Explorax Backend - Documentación y Decisiones de Diseño

Introducción

Explorax Backend es una API REST desarrollada en Go utilizando el framework Gin. La API proporciona funcionalidades de autenticación, administración de misiones y seguimiento del progreso de los usuarios.

Arquitectura de la Solución

El backend sigue una arquitectura modular para mejorar la escalabilidad y el mantenimiento. La organización de carpetas es la siguiente:

```
/internal
├── database       # Lógica de acceso a la base de datos (MongoDB)
├── handlers       # Controladores HTTP para las rutas
├── middleware     # Middleware para autenticación JWT
├── models         # Definición de estructuras de datos
├── utils          # Funcionalidades auxiliares como generación de JWT
├── docs           # Documentación generada por Swagger
/cmd
├── main.go        # Punto de entrada de la aplicación
```

Modelos de Datos

La API utiliza MongoDB como base de datos principal. Los modelos más importantes incluyen:

Modelo de Usuario

```
type User struct {
    ID           primitive.ObjectID `bson:"_id,omitempty"`
    Username     string             `bson:"username"`
    Email        string             `bson:"email"`
    PasswordHash string             `bson:"password_hash"`
    CreatedAt    time.Time          `bson:"created_at"`
}
```

Modelo de Misión

```
type Mission struct {
    ID          primitive.ObjectID `bson:"_id,omitempty"`
    Title       string             `bson:"title"`
    Description string             `bson:"description"`
    CreatedAt   time.Time          `bson:"created_at"`
}
```

Modelo de Progreso de Misión

```
type MissionProgress struct {
    ID        primitive.ObjectID `bson:"_id,omitempty"`
    UserID    primitive.ObjectID `bson:"user_id"`
    MissionID primitive.ObjectID `bson:"mission_id"`
    Status    string             `bson:"status"`
    StartDate time.Time          `bson:"start_date"`
}
```

Decisiones de Diseño

1. Uso de Gin como framework: Se eligió Gin por su alto rendimiento y facilidad de uso.
2. Autenticación con JWT: Para garantizar la seguridad de los endpoints protegidos, los usuarios deben autenticarse mediante un token JWT.
3. Persistencia en MongoDB: Se eligió MongoDB debido a su flexibilidad en el almacenamiento de datos estructurados y no estructurados.
4. Swagger para documentación: Se utilizó Swagger para facilitar la exploración de la API y permitir pruebas interactivas.

Además, se validó generar los id's como ObjectD para que se almacenara correctamente en la base de datos y no hubiera problema al hacer las peticiones.

Uso de JWT en Swagger

Para hacer pruebas en Swagger, es necesario incluir el token JWT en el encabezado Authorization para los endpoints que requieren autenticación . El formato correcto es:

Bearer <token>

Para poder realizar el login, primero se debe registrar en /auth/register

Pasos:

1. Realizar una petición POST a /auth/login con un JSON que contenga email y password.
2. Copiar el token generado en la respuesta.
3. En Swagger, hacer clic en "Authorize" y pegar el token con el formato Bearer <token>.
4. Ahora puedes probar los endpoints protegidos.

Rutas de la API

Autenticación

- POST /auth/register - Registro de un nuevo usuario.
- POST /auth/login - Inicio de sesión y generación de JWT.

Misiones

- GET /missions/all - Obtener todas las misiones.
- POST /missions/start - Iniciar una misión.
- POST /missions/complete - Completar una misión.
- GET /missions/progress - Ver progreso de misiones.
- GET /missions/active - Ver misiones activas.
- GET /missions/completed - Ver misiones completadas.
- GET /missions/statistics - Obtener estadísticas del usuario.
- GET /missions/leaderboard - Ver el ranking de usuarios.
- GET /missions/overview - Obtener una visión general de las misiones.