TRABAJOS FIN DE MÁSTER

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Un Sistema de Recomendación Integrado en un mundo Virtual 3D con Gafas Meta Quest 3

Autor: Óscar Ortega González

Máster: Ingeniería en Informática

Directores: Luis Martínez López y Álvaro Labella Romero
Departamento del director: Informática

Fecha: 8/1/2026

Agradecimientos

Un agradecimiento a los tutores, Álvaro Labella Romero y Luis Martínez López, ya que sin su ayuda este proyecto no hubiese sido posible. Su apoyo, guía y correcciones de la presente memoria han sido vitales.

Otro agradecimiento especial al profesor Rafael Segura Sánchez, ya que sin él nunca hubiese conocido a Luis Martínez.

Finalmente, un agradecimiento especial a un compañero, Álvaro Garrido. Entre sus aportaciones se encuentran su ofrecimiento voluntario en varias ocasiones para ayudar con las pruebas del sistema y su ayuda en la configuración de las gafas de realidad virtual.

Tabla de contenidos

Lista de figuras

Lista de tablas

Lista de listados de código

Capítulo 1

ESPECIFICACIÓN DEL TRABAJO

1.1. Introducción

El ser humano pasa gran parte de su tiempo realizando actividades que, probablemente, no haría si no fuera por necesidad o subsistencia. Cuando no tiene que dedicarse a ellas, busca otras que le resulten más agradables o satisfactorias. En el pasado, solo quienes contaban con un alto poder adquisitivo podían permitirse tener tiempo libre y la posibilidad de buscar formas de entretenimiento. En la actualidad, pese a que la mayoría de las personas mantienen obligaciones que consumen gran parte de su tiempo, el tiempo destinado al ocio ha aumentado.

Uno de los métodos de entretenimiento más populares es el cine, un medio que a menudo transmite enseñanzas o mensajes que pueden considerarse beneficiosos para el público general. Además, una película tiene una duración conocida, lo que permite saber de antemano cuánto tiempo libre habrá que invertir en este tipo de entretenimiento. Por todo ello, no sorprende que sea un medio tan popular y que constituya una de las industrias más potentes y rentables del mundo.

Ahora bien, en la industria del cine, se produce una situación llamativa. Las personas, frente a la gran variedad de películas que tiene para elegir, tiende a dedicar más tiempo a decidir qué película ver que viéndola. Como se ha mencionado anteriormente, el tiempo de ocio es limitado y es aquí donde los sistemas de recomendación (SR) de películas son fundamentales, ya que su objetivo principal es recomendar películas, según los gustos del usuario, para evitar perder el tiempo buscando una entre tantas opciones.

Por otro lado, las personas buscan formas de intentar evadirse de su realidad. Esto se ve reflejado en la gran cantidad de medios de entretenimiento cuyo objetivo principal es escapar a una realidad “mejor”. Actualmente, la tecnología que más se aproxima a esto es la realidad virtual (RV), que busca transportar al usuario a una nueva realidad completamente diferente a la suya, generando mundos interactivos virtuales. Esta tecnología, que hasta hace unos años era muy costosa e inaccesible, se ha vuelto mucho más accesible a al público general, como se puede observar en el

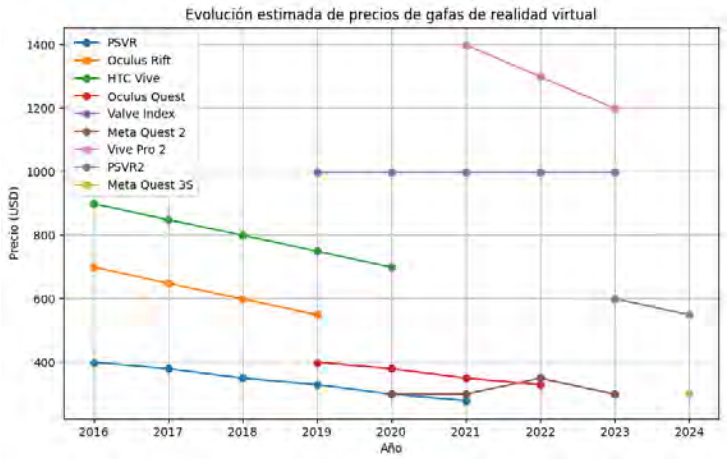


Figura 1.1: Evolución del precio en gafas de RV

descenso de precios de los dispositivos de RV en los últimos años (ver Figura 1.1).

Teniendo en cuenta todo lo anterior, en este proyecto se va a implementar un SR que tenga como objetivo sugerir al usuario películas, basándose en sus gustos y preferencias y en las de otros usuarios. Además, este sistema buscará beneficiar lo máximo posible a los usuarios considerando, no solo las películas almacenadas en una base de datos (BBDD), sino también nuevas películas que se estrenan cada semana en la cartelera. Sin embargo, esa no es la gran novedad del proyecto, pues varios SR cubren estas necesidades.

Si bien existen proyectos anteriores similares, como VR cinema [1] o metaversos que implementan sistemas de recomendación, como Roblox [2], la aplicación de SR en entornos de realidad virtual sigue siendo un campo en desarrollo, especialmente en el contexto académico, ya que los ejemplos citados tienen un origen principalmente industrial. Este proyecto busca explorar de manera experimental cómo un SR puede integrarse en dispositivos de RV para mejorar la experiencia del usuario, ofreciendo un enfoque formativo y experimental distinto al de los prototipos industriales existentes. Aunque toda la lógica de la aplicación se construirá sobre conceptos previamente establecidos y optimizados en la literatura, particularmente en lo relacionado con SR, la interfaz estará diseñada específicamente para su uso en dispositivos de RV.

Es importante recalcar que el proyecto presenta una novedad muy importante dentro del área de los SR y RV, ya que actualmente las gafas se utilizan principalmente para ver películas que el usuario tenga en propiedad o para jugar videojuegos en RV. La integración de un SR para este tipo de dispositivos, todavía sigue siendo un área poco explorada en el ámbito académico y formativo.Por ello, este proyecto permite explorar de manera preliminar la viabilidad de nuevas experiencias de recomendación en RV, contribuyendo al conocimiento sobre la interacción de los usuarios con contenidos recomendados en entornos virtuales.

El nombre por el que se dará a conocer este proyecto será S.C.E.N.E., que es el acrónimo de *Smart Cinematic Experience & Navigation Engine*.

1.2. Objetivos del trabajo

En esta sección se enumeran todos los objetivos que se han considerado necesarios para concluir de forma satisfactoria este proyecto.

1.2.1. Objetivos generales

Uno de los objetivos principales en este trabajo es emplear distintas técnicas de recomendación de películas a usuarios para mejorar significativamente su experiencia considerando, no solo películas almacenadas en una BBDD, sino también aquellas que están en la cartelera actualmente. Para ello, implementaremos un SR basado en filtrado colaborativo (SRFC) para recomendar películas almacenadas en una BBDD y, por otra parte, se empleará un SR basado en contenidos (SRBC) para aquellas películas que se encuentren en cartelera.

Además, y como gran novedad, este proyecto busca la inmersión del usuario en el SR experimentando con técnicas de RV. Es por ello que se deberá establecer un entorno virtual completamente funcional que cubra las necesidades que el usuario pudiera presentar en lo que respecta a las puntuaciones de películas y obtención de las recomendaciones.

1.2.2. Objetivos específicos

Los objetivos específicos, que surgen de desglosar el objetivo general expuesto en el apartado anterior, se presentan a continuación:

- **#OBJ01**: Revisión del estado del arte en el ámbito de SR.
- **#OBJ02**: Revisión del estado del arte en el ámbito de los sistemas de RV.
- **#OBJ03**: Estudio y análisis de conjuntos de datos relacionados con la recomendación a realizar, en este caso, películas.
- **#OBJ04**: Preprocesamiento y limpieza de los datos con el objetivo de adaptarlos a las necesidades de este proyecto.
- **#OBJ05**: Implementación de un backend que cumpla con todas las necesidades que el SR precise.
- **#OBJ06**: Implementación de un frontend que cumpla con todas las necesidades que el SR precise.
- **#OBJ07**: Conexión entre backend y frontend.
- **#OBJ08**: Experimentación sobre el SR.

- **#OBJ09**: Elaboración de la documentación relacionada con el uso y funcionamiento del SR.
- **#OBJ10**: Despliegue del sistema en un entorno real y elaboración de una documentación final del proyecto.

1.3. Metodología de desarrollo software y definición de tareas

Este proyecto adoptará una metodología en cascada, un enfoque estructurado y secuencial ampliamente utilizado en la gestión de proyectos [3, 4]. Este modelo organiza el desarrollo en una serie de fases lineales, donde cada etapa depende de la finalización de la anterior antes de que se pueda proceder a la siguiente. Su naturaleza progresiva permite un control riguroso del proyecto y una planificación detallada en cada fase, asegurando claridad y orden a lo largo del proceso.

La metodología en cascada consiste en varias fases bien definidas. En primer lugar, se realiza una recolección y análisis de requisitos, donde se identifican y documentan todas las necesidades y objetivos del sistema, asegurando que se comprenden completamente antes de avanzar. Luego, en la fase de diseño, se desarrolla una arquitectura detallada del sistema, incluyendo diagramas, especificaciones y estrategias técnicas para cumplir con los requisitos establecidos. A continuación, la implementación o codificación transforma esos diseños en un sistema funcional. Posteriormente, en la fase de pruebas, se verifica que el sistema desarrollado cumple con los requisitos definidos y funciona correctamente, identificando y corrigiendo posibles errores. Una vez finalizadas las pruebas, se procede al despliegue, donde el sistema se entrega al cliente y se pone en funcionamiento. Finalmente, durante la fase de mantenimiento, se abordan ajustes, correcciones y mejoras necesarias para garantizar el correcto funcionamiento del sistema a lo largo del tiempo.

Este enfoque permite una estructura clara y un control eficiente del proyecto, con documentación detallada en cada etapa que sirve como guía y referencia. Su enfoque metódico y su énfasis en la planificación detallada aseguran un desarrollo ordenado y predecible, alineado con los objetivos planteados desde el inicio. Ademas de todas estas razones que pueden ayudar a comprender porque es tan usada esta metodología, es relevante destacar que el estudiante a cargo del proyecto ya está familiarizado con ella, lo que permitirá una adaptación inmediata tanto en la planificación como en la ejecución del trabajo. Asimismo, al tratarse de un proyecto complejo llevado a cabo por un único estudiante, esta metodología facilitará desde el inicio un control claro del progreso y de los tiempos, permitiendo ajustarse de manera ágil a posibles imprevistos

Una vez definida la metodología que se llevará a cabo, a continuación, se definirán las tareas a realizar en el proyecto. Debido a la complejidad de un proyecto como este, se ha decidido definir una serie de tareas generales, desglosar estas en tareas más pequeñas y concretas para, finalmente, a partir de estas últimas, obtener las actividades.

Por tanto, primero se debe de definir las tareas generales:

- **#EDA**: Estado del arte. Esta tarea se corresponde con la investigación sobre el estado del arte. Durante esta etapa se obtendrán todos los conocimientos necesarios para poder llevar a cabo el desarrollo del proyecto. Esta tarea permite cumplir con los objetivos: #OBJ01 y #OBJ02.
- **#ALD**: Análisis y limpieza de datos. Esta tarea consiste en el estudio de varias BBDD de películas, su análisis y la elección de la que más se alinee con las necesidades del proyecto. Esta tarea permite cumplir con el objetivo #OBJ03.
- **#PREP**: Preprocesamiento de los datos. Una vez se seleccione la BBDD con la que trabajar, se realizará un preprocesamiento de los datos para adaptarlos a las necesidades del proyecto. Esta tarea permite cumplir con el objetivo #OBJ04.
- **#IMP**: Implementación. En esta tarea se busca realizar toda la implementación del sistema a nivel lógico. Esta tarea permite cumplir con los objetivos #OBJ05, #OBJ06 y #OBJ07.
- **#EXP** Experimentación. En esta tarea se testeará el sistema ya implementado con la finalidad de validar su comportamiento. Esta tarea permite cumplir con el objetivo #OBJ08.
- **#COOR** Coordinación del proyecto. Esta tarea busca llevar a cabo la coordinación y documentación de todos los aspectos relevantes del proyecto. Esta tarea permite cumplir con el objetivo #OBJ09.
- **#SETUP**: Despliegue. Esta tarea finalizará con el despliegue de la aplicación en un entorno real, cumpliendo con el objetivo #OBJ10.

1.3.1. EDT

La estructura de desglose de trabajo (EDT), se define como una descomposición jerárquica del trabajo en partes más pequeñas. Su finalidad consiste en ser una ayuda visual y organizativa de las tareas y actividades que componen el proyecto. La diferencia entre el EDT y la matriz de trazabilidad (que se verá a continuación en la Sección 1.3.3) es que, mientras que la primera busca una descomposición y organización del trabajo de forma general (ver Figura 1.2), la segunda estudia el comportamiento de cada elemento que se encuentra en lo más profundo de la descomposición [5].

1.3.2. Diccionario de la EDT para SCENE

Una vez presentada la EDT, a continuación se explicarán las tareas específicas que nacen de las más generales (representadas en color azul en la Figura 1.2).

- **#EDA:INVESTIGACIÓN SOBRE ESTADO DEL ARTE**

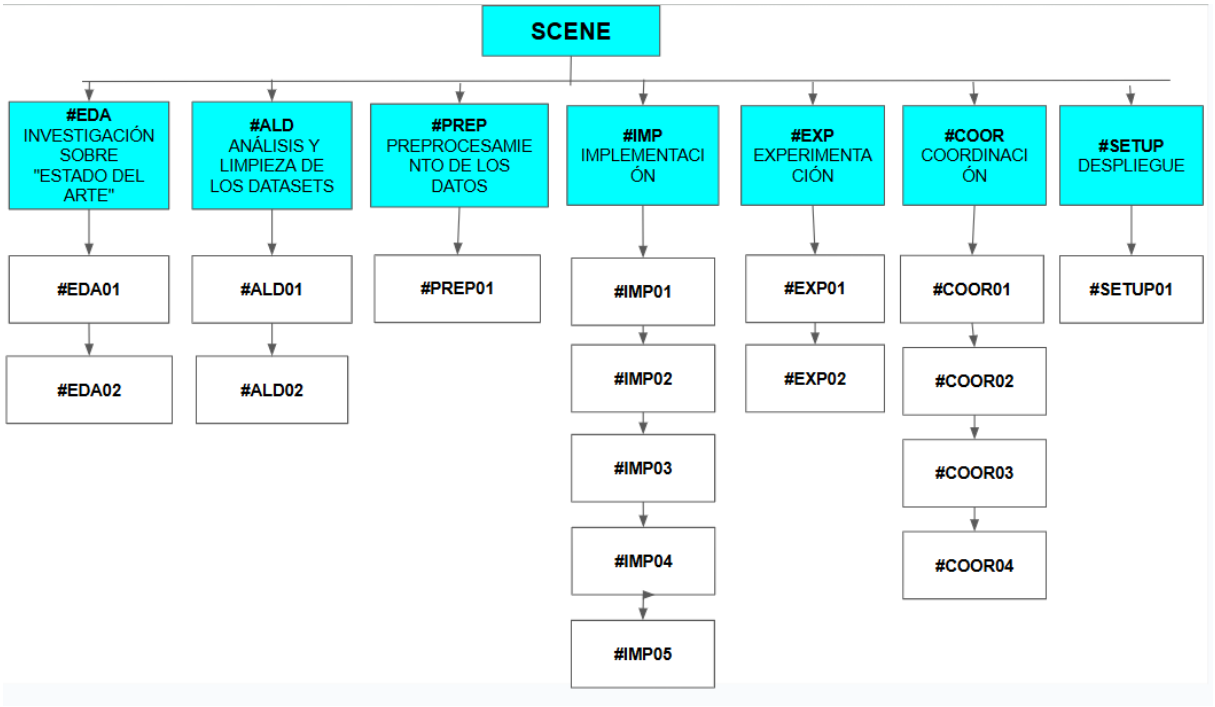


Figura 1.2: EDT

- **#EDA01: Tecnologías existentes para recomendación**
  - Descripción: Investigación sobre algoritmos y bibliotecas empleadas en la creación de SR, tanto de SRFC, como SRBC.
  - Entregable: Informe sobre algoritmos y bibliotecas que implementan SR.
  - Objetivo: #OBJ01.
- **#EDA02: Tecnologías existentes de RV**
  - Descripción: Investigación sobre herramientas, procedimientos y elementos que puedan ayudar a guiar a los desarrolladores en el proceso de implementación de un programa de RV.
  - Entregable: Informe sobre todas las tecnologías, elementos y material adicional que se necesitará conocer para la implementación de un sistema de RV.
  - Objetivo: #OBJ02.

- **#ALD: ANÁLISIS Y LIMPIEZA DE LOS DATASETS**
  - **#ALD01: Recopilación de datasets de películas**
    - Descripción: Se realizará una búsqueda en los principales sitios web que albergan BBDD gratuitas para su descarga y análisis.
    - Entregable: Varias BBDD de películas.
    - Objetivo: #OBJ03.
  - **#ALD02: Análisis de datasets de películas**
    - Descripción: Se realizará un estudio en profundidad de las BBDD obtenidas previamente. Se descartarán las que se consideren inadecuadas.

- Entregable: La BBDD con la que se trabajará en el proyecto.
- Objetivo: #OBJ03.

- **#PREP: PREPROCESAMIENTO DE LOS DATOS**
  - **#PREP01: Preparación de los datasets**
    - Descripción: Se realizará un preprocesamiento de la BBDD. Se implementarán nuevos archivos que cumplan con el formato de la BBDD ya existente y se realizarán las modificaciones pertinentes.
    - Entregable: Una BBDD completamente adaptada a las necesidades del proyecto.
    - Objetivo: #OBJ04.
- **#IMP: IMPLEMENTACIÓN**
  - **#IMP01: Implementación de algoritmo de SRFC**
    - Descripción: Se implementa en el backend un algoritmo de recomendación mediante SRFC para películas.
    - Entregable: Código con la implementación de un SRFC funcional con el dataset.
    - Objetivo: #OBJ05.
  - **#IMP02: Implementación de algoritmo de recomendación SRBC**
    - Descripción: Se implementa en el backend un algoritmo de recomendación mediante SRBC para películas que están en cartelera.
    - Entregable: Código con la implementación de un SRBC funcional con el dataset.
    - Objetivo: #OBJ05.
  - **#IMP03: Implementación de toda la lógica de gestión interna del backend**
    - Descripción: Se implementa toda la lógica de gestión interna de los usuarios del sistema. Lo que implica la creación de nuevos usuarios, el login de usuarios ya existentes y la modificación de los ratings de las películas.
    - Entregable: Código con toda la implementación de la gestión de usuario completamente operativa.
    - Objetivo: #OBJ05.
  - **#IMP04: Implementación de un frontend completo para RV**
    - Descripción: Se implementa toda la lógica del frontend de la aplicación. El frontend deberá ser capaz de visualizar y procesar de forma correcta no solo el escenario, sino también toda la información que se le envíe desde el backend.
    - Entregable: Código implementado de un frontend completamente funcional.
    - Objetivo: #OBJ06.
  - **#IMP05: Conexión frontend y backend**

- Descripción: Se implementará un sistema de comunicación mediante sockets tanto en el frontend como en el backend para permitir que ambos puedan comunicarse.
- Entregable: Código de la aplicación completamente funcional.
- Objetivo: #OBJ07.

- **#EXP : EXPERIMENTACIÓN**
  - **#EXP01: Pruebas de laboratorio**
    - Descripción: Pruebas del sistema en un entorno controlado.
    - Entregable: Informe de resultados experimentales en laboratorio.
    - Objetivo: #OBJ08.
  - **#EXP02: Pruebas en entornos reales**
    - Descripción: Pruebas con usuarios reales para validar la aplicación en entornos reales.
    - Entregable: Informe de resultados experimentales en entornos reales.
    - Objetivo: #OBJ08.
- **#COOR: COORDINACIÓN**
  - **#COOR01: Coordinación del proyecto**
    - Descripción: Proceso de coordinación del proyecto completo.
    - Entregable: Memoria del proyecto y documentación asociada.
    - Objetivo: #OBJ10.
  - **#COOR02: Manual de usuario del modelo jerárquico.**
    - Descripción: Se desarrollará una guía de implementación y uso del modelo desarrollado.
    - Entregable: Manual de usuario.
    - Objetivo: #OBJ09.
  - **#COOR03: Funcionalidad del modelo.**
    - Descripción: Documentación técnica sobre las funcionalidades y limitaciones del sistema.
    - Entregable: Documento funcional.
    - Objetivo: #OBJ09.
  - **#COOR04: Memoria del proyecto.**
    - Descripción: Memoria final del proyecto, incluyendo toda la documentación generada anteriormente.
    - Entregable: Memoria del proyecto estructurada.
    - Objetivo: #OBJ10.
- **#SETUP: DESPLIEGUE**
  - **#SETUP01: Despliegue del modelo en producción**
    - Descripción: Implementación del modelo en entornos de producción.
    - Entregable: Sistema funcional y operativo en producción.
    - Objetivo: #OBJ10.

ID	Tipo	Prioridad	Objetivo	Funcionalidad
#EDA01	Investigación	Alta	#OBJ01	Formación del personal
#EDA02	Investigación	Alta	#OBJ02	Formación del personal
#ALD01	Análisis	Alta	#OBJ03	Recolección de datasets
#ALD02	Análisis	Alta	#OBJ03	Análisis de datasets y elección del dataset de trabajo
#PREP01	Preprocesamiento	Alta	#OBJ04	Preparación del dataset
#IMP01	Implementación	Alta	#OBJ05	Implementación del algoritmo de recomendación
#IMP02	Implementación	Alta	#OBJ05	Implementación del algoritmo de recomendación
#IMP03	Implementación	Alta	#OBJ05	Implementación de gestión de datos interno
#IMP04	Implementación	Alta	#OBJ06	Implementación de un frontend
#IMP05	Implementación	Alta	#OBJ07	Implementación de la comunicación entre backend y frontend
#EXP01	Experimentación	Media	#OBJ08	Pruebas en entornos controlados
#EXP02	Experimentación	Media	#OBJ08	Pruebas en entornos reales
#COOR01	Coordinación	Alta	#OBJ10	Coordinación del proyecto
#COOR02	Coordinación	Media	#OBJ09	Manual de usuario
#COOR03	Coordinación	Media	#OBJ09	Documentación técnica
#COOR04	Coordinación	Media	#OBJ10	Estrategia de mercado
#SETUP01	Despliegue	Baja	#OBJ10	Lanzamiento del sistema

Tabla. 1.1: Matriz de trazabilidad

1.3.3. Matriz de trazabilidad de requisitos

En este apartado se presenta una matriz de trazabilidad de requisitos (ver Tabla 1.1). Esta matriz permite relacionar y rastrear cada requisito del proyecto con los elementos que garantizan su cumplimiento. Además, presentar la información de esta forma estructurada y visual, permite asegurar que todos los requisitos se cumplan y visualizar como cada pieza del proyecto contribuye a la totalidad del mismo [6].

1.4. Planificación temporal

Un factor clave en cualquier proyecto es la planificación temporal del mismo. Es fundamental tener en cuenta que el tiempo es limitado y tiene un coste. Por tanto, se debe de organizar y planificar el proyecto de la mejor manera posible para se puedan obtener los resultados planteados en el tiempo previsto. En este apartado, se definen las actividades a realizar para completar las tareas indicadas anteriormente y se establecerán los hitos más importantes del proyecto.

1.4.1. Lista de actividades

Para empezar con la planificación temporal de la ejecución del proyecto deberemos de establecer las actividades concretas necesarias para completarlo. Para facilitar su comprensión, a continuación se conectan dichas actividades a una tarea definida anteriormente:

- #COOR01:
  - **Código: #ACT01** El alumno se encargará de supervisar al resto de trabajadores en cada una de las actividades asignadas. En nuestro caso, solo

habrá una persona en el equipo que se encargará de la supervisión y desarrollo de las actividades. Por lo que la misma persona asumirá todos los roles dentro del proyecto.

- **Código: #ACT02** El alumno se encargará de realizar un informe sobre la información recopilada en la tarea #EDA01.
- **Código: #ACT03** El alumno se encargará de realizar un informe sobre la información recopilada en la tarea #EDA02.
- **Código: #ACT04** El alumno se encargará de realizar un informe sobre el dataset elegido para trabajar y las características del mismo. Es decir, un informe de los resultados de la tarea #ALD02.
- **Código: #ACT05** El alumno se encargará de hacer una documentación de todos los resultados obtenidos tras todas las pruebas reales asociadas a la tarea #EXP02.

- #COOR02:
  - **Código: #ACT06** El alumno deberá, de forma estructurada, elaborar un documento de Google Docs con información sobre: como instalar el sistema, incluir ejemplos prácticos de uso y proporcionar soluciones a problemas comunes.

- #COOR03:
  - **Código: #ACT07** El alumno deberá, de forma estructurada, elaborar un documento de Google Docs con información sobre: descripción de la arquitectura técnica del sistema, detalles sobre configuración y parámetros y determinar dependencias necesarias del sistema.

- #COOR04
  - **Código: #ACT08** El alumno deberá, de forma estructurada, elaborar un documento de Google Docs con información sobre un plan de negocio donde se incluya: impacto de mercado, costes y beneficios y estrategias de mercado.

- #EDA01:
  - **Código: #ACT09** El alumno deberá hacer una búsqueda bibliográfica usando las siguientes herramientas: IEEE Xplore, Google Scholar, Springer Link y los libros relacionados con la temática en la biblioteca de la Universidad de Jaén.
  - **Código: #ACT10** El alumno realizará un estudio sobre las principales librerías existentes que cumplen con las necesidades del proyecto.
  - **Código: #ACT11** El alumno, una vez terminado los puntos anteriores, se encargará de hacer pruebas con las librerías anteriores para seleccionar las que más convengan al proyecto.

- #EDA02:

- **Código: #ACT12** El alumno se encargará de realizar una búsqueda bibliográfica sobre la documentación de Meta sobre gafas RV, documentación oficial de Unity y Google Scholar.
- **Código: #ACT13** El alumno se encargará, con toda la información obtenida de la tarea anterior, de definir el apartado técnico que será necesario para la construcción del backend.

- #ALD01:
  - **Código: #ACT14** El alumno será el encargado de recopilar distintos datasets de películas, destacando lugares como tmdb, MovieLens, Kaggle o cualquiera que se considere relevante.
- #ALD02:
  - **Código: #ACT15** El alumno realizará un estudio en profundidad sobre la estructura y contenido que presentan estas BBDD.
  - **Código: #ACT16** El alumno con la información anterior, decidirá cuál será el dataset con el que trabajar.
- #PREP01:
  - **Código: #ACT17** El alumno será el encargado de preprocesar los datos según las necesidades del sistema.
- #IMP01:
  - **Código: #ACT18** El alumno se encargará de la configuración del entorno de programación mediante la instalación de las librerías necesarias para trabajar con Python.
  - **Código: #ACT19** El alumno se encargará mediante el uso de librerías de implementar un SRFC.
  - **Código: #ACT20** El alumno se encargará de hacer pruebas controladas del algoritmo y realizar las modificaciones necesarias.
- #IMP02:
  - **Código: #ACT21** El alumno se encargará mediante el uso de librerías de implementar un SRBC.
  - **Código: #ACT22** El alumno se encargará de hacer pruebas controladas del algoritmo y realizar las modificaciones necesarias.
- #IMP03:
  - **Código: #ACT23** El alumno será el encargado de implementar un sistema de suscripción de usuarios: login, creación de usuarios, autentificación, etc.
  - **Código: #ACT24** El alumno será el encargado de unificar todas las funciones ya programadas: los dos SR y la gestión de usuarios.
- #IMP04:

- **Código: #ACT25** El alumno se encargará de la configuración del entorno de programación mediante la instalación de las bibliotecas necesarias para trabajar con Unity.
- **Código: #ACT26** El alumno será el encargado de la búsqueda y descarga de assets para la correcta implementación del sistema y la mejora visual del mismo.
- **Código: #ACT27** El alumno creará el entorno inmersivo para la recomendación de películas, junto con todas las necesidades del mismo.
- **Código: #ACT28** El alumno se encargará de realizar las pruebas del entorno inmersivo y realizar las mejoras necesarias.

■ #IMP05

- **Código: #ACT29** El alumno se encargará de implementar un sistema de comunicación de sockets en Python.
- **Código: #ACT30** El alumno se encargará de implementar un sistema de comunicación de sockets en Unity.
- **Código: #ACT31** El alumno se encargará de hacer pruebas de comunicación entre ambos sistemas y realizar las modificaciones necesarias.

■ #EXP01

- **Código: #ACT32** El alumno hará pruebas para analizar el funcionamiento del sistema.
- **Código: #ACT33** El alumno, a partir de los datos anteriores, se encargará de ajustar el sistema.

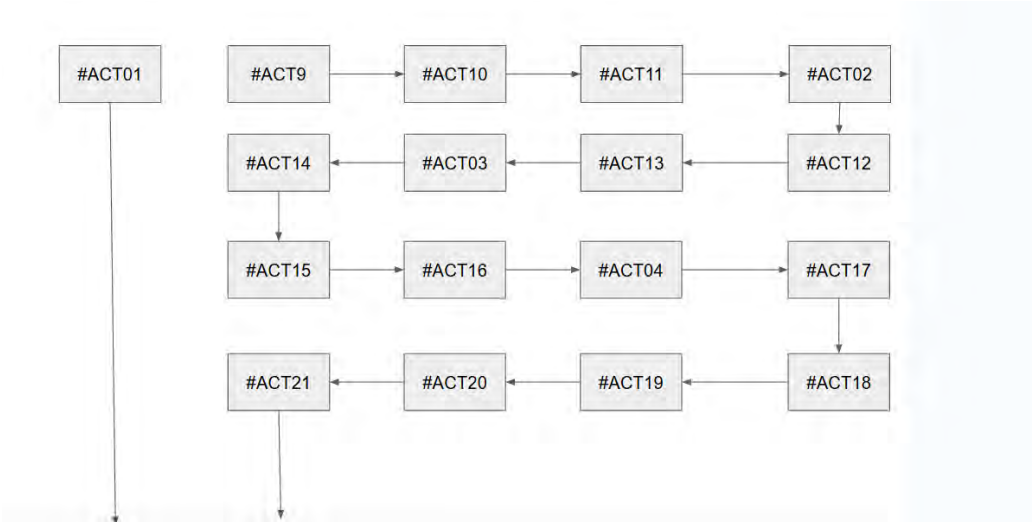
■ #EXP02

- **Código: #ACT34** El alumno implementará el backend en un servidor real local, en este caso a través de Docker.
- **Código: #ACT35** Un usuario real probará el sistema en unas gafas de RV y dará su opinión sobre el sistema.
- **Código: #ACT36** El alumno, a partir de los datos anteriores, se encargará de ajustar el sistema.

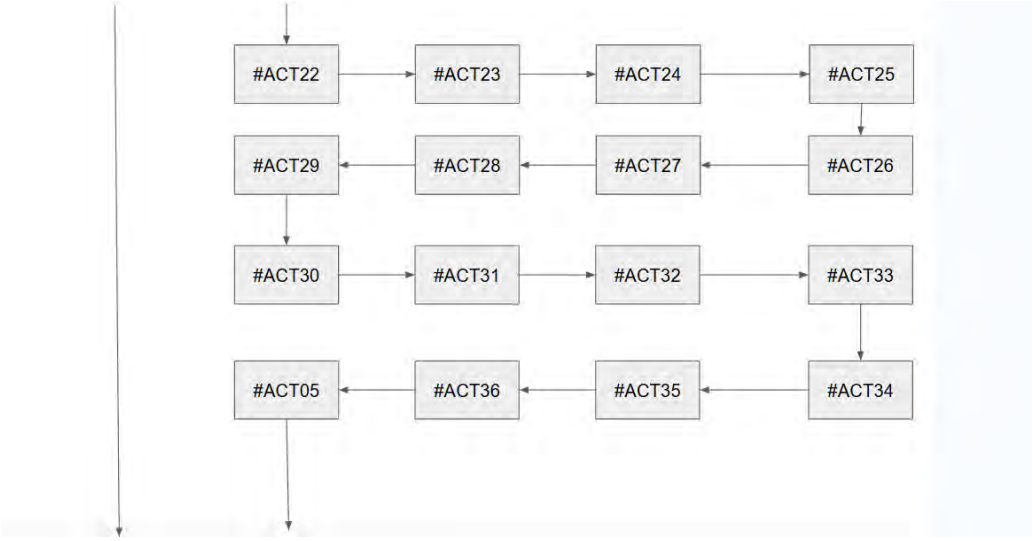
■ #SETUP01

- **Código: #ACT37** El alumno se encargará de la creación de un entorno de producción, en este caso usando Docker.
- **Código: #ACT38** El alumno creará un repositorio en Google Drive para publicar el producto.
- **Código: #ACT39** El alumno se encargará de subir el producto a Google Drive.

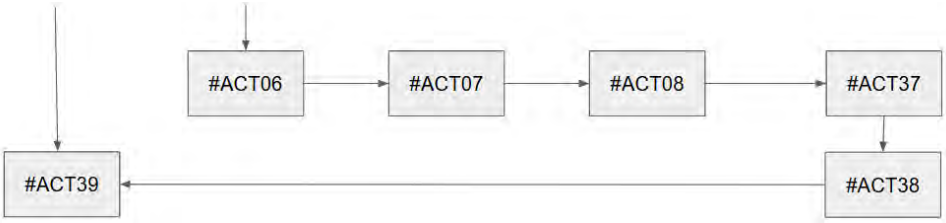
Una vez presentadas las tareas, se deberá definir el flujo de secuencia de las mismas. Gracias a este flujo, se podrá visualizar la correlación entre las actividades. Este flujo corresponderá a las Figuras 1.3a, 1.3b y 1.3c



(a) Flujo de actividades 1



(b) Flujo de actividades 2



(c) Flujo de actividades 3

Figura 1.3: Flujo de actividades

1.4.2. Lista de hitos

A lo largo de este proyecto se deberá crear una serie de eventos significativos que nos guíen sobre el estado de madurez en el que se encuentra el proyecto denominados hitos. En este punto presentaremos la lista de hitos:

- **Código: #HITO01** Preparación y formación previa de personal. Este hito quedará completado cuando se hayan cumplido las tareas asociadas a las tareas #EDA01 y #EDA02.
- **Código: #HITO02** Preparación del dataset a trabajar. Este hito quedará completado cuando se hayan cumplido las tareas asociadas a las tareas #ALD01, #ALD02 y #PREP01.
- **Código: #HITO03** Implementación del sistema. Este hito quedará completado cuando se hayan cumplido las tareas asociadas a las tareas #IMP01, #IMP02, #IMP03, #IMP04 y #IMP05.
- **Código: #HITO04** Experimentación sobre el trabajo ya implementado. Este hito se considerará superado una vez terminadas las tareas asociadas a las tareas #EXP01 y #EXP02.
- **Código: #HITO05** Documentación y despliegue del sistema. Este hito quedará completado cuando se hayan cumplido las tareas asociadas a las tareas #COOR01, #COOR02, #COOR03, #COOR04 y #SETUP01.

1.4.3. Estimación de la duración de actividades

Para estimar la duración de las actividades previamente planteadas, en este proyecto se ha optado por preguntar a trabajadores con experiencia en el desarrollo de proyectos software. De esta forma, se ha podido estimar la duración de cada actividad (ver Tabla 1.2). Estos tiempos, no solo incluyen el tiempo estimado para la realización de la actividad, sino también posibles retrasos. A partir de estos datos, se puede definir un PERT que aporte, de forma visual, información sobre el orden en el que se deben de realizar las actividades y el tiempo asignado a las mismas (ver Figuras 1.4, 1.5, 1.6 y 1.7).

Finalmente, también se define un diagrama de GAntt que permita de forma visual tener una organización temporal del proyecto (ver Figura 1.8).

Actividad	Duración estimada (h)	Precedentes
#ACT01	450	-
#ACT02	6	#ACT11
#ACT03	6	#ACT13
#ACT04	6	#ACT16
#ACT05	6	#ACT35
#ACT06	12	#ACT05
#ACT07	12	#ACT06
#ACT08	12	#ACT07
#ACT09	12	-
#ACT10	12	#ACT09
#ACT11	12	#ACT10
#ACT12	12	#ACT02
#ACT13	12	#ACT12
#ACT14	12	#ACT03
#ACT15	12	#ACT14
#ACT16	6	#ACT15
#ACT17	12	#ACT04
#ACT18	6	#ACT17
#ACT19	30	#ACT18
#ACT20	12	#ACT19
#ACT21	30	#ACT20
#ACT22	12	#ACT21
#ACT23	18	#ACT22
#ACT24	12	#ACT23
#ACT25	12	#ACT24
#ACT26	12	#ACT25
#ACT27	54	#ACT26
#ACT28	12	#ACT27
#ACT29	6	#ACT28
#ACT30	6	#ACT29
#ACT31	12	#ACT30
#ACT32	6	#ACT31
#ACT33	6	#ACT32
#ACT34	6	#ACT33
#ACT35	6	#ACT34
#ACT36	6	#ACT35
#ACT37	6	#ACT08
#ACT38	12	#ACT37
#ACT39	6	#ACT38, #ACT01

Tabla. 1.2: Duración estimada y precedentes de actividades

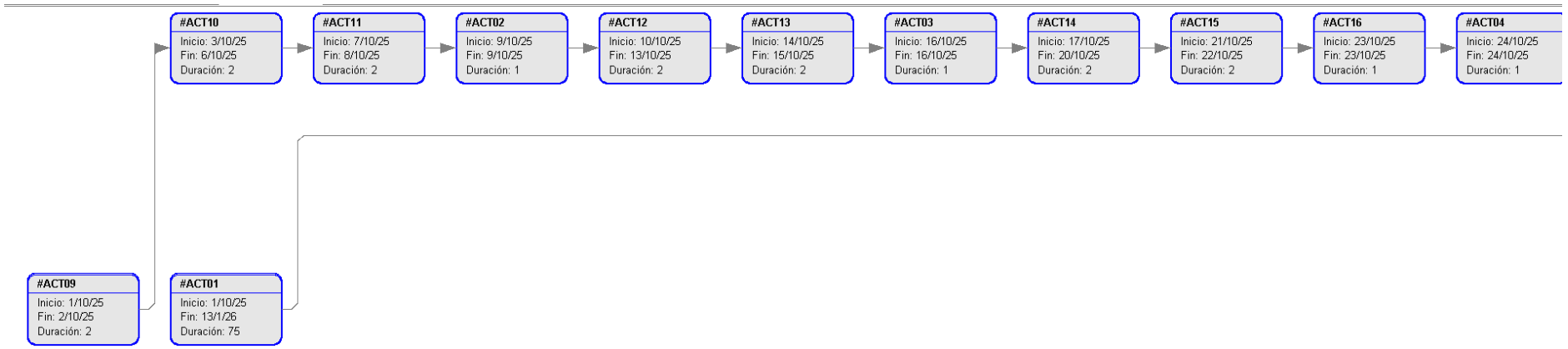


Figura 1.4: PERT 1

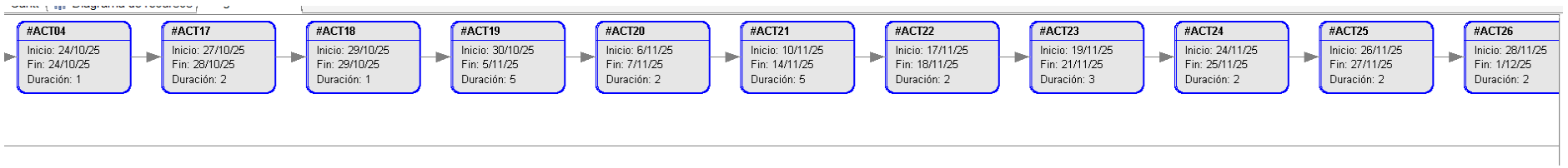


Figura 1.5: PERT 2

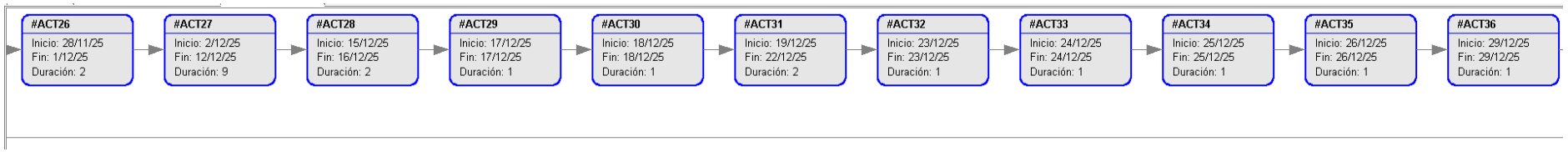


Figura 1.6: PERT 3

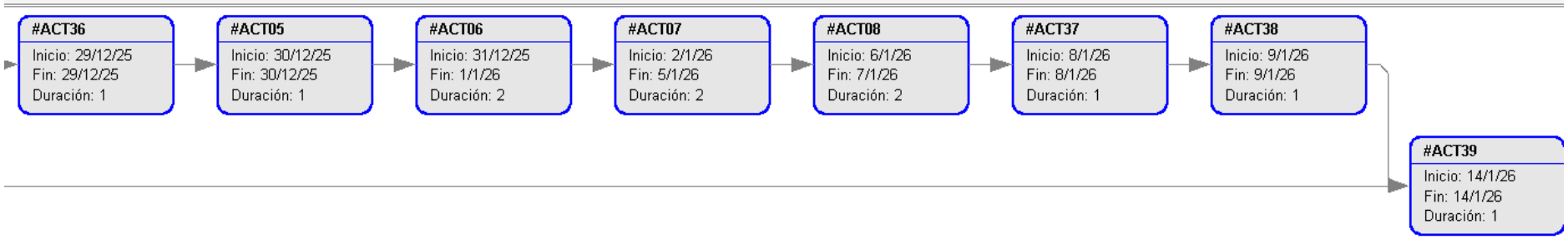


Figura 1.7: PERT 4

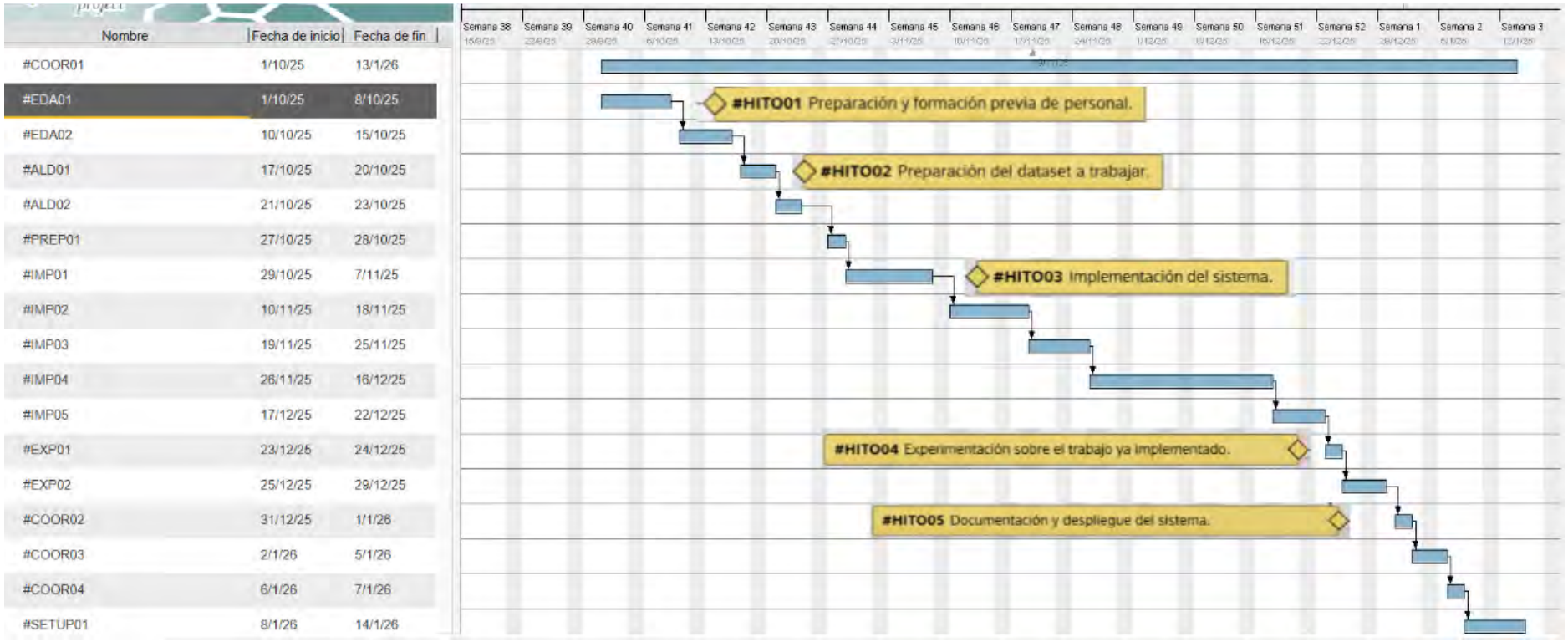


Figura 1.8: GANTT organizado por tareas

1.5. Presupuesto

Recurso	Horas totales	Coste total (€)
Conjunto de datos - [#RH01]	–	0
Estudiante - [#RH02]	450	15750
PC de estudiante - [#RH03]	450	400
Gafas VR - [#RH04]	30	36.58
Servidor de la universidad - [#RH05]	18	28.33
Assets Unity - [#RH06]	–	0
Total	450	16214.91

Tabla. 1.3: Recursos utilizados y costes asociados

En este apartado se busca desglosar lo máximo posible todas las necesidades financieras que estarán presentes en este proyecto (ver Tabla 1.3), teniendo en cuenta las peculiaridades propias de un Trabajo de Fin de Máster (TFM). De esta forma, se enumerarán todas los recursos materiales y humanos que serán indispensables para el correcto desarrollo de este proyecto:

- Los conjuntos de datos necesarios para llevar a cabo el proyecto son de acceso libre, y no conllevarán ningún coste añadido.
- El personal estará compuesto únicamente por un estudiante de máster realizando prácticas de empresa, por lo que no supondrá gasto alguno para la universidad. Sin embargo, con el fin de contemplar posibles circunstancias imprevistas que requieran la contratación con su respectivo sueldo del estudiante, se definirá un presupuesto específico para tal eventualidad. Este alumno realizará las funciones de: coordinador de proyecto, diseñador gráfico e ingeniero software. Sin embargo, vamos a similar lo que supondría contratar a una persona con poca experiencia laboral pero con los conocimientos requeridos para este trabajo. El sueldo medio de un programador junior suele estar entre los 21 y 35 euros la hora [7]. Debido a que todo el trabajo es realizado por una sola persona, se considerará el pago de 35 euros la hora. Teniendo en cuenta que se van a dedicar al proyecto 450 horas, el total asciende a 15750 euros.
- 1 - PcCom Ready Intel Core / 16GB / 1TB SSD / Windows 11. Será el equipo informático donde se desarrolle todo el proyecto. En este caso será proporcionado por el propio alumno sin coste alguno. Vamos a simular el coste de un equipo de estas características. El precio del equipo es de 1200 euros. Teniendo en cuenta que será usado durante un periodo de 4 meses, eso asciende a una suma total de 400 euros.
- 1 - Gafas de RV meta quest 3. Esta tecnología no conlleva ningún coste adicional, pues es propiedad del grupo de investigación Sinbad2. Sin embargo, vamos a simular su compra, cuyo coste será de 439 euros, según el precio actual de mercado. Teniendo en cuenta la amortización de un mes del dispositivo, el precio sería: 36.58 euros.

- 1 - Servidor para implementar el backend del sistema. Este sistema no supondrá gasto alguno, pues el grupo de investigación ya dispone de un servidor completamente operativo para la implementación de este sistema. Vamos a simular el costo de este equipo que asciende a 340 euros. teniendo en cuenta la amortización del mismo, el precio final sería de 28.33 euros.
- Todos los objetos que componen la escena desarrollada Unity serán gratuitos.
- Por todo ello, se estima que el gasto de todo el proyecto será de 16214.91 euros.

Es necesario remarcar un detalle importante. Debido a que el alumno será el único encargado del desarrollo de todo el proyecto, no será necesaria la realización de una tabla de asignación de horas a trabajadores, pues todas las actividades serán realizadas por el mismo, como se indicó en los apartados anteriores.

1.6. Estructura de la memoria

Una vez presentada toda la planificación previa al desarrollo del proyecto, en este apartado se describe la estructura de este documento.

En el Capítulo 1 se ha realizado una presentación de los objetivos del proyecto y toda la planificación previa al mismo. Una vez terminado este apartado, el usuario debería tener un conocimiento amplio sobre el trabajo que se va a desarrollar.

El Capítulo 2 abarca los antecedentes. En este apartado se busca revisar el estado del arte de las tecnologías y técnicas principales que se relacionan con el trabajo presentado. Cuando el lector termine este capítulo, habrá obtenido conocimientos amplios sobre la materia de la que trata este proyecto.

El Capítulo 3 expone el diseño del sistema. En este capítulo profundizaremos en el diseño del sistema que vamos a desarrollar. Además, en este apartado también se introducirá toda la colección de tecnologías seleccionadas para llevar a cabo dicho desarrollo.

El Capítulo 4 aborda la implementación del sistema. En este capítulo se describe en detalle la implementación de la mayoría de elementos que componen el proyecto.

El Capítulo 5 ofrece los resultados de la experimentación final y una conclusión, con su respectiva reflexión, sobre los resultados obtenidos.

Capítulo 2

ANTECEDENTES

2.1. Sistemas de recomendación

El fenómeno de la obsolescencia tecnológica ha estado presente en muchos ámbitos y el sector audiovisual no es una excepción. Un ejemplo claro es el cambio que se produjo desde los antiguos videoclubs hasta las actuales plataformas de distribución digital. Durante décadas, los videoclubs fueron la forma habitual de alquilar películas, pero con el tiempo fueron desapareciendo y dando paso a servicios como Netflix o Amazon Prime Video, que ofrecen un funcionamiento similar aunque basado completamente en contenidos digitales y accesibles desde cualquier lugar. Esta transición muestra cómo ciertos formatos físicos pueden dejar de existir, mientras que las ideas o servicios que ofrecían continúan adaptados a nuevas tecnologías.

El gran volumen de usuarios que manejan las plataformas actuales supone un reto tecnológico importante, ya que obliga a garantizar un servicio estable, eficiente y adaptado a cada persona. Dado que se trata de un mercado muy competitivo, muchas empresas consideran que mejorar la experiencia del usuario es fundamental para mantener su posición y atraer a nuevos clientes. Una de las herramientas más utilizadas con este fin son los SR, que analizan los gustos y comportamientos de cada usuario para sugerir contenidos que puedan resultar de su interés. Un ejemplo conocido es el Netflix Prize, un concurso que ofreció una recompensa económica importante con el objetivo de desarrollar un algoritmo de recomendación más preciso para la plataforma [8, 9].

Otro aspecto que explica la relevancia de los SR es la denominada paradoja de la elección, un concepto estudiado en psicología que señala que la capacidad de decisión del ser humano disminuye a medida que aumenta el número de opciones disponibles. Para demostrarlo, uno de los experimentos más citados comparó el comportamiento de dos grupos de consumidores: cuando se ofrecieron 6 variedades de mermelada, aproximadamente un 12 % realizó una compra, mientras que en el grupo al que se le presentaron 30 variedades, el porcentaje descendió al 2 %. Si se extrapola esta idea a plataformas como Netflix, que cuentan con miles de títulos, se entiende la necesidad de contar con sistemas que ayuden al usuario a encontrar contenidos rele-

vantes sin abrumarlo. Esta situación ha impulsado a muchas empresas a competir por desarrollar algoritmos de recomendación cada vez más precisos y eficientes [10].

Una vez expuesta la información necesaria sobre la importancia y los antecedentes del tema tratado, es apropiado pasar al apartado técnico. En las siguientes secciones se presentarán los principales tipos de SR utilizados en la actualidad, así como sus características más destacadas junto con sus principales ventajas e inconvenientes.

2.1.1. SRFC

El SRFC es una técnica de recomendación que sugiere contenido a los usuarios basándose en las similitudes entre ellos. En términos simples, la idea central es que si dos personas comparten gustos similares, es probable que también disfruten de productos similares. Por ejemplo, si a una persona X le ha gustado un determinado producto y tiene un perfil de preferencias similar al de otra persona Y, entonces es razonable suponer que a Y también podría gustarle ese producto [11].

Existen 3 tipos de metodologías distintas:

1. SRFC user-user, basado en la similitud entre usuarios.
2. SRFC item-item, basado en la similitud entre ítems (en nuestro caso, películas).
3. Factorización matricial, una técnica más avanzada que modela las interacciones usuario-ítem mediante representaciones latentes.

Primero, el método user-user [11]. Este método se puede considerar como la forma más representativa del SRFC. Su objetivo es encontrar usuarios que sean lo más similares posible a una persona en concreto y, a partir de ahí, analizar sus opiniones sobre un determinado ítem. Si estas opiniones son positivas, se recomienda dicho objeto al usuario objetivo.

Los pasos para realizar este filtrado son los siguientes:

1. Creación de la matriz con los datos disponibles:
   - Las filas representan a los usuarios.
   - Las columnas corresponden a los ítems a recomendar.
   - Las celdas contienen la valoración que cada usuario ha dado a cada ítem.
2. Identificación de los usuarios más similares al usuario objetivo. Para ello, se emplea una medida de similitud. Normalmente, se suelen utilizar las siguientes medidas:
   - Fórmula de la correlación de Pearson:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.1}$$

- Fórmula de la similitud del coseno:

$$Sim_{cos}(u,n) = \frac{\sum_{i \subset CR_{u,n}} r_{ui}\, r_{ni}}{\sqrt{\sum_{i \subset CR_{u,n}} r_{ui}^2}\, \sqrt{\sum_{i \subset CR_{u,n}} r_{ni}^2}} \tag{2.2}$$

3. Selección de los usuarios más similares:
   - Una vez que se conoce cuáles son los usuarios más similares al sujeto de estudio, se debe elegir un número limitado de ellos con los que trabajar. Normalmente, por razones de eficiencia, solo interesa trabajar con un grupo reducido de usuarios, los más similares al usuario objetivo.
4. Predicción de la valoración:
   - Una vez seleccionados estos vecinos similares al sujeto objetivo, se realiza una predicción sobre cuánto le podría gustar un objeto al usuario. Se basará en las puntuaciones dadas por los otros usuarios. Para ello, se tiene en cuenta tanto la media de las puntuaciones dadas a ese ítem por los usuarios vecinos como la similitud entre cada vecino y el usuario objetivo. Para ello, se puede utilizar la siguiente fórmula:

$$p_{u,i} = s_u + \frac{\sum_{v \in V} sim(u,v)\,[s_v(i) - s_v]}{\sum_{v \in V} |sim(u,v)|} \tag{2.3}$$

5. Ordenación y recomendación:
   - Finalmente, se ordenan las predicciones generadas y se le ofrece al usuario aquellas que se consideran más interesantes.

El segundo método es el item-item [11]. Este método tiene una premisa similar a la anterior, pero con una diferencia clave: en lugar de basarse en la similitud entre usuarios, se basa en la similitud entre los productos. Si a un usuario le gusta un producto X, entonces, se supone que también le gustará un producto similar a X. Los pasos para realizar este filtrado son los siguientes:

1. Creación de la matriz:
   - Al igual que en el método user-user, lo primero es crear una matriz organizando los datos con los que se va a trabajar:
     - Las filas representan a los usuarios.
     - Las columnas corresponden a los ítems.
     - Las celdas contienen la valoración que cada usuario ha dado a cada ítem.
2. Identificación de los ítems más similares:
   - A diferencia del método user-user, que mide la similitud entre todos los usuarios y el usuario objetivo, en este caso, se busca conocer la similitud entre todos los pares de ítems. Es decir, se determina cuán similares son los productos en función de las calificaciones de los usuarios. Al igual que en el caso anterior, las fórmulas más utilizadas son la correlación de Pearson y la similitud del coseno (ver Ecuaciones 2.1 y 2.2).

Figura 2.1: SRFC, Fuente: consultar Apéndice C

3. Predicción para los ítems:

   - Una vez que se conoce cuáles son los ítems más similares entre sí, se debe buscar cuáles son los ítems más similares a aquellos que el usuario ya ha valorado anteriormente. De esta forma, se puede crear una lista de los ítems más parecidos a los que el usuario ya ha valorado.

4. Evaluación de la similitud global:

   - Sin embargo, un producto puede ser similar a uno que le gustó al usuario, pero muy diferente de otro que también le gustó. Por lo tanto, existe la necesidad de revisar cada producto que el usuario no ha valorado anteriormente y verificar cuán similar es con todos los productos que sí ha valorado, eligiendo los más similares en general. Se utiliza la siguiente fórmula para calcular la predicción final:

$$r_{u,i} = \frac{\sum_{j \in S(i)} sim(i,j) \cdot r_{u,j}}{\sum_{j \in S(i)} |sim(i,j)|} \tag{2.4}$$

5. Ordenación y recomendación:

   - Finalmente, se ordenan los productos según su valor de predicción y se presentan al usuario las recomendaciones más relevantes.

El último método es la factorización matricial [11], que puede entenderse como una combinación de los dos enfoques anteriores. La lógica detrás de este método es sencilla: en función de ciertos criterios, se elige aplicar una u otra estrategia de recomendación. La forma más intuitiva de explicarlo es a través de un ejemplo. Contamos con dos modelos: user-user e item-item y, dependiendo de un umbral predefinido, se decide cuál aplicar. Por ejemplo, si un usuario tiene más de 10 recomendaciones conocidas, se le aplica el modelo user-user; en caso contrario, se utiliza el modelo item-item. (ver Figura 2.1)

Figura 2.2: SRBC, Fuente: consultar Apéndice C

2.1.2. SRBC

Un método de SRBC mediante etiquetas es un sistema que sugiere productos tomando como referencia las características asociadas a dichos productos. En este caso, las etiquetas asignadas a cada producto buscan describir sus atributos más relevantes. Se parte de la premisa de que un usuario tiene preferencia por productos con ciertas características o etiquetas. Por lo tanto, el objetivo es identificar qué productos comparten esas características, con el fin de recomendarlos. Para lograrlo, se utilizará el historial de productos que previamente le han gustado al usuario. En resumen, la idea es que si a un usuario le gustó el producto X, es probable que también le guste el producto Y, siempre que ambos compartan características similares (ver Figura 2.2) [11].

Para crear este sistema, en su forma más elemental, se recurre al algoritmo TF-IDF [11]. Este método busca determinar cuán importante es un término dentro de un conjunto de documentos; en este caso, se trata de evaluar la relevancia de cada etiqueta en un conjunto de elementos. Las etiquetas que aparezcan en pocos elementos serán consideradas características distintivas y relevantes de aquellas en las que sí estén presentes. El primer paso para poder realizar este algoritmo será obtener una matriz de perfiles de producto. Este perfil de producto ayudará a calcular el valor real, es decir, el peso que tiene cada etiqueta de cada objeto, para conocer si debemos recomendarla o no a un determinado usuario. Este proceso de creación tiene varios pasos intermedios para completarse.

- Cálculo de TF [11].
    - La matriz de TF (frecuencia de término) indica cuántas veces una etiqueta ha sido asignada a un producto. Este valor resulta útil porque permite evaluar la relevancia de una etiqueta, ya sea por su alta frecuencia en un producto específico o por su aparición en múltiples productos. Una etiqueta puede estar concentrada en un único producto o distribuida entre varios; en cualquier caso, este será el primer valor a calcular. La frecuencia se obtiene

contando cuántas veces una etiqueta aparece asociada a cada producto.

$$TF_{p,e} = \#\{\text{asignaciones de la etiqueta } e \text{ al producto } p\}$$

- Cálculo de IDF [11].
  - Como ya se ha mencionado, el valor de TF puede estar muy concentrado en un único producto o distribuido uniformemente entre varios. Esta situación puede generar problemas al realizar recomendaciones, ya que no todas las etiquetas aportan el mismo nivel de información. Por ello, es necesario medir cuán rara o específica es una etiqueta dentro del conjunto de productos. Si no se aplicara esta corrección, las etiquetas presentes en la mayoría de los productos tendrían una sobrerrepresentación, lo cual reduciría la calidad de las recomendaciones. Gracias a esta medida, se puede dar mayor peso a aquellas etiquetas que realmente aportan información distintiva. Esta corrección se realiza mediante el cálculo del IDF (frecuencia inversa de documento), definido como:

$$IDF_e = \log \frac{|productos|}{|productos_e|} \tag{2.5}$$

    donde $|productos|$ es el número total de productos del sistema y $|productos_e|$ es el número de productos a los que se ha asignado la etiqueta *e* al menos una vez.

- Cálculo de TF-IDF [11].
  - Conociendo la utilidad de ambas variables, TF e IDF, el siguiente paso es combinarlas. El objetivo de esta combinación es asignar un peso a cada etiqueta, de modo que las más comunes tengan una menor influencia en el proceso de recomendación, mientras que las más distintivas adquieran mayor relevancia. Esta ponderación se calcula mediante la fórmula del TF-IDF, que se expresa como:

$$TF - IDF_{p,e} = TF_{p,e} \cdot IDF_e \tag{2.6}$$

    Una vez realizado este proceso, se habrá generado un perfil completo para cada producto. No obstante, aún queda un último paso por llevar a cabo. Aunque ya se han ponderado los valores mediante TF-IDF (dando mayor peso a las etiquetas más raras y reduciendo la influencia de las más comunes), sigue existiendo el riesgo de que los productos con muchas etiquetas, especialmente si son populares y tienen etiquetas muy variadas, dominen sobre el resto. Para evitar este desequilibrio, se aplica un proceso de normalización, cuyo objetivo es equilibrar la magnitud de los vectores de cada producto y asegurar una comparación justa entre ellos.

$$p'_{p,e} = \frac{p_{p,e}}{\sqrt{\sum_{e_2} p^2_{p,e_2}}} \tag{2.7}$$

- Perfil usuario [11].

- Una vez finalizado el perfil del producto, el siguiente paso será construir el perfil del usuario. El primer aspecto a considerar en este proceso es entender cómo puntúa el usuario. Analizar su forma de valorar los elementos no solo permite identificar cuáles son sus gustos (es decir, qué objetos le gustan o no), sino también comprender su criterio de puntuación y su nivel de exigencia. Este análisis se realiza de la siguiente manera:

$$w_{u,p} = r_{u,p} - \bar{r}_u \tag{2.8}$$

donde $r_{u,p}$ es la calificación del usuario u sobre el producto p y $\bar{r}_u$ es la valoración media del usuario. Una vez realizado este cálculo, el siguiente paso es determinar cómo valora el usuario la presencia de cada una de las etiquetas. Conociendo cuáles son sus etiquetas preferidas, se puede identificar las recomendaciones más adecuadas para él. Para ello, se utilizará la relación entre el perfil de los productos y las valoraciones del usuario, estableciendo un vínculo entre usuarios y etiquetas a través de las películas que ha puntuado:

$$p_{u,e} = \sum_{producto=1}^{n} p'_{p,e} \cdot w_{u,p} \tag{2.9}$$

- Recomendación [11].

  - Una vez generados el perfil del usuario y el perfil de las películas, el siguiente paso consiste en analizar cómo interactúan para realizar la recomendación. Este proceso se lleva a cabo utilizando la similitud del coseno, una técnica ampliamente empleada en SR [11]. La razón es sencilla: esta medida permite calcular el grado de similitud entre dos vectores (en este caso, el perfil del usuario y el de cada elemento), lo cual ayuda a identificar qué elementos son más parecidos a los gustos previamente expresados por el usuario. La fórmula utilizada es la siguiente:

$$\cos(P_u, P_i) = \frac{\sum_e p'_{u,e} \cdot p'_{i,e}}{\sqrt{\sum_e (p'_{u,e})^2} \cdot \sqrt{\sum_e (p'_{i,e})^2}} \tag{2.10}$$

2.1.3. Comparativa de SR

Una vez se han expuesto las bases en las que se fundamentan los diferentes tipos de SR, se estudiarán las ventajas y desventajas que uno presenta frente a otro.

El SRFC, destaca por su capacidad de encontrar patrones complejos de preferencias de usuarios, sin necesidad de conocer las características propias del producto, lo que termina consiguiendo que el usuario acceda a contenido que originalmente no había ni siquiera considerado, debido a la diversidad del mismo. Desgraciadamente, el SRFC presenta el problema del arranque en frío. Es decir, si el número de datos de usuarios e ítems es escaso, las recomendaciones serán bastante ineficientes. Sin embargo, los problemas no se presentan solo en caso de escasez de datos, en caso de excesiva abundancia, los cálculos computacionales pueden llegar a ser bastante

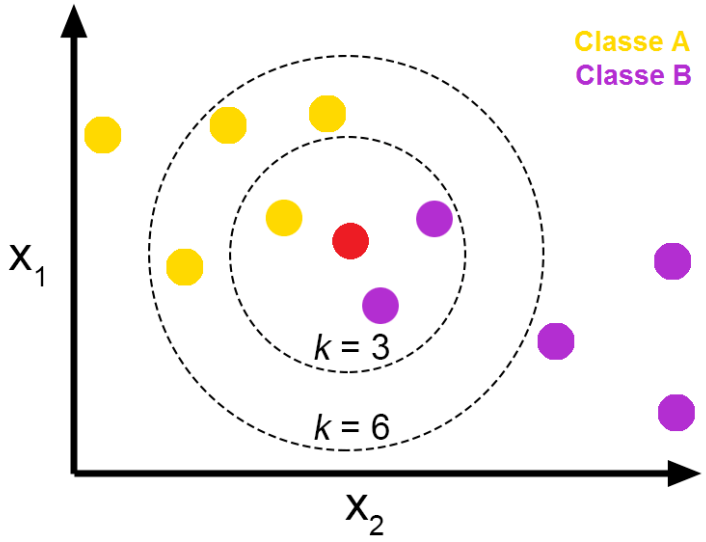


Figura 2.3: KNN, Fuente: consultar Apéndice C

costosos. Finalmente, otro inconveniente que presenta, es que los ítems que tienden a ser más populares suelen ser más recomendados.

Por otro lado, los SRBC sirve como contraposición a los SRFC. En primer lugar, al no depender de otros usuarios, funciona bastante bien para nuevos productos, evitando el problema del arranque en frío. Sin embargo, pueden producir problemas de *burbujas*, pues al usar las características de los ítems que ya le gustaban al usuario, tienden a no descubrir nuevos ítems. Otro problema que presentan es que necesitan de una descripción de las características del producto. Para concluir, al igual que el SRFC, presenta una mala escalabilidad.

Hoy en día, la mayoría de SR son mixtos, es decir, utilizan una combinación de ambos enfoques para poder obtener una mejor recomendación. Normalmente, para productos nuevos se suelen emplear SRBC, mientras que para recomendaciones con BBDD ya construidas, se suelen emplear SRFC [11].

2.1.4. Machine learning

La tecnología avanza y con la llegada de las técnicas de machine learning, muchos problemas clásicos se han resuelto de forma más eficiente. Los SR no son una excepción, pues existe una amplia lista de librerías con algoritmos de machine learning que pueden ayudar a realizar estas recomendaciones de manera mucho más eficiente.

Algunos de los algoritmos más utilizados en esta área y que producen muy buenas recomendaciones son los siguientes:

- KNN [11] (ver Figura 2.3)
    - Es un algoritmo de aprendizaje supervisado que se basa en el principio de similitud entre usuarios, también conocido como el principio del vecino más

cercano. Comúnmente para medir esa similitud se suelen utilizar algoritmos de medida como la distancia Euclídea o el coseno. En lo referente a los SR, se pueden encontrar principalmente dos implementaciones del mismo, de tipo user-based y de tipo item-based.

- En el caso de user-based, lo que se pretende es encontrar usuarios similares al usuario objetivo y recomendarle un producto que haya gustado a dichos usuarios similares. Se utiliza para crear un SRFC user-user.
- Por otra parte, el tipo item-based, busca encontrar similitudes entre ítems que hayan gustado al usuario y otros ítems que el usuario no haya valorado, con el fin de recomendar estos últimos. Se suele utilizar para la implementación de un SRFC ítem-ítem.

- SVD [11] (ver Figura 2.4)
  - SVD es una técnica de factorización de matrices. Su principio es descomponer una matriz de gran dimensionalidad en matrices más pequeñas para, de esta forma, captar patrones latentes. Se suelen utilizar para la creación de SRFC basados en filtrado matricial. Además, también permite predecir calificaciones de un usuario para un ítem que aún no ha consumido.

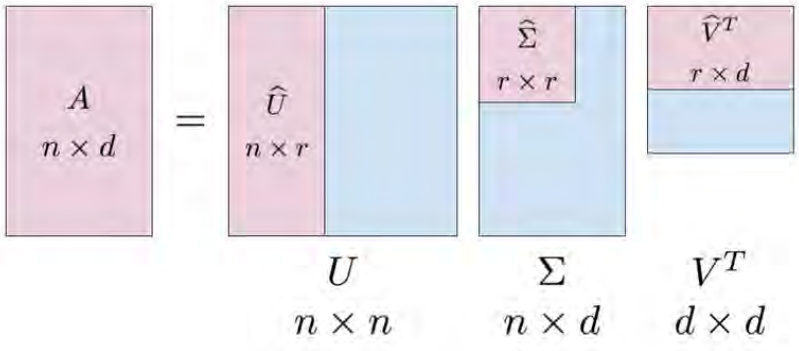


Figura 2.4: SVD, Fuente: consultar Apéndice C

- Transformers [11, 12] (ver Figura 2.5)
  - Los transformers se definen como modelos de lenguaje profundo con el fin de procesar secuencias de datos. Lo más característico de estos sistemas es su mecanismo de atención, que identifica los datos más relevantes para hacer predicciones. De esta manera, consiguen captar relaciones complejas entre elementos de manera eficiente.
  - Su funcionamiento es el siguiente. Se convierte cada ítem de la secuencia en un embedding, un vector numérico que representa sus características o significado. Para cada ítem, el modelo evalúa su importancia respecto al resto, obteniendo los elementos que considera más importantes.
  - En lo referente a los SR, se utiliza especialmente para analizar los historiales del usuario. Se convierte cada acción del usuario en un embedding, se evalúa la relación entre los mismos y se obtiene una recomendación. Principalmente, se utiliza en los SRBC.

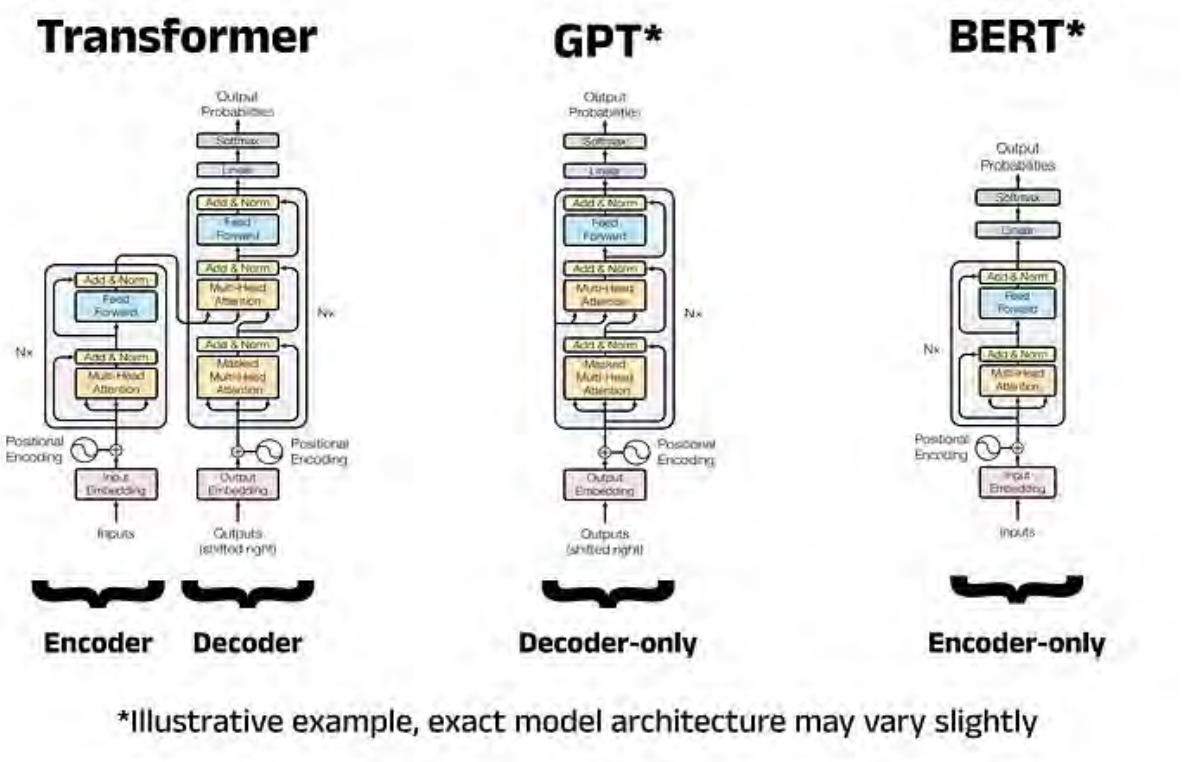


Figura 2.5: Diferencia entre transformer y BERT, Fuente: consultar Apéndice C

- BERT [11, 13] (ver Figura 2.5)
  - Un BERT se puede definir como un transformer entrenado de una manera específica, principalmente para trabajar con PLN (procesamiento de lenguaje natural).
  - Se caracteriza principalmente por su mecanismo de atención bidireccional para comprender el contexto de cada palabra dentro de una frase, lo que le permite obtener todo el contexto de una palabra, obteniendo así ciertos matices de la misma.
  - También se emplea en SRBC, de forma similar al resto de transformers.

2.2. RV

La RV se puede definir como toda tecnología que permite a los usuarios interactuar con entornos digitales tridimensionales, de manera inmersiva, haciendo creer al usuario de esta manera, “estar presentes”, en un espacio simulado. Esta tecnología combina hardware especializado, como cascos o gafas de RV, sensores de movimiento y controladores, con software capaz de crear escenarios interactivos y realistas [14].

La RV es una tecnología que tiene su origen en el siglo XIX. En aquel entonces, el término RV era muy amplio, ya que englobaba todo elemento que crease artificialmente una ilusión de la realidad. Principalmente, abarcaba espectáculos de carácter

circense, como puede ser un juego de espejos. Sin embargo, la tecnología empleada en aquella época difiere mucho de la actual. Los inicios de lo que hoy en día conocemos como RV surgen alrededor de 1960. En esta época aparatos como Sensorama o el HMD, empiezan a aparecer y establecer las bases de la RV que actualmente conocemos. Como punto importante se debe destacar que las tendencias actuales buscan: la mejora de la resolución visual, la reducción de la latencia, la integración de la inteligencia artificial para experiencias adaptativas y la combinación con otras tecnologías inmersivas como la realidad aumentada (RA) y la realidad mixta (RM) [15, 16].

Uno de los acontecimientos más importantes relacionados con esta tecnología tuvo lugar cuando la empresa anteriormente conocida como Facebook decidió cambiar su nombre al actual, Meta, en el año 2021, poniendo de manifiesto su deseo de centrarse principalmente en la creación del metaverso. El metaverso es un proyecto que intenta crear un entorno virtual compartido, persistente y tridimensional, en el que las personas puedan interactuar entre sí, trabajar, jugar, asistir a eventos y consumir contenido digital mediante avatares. En otras palabras, es un universo virtual conectado donde se combinan RV, RA y redes sociales [15, 16].

Finalmente, es conveniente resaltar que el principal uso que esta tecnología presenta a día de hoy es lúdico, más específicamente su gran nicho de usuarios son jugadores de videojuegos, por lo que no es de extrañar que la mayoría de la industria que trabaja con RV actualmente se centre principalmente en ese sector. A continuación, se presentará la información necesaria para que el lector conozca las principales tecnologías relacionadas con la RV, incluyendo tanto los dispositivos utilizados como los motores principales sobre los que se han desarrollado.

2.2.1. Dispositivos

La RV ha avanzado constantemente y no ha parado de desarrollarse desde sus inicios. Actualmente, existen gran variedad de dispositivos, con sus pros y sus contras. En lo fundamental, no suelen ser demasiado dispares. Las diferencias más significativas están ligadas a los controladores y a las empresas que los desarrollan. Los controladores son todos aquellos periféricos, aparte de las gafas, que utiliza el usuario para comunicarse con el entorno virtual que lo rodea. Los controladores más comunes son los joysticks que permiten a los usuarios moverse por la escena. Cuanto más numerosos y más complejos son los controladores, más diversas podrán ser las experiencias. Aparte de los controladores, también existen ciertas diferencias en lo que respecta a potencia gráfica y al sistema operativo. Cada gafa usa su propio sistema operativo, aunque en la práctica la mayoría son versiones adaptadas de sistemas ya conocidos como Android [17].

A continuación, se presentarán algunos de los dispositivos más importantes actualmente (ver Tabla 2.1).

Una vez presentada la tecnología actual en lo referente a las gafas, falta conocer los motores gráficos que se usan para modelar los entornos virtuales. Es importante remarcar que las gafas de RV son el periférico donde se visualiza la escena, no el

Modelo	Ventajas	Inconvenientes	Precio (€)
Google Cardboard VR	Económicas y compatibles con diferentes móviles	Experiencia limitada comparada con otras gafas de RV	21,90 (Amazon)
HTC VIVE Focus 3	Experiencia muy inmersiva y compatibles con SteamVR y Meta	Difíciles de configurar	1.559 (PcComponentes)
Sony PlayStation VR2	Buena calidad y pantallas OLED	Aunque son compatibles con SteamVR, es necesario comprar un adaptador caro	445 (Amazon)
Meta Quest 3S	Precio más asequible que otros modelos de Meta	Calidad de imagen inferior a otros modelos de Meta	439 (Amazon)
Meta Quest Pro	Excelente calidad y experiencia de juego	Precio	549,99 (Meta)

Tabla. 2.1: Comparativa de modelos de gafas de RV

creador de la misma. Los motores gráficos que se utilizan para trabajar con RV son plataformas de desarrollo de software que permiten crear entornos 3D interactivos, integrando gráficos, físicas, audio y controladores. Teniendo en cuenta el mercado de las gafas RV, los principales motores de este tipo están especializados en la creación de videojuegos. Los más destacados son:

- Unity [18]
  - Descripción
    - Unity es la plataforma líder mundial para crear y operar contenido interactivo en tiempo real 3D. Creadores, que van desde desarrolladores de juegos hasta artistas, arquitectos y diseñadores automotrices hasta cineastas y más, utilizan Unity para dar vida a sus imaginaciones. La plataforma Unity proporciona un conjunto completo de soluciones de software para crear y operar contenido en tiempo real 2D y 3D para múltiples plataformas, incluidos teléfonos móviles, tabletas, PC, consolas y dispositivos de realidad aumentada y virtual.
  - Pros
    - Fácil de usar, fuerte apoyo de la comunidad, gratuito para uso personal, interfaz intuitiva y aprendizaje sencillo.
  - Contras
    - Problemas de rendimiento, precios demasiados altos para empresas pequeñas, suscripciones caras, errores de software o incapacidad de trabajar con Git sin corrupción son algunos de sus problemas. Sobre esto último se hará un pequeño énfasis. Si bien Unity permite trabajar con plataformas como GitHub, lo cierto es que gran la cantidad de elementos que un proyecto Unity usa para la creación de la escena, creándose y destruyéndose continuamente, causa que irremediablemente en

algún momento exista una corrupción del proyecto Git. Ante este problema, Unity ha creado su propia plataforma para trabajar de forma remota que sustituye a Git. Sin embargo, la versión gratuita está muy limitada y casi es obligatorio usar la versión de pago, lo que muchos usuarios consideran abusivo.

- Unreal Engine [18]
  - Descripción
    - Fundada en 1991, Epic Games es la creadora de las series de juegos Unreal, Gears of War e Infinity Blade. Hoy en día, Epic está desarrollando Paragon, Fortnite, SPYJiNX y el nuevo Unreal Tournament. La tecnología Unreal Engine de Epic es utilizada por equipos de todos los tamaños para lanzar juegos y experiencias visualmente impresionantes y de alta calidad en plataformas de PC, consola, RV y móviles. Los desarrolladores también eligen Unreal Engine para visualización, diseño, entretenimiento lineal y simulación.
  - Pros
    - Es multiplataforma, gran realismo en el entorno 3D, los precios en caso de comercialización no son abusivos y el lenguaje de programación en el que se basa, c++, está muy extendido.
  - Contras
    - El tamaño de los proyectos suele ser bastante grande, problemas de optimización y la comunidad no es tan grande como en otros motores.
- 3ds Max Design [18]
  - Descripción
    - El software 3ds Max Design es una solución integral de diseño, modelado, animación y renderizado en 3D para arquitectos, diseñadores, ingenieros civiles y especialistas en visualización. Este motor gráfico no está pensado para el desarrollo de videojuegos.
  - Pros
    - Creación de diseños muy realistas, gran versatilidad, animaciones de muy alta calidad.
  - Contras
    - Curva de aprendizaje difícil, altos requerimientos de hardware, fallos y retrasos frecuentes.

2.3. Docker

Docker [19] es una plataforma de código abierto que da a los usuarios la capacidad de crear y gestionar contenedores y ejecutar aplicaciones en ellos. Coloquialmente,

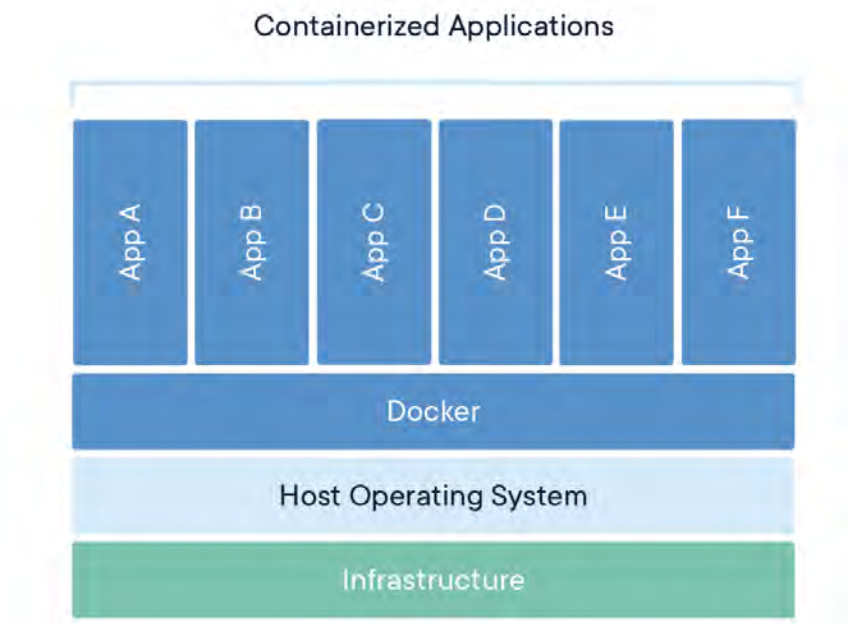


Figura 2.6: Arquitectura Docker, Fuente: consultar Apéndice C

recibe el nombre de Docker, debido al nombre de la empresa que ha desarrollado la tecnología, Docker Inc.

Los contenedores Docker son componentes estandarizados que contienen como elementos principales: el código de la aplicación que se desea ejecutar, bibliotecas del propio sistema operativo y las dependencias necesarias para ejecutar el mismo. Ahora bien, toda la tecnología de contenedores se sustenta en la capacidad de virtualización integrada en un núcleo de Linux. Gracias a esto, se pueden asignar recursos entre procesos. Esta gestión de recursos se asemeja bastante a como un hipervisor permite que varias máquinas virtuales compartan el uso de la CPU (ver Figura 2.6).

Es gracias a esta virtualización, que los contenedores Docker ofrecen todas las ventajas y funciones que se pueden encontrar de forma habitual en las máquinas virtuales. Sin embargo, no se limita únicamente a proporcionar aislamiento de recursos y escalabilidad rentable, entre otras ventajas, podemos destacar [19, 20]:

- Menor peso: Los contenedores destacan por tener únicamente, además del código a ejecutar, los procesos del sistema operativo y las dependencias necesarias para ejecutar dicho código. Los que hace a estos contenedores sean más ligeros que las máquinas virtuales, que suelen tener una instancia completa del sistema operativo. Por ejemplo, el tamaño de una máquina virtual suele medirse en gigabytes, mientras que el de un contenedor se mide en megabytes.

- Mejora de la productividad: Una característica de las aplicaciones en contenedores, es que se pueden escribir una vez y ejecutarse en cualquier lugar. Si hacemos una comparativa, son más rápidos y sencillos de gestionar que las máquinas virtuales.

- Mayor eficiencia: Los contenedores tienen la capacidad de ejecutar varias copias de una aplicación en el mismo hardware. Esto permite reducir gastos en la nube.

Es por todo esto que se puede afirmar que los contenedores simplifican el desarrollo y entrega de aplicaciones, sobre todo en el ámbito distribuido. Esto, unido al paso de muchas organizaciones a trabajar en la nube, ha implicado que los contenedores sean una herramienta bastante utilizada y famosa actualmente.

Finalmente, cabe destacar el papel de Docker en este proyecto, ya que se ha optado por su uso para llevar a cabo el despliegue del sistema. Aunque es posible crear contenedores directamente mediante herramientas nativas de Linux, Docker facilita notablemente este proceso al permitir la creación y gestión de contenedores de forma rápida y sencilla. Además, su amplia adopción en el sector, con una cuota de mercado superior al 80 %, lo convierte en una solución consolidada y adecuada para el desarrollo y despliegue de aplicaciones como la propuesta en este trabajo.

Capítulo 3

DISEÑO INICIAL

3.1. Especificaciones del sistema

El objetivo principal de este apartado es definir las especificaciones del sistema. Estas especificaciones sirven como referencia para comprobar si el desarrollo, la validación y la evaluación del sistema se han llevado a cabo de forma adecuada, y constituyen la base sobre la que se sustenta la propuesta. Una vez establecidos los objetivos, las especificaciones permiten evaluar de manera objetiva si estos se han cumplido correctamente.

Existen dos tipos de especificaciones. Los requisitos funcionales y los no funcionales. Los requisitos funcionales representan las acciones que el sistema debe realizar. Mientras que los requisitos no funcionales indican en qué condiciones deben cumplirse dichas acciones.

3.1.1. Requisitos funcionales

Los requisitos funcionales, especificados por el cliente, son los siguientes:

- #RF01: El sistema debe ofrecer recomendaciones utilizando un SRFC.
- #RF02: El sistema debe ofrecer recomendaciones utilizando un SRBC.
- #RF03: El sistema deberá mostrar la cartelera actual de los cines.
- #RF04: Nuevos usuarios deben de poder registrarse en la aplicación.
- #RF05: Usuarios ya registrados podrán autenticarse en su cuenta.
- #RF06: Debe existir la posibilidad de entrar como invitado a la aplicación.
- #RF07: El usuario debe de ser capaz de valorar películas.

- #RF08: La cartelera debe mostrar el nivel de recomendación de una película.
- #RF09: La cartelera debe mostrar el porqué se ha realizado esa recomendación.
- #RF10: El frontend de la aplicación debe diseñarse en un sistema de RV.
- #RF11: El sistema debe de dockerizarse.
- #RF12: Al registrarse, cada usuario debe de valorar 20 películas por defecto.

3.1.2. Requisitos no funcionales

Los requisitos no funcionales de este proyecto son los siguientes:

1. #RNF01: Rendimiento.
    - El sistema, en lo referente al backend, deberá ser capaz de generar las recomendaciones tanto con un SRFC como con un SRBC, en un tiempo razonable (menos de 2 segundos).
    - La renderización de la escena en el frontend deberá realizarse en un tiempo razonable (menos de 7 segundos).
    - Se usará de forma eficiente el uso de la memoria del sistema. Intentando evitar los bloqueos o caídas del sistema.
2. #RNF02: Fiabilidad y disponibilidad
    - Se debe asegurar la disponibilidad del backend en todo momento. Se considera aceptable un mínimo de disponibilidad del 95 %.
    - El sistema debe de gestionar de forma adecuada los errores de conexión para no perjudicar la experiencia del usuario.
    - En caso de error de conexión con la API de la cartelera, se deberán de usar los datos internos del sistema para no perjudicar la experiencia del usuario.
3. #RNF03: Escalabilidad y mantenibilidad
    - La arquitectura del sistema deberá permitir la incorporación de nuevos algoritmos de recomendación sin necesidad de modificar la estructura central del backend.
    - El código del backend y del frontend deberá estar modularizado para facilitar la ampliación del proyecto y el mantenimiento futuro.
    - La estructura de la BBDD debe estar preparada para la incorporación de nuevos atributos o elementos.
4. #RNF04: Seguridad
    - El sistema deberá validar toda entrada procedente del usuario y del cliente RV para prevenir errores o comportamientos inesperados.

- El sistema responderá únicamente a los mensajes que cumplan con la estructura correcta y definida por el equipo de desarrollo, cualquier mensaje que no cumpla este formato será descartado al considerarse sospechoso.
- El sistema no debe de revelar ninguna información considerada sensible de los usuarios registrados.

5. #RNF05: Usabilidad

   - La interfaz en RV deberá ser intuitiva, de fácil navegación y con elementos interactivos claramente identificables.
   - El sistema debe de proporcionar una cierta retroalimentación a las acciones del usuario.

6. #RNF06: Compatibilidad

   - El frontend deberá funcionar en las gafas de RV seleccionadas durante la fase de investigación, cumpliendo los requisitos técnicos de Unity.
   - El backend deberá ser compatible con la infraestructura Docker definida en el proyecto, permitiendo su despliegue en distintos entornos sin modificaciones.

7. #RNF07: Portabilidad

   - El sistema, una vez dockerizado, deberá de poder ser instalado en cualquier servidor.
   - Los distintos componentes que forman el sistema (contenedores dockers y frontend) deberán ser independientes entre sí para facilitar su despliegue y migración.

8. #RNF08: Documentación

   - Deberá elaborarse una documentación técnica que describa la arquitectura, dependencias, instalación y funcionamiento del sistema.
   - El manual de usuario deberá permitir que cualquier persona, sin conocimientos previos, pueda utilizar el sistema SCENE.

3.2. Análisis y diseño del sistema

Una vez presentadas las especificaciones del sistema, en este apartado se analizan las necesidades identificadas y se plantea un diseño inicial que permita dar respuesta a dichas necesidades.

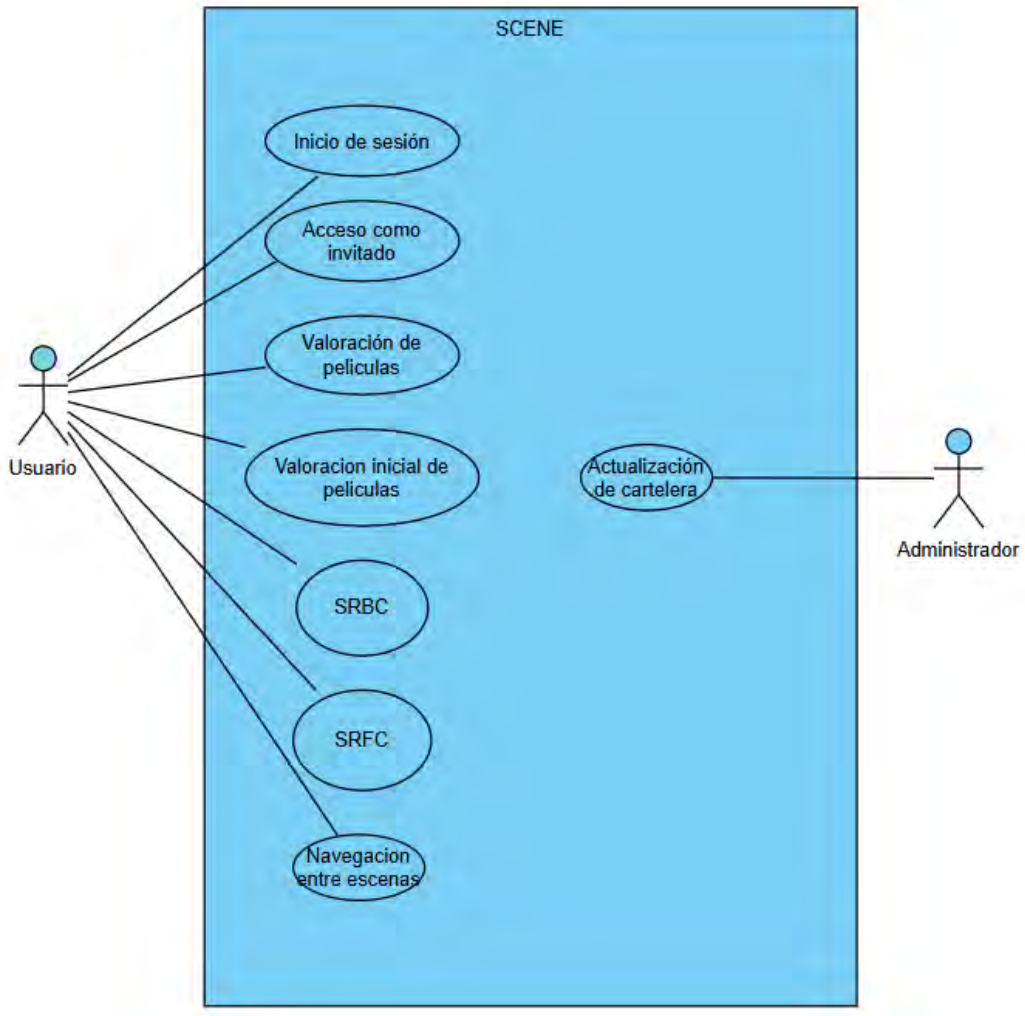


Figura 3.1: Caso de uso: general

3.2.1. Casos de uso

Los casos de usos se definen para poder establecer, de manera clara y orientada al usuario final de la aplicación, su interacción con la misma. Este apartado servirá de puente entre los requisitos del proyecto y su posterior implementación.

Diagrama general de casos de uso

En la Figura 3.1 podemos ver el diagrama de casos de uso general del sistema.

Lista de casos de uso

En este apartado se listará, de forma ordenada, cada caso de uso. Cada uno deberá incluir no solo el flujo de acciones, sino también cierta información adicional que

ayude a entender mejor el contexto de la aplicación. Indicar que, dada su simplicidad, no se han incluido los diagramas de caso de uso para cada caso.

1. Registro en el sistema
    - Actor: Usuario.
    - Objetivo: Crear una cuenta en la aplicación.
    - Contexto: Un usuario no autenticado desea registrarse en la aplicación.
    - Precondiciones: El usuario no dispone de una cuenta previamente registrada en el sistema.
    - Condiciones de éxito: La cuenta del usuario queda creada correctamente y el sistema devuelve la información inicial asociada.
    - Flujo principal:
        a) El usuario interactúa con el teclado en el entorno de RV.
        b) El usuario introduce su nombre de usuario y contraseña.
        c) El usuario envía la información pulsando la opción de registro.
        d) El sistema comprueba que el nombre de usuario no existe previamente.
        e) El sistema registra al usuario en caso de que no exista.
        f) El sistema selecciona un conjunto inicial de 40 películas de forma aleatoria.
        g) El sistema devuelve al usuario su identificador y el conjunto de películas seleccionadas.
2. Inicio de sesión
    - Actor: Usuario.
    - Objetivo: Acceder a la aplicación utilizando sus credenciales.
    - Contexto: Un usuario registrado desea autenticarse en el sistema.
    - Precondiciones: El usuario dispone de una cuenta previamente registrada y no se encuentra autenticado.
    - Condiciones de éxito: El usuario queda autenticado correctamente y el sistema devuelve su identificador.
    - Flujo principal:
        a) El usuario interactúa con el teclado en el entorno de RV.
        b) El usuario introduce su nombre de usuario y contraseña.
        c) El usuario envía la información seleccionando la opción de inicio de sesión.
        d) El sistema valida las credenciales introducidas.
        e) El sistema autentica al usuario y devuelve su identificador.
3. Acceso como invitado
    - Actor: Usuario.

- Objetivo: Acceder a la aplicación sin necesidad de registro previo.
- Contexto: Un usuario desea utilizar la aplicación sin crear una cuenta permanente.
- Precondiciones: El usuario no se encuentra autenticado en el sistema.
- Condiciones de éxito: El usuario accede a la aplicación como invitado y el sistema genera un identificador temporal.
- Flujo principal:
  a) El usuario interactúa con el teclado en el entorno de RV.
  b) El usuario selecciona la opción de acceso como invitado.
  c) El sistema crea un usuario temporal.
  d) El sistema autentica al usuario como invitado y devuelve su identificador.

4. Valoración de películas

   - Actor: Usuario.
   - Objetivo: Asignar una valoración numérica a una película.
   - Contexto: Un usuario autenticado desea valorar una película disponible en la aplicación.
   - Precondiciones: El usuario se encuentra autenticado y la película seleccionada existe en el sistema.
   - Condiciones de éxito: La valoración queda correctamente registrada y asociada al usuario y a la película.
   - Flujo principal:
     a) El usuario interactúa con la interfaz de la película para seleccionar una puntuación.
     b) El usuario confirma la valoración mediante la opción de envío.
     c) El sistema registra la valoración asociándola al usuario y a la película.
     d) El sistema muestra una notificación indicando que la valoración se ha registrado correctamente.

5. Valoración inicial de películas

   - Actor: Usuario autenticado (registrado o invitado).
   - Objetivo: Realizar la valoración inicial de un conjunto de películas para generar un perfil de preferencias.
   - Contexto: El usuario accede por primera vez al sistema tras el registro o acceso como invitado.
   - Precondiciones: El usuario se encuentra autenticado y dispone de un conjunto inicial de películas asignadas por el sistema.
   - Condiciones de éxito: El usuario completa la valoración de al menos 20 películas y el sistema permite continuar al siguiente escenario.
   - Flujo principal:

a) El sistema presenta al usuario el conjunto inicial de películas.
b) El usuario selecciona y registra la valoración de las películas mostradas.
c) El sistema contabiliza el número de valoraciones realizadas.
d) Una vez alcanzadas 20 valoraciones, el sistema avanza automáticamente al siguiente escenario.

6. Obtención de recomendaciones desde la sala de cine (SRFC)

   - Actor: Usuario autenticado (registrado o invitado).
   - Objetivo: Obtener una recomendación de películas de la BBDD mediante filtrado colaborativo.
   - Contexto: El usuario accede a la sala de cine.
   - Precondiciones: El usuario se encuentra autenticado y dispone de valoraciones previas suficientes para aplicar el SR
   - Condiciones de éxito: El sistema genera y presenta al usuario un conjunto de 10 películas recomendadas.
   - Flujo principal:
     a) El usuario accede a la escena de la biblioteca.
     b) El sistema genera automáticamente una recomendación utilizando el sistema de filtrado colaborativo.
     c) El sistema presenta al usuario el conjunto de películas recomendadas.

7. Obtención de recomendaciones de cartelera (SRBC)

   - Actor: Usuario autenticado (registrado o invitado).
   - Objetivo: Obtener una recomendación de películas en cartelera mediante un SRBC.
   - Contexto: El usuario accede a la sección de cartelera de la aplicación.
   - Precondiciones: El usuario se encuentra autenticado y dispone de un perfil de preferencias generado a partir de valoraciones previas.
   - Condiciones de éxito: El sistema genera y presenta al usuario un conjunto de películas recomendadas de la cartelera.
   - Flujo principal:
     a) El usuario accede a la escena de cartelera.
     b) El sistema genera automáticamente una recomendación utilizando el sistema basado en contenido.
     c) El sistema presenta al usuario el conjunto de películas recomendadas.

8. Navegación entre escenas

   - Actor: Usuario autenticado (registrado o invitado).
   - Objetivo: Desplazarse entre las diferentes escenas del entorno de RV.
   - Contexto: El usuario se encuentra interactuando con el entorno inmersivo de la aplicación.

- Precondiciones: El usuario ha accedido correctamente al sistema mediante registro, inicio de sesión o acceso como invitado.
- Condiciones de éxito: El sistema carga correctamente la escena seleccionada por el usuario.
- Flujo principal:
  a) El usuario se desplaza dentro del entorno de RV hasta una zona de transición señalizada.
  b) El sistema detecta la interacción del usuario con la zona de transición.
  c) El sistema carga la escena correspondiente.

9. Actualización de la cartelera

- Actor: Administrador.
- Objetivo: Actualizar la información de las películas en cartelera del sistema.
- Contexto: El administrador necesita sincronizar la cartelera local con una fuente externa.
- Precondiciones: El sistema se encuentra detenido o fuera de servicio para los usuarios.
- Condiciones de éxito: La cartelera queda actualizada correctamente en el sistema.
- Flujo principal:
  a) El administrador inicia el proceso de actualización de la cartelera.
  b) El sistema obtiene la información actualizada desde la fuente externa correspondiente.
  c) El sistema actualiza la información de la cartelera almacenada localmente.

3.2.2. Diagramas de secuencia

En este apartado se presentan los diagramas de secuencia utilizados durante el desarrollo del proyecto. El objetivo de estos diagramas es representar de forma clara el flujo de mensajes y la interacción que se produce entre los distintos actores y componentes software implicados en el sistema, permitiendo así comprender el comportamiento dinámico de la aplicación. En el contexto de este proyecto, se consideran tres actores principales: el usuario, el frontend y el backend, que intervienen de manera coordinada en la ejecución de las distintas funcionalidades.

Con el fin de facilitar la comprensión y la visualización de estas interacciones, se ha elaborado un diagrama de secuencia específico para cada caso de uso previamente definido. De esta forma, se puede analizar de manera individual el comportamiento del sistema en cada escenario, mostrando con mayor detalle cómo se intercambia la información y cómo se ejecutan las acciones necesarias para cumplir los objetivos del sistema.

Registro en el sistema (Figura 3.2)

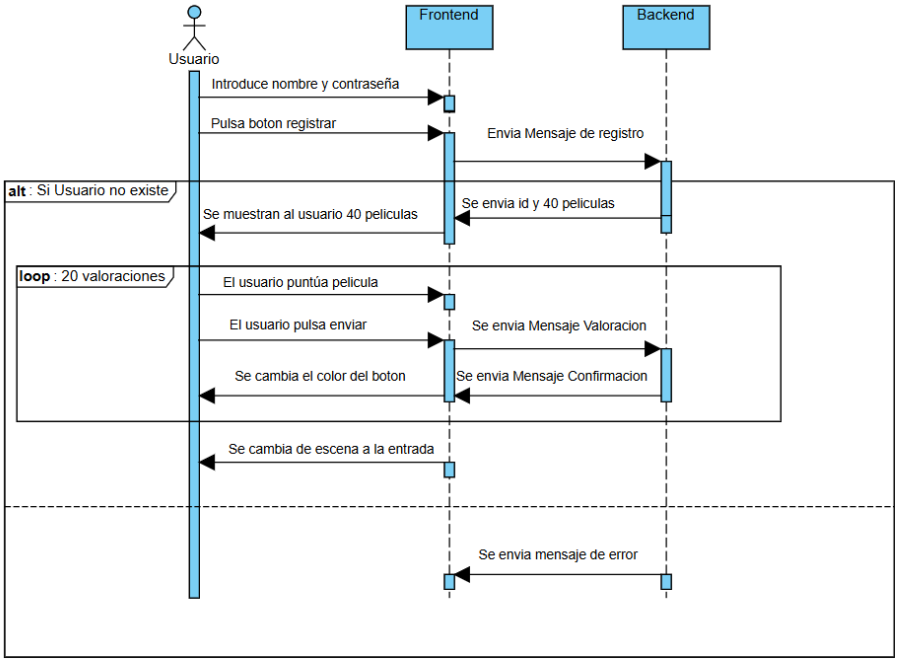


Figura 3.2: Diagrama de flujo: registro

Inicio de sesión (Figura 3.3)

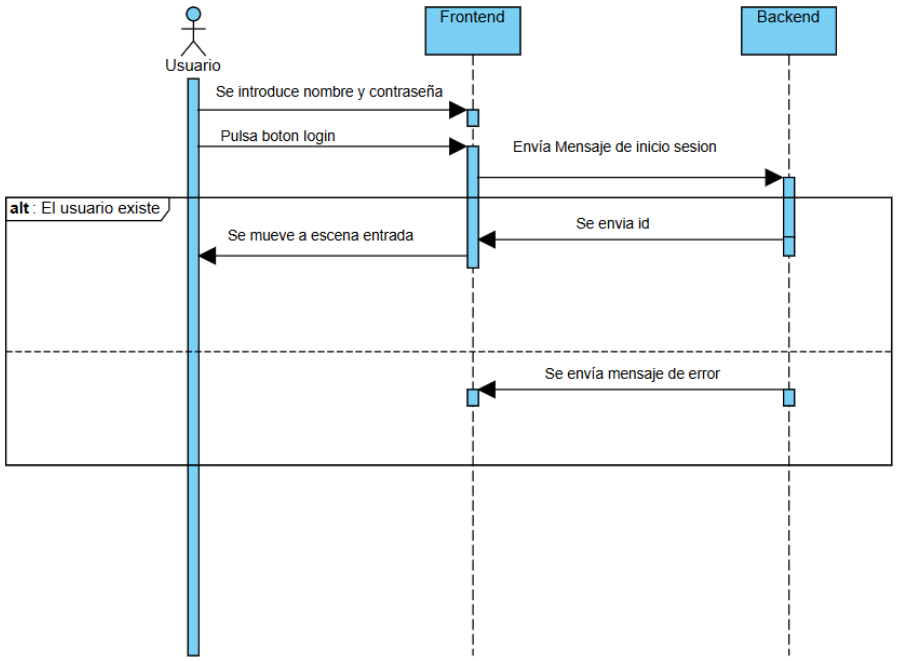


Figura 3.3: Diagrama de flujo: login

Acceso como invitado (Figura 3.4)

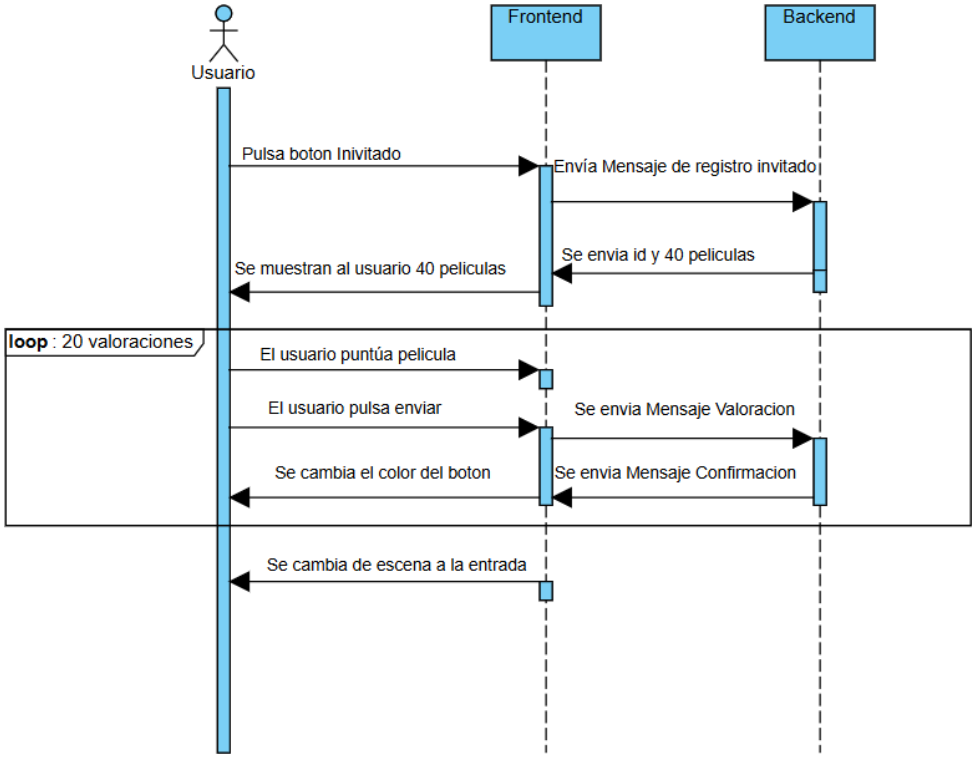


Figura 3.4: Diagrama de flujo: invitado

Valoración de películas (Figura 3.5)

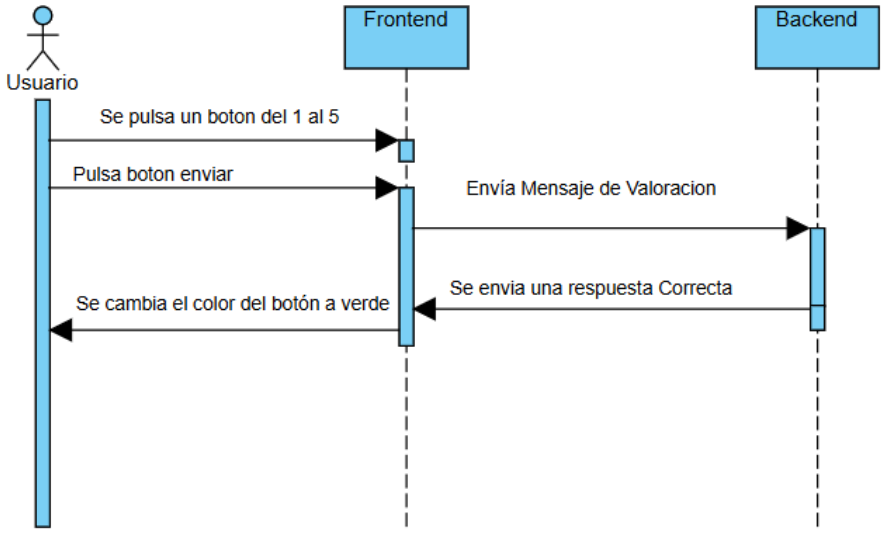


Figura 3.5: Diagrama de flujo: valorar película

Valoración inicial de películas (Figura 3.6)

La valoración de las 20 películas presenta exactamente el mismo flujo que la valoración de 1 película, repetido 20 veces.

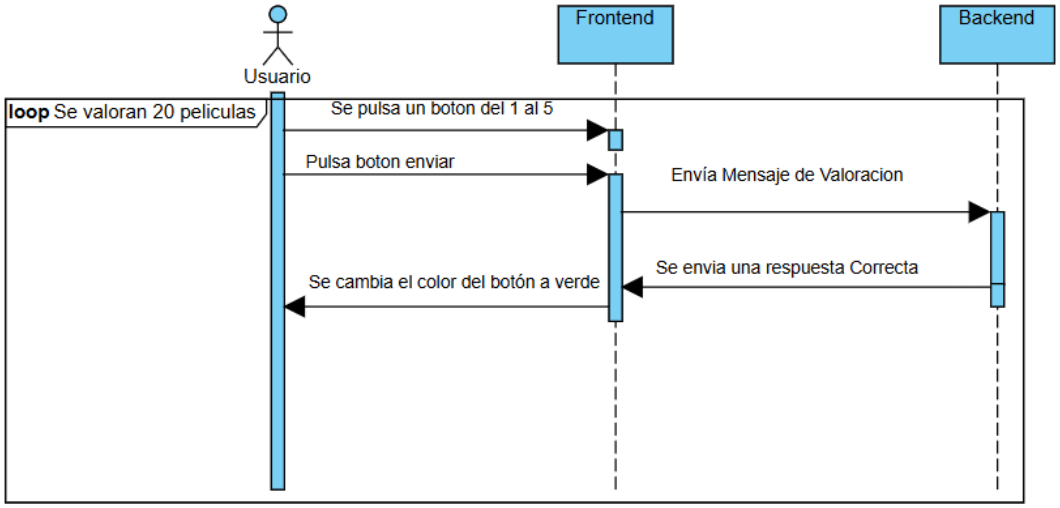


Figura 3.6: Diagrama de flujo: valorar 20 películas

Obtener recomendaciones de biblioteca (SRFC) (Figura 3.7)

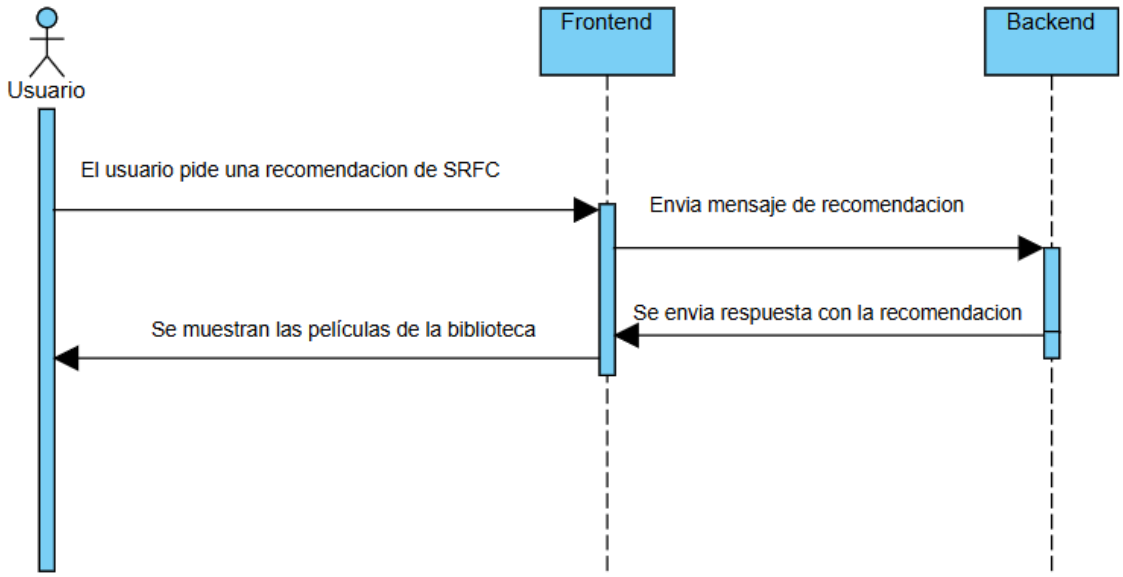


Figura 3.7: Diagrama de flujo: SRFC

Obtener recomendaciones de cartelera (SRBC) (Figura 3.8)

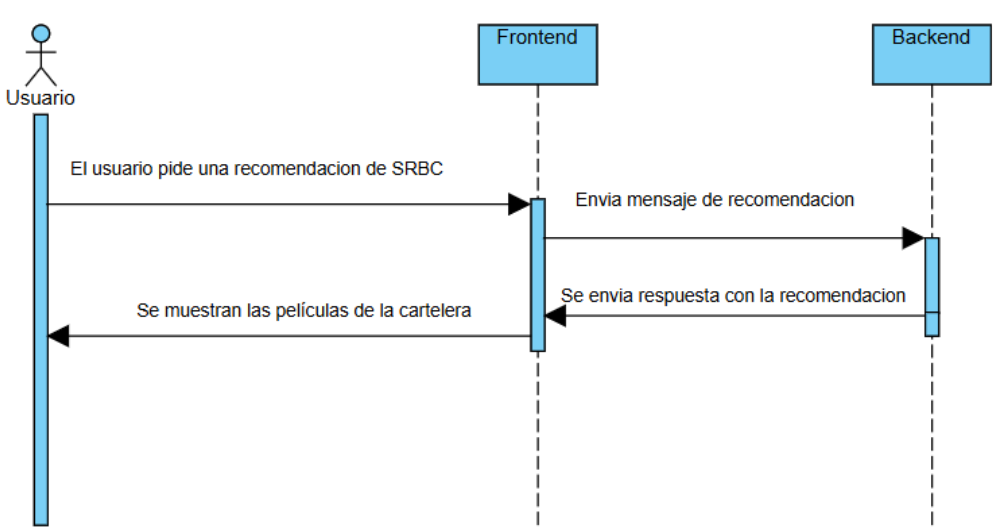


Figura 3.8: Diagrama de flujo: registro

Navegar entre escenas (Figura 3.9)

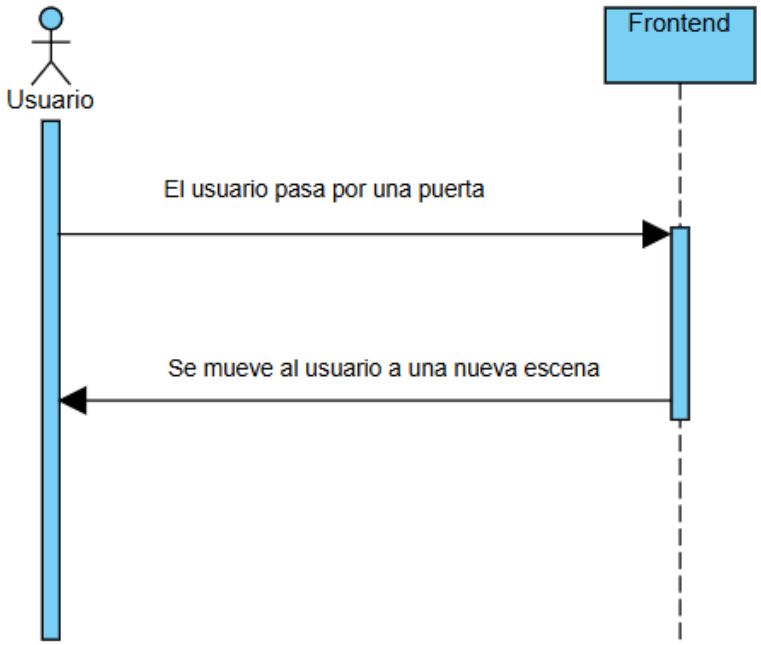


Figura 3.9: Diagrama de flujo: navegar entre escenas

Actualizar cartelera (Figura 3.10)

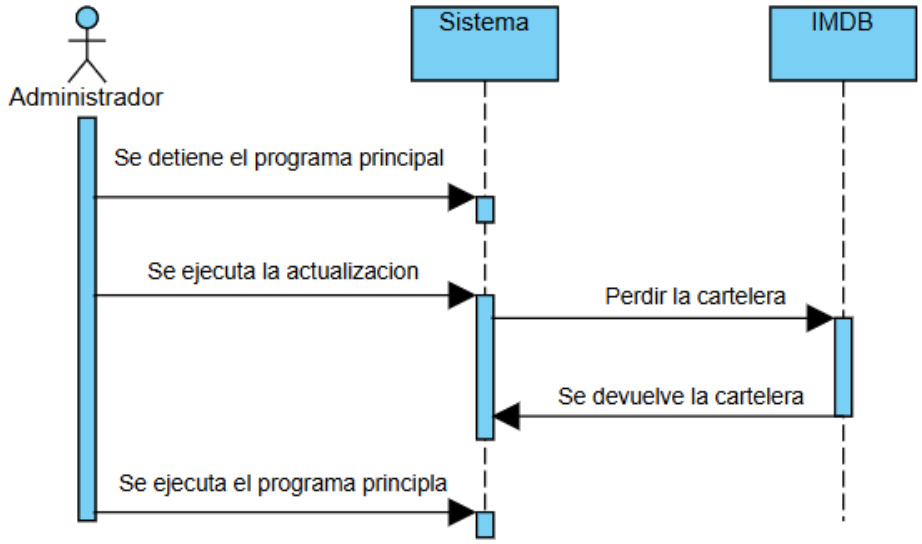


Figura 3.10: Diagrama de flujo: Actualizar cartelera

3.2.3. Diseño de la BBDD

Uno de los elementos más relevantes del sistema es la BBDD. No obstante, aunque las tareas de búsqueda del conjunto de datos (#ALD) y de preprocesamiento (#PREP) han sido definidas previamente, la estructura final de la BBDD no puede determinarse con exactitud hasta que dichas tareas se completen. Aun así, debido a la naturaleza del proyecto y a los requisitos mínimos necesarios para su correcto funcionamiento, es posible establecer una serie de condiciones generales que la BBDD deberá cumplir.

En cuanto a la estructura del sistema, inicialmente se decidió trabajar con una BBDD almacenada de forma local, de manera que la aplicación tenga siempre acceso a la información necesaria sin depender de peticiones externas. Esta decisión garantiza una mayor estabilidad y reduce la dependencia de servicios de terceros. No obstante, esta situación presenta una excepción en el caso de la funcionalidad relacionada con la cartelera actual, ya que, para mantenerla actualizada, es necesario consultar una API externa.

Aunque las peticiones a dicha API son imprescindibles para obtener la información más reciente, los datos recuperados se almacenan posteriormente en un archivo local. Esta decisión responde a la posibilidad de que el servicio externo no esté disponible temporalmente, lo que podría afectar al funcionamiento de la aplicación. Al mantener una copia local de la cartelera, el sistema puede seguir ofreciendo recomendaciones y funcionalidades básicas, aun cuando la información no esté completamente actualizada. Una vez restablecida la comunicación con la API, el fichero local se actualiza con los nuevos datos, asegurando así la continuidad del servicio.

Otro factor clave del proyecto es el formato de almacenamiento de la BBDD. En sistemas de este tipo es habitual emplear BBDD relacionales o soluciones NoSQL. Sin embargo, en este caso se ha optado por el uso de archivos de texto, decisión que responde a una serie de motivos que se detallan en el Capítulo 4. El lenguaje de programación seleccionado para el desarrollo del sistema es Python, que dispone de bibliotecas ampliamente utilizadas en ámbitos como la ciencia de datos, el análisis de datos y el aprendizaje automático. Entre ellas destaca pandas, cuya funcionalidad principal se basa en la lectura, gestión y manipulación eficiente de datos mediante estructuras denominadas DataFrames.

La relevancia de esta elección radica en que la biblioteca utilizada para la implementación del SR, Cornac, está preparada para trabajar directamente con estructuras de tipo DataFrame. Esto implica que, independientemente del formato original de almacenamiento, los datos deberían transformarse a este tipo de estructura para su procesamiento. Además, con el objetivo de ofrecer un servicio eficiente y evitar accesos innecesarios al almacenamiento, los datos se cargan una única vez durante la inicialización del sistema, realizándose posteriormente múltiples operaciones de lectura y escritura en memoria a lo largo de su ejecución. En este contexto, pandas ofrece un rendimiento adecuado, permitiendo operaciones rápidas sin introducir una carga adicional significativa.

Teniendo en cuenta estos aspectos, la incorporación de una capa adicional de almacenamiento, como una BBDD relacional o NoSQL, resultaría innecesaria y añadiría complejidad al sistema sin aportar beneficios claros. Asimismo, la estructura de los conjuntos de datos utilizados es relativamente simple, lo que refuerza la idoneidad de una solución ligera. Por todo ello, se ha optado por emplear archivos de texto como método de almacenamiento, ya que ocupan poco espacio y permiten una lectura y escritura eficientes, constituyendo una solución sencilla y adecuada para las necesidades de este proyecto.

En lo referente a los documentos que se van a necesitar y las características de los mismos. Se deberán de establecer unas características mínimas que deben de incluir.

- users
  - Este documento representa a los usuarios.
  - Cada usuario debe tener la siguiente información.
    - id: Identificador del usuario.
    - nombre: Nombre del usuario.
    - contraseña: Contraseña del usuario.
- items
  - Este documento representa la información de las películas.
  - Cada película debe tener la siguiente información.
    - id: Identificador de la película.
    - title: Título de la película.

    - tags: Características de la película. Indican características como el género de la película: Acción, aventuras entre otras y además algunas características adicionales como el director, actores o elementos descriptivos de la misma.
    - cartelera: Booleano que indique si la película se encuentra o no en la cartelera.

- ratings

    - Representan las valoraciones que los usuarios han hecho de las distintas películas.
    - Su estructura es la siguiente:
        - idUsuario: Identificador del usuario.
        - idItem: Identificador de película.
        - valor: Valoración que el usuario ha hecho de la película.

3.2.4. Diseño de la interfaz

En este apartado se presentan bocetos del diseño preliminar presentar de la interfaz de la aplicación. Es importante recalcar que la aplicación emplea un entorno inmersivo, lo que implica que los diseños se tendrán que hacer, no solo teniendo en cuenta las distintas pantallas en las que puede estar el usuario, sino también el espacio tridimensional. Se han previsto 4 escenarios principales. El inicio, la entrada al cine, la sala de cine y el pasillo de la cartelera.

Escena 1 - Inicio

Esta escena es la primera que ve el jugador al abrir la aplicación. Representa una calle, en la que se pueda ver un cine y un teclado para introducir las credenciales del usuario (ver Figuras 3.11 y 3.12). En caso de registro de un nuevo usuario, este verá un pasillo donde aparecerán una serie de películas que el usuario debe valorar (ver Figura 3.11). En concreto, se mostrarán 40 películas elegidas aleatoriamente entre toda la BBDD y el usuario deberá introducir la valoración de 20 de forma obligatoria. De esta manera, el programa tiene una idea preliminar de los gustos del usuario.

Escena 2 - Entrada

Cuando el usuario se identifica, entra en el pasillo del cine. En ese pasillo habrá 3 puertas, una que lo lleva inicio, otra a la sala de cine y, la última, a la cartelera. A su vez, existirá un indicador que lo señale. (ver Figura 3.13)

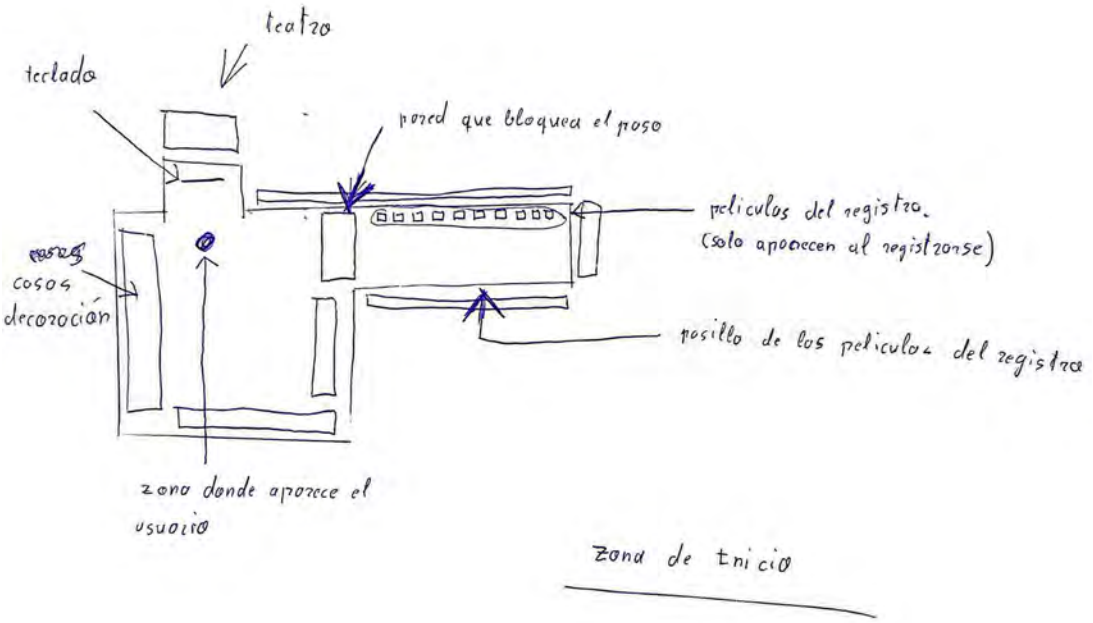


Figura 3.11: Mockup: inicio

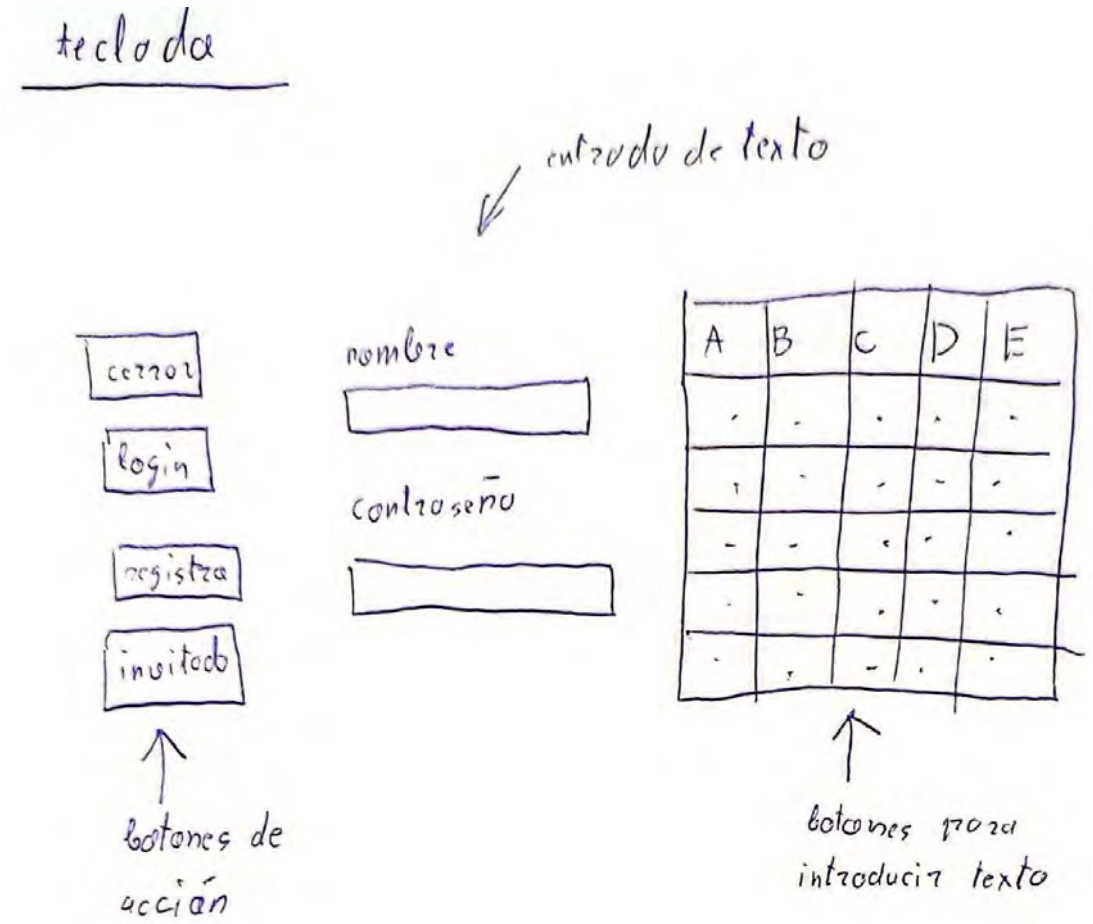


Figura 3.12: Mockup: teclado

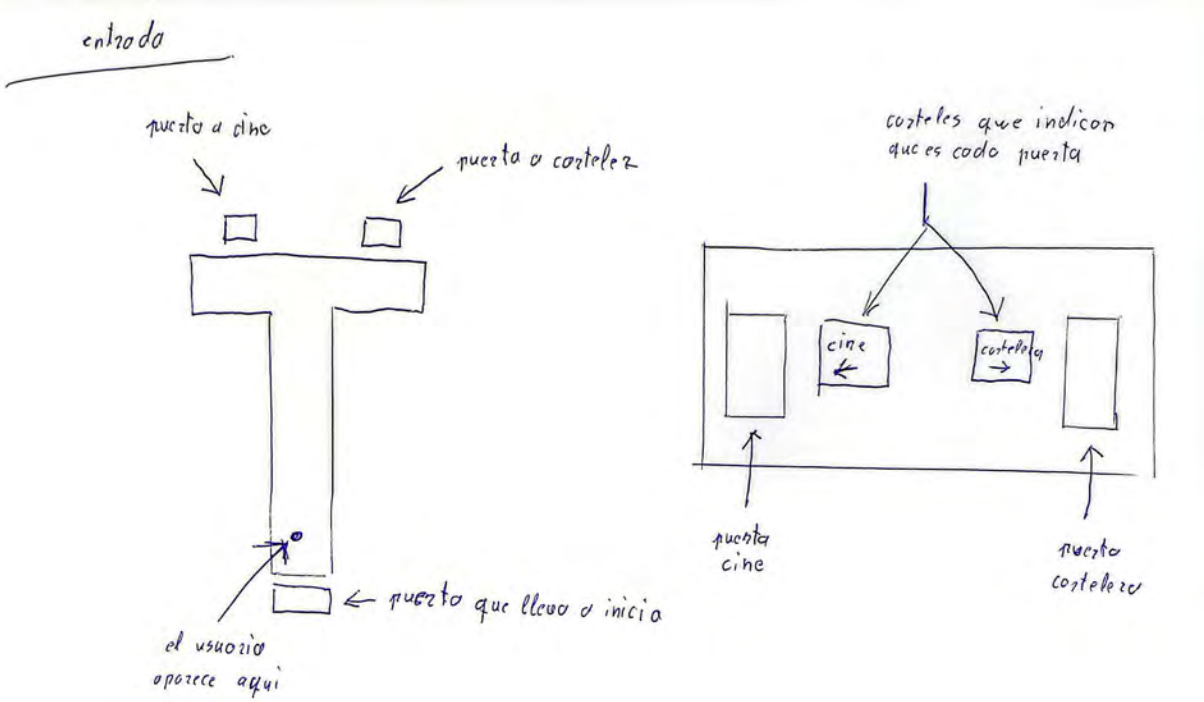


Figura 3.13: Mockup: entrada

Escena 3 - Sala de cine

Es la sala cine, donde se proyecta la película, aparecerán las películas valorar. El usuario también tiene acceso a una puerta de salida para volver al pasillo del cine. (ver Figura 3.14)

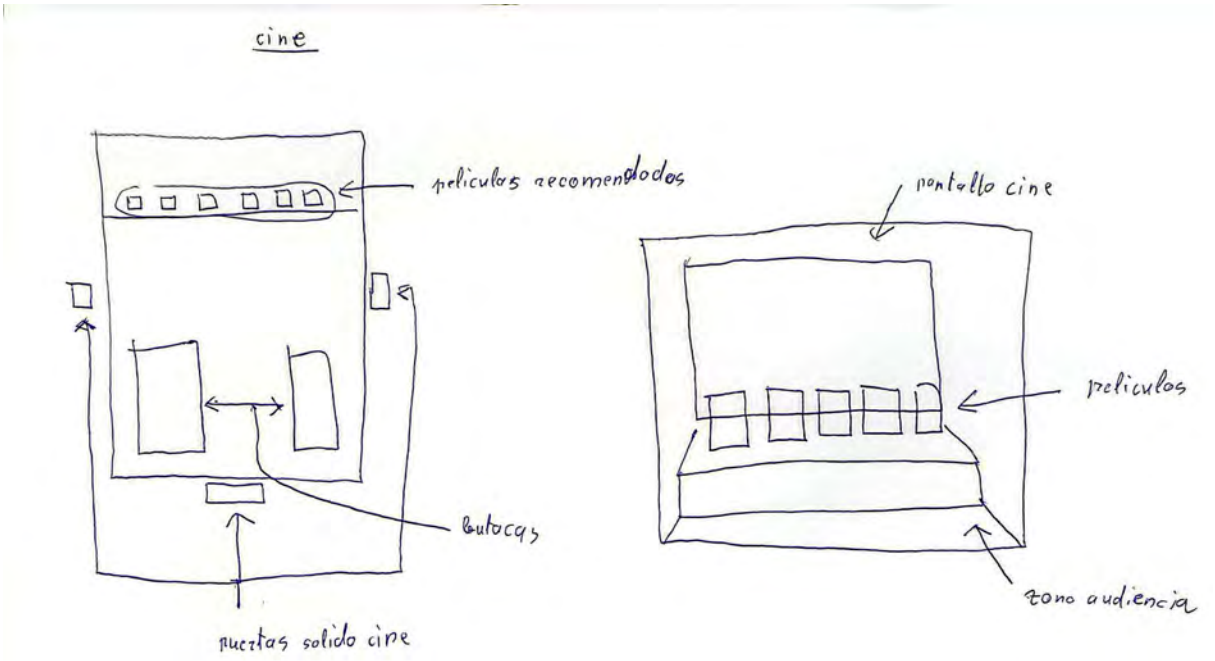


Figura 3.14: Mockup: cine

Escena 4 - Pasillo de la cartelera

Se visualiza Una sala en el que se encuentran, en línea, las películas de la cartelera. Además, el usuario tendrá acceso a una puerta con la que puede volver al pasillo de la entrada. (ver Figura 3.15)

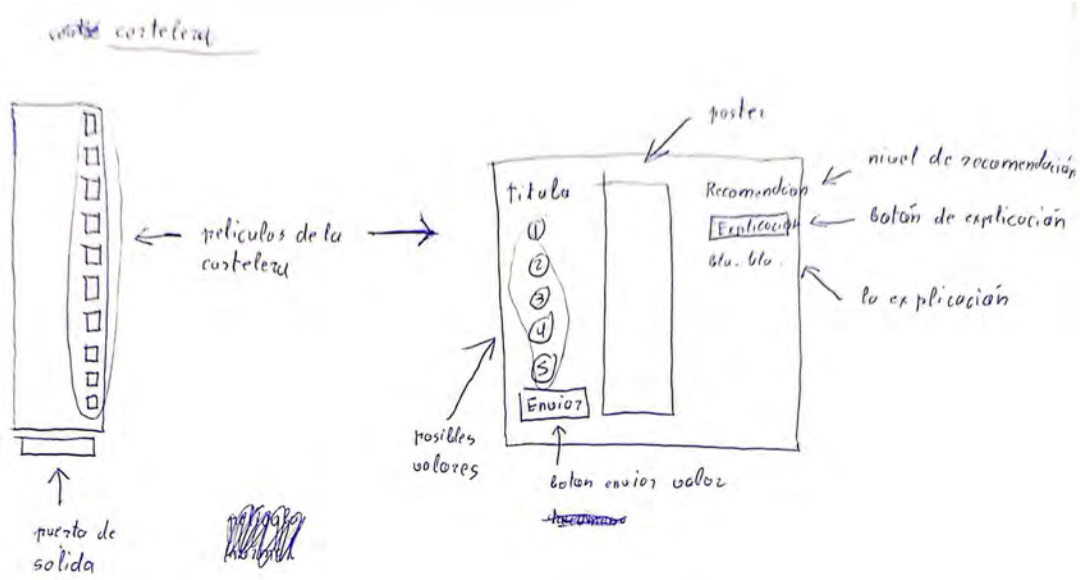


Figura 3.15: Mockup: cartelera

Por otro lado, la comunicación entre escenas, se representa en la Figura 3.16.

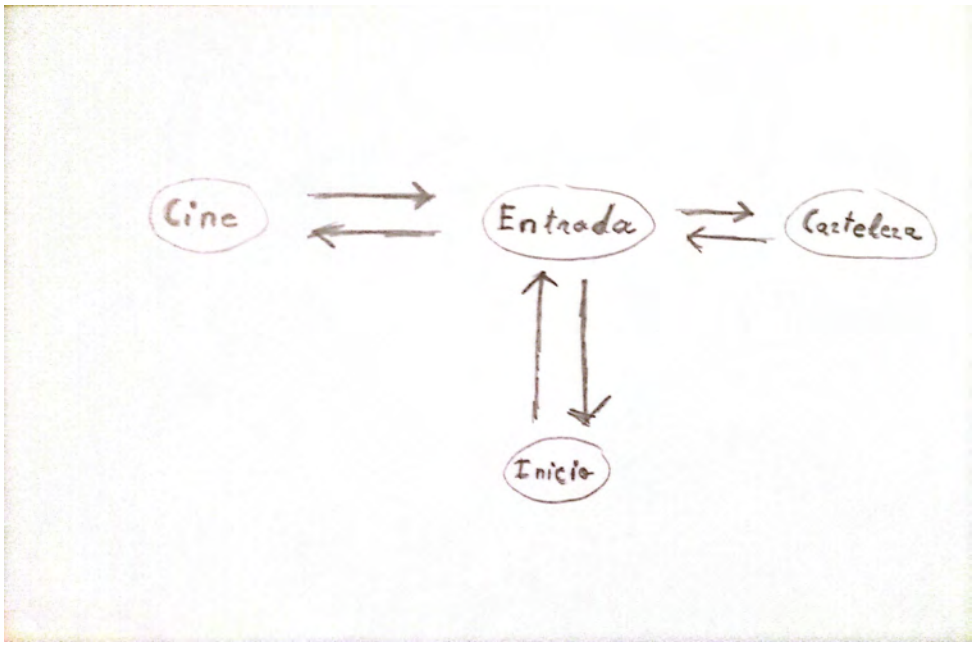


Figura 3.16: Mockup: movimiento entre escenas

Capítulo 4

DESARROLLO DEL PROYECTO

Una vez desarrollada la planificación teórica del proyecto, esta sección tiene como objetivo exponer su desarrollo práctico. Los apartados que se explorarán son los siguientes: la fase de investigación 4.1, que abarca el estudio relacionado con el backend, el frontend y la base de datos; la fase de implementación 4.2, en la que, a partir de la investigación previa, se llevará a cabo la implementación del backend y del frontend; la fase de experimentación 4.3, donde se realizarán pruebas sobre los componentes implementados; y, finalmente, la documentación y el despliegue 4.4, en la que se registrará toda la información anterior y se describirá el proceso de despliegue.

4.1. Investigación

El objetivo final de esta sección es proporcionar a los lectores un conocimiento más profundo sobre SR, RV y dockerización. Se expondrán todos los aspectos relevantes relacionados con tecnologías, técnicas y bibliotecas necesarios para completar el proyecto y tener un pleno conocimiento sobre el desarrollo. La organización de esta fase en el proyecto, queda plasmada en la Tabla 4.1

Fase	**Información / Subtareas**	**Inicio**	**Fin**
Backend	#EDA01	1/10/25	8/10/25
Frontend	#EDA02	10/10/25	15/10/25
Bases de datos	#ALD01, #ALD02, #PREP01	17/10/25	28/10/25

Tabla. 4.1: Planificación de fase de investigación

Fase	**Herramienta**
Backend	IEEE Xplore, Google Scholar, Springer Link y libros [11]
Frontend	IEEE Xplore, Google Scholar, Springer Link y libros [11]
Base de datos	Varias bases de datos C

Tabla. 4.2: Principales herramientas utilizadas

4.1.1. Backend

El fin último de esta fase es la obtención del conocimiento necesario para poder diseñar un backend que cubra todas las necesidades que el proyecto requiere.

Aunque el alumno partía de un conocimiento previo de la materia, ha sido necesaria la realización de una investigación bibliográfica por su parte. Uno de los mayores puntos de investigación se centró en las bibliotecas de algoritmos de recomendación mediante filtrado colaborativo.

Actualmente, las librerías más importantes que implementan algoritmos de recomendación están desarrolladas en Python, por lo que uno de los aspectos más interesantes del proyecto fue el uso Python como lenguaje de programación. En cuanto a las librerías, se destacan, entre otras: Surprise, TensorFlow Recommenders, PyTorch Recommenders, LightFM, libRecomender y Cornac (ver Anexo C).

Entre todas ellas, la elegida fue Cornac, por las siguientes razones. En primer lugar, Cornac poseía una biblioteca bastante amplia de algoritmos, de hecho, tenía integrados algoritmos de otras bibliotecas como PyTorch, TensorFlow o LightFM. Otra característica importante que presentaba es que, a diferencia de bibliotecas como Surprise o libRecomender, no presenta problemas de dependencias internas, ya sea con Python o con TensorFlow. Finalmente, es una biblioteca cuyo uso está muy optimizado.

Una vez definida la biblioteca de datos con la que se va a trabajar, el siguiente paso consiste en establecer los algoritmos de recomendación que se emplearán en el sistema. Tal y como se ha mencionado en los Capítulos 1 y 2, el proyecto contempla la implementación de dos enfoques distintos: un SRFC y SRBC.

En el caso del SRFC, este tipo de algoritmos se utilizan principalmente para generar recomendaciones a partir de las preferencias de otros usuarios con comportamientos similares. Tras el análisis realizado en el estado del arte, se consideraron dos alternativas principales: el uso de algoritmos basados en vecinos más cercanos (KNN) y la descomposición en valores singulares (SVD) (ver Capítulo 2). Finalmente, por motivos de rendimiento y por los resultados que ofrece en este tipo de problemas, se optó por SVD, es decir, por la implementación de un algoritmo de filtrado colaborativo de tipo matricial.

Además, una de las características determinantes del SVD frente al KNN, al menos en la implementación disponible en la biblioteca Cornac, es la posibilidad de incorporar nueva información de usuarios y películas sin necesidad de reentrenar completamente el modelo. Este aspecto resulta especialmente relevante en un sistema dinámico como el propuesto, ya que permite mejorar de forma considerable el rendimiento y la eficiencia del sistema a medida que se incorporan nuevas valoraciones.

Una vez definido el algoritmo para el SRFC, el siguiente paso consiste en seleccionar el enfoque adecuado para el SRBC. Conviene recordar que los SRBC generan sugerencias a partir de las características propias de los objetos, sin requerir valoraciones previas por parte de los usuarios. Esta característica los hace especialmente adecuados para escenarios como la cartelera, donde las películas pueden no haber

sido aún valoradas.

A partir del estudio de los métodos más utilizados en la actualidad, se observó que muchas de las implementaciones modernas de SRBC se basan en modelos Transformer. En el estado del arte (ver Capítulo 2) se ha introducido la información necesaria sobre este tipo de modelos y su funcionamiento. En este proyecto se ha optado por emplear BERT, un modelo basado en Transformers diseñado para el procesamiento bidireccional del lenguaje natural. Para generar recomendaciones, se utilizará información descriptiva de las películas, concretamente los tags asociados a cada una de ellas, junto con la información de las películas previamente vistas por el usuario, con el objetivo de considerar un perfil de preferencias que permita ofrecer recomendaciones más ajustadas.

4.1.2. Frontend

Para el desarrollo de este proyecto se han seleccionado las Meta Quest 3 de Meta como dispositivo de RV. A lo largo del trabajo se ha realizado un estudio detallado sobre el funcionamiento de estas gafas, analizando aspectos como la gestión de cuentas de usuario necesarias para su uso, la instalación de aplicaciones, los métodos de conexión entre las gafas y el ordenador, los requisitos mínimos para su correcta ejecución, así como la configuración interna del propio dispositivo. Asimismo, se ha llevado a cabo una identificación de plataformas que proporcionan assets compatibles con el desarrollo de aplicaciones de RV. Cabe recordar que los assets son recursos digitales reutilizables que se emplean durante el desarrollo de aplicaciones, tales como modelos tridimensionales, texturas, animaciones, sonidos o elementos de interfaz. Estos recursos permiten acelerar el proceso de desarrollo y mejorar el apartado visual de la aplicación sin necesidad de crear todos los elementos desde cero.

En cuanto al motor gráfico utilizado para implementar el frontend del sistema, se ha optado por Unity. Aunque este motor presenta ciertas desventajas (ver Sección 2.2.1), la elección se justifica por la experiencia previa del alumno con la herramienta, la disponibilidad de bibliotecas específicas de Meta para el desarrollo en realidad virtual, su coste gratuito y la amplia oferta de assets orientados a este ámbito. Estos factores convierten a Unity en una opción adecuada para el desarrollo del entorno inmersivo propuesto en este proyecto.

4.1.3. BBDD

Selección de datasets

La finalidad de este apartado es la de encontrar, estudiar y preparar una BBDD que cumpla todas las necesidades para la creación del SR y el desarrollo del sistema en general.

En cuanto a la búsqueda de datasets de películas, se contemplaron 3 posibles op-

ciones: uno desarrollado en la Universidad de Jaén, TMDB y MovieLens (ver Anexo C). Primeramente, se consideró un dataset ya creado por el grupo de investigación de la Universidad de Jaén Sinbad2 (TIC206). La principal ventaja era, a priori, la gran cantidad de películas que contenía, cada una de ellas, con sus descripciones e imágenes de la carátula. Además, este dataset debía tener asociado una BBDD junto con un sistema de autentificación de usuarios. Sin embargo, al estudiar la BBDD, se identificó que el número de películas era insuficiente (menos de mil películas) y no existía ninguna información sobre usuarios y tampoco un sistema de autenticación. De hecho, la propia base BBDD de películas, se obtenía a partir de la API de TMDB.

La segunda opción, TMDB, se presenta como uno de los datasets más completos sobre películas. El principal problema de este dataset viene con respecto a la información de sus usuarios. TMDB no permite obtener ninguna información de sus usuarios, como por ejemplo sus valoraciones. A la única información a la que se puede acceder es a la del propio usuario que esté registrado previamente. Una posible solución para este problema hubiese sido crear manualmente miles de cuentas “ficticias"de usuarios, usando miles de correos y guardar toda la información en BBDD físicas. Esto era posible, pero impracticable. Sin embargo, este dataset sí que permite descargar la cartelera actual de películas con: imagen, título, géneros y descripción.

Finalmente, MovieLens. Inicialmente, la opción más atractiva, ya que Cornac, no solo permite trabajar con MovieLens, sino que está optimizada para ello. Por otra parte, se encuentra en un repositorio público y con varias versiones con diferentes características, por lo que su descarga es muy sencilla. Además, un factor fundamental y decisivo, es que a diferencia de otros datasets que no ofrecen información sobre los usuarios y sus valoraciones de películas, este dataset ofrece un conjunto de usuarios creados con sus respectivos ratings sobre películas.

Por todo lo anterior, se eligió trabajar con la biblioteca de MovieLens, más concretamente con la versión de 1 millón, que incluye 1 millón de ratings, 6040 usuarios y 3952 películas, y también trabajar con TMDB para obtener información sobre la cartelera de la semana. De esta manera, se partía de una base sólida para llevar a cabo la recomendación a nuevos usuarios basada en filtrado colaborativo y se tenía disponible la información actualizada de la cartelera actual a partir de TMDB. Por lo tanto, se parte de un banco de películas con una estructura preestablecida y se va incrementando con la información de nuevas películas y de nuevos usuarios reales.

Sin embargo, aunque MovieLens incluye información de usuarios, películas y ratings, no tiene disponibles imágenes de las carátulas de las películas. Para obtener dichas imágenes, existen varios repositorios Git que permiten obtener la mayoría de imágenes de pósteres de películas y tags de cada una. Usando estos repositorios (ver Anexo C), la integración fue prácticamente inmediata. Cabe destacar que, si bien el dataset original contenía ciertos tags, estos eran muy básicos. Es por ello que se buscó un repositorio que no solo tuviera los tags básicos de las películas, sino más información de las mismas.

Ahora bien, debido a la necesidad de autentificación de cada usuario para tener recomendaciones personalizadas, fue necesario la creación de un documento que integrase los usuarios ya existentes (provenientes de MovieLens) con los nuevos usuarios.

Este documento debería tener la identificación de los nuevos usuarios considerando los ya existentes, un nombre de usuario y su contraseña.

Finalmente, el producto final es una BBDD, completamente preparada para recomendaciones por filtrado colaborativo y por recomendaciones basadas en contenidos. Además, se incluye un sistema de autentificación junto con un dataset con imágenes de películas, todo enfocado a las necesidades de la aplicación.

Estructura de la BBDD

Una vez ya establecida la BBDD con la que se va a implementar este sistema, es necesario conocer internamente como está conformada la misma.

1. **users.dat**
   - Contiene toda la información de los usuarios. Toda esta información ha sido aportada por el usuario voluntariamente y no ha sido comprobada. Aprovecharemos esto, pues para la recomendación no es necesario tener toda esta información, únicamente tener registrado el ID del usuario.
   - El formato que presenta es el siguiente: UserID::Gender::Age::Occupation::Zip-code.
     - UserID: Es el identificador del usuario.
       - Este dataset contiene un total de 6040 usuarios, los cuales están identificados por este valor entero secuencial desde el 1 hasta 6040.
     - Gender: Es el género del usuario, M es masculino y F es femenino.
       - Este valor no es relevante, pero es necesario para poder recopilar los datos de forma correcta, ya que permite identificar a los nuevos usuarios de la aplicación. En el documento, se introducirá un valor arbitrario que no desvele información personal del usuario, en este caso T.
     - Age: La edad del usuario se da por rangos:
       - 1: 18.
       - 18: 18-24.
       - 25: 25-34.
       - 35: 35-44.
       - 45: 45-49.
       - 50: 50-55.
       - 56: 56+.
     - Occupation: Indica el trabajo que tiene el individuo. Se da en estos parámetros:
       - 0: “otro” o no especificado.
       - 1: “académico/educador”.
       - 2: “artista”.

        - 3: “administrativo/oficina”.
        - 4: “estudiante universitario/de posgrado”.
        - 5: “atención al cliente”.
        - 6: “médico/profesional de la salud”.
        - 7: “ejecutivo/gerente”.
        - 8: “agricultor”.
        - 9: “amo/a de casa”.
        - 10: “estudiante de primaria o secundaria”.
        - 11: “abogado”.
        - 12: “programador”.
        - 13: “jubilado”.
        - 14: “ventas/mercadotecnia”.
        - 15: “científico”.
        - 16: “autónomo/independiente”.
        - 17: “técnico/ingeniero”.
        - 18: “artesano/obrero especializado”.
        - 19: “desempleado".
        - 20: “escritor".
    - Zip-code: Identificador del zip.

2. **movies.dat**

    - Este documento contiene toda la información de las películas con las que vamos a trabajar.
    - La estructura que contiene es la siguiente: MovieID::Title::Genres.
        - MovieID: Identificador de la película.
            - Este dataset contiene un total de 3952 películas, los cuales están identificados por este valor entero secuencial, cuyos valores son del 1 hasta 3952.
        - Title: El título de la película.
        - Genres: Los distintos géneros de la película.
            - Después de analizar todo el dataset, se comprobó que únicamente existían 18 tags distintos entre todas las películas.
    - Esta BBDD es la que obtenemos del dataset oficial de MovieLens. Se aplicarán distintas actualizaciones a la BBDD, que crecerá de forma continua incorporando nuevas películas cada semana descargadas a partir de TMDB.

3. **ratings.dat**

    - Este documento guardará toda la información de cada valoración que un usuario ha realizado sobre una película. Este archivo es especialmente importante para trabajar con el filtrado colaborativo.
    - El formato que presenta es el siguiente: UserID::MovieID::Rating::Timestamp.
        - UserIDs: Identificador de usuario.

        - Valores enteros entre 1 y 6040.
      - MovieIDs: Identificador de película.
        - Valores enteros 1 y 3952.
      - Ratings: Valoración de cada película.
        - Un valor entero comprendido entre 1 y 5.
      - Timestamp es la marca de tiempo en segundos.
        - Este valor indica el momento de tiempo en el que se introdujo esta valoración en concreto. Para el trabajo aquí presente, resulta completamente innecesario. De hecho, para no consumir tiempo de procesamiento, se introducirá un número por defecto.
      - Cada usuario tiene mínimo 20 valoraciones.
  - Esta BBDD es la que obtenemos del dataset oficial de MovieLens. Se aplicarán distintas actualizaciones a la BBDD, que crecerá de forma continua incorporando nuevas películas cada semana descargadas a partir de TMDB.

4. **moviestagsoriginal.txt**

   - Debido al limitado número de tags que presenta el documento original, solo 18, se ha optado por la utilización de un archivo adicional, que no solo contuviese los géneros de la película, también más información adicional de la misma. Con ese fin, se utiliza este archivo, que contiene mucha más información de las características de las películas y que aporta mayor precisión a los resultados obtenidos por el SRBC.
   - La estructura es la siguiente: MovieID::Title::Genres.
     - MovieID: Identificador de la película.
       - Entre 1 y 3952.
       - El dataset no está completo, presentando un total de 3883 películas en lugar de las 3952 películas originales en el dataset de MovieLens.
     - Title: El título de la película.
     - Genres: Los distintos géneros de la película.
       - Número de tags: 577.
   - Esta BBDD empieza con un número de valores determinados, como se ha indicado anteriormente. Durante las distintas actualizaciones del sistema, se añadirán de forma progresiva valores a partir de la información de películas que se obtenga de TMDB

5. **moviestags.txt**

   - Su estructura es exactamente igual al caso anterior. La única diferencia es su tamaño y propósito. Este documento representa a las películas que se encuentran en cartelera. Actualmente, son solo las 10 que se busca recomendar.
   - Toda la información está constantemente actualizada tomando la cartelera de TMDB.

6. **privados.dat**

- Este archivo contiene la información de los suscriptores a la aplicación. Se usa para identificar a los usuarios.
- La estructura es userID::nombre::contraseña.
  - userID: Corresponde al identificador de usuario.
  - nombre: Es el nombre de usuario.
  - contraseña: Es la contraseña del usuario.

El esquema relacional de la base de datos aqui planteada, quedaria de la siguiente forma (ver Figura: 4.1):

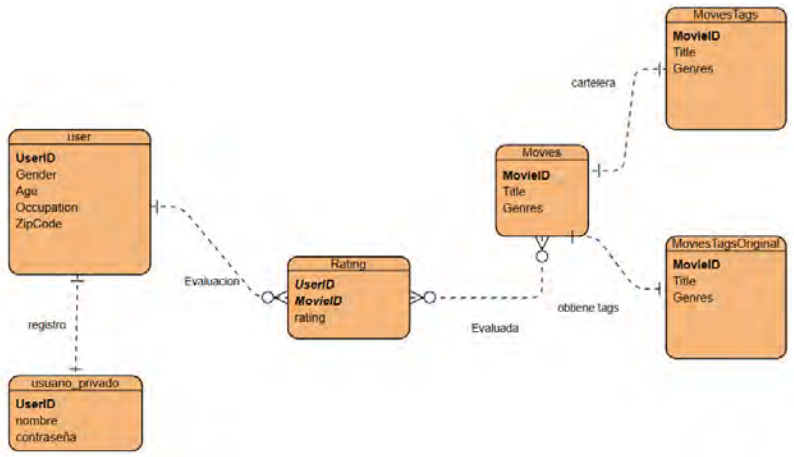


Figura 4.1: fig: esquema relacional bbdd

4.2. Implementación

La misión principal de este apartado es describir y obtener una implementación completamente funcional del sistema y preparada para realizar pruebas. Una vez terminada esta fase, tendremos: (i) un backend capaz de hacer recomendaciones de películas teniendo en cuenta todas las casuísticas dadas y (ii) un frontend de RV con el cual el usuario pueda interactuar. Destacar que, la versión que se tendrá al terminar esta fase, es un sistema que funcionará para pruebas locales, no el proyecto en despliegue. La organización de esta fase en el proyecto, queda plasmada en la Tabla 4.3.

Fase	Información / Subtareas	Inicio	Fin
Backend	#IMP01, #IMP02, #IMP03	29/10/25	25/11/25
Frontend	#IMP04	26/11/25	16/12/25
Comunicación	#IMP05	17/12/25	22/12/25

Tabla. 4.3: Planificación de la fase de Implementación

Fase	Herramienta
Backend	Visual Studio Code
Frontend	Unity, gafas meta quest 3
Comunicación	Visual Studio Code, Unity, gafas meta quest 3

Tabla. 4.4: Principales herramientas utilizadas

4.2.1. Comunicación

Uno de los puntos más importantes en el diseño de la aplicación es la estructura cliente-servidor que presenta. Con esto en mente, lo que se plantea para la implementación de esta comunicación, es establecer conexiones de tipo TCP/IP mediante el uso de sockets, por diferentes razones. Las conexiones son rápidas y las respuestas son igual de rápidas. Mediante multi-hilo, se pueden establecer varias conexiones simultáneas con el servidor. De esta forma se permiten varios dispositivos conectados de forma simultánea. Establecemos una IP y un puerto por defecto, por lo que las comunicaciones de entrada y salida en el servidor están muy controladas.

Una vez definida la comunicación, su implementación, en pseudocódigo, se representa en los Listados 4.1, y 4.2

```
conexionesBackend():
    while(true):
        if(alguienAbreConexion())
            inicializarConexionNuevoHilo()
            mensaje <- obtenerMensajeCliente()
            codigo, datos <- obtenerPeticion(mensaje)
            solucion <- obtenerSolucionPeticion(codigo, datos)
            respuesta <- CrearMensajeRespuesta(codigo, solucion)
            devolverRespuesta(respuesta)
            cerrarConexion()
```

Listado 4.1: Conexiones Backend

```
conexionesFrontEnd():
    peticion <- clienteHacePeticion()
    mensaje <- crearMensajePorPeticion(peticion)
    mandarMensaje(mensaje)
    respuesta <- respuestaDelMensaje()
    realizarAccion(respuesta)
```

Listado 4.2: Conexiones Frontend

Como se puede observar en el Listado 4.1, el código, en Python, hace de servidor de tal manera que se encuentra en un bucle infinito y cada vez que el cliente de Unity intenta conectarse, se abrirá un hilo de conexión. En ese hilo se obtendrá el mensaje del cliente, se procesa de forma interna y se le enviará la respuesta necesaria. Acto seguido se cierra la conexión. Por parte de Unity, el proceso no es muy complejo. Unity hace una petición, y cuándo envía el mensaje, espera la respuesta y, cuando la obtiene, continúa.

Un punto importante a destacar es que Unity siempre enviará peticiones en formato

texto. Sin embargo, el servidor puede proporcionar dos tipos de respuestas, tanto en texto como en imagen, en este caso la imagen serían las distintas carátulas de las películas. Tanto el cliente como el servidor gestionan sus respuestas dependiendo del contenido y tipo del mensaje. Sin embargo, en el caso de las recomendaciones de películas, nos encontramos un caso especial. En un mensaje se deberá obtener la recomendación de las películas, con los datos de texto, y acto seguido se deberá llevar a cabo una segunda petición para obtener la imagen de la portada de dichas películas. De tal forma que cada recomendación funcionará de la siguiente manera: 1 respuesta con todas las películas y X respuestas, donde X es el número de imágenes obtenidas correspondientes a dichas películas.

Otro punto a tratar en este apartado, y que resulta fundamental, es la definición de la clase Mensaje, que representa la información que se envía en cada comunicación entre el cliente y el servidor. El mensaje variará dependiendo del tipo de petición que realice el cliente. Para identificar el tipo de mensaje, se usa el atributo de la clase *código*. Código representa un atributo numérico de tipo entero, que dependiendo del valor del mismo, determina el tipo de mensaje y la forma de tratarlo. Actualmente, existen 6 tipos de peticiones distintos, por lo que este atributo tendrá asignados valores de 0 a 5. En caso de futuras mejoras, se podrán añadir más valores a este atributo

1. Petición de recomendación a la librería
    - En esta petición se solicita una recomendación por filtrado colaborativo al servidor.
    - El mensaje del cliente:
        - *codigo:::id_usu*.
            - código = 0.
            - id_usu = Identificador del cliente de Unity.
    - El mensaje del servidor:
        - *codigo:::id_us:::id_peli:::nombre_peli*.
            - código = 0.
            - id_us = Identificador del cliente de Unity.
            - id_peli = Identificador de la película.
            - nombre_peli = Nombre de la película.
        - El mensaje suele tener varias películas: *codigo:::id_us:::id_peli1:::nombre_peli1:::id_peli2:::nombre_peli2.*
2. Petición de recomendación sobre la cartelera.
    - En esta petición el cliente solicita la cartelera con las recomendaciones sobre la misma.
    - El mensaje del cliente:
        - *codigo:::id_usu*.
            - código = 1.
            - id_usu = Identificador del cliente de Unity.

- El mensaje del servidor:
  - *codigo:::id_us:::id_peli:::nombre_peli:::similaridad:::explicacion*.
    - ◇ código = 1.
    - ◇ id_us = Identificador del cliente de Unity.
    - ◇ id_peli = Identificador de la película.
    - ◇ nombre_peli = Nombre de la película.
    - ◇ similaridad = Valor numérico que determina cuánto le va a gustar una película al usuario.
    - ◇ explicacion = Los tags en los que se basa la recomendación de dicha película.
  - El mensaje suele incluir varias películas: *codigo:::id_us:::id_peli1:::nombre_peli1:::similaridad1:::explicacion1::: id_peli2:::nombre_peli2:::similaridad2:::explicacion2*.

3. Registrar usuario nuevo

   - Esta petición sirve para indicarle al servidor si puede registrar a un nuevo usuario.
   - Mensaje del cliente:
     - *codigo:::nombre:::contraseña*.
       - codigo = 2.
       - nombre= Nombre de usuario.
       - contraseña= Contraseña de ese usuario.
   - Mensaje del servidor:
     - *codigo:::id_us:::id_pelicula:::nombre_pelicula*.
       - codigo = 2.
       - id_us = El id asignado a ese usuario.
       - id_pelicula = Es el identificado de la película.
       - nombre_pelicula = Es el título de la película.
     - Al registrarse correctamente se le envían al usuario 40 películas. No es necesario que el usuario valore las 40, solo debe valorar 20. Se envían 40 películas con el fin de que haya variedad de elección por parte del usuario
     - En caso de haber un error en el registro, el mensaje que se envía es *código:::error*. Donde “error” es un string que describe el error.

4. Login de usuario

   - En esta petición, el cliente está registrado previamente y quiere acceder con su cuenta.
   - Mensaje del cliente:
     - codigo:::nombre:::contraseña.
       - código = 3.
       - nombre= Nombre de usuario.

        - contraseña= Contraseña de ese usuario.
    - Mensaje del servidor:
        - codigo:::id_us.
            - código = 3.
            - id_us = Identificador de usuario.
        - En caso de haber un error en el registro, el mensaje que se envía es código:::error. Donde código=2 y error es un string que dice error.

5. Evaluar una película.

    - En este mensaje lo que el cliente quiere es que se guarde una valoración de película que él mismo ha realizado.
    - Mensaje del cliente:
        - codigo:::id_us:::id_item:::valor.
            - código = 4.
            - id_us = Identificador de usuario.
            - id_item = es el identificado de la película.
            - valor = Valoración que el cliente hace de la película.
    - Mensaje del servidor:
        - codigo:::correcto.
            - código = 4.
            - correcto = Mensaje que indica que se ha completado, diciendo correcto.

6. Petición de imagen

    - El cliente quiere la portada de una película.
    - Los mensajes de texto e imágenes son dos mensajes distintos. Es por eso que una vez el cliente tiene una película después de una petición, hace una segunda petición de forma automática para obtener la imagen de la película.
    - Mensaje del cliente:
        - codigo:::id_tem.
            - codigo = 5.
            - id_item = Identificador de la película.
    - Mensaje del servidor:
        - Se devuelve la imagen.

4.2.2. Backend

El backend, desarrollado en Python, tendrá la misión de, a partir de una petición del usuario, procesar la misma para dar una solución satisfactoria.

Lo primero a tener en cuenta es que existen dos programas en el servidor. Uno de los programas es el programa principal del sistema y el otro, es el encargado de actualizar la cartelera de películas y los datos asociados con la cartelera actual del cine.

Un apunte importante sobre el código del fichero main, fichero principal de la aplicación y que contiene todas las funciones para dar respuestas a las peticiones de los usuarios. Buscando la mayor eficiencia posible, el fichero main, antes de permitir conexiones, cargará en memoria tanto la BBDD: usuarios, películas, ratings, cartelera, como los modelos de machine learning con los que va a trabajar: SVD y BERT. Ninguno de los dos modelos será entrenado hasta que la aplicación se reinicie, solo actualizado. El comportamiento de inicialización del main queda reflejado en el Listado 4.3,

```
1 Inicializacion():
2     base_datos <- cargarLaBaseDeDatos()
3     SVD <- crearSVD(base_datos)
4     BERT, tokenizer <- crearBERT(base_datos)
5     EsperarConexiones(base_datos, SVD, BERT, tokenizer)
```

Listado 4.3: Inicializar Backend

Actualizar cartelera

El proceso que se sigue para actualizar la cartelera de películas es el siguiente. Primero se hace una conexión con la BBDD de TMDB y se solicitan 10 películas que se encuentran en cartelera. De esta comunicación, se obtiene de cada película: título, descripción, tags y la imagen del póster. Segundo, se analiza si esa película existía previamente en la BBDD. En caso afirmativo, se trabaja con los datos de la BBDD. En caso contrario, se deberá crear un objeto nuevo en la BBDD. Los tags por defecto que ofrece TMDB son bastante limitados, es por eso que mediante la librería spicy y usando la descripción de la película, se extraen más tags a partir de la descripción.

Finalmente. se guarda la información. Para ello, primero se adaptan los datos de la película a la estructura de la BBDD, definida en la Sección 4.1.3. Acto seguido, se comprueba que no existía previamente esa película en la BBDD. En caso de no existir previamente se introduce una nueva. Para la cartelera actual de cine, siempre se sobrescriben estos datos con las películas que se encuentran en cartelera de TMDB.

De forma adicional, la fase de actualización de la cartelera se utiliza también para eliminar los datos asociados a los usuarios invitados. Los usuarios invitados que se crean durante la ejecución del programa principal no se eliminan automáticamente, ya que el sistema no dispone de un mecanismo que permita detectar el momento en que finaliza su actividad. Como consecuencia, estos usuarios quedan registrados en la BBDD una vez finalizada la sesión.

Teniendo en cuenta esta situación, el proceso de actualización de la cartelera se ejecuta con el sistema principal detenido, evitando así accesos concurrentes a la BBDD que podrían provocar inconsistencias o incluso su corrupción. Aprovechando

este estado controlado, durante la actualización se identifican todos los usuarios de tipo invitado almacenados en la BBDD y se procede a su eliminación. De este modo, se garantiza la limpieza de los datos temporales y se mantiene la integridad del sistema.

El comportamiento del proceso de actualización descrito puede observarse en el Listado 4.4.

```
ActualizacionSemanal():
    base_datos <- cargarBaseDeDatos()
    credenciales <- crearCredencialesTMDB()
    carteleraActual <- peticionATMDB(credenciales)
    for pelicula in carteleraActual:
        if(peliculaExistiaEnBaseDeDatos(base_datos, pelicula)):
            pelicula_nueva <- sacarBaseDeDatosPelicula(base_datos, pelcula)
        else:
            pelcula_nueva <- crearPeliculaNueva(pelicula)
            imagen <- descargarImagen(pelicula)
        actualizarBaseDatos(pelicula_nueva,imagen,base_datos)
    borrarInvitados(base_datos)
```

Listado 4.4: Actualizacion cartelera

SRFC

Por otra parte, y como se indicó en el apartado de investigación (ver Sección 4.1), se ha decidido llevar a cabo la implementación de un SR de filtrado colaborativo mediante un SVD. La BBDD que se usará para obtener las valoraciones de los usuarios será MovieLens.

El funcionamiento del modelo es simple, una vez el usuario hace la petición, se toma su id y el modelo, entrenado previamente con la BBDD de ratings de usuarios, obtiene la recomendación de varias películas, se descartan las que el usuario ha visto anteriormente y se seleccionan las 10 mejores películas, es decir, aquellas que el algoritmo determina que más le van a gustar al usuario. Acto seguido, se devuelven al usuario.

Este sería el funcionamiento normal del modelo. No obstante, es necesario tener en cuenta distintas casuísticas, siendo la más relevante el problema del arranque en frío. Tal y como se ha indicado anteriormente, el modelo basado en SVD se entrena una única vez durante su ejecución y no vuelve a entrenarse hasta que el sistema se reinicia.

Como consecuencia, cuando se incorporan nuevos usuarios o nuevos ítems que no estaban presentes en el conjunto de datos utilizado durante el entrenamiento, el modelo no dispone de la información necesaria para generar predicciones sobre ellos. En estos casos, el algoritmo no es capaz de reconocer dichos elementos al introducirlos en el sistema, lo que provoca una excepción interna que finaliza la ejecución del programa con un mensaje de error.

Para este problema, existen dos soluciones. La primera es entrenar el modelo con cada recomendación para que de esta manera tenga toda la información actualiza-

da en el momento de realizar una recomendación. El inconveniente de esta solución es que reentrenar el modelo continuamente, lo que es increíblemente ineficiente. El segundo método, más eficiente y elegido para este sistema, es el de introducir manualmente los datos en un SVD previamente entrenado. Por lo tanto, en cada petición, se comprueba si los datos del usuario del SVD y los de la BBDD coinciden, es decir, se han hecho modificaciones en tiempo de ejecución. En caso de que sí, se añade manualmente el usuario o las películas nuevas al SVD y se le pide una recomendación

Este comportamiento, aunque técnicamente correcto, tiene implicaciones en la percepción del usuario. Durante una misma ejecución, las recomendaciones generadas por el modelo SVD para un usuario tienden a ser prácticamente idénticas. Cuando se registran nuevas valoraciones, el modelo no las incorpora hasta que se vuelve a entrenar, por lo que el sistema únicamente excluye de la recomendación el ítem recién valorado, manteniendo sin cambios el resto de las películas recomendadas. En la práctica, esto se traduce en que solo se introduce un nuevo elemento en la lista de recomendaciones.

Aunque este comportamiento no constituye un error desde el punto de vista del algoritmo, puede generar en el usuario la sensación de que no se están produciendo nuevas recomendaciones. Para mitigar este efecto y mejorar la experiencia de uso, el sistema no selecciona directamente las 10 películas con mayor puntuación, sino que obtiene un conjunto ampliado de las 30 mejores recomendaciones. A partir de este conjunto, se seleccionan aleatoriamente 10 películas que se presentan al usuario. De este modo, se introduce variabilidad en las recomendaciones y se transmite la percepción de un recálculo dinámico, sin necesidad de reentrenar el modelo.

El funcionamiento descrito puede observarse en el Listado 4.5.

```
SRFC(SVD, base_datos, id_usuario):
    peliculas_vistas_usuario <- obtenerPeliculasVistasPorElUsuario(base_datos,id_usuario)
    recomendaciones <- HacerRecomendacion(SVD,id_usuario)
    recomendacionFiltrada <- eliminarElementosVistos(recomendaciones,
        peliculas_vistas_usuario)
    recomendacionesOrdenadas <- OrdenarRecomendaciones(recomendacionFiltrada)
    top30 <- TomarLos30Mejores(recomendacionesOrdenadas)
    top10 <- tomar10ValoresAleatorios(top30)
    devolverSolucion(top10)
```

Listado 4.5: Recomendacion mediante SRFC

SRBC

El modelo seleccionado para este tipo de recomendación es BERT, el cual se inicializa al principio de la ejecución. La BBDD que se utiliza en este caso será la de la cartelera actual del cine, obtenida de TMDB. Para obtener todos los tags, se utilizará la BBDD de MovieLens, más concretamente la que contiene la extensión de tags y no la original de MovieLens.

El proceso de inicialización del modelo es rápido y sencillo. Únicamente habrá que definir el modelo de BERT exacto que se va a usar. Es importante recordar que BERT

es, básicamente, un tipo de Transformer, por lo que únicamente hay que definir las variables tipo que tiene por defecto, indicando que se quiere usar BERT. Acto seguido, se obtiene de la BBDD todas las películas de la cartelera y se crea una nueva columna en la BBDD. Esta nueva columna será el resultado de hacer un embedding con el modelo a los tags y al nombre de la película. Una vez ya se tiene todo el modelo y BBDD preparada, se esperará la petición. Cuando llega la petición del usuario, lo primero que se realiza es la creación del perfil de usuario. Para ello, se buscan todas las películas que el usuario ha visto y se obtienen todos sus tags. Acto seguido se realiza un embedding de los gustos del usuario. Para cada película de la cartelera se mide la similitud entre el usuario y la película y se obtiene un valor. A partir de ese valor se reordena la cartelera en función de sus gustos.

Por otra parte, se considera relevante dotar al sistema de cierto grado de explicabilidad en las recomendaciones ofrecidas al usuario. Para ello, y teniendo en cuenta que el modelo basado en BERT genera recomendaciones a partir de la similitud entre los tags asociados a una película y los tags que definen las preferencias del usuario, es posible proporcionar una explicación sencilla del motivo de cada recomendación. En este contexto, cuanto mayor sea el número de tags compartidos entre una película y el perfil del usuario, mayor será la similitud entre ambos y, por tanto, mayor la probabilidad de que dicha película sea recomendada.

A partir de esta premisa, el sistema compara los tags de las películas previamente vistas por el usuario con los de la película recomendada, y devuelve como explicación aquellos tags comunes a ambos conjuntos. De este modo, el usuario puede comprender de forma básica la razón por la que una película ha sido seleccionada. El funcionamiento de este proceso puede observarse en el Listado 4.6.

```
1 SRBC(BERT, tokenizer, base_datos, id_usuario):
2     peliculas_vistas_usuario <- obtenerPeliculasVistasPorElUsuario(base_datos,id_usuario)
3     tags_usuario <- obternerTagsPeliculas(peliculas_vistas_usuario)
4     peliculas_embeding <-hacerEmbeding(tokenizer, peliculas_vistas_usuario)
5     recomendaciones <- hacerRecomendacion(BERT, tags_usuario)
6     recomendacionesOrdenadas <- ordenar(recomendaciones)
7     explicabilidad <- obtenerExplicabilidad(recomendacionesOrdenadas, tags_usuario)
8     solucion <- CrearRespuesta(recomendacionesOrdenadas, explicabilidad)
9     devolverSolucion(solucion)
```

Listado 4.6: Recomendacion mediante SRBC

Gestión de Usuarios

El sistema, aparte de ofrecer distintas recomendaciones a los usuarios, también debe de ser capaz de llevar a cabo una serie de gestiones y peticiones relacionadas con los mismos.

- Registro
  - Cada usuario, al entrar por primera vez a la aplicación, puede registrarse como un nuevo usuario del sistema. Para ello, introduce un nombre de usuario y contraseña. El sistema identificará que ese usuario no exista. En caso

de existir aparecerá un mensaje de error, en caso contrario, se creará el usuario en la BBDD y se le mostrarán 40 películas para que evalúe 20.

- Usuario invitado
  - Funciona como un registro. Se realiza un registro en la BBDD de un nuevo usuario invitado. Para ello, cuando se realiza la autentificación se debe obtener una respuesta con un id = 0. Se registra entonces un nuevo usuario invitado. Cuando se hace la renovación de cartelera, se limpia a su vez la BBDD. (ver Listado 4.7)

```
registro(base_datos, nombre, contraseña):
    existe <- verSiExisteUsuario(nombre)
    if(existe == False):
        usuario_nuevo <- crearUsuario(nombre, contraseña)
        actualizarBaseDatos(base_datos, usuario_nuevo)
        peliculas40 <- obtener40PeliculasAleatorias(base_datos)
        solucion <- solucionRegistro(usuario_nuevo, peliculas40)
    else:
        solucion <- "error"
    devolverSolucion(solucion)
```

Listado 4.7: Registro Backend

- Login
  - Cada usuario registrado podrá entrar en su cuenta para, de esta forma, obtener sus recomendaciones. Para ello, el usuario introduce su nombre de usuario y contraseña. Si el usuario existe en la BBDD, se le devolverá al usuario su identificador, en caso contrario, se le devolverá un mensaje de error. (ver Listado 4.8)

```
login(base_datos, nombre, contraseña):
    existe <- existeUsuario(base_datos, nombre, contraseña)
    if(existe == False):
        solucion <- "error"
    else:
        solucion <- obtenerUsuario(base_datos, nombre, contraseña)
    devolverSolucion(solucion)
```

Listado 4.8: Login Backend

- Valorar película
  - Todo usuario podrá valorar una película. Para valorar una película se modificará la BBDD ya sea introduciendo un nuevo rating o modificando uno previamente existente. (ver Listado 4.9)

```
Valoracion(base_datos, id_usuario, id_item, valor):
    existe <- existePreviamenteLaValoracion(base_datos, id_usuario, id_item,
        valor)
    if(existe == True):
        modificarValor(base_datos, id_usuario, id_item, valor)
    else:
        añadirValor(base_datos, id_usuario, id_item, valor)
    solucion <- "correcto"
    devolverSolucion(solucion)
```

Listado 4.9: Valorar Pelicula Backend

4.2.3. Frontend

Para diseñar el frontend, se usará el motor gráfico Unity. Una de las premisas a tener en cuenta al usar este motor es la definición de un mundo dividido en escenas. Por ello, se trabaja con una serie de elementos prefabricados, es decir, que ya han sido construidos y preparados para adaptarse a la escena. Cada escena será independiente, algunas usarán los mismos elementos prefabricados o variaciones de los mismos, dependiendo de las necesidades de la propia escena. Además, existirán una serie de elementos comunes entre escenas que se irán moviendo sin destruirse entre escenas.

Prefabs

Existen varios elementos prefabricados que se crean en la escena a medida que esta se inicia y desarrolla. Estos prefabricados suelen recibir el nombre de prefabs. En este apartado únicamente se destacarán los más importantes, es decir, aquellos que tienen una funcionalidad relevante en la escena y no una meramente decorativa.

- Botón recarga (ver Figura 4.2)
  - La principal función de este elemento es que el usuario vuelva a pedir una recomendación de tipo filtrado colaborativo sin salir de la escena.
  - El funcionamiento es el siguiente:
    - Al pulsar el botón, se le pide al gestor de la escena interno que elimine las películas que se le están mostrando al usuario y que vuelva a pedir una nueva recomendación de películas (ver Listado 4.10).

```
BotonRecarga():
    gestorEscena <- obtenerGestorEscena()
    if(elBotonSeHaPulsado() == True):
        borrarPeliculasActuales(gestorEscena)
        mensaje
        hacerPeticionDeSRFC(gestorEscena)
```

Listado 4.10: Prefab boton Recarga

- Número de películas votadas (ver Figura 4.3)
  - La función de este elemento, aunque ciertamente es estético, es ayudar al usuario durante su registro, mostrando las películas que le faltan por valorar.
  - Es una interfaz que constantemente está siendo actualizada por el gestor de escena para indicar el número de películas que faltan por evaluar en el registro (ver Listado 4.11).

```
Numero_pelis_botadas():
    gestorEscena <- obtenerGestorEscena()
    if(Un UsuarioSeHaRegistrado() == True):
        numeroPeliculasSinValorar <- numeroPeliculasSinValorar(gestorEscena)
```

Figura 4.2: Prefab: botón recarga

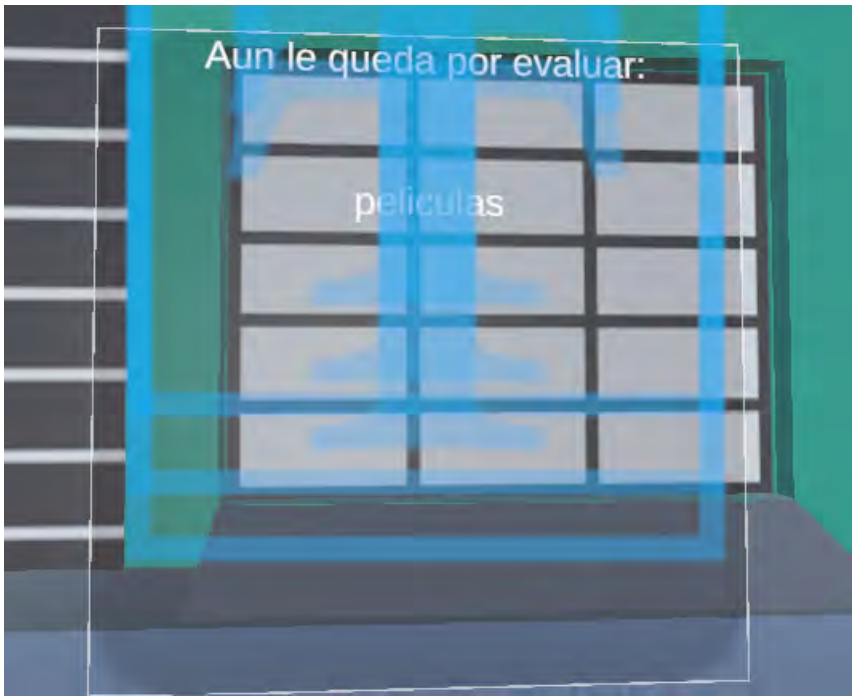


Figura 4.3: Prefab: interfaz con las películas que has valorado en el registro

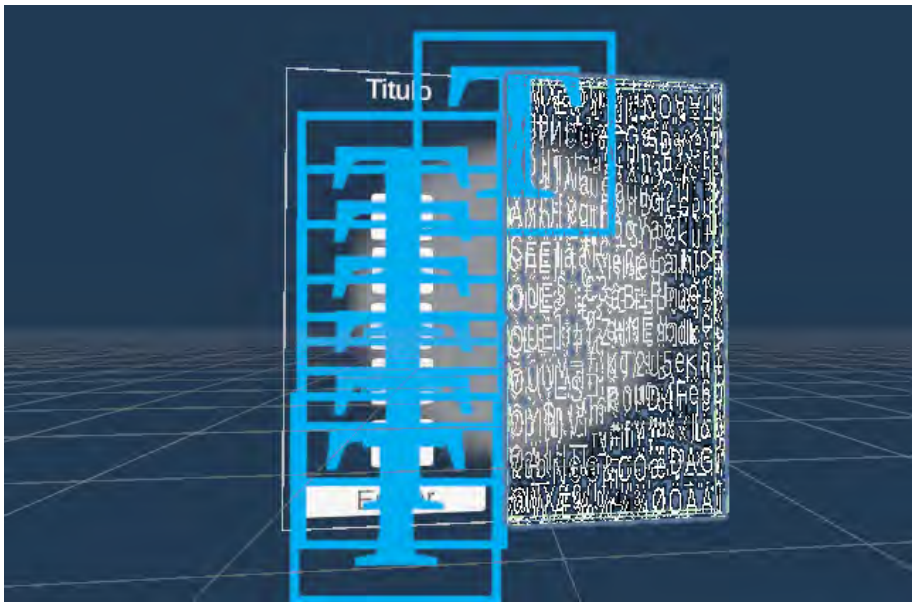


Figura 4.4: Prefab: película

```
5         mostrarNumero(numeroPeliculasSinValorar)
6         direccion <- determinarDireccionMovimiento()
7         moverInterfaz(direccion)
```

Listado 4.11: Prefab numero de pelis votadas

- Película (ver Figura 4.4)
  - El objeto que representa la película que se le recomienda y valores relacionados con esta, además de ofrecer la posibilidad de evaluarla.
  - Este prefab se utiliza para la representación gráfica de la película dada por la recomendación del SRFC y para las películas que aparecen y que se deben valorar al registrarse.
  - Está formada a nivel visual por:
    - Un texto que indica el nombre de la película.
    - Una imagen que muestra el póster de la película.
    - 6 botones, cinco presentan un número del 1 al 5 para realizar las valoraciones y en el sexto botón incluye el texto “enviar”. La idea es que el usuario, al pulsar un botón numérico, este cambie de color y al pulsar el botón de “enviar”, se envíe la evaluación al servidor.
  - A nivel lógico
    - Presenta un script que contiene toda la información asociada a una película.
      - id_pelicula = identificador de la película.
      - nombre = nombre de la película.
      - textura = imagen del póster de la película.
      - rating = valoración de la película.
      - ranking = nivel de recomendación de la película. Se usa únicamente para cartelera.

        - ◇ razón = explicación en texto de por qué se le recomienda una película. Se usa únicamente para cartelera.
        - ◇ texto = es el objeto que representa a un elemento texto de la película, se actualiza dependiendo del valor de nombre.
        - ◇ nivelRecomendacion = es el objeto que representa a un elemento texto de la película, se actualiza dependiendo del valor de ranking.
        - ◇ razonRecomendacion = es el objeto que representa a un elemento texto de la película, se actualiza dependiendo del valor de razón.
    - Además presenta un script de botones que se encarga de saber cuál botón se ha pulsado y, de esta manera, cambiar de color de blanco a verde.
    - Existen dos scripts de botones más, uno para conocer qué botón de valoración se ha pulsado, además del botón de "enviar" para mandar al servidor la calificación de la película (ver Listado 4.12).

```
gestionBotonesPelicula():
    gestorMensajes <- dameGestorMensajes()
    datosPelicula <- dameDatosPelicula()
    datosUsuario <- dameDatosUsuario()
    listaBotones <- dameTodosBotones()
    for boton in listaBotones:
        if(botonPulsado(boton) == True):
            if(tipoBoton(boton) == "valor"):
                actualizarValorActual(boton)
                cambiarColorVerde(boton)
                cambiarRestoColor(boton, listaBotones)
            else:
                valorActual <- valorActualPelicula()
                mensaje <- crearMensaje(valorActual, datosUsuario,
                    datosPelicula)
                enviarMensaje(mensaje, gestorMensajes)
```

Listado 4.12: Prefab gestion botones pelicula

- Película cartelera (ver Figura 4.5)
    - Es el prefab que se utiliza para la recomendación de cartelera.
    - Es prácticamente igual al prefabricado de películas, pero con una diferencia estética. En este elemento, se muestra el nivel de recomendación de la película al usuario y también presenta un botón que al pulsarlo muestra las razones de la recomendación.
    - (ver Listado 4.12)
- prefab_inicio (ver Figuras 4.6, 4.7, 4.8 y 4.9)
    - Este elemento es el prefabricado que contiene la primera escena. Contiene todos los elementos estéticos y funcionales de la misma. En ella se puede encontrar:
        - Todos los edificios con un fin decorativo.
        - Paredes invisibles para que el usuario no salga del mapa.
        - Se utiliza un asset de Unity que muestra un teclado para RV.
            - ◇ Este teclado se ha modificado añadiendo cuatro botones:

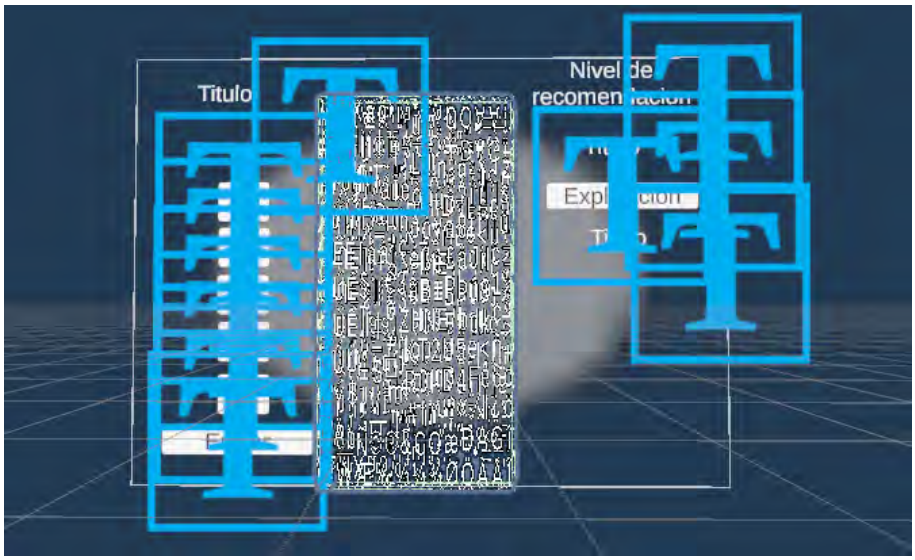


Figura 4.5: Prefab: pelicula en cartelera

1. login: Para iniciar sesión con un usuario.
2. register: Para crear un nuevo usuario.
3. cerrar: Para cerrar el programa.
4. invitado: Entra con un usuario temporal.

- Estos últimos botones no se encuentran en el propio prefab, sino en la versión modificada que se usa en la escena. Sin embargo, debido a problemas con los assets, se decidió guardar esta versión y trabajar sobre ella en el producto final.

- El funcionamiento es el siguiente. Al inicializarse el programa, se crea este prefabricado. Cuando el usuario accede al teclado indica que acción quiere realizar. En caso de login, una vez se asegura que es correcto, se traslada a la siguiente escena. En caso de registro, se elimina el edificio de la derecha del teclado, se cargan las 40 películas y el prefabricado de número de películas.

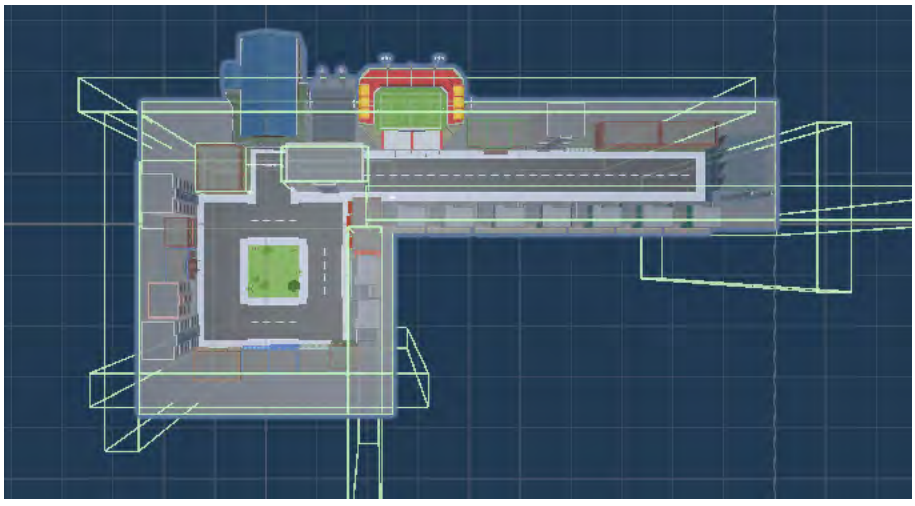


Figura 4.6: Prefab: zona inicio general

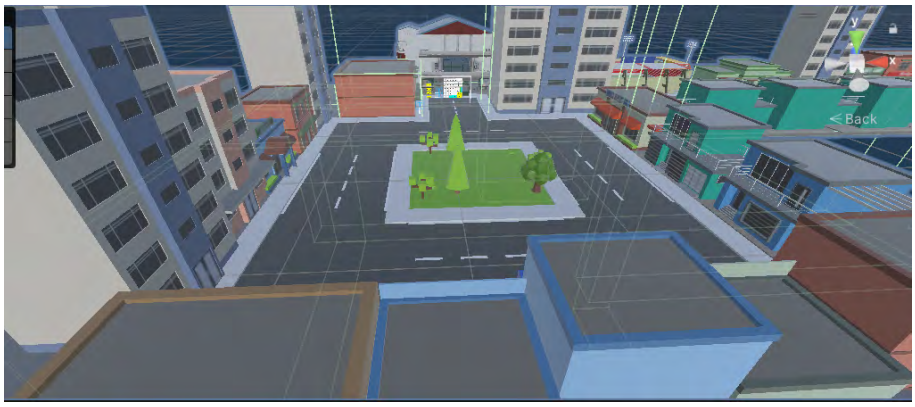


Figura 4.7: Prefab: zona inicio general centro

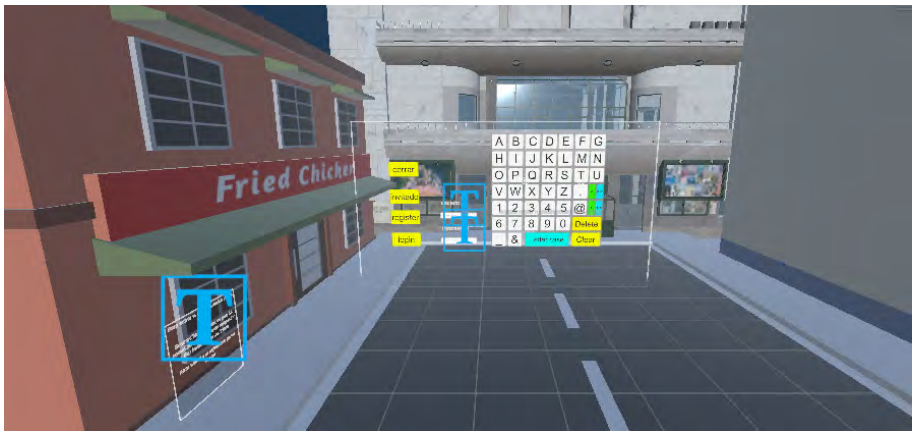


Figura 4.8: Prefab: zona inicio teclado

Figura 4.9: Prefab: zona inicio calle registro

- Puerta entrada (ver Figuras 4.11 y 4.10)
  - Lo que aquí se busca es replicar la entrada interior del teatro, en esta escena, se tienen varias puertas y señales indicando lo que hace cada puerta. Dependiendo de la puerta en la que se entre, el usuario se moverá a una escena u otra.

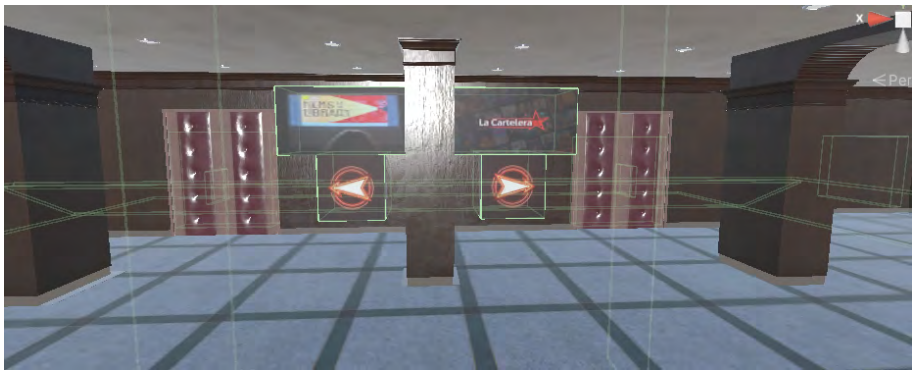


Figura 4.10: Prefab: entrada 1

Figura 4.11: Prefab: entrada2

- zona_pelicula (ver Figuras 4.12 y 4.13)
  - Este prefab representa la escena donde se desarrollará la recomendación basada en contenidos, en otras palabras, la cartelera.
  - Su misión es simular un pasillo del cine que contiene la cartelera de películas actuales.
  - Está formada a partir del modelo del teatro y la única puerta presente permitirá al usuario salir de la sala para volver de nuevo a la entrada.

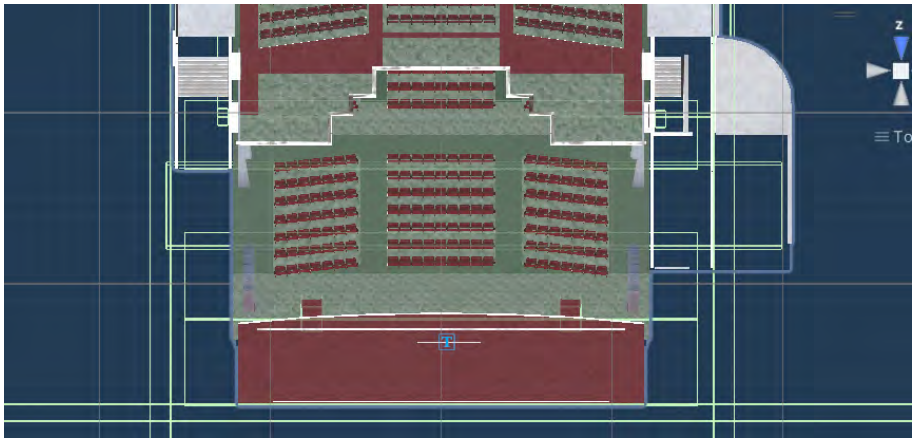


Figura 4.12: Prefab: sala cine 1

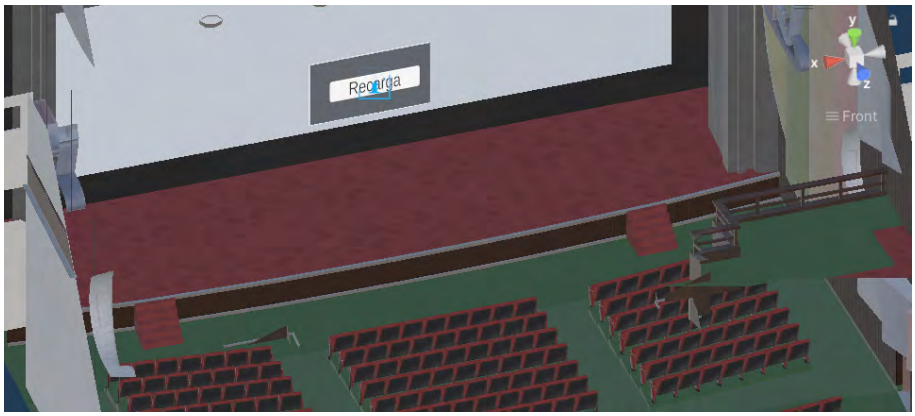


Figura 4.13: Prefab: sala cine 2

- zona_pelicula_colaborativo (ver Figuras 4.14, 4.15 y 4.16)

  - Este prefab representa la escena donde se desarrollará la recomendación por filtrado colaborativo, en otras palabras, la sala de cine.
  - Su misión es simular una sala de cine que contiene la portada de 10 películas.
  - Está formada a partir del modelo del teatro, sus diferentes puertas únicamente permiten salir a la entrada y presenta un botón de reinicio para generar una nueva recomendación.

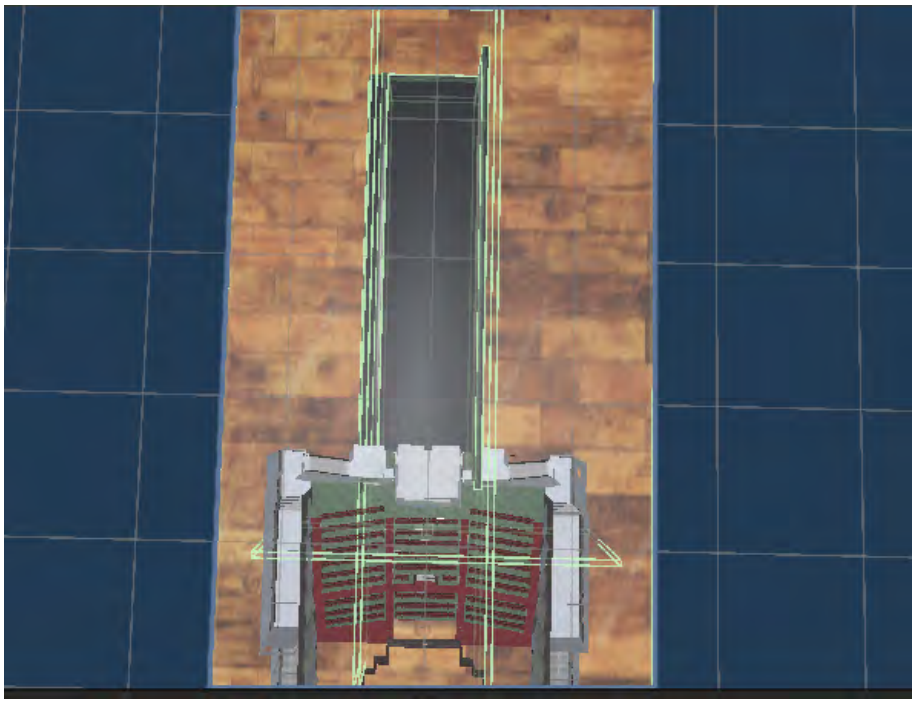


Figura 4.14: Prefab: sala de cartelera 1

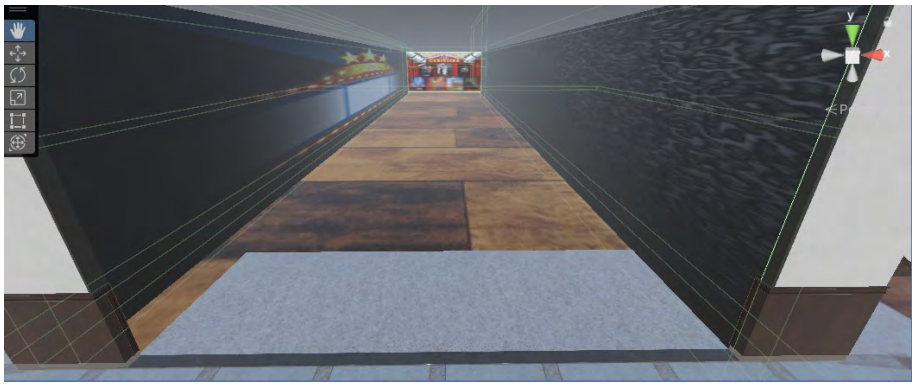


Figura 4.15: Prefab: sala de cartelera 2

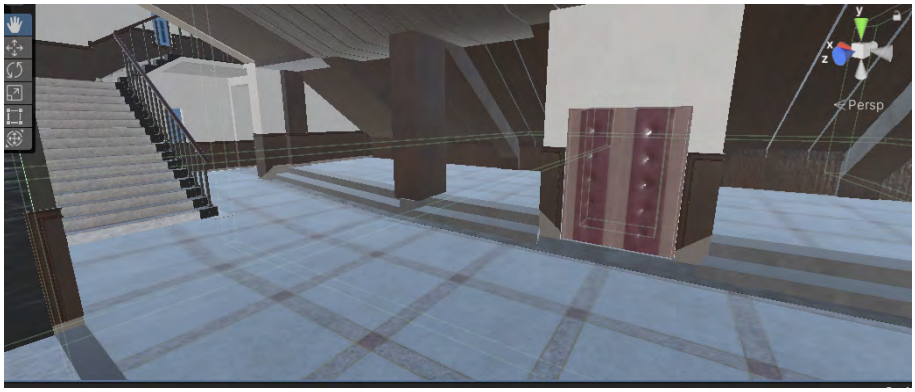


Figura 4.16: Prefab: sala de cartelera 3

General

Será necesario tener una serie de objetos que no se destruyan entre escenas. Estos elementos serán gestores de información y tareas, almacenando toda aquella información y funciones que se consideran comunes y necesarias para todas las escenas. Todas ellas estarán contenidas en un objeto gestor que estará compuesto por los siguientes elementos.

- gestor
  - gestorUsuario
    - Este elemento contiene un script que se encargará de almacenar toda la información del usuario. Más concretamente almacenará:
      - id = Identificador del usuario.
      - nombre = Nombre del usuario.
      - contraseña = Contraseña del usuario.
  - gestorMovimiento
    - Este elemento se encargará de mover al usuario entre las distintas escenas.

- El funcionamiento es el siguiente, cuando un usuario colisiona con una puerta, se llama a esta función para indicarle que escena debe cargar.

- gestorComunicaciones
  - Este objeto tiene como función principal la conexión con la BBDD. De tal forma que, dependiendo del mensaje, se obtendrá una respuesta que puede ser tanto imagen como texto (ver Listado 4.13).

```
comunicacion(mensaje):
    ip, port <- obtenerIPPuerto()
    conexion <- establecerConexion(ip, port)
    mandarMensaje(mensaje, conexion)
    respuesta <- obtenerRespuesta()
    devolverRespuesta(respuesta)
```

Listado 4.13: Gestor Comunicacion

- gestorParedes
  - En el escenario técnicamente no existe el contacto con las paredes. Cuando el usuario se acerca a una pared, no está colisionando con ella.
  - Se han creado elementos invisibles y se han superpuesto a las paredes. Cuando el usuario se acerca a estos elementos, el gestor lo detecta y mueve al usuario en la dirección contraria, simulando así la existencia de paredes y colisiones con elementos (ver Listado 4.14).

```
gestionColisionesParedes(mensaje):
    usuario <- damedatosUsuario()
    listaParedes <- dameListaParedes()
    distanciaMaxima <- dameUmbralDistancia()
    for pared in listaParedes:
        distancia <- calculoDistancia(pared, usuario)
        if(distancia > distanciaMaxima):
            apartarUsuario(usuario, pared)
```

Listado 4.14: Gestor Paredes

Escenas

Las escenas son elementos independientes unos de los otros. Cada escena tendrá un gestor interno que se encargará de la creación del escenario mediante los elementos prefabricados ya establecidos. Cabe destacar que estos gestores son objetos vacíos, es decir, estos objetos no tienen una representación física en la escena, únicamente son elementos que ejecutan código. Por lo que serán estos elementos los que, a nivel de código, ejecuten los scripts que inicialicen toda la escena y gestionen los distintos aspectos de la misma. Por tanto, serán estos gestores los encargados de la renderizar y gestionar todos los prefabricados mencionados anteriormente.

1. Inicio

   La función de esta escena es dar la bienvenida al usuario. El funcionamiento es el siguiente (ver Listado 4.16).

Primero, se crea tanto la escena de inicio con el prefab de inicio (ver Listado 4.2.3) como el usuario. El usuario se encontrará de frente con un teatro y un teclado que le dará 3 opciones, iniciar sesión, registrarse o salir. En caso de iniciar sesión, se moverá directamente al siguiente escenario, en caso de cerrarse, se cerrará el programa. En caso de registro, existe un procedimiento concreto (ver Listado 4.15).

El procedimiento de registro es el siguiente. Una vez el usuario introduce sus credenciales y se aceptan, el edificio que se encontraba a su derecha desaparecerá, en ese momento, el usuario verá una larga calle, con 40 películas, usando el prefab de películas (ver Figura 4.4) y el prefab de número de películas (ver Figura 4.3), que le indicará en todo momento cuántas películas le faltan por valorar. Una vez valore 20, se mueve el escenario a la siguiente pantalla, en este caso, la entrada. La escena se puede apreciar en las Figuras 4.17, 4.18 y 4.19

```
teclado():
    gestorEscena <- dameGestorEscena()
    gestorComunicacion <- dameGestorComunicacion()
    gestorMovimiento <- dameGestorMovimiento()
    nombre <- dameNombre()
    contraseña <- dameContraseña()
    teclas <- dameTeclas()
    for tecla in teclas:
        if(SeHaPulsado(tecla) == True):
            if(tipoTecla(tecla) == "letra"):
                añadirLetra(nombre, contraseña, tecla)
            if(tipoTecla(tecla) == "modificadorTeclado"):
                modificarTeclado(teclas)
            if(tipoTecla(tecla) == "modificadorPalabras"):
                modificarPalabras(tecla, nombre, contraseña)
            if(tipoTecla(tecla) == "registro"):
                mensaje <- crearMensajeRegistro(nombre,contraseña)
                enviarMensaje(mensaje)
                respuesta <- obtenerRespuestaMensaje()
                generarPeliculasRegistro(gestorEscena)
            if(tipoTecla(tecla) == "login"):
                mensaje <- crearMensajeLogin(nombre,contraseña)
                enviarMensaje(mensaje)
                respuesta <- obtenerRespuestaMensaje()
                if(existEnOperacion(respuesta)):
                    irALaEntrada(gestorMovimiento)

            if(tipoTecla(tecla) == "invitado"):
                mensaje <- crearMensajeInvitado()
                enviarMensaje(mensaje)
                respuesta <- obtenerRespuestaMensaje()
                if(existEnOperacion(respuesta)):
                    irALaEntrada(gestorMovimiento)
            if(tipoTecla(tecla) == "cerrar"):
                cerrarPrograma()
```

Listado 4.15: Teclado Inicio 4.2.3

```
gestorInicio():
    escenaInicio <- CrearEscenaInicio()
    usuario <- damedatosUsuario()
    gestorMovimiento <- dameGestorMovimiento()
    if(elUsuarioSeRegistra() == True):
        peliculas <- damePeliculasRegistro()
        generarPeliculas(peliculas)
        numeroPeliculas <- 20
        numeroActual <- peliculasValoradas(peliculas)
        while(numeroPeliculas != numeroActual):
```

```
11             numeroActual <- peliculasValoradas(peliculas)
12         irALaEntrada(gestorMovimiento)
```

Listado 4.16: Gestor Inicio 1

Figura 4.17: Escena: escena inicio ciudad

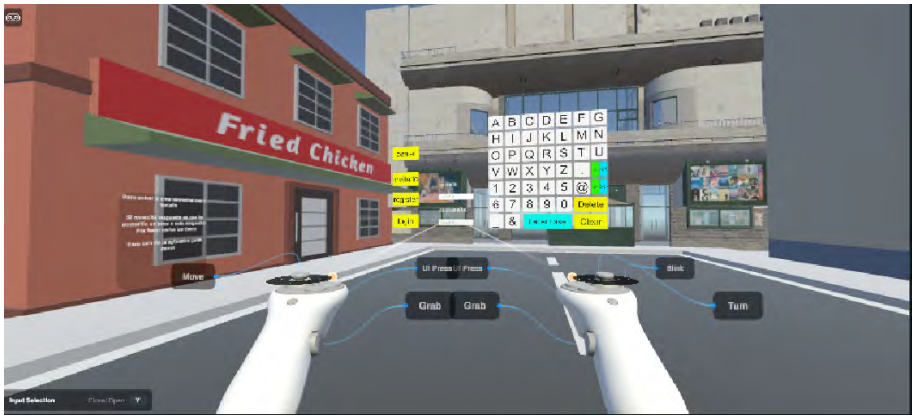


Figura 4.18: Escena: escena inicio teatro

Figura 4.19: Escena: escena inicio registro

2. Entrada

El gestor de la entrada presenta el siguiente funcionamiento. En primer lugar, el usuario aparece en mitad de una sala que corresponde al prefab de la escena de entrada (ver Figura 4.2.3). Delante de él, hay unas puertas de madera que lo devolverán a la zona de inicio si las traspasa (ver Figura 4.20).

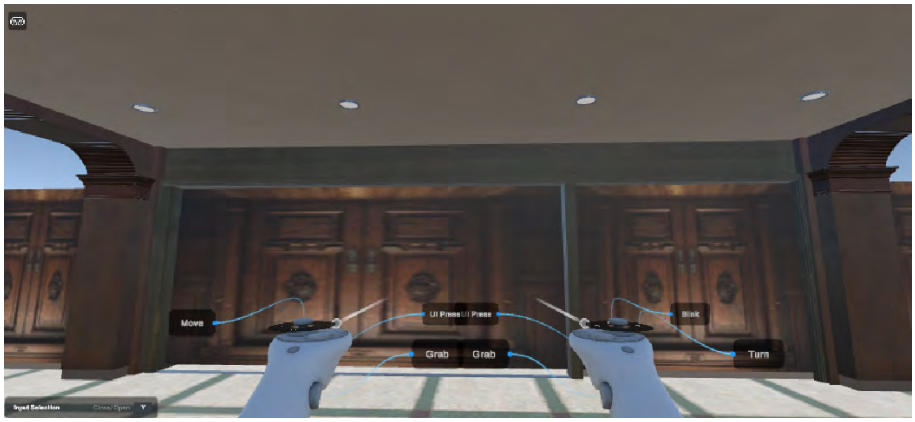


Figura 4.20: Escena: entrada puertas salida cine

Una vez se dé la vuelta, encontrará dos puertas. Al lado de cada puerta hay un cartel indicando a dónde lleva cada puerta. El que indica “biblioteca” se refiere a la escena donde se hace la recomendación por SRFC. Mientras que, el que indica “cartelera”, realiza la recomendación por SRBC de la cartelera actual. Al pasar por cada puerta, se activará el gestor de movimiento y identificando la puerta mediante un tag asignado a la misma. Acto seguido, el usuario aparecerá en la sala seleccionada (ver Figura 4.21).

Figura 4.21: Escena: entrada puertas cine

3. Biblioteca o sala de cine

   Esta escena corresponde a la sala de cine donde al usuario se le da una recomendación de películas mediante un filtrado colaborativo. El gestor de la sala, funciona de la siguiente manera. Inicialmente, crea el elemento prefabricado de la sala de cine (ver Sección 4.2.3). Acto seguido, una vez creado el escenario, realiza la petición de la recomendación por filtrado colaborativo. Una vez obtenida la respuesta, pide las carátulas de las 10 películas que se van a recomendar.

Una vez esto finaliza, crea los prefab de películas (ver Sección **??**) que el usuario contempla (ver Listado 4.17).

El usuario puede pasear por la zona de cine, valorar cualquiera de las 10 películas o incluso utilizar el botón de reinicio 4.2.3 para pedir nuevamente que se le recarguen las películas. Cuando el usuario quiera salir, existen varias puertas que llevan a la entrada por las que puede hacerlo (ver Imagen 4.22).

```
SRFC():
    escena <- generarPrefabCine()
    usuario <- dameDatosUsuario()
    gestorMovimiento <- dameGestorMovimiento()
    gestorComunicacion <- dameGestorComunicacion()

    mensaje <- peticionDeSRFC(usuario)
    enviarMensaje(mensaje, gestorComunicacion)
    respuesta <- obtenerRespuesta(mensaje, gestorComunicacion)
    generarLasPeliculas(respuesta)
```

Listado 4.17: Gestor biblioteca

Figura 4.22: Escena: cine, la biblioteca

4. Cartelera

   Esta escena representa las películas que se encuentran en cartelera. El gestor tiene el siguiente comportamiento (ver Listado 4.18). Primero se crea el prefab de escenario de esta área (ver Sección 4.2.3). Acto seguido se carga toda la cartelera y se piden sus respectivas imágenes. En este caso, las películas serán las de la cartelera actual (ver Figura 4.5) Una vez creado todo, el usuario podrá moverse por la escena, con sus respectivas paredes invisibles limitando el movimiento. El usuario podrá ver las imágenes de las películas e interactuar con ellas. Para salir, tomará la única puerta de la escena, que lo conducirá a la escena de la entrada (ver Imagen 4.23).

```
SRBC():
    escena <- generarPrefabCine()
    usuario <- dameDatosUsuario()
    gestorMovimiento <- dameGestorMovimiento()
    gestorComunicacion <- dameGestorComunicacion()

    mensaje <- peticionDeSRBC(usuario)
    enviarMensaje(mensaje, gestorComunicacion)
    respuesta <- obtenerRespuesta(mensaje, gestorComunicacion)
```

```
10    generarLasPeliculas(respuesta)
```

Listado 4.18: Gestor Cartelera

Figura 4.23: Escena: cine, la cartelera

4.3. Experimentación

La función principal de esta fase es la de probar el sistema y obtener una idea general sobre cuán satisfactoria ha sido la experiencia del usuario, es decir, si ha conseguido cumplir los objetivos inicialmente establecidos y si hay algunas áreas que se puedan mejorar. La organización de esta fase en el proyecto, queda plasmada en la Tabla 4.5.

Fase	Información / Subtareas	Inicio	Fin
Backend	#EXP01, #EXP02	23/12/25	29/12/25

Tabla. 4.5: Planificación de la fase de experimentación del proyecto

Fase	Herramienta
Backend	personas reales, lapiz y papel

Tabla. 4.6: Principales herramientas utilizadas

4.3.1. Pruebas realizadas

En esta sección se describen cuáles fueron las pruebas que se realizaron a la aplicación y las condiciones en las que se realizaron dichas pruebas. Lo primero y más importante es destacar los dos tipos de pruebas que se llevaron a cabo: las de laboratorio y las reales.

Las pruebas de laboratorio se caracterizaron principalmente por los siguientes puntos:

- Fueron realizadas con una aplicación en un entorno local. Es decir, el backend y el frontend se encontraban en la misma red, más concretamente en el mismo equipo informático.
- Los usuarios que realizaron estas pruebas eran miembros del propio equipo desarrollador o conocidos de los mismos que de forma gratuita accedieron a realizar las pruebas.

Por otra parte, las pruebas en entornos reales presentaron las siguientes características:

- Fueron realizadas en un entorno en que tanto el backend como el frontend se encontraban separados y se simulaba un entorno completamente real.
- Los usuarios que las llevaron a cabo, fueron usuarios reales de la aplicación. Ninguno de ellos tenía relación alguna con el proyecto o con el equipo.

Teniendo en cuenta las particularidades presentes en cada tipo de prueba, el procedimiento y acciones que se realizaron fueron las mismas en ambos casos:

- #EXP01 El usuario debía de recorrer el escenario de inicio.
- #EXP02 El usuario debía interactuar con el teclado.
- #EXP03 El usuario debía de registrarse en la aplicación como nuevo usuario.
- #EXP04 El usuario debía validar las 20 películas del registro.
- #EXP05 El usuario debía recorrer la escena de entrada.
- #EXP06 El usuario debía entrar a la zona de cartelera.
- #EXP07 El usuario debía recorrer la cartelera.
- #EXP08 El usuario debía examinar el área de explicabilidad de las películas.
- #EXP09 El usuario debía valorar una película de la cartelera.
- #EXP10 El usuario debía salir de la escena para ir a la entrada y volver a entrar en la cartelera.
- #EXP11 El usuario debía examinar el área de explicabilidad de las películas para ver si existía algún cambio.
- #EXP12 El usuario debía salir a la entrada.
- #EXP13 El usuario debía entrar en la biblioteca.
- #EXP14 El usuario debía valorar una película.
- #EXP15 El usuario debía recargar la recomendación.

- #EXP16 El usuario debía salir a la entrada.
- #EXP17 El usuario de la entrada debía ir a inicio.
- #EXP18 El usuario iniciaba sesión con el usuario con el que se había registrado previamente.
- #EXP19 El usuario repetirá los pasos desde el 5 al 16, comprobando que los cambios realizados en la ronda anterior siguen vigentes.
- #EXP20 El usuario saldrá a inicio y volverá a entrar como invitado.
- #EXP21 El usuario repetirá los mismos pasos realizados la vez anterior (del 4 al 16), comprobando que los cambios realizados en la ronda anterior no aparecen.
- #EXP22 Finalmente, el usuario cierra el programa.

Una vez expuestas las distintas pruebas realizadas, se presenta una tabla que correlaciona cada una de ellas con los requisitos funcionales que se pretende validar. Cabe destacar que el requisito #RF11, al estar relacionado con la dockerización del sistema, no es susceptible de validación mediante pruebas de usuario. Asimismo, los requisitos no funcionales, al tener un carácter transversal, no se reflejan en la tabla siguiente (Ver tabla 4.7).

Requisito	**Experimentos asociados**
#RF01	#EXP06, #EXP07, #EXP08, #EXP15
#RF02	#EXP06, #EXP07, #EXP08, #EXP15
#RF03	#EXP06, #EXP07
#RF04	#EXP03, #EXP04
#RF05	#EXP18
#RF06	#EXP20, #EXP21
#RF07	#EXP09, #EXP14
#RF08	#EXP07, #EXP11
#RF09	#EXP08, #EXP11
#RF10	#EXP01, #EXP02, #EXP05, #EXP06, #EXP07, #EXP13
#RF12	#EXP04

Tabla. 4.7: Trazabilidad entre requisitos funcionales y experimentos realizados

4.3.2. Resultados

A partir de las pruebas correspondientes, tanto las de laboratorio como las reales, se han obtenido los siguientes resultados:

En las pruebas de laboratorio, los usuarios que las llevaron a cabo fueron todos miembros del equipo de desarrollo, por lo que tenían experiencia previa con la aplicación. Esto se tradujo en que la mayoría de interacciones con el sistema eran conocidas

de antemano, por lo que no fueron necesarias explicaciones previas y toda acción se realizaba de forma intuitiva. Las pruebas indicadas anteriormente dieron en general buenos resultados. Sin embargo, algunos usuarios encontraron problemas de rendimiento y una peculiaridad respecto a la recomendación.

- En lo referente al rendimiento. Se debe recordar que en esta prueba el backend y el frontend estaban ejecutándose en el mismo equipo, lo cual suponía una carga de cómputo bastante elevada. Para solucionar este problema, la arquitectura inicial contemplaba separar el backend y el frontend. Además, se llevó a cabo una optimización de todas las operaciones del sistema, intentando mejorar lo máximo posible la experiencia.

- En lo referente a la recomendación, el problema no venía de la recomendación en sí mismo, sino de la generación. En la SRFC, se debía entrar a la sala de cine para obtener una recomendación y si se quería otra, el usuario debía salir y volver a entrar, resultando tedioso para el usuario. Para solucionarlo, se optó finalmente por incluir un botón de recarga (ver Sección 4.2.3) de la petición en el frontend.

Una vez terminadas las pruebas de laboratorio e implementadas las mejoras, se llevaron a cabo las pruebas en entornos reales. Estas pruebas, aparte de cumplir con las condiciones presentadas en el apartado anterior, incorporaron a una persona que tenía conocimientos de la aplicación para ayudar a los usuarios en caso de que tuviesen dudas sobre algún elemento del sistema. En general, los usuarios cumplieron con todas las tareas de forma satisfactoria y sin la necesidad de mucha guía. En general, todos tuvieron una muy buena impresión de la aplicación, tanto a nivel visual, como a nivel de funcionalidad. Sin embargo, se detectaron algunos detalles a tener en cuenta.

- En primer lugar, la exploración. En lo referente a la ciudad, a la mayoría de usuarios les gustaba explorar la zona de inicio. De hecho, los usuarios buscaban la forma, moviéndose por la escena, de encontrar un “hueco” para entrar en la zona de registro (ver Sección 4.2.3). Aquí aparecen las películas a valorar para poder entrar en la aplicación cuando el usuario se registra o cuando entra como invitado. Los usuarios encontraban esto divertido y satisfactorio, ya que creían haber encontrado la forma de engañar al sistema y entrar en una zona secreta, aunque no era así.

- En lo referente al teclado, si bien las personas no tenían problemas a la hora de introducir los datos, ciertamente algunos usuarios tenían que acercarse mucho para darse cuenta de que había que interactuar con él y que no podían entrar directamente por la puerta del teatro.

- Algunos usuarios, al fallar en poner las credenciales y no recibir una respuesta inmediata, se frustraban porque no se les mostraba el error específico, fallo en la contraseña o en el usuario. Información que se les indicó que no se les iba a facilitar para no poner en riesgo la integridad de los datos de los usuarios.

4.4. Documentación y Despliegue

Esta fase corresponde a la última del proyecto. Su misión es la de redactar una documentación que incluya toda la información relevante de SR SCENE. Además, en esta fase también se prepara la aplicación para que sea completamente funcional y esté dockerizada, de manera que cualquiera pueda usarla. La organización de esta fase en el proyecto, queda plasmada en la Tabla 4.8.

Fase	Información / Subtareas	Inicio	Fin
Documentación	#COOR01, #COOR02, #COOR03, #COOR04	1/10/25	13/1/26
Despliegue	#SETUP01	8/1/26	14/1/26

Tabla. 4.8: Planificación de la fase de documentación y despliegue del proyecto

Fase	Herramienta
Documentación	Overleaf C
Despliegue	Docker, gitlab

Tabla. 4.9: Principales herramientas utilizadas

4.4.1. Documentación

La elaboración de la documentación del proyecto ha sido un proceso largo, complejo y continuo. Han existido dos fases en el proceso: una general y otra específica.

En la documentación de carácter general se ha seguido un proceso continuo. Esta fase se inició al comenzar el proyecto. Su misión principal era monitorizar el desarrollo del proyecto en todos sus aspectos. Además, esta documentación, una vez finalizada, debía incluir diferentes documentos de carácter específico, incluidos en los apéndices (ver Apéndices A, B y C), que se desarrollaron a la par de esta. Como resultado, esta memoria de TFM se corresponde al documento final nacido de unificar toda la planificación y monitorización del sistema, con los Apéndices A, B y C. En cuanto al formato. La documentación se ha elaborado utilizando la aplicación web Overleaf. Esta aplicación web permite la creación de documentos compartidos en LaTeX. (ver Apéndice C)

La documentación específica se traduce en la realización de documentos que correspondan con actividades concretas que se realizan una vez conseguido un hito o una acción importante que define gran parte del proyecto. Esta documentación viene determinada por unas tareas concretas que han sido indicadas previamente en la Sección 4.8. Los principales documentos que se han elaborado son:

- Manual de usuario:
  - Se ha creado un manual de usuario y de implementación. Lo que se busca es que tanto el usuario final, como el administrador del backend, sean

capaces de conocer todos los procesos relacionados con la descarga, implementación y uso del sistema.
    - Esta documentación se encuentra en este proyecto en los Apéndices A y B.

- Funcionalidad del modelo
    - La funcionalidad del modelo debe indicar las capacidades funcionales del sistema.
    - Este apartado queda reflejado a lo largo de todo el documento, pero mas específicamente en el apartado de Diseño del sistema, en el Capítulo 3.

4.4.2. Despliegue

Esta fase se corresponde a la última fase del proyecto. Lo que se busca en esta fase en concreto, teniendo previamente todo el sistema terminado, probado y documentado, es hacer un despliegue del sistema en un entorno real. De tal forma que la aplicación fuese capaz de comercializarse sin ningún problema.

Una vez cumplidas todas las condiciones, se lleva a cabo el despliegue del sistema. Para ello, se ha creado un Docker del backend del sistema. Este Docker presenta la siguiente estructura:

- Contenedor con el código principal del proyecto, de tal forma que una vez se ejecute, el sistema queda escuchando peticiones del cliente y respondiendo.
- Contenedor con la función update. El flujo natural es el siguiente. Se para el contenedor principal, se ejecuta este contenedor para actualizar la cartelera y finalmente se vuelve a ejecutar el programa principal.
- En lo referente a la BBDD. Ambos contenedores trabajan sobre la misma BBDD.

Como nota adicional, indicar este sistema se ha desplegado en un servidor de la Universidad de Jaén, propiedad del grupo de investigación Sinbad2, llamado Sinbad2ia. Un factor importante a tener en cuenta es que la universidad controla el acceso al mismo y solo permite el acceso a aquellos usuarios conectados a redes cuyo origen proviene de la misma universidad.

Capítulo 5

RESULTADOS, CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Resultados

Una vez que el sistema está completamente operativo, se puede discutir sobre los resultados obtenidos. En lo referente a los requisitos funcionales que se debían cumplir y que fueron indicados en la Sección 3.1.1, todos se han cumplido de forma exitosa. Es por todo ello que se puede afirmar que:

- El sistema permite el registro de nuevos usuarios.
- El sistema permite identificar a usuarios registrados.
- Se ha optado por una implementación de un usuario de tipo invitado, que permite eliminar toda la información registrada una vez el usuario cierra la sesión, lo que resulta muy eficiente.
- En cuanto a la valoración de películas. Se consigue crear la ilusión al usuario de que el proceso es inmediato. Además, no existe ningún riesgo de perdida de memoria o sobrescritura. Esto se debe a que el programa en todo momento tiene los resultados de las acciones del usuario en memoria, aunque tarde un poco en escribirlos en la BBDD.
- Las recomendaciones del SRFC, resultan muy rápidas y consiguen que el usuario acceda a una gran variedad de recomendaciones a partir de unos gustos mínimos introducidos. Además, a medida que introduce nuevas valoraciones, se consiguen mejores recomendaciones sin perder la variedad en las mismas.
- El SRBC es muy eficiente y cumple con todos los requisitos que en un primer momento se marcaron.
- Se tiene una cartelera actualizada semanalmente. En caso de no poder acceder a la cartelera, el sistema utiliza la cartelera de la semana anterior, guardada previamente. Por lo que se controla un posible fallo de la API de TMDB.

Por otra parte, es importante destacar la eficiencia del sistema, que es capaz de responder a las peticiones del usuario en tiempos aceptables. En particular, en las operaciones de escritura de datos en la BBDD, el sistema mantiene en memoria, durante la ejecución, la versión más reciente y coherente de la información. Esto implica que, aunque se produjera un error puntual durante una operación de escritura en la base de datos, dicha inconsistencia podría corregirse en una escritura posterior, ya que el sistema continúa trabajando sobre la versión actualizada de los datos almacenados en memoria antes de persistirlos de forma definitiva.

Finalmente, con todo esto en mente, se ha conseguido desarrollar un SR con un frontend en RV. Este sistema combina varios tipos de recomendaciones, supliendo así las necesidades y carencias que cada tipo de SR presenta por separado. Además, es un sistema muy eficiente y con el que se puede trabajar de forma sencilla e incluir futuras mejoras.

5.2. Líneas de mejora

En lo referente a las mejoras que se podrían implementar en este sistema, se han detectado principalmente dos.

La primera es acerca de las películas. Una buena función que se podría agregar es la de poder reproducir películas. Este sistema en un principio se imaginó como un SR de películas y no uno de visualización. Sin embargo, es algo innegable que ambos sistemas hoy en día son co-dependientes, aunque no es la primera vez que se han creado de forma separada. Es por ello que una mejora planteada es la de poder permitir al usuario ver películas en el frontend. Existiría dos formas de implementarlo, mediante una API de Netflix o Amazon Prime Video, la cual permita ver las películas, gastos aparte. Otra forma sería obtener las películas y guardarlas en la BBDD, lo cual incrementaría bastante el tamaño del contenedor. Por lo que, en definitiva, es una decisión que pone en una balanza la inversión económica, el trabajo asociado a la creación del sistema y la BBDD.

La segunda es mejora estaría relacionada con la Blockchain [21], donde se podría implementar un sistema seguro y confiable donde vender cada imagen de la cartelera a cada usuario mediante un precio preestablecido. Obviamente, sería necesario tener grandes conocimientos de Blockchain y trabajar con una empresa que gestionase los pagos.

5.3. Conclusiones

El desarrollo de este trabajo pone de manifiesto las dificultades a las que se enfrenta cualquier persona que decide abordar un proyecto informático de cierta envergadura. En primer lugar, la planificación resulta un elemento fundamental. Sin una

planificación inicial bien estructurada y detallada, como la realizada en este proyecto, habría sido complicado completar el trabajo en el tiempo y la forma establecidos.

Otro aspecto especialmente relevante ha sido la búsqueda de información y de conjuntos de datos adecuados. Este proceso ha requerido un trabajo de investigación considerable, así como la limpieza y adaptación de distintas BBDD, una tarea que ha resultado más compleja de lo esperado. A ello se suma la dificultad inherente a trabajar con múltiples componentes, lenguajes de programación y herramientas desarrolladas por terceros. Integrar todos estos elementos en un único sistema funcional es un proceso complejo que, aunque ocurre de forma habitual en el desarrollo de software, no siempre es suficientemente valorado.

No obstante, si hay un elemento que merece especial atención, es el uso de tecnologías de RV, en este caso las gafas Meta Quest. Estas han supuesto uno de los mayores retos del proyecto, no solo por su complejidad técnica, sino también por las dificultades adicionales asociadas a su configuración, integración y mantenimiento dentro del sistema.

Como conclusión final, se puede afirmar que se ha desarrollado un SR de películas completo y funcional. Aunque sería necesaria una actualización periódica de la BBDD para incorporar los estrenos más recientes, las recomendaciones generadas resultan adecuadas y el sistema cumple con los requisitos esperables de una solución de este tipo. Además, el proyecto presenta potencial tanto desde un punto de vista comercial, al tratarse de un sistema de acceso gratuito, como desde un enfoque divulgativo, gracias a una interfaz gráfica intuitiva y atractiva para el público general.

Apéndice A

Manual de Usuario

Este manual tiene como fin último dar a conocer al usuario final del sistema los pasos que debe llevar a cabo para la instalación y el uso de la aplicación.

En primer lugar, debe descargarse la aplicación desde el Drive oficial: https://drive.google.com/drive/u/1/folders/1zRLUY2QznQ8NTt1zliTQhFdz1TnUmoWt. Una vez descargado el archivo, se debe descomprimir el contenido. Al hacerlo, se generará una carpeta; dentro de esta, deberá ejecutarse el archivo correspondiente para iniciar la aplicación.

De forma previa a la ejecución, es necesario cumplir una serie de requisitos:

- Disponer de unas gafas de realidad virtual Meta Quest 3.
- Tener una cuenta de Meta.
  - En caso de no disponer de una, deberá crearse. Existen varias formas de hacerlo:
    - Descargar la aplicación móvil Meta Horizon, disponible en la Play Store. Al ejecutarla por primera vez, se solicitará la creación de una cuenta.
    - Descargar la aplicación para PC Meta Quest Link desde la página oficial de Meta. Durante el proceso inicial de configuración, se solicitará la creación de la cuenta.
    - En ambos casos introducir la información requerida y se creara la cuenta automáticamente
  - Dado que se utilizará obligatoriamente Meta Quest Link, se recomienda emplear este método.
- Instalar la aplicación Meta Quest Link.
  - Verificar que el PC cumple con las especificaciones técnicas necesarias.
  - Las especificaciones pueden consultarse en el siguiente enlace: https://www.meta.com/es-es/help/quest/140991407990979/
- Configuración en Meta Quest Link (ver Imagen A.1):

- Acceder a las opciones de configuración y, dentro del apartado *General*, habilitar la opción que permite ejecutar aplicaciones de origen desconocido.
- En caso de que la opción esté disponible, habilitar OpenXR en tiempo de ejecución. Si no se encuentra activada por defecto, deberá activarse manualmente.

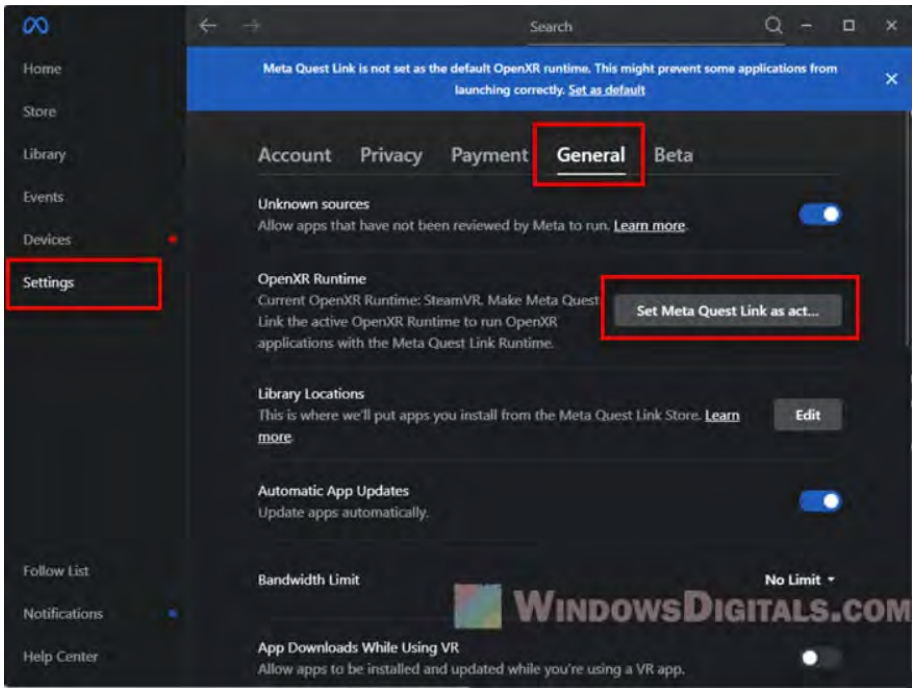


Figura A.1: Imagen manual: Pantalla configuración Meta, Fuente: consultar Apéndice C

Una vez cumplidos todos los requisitos previos y descargado el ejecutable, se deben conectar las gafas de realidad virtual al ordenador mediante un cable USB tipo C. Tras la conexión, aparecerá una ventana en las gafas solicitando la activación de Meta Quest Link. Se deberá aceptar y esperar a que cargue la interfaz de Link.

Cuando la carga haya finalizado, se ejecutará el programa desde el ordenador mediante el archivo `.exe`. En las gafas se mostrará el logotipo de Unity y, a continuación, el usuario accederá a la primera escena de la aplicación.

En esta escena, el usuario deberá interactuar con el teclado para seleccionar una de las siguientes opciones (ver Imagen A.2):

- Crear usuario
- Iniciar sesión
- Cerrar sesión
- Entrar a la aplicación como invitado

Figura A.2: Imagen manual: Teclado

Si el usuario accede como invitado o crea un nuevo usuario, será obligatorio valorar al menos 20 de las 40 películas disponibles, las cuales se mostrarán en la escena correspondiente.

Una vez dentro de la aplicación, el usuario podrá seleccionar el tipo de recomendación que desea obtener y acceder a ella atravesando las puertas asociadas a cada opción. Asimismo, el usuario podrá valorar cada película que se le recomiende.

Apéndice B

Manual de Instalación

Por parte del usuario final, todo el funcionamiento del sistema ha sido explicado en el Apéndice A.

El backend del sistema funciona sobre un servidor que tenga Docker instalado. Para ello, es necesario instalar Docker previamente. En una consola Linux, se deben ejecutar los siguientes comandos:

- Actualizar los paquetes del sistema:
  - `sudo apt update`
- Instalar las dependencias necesarias:
  - `sudo apt install -y ca-certificates curl gnupg`
- Añadir la clave GPG oficial de Docker:
  - `sudo install -m 0755 -d /etc/apt/keyrings`
  - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg -dearmor -o /etc/apt/keyrings/docker.gpg`
- Añadir el repositorio de Docker:
  - `echo "deb [arch=$(dpkg -print-architecture) signed-by=/etc/apt/keyrings/docke https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"| sudo tee /etc/apt/sources.list.d/docker.list >/dev/null`
- Instalar Docker:
  - `sudo apt update`
  - `sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugi docker-compose-plugin`

En cuanto al backend, el sistema ha sido implementado como un contenedor Docker y se encuentra desplegado mediante un sistema de integración y despliegue continuo (CI/CD) utilizando GitLab: http://serezade.ujaen.es:8030/oog/backendscene/-/tree/main?ref_type=heads. Esto implica que cualquier cambio realizado en el código se refleja automáticamente en el servidor al actualizar el repositorio correspondiente.

Dentro del despliegue Docker se encuentran los siguientes elementos:

- Código del programa:
  - Contiene la totalidad del código del backend. Cualquier modificación en este afecta directamente a su funcionamiento.
- Archivo `docker-compose.yml`:
  - Define el comportamiento del sistema Docker.
  - Especifica los dos contenedores del sistema: el contenedor principal y el contenedor encargado de realizar las actualizaciones.
  - El contenedor principal se denomina `app` y el contenedor de actualización `actualizar`.
- Archivo `gitlab-ci.yml`:
  - Define la conexión con el servidor.
  - Especifica las órdenes necesarias para el despliegue automático del sistema.

El sistema se encuentra completamente preparado para su funcionamiento. En caso de querer desplegarlo en otro servidor, únicamente será necesario modificar la información correspondiente en el archivo `gitlab-ci.yml` para indicar el nuevo servidor de destino.

Actualmente, el sistema está activo y conectado a un servidor. Cualquier cambio en el código o en la estructura de los contenedores Docker se realiza mediante modificaciones en el repositorio Git, las cuales se despliegan automáticamente. No obstante, el servidor que aloja los contenedores debe contar con un seguimiento periódico por parte del personal técnico.

Este seguimiento consiste en detener semanalmente el contenedor principal y ejecutar el proceso de actualización. Para ello, desde la consola de Linux, se deben ejecutar los siguientes comandos:

- Detener el contenedor principal:
  - `docker stop app`
- Ejecutar el contenedor de actualización:

    - `docker compose run -rm actualizar`

- Iniciar nuevamente el contenedor principal:

    - `docker compose up app`

En caso de que el contenedor principal se detenga de forma inesperada, se deberá ejecutar el comando `docker compose up app` desde la consola de Linux para restaurar el servicio.

Apéndice C

Fuentes de datos

En este apartado se indican las referencias de las BBDD y assets que se han utilizado en el desarrollo del proyecto. De esta forma, cualquiera puede acceder a las fuentes originales sin modificar. Todos los recursos son de acceso libre en el momento que se está redactando este documento.

- Base da datos
  - MovieLens
    - Se ha utilizado de base la versión de 1 millón. Fuente
    - La versión 1 millón, con los tags de usuarios, se ha obtenido de un Git público. Fuente
    - Las imágenes se han obtenido de un Git público. Fuente
  - TMDB
    - La BBDD de la cartelera se obtiene de tmdb usando su API oficial
- Assests de Unity
  - Teclado (Fuente)
  - Decoración visual
    - Fuente 1: https://assetstore.unity.com/packages/3d/environments/gwangju-theater-282533
    - Fuente 2: https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224
    - Fuente 3: https://assetstore.unity.com/packages/3d/environments/simplepoly-city-low-poly-assets-58899
- Otras fuentes de investigación y librerías destacadas
  - LibRecommender
    - Biblioteca de SR para Python
    - Fuente: https://pypi.org/project/LibRecommender/

    - Overleaf
        - Editor de texto LaTeX para la creación de documentos
        - Fuente: https://es.overleaf.com/

- Imágenes adicionales. Todas las imágenes han sido obtenidas de Google Imágenes
    - Figura 2.1 (Fuente)
    - Figura 2.2 (Fuente)
    - Figura 2.3 (Fuente)
    - Figura 2.4 (Fuente)
    - Figura 2.5 (Fuente)
    - Figura 2.6 (Fuente)
- Adicionalmente, en al manual de usuario, se ha usado una imagen sobre configuración del sistema. Esa imagen es de un articulo que explica como arreglar problemas con el OpenXR, esto es muy raro y no debería darse.
    - Fuente: https://www.windowsdigitals.com/why-set-meta-quest-link-as-default-op

Bibliografía

[1] Vr cinema, 2026. URL https://play.google.com/store/apps/details?id=com.fibrum.vrcinema&hl=es. Comprobado en 2026-1-7.

[2] ¿es roblox seguro para nuestros hijos? guía de seguridad para padres, 2026. URL https://www.qustodio.com/es/blog/is-roblox-safe-for-kids-app-safety-guide-for-parents/. Comprobado en 2026-1-7.

[3] Modelo de cascada (waterfall): qué es y cuándo conviene usarlo, 2025. URL https://blog.ganttpro.com/es/metodologia-de-cascada/. Comprobado en 2025-11-07.

[4] Qué es la metodología waterfall y cuándo utilizarla, 2025. URL https://asana.com/es/resources/waterfall-project-management-methodology. Comprobado en 2025-11-07.

[5] Edt: cómo hacer una para tu proyecto con un ejemplo, 2025. URL https://asana.com/es/resources/work-breakdown-structure. Comprobado en 2025-11-07.

[6] ¿qué es la matriz de trazabilidad de requisitos (rtm) en las pruebas?, 2025. URL https://www.guru99.com/es/traceability-matrix.html. Comprobado en 2025-11-07.

[7] ¿cuánto gana un programador en españa?, 2025. URL https://es.talent.com/salary?job=programador. Comprobado en 2025-12-23.

[8] Netflix prize, 2025. URL https://en.wikipedia.org/wiki/Netflix_Prize. Comprobado en 2025-11-04.

[9] The netflix prize: How a $1 million contest changed binge-watching forever, 2025. URL https://www.thrillist.com/entertainment/nation/the-netflix-prize. Comprobado en 2025-11-04.

[10] La paradoja de elección: muchas opciones, pocas decisiones., 2025. URL https://psicopico.com/la-paradoja-de-eleccion-muchas-opciones-pocas-decisiones/. Comprobado en 2025-11-04.

[11] An introduction recomender systems., 2025. URL https://www.cambridge.org/core/books/recommender-systems/C6471B59388D8A9F684C49C198691B53. Comprobado en 2025-11-04.

[12] Transformer (deep learning architecture), 2025. URL https://en.wikipedia.org/wiki/Transformer_%28deep_learning_architecture%29. Comprobado en 2025-11-04.

[13] Bert (modelo de lenguaje), 2025. URL https://es.wikipedia.org/wiki/BERT_%28modelo_de_lenguaje%29. Comprobado en 2025-11-04.

[14] ¿qué es el metaverso?, 2025. URL https://www.meta.com/es-es/metaverse/what-is-the-metaverse/. Comprobado en 2025-11-04.

[15] Historia de la realidad virtual, 2025. URL https://es.wikipedia.org/wiki/Historia_de_la_realidad_virtual. Comprobado en 2025-11-04.

[16] Qué es el metaverso, qué posibilidades ofrece y cuándo será real, 2025. URL https://www.xataka.com/basics/que-metaverso-que-posibilidades-ofrece-cuando-sera-real. Comprobado en 2025-11-04.

[17] Mejores gafas de realidad virtual. cuál comprar y siete modelos recomendados para todos los presupuestos, 2025. URL https://www.xataka.com/seleccion/mejores-gafas-realidad-virtual-cual-comprar-siete-modelos-recomendados-para-todos
Comprobado en 2025-11-04.

[18] Mejor motores de juegos de realidad virtual (vr), 2025. URL https://www.g2.com/es/categories/vr-game-engine. Comprobado en 2025-11-04.

[19] ¿qué es docker?, 2025. URL https://www.ibm.com/es-es/think/topics/docker. Comprobado en 2025-11-04.

[20] Contenedores docker: Ventajas para el desarrollo y despliegue de aplicaciones, 2026. URL https://www.sysadminok.es/blog/administracion-de-sistemas/contenedores-docker-ventajas-para-el-desarrollo-y-despliegue-de-aplicaciones/. Comprobado en 2026-1-7.

[21] Cadena de bloques, 2025. URL https://es.wikipedia.org/wiki/Cadena_de_bloques. Comprobado en 2025-12-17.