Azkernia

Introducción

Azkernia es un ambicioso proyecto web diseñado como un juego interactivo que combina entretenimiento y aprendizaje, permitiendo a los jugadores defender un reino mediante misiones que involucran el uso de consultas SQL. Este documento proporciona una descripción extremadamente detallada del proyecto, incluyendo su estructura, funcionalidades, y el propósito de cada archivo involucrado.

Estructura del Proyecto

El proyecto está organizado en varias carpetas principales que facilitan la separación de responsabilidades y la gestión de los recursos. Cada carpeta tiene un propósito específico y contiene los archivos necesarios para cumplir dicho propósito.

- **css/**: Contiene las hojas de estilo que definen la apariencia visual de la aplicación. Estas hojas permiten un diseño atractivo y consistente a lo largo del proyecto.

- **images/**: Almacena las imágenes utilizadas en la interfaz gráfica, tales como fondos, íconos y otros elementos visuales que enriquecen la experiencia del usuario.

- **js/**: Incluye los archivos JavaScript encargados de manejar la lógica y las interacciones en el lado del cliente, permitiendo una experiencia interactiva y dinámica.

- **views/**: Esta carpeta contiene las vistas principales de la aplicación, implementadas mediante Laravel. Aquí se define la presentación de cada página del proyecto.

- **routes/**: Define las rutas disponibles en la aplicación, conectando las solicitudes del usuario con los controladores que las procesan.

- **controllers/**: Contiene los controladores que gestionan la lógica del servidor, procesan las solicitudes HTTP y retornan las respuestas adecuadas.

- **database/**: Alberga las migraciones y configuraciones relacionadas con la base de datos. Aquí se define la estructura de las tablas utilizadas.

- **tests/**: Incluye pruebas automatizadas para verificar la funcionalidad y confiabilidad del proyecto.

Archivos Principales y Su Funcionalidad

A continuación, se detallan los archivos clave del proyecto, describiendo en profundidad su propósito y la lógica detrás de su implementación.

1. `routes/web.php`

Este archivo define todas las rutas principales del proyecto, actuando como un mapa que conecta las URLs con los métodos correspondientes en los controladores.

Propósito

Facilitar la navegación y garantizar que las solicitudes sean gestionadas por la lógica adecuada.

Ejemplo de Ruta

Route::get('/', [WelcomeController::class, 'index'])->name('welcome');

- **Descripción:** Muestra la página de bienvenida.
- **Ventajas:** Centraliza la gestión de rutas, simplificando futuras modificaciones.

2. `app/Http/Controllers/WelcomeController.php`

Este controlador es responsable de manejar la lógica asociada a la página inicial.

Métodos Clave

- **`index`:** Renderiza la vista `welcome.blade.php`.
    - **Propósito:** Mostrar la página inicial del juego y ofrecer opciones para iniciar partidas o leer instrucciones.

3. `resources/views/welcome.blade.php`

Este archivo Blade actúa como la página de inicio del proyecto.

Funcionalidades Principales

- Contiene botones para:
    - "Iniciar Juego": Lleva al jugador al área principal del juego.

Diseño Visual

Incluye un diseño responsivo para garantizar una experiencia óptima en dispositivos de diferentes tamaños.

4. `resources/views/principal.blade.php`

Esta vista constituye el núcleo del juego. Es donde los jugadores interactúan directamente con el sistema.

Elementos Clave

- **Formulario:** Permite a los jugadores ingresar consultas SQL.
- **Sección de Misiones:** Muestra las misiones activas con sus detalles y requisitos.
- **Resultados:** Muestra el feedback en tiempo real de las consultas ejecutadas.
- **Botón de Regreso:** Facilita la navegación de vuelta a la página principal.

5. `public/css/principal.css`

Define los estilos aplicados a las vistas principales, garantizando una estética coherente y atractiva.

Estilos Notables

- **Cuerpo:**

```
body {
    background-image: url('/images/1.jpg');
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-position: center;
}
```

- **Botones Interactivos:**

```
.btn-warning {
    background-color: #d4a373;
    color: #3e3b36;
    padding: 18px 35px;
    border-radius: 8px;
}
```

- Proporcionan una apariencia elegante y funcional.

6. `public/js/gameLogic.js`

Este archivo gestiona la lógica del lado del cliente, siendo esencial para las interacciones dinámicas del juego.

Responsabilidades

- **Envía Consultas:** Conecta al cliente con el servidor para procesar las misiones SQL.
- **Procesa Respuestas:** Muestra resultados de las consultas en tiempo real.

7. `database/migrations`

Define la estructura de las tablas de la base de datos necesarias para el funcionamiento del juego.

Tablas Clave

1. **resources:** Almacena los recursos del reino.
    - Campos: id, kingdom_id, gears, steam, etc.
2. **troops**: Contiene información sobre las tropas del jugador.
    - Campos: id, type, quantity.

8. `public/images/`

Incluye imágenes utilizadas para enriquecer la experiencia visual del juego.

Ejemplo

- **1.jpg:** Fondo principal utilizado en las vistas principales.

9. `tests/`

Contiene pruebas unitarias diseñadas para verificar que cada funcionalidad del proyecto opere correctamente.

Ejemplo

- Prueba de que las consultas SQL devuelvan los resultados esperados.

10. `composer.json`

Archivo que gestiona las dependencias del proyecto Laravel.

Propósito

- Garantiza que todas las bibliotecas necesarias estén instaladas y actualizadas.

Funcionalidades Clave

1. Autenticación de Usuarios

- Permite el acceso exclusivo a usuarios registrados, protegiendo el contenido del juego.

2. Sistema de Misiones

- Genera misiones dinámicas que involucran consultas SQL.
- Proporciona retroalimentación inmediata sobre el progreso.

3. Estadísticas del Jugador

- Registra las victorias y derrotas.

4. Interfaz de Usuario

- Diseño atractivo y responsivo.
- Elementos interactivos y fáciles de usar.

5. Gestión de Base de Datos

- Migraciones que simplifican la configuración inicial del proyecto.
- Seeds para poblar la base de datos con datos iniciales.

Diagrama de casos de uso

Los **casos de uso** en una página web describen cómo los usuarios interactúan con el sitio para lograr objetivos específicos. Estos casos de uso se centran en las acciones que los usuarios realizan y cómo el sistema responde a esas acciones.

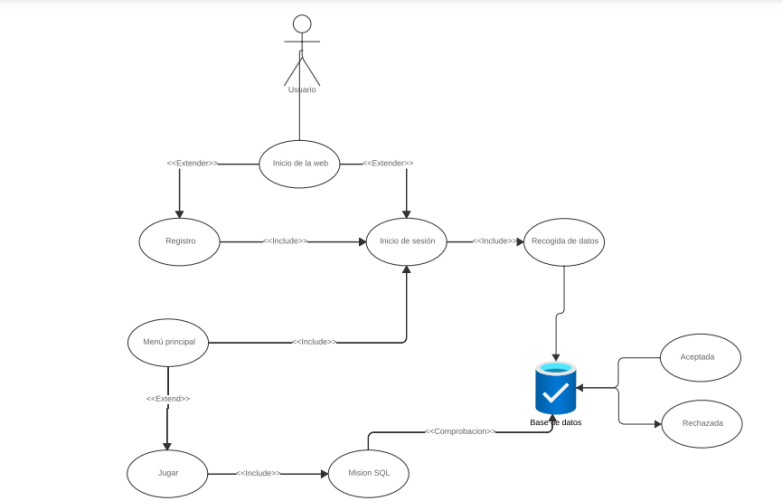

Diagrama relacional de la base de datos

El diagrama de entidad-relacional es una representación visual de la estructura de una base de datos. Muestra las entidades (objetos o conceptos) y las relaciones entre ellas.