Sprint 1 — Disseny del sistema de control de visites

En aquest primer sprint hem plantejat com funcionaria un sistema real de control d'entrades per a un centre educatiu. L'objectiu era detectar automàticament quan una persona entra al centre, registrar l'hora i vincular-la amb les seues dades personals (com ara el nom i DNI). He proposat una arquitectura on la **Raspberry Pi** llig targetes NFC i envia la informació a una **API** que després guarda la visita en una base de dades.

També he fet un esquema funcional que mostra:

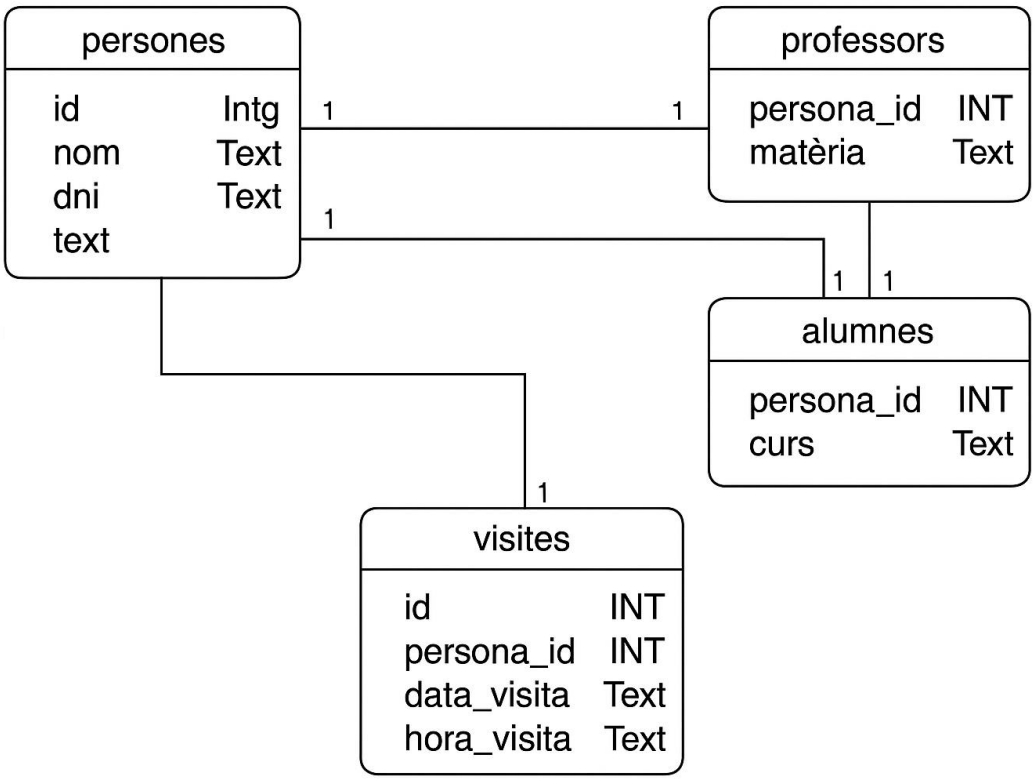
- Lector NFC → Raspberry Pi → API (Flask) → SQLite
- Consultes posteriors per veure qui ha entrat i quan.

Sprint 2 — Disseny avançat de la base de dades (amb rols)

Descripció:
En aquest sprint he redissenyat la base de dades per incloure informació diferenciada segons si la persona registrada és **professor** o **alumne**.
He creat una estructura jeràrquica:

- **persones**: conté la informació comuna a tots (nom, dni, rol).
- **professors**: taula associada només a les persones amb rol "professor", amb la matèria que imparteixen.
- **alumnes**: taula associada només a les persones amb rol "alumne", amb el seu curs escolar.

D'aquesta manera, mantinc les dades normalitzades, separant el que és comú del que és específic. Això també facilita consultes segons el rol.



Sprint 3 — Creació de la base de dades amb rols diferenciats

🔧 Codi SQL avançat (amb professors i alumnes)

```
-- Taula base de persones
CREATE TABLE persones (
id INTEGER PRIMARY KEY AUTOINCREMENT,
nom TEXT NOT NULL,
dni TEXT UNIQUE NOT NULL,
rol TEXT CHECK(rol IN ('alumne', 'professor')) NOT NULL
);


-- Taula per a alumnes
CREATE TABLE alumnes (
persona_id INTEGER PRIMARY KEY,
curs TEXT NOT NULL,
FOREIGN KEY (persona_id) REFERENCES persones(id)
);

-- Taula per a professors
CREATE TABLE professors (
```

```
persona_id INTEGER PRIMARY KEY,
materia TEXT NOT NULL, FOREIGN KEY (persona_id) REFERENCES persones(id)
);

-- Taula de visites
CREATE TABLE visites (
id INTEGER PRIMARY KEY AUTOINCREMENT,
persona_id INTEGER NOT NULL,
data_visita TEXT NOT NULL,
hora_visita TEXT NOT NULL,
FOREIGN KEY (persona_id) REFERENCES persones(id) );

-- Insercions de persones (amb rol)
INSERT INTO persones (nom, dni, rol) VALUES
('Jordi Andreu', '11111111A', 'professor'),
('Clara Requena', '22222222B', 'alumne'),
('Pau Soler', '33333333C', 'professor'),
('Laia Serra', '44444444D', 'alumne');

-- Informació específica de professors
INSERT INTO professors (persona_id, materia)
VALUES (1, 'Matemàtiques'), (3, 'Informàtica');

-- Informació específica d'alumnes
INSERT INTO alumnes (persona_id, curs)
VALUES (2, '1r Batxillerat A'),
(4, '2n ESO B');

-- Inserció de visites
INSERT INTO visites (persona_id, data_visita, hora_visita) VALUES (1, '2025-06-10',
'08:45'), (2, '2025-06-10', '08:50'), (3, '2025-06-10', '09:00'), (4, '2025-06-10', '09:05');
```

Desenvolupament de l'API de connexió

En aquest sprint hem fet un pas molt important: **crear una API funcional amb Python** que permet a la Raspberry Pi **connectar-se a la base de dades SQLite** i **registrar entrades en temps real**. He utilitzat el framework `Flask`, ja que és lleuger, fàcil d'aprendre i ideal per a petits projectes educatius com aquest.

S'han creat dues rutes principals:

- `GET /visites`: retorna totes les visites emmagatzemades a la base de dades, útil per comprovar què s'ha registrat.

- `POST /visites`: permet afegir noves visites enviant dades com l'ID de persona, la data i l'hora.

També he configurat correctament la connexió amb SQLite i gestionat possibles errors amb respostes JSON clares. Ara la Raspberry Pi podrà fer peticions POST cada vegada que detecti una targeta NFC, enviant la informació automàticament.

A més, he utilitzat Postman per provar tot abans de connectar la Raspberry. Això m'ha ajudat a entendre millor com funcionen les API REST i com es comuniquen els dispositius amb un servidor local.

Aprenentatge clau:
He après a construir un servei web real, a fer-lo segur i útil, i he descobert com aplicar Python fora dels scripts normals per a fer aplicacions pràctiques i útils.

Sprint 5 — Integració final amb Raspberry Pi i sistema real

En aquest últim sprint hem integrat tot el treball realitzat: **la Raspberry Pi, el lector NFC, l'API en Python i la base de dades SQLite**. El repte ha sigut fer que tot funcione automàticament quan una targeta és detectada.

He configurat la Raspberry perquè, cada vegada que llig una targeta NFC, execute un script en Python que:

1. Llig el codi UID de la targeta.
2. Identifica a quina persona correspon.
3. Envia una petició POST a la nostra API local amb la data i l'hora de l'entrada.
4. L'API insereix la visita en la base de dades.

He fet proves reals amb targetes NFC i el lector funcionant amb `libnfc` o `MFRC522`, depenent del model. També he inclòs missatges d'error útils i un log de registre per depurar fàcilment.

Aprenentatge clau:
Aquest sprint m'ha fet entendre el funcionament complet d'un sistema connectat: des del sensor físic fins al registre en una base de dades. M'ha motivat molt veure que el nostre projecte ja funciona com un sistema real. Ara sé muntar un petit servei tècnic des de zero!