



Tradewing

Pablo de Baro Escapa
Marco Fernández Llamas
Javier García Turiel

Introducción

Tradewing es una aplicación web diseñada para facilitar la compraventa de productos de segunda mano entre particulares a través de Internet. Permite a los usuarios publicar artículos usados, que pueden ser vistos y comprados por cualquier persona con acceso a internet.

La aplicación está dirigida a personas de todas las edades con conocimientos básicos de navegación por internet, interesadas en vender o adquirir productos usados.

El objetivo principal de Tradewing es crear una plataforma web funcional y accesible que facilite el intercambio de artículos usados entre usuarios particulares, promoviendo la economía circular y ofreciendo una alternativa práctica, rápida y económica al consumo tradicional.

Equipo de Trabajo

Miembros del equipo:

- Pablo De Baro Escapa - Scrum Master
- Marco Fernández Llamas - Desarrollador software
- Javier García Turiel - Desarrollador software

Para la organización de tareas, hemos creado un Project de Github, donde en una sesión inicial, establecimos todas las tareas necesarias para completar nuestro producto software. En cada sesión de prácticas semanal, decidíamos que tareas completar esa semana y a lo largo de esa semana, cada uno iba haciendo las tareas de esa pila de tareas semanales.

Requisitos y organización del proyecto

- RF1. El usuario puede registrarse, iniciar sesión y cerrar sesión.
- RF2. El usuario puede crear y eliminar anuncios de productos.
- RF3. Los anuncios pueden incluir imágenes, título, descripción, y precio.
- RF4. Los usuarios pueden buscar productos por texto.
- RF5. Es posible visualizar los detalles de un producto publicado.
- RF6. El usuario puede comprar productos a través de la plataforma.
- RF7. El comprador puede consultar el estado de sus pedidos y realizar su seguimiento.
- RF8. El usuario puede gestionar su perfil y ver sus anuncios activos.
- RF9. Se valida la autenticación y autorización para acceder a funciones protegidas.

Diseño y Arquitectura

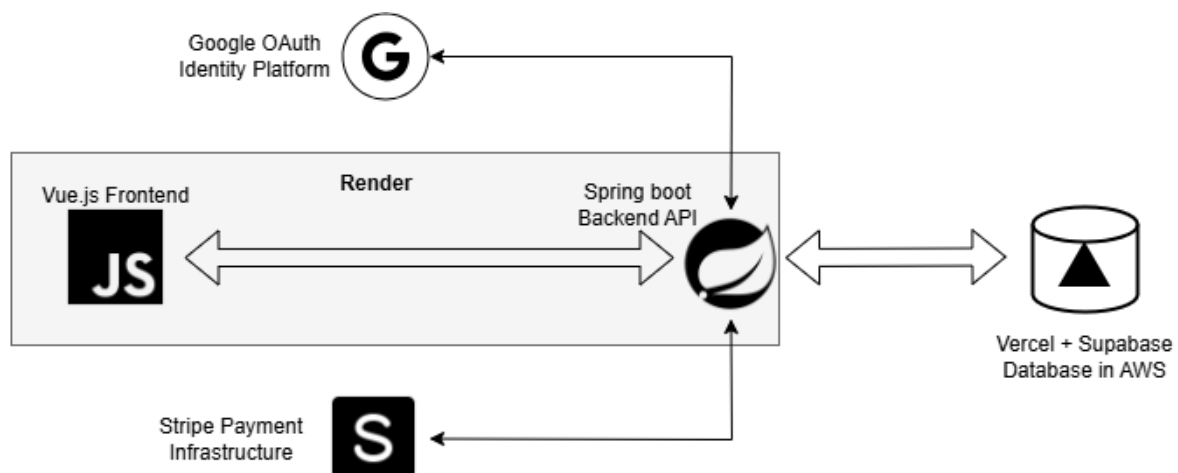
Tradewing sigue una arquitectura de tipo cliente-servidor dividida en tres capas principales:

- **Cliente (frontend):** Una aplicación web desarrollada con Vue.js que se ejecuta en el navegador del usuario. Se encarga de la interacción directa con el usuario, permitiendo visualizar y comprar productos, publicar anuncios, y gestionar el perfil personal.

- **Servidor (backend):** Una API REST desarrollada con Spring Boot que actúa como intermediario entre el cliente y la base de datos. Se encarga de la lógica de negocio, gestión de usuarios, seguridad, validación de datos y comunicación con la base de datos.
- **Base de datos:** Un sistema PostgreSQL alojado en AWS, donde se almacena de forma persistente toda la información de usuarios, productos, mensajes, categorías, imágenes, etc.

Tecnologías utilizadas

- **Vue.js:** Elegido como framework frontend por su curva de aprendizaje baja, su enfoque reactivo, su capacidad de crear interfaces de usuario dinámicas y su facilidad de integración con servicios REST.
- **Spring Boot:** Utilizado para construir el backend gracias a su robustez, modularidad, soporte para REST, y su integración con herramientas de seguridad, persistencia y gestión de dependencias.
- **PostgreSQL:** Seleccionado como sistema de gestión de bases de datos por su fiabilidad, soporte para integridad referencial, funciones avanzadas y buen rendimiento en entornos transaccionales.
- **AWS (Amazon Web Services):** Se ha utilizado para alojar la base de datos debido a su alta disponibilidad, escalabilidad y facilidad de gestión en entornos en la nube, a través del marketplace de Vercel + Supabase.
- **Render:** Plataforma de despliegue automático utilizada para alojar tanto el frontend como el backend. Permite una integración continua sencilla, despliegues automáticos desde el repositorio de código y una gestión simplificada de servicios web.



Desarrollo e Implementación

Buenas prácticas aplicadas:

- **Uso de Pull Requests:** Todo el código nuevo o modificado fue incorporado al repositorio principal a través de pull requests. Esto permitió revisar los cambios antes de integrarlos, asegurando que se cumplieran los estándares de calidad establecidos y que no se introdujeran errores.
- **Revisiones de código (code review):** Cada pull request fue revisado por al menos otro miembro del equipo antes de ser aceptado.
- **Control de versiones con Git:** Se utilizó Git como sistema de control de versiones, siguiendo una estrategia de ramas para mantener el entorno principal (main) siempre estable y operativo.
- **Uso de patrones de diseño:**
 - En el backend, se aplicó el patrón MVC (Modelo-Vista-Controlador) para estructurar la lógica de la aplicación, separando responsabilidades entre la gestión de datos, la lógica de negocio y la presentación.
 - También se emplearon otros patrones como Repository para el acceso a datos y Service para encapsular la lógica de negocio, promoviendo la reutilización de código y la independencia entre capas.
 - En el frontend, se organizaron los componentes de Vue siguiendo una estructura modular, facilitando la reutilización y el mantenimiento.
- **Despliegue continuo con Render:** Se configuró la integración continua para que cada cambio en la rama principal del repositorio desencadene automáticamente un despliegue en la plataforma Render, permitiendo así una validación rápida de nuevas funcionalidades.
- **Separación de entornos:** Se gestionaron diferentes entornos (desarrollo, pruebas y producción) para poder realizar pruebas sin afectar a la versión estable del sistema.

Ejemplo de una funcionalidad (Proceso de compra de un producto)

Objetivo de la funcionalidad:

El objetivo de esta funcionalidad es permitir al usuario realizar el pago de un producto publicado en la plataforma Tradewing a través del proveedor Stripe, y posteriormente recibir un correo electrónico con los detalles del producto adquirido como confirmación de la compra.

Flujo de pago

El usuario accede a la página de detalle de un producto y pulsa el botón de compra.

El frontend envía una petición al backend con los datos del producto (nombre, precio e ID).

El backend genera una sesión de pago mediante la API de Stripe.

Stripe redirige al usuario a su interfaz de Checkout.

Si el pago se completa correctamente, Stripe redirige al usuario a una URL de éxito definida por la aplicación.

En esa página de éxito, el frontend llama al backend para enviar el correo de confirmación con los detalles del producto adquirido.

Implementación en el backend

Para gestionar los pagos, se ha utilizado la librería oficial de Stripe para Java. Esta permite la creación de sesiones de pago, que encapsulan toda la información necesaria sobre la transacción: producto, cantidad, moneda y URLs de redirección tras el éxito o cancelación del proceso de pago.

El backend recibe desde el frontend los datos del producto que se desea comprar. A partir de estos datos, se crea una sesión en Stripe, y el backend devuelve un identificador de sesión que permite al cliente redirigirse a la interfaz de pago proporcionada por Stripe.

Una vez completado el pago, se inicia el proceso de envío de un correo electrónico al comprador, confirmando los detalles de la compra.

Para esta funcionalidad se ha utilizado JavaMailSender, una herramienta proporcionada por Spring para el envío de correos electrónicos. Esta librería facilita la construcción de mensajes con formato HTML y permite el uso de servidores SMTP externos como Gmail.

El sistema recupera los datos del producto desde la base de datos, y construye un mensaje con los detalles relevantes (nombre, precio, descripción, imagen, etc.). A continuación, se envía al correo electrónico del usuario que ha realizado la compra.

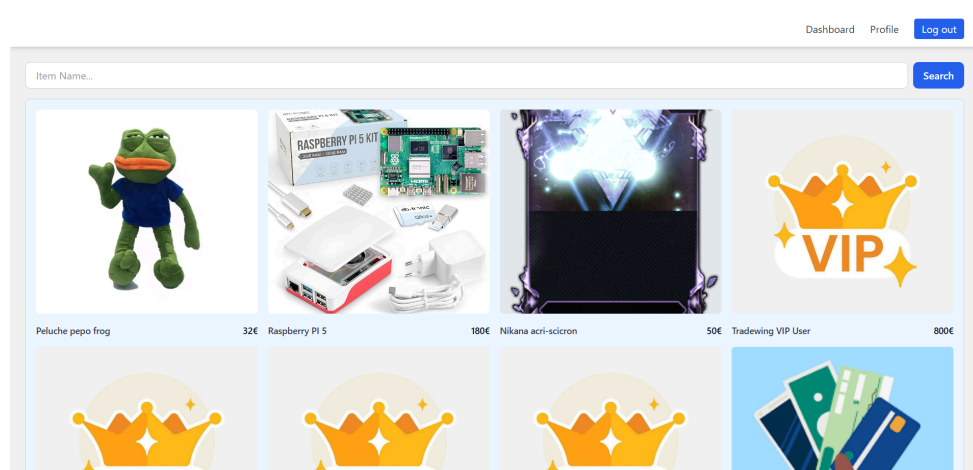
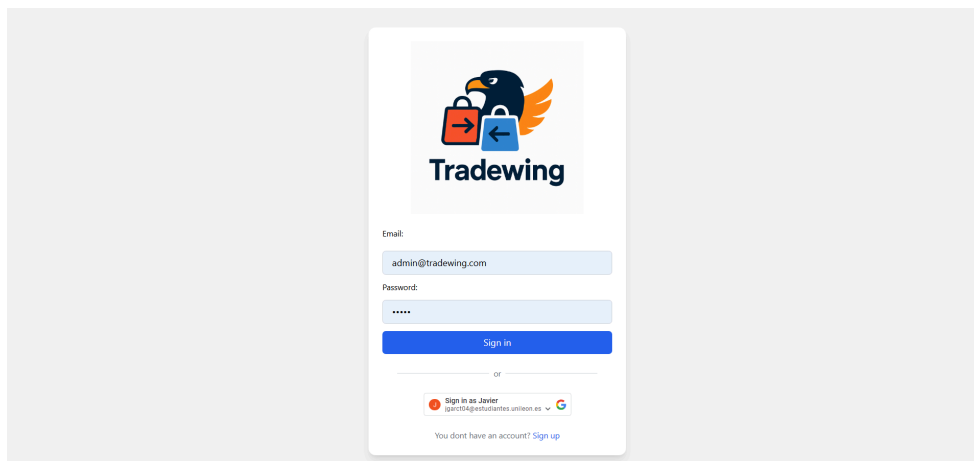
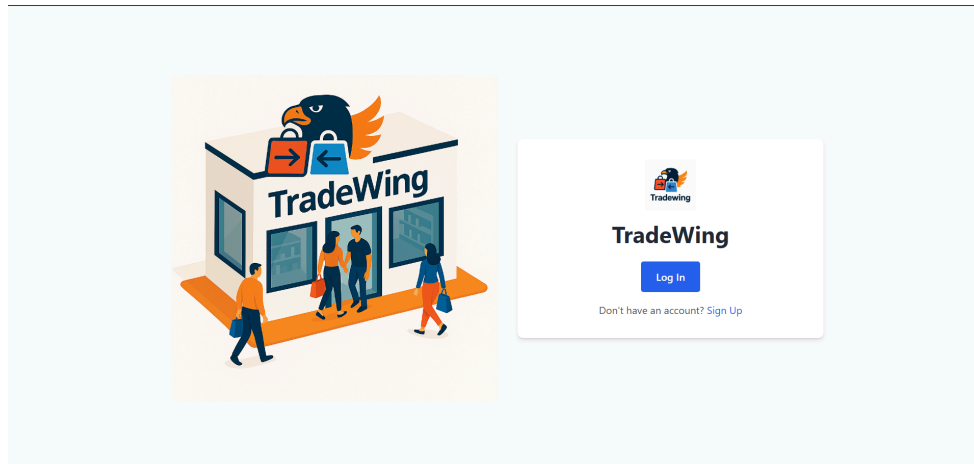
Las claves de Stripe y otros valores sensibles como las credenciales del servidor de correo se gestionan mediante el fichero `application.properties`. Esto permite adaptar fácilmente la configuración al entorno de desarrollo o producción sin modificar el código.

CI/CD y Despliegue


Para el despliegue de Tradewing se ha utilizado la plataforma Render, que permite la integración continua (CI) y el despliegue automático (CD) a partir del repositorio de GitHub. Cada vez que se realiza un push a la rama principal, Render detecta los cambios, construye automáticamente el frontend (Vue.js) y el backend (Spring Boot), y actualiza los servicios desplegados en producción sin intervención manual.

Esta automatización ha permitido mantener un flujo de trabajo ágil y controlado, reduciendo errores humanos y facilitando la entrega continua de nuevas funcionalidades. La base de datos PostgreSQL se aloja en AWS, asegurando alta disponibilidad y rendimiento en la capa de almacenamiento.

Resultados Finales y Capturas



DashboardProfileLog out



Javier García Turiel
turi3d@gmail.com

Modify Image

Upload Product

Name

Price


Image

Description

Save

For SalePurchased

DashboardProfileLog out




Peluche pepo frog
Peluche pepo frog muy bonito
€32
Shipping address:

Street, number, city, postal code...

Buy now

Seller information



Turi pro
turi3d@gmail.com

TradeWing - CheckoutEntorno de prueba

Peluche pepo frog

32,00 €

Pagar con link

Correo electrónico
correoelectronico@ejemplo.com

Método de pago

Tarjeta

Bancontact

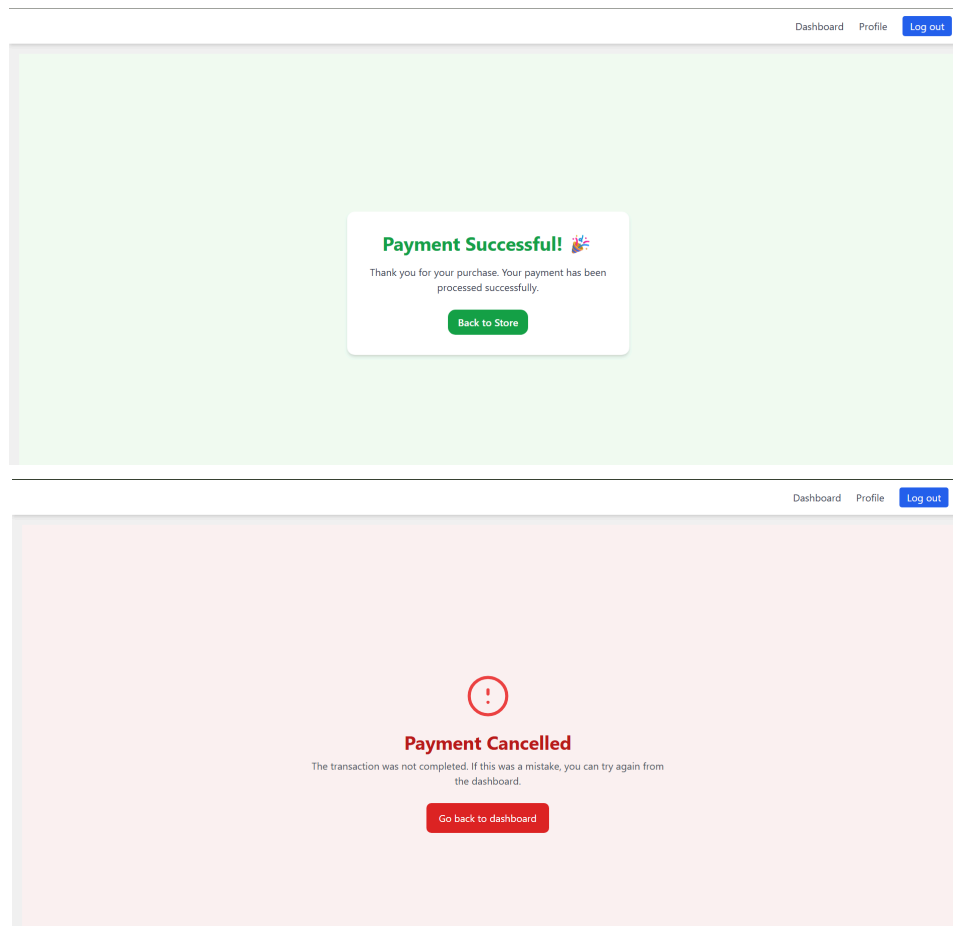
EPS

Klarna

Guardar mis datos de forma segura para un proceso de compra en un clic

Paga más rápido en TradeWing - Checkout y en todos los comercios que acepten Link.

Pagar



Proyecto Desplegado: <https://tradewing-lri3.onrender.com>

Conclusiones y Lecciones Aprendidas

Durante el desarrollo de Tradewing aprendimos a trabajar con una arquitectura web completa, integrando tecnologías modernas como Vue.js en el frontend, Spring Boot en el backend y PostgreSQL como base de datos. También adquirimos experiencia en el despliegue automático mediante Render y en la gestión de código colaborativo con Git y GitHub. Este proyecto nos permitió aplicar buenas prácticas de desarrollo, como el uso de pull requests, revisiones de código y separación de responsabilidades entre capas.

Entre las principales dificultades encontramos la configuración inicial del despliegue y la conexión con la base de datos en la nube, que resolvimos tras investigar la documentación. También tuvimos dificultades a la hora de gestionar variables de entorno correctamente y realizar pruebas controladas. También supuso un reto coordinar el desarrollo simultáneo entre frontend y backend, algo que logramos mejorar mediante una planificación clara y el uso de Github Project.

Si hubiéramos tenido más tiempo, nos habría gustado implementar un chat integrado, valoraciones de usuarios y mejorar la interfaz de usuario con un diseño más pulido. Aun así, el proyecto ha sido una experiencia muy satisfactoria tanto a nivel técnico como organizativo.

Enlaces Relevantes

Repositorio de Github: <https://github.com/GitDebaro/Tradewing>

Github Project: <https://github.com/users/GitDebaro/projects/1>

Proyecto Desplegado: <https://tradewing-lri3.onrender.com>