Planificador UCN (NestJS + React + Mongo)

Tecnologias

- Backend: NestJS + Mongoose + Axios + Swagger opcional + Helmet + rate limit
- Frontend: React + Vite + Tailwind CSS
- Base de datos: MongoDB
- Integraciones: APIs UCN (Login, Malla, Avance) con stubs y respaldos CSV/JSON

Requisitos

- Node.js 20+
- MongoDB 7+ (local o contenedor Docker)

Puesta en marcha rapida

1. Clona el repo y entra a la carpeta `yio Malla proyecto`.
2. Instala dependencias de la raiz (necesarias para los scripts del monorepo): `npm install`.
3. Instala dependencias del backend y del frontend en un solo paso: `npm run bootstrap`.
4. Copia los archivos de entorno base (solo lo hace si faltan): `npm run setup`.
5. Ajusta los valores generados en `backend/.env` y `frontend/.env` si lo necesitas.
6. Asegura que MongoDB este corriendo (`mongodb://localhost:27017/planificador` por defecto) o usa Docker (`docker compose up --build`).
7. Levanta los servicios:
    - Backend: `npm run dev:backend`
    - Frontend: `npm run dev:frontend`
    - Ambos al mismo tiempo: `npm run all:dev`

> Tip: el backend expone Swagger en `http://localhost:3000/api/docs` cuando `SWAGGER_ENABLED=true`.

Configuracion de entorno

Backend

`backend/.env.example` ya viene con valores pensados para desarrollo offline (`USE_STUBS=true`). Asegurate de definir:

- `MONGO_URI`: cadena de conexion (por defecto local).
- `ALLOWED_ORIGINS`: agrega URLs adicionales de frontends si Vite usa otro puerto.
- `ADMIN_API_KEY`: clave para los endpoints protegidos de respaldo y oferta.
- `USE_BACKUP_FALLBACK=true` si quieres servir respaldos cuando las APIs UCN fallen.

Frontend

`frontend/.env.example` apunta al backend local (`VITE_API_BASE=http://localhost:3000/api`). Ajusta si cambiaste puerto o URL.

Ejecutar con Docker

- `docker compose up --build`
- Frontend: `http://localhost:8080`
- Backend: `http://localhost:3000`

Uso (UI)

1. **Login demo**
   - Ingresa correo y password (ver credenciales demo). Se listan las carreras/catalogos asociados.
   - "Olvide mi contrasena" valida el RUT y devuelve un mensaje (no envia correos reales).
   - Acceso administrador disponible desde la misma pantalla de login.
2. **Plan (/plan)**
   - Define tope de creditos y genera la seleccion principal (prioriza reprobados, luego atrasados segun nivel objetivo y luego prioritarios, respetando prerrequisitos).
   - Desde la vista puedes marcar prioritarios y generar opciones alternativas.
3. **Mis proyecciones (/proyecciones)**
   - Lista proyecciones guardadas, expande la malla, marca favoritas, renombra o elimina.
4. **Demanda (/demanda)**
   - Muestra demanda agregada por codigo o por NRC (solo desde favoritas).
5. **Admin / oferta**
   - En `/admin` ingresa la cabecera `X-ADMIN-KEY` configurada para cargar oferta CSV en `/oferta`.

Credenciales demo

- Estudiante 1: rut `333333333`, email `juan@example.com`, password `1234` (ICCI 201610)
- Estudiante 2: rut `222222222`, email `maria@example.com`, password `abcd` (ITI 202410)
- Estudiante 3: rut `111111111`, email `ximena@example.com`, password `qwerty` (ICCI 201610)

API destacado

- Health: `GET /api/health`
- Auth (demo): `POST /api/auth/login`, `POST /api/auth/forgot`
- UCN (proxy): `GET /api/ucn/malla/:cod/:catalogo`, `GET /api/ucn/avance?rut&codcarrera`
- Proyecciones: generar / generar-opciones / guardar / favorita / renombrar / borrar / demanda
- Admin: carga de oferta CSV y respaldos UCN (requiere `X-ADMIN-KEY`)

Comandos para resolver problemas comunes

- Utilizar comando:
  copy backend.env.example backend.env
- Cambiar el .env de backend a USE_STUBS=true
- Levantar de nuevo:
  docker compose down
  docker compose up --build

Para test:

- Utilizar comando
  cd backend
  npm install
  npm test