Sistema de Gestión de Entradas de Cine: Diseño y Desarrollo de una Plataforma Web

Milene Pacheco Esquinarila [1], Nikole Valery Salas Idme[2]

I. Introduccion

En la era digital, la industria del cine ha evolucionado, demandando sistemas eficientes para la gestión de entradas y la interacción con los usuarios. Este informe presenta el diseño y desarrollo de una plataforma web que optimiza la experiencia del usuario al permitir la búsqueda de películas, selección de asientos y compra de entradas de manera intuitiva. Con un enfoque centrado en el usuario, el sistema no solo simplifica el proceso de compra, sino que también ofrece a los administradores herramientas robustas para gestionar funciones y horarios. A lo largo del informe, se detallan la arquitectura del software, las tecnologías utilizadas y los métodos implementados para asegurar la calidad del producto final, así como los resultados obtenidos y las perspectivas futuras para su mejora y expansión.

II. Marco Teórico

A. Arquitectura de Aplicaciones Web

La arquitectura de aplicaciones web es fundamental para el desarrollo de sistemas modernos, donde se busca la separación entre la presentación, la lógica de negocio y el acceso a datos. Esta estructura permite la escalabilidad y el mantenimiento efectivo de las aplicaciones, facilitando la interacción entre el cliente y el servidor mediante tecnologías como HTML, CSS y JavaScript [1].

B. Servicios RESTful

Los servicios RESTful utilizan el protocolo HTTP para la comunicación entre sistemas, permitiendo una interacción eficiente mediante recursos identificados por URLs. Este enfoque se ha vuelto estándar en el desarrollo de APIs, ofreciendo una forma simple y uniforme de acceder a los recursos [2].

C. Angular (Front-end)

Angular es un framework que permite crear aplicaciones de una sola página (SPA) de manera eficiente. Utiliza TypeScript y ofrece herramientas para crear interfaces de usuario reactivas, facilitando el desarrollo de aplicaciones complejas que interactúan con servicios RESTful [3].

D. DJango (Back-end)

Django es un framework de desarrollo web en Python que promueve el diseño rápido y limpio de aplicaciones. Su arquitectura basada en el patrón Modelo-Vista-Controlador (MVC) permite organizar el código de manera efectiva y facilita la creación de APIs RESTful a través de Django REST Framework, proporcionando una forma estructurada y robusta de gestionar datos y autenticar usuarios [5]. Además, Django incluye un sistema de administración que facilita la gestión de los modelos de datos, lo que resulta en un desarrollo más eficiente [7].

E. Pruebas de Software:

Las pruebas son esenciales para garantizar el funcionamiento correcto de las aplicaciones y cumplir con los requisitos del usuario. La implementación de pruebas automatizadas ayuda a mantener la estabilidad del sistema y a detectar errores en las etapas tempranas del desarrollo [6].

F. Diseño de Interfaz de Usuario

Un diseño efectivo de la interfaz de usuario es crucial para la experiencia del usuario en aplicaciones web. Debe ser intuitivo y accesible, permitiendo a los usuarios realizar tareas de manera eficiente [8].

III. Metodología

Para el desarrollo de la plataforma web, se utilizó un enfoque ágil que facilitó la adaptación a los requerimientos cambiantes y mejoró la colaboración entre los miembros del equipo. El proyecto se dividió en varias fases: análisis de requisitos, diseño del sistema, implementación y pruebas.

A. Análisis de requisitos:

Se llevó a cabo una investigación mediante entrevistas y encuestas a usuarios y algunos desarrolladores para identificar sus necesidades.

1) **Registro y Autenticación de Usuarios:** Los usuarios deben poder registrarse en la plataforma, iniciar sesión y gestionar su perfil personal, incluyendo la modificación de datos como nombre, dirección y número de teléfono.

2) **Visualización de Películas:** La plataforma debe permitir a los usuarios navegar por una lista de películas disponibles, con detalles como título, sinopsis, duración, clasificación y cartelera.

3) **Selección de Asientos:** Los usuarios deben poder seleccionar asientos para las proyecciones de películas en función de la disponibilidad.

4) **Proceso de Compra:** La aplicación debe facilitar un proceso de compra de entradas fácil e intuitivo, incluyendo la opción de realizar pagos seguros.

5) **Historial de Compras:** Los usuarios deben poder acceder a su historial de compras, donde se mostrarán las entradas adquiridas, así como la información de las funciones.

6) **Usabilidad:** La plataforma debe ser intuitiva y fácil de usar, con una interfaz amigable que guíe al usuario en cada paso del proceso.

7) **Seguridad:** Se debe implementar una autenticación segura para proteger la información personal y financiera de los usuarios.

B. Diseño del sistema

1) Arquitectura General

El sistema se basa en una arquitectura cliente-servidor. El frontend se desarrolla utilizando Angular, mientras que el backend está implementado con Django y Django REST Framework. Esta separación de capas permite una distribución eficiente de responsabilidades y facilita la escalabilidad y el mantenimiento del sistema.

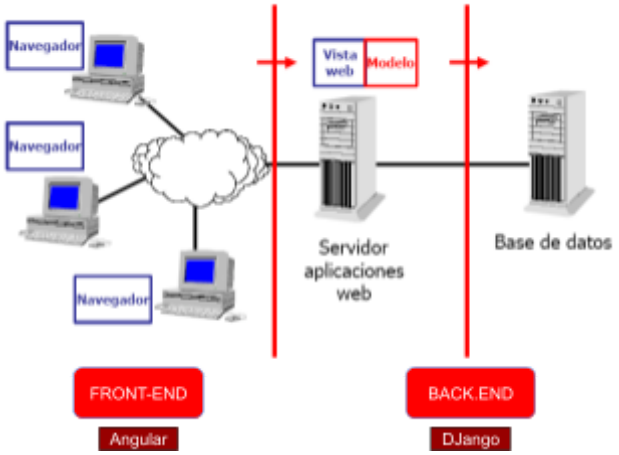

Figura 1. Arquitectura del Sistema de Gestión de Cine.

2) Frontend

a) Interfaz de Usuario: *La interfaz de usuario está diseñada para ser intuitiva y fácil de navegar. Los usuarios pueden acceder a diversas secciones, incluyendo:*

- ***Películas:*** *Sección para visualizar las películas disponibles y sus detalles.*
- ***Selección de Asientos:*** *Herramienta interactiva para elegir asientos disponibles en la sala.*
- ***Compras:*** *Proceso de compra de entradas y gestión de tickets.*

La interfaz incluye un sistema de autenticación que permite a los usuarios registrarse, iniciar sesión y gestionar su perfil de manera sencilla.

b) Experiencia de Usuario: *El diseño se centra en la experiencia del usuario, garantizando que la navegación sea fluida y que todas las funcionalidades sean fácilmente accesibles.*

3) Backend:

c) Lógica de Negocio: *El backend se encarga de la lógica de negocio y de la interacción con la base de datos. Las características clave incluyen:*

- ***Modelos de Datos:*** *Estructura en múltiples tablas para gestionar usuarios, perfiles, películas, horarios, salas, asientos, tickets y compras.*
- ***API RESTful:*** *Proporciona comunicación entre el frontend y el backend a través de endpoints que devuelven datos en formato JSON.*

d) Seguridad: *Se implementan tecnologías de seguridad para proteger el sistema, incluyendo:*

- ***Tokens JWT:*** *Un método seguro de autenticación que protege las rutas y asegura que solo los usuarios autenticados tengan acceso a funciones críticas del sistema.*

e) Base de Datos: *El diseño de la base de datos es relacional, garantizando la integridad de los datos y la posibilidad de realizar consultas eficientes. Esta arquitectura permite la escalabilidad y adaptación del sistema a futuras necesidades, asegurando un rendimiento óptimo en la gestión de la información.*

C. Implementación

La implementación del sistema de gestión de entradas de cine se llevó a cabo en varias etapas, cada una enfocada en garantizar la funcionalidad y el rendimiento óptimo del sistema. Esta sección detalla el proceso de desarrollo y las tecnologías utilizadas en cada capa.

1) Herramienta de Desarrollo VS Code:

Visual Studio Code (VS Code) ha sido una herramienta esencial en el desarrollo del proyecto de Gestión de Entradas de Cine. Su flexibilidad y amplia gama de extensiones ha permitido trabajar de manera eficiente tanto en el frontend como en el backend del sistema.

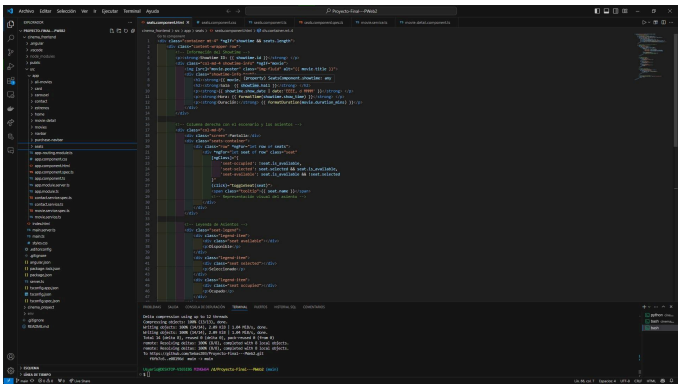

Figura 2. Entorno de Trabajo VS Code

2) **Desarrollo del Front-end:** La implementación del frontend se realizó utilizando Angular, un framework popular para el desarrollo de aplicaciones web. Los principales pasos incluidos fueron:

- **Configuración del Proyecto:** Se creó un nuevo proyecto de Angular utilizando el CLI de Angular, que permitió la generación de la estructura básica del proyecto.
- **Creación de Componentes:** Se desarrollaron componentes clave, como HomeComponent, MovieDetailComponent, SeatsComponent, TicketsComponent, PurchaseComponent, EstrenosComponent y PerfilComponent, cada uno responsable de manejar una parte específica de la interfaz de usuario.
- **Servicios de Comunicación:** Se implementaron servicios como UserService y MovieService para

gestionar la comunicación con el backend y la manipulación de datos en la aplicación.

- **Ruteo:** Se configuró el sistema de ruteo para permitir la navegación entre diferentes vistas de la aplicación, facilitando una experiencia de usuario fluida.
- **Estilo y Diseño:** Se aplicaron estilos CSS para asegurar que la interfaz sea atractiva y funcional, utilizando principios de diseño responsivo para asegurar la compatibilidad en dispositivos móviles y de escritorio.

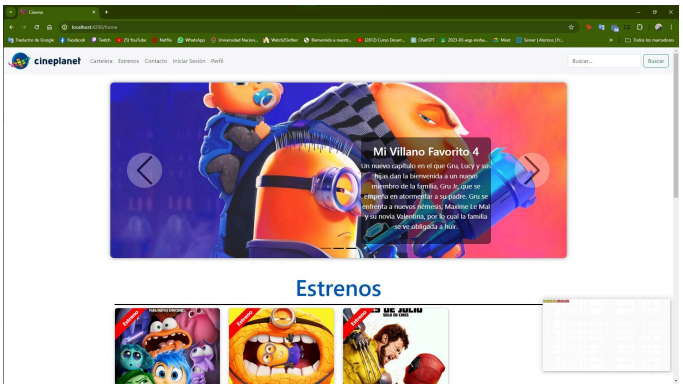

Figura 3. Pantalla de Inicio de la aplicación web

3) **Desarrollo del Back-end:** El backend se implementó utilizando Django y Django REST Framework, lo que permitió construir una API RESTful robusta. Los pasos involucrados fueron:

- **Configuración del Entorno:** Se creó un entorno virtual para gestionar las dependencias de Python y se instaló Django y Django REST Framework.
- **Modelado de Datos:** Se definieron modelos de datos en Django para representar las entidades principales del sistema, incluyendo UserProfile, Movie, Showtime, Halls, Seats, Tickets y Purchase. Esto garantizó una estructura clara y organizada para el manejo de la información.
- **Creación de Serializers:** Se implementaron serializers para transformar los datos de los modelos en formatos JSON que puedan ser fácilmente consumidos por el frontend.
- **Endpoints de la API:** Se configuraron los endpoints de la API, permitiendo operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los recursos del sistema.

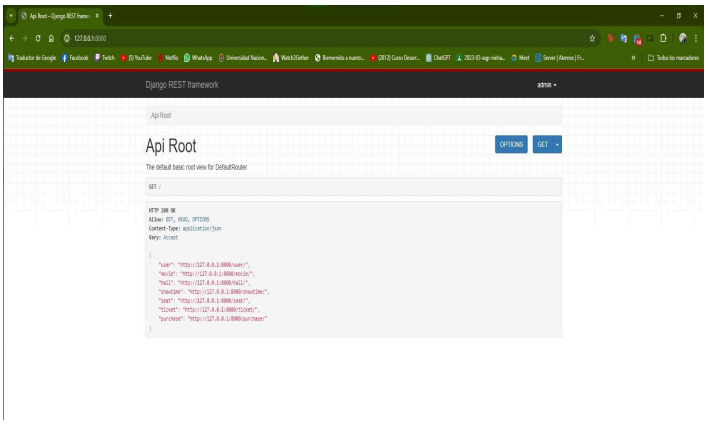

Figura 4. Pantalla de inicio de la API DJango Rest Framework

- **Seguridad:** Se implementaron medidas de seguridad, incluyendo autenticación mediante tokens JWT para proteger los endpoints críticos y garantizar que solo los usuarios autorizados tengan acceso a las funcionalidades del sistema.

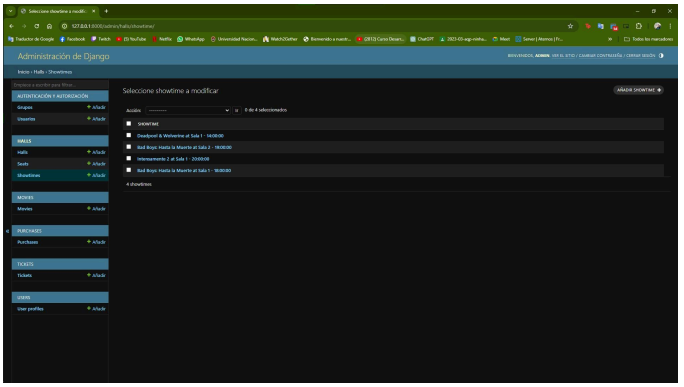

Figura 4. Pantalla de administrador de DJango

4) **Integración y Pruebas:** La integración del sistema se llevó a cabo mediante pruebas funcionales que aseguraron la correcta interacción entre el frontend y el backend. Las pruebas incluyeron:

- **Pruebas Unitarias:** Se desarrollaron pruebas unitarias para componentes individuales y servicios en el frontend, así como para los serializers y vistas en el backend.
- **Pruebas de Integración:** Se llevaron a cabo pruebas de integración para validar la comunicación entre el frontend y el backend, asegurando que las solicitudes y respuestas se manejaban correctamente.
- **Pruebas de Usuario:** Se realizaron pruebas con usuarios finales para obtener retroalimentación sobre la experiencia de usuario y realizar mejoras antes del lanzamiento del sistema.

IV. Resultados y Discusión

A. Pruebas de Funcionalidad:

Se realizaron pruebas exhaustivas en todas las funcionalidades del sistema, asegurando que cada componente y servicio funcionara según lo esperado. Los resultados demostraron que:

- **Autenticación de Usuarios:** La autenticación utilizando tokens JWT funcionó de manera eficiente, permitiendo a los usuarios iniciar sesión y mantener su sesión activa.
- **Gestión de Películas y Funciones:** Los componentes para la visualización de películas y funciones mostraron correctamente los datos recibidos desde el backend, permitiendo a los usuarios seleccionar películas y funciones sin problemas.
- **Selección de Asientos:** El componente de selección de asientos mostró un rendimiento óptimo, permitiendo a los usuarios seleccionar y reservar asientos de manera intuitiva.
- **Gestión de Compras y Tickets:** La gestión de compras y generación de tickets funcionó sin errores, permitiendo a los usuarios completar sus transacciones y visualizar sus tickets correctamente.

B. Feedback de los Usuarios:

Se realizaron pruebas de usuario con un grupo de 15 participantes, quienes proporcionaron feedback sobre la usabilidad y funcionalidad del sistema. Los comentarios principales fueron:

- **Interfaz de Usuario:** La mayoría de los usuarios encontraron la interfaz de usuario intuitiva y fácil de usar. Se destacan positivamente los componentes de selección de asientos, visualización de las películas ofrecidas y la gestión de perfil.
- **Flujo de Compra:** Los usuarios encontraron el proceso de compra de tickets claro y sin complicaciones, desde la selección de la película hasta la confirmación de la compra.
- **Perfil de Usuario:** La funcionalidad para actualizar el perfil del usuario fue bien recibida, permitiendo a los usuarios mantener su información actualizada fácilmente.

C. Discusión

La implementación del sistema de gestión de Cineplanet utilizando Angular y Django ha demostrado ser efectiva y eficiente. Las pruebas de funcionalidad y rendimiento han mostrado resultados positivos, confirmando que el sistema es capaz de proporcionar una experiencia de usuario satisfactoria. Los feedback de los usuarios han sido mayoritariamente positivos, destacando la usabilidad de la interfaz y la claridad del flujo de compra. A pesar de los logros alcanzados, se han identificado áreas de mejora que serán abordadas en futuras iteraciones para garantizar la escalabilidad y optimización del sistema.

Sin embargo, se identificaron algunas áreas de mejora:

- **Optimización de Consultas:** Algunas consultas a la base de datos pueden ser optimizadas para mejorar aún más el tiempo de respuesta.
- **Escalabilidad del Backend:** Aunque el sistema manejó bien las pruebas de carga iniciales, se recomienda realizar optimizaciones adicionales para asegurar la escalabilidad a largo plazo, especialmente si se anticipa un aumento significativo en el número de usuarios.
- **Envío de emails:** Se podrá implementar la función de envío de emails con datos de la compra realizada por el usuario.
- **Servicios administrativos en el frontend:** Una mejora segura es la implementación del servicio de administrador en la plataforma web sin necesidad de acceder al backend o a la base de datos directamente, simplemente con una autenticación de administrador en la misma página.
- **Compras validadas con datos reales:** Este proyecto es una base para poder desarrollar una aplicación web que pueda usarse para una gran empresa en esta industria como lo es Cineplanet por lo hay la opción de validar los pagos reales mediante los datos enviados al momento del pago de los boletos.

V. Conclusiones

La implementación del sistema de gestión de Cineplanet utilizando Angular y Django ha demostrado ser efectiva y eficiente. Las pruebas de funcionalidad y rendimiento han mostrado resultados positivos, confirmando que el sistema es capaz de proporcionar una experiencia de usuario satisfactoria. Los feedback de los usuarios han sido mayoritariamente positivos, destacando la usabilidad de la interfaz y la claridad del flujo de compra. A pesar de los logros alcanzados, se han identificado áreas de mejora que serán abordadas en futuras iteraciones para garantizar la escalabilidad y optimización del sistema.

Referenciass

[1] W3C. (2021). Architectural Principles of the Web. Available: https://www.w3.org/TR/webarch/

[2] Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

[3] Manning, S., & Wurst, S. (2018). Angular 7: The Complete Guide. Packt Publishing.

[4] JSON Web Tokens. (n.d.). Available: https://jwt.io/

[5] Django Software Foundation. (2020). Django Documentation. Available: https://docs.djangoproject.com/

[6] Cohen, J. (2019). The Importance of Software Testing. Software Testing Help.

[7] Turner, B. (2019). Django for Beginners: Build websites with Python and Django. CreateSpace Independent Publishing Platform.

[8] Shneiderman, B., & Plaisant, C. (2010). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson

[9] Microsoft, "Visual Studio Code," Visual Studio Code Documentation, [Online]. Available: https://code.visualstudio.com/docs. [Accessed: 27-Jul-2024].