Proyecto TITE: Grupo 4 - Autenticación y Perfiles.

Backend desarrollado con NestJS y MongoDB para el template GPI de la Universidad de Valparaíso. Expone una API RESTful para autenticación y gestión de usuarios, pensada para integrarse con un frontend React.

Tecnologías principales

- NestJS + TypeScript
- MongoDB con Mongoose
- JWT y Passport para autenticación
- Class Validator y bcrypt

Puesta en marcha rápida

1) Instala dependencias: `pnpm install`

2) Configura variables en `.env`: Estos .env ,**en caso de no estar creado**, debe crearse uno tanto en el front como el back y debe aplicarse esto en su interior:

.ENV FRONT:

VITE_API_URL=http://localhost:3000/api

VITE_RECAPTCHA_SITE_KEY=6LempAIsAAAAAG67RdIhapeVuFuzEgV70L5ukLGp

.ENV BACK:

NODE_ENV=development

PORT=3000

MONGODB_URI=mongodb://localhost:27017/gpi_database

JWT_SECRET=EstoEsUnSecretoSuperSeguroParaElCursoGPI

JWT_EXPIRES_IN=1d

VITE_API_URL=http://localhost:3000/api

RECAPTCHA_V2_SECRET_KEY=6LempAIsAAAAAIeCdDlrMBDKweYSN4smgsmV8Vfd

GOOGLE_CLIENT_ID=230221172309-knmlopi70nkkee47k4rnbhtdbkaed4r3.apps.googleusercontent.com

GOOGLE_CLIENT_SECRET=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

SMTP_HOST=smtp.gmail.com

SMTP_PORT=587

SMTP_SECURE=false

SMTP_USER=testpulgashopuv@gmail.com

SMTP_PASSWORD=bmdz zwwt dsxl csgt

EMAIL_FROM="PulgaShop <no-reply@pulgashop.cl>"

FRONTEND_BASE_URL=http://localhost:5173

FRONTEND_REGISTER_PATH=/auth/register

PASSWORD_RESET_TOKEN_EXPIRES_IN=30m

3) Inicia MongoDB y levanta el servidor: `pnpm start:dev` (por defecto en `http://localhost:3000/api`).

Datos útiles

- Backup de la base: `backups/gpi_dump` (usa `mongorestore --drop ... ./backups/gpi_dump/gpi_database`).

Comandos: Drop de databases

mongorestore --drop --uri "mongodb://localhost:27017/gpi_database" --nsInclude "gpi_database.*" ".\backups\gpi_dump\gpi_database"

Y para comprobar aplicacion:
(Para mostrar las colecciones y confirmar):
mongosh "mongodb://localhost:27017/gpi_database" --quiet --eval "show collections"
(Para mostrar un usuario random entre los que hay (hacerlo hasta ver que sale Admin1234Admin)):
mongosh "mongodb://localhost:27017/gpi_database" --quiet --eval "db.usuarios.findOne()"

- Usuario de prueba tras restaurar:

Credenciales:

ROL	CORREO	CONTRASEÑA
Admin	admin@admin.com	Admin1234
Admin	admindos@admin.com	Admin21234
Moderador	moderador@mod.com	mod12345
Moderador	moderador2@mod.com	mod21234
Vendedor	vendedor@yopmail.com	venuno1234
Vendedor	vendedordos@yopmail.com	vendos1234
Cliente	oscarastudillo@yopmail.com	oscar12345
Cliente	marianati@yopmail.com	marianatasha12

- Scripts de producción: `pnpm build` y `pnpm start:prod`.

Autenticación y uso de token

- Flujos soportados: credenciales (`/api/auth/login`), registro (`/api/auth/register`) y Google OAuth (`/api/auth/google` → callback).

- El login devuelve un `accessToken`. Guárdalo en memoria (no en localStorage si quieres evitar XSS) y envíalo en cada petición protegida con `Authorization: Bearer <token>`.

- El backend deriva el usuario desde el token; no enviamos el ID en cuerpo para rutas protegidas.

- Rutas que requieren bearer según el Swagger: usuarios y roles (create/list/get/update/delete). Para trabajar seguro, usa el token en TODAS las rutas salvo `login`, `register` y `google`/`google/callback`.

- Ejemplo con token:

```
curl -H "Authorization: Bearer $TOKEN" http://localhost:3000/api/users
```

Endpoints principales:

Auth:

- **POST** /api/auth/register: *Crea un nuevo usuario en el sistema.*
- **POST** /api/auth/login: *valida credenciales y entrega un JWT de acceso*.
- **GET** /api/auth/me: *devuelve los datos del usuario autenticado mediante el token.*
- **GET** /api/auth/google: *redirige al flujo de autenticación de Google OAuth*.
- **GET** /api/auth/google/callback: *recibe el callback de Google, valida el login y genera el token*.

Usuarios *(Requiere token; algunos solo admin/permisos):*

- **POST** /api/users: *Crea un nuevo usuario manualmente (administración).*
- **GET** /api/users: *Lista todos los usuarios*.
- **GET** /api/users/{id}: *Obtiene los datos de un usuario específico.*
- **PATCH** /api/users/{id}: *Actualiza los datos de un usuario.*
- **DELETE** /api/users/{id}: *Elimina un usuario.*

Roles:

- **POST** /api/roles: *Crea un nuevo rol.*
- **GET** /api/roles: *Lista todos los roles.*
- **GET** /api/roles/{id}: *Obtiene un rol por ID.*
- **PATCH / DELETE** /api/roles/{id}: *Actualiza un rol existente / elimina un rol.*

Permisos:

- **POST** /api/permisos: *Crea un permiso nuevo*.
- **GET** /api/permisos: *Lista todos los permisos disponibles.*
- **GET** /api/permisos/{id}: *Obtiene un permiso por ID*.
- **PATCH / DELETE** /api/permisos/{id}: *Actualiza un permiso / elimina un permiso.*

Verificaciones de correo:

- **POST** /api/verificaciones-correo: *Genera una solicitud de verificación de correo*
- **GET** /api/verificaciones-correo: *Lista verificaciones creadas.*
- **GET** /api/verificaciones-correo/{id}: *Consulta una verificación por ID.*
- **PATCH / DELETE** /api/verificaciones-correo/{id}: *Actualiza una verificación / elimina una verificación.*

Vendor Accreditations *(requiere token; solo admin para listar/eliminar):*

- **POST** /api/vendor-accreditations: *Envía solicitud de acreditación de vendedor.*
- **GET** /api/vendor-accreditations: *Lista todas las solicitudes de acreditación.*
- **DELETE** /api/vendor-accreditations/{id}: *Elimina una acreditación*.