

Seminario de Práctica Informática

Trabajo Práctico 4

Lic. Informática.

Año 2024

Jonatan Diaz

Contenido

Primer TP	3
Título del Proyecto	3
Introducción	3
Descripción del CineTech:	3
Descripción de la Problemática:	3
Justificación	3
Mejora de la Experiencia del Cliente	4
Adaptación a las Tendencias Digitales	4
Optimización de la Gestión de Ventas	4
Definiciones del Proyecto y del Sistema	4
Definición del Proyecto	4
Alcance del Proyecto:	4
Stakeholders Involucrados:	4
Definición del Sistema	5
Elicitación	5
Fuentes de Elicitación	5
Requisitos Identificados	5
Validación de Requisitos	6
Conocimiento del Negocio	6
Análisis del Mercado Cinematográfico	6
Prácticas Operativas de CineTech	6
Necesidades y Expectativas de los Clientes	6
Desafíos y Oportunidades	7
Diagrama de Dominio	7
Propuesta de Solución	7
Arquitectura del Sistema	7
Capa de Presentación (Frontend):	7
Capa de Lógica de Negocio (Backend):	8
Capa de Almacenamiento de Datos (Base de Datos):	8
Tecnologías y Herramientas	8
Implementación del Sistema	8
Despliegue y Mantenimiento	9
Inicio del Análisis: Casos de Uso	9
Requerimientos Funcionales	9
Requerimientos No Funcionales	9
Diagrama de Casos de Uso:	10

Trazabilidad:	10
Descripción de los casos de uso:	11
Segundo TP	14
Etapa de análisis.....	14
Diagramas de colaboración	14
Etapa de diseño	15
Arquitectura del Sistema.....	15
Diagrama de clases.....	15
Etapa de implementación.....	15
Tecnologías Utilizadas.....	15
Diagrama de implementación	16
Etapa de pruebas.	16
Plan de pruebas.....	16
Análisis de casos de prueba	17
Definición de base de datos para el sistema.....	18
Diagrama entidad-relación de la base de datos.	18
Creación de las tablas MySQL.	19
Inserción, consulta y borrado de registros.	20
Definiciones de comunicación.	21
1. Interacción Cliente-Servidor.....	21
2. Protocolo de Comunicación	21
3. Endpoints de la API.....	21
Tercer TP	21
Explicación del Desarrollo en Java	21
Presentación del Desarrollo en Java.....	21
Estructura del Proyecto Maven	22
Explicación del Código.....	22
Repositorio	25
Cuarto TP	26
1. Selección del patrón de diseño y justificación.....	26
2. Persistencia y consulta de datos en una base de datos MySQL	26
Modelo	27
Controlador	28
Servicio.....	29
Objetos de Acceso a datos	29

Primer TP

Título del Proyecto

Venta de Entradas de Cine Online para CineTech

Introducción

Nombre del Cine: CineTech

Descripción del CineTech:

CineTech es una cadena de cines moderna y tecnológicamente avanzada que ofrece a sus clientes una experiencia cinematográfica única. Con múltiples salas equipadas con la última tecnología de proyección y sonido, CineTech se ha consolidado como un referente en la industria del entretenimiento.

Además de ofrecer una amplia selección de películas de estreno y clásicos, CineTech se destaca por su compromiso con la innovación y la comodidad del cliente. Con servicios como asientos reclinables, pantallas de gran formato y una variedad de opciones gastronómicas, CineTech se esfuerza por brindar una experiencia cinematográfica inigualable.

Descripción de la Problemática:

A pesar de su éxito y popularidad, CineTech enfrenta el desafío de optimizar el proceso de venta de entradas para mejorar la experiencia del cliente y aumentar la eficiencia operativa. Actualmente, el proceso de compra de entradas se realiza principalmente en las taquillas del cine o a través de quioscos de autoservicio en el lugar.

Sin embargo, con la creciente demanda de servicios en línea y la necesidad de adaptarse a las tendencias digitales, existe una oportunidad significativa para desarrollar una plataforma de venta de entradas en línea que permita a los clientes reservar y comprar entradas de manera conveniente desde cualquier dispositivo.

Este proyecto tiene como objetivo diseñar y desarrollar una aplicación web para CineTech que permita a los usuarios explorar películas disponibles, consultar horarios, reservar asientos y comprar entradas de forma segura y eficiente, mejorando así la experiencia del cliente y optimizando la gestión de ventas para el cine.

Justificación

En la era digital actual, la adaptación tecnológica se ha convertido en un pilar fundamental para el éxito y la competitividad de las empresas en diversos sectores, incluido el entretenimiento y la industria cinematográfica. CineTech, como una cadena de cines moderna y tecnológicamente avanzada, reconoce la importancia de ofrecer a sus clientes una experiencia de usuario mejorada y conveniente a través de soluciones digitales innovadoras.

Mejora de la Experiencia del Cliente

La implementación de una plataforma de venta de entradas en línea para CineTech tiene como objetivo principal mejorar la experiencia del cliente al proporcionar un proceso de compra de entradas más rápido, conveniente y accesible. Los clientes podrán explorar la cartelera de películas, consultar horarios, seleccionar asientos y realizar compras desde la comodidad de sus hogares o dispositivos móviles, evitando las colas y las esperas en las taquillas físicas.

Adaptación a las Tendencias Digitales

Con el crecimiento continuo del comercio electrónico y la preferencia de los consumidores por los servicios en línea, es esencial que CineTech se adapte a las tendencias digitales actuales para mantenerse relevante y competitivo en el mercado. La implementación de una plataforma de venta de entradas en línea permitirá a CineTech expandir su alcance, atraer a una audiencia más amplia y aumentar las ventas al ofrecer un canal de venta adicional y conveniente para sus clientes.

Optimización de la Gestión de Ventas

Además de mejorar la experiencia del cliente, la plataforma de venta de entradas en línea proporcionará a CineTech herramientas avanzadas para optimizar la gestión de ventas, la planificación de horarios y la asignación de asientos. Con funcionalidades como la visualización en tiempo real de la disponibilidad de asientos y la integración con sistemas de pago seguros, CineTech podrá gestionar de manera más eficiente sus operaciones y maximizar los ingresos potenciales.

Definiciones del Proyecto y del Sistema

Definición del Proyecto

El proyecto "Venta de Entradas de Cine Online para CineTech" tiene como objetivo desarrollar una plataforma de venta de entradas en línea que permita a los clientes explorar la cartelera de películas, consultar horarios, seleccionar asientos y realizar compras de manera conveniente y segura. Esta iniciativa busca mejorar la experiencia del cliente, adaptarse a las tendencias digitales actuales y optimizar la gestión de ventas para CineTech.

Alcance del Proyecto:

Desarrollo de una aplicación web responsiva accesible desde diferentes dispositivos.

Implementación de funcionalidades de registro, inicio de sesión y gestión de perfiles de usuario.

Integración con una base de datos MySQL para la persistencia de datos.

Implementación de un sistema de selección de asientos en tiempo real.

Desarrollo de un panel de administración para la gestión de películas, horarios, salas y ventas.

Stakeholders Involucrados:

Clientes: Usuarios finales que utilizarán la plataforma para comprar entradas.

Administradores: Personal de CineTech responsable de la gestión y administración del sistema.

Desarrolladores: Equipo encargado del diseño, desarrollo e implementación del sistema.

Definición del Sistema

El sistema de "Venta de Entradas de Cine Online para CineTech" estará compuesto por varios componentes interrelacionados que trabajarán juntos para ofrecer una experiencia de usuario fluida y eficiente. A continuación, se describen los componentes principales y sus funcionalidades:

Frontend: Interfaz de usuario desarrollada con tecnologías web estándar como HTML, CSS y JavaScript para garantizar una experiencia de usuario intuitiva y responsiva.

Backend: API RESTful desarrollada en Java para gestionar la lógica de negocio, la autenticación de usuarios, la interacción con la base de datos y la integración con sistemas externos, como el sistema de pago.

Base de Datos MySQL: Sistema de gestión de bases de datos relacional utilizado para almacenar información de usuarios, películas, horarios, reservas, transacciones y otros datos relevantes del sistema.

Elicitación

La elicitación es un proceso sistemático que implica recopilar información de diversas fuentes para comprender completamente los requisitos del sistema. En el contexto del proyecto "Venta de Entradas de Cine Online para CineTech", se han utilizado varias técnicas de elicitación para capturar las necesidades, expectativas y restricciones del negocio y los usuarios finales.

Fuentes de Elicitación

Entrevistas con Stakeholders: Se llevaron a cabo entrevistas con clientes, administradores y personal de CineTech para obtener insights sobre las necesidades y desafíos actuales en la gestión de ventas de entradas y la experiencia del cliente.

Encuestas y Cuestionarios: Se diseñaron encuestas y cuestionarios para recopilar información adicional sobre las preferencias de los usuarios, las funcionalidades deseadas y las áreas de mejora percibidas en el proceso actual de compra de entradas.

Análisis de Documentación Existente: Se revisaron documentos, manuales y registros existentes relacionados con las operaciones de venta de entradas de CineTech para identificar requisitos y procesos relevantes que deben ser considerados en el nuevo sistema.

Observación Directa: Se realizaron observaciones directas de las operaciones en las taquillas y los quioscos de autoservicio de CineTech para comprender mejor los flujos de trabajo actuales y las interacciones de los usuarios con el sistema existente.

Requisitos Identificados

Basado en el proceso de elicitación, se han identificado los siguientes tipos de requisitos para el sistema:

Requisitos Funcionales: Estos son requisitos que describen las funcionalidades específicas que el sistema debe ofrecer, como la gestión de usuarios, la búsqueda de películas, la selección de asientos y el proceso de compra de entradas.

Requisitos No Funcionales: Estos son requisitos que abordan aspectos de rendimiento, seguridad, usabilidad y otros criterios de calidad que el sistema debe cumplir para satisfacer las necesidades del negocio y los usuarios.

Requisitos de Datos: Estos son requisitos relacionados con la estructura y la gestión de la base de datos, incluyendo la información de usuarios, películas, horarios, reservas y transacciones.

Validación de Requisitos

Para garantizar la precisión y la relevancia de los requisitos identificados, se llevaron a cabo sesiones de validación con los stakeholders clave, donde se revisaron y confirmaron los requisitos para asegurar que reflejan con precisión las necesidades y expectativas del negocio y los usuarios finales.

Conocimiento del Negocio

El conocimiento del negocio proporciona información sobre la industria del cine, las operaciones de CineTech y las expectativas de los clientes y stakeholders. Esta sección se centra en analizar el mercado cinematográfico, las prácticas operativas de CineTech y los factores clave que influyen en la implementación del sistema de venta de entradas en línea.

Análisis del Mercado Cinematográfico

CineTech opera en un mercado competitivo y dinámico, caracterizado por una amplia variedad de opciones de entretenimiento y una demanda cambiante de los consumidores. Algunos aspectos importantes del mercado cinematográfico incluyen:

Competencia: CineTech compite con otras cadenas de cines, así como con servicios de streaming y otras formas de entretenimiento.

Tendencias de Consumo: Los consumidores están cada vez más interesados en la conveniencia y la experiencia de usuario al elegir entretenimiento.

Estacionalidad: La demanda de películas puede variar según la temporada, los días festivos y los eventos especiales.

Prácticas Operativas de CineTech

CineTech opera múltiples complejos cinematográficos y se esfuerza por proporcionar una experiencia de cine de alta calidad para sus clientes. Algunos aspectos relevantes de las operaciones de CineTech son:

Cartelera de Películas: CineTech ofrece una variedad de películas de estreno y clásicas en sus complejos.

Horarios y Salas: Los horarios y la disponibilidad de salas varían según la película, la ubicación y otros factores.

Venta de Entradas: Actualmente, las entradas se venden principalmente en taquillas físicas y quioscos de autoservicio en el lugar.

Necesidades y Expectativas de los Clientes

Los clientes de CineTech valoran la calidad de la experiencia cinematográfica, la conveniencia en el proceso de compra de entradas y la variedad de películas disponibles. Algunas necesidades y expectativas de los clientes son:

Accesibilidad: Los clientes desean poder comprar entradas de manera conveniente desde cualquier lugar y en cualquier momento.

Variedad de Películas: Los clientes valoran la disponibilidad de una amplia variedad de películas para elegir.

Experiencia del Usuario: Los clientes esperan una experiencia de compra de entradas intuitiva y sin problemas.

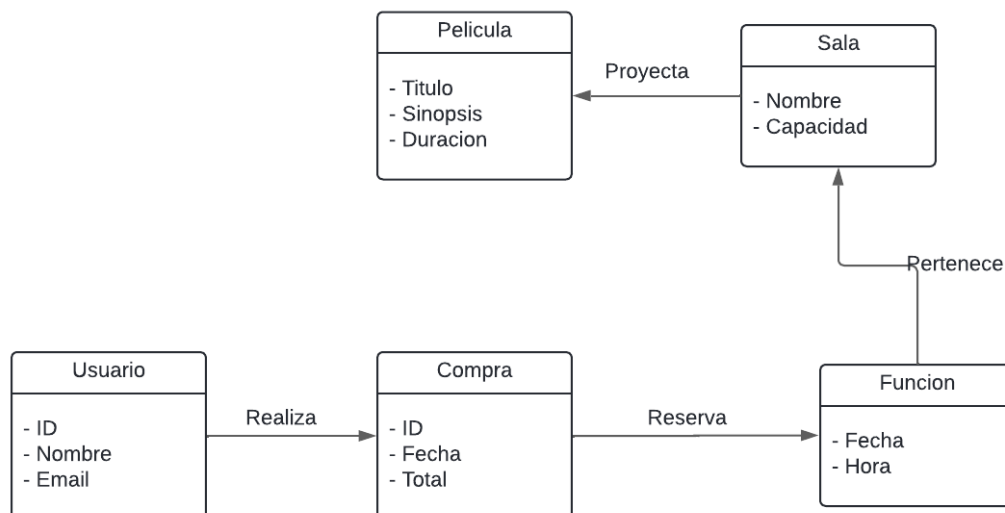
Desafíos y Oportunidades

La implementación de un sistema de venta de entradas en línea presenta varios desafíos y oportunidades para CineTech:

Desafíos: Adaptarse a las tendencias digitales, garantizar la seguridad de las transacciones en línea y mantener la experiencia del cliente como prioridad.

Oportunidades: Ampliar el alcance y la accesibilidad de la venta de entradas, mejorar la eficiencia operativa y diferenciarse de la competencia.

Diagrama de Dominio



Propuesta de Solución

Arquitectura del Sistema

La arquitectura del sistema estará compuesta por tres capas principales: la capa de presentación (frontend), la capa de lógica de negocio (backend) y la capa de almacenamiento de datos (base de datos).

Capa de Presentación (Frontend):

Desarrollada utilizando tecnologías web estándar como HTML, CSS y JavaScript.

Se enfocará en proporcionar una interfaz de usuario intuitiva y responsiva para que los usuarios puedan buscar películas, consultar horarios, seleccionar asientos y realizar compras de entradas.

Se implementará una arquitectura de diseño modular y componetizada para facilitar la escalabilidad y el mantenimiento del frontend.

Capa de Lógica de Negocio (Backend):

Desarrollada utilizando el lenguaje de programación Java y el framework Spring Boot para la creación de APIs RESTful.

Se encargará de gestionar la lógica de negocio, la autenticación de usuarios, la integración con la base de datos y la integración con sistemas externos, como el sistema de pago.

Se implementarán patrones de diseño como MVC (Modelo-Vista-Controlador) para separar las responsabilidades y facilitar la modularidad y la mantenibilidad del código.

Capa de Almacenamiento de Datos (Base de Datos):

Utilizará MySQL como sistema de gestión de bases de datos relacional para almacenar y gestionar la información del sistema.

Se diseñará un esquema de base de datos eficiente y normalizado que incluya tablas para usuarios, películas, funciones, salas, reservas, transacciones y otros datos relevantes del sistema.

Se aplicarán técnicas de optimización de consultas y de índices para garantizar un rendimiento óptimo de la base de datos.

Tecnologías y Herramientas

Frontend:

- HTML, CSS y JavaScript para el desarrollo de la interfaz de usuario.
- Frameworks como Bootstrap o Materialize para el diseño y la maquetación.
- Bibliotecas como React.js o Vue.js para la creación de componentes interactivos y reutilizables.

Backend:

- Java como lenguaje de programación principal.
- Framework Spring Boot para la creación de APIs RESTful.
- Gestión de dependencias con Maven o Gradle.
- Seguridad con Spring Security para la autenticación y autorización de usuarios.

Base de Datos:

- MySQL como sistema de gestión de bases de datos relacional.
- Herramientas de modelado de datos como MySQL Workbench o DBeaver para el diseño del esquema de base de datos.
- Conexión y acceso a la base de datos a través de JDBC (Java Database Connectivity).

Implementación del Sistema

El desarrollo del sistema seguirá un enfoque iterativo e incremental, utilizando metodologías ágiles como Scrum o Kanban para la gestión del proyecto. Se establecerán sprints cortos con entregas incrementales de funcionalidades para obtener retroalimentación temprana de los stakeholders y garantizar la adaptabilidad a los cambios.

Se llevará a cabo un riguroso proceso de pruebas que incluirá pruebas unitarias, pruebas de integración y pruebas de aceptación para garantizar la calidad y la fiabilidad del sistema. Se

utilizarán herramientas de automatización de pruebas como JUnit y Selenium para agilizar el proceso de prueba y detectar posibles fallos de manera temprana.

Despliegue y Mantenimiento

Una vez completada la fase de desarrollo, el sistema se desplegará en un entorno de producción utilizando servicios de alojamiento en la nube como AWS o Azure. Se implementarán prácticas de DevOps para la automatización del despliegue, la monitorización del sistema y la gestión de la infraestructura.

Se establecerá un plan de mantenimiento proactivo que incluirá actualizaciones regulares del sistema, monitoreo continuo del rendimiento y la disponibilidad, y respuesta rápida a posibles problemas o incidencias.

Inicio del Análisis: Casos de Uso

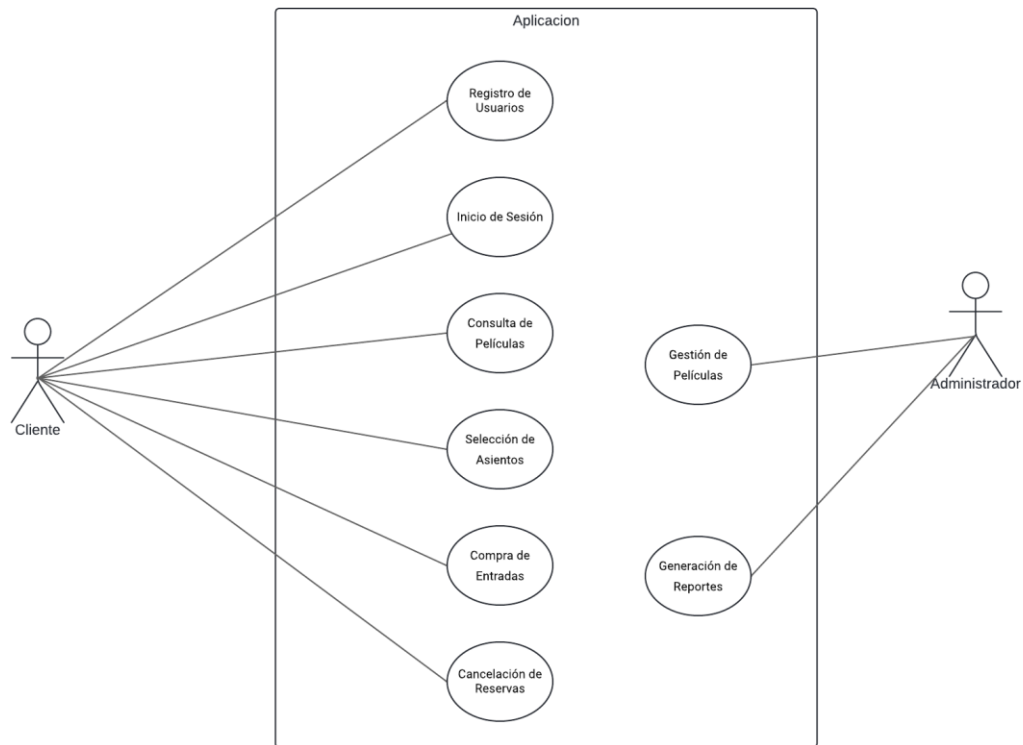
Requerimientos Funcionales

ID	Requerimiento
RF1	Registro de usuarios: Los usuarios deben poder registrarse en la plataforma.
RF2	Inicio de sesión: Los usuarios registrados deben poder iniciar sesión en la plataforma.
RF3	Búsqueda de películas: Los usuarios deben poder buscar películas disponibles en la plataforma.
RF4	Consulta de horarios: Los usuarios deben poder consultar los horarios de las películas.
RF5	Selección de asientos: Los usuarios deben poder seleccionar los asientos deseados.
RF6	Compra de entradas: Los usuarios deben poder comprar las entradas seleccionadas.
RF7	Cancelación de reservas: Los usuarios deben poder cancelar sus reservas activas.
RF8	Gestión de películas: Los administradores deben poder agregar, editar y eliminar películas.
RF9	Generación de reportes: Los administradores deben poder generar reportes de ventas.

Requerimientos No Funcionales

ID	Requerimiento
RNF1	Usabilidad: La interfaz de usuario debe ser intuitiva y fácil de usar.
RNF2	Seguridad: El sistema debe garantizar la seguridad de los datos del usuario y las transacciones financieras.
RNF3	Rendimiento: El sistema debe ser capaz de manejar un alto volumen de usuarios y transacciones simultáneas.
RNF4	Compatibilidad: El sistema debe ser compatible con diferentes dispositivos y navegadores web.
RNF5	Mantenibilidad: El sistema debe ser fácil de mantener y actualizar en el futuro.

Diagrama de Casos de Uso:



Cliente: Usuario final que utiliza la plataforma para buscar películas, consultar horarios, seleccionar asientos y comprar entradas.

Administrador: Personal autorizado de CineTech responsable de gestionar y administrar el sistema, incluyendo la gestión de películas, horarios y ventas.

Trazabilidad:

ID Req.	ID Caso de Uso	Actor	Paquete de Análisis	Comentario
RF1	CU01	Cliente	Gestión de Usuarios	Los usuarios deben poder registrarse en la plataforma.
RF2	CU02	Cliente	Gestión de Usuarios	Los usuarios registrados deben poder iniciar sesión en la plataforma.
RF3	CU03	Cliente	Gestión de Películas	Los usuarios deben poder buscar películas disponibles.

RF4	CU03	Cliente	Gestión de Películas	Los usuarios deben poder consultar los horarios de las películas.
RF5	CU04	Cliente	Gestión de Películas	Los usuarios deben poder seleccionar los asientos deseados.
RF6	CU05	Cliente	Compra de Entradas	Los usuarios deben poder comprar las entradas seleccionadas.
RF7	CU06	Cliente	Compra de Entradas	Los usuarios deben poder cancelar sus reservas activas.
RF8	CU07	Administrador	Gestión de Películas	Los administradores deben poder agregar, editar y eliminar películas.
RF9	CU08	Administrador	Gestión de Películas	Los administradores deben poder generar reportes de ventas.

Descripción de los casos de uso:

Campo	Descripción
Caso de Uso	CU01
Actores	Cliente
Referencias	RF1
Descripción	Este caso de uso describe el proceso que sigue un usuario para registrarse en la plataforma.
Precondición	El usuario no está registrado en la plataforma.

Flujo Principal	1. El usuario accede a la página de registro. 2. El usuario completa el formulario de registro con su información personal. 3. El usuario confirma el registro.
Postcondición	El usuario queda registrado en la plataforma.
Flujo Alternativo	-
Excepciones	-

Campo	Descripción
Caso de Uso	CU02
Actores	Cliente
Referencias	RF2
Descripción	Este caso de uso describe cómo un usuario registrado inicia sesión en la plataforma.
Precondición	El usuario está registrado en la plataforma.
Flujo Principal	1. El usuario accede a la página de inicio de sesión. 2. El usuario ingresa su correo electrónico y contraseña. 3. El sistema valida las credenciales del usuario. 4. El usuario accede a su cuenta.
Postcondición	El usuario inicia sesión correctamente en la plataforma.
Flujo Alternativo	-
Excepciones	-

Campo	Descripción
Caso de Uso	CU03
Actores	Cliente
Referencias	RF3
Descripción	Este caso de uso permite a los usuarios buscar y consultar la cartelera de películas.
Precondición	-
Flujo Principal	1. El usuario accede a la página de consulta de películas. 2. El usuario realiza una búsqueda de películas. 3. El sistema muestra los resultados de la búsqueda.
Postcondición	El usuario visualiza la cartelera de películas.
Flujo Alternativo	-
Excepciones	-

Campo	Descripción
Caso de Uso	CU04
Actores	Cliente
Referencias	RF5

Descripción	Este caso de uso describe cómo un usuario selecciona los asientos deseados para una película específica.
Precondición	El usuario ha consultado la cartelera de películas y ha seleccionado una película.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona una película y un horario específico. 2. El sistema muestra el mapa de asientos disponibles para esa función. 3. El usuario selecciona los asientos deseados. 4. El usuario confirma la selección de asientos.
Postcondición	El sistema registra la selección de asientos del usuario.
Flujo Alternativo	-
Excepciones	-

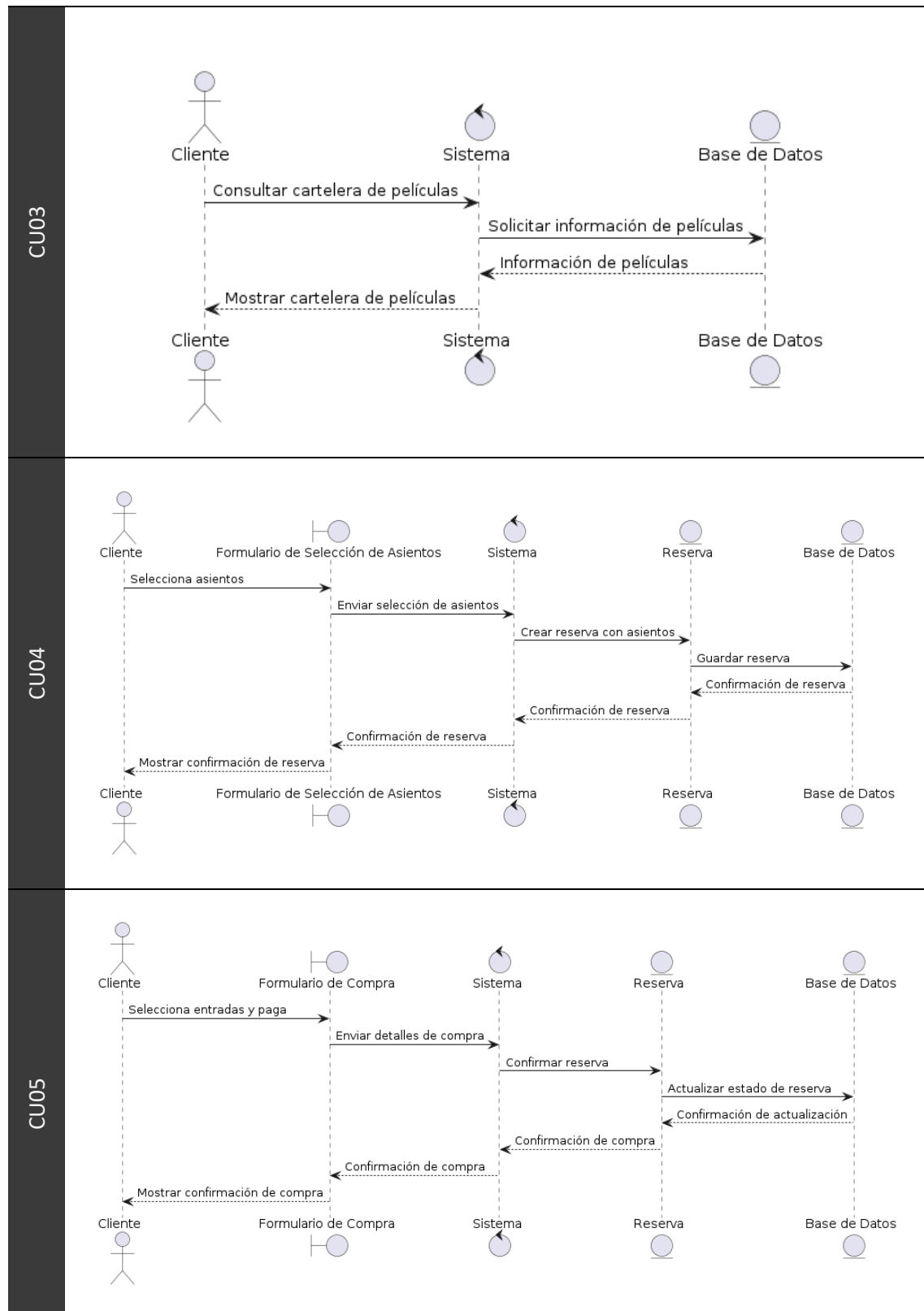
Campo	Descripción
Caso de Uso	CU05
Actores	Cliente
Referencias	RF6
Descripción	Este caso de uso permite a los usuarios comprar entradas para las películas seleccionadas, completando el proceso de pago.
Precondición	El usuario ha seleccionado los asientos deseados para una película y ha confirmado la selección.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona los asientos deseados para una película. 2. El sistema muestra un resumen de la selección de asientos y el precio total. 3. El usuario completa el proceso de pago proporcionando la información de facturación y seleccionando el método de pago. 4. El sistema procesa el pago y genera las entradas. 5. El usuario recibe la confirmación de la compra.
Postcondición	El usuario completa la compra de las entradas y recibe la confirmación.
Flujo Alternativo	-
Excepciones	-

Segundo TP

De los casos vistos en la primera parte utilizare los casos de uso CU03, CU04 y CU05 que considero que representan el funcionamiento principal del objetivo de la aplicación.

Etaapa de análisis.

Diagramas de colaboración



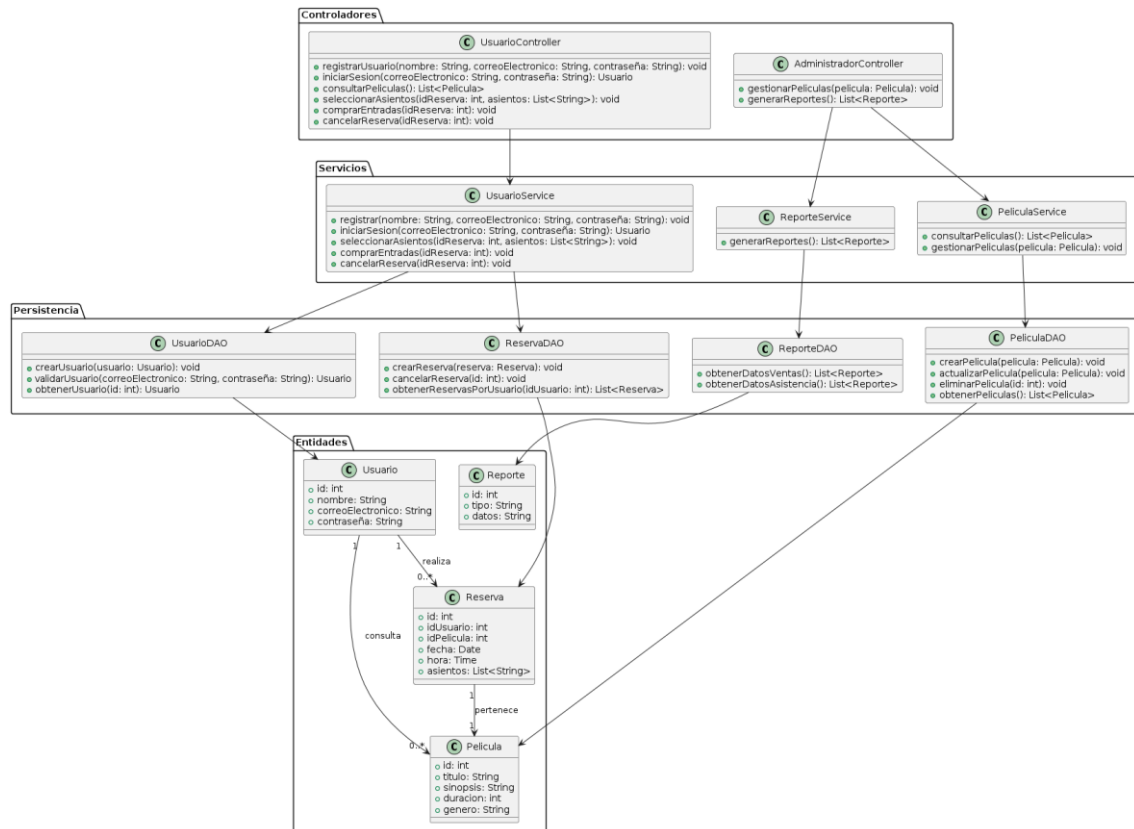
Etapas de diseño

Arquitectura del Sistema

Capas:

- Presentación: Interfaz de usuario (HTML/CSS/JavaScript).
- Lógica de Negocio: Procesamiento de la lógica de negocio (Java).
- Persistencia: Gestión de la base de datos (MySQL).

Diagrama de clases

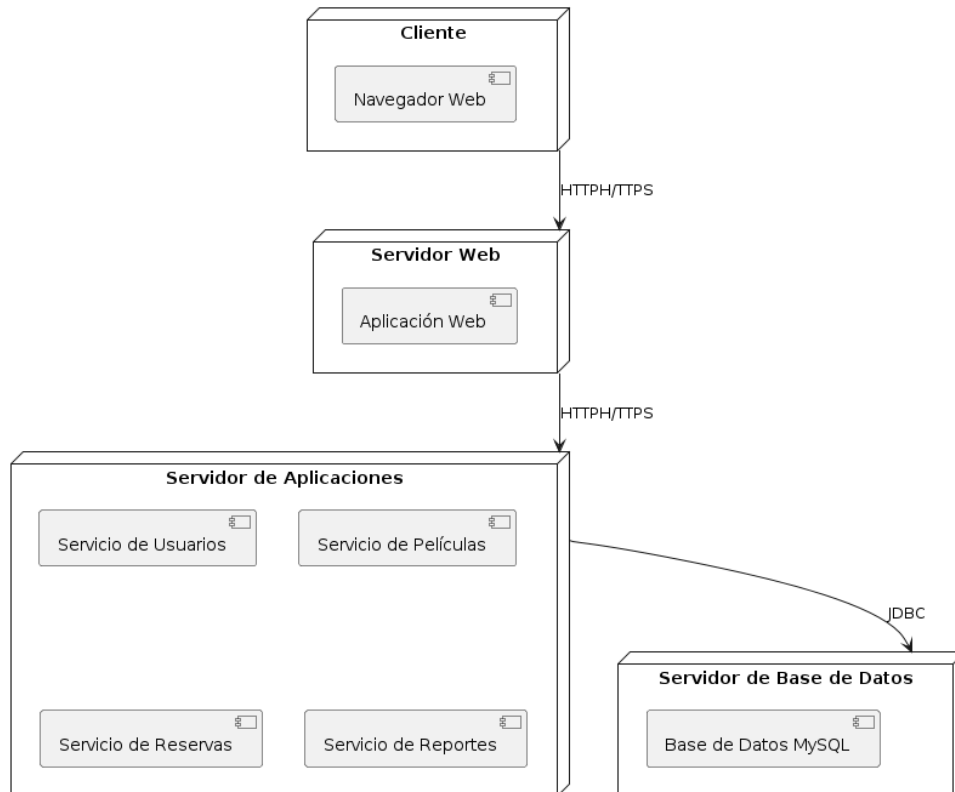


Etapas de implementación.

Tecnologías Utilizadas

- Frontend: HTML, CSS, JavaScript.
- Backend: Java.
- Base de Datos: MySQL.

Diagrama de implementación



Etapas de pruebas.

Plan de pruebas

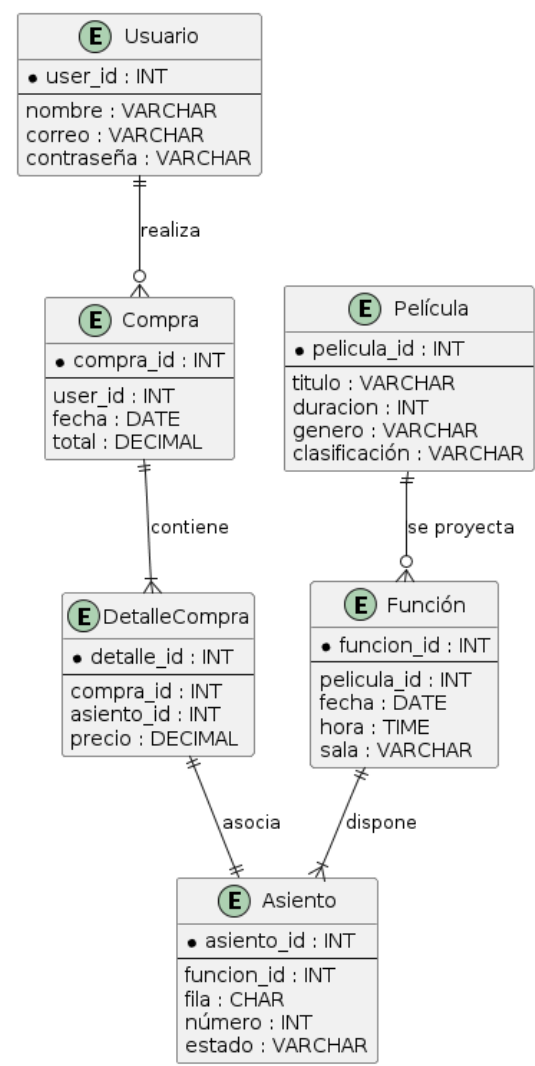
Caso de Uso	Código Prueba	Tipo de Prueba	Técnica Propuesta	Observaciones
CU03	CP003-01	Prueba Unitaria	Caja Negra	Verificar que la lista de películas se obtenga correctamente desde la base de datos
CU03	CP003-02	Prueba de Integración	Caja Negra	Asegurar que la consulta de películas desde la interfaz de usuario se comunice correctamente con el backend
CU03	CP003-03	Prueba de Sistema	Caja Negra	Verificar que se muestren todas las películas disponibles y se manejen adecuadamente los casos de error
CU04	CP004-01	Prueba Unitaria	Caja Negra	Verificar que se puedan seleccionar asientos disponibles correctamente
CU04	CP004-02	Prueba de Integración	Caja Negra	Asegurar que la selección de asientos actualice correctamente el estado de los asientos en la base de datos
CU04	CP004-03	Prueba de Sistema	Caja Negra	Verificar que se impida la selección de asientos ya reservados por otros usuarios
CU05	CP005-01	Prueba Unitaria	Caja Negra	Verificar que el proceso de compra de entradas se complete correctamente
CU05	CP005-02	Prueba de Integración	Caja Negra	Asegurar que la compra de entradas actualice correctamente las reservas en la base de datos
CU05	CP005-03	Prueba de Sistema	Caja Negra	Confirmar que se emitan correctamente los comprobantes de compra y se manejen los pagos de manera segura

Análisis de casos de prueba

Código Prueba	Dato de Entrada	Comportamiento Esperado	Mensaje Emitido por el Sistema
CP003-01	Solicitud de lista de películas	La lista de películas se obtiene correctamente	"Películas disponibles cargadas correctamente"
CP003-02	Solicitud desde la interfaz de usuario	La interfaz se comunica correctamente con el backend	"Películas obtenidas exitosamente"
CP003-03	Error en la consulta de películas	Manejo adecuado del error, sin interrupción del servicio	"Error al cargar las películas, por favor intente nuevamente"
CP004-01	Selección de asientos disponibles	Los asientos seleccionados se marcan como reservados	"Asientos seleccionados correctamente"
CP004-02	Actualización de estado en la base de datos	La base de datos se actualiza correctamente con la selección de asientos	"Estado de asientos actualizado correctamente"
CP004-03	Selección de asientos ya reservados	Impedir la selección y notificar al usuario	"Asiento no disponible, seleccione otro asiento"
CP005-01	Datos de compra (usuario, asientos, pago)	La compra se procesa correctamente	"Compra realizada con éxito"
CP005-02	Actualización de reservas en la base de datos	Las reservas se actualizan correctamente en la base de datos	"Reservas actualizadas correctamente"
CP005-03	Confirmación de compra	Se emite el comprobante de compra	"Comprobante de compra emitido"

Definición de base de datos para el sistema.

Diagrama entidad-relación de la base de datos.



Creación de las tablas MySQL.

```
1 CREATE TABLE Usuario (  
2     user_id INT AUTO_INCREMENT PRIMARY KEY,  
3     nombre VARCHAR(100) NOT NULL,  
4     correo VARCHAR(100) NOT NULL UNIQUE,  
5     contrasenia VARCHAR(100) NOT NULL  
6 );  
7  
8 CREATE TABLE Pelicula (  
9     pelicula_id INT AUTO_INCREMENT PRIMARY KEY,  
10    titulo VARCHAR(100) NOT NULL,  
11    duracion INT NOT NULL,  
12    genero VARCHAR(50) NOT NULL,  
13    clasificacion VARCHAR(20) NOT NULL  
14 );  
15  
16 CREATE TABLE Funcion (  
17     funcion_id INT AUTO_INCREMENT PRIMARY KEY,  
18     pelicula_id INT NOT NULL,  
19     fecha DATE NOT NULL,  
20     hora TIME NOT NULL,  
21     sala VARCHAR(10) NOT NULL,  
22     FOREIGN KEY (pelicula_id) REFERENCES Pelicula(pelicula_id)  
23 );  
  
25 CREATE TABLE Asiento (  
26     asiento_id INT AUTO_INCREMENT PRIMARY KEY,  
27     funcion_id INT NOT NULL,  
28     fila CHAR(1) NOT NULL,  
29     numero INT NOT NULL,  
30     estado VARCHAR(20) NOT NULL,  
31     FOREIGN KEY (funcion_id) REFERENCES Funcion(funcion_id)  
32 );  
33  
34 CREATE TABLE Compra (  
35     compra_id INT AUTO_INCREMENT PRIMARY KEY,  
36     user_id INT NOT NULL,  
37     fecha DATE NOT NULL,  
38     total DECIMAL(10, 2) NOT NULL,  
39     FOREIGN KEY (user_id) REFERENCES Usuario(user_id)  
40 );  
41  
42 CREATE TABLE DetalleCompra (  
43     detalle_id INT AUTO_INCREMENT PRIMARY KEY,  
44     compra_id INT NOT NULL,  
45     asiento_id INT NOT NULL,  
46     precio DECIMAL(10, 2) NOT NULL,  
47     FOREIGN KEY (compra_id) REFERENCES Compra(compra_id),  
48     FOREIGN KEY (asiento_id) REFERENCES Asiento(asiento_id)  
49 );
```

Inserción, consulta y borrado de registros.

Inserción

```
51 -- Inserción de Registros
52 INSERT INTO Usuario (nombre, correo, contraseña) VALUES
53 ('Juan Perez', 'juan.perez@example.com', 'password123'),
54 ('Maria Lopez', 'maria.lopez@example.com', 'securepassword');
55
56 INSERT INTO Pelicula (titulo, duracion, genero, clasificación) VALUES
57 ('Inception', 148, 'Ciencia Ficción', 'PG-13'),
58 ('Toy Story', 81, 'Animación', 'G');
59
60 INSERT INTO Funcion (pelicula_id, fecha, hora, sala) VALUES
61 (1, '2024-06-01', '18:00:00', 'Sala 1'),
62 (2, '2024-06-01', '14:00:00', 'Sala 2');
63
64 INSERT INTO Asiento (funcion_id, fila, número, estado) VALUES
65 (1, 'A', 1, 'disponible'),
66 (1, 'A', 2, 'disponible'),
67 (2, 'B', 1, 'disponible'),
68 (2, 'B', 2, 'disponible');
69
70 INSERT INTO Compra (user_id, fecha, total) VALUES
71 (1, '2024-06-01', 20.00),
72 (2, '2024-06-01', 15.00);
73
```

Consulta

```
79 -- Consultas de Registros
80 SELECT * FROM Usuario;
81
82 SELECT * FROM Pelicula;
83
84 SELECT * FROM Funcion WHERE pelicula_id = 1;
85
86 SELECT * FROM Asiento WHERE funcion_id = 1 AND estado = 'disponible';
87
88 SELECT * FROM Compra WHERE user_id = 1;
89
90 SELECT * FROM DetalleCompra WHERE compra_id = 1;
91
```

Borrado

```
92 -- Borrado de Registros
93 DELETE FROM Usuario WHERE user_id = 1;
94
95 DELETE FROM Pelicula WHERE pelicula_id = 1;
96
97 DELETE FROM Funcion WHERE funcion_id = 1;
98
99 DELETE FROM Asiento WHERE asiento_id = 1;
100
101 DELETE FROM Compra WHERE compra_id = 1;
102
103 DELETE FROM DetalleCompra WHERE detalle_id = 1;
104
```

Definiciones de comunicación.

1. Interacción Cliente-Servidor

- 1) Frontend: HTML, CSS, JavaScript envían solicitudes HTTP al backend.
- 2) Backend: Java recibe las solicitudes, procesa la lógica de negocio y accede a la base de datos MySQL.
- 3) Base de Datos: MySQL almacena los datos y responde a las consultas del backend.

2. Protocolo de Comunicación

- 1) HTTP/HTTPS: Para garantizar la seguridad en la transmisión de datos.
- 2) JSON: Formato para el intercambio de datos entre frontend y backend.

3. Endpoints de la API

- 1) Registro de Usuarios: POST /api/usuarios
- 2) Inicio de Sesión: POST /api/sesion
- 3) Consulta de Películas: GET /api/peliculas
- 4) Selección de Asientos: POST /api/reservas/seleccionar
- 5) Compra de Entradas: POST /api/entradas/comprar
- 6) Cancelación de Reservas: DELETE /api/reservas/{id}
- 7) Gestión de Películas: POST /api/peliculas (agregar), PUT /api/peliculas/{id} (editar), DELETE /api/peliculas/{id} (eliminar)
- 8) Generación de Reportes: GET /api/reportes

Tercer TP

Explicación del Desarrollo en Java

El desarrollo de los casos de uso 4, 5 y 6 en Java se enfoca en la gestión de las compras de entradas de cine, incluyendo la selección de funciones, la reserva de asientos y la confirmación de compras. Usaremos Maven para gestionar las dependencias y el proyecto.

Presentación del Desarrollo en Java

1. Correcta utilización de sintaxis, tipos de datos, estructuras de control: Usaremos clases, interfaces, listas, bucles y condicionales para implementar la lógica del sistema.
2. Tratamiento y manejo de excepciones: Implementaremos manejo de excepciones para errores comunes como entradas inválidas o problemas de base de datos.
3. Encapsulamiento, herencia, polimorfismo y abstracción: Definiremos clases y métodos siguiendo principios de POO.
4. Menú de selección: Implementaremos un menú interactivo para el usuario.
5. Estructuras condicionales y repetitivas: Usaremos if-else y bucles for y while.
6. Declaración y creación de objetos en Java: Creamos objetos para representar películas, funciones, asientos y compras.
7. Constructores para inicializar objetos: Usaremos constructores parametrizados para inicializar objetos.

8. Uso de algoritmos de ordenación y búsqueda: Implementaremos búsqueda de funciones y asientos.

Estructura del Proyecto Maven

src/main/java

com.mycompany.spi.proyectocine: Clases que representan el modelo y la lógica del negocio.

src/main/resources: Archivos de configuración y recursos del proyecto.

pom.xml: Archivo de configuración de Maven.

Explicación del Código

Modelo de Negocio: Clases Usuario, Pelicula, Funcion, Asiento, Compra, y DetalleCompra representan los elementos del sistema.

Usuario

```
8      *
9      * @author IA-JED
10     */
11     public class Usuario {
12         private int id;
13         private String nombre;
14         private String correo;
15         private String contrasena;
16
17         public Usuario(int id, String nombre, String correo, String contrasena) {
18             this.id = id;
19             this.nombre = nombre;
20             this.correo = correo;
21             this.contrasena = contrasena;
22         }
23
24         // Getters and Setters
25     }
```

Pelicula

```
5     package com.mycompany.spi.proyectocine;
6
7     /**
8      *
9      * @author IA-JED
10     */
11     public class Pelicula {
12         private int id;
13         private String titulo;
14         private int duracion;
15         private String genero;
16         private String clasificacion;
17
18         public Pelicula(int id, String titulo, int duracion, String genero, String clasificacion) {
19             this.id = id;
20             this.titulo = titulo;
21             this.duracion = duracion;
22             this.genero = genero;
23             this.clasificacion = clasificacion;
24         }
25
26         // Getters and Setters
27     }
```

Funcion

```
5 package com.mycompany.spi.proyectocine;
6
7 /**
8  *
9  * @author IA-JED
10 */
11 import java.time.LocalDate;
12 import java.time.LocalTime;
13
14 public class Funcion {
15     private int id;
16     private Pelicula pelicula;
17     private LocalDate fecha;
18     private LocalTime hora;
19     private String sala;
20
21     public Funcion(int id, Pelicula pelicula, LocalDate fecha, LocalTime hora, String sala) {
22         this.id = id;
23         this.pelicula = pelicula;
24         this.fecha = fecha;
25         this.hora = hora;
26         this.sala = sala;
27     }
28
29     // Getters and Setters
```

Asiento

```
5 package com.mycompany.spi.proyectocine;
6
7 /**
8  *
9  * @author IA-JED
10 */
11 public class Asiento {
12     private int id;
13     private Funcion funcion;
14     private char fila;
15     private int numero;
16     private String estado;
17
18     public Asiento(int id, Funcion funcion, char fila, int numero, String estado) {
19         this.id = id;
20         this.funcion = funcion;
21         this.fila = fila;
22         this.numero = numero;
23         this.estado = estado;
24     }
25
26     // Getters and Setters
```


Compra

```
5 package com.mycompany.spi.proyectocine;
6
7 /**
8  *
9  * @author IA-JED
10 */
11 import java.time.LocalDate;
12 import java.util.List;
13
14 public class Compra {
15     private int id;
16     private Usuario usuario;
17     private LocalDate fecha;
18     private double total;
19     private List<DetalleCompra> detalles;
20
21     public Compra(int id, Usuario usuario, LocalDate fecha, double total, List<DetalleCompra> detalles) {
22         this.id = id;
23         this.usuario = usuario;
24         this.fecha = fecha;
25         this.total = total;
26         this.detalles = detalles;
27     }
28
29     // Getters and Setters
```

DetalleCompra

```
5 package com.mycompany.spi.proyectocine;
6
7 /**
8  *
9  * @author IA-JED
10 */
11 public class DetalleCompra {
12     private int id;
13     private Compra compra;
14     private Asiento asiento;
15     private double precio;
16
17     public DetalleCompra(int id, Compra compra, Asiento asiento, double precio) {
18         this.id = id;
19         this.compra = compra;
20         this.asiento = asiento;
21         this.precio = precio;
22     }
23
24     // Getters and Setters
```

Lógica del Negocio: Clase CinemaService contiene métodos para buscar funciones y asientos, reservar asientos y confirmar compras. En esta clase también inicializo algunas Funciones, Asientos y Películas para realizar la validación y prueba del código.

```

61 public List<Funcion> buscarFuncionesPorPelicula(int peliculaId) {
62     List<Funcion> resultado = new ArrayList<>();
63     for (Funcion funcion : funciones) {
64         if (funcion.getPelicula().getId() == peliculaId) {
65             resultado.add(funcion);
66         }
67     }
68     return resultado;
69 }
70
71 public List<Asiento> buscarAsientosDisponibles(int funcionId) {
72     List<Asiento> resultado = new ArrayList<>();
73     for (Asiento asiento : asientos) {
74         if (asiento.getFuncion().getId() == funcionId && asiento.getEstado().equals("disponible")) {
75             resultado.add(asiento);
76         }
77     }
78     return resultado;
79 }
80
81 public boolean reservarAsiento(int asientoId, Usuario usuario) {
82     for (Asiento asiento : asientos) {
83         if (asiento.getId() == asientoId && asiento.getEstado().equals("disponible")) {
84             asiento.setEstado("reservado");
85             return true;
86         }
87     }
88     return false;
89 }
90
91 public Compra confirmarCompra(Usuario usuario, List<Asiento> asientosSeleccionados) {
92     double total = 0;
93     for (Asiento asiento : asientosSeleccionados) {
94         total += 10.00; // Precio fijo por asiento para simplificación
95     }
96     return new Compra(usuario, asientosSeleccionados, total);
97 }

```

Interacción con el Usuario: Clase SpiProyectocine que contiene la clase Main implementa un menú interactivo que permite al usuario realizar acciones como ver funciones, ver asientos disponibles, reservar asientos y confirmar compras.

```

15 public class SpiProyectocine {
16
17     public static void main(String[] args) {
18         CinemaService cinemaService = new CinemaService();
19         Scanner scanner = new Scanner(System.in);
20         Usuario usuario = new Usuario(1, "Juan Perez", "juan.perez@example.com", "password123");
21
22         while (true) {
23             System.out.println("Menú:");
24             System.out.println("1. Ver funciones por pelicula");
25             System.out.println("2. Ver asientos disponibles");
26             System.out.println("3. Reservar asiento");
27             System.out.println("4. Confirmar compra");
28             System.out.println("5. Salir");
29             System.out.print("Seleccione una opción: ");
30             int opcion = scanner.nextInt();
31
32             switch (opcion) {
33                 case 1:
34                     System.out.print("Ingrese el ID de la pelicula: ");
35                     int peliculaId = scanner.nextInt();
36                     List<Funcion> funciones = cinemaService.buscarFuncionesPorPelicula(peliculaId);
37                     for (Funcion funcion : funciones) {
38                         System.out.println("Función ID: " + funcion.getId() + ", Fecha: " + funcion.getFecha() + ", Hora: " + funcion.getHora());
39                     }
40                     break;
41                 case 2:
42                     System.out.print("Ingrese el ID de la función: ");
43                     int funcionId = scanner.nextInt();
44                     List<Asiento> asientos = cinemaService.buscarAsientosDisponibles(funcionId);
45                     for (Asiento asiento : asientos) {
46                         System.out.println("Asiento ID: " + asiento.getId() + ", Fila: " + asiento.getFila() + ", Número: " + asiento.getNumero());
47                     }
48                     break;
49             }
50         }
51     }
52 }

```

Repositorio

Todo el código se encuentra subido en <https://github.com/Jontuc/proyectocine>

Cuarto TP

1. Selección del patrón de diseño y justificación

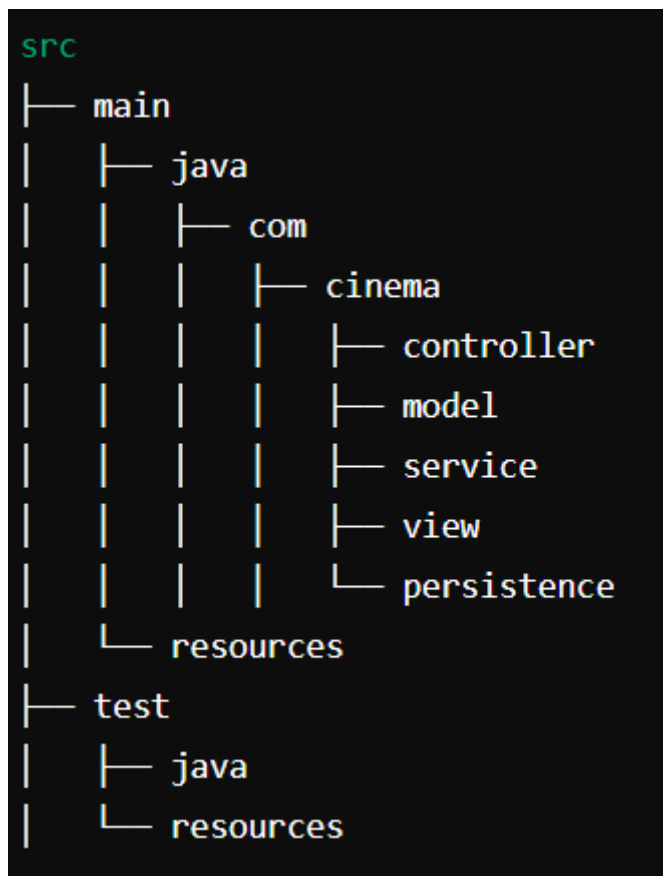
Patrón MVC (Model-View-Controller)

Justificación: El patrón MVC es adecuado para este proyecto porque permite separar la lógica de negocio (Modelo), la interfaz de usuario (Vista) y el control del flujo de la aplicación (Controlador). Esto facilita la mantenibilidad, escalabilidad y organización del código. Además, al usar MVC, podemos cambiar o actualizar una parte de la aplicación sin afectar las otras partes. Por ejemplo, podemos cambiar la base de datos o la interfaz de usuario sin tener que reescribir la lógica de negocio.

2. Persistencia y consulta de datos en una base de datos MySQL

Para esta parte, crearemos un paquete que manejará la persistencia en la base de datos. Utilizaremos JDBC para conectarnos a la base de datos MySQL.

Estructura del proyecto



Conexión a MySQL

Primero, vamos a crear una clase `DatabaseConnection` en el paquete `persistence` para gestionar la conexión con la base de datos.

```

package com.mycompany.spi.proyectocine.persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

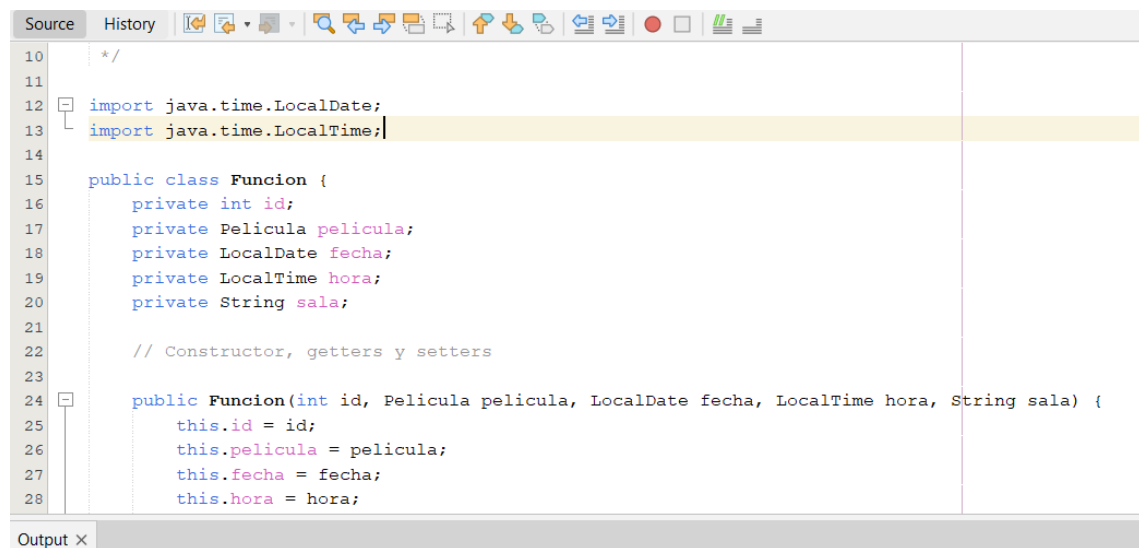
/**
 *
 * @author IA-JED
 */
public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/cinema";
    private static final String USER = "root";
    private static final String PASSWORD = "Jon@19870402";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}

```

Modelo

Función



```

Source History
10 */
11
12 import java.time.LocalDate;
13 import java.time.LocalTime;
14
15 public class Funcion {
16     private int id;
17     private Pelicula pelicula;
18     private LocalDate fecha;
19     private LocalTime hora;
20     private String sala;
21
22     // Constructor, getters y setters
23
24     public Funcion(int id, Pelicula pelicula, LocalDate fecha, LocalTime hora, String sala) {
25         this.id = id;
26         this.pelicula = pelicula;
27         this.fecha = fecha;
28         this.hora = hora;
29
30     }
31 }

```

Output ×

Pelicula

```

4  /**
5  package com.mycompany.spi.projectocine.model;
6
7  /**
8  *
9  * @author IA-JED
10 /**
11 public class Pelicula {
12     private int id;
13     private String titulo;
14     private String director;
15     private int duracion; // en minutos
16
17     // Constructor, getters y setters
18
19     public Pelicula(int id, String titulo, String director, int duracion) {
20         this.id = id;
21         this.titulo = titulo;
22         this.director = director;

```

Asiento

```

4  /**
5  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
6  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
7  */
8  package com.mycompany.spi.projectocine.model;
9
10 /**
11 *
12 * @author IA-JED
13 */
14 public class Asiento {
15     private int id;
16     private Funcion funcion;
17     private char fila;
18     private int numero;
19     private String estado;
20
21     public Asiento(int id, Funcion funcion, char fila, int numero, String estado) {
22         this.id = id;

```

Controlador

```

1  /**
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package com.mycompany.spi.projectocine.controller;
6
7
8
9  import com.mycompany.spi.projectocine.model.Asiento;
10 import com.mycompany.spi.projectocine.model.Funcion;
11 import com.mycompany.spi.projectocine.model.Pelicula;
12 import com.mycompany.spi.projectocine.service.CinemaService;
13 import java.util.List;
14
15 public class CinemaController {
16     private CinemaService cinemaService = new CinemaService();
17
18     public List<Pelicula> getPeliculas() {

```

Servicio

```
6
7
8  import com.mycompany.spi.proyectocine.model.*;
9  import com.mycompany.spi.proyectocine.persistence.PeliculaDAO;
10 import com.mycompany.spi.proyectocine.persistence.FuncionDAO;
11
12
13
14 import java.time.LocalDate;
15
16 import java.util.ArrayList;
17 import java.util.List;
18
19 public class CinemaService {
20     private PeliculaDAO peliculaDAO = new PeliculaDAO();
21     private FuncionDAO funcionDAO = new FuncionDAO();
22
23     private List<Funcion> funciones = new ArrayList<>();
24     private List<Asiento> asientos = new ArrayList<>();
```

Output X

Objetos de Acceso a datos

```
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class PeliculaDAO implements DAO<Pelicula> {
12
13     @Override
14     public List<Pelicula> getAll() {
15         List<Pelicula> peliculas = new ArrayList<>();
16         String query = "SELECT * FROM peliculas";
17         try (Connection connection = DatabaseConnection.getConnection();
18             PreparedStatement statement = connection.prepareStatement(query);
19             ResultSet resultSet = statement.executeQuery()) {
20
21             while (resultSet.next()) {
```

put X

```

8      import com.mycompany.spi.proyectocine.model.Pelicula;
9
10
11      import java.sql.Connection;
12      import java.sql.PreparedStatement;
13      import java.sql.ResultSet;
14      import java.sql.SQLException;
15      import java.util.ArrayList;
16      import java.util.List;
17
18      public class FuncionDAO implements DAO<Funcion> {
19
20          private PeliculaDAO peliculaDAO = new PeliculaDAO();
21
22          @Override
23          public List<Funcion> getAll() {
24              List<Funcion> funciones = new ArrayList<>();
25              String query = "SELECT * FROM funciones";

```

Output X