Universidad Politécnica

de Madrid

**Escuela Técnica Superior de**

**Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**APLICACIÓN DE GESTIÓN DE UNA BASE DE DATOS DE VIDEOJUEGOS**

Autor: Katherin Ambia Payano

Tutor(a): Antonio Latorre De La Fuente

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Grado*

*Grado en Ingeniería Informática*

*Título:* *APLICACIÓN DE GESTIÓN DE UNA BASE DE DATOS DE VIDEOJUEGOS*

Mes Año

*Autor:* Katherin Ambia Payano

*Tutor:*

Antonio Latorre De La Fuente

<<departamento>>

ETSI Informáticos

Universidad Politécnica de Madrid

**Resumen**

El siguiente trabajo implementa una aplicación web que gestiona una base de datos de videojuegos.

La memoria esta divida en 10 apartados (secciones) de las cuales los apartados de especificación, diseño, desarrollo, prueba y mantenimento llevan los nombres de las actividades de Ingeniería de Software. Para el desarrollo del apartado de **especificación de requisitos** se llevo acabo actividades de obtencion, análisis, especificación y validación de requisitos con el fin de obtener la documentacion de especificacion de requisitos donde se muestran principalmente las funcionalidades especificas y no especificas del software. La estructura de este apartado hace uso de estandar IEEE 830:1980.

En el aparatado de diseño se ha desarrollada el diseño de arquitectura o de alto nivel y el diseño detallado. Haremos uso diagramas como DFD (Diagrama de flujo de datos), E/R (Entidad-Relacion), diccionarios de Datos entre otros. Tambien haremos uso de notaciones como UML. El diseño de los paths se basa en los principios API-REST.

**Abstract**

<<Abstract of the Final Degree Project. Maximum length: 2 pages.>>

The following paper implements a web application that manages a video game database.

The report is divided into 10 sections of which the specification, design, development, testing and maintenance sections are named after the Software Engineering activities. For the development of the requirements specification section, activities were carried out to obtain, analyze, specify and validate requirements in order to obtain the requirements specification documentation where the specific and non-specific functionalities of the software are mainly shown. The structure of this section makes use of the IEEE 830:1980 standard.

In the design apparatus, architectural or high-level design and detailed design have been developed. We will make use of diagrams such as DFD (Data Flow Diagram), E/R (Entity-Relation), Data dictionaries among others. We'll also make use of notations like UML. The design of the paths is based on API-REST principles.

**Plan de trabajo**

En la primera entrega los objetivos contaban con una breve descripción. Los objetivos como tal no han cambiado, pero la descripción ha variado ya que se ha accedido a más fuentes de información. Lo mismo sucede con las lista de tareas. Los tiempos en la fase de especificación han variado por que es la fase donde hay que realizar actividades claves para obtener los requisitos funcionales y no funciones. Hay que analizar las funcionalidades y validarlas usando técnicas que aseguren que son realmente son las funcionalidades que el cliente desea y por lo tanto require de más tiempo de lo que se planificó.

**Lista de objetivos**

* Identificación y especificación de requisitos. Para la identificación y especificación de requisitos se lleva a cabo actividades de obtención, análisis, especificación y validación de requisitos con el fin de elaborar el Documentos de requisitos del software
* Diseño de la herramienta software (aplicación web).

Desarrollo del diseño de la arquitectura o de algo nivel.

Diseño detallado.

Diseño de paths.

Para el desarrollo del diseño de hará uso de diagramas como DFD, diagrama E/R y diccionario de datos. También haremos uso de notaciones como UML.

* Implementación de la herramienta (aplicación web). Se hará usando el framework Django.
* Validación de la herramienta (aplicación web) con un plan de pruebas que cubra escenarios (casuísticas) de distinta complejidad.
* Empaquetado de la herramienta como un contenedor Docker.

**Lista de tareas**

* Estudio de la metodología de lenguaje Python y framework Django. (6 horas)
* Especificación de requisitos.
* Actividades de obtención.
* Actividades de análisis.
* Actividades de especificación.
* Actividades de validación
* Documentación de especificación de requisitos.
* Diseño de la aplicación. (55 horas).
* Desarrollo de diseño de la arquitectura o de alto nivel
* Diseño detallado.
* Diseño de paths.
* Implementación (codificación y base de datos) de la aplicación. (165 horas)
* Implementar las tablas de la base de datos.
* Implementar back end.
* Implementar frond end.
* Realización de pruebas para las distintas casuísticas. (28 horas)
* Empaquetado de la aplicación como un contenedor Docker.
* Documentación del trabajo.
* Presentación y defensa del trabajo.

Tabla de contenidos

[1 Introducción 1](#_Toc153846864)

[1.1 Objetivos 1](#_Toc153846865)

[2 Especificación de Requisitos 2](#_Toc153846866)

[2.1 Introducción 2](#_Toc153846867)

[2.1.1 Propósito 2](#_Toc153846868)

[2.1.2 Alcance 2](#_Toc153846869)

[2.1.3 Definiciones, siglas y abreviaturas 2](#_Toc153846870)

[2.1.4 Referencias 2](#_Toc153846871)

[2.1.5 Visión general 2](#_Toc153846872)

[2.2 Descripción General 2](#_Toc153846873)

[2.2.1 Perspectiva del producto 2](#_Toc153846874)

[2.2.2 Funcionalidades 3](#_Toc153846875)

[2.2.3 Características de usuario 3](#_Toc153846876)

[2.2.4 Restricciones 3](#_Toc153846877)

[2.2.5 Asunciones y dependencias 3](#_Toc153846878)

[2.3 Requisitos específicos 3](#_Toc153846879)

[2.3.1 Requisitos funcionales 3](#_Toc153846880)

[2.3.2 Requisitos no funcionales 3](#_Toc153846881)

[2.3.2.1 Requisito de usabilidad: 3](#_Toc153846882)

[2.3.2.2 Requisito de entorno de ejecución: 3](#_Toc153846883)

[2.3.2.3 Atributos de calidad: 3](#_Toc153846884)

[2.3.2.4 Requisitos de seguridad: 3](#_Toc153846885)

[2.3.2.5 Requisito de rendimiento: 3](#_Toc153846886)

[2.4 Representación tabular de casos de uso de los requisitos. 4](#_Toc153846887)

[2.5 Representación de requisitos mediante lenguaje formal. 7](#_Toc153846888)

[3 Diseño 11](#_Toc153846889)

[3.1 Diseño de arquitectura o de alto nivel. 11](#_Toc153846890)

[3.1.1 Componentes del sistema. 12](#_Toc153846891)

[3.1.2 Diseño físico de la base de datos. 12](#_Toc153846892)

[3.2 Diseño detallado 12](#_Toc153846893)

3.3 [Diseño de paths 2](#_Toc153846894)

[4 Desarrollo 2](#_Toc153846895)

[4.1 Desarrollo de backend 2](#_Toc153846896)

[4.1.1 implementación de pseudocódigo 2](#_Toc153846897)

[4.1.2 implementación de código. 2](#_Toc153846898)

[4.2 Desarrollo de frontend 2](#_Toc153846899)

[5 Pruebas 2](#_Toc153846900)

[5.1 Pruebas unitarias 2](#_Toc153846901)

[5.2 Pruebas de sistema 2](#_Toc153846902)

[6 Mantenimiento 2](#_Toc153846903)

[7 Resultados y conclusiones 3](#_Toc153846904)

[8 Análisis de Impacto 4](#_Toc153846905)

[9 Bibliografía 5](#_Toc153846906)

[10 Anexos 6](#_Toc153846907)

# Introducción

Las bases de datos de videojuegos son de gran importancia en la industria de los videojuegos por que permiten almacenar y organizar de manera eficiente información sobre los videojuegos, como títulos, géneros, desarrolladores, fechas de lanzamiento, plataformas. Una aplicación de base de datos de videojuegos con un front end intuitivo seria beneficiosa para usuarios que buscan información sobre videojuegos.

Este TFG implementa una aplicación que gestiona una base de datos de videojuegos. Para el desarrollo del software llevaremos a cabo las siguientes actividades de Ingeniería de Software: especificación de requisitos, diseño, desarrollo, pruebas y mantenimiento.

El modelo de ciclo de vida del software que aplicaremos para este TFG será el modelo cascada que permita la iteración entre fases para la corrección de errores. También aplicaremos el modelo de proceso casados en prototipos para validad las diferentes vistas del front-end.

Para el diseño de los paths (rutas o enpoints) de las operaciones CRUD (GET, POST, PUT y DELETE), nos basaremos en los principios REST.

Para el control de versiones del TFG hemos usado github.

## Objetivos

* Identificación y especificación de requisitos. Para la identificación y especificación de requisitos se lleva a cabo actividades de obtención, análisis, especificación y validación de requisitos con el fin de elaborar el Documentos de requisitos del software
* Diseño de la herramienta software (aplicación web).

Desarrollo del diseño de la arquitectura o de algo nivel.

Diseño detallado.

Diseño de paths.

Para el desarrollo del diseño de hará uso de diagramas como DFD, diagrama E/R y diccionario de datos. También haremos uso de notaciones como UML.

* Implementación de la herramienta (aplicación web). Se hará usando el framework Django.
* Validación de la herramienta (aplicación web) con un plan de pruebas que cubra escenarios (casuísticas) de distinta complejidad.
* Empaquetado de la herramienta como un contenedor Docker.

# Especificación de Requisitos

Para la obtención del Documento de especificación de requisitos del software (SRS) se lleva a cabo actividades de obtención, análisis, especificación y validación de requisitos para nuestra aplicación. El objetivo de realizar estas actividades de requisitos es obtener información para realizar el Documento de especificación de requisitos, documento clave para llevar a cabo las tareas de diseño, desarrollo y pruebas. En este documento solo se establece lo que hará la aplicación.

Para la estructura del apartado de Especificación de Requisitos nos basaremos en el estándar IEEE 830-1998.

## Introducción

### Propósito

El propósito del apartado de Especificación de Requisitos Software (ERS) es **proporcionar descripción detallada y precisa de las funcionalidades de la aplicación.**

### Alcance

* Nombre de la aplicación: GamesDB
* Función de la aplicación: Muestra, descarga información sobre juegos y se podrá guardar comentarios de los juegos.

### Definiciones, siglas y abreviaturas

* Software: Tecnología o producto resultante de las actividades de Ingeniería de Software.
* Actividad: es un proceso tiene lugar en el tiempo y espacio, y en el cual un agente actúa con unos objetivos determinados.
* IEEE 830-1998: para la especificación de requisitos del software describe el contenido que debe tener un buen documento de especificación de requisitos.
* Actividades de requisitos: son obtención, análisis, especificación y validación de requisitos software.
* Requisitos Software:
* Usuario: Personal que usara la aplicación.
* Especificación de requisitos: Proceso de documentar el comportamiento requerido de un sistema software.
* Requisito funcional: específica una función que un sistema o componente de un sistema debe de ser capaz de llevar a cabo.
* Operaciones CRUD:
* Requisito no funcional: especifican aspectos técnicos que debe incluir el sistema y que pueden clasificarse en restricciones y calidades

### Referencias

* Estándar IEEE 830.

### Visión general

## Descripción General

### Perspectiva del producto

Aplicación web que permita gestionar una base de datos de videojuegos. La aplicación deberá permitir llevar a cabo operaciones CRUD (GET, POST, PUT and DELETE) sobre los datos, además de descargar la información (de los videos juegos) del juego desde la aplicación.

### Funcionalidades

Muestra información de los juegos, permite descargarse esta información y guardar comentarios de los usuarios sobre los juegos.

### Características de usuario

**Usuarios finales:** Personas que buscan información sobre videojuegos, insertar comentarios de videojuegos, descargar información de videojuegos y hacer uso de la API.

**Administradores:** Usuarios responsables de mantener actualizada la base de datos de videojuegos.

### Restricciones

* No es una aplicación donde se pueda jugar.
* Se usará el framework Django para desarrollar la aplicación.

### Asunciones y dependencias

* Se asume que los usuarios tendrán acceso a internet para usar la aplicación.
* La aplicación dependerá de un servidor web y una base de datos relacional.

## Requisitos específicos

### Requisitos funcionales

A continuación, se detallarán las funcionalidades que lleva a cabo la aplicación.

* La aplicación debe permitir **crear al usuario** un identificador y contraseña.
* Debe permitir al usuario o usuario anónimo **buscar información sobre video juegos**.
* La aplicación debe permitir al usuario o usuario anónimo acceder a las url de noticias sobre video juegos.
* Después de registrarse el usuario, puede loggearse.
* El usuario puede reestablecer la contraseña.
* El usuario puede insertar comentarios que servirán de feedback para otros usuarios con el fin de mejorar la experiencia de usuario.
* El usuario puede modificar su comentario.
* El usuario puede eliminar su comentario.
* El usuario imprimir información sobre el video juego.
* El usuario puede cerrar sesión.
* Eliminar cuenta.

### Requisitos no funcionales

#### Requisito de usabilidad:

* Debe de tener una interfaz intuitiva (se utilizarán iconos que permite al usuario), usable (de fácil manejo).
* Tendrá una barra de menú en la parte superior de la pantalla fija
* Deben de mostrarse los datos del usuario que esté conectado.
* Mostrar un ranking de los videojuegos más buscados.

#### Requisito de entorno de ejecución:

* Al ser una aplicación que se va a dockerizar, se podrá … en cualquier entorno.

#### Atributos de calidad:

* El tiempo de respuesta será?????

#### Requisitos de seguridad:

* Se protegera la información del usuario, mediante encriptaciones.

#### Requisito de rendimiento:

* Como vamos a realizar una aplicación que no realizan operaciones de gran complejidad no tenemos requisitos de rendimiento.

#### Requisitos no clasificados

#### La aplicación debe estar dockerizada.

La aplicación debe estar una API.

## Representación tabular de casos de uso de los requisitos.

|  |
| --- |
| **R1**  **Código:** CU0001  **Nombre:** Crear usuario/ Registrar usuario.  **Autores:** Usuario final.  **Descripción:** Crear un usuario para acceder a la aplicación. Tiene que introducir nombre de usuario y crear una contraseña.  **Precondiciones:** No estar registrado.  **Postcondiciones:** Se crea un nuevo usuario. Y sus datos se registran en la base de datos. |

|  |
| --- |
| **R2**  **Código:** CU0003  **Nombre:** Buscar juegos.  **Autores:** Usuario final.  **Descripción:** Filtrar juegos según atributos.  **Precondiciones:** El usuario o usuario anónimo puede estar o no registrado como  **Postcondiciones:** El usuario o usuario anónimo puede visualizar el juego o juegos según los filtros. |

|  |
| --- |
| **R3**  **Código:** CU0009  **Nombre:** Acceder a link de noticias sobre videojuegos.  **Autores:** Usuario o usuario anonimo.  **Descripción:** El usuario visualiza las noticias sobre videojuegos.  **Precondiciones:** Haber iniciado sesión o ser un usuario anonimo.  **Postcondiciones:** El usuario visualiza una sesión de las noticias sobre videojuegos. |

|  |
| --- |
| **R4**  **Código:** CU0002  **Nombre:** Usuario inicia sesión.  **Autores:** Usuario final.  **Descripción:** El usuario inicia sesión y puede acceder a la aplicación.  **Precondiciones:** CU0001 El usuario debe de estar registrado en la base de datos.  **Postcondiciones:** El usuario accede a la aplicación. |

|  |
| --- |
| **R5**  **Código:** CU0011  **Nombre:** Reestablecer contraseñar.  **Autores:** Usuario final.  **Descripción:** El usuario en caso de haber olvidado la contraseña tendrá que ir al botón **olvidar contraseña y luego introducir su correo electrónico.**  **Precondiciones:** Tener un usuario existente.  **Postcondiciones:** El usuario reestablece contraseña y puede loggearse. |

|  |
| --- |
| **R6**  **Código:** CU0004  **Nombre:** Hacer comentarios sobre los videojuegos.  **Autores:** Usuario final.  **Descripción:** El usuario hacer comentarios de los juegos.  **Precondiciones:** Ser un usuario.  **Postcondiciones:** Se registra un comentario en la base de datos del videojuego. Este comentario puede ser visto por otros usuarios. |

|  |
| --- |
| **R7**  **Código:** CU0005  **Nombre:** Usuario modifica comentario.  **Autores:** Usuario final.  **Descripción:** El usuario modifica comentario.  **Precondiciones:** Ser usuario y el usuario haber encontrado alguna falta ortográfica por ejemplo.  **Postcondiciones:** Se modifica el comentario. |

|  |
| --- |
| **R8**  **Código:** CU0006  **Nombre:** Usuario elimina comentario.  **Autores:** Usuario final.  **Descripción:** Ser usuario.  **Precondiciones:**  **Postcondiciones:** El usuario elimina el comentario del videojuego. |

|  |
| --- |
| **R9**  **Código:** CU0007  **Nombre:** Imprimir información de videojuegos  **Autores:** Usuario final.  **Descripción:** Usuario imprime información de los videojuegos que filtres.  **Precondiciones:** Ser usuario y haber aplicado los filtros para buscar el videojuego o los videojuegos.  **Postcondiciones:** Se imprime información del videojuego. |

|  |
| --- |
| **R10**  **Código:** CU0010  **Nombre:** Cerrar sesión.  **Autores:** Usuario final.  **Descripción:** El usuario podrá cerrar sesión.  **Precondiciones:** Haber iniciado sesión.  **Postcondiciones:** El usuario no estará loggeado. |

|  |
| --- |
| **R11**  **Código:** CU0012  **Nombre:** Eliminar cuenta.  **Autores:** Usuario final.  **Descripción:** Usuario puede eliminar cuenta.  **Precondiciones:** Usuario debe existir  **Postcondiciones:** Usuario eliminado. |

## Representación de requisitos mediante lenguaje formal.

Definición de espacio de estados

GameDBApp

comentarios = dom comentariousr

usuario: P USUARIO

videojuego: P VIDEOJUEGO

noticia: P NOTICIA

comentario: P COMENTARIO

categoría: P CATEGORIA

comentariousr: P COMENTARIO USUARIO

videojuegocategoria: V

InicializarGameDBApp

GameDBApp

usuario = Ǿ

videojuego = Ǿ

noticia = Ǿ

comentario = Ǿ

categoria = Ǿ

RegistrarUsuario

Δ GameDBApp

usr ? = USUARIO

usr ? € USUARIO

usuario’ = usuario U { usr? }

BuscarVideoJuego

Ξ GameDBApp

vj ? = VIDEOJUEGO

vj ? € VIDEOJUEGO

falta poner notación que devuelve una lista de videos según filtros

AccederUrlNoticia

Ξ GameDBApp

nt? = noticia

nt? € NOTICIA

IniciarSession

Δ GameDBApp

usr ? = USUARIO

usr ? € USUARIO

ReestablecerContraseña

Δ GameDBApp

usr ? = USUARIO

usr ? € USUARIO

usuario’ = usuario  { usr ? }

HacerComentario

Δ GameDBApp

usr ? = USUARIO

vj ? = VIDEOJUEGO

comentario ? = COMENTARIO

usr ? € USUARIO

vj ? € VIDEOJUEGO

comentariousrvj’ = comentariousrvj U { comentariousr? → vj? }

ActualizarComentario

Δ GameDBApp

usr ? = USUARIO

vj ? = VIDEOJUEGO

comentario ? = COMENTARIO

usr ? € USUARIO

vj ? € VIDEOJUEGO

comentariousrvj’ = comentariousrvj { comentariousr? → vj? }

EliminarComentario

Δ GameDBApp

usr ? = USUARIO

vj ? = VIDEOJUEGO

comentario ? = COMENTARIO

usr ? € USUARIO

vj ? € VIDEOJUEGO

comentario ? € COMENTARIO

comentariousrvj’ = { comentario? } comentariousrvj

ImprimirInformacionVideoJuego

Ξ GameDBApp

vj ? = VIDEOJUEGO

vj ? € VIDEOJUEGO

CerrarSesion

Δ GameDBApp

usr ? = USUARIO

usr ? € USUARIO

usuario’ = usuario  { usr? }

EliminarCuenta

Δ GameDBApp

usr ? = USUARIO

usr ? € USUARIO

usuario’ = { usr? }  usuario

# Diseño

En la fase de diseño tomaremos como punto de partida los requisitos funcionales y no funcionales.

En este apartado vamos a mostrar los componentes del sistema y como se relacionan. Para describir el diseño utilizaremos las siguientes técnicas: estilos arquitectónicos y arquitectura de base de datos. Dentro de la técnica estilos arquitectónicos utilizaremos la orientada a objetos.

## Diseño de arquitectura o de alto nivel.

En este apartado mostraremos los componentes del sistema. Utilizaremos el método orientado a datos. Partiremos de diagrama E/R para diseñar las tablas de la base de dato.

### Componentes del sistema.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

PC

Aplicación web

Arquitectura multinivel

### Diseño físico de la base de datos.

Para realizar el diseño de base de datos partiremos del diagrama E/R. Ya que las entidades del diagrama E/R serán las tablas de la base de datos de la aplicación y las elipses las columnas de las tablas.

Diagrama

Descripción generada automáticamente

## Diseño detallado.

## Interfaz de usuario gráfica, Aplicación Descripción generada automáticamente

## Diseño de paths:

Para el diseño de paths haremos uso de REST (Representational State Transfer), una arquitectura para construir servicios web que utiliza los métodos HTTP estándar y los recursos identificados por URLs. Esto nos permitirá crear endpoints claros y organizados para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre nuestros recursos.

Cada endpoint será diseñado para interactuar con las tablas en la base de datos. Los métodos HTTP se utilizarán de la siguiente manera:

* GET para obtener recursos, como la lista de todas las instancias o una instancia en específico.
* POST para crear una nueva instancia de la tabla.
* PUT para actualizar una instancia existente en la tabla.
* DELETE para eliminar una instancia de la tabla.

Los endpoints REST para nuestra API serán:

**Tabla: Juegos**

* GET /juegos para obtener todos los juegos.
* GET /juegos/?nombre=<nombre>&genero=<genero>&plataforma=<plataforma> para obtener un juego específico por nombre, género y plataforma.
* POST /juegos hacer un comentario.

**Tabla: Usuarios**

* POST /autenticación para crear usuario

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

1. Imagen Home

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

1. Vista Registrarse

Estos endpoints seguirán las convenciones REST, permitiendo una fácil comprensión y manipulación de los recursos mediante el uso de URLs significativas y métodos HTTP adecuados.

# Desarrollo

## Desarrollo de backend

Para el desarrollo del backend de la aplicación de gestión de video juegos, he utilizado las siguientes tecnologías:

**Python:** El **lenguaje de programación** utilizado para desarrollar la aplicación.

**Django:** Un **framework** de desarrollo web en Python que facilita la creación de aplicaciones web robustas y escalables.

**SQLite:** **Sistema de gestión de bases de datos relacional** para almacenar los datos de los juegos y usuarios.

**Estructura del proyecto**

Aquí voy a poner una imagen de como esta estructurado las carpetas.

1. Estructura de proyecto

### implementación de pseudocódigo

Para la implementación del pseudocódigo, me he basado en los elementos del framework Django, que incluye modelos, serializadores, vistas y enrutadores. A continuación, se presenta el pseudocódigo para las operaciones CRUD utilizando Django y Django REST Framework.

#### Configuración de la Base de Datos

1. **Conectar a la Base de Datos**

Definir la configuración de la base de datos en **settings.py**

Establecer las credenciales y parámetros necesarios

#### Definicion del modelo

1. **Definición de Modelos**

Crear una clase X que hereda de **models.Model**

Definir los campos necesarios de la clase (titulo, genero, fecha\_lanzamiento, etc.)

#### Serializadores

#### Definir el Serializador

Crear una clase JuegoSerializer que hereda de serializers.ModelSerializer

Especificar los campos que deben ser serializados

#### Crear las Vistas para el CRUD

Crear una clase JuegoViewSet que hereda de viewsets.ModelViewSet

Configurar el queryset y el serializador

#### Definir las Rutas para la API

Configurar un router y registrar el viewset JuegoViewSet

Incluir las rutas en el archivo urls.py

#### Pseudocódigo de Operaciones CRUD

**Obtener Todos los Juegos (GET)**

Recibir una solicitud GET en /api/juegos/ Consultar todos los juegos en la base de datos Serializar los juegos Devolver una respuesta con la lista de juegos y el estado 200.

**Obtener un Juego por parámetros (GET)**

Recibir una solicitud GET en /api/juegos?parametro1=valor1&parametro2=valor2&parametro3=valor3 Consultar el juego con el ID especificado en la base de datos Si el juego existe: Serializar el juego Devolver una respuesta con los datos del juego y el estado 200 Si el juego no existe: Devolver una respuesta de error con el estado 404.

### implementación de código.

**Configuración de la Base de Datos.**

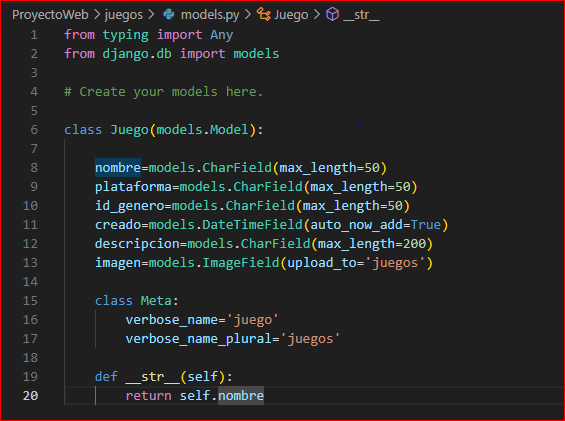
En el archivo settings.py, configuramos la conexión a la base de datos. Texto

Descripción generada automáticamente

1. Fichero settings.py

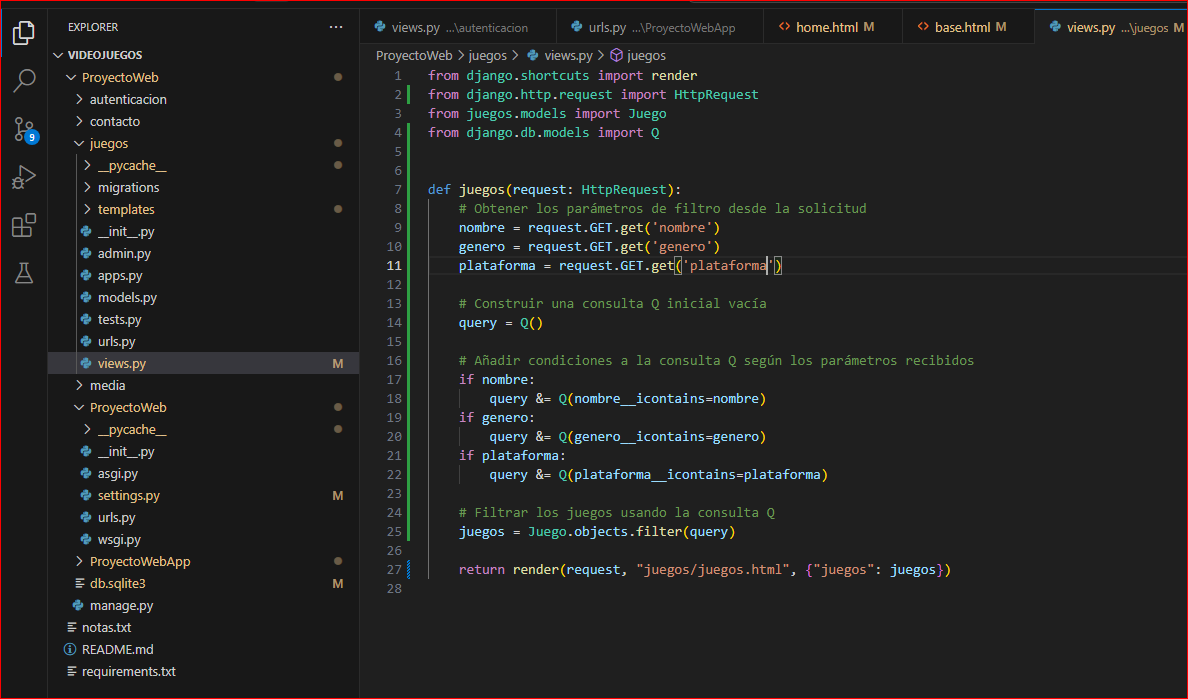
**Definición de Modelos**

En el archivo models.py, definimos el modelo para los juegos:



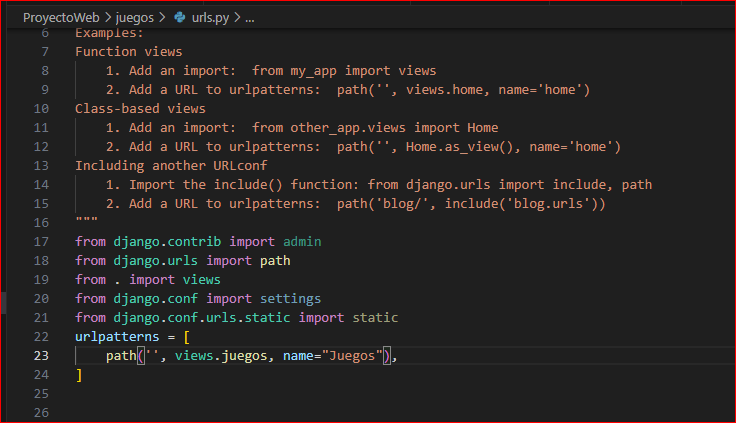
**Vistas**

En el archivo views.py, implementamos las vistas para manejar las solicitudes:

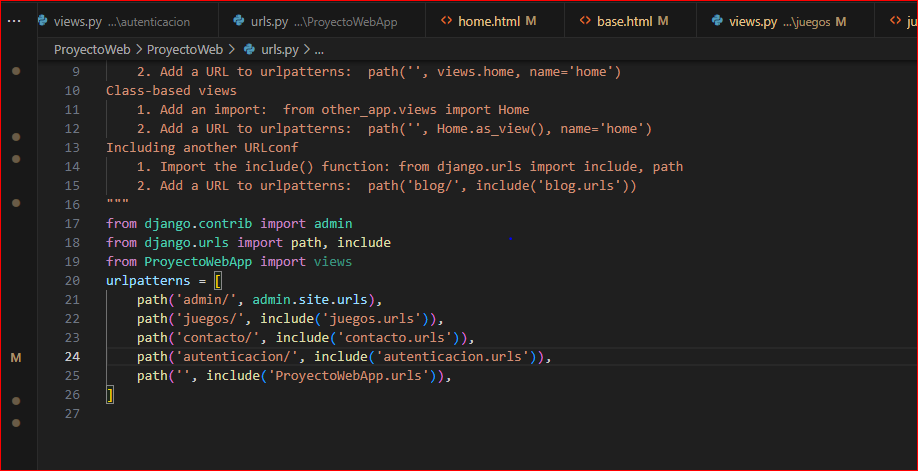


**Rutas**

En el archivo urls.py de la aplicación juegos, definimos las rutas de la API:



En el archivo urls.py del proyecto principal (juegos\_api), incluimos las rutas de la aplicación:



## Desarrollo de frontend

El desarrollo del frontend de nuestra aplicación de gestión de juegos se ha realizado utilizando HTML, CSS y Django Templates. La integración de Django Templates nos permite generar páginas web dinámicas y reutilizar componentes comunes en diferentes vistas de la aplicación, lo que mejora la mantenibilidad y la coherencia del diseño.

**Estructura del Proyecto**

La estructura del proyecto frontend en Django se organiza de la siguiente manera:

**Plantillas (Templates)**

**base.html**

En la plantilla base (base.html) definimos la estructura común para todas las páginas de la aplicación. Incluye el encabezado, pie de página, y enlaces a los archivos CSS y JavaScript.

**juego\_lista.html**

Esta plantilla se muestra la lista de juegos. Hereda de **base.html** y define un bloque de contenido específico para la lista de juegos.

Mencionar los html

**Archivos estáticos**

**styles.css**

Archivo CSS que define los estilos para la aplicación.

**scripts.js**

Archivo JavaScript para la aplicación (si es necesario).

El desarrollo del frontend utilizando HTML y Django Templates permite una integración eficiente y coherente con el backend de Django, facilitando la generación de páginas dinámicas y el manejo de datos. Esta estructura modular y reutilizable mejora la mantenibilidad y escalabilidad de la aplicación. El uso del archivo base base.html garantiza una estructura uniforme en todas las páginas, simplificando la gestión del diseño y estilos.

# Pruebas

Para la realización de la pruebas con el software

## Pruebas unitarias

## Pruebas de sistema

# Mantenimiento

# Resultados y conclusiones

Resumen de resultados obtenidos en el TFG. Y conclusiones personales del estudiante sobre el trabajo realizado.

# Análisis de Impacto

En este capítulo se realizará un análisis del impacto potencial de los resultados obtenidos durante la realización del TFG en diferentes contextos:

* personal
* empresarial
* social
* económico
* medioambiental
* cultural

En dicho análisis se destacarán los beneficios esperados, así como también los posibles efectos perjudiciales.

Se recomienda analizar también el potencial impacto haciendo referencia a los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 que sean relevantes para la solución propuesta (https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/).

En este capítulo se destacarán también aquellas decisiones tomadas a lo largo del trabajo que tienen como base la consideración del impacto.

# Bibliografía

Publicaciones utilizadas en el estudio y desarrollo del trabajo.

Hay que utilizar un sistema internacional para referencias bibliográficas, de acuerdo con las indicaciones del tutor. Por ejemplo, el [sistema de IEEE](https://www.etsiinf.upm.es/docs/estudios/grado/1475_ieeecitationref.pdf).

Salvador Sanchez, Miguel Angel Silicia y Daniel Rodriguez (2011). Ingeniería de Software (libro) Un enfoque desde la guía SWEBOK.

Shari Lawrence Pfleeger (2002). Ingenieria de software (libro)

Ian Sommerville (2005). Ingenieria de Software (libro).

Estándar IEEE 1990. Glosario IEEE de Tecnica de Ingenieria del Software.

Estándar IEEE 830.1998. Plantilla de Documento de especificación de requisitos.

Estándar IEEE de Practicas Recomendadas para la descripción de Arquitectura Software (ISO/IEC 42010:2007.

# Anexos

Este capítulo es opcional, y se escribirá de acuerdo con las indicaciones del Tutor.