PROYECTO INTERMODULAR
“TAQUILLA DE CINE ONLINE”

INTEGRANTES DE GRUPO

Andrés Pérez Guardiola
Siham Fodil Ferouli
Ismael Lucas Aparicio
Juan Mira Alcántara - TEAM LEADER

Repositorio GIT:

https://github.com/juanmiraaaa/ProyectoIntermodular.git

ÍNDICE:

1. Resumen

Al Índice

El objetivo principal del proyecto es crear una taquilla online para un cine. Se pretende que la aplicación permita a los clientes de la misma consultar las películas disponibles en los distintos cines, mirar la disponibilidad de sesiones y butacas en las distintas sesiones, y comprar sus entradas para ver la película. La aplicación es una aplicación web e incorpora una api escrita en Java (Spring Boot).

2. Introducción

Al Índice

La aplicación web “Taquilla de Cine Online” permite consultar las diferentes carteleras de los cines y comprar entradas para las sesiones.

El objetivo es automatizar el proceso, al mismo tiempo que permitimos un mejor control de que butacas se venden, y permitir a los clientes hacerlo todo de forma automática y sencilla.

3. Descripción del proyecto

Al Índice

Una empresa opera varios cines en el entorno. La empresa quiere que sus clientes puedan consultar la cartelera, ver los horarios de las distintas sesiones disponibles, y que se les permita escoger una butaca de la sala y comprar su entrada para ver la película.

La App debe registrar las Películas, Sesiones, Salas, Butacas y clientes que han reservado.
Los clientes deben poder ver los detalles de las películas, director, elenco de actores, breve sinopsis, duración, género, edad recomendada(PEGI), y trailer de la misma. Al usuario le aparecerán las sesiones de los distintos días, una vez seleccionado el día, a este le aparecerán las butacas disponibles.

Los usuarios se pueden registrar. Si quieres comprar tienes que estar registrado. Una vez compras la entrada, esta aparecerá en el área de cliente junto con la factura.

4. Organización del proyecto

Al Índice

Tareas y su estimación temporal. (La duración ha sido ajustada a la duración real una vez fuimos terminando las mismas)

Planificación:

- Organización del equipo
- Establecer metodologías de trabajo grupal
- Brainstorming
- Detallar la idea del proyecto

Análisis de requisitos:

- Requisitos funcionales
- Requisitos no funcionales
- Requisitos Base de Datos (BBDD)
- Casos de uso

Diseño:

- Diseño conceptual y lógico de BBDD
- Diseño Diagrama de Clases
- Wireframes y Mockups

Codificación:

- Codificación Lado Cliente
- Codificación Lado Servidor

Testing:

Planificación temporal inicial del proyecto:

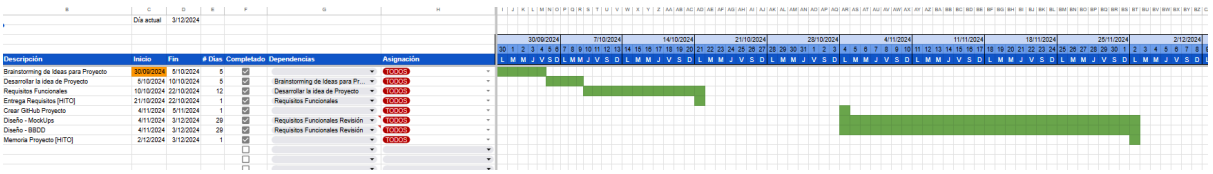
Tarea	Duración	Dependencia
Planificación	10	Ninguna

Tarea	Duración	Dependencia
Análisis de requisitos	14	Planificación
Diseño BBDD	30	Análisis de requisitos
Diseño MockUps	30	Análisis de requisitos
Diseño Diagrama de Clases	30	Análisis de requisitos
Codificación Lado Cliente	60	Diseño MockUps
Codificación Lado Servidor	90	Diagrama de Clases Diseño BBDD
Testing	10	Codificación Lado Cliente Codificación Lado Servidor
Despliegue Producción	5	Testing

Planificación temporal REAL del proyecto:

Tarea	Duración	Dependencia
Planificación	10	Ninguna
Análisis de requisitos	12	Planificación
Diseño BBDD	35	Análisis de requisitos
Diseño MockUps	29	Análisis de requisitos
Diseño Diagrama de Clases	14	Análisis de requisitos
Codificación Lado Cliente	– (Inconcluso)	Diseño MockUps
Codificación Lado Servidor	– (Inconcluso)	Diagrama de Clases Diseño BBDD
Testing		Codificación Lado Cliente Codificación Lado Servidor
Despliegue Producción		Testing

Diagrama de Gantt:



Descripción	Inicio	Fin	# Dias	Completado	Dependencias	Asignación
Brainstorming de Ideas para Proyecto	30/09/2024	5/10/2024	5	☑		TODOS
Desarrollar la idea de Proyecto	5/10/2024	10/10/2024	5	☑	Brainstorming de Ideas para Pr...	TODOS
Requisitos Funcionales	10/10/2024	22/10/2024	12	☑	Desarrollar la idea de Proyecto	TODOS
Entrega Requisitos [HITO]	21/10/2024	22/10/2024	1	☑	Requisitos Funcionales	TODOS
Crear GitHub Proyecto	4/11/2024	5/11/2024	1	☑		TODOS
Diseño - ModUps	4/11/2024	3/12/2024	29	☑	Requisitos Funcionales Revisión	TODOS
Diseño - BBDD	4/11/2024	3/12/2024	29	☑	Requisitos Funcionales Revisión	TODOS
Memoria Proyecto [HITO]	2/12/2024	3/12/2024	1	☑		TODOS

Comentarios Planificación Temporal:

Se planteó que el final de la etapa de codificación sería el día 12/05/2025:

Descripción	Inicio	Fin	# Dias	Completado	Dependencias	Asignación
				☐		
Diagrama Lógico - BBDD	24/02/2025	3/03/2025		☐	Diagrama Conceptual - BBDD	
	10/03/2025	17/03/2025		☐		
	17/03/2025	24/03/2025		☐		
Diseño Físico - BBDD	24/03/2025	31/03/2025		☐	Diagrama Lógico - BBDD	
	31/03/2025	07/04/2025		☐		
	14/04/2025	21/04/2025		☐		
	21/04/2025	28/04/2025		☐		
Crear API - en Java	28/04/2025	5/05/2025		☐		
	5/05/2025	12/05/2025		☐		
	12/05/2025			☐		
	vacaciones			☐		

pero la planificación real no se cumplió, llegando al día 25/05/2025 con solo el diseño web, teniendo implementado el diseño responsive en gran parte de nuestras vistas, pero con mucho por hacer todavía. La página web todavía no tiene servidor con el que comunicarse, y la API está construida en los modelos (clases cine, butacas, etc), pero no dispone de ningún servicio ni lógica de negocio implementada.

Estimamos que necesitaríamos al menos de 2 a 4 meses más para tener algo medianamente funcional.

5. Análisis de Requisitos

Al Índice

Al desglosar la descripción del proyecto, hemos identificado 5 apartados: Cartelera, Proceso de Selección de Butaca y Sesiones, Proceso de Compra, Registro y Área Cliente. Y en cada parte hemos identificado una serie de vistas que el cliente puede ver y dentro de cada vista, las funcionalidades que puede hacer el cliente con ellas.

Cartelera:

La cartelera mostrará las películas disponibles y estrenos.
Los usuarios pueden consultar detalles de las películas, así como detalles de la emisión, como en qué cine o qué horarios.

Butacas:

Los usuarios pueden consultar la disponibilidad de butacas para una sesión. Pueden seleccionar butacas libres y empezar un proceso de compra.

Compra:

El usuario puede pagar por esas entradas y recibirá una entrada digital y una factura de su compra.

Registro:

Para comprar, es necesario que el usuario esté registrado, ya que necesitamos relacionar la entrada con el cliente que la compró.

Área Cliente:

Los usuarios registrados podrán consultar toda la información relativa a las entradas y sus compras en el área cliente.

Análisis de requisitos y Casos de uso:

Análisis de Requisitos (Enlace)

6. Tecnologías Usadas

Al Índice

Sobre la tecnología, hemos decidido utilizar la tecnología listada a continuación:

MockUps:

- Figma

Frameworks:

- FrontEnd: next.js, bootstrap
- BackEnd: SpringBoot

IDEs:

- VS Code
- IntelliJ

BBDD:

- BBDD: MySQL

Testing:

- Front: Playwright
- Back: JUnit

Sistema Operativo:

- Windows

Hosting:

- Vercel

Spring Boot se emplea para crear una API con la que comunicar el frontend con la base de datos. Tendrá toda la lógica de negocio. La base de datos será una base de datos MySQL. En caso que se desarrolle una aplicación móvil a futuro, esta aplicación se podría conectar a la misma API.

Next.js se usa para la capa visible para los usuarios. Esta capa se comunica directamente con los usuarios y realiza peticiones a la API cuando es necesario. Next.js nos permite trabajar con componentes.

Una vez tengamos las dos capas, se podrán implementar en servidores distintos, siendo el next.js un proxy entre cliente y servidor.

7. Diseño

Al Índice

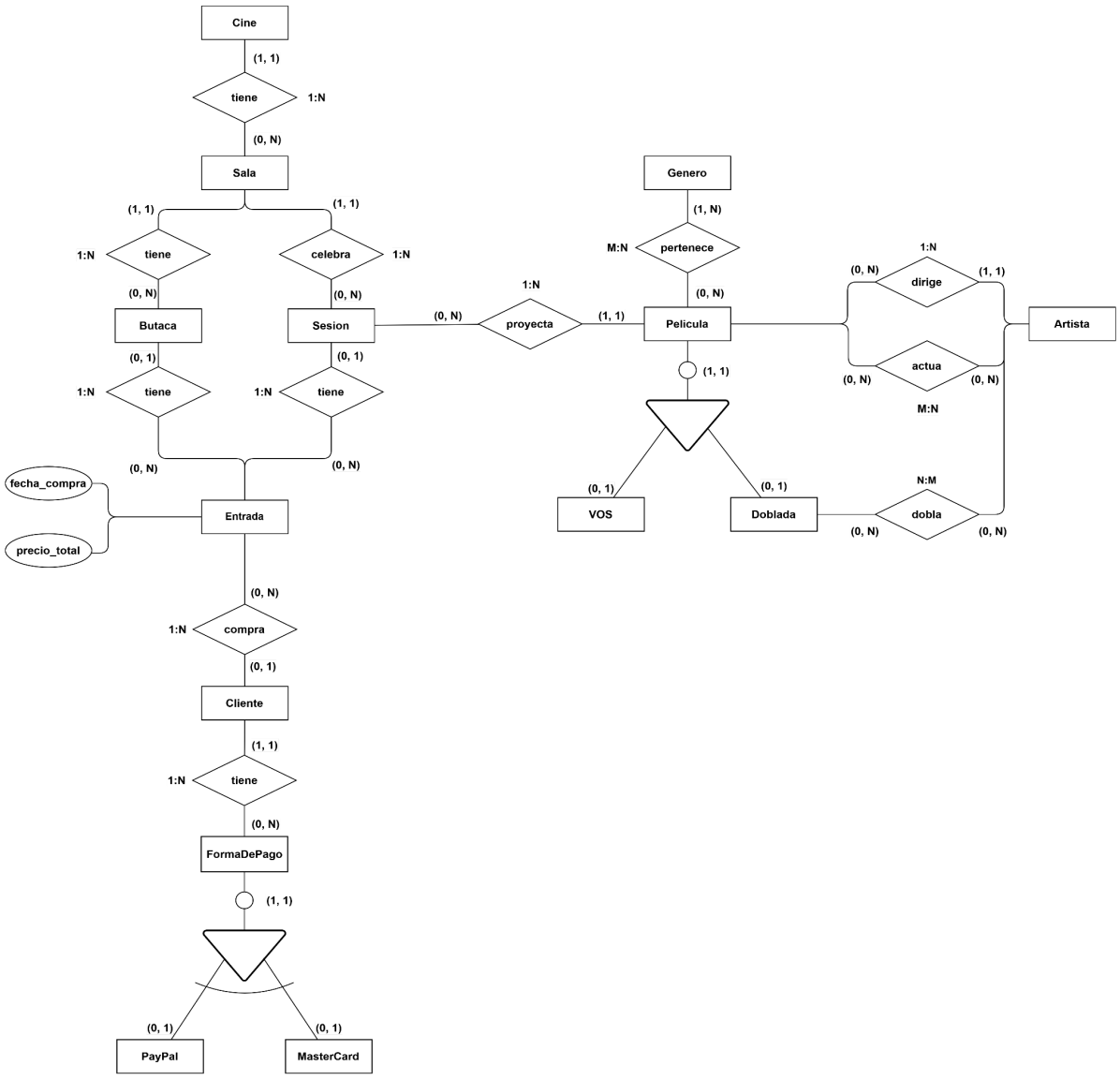
Diseño BBDD:

Para la realización del **diseño conceptual** de la base de datos, realizamos una extensión de los requisitos funcionales de la aplicación, teniendo especial atención a qué datos eran necesarios guardar en nuestra base de datos.

Encontramos varias opciones que satisfacen los requisitos, el primero incluye una relación ternaria de muchos a muchos sobre la que recae el peso de la aplicación, ya que relaciona las sesiones de cine, con los clientes y las butacas. Dicha relación de compra se convierte en una entrada donde se registran las entradas, una vez el cliente completa todo el proceso de compra en la aplicación.

En nuestro planteamiento decidimos separar los campos de película de los actores, directores, etc. Esto permite mantener una entidad llamada Artísticas que se relaciona con Películas. La relación es, en muchos casos, de muchos a muchos, lo cual se traduce en la aparición de nuevas tablas como: actua_pelicula, dobla_pelicula. De esa forma podemos buscar y relacionar un actor con todas las películas en las que ha participado, lo mismo con directores y actores de doblaje.

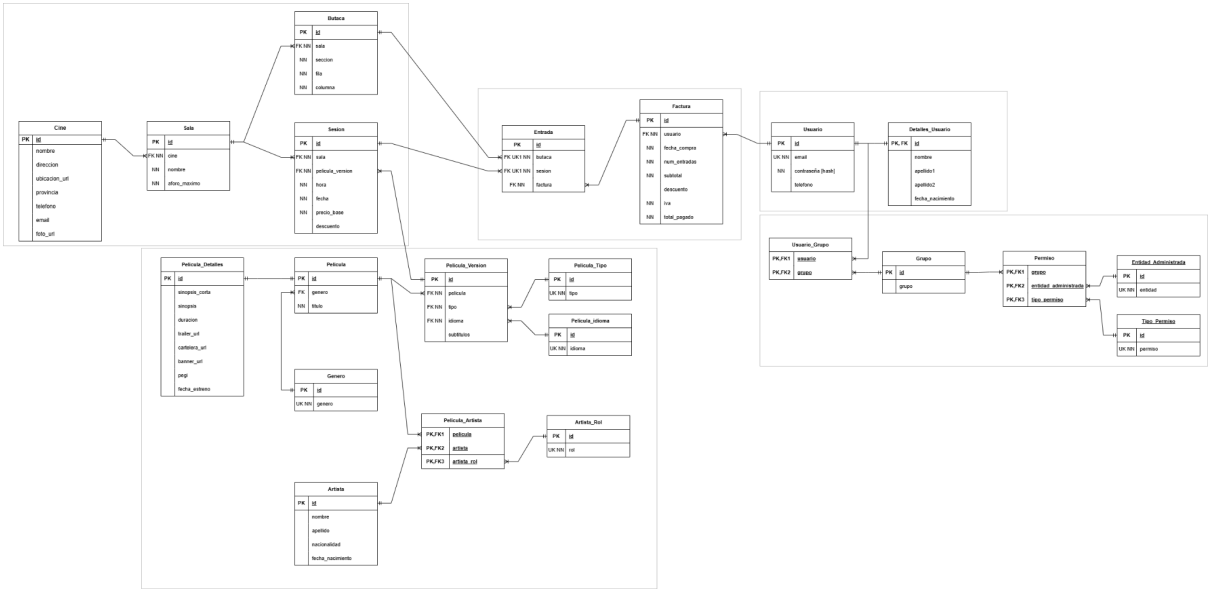
En cuanto a los géneros, hemos optado por permitir que sea una relación muchos a muchos, por lo que podemos tener películas que pertenezcan a muchos géneros distintos.



[Diagrama ER](#) (enlace)

TODO: Determinar si es necesario tener guardado la tarjeta de crédito o el PayPal del cliente en nuestra base de datos o pedirle que introduzca esta información en un formulario.
TODO: Decidir si es necesario tener artistas, o separarlo en actores, directores, o directamente no tener artistas y tener un campo String largo que contenga todo los actores y directores. (Si lo tenemos en una entidad, lo podemos buscar y relacionar en un buscador).

En el **diseño lógico** de la base de datos se tomaron decisiones que cambiaron el diseño conceptual inicial. Entre los cambios, se ha modificado como se guarda la información de películas. Esto se debe a que una película se puede emitir con ciertas características, no solo su idioma, también depende de la equipación de la sala y más detalles. Sobre cliente, pensamos en usuarios y que un usuario podría ser cliente, o tomar distintos roles como administrador, vendedor, etc. Estos tendrían unos permisos asociados al grupo al que pertenecen que les permite, mediante un área administrador de sitio, modificar las sesiones del cine, o hacer otras operaciones de gestión dentro del cine.



MockUp:

Llevamos los casos de uso a una maqueta realizada en Figma. Actualmente tenemos 13 vistas creadas y donde hemos estado discutiendo cómo organizar cada una de las vistas y hemos tomado decisiones en consecuencia.

A modo de presentación, mostramos las vistas que definen el proceso de compra de una entrada. **También incluimos un enlace al mockup en FIGMA donde se puede ver y testear la navegación.**

MockUp Interactivo (Enlace FIGMA)

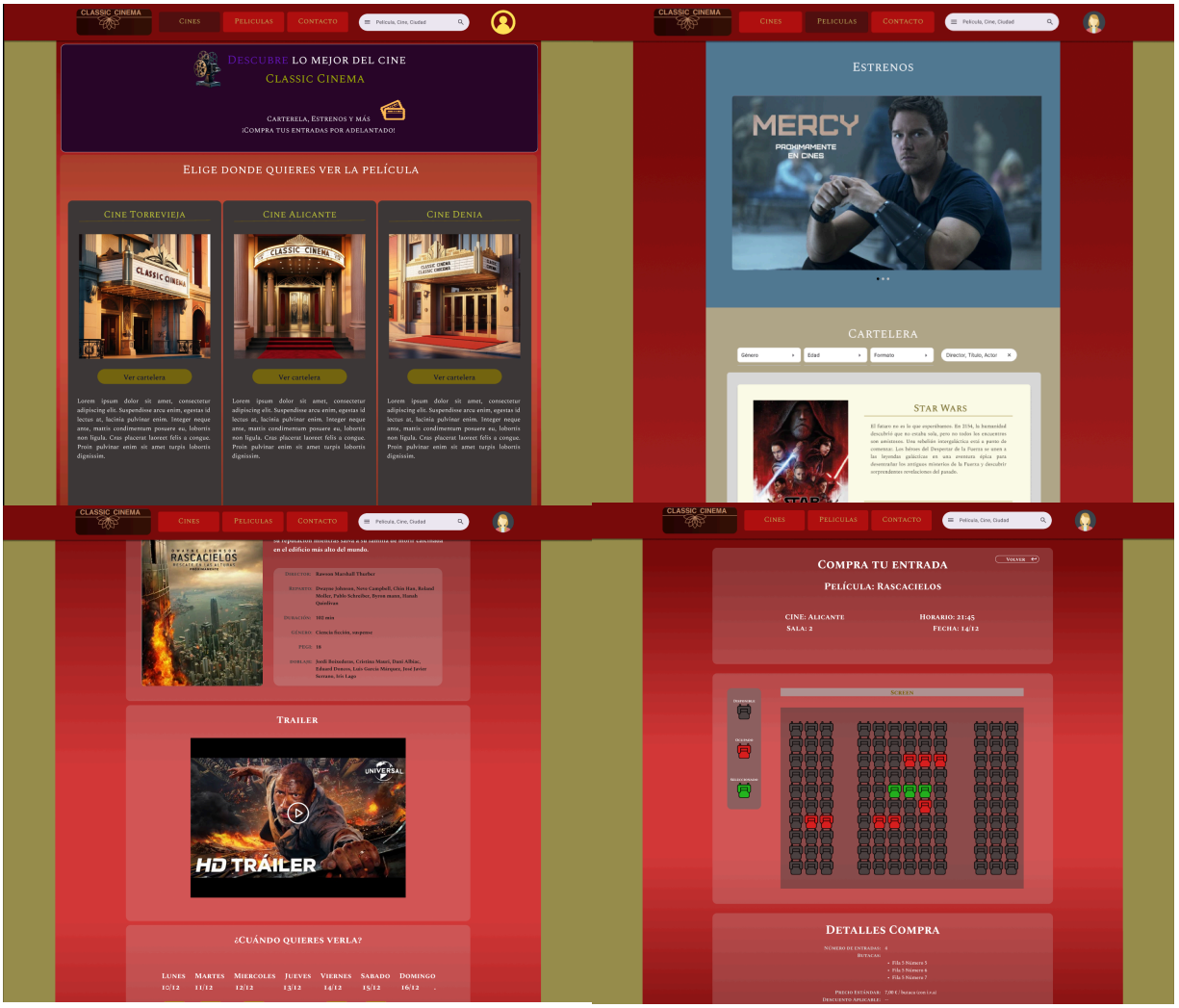
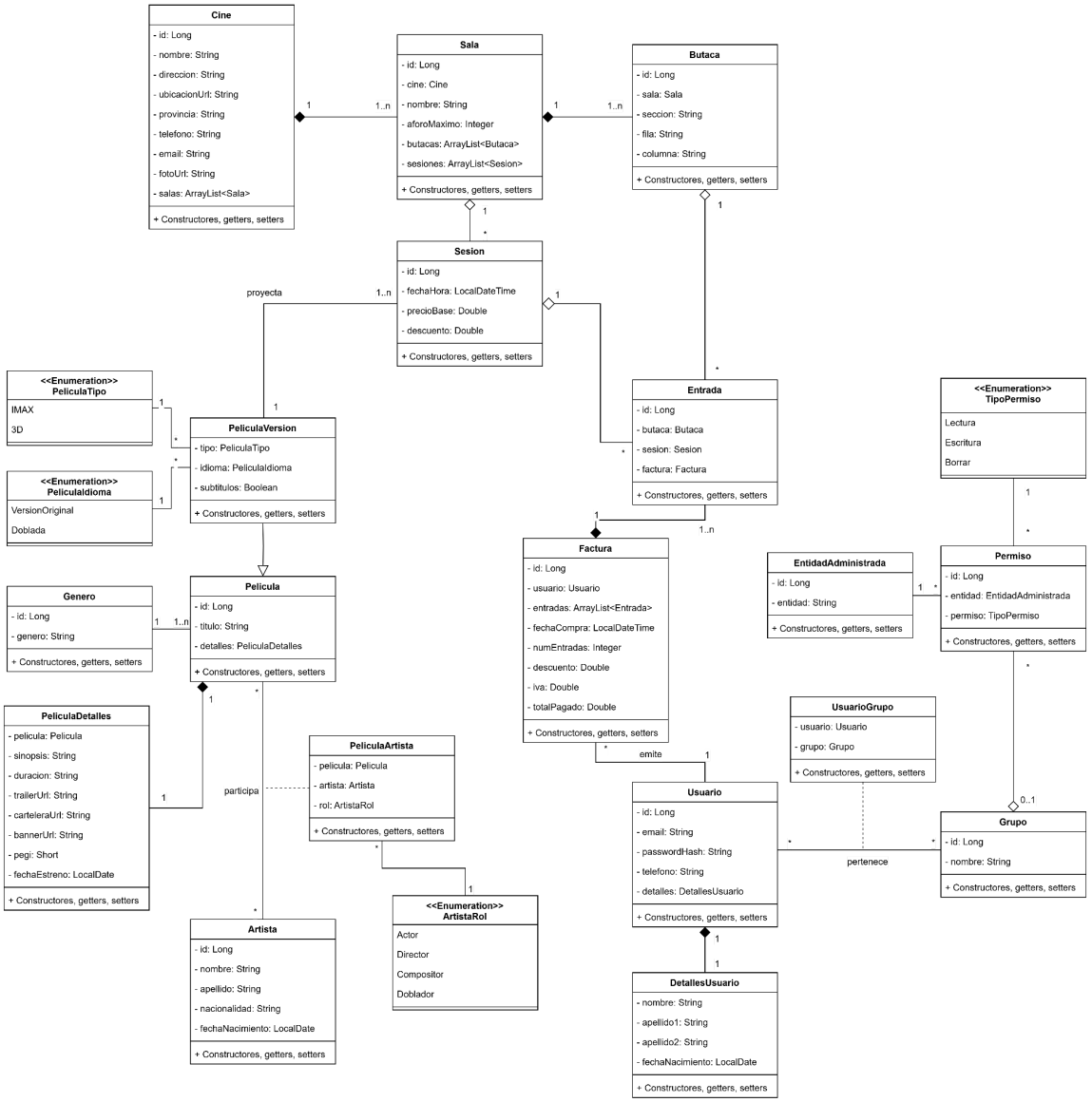


Diagrama de Clases:



El diagrama de clases aquí presentado muestra el modelo del cine y tiene prácticamente una correlación directa con el diseño lógico de la base de datos.

8. Codificación

Al Índice

Lado Cliente:

Hemos descartado el uso de next.js como habíamos planteado al principio en el apartado 6 al no tener conocimientos de javascript y menos aún de un framework

como next. Hemos realizado nuestras webs usando html y css puro, al que luego hemos incorporado **bootstrap** y algunos componentes **SASS** creados por nosotros mismos. Hemos conseguido que el resultado visual de nuestra web sea prácticamente el mismo que el de nuestros MockUps, aunque algunos componentes se ven ligeramente distintos en nuestra versión web.

A continuación vamos a reparar los puntos que deberían de mejorarse:

- Hemos priorizado tener todas las vistas posibles en los plazos propuestos. Nos encontramos ahora mismo en mitad del proceso de implementación de bootstraps a nuestras vistas.

- Algunas páginas implementan bootstrap con algunos componentes de SASS custom y algunas pinceladas de CSS por encima, por lo que quizás plantea el problema de que nuestras páginas cargan demasiados recursos individuales y sería interesante terminar de integrar todo dentro de nuestra versión modificada de bootstrap, para cargar un solo archivo de CSS y que estén todas las clases para el sitio web.

- Nuestra web muestra un contenido estático. Deberíamos recibir información del Lado Servidor y mostrarla al cliente.

- Página de cartelera debería de mostrar listado de películas en emisión (recibir listado desde servidor) y permitir paginación o buscar mediante filtros. Requiere JavaScript para poder hacerlo.

- Plantear nuevos requisitos funcionales, como un área de administración de sitio para poder ingresar nuevas películas en la base de datos, programar emisiones, etc.

Lado Servidor:

Hemos implementado el modelo de datos y, usado el ORM de spring boot, implementado también la base de datos para la persistencia.

Aunque tenemos el modelo de datos que vamos a emplear, no tenemos ni un solo servicio que implemente lógica de negocio, como comprar entradas, obtener películas en emisión, ver entradas disponibles, etc. Tampoco tenemos ningún endpoint para iniciar nuestros inexistentes servicios desde web.

La falta de conocimientos sobre el tema nos deja en standby por el momento en este tema.

9. Testing

Al Índice

(Requiere de terminar la codificación, o tener planteado hacer codificación y testing simultáneamente)

10. Integración y Despliegue

Al Índice

11. Conclusión

Valoración del resultado obtenido:

Por lo general estamos bastante satisfechos en lo que hemos sido capaces de hacer, aunque no hemos conseguido satisfacer los requisitos funcionales del cliente en el tiempo establecido. Pensamos que los conocimientos que teníamos para completar los requisitos funcionales del proyecto eran insuficientes, incluso a finalización del primer año del curso académico.

Los retos y dificultades que hemos ido encontrando son:

- Trabajo en equipo. Tanto en las herramientas como en la propia organización.
    - Nos ha costado unificar criterios a la hora de hacer las cosas, o cambios de criterios en mitad del proceso de desarrollo que afectaba críticamente a todo el trabajo realizado previamente.
    - La realización de reuniones semanales se vió truncada por la actividad de la academia (exámenes), por lo que en momentos del proyecto hemos estado vagando cada uno en solitario.
    - GIT. Ha causado en algunas situaciones algunas crisis, como de colisiones entre distintos merges, problemas de manejo en general y falta de disciplina a la hora de hacer las cosas. (Mal uso de ramas, conflictos entre ramas). En muchas situaciones las reuniones se han centrado sobre aclarar conceptos y manejo de GIT, y no tanto sobre el proyecto en sí mismo.
- HTML.
    - Organizar el contenido de forma similar en todas las vistas de la web.
    - Emplear componentes reutilizables, como navbar o footer. Empezamos usando IFRAMEs para conseguir este efecto, pero vimos que muchas vistas no funcionaban bien debido a reglas CSS propias del documento.

- CSS.
    - Reglas CSS propias para cada documento. Muchos componentes HTML reutilizables se veían de forma completamente distinta dependiendo de la vista en la que se utilizara, al colisionar con reglas propias de la página en la que se usaba. (esto se soluciona en parte al usar sass y bootstrap)
    - Conseguimos hacer alguna página responsive usando media-queries, pero requería de crear muchas reglas para distintos tamaños de pantalla y hacía el CSS ilegible.
- Bootstrap.
    - Integración a veces complicada, al tener mucho hecho previamente en CSS, muchas cosas estaban siendo modificadas en el CSS propio de la página y requería de primeramente organizar todos los elementos empleando clases de bootstrap y, posteriormente, aplicar reglas CSS para incluir colores o detalles menores.
    - Aún hay algunas reglas CSS que deberían de ser eliminadas y permanecen en el CSS porque no se sabe que efectos puede tener el eliminarlas.

12. Documentos adjuntos y Enlaces de Interés

Al Índice

Documentos Adjuntos:

- **Diagrama Gantt:**

  Diagrama con la Distribución temporal de los distintos apartados del proyecto, así como su estimación en días para cada uno
  doc/DiagramaGantt/**DiagramaGantt.pdf**

- **Análisis de Requisitos:**

  Requisitos funcionales y no funcionales del proyecto, así como su caso de uso:
  doc/AnalisisRequisitos/**AnalisisRequisitos.pdf**

- **Diagrama ER BBDD:**

  Diseño conceptual de la base de datos:
  doc/DiagramaBBDD/**DisenyoBBDD.pdf**

- **MockUp:**

  Versión PDF por si fuera necesario o hubiera fallado la versión online doc/MockUp/**TaquillaClassicCinema.pdf**

Enlaces de Interés:

- **GitHub del grupo:**

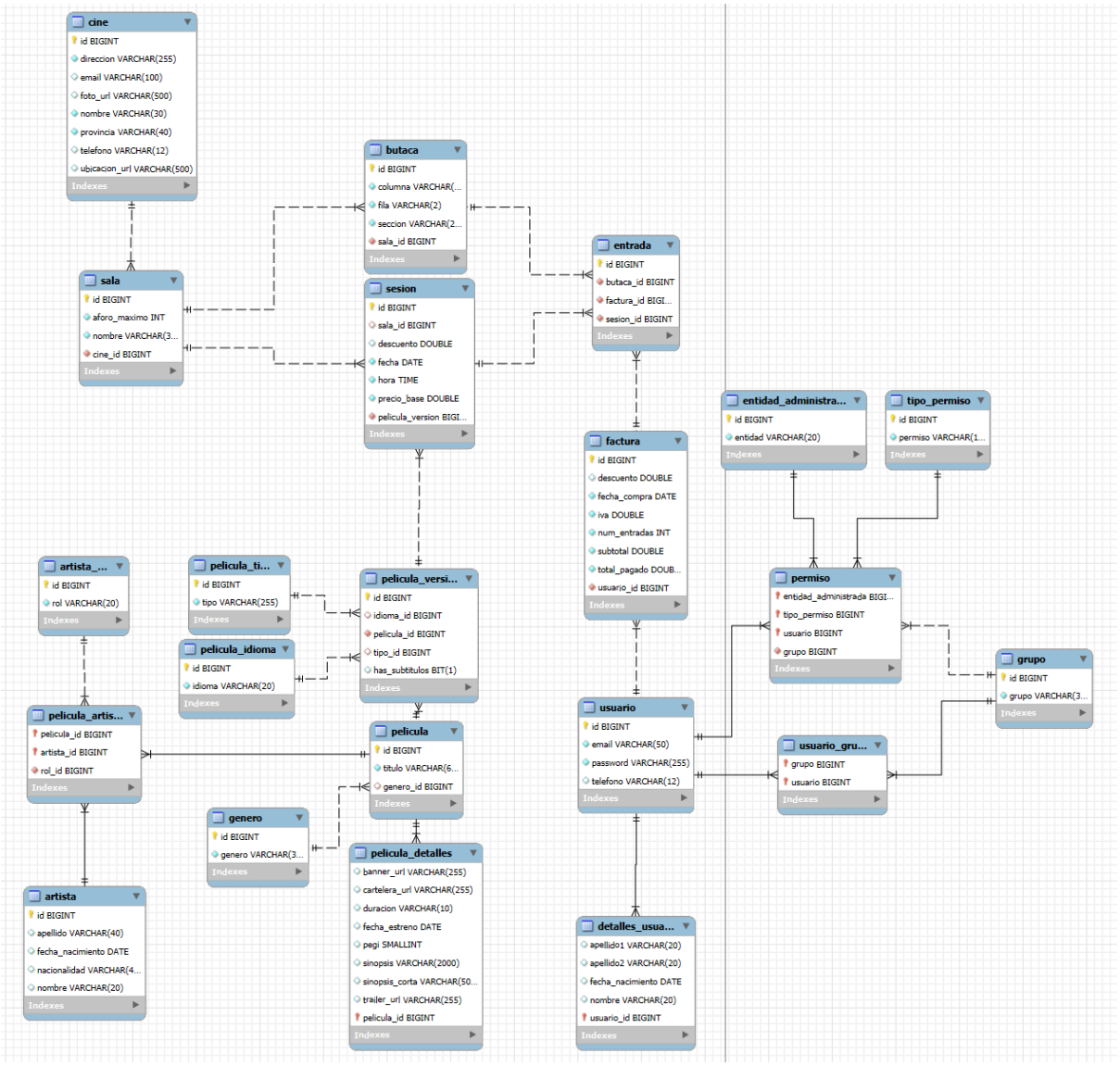
  GitHub público de grupo.
  GitHub

- **MockUp interactivo:**

  MockUp interactivo en Figma. Se puede probar la navegación.
  MockUP

Anexos:

Diseño Físico Base de datos:



Este es el diagrama lógico de la base de datos implementada en MySQL. La implementación de la base de datos se realiza a partir de la propia API. El ORM de Spring Boot levanta la base de datos en caso de que no exista la primera vez que se ejecuta.