Sistemas de Información Práctica 4

GameHeaven

23 de noviembre de 2023

Daniel Jal Burguete, 815450

Álvaro López Berga, 820800

Mario Ortega Cubero, 817586

Índice

Introducción

En el contexto de la Práctica 4, insertados en el ámbito de la asignatura de Sistemas de Información de Ingeniería Informática en la Universidad de Zaragoza, nos adentramos en la fase crucial de implementación de nuestro sistema de gestión de información para la tienda de videojuegos "GameHeaven".

Este documento tiene como propósito ofrecer una visión detallada de todo el proceso relacionado con la creación y desarrollo de la interfaz web y las reglas de negocio de nuestra aplicación. La conjunción de estos elementos es esencial para proporcionar a nuestros usuarios una experiencia fluida y eficiente al explorar e interactuar con los productos y servicios ofrecidos por GameHeaven.

La memoria generada durante esta etapa no solo sirve como un registro detallado de las decisiones tomadas y del código implementado, sino que también actúa como un recurso fundamental para comprender la arquitectura de nuestro sistema. Al suministrar una visión completa de la estructura y gestión de la información clave para nuestra plataforma, esta documentación se convierte en un valioso activo para futuros desarrollos y mantenimientos.

Adicionalmente, para facilitar el seguimiento y colaboración, hemos optado por emplear Git como nuestro sistema de gestión de versiones. Repositorio: gameheaven. A través de esta plataforma, proporcionamos una ventana transparente a nuestro código fuente y sus evoluciones, fomentando la colaboración y revisión por parte de compañeros y docentes.

En resumen, esta fase de implementación marca un hito crucial en el camino de GameHeaven hacia su plena funcionalidad. Con este informe, no sólo documentamos nuestro progreso sino que también sentamos las bases para futuras iteraciones y mejoras en nuestra aplicación. ¡Bienvenidos a GameHeaven, donde la experiencia de juego comienza en cada clic!

Metodología

Diseño

Nuestra aplicación web se estructura en tres capas fundamentales para garantizar un funcionamiento eficiente y una experiencia del usuario fluida. Estas capas son:

Capa de Base de Datos:

- Diseñada en prácticas anteriores, esta capa constituye la infraestructura donde almacenamos y gestionamos nuestra información crítica. La base de datos, junto con los Objetos de Acceso a Datos (DAOs), se encarga de interactuar con los datos almacenados, proporcionando una interfaz consistente para su manipulación.

Capa de Servidor de Aplicación Web:

- En el corazón de nuestra arquitectura, esta capa es responsable tanto de la lógica de negocio como de la interfaz web. Aquí, la lógica de negocio se ejecuta, asegurando que nuestras operaciones comerciales sean sólidas y eficientes. Además, la interfaz web toma forma, utilizando plantillas base para generar páginas HTML dinámicas. La capa de servidor no solo procesa la lógica interna, sino que también renderiza los datos de la base de datos en HTML, proporcionando así páginas estáticas al cliente.

Capa del Cliente:

- Esta es la capa a la que interactúan directamente los usuarios. Utilizando un navegador, los clientes realizan solicitudes HTTP a nuestra aplicación web. La capa del cliente recibe y presenta la información proporcionada por la capa de servidor, representando visualmente los HTMLs estáticos generados. Así, los

usuarios pueden acceder a la diversa información que ofrece nuestra aplicación de manera intuitiva y eficiente.

Con esta estructura de tres capas, hemos logrado una separación clara de responsabilidades, permitiendo una fácil escalabilidad y mantenimiento. Desde la gestión de datos hasta la presentación en el navegador, cada capa desempeña un papel vital en la creación de una experiencia coherente y eficaz para nuestros usuarios.

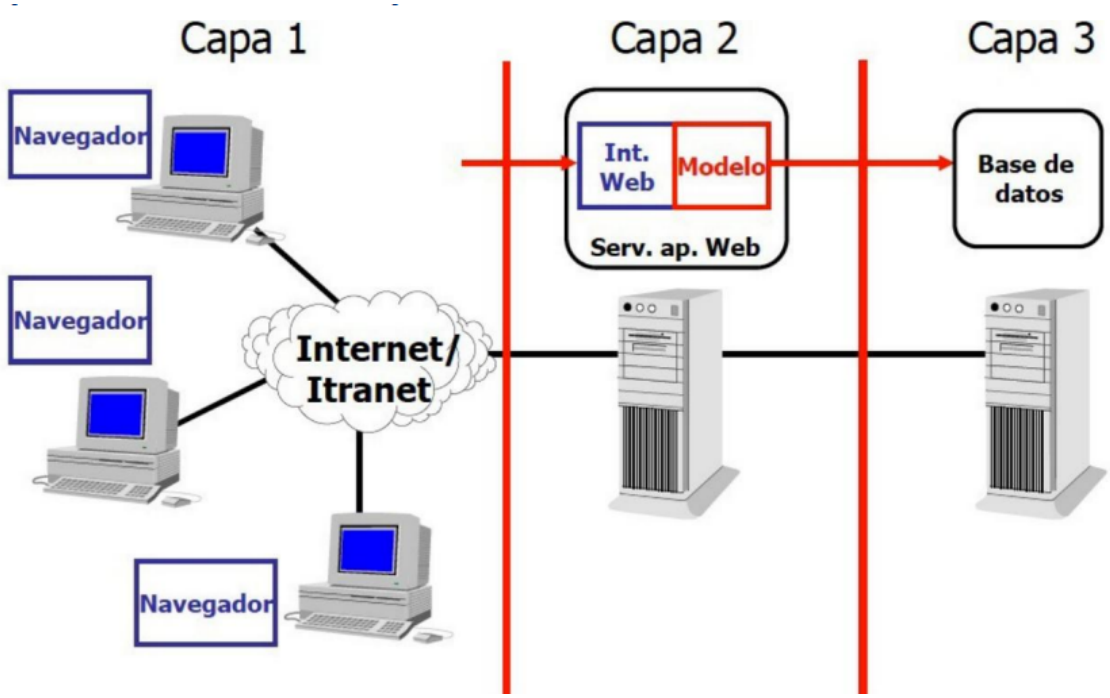

Figura 1: Arquitectura web de 3 capas

Decisiones

Toma de Decisiones en la Implementación de la Lógica de Negocio:

En el diseño y desarrollo de la lógica de negocio de nuestra aplicación, se tomaron decisiones cruciales para garantizar un flujo eficiente y seguro. La gestión de permisos se abordó mediante el uso de los grupos y permisos proporcionados por Django. Estos mecanismos resultaron ser herramientas fundamentales para restringir acciones específicas a determinados tipos de usuarios, permitiéndonos controlar de manera precisa el acceso a nuestra API.

Además, se adoptó la estrategia inicial de agregar videojuegos a través de un formulario tradicional. No obstante, en una fase posterior, se introdujo una funcionalidad adicional que permite la incorporación de videojuegos desde la plataforma Steam. Esta nueva característica se implementa mediante el uso de un módulo de Python que encapsula la API de Steam. Esta decisión nos habilita no solo para añadir juegos mediante un formulario convencional sino también para realizar búsquedas en la API de Steam y obtener información directamente.

Decisiones en la Implementación de la Interfaz Web:

En lo que respecta a la interfaz web, las decisiones se centraron en la estructuración de los HTML y la implementación de formularios. Django ofrece un lenguaje propio que facilita la inserción de código y la creación de plantillas base, las cuales se incorporan en otros HTML para una mayor modularidad y mantenibilidad del código.

Para la creación de formularios, se utilizaron tanto las funcionalidades nativas de Django, que permiten la creación de formularios de manera sencilla mediante clases con atributos, como el módulo de Python denominado "crispy-forms". Este último, al recibir un formulario de Django como contexto al renderizar un HTML, genera automáticamente el código HTML necesario, simplificando así el proceso de diseño y optimizando la presentación de los formularios.

Decisiones en el Estilo de la Interfaz:

La estilización de nuestra interfaz se abordó utilizando Bootstrap 5, un marco de diseño ampliamente reconocido por su capacidad de respuesta y flexibilidad. Se optó por un estilo específico, el cual se encuentra detallado en la Bibliografía de nuestro proyecto, proporcionando coherencia estética y mejorando la experiencia visual para los usuarios finales.

Estas decisiones han sido fundamentales en la construcción de una aplicación web sólida, eficiente y visualmente atractiva. Cada elección fue cuidadosamente evaluada para asegurar la coherencia y el rendimiento de nuestra plataforma.

Recursos y herramientas

La elección de Django para la implementación de la base de datos fue una decisión estratégica que demostró ser altamente beneficiosa durante la asignatura de Proyecto Software ya realizada por dos de los integrantes del equipo el año anterior. La adopción de este marco de alto nivel se fundamentó en su capacidad para proporcionar utilidades prediseñadas, especialmente en la generación automática de tablas en PostgreSQL a través de migraciones. Los modelos de Django, que se corresponden con nuestras clases (Value Objects), fueron instrumentalizados para facilitar la creación de las tablas necesarias.

La ventaja adicional de Django reside en su API interna que permite el acceso a los datos sin la necesidad de realizar consultas SQL directas. Este aspecto simplificó significativamente la implementación de los métodos de acceso a la base de datos, en particular la creación de Data Access Objects (DAOs) de nivel superior, que se pueden examinar en la carpeta DAOs del proyecto entregado.

En cuanto a la capa de presentación, se empleó Bootstrap 5 para la personalización visual, Crispy para la gestión de formularios, y se integraron elementos de Javascript para animaciones básicas. Además, se aprovechó la potencia de Django en la creación de formularios, facilitando la gestión de las solicitudes POST mediante la presentación de formularios renderizados directamente en los templates.

Para mejorar la información relacionada con los videojuegos disponibles en Steam, se implementó la SteamAPI. Esta interfaz permite obtener datos cruciales como imágenes, precios, descripciones y valoraciones, enriqueciendo así la calidad de la información presentada en la aplicación.

Distribución de trabajo

Desde el inicio del proyecto, se ha adoptado la práctica de colaborar estrechamente, incluso cuando cada miembro del equipo se encontraba trabajando en secciones distintas de la aplicación. Este enfoque, que ha demostrado ser eficaz en prácticas anteriores, se ha mantenido durante todo el desarrollo del presente proyecto, produciendo, en nuestra evaluación, resultados satisfactorios.

En la fase final del proyecto, dedicamos la última semana a realizar pruebas exhaustivas del sistema. Durante este período, nos enfocamos de manera conjunta en identificar y abordar los diversos errores detectados, asegurándonos así de la estabilidad y coherencia del sistema implementado. Este enfoque colaborativo permitió una resolución efectiva de los problemas surgidos y contribuyó al logro de un producto final más robusto y fiable.

Principales dificultades encontradas

En el transcurso de nuestro primer encuentro con Django y PostgreSQL, se requirió consultar la documentación para adquirir un entendimiento profundo de los fundamentos y así aprovechar al máximo las capacidades de integración de Django con un sistema de gestión de bases de datos robusto como PostgreSQL.

En una etapa posterior del desarrollo, al llevar a cabo pruebas, nos enfrentamos a desafíos, identificando errores de complejidad inicial. No obstante, mediante el estudio de la documentación correspondiente, logramos comprender más a fondo el funcionamiento interno de Django, permitiéndonos abordar y resolver eficazmente dichos problemas.

Durante la fase avanzada del proyecto, específicamente en la implementación de la capa del servidor de aplicación, no se presentaron desafíos sustanciales, a excepción de las consideraciones relativas al formato y paso de datos.

En lo que respecta a la integración con la API de Steam, surgieron diversas problemáticas. Por ejemplo, al tratar con videojuegos gratuitos, observamos que el campo de precio devuelto por las consultas podía presentar varias variaciones de texto(“Free to Play”, “Free To Play”, “0”, “free to play”...). Para abordar esta variabilidad, se implementó un mecanismo que envuelve la recuperación del precio en un bloque "try-except", intentando convertir el valor devuelto por la API a un formato de número flotante. En caso de error, se asigna el valor cero al precio.

Inicialmente, se contempló la utilización de la API de una plataforma específica para publicar automáticamente tweets cada vez que un trabajador añadiera un nuevo producto a la tienda. Sin embargo, esta iniciativa fue abandonada debido a las limitaciones significativas de la API, que requería un pago para habilitar dicha funcionalidad. A pesar de ello, se implementó una sección en la interfaz web, denominada "Acerca de", donde se pueden visualizar los tweets que los trabajadores han publicado manualmente.

En el ámbito del desarrollo del frontend, surgieron desafíos relacionados con la manipulación de formularios, los cuales fueron resueltos mediante la adopción de formularios de Crispy. Esta herramienta facilitó la renderización de formularios de Python en HTML de manera eficiente. Asimismo, para abordar la necesidad de insertar código en múltiples instancias, recurrimos al uso de Custom Tags and Filters de Django, lo que nos permitió integrar código directamente en el HTML.

En la capa de personalización, además de Crispy, incorporamos Bootstrap 5 para mejorar la estética y la presentación, dado que comenzar desde cero con CSS se percibía como un proceso tedioso.

Plan de pruebas llevadas a cabo

Para probar nuestros DAOs y VOs utilizamos los tests de django para comprobar algunos casos ya que no era realmente necesario testear todos los DAOs para comprobar la consistencia de la base

Para probar nuestra web, decidimos que la mejor opción sería ir haciendo casos de uso que pudiéramos creer útiles como por ejemplo añadir juegos de Steam que no hayan salido todavía.

Aparte de estos que tratan de buscar vulnerabilidades en casos poco comunes también hemos comprobado todos los casos de uso posibles tanto de clientes como de trabajadores.

Se hicieron numerosas pruebas con la API de Steam y se detectaron varios problemas (detallados en la sección anterior).

Plan de mantenimiento

El plan de mantenimiento para nuestra web representa un compromiso con la excelencia operativa y experiencia continua del usuario. En el mundo en el que vivimos de constante y rápida evolución de las tecnologías es necesario garantizar estabilidad y seguridad de nuestro sistema de información. Esto es esencial para mantener la confianza de nuestros usuarios.

A través del plan que se va a detallar a continuación, se pretende ofrecer un sistema robusto, seguro y en continuo crecimiento para garantizar el liderazgo de Gameheaven en el mercado de los videojuegos.

Gestión de errores y excepciones

Debido al limitado tiempo del que hemos dispuesto para desarrollar el sistema, algunos de los errores que pueden encontrarse (los menos comunes) se han dejado con la respuesta por defecto. A futuro, se planea integrar en una próxima actualización una gestión de errores más robusta en la que el usuario sea provisto con información relevante.

Actualizaciones de seguridad y parches

Una vez realizado el despliegue, se aplicarán tanto a nuestro sistema como a la máquina en la que se ha desplegado los parches de seguridad y actualizaciones requeridos semanalmente. Así mismo si alguna de las aplicaciones requiriese algún reinicio este se realizaría durante las horas de menos tráfico de peticiones para minimizar el daño a los clientes.

Documentación

De momento se dispone de acceso al repositorio público con todo el código fuente bien estructurado de forma que se facilita su comprensión, modificación e integración de nuevas funcionalidades. A futuro se planea documentar todo el código y utilizar herramientas como Sphinx para generar la documentación automática.

Rendimiento

De momento el sistema se va a desplegar sobre una única máquina, lo que puede derivar eventualmente en fallos por caídas, lo que empeora la experiencia de usuario notablemente.

Para solucionarlo, se plantea desarrollar un sistema distribuido usando kubernetes en 3 máquinas distintas. De este modo, se podrían tener 3 réplicas de nuestro sistema y tolerar fallos de conexión o caídas de nuestro sistema en alguna de las réplicas y garantizar el acceso al sistema como cliente.

Además, permitiría actualizar por partes, dejando siempre una de las réplicas en la versión anterior hasta que se verifica con pruebas exhaustivas que la nueva funciona. Así si se integra alguna funcionalidad nueva y falla, no llegaría a ser detectado por el cliente.

Copias de seguridad

Además de la tolerancia a fallos que se busca lograr con el uso de kubernetes en el futuro (ya garantiza cierta robustez en cuanto a los datos), se realizan copias de seguridad diarias en nuestros servidores centrales.

Capacitación del personal

Ya que nos encontramos ante un sistema que va a ser usado por los trabajadores de las distintas sucursales para gestionar reservas de videojuegos y stock, hay que asegurar que dichos trabajadores lo utilicen de mejor manera.

Para ello, conforme se vaya integrando el uso de la web en las distintas sucursales, uno de nuestros técnicos realizará un cursillo rápido para los trabajadores. Así mismo, se les proporcionará un pequeño manual de usuario.

Al ser una empresa relativamente nueva no se dispone de personal para poder dar soporte las 24h del día, pero se garantiza un tiempo de respuesta inferior a 5 horas para problemas menores y de 1 hora para defectos graves (mal funcionamiento completo del sistema, pérdidas de datos importantes…)

Bibliografía

Documentación Django: https://docs.djangoproject.com/en/4.2/topics/db/models/

Proyecto en github: https://github.com/mariooc04/gameheaven

Steam API: https://pypi.org/project/python-steam-api/

Bootstrap: https://bootswatch.com/vapor/

Horas

A continuación las horas totales invertidas en el proyecto. Las filas azules son las correspondientes al desarrollo de esta práctica.

	Horas Mario	Fecha	Tiempo	Horas Daniel	Fecha	Tiempo	Horas Álvaro	Fecha	Tiempo	
		22/09/2023	2		22/09/2023	2		22/09/2023	2	
		24/09/2023	1		24/09/2023	1		24/09/2023	1	
		29/09/2023	2		29/09/2023	2		29/09/2023	2	
		01/10/2023	2		30/09/2023	4		01/10/2023	2	
		6/10/2023	2		01/10/2023	2		6/10/2023	2	
		7/10/2023	2		6/10/2023	2		7/10/2023	2	
		9/10/2023	2		9/10/2023	2		9/10/2023	2	
		13/10/2023	2,5		13/10/2023	2,5		13/10/2023	2,5	
		14/10/2023	2,5		14/10/2023	2,5		14/10/2023	2,5	
		21/10/2023	2		21/10/2023	2		21/10/2023	2	
		22/10/2023	5		22/10/2023	5		22/10/2023	4	
		25/10/2023	2		25/10/2023	2		25/10/2023	2	

		27/10/2023	2		27/10/2023	2		27/10/2023	2	
		29/10/2023	2		29/10/2023	2		29/10/2023	2	
		30/10/2023	2					30/10/2023	2,5	
		01/11/2023	2,5		01/11/2023	2,5		01/11/2023	2,5	
		03/11/2023	2		03/11/2023	2		03/11/2023	2,5	
		04/11/2023	3		04/11/2023	3		04/11/2023	3	
		07/11/2023	3		07/11/2023	3		07/11/2023	3	
		09/11/2023	2		09/11/2023	2		09/11/2023	2	
		15/11/2023	2		15/11/2023	2		15/11/2023	2	
		17/11/2023	2		17/11/2023	2		17/11/2023	2	
		21/11/2023	2		21/11/2023	2		21/11/2023	2	
		22/11/2023	2		22/11/2023	2		22/11/2023	2	
		23/11/2023	2		23/11/2023	2		23/11/2023	2	
Total			55,5			55,5			55,5	166,5