Entrega 2 — VisiControl (Gestión de visitas penitenciarias)

Integrantes

- Kevin Adrián Gil Soto (@kevingS0712)
- Boris Guillermo Briones Dupla
- Fernando Moreno Mora
- Rafael Alejandro Ajila Gallegos

Carrera: Ingeniería de Software 2
Fecha: 14/10/2025
Repositorio Git: https://github.com/kevings0712/VisiControl

---

1) Objetivo de la entrega

Demostrar trabajo colaborativo y automatización (CI/CD) con:

- **2 funcionalidades operativas** (Login + Visitas).
- **Build pasando** en GitHub Actions (typecheck + lint + build).
- **Despliegue** en cloud (Render) con `/api/health` OK.

---

2) Flujo de trabajo con Git (GitHub Flow)

- **Rama por defecto:** `develop`.
- Trabajo en ramas: `feat/*`, `fix/*`, `chore/*`, `ci/*`.
- **Pull Requests** siempre hacia `develop`.
- **main protegida** (sin pushes directos; solo PR desde `develop` cuando se libere versión).

Reglas base de PRs

- Título claro (`feat:`, `fix:`, `chore:`, `ci:`).
- Descripción breve de cambios.
- CI en verde para poder mergear.

---

3) Evidencia de colaboración

- Commits con convención (`feat: …`, `fix: …`, `chore: …`, `ci: …`).
- PRs abiertos y revisados en https://github.com/kevings0712/VisiControl/pulls
- Workflows ejecutados y en verde en https://github.com/kevings0712/VisiControl/actions

---

4) CI/CD (GitHub Actions)

Workflow: `.github/workflows/ci.yml`

Job `backend` (resumen):

1. `actions/checkout@v4`
2. Node 20
3. **Install**: `npm ci` - `npm install`
4. **Typecheck**: `npm run typecheck`
5. **Lint**: `npm run lint`
6. **Build**: `npm run build`

> Resultado: el pipeline **frena PRs** cuando falla typecheck/lint/build.

---

5) Estructura del repo (monorepo)

```
VisiControl/
└─ src/
├─ backend/ # Node/Express/TypeScript
│ ├─ src/{config,controllers,routes,services,repositories,utils}
│ ├─ sql/{schema_users.sql, schema_visits.sql}
│ ├─ package.json
│ └─ .env.example
└─ frontend/ # Placeholder React/Vite/TS (futuro)
```

6) Correr backend en **local**

> **Requisitos**: Node 20, PostgreSQL, `psql`, `jq` (en Mac: `brew install jq`).

```
git clone https://github.com/kevings0712/VisiControl.git
cd VisiControl/src/backend
cp .env.example .env
# Edita .env con tus credenciales locales.
.env ejemplo (ajusta a tu instalación local):
NODE_ENV=development
PORT=4000
CORS_ORIGIN=*

PGHOST=127.0.0.1
PGPORT=5433
PGDATABASE=visicontrol
PGUSER=visictrl_admin
PGPASSWORD=visipass

JWT_SECRET=supersecret
JWT_EXPIRES_IN=1d
```

```
#Inicializar DB (tablas):
psql -h "$PGHOST" -p "$PGPORT" -U "$PGUSER" -d "$PGDATABASE" -f
sql/schema_users.sql
psql -h "$PGHOST" -p "$PGPORT" -U "$PGUSER" -d "$PGDATABASE" -f
sql/schema_visits.sql


#Seed del admin
psql -h "$PGHOST" -p "$PGPORT" -U "$PGUSER" -d "$PGDATABASE" -c \
"INSERT INTO users (name, email, password_hash, role)
 VALUES ('Administrador', 'admin@visicontrol.dev',
 '$2b$10$MSqt1pmN7e04QRVX2Gn7D0C8lNgMBSBjdQ2oXouMBTDrwUewfENh6', --
Admin123!
 'ADMIN')
 ON CONFLICT (email) DO NOTHING;"

#Levantar API
npm run dev

#Healthcheck
curl -s http://localhost:4000/api/health
```

7) Pruebas manuales de funcionalidades

```
#A) Localhost (http://localhost:4000) API corriendo npm run dev
LOGIN
curl -s -X POST http://localhost:4000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@visicontrol.dev","password":"Admin123!"}'

Capturar token
TOKEN=$(curl -s -X POST http://localhost:4000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@visicontrol.dev","password":"Admin123!"}' | jq -r
'.token')
echo "TOKEN=${TOKEN:0:40}..."

GET vistias
curl -s -H "Authorization: Bearer $TOKEN" \
  http://localhost:4000/api/visits | jq .

POST visita
curl -s -X POST http://localhost:4000/api/visits \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"visitor_name":"Kevin Gil","inmate_name":"Maholy
Mera","visit_date":"2025-10-
25","visit_hour":"18:15:00","status":"PENDING","notes":"Primera visita:
```

```
visita conyugal"}' | jq .
```

8) Rutas clave (backend)

- GET /api/health – Sanidad del servicio.
- POST /api/auth/login – Login con JWT (staff).
- GET /api/visits – Lista visitas (requiere Authorization: Bearer).
- POST /api/visits – Crea visita (requiere JWT).

s9) Estado de funcionalidades (entrega)

- Login (JWT) funcionando (local y Render).
- Visitas: listar y crear funcionando (local y Render).
- CI en verde: typecheck + lint + build.

10) Enlaces

Repositorio: https://github.com/kevings0712/VisiControl

Pull Requests: https://github.com/kevings0712/VisiControl/pulls

Actions (CI): https://github.com/kevings0712/VisiControl/actions

URL Render (API): https://visicontrol.onrender.com