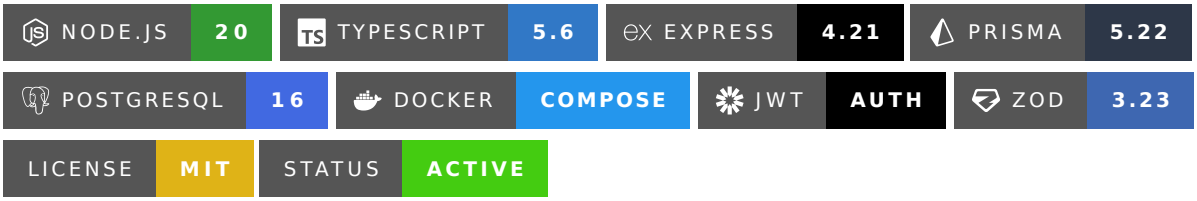
Backend DWEC - Prisma + JWT



Backend educativo completo con Node.js + Express + TypeScript + Prisma + JWT diseñado para que alumnos de React practiquen autenticación y operaciones CRUD.

Características

- Autenticación JWT (registro, login, me)
- CRUD completo de tareas con ownership
- Sistema de roles (USER/ADMIN)
- Paginación y búsqueda
- Validación estricta con Zod
- Error handling centralizado
- PostgreSQL + Prisma ORM
- Docker Compose (backend, postgres, pgAdmin)
- TypeScript estricto con ESM
- CORS configurado para React
- Documentación completa
- Colección de Insomnia
- Plantilla para crear nuevos recursos

Stack tecnológico

Tecnología	Propósito
Node.js 20	Runtime
TypeScript	Tipado estático
Express	Framework web
Prisma	ORM
PostgreSQL	Base de datos
JWT	Autenticación
Zod	Validación
bcrypt	Hash de contraseñas
Docker	Containerización

Inicio rápido

1. Configurar variables de entorno

```
cp .env_example .env
# El archivo .env ya está listo para usar
```

2. Levantar servicios con Docker

```
docker compose up -d
```

Servicios disponibles:

- Backend: http://localhost:3500
- pgAdmin: http://localhost:3502
- PostgreSQL: http://localhost:3501

3. Ejecutar migraciones y seed

```
# Migraciones
docker compose exec backend npx prisma migrate dev

# Generar Prisma Client
docker compose exec backend npx prisma generate

# Seed (datos iniciales)
docker compose exec backend npm run prisma:seed
```

4. Verificar que funciona

```
curl http://localhost:3500/health
```

Credenciales del seed:

- Admin: `admin@dwec.com` / `admin123`
- User: `user@dwec.com` / `user123`

Endpoints principales

Autenticación

```
POST   /api/auth/register    - Registrar usuario
POST   /api/auth/login       - Iniciar sesión
GET    /api/auth/me          - Obtener usuario actual (protegido)
```

Tareas (protegido con JWT)

```
GET    /api/tasks            - Listar tareas (paginación, búsqueda)
GET    /api/tasks/:id        - Obtener tarea por ID
POST   /api/tasks            - Crear tarea
PUT    /api/tasks/:id        - Actualizar tarea
DELETE /api/tasks/:id        - Eliminar tarea
```

Admin (solo ADMIN)

```
GET    /api/admin/users      - Listar usuarios
GET    /api/admin/stats      - Estadísticas del sistema
```

General

```
GET    /health               - Healthcheck
```

Uso desde React

Login

```javascript
async function login(email, password) {
  const response = await fetch("http://localhost:3500/api/auth/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ email, password }),
  });

  const data = await response.json();

  if (data.ok) {
    localStorage.setItem("token", data.data.token);
    return data.data;
  } else {
    throw new Error(data.error.message);
```

```
  }
}
```

Peticiones protegidas

```
async function getTasks() {
  const token = localStorage.getItem("token");

  const response = await fetch("http://localhost:3500/api/tasks", {
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });

  const data = await response.json();
  return data.ok ? data.data : null;
}
```

Estructura del proyecto

```
backend-dwec-prisma-jwt/
├── src/
│   ├── app.ts                 # Configuración Express
│   ├── server.ts              # Servidor
│   ├── config/                # Configuración (env, prisma)
│   ├── middlewares/           # Auth, validación, errores
│   ├── utils/                 # Utilidades (AppError, response)
│   └── modules/               # Módulos funcionales
│       ├── auth/             # Autenticación JWT
│       ├── tasks/            # CRUD tareas
│       ├── admin/            # Rutas administrativas
│       └── template-resource/ # Plantilla para nuevos recursos
├── prisma/
│   ├── schema.prisma         # Modelos de datos
│   └── seed.ts               # Datos iniciales
├── documentacion/            # Documentación completa
├── insomnia/                 # Colección de peticiones
├── docker-compose.yml        # Orquestación de servicios
├── Dockerfile                # Imagen del backend
└── README.md                 # Este archivo
```

Documentación

Toda la documentación está en la carpeta `/documentacion`:

1. **00-overview.md** - Arquitectura y visión general
2. **01-setup.md** - Instalación paso a paso
3. **02-auth.md** - Autenticación JWT
4. **03-crud-tasks.md** - CRUD de tareas
5. **04-how-to-create-a-new-resource.md** - Crear nuevos recursos
6. **05-troubleshooting.md** - Solución de problemas

Insomnia

Importa la colección desde `insomnia/insomnia-collection.json` para probar todos los endpoints.

Ver insomnia/README.md para instrucciones.

Comandos útiles

```
# Docker
docker compose up -d          # Levantar servicios
docker compose down           # Detener servicios
docker compose logs -f        # Ver logs
docker compose restart backend # Reiniciar backend

# Prisma
docker compose exec backend npx prisma studio  # Abrir Prisma Studio
docker compose exec backend npx prisma migrate dev # Nueva migración
```

Autor

Isaías Fernández Lozano

- Email: ifernandez@ieshlanz.es
- GitHub: @isaiasfl
- Módulo: **DWEC** (Desarrollo Web en Entorno Cliente)
- Centro: IES Hermenegildo Lanz
- Fecha: Febrero 2026

Licencia

```
npm run prisma:generate       # Generar cliente
npm run prisma:migrate        # Crear/aplicar migraciones
npm run prisma:seed           # Ejecutar seed
npm run prisma:studio         # Abrir Prisma Studio

# Desarrollo
npm run dev                   # Modo desarrollo
npm run build                 # Compilar TypeScript
npm start                     # Ejecutar compilado
```

Crear un nuevo recurso

1. Añade el modelo en `prisma/schema.prisma`
2. Ejecuta `npx prisma migrate dev`
3. Copia `src/modules/template-resource` con nuevo nombre
4. Adapta los archivos al nuevo recurso
5. Registra las rutas en `src/app.ts`

Ver documentacion/04-how-to-create-a-new-resource.md para guía completa.

Acceso a pgAdmin

1. Abre http://localhost:3502
2. Login: `isaias@dwec.com` / `dwec-2026`
3. Añadir servidor:
   - Host: `postgres`
   - Port: `5432`
   - User: `dwec_user`
   - Password: `dwec_password`
   - Database: `dwec_db`

Cambiar puertos

Si los puertos están ocupados, edita `.env` y `docker-compose.yml`:

Backend (3500):

```
# docker-compose.yml
backend:
  ports:
    - "NUEVO_PUERTO:NUEVO_PUERTO"

# .env
PORT=NUEVO_PUERTO
```

PostgreSQL (3501) y pgAdmin (3502):

```
# docker-compose.yml
postgres:
  ports:
    - "NUEVO_PUERTO:5432"

pgadmin:
  ports:
    - "NUEVO_PUERTO:80"
```

Desarrollo local (sin Docker para backend)

```
# Instalar dependencias
npm install

# Asegúrate de que PostgreSQL esté corriendo
# y cambia DATABASE_URL en .env a localhost:3501

# Ejecutar migraciones
npm run prisma:migrate
npm run prisma:generate
npm run prisma:seed

# Iniciar servidor
npm run dev
```

Seguridad

- Helmet para headers seguros
- CORS configurado
- Rate limiting (100 req/15min)
- Validación estricta con Zod
- Passwords hasheados con bcrypt
- JWT con expiración
- Ownership de recursos

Licencia

MIT

Autor

DWEC - Proyecto educativo para alumnos de React