SSO "desde cero" súper sencillo (con este repo)

Demo educativa de SSO con Node.js + Express. Incluye un IdP minimalista y un Cliente que confía en él, organizados en `idp/` y `client/`.

No es seguro para producción. Sirve para entender el flujo Authorization Code y el concepto de sesión única.

0) Requisitos

- Node.js 18+ (ideal 20+)
- Puertos libres: 4000 (IdP) y 3000 (Cliente)
- Opcional para despliegue: UFW (firewall) y una VM Linux

1) Estructura del proyecto

- `idp/`
  - `server.js` (app Express + sesión)
  - `routes.js` (rutas `/login`, `/authorize`, `/token`, `/userinfo`, `/me`, `/logout`)
  - `tokenService.js` (firma de JWT con `jose`)
  - `config.js` (carga `.env`, `BASE_URL`, `JWT_SECRET`, `PUERTO`)
  - `storage.js` (usuarios y códigos en memoria)
  - `.env` (incluye valores de ejemplo)
- `client/`
  - `server.js` (app Express + cookie `cliente.sid`)
  - `src/rutas.js` (home, login, callback, dashboard, me, logout, api/perfil)
  - `src/configuracion.js` (carga `.env` y puertos)
  - `src/middlewares/*`, `src/servicios/*`, `src/vistas/*`
  - `.env` (incluye valores de ejemplo)

2) Variables de entorno

- `idp/.env`:
  - `BASE_URL=http://127.0.0.1:4000`
  - `JWT_SECRET=super-secreto-solo-lab`
- `client/.env`:
  - `IDP_BASE=http://127.0.0.1:4000`
  - `BASE_URL=http://127.0.0.1:3000`

Para exponer la demo fuera de la VM, usa la IP real de la VM:

- `idp/.env` → `BASE_URL=http://<IP_VM>:4000`
- `client/.env` → `IDP_BASE=http://<IP_VM>:4000` y `BASE_URL=http://<IP_VM>:3000`

3) Instalación y ejecución

Desde la raíz del repo:

```
npm install          # instala dev deps (p. ej., concurrently)
cd idp && npm install
cd ../client && npm install
```

Puedes arrancar por separado o en paralelo:

- Solo IdP: `npm run start:idp` (raíz) o `cd idp && npm start`
- Solo Cliente: `npm run start:client` (raíz) o `cd client && npm start`
- Ambos a la vez: `npm start` (raíz) → usa `concurrently`

Direcciones por defecto:

- IdP: `http://127.0.0.1:4000`
- Cliente: `http://127.0.0.1:3000`

Usuario demo del IdP: `demo@lab.local` / `demo`.

4) Flujo de autenticación (cómo funciona)

1. Cliente (`/login`) genera `state` aleatorio y redirige a `IdP /authorize` con `client_id`, `redirect_uri`, `state`.
2. Si no hay sesión en el IdP, el usuario inicia sesión en `IdP /login`.
3. IdP emite un `code` efímero y redirige a `redirect_uri` (cliente) con `code` y `state`.
4. Cliente valida `state` (guardado en sesión) y canjea `code` en `IdP /token` → obtiene `access_token` (JWT HS256).
5. Cliente guarda el token en su sesión y protege rutas (`/dashboard`, `/me`, `/api/perfil`). El cliente solo decodifica el JWT para mostrar datos (no valida firma; simplificado para educación).

Endpoints principales del IdP:

- `GET /login` (form), `POST /login`
- `GET /authorize` → redirección con `code` y `state`
- `POST /token` → `{ access_token, token_type, expires_in }`
- `GET /userinfo` (JSON), `GET /me` (HTML)
- `GET /logout`

5) Despliegue en Ubuntu Server (VM) y acceso externo

1. Red de VM: usa modo Puente (Bridged) para IP propia o configura NAT con port-forward (3000 y 4000).
2. Instala Node 20+:

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt -y install nodejs
node -v && npm -v
```

3. Clona el repo e instala dependencias:

```
sudo apt -y install git
git clone <URL_DEL_REPO>
cd sso
npm install
cd idp && npm install
cd ../client && npm install
```

4. Ajusta `.env` con IP de la VM (ver sección 2).
5. Abre el firewall (UFW):

```
sudo ufw allow 3000/tcp
sudo ufw allow 4000/tcp
sudo ufw enable   # si no estaba activo
sudo ufw status
```

6. Arranca ambos: `npm start` (raíz). Accede desde tu equipo: `http://<IP_VM>:3000`.
7. Opcional servicios con PM2:

```
sudo npm i -g pm2
pm2 start idp/server.js --name sso-idp
pm2 start client/server.js --name sso-client
pm2 save
pm2 startup systemd
```

6) Pruebas

El repo incluye pruebas para el IdP en `tests/idp/*` (usa `node:test` y `supertest`). Desde la raíz:

```
npm install
node --test
```

7) Solución de problemas

- No se ve desde fuera: verifica modo de red (Bridged/port-forward), IP de la VM, puertos abiertos (UFW) y que los procesos estén escuchando (`ss -tulpn | grep -E '3000|4000'`).
- Error de `state`: borra la cookie `cliente.sid` y reintenta el login.
- Puertos ocupados: ajusta `BASE_URL` en los `.env` o libera 3000/4000.
- Node antiguo: instala Node 18+ (ideal 20+) para `fetch` y `node:test`.

8) Seguridad (recordatorio)

- Demo educativa: no apta para producción.
- Sin HTTPS, sin PKCE, secreto compartido HS256, usuarios y códigos en memoria, cookies no seguras.

- Para producción: OIDC (p. ej., Keycloak), HTTPS, Authorization Code + PKCE, validación de tokens en backend, rotación de tokens, MFA, CSP, etc.