Manual de Código — Biblioteca Virtual

Este manual documenta la arquitectura, estructura, rutas de API, flujos, variables de entorno y procedimientos para ejecutar, desarrollar y desplegar el proyecto de la Biblioteca Virtual, desarrollado por los alumnos de PRÁCTICAS PROFESIONALIZANTES III, a cargo del profesor David Mauro, durante el año 2025: Alvarez Misael, Di Blasio Facundo, Laddaga Bárbara, Madonia Luca y Spinetto Eduardo.

Manual de Código — Biblioteca Virtual 1
1) Visión general 3
2) Estructura del repositorio

Estructura 3
3) Requisitos y versiones 3
4) Variables de entorno 3
5) Puesta en marcha 4
Con Docker 4
Dev sin Docker 4
6) CORS, cookies y auth 4
7) Rutas del Frontend (React Router) 4
8) API del Backend 5
Auth (/api/auth) 5
Documents (/api/documents) 5
Folders (/api/folders) 5
News (/api/news) 5
9) Modelo de datos 6
10) Subidas (Multer) 6
11) calidad y seguridad 6
12) Despliegue 6
13) Mantenimiento 6

1) Visión general

Objetivo: Sitio web para gestionar material académico por **carrera, año** y **materia,** con módulo de **noticias, gestión de usuarios**(para admins) y **carnet de estudiante imprimible.**

Arquitectura: Frontend (Vite + React Router) → Backend (Express 5 + Sequelize) → DB MySQL 8.0.19. Orquestación con **Docker Compose** (redes public/private, volumen db-data y secreto db-password).

Cron & Mailer: El backend ejecuta startCronCheckUp() y verifica nodemailer al iniciar.

2) Estructura del repositorio

Estructura

3) Requisitos y versiones

- **Node.js: LTS 18+**
- **NPM: 9+**
- **Docker** y **Docker Compose**: actuales
- **Base de datos:** MySQL 8.0.19

4) Variables de entorno

```
DB_USER = 'root'
DB_PASSWORD = ''
DB_DATABASE = 'digital_library'
DB_SERVER = 'localhost'
DB_PORT = 3306
JWT_SECRET=x92n13h8vjasdu9vh1923hfa9!@#%^ABCD
GMAIL_USER: bibliotecadigitalisfdyt2@gmail.com
GMAIL_PASS: qkva tzob kfbm mjkp
```

GMAIL_FROM: Biblioteca Digital
<bibliotecadigitalisfdyt2@gmail.com>

5) Puesta en marcha

Con Docker

```
docker compose up -d --build
# Frontend: http://localhost:5173
# Backend : http://localhost:3000
```

Redes: public (frontend/backend), private (backend/db). Volumen: db-data.

Dev sin Docker

```
# Backend
cd backend
npm install
npm run dev

# Frontend
cd frontend
npm install
npm run dev
```

6) CORS, cookies y auth

- **Backend CORS**: origin: true, credentials: true, métodos GET,POST,PUT,DELETE,OPTIONS, headers Content-Type, Authorization.
- **Frontend**: mezcla cookies (credentials: "include") y header Bearer (localStorage.getItem("token")).
  Vite server.allowedHosts: i2azul.mooo.com

7) Rutas del Frontend (React Router)

En src/App.jsx usando <BrowserRouter>:

- / → ContentRenderer + Home
- /documentos → ContentRenderer + DocumentBarLayout + Documents + CreateDocument
- /noticias → ContentRenderer + News
- /login → ContentRenderer + LoginContent
- /signIn → ContentRenderer + SignInForm
- /activation → ContentRenderer + UserActivationTable

- /profile → ContentRenderer + ProfilePanel
- /gestor-noticias → ContentRenderer + NewsManager
- /admin → ContentRenderer + AdminPage
- /carnet → ContentRenderer + CarnetPage

Protecciones:
AdminPage verifica admin con POST {VITE_API_URL}/api/auth/tokenchk (credentials: "include") y redirige si no es admin.

8) API del Backend

Base URL: http://localhost:3000/api

Auth (/api/auth)

- POST /login — loginSchema → JWT/cookie.
- POST /register — registerSchema.
- GET /profile — **protegido** (chkToken, checkAdmin).
- GET /unactive — lista inactivos.
- PATCH /profile — **protegido** (chkToken, checkAdmin, patchUserSchema).
- PATCH /lowuser — **protegido** (chkToken, checkAdmin, activateUserSchema).
- DELETE /delete — **protegido** (chkToken, checkAdmin).
- DELETE /delete/:id — **protegido** (chkToken).
- POST /logout
- POST /tokenchk — **protegido** (chkToken).

Documents (/api/documents)

- GET /
- GET /find_tag/:tag
- GET /findByFolder/:folder_id
- POST /createDocument — **protegido** + multer.single('file').

Folders (/api/folders)

- GET /
- GET /parentsFolders
- GET /byParent/:parent_id

News (/api/news)

- POST / — **protegido** + upload.
- DELETE /delete — **protegido**.
- PATCH /update — **protegido** + upload.

- GET / — público.

9) Modelo de datos

SQL database

10) Subidas (Multer)

- config/multerConfig.js: diskStorage → uploads/.
- /api/news define propio storage → uploads/news.
- Servido estático en /uploads.

11) calidad y seguridad

- camelCase / PascalCase; REST plural.
- Validación: Joi (auth/news).
- JWT (cookie/bearer a unificar).
- Roles: checkAdmin, authorize.

12) Despliegue

- Backend Dockerfile (node:lts, EXPOSE 80 9229 9230, CMD node index.js).
- Compose: redes public/private, db-data, db-password.
- .env prod: JWT_SECRET, SMTP, CORS restringido, volumen persistente de uploads/.

13) Mantenimiento

Cualquier cambio en rutas, puertos, variables o DDL debe actualizar este documento.