# 🚀 Documentación Completa - Dashboard Yega

**Dashboard profesional para análisis y gestión del ecosistema Yega en GitHub**

[PHP Version](https://php.net) [MySQL](https://mysql.com) [Node.js](https://nodejs.org) [License](LICENSE)

## 📋 Tabla de Contenidos

1. [🔧 Requisitos del Sistema](#requisitos-del-sistema)
2. [⚡ Instalación Automatizada](#instalación-automatizada)
3. [🛠️ Instalación Manual](#instalación-manual)
4. [🔑 Configuración de GitHub API](#configuración-de-github-api)
5. [🗄️ Configuración de Base de Datos](#configuración-de-base-de-datos)
6. [⚙️ Variables de Entorno](#variables-de-entorno)
7. [🔄 Migraciones y Schema](#migraciones-y-schema)
8. [📖 Guía de Uso del Dashboard](#guía-de-uso-del-dashboard)
9. [🚨 Troubleshooting](#troubleshooting)
10. [🚀 Scripts de Deployment](#scripts-de-deployment)
11. [🔄 Mantenimiento y Actualizaciones](#mantenimiento-y-actualizaciones)
12. [📞 Soporte y Contacto](#soporte-y-contacto)

## 🔧 Requisitos del Sistema

### 📋 Requisitos Mínimos

| Componente | Versión Mínima | Recomendada | Notas |
| --- | --- | --- | --- |
| **PHP** | 8.1 | 8.2+ | Con extensiones requeridas |
| **MySQL** | 8.0 | 8.0.32+ | O MariaDB 10.6+ |
| **Composer** | 2.0 | 2.5+ | Gestor de dependencias PHP |
| **Node.js** | 16.0 | 18.0+ LTS | Para herramientas frontend |
| **NPM** | 8.0 | 9.0+ | O Yarn 1.22+ |
| **Git** | 2.25 | 2.40+ | Control de versiones |
| **Memoria RAM** | 2GB | 4GB+ | Para operaciones complejas |
| **Espacio en Disco** | 1GB | 5GB+ | Incluyendo logs y cache |

### 🔌 Extensiones PHP Requeridas

# Extensiones esenciales  
php-mysql # Conexión a MySQL/MariaDB  
php-pdo # PHP Data Objects  
php-mbstring # Manipulación de strings multibyte  
php-xml # Procesamiento XML  
php-curl # Cliente HTTP  
php-json # Procesamiento JSON  
php-tokenizer # Tokenización de código  
php-bcmath # Matemáticas de precisión arbitraria  
php-ctype # Funciones de tipo de carácter  
php-fileinfo # Información de archivos  
php-openssl # Funciones criptográficas  
php-zip # Manejo de archivos ZIP  
php-gd # Manipulación de imágenes  
php-intl # Internacionalización  
php-redis # Cache Redis (opcional)

### 🖥️ Sistemas Operativos Soportados

#### 🐧 Linux (Recomendado)

* **Ubuntu**: 20.04 LTS, 22.04 LTS
* **Debian**: 11 (Bullseye), 12 (Bookworm)
* **CentOS**: 8, 9
* **RHEL**: 8, 9
* **Amazon Linux**: 2

#### 🍎 macOS

* **macOS Big Sur**: 11.0+
* **macOS Monterey**: 12.0+
* **macOS Ventura**: 13.0+
* **Homebrew**: Para instalación de dependencias

#### 💻 Windows

* **Windows 10**: Build 19041+
* **Windows 11**: Todas las versiones
* **WSL2**: Altamente recomendado
* **Docker Desktop**: Para contenedores

### 🔍 Verificación de Requisitos

# Descargar script de verificación  
wget https://raw.githubusercontent.com/tu-usuario/yega-dashboard/main/code/setup/check-system.sh  
chmod +x check-system.sh  
./check-system.sh  
  
# O ejecutar manualmente  
php --version  
mysql --version  
composer --version  
node --version  
npm --version  
git --version

## ⚡ Instalación Automatizada

### 🚀 Instalación de Un Solo Comando

# Método 1: Script directo desde GitHub  
curl -fsSL https://raw.githubusercontent.com/tu-usuario/yega-dashboard/main/code/setup/install.sh | bash  
  
# Método 2: Clonar y ejecutar localmente  
git clone https://github.com/tu-usuario/yega-dashboard.git  
cd yega-dashboard  
chmod +x code/setup/install.sh  
./code/setup/install.sh

### 📋 Lo que Hace el Script Automatizado

1. **✅ Verificación del Sistema**
   * Comprueba versiones de PHP, MySQL, Node.js
   * Verifica extensiones PHP requeridas
   * Valida permisos de escritura
2. **📦 Instalación de Dependencias**
   * Instala dependencias PHP con Composer
   * Instala paquetes Node.js con NPM
   * Configura autoloader optimizado
3. **🗄️ Configuración de Base de Datos**
   * Crea base de datos MySQL
   * Configura usuario y permisos
   * Ejecuta migraciones de Prisma
4. **⚙️ Configuración del Entorno**
   * Genera archivo .env desde plantilla
   * Configura claves de seguridad
   * Establece permisos de archivos
5. **🚀 Inicio del Servicio**
   * Compila assets frontend
   * Inicia servidor de desarrollo
   * Ejecuta sincronización inicial

### 🖥️ Scripts por Sistema Operativo

#### Linux (Ubuntu/Debian)

# Instalación completa para Ubuntu/Debian  
curl -fsSL https://raw.githubusercontent.com/tu-usuario/yega-dashboard/main/code/setup/install-linux.sh | bash

#### macOS

# Instalación completa para macOS  
curl -fsSL https://raw.githubusercontent.com/tu-usuario/yega-dashboard/main/code/setup/install-macos.sh | bash

#### Windows (PowerShell)

# Instalación completa para Windows  
iwr -useb https://raw.githubusercontent.com/tu-usuario/yega-dashboard/main/code/setup/install-windows.ps1 | iex

### 📄 Configuración Personalizada

# Instalación con configuración personalizada  
./code/setup/install.sh --config custom-config.json  
  
# Instalación en modo silencioso  
./code/setup/install.sh --silent  
  
# Instalación solo para desarrollo  
./code/setup/install.sh --dev  
  
# Instalación para producción  
./code/setup/install.sh --production

## Instalación Manual

### Paso 1: Clonar el Repositorio

git clone https://github.com/tu-usuario/yega-dashboard.git  
cd yega-dashboard

### Paso 2: Instalar Dependencias PHP

# Verificar versión de PHP  
php --version  
  
# Instalar dependencias con Composer  
composer install --no-dev --optimize-autoloader

### Paso 3: Instalar Dependencias Frontend

# Instalar dependencias Node.js  
npm install  
  
# O usando Yarn  
yarn install

### Paso 4: Configurar Variables de Entorno

# Copiar archivo de ejemplo  
cp code/setup/.env.example .env  
  
# Editar variables de entorno  
nano .env

### Paso 5: Configurar Base de Datos

# Crear base de datos  
mysql -u root -p  
CREATE DATABASE yega\_dashboard;  
CREATE USER 'yega\_user'@'localhost' IDENTIFIED BY 'tu\_password\_seguro';  
GRANT ALL PRIVILEGES ON yega\_dashboard.\* TO 'yega\_user'@'localhost';  
FLUSH PRIVILEGES;  
EXIT;

### Paso 6: Ejecutar Migraciones

# Ejecutar migraciones de Prisma  
npx prisma migrate deploy  
  
# Generar cliente Prisma  
npx prisma generate

### Paso 7: Compilar Assets

# Desarrollo  
npm run dev  
  
# Producción  
npm run build

### Paso 8: Iniciar Servidor

# Servidor de desarrollo  
php -S localhost:8000 -t public  
  
# O usando Laravel Artisan (si es Laravel)  
php artisan serve

## 🔑 Configuración de GitHub API

### 📋 Pasos Detallados para Obtener Token

#### 1. Acceso a GitHub Settings

# Navegar directamente a la configuración  
https://github.com/settings/tokens  
  
# O seguir la ruta manual:  
# GitHub.com → Avatar (esquina superior derecha) → Settings →   
# Developer settings → Personal access tokens → Tokens (classic)

#### 2. Crear Nuevo Token

1. **Click en “Generate new token”** → **“Generate new token (classic)”**
2. **Configurar Token**:

* Token name: "Yega Dashboard API Access"  
  Expiration: 90 days (recomendado para seguridad)  
  Note: "Token para sincronización de repositorios Yega"

1. **Seleccionar Permisos Esenciales**:

| Permiso | Descripción | Necesario |
| --- | --- | --- |
| repo | Acceso completo a repositorios públicos y privados | ✅ **SÍ** |
| repo:status | Acceso a estado de commits | ✅ **SÍ** |
| repo\_deployment | Acceso a deployments | 🔄 *Opcional* |
| public\_repo | Acceso solo a repositorios públicos | ✅ **SÍ** |
| read:user | Leer información de perfil de usuario | ✅ **SÍ** |
| user:email | Acceso a direcciones de email | ✅ **SÍ** |
| read:org | Leer información de organizaciones | ✅ **SÍ** |
| project:read | Leer proyectos | 🔄 *Opcional* |
| workflow | Acceso a GitHub Actions | 🔄 *Opcional* |

#### 3. Configuración de Permisos Detallada

✅ \*\*PERMISOS OBLIGATORIOS\*\*  
┌─ repo  
│ ├─ repo:status  
│ ├─ repo\_deployment (opcional)  
│ └─ public\_repo  
├─ user  
│ ├─ read:user  
│ └─ user:email  
├─ org  
│ └─ read:org  
└─ project (opcional)  
 └─ project:read

#### 4. Copiar y Guardar Token

⚠️ **IMPORTANTE**: El token solo se muestra UNA VEZ

# Ejemplo de token generado:  
ghp\_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
  
# Guárdalo inmediatamente en lugar seguro:  
# 1. En tu gestor de contraseñas  
# 2. En variable de entorno temporal  
# 3. En archivo .env del proyecto

### 🔐 Configuración en el Dashboard

#### Método 1: Variable de Entorno

# Agregar al archivo .env  
GITHUB\_TOKEN=ghp\_tu\_token\_personal\_aqui  
GITHUB\_USERNAME=tu\_usuario\_github  
GITHUB\_API\_URL=https://api.github.com

#### Método 2: Configuración Interactiva

# Ejecutar configurador  
php bin/console github:configure  
  
# O usar el script de configuración  
./code/setup/configure.sh --github-token

### ⚡ Rate Limits y Optimización

#### Límites de GitHub API

| Tipo de Token | Requests/hora | Búsquedas/minuto |
| --- | --- | --- |
| **Personal Access Token** | 5,000 | 30 |
| **OAuth App** | 5,000 | 30 |
| **GitHub App** | 15,000 | 30 |
| **Sin autenticación** | 60 | 10 |

#### Configuración de Rate Limiting

# En .env - Configuración conservadora  
GITHUB\_API\_RATE\_LIMIT=5000  
GITHUB\_REQUESTS\_PER\_MINUTE=60  
GITHUB\_RETRY\_ATTEMPTS=3  
GITHUB\_RETRY\_DELAY=5  
  
# Configuración de timeouts  
GITHUB\_TIMEOUT=30  
GITHUB\_CONNECT\_TIMEOUT=10

#### Monitoreo de Rate Limits

# Verificar límites actuales  
curl -H "Authorization: token $GITHUB\_TOKEN" https://api.github.com/rate\_limit  
  
# Ver headers de respuesta  
curl -I -H "Authorization: token $GITHUB\_TOKEN" https://api.github.com/user  
  
# Comando del dashboard  
php bin/console github:rate-limit

### 🔍 Validación de Token

#### Script de Validación Automática

#!/bin/bash  
# code/setup/validate-github-token.sh  
  
TOKEN="$1"  
  
if [ -z "$TOKEN" ]; then  
 echo "Uso: $0 <github\_token>"  
 exit 1  
fi  
  
# Verificar token  
RESPONSE=$(curl -s -H "Authorization: token $TOKEN" https://api.github.com/user)  
  
if echo "$RESPONSE" | grep -q '"login"'; then  
 USERNAME=$(echo "$RESPONSE" | grep '"login"' | cut -d'"' -f4)  
 echo "✅ Token válido para usuario: $USERNAME"  
   
 # Verificar permisos  
 SCOPES=$(curl -s -I -H "Authorization: token $TOKEN" https://api.github.com/user | grep -i 'x-oauth-scopes' | cut -d':' -f2)  
 echo "📋 Permisos disponibles:$SCOPES"  
else  
 echo "❌ Token inválido o expirado"  
 exit 1  
fi

#### Validación Desde el Dashboard

# Validar configuración completa  
php bin/console github:validate  
  
# Resultado esperado:  
# ✅ Token válido  
# ✅ Permisos suficientes  
# ✅ Rate limit: 4,989/5,000 restantes  
# ✅ Conexión a API exitosa

### 🔄 Renovación y Mantenimiento

#### Configurar Alertas de Expiración

# Agregar al crontab para verificación diaria  
0 9 \* \* \* /path/to/yega-dashboard/bin/console github:check-expiry  
  
# Script de verificación  
#!/bin/bash  
# bin/console github:check-expiry  
  
# Verificar días hasta expiración  
EXPIRY\_DAYS=$(php -r "  
\$token = getenv('GITHUB\_TOKEN');  
\$response = json\_decode(file\_get\_contents('https://api.github.com/user', false, stream\_context\_create([  
 'http' => ['header' => 'Authorization: token ' . \$token]  
])), true);  
echo isset(\$response['message']) ? 0 : 30; // Simplificado  
")  
  
if [ "$EXPIRY\_DAYS" -lt 7 ]; then  
 echo "⚠️ Token de GitHub expira en $EXPIRY\_DAYS días"  
 # Enviar notificación por email/Slack  
fi

#### Rotación de Tokens

# 1. Generar nuevo token en GitHub  
# 2. Actualizar configuración  
php bin/console github:update-token --new-token="ghp\_nuevo\_token"  
  
# 3. Validar nuevo token  
php bin/console github:validate  
  
# 4. Revocar token anterior en GitHub

### 🐛 Troubleshooting GitHub API

#### Errores Comunes

**Error 401 - Unauthorized**

# Causa: Token inválido, expirado o mal configurado  
# Solución:  
1. Verificar token en .env  
2. Regenerar token en GitHub  
3. Verificar permisos del token  
  
# Debug:  
curl -H "Authorization: token $GITHUB\_TOKEN" https://api.github.com/user

**Error 403 - Forbidden**

# Causa: Permisos insuficientes o rate limit excedido  
# Solución:  
1. Verificar permisos del token  
2. Esperar reset de rate limit  
3. Implementar cache para reducir requests  
  
# Debug:  
curl -I -H "Authorization: token $GITHUB\_TOKEN" https://api.github.com/rate\_limit

**Error 422 - Unprocessable Entity**

# Causa: Parámetros de API incorrectos  
# Solución:  
1. Verificar formato de requests  
2. Validar parámetros enviados  
3. Consultar documentación de GitHub API

## 🔄 Migraciones y Schema

### 💾 Setup Inicial de Prisma

#### Inicialización del Proyecto

# Inicializar Prisma en proyecto existente  
npx prisma init  
  
# Esto crea:  
# ├── prisma/  
# │ └── schema.prisma  
# └── .env (si no existe)

#### Configuración del Schema Principal

// prisma/schema.prisma  
generator client {  
 provider = "prisma-client-js"  
 output = "./generated/client"  
}  
  
generator docs {  
 provider = "node node\_modules/prisma-docs-generator"  
 output = "../docs/database"  
}  
  
datasource db {  
 provider = "mysql"  
 url = env("DATABASE\_URL")  
}  
  
// ============================================================================  
// MODELOS PRINCIPALES  
// ============================================================================  
  
model User {  
 id Int @id @default(autoincrement())  
 username String @unique @db.VarChar(50)  
 email String @unique @db.VarChar(100)  
 github\_id String? @unique @db.VarChar(20)  
 avatar\_url String? @db.Text  
 access\_token String? @db.Text  
 refresh\_token String? @db.Text  
 name String? @db.VarChar(100)  
 company String? @db.VarChar(100)  
 location String? @db.VarChar(100)  
 bio String? @db.Text  
 blog String? @db.VarChar(200)  
 twitter String? @db.VarChar(50)  
 public\_repos Int @default(0)  
 public\_gists Int @default(0)  
 followers Int @default(0)  
 following Int @default(0)  
 is\_active Boolean @default(true)  
 last\_login DateTime?  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
   
 // Relaciones  
 repositories Repository[]  
 activities Activity[]  
 collaborations Collaboration[]  
 settings UserSetting?  
   
 @@map("users")  
 @@index([github\_id])  
 @@index([email])  
 @@index([username])  
}  
  
model Repository {  
 id Int @id @default(autoincrement())  
 github\_id String @unique @db.VarChar(20)  
 name String @db.VarChar(100)  
 full\_name String @db.VarChar(200)  
 description String? @db.Text  
 private Boolean @default(false)  
 fork Boolean @default(false)  
 archived Boolean @default(false)  
 disabled Boolean @default(false)  
 stars\_count Int @default(0)  
 forks\_count Int @default(0)  
 watchers\_count Int @default(0)  
 open\_issues Int @default(0)  
 language String? @db.VarChar(50)  
 size Int @default(0)  
 default\_branch String @default("main") @db.VarChar(50)  
 homepage String? @db.VarChar(200)  
 topics Json?  
 license\_key String? @db.VarChar(50)  
 license\_name String? @db.VarChar(100)  
 visibility String @default("public") @db.VarChar(20)  
 clone\_url String? @db.Text  
 ssh\_url String? @db.Text  
 html\_url String? @db.Text  
 last\_sync DateTime?  
 sync\_status String @default("pending") @db.VarChar(20)  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 pushed\_at DateTime?  
   
 // Relaciones  
 user\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 commits Commit[]  
 issues Issue[]  
 pull\_requests PullRequest[]  
 releases Release[]  
 collaborations Collaboration[]  
 languages LanguageStat[]  
   
 @@map("repositories")  
 @@index([user\_id])  
 @@index([github\_id])  
 @@index([name])  
 @@index([language])  
 @@index([visibility])  
}  
  
model Commit {  
 id Int @id @default(autoincrement())  
 sha String @unique @db.VarChar(40)  
 message String @db.Text  
 author\_name String @db.VarChar(100)  
 author\_email String @db.VarChar(100)  
 committer\_name String @db.VarChar(100)  
 committer\_email String @db.VarChar(100)  
 additions Int @default(0)  
 deletions Int @default(0)  
 changes Int @default(0)  
 files\_changed Json?  
 html\_url String? @db.Text  
 committed\_at DateTime  
 created\_at DateTime @default(now())  
   
 // Relaciones  
 repository\_id Int  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("commits")  
 @@index([repository\_id])  
 @@index([sha])  
 @@index([committed\_at])  
 @@index([author\_email])  
}  
  
model Issue {  
 id Int @id @default(autoincrement())  
 github\_id String @unique @db.VarChar(20)  
 number Int  
 title String @db.VarChar(500)  
 body String? @db.LongText  
 state String @db.VarChar(20)  
 labels Json?  
 assignees Json?  
 milestone String? @db.VarChar(200)  
 comments\_count Int @default(0)  
 author String @db.VarChar(100)  
 author\_avatar String? @db.Text  
 html\_url String? @db.Text  
 closed\_at DateTime?  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
   
 // Relaciones  
 repository\_id Int  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("issues")  
 @@index([repository\_id])  
 @@index([github\_id])  
 @@index([state])  
 @@index([number, repository\_id])  
}  
  
model PullRequest {  
 id Int @id @default(autoincrement())  
 github\_id String @unique @db.VarChar(20)  
 number Int  
 title String @db.VarChar(500)  
 body String? @db.LongText  
 state String @db.VarChar(20)  
 head\_branch String @db.VarChar(100)  
 base\_branch String @db.VarChar(100)  
 merged Boolean @default(false)  
 mergeable Boolean?  
 draft Boolean @default(false)  
 additions Int @default(0)  
 deletions Int @default(0)  
 changed\_files Int @default(0)  
 comments\_count Int @default(0)  
 review\_comments Int @default(0)  
 commits\_count Int @default(0)  
 author String @db.VarChar(100)  
 author\_avatar String? @db.Text  
 html\_url String? @db.Text  
 merged\_at DateTime?  
 closed\_at DateTime?  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
   
 // Relaciones  
 repository\_id Int  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("pull\_requests")  
 @@index([repository\_id])  
 @@index([github\_id])  
 @@index([state])  
 @@index([number, repository\_id])  
}  
  
model Release {  
 id Int @id @default(autoincrement())  
 github\_id String @unique @db.VarChar(20)  
 tag\_name String @db.VarChar(100)  
 name String? @db.VarChar(200)  
 body String? @db.LongText  
 draft Boolean @default(false)  
 prerelease Boolean @default(false)  
 target\_commitish String @db.VarChar(100)  
 author String @db.VarChar(100)  
 download\_count Int @default(0)  
 assets\_count Int @default(0)  
 html\_url String? @db.Text  
 published\_at DateTime?  
 created\_at DateTime @default(now())  
   
 // Relaciones  
 repository\_id Int  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("releases")  
 @@index([repository\_id])  
 @@index([tag\_name])  
 @@index([published\_at])  
}  
  
model Activity {  
 id Int @id @default(autoincrement())  
 type String @db.VarChar(50)  
 action String @db.VarChar(50)  
 description String @db.VarChar(500)  
 metadata Json?  
 entity\_type String? @db.VarChar(50)  
 entity\_id String? @db.VarChar(20)  
 repository\_name String? @db.VarChar(200)  
 public Boolean @default(true)  
 created\_at DateTime @default(now())  
   
 // Relaciones  
 user\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 @@map("activities")  
 @@index([user\_id])  
 @@index([type])  
 @@index([created\_at])  
 @@index([entity\_type, entity\_id])  
}  
  
model Collaboration {  
 id Int @id @default(autoincrement())  
 permission String @db.VarChar(20)  
 role String? @db.VarChar(50)  
 created\_at DateTime @default(now())  
   
 // Relaciones  
 user\_id Int  
 repository\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("collaborations")  
 @@unique([user\_id, repository\_id])  
 @@index([user\_id])  
 @@index([repository\_id])  
}  
  
model LanguageStat {  
 id Int @id @default(autoincrement())  
 language String @db.VarChar(50)  
 bytes Int @default(0)  
 percentage Decimal @default(0.00) @db.Decimal(5,2)  
 updated\_at DateTime @updatedAt  
   
 // Relaciones  
 repository\_id Int  
 repository Repository @relation(fields: [repository\_id], references: [id], onDelete: Cascade)  
   
 @@map("language\_stats")  
 @@unique([repository\_id, language])  
 @@index([language])  
 @@index([bytes])  
}  
  
model UserSetting {  
 id Int @id @default(autoincrement())  
 theme String @default("dark") @db.VarChar(20)  
 language String @default("es") @db.VarChar(5)  
 timezone String @default("America/Mexico\_City") @db.VarChar(50)  
 notifications Json?  
 privacy Json?  
 sync\_frequency Int @default(300) // segundos  
 auto\_sync Boolean @default(true)  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
   
 // Relaciones  
 user\_id Int @unique  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 @@map("user\_settings")  
}  
  
model SyncLog {  
 id Int @id @default(autoincrement())  
 type String @db.VarChar(50)  
 status String @db.VarChar(20)  
 message String? @db.Text  
 details Json?  
 duration Int? // milisegundos  
 records\_processed Int @default(0)  
 started\_at DateTime  
 completed\_at DateTime?  
 created\_at DateTime @default(now())  
   
 @@map("sync\_logs")  
 @@index([type])  
 @@index([status])  
 @@index([started\_at])  
}  
  
model ApiToken {  
 id Int @id @default(autoincrement())  
 name String @db.VarChar(100)  
 token String @unique @db.VarChar(255)  
 permissions Json?  
 last\_used DateTime?  
 expires\_at DateTime?  
 is\_active Boolean @default(true)  
 created\_at DateTime @default(now())  
   
 // Relaciones  
 user\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 @@map("api\_tokens")  
 @@index([user\_id])  
 @@index([token])  
 @@index([expires\_at])  
}

### 🛠️ Comandos de Migración

#### Comandos Esenciales

# Generar cliente Prisma  
npx prisma generate  
  
# Ver estado de migraciones  
npx prisma migrate status  
  
# Crear nueva migración (desarrollo)  
npx prisma migrate dev --name "descripcion\_cambio"  
  
# Aplicar migraciones (producción)  
npx prisma migrate deploy  
  
# Reset completo de BD (CUIDADO - solo desarrollo)  
npx prisma migrate reset  
  
# Introspect BD existente  
npx prisma db pull  
  
# Push schema sin crear migración (prototipado)  
npx prisma db push

#### Workflow de Desarrollo

# 1. Modificar schema.prisma  
vim prisma/schema.prisma  
  
# 2. Crear migración  
npx prisma migrate dev --name "add\_user\_preferences"  
  
# 3. Verificar archivos generados  
ls prisma/migrations/  
  
# 4. Regenerar cliente  
npx prisma generate  
  
# 5. Actualizar código que usa el cliente

#### Workflow de Producción

# 1. Backup de BD  
mysqldump -u user -p database > backup\_pre\_migration.sql  
  
# 2. Aplicar migraciones  
npx prisma migrate deploy  
  
# 3. Regenerar cliente  
npx prisma generate  
  
# 4. Verificar aplicación  
php bin/console app:health-check

### 🌱 Seeders y Datos Iniciales

#### Configuración de Seeder

// prisma/seed.js  
const { PrismaClient } = require('@prisma/client')  
const prisma = new PrismaClient()  
  
async function main() {  
 console.log('🌱 Iniciando seeding...')  
   
 // Usuario de ejemplo  
 const user = await prisma.user.upsert({  
 where: { email: 'admin@yega.dev' },  
 update: {},  
 create: {  
 username: 'yega-admin',  
 email: 'admin@yega.dev',  
 name: 'Yega Administrator',  
 github\_id: '1',  
 public\_repos: 0,  
 settings: {  
 create: {  
 theme: 'dark',  
 language: 'es',  
 auto\_sync: true  
 }  
 }  
 }  
 })  
   
 // Repositorio de ejemplo  
 await prisma.repository.upsert({  
 where: { github\_id: '1' },  
 update: {},  
 create: {  
 github\_id: '1',  
 name: 'yega-dashboard',  
 full\_name: 'yega/yega-dashboard',  
 description: 'Dashboard para el ecosistema Yega',  
 language: 'PHP',  
 stars\_count: 10,  
 user\_id: user.id  
 }  
 })  
   
 console.log('✅ Seeding completado')  
}  
  
main()  
 .catch((e) => {  
 console.error(e)  
 process.exit(1)  
 })  
 .finally(async () => {  
 await prisma.$disconnect()  
 })

#### Ejecutar Seeders

# Configurar en package.json  
{  
 "prisma": {  
 "seed": "node prisma/seed.js"  
 }  
}  
  
# Ejecutar seeder  
npx prisma db seed  
  
# Ejecutar seeder en migración reset  
npx prisma migrate reset

## 🗄️ Configuración de Base de Datos

### 📋 Setup de MySQL/MariaDB

#### Instalación y Configuración Básica

-- Crear base de datos  
CREATE DATABASE yega\_dashboard   
 CHARACTER SET utf8mb4   
 COLLATE utf8mb4\_unicode\_ci;  
  
-- Crear usuario dedicado  
CREATE USER 'yega\_user'@'localhost' IDENTIFIED BY 'password\_seguro\_aqui';  
  
-- Otorgar permisos completos  
GRANT ALL PRIVILEGES ON yega\_dashboard.\* TO 'yega\_user'@'localhost';  
GRANT SELECT ON performance\_schema.\* TO 'yega\_user'@'localhost';  
FLUSH PRIVILEGES;  
  
-- Verificar usuario  
SHOW GRANTS FOR 'yega\_user'@'localhost';

#### Configuración Optimizada para Producción

# /etc/mysql/mysql.conf.d/yega-dashboard.cnf  
[mysqld]  
  
# Configuraciones básicas  
default\_storage\_engine = InnoDB  
default\_table\_type = InnoDB  
  
# Optimizaciones de memoria  
innodb\_buffer\_pool\_size = 512M  
innodb\_log\_file\_size = 128M  
innodb\_log\_buffer\_size = 16M  
innodb\_flush\_method = O\_DIRECT  
  
# Conexiones  
max\_connections = 300  
max\_connect\_errors = 1000  
connect\_timeout = 10  
wait\_timeout = 600  
interactive\_timeout = 600  
  
# Query Cache  
query\_cache\_type = 1  
query\_cache\_size = 64M  
query\_cache\_limit = 2M  
  
# MyISAM (para tablas de sesión si se usan)  
key\_buffer\_size = 32M  
  
# Logging  
log\_error = /var/log/mysql/error.log  
slow\_query\_log = 1  
slow\_query\_log\_file = /var/log/mysql/slow-query.log  
long\_query\_time = 2  
  
# Binlog (para replicación)  
log\_bin = /var/log/mysql/mysql-bin.log  
binlog\_expire\_logs\_seconds = 2592000  
max\_binlog\_size = 100M  
  
# InnoDB específicas  
innodb\_file\_per\_table = 1  
innodb\_open\_files = 400  
innodb\_io\_capacity = 400  
innodb\_flush\_neighbors = 0  
  
# Configuraciones de threads  
thread\_cache\_size = 16  
table\_open\_cache = 2000  
table\_definition\_cache = 1000  
  
# Configuraciones de red  
max\_allowed\_packet = 64M  
net\_buffer\_length = 32K  
  
# Configuraciones de seguridad  
local\_infile = 0

#### Monitoreo y Mantenimiento

-- Verificar estado de InnoDB  
SHOW ENGINE INNODB STATUS\G  
  
-- Ver tamaños de tablas  
SELECT   
 table\_schema as 'Database',  
 table\_name as 'Table',  
 round(((data\_length + index\_length) / 1024 / 1024), 2) as 'Size (MB)'  
FROM information\_schema.tables   
WHERE table\_schema = 'yega\_dashboard'  
ORDER BY (data\_length + index\_length) DESC;  
  
-- Optimizar tablas  
OPTIMIZE TABLE repositories, commits, issues, pull\_requests;  
  
-- Verificar fragmentación  
SELECT   
 table\_name,  
 round(data\_length/1024/1024,2) as data\_mb,  
 round(data\_free/1024/1024,2) as data\_free\_mb,  
 round(data\_free/(data\_length+data\_free)\*100,2) as fragmentation\_pct  
FROM information\_schema.tables   
WHERE table\_schema = 'yega\_dashboard' AND data\_free > 0;

### 🔒 Backup y Restauración

#### Scripts de Backup Automatizado

#!/bin/bash  
# code/setup/backup.sh  
  
set -e  
  
# Configuración  
DB\_NAME="yega\_dashboard"  
DB\_USER="yega\_user"  
DB\_PASS="$(grep DB\_PASSWORD .env | cut -d'=' -f2)"  
BACKUP\_DIR="backups"  
DATE=$(date +"%Y%m%d\_%H%M%S")  
BACKUP\_FILE="$BACKUP\_DIR/yega\_dashboard\_$DATE.sql"  
COMPRESSED\_FILE="$BACKUP\_DIR/yega\_dashboard\_$DATE.sql.gz"  
  
# Crear directorio de backup  
mkdir -p "$BACKUP\_DIR"  
  
echo "📋 Iniciando backup de base de datos..."  
  
# Crear backup  
mysqldump \  
 --user="$DB\_USER" \  
 --password="$DB\_PASS" \  
 --host=localhost \  
 --single-transaction \  
 --routines \  
 --triggers \  
 --add-drop-table \  
 --extended-insert \  
 --create-options \  
 --comments \  
 --default-character-set=utf8mb4 \  
 "$DB\_NAME" > "$BACKUP\_FILE"  
  
# Comprimir backup  
gzip "$BACKUP\_FILE"  
  
echo "✅ Backup completado: $COMPRESSED\_FILE"  
echo "📏 Tamaño: $(du -h "$COMPRESSED\_FILE" | cut -f1)"  
  
# Limpiar backups antiguos (mantener últimos 30 días)  
find "$BACKUP\_DIR" -name "yega\_dashboard\_\*.sql.gz" -mtime +30 -delete  
  
echo "🧹 Backups antiguos limpiados"

#### Script de Restauración

#!/bin/bash  
# code/setup/restore.sh  
  
set -e  
  
BACKUP\_FILE="$1"  
  
if [ -z "$BACKUP\_FILE" ]; then  
 echo "Uso: $0 <archivo\_backup.sql.gz>"  
 echo "Backups disponibles:"  
 ls -la backups/yega\_dashboard\_\*.sql.gz 2>/dev/null || echo "No hay backups disponibles"  
 exit 1  
fi  
  
if [ ! -f "$BACKUP\_FILE" ]; then  
 echo "Error: Archivo $BACKUP\_FILE no encontrado"  
 exit 1  
fi  
  
# Configuración  
DB\_NAME="yega\_dashboard"  
DB\_USER="yega\_user"  
DB\_PASS="$(grep DB\_PASSWORD .env | cut -d'=' -f2)"  
  
echo "⚠️ ADVERTENCIA: Esto sobrescribirá la base de datos actual"  
read -p "¿Estás seguro? (yes/no): " confirm  
  
if [ "$confirm" != "yes" ]; then  
 echo "Operación cancelada"  
 exit 0  
fi  
  
echo "📋 Iniciando restauración desde $BACKUP\_FILE..."  
  
# Descomprimir y restaurar  
if [[ "$BACKUP\_FILE" == \*.gz ]]; then  
 gunzip -c "$BACKUP\_FILE" | mysql \  
 --user="$DB\_USER" \  
 --password="$DB\_PASS" \  
 --host=localhost \  
 "$DB\_NAME"  
else  
 mysql \  
 --user="$DB\_USER" \  
 --password="$DB\_PASS" \  
 --host=localhost \  
 "$DB\_NAME" < "$BACKUP\_FILE"  
fi  
  
echo "✅ Restauración completada"  
echo "🔄 Regenerando cliente Prisma..."  
npx prisma generate  
  
echo "✅ Base de datos restaurada correctamente"

## ⚙️ Variables de Entorno

### 📋 Configuración Completa del .env

El archivo .env es el corazón de la configuración del dashboard. Copia el archivo de ejemplo y personaliza según tu entorno:

# Copiar archivo de ejemplo  
cp code/setup/.env.example .env  
  
# Editar configuración  
nano .env

### 🔑 Generación de Claves de Seguridad

# Generar clave de aplicación  
echo "APP\_KEY=base64:$(openssl rand -base64 32)"  
  
# Generar clave JWT  
echo "JWT\_SECRET=$(openssl rand -base64 64)"  
  
# Generar clave de encriptación adicional  
echo "ENCRYPTION\_KEY=$(openssl rand -base64 32)"  
  
# Script automatizado  
./code/setup/generate-keys.sh

### 🌍 Configuración por Entornos

#### Desarrollo Local

APP\_ENV=local  
APP\_DEBUG=true  
LOG\_LEVEL=debug  
CACHE\_DRIVER=file  
SESSION\_DRIVER=file  
QUEUE\_CONNECTION=sync

#### Staging

APP\_ENV=staging  
APP\_DEBUG=true  
LOG\_LEVEL=info  
CACHE\_DRIVER=redis  
SESSION\_DRIVER=redis  
QUEUE\_CONNECTION=redis

#### Producción

APP\_ENV=production  
APP\_DEBUG=false  
LOG\_LEVEL=warning  
CACHE\_DRIVER=redis  
SESSION\_DRIVER=redis  
QUEUE\_CONNECTION=redis

## 📖 Guía de Uso del Dashboard

### 🚀 Primer Acceso y Configuración

#### 1. Iniciar el Servidor

# Método 1: Servidor PHP integrado  
composer serve  
# O equivalente:  
php -S localhost:8000 -t public  
  
# Método 2: NPM script  
npm run serve  
  
# Método 3: Con hot reload para desarrollo  
npm run dev

#### 2. Acceso al Dashboard

# Abrir en navegador  
http://localhost:8000  
  
# URLs principales:  
http://localhost:8000/ # Dashboard principal  
http://localhost:8000/repositories # Lista de repositorios  
http://localhost:8000/activity # Feed de actividad  
http://localhost:8000/stats # Estadísticas detalladas  
http://localhost:8000/settings # Configuración de usuario

#### 3. Configuración Inicial

1. **Configurar Token de GitHub**
   * Ve a Configuración (Settings)
   * Introduce tu token de GitHub
   * Verifica la conexión
2. **Primera Sincronización**

* # Sincronización manual  
  php sync.php  
    
  # O desde el dashboard  
  # Click en "Sincronizar" en la barra superior

1. **Personalizar Configuración**
   * Tema (claro/oscuro/automático)
   * Idioma (español/inglés)
   * Frecuencia de sincronización
   * Notificaciones

### 📋 Funcionalidades Principales

#### Dashboard Principal

**Vista General** - Resumen de repositorios totales - Actividad reciente (commits, issues, PRs) - Gráficos de productividad - Repositorios más activos - Lenguajes de programación utilizados

**Widgets Disponibles** - 📊 Gráfico de commits por día - 📋 Lista de issues abiertos - 🔄 Pull requests pendientes - 🏆 Repositorios con más estrellas - 👥 Colaboradores activos

#### Gestión de Repositorios

**Listado de Repositorios**

┌─ Filtros Disponibles  
├─ 🔍 Búsqueda por nombre  
├─ 🏷️ Filtro por lenguaje  
├─ 🔒 Filtro por visibilidad (público/privado)  
├─ ⭐ Ordenar por estrellas  
├─ 📅 Ordenar por última actualización  
└─ 📊 Ordenar por actividad

**Detalles de Repositorio** - Información general (descripción, licencia, temas) - Estadísticas (estrellas, forks, issues) - Commits recientes - Issues y Pull Requests - Colaboradores - Distribusión de lenguajes - Releases y tags

#### Análisis de Actividad

**Feed de Actividad** - Commits recientes - Issues creados/cerrados - Pull Requests - Releases publicados - Nuevos repositorios

**Métricas y Estadísticas** - Commits por hora/día/semana/mes - Productividad por repositorio - Análisis de lenguajes - Trends de actividad - Colaboración en equipo

### 🛠️ Herramientas y Utilidades

#### Comandos de Consola

# Sincronización manual  
php sync.php  
composer sync  
  
# Verificar estado del sistema  
php bin/console app:health-check  
  
# Limpiar cache  
php bin/console cache:clear  
  
# Ver logs en tiempo real  
tail -f logs/sync.log  
npm run logs  
  
# Backup de base de datos  
composer backup  
./code/setup/backup.sh  
  
# Ejecutar tests  
composer test  
npm test  
  
# Actualizar dependencias  
composer update  
npm update

#### API REST

**Endpoints Principales**

# Estado del sistema  
GET /api/health  
  
# Información de usuario  
GET /api/user  
  
# Lista de repositorios  
GET /api/repositories  
GET /api/repositories/{id}  
  
# Actividad reciente  
GET /api/activity  
GET /api/activity/user/{userId}  
  
# Estadísticas  
GET /api/stats/overview  
GET /api/stats/repositories  
GET /api/stats/languages  
  
# Sincronización  
POST /api/sync/repositories  
POST /api/sync/user  
GET /api/sync/status

**Autenticación API**

# Generar token de API  
php bin/console api:token:create --name="Mi App"  
  
# Usar token en requests  
curl -H "Authorization: Bearer tu\_token\_aqui" \  
 http://localhost:8000/api/repositories

### 📁 Exportación de Datos

#### Formatos Disponibles

# Exportar repositorios a CSV  
php bin/console export:repositories --format=csv  
  
# Exportar actividad a JSON  
php bin/console export:activity --format=json --from=2024-01-01  
  
# Exportar estadísticas a Excel  
php bin/console export:stats --format=xlsx  
  
# Backup completo  
php bin/console export:full-backup

#### Integraciones Externas

# Webhook para Slack  
POST /webhooks/slack  
  
# Webhook para Discord  
POST /webhooks/discord  
  
# Webhook genérico  
POST /webhooks/generic

## Configuración de Variables de Entorno

### Archivo .env Completo

# Configuración de Aplicación  
APP\_NAME="Yega Dashboard"  
APP\_ENV=production  
APP\_DEBUG=false  
APP\_URL=http://localhost:8000  
APP\_TIMEZONE=America/Mexico\_City  
  
# Configuración de Base de Datos  
DB\_CONNECTION=mysql  
DB\_HOST=localhost  
DB\_PORT=3306  
DB\_DATABASE=yega\_dashboard  
DB\_USERNAME=yega\_user  
DB\_PASSWORD=tu\_password\_seguro  
  
# URL de Base de Datos para Prisma  
DATABASE\_URL="mysql://yega\_user:tu\_password\_seguro@localhost:3306/yega\_dashboard"  
  
# Configuración de GitHub API  
GITHUB\_TOKEN=ghp\_tu\_token\_personal\_aqui  
GITHUB\_USERNAME=tu\_usuario\_github  
GITHUB\_API\_URL=https://api.github.com  
GITHUB\_WEBHOOK\_SECRET=webhook\_secret\_aleatorio  
  
# Configuración de Cache  
CACHE\_DRIVER=redis  
REDIS\_HOST=localhost  
REDIS\_PORT=6379  
REDIS\_PASSWORD=null  
REDIS\_DB=0  
  
# Configuración de Sesiones  
SESSION\_DRIVER=database  
SESSION\_LIFETIME=120  
SESSION\_ENCRYPT=true  
  
# Configuración de Email  
MAIL\_MAILER=smtp  
MAIL\_HOST=localhost  
MAIL\_PORT=587  
MAIL\_USERNAME=null  
MAIL\_PASSWORD=null  
MAIL\_ENCRYPTION=tls  
MAIL\_FROM\_ADDRESS=noreply@yega-dashboard.local  
MAIL\_FROM\_NAME="Yega Dashboard"  
  
# Configuración de Logging  
LOG\_CHANNEL=daily  
LOG\_LEVEL=info  
LOG\_DAYS=14  
  
# Configuración de Seguridad  
APP\_KEY=base64:tu\_app\_key\_generada\_aqui  
JWT\_SECRET=tu\_jwt\_secret\_aqui  
ENCRYPTION\_KEY=tu\_encryption\_key\_aqui  
  
# Configuración de API Rate Limiting  
API\_RATE\_LIMIT=60  
API\_RATE\_LIMIT\_WINDOW=60  
  
# Configuración de GitHub Sync  
GITHUB\_SYNC\_INTERVAL=300  
GITHUB\_BATCH\_SIZE=50  
GITHUB\_TIMEOUT=30

### Generar Keys de Seguridad

# Generar APP\_KEY (Laravel)  
php artisan key:generate  
  
# Generar JWT Secret  
php artisan jwt:secret  
  
# Generar clave de encriptación personalizada  
openssl rand -base64 32

## Migraciones de Base de Datos

### Setup Inicial de Prisma

# Inicializar Prisma  
npx prisma init  
  
# Generar migración inicial  
npx prisma migrate dev --name init  
  
# Aplicar migraciones en producción  
npx prisma migrate deploy

### Comandos de Migración

# Crear nueva migración  
npx prisma migrate dev --name nombre\_migracion  
  
# Aplicar migraciones pendientes  
npx prisma migrate deploy  
  
# Reset completo de base de datos (CUIDADO)  
npx prisma migrate reset  
  
# Verificar estado de migraciones  
npx prisma migrate status  
  
# Generar cliente después de cambios  
npx prisma generate

### Schema Base

// prisma/schema.prisma  
generator client {  
 provider = "prisma-client-js"  
}  
  
datasource db {  
 provider = "mysql"  
 url = env("DATABASE\_URL")  
}  
  
model User {  
 id Int @id @default(autoincrement())  
 username String @unique  
 email String @unique  
 github\_id String? @unique  
 avatar\_url String?  
 access\_token String?  
 refresh\_token String?  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
   
 repositories Repository[]  
 activities Activity[]  
   
 @@map("users")  
}  
  
model Repository {  
 id Int @id @default(autoincrement())  
 github\_id String @unique  
 name String  
 full\_name String  
 description String?  
 private Boolean @default(false)  
 fork Boolean @default(false)  
 stars\_count Int @default(0)  
 forks\_count Int @default(0)  
 language String?  
 size Int @default(0)  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 pushed\_at DateTime?  
   
 user\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 commits Commit[]  
 issues Issue[]  
 pull\_requests PullRequest[]  
   
 @@map("repositories")  
}  
  
model Activity {  
 id Int @id @default(autoincrement())  
 type String  
 description String  
 metadata Json?  
 created\_at DateTime @default(now())  
   
 user\_id Int  
 user User @relation(fields: [user\_id], references: [id], onDelete: Cascade)  
   
 @@map("activities")  
}

## Guía de Uso

### Primer Acceso

1. **Acceder al Dashboard**:

* http://localhost:8000

1. **Autenticación con GitHub**:
   * Click en “Login with GitHub”
   * Autorizar la aplicación
   * Serás redirigido al dashboard
2. **Configuración Inicial**:
   * Revisar perfil de usuario
   * Sincronizar repositorios
   * Configurar preferencias

### Funcionalidades Principales

#### Dashboard Principal

* **Vista General**: Estadísticas de repositorios y actividad
* **Repositorios**: Lista de todos tus repositorios
* **Actividad Reciente**: Últimos commits, issues, PRs
* **Estadísticas**: Gráficos de actividad y productividad

#### Gestión de Repositorios

* **Sincronización**: Actualizar datos desde GitHub
* **Filtrado**: Por lenguaje, estado, visibilidad
* **Búsqueda**: Encontrar repositorios específicos
* **Detalles**: Ver información completa del repositorio

#### Análisis de Código

* **Métricas de Commits**: Frecuencia y patrones
* **Análisis de Lenguajes**: Distribución de código
* **Colaboradores**: Actividad del equipo
* **Issues y PRs**: Estado y tendencias

### Comandos Útiles

# Sincronizar datos de GitHub  
php artisan github:sync  
  
# Limpiar cache  
php artisan cache:clear  
  
# Optimizar aplicación  
php artisan optimize  
  
# Ver logs en tiempo real  
tail -f storage/logs/laravel.log  
  
# Backup de base de datos  
php artisan backup:run

## Solución de Problemas

### Problemas Comunes

#### Error de Conexión a GitHub API

**Síntoma**: 401 Unauthorized o 403 Forbidden

**Solución**:

# Verificar token  
curl -H "Authorization: token $GITHUB\_TOKEN" https://api.github.com/user  
  
# Regenerar token si es necesario  
# Verificar permisos del token  
# Actualizar .env con nuevo token

#### Error de Conexión a Base de Datos

**Síntoma**: SQLSTATE[HY000] [2002] Connection refused

**Solución**:

# Verificar servicio MySQL  
sudo systemctl status mysql  
  
# Iniciar si está detenido  
sudo systemctl start mysql  
  
# Verificar credenciales  
mysql -u yega\_user -p yega\_dashboard

#### Problemas de Permisos

**Síntoma**: Errores de escritura en archivos

**Solución**:

# Configurar permisos correctos  
sudo chown -R www-data:www-data storage  
sudo chown -R www-data:www-data bootstrap/cache  
sudo chmod -R 775 storage  
sudo chmod -R 775 bootstrap/cache

#### Problemas de Memoria PHP

**Síntoma**: Fatal error: Allowed memory size exhausted

**Solución**:

# Aumentar límite de memoria en php.ini  
memory\_limit = 512M  
  
# O temporalmente  
php -d memory\_limit=512M artisan comando

### Logs y Debugging

# Ver logs de aplicación  
tail -f storage/logs/laravel.log  
  
# Ver logs de web server  
sudo tail -f /var/log/apache2/error.log  
sudo tail -f /var/log/nginx/error.log  
  
# Ver logs de MySQL  
sudo tail -f /var/log/mysql/error.log  
  
# Modo debug  
# En .env  
APP\_DEBUG=true  
LOG\_LEVEL=debug

### Contacto y Soporte

* **Documentación**: [Wiki del proyecto](https://github.com/tu-usuario/yega-dashboard/wiki)
* **Issues**: [GitHub Issues](https://github.com/tu-usuario/yega-dashboard/issues)
* **Discussions**: [GitHub Discussions](https://github.com/tu-usuario/yega-dashboard/discussions)
* **Email**: soporte@yega-dashboard.com

### Actualizaciones

# Actualizar a última versión  
git pull origin main  
composer install --no-dev --optimize-autoloader  
npm install  
npm run build  
npx prisma migrate deploy  
php artisan cache:clear  
php artisan config:cache

**Última actualización**: 17 de Agosto, 2025 **Versión**: 1.0.0