `coupon` (
19     `id` int(11) NOT NULL,
20     `user_id` int(11) DEFAULT NULL,
21     `code` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
22     `active` tinyint(1) NOT NULL,
23     `discount` int(11) NOT NULL,
24     `expiration` datetime DEFAULT NULL
25 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
26
27 CREATE TABLE `event_data` (
28     `id` int(11) NOT NULL,
29     `title` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
30     `type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
31     `gender` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
32     `description` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,
33     `duration` int(11) DEFAULT NULL,
34     `release_date` date DEFAULT NULL,
35     `actors` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,
36     `poster_photo` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
37     `rating` double DEFAULT NULL,
38     `status` tinyint(1) NOT NULL,
39     `age_rating` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
40     `director` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,
41     `tag_line` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,
42     `backdrop_path` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
43     `youtube_trailer` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL
44 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
45
46 CREATE TABLE `log_info` (
47     `id` int(11) NOT NULL,
48     `date` datetime NOT NULL,
49     `info` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
50     `type` int(11) NOT NULL
51 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
52
53 CREATE TABLE `room` (
54     `id` int(11) NOT NULL,
55     `cinema_id` int(11) NOT NULL,
56     `n_rows` int(11) NOT NULL,
57     `n_columns` int(11) NOT NULL,
58     `number` int(11) NOT NULL
59 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
60
61 CREATE TABLE `seat_booked` (
62     `id` int(11) NOT NULL,
63     `session_id` int(11) NOT NULL,
64     `ticket_id` int(11) NOT NULL,
65     `row` int(11) NOT NULL,
66     `number` int(11) NOT NULL,
67     `room_id` int(11) NOT NULL
68 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
69
70 CREATE TABLE `session` (
71     `id` int(11) NOT NULL,
72     `event_id` int(11) NOT NULL,
73     `cinema_id` int(11) NOT NULL,
74     `room_id` int(11) NOT NULL,
75     `schedule` datetime NOT NULL,
76     `language` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
77     `schedule_end` datetime NOT NULL
78 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
79
80 CREATE TABLE `ticket` (
81     `id` int(11) NOT NULL,
82     `user_id` int(11) DEFAULT NULL,
83     `price` double NOT NULL,
84     `qr_code` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,
85     `sale_date` datetime DEFAULT NULL,
86     `session_id` int(11) NOT NULL
87 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
88
89 CREATE TABLE `user` (
90     `id` int(11) NOT NULL,
91     `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
92     `surname` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
93     `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
94     `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
95     `privileges` int(11) DEFAULT NULL,
```

```
96   `phone_number` varchar(14) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
97   `profile_photo` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
98   `created_at` datetime NOT NULL,
99   `city` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL
100 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
101 --
102 -- Filtros para la tabla `comment`
103 --
104 ALTER TABLE `comment`
105   ADD CONSTRAINT `FK_9474526C71F7E88B` FOREIGN KEY (`event_id`) REFERENCES `event_data`(`id`),
106   ADD CONSTRAINT `FK_9474526CA76ED395` FOREIGN KEY (`user_id`) REFERENCES `user`(`id`);
107 --
108 --
109 -- Filtros para la tabla `coupon`
110 --
111 ALTER TABLE `coupon`
112   ADD CONSTRAINT `FK_64BF3F02A76ED395` FOREIGN KEY (`user_id`) REFERENCES `user`(`id`);
113 --
114 --
115 -- Filtros para la tabla `room`
116 --
117 ALTER TABLE `room`
118   ADD CONSTRAINT `FK_729F519BB4CB84B6` FOREIGN KEY (`cinema_id`) REFERENCES `cinema`(`id`);
119 --
120 --
121 -- Filtros para la tabla `seat_booked`
122 --
123 ALTER TABLE `seat_booked`
124   ADD CONSTRAINT `FK_A6DA6E0354177093` FOREIGN KEY (`room_id`) REFERENCES `room`(`id`),
125   ADD CONSTRAINT `FK_A6DA6E03613FECDF` FOREIGN KEY (`session_id`) REFERENCES `session`(`id`),
126   ADD CONSTRAINT `FK_A6DA6E03700047D2` FOREIGN KEY (`ticket_id`) REFERENCES `ticket`(`id`);
127 --
128 --
129 -- Filtros para la tabla `session`
130 --
131 ALTER TABLE `session`
132   ADD CONSTRAINT `FK_D044D5D454177093` FOREIGN KEY (`room_id`) REFERENCES `room`(`id`),
133   ADD CONSTRAINT `FK_D044D5D471F7E88B` FOREIGN KEY (`event_id`) REFERENCES `event_data`(`id`),
134   ADD CONSTRAINT `FK_D044D5D4B4CB84B6` FOREIGN KEY (`cinema_id`) REFERENCES `cinema`(`id`);
135 --
136 --
```

```

137 -- Filtros para la tabla `ticket`
138 --
139 ALTER TABLE `ticket`
140   ADD CONSTRAINT `FK_97AOADA3613FECDF` FOREIGN KEY (`session_id`) REFERENCES `session`(`id`),
141   ADD CONSTRAINT `FK_97AOADA3A76ED395` FOREIGN KEY (`user_id`) REFERENCES `user`(`id`);
142 COMMIT;

```

### 5.3.2. Creación de entidades y sus relaciones en Symfony

Aunque el paso a tablas de forma manual es muy necesario para conocer la estructura de las entidades con sus relaciones cardinales, el framework Symfony que es el que ha sido escogido para el proyecto, tiene un sistema para crear las entidades con sus atributos y relaciones guiado. Con el comando `php bin/console make:entity` se puede crear una entidad y como se observa en la figura 5.4, está compuesto por una serie de preguntas para saber como se llama la entidad, qué atributos tiene y cual es su tipo, si está relacionado con alguna entidad y que tipo de relación tiene en el caso de estar relacionado.

```

Add another property? Enter the property name (or press <return> to stop adding fields):
> rating

Field type (enter ? to see all types) [string]:
> ?

Main types
* string
* text
* boolean
* integer (or smallint, bigint)
* float

Relationships / Associations
* relation (a wizard will help you build the relation)
* ManyToOne
* OneToMany
* ManyToMany
* OneToOne

Array/Object Types
* array (or simple_array)
* json
* object
* binary
* blob

Date/Time Types
* datetime (or datetime_immutable)
* datetimetz (or datetimetz_immutable)
* date (or date_immutable)
* time (or time_immutable)
* dateinterval

Other Types
* ascii_string
* decimal
* guid
* json_array

```

Figura 5.4: Ayuda para indicar de qué tipo es el atributo

Como se observa en la figura 5.5, si elegimos que el atributo es de tipo relacional, aparece una pantalla de ayuda con una tabla de correspondencia entre los tipos de relaciones con nuestras dos

entidades relacionadas. En el ejemplo que podemos observar, la relación es entre la entidad User y la entidad Ticket. Hay cuatro tipo de relaciones:

- La relación ManyToOne (muchos a uno) nos indica que sería que cada usuario estaría relacionado exactamente con un solo ticket y que cada ticket puede estar relacionado con varios usuarios.
- La relación OneToMany (uno a muchos) nos indica que un usuario puede estar relacionado con varios tickets pero que cada ticket está relacionado con un solo usuario.
- La relación ManyToMany (muchos a muchos) nos indica que un usuario puede estar relacionado con varios tickets y que cada ticket puede estar relacionado con varios usuarios.
- Y por último la relación OneToOne (uno a uno) nos indica que un usuario está relacionado exactamente con un solo ticket y un ticket está relacionado sólo con un usuario.

Acorde con la esquema entidad relación de nuestra base de datos, la relación correspondiente es OneToMany ya que a un usuario pueden pertenecerle varios tickets de cine, sin embargo a un ticket sólo le puede pertenecer un usuario.

```
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

What class should this entity be related to?:
> Ticket

What type of relationship is this?
-----
Type      Description
-----
ManyToOne  Each User relates to (has) one Ticket.
            Each Ticket can relate to (can have) many User objects

OneToMany   Each User can relate to (can have) many Ticket objects.
            Each Ticket relates to (has) one User

ManyToMany  Each User can relate to (can have) many Ticket objects.
            Each Ticket can also relate to (can also have) many User objects

OneToOne    Each User relates to (has) exactly one Ticket.
            Each Ticket also relates to (has) exactly one User.

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> OneToMany

A new property will also be added to the Ticket class so that you can access and set the related User object from it.

New field name inside Ticket [user]:
>

Is the Ticket.user property allowed to be null (nullable)? (yes/no) [yes]:
> yes

updated: src/Entity/User.php
updated: src/Entity/Ticket.php
```

Figura 5.5: Ayuda para saber qué tipo de relación tiene

## 5.4. Desarrollo de aplicación web

En esta sección se explicarán los procesos más relevantes de la plataforma web y su funcionamiento con el código asociado. El código completo se encuentra en el Anexo [8]. Para el servidor de desarrollo se ha utilizado el que trae *Symfony* por defecto [25], que permite desplegar el proyecto en local antes de desplegarlo en un servidor en producción. Para alojar la base de datos en local he utilizado XAMPP [26], que es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.

### 5.4.1. Gestión de usuarios

Como se indicó en el capítulo de Análisis de Requisitos, habrá dos tipos de usuarios con distintos tipos de privilegios. Un usuario con privilegios de administrador (*USER\_ADMIN*) y un usuario sin privilegios (*USER*). Para las rutas que sólo sean para usuarios administradores se utilizará el fragmento de código 5.1 que deniega el acceso a todo usuario que no esté identificado o que esté identificado sin privilegios de administrador.

Listing 5.1: UserController.php

```

1 if (!(!$this->getUser() || ($this->getUser() &&
2     !in_array(User::ROLE_ADMIN, $this->getUser()->getRoles(), true))):
3     $this->addFlash(
4         'error', 'No tienes acceso a la ruta ' . $request->getBaseUrl());
5     return $this->redirectToRoute('home');
6 endif;
```

Todos los procesos relacionados con usuarios lo maneja el controlador de usuarios (*UserController*). Para identificarse en el sistema, *Symfony* utiliza varios archivos para administrar la seguridad de la identificación en el sistema. Cuenta con un sistema de *hashing* llamado *bcrypt* [27] que es una función de hash de contraseña basada en el cifrado Blowfish. Incorpora una *salt*(datos aleatorios) para proteger contra los ataques de la tabla de arco iris , además es una función adaptativa con el tiempo, esto lo hace resistente a los ataques de búsqueda de fuerza bruta incluso con una potencia de cálculo creciente.

Se realiza el *hashing* a las contraseñas para evitar que un atacante que haya comprometido la base de datos use las contraseñas. Por lo que se aplica en el lado del servidor para que el atacante primero tenga que descifrarlas para usarlas, si lo hacemos en la parte del cliente el *hash* estaría tal y como está almacenado en la base de datos.

Para el *logout* simplemente hay que indicar el *path* en el sistema de *firewall* de *Symfony*.

### 5.4.2. Asignación de salas para sesiones

Cuando se quiere añadir una sesión nueva a la cartelera se debe escoger la película, el cine, la fecha y hora y el idioma de la sesión. Por lo que en el proceso para asignarle una sala hay que realizar ciertas comprobaciones.

Una vez que el usuario con privilegios de administrador ha introducido todos los datos descritos anteriormente, se realizan ciertas comprobaciones cómo que la fecha no sea anterior a la actual. Una vez realizadas, se encuentran todas las sesiones activas del cine elegido para esa fecha, es decir,

todas las sesiones de ese cine para la fecha elegida. Se calcula la hora a la que termina el evento para tener el tramo horario que ocupará la sesión en la sala.

Si las sesiones activas están vacías significa que no hay ninguna sesión para esa fecha aún por lo que se puede ir a la primera sala. De lo contrario recorremos todas las salas del cine y hacemos una consulta para comprobar si en ese cine y sala en concreto hay alguna sesión entre la hora de inicio de la película y la hora final de la película como muestra el fragmento de código 5.2.

Listing 5.2: SessionRepository.php

```

1  public function findSessionsByRoom(Cinema $cinema, Room $room,
2                                     DateTime $start, DateTime $end): ?array
3  {
4      $qb=$this->createQueryBuilder('s');
5      return $qb
6          ->where(':start BETWEEN s.schedule AND s.schedule_end')
7          ->orWhere(':end BETWEEN s.schedule AND s.schedule_end')
8          ->orWhere('s.schedule BETWEEN :start AND :end')
9          ->andWhere($qb->expr()->eq('s.cinema', ':cinema'))
10         ->andWhere($qb->expr()->eq('s.room', ':room'))
11         ->setParameter('start', $start)
12         ->setParameter('end', $end)
13         ->setParameter('cinema', $cinema)
14         ->setParameter('room', $room)
15         ->getQuery()->execute()
16     ;
17 }
```

Si la consulta devuelve un *array* vacío significa que no hay ninguna sesión en ese tramo por lo que la sesión se añadiría a esa sala. En el fragmento de código 5.3 se puede observar como sería el proceso completo obviando algunas partes.

Listing 5.3: SessionController.php

```

1 ...
2 $sessions = $this->getDoctrine()->getRepository(Session::class)->
3     findActiveSessions($cinema,$date);
4 // Comprobaciones
5 if ($formData['schedule'] < $now):
6 ...
7 ...
8 elseif ($sessions === null && $rooms === null):
9 ...
10    $session = new Session($cinema,$event,$formData['language'], $schedule_start,
11                           $schedule_end);
12    $session->setRoom($rooms[0]);
13 ...
14 ...
15
16    return $this->redirectToRoute('home');
```

```

18     else:
19         $assigned = false;
20         foreach ($rooms as $room):
21             if (empty($this->getDoctrine()->getRepository(Session::class)
22                     ->findSessionsByRoom($cinema, $room,
23                         $schedule_start, $schedule_end))):
24                 $session = new Session($cinema,$event,
25                     formData['language'], $schedule_start
26                     , $schedule_end);
27                 $session->setRoom($room);
28
29                 if (!$event->getStatus()):
30                     $event->setStatus(true);
31                     $em->persist($event);
32                 endif;
33
34                 $assigned = true;
35                 ....
36                 break;
37             endif;
38         endforeach;
39
40         if ($assigned):
41             ...
42             return $this->redirectToRoute('home');
43         else:
44             ...
45         endif;
46     endif;

```

#### 5.4.3. Proceso de reserva de entradas

El proceso de reserva de entradas es el más importante del proyecto ya que abarca la principal característica de él: selección de cine para elegir sesiones, selección de una sesión de cine en concreto para reservar entradas, selección de asientos de forma interactiva y pasarela de pago de las entradas generadas.

Se crea un formulario en el controlador de Eventos para que el usuario elija el cine y la fecha de las cuáles quiere saber las sesiones, el formulario recibe un listado con todos los cines de la plataforma y un rango de los años fijo que será entre este año y 2023.

codigo/EventController.php

```

1
2     $form = $this->createFormBuilder(array('csrf_protection' => FALSE))
3             ->setMethod(Request::METHOD_GET)
4             ->setAction($this->generateUrl(static::ROUTE_SHOW_TIMES))
5             ->add('cinema', ChoiceType::class, array(
6                 'label' => 'Cine',
7                 'choices' => $cinemas
8             ))
9             ->add('schedule', DateType::class, array(

```

```

10         'label' => 'Fecha',
11         'placeholder' => [
12             'year' => 'Año', 'month' => 'Mes', 'day' => 'Dia'
13         ],
14         'years' => range(2021, 2023),
15         'data' => new DateTime()
16     ))
17     ->add('submit', SubmitType::class, array(
18         'label' => 'Buscar',
19     ))
20     ->getForm();
21
22     $form->handleRequest($request);

```

Una vez el formulario se ha enviado, se realiza una búsqueda de las sesiones activas del cine y fechas elegidos en el repositorio de las sesiones. Como fecha inicial será la fecha escogida por el usuario y como fecha final el final del día de esa fecha, así se buscará las sesiones que comprenden entre todas las horas de un día completo.

codigo/SessionRepository.php

```

1     public function findByDate(Cinema $cinema, DateTime $start, DateTime $end): ?
2     array
3     {
4
5         $qb=$this->createQueryBuilder('s');
6         return $qb
7             ->where('s.schedule BETWEEN :start AND :end')
8             ->andWhere($qb->expr()->eq('s.cinema',':cinema'))
9             ->setParameter('start',$start)
10            ->setParameter('end',$end)
11            ->setParameter('cinema',$cinema)
12            ->getQuery()->execute()
13    ;

```

Si hay sesiones activas para esa fecha y cines se pasa a la vista para que el usuario seleccione la sesión deseada. Una vez que se ha seleccionado una sesión concreta, se le pasa como parámetro la sesión al controlador de reservas, el cuál obtiene todos los datos asociados a esa sesión. El controlador obtiene todos los asientos reservados para esa sesión y ejecuta un bucle *foreach* recorriendo los asientos reservados para usar como índices la fila y columna en una matriz booleana. Se puede observar este fragmento en el código de abajo.

codigo/BookingController.php

```

1     if ($session !== null):
2         $cinema = $session->getCinema();
3         $event = $session->getEvent();
4         $room = $session->getRoom();
5         if ($room !== null):
6             $matrixStatusSeats=array();
7             $reservedSeats = $this->getDoctrine()->getRepository(SeatBooked::
class)->findBy(array('session' => $session));

```

```

8         foreach ($reservedSeats as $seatBooked):
9             $matrixStatusSeats[$seatBooked->getRow()] [$seatBooked->
10                getNumber()]=true;
11            endforeach;
12        endif;
13    endif;

```

Todos los datos asociados a la sesión junto con la matriz de estados de los asientos de la sesión los recibe la plantilla *twig*. En dicha vista, el usuario tendrá que introducir su email (si está identificado en el sistema aparecerá su email asociado) y el número de entradas que desee, una vez introducidos los datos, se comprobarán que son correctos con una función en *Javascript* y se le alertará de que ya puede seleccionar los asientos de forma interactiva.

```

1     function checkData() {
2         // Si no se ha introducido ningún dato
3         if ((($("#userEmail").val().length == 0) || ($("#numSeats").val() .
4             length == 0)) {
5             alert("Introduce tu email y el número de asientos");
6         } else if ((($("#numSeats").val() < 0) || ($("#numSeats").val() > 10))
7         {
8             alert("Introduce un número de asientos válido");
9         } else {
10             $(".inputForm *").prop("disabled", true);
11             $(".seatStructure *").prop("disabled", false);
12             document.getElementById("notification").innerHTML = "<b style='
13 margin-top:2px;background:lightyellow;'>" +
14                 "<i class='fas fa-hand-pointer'></i> Selecciona tus asientos
15             </b>";
16         }
17     }

```

codigo/room\_booking.html.twig

Toda la parte interactiva será una tabla donde cada celda será un asiento, si está libre se rellenará con *input* de tipo *checkbox*, sin embargo si está ocupado se rellanará con un *div* genérico asociado a una clase que creará una casilla de las mismas dimensiones de un *checkbox* pero de color rojo.

room\_booking.html.twig:

```

{% for i in 1..room.getNRows() %}
<tr>
    <td class="text-white">{{ i }}</td>
    {% for j in 1..room.getNColumns() %}
        <td>
            {% if matrixStatusSeats[i][j] is defined and
               matrixStatusSeats[i][j] == true %}
                <div class="smallBox redBox"></div>
            {% else %}
                <input type="checkbox" class="seats"
                   value="Fila {{ i }} Asiento {{ j }},">
            {% endif %}
        </td>
    {% endfor %}
</tr>

```

```

        </td>
    {% endfor %}
</tr>
{% endfor %}

```

Una vez que el usuario ha seleccionado los asientos que desea reservar para la sesión y haya confirmado dicha selección, se le redirigirá a una ruta en el controlador de reservas para mostrar toda la información sobre la compra de dichas entradas y la posibilidad de aplicar un cupón de descuento. La ruta muestra toda la información al usuario sobre la compra y genera la entrada que compraría con su QR asociado. Para la generación del código QR se ha usado la API de la web *QR Code Generator* [28], la función que se muestra en el código de abajo llama a la API con el *token* generado al crearse una cuenta en dicha web y como URL será una ruta para validar el ticket de compra que recibe como parámetro el ID del ticket.

codigo/BookingController.php

```

1     public function generateTicketQR(int $ticketID): string
2     {
3         $result = NULL;
4
5         $url = 'https://api.qr-code-generator.com/v1/create?access-token=' .
6             static::API_QR_GENERATOR_KEY;
7         $response = $this->client->request(
8             'POST',
9             $url,
10            [
11                'body' => [
12                    'frame_name' => 'no-frame',
13                    'qr_code_text' => 'ticket/validate?id=' . $ticketID,
14                    'image_format' => 'SVG',
15                    'qr_code_logo' => 'scan-me-square'
16                ]
17            ]
18        );
19
20
21        if ($response->getStatusCode() === EventController::SUCCESS_STATUS_CODE):
22            $result = $response->getContent();
23        endif;
24
25        return $result;
26    }

```

Para aplicar el cupón introducido por el usuario (si introduce alguno), una función en *javascript* comprueba que el código de cupón introducido coincide con alguno de los cupones asociados al usuario, si es así comprueba que está activo, es decir, que no se ha utilizado nunca. Si el cupón es válido, el precio total de las entradas cambiará, de lo contrario se le informará de que el cupón introducido no es válido.

Para la pasarela de pago, se ha utilizado una ventana *modal* donde te da la posibilidad de pagarla a través de *PayPal*, tarjeta de crédito, *Sofort* o pagarla en taquilla. Las tres opciones

que implican pagar a través de la plataforma se han implementado con *Paypal Checkout* y *Smart Payment Button* [29]. El código del proceso se puede observar justo abajo con la función de *javascript initPaypalButton*. Consiste en crear el botón de *Paypal* con el estilo que se desee, crear la configuración de la transacción con *createOrder* estableciendo la divisa y la cantidad y capturar la transacción con *onApprove* redirigiendo a la ruta que deseemos si el proceso de compra ha sido satisfactorio. El *client-id* para que las compras vayan hacia una cuenta se encuentra en la plantilla base del proyecto.

```

1  function initPayPalButton() {
2      paypal.Buttons({
3          style: {
4              shape: 'rect',
5              color: 'gold',
6              layout: 'vertical',
7              label: 'paypal',
8          },
9
10         createOrder: function (data, actions) {
11             var amount={{ totalPrice }};
12             return actions.order.create({
13                 purchase_units:
14                     [{"amount": {"currency_code": "EUR", "value": amount}}]
15             });
16         },
17
18         onApprove: function (data, actions) {
19             return actions.order.capture().then(function (orderData) {
20
21                 // Full available details
22                 console.log('Capture result', orderData,
23                             JSON.stringify(orderData, null, 2));
24
25                 // Show a success message within this page, e.g.
26                 const element = document.getElementById('paypal-button-container'
27 );
27                 element.innerHTML = '';
28                 element.innerHTML = '<h3>Gracias por el pago!</h3>';
29
30                 window.location= window.location.origin +
31                     "/booking/payment?method=cash&seats=" + "{{ seats|join(', ') }}"
32                     +"&id_session=" + {{ session.getId() }} +'&method=paypal' +
33                     '&price=' +{{ cinema.getTicketsPrice() }}
34                     +'&email=' + "{{ email }}" + '&id_coupon=' + resultCoupon;
35             });
36         },
37
38         onError: function (err) {
39             console.log(err);
40         }
41     }).render('#paypal-button-container');
```

42 }

Listing 5.4: process\_booking.html.twig

Después de que el proceso de compra y pago se haya completado correctamente, el controlador de reservas se encargará de crear los *tickets* de la reserva asociados al usuario, los asientos reservados y enviar el email con las entradas asociadas.

Para este proceso se utilizarán los candados que proporciona Symfony para evitar la concurrencia y garantizar el acceso exclusivo a algún recurso compartido, que en este caso son la reserva de asientos. Ya que existe la posibilidad de que otro usuario mientras estamos realizando el proceso de compra, pueda elegir los mismos asientos que ha elegido el usuario que aún no ha finalizado el proceso. Los candados se crean usando una clase LockFactory [30]. El bloqueo se crea llamando al método `createLock()`. Su primer argumento es una cadena arbitraria que representa el recurso bloqueado. Luego, una llamada al método `acquire()` intentará adquirir el bloqueo, a no ser que le pases como argumento `true`, que adquiere el cerrojo y hasta que no lo libere no podrá obtenerlo el siguiente recurso.

Recorremos la matriz de asientos, que en este caso son los asientos que ha elegido el usuario y buscamos si hay alguno reservado en esa fila y columna, que no debería ya que se está haciendo con candados este proceso pero se comprueba igualmente. Si está vacío, entonces creamos el objeto *Ticket* asociándolo al usuario si está identificado. Si hay utilizado algún cupón, lo ponemos a no activo y cambiamos el precio del *Ticket* comprado. Creamos el objeto *SeatBooked* con todos los datos junto con la fila y columna y generamos el QR para el *Ticket*.

Listing 5.5: BookingController.php

```
1 $em = $this->getDoctrine()->getManager();
2 $lock = $lockFactory->createLock('seat-booking');
3
4 foreach ($matrixSeats as $row => $columns):
5     foreach ($columns as $column):
6         $lock->acquire(true);
7         $seatBooked = $this->getDoctrine()->getRepository(SeatBooked::class)->
8             findBy(
9                 array('session' => $session, 'row' => (int)$row,
10                   'number' => (int)$column, 'room' => $room->getId()));
11         if (empty($seatBooked)):
12             $ticket = new Ticket($session,new DateTime());
13             if ($this->getUser()):
14                 $ticket->setUser($this->getUser());
15             endif;
16
17             if ($couponID !== 0):
18                 $coupon =
19                     $this->getDoctrine()->getRepository(Coupon::class)
20                     ->find($couponID);
21                 if ($coupon !== null):
22                     $coupon->setActive(false);
23                     $ticket->setPrice(
24                         $price - ((($coupon->getDiscount() / 100)
25                         * $price));
26                     $em->persistent($coupon);
27             endif;
28         endif;
29     endforeach;
30 endforeach;
31
32 $lock->release();
33
34 $em->flush();
35
36 return $ticket;
```



```

16
17 if ($mailer->send($emailMessage)):
18 ...
19 else:
20 ...
21 endif;

```

#### 5.4.4. The Movie Database API

Para obtener las películas que se estrenarán próximamente se utiliza la API de The Movie Database (TMDB) [31], esta API proporciona un listado de métodos de películas, televisión, actores e imágenes que consultar. Para usar la API hace falta registrarse y obtener un token que se utilizará de identificador al usar los diferentes métodos. Los métodos que se han usado en este proyecto son la de obtener los datos de una película por su ID y obtener un listado de las películas que se estrenarán en los cines próximamente. Inicialmente se pensó combinar películas y eventos de la base de datos del proyecto con películas que se recogían de la API de TMDB, así no sería necesario tener almacenado datos de película para su visualización pero al usar las películas para sesiones, comentarios y entradas era complejo su administración ya que había que tener al menos una tabla con IDs de películas de TMDB para tener una referencia a la hora de crear sesiones o comentarios que estaban relacionados. A pesar de no haberlo tenido en cuenta para todos los procesos comentados anteriormente, la llamada está incluida en el código del proyecto y la ruta que procesa los datos de las películas funciona si se le pasan los datos de una película obtenida por TMDB.

El método que sí se utiliza es el de obtener el listado de películas que se estrenarán próximamente como se observa en el fragmento de código 5.7. La llamada es muy simple, como parámetros recibe el token que nos proporcionan, el idioma y la página que se quiere obtener (cada página contiene 20 películas), en este caso la primera. Para la petición HTTP se utiliza *client*, una instancia del *HttpClientInterface* que proporciona *Symfony* para realizar peticiones HTTP.

Listing 5.7: EventController.php

```

1 public function getTMDBFilmsUpcoming(): ?array
2 {
3     $result = NULL;
4
5     $response = $this->client->request(
6         'GET',
7         'https://api.themoviedb.org/3/movie/upcoming?api_key=' . static::API_KEY
8         . '&language=es-ES&page=1'
9     );
10
11    if ($response->getStatusCode() === static::SUCCESS_STATUS_CODE):
12        $result = $response->toArray();
13    endif;
14
15    return $result;
16 }

```

Tras obtener el listado, recorremos el vector y normalizamos sus datos para pasarlo a la vista para el usuario. En esta vista se verán estos detalles en pequeñas “cartas” de presentación si se quieren visualizar sus datos completamente hay un botón disponible para ello.

Listing 5.8: EventController.php

```

1 foreach ($upcomingsFilms as $filmTMDB):
2     if ($filmTMDB !== NULL):
3         $overview = EventData::getShortenSummary($filmTMDB['overview']);
4
5         $data[] = array(
6             'tmdb_id' => $filmTMDB['id'],
7             'title' => strtoupper($filmTMDB['title']),
8             'release_date' => $filmTMDB['release_date'],
9             'summary' => $overview,
10            'poster_photo' => $this->getImageBaseURLIMDB() . 'w154/' . $filmTMDB[
11                'poster_path'],
12                'mark' => $filmTMDB['vote_average']
13            );
14        endif;
15    endforeach;

```

#### 5.4.5. Paginación de resultados

Para la paginación de resultados se ha utilizado el *bundle* de *Knp Paginator* [32]. Un *bundle* es pack de archivos propios o de terceros que implementan una funcionalidad dentro de un sistema Symfony, este *bundle* ofrece una forma de paginar resultados muy sencilla, como muestra el fragmento 5.9, recibe tres parámetros: el primero de ellos es la *query* de la consulta que se quiere hacer, el segundo es el número de página por el que se quiere empezar y el último el número de *items* por página.

Listing 5.9: Sintaxis para usar *Knp Paginator*

```

1 $searchResults = $paginator->paginate(
2     $searchQuery,
3     $request->query->getInt('page', 1),
4     5
5 );

```

Una vez que se han paginado los resultados en el *back-end*, en el *front-end* hay que incluir el fragmento de código de abajo para añadir la interfaz de paginado y que el usuario puede cambiar de página fácilmente. El parámetro *results* serían los resultados obtenidos anteriormente y lo demás para darle un estilo con CSS.

```

{{ knp_pagination_render(results, null, {}, {
    'align': 'center',
    'size': 'large',
    'rounded': true,
}) }} 
```

# Capítulo 6

## Pruebas

En este capítulo se realizarán pruebas de software para comprobar que el funcionamiento de todo el código, evaluar su calidad y encontrar defectos o errores. Se utilizarán dos métodos que son los *tests* de caja blanca (*white box*) y caja negra (*black box*). A continuación se explicarán ambos métodos y sus diferencias.

### 6.1. Caja blanca

Este método consiste en analizar el código y la estructura del producto que se va a probar y usar ese conocimiento para la realización de las pruebas. Se usa en la fase de *Unit testing*, que es un tipo de prueba de software en el que se prueban unidades o componentes individuales de un software. También puede ocurrir en otras fases como en las pruebas de sistema o de integración. Este método requiere tener amplio conocimiento sobre la tecnología y arquitectura del proyecto. [33]

### 6.2. Caja negra

Este método consiste en estudiar las entradas y salidas que produce la aplicación sin tener en cuenta el funcionamiento interno. Dichas pruebas se realizan desde la interfaz gráfica. Estas pruebas pueden ser funcionales o no funcionales, aunque generalmente son funcionales

### 6.3. Pruebas en *Symfony*

Para realizar las pruebas de caja blanca y caja negra se pueden utilizar una gran variedad de *frameworks* que incluyen herramientas muy útiles para facilitar el desarrollo de los *tests*. Para este proyecto se ha escogido PHPUnit [34] que viene incluido con *Symfony*, este *framework* se utiliza para realizar pruebas unitarias en PHP. PHPUnit permite realizar muchas *assertions* simples y flexibles que permiten probar fácilmente el código, lo que funciona muy bien cuando está probando componentes específicos. Una vez hemos instalado la biblioteca, para crear un *test* tiene una pequeña guía integrada para ayudarte, indicarte y crearte el archivo como se muestra en la figura 6.1

```
D:\Usuario\Desktop\Informatica\QUINTO FACTORIAL\SEGUNDO CUATRI\TFG\CineLify>symfony console make:test

Which test type would you like?:
[TestCase] basic PHPUnit tests
[KernelTestCase] basic tests that have access to Symfony services
[WebTestCase] to run browser-like scenarios, but that don't execute JavaScript code
[ApiTestCase] to run API-oriented scenarios
[PantherTestCase] to run e2e scenarios, using a real-browser or HTTP client and a real web server
> WebTestCase

Choose a class name for your test, like:
* UtilTest (to create tests/UtilTest.php)
* Service\UtilTest (to create tests/Service/UtilTest.php)
* \App\Tests\Service\UtilTest (to create tests/Service/UtilTest.php)

The name of the test class (e.g. BlogPostTest):
> EventControllerTest

created: tests/EventControllerTest.php

Success!
```

Next: Open your new test class and start customizing it.  
Find the documentation at <https://symfony.com/doc/current/testing.html#functional-tests>

Figura 6.1: Guía para crear un *test* en *Symfony*

Se dividen en cinco tipos de *tests*:

- ***TestCase***: son para pruebas básicas como funciones o métodos sencillos.
- ***KernelTestCase***: son para pruebas básicas que tienen acceso a servicios de *Symfony*
- ***WebTestCase***: agrega una lógica especial además de *KernelTestCase*, se ejecutan en el contexto de una petición HTTP y corren en escenarios similares a los de un navegador.
- ***ApiTestCase***: para ejecutar escenarios orientados a API
- ***PantherTestCase***: *Panther* permite ejecutar exactamente el mismo escenario *WebTestCase* pero en navegadores reales. Es una

Se utilizarán principalmente *WebTestCase* y *KernelTestCase*, ya que al ser una aplicación web el contexto son peticiones HTTP y pruebas para servicios de *Symfony*

Para el primer *test* como muestra el código, cargamos la página principal y comprobamos que carga correctamente comprobando el estado de la petición HTTP y que contiene el título de la página.

Listing 6.1: MainControllerTest.php

```
1 use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;
2
```

```

3 class MainControllerTest extends WebTestCase
4 {
5     public function testHomePage(): void
6     {
7         $client = static::createClient();
8         $crawler = $client->request('GET', 'http://127.0.0.1:8000/');
9
10        static::assertResponseIsSuccessful();
11        static::assertSelectorTextContains('h1', 'Cinefilly');
12
13    }
14 }
```

Comprobaremos que los comentarios en los detalles de una película y evento se cargan correctamente comprobando el número de comentarios con su clase asociada, que es “media” como muestra el código. Cargamos la URL de una película que sepamos que tiene un comentario y se comprueba su tamaño.

Listing 6.2: EventControllerTest.php

```

1     public function testEventDetails(): void
2     {
3         $client = static::createClient();
4         $crawler = $client->request('GET', 'http://127.0.0.1:8000/event/details?
5 id=15');
6
7         static::assertResponseIsSuccessful();
8         static::assertCount(1, $crawler->filter('.media'));
9     }
```

Como vemos en la figura 6.2, ambos *tests* se ejecutan correctamente. Sin embargo si cambiamos el tamaño de comentarios del código anterior por 2 en vez de 1, vemos en la figura 6.3 como falla uno de los *test* con el mensaje “Failed asserting that actual size 1 matches expected size 2”, es decir, “no se pudo afirmar que el tamaño real 1 coincide con el tamaño 2 esperado”.

```

D:\Usuario\Desktop\Informatica\QUINTO FACTORIAL\SEGUNDO CUATRI\TFG\Cinefilly>php ./vendor/bin/phpunit
PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

Warning: Your XML configuration validates against a deprecated schema.
Suggestion: Migrate your XML configuration using "--migrate-configuration"!

Testing
..
2 / 2 (100%)

Time: 00:00.665, Memory: 32.00 MB

OK (2 tests, 5 assertions)
```

Figura 6.2: Ejecución correcta de ambos *tests*

```
D:\Usuario\Desktop\Informatica\QUINTO FACTORIAL\SEGUNDO CUATRI\TFG\Cinelify>php ./vendor/bin/phpunit
PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

Warning: Your XML configuration validates against a deprecated schema.
Suggestion: Migrate your XML configuration using "--migrate-configuration"!

Testing
.
2 / 2 (100%)

Time: 00:00.719, Memory: 32.00 MB

There was 1 failure:

1) App\Tests\EventControllerTest::testEventDetails
Failed asserting that actual size 1 matches expected size 2.

D:\Usuario\Desktop\Informatica\QUINTO FACTORIAL\SEGUNDO CUATRI\TFG\Cinelify\tests\EventControllerTest.php:15

FAILURES!
Tests: 2, Assertions: 5, Failures: 1.
```

Figura 6.3: Ejecución fallida de uno de los *tests*

Para comprobar la carga de los asientos a la hora de seleccionar y verificar que muestran los asientos ocupados utilizaremos la misma técnica que anteriormente, cada asiento reservado tiene una clase asociada que es “redBox”. El código 6.3 muestra que se utilizará una sesión en la que haya dos asientos reservados y se comprobará que hay dos clases “redBox” asociadas a los asientos y una en la leyenda, en total 3.

Listing 6.3: BookingControllerTest.php

```
1  public function testSessionBooking(): void
2  {
3      $client = static::createClient();
4      $crawler = $client->request('GET', 'http://127.0.0.1:8000/booking?session
=14');
5
6      static::assertResponseIsSuccessful();
7      static::assertCount(3, $crawler->filter('.redBox'));
8
9  }
```

También se va a hacer una prueba para el *login*, es decir la identificación del usuario en el sistema. Como muestra el código 6.4 se buscará un usuario en la base de datos, se comprobará que accede correctamente a su cuenta y se hará una petición a la zona de “Mi perfil” para verificar que su nombre aparece, de lo contrario saltaría un error en la prueba.

Listing 6.4: UserControllerTest.php

```
1  public function testLogin(): void
2  {
3      $client = static::createClient();
4      $entityManager = $client->getContainer()->get('doctrine')->getManager();
5      $user=$entityManager
```

```
6      ->getRepository(User::class)
7      ->findOneBy(array('email'=>'sergiocantero8@gmail.com'));
8
9      static::assertSame('sergiocantero8@gmail.com', $user->getEmail());
10
11
12     $client->loginUser($user);
13     $client->request('GET', 'http://127.0.0.1:8000/user/profile');
14     static::assertResponseIsSuccessful();
15     static::assertSelectorTextContains('h4', 'Sergio');
16
17 }
```

El código 6.5 muestra el *test* para un formulario, en concreto, el de añadir un comentario a una película o evento. Para ello, debemos identificarnos ya que no se puede comentar sin estar identificado, una vez nos hemos identificado, se busca el primer formulario que tenga la palabra “Enviar” y se añade un comentario de prueba.

Listing 6.5: EventControllerTest.php

```
1   public function testAddCommentToEvent(): void
2   {
3       $client = static::createClient();
4       $entityManager = $client->getContainer()->get('doctrine')->getManager();
5       $user=$entityManager
6           ->getRepository(User::class)
7           ->findOneBy(array('email'=>'sergiocantero8@gmail.com'));
8
9       static::assertSame('sergiocantero8@gmail.com', $user->getEmail());
10
11
12      $client->loginUser($user);
13      $client->request('GET', 'http://127.0.0.1:8000/event/details?id=10');
14
15
16      $crawler=$client->submitForm('Enviar', [
17          'form[comment]' => 'Comentario para test',
18      ]);
19
20      static::assertResponseIsSuccessful();
21
22 }
```

El código 6.5 muestra el *test* para el formulario de añadir una película o evento. Como este formulario contiene muchos campos para llenar, sólo se llenarán los que no puedan ser nulos, los demás se dejarán en blanco ya que al ser un *test* sólo es necesario comprobar que se añade correctamente. Debemos identificarnos como un usuario administrador ya que este proceso necesita privilegios de administrador. Una vez que nos hemos identificado, obtenemos el formulario por su etiqueta del botón *submit* y llenamos los campos requeridos. Para los selectores se utiliza la sentencia *select* y para los botones de tipo *checkbox* se utiliza la sentencia *tick*.

Listing 6.6: EventControllerTest.php

```
1  public function testAddEvent(): void
2  {
3      $client = static::createClient();
4      $entityManager = $client->getContainer()->get('doctrine')->getManager();
5      $user=$entityManager
6          ->getRepository(User::class)
7          ->findOneBy(array('email'=>'sergiocantero8@gmail.com'));
8
9      static::assertSame('sergiocantero8@gmail.com', $user->getEmail());
10
11
12     $client->loginUser($user);
13     $crawler=$client->request('GET', 'http://127.0.0.1:8000/admin/event/add')
14 ;
15
16     $buttonCrawlerNode = $crawler->selectButton('Añadir');
17
18     $form = $buttonCrawlerNode->form();
19
20     $form['add_event[title]'] = 'Titulo película';
21     $form['add_event[type]']->select('película');
22     $form['add_event[description]'] ='Sinopsis';
23     $form['add_event[status]']->tick();
24
25     $client->submit($form);
26
27     static::assertResponseIsSuccessful();
28 }
```

# Capítulo 7

## Conclusiones

### 7.1. Cronología real de desarrollo

En esta sección se analizará la cronología real de desarrollo que se ha seguido a lo largo de la realización de todo el proyecto. Como se analizó al principio del proyecto, en la sección 2.2 del capítulo 2, se estimó una planificación para todas las fases del proyecto.

En la figura 7.1 podemos observar cuál ha sido la cronología real, comparativamente con la inicial, se ha tardado más en el inicio del proyecto ya que se produjeron varios conflictos sobre qué *frameworks* y *herramientas* escoger. En la parte de análisis se ha invertido el tiempo estimado, igual que con la parte de diseño. Sin embargo la fase de aprendizaje ha sido menor de lo esperado ya que se tenía una experiencia previa con *Symfony* y *Twig*, al contrario con la API de TMDB que no se tenían conocimientos. La fase de implementación ha tenido la planificación esperada pero repartida de forma distinta ya que la parte del *front-end* se ha invertido más tiempo al no tener un *framework* específico para ello y tener que diseñar todo con HTML, CSS y Bootstrap. Para las fases de redacción de memoria y correcciones se han ajustado perfectamente a lo planificado inicialmente.

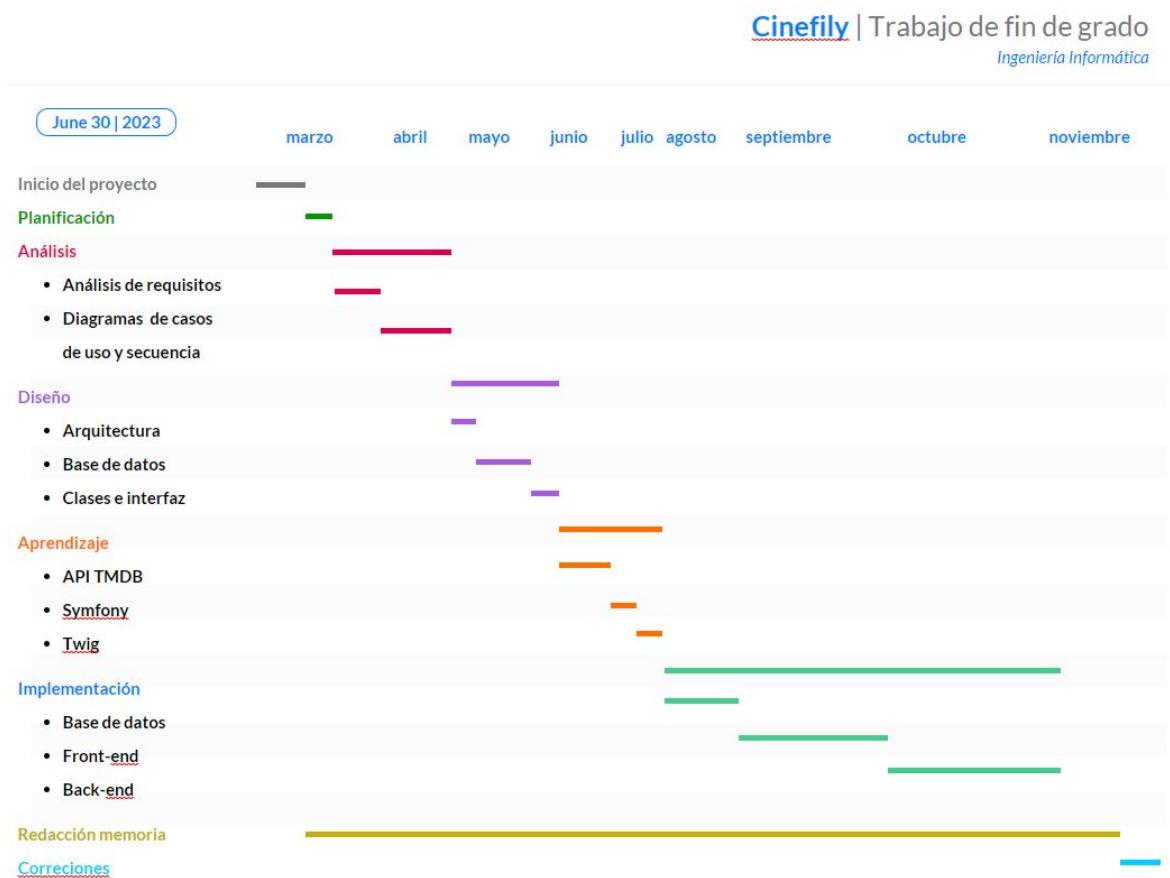


Figura 7.1: Cronología real del proyecto

En general, unas fases se ha tardado más en realizarlas pero se han compensado con otras que se han tardado menos por lo que se ha ajustado muy bien la planificación inicial con la real.

## 7.2. Objetivos cumplidos

En líneas generales todos los objetivos propuestos inicialmente se han completado correctamente. El objetivo general se ha cumplido ya que se ha dado solución a un problema propuesto en el inicio del proyecto. Los objetivos específicos que surgían del objetivo general se han conseguido realizar correctamente explicando todo su proceso a lo largo del proyecto. A nivel técnico las nuevas tecnologías propuestas para su aprendizaje y su posterior ejecución se han aplicado adecuadamente. A nivel personal, satisfecho por el proyecto que he realizado y de superar ciertos defectos a la hora de desarrollar un proyecto completo con todas sus fases y su documentación para el Trabajo de Fin de Grado de Ingeniería Informática.

### 7.3. Trabajos futuros

Cualquier aplicación que se desarrolla y se establecen unos objetivos, tiene un margen de mejora que se trabaja con el tiempo. Este proyecto ha cubierto todos los objetivos que se propusieron inicialmente pero hay varias mejoras y funcionalidad que podrían añadirse a la aplicación web, como pueden ser:

- Carga por AJAX de todos los elementos de la página web para reducir su tiempo de carga y que pueda modificarse el contenido sin tener que volver a cargarse.
- Búsqueda de películas y eventos con autocompletado, actualmente se pueden buscar pero sin el autocompletado que daría mucha facilidad y rapidez al usuario para escoger la película que desea.
- Descarga de las entradas en formato PDF desde la propia aplicación web
- Seguimiento y envío de mensajes entre usuarios de la plataforma
- Búsqueda de películas que no están en la plataforma pero sí en TMDB a través de su API
- Añadir sistema de recomendación para mostrar películas al usuario que estén relacionadas con sus gustos o sus visitas recientes
- Añadir *widgets* para saber cuáles son las películas más populares, peor valoradas, más comentadas...

# Capítulo 8

## Anexos

### 8.1. Manual de usuario

En este manual de usuario se explicará de forma visual como acceder a todas las opciones de la plataforma y realizar sus respectivos procesos.

Al entrar en la web, aparece como primera página el *home*, donde aparecerán las películas divididas por géneros. La información sobre las películas está contenida en una “carta”, con un botón para mostrar más información y otro para el *trailer*. (Figura 8.1)

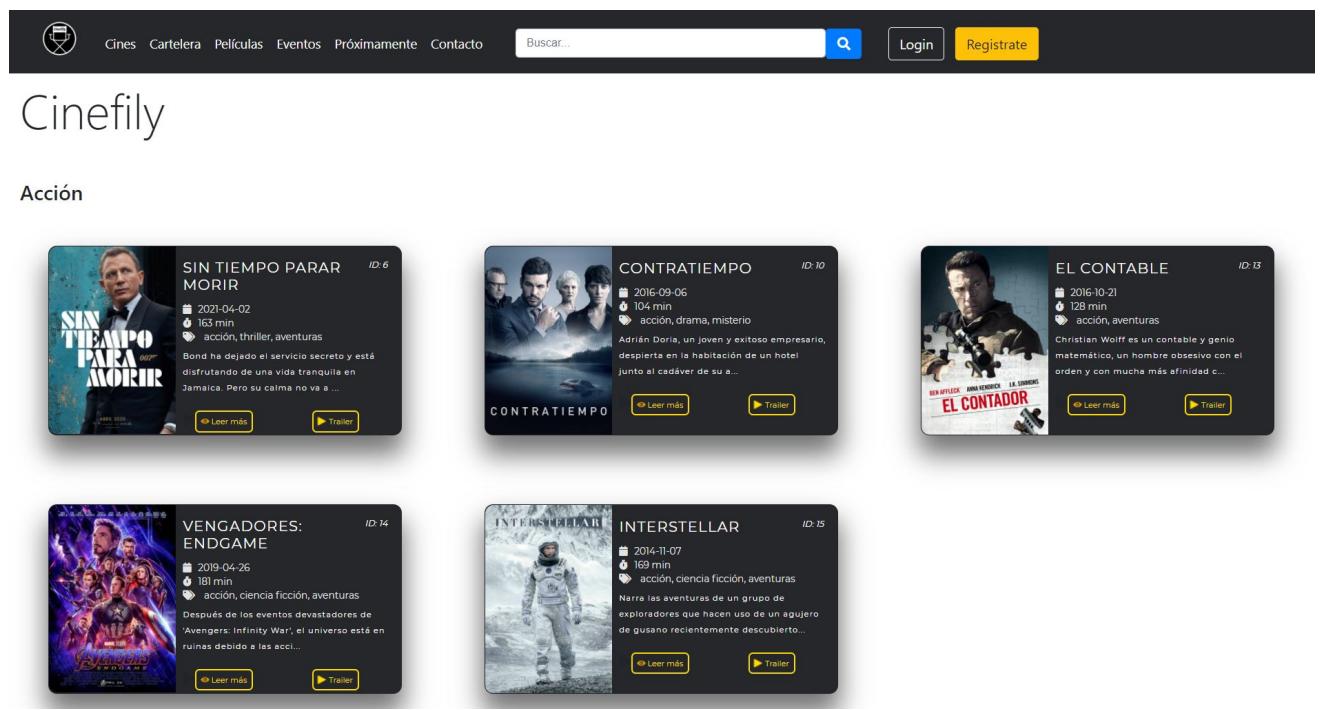


Figura 8.1: Página de inicio

Los menús de la plataforma se pueden encontrar en la parte superior dependiendo de si el usuario está identificado o si tiene privilegios de administrador tendrá un menú diferente como se observa en las tres figuras 5.4

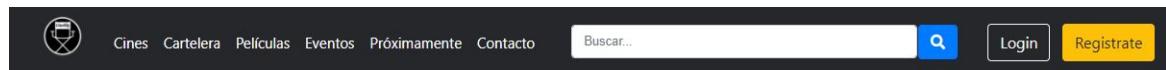


Figura 8.2: *Header* sin identificar

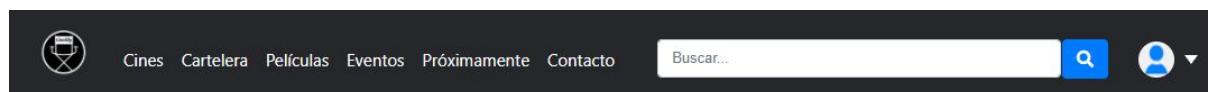


Figura 8.3: *Header* identificado como usuario

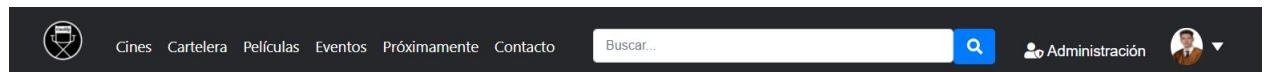


Figura 8.4: *Header* identificado como admin

Si el usuario está identificado y tiene privilegios de administrador tendrá un nuevo menú desplegable llamado “Administración” como aparece en la figura 8.5 donde se pueden ver todas las opciones: añadir un evento, añadir una sesión, añadir un cine, obtener un listado de todos los usuarios y obtener el *log* del sistema.

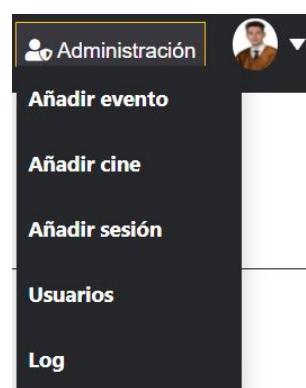
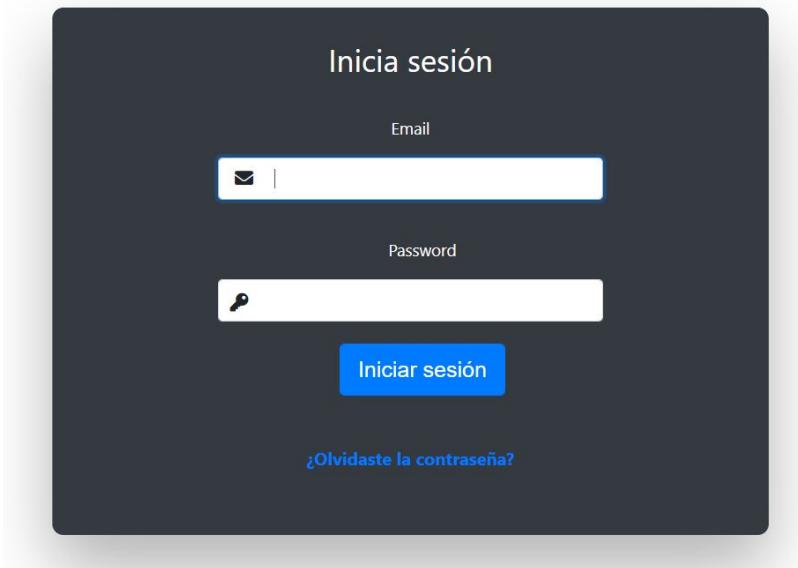


Figura 8.5: Opciones de administrador

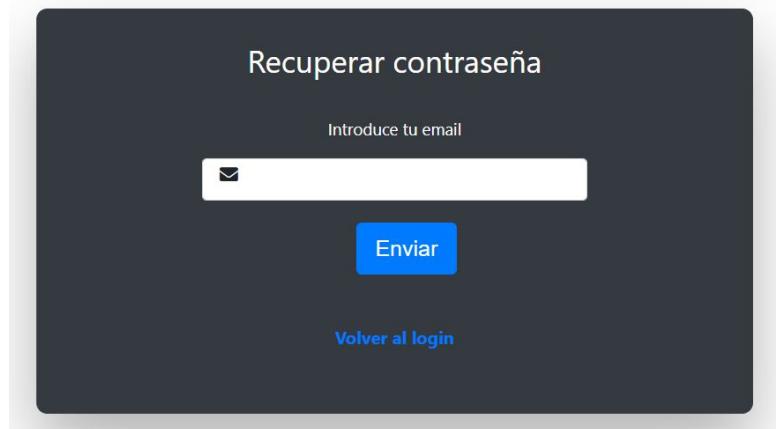
Para identificarse en la plataforma, se debe hacer *click* en el botón de *Login* y aparecerá un formulario como se muestra en la figura 8.6 para llenar con su email y contraseña.



The image shows a dark-themed login form titled "Inicia sesión". It features two input fields: "Email" with an envelope icon and "Password" with a keyhole icon. Below the fields is a blue "Iniciar sesión" button. At the bottom, there is a link in blue text: "¿Olvidaste la contraseña?".

Figura 8.6: Formulario para identificarse

Si se ha olvidado la contraseña, accediendo a la opción “¿Olvidaste la contraseña?” se mostrará un formulario como muestra la figura 8.7 para recuperar la contraseña, introduciendo su email se le enviará un correo con la nueva contraseña generada. Se puede volver atrás pulsando “Volver al login”.



The image shows a dark-themed form titled "Recuperar contraseña". It has a single input field labeled "Introduce tu email" with an envelope icon. Below it is a blue "Enviar" button. At the bottom, there is a link in blue text: "Volver al login".

Figura 8.7: Formulario para recuperar la contraseña

Para ver la información sobre una película se deberá hacer *click* en el botón “Leer más” de la “carta” de la película. En las figuras 8.8 y 8.9 se muestran los detalles de una película o evento con toda su información y el *trailer* (si tiene) respectivamente.

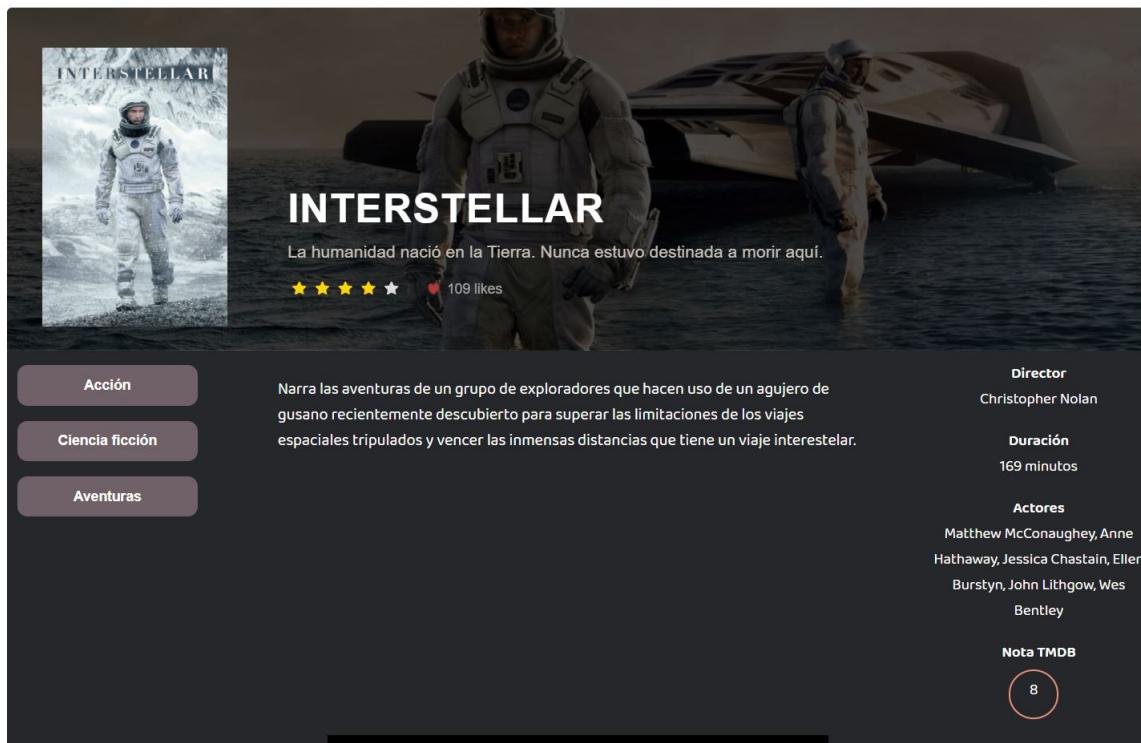


Figura 8.8: Detalles de la película o evento

Para reproducir el *trailer* hay que hacer *click* sobre él y se reproducirá dentro de la propia página sin abrir otra ventana



Figura 8.9: *Trailer* de película

La sección de comentarios se encuentra justo abajo de los detalles comentados anteriormente, si no se está identificado en la plataforma, no se podrá comentar como muestra la figura 8.10 pero si visualizar los comentarios.

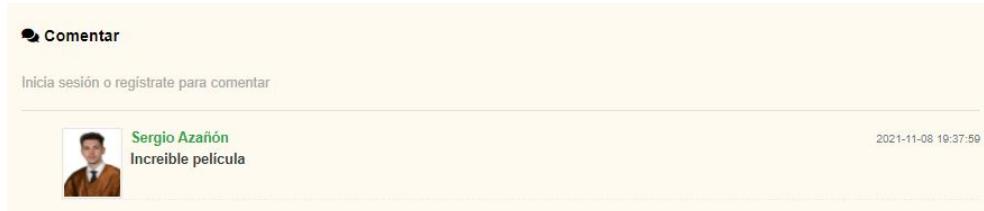


Figura 8.10: Sección de comentarios (no identificado)

Sin embargo si se está identificado, se podrá comentar y además visualizar los comentarios. Todos los comentarios que sean del usuario identificado podrán borrarlo pulsando sobre el ícono de la papelera de la esquina superior derecha.

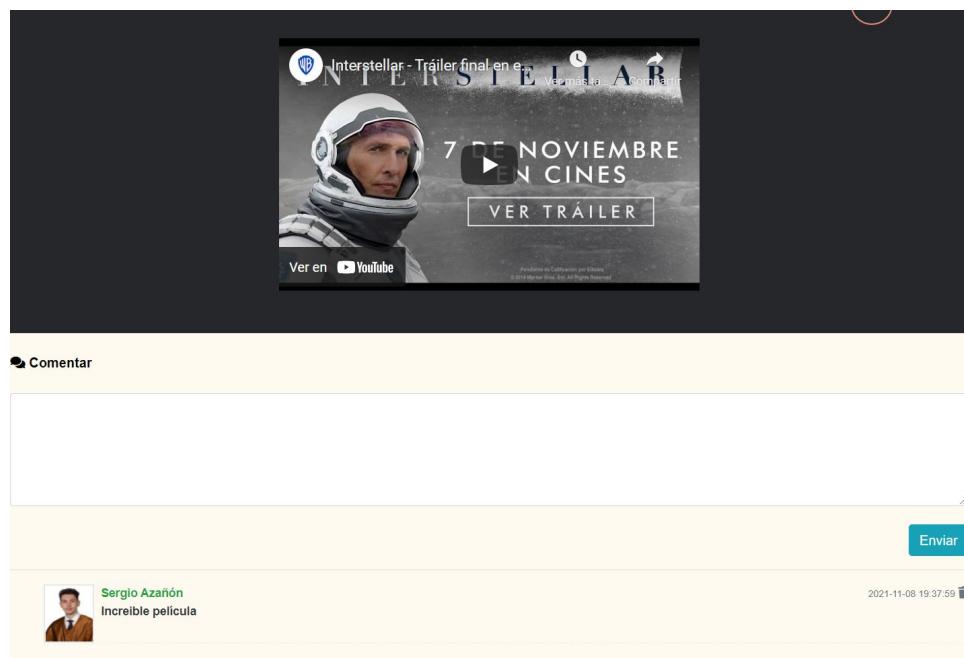


Figura 8.11: Trailer y sección de comentarios

Para visualizar la cartelera, se debe hacer *click* en el botón de “Cartelera” del *header* superior. En la figura 8.12 se observa la cartelera con las películas para una fecha y un cine concretos, en la parte superior está el formulario donde se elige dichos parámetros.

Cada película o evento que tenga sesiones para la fecha indicada aparecerán su título, su sinopsis, un botón para ver más detalles de la película, un botón para ver el *trailer* de la película en otra ventana y los horarios de las sesiones con un botón para entrar a reservar entradas para dicha sesión.

#### Cartelera

Cine

Cine Kinefly

Fecha

Nov 17 2021

Buscar

---

Evento	Horario
	Sin tiempo para morir Bond ha dejado el servicio secreto y está disfrutando de una vida tranquila en Jamaica. Pero su calma no va a durar mucho tiempo. Su amigo de la CIA, Félix Leiter, aparece para pedirle ayuda. La misión de rescatar a un científico secuestrado resulta ser mucho más arriesgada de lo esperado, y lleva a Bond tras la pista de un misterioso villano armado con una nueva y peligrosa tecnología. <a href="#">Ver detalles</a> <a href="#">Trailer</a>
	Vengadores: Endgame Después de los eventos devastadores de 'Avengers: Infinity War', el universo está en ruinas debido a las acciones de Thanos, el Titán Loco. Con la ayuda de los aliados que quedaron, los Vengadores deberán reunirse una vez más para intentar deshacer sus acciones y restaurar el orden en el universo de una vez por todas, sin importar cuáles son las consecuencias... Cuarta y última entrega de la saga "Vengadores". <a href="#">Ver detalles</a> <a href="#">Trailer</a>

Figura 8.12: Cartelera

Si se ha hecho *click* en uno de los botones con el horario para una sesión, aparecerá una vista como la de la figura 8.13 que muestra la pantalla de selección de asientos para una sesión con los detalles de la misma a la izquierda, en el centro los *inputs* para el email donde se enviarán las entradas y el número de asientos y en la parte inferior la zona interactiva donde el usuario elegirá sus asientos.

Los asientos grises son todos los que están vacíos, los rojos los que están ya reservados y los verdes los que se han seleccionado. Una vez se hayan seleccionado, hay que pulsar el botón de confirmar selección para procesar la reserva.

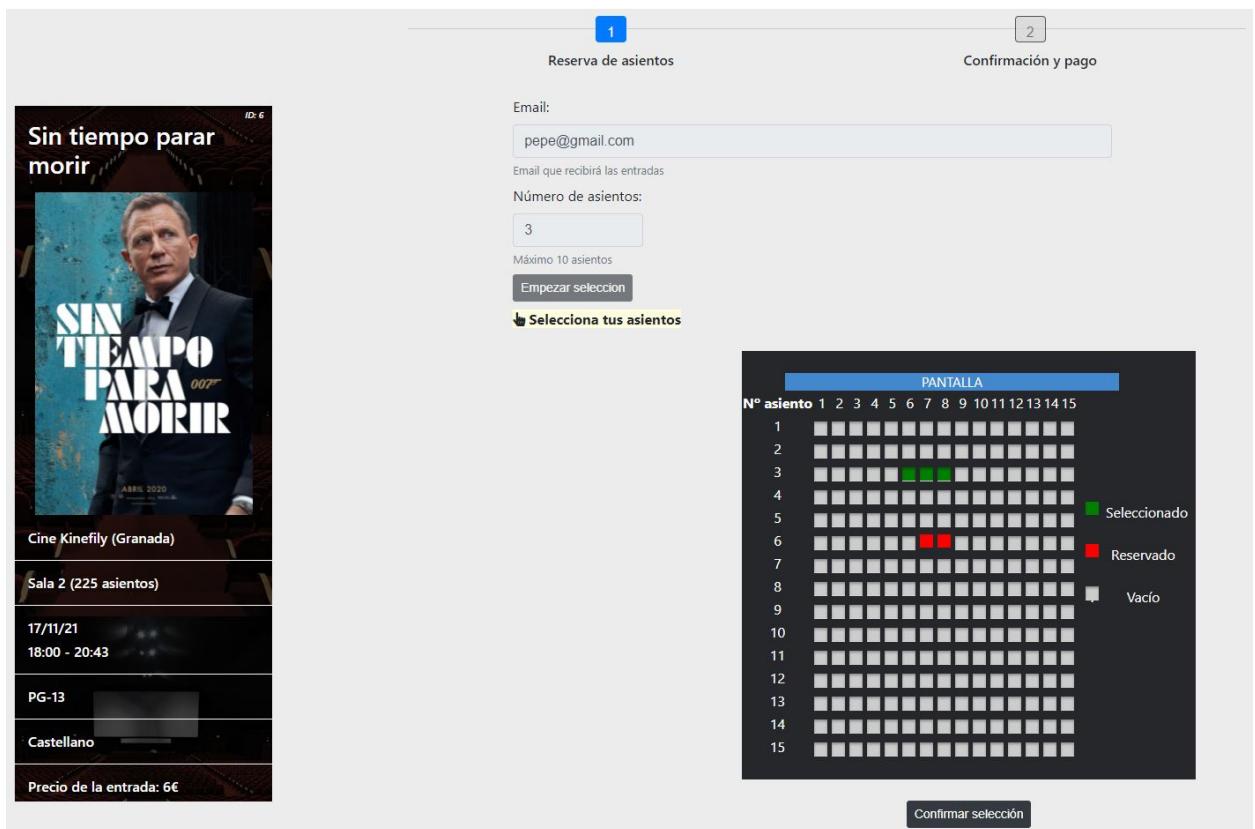


Figura 8.13: Pantalla de selección de asientos

Tras la elección de asientos, el segundo paso es la parte de “Confirmación y pago” como muestra la figura 8.14, esta vista es para confirmar la reserva, ver la información detallada de la reserva, mostrar la entrada generada, aplicar un cupón de descuento si lo tiene y pagar la reserva.

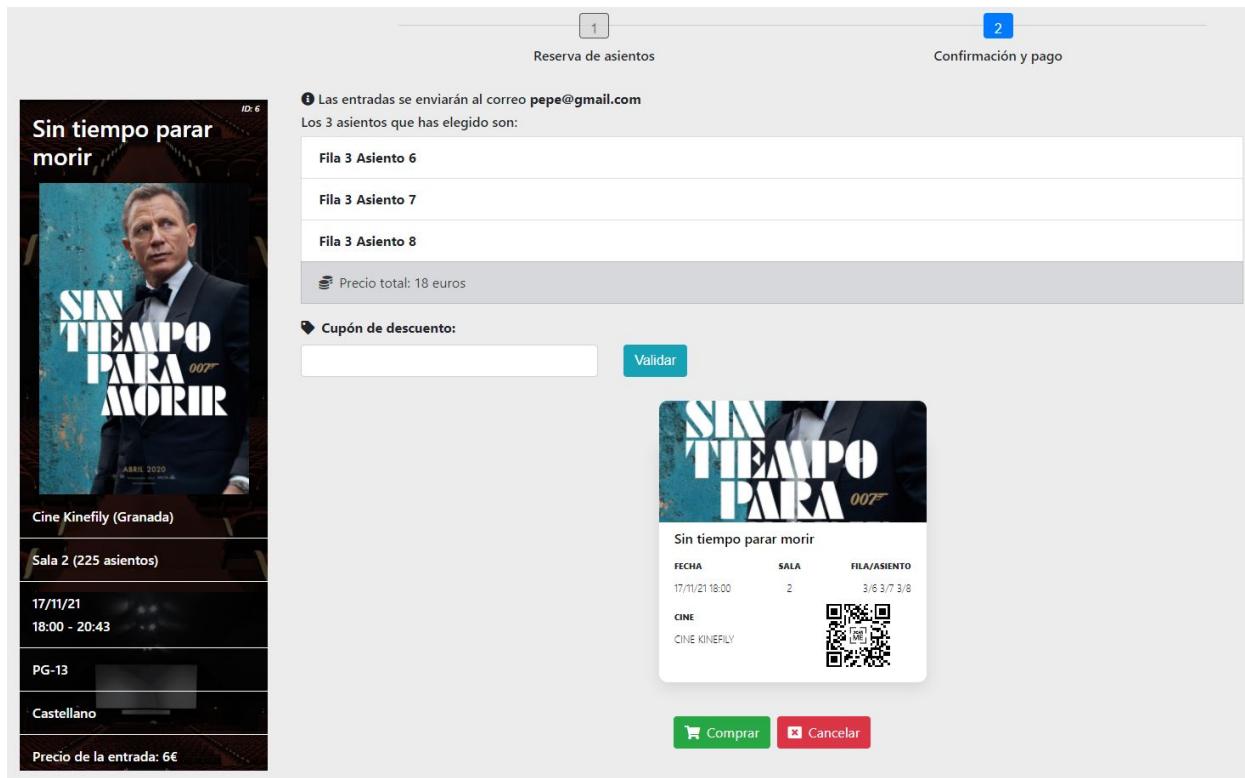


Figura 8.14: Pantalla de confirmación de reserva

La figura 8.15 muestra la vista para cuándo se aplica un cupón de descuento correctamente, tachando el precio anterior y añadiendo el nuevo.

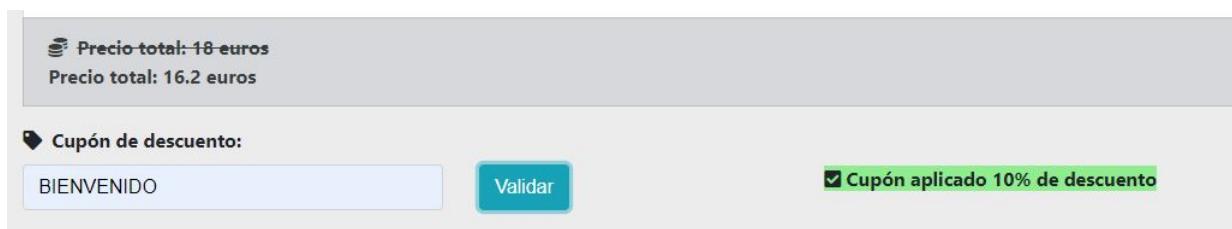


Figura 8.15: Aplicación de cupón de descuento

Si se pulsa el botón de “Comprar” se mostrarán las opciones de pago en una ventana *modal* como muestra la figura 8.16, aparece el precio total a pagar y las diferentes opciones para el pago: *Paypal*, *Sofort*, tarjeta de crédito o el pago en taquilla.

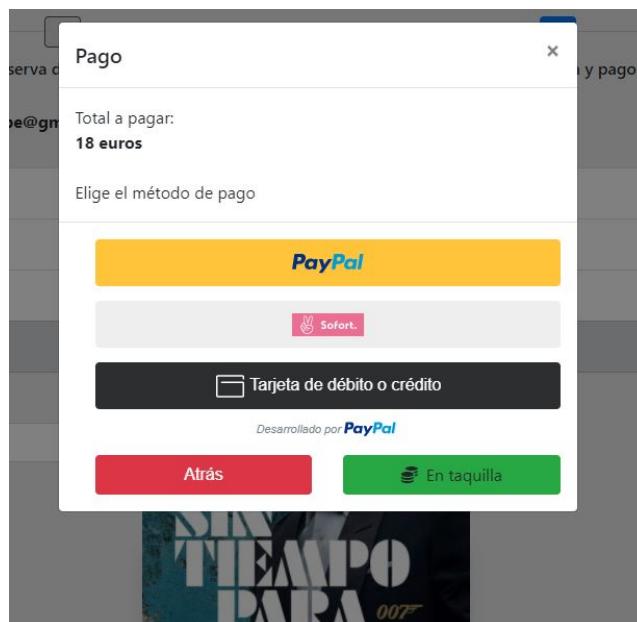


Figura 8.16: Ventana *modal* de todas las opciones de pago

Si se quiere saber qué cines hay disponible en la plataforma e información sobre ellos hay que pulsar el botón del *header* superior “Cines”. En la figura 8.17 se puede ver el listado de todos los cines que hay en la plataforma con información sobre ellos. La columna de acciones sólo la puede ver el usuario con privilegios de administrador.

Nombre	Localización	Número de salas	Asientos totales	Precio de las entradas	Número de sesiones activas	Acciones
Cine Kinefilly	Granada	5	1125	6€	3	
Cine Yelmo	Madrid	2	200	5€	0	

Figura 8.17: Listado de todos los cines

Para ver todas las películas que hay en la plataforma se debe pulsar el botón de “Películas” del *header* superior y aparecerá un listado como muestra la figura 8.18. El listado incluye una paginación de diez películas por página con información sobre ellas y las sesiones activas. Además hay dos botones para usuarios con privilegios de administrador, uno para editar la película y otro para eliminarla.

Todas las películas

Evento	Descripción	Sesiones activas	Acciones
 Ghostland	Una madre y sus dos hijas heredan una casa. Pero en su primera noche, aparecen unos asesinos y la madre se ve obligada a luchar para salvar a sus hijas.	No hay sesiones programadas	  
 Sin tiempo parar morir	Bond ha dejado el servicio secreto y está disfrutando de una vida tranquila en Jamaica. Pero su calma no va a durar mucho tiempo. Su amigo de la CIA, Felix Leiter, aparece para pedirle ayuda. La misión de rescatar a un científico secuestrado resulta ser mucho más arriesgada de lo esperado, y lleva a Bond tras la pista de un misterioso villano armado con una nueva y peligrosa tecnología.	Cine Kinefly 20:00 Cine Kinefly 18:00	  

Figura 8.18: Todas las películas

Lo mismo pasa con los eventos pero pulsando el botón de “Eventos”. La figura 8.19 muestra un listado de todos los eventos de la plataforma con información sobre ellos y las sesiones activas. Además hay dos botones para usuarios con privilegios de administrador, uno para editar el evento y otro para eliminarlo.

Todos los eventos

Evento	Descripción	Sesiones activas	Acciones
 Monólogo Christian García	“En verdad... soy buena gente” es un espectáculo de monólogos acompañado de videos inéditos, anécdotas reales, algo de música, y mucha tontería. Un show lleno de malafollá y de ese humor canalla y salvaje que dan a Christian García sus demonios granadinos. De igual manera, es una oportunidad para conocer al artista un poco más como persona y darse cuenta, de que a pesar de los demonios... en verdad, ES BUENA GENTE.	No hay sesiones programadas	  

Figura 8.19: Todos los eventos

Para ver todas las películas que se estrenarán próximamente se debe hacer *click* en el botón de “Próximamente” en el *header* de la parte superior. La figura 8.21 muestra todas las películas que se estrenarán próximamente junto con toda su información y un botón para ver más detalles, se usa la API de TMDB.

#### ► Próximamente

Estas son las próximas películas que serán estrenadas en los cines

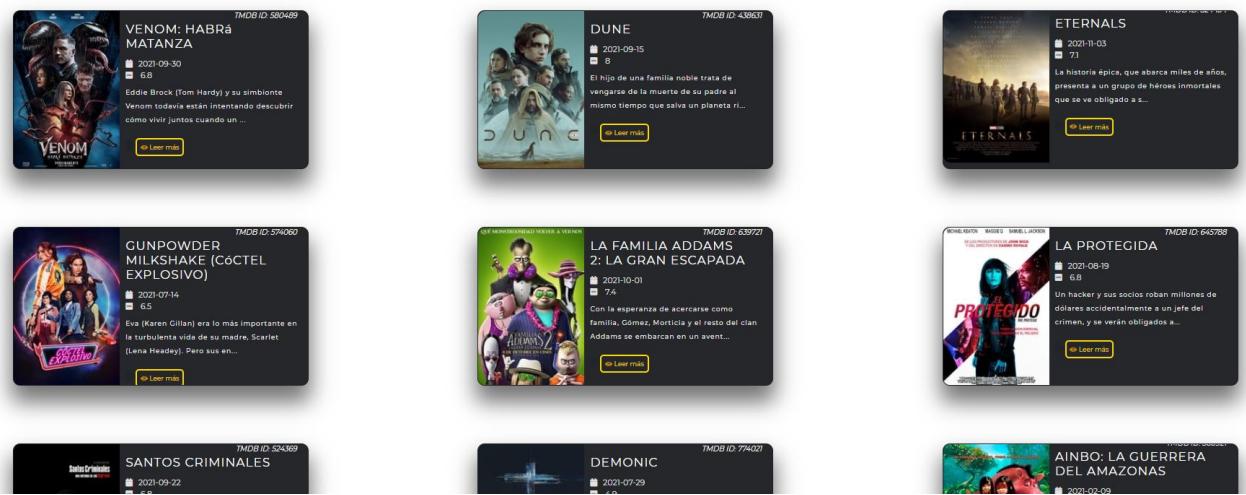


Figura 8.20: Películas que se estrenarán próximamente

Para enviar la opinión o querer alquilar una sala de cine hay que pulsar en el botón de “Contacto” del *header* y como muestra la figura 8.21, aparece el formulario de contacto para enviar un mensaje eligiendo el motivo con un selector estando disponibles: “Opinión”, “Solicitud de sala para evento” y “Mejoras”.

Figura 8.21: Formulario de contacto

En la figura 8.22 muestran las opciones que tiene un usuario cuando hace *click* sobre su avatar en el *header* fijo, tiene las opciones de acceder a su perfil, visualizar las entradas que ha comprado o los cupones que tiene disponibles.

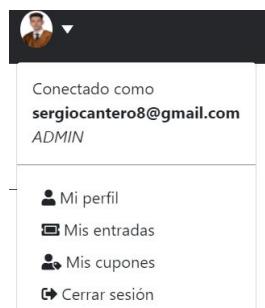


Figura 8.22: Opciones del perfil

En la figura 8.23 muestra la vista que tiene un usuario al acceder a su perfil, en la izquierda tiene sus datos además de estadísticas como el número de comentarios que ha realizado, el número de entradas que ha comprado o el número de cupones que tiene disponibles. También puede acceder a las entradas que tiene compradas y sus cupones disponibles. A la derecha tiene un formulario con todos sus datos donde puede modificarlos si lo desea

A screenshot of a user profile edit form. On the left, there is a sidebar with a user's profile picture, name ('Sergio'), member since date ('18/06/21'), location ('Granada'), and three buttons: 'Mis entradas', 'Mis cupones', 'Comentarios realizados' (1), 'Cupones' (2), and 'Entradas compradas' (9). The main area contains a form with fields for 'Nombre' (Sergio), 'Apellidos' (Azañón), 'Email' (sergiocantero8@gmail.com), 'Teléfono (Opcional)' (12345678), 'Ciudad (Opcional)' (Granada), a 'Cambiar foto de perfil' section with a 'Browse' button, a 'Cambiar contraseña' section with a password input field, and a 'Privilegios' section showing 'ADMIN'. A 'Guardar' button is at the bottom.

Figura 8.23: Vista del perfil de un usuario

En la figura 8.24 muestra la vista que tiene un usuario al acceder al menú “Mis cupones” que visualiza todos los cupones que tiene disponibles junto con sus detalles.

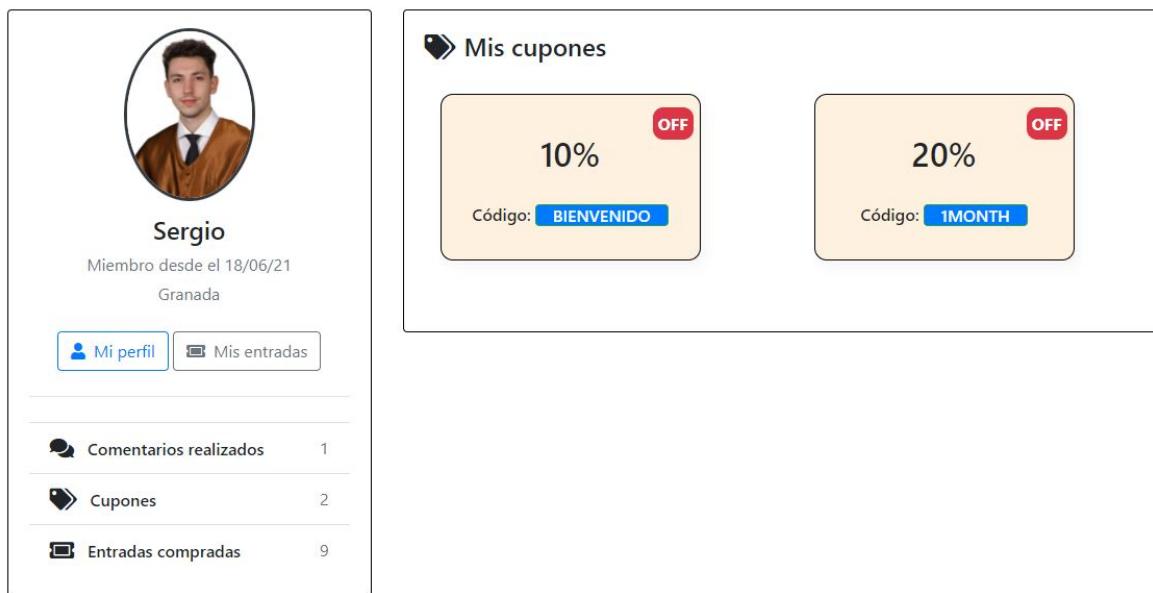


Figura 8.24: Vista de todos los cupones de un usuario

En la figura 8.25 muestra la vista que tiene un usuario al acceder al menú “Mis entradas” que visualiza todos los tickets que ha comprado el usuario con los detalles de cada ticket respectivamente.

Sergio  
Miembro desde el 18/06/21  
Granada

[Mi perfil](#)

[Mis cupones](#)

[Comentarios realizados](#) 1

[Cupones](#) 2

[Entradas compradas](#) 1

**Mis entradas**

31/10/21 14:12

Sin tiempo parar morir

FECHA	SALA	FILA/ASIENTO
17/11/21 20:00	1	6 / 10

CINE CINEKINELLY

31/10/21 14:12

Sin tiempo parar morir

FECHA	SALA	FILA/ASIENTO
17/11/21 20:00	1	6 / 11

CINE CINEKINELLY

31/10/21 14:12

Sin tiempo parar morir

FECHA	SALA	FILA/ASIENTO
17/11/21 20:00	1	6 / 9

CINE CINEKINELLY

27/10/21 13:52

Interstellar

FECHA	SALA	FILA/ASIENTO
27/10/21 21:00	1	4 / 6

CINE CINEKINELLY

Figura 8.25: Vista de todas las entradas de un usuario

Para añadir un nuevo cine la plataforma debemos pulsar en el botón “Añadir cine” en el menú de “Administración” (sólo para usuarios administradores) y aparecerá un formulario como el que muestra la figura 8.26, se deben llenar todos los datos del cine y cuándo se han llenado pulsar el botón de “Configurar salas” para indicar el tamaño de filas y asientos que tendrá cada sala.

Añadir cine		
Nombre	Localización	Precio de las entradas (estándar)
Cine de pruebas	Barcelona	5
Número de salas	<input type="button" value="Configurar salas"/>	
3		
Sala 1	<input type="button" value="Número de filas"/> <input type="button" value="Número de asientos por filas"/>	
Sala 2	<input type="button" value="Número de filas"/> <input type="button" value="Número de asientos por filas"/>	
Sala 3	<input type="button" value="Número de filas"/> <input type="button" value="Número de asientos por filas"/>	
<input type="button" value="Añadir"/>		

Figura 8.26: Formulario para añadir un nuevo cine

Para añadir una nueva película o evento a la plataforma debemos pulsar en el botón “Añadir evento” en el menú de “Administración” (sólo para usuarios administradores) y aparecerá un formulario como el que muestra la figura 8.27, se deben llenar todos los datos de la película, muchos de ellos opcionales y pulsar el botón de “Añadir” para que se guarden los datos en la plataforma.

The screenshot shows a dark-themed web form titled "Añadir evento". The form fields are organized into several sections:

- Título:** A text input field.
- Tipo:** A dropdown menu set to "Película".
- Género:** A dropdown menu listing "Acción", "Comedia", "Drama", and "Fantasía".
- Descripción:** A large text area.
- Duración (mins):** A text input field.
- Valoración:** A text input field.
- Fecha de estreno:** A date selection field with dropdowns for "Mes", "Dia", and "Año".
- Recomendada para:** A dropdown menu set to "Todos los públicos".
- Director/es:** A text input field.
- Actores:** A text input field with placeholder text "Los nombres de los actores separados por comas".
- Frase de película:** A text input field.
- Foto de portada:** A file upload section with "Buscar..." and "Browse" buttons.
- Foto de fondo:** A file upload section with "Buscar..." and "Browse" buttons.
- Trailer (URL Youtube):** A text input field.
- En cartelera:** A toggle switch.
- Añadir:** A blue "Añadir" button at the bottom.

Figura 8.27: Formulario para añadir un nuevo evento o película

Para añadir una nueva sesión a la plataforma debemos pulsar en el botón “Añadir sesión” en el menú de “Administración” (sólo para usuarios administradores) y aparecerá un formulario como el de la figura 8.28, se deberá indicar la película o evento del queremos añadir la sesión, el cine, la fecha y el idioma.



El formulario se titula “Añadir nueva sesión”. Contiene los siguientes campos:

- Evento:** Un campo desplegable con la opción “Interstellar”.
- Cine:** Un campo desplegable con la opción “Cine Kinefily”.
- Fecha:** Un campo que incluye selectores para Mes, Dia, Año, Hora y Minutos.
- Idioma:** Un campo desplegable con la opción “Castellano”.
- Añadir:** Un botón azul.

Figura 8.28: Formulario para añadir una nueva sesión

Para obtener un listado de todos los usuarios registrados en la plataforma se deberá pulsar el botón de “Usuarios” en el menú de “Administración” y se tendrá un listado como el que aparece en la figura 8.29 con los detalles del usuario. Pueden verse todos sus detalles pulsando el botón con el icono de un ojo o eliminar el usuario pulsando el botón con el icono de la papelera.

Todos los usuarios				
Foto	Usuario	Email	Miembro desde	Acciones
	Sergio Azañón	sergiocantero8@gmail.com	2021/06/18 01:40	 
	Pepe Gonzalez	pepe@gmail.com	2021/10/14 18:55	 
	Jose Rodriguez	jose@gmail.com	2021/10/25 14:43	 

Figura 8.29: Usuarios del sistema

Para visualizar el *log* del sistema deberemos pulsar el botón de *Log* en el menú de “Administración” (solo para usuarios administradores) y aparecerá un listado con toda la información de procesos que han ocurrido en la plataforma como pueden ser compra de entradas, registro de usuarios, mensajes de error...etc.

ID	Fecha	Tipo	Información
67	2021-11-08 20:33	EXITO	Se ha eliminado el usuario con ID 21
64	2021-10-31 14:16	EXITO	Se han reservado 2 asientos para la session con ID13 y con email asociado jose@hotmail.com
65	2021-10-31 14:16	EXITO	Se han reservado 2 asientos para la session con ID13 y con email asociado jose@hotmail.com
66	2021-10-31 14:16	EXITO	Se le han enviado las entradas correctamente al email jose@hotmail.com
61	2021-10-31 14:12	EXITO	Se han reservado 3 asientos para la session con ID12 y con email asociado sergiocantero8@gmail.com
62	2021-10-31 14:12	EXITO	Se han reservado 3 asientos para la session con ID12 y con email asociado sergiocantero8@gmail.com

Figura 8.30: *Log* del sistema

Si entramos a la plataforma desde un dispositivo móvil como muestra la figura 8.31, la página está diseñada para ser *responsive* y el menú que se encontraba en el *header* pasa a ser un menú lateral desplegable.

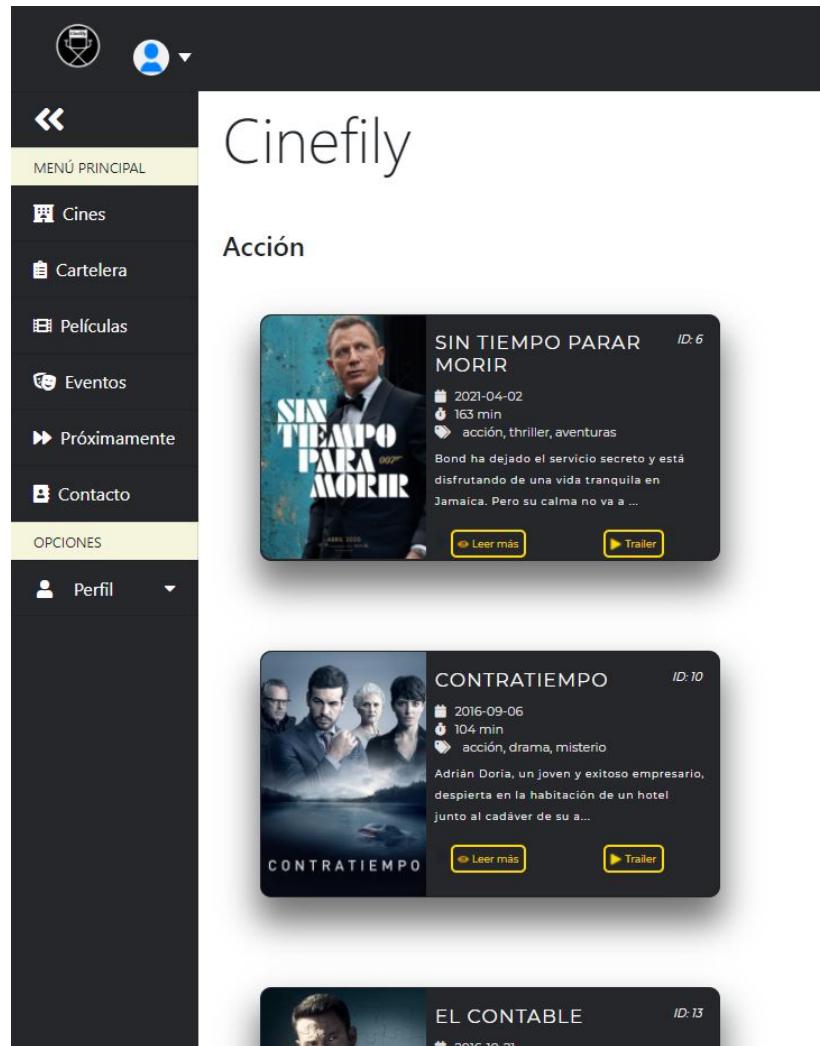


Figura 8.31: Vista de la página principal desde las dimensiones de un móvil

Todos los listados y visualizaciones de objetos tienen una paginación como se muestra en la figura 8.32

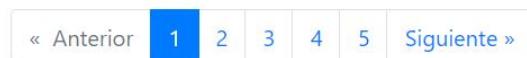


Figura 8.32: Paginación de un listado

## 8.2. Código fuente

El código fuente del proyecto se encuentra en <https://github.com/sergiocantero8/Cinefily> con todos los *commits* que se han ido haciendo a lo largo de su desarrollo.

# Bibliografía

- [1] Aguilar®, José María: *Diagrama MVC*. <https://www.campusmpv.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx>, Octubre 2019.
- [2] AIMC®: *23º Edición del censo de Salas de Cine*. [https://www.aimc.es/a1mc-c0nt3nt/uploads/2021/07/2021\\_07\\_22\\_NP\\_Censo\\_Cine\\_AIMC\\_2021.pdf](https://www.aimc.es/a1mc-c0nt3nt/uploads/2021/07/2021_07_22_NP_Censo_Cine_AIMC_2021.pdf), 2021.
- [3] ®: *Kinepolis*. <https://kinepolis.es>.
- [4] ®: *IMDB*. <https://imdb.com>.
- [5] ®: *Cinesa*. <https://www.cinesa.es/>.
- [6] Wikipedia®: *Proceso para el desarrollo de software*. [https://es.wikipedia.org/wiki/Proceso\\_para\\_el\\_desarrollo\\_de\\_software](https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software), M.
- [7] Ionos®, Digital Guide: *El modelo en cascada: desarrollo secuencial de software*. <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/>, Marzo 2019.
- [8] Billin®: *Calculadora para contratar a trabajador*. <https://www.billin.net/calculadora-contratar-trabajador/>.
- [9] país®, El: *Calculadora sueldo neto trabajador*. <https://cincodias.elpais.com/herramientas/calculadora-sueldo-neto/>.
- [10] ®: Universidad de Sevilla: *Guía para la gestión de proyectos de investigación*. [https://www.upo.es/cms1/export/sites/upo/area-investigacion/guia-gestion/GUIA-GESTION-PROYECTOS\\_UP0-V05\\_03\\_19.pdf](https://www.upo.es/cms1/export/sites/upo/area-investigacion/guia-gestion/GUIA-GESTION-PROYECTOS_UP0-V05_03_19.pdf).
- [11] 830®, IEE: *Especificación de Requisitos Software según el estándar de IEEE 830*. <https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>, Octubre 2008.
- [12] UGR: *Asignatura Fundamentos de la Ingeniería del Software (UGR) - Tema 2.1 (Introducción a los requisitos)*. 2018.
- [13] UGR: *Asignatura Fundamentos de la Ingeniería del Software (UGR) - Tema 2.3 (Casos de uso)*. 2018.

- [14] Etecé®, Editorial: *Definición de diagrama*. <https://concepto.de/diagrama/>, Agosto 2021.
- [15] ®: *Digital Guide Ionos*. <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagramas-de-secuencia/>.
- [16] Wikipedia®: *Normalización de bases de datos*. [https://es.wikipedia.org/wiki/Normalizaci%C3%B3n\\_de\\_bases\\_de\\_datos](https://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_bases_de_datos).
- [17] Wikipedia®: *Arquitectura cliente-servidor*. <https://es.wikipedia.org/wiki/Cliente-servidor>.
- [18] Wikipedia®: *Patrón MVC*. <https://es.wikipedia.org/wiki/Modelo-Vista-Controlador>.
- [19] Spittel®, Ali: *What is a Web Framework, and Why Should I use one?* <https://welearncode.com/what-are-frontend-frameworks/>, 2021.
- [20] Courseya®: *What is laravel features and advantages*. <https://www.courseya.com/blog/what-is-laravel-features-and-advantages/>, 2021.
- [21] Oragency®: *Everything about Zend framework(Laminas Project)*. <https://oragency.com/web-design/everything-about-zend-frameworklaminas-project/>, 2021.
- [22] Wikipedia®: *Lenguaje UML*. [https://es.wikipedia.org/wiki/Lenguaje\\_unificado\\_de\\_modelado](https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado), 2021.
- [23] UML®, Diagramas: *Diagramas UML*. <https://diagramasuml.com/diagrama-de-clases/>, 2021.
- [24] Doctrine®: *Doctrine*. <https://www.doctrine-project.org/>, 2021.
- [25] Symfony®: *Symfony Local Server*. [https://symfony.com/doc/current/setup/symfony\\_server.html](https://symfony.com/doc/current/setup/symfony_server.html), 2021.
- [26] XAMPP®: *Apache + MariaDB + PHP + Perl*. <https://www.apachefriends.org/es/index.html>, 2021.
- [27] Wikipedia®: *Bcrypt*. <https://en.wikipedia.org/wiki/Bcrypt>, 2021.
- [28] Generator®, QR Code: *QR Code Generator*. <https://es.qr-code-generator.com/>, 2021.
- [29] Paypal®: *Create Smart Payment Button*. <https://www.paypal.com/buttons/smart?flowloggingId=f23728612526c>, 2021.
- [30] Symfony®: *The Lock Component*. <https://symfony.com/doc/current/components/lock.html>, 2021.
- [31] TMDB®: *The Movie Database API*. <https://developers.themoviedb.org/3/getting-started/introduction>, 2021.

- [32] Github<sup>®</sup>: *Knp Paginator Bundle*. <https://github.com/KnpLabs/KnpPaginatorBundle>, 2021.
- [33] Tester<sup>®</sup>: *Caja blanca vs caja negra*. <https://www.testermoderno.com/caja-blanca-vs-caja-negra/>, 2021.
- [34] PHPUnit<sup>®</sup>: *PHPUnit Framework*. <https://phpunit.de/>, 2021.