



**DPTO. INFORMÁTICA - I.E.S. DELGADO HERNÁNDEZ**

**MÓDULO PROYECTO C.F.G.S.**

**DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

**Applimon Connect**

**Autor/es: Sergio González Membrilla**

**Tutor: Daniel Cruz Fuentes**

---

**HOJA RESUMEN-PROYECTO**

<b>Título del proyecto:</b> Applimon Connect	
<b>Autor/a:</b> Sergio González Membrilla	<b>Fecha:</b> 24/04/2025
<b>Tutor/a:</b> Daniel Cruz Fuentes	
<b>Titulación:</b> DESARROLLO DE APLICACIONES MULTIPLATAFORMA	
<b>Palabras clave:</b>	
<ul style="list-style-type: none"><li>● Sitio web.</li><li>● Laravel.</li><li>● Coste 0.</li><li>● Entretenimiento.</li></ul>	
<b>Resumen del proyecto</b>	
Realizar un sitio web en el que accederemos a un juego del estilo Tamagotchi donde coleccionaremos monstruos digitales con los que podremos combatir contra otros usuario	

## **ÍNDICE**

### **1. Introducción**

- 1.1 Descripción
- 1.2 Objetivos generales
- 1.3 Motivaciones personales

### **2. Estudio de viabilidad**

- 2.1 Estudio de la situación actual
  - 2.1.1 Contexto.
  - 2.1.2 Lógica del sistema.
  - 2.1.3 Descripción física.
  - 2.1.4 Normativa, legislación y licencias.
- 2.2 Requisitos del sistema
  - 2.2.1 Requisitos.
  - 2.2.2 Restricciones del sistema.
  - 2.2.3 Catalogación y priorización de los requisitos.
- 2.3 Alternativas y selección de la solución
  - 2.3.1 Tecnología.
  - 2.3.2 Hosting.
  - 2.3.3 Dominio.
  - 2.3.4 Certificado SSL.
  - 2.3.5 Elección y conclusiones.
- 2.4 Planificación del proyecto – Diagrama de Gantt
  - 2.4.1 Recursos del proyecto.
  - 2.4.2 Tareas del proyecto.
  - 2.4.3 Planificación temporal.
- 2.5 Evaluación de riesgos
  - 2.5.1 Lista de riesgos.
  - 2.5.2 Plan de contingencia.
- 2.6 Presupuesto (Estudio de Viabilidad)
  - 2.6.1 Estimación de coste material.
  - 2.6.2 Estimación de coste personal.
  - 2.6.3 Resumen y análisis coste beneficio.

### **3. Diseño**

- 3.1 Descripción del diseño
  - 3.1.1 Selección de la arquitectura del sistema .
  - 3.1.2 Selección de base de datos.

- 4. Implementación**
  - 4.1 Configuración de los distintos subsistemas
  - 4.2 Integración de las herramientas externas
  - 4.3 Despliegue del sistema
- 5. Pruebas**
  - 5.1 Pruebas realizadas
  - 5.2 Resultados obtenidos
- 6. Conclusiones**
  - 6.1 Conclusiones finales
  - 6.2 Desviaciones temporales
  - 6.3 Possibles ampliaciones y modificaciones
  - 6.4 Valoración personal
- 7. Coste Real del Proyecto**
  - 7.1 Coste material
  - 7.2 Coste personal
  - 7.3 Resumen coste-beneficio
- 8. Bibliografía**
- 9. Glosario**
- 10. Anexo**

## **1. Introducción**

En este documento seguiremos un enfoque de desarrollo top-down, es decir, parte de lo general para luego descomponerlo en partes más detalladas hasta llegar al diseño e implementación específicos. A lo largo del documento abordaremos todas las etapas del desarrollo del proyecto, desde el planteamiento inicial hasta las pruebas finales, incluyendo conclusiones y decisiones técnicas tomadas durante su desarrollo.

“Applimon Connect” es una aplicación web de tipo videojuego que simula un entorno de crianza y combate de criaturas digitales (inspiradas en la franquicia de Digimon), la cual ha sido desarrollada con el framework Laravel y utiliza como base de datos principal PostgreSQL. La idea principal es ofrecer una experiencia entretenida y personalizable al usuario, permitiéndole capturar, entrenar, evolucionar e intercambiar criaturas para usarlas en distintos modos de juego, como combates online y modos de exploración/aventura.

El sistema está actualmente en funcionamiento de forma local, pero contempla una posible migración a un entorno online en el futuro. El proyecto no solo supone un desafío técnico completo, abarcando backend, frontend y gestión de datos, sino que también representa una oportunidad creativa para construir una mecánica de juego compleja, divertida y estratégica.

### **1.1 Descripción**

Este proyecto ha sido creado como una aplicación web tipo juego interactivo llamada **Applimon Connect**, orientada a usuarios interesados en los videojuegos de recolección, evolución y combate de criaturas virtuales, similar a lo que representan franquicias como Digimon o Pokémon.

Aunque actualmente se utiliza la marca y criaturas de Digimon (de manera no comercial), está prevista la posibilidad de reemplazar estas criaturas por diseños propios en caso de expandirse el proyecto de manera oficial o pública. En su versión actual, Applimon Connect funciona de forma local, y permite a cada usuario:

- Registrar una cuenta, pudiendo elegir un Digimon inicial entre dos opciones.
- Criar y cuidar a su criatura, atendiendo a aspectos como el hambre, la higiene, la salud y la necesidad de hacer “caca”.
- Acceder a una “DigiDex” que registra todas las criaturas capturadas, intercambiadas o evolucionadas.
- Usar una “Caja” para organizar y seleccionar a su Digimon activo.
- Entrenar diferentes estadísticas del Digimon, como el ataque, la defensa o la vida.
- Participar en combates contra Digimon salvajes o en modo competitivo online.
- Realizar intercambios con otros usuarios.
- Evolucionar o involucionar a sus criaturas, lo que desbloquea nuevas formas y capacidades.

El sistema ha sido desarrollado con Laravel (PHP) y usa una arquitectura de base de datos distribuida con una base de datos principal PostgreSQL y conexiones con APIs externas como YouTube o Imgur. Cada criatura cuenta con su propia animación, cada ventana tiene su propia banda sonora y se ha trabajado en ofrecer una interfaz amigable con capacidad de añadir más funcionalidades y actualizaciones en el futuro.

## **1.2 Objetivos generales**

A continuación, describiremos los objetivos generales del proyecto, para lograr que la aplicación sea funcional, extensible y entretenida:

### **OBJ1. Establecer el entorno de desarrollo y la arquitectura básica**

Elegir las tecnologías adecuadas para la construcción de una aplicación interactiva con funcionalidades de juego, incluyendo Laravel para la lógica del servidor y PostgreSQL para el almacenamiento de datos.

### **OBJ2. Desarrollar un sistema de usuario con autenticación**

Permitir que nuevos usuarios se registren, inicien sesión y seleccionen su primer Digimon para comenzar la aventura.

### **OBJ3. Implementar las mecánicas básicas de crianza**

Desarrollar funciones que simulen el cuidado de criaturas, afectando a su rendimiento en combate según sus estadísticas internas.

### **OBJ4. Crear un sistema de combate funcional**

Diseñar un sistema que permita enfrentamientos entre Digimon, tanto en modo PvE (contra la IA) como en PvP (jugador contra jugador), que tenga en cuenta estadísticas, niveles y experiencia.

### **OBJ5. Gestionar evolución e involución de Digimon**

Permitir al usuario modificar el estado evolutivo de su criatura según requisitos específicos, alterando así sus atributos y posibilidades estratégicas.

### **OBJ6. Añadir la funcionalidad de DigiDex y Caja**

Construir interfaces visuales donde se pueda consultar el historial de criaturas adquiridas y gestionar la colección personal del usuario.

### **OBJ7. Implementar funcionalidades sociales**

Desarrollar sistemas de intercambio de criaturas entre usuarios, combates online con rankings, y otras formas de interacción(para implementarlas en versiones futuras).

### **OBJ8. Almacenar y recuperar información multimedia**

Conectar con servicios externos como Imgur o YouTube para gestionar las animaciones de los Digimon o videos que contienen la banda sonora de la aplicación de forma eficiente.

### **OBJ9. Preparar la aplicación para despliegue futuro**

Dejar preparada una arquitectura que permita trasladar fácilmente el sistema desde un entorno local a uno público, incluyendo la posibilidad de añadir certificados SSL, dominios y alojamiento en servidores online.

### 1.3 Motivaciones personales

He elegido este proyecto porque representa uno de mis principales intereses: los videojuegos. Desde pequeño me han gustado los juegos de criaturas como Digimon, y crear una aplicación que simule esa experiencia ha sido una forma entretenida de aplicar todos los conocimientos adquiridos durante el grado y las prácticas.

Inicié este proyecto en Android Studio como una entrega de una asignatura. Al principio era pequeña, pero al añadirle una nueva función tras otra, no sólo sobrepasaba lo que pedía la entrega inicialmente, si no que empezaba a tener “personalidad propia”. Al ver su potencial, decidí proponerla como mi proyecto de fin de grado. Después de una revisión, que implicaba el no poder usar el código original (algo que se tenía que hacer por fuerza ya que quería aprovechar lo visto en las prácticas y pasar la idea a un proyecto laravel), se dió el visto bueno.

Una de las partes que más me ha gustado ha sido diseñar las mecánicas de juego (evolución, combates, entrenamiento), ya que me ha permitido pensar no sólo en la implementación técnica, sino también en la experiencia del usuario. Además, me ha resultado motivador trabajar con Laravel y bases de datos relacionales, herramientas que ya conocía, pero en las que ahora tengo un mayor manejo.

Por otro lado, una de las mayores dificultades ha sido la gestión de múltiples estados para los Digimon, como los cambios al evolucionar/involucionar, el impacto de sus estadísticas tras sus entrenamientos o la reducción de las variables que controlan la “Felicidad” del Digimon, que el usuario tendrá que controlar. Aún así, ha sido una oportunidad para aprender sobre lógica compleja en entornos web, control de sesiones, relaciones entre tablas, y planificación de interfaces interactivas.

Este proyecto no solo me ha ayudado a reforzar mis habilidades técnicas obtenidas durante mis prácticas, sino que también me ha dado confianza para enfrentarme a desarrollos más ambiciosos y creativos en el futuro.

## 2. Estudio de viabilidad

### 2.1 Estudio de la situación actual

#### 2.1.1 Contexto

**Applimon Connect** nace con la idea de ofrecer a los usuarios una experiencia interactiva nostálgica, donde puedan colecciónar, criar, combatir y relacionarse a través de monstruos digitales, o comúnmente llamadas: "Digimon" (Digital Monster). El sistema está inspirado en mecánicas popularizadas por franquicias clásicas, pero adaptadas al entorno web moderno.

El proyecto actualmente se ejecuta de manera local, usando Laravel como framework base y PostgreSQL como base de datos principal, además de conectarse a otras fuentes como YouTube e Imgur. Cada usuario puede seleccionar un Digimon inicial y, a partir de ahí, gestionar su experiencia a través de diferentes funcionalidades dentro del menú principal de la aplicación.

Los principales actores del sistema son:

- **Usuarios registrados:** gestionan Digimon, entrenan, combaten, evolucionan, etc.
- **Sistema:** gestiona estados (hambre, salud, etc.), lógica de combate, base de datos y animaciones.

**Objetivo general:** ofrecer una plataforma funcional, gamificada y escalable que en un futuro pueda evolucionar hacia un producto en línea profesional, acatando licencias y derechos de uso correspondientes.

#### 2.1.2 Lógica del sistema

El sistema estará compuesto por una aplicación web desarrollada en Laravel. Requiere de un servidor (initialmente local) donde residirá tanto la lógica de la aplicación como la base de datos. En el futuro, se contempla su despliegue en un hosting gratuito (InfinityFree) o en una Raspberry Pi para su funcionamiento 24/7.

**Componentes:**

- **Frontend:** Blade, HTML, CSS, JavaScript
- **Backend:** Laravel (PHP)
- **Base de datos principal:** PostgreSQL
- **Bases de datos auxiliares:** YouTube (para audio) e Imgur (para imágenes/animaciones)
- **Actores:**
  - Usuario (jugador)
  - Sistema (motor de reglas y lógica del juego)
  - Administrador (futuro: revisión de comportamiento de usuarios)

### 2.1.3 Descripción física

En la etapa actual, el sistema opera en un entorno local. El único hardware requerido es un equipo capaz de ejecutar Laravel y PostgreSQL(en nuestro caso, hemos usado un MacBook Pro), lo cual puede incluir un ordenador personal.

Para un despliegue futuro se consideran:

- Raspberry Pi 4 (24/7, bajo consumo)
- Servidores gratuitos como InfinityFree
- PC de escritorio para desarrollo y pruebas

### 2.1.4 Normativa, legislación y licencias

Actualmente, en Applimon Connect utilizamos personajes y recursos de la franquicia Digimon y Pokémon(este último en muy menor medida y fácil de cambiar), por lo que para una eventual publicación o monetización será **necesaria la adquisición de licencias oficiales** de la IP o creación de recursos originales.

A su vez, se garantizará el cumplimiento de la legislación vigente:

- **RGPD y LOPDGDD:** para la protección de datos de los usuarios
- **Aviso legal y política de privacidad:** implementadas en el sitio
- **Consentimiento explícito del usuario** al registrarse y al compartir datos
- **Medidas de seguridad:** sanitización de inputs, CSRF tokens, protección contra inyecciones SQL

## 2.2 Requisitos del sistema

### 2.2.1 Requisitos

ID	Descripción
REQ 1.0	El sistema debe permitir el registro e inicio de sesión de usuarios.
REQ 2.0	El usuario podrá elegir entre dos Digimon iniciales al crearse su usuario.
REQ 3.0	El usuario podrá acceder a un menú principal con las distintas opciones de gestión.
REQ 4.0	Se podrán visualizar y administrar las estadísticas del Digimon activo.
REQ 5.0	El sistema actualizará automáticamente valores como hambre o salud con el tiempo, mostrándole al usuario la “Felicidad” de su compañero activo.
REQ 6.0	El usuario podrá entrenar atributos como ataque, defensa y vida.
REQ 7.0	El usuario podrá combatir contra Digimon salvajes, obteniendo experiencia y dinero(Bits).
REQ 8.0	El sistema permitirá la captura de Digimon en una batalla si se cumple la probabilidad.
REQ 9.0	El usuario podrá intercambiar Digimon con otros jugadores.
REQ 10.0	Existirá un sistema de evolución/involución con líneas evolutivas únicas por Digimon.
REQ 11.0	Combate online con clasificación por puntos ganados o perdidos.

### 2.2.2 Restricciones del sistema

ID	Restricción
REX 1	Es obligatorio registrarse para acceder a las funcionalidades del sistema.
REX 2	No se podrá usar Digimon licenciados en un producto final sin autorización oficial.
REX 3	El número de entrenamientos disponibles es limitado, es decir; en su versión inicial, en Applimon no se podrán revertir los cambios hechos en las estadísticas del Digimon, haciendo que los entrenamientos del mismo sean irreversibles.
REX 4	El sistema sólo funcionará localmente en su versión inicial.
REX 5	El sistema no contará inicialmente con soporte multilingüe ni responsividad completa.

### 2.2.3 Catalogación y priorización de los requisitos

Prioridad	Requisitos
Alta	REQ 1.0, REQ 6.0, REQ 3.0, REQ 7.0, REQ 10.0
Media	REQ 9.0, REQ 4.0, REQ 5.0, REQ 8.0
Baja	REQ 2.0, REQ 11.0

## 2.3 Alternativas y selección de la solución

### 2.3.1 Tecnología

- **Laravel (seleccionado)**: Framework PHP moderno y robusto que permite crear aplicaciones web seguras, escalables y bien estructuradas.
- **PHP puro y JavaScript**: Alternativa viable, pero más difícil y menos estructurada.
- **WordPress**: Con limitaciones para personalizar lógicas de juego.

### 2.3.2 Hosting

- **InfinityFree (seleccionado)**: Gratuito, con PHP y bases de datos MySQL, SSL incluido.
- **Hosting local con Raspberry Pi**: Perfecto para tests locales o demostraciones offline.
- **000Webhost**: Alternativa básica con muchas limitaciones en almacenamiento.

### 2.3.3 Dominio

- **Subdominio gratuito de InfinityFree (seleccionado)**: Suficiente para una fase inicial de pruebas públicas en el futuro.
- **FreeDNS / Red.es / GoDaddy**: Opciones viables en caso de profesionalización del producto.

### 2.3.4 Certificado SSL

- **Let's Encrypt / InfinityFree (seleccionado)**: Solución gratuita y segura.
- **Actalis**: Alternativa de pago si se requieren más validaciones en el futuro.

### 2.3.5 Elección y conclusiones

Se opta por desarrollar Applimon Connect usando Laravel(aparte de para demostrar lo aprendido en las prácticas), ya que ofrece una estructura robusta para manejar las múltiples funcionalidades requeridas por el proyecto. Para la fase inicial se utilizará un entorno local, y eventualmente se migrará a un entorno gratuito como InfinityFree, con certificado SSL incluido.

En un futuro, se contempla la compra de una licencia oficial para el uso de personajes Digimon o el desarrollo de nuevas criaturas propias que permita independencia comercial total.

## 2.4 Planificación del proyecto – Diagrama de Gantt

### 2.4.1 Recursos del proyecto

#### **Humanos**

- 1 desarrollador full stack (yo mismo).

#### **Materiales**

- Ninguno (el desarrollo se realiza en entorno digital).

#### **Software**

- Laravel (framework PHP para backend).
- PostgreSQL (base de datos principal).
- APIs externas (YouTube, Imgur).
- GitHub (control de versiones).
- XAMPP / Laravel / Docker (entorno local).
- Piskel (Herramientas de diseño y edición de sprites).
- [Opcional futuro] Servidor web (ej: DigitalOcean, AWS).

### 2.4.2 Tareas del proyecto

Estas tareas se organizan de forma lógica en base al flujo de desarrollo de una app tipo RPG online multicomponente como Applimon Connect:

#### **1. Diseño conceptual del sistema**

- Definir funcionalidades generales y específicas (colección, combate, evolución, etc.).
- Crear primero las tablas en la base de datos.

#### **2. Configuración del entorno de desarrollo local**

- Instalación de Laravel y dependencias.
- Configuración de PostgreSQL y creación de la base de datos principal.
- Integración con bases de datos secundarias (YouTube API, Imgur).

#### **3. Desarrollo del sistema de usuarios**

- Registro e inicio de sesión.
- Elección inicial del Digimon.
- Asociación de usuario ↔ Digimon activo.

#### **4. Diseño e implementación de vistas principales (UI/UX básico)**

- Pantalla de menú principal.
- Vistas para: Cuidar, Digidex, Caja, Entrenamiento, Aventura, Intercambio, Combate online, Evolución.

#### **5. Desarrollo de funcionalidades del sistema**

- Sistema de estadísticas y su degradación temporal (hambre, salud, etc.).
- Sistema de combate (IA básica contra Digimon salvajes y lógica de PvP asincrónico).
- Sistema de experiencia, niveles y economía interna(Bits).
- Sistema de evolución/involución con condiciones.
- Interfaz de intercambio con otros usuarios.
- Almacenamiento y recuperación desde la "Caja".

- 6. Implementación de animaciones y multimedia**
  - Integrar animaciones para cada Digimon usando Imgur.
  - Conexión con YouTube (para reproducir música de fondo).
- 7. Pruebas locales y beta testing**
  - Validación de funcionalidades.
  - Corrección de errores y ajustes.
- 8. Implementación de medidas básicas de seguridad**
  - CSRF, validación de entradas.
- 9. Preparación para despliegue (futuro)**
  - Diseño de plan de hosting (opcional en esta versión).
  - Estudio de requerimientos legales (licencia Digimon, privacidad de datos).

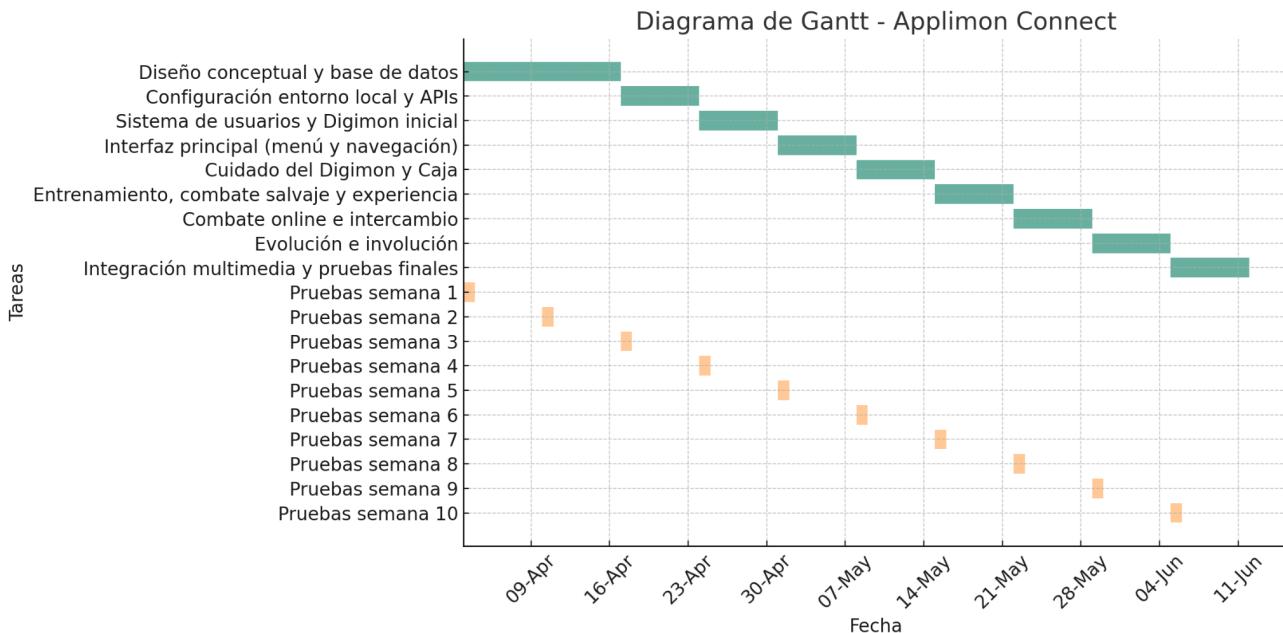
#### 2.4.3 Planificación temporal

**Periodo: 3 de abril al 12 de junio (aproximadamente 10 semanas)**

Semana	Tareas
1-2	Diseño conceptual, definición de funcionalidades, diagrama de base de datos.
2-3	Configuración del entorno local, creación del proyecto Laravel, conexión a base de datos PostgreSQL y APIs.
3-4	Sistema de usuarios, login y elección inicial de Digimon.
4-5	Implementación de la interfaz principal (menú, navegación, primeras vistas).
5-6	Desarrollo del sistema de estadísticas, cuidado del Digimon y funcionalidades de la "Caja".
6-7	Entrenamiento, combate (vs salvajes) y sistema de experiencia.
7-8	Combate online, clasificación, sistema de intercambio.
8-9	Evolución e involución, validación de líneas evolutivas.
9-10	Integración de animaciones, pruebas generales, validaciones, medidas de seguridad básicas.
Final	Documentación, resumen de funcionalidades, y presentación del proyecto.

El cronograma inicial se ajustaba dependiendo de la complejidad de ciertas funciones como el combate online o el sistema de evolución, otorgándoles más tiempo. Al final de cada semana se hacían pruebas para comprobar que todo funcionara correctamente.

Lo que vemos ahora es el Diagrama de Gantt que se esperaba realizar idealmente, es decir; sin cambios, retrasos ni ajustes.



## 2.5 Evaluación de riesgos

### 2.5.1 Lista de riesgos

Los riesgos identificados en el desarrollo del proyecto *Applimon Connect* son:

- **Fallas de seguridad:** Posibles ataques como inyecciones SQL, CSRF, etc.
- **Errores en la lógica de la aplicación:** Bugs o errores en la lógica que afecten al comportamiento del sistema.
- **Falta de cumplimiento con normativas:** Especialmente aquellas relacionadas con la protección de datos.
- **Dependencia de servicios de terceros:** Como YouTube o Imgur.
- **Problemas de escalabilidad:** Posible degradación del rendimiento con el aumento de usuarios y criaturas.
- **Desconocimiento del usuario:** Interfaz poco intuitiva que podría desalentar el uso.
- **Pérdida de acceso a la cuenta de administración:** Riesgo de que se pierdan las credenciales del usuario administrador.
- **Interrupciones del servicio:** Caídas del servidor o problemas con el hosting gratuito (riesgo de futuras versiones).
- **Incompatibilidad del navegador:** La aplicación puede no funcionar correctamente en todos los navegadores o dispositivos.

## 2.5.2 Plan de contingencia

A cada uno de los riesgos se le asigna una solución preventiva o paliativa:

- **Fallas de seguridad**

*Solución:* Uso de Eloquent ORM de Laravel, validación de entradas, tokens CSRF. Actualización continua del software.

- **Errores en la lógica de la aplicación**

*Solución:* Realizar pruebas unitarias y funcionales frecuentes. Involucrar a testers para detectar fallos antes de pasar a producción.

- **Falta de cumplimiento con normativas**

*Solución:* Se implementará una política de privacidad clara, y el sistema solicitará consentimiento explícito antes de recoger datos sensibles.

- **Dependencia de servicios de terceros**

*Solución:* En caso de fallos de APIs externas, se considerará almacenamiento interno alternativo o migración a otros servicios gratuitos.

- **Problemas de escalabilidad**

*Solución:* Desde el diseño inicial, se emplearán prácticas como consultas optimizadas y almacenamiento en caché.

- **Desconocimiento del usuario**

*Solución:* El sistema se diseñará de forma intuitiva, inspirado en interfaces conocidas por los usuarios.

- **Pérdida de acceso a la cuenta de administración**

*Solución:* Se implementarán mecanismos de recuperación (preguntas de seguridad, verificación por correo). Este punto sólo se abordará si se dispone de tiempo adicional.

- **Interrupciones del servicio**

*Solución:* Realizar copias de seguridad periódicamente. Si el servicio gratuito no es suficiente, se propondrá la migración a un hosting de pago más robusto.

- **Incompatibilidad del navegador**

*Solución:* Pruebas en los navegadores más usados (Chrome, Firefox, Edge, Safari). El diseño responsive no será prioridad inicial, pero se tendrá en cuenta si el tiempo lo permite o en versiones futuras.

## 2.6 Presupuesto (Estudio de Viabilidad)

### 2.6.1 Estimación de coste material

El desarrollo de *Applimon Connect* se ha planteado utilizando herramientas y servicios gratuitos:

Elemento	Recurso	Coste
Hosting(futuro)	InfinityFree	0 €
Dominio(futuro)	Subdominio gratuito de InfinityFree	0 €
Framework	Laravel (open-source)	0 €
Entorno local	XAMPP / PHP / Composer	0 €
Base de datos	PostgreSQL y MySQL (gratuitos)	0 €

**Total estimado: 0 €**

### 2.6.2 Estimación de coste personal

El desarrollo del proyecto se ha llevado a cabo entre el **3 de abril y el 12 de junio**, lo que equivale a aproximadamente **10 semanas naturales**. Si se estima que se han trabajado **5 días por semana, 5 horas por día**, obtenemos:

$$10 \text{ semanas} \times 5 \text{ días/semana} \times 7 \text{ horas/día} = 350 \text{ horas}$$

$$350 \text{ horas} \times 12 \text{ €/hora} = 4200 \text{ €}$$

**Coste personal estimado: 4200 €**

### 2.6.3 Resumen y análisis coste beneficio

Dado que el coste material es de **0 €**, el único coste es el tiempo invertido, valorado en **4200 €**. El beneficio neto estimado, sería de esa misma cantidad, ya que todo el desarrollo ha sido autogestionado y sin inversión externa.

Además, en caso de venta:

- Se podría establecer una **tarifa anual de 50 €** para mantenimiento básico (renovación de certificados SSL, ajustes menores, administración de base de datos, etc.).
- El sistema puede mejorar en el futuro con mínima inversión adicional, haciéndolo viable tanto técnica como económicaamente.

### **3. Diseño**

#### **3.1 Descripción del diseño**

##### **3.1.1. Selección de la arquitectura del sistema**

Actualmente, **Applimon Connect** se ejecuta en un entorno **local** para su desarrollo y prueba. Esto significa que tanto el código como las bases de datos están en el mismo equipo. Sin embargo, se ha diseñado con la previsión de poder migrarlo fácilmente a un **servidor remoto**, donde cada componente esté correctamente segmentado y protegido.

En un entorno futuro de producción, se podría plantear una arquitectura física clásica con los siguientes componentes:

- Una **zona DMZ** para alojar los archivos públicos (accesibles desde la web), como las vistas y scripts JavaScript.
- Una **base de datos principal PostgreSQL**, alojada en un entorno protegido, con acceso restringido únicamente desde el backend de la aplicación.
- Bases de datos externas **secundarias**, como las de **YouTube** (para visualización de animaciones) e **Imgur** (para las imágenes de los Digimon), que son utilizadas de forma no intrusiva y vía API.
- En el futuro, se contemplaría el uso de un servidor de correos externo para gestión de notificaciones (registro, recuperación de contraseña, etc.), preferiblemente mediante servicios como SendGrid o Mailgun.

##### **3.1.2. Selección de la arquitectura del sitio web**

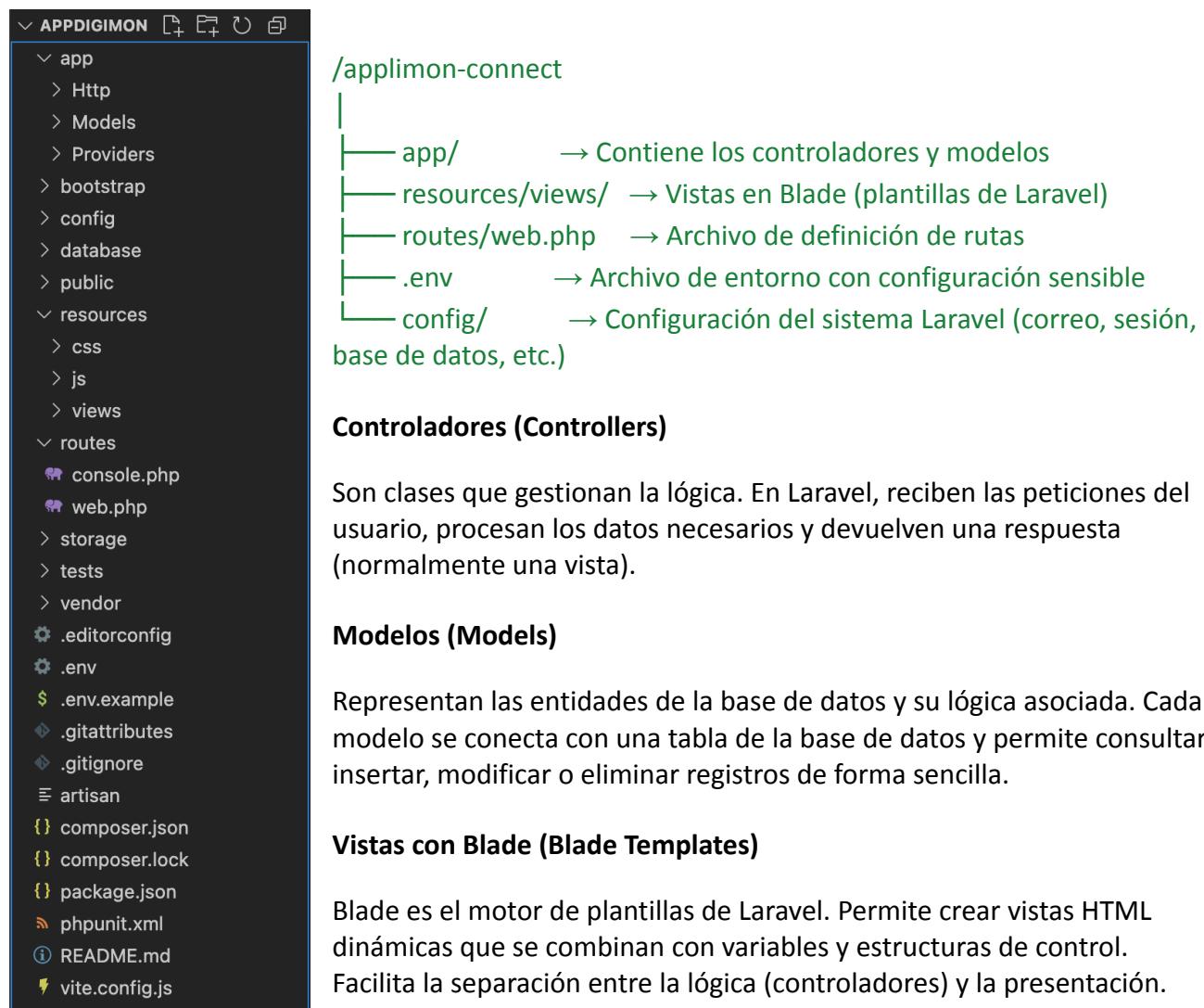
**Applimon Connect** está desarrollado con **Laravel**, un framework PHP moderno que sigue la arquitectura **Modelo – Vista – Controlador (MVC)**. Esta estructura permite separar claramente la lógica del negocio (modelos y controladores) de la presentación (vistas), lo que facilita tanto el mantenimiento como la escalabilidad del sistema.

Laravel, además, incluye herramientas integradas para:

- Gestión de rutas.
- Validación de formularios.
- Control de sesiones y autenticación.
- Seguridad frente a ataques comunes (CSRF, XSS, SQL Injection).
- Uso de archivos de configuración separados mediante el archivo `.env`, incluyendo credenciales de base de datos y claves API.

## Estructura general del proyecto

A continuación, se resume la organización principal de carpetas y archivos en el entorno de desarrollo:



### web.php

Archivo de rutas principales en Laravel, ubicado en routes/web.php. Aquí se definen las URLs que puede visitar el usuario y qué controlador o acción debe ejecutarse para cada una.

### .env

Archivo de configuración de entorno (ubicado en la raíz del proyecto). Contiene variables sensibles y específicas del entorno, como las credenciales de la base de datos, claves de API, nombre del proyecto, URL base, etc. Laravel carga automáticamente estas variables al arrancar la aplicación.

## Detalle por módulos funcionales

El proyecto se estructura de forma modular según las funcionalidades principales:

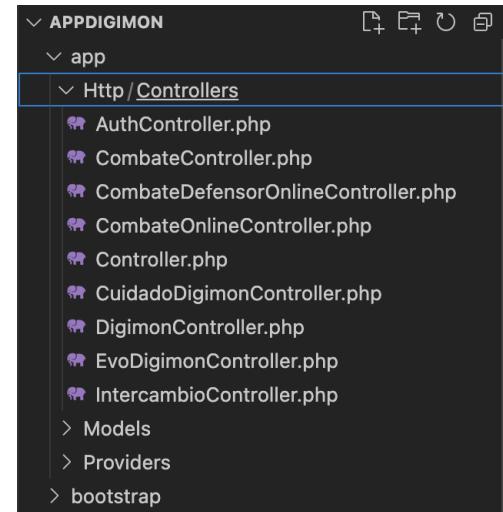
### Controladores:

#### Módulo de autenticación (**AuthController**)

- Registro, inicio de sesión, cierre de sesión.
- Validaciones básicas de seguridad.
- Control del decremento de los valores de la Felicidad.

#### Módulo de combate (**CombateController**)

- Maneja todo lo que tenga que ver con una batalla contra un Digimon salvaje, desde las acciones durante éste hasta los resultados (como la experiencia o los Bits ganados).
- Control sobre la dificultad de las rutas para luchar contra Digimon salvajes.



#### Módulo de combate online (**CombateDefensorOnlineController**)

- Igual que el anterior, pero contra Digimon de usuarios y controlando la clasificación de éstos.

#### Módulo de Gestión de combates online (**CombateOnlineController**)

- Menú donde puedes dejar a tu Digimon defensor, retar a otros jugadores y ver tu clasificación y estadísticas.

#### Módulo de cuidado (**CuidadoDigimonController**)

- Estadísticas como hambre, salud, higiene y “caca” se gestionan aquí.
- El coste de mejora está relacionado con los Bits ganados en combate.

#### Módulo Digimon (**DigimonController**)

- Controla todo lo relacionado con los Digimon que no estén en otros controladores, como la creación de estos, entrenamientos, Digidex, la caja, etc.

#### Módulo de evolución (**EvoDigimonController**)

- Permite evolucionar o involucionar al Digimon activo.
- Cada línea evolutiva tiene sus propios requisitos.
- La involución no tiene restricciones y proporciona bonificaciones especiales.

#### Módulo de intercambio (**IntercambioController**)

- Gestión de ofertas y solicitudes de intercambio entre usuarios.
- Búsqueda de Digimon disponibles en el mercado.
- Control de intercambios activos y pendientes.

## Modelos:

Representan las tablas y las relaciones en la base de datos.

### Tabla de Torneos (**CombateOnline**)

- Registro de la clasificación, victorias, derrotas, puntos y Digimon defensor de cada usuario.

Models
CombatOnline.php
Digimon.php
Intercambio.php
ListaCancion.php
ListaDigimon.php
User.php
Usuario.php
> Providers

### Tabla de Digimon (**Digimon**)

- Datos que todos los Digimon capturados por todos los usuarios.
- Entre los más importantes están su nivel, estadísticas entrenadas, usuario al que pertenece, bonificaciones, sus llaves(desbloquea evoluciones),etc.
- Todo Digimon creado se basa según los que existen en ListaDigimon.

### Tabla de Intercambios (**Intercambios**)

- Parecida a la de torneos. Gestiona todos los intercambios que los usuarios generan.
- El usuario ofrece un Digimon y pide otro a cambio.

### Tabla de Canciones (**ListaCancion**)

- Donde se guardan ordenadamente todas las canciones

### Tabla de Digimon Originales (**ListaDigimon**)

- Como ya se ha dicho, aquí guardamos todos los datos básicos con los que se crea un digimon de un usuario.
- Entre dichos datos están su animación, evoluciones e involuciones, etapa, elemento, estadísticas base(necesarias para calcular las reales), entre otros.

### Tabla de Usuarios (**Usuarios**)

- Donde se guarda toda la información del usuario, como su nombre, contraseña, correo, su Digimon actual, Bits y una lista con los Digimon que ha adquirido.

## Vistas:

Todo lo que el usuario llega a ver en la aplicación al final.

Botones, colores, tablas, despliegues, columnas, etc.

Todas estas vistas son manejadas tanto por los controladores como por el archivo web.php, que contiene las rutas que indica qué función utilizará qué vista.

views
combate_online
buscar.blade.php
combate2.blade.php
depositar.blade.php
index.blade.php
intercambio
buscar.blade.php
depositar.blade.php
index.blade.php
resultados.blade.php
seleccionar_digimon_usuario.blade.php
seleccionar.blade.php
cambiarDigimon.blade.php
combate.blade.php
crearDigimon.blade.php
cuidar.blade.php
digidex.blade.php
eleccion.blade.php
entrenar.blade.php
evolucionar.blade.php
home.blade.php
image_redirect.blade.php
login.blade.php
register.blade.php
rutaseleccion.blade.php
show.blade.php
welcome.blade.php

### 3.1.2. Selección de base de datos

Para el desarrollo de **Applimon Connect**, se ha utilizado como base de datos principal **PostgreSQL**, elegida por su robustez, soporte para integridad referencial, eficiencia en consultas complejas y buena escalabilidad para proyectos que manejan relaciones entre múltiples entidades como Digimon, usuarios, combates, intercambios, etc.

\*Las tablas no mencionadas no son importantes para la aplicación

En el futuro, esta estructura puede migrarse sin problemas a una instancia en la nube como **Amazon RDS**, **Supabase** o **Render**, manteniendo compatibilidad gracias a la estructura ORM de Laravel (Eloquent).

Además, se integran **fuentes de datos externas** como:

- **YouTube**: para mostrar animaciones propias de cada Digimon, embebidas mediante enlaces o API.
- **Imgur**: utilizada como CDN para almacenar imágenes estáticas de Digimon, sprites o fondos del juego.

Tables (17)
> cache
> cache_locks
> combates_online
> cursos
> digimon
> failed_jobs
> gifs
> intercambios
> job_batches
> jobs
> lista_canciones
> lista_digimon
> migrations
> password_reset_tokens
> sessions
> users
> usuarios

### Estructura de las principales tablas

El contenido de las tablas ya se han descrito en los modelos del punto **3.1.2. Selección de la arquitectura del sitio web**, pero aquí podremos ver los ejemplo más importantes para ver cómo se guardan los datos:

#### Digimon:

	<b>id [PK] integer</b>	<b>id_usuario integer</b>	<b>id_lista_digimon integer</b>	<b>nombre character varying (255)</b>	<b>nivel integer</b>	<b>ataquebase integer</b>	<b>defensabase integer</b>	<b>vidabase integer</b>	<b>experienciabase integer</b>	<b>experienciaactual integer</b>	<b>experienciasiguientenivel integer</b>	<b>idevolv integer</b>
1	95	27	19	Guardromon	14	20	35	125	150	30	110	
2	96	27	2	Coronamon	11	18	12	90	100	30	60	
3	97	27	26	Hyokomon	1	28	10	82	100	0	20	
4	98	28	30	Lopmon	1	15	18	85	100	0	20	
5	99	28	1	Sunmon	1	12	8	40	50	0	10	
6	100	27	26	Hyokomon	1	28	10	82	100	0	20	
7	101	26	31	Wendimon	1	30	25	140	150	0	50	
8	102	27	9	Chicomon	1	14	8	38	50	0	10	
9	103	27	24	Shawujinmon	1	25	35	180	200	0	75	
10	104	26	10	Veemon	3	24	10	86	100	10	40	
11	105	28	2	Coronamon	2	18	12	90	100	0	30	
12	106	28	14	Terriermom	4	15	12	93	100	10	50	

## ListaDigimon:

	<b>id [PK] integer</b>	<b>nombre character varying (100)</b>	<b>nivel integer</b>	<b>ataquebase integer</b>	<b>defensabase integer</b>	<b>vidabase integer</b>	<b>experienciabase integer</b>	<b>experienciaactual integer</b>	<b>experienciasiguientenivel integer</b>	<b>idevolucion integer</b>	<b>idinvolucion integer</b>	<b>etapa character</b>
1	1	Sunmon	1	12	8	40	50	0	10	2	0	Bebé
2	2	Coronamon	1	18	12	90	100	0	20	3	1	Principiar
3	3	Firamon	1	30	20	130	150	0	50	4	2	Campeón
4	4	Flaremon	1	30	25	185	200	0	75	0	3	Mega-Car
5	5	Minomon	1	8	12	40	50	0	10	6	0	Bebé
6	6	Wormmon	1	22	18	80	100	0	20	7	5	Principiar
7	7	Stingmon	1	35	30	115	150	0	50	8	6	Campeón
8	8	Dinobeamon	1	50	40	150	200	0	75	0	7	Mega-Car
9	9	Chicomon	1	14	8	38	50	0	10	10	0	Bebé
10	10	Veemon	1	24	10	86	100	0	20	11	9	Principiar
11	11	Veedramon	1	42	15	123	150	0	50	12	10	Campeón
12	12	AeroVeedramon	1	50	20	170	200	0	75	0	11	Mega-Car
13	13	Gummymon	1	10	9	41	50	0	10	14	0	Bebé

## Usuarios:

	<b>id [PK] integer</b>	<b>nombreusuario character varying (255)</b>	<b>correousuario character varying (255)</b>	<b>contraseñausuario character varying (255)</b>	<b>digimon_id bigint</b>	<b>digimones_adquiridos text</b>	<b>ultima_salida timestamp without time zone</b>
1	26	1	1@1	\$2y\$12\$8k4PVOrUsoh3sK0jOpXI.avvKpsl.aQ7Ime0SgiuhC0soysv88vW	104	30,16,4,32,14,9,25,21,13,29,11,10,26,31,18	2025-06-05 15:25:1
2	27	2	2@2	\$2y\$12\$8Af600orAhtGy9XpH0afeT7b/LKUbYbtX0RwIY3OCtJHUVEJ...	96	2,26,31,1,22,4,11,23,32,22	2025-06-05 14:56:1
3	28	3	3@3	\$2y\$12\$sbAwbPWQqA.2EpliWS2u0tmBabE5xspXjNtmoD0kiE9Ypfea...	105	30,1,32,4,20,14,10,3,2,7	2025-06-05 15:39:4
4	29	4	4@4	\$2y\$12\$BoxcJhhKkpE1UsFIx6siOt63Nk26Xd9kYYhC8yH.hdODHd.7D...	114	30,22	2025-06-05 14:55:3
5	30	5	5@5	\$2y\$12\$Ylp4R3q7HYm7Jwmn0WQcOF6AeoPc.1WXMTjEBk6/G1hxYfw...	134	2,3,5,31,23,12,11,10	2025-06-05 15:17:4
6	31	rub2	rub@gmail.com	\$2y\$12\$vspx4BzrZs9.kQlJhM300HJCo.Cvg0CuvbcPbTau21TECBrq0yW	141	2,21	2025-06-05 16:37:3

## Combates Online:

	<b>id [PK] integer</b>	<b>id_usuario integer</b>	<b>id_digimon_defensor integer</b>	<b>victorias integer</b>	<b>derrotas integer</b>	<b>empates integer</b>	<b>clasificacion integer</b>	<b>creado_en timestamp without time zone</b>	<b>puntos integer</b>	<b>usuarios_combatidos text</b>	<b>ultima_reset timestamp without time zone</b>
1	1	26	128	2	2	0	2	2025-06-01 18:17:55.069884	98	28,29,30,27	2025-06-03 20:58:31
2	3	28	99	1	1	0	1	2025-06-01 19:39:54.936647	105	26,27	2025-06-03 22:28:54
3	4	27	112	6	0	0	3	2025-06-01 20:14:09.885096	63	26,29,28,30	2025-06-01 18:46:31
4	5	29	[null]	0	0	0	5	2025-06-01 20:54:56.630896	6	[null]	2025-06-01 18:54:56
5	6	30	133	2	0	0	4	2025-06-04 00:30:06.497308	20	28,26	2025-06-03 22:30:06

## Intercambios:

	<b>id [PK] integer</b>	<b>id_usuario bigint</b>	<b>id_digimon_ofrecido bigint</b>	<b>id_digimon_buscado bigint</b>	<b>realizado boolean</b>	<b>created_at timestamp without time zone</b>	<b>updated_at timestamp without time zone</b>	<b>id_digimon_recibido integer</b>
1	39	26	106	7	true	2025-06-05 10:40:34	2025-06-05 15:37:34	139
2	42	27	95	17	false	2025-06-05 11:19:01	2025-06-05 11:19:01	[null]
3	43	28	98	2	false	2025-06-05 14:51:41	2025-06-05 14:51:41	[null]
4	44	30	121	8	false	2025-06-05 14:56:38	2025-06-05 14:56:38	[null]

## 4. Implementación

### 4.1 Configuración de los distintos subsistemas( realizado en MAC)

\*Para una guía más profunda, diríjase a **Bibliografía**

#### Instalación y configuración inicial (local)

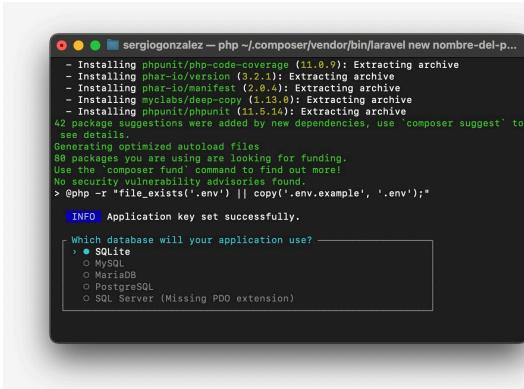
Para el desarrollo local de **Applimon Connect**, se utilizó la última versión de **Laravel** y **PostgreSQL** como base de datos principal. A continuación, se detallan los pasos seguidos para configurar el entorno de desarrollo:

#### Requisitos previos instalados:

- **Composer**: gestor de dependencias para PHP.
- **Laravel CLI**
- **PostgreSQL** (v14 o superior)
- **Node.js y NPM**: para compilar assets con Laravel Mix (animaciones, CSS, JS).
- **XAMPP o Laravel Valet**: como entorno de servidor local.
- **DBeaver o pgAdmin**: para la gestión de la base de datos.

#### Instalación del proyecto:

```
composer create-project laravel/laravel AppDigimon
cd AppDigimon
```



```
sergiogonzalez — php ~./composer/vendor/bin/laravel new nombre-del-p...
- Installing phpunit/php-code-coverage (11.0.0): Extracting archive
- Installing phar-io/version (3.2.1): Extracting archive
- Installing phar-io/manifest (2.0.4): Extracting archive
- Installing myclabs/deep-copy (1.13.0): Extracting archive
- Installing phpunit/phpunit (11.5.14): Extracting archive
47 packages and 0 dependency suggestions were added by new dependencies, use "composer suggest" to see details.
Generating optimized autoload files
88 packages you are using are looking for funding.
Use the "composer fund" command to find out more!
No security vulnerabilities advisories found.
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
INFO Application key set successfully.
Which database will your application use?
> SQLite
  ○ MySQL
  ○ MariaDB
  ○ PostgreSQL
  ○ SQL Server (Missing PDO extension)
```

#### Inicio del servidor del proyecto laravel para acceder a él(Mac):

```
sergiogonzalez@MacBook-Pro-de-Sergio ~ % cd /Users/sergiogonzalez/AppDigimon
←Accedemos a la carpeta del proyecto
sergiogonzalez@MacBook-Pro-de-Sergio AppDigimon % php artisan serve ←Iniciamos el servidor
```

#### Configuración del entorno (.env):

Se edita el archivo .env para conectar correctamente con la base de datos local PostgreSQL:

```
DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=AppDigimon
DB_USERNAME=postgres
DB_PASSWORD=nueva_contraseña
```

## Creación de la base de datos:

Se crean las tablas con los modelos definidos en Laravel, para ello usamos consultas(Querys en Postgre), ejemplo:

```
CREATE TABLE intercambios (
    id SERIAL PRIMARY KEY,
    digimon_id INTEGER NOT NULL REFERENCES digimon(id) ON DELETE CASCADE,
    lista_digimon_id INTEGER NOT NULL REFERENCES lista_digimon(id) ON DELETE CASCADE,
    realizado BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

## Datos multimedia (Digimon):

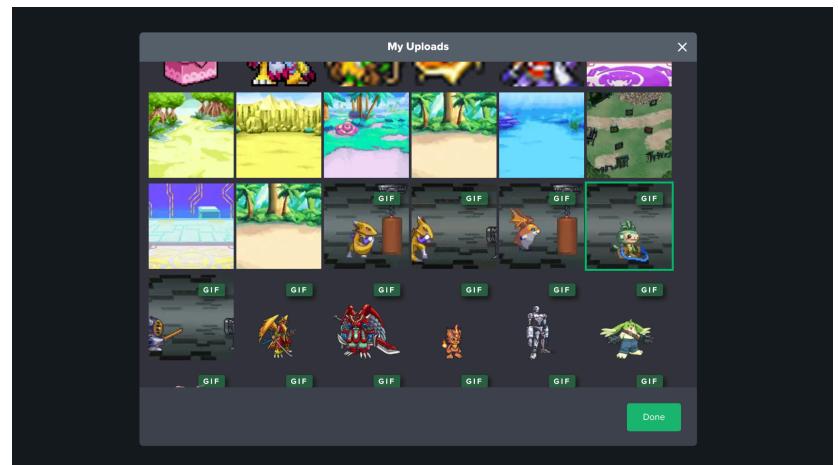
- La **banda sonora** se obtienen desde **YouTube**



- Las **imágenes** están alojadas en **Imgur**(las animaciones fueron previamente retocadas en Piskel), y sus URLs se almacenan directamente en la base de datos ListaDigimon en el dato **videogif**.



Piskel



Imgur

## 4.2 Integración de las herramientas externas

### Sistemas externos conectados:

- **YouTube API (opcional):** para cargar música de fondo en las vistas.
- **Imgur API:** si se desea gestionar las animaciones específicas de Digimon.

### Pruebas iniciales:

Nos crearemos usuarios de ejemplos, con los que:

- Capturaremos varios Digimon.
- Simularemos combates.
- Comprobaremos su crecimiento y desarrollo.
- Pondremos Digimon en intercambios y torneos para que otros usuarios puedan probarlo..

## 4.3 Despliegue del sistema

### Función de registro:

Desde la primera pestaña, el usuario puede completar el formulario de registro. Al enviarlo, los datos son procesados por el controlador, el cual:

1. Verifica si el correo ya se encuentra registrado.
2. Te da a elegir entre dos Digimon iniciales.
3. Crea una nueva cuenta de usuario.

### Función de login y logout:

Para acceder al sistema, el usuario debe iniciar sesión, donde se verifican:

- Si el correo electrónico está registrado.
- Si la combinación email/contraseña es correcta.

En caso afirmativo, se inicia una **sesión segura de usuario** y se redirige al menú principal.

La función de logout(accedida por el botón de cerrar sesión del menú principal) redirige al usuario a la página de inicio,cerrando su sesión.

## Funciones principales del sistema:

### 1. Cuidado del Digimon

Permite mantener el estado óptimo del Digimon activo. Las estadísticas que se gestionan incluyen:

- Hambre
- Higiene
- Salud
- Caca

Estas bajan de forma progresiva con el tiempo (por eventos programados) y pueden mejorarse usando **Bits(Dinero)** que se gana en los combates. Si estas estadísticas son bajas, afectarán negativamente el rendimiento en batalla.

### 2. Digidex

Muestra un **registro visual e interactivo** de todos los Digimon que el usuario ha obtenido (capturados, evolucionados, intercambiados, etc.). También permite ver los Digimon que existen, aunque aún no hayan sido obtenidos (como la Pokédex de Pokémon, pero con Digimon).

### 3. Caja de Digimon

En esta vista se muestran **todos los Digimon capturados**. El usuario puede:

- Ver estadísticas detalladas (niveles, estadísticas, evolución actual, etc.).
- Cambiar cuál es su Digimon activo para combate y entrenamiento.

### 4. Entrenamiento

Permite mejorar las estadísticas del Digimon activo:

- Vida
- Ataque
- Defensa

Cada evolución/involución concede **un número limitado de sesiones de entrenamiento**, que deben usarse sabiamente.

## 5. Intercambio entre usuarios

Sistema de **intercambio** de Digimon entre jugadores.

- Los usuarios pueden **depositar Digimon** y especificar qué tipo de Digimon buscan a cambio.

El sistema se basa en sistema de oferta y demanda, el cuál puede cambiar/mejorar en el futuro.

## 6. Vamos de aventura

Permite combatir contra **Digimon salvajes** seleccionando el nivel de dificultad. Si el usuario gana:

- Obtiene **Bits(Dinero)** y **experiencia**.
- Antes de derrotar al Digimon salvaje, existe una **probabilidad** de capturar al Digimon, ganando el combate, pero sin ganar Bits o experiencia.

Si pierde, no gana nada.

## 7. Combate online

Sistema de combate **jugador vs jugador asincrónico**:

- El jugador puede **depositar un Digimon para que lo defienda** pasivamente.
- También puede **buscar y desafiar activamente** a otros jugadores.
- Las victorias y derrotas afectan a una **clasificación online de puntos (ranking)**.

Este sistema implementa un sistema de matchmaking simple según rangos.

## 8. Evolución / Involución

Cada Digimon tiene **líneas evolutivas específicas**. El usuario puede:

- **Evolucionar** su Digimon (a uno más fuerte), cumpliendo ciertas condiciones: nivel mínimo, llaves conseguidas(al evolucionar/involucionar a ciertos Digimon) etc.
- **Involucionar** a versiones anteriores (más débiles) para obtener **bonificaciones de entrenamiento adicionales**.

Involucionar **no tiene restricciones** y se puede hacer libremente.

## 5. Pruebas

En esta sección se presentan una serie de pruebas funcionales realizadas al sistema **Applimon Connect** para verificar el correcto funcionamiento de sus principales módulos. También se intentaron forzar errores o comportamientos inesperados para comprobar la robustez de las validaciones. La mayoría de los errores se controlan al deshabilitar los botones.

### 5.1 Pruebas realizadas

#### 1. Comprobación del sistema de registro y login

##### Escenario 1: Registro con correo ya existente

- Intentamos registrarnos con un correo que ya está en uso.
- Resultado esperado: Se muestra un mensaje de error indicando que el correo ya está registrado.

The diagram illustrates the process of attempting to register with an existing email address. It consists of two screenshots of the 'Crear Cuenta' (Create Account) form. A large green arrow points from the left screenshot to the right one. The left screenshot shows a successful registration attempt with the following input fields: Nombre de usuario ('Gero'), Correo electrónico ('Sergiogm99@hotmail.com'), and Contraseña ('.....'). The right screenshot shows an error message: 'The correousuario has already been taken.' Below the message, the same input fields are shown, with the email field now containing '.....'. The 'Crear Cuenta' button is visible at the bottom of both forms.

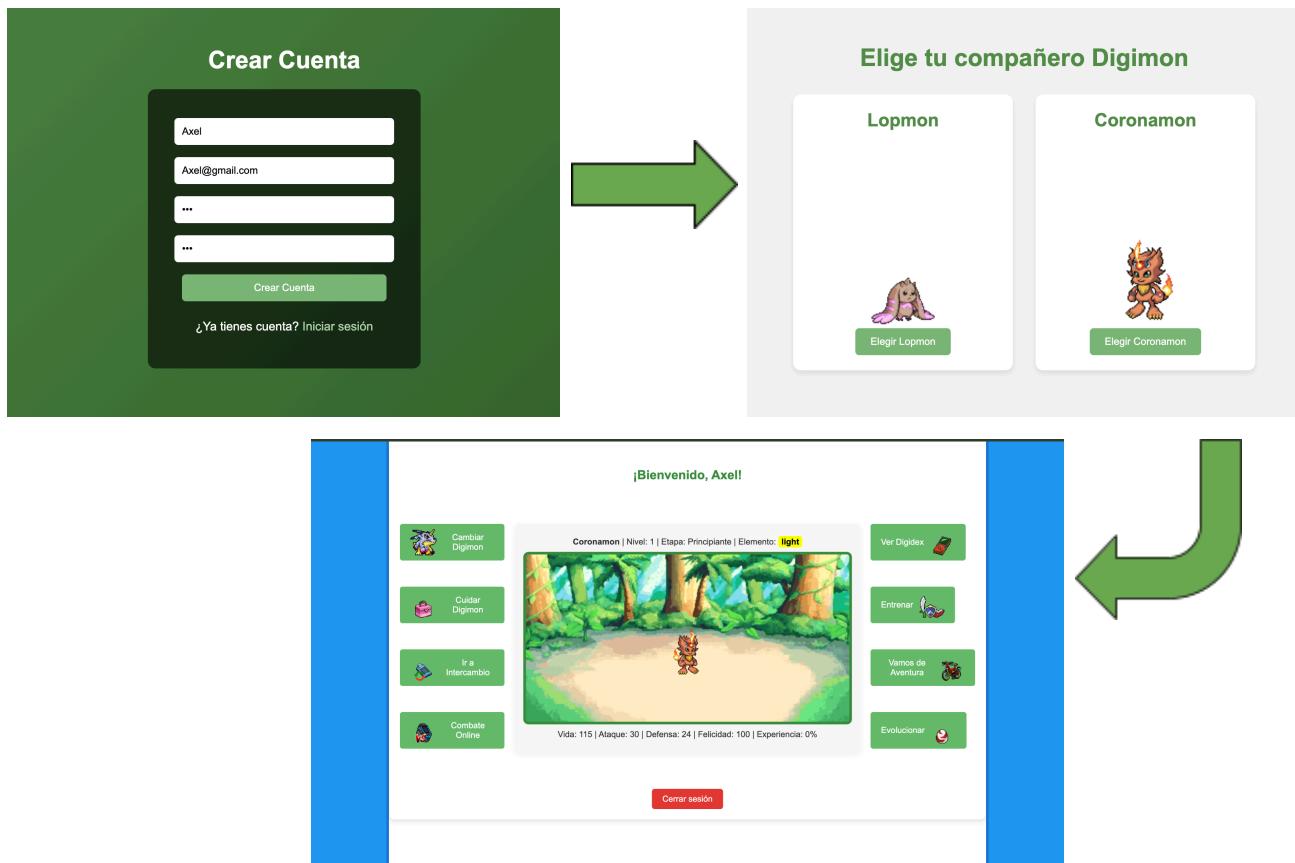
##### Escenario 2: Registro con correo no válido

- Introducimos un email con formato incorrecto (ej: Axelgmail.com).
- Resultado esperado: El formulario no se envía y se muestra un mensaje de validación.

The diagram illustrates the process of attempting to register with an invalid email address. It consists of two screenshots of the 'Crear Cuenta' (Create Account) form. A large green arrow points from the left screenshot to the right one. The left screenshot shows an invalid email entry ('Axelgmail.com') in the 'Correo electrónico' field. The right screenshot shows a validation error message: 'Incluye un signo "@" en la dirección de correo electrónico. La dirección "Axelgmail.com" no incluye el signo "@".' The email field now contains 'Axelgmail.com'. The other input fields ('Nombre de usuario' and 'Contraseña') remain the same as in the first screenshot. The 'Crear Cuenta' button is visible at the bottom of both forms.

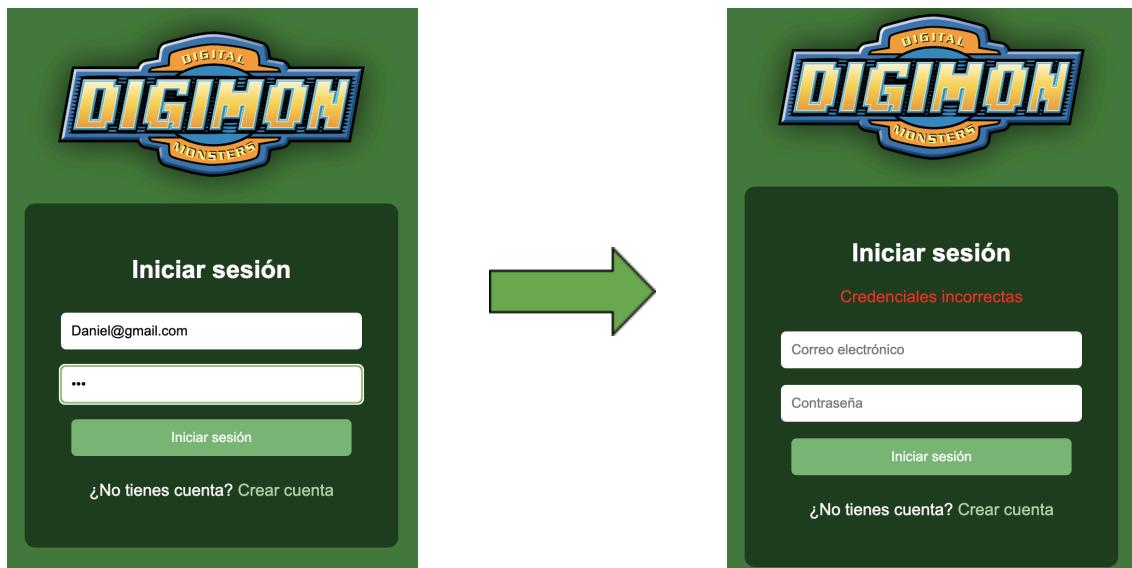
### Escenario 3: Registro exitoso

- Rellenamos el formulario correctamente.
- El sistema crea el usuario, pasamos a la elección del primer compañero y tras elegirlo, ya estamos en la página principal.



### Escenario 4: Intento de login sin confirmar correo

- Resultado esperado: El sistema no permite el acceso y redirige al login.



## Escenario 5: Login correcto tras confirmar correo

- Resultado esperado: El sistema inicia sesión y redirige al menú principal.



## 2. Cuidado del Digimon

**Escenario:** Accedemos al módulo de cuidado.

- Mostramos los valores actuales de hambre, salud, higiene y caca.
- Usamos acciones para mejorar dichas estadísticas (gastando Bits).
- Se bloquean los botones una vez alcanzado el límite



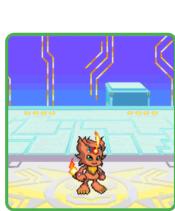
### 3. Entrenamiento

**Escenario:** Digimon ha evolucionado recientemente, tiene puntos de entrenamiento disponibles.

- Entrenamos vida, ataque o defensa(en este caso vida) y se desactivarán los botones hasta que acabe el entrenamiento.
- Se reduce el contador de entrenamientos disponibles.

#### Entrenamiento de Coronamon

| Nivel Actual: 1 | | Vida Actual: 110.4 Ataque Actual: 27.7 Defensa Actual: 21.7 | | Entrenamientos restantes: 5 |



Entrenamiento vigoroso con Monmon 🌋

Entrenamiento Ofensivo con Patamon 🔥

Entrenamiento Defensivo con Kotemon 🏰

#### Entrenamiento de Coronamon

| Nivel Actual: 1 | | Vida Actual: 110.4 Ataque Actual: 27.7 Defensa Actual: 21.7 | | Entrenamientos restantes: 5 |



#### Entrenamiento de Coronamon

| Nivel Actual: 1 | | Vida Actual: 111.4 Ataque Actual: 27.7 Defensa Actual: 21.7 | | Entrenamientos restantes: 4 |



Entrenamiento vigoroso con Monmon 🌋

Entrenamiento Ofensivo con Patamon 🔥

Entrenamiento Defensivo con Kotemon 🏰

Entrenamiento Ofensivo con Patamon 🔥

Entrenamiento Defensivo con Kotemon 🏰

### 4. Evolución e involución (si no se cumplen los requisitos, el botón se desactiva)

#### Evolución:

- Cumplimos con los requisitos(nivel 10) y evolucionamos a nuestro Digimon.
- Resultado: Nuevo sprite y estadísticas mejoradas.

Volver a Inicio | ¡Es momento de Digi-Evolucionar! | Ver Digidex

**Involuciones**

Sumon

**Digimon Actual**

Coronamon  
Etapa: Principiante

**Evoluciones**

Firamon

Stingmon

**Evolucionar**

Nivel suficiente: 10 / 10  
Necesitas las llaves: [LIGHT2] o [INSECT1] y [INSECT2]



Volver a Inicio | ¡Es momento de Digi-Evolucionar! | Ver Digidex

**Éxito: ¡Tu Digimon ha evolucionado!**

**Involuciones**

Sumon

**Digimon Actual**

Firamon  
Etapa: Campeón

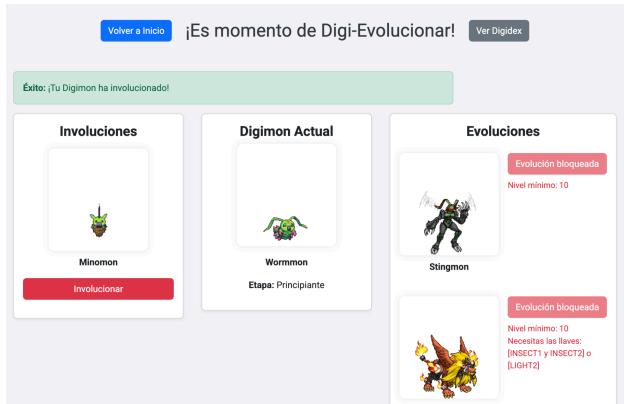
**Evoluciones**

Flaremon

**Evolución bloqueada**  
Nivel mínimo: 15

## Involución:

- Volvemos a una fase anterior.
- Resultado: estadísticas más bajas, pero se ganan puntos extra de entrenamiento(5 puntos por haber evolucionado y otros 5 por involucionar) y bonos ocultos.



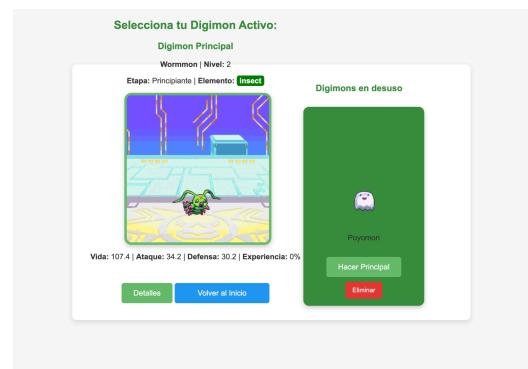
## 5. Combates contra Digimon salvajes(acciones instantáneas, no se puede apretar un botón durante otra acción )

**Escenario:** Seleccionamos dificultad fácil y peleamos.

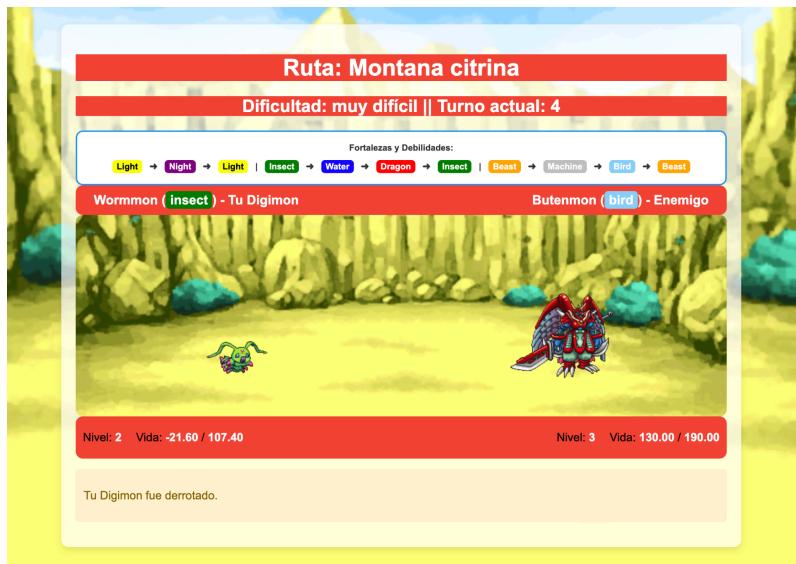
- Si ganamos: recibimos experiencia y Bits.



- Si capturamos: aumentamos el número de Digimon a nuestra disposición



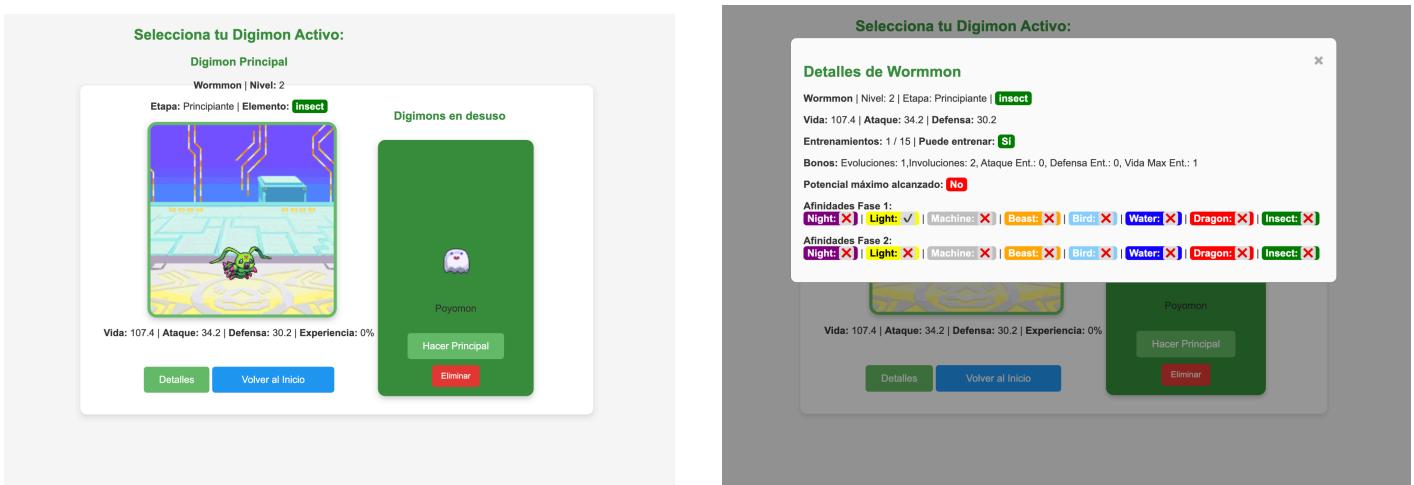
- Si perdemos: no recibimos recompensas (he cambiado la dificultad para forzar la derrota).



## 6. Caja

**Escenario:** Anteriormente hemos capturado un Digimon

- Comprobamos que se visualizan tanto nuestras capturas correctamente como los detalles de nuestro compañero activo. Podremos eliminarlos permanentemente si queremos, pero nunca al Digimon activo.



## 7. Intercambio entre usuarios

**Escenario:** Usuario A deposita un Digimon(Poyomon) y solicita uno específico(Lopmon). Usuario B visualiza la oferta y acepta.Si no se tiene al Digimon solicitado, no se da la opción del intercambio.

### Usuario A

**Zona de Intercambio**

Volver a Home Buscar Digimon para Intercambio Cancelar depósito

Poyomon

Aún no has recibido un Digimon.

Intercambio creado exitosamente.



### Usuario B

← Volver a intercambio Buscar Digimon para Intercambio

Buscar por nombre del Digimon Buscar

Intercambios Disponibles:

- Firamon ofrecido por el usuario #30 Intercambiar (Buscado: Dinobeemon)
- Poyomon ofrecido por el usuario #34 Intercambiar (Buscado: Lopmon)
- Guardromon ofrecido por el usuario #27 Intercambiar (Buscado: Kapurimon)
- Lopmon ofrecido por el usuario #28 Intercambiar (Buscado: Coronamon)



**Zona de Intercambio**

Volver a Home Buscar Digimon para Intercambio Recibir Digimon

Esperando que otro usuario complete el intercambio...

Intercambio Realizado Lopmon



← Volver a Intercambio Seleccióna uno de tus Digimon para intercambiar

Lopmon Nivel: 1 Etapa: Principiante Elemento: Night Intercambiar



## 8. Combate online

**Escenario:** Usuario A deja un Digimon en defensa, Usuario B lo desafía.

- El sistema evalúa ambos Digimon y determina el resultado según sus estadísticas.
- Puntos ganados o perdidos son registrados en el ranking(en este caso hemos ganado).



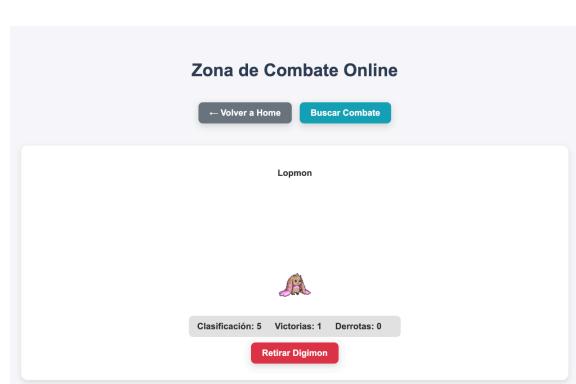
← Volver a Home Buscar Combate

Zona de Combate Online

Lopmon

Clasificación: 5 Victorias: 1 Derrotas: 0

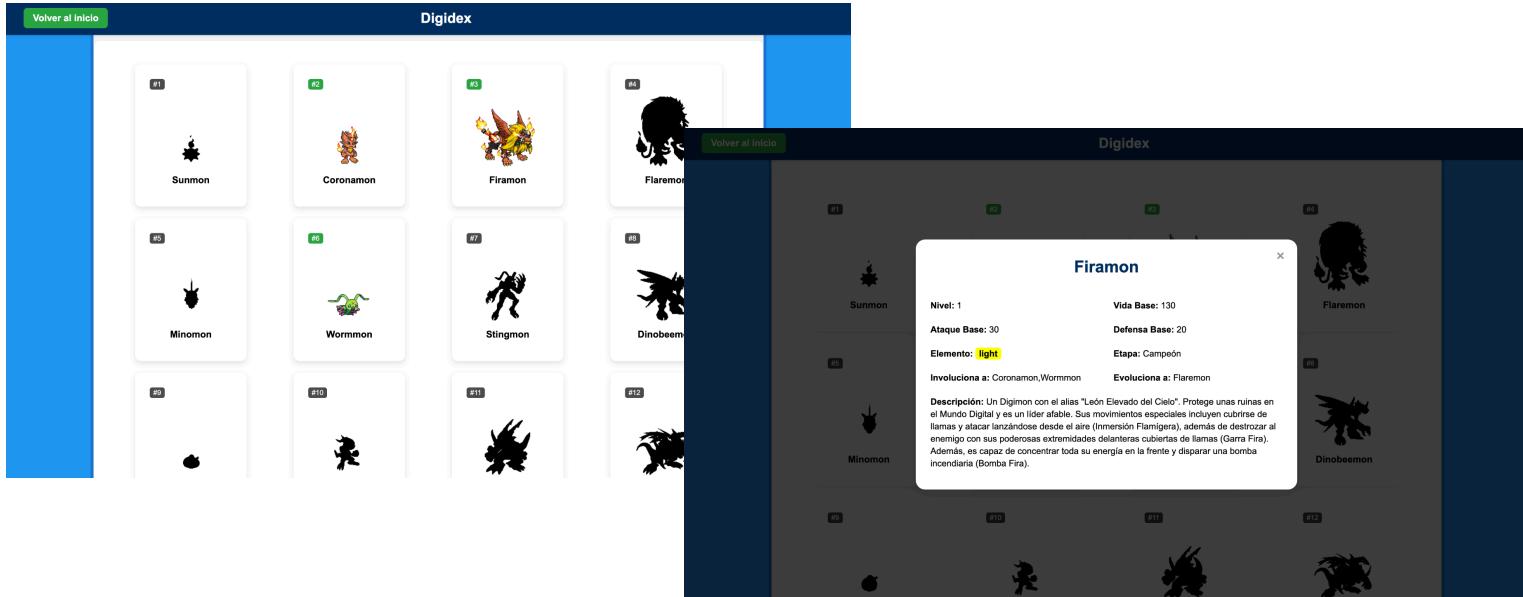
Retirar Digimon



## 9. Digidex

**Escenario:** Todo lo hecho anteriormente: Evolucionar, involucionar, intercambiar y capturar.

- El sistema controla todos los digimon que hayas tenido una vez, registrándolos y dando los datos básicos de aquellos registrados.



### 5.2 Resultados obtenidos

Prueba	Resultado
Registro/login/activación por correo	✓ Funciona correctamente con validaciones.
Asignación de Digimon inicial	✓ Sistema funcional, asignación exitosa.
Cuidado del Digimon	✓ Estadísticas modificables, afectan combates.
Entrenamiento	✓ Bonificaciones otorgadas tras evolución o involución.
Evolución / Involución	✓ Mecanismo funcionando según condiciones establecidas.
Combate salvaje y captura	✓ Resultados variables, lógica de victoria, experiencia y captura.
Caja	✓ Vemos que el Digimon que hemos capturado está aquí y también que podemos comprobar los detalles.
Intercambio entre usuarios	✓ Sistema básico funcional, verificación manual de condiciones.
Combate online y ranking	✓ Sistema asíncrono funcional, ranking actualizado tras combates.
Digidex	✓ Control total de los Digimon adquiridos alguna vez

## **6. Conclusiones**

### **6.1 Conclusiones finales**

El desarrollo de **Applimon Connect** ha sido una experiencia tanto intensa como divertida. Ha supuesto un verdadero reto técnico y organizativo, desde el diseño de la base de datos en PostgreSQL hasta la integración de funciones complejas, como la evolución, combates y el intercambio entre usuarios.

Gracias a elegir **Laravel** como framework principal, se pudo organizar el proyecto de forma limpia, estructurada y segura, facilitando la integración con herramientas como el encriptado de contraseñas. El uso de bases de datos externas como YouTube (para la banda sonora) e Imgur (para la gestión de imágenes y animaciones) aportó un toque multimedia al sistema que mejoró significativamente la experiencia del usuario.

A lo largo del proyecto también surgieron dificultades, como la generación de los Digimon o el control de la reducción de estadísticas (hambre, higiene, etc.). Otro punto que requirió mucho ajuste fue el sistema de combate y evolución, donde se buscó un balance justo entre progresión y jugabilidad.

Aunque el sistema solo está disponible de forma **local**, se ha desarrollado teniendo en cuenta su futura escalabilidad a producción. Este proyecto ha demostrado que, con una planificación adecuada y la elección correcta de tecnologías, es posible desarrollar un sistema complejo desde cero con recursos gratuitos y abiertos.

### **6.2 Desviaciones temporales**

Durante el desarrollo hubo algunas desviaciones tanto técnicas como de calendario:

- **Tiempo estimado vs. real:** El tiempo total se ajustó bastante a lo planeado (250 horas), pero ciertas tareas, como el sistema de evolución o el combate online, requirieron más trabajo del previsto inicialmente.
- **Complejidad de diseño:** Al tratarse de un sistema con múltiples módulos dependientes unos de otros, fue necesario revisar y reestructurar varias partes del código para garantizar un flujo lógico y sin errores.

A pesar de estas desviaciones, se logró cumplir con los objetivos fundamentales.

### **6.3 Posibles ampliaciones y modificaciones**

El sistema actual funciona correctamente en entorno local, pero hay una larga lista de mejoras previstas que podrían llevar el proyecto al siguiente nivel:

- **Publicación en servidor con dominio propio.**
- **Sistema de recuperación de contraseña por correo.**
- **Creación de criaturas originales para reemplazar a los Digimon en caso de problemas con derechos de autor.**
- **Ampliación de las líneas evolutivas.**
- **Balanceo de las estadísticas y ataques de los Digimon.**
- **Incorporación de eventos temporales o logros (como nuevas zonas a la que acceder).**
- **Sistema de notificaciones internas (ej: avisos de eventos ).**
- **Interfaz gráfica mejorada con animaciones más fluidas entre pantallas.**
- **Posibilidad de combate en tiempo real.**
- **Sistema de logros o misiones diarias.**

Estas ideas permitirían ampliar y mejorar la experiencia de usuario, y también facilitarían la escalabilidad del sistema.

### **6.4 Valoración personal**

A nivel personal, **Applimon Connect** ha sido un proyecto muy especial. Me ha permitido aplicar de forma práctica conocimientos de bases de datos, manejo de usuarios, sistemas de lógica compleja y diseño de interfaces. También ha sido un desafío creativo al tratar de construir un “mundo” propio con Digimon como base conceptual.

Aunque queda un largo camino por recorrer, me siento orgulloso de haber logrado desarrollar un sistema funcional, jugable y relativamente estable en un plazo razonable. Aunque cada módulo puede llegar a mejorar en el futuro, esto es parte del aprendizaje continuo que tendré que realizar..

Para el tiempo y los recursos usados, considero que he hecho un buen trabajo, el cual cumple con los requisitos propuestos, y si tuviera que valorar el proyecto, me pondría un **8,5/10**, ya que hay muchas cosas por pulir.

## 7. Coste Real del Proyecto

### 7.1 Coste material

Recurso	Herramienta / Servicio	Costo estimado
Hosting (futuro)	InfinityFree	0 €
Dominio(futuro)	Subdominio gratuito	0 €
Framework	Laravel (open-source)	0 €
Entorno local	XAMPP / PHP / Composer	0 €
Base de datos principal	PostgreSQL	0 €
Bases de datos secundarias	YouTube, Imgur (uso parcial)	0 €

**Total coste material: 0 €**

### 7.2 Coste personal

- Tiempo invertido: **250 horas**
- Valor hora estimado: **12 €/hora**

**Total coste personal: 4200 €**

### 7.3 Resumen coste-beneficio

El proyecto ha sido desarrollado íntegramente con herramientas gratuitas, por lo que el único coste ha sido el tiempo de desarrollo. Esto lo convierte en un proyecto económicamente viable, que podría ponerse en producción con una inversión mínima.

Además, se estima un coste de **50 €/año** en caso de desplegar el sistema en servidor (SSL, base de datos, mantenimiento).

## **8. Bibliografía**

### **Herramientas y tecnologías**

- Laravel: <https://laravel.com/>
- PostgreSQL: <https://www.postgresql.org/>
- Mailtrap: <https://mailtrap.io/>
- Composer: <https://getcomposer.org/>
- XAMPP: <https://www.apachefriends.org/>

### **Recursos utilizados**

- Animaciones e imágenes: <https://imgur.com/>
- Datos sobre Digimon: [https://wikimon.net/Main\\_Page](https://wikimon.net/Main_Page)
- Música: <https://www.youtube.com/watch?v=QBv0YjElMI&list=PL83ABBC8D3214BA9C>
- Fondos y otras imágenes: <https://amiaiba.tumblr.com/>
- Videos de descarga e instalación:
  - Postgre: [https://www.youtube.com/watch?v=gEJcMrk3E-Q&list=PLgqdACsQ8US2DCzQVrdZDZCTtTFHMY0as&ab\\_channel=VitoDev](https://www.youtube.com/watch?v=gEJcMrk3E-Q&list=PLgqdACsQ8US2DCzQVrdZDZCTtTFHMY0as&ab_channel=VitoDev)
  - Laravel: [https://www.youtube.com/watch?v=3e1lsZJuYAw&ab\\_channel=CodersFree](https://www.youtube.com/watch?v=3e1lsZJuYAw&ab_channel=CodersFree)
  - Exportar/Importar bases de datos: <https://www.youtube.com/watch?v=icEvkylXqug>

## 9. Glosario

### **Digimon:**

O “Digital Monster”, son criaturas digitales con características únicas que los usuarios pueden colecciónar, cuidar, entrenar, evolucionar o involucionar. Son el eje central del sistema de juego, utilizados tanto en combates como en actividades interactivas dentro de la aplicación.

### **Digi-dex:**

Enciclopedia virtual donde se almacenan todos los Digimon que un usuario ha conseguido a lo largo del juego. Incluye datos como el nombre, estadísticas, línea evolutiva, estado y descripción.

### **Laravel:**

Framework PHP de código abierto que facilita el desarrollo web mediante una estructura elegante y legible. Laravel incorpora herramientas para enrutamiento, autenticación, pruebas, seguridad y ORM, lo que agiliza la creación de aplicaciones complejas como Applimon Connect.

### **ORM (Object-Relational Mapping):**

Técnica que permite manipular y consultar bases de datos usando objetos del lenguaje de programación, en lugar de consultas SQL directas. En Laravel, se implementa a través de Eloquent, facilitando el acceso a modelos como Usuario, Digimon, Combate, etc.

### **CSRF (Cross-Site Request Forgery) / XSS (Cross-Site Scripting):**

Dos vulnerabilidades comunes en aplicaciones web:

- **CSRF:** Ataques donde un usuario autenticado es inducido a ejecutar acciones no deseadas sin saberlo.
- **XSS:** Inyecciones de código malicioso (generalmente JavaScript) en páginas vistas por otros usuarios. Laravel mitiga ambos riesgos por defecto mediante el uso de tokens de autenticación y escape de datos.

### **Responsive:**

Diseño web que se adapta automáticamente a distintos tamaños de pantalla (móviles, tablets, ordenadores). Se basa en tecnologías como CSS Flexbox, media queries y frameworks como Bootstrap o Tailwind, garantizando una experiencia de usuario óptima en todos los dispositivos.

### **Combate online asíncrono:**

Sistema de enfrentamientos multijugador donde un usuario puede dejar a su Digimon activo como defensor pasivo, mientras otros usuarios pueden desafiarlo en cualquier momento. Los resultados afectan a la clasificación del usuario, pero no requieren que ambos estén conectados simultáneamente.

### **Backend:**

Parte del sistema que maneja la lógica de negocio, bases de datos, autenticaciones y comunicaciones del servidor. En Applimon Connect, el backend está desarrollado en Laravel y maneja tareas como registrar usuarios, procesar combates, gestionar Digimon, etc.

**Frontend:**

Parte visible de la aplicación con la que interactúan los usuarios. Incluye interfaces gráficas, formularios, animaciones y navegación. El frontend de Applimon Connect combina HTML, CSS y JavaScript, aprovechando Blade (motor de plantillas de Laravel) y bibliotecas externas.

**DMZ (Demilitarized Zone):**

Zona intermedia de red entre el servidor público (como el que aloja el frontend) y la red interna (donde se encuentran la base de datos y servicios protegidos). Se utiliza para aumentar la seguridad, evitando accesos directos al núcleo de la infraestructura desde el exterior.

**Token:**

Es una cadena única generada por el sistema que se utiliza para verificar la identidad del usuario o la validez de ciertas acciones en la aplicación. Se pueden usar para proteger los formularios web de ataques maliciosos o para recuperar contraseñas enviando un token único al correo.

**RGPD y LOPDGDD:**

Son leyes que regulan cómo deben tratarse los datos personales de los usuarios en Europa y España, exigiendo transparencia, consentimiento y medidas de protección. Son fundamentales para cualquier aplicación que almacene información de personas.

**10. Anexo**

Aquí describiré un problema que he tenido con la banda sonora, que en futuras versiones del proyecto se solucionará:

Como ya se ha dicho a lo largo de este documento, Applimon Connect cuenta con una banda sonora(prestada de la franquicia de Digimon) que otorga una mejor inmersión al usuario en la aplicación. Cada módulo tiene su propia melodía, para así diferenciar una vista de otra.

El problema es a la hora de cómo se pasa de una vista a otra, ya que cuando, por ejemplo, vamos de la página principal a “Cuidar”, usamos una función, que llama a la nueva vista y consigue que la música asociada se reproduzca y la anterior cese. Hasta aquí todo ha ido como debería, el problema viene cuando en dicha página queremos refrescar la página para actualizar los datos o porque hemos hecho uso de una función(por ejemplo, pulsando un botón); en este caso llamamos a la misma vista, lo que produce que la canción se reinicie cuando no es la intención.

Aunque no supone un daño al código, ni causa problemas con el usuario, sí que hace que el usuario pierda un poco de esa “inmersión” que queremos conseguir en nuestra aplicación. Por ello, para versiones futuras se propondrán soluciones para abarcar este dilema.