# TFG: Microgames

## Descripción y Justificación del Proyecto

**Microgames** es una plataforma web que ofrece a los usuarios la posibilidad de acceder a un catálogo de minijuegos desarrollados con **Phaser.js**. El objetivo principal es permitir que los usuarios puedan desbloquear y jugar distintos títulos mediante un sistema de pago seguro con **Stripe**. El proyecto está diseñado como un entorno de práctica completo para el desarrollo web, integrando tecnologías de frontend, backend y despliegue en contenedores.

El proyecto nace de la necesidad de disponer de una plataforma modular que combine entretenimiento interactivo con una infraestructura web profesional. En un contexto donde la gamificación y los micropagos están presentes en numerosos servicios digitales, **Microgames** busca demostrar cómo un desarrollador web puede integrar motores de juego (**Phaser.js**), animaciones, y un backend robusto (**Laravel con MongoDB**) dentro de un sistema escalable y dockerizado.

---

## Alcance del Proyecto

**Microgames** incluirá:

- Autenticación de usuarios
- Asignación de roles (administrador y jugador)
- Gestión de pagos con **Stripe**
- Panel de administración para configurar juegos disponibles

Los usuarios podrán crear una cuenta, explorar los minijuegos y desbloquearlos mediante un sistema de pago integrado. La aplicación estará diseñada para funcionar tanto en entornos locales como en contenedores **Docker**, facilitando su despliegue en **Proxmox**.

---

## Valoración de Alternativas

Se valoraron distintas alternativas tecnológicas:

- Para el **frontend**, **React** fue elegido sobre Vue.js por su integración nativa con **Phaser Editor** y su ecosistema más amplio.
- En **backend**, **Laravel** se seleccionó frente a **Express.js** por su robustez y estructura MVC clara.
- **MongoDB** se prefirió sobre **MySQL** debido a su flexibilidad para manejar datos no estructurados y usuarios con listas de juegos comprados.
- El uso de **Docker** facilita la portabilidad y reproducibilidad del entorno de desarrollo.

---

# Stack Tecnológico

| Capa | Tecnología | Motivo de elección |
|---|---|---|
| Frontend | React + Phaser.js | Interactividad, animaciones y soporte de plantillas desde Phaser Editor |
| Backend | Laravel (PHP) | Framework estable y seguro con soporte REST API |
| Base de datos | MongoDB | Flexibilidad en documentos y relaciones anidadas |
| Pasarela de pago | Stripe | Integración sencilla y entorno de pruebas gratuito |
| Despliegue | Docker + Proxmox | Facilita pruebas en distintos entornos |

# Objetivos del Proyecto

- Desarrollar una plataforma web modular para minijuegos con autenticación y roles.
- Implementar un sistema de micropagos mediante **Stripe** para desbloquear juegos.
- Integrar un motor de juegos (**Phaser.js**) con **React**.
- Diseñar un backend escalable con **Laravel** y base de datos **MongoDB**.
- Dockerizar el proyecto para facilitar su despliegue y mantenimiento.

---

# Requisitos del Sistema

**Requisitos funcionales:**

- Registro, inicio de sesión y gestión de cuentas
- Roles de usuario: administrador, usuario y usuario “Premium”
- Listado de minijuegos con sistema de compra/desbloqueo
- Integración de **Stripe** para pagos
- Panel de administración para añadir o modificar juegos

**Requisitos no funcionales:**

- Escalabilidad mediante contenedores **Docker**
- Seguridad en transacciones (HTTPS y Stripe SDK)
- Compatibilidad con dispositivos móviles
- Rendimiento óptimo en navegadores modernos

**Requisitos de interfaz:**

- Interfaz React con diseño responsive
- Integración visual de minijuegos desarrollados con **Phaser.js**

---

# Casos de Uso Principales

1. **Registro e inicio de sesión:** un usuario se registra con correo y contraseña, obteniendo acceso a su perfil.
2. **Compra de minijuego:** el usuario selecciona un título, realiza el pago con **Stripe** y lo desbloquea en su cuenta.

3. **Acceso al minijuego:** el usuario accede a los minijuegos de pago y los juega directamente desde la web.

4. **Gestión de juegos (administrador):** el administrador añade, edita o elimina minijuegos y consulta estadísticas.

---

# Modelo de Base de Datos (MongoDB)

El modelo de datos se compone principalmente de tres colecciones:

- **users** → { _id, name, email, password, role, purchased_games: [game_id] }
- **games** → { _id, title, description, price, asset_url }
- **transactions** → { _id, user_id, game_id, amount, stripe_id, date }