



# Documentación Completa - Dashboard Yega

Dashboard profesional para análisis y gestión del ecosistema Yega en  
GitHub

PHP 8.1+

MySQL 8.0+

Node.js 16+

License MIT



## Tabla de Contenidos

1. [Requisitos del Sistema](#)
2. [Instalación Automatizada](#)
3. [Instalación Manual](#)
4. [Configuración de GitHub API](#)
5. [Configuración de Base de Datos](#)
6. [Variables de Entorno](#)
7. [Migraciones y Schema](#)
8. [Guía de Uso del Dashboard](#)
9. [Troubleshooting](#)
10. [Scripts de Deployment](#)
11. [Mantenimiento y Actualizaciones](#)
12. [Soporte y Contacto](#)

# Requisitos del Sistema

## Requisitos Mínimos

Componente	Versión Mínima	Recomendada	Notas
<b>PHP</b>	8.1	8.2+	Con extensiones requeridas
<b>MySQL</b>	8.0	8.0.32+	O MariaDB 10.6+
<b>Composer</b>	2.0	2.5+	Gestor de dependencias PHP
<b>Node.js</b>	16.0	18.0+ LTS	Para herramientas frontend
<b>NPM</b>	8.0	9.0+	O Yarn 1.22+
<b>Git</b>	2.25	2.40+	Control de versiones
<b>Memoria RAM</b>	2GB	4GB+	Para operaciones complejas
<b>Espacio en Disco</b>	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"
```

3. Seleccionar Permisos Esenciales:

Permiso	Descripción	Necesario
<code>repo</code>	Acceso completo a repositorios públicos y privados	✓ <b>SÍ</b>
<code>repo:status</code>	Acceso a estado de commits	✓ <b>SÍ</b>
<code>repo_deployment</code>	Acceso a deployments	 <i>Opcional</i>
<code>public_repo</code>	Acceso solo a repositorios públicos	✓ <b>SÍ</b>
<code>read:user</code>	Leer información de perfil de usuario	✓ <b>SÍ</b>
<code>user:email</code>	Acceso a direcciones de email	✓ <b>SÍ</b>
<code>read:org</code>	Leer información de organizaciones	✓ <b>SÍ</b>
<code>project:read</code>	Leer proyectos	 <i>Opcional</i>
<code>workflow</code>	Acceso a GitHub Actions	 <i>Opcional</i>


### 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
<b>Personal Access Token</b>	5,000	30
<b>OAuth App</b>	5,000	30
<b>GitHub App</b>	15,000	30
<b>Sin autenticación</b>	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=<span class="math-inline" style="display: inline;"><math
xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mo stretchy="false">&#x00028;</mo><mi>c</
mi><mi>u</mi><mi>r</mi><mi>l</mi><mo>&#x02212;</mo><mi>s</
mi><mo>&#x02212;</mo><mi>H</mi><mi>"</mi><mi>A</mi><mi>u</
mi><mi>t</mi><mi>h</mi><mi>o</mi><mi>r</mi><mi>i</mi><mi>z</
mi><mi>a</mi><mi>t</mi><mi>i</mi><mi>o</mi><mi>n</mi><mi>:</
mi><mi>t</mi><mi>o</mi><mi>k</mi><mi>e</mi><mi>n</mi></mrow></
math></span>TOKEN" https://api.github.com/user)

if echo "$RESPONSE" | grep -q '"login"'; then
    USERNAME=<span class="math-inline" style="display:
inline;"><math xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mo stretchy="false">&#x00028;</mo><mi>e</
mi><mi>c</mi><mi>h</mi><mi>o</mi><mi>"</mi></mrow></math></
span>RESPONSE" | grep '"login"' | cut -d'"' -f4)
    echo "✅ Token válido para usuario: $USERNAME"

    # Verificar permisos
    SCOPES=<span class="math-inline" style="display:
inline;"><math xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mo stretchy="false">&#x00028;</mo><mi>c</
mi><mi>u</mi><mi>r</mi><mi>l</mi><mo>&#x02212;</mo><mi>s</
mi><mo>&#x02212;</mo><mi>I</mi><mo>&#x02212;</mo><mi>H</mi><mi>"</
mi><mi>"</mi></mrow></math></span>
```



```

mi><mi>A</mi><mi>u</mi><mi>t</mi><mi>h</mi><mi>o</mi><mi>r</mi><mi>i</mi><mi>z</mi><mi>a</mi><mi>t</mi><mi>i</mi><mi>o</mi><mi>n</mi><mi>:</mi><mi>t</mi><mi>o</mi><mi>k</mi><mi>e</mi><mi>n</mi></mrow></math></span>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="

```

```

xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mi>D</mi><msub><mi>B</mi><mi>N</mi></msub><mi>A</mi><mi>M</mi><mi>E</mi><mi>"</mi><mo>&#x0003E;</mo><mi>"</mi></mrow></math></span>BACKUP_FILE"

# Comprimir backup
gzip "$BACKUP_FILE"

echo "✅ Backup completado: $COMPRESSED_FILE"
echo "📏 Tamaño: <span class="math-inline" style="display: inline;"><math xmlns="http://www.w3.org/1998/Math/MathML" display="inline"><mrow><mo stretchy="false">&#x00028;</mo><mi>d</mi><mi>u</mi><mo>&#x02212;</mo><mi>h</mi><mi>"</mi></mrow></math></span>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 \
        "<span class='math-inline' style='display: inline;'"><math
xmlns='http://www.w3.org/1998/Math/MathML'
display='inline'"><mrow><mi>D</mi><msub><mi>B</mi><mi>N</mi></
msub><mi>A</mi><mi>M</mi><mi>E</mi><mi>"</mi><mo>&#x0003C;</
mo><mi>"</mi></mrow></math></span>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

```
```bash
```

```
# 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
- Distribución 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`

### 2. Autenticación con GitHub:

- Click en "Login with GitHub"
- Autorizar la aplicación
- Serás redirigido al dashboard

### 3. 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](#)
- **Issues:** [GitHub Issues](#)
- **Discussions:** [GitHub 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