Planificador UCN — Backend (NestJS) + Frontend (React + Vite)

Descripción general

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

Requisitos

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

Configuración

Backend

1. Copiar `backend/.env.example` a `backend/.env` y ajustar:
   - `MONGO_URI`: cadena de conexión a Mongo
   - `ALLOWED_ORIGINS`: por defecto `http://localhost:5173,http://localhost:8080`
   - `USE_STUBS=true` para trabajar offline con datos de ejemplo
   - `ADMIN_API_KEY`: clave secreta para endpoints de administración
   - `USE_BACKUP_FALLBACK=true` para servir respaldos cuando UCN no responda
   - `SWAGGER_ENABLED=true` para exponer `/api/docs` en desarrollo

Frontend

1. Copiar `frontend/.env.example` a `frontend/.env` y ajustar si es necesario:
   - `VITE_API_BASE=http://localhost:3000/api`

Ejecución en desarrollo

Opción A (local)

- Backend: `cd backend && npm install && npm run start:dev`
- Frontend: `cd frontend && npm install && npm run dev` → abrir `http://localhost:5173`
- Desde la raíz del repo (ambos en paralelo):
  - Primera vez: `npm install` (para instalar `concurrently` en el root)
  - Ejecutar: `npm run all:dev`

Opción B (Docker stack completo)

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

- Backend: `http://localhost:3000`
- Mongo: `mongodb://localhost:27017`

Swagger

- **Dev:** `http://localhost:3000/api/docs`
- **Producción:** deshabilitado por defecto (controlado por `SWAGGER_ENABLED`)

Uso de la aplicación (UI)

- Abrir `http://localhost:5173` (o `http://localhost:8080` con Docker Compose)

Login

- Con `USE_STUBS=true`, ejemplos de credenciales: `juan@example.com` / `1234`
- Elegir carrera + catálogo de la lista

Generar proyección (`/plan`)

- Definir tope de créditos (ej. 22)
- Opcional: definir período (ej. `202520`) y usar “Con Oferta” si cargaste CSV de oferta
- Revisar selección; guardar y/o marcar como favorita

Mis proyecciones (`/proyecciones`)

- Listar historial, marcar favorita (confirma reemplazo), y borrar (con confirmación)

Demanda (`/demanda`)

- Ver demanda agregada por código de curso o por NRC (solo favoritas)

Oferta (`/oferta`)

- Primero ir a `/admin` e ingresar tu `X-ADMIN-KEY` (debe coincidir con `ADMIN_API_KEY` del backend)
- Subir CSV (cabecera: `period,nrc,course,codigoparalelo,dia,inicio,fin,sala,cupos`) y luego listar ofertas

Funcionalidades

- Prioriza ramos reprobados y respeta prerrequisitos y tope de créditos
- Permite guardar múltiples proyecciones y marcar una favorita
- Selecciona NRCs de la oferta evitando choques de horario
- Calcula demanda agregada por curso o NRC a partir de favoritas
- Integración con APIs de UCN (login, malla, avance), con stubs para demo y respaldos CSV/JSON con fallback
- Carga de oferta CSV (admin) para potenciar la selección de NRCs

Arquitectura y flujo

Backend (NestJS)

- **domain:** reglas de negocio (ProjectionService arma la selección; ScheduleService revisa choques de horario)
- **application:** casos de uso (GenerateProjection*, etc.)
- **infra:**
  - **db:** esquemas y repositorios Mongoose (proyecciones, ofertas) con índices útiles
  - **ucn:** gateways HTTP a UCN y controladores de respaldo admin (seguros con X-ADMIN-KEY)
- **web:** controladores HTTP (ofertas, proyecciones, health, proxies ucn)

Estrategia de datos

- Malla y avance desde UCN; con `USE_STUBS=true` se usan datos de demo
- Si UCN falla y `USE_BACKUP_FALLBACK=true`, se sirven respaldos desde Mongo
- Oferta proviene de CSV, guardada en Mongo, indexada por `course` y `period`

Seguridad backend

- CORS controlado por `ALLOWED_ORIGINS`, Helmet, límites globales y específicos en `/api/ucn`
- Validación estricta de DTOs (ValidationPipe + class-validator)
- Endpoints admin requieren `X-ADMIN-KEY`

Frontend (React + Vite + Tailwind)

- Rutas: Login, Plan, Proyecciones, Demanda, Oferta, Admin, 404
- Estado global persistente (rut, carreras, selección, tope, período, adminKey)
- UX: toasts, confirmaciones, guards de rutas por rut y adminKey
- **Dev:** Vite proxy `/api` → backend
- **Prod (Docker):** Nginx sirve la SPA y proxya `/api`

Resumen de API

- **Health:** `GET /api/health` (incluye `mongo.readyState`)
- **UCN (proxy):** `GET /api/ucn/login`, `GET /api/ucn/malla/:cod/:catalogo`, `GET /api/ucn/avance?rut&codcarrera`
- **Proyecciones:** generar/guardar/listar/favorita/borrar/demanda
- **Oferta CSV (admin):** `POST /api/oferta/cargar` (requiere `X-ADMIN-KEY`), `GET /api/oferta/listar`
- **Respaldos (admin):** `POST/GET /api/ucn/respaldo/malla`, `POST/GET /api/ucn/respaldo/avance` (requieren `X-ADMIN-KEY`)

Problemas comunes

- IDE muestra “module not found” en módulos locales: reinicia TS Server (VS Code → “TypeScript: Restart TS Server”)
- Mongo no disponible: revisar `MONGO_URI` o usar `docker compose`
- CORS: ajustar `ALLOWED_ORIGINS`
- UCN caída: habilitar `USE_STUBS=true` y/o cargar respaldos con `USE_BACKUP_FALLBACK=true`

Licencia

- Proyecto académico. Adaptar según sea necesario