ESGames: Desarrollo de una Plataforma Web para la Venta de Videojuegos Utilizando Django y Vue.js

*Note: Sub-titles are not captured in Xplore and should not be used

1st Flores Nuñes, Rodrigo Francisco
Universidad Nacional de San Agustín
Arequipa, Perú
email address or ORCID

2nd Jimenez Paredes, Fabricio Gabriel
Universidad Nacional de San Agustín
Arequipa, Perú
fjimenezp@unsa.edu.pe

3rd Leon Ramos, Mijael Paul
Universidad Nacional de San Agustín
Arequipa, Perú
fjimenezp@unsa.edu.pe

4th Paredes Miranda, Mauricio Antonio
Universidad Nacional de San Agustín
Arequipa, Perú
fjimenezp@unsa.edu.pe

***Abstract*—This paper presents the design and development of GameStore, a web-based platform intended to simulate the operation of a digital video game store. The system is structured using a client-server architecture, with Django employed for backend development and Vue.js for frontend implementation. The application allows users to register, log in, browse a catalog of video games, add products to a shopping cart, and manage a personal library of purchased titles. Additional features include user balance management, game reviews, ratings, and favorites, aiming to improve user interaction and engagement. The system implements a RESTful API to ensure efficient communication between client and server, facilitating modularity and scalability.**

***Index Terms*—Video game store, web development, Django, Vue.js, REST API.**

I. Introduccion

El crecimiento sostenido del comercio electrónico ha transformado la manera en que los usuarios acceden y consumen productos digitales, incluyendo los videojuegos. En este contexto, las plataformas en línea han adquirido una relevancia fundamental al permitir la distribución eficiente de títulos, la gestión de bibliotecas personales y la interacción entre usuarios mediante reseñas y valoraciones.

El presente trabajo describe el desarrollo de GameStore, una aplicación web que simula una tienda digital de videojuegos. La plataforma permite a los usuarios registrarse, iniciar sesión, explorar un catálogo de juegos, añadir productos a un carrito de compras y adquirirlos utilizando un sistema de saldo virtual. Asimismo, el sistema ofrece funcionalidades complementarias como la biblioteca de juegos adquiridos, el sistema de favoritos, y módulos para realizar valoraciones y comentarios.

Para la implementación se empleó una arquitectura cliente-servidor, donde el backend fue desarrollado con Django, un framework robusto basado en el patrón modelo-vista-controlador (MVC), ampliamente utilizado en aplicaciones web modernas por su seguridad y escalabilidad [1]. El frontend se desarrolló utilizando Vue.js, un framework progresivo de JavaScript orientado a componentes, que permite construir interfaces reactivas bajo el paradigma de aplicaciones de una sola página (SPA, por sus siglas en inglés) [2].

El desarrollo de GameStore busca demostrar la viabilidad de construir plataformas especializadas de comercio digital utilizando tecnologías web modernas, priorizando aspectos como modularidad, rendimiento y experiencia de usuario.

Identify applicable funding agency here. If none, delete this.

II. Marco Teorico

El desarrollo de aplicaciones web modernas requiere el uso de herramientas y marcos que permitan separar responsabilidades, garantizar la escalabilidad del sistema y mejorar la experiencia del usuario final. En este proyecto se utilizaron tecnologías ampliamente adoptadas en la industria como Django, Vue.js, y la arquitectura basada en API REST. A continuación, se detallan los fundamentos teóricos y tecnológicos empleados.

A. Django

Django es un framework de desarrollo web de alto nivel escrito en Python que sigue el patrón modelo-vista-controlador (MVC), aunque su implementación se describe como modelo-vista-template (MVT). Django permite una gestión eficiente de modelos de datos mediante un ORM (Object-Relational Mapper), además de contar con herramientas integradas para la autenticación, administración y enrutamiento [1]. Su robustez y modularidad lo hacen adecuado para aplicaciones escalables y seguras, siendo ampliamente utilizado en sistemas de comercio electrónico, gestión de contenidos y redes sociales.

B. Django REST Framework

Django REST Framework (DRF) es una extensión poderosa para la creación de APIs RESTful dentro del ecosistema Django. Proporciona herramientas para la serialización de

datos, control de permisos, autenticación, paginación y filtrado, facilitando la interoperabilidad entre el servidor y el cliente [2]. La arquitectura REST permite que distintos componentes del sistema (como aplicaciones móviles o frontends independientes) se comuniquen de manera estándar mediante peticiones HTTP.

C. Vue

Vue es un framework progresivo para la construcción de interfaces de usuario interactivas. Su enfoque basado en componentes reutilizables, junto con su sistema de enlace bidireccional (two-way binding), lo convierten en una solución ideal para el desarrollo de aplicaciones de una sola página (SPA) [3]. Además, Vue.js permite una integración progresiva, lo cual significa que puede ser utilizado tanto para partes específicas de una aplicación como para construir interfaces completas.

D. Arquitectura Cliente-Servidor y SPA

La arquitectura cliente-servidor divide la lógica del sistema entre dos capas principales: el servidor (backend), que gestiona los datos y la lógica empresarial, y el cliente (frontend), que presenta la información al usuario y gestiona las interacciones. Las aplicaciones SPA ofrecen una experiencia fluida al cargar una sola página HTML y actualizar dinámicamente su contenido mediante JavaScript, lo que reduce la necesidad de recargar completamente el sitio web [4]. Este enfoque mejora la velocidad de navegación y la interactividad del sistema.

III. Desarollo e Implementacion

A. Backend

El backend de GameStore fue desarrollado utilizando el framework Django, y se encarga de gestionar la lógica de negocio, la autenticación de usuarios y la comunicación con la base de datos alojada en Supabase. A pesar de que Django proporciona herramientas para definir modelos y administrar bases de datos localmente, en este proyecto se optó por externalizar la gestión de datos, accediendo a ellos mediante llamadas HTTP a la API de Supabase.

El sistema se organiza en una aplicación modular denominada ESGames, que agrupa las vistas (views.py), rutas (urls.py) y serializadores (serializers.py) necesarios para el manejo de las operaciones de negocio. Las vistas están implementadas como clases basadas en Django REST Framework [3], lo que permite estructurar de forma clara los distintos endpoints y aplicar control de acceso según el tipo de usuario.

Las funcionalidades implementadas en el backend incluyen:

- Registro e inicio de sesión de usuarios, integrando autenticación personalizada.
- Gestión del flujo de compra, que valida el saldo del usuario, registra la transacción y actualiza la biblioteca en Supabase.
- Manejo del carrito de compras y favoritos, controlando qué elementos están asociados a cada usuario.
- Generación de comprobantes de compra, mediante plantillas HTML dinámicas renderizadas desde Django.
- Enrutamiento y control de acceso, con protección de rutas mediante tokens de autenticación.
- Conexión con Supabase, mediante peticiones programadas que interactúan con su API REST, en lugar de usar el ORM de Django.

El backend actúa como intermediario entre la interfaz de usuario y la base de datos, implementando las reglas del negocio que rigen el comportamiento de la aplicación. Esta separación permite mantener un control centralizado sobre la lógica crítica del sistema, a la vez que se delega la persistencia de datos a un servicio externo.

B. Frontend

El frontend se desarrolló utilizando Vue.js, organizando la aplicación en una estructura de componente raíz (App.vue) y múltiples vistas contenidas en el directorio src/views. El punto de entrada (main.js) monta la aplicación y configura el enrutador.

Las vistas principales son:

- Home.vue: página de inicio que lista los videojuegos disponibles.
- Login.vue / Register.vue: formularios para iniciar sesión y registrarse..
- Profile.vue: vista del perfil del usuario, incluyendo su saldo, juegos comprados y favoritos.

El sistema de rutas se gestiona mediante Vue Router, y el estado de sesión del usuario se mantiene a través de un archivo auth.js que actúa como almacén global.

La comunicación con la API se realiza mediante Axios, centralizada en módulos como api.js y videogames.js, lo que permite desacoplar la lógica de negocio del consumo de datos. El estilo general se define en style.css y se aplica mediante clases y estilos scoped en los componentes.

C. Base de Datos y Entidades

La base de datos utilizada en GameStore está alojada en Supabase, una plataforma de desarrollo backend como servicio que proporciona una base de datos PostgreSQL completamente gestionada en la nube. Supabase permite definir tablas, relaciones y políticas de acceso mediante una interfaz gráfica, además de generar automáticamente una API REST para cada entidad del sistema [5].

Aunque el backend del sistema fue desarrollado con Django, la lógica de persistencia no se maneja mediante el ORM tradicional, sino que se conecta directamente con Supabase a través de la API proporcionada por esta plataforma. Esto permite separar la gestión de datos del entorno del servidor, facilitando la escalabilidad y la implementación en producción.

Las entidades principales definidas en Supabase incluyen:

- UserProfile: almacena información del usuario y su saldo virtual.
- VideoGame: contiene los datos de los videojuegos disponibles.

- CartItem, Favorite, Review, Library: definen las relaciones entre los usuarios y los juegos.

La estructura de datos se rige por un modelo relacional, donde:

- Un UserProfile puede tener múltiples CartItems, Reviews, y Favorites.
- Un VideoGame puede estar vinculado a muchas Reviews, Favorites y elementos de la biblioteca (Library).
- Library representa una relación muchos a muchos (N:M) entre usuarios y videojuegos.

Las operaciones de lectura y escritura se realizan a través de peticiones HTTP a la API REST de Supabase, o bien mediante el cliente oficial en JavaScript. Esta aproximación permite una integración directa desde el frontend o desde servicios externos sin necesidad de exponer el backend de Django para estas tareas específicas.

IV. Resultados

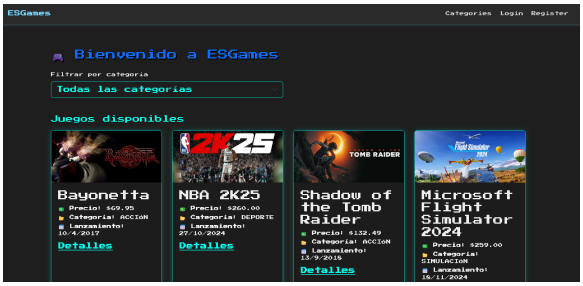

Fig. 1. Example of a figure caption.

Fig. 2. Example of a figure caption.

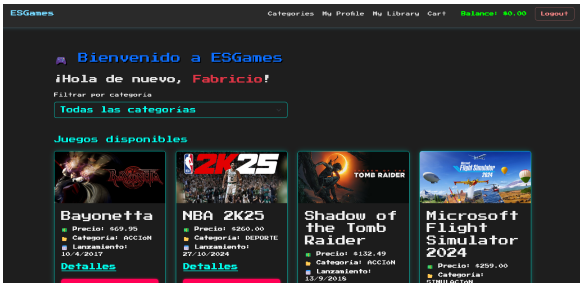

Fig. 3. Example of a figure caption.

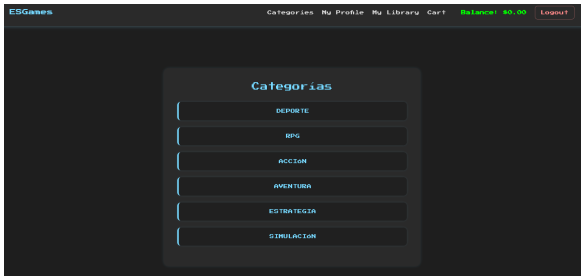

Fig. 4. Example of a figure caption.

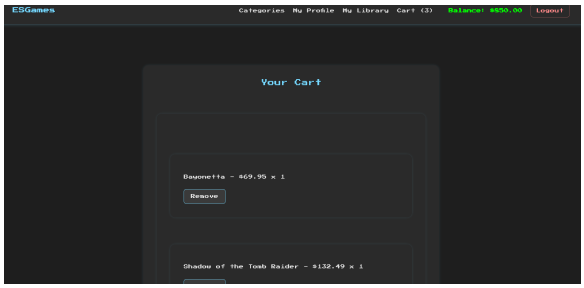

Fig. 5. Example of a figure caption.

References

[1] Django Software Foundation, "Django," https://www.djangoproject.com/, Accedido: 24-jul-2025.

[2] Vue.js Developers, "Vue.js - The Progressive JavaScript Framework," https://vuejs.org/, Accedido: 24-jul-2025.

[3] Django REST Framework, "DRF - Django REST framework," https://www.django-rest-framework.org/, Accedido: 24-jul-2025.

[4] M. Stojanovic, "What is a Single Page Application (SPA)?," Telerik, 2023. [En línea]. Disponible: https://www.telerik.com/blogs/what-is-a-single-page-application

[5] Supabase, "The Open Source Firebase Alternative," https://supabase.com/, Accedido: 24-jul-2025.

[6] .

[7] .

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.