OdontoCare

UOC Treball final Python C1

Gener 2026 by Carles Person

Arquitectura

Microserveis

CLEINT -> API (blueprints)

- ***auth_bp***: obtenim el token (user/pass)
- ***admin_bp***: gestionem usuaris (depenen del nostre rol). Obté el ROL del token.
- **agenda_bp:** gestionem cites. Funcionalitat limitada per usuari i/o rol.

API -> Auth_svc

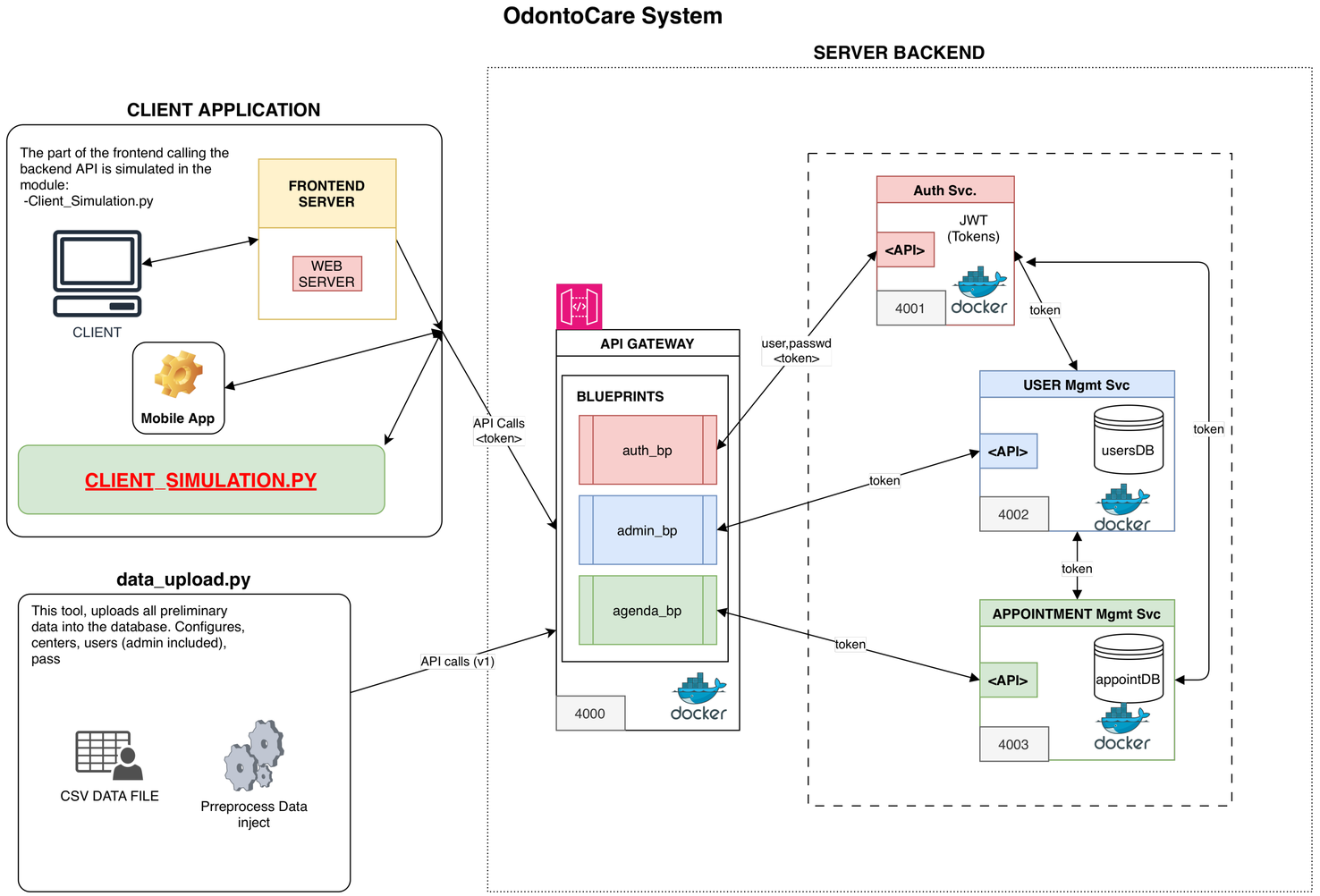
- obtenim el token (forward user/pass).

API -> Admin_svc: (user management)

- Forward sol·licituds per crear, destruir i comprovar usuaris.

API-> Agenda_svc: (agenda management)

- Forward sol·licituds per crear, modificar, eliminar entrades en agenda.



ROLS: ‘admin’, ‘assistant’, ‘patient’ i ‘doctor’

Persistència de Dades

SQLite / MySQL

MySQL:

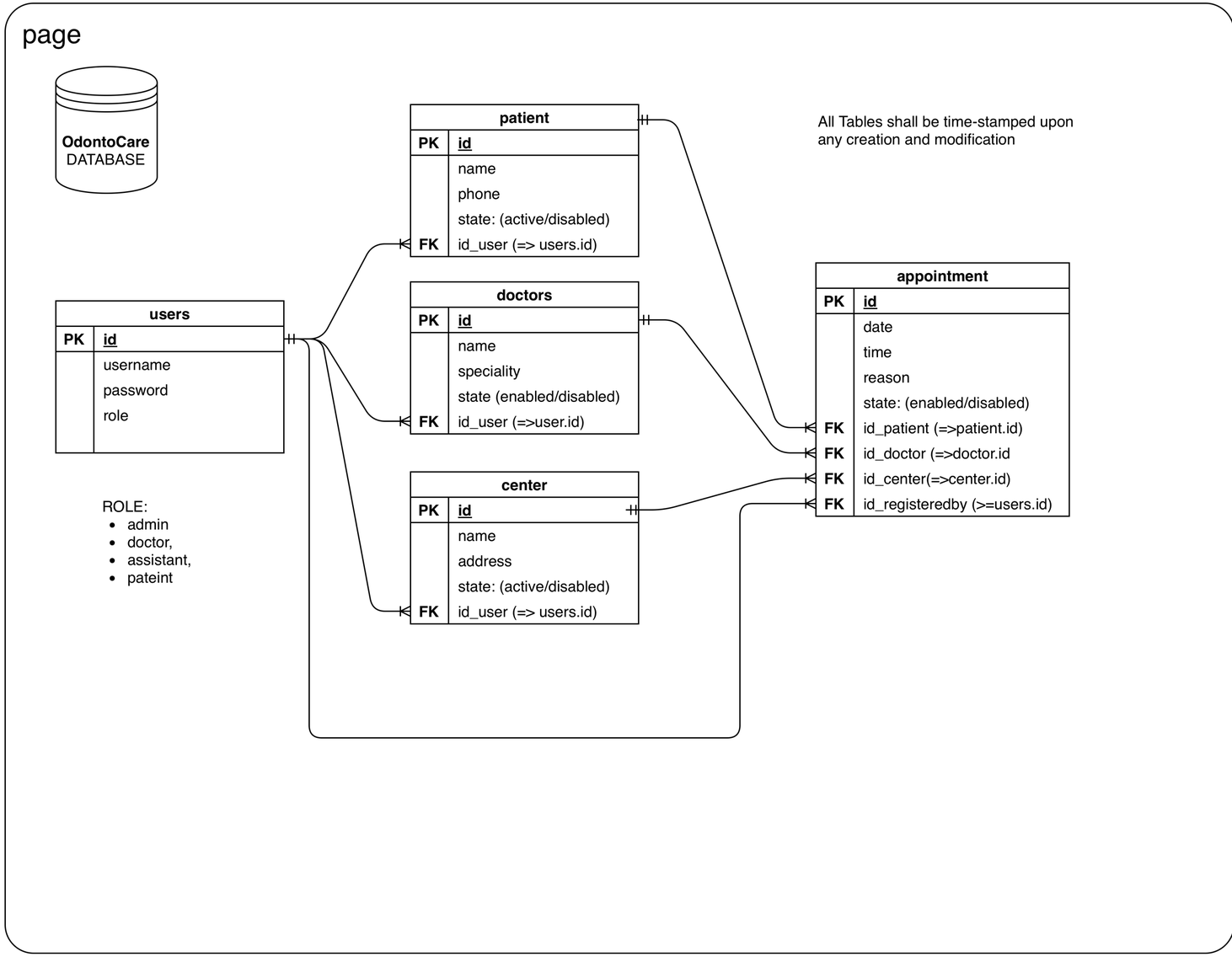
- Utilitzada durant desenvolupament per facilitat interacció amb les dades.
- una única base de dades

SQLite:

- DEMO final
- 3 fitxers bases de dades
  - auth.db (mòdul **auth_svc**)
  - admin.db (mòdul **admin_svc**)
  - agenda.db (mòdul **agenda_svc**)

Foreign keys:

- Al estar els 4 serveis en contenidors diferents, no es pot configurar les **FK** utilitzar en motor de DB.
- Gestionat internament cada servei



Consideracions

```python
@agenda_bp.route('/get', methods=['GET'])
@role_required([OCROL.ADMIN, OCROL.ASSISTANT, OCROL.DOCTOR], SECRET_KEY)
def get_appointments():
    token = request.headers.get('Authorization')
    header = {
        'content-type': "application/json",
        "Authorization": token
    }
```

- **Seguretat:**
  - Usuari només té access a la API pública (**api_svc**)
    - Contenidors: **auth_svc**, **admin_svc** i **agenda_svc** no son accessibles
  - API pública limita (implementat en els blueprints) tipus accés (següent slide)
  - Serveis, només requereixen “token” vàlid
- **Token**: generació amb “secret_key” que es randomitza en inicialitzar els contenidors
- **Password:** no-“hashed” a la DB, per facilitar “visualització” .

Consideracions

Seguretat ROL i TOKEN

api-server.py

blueprint: agenda_bp.py

```python
@agenda_bp.route('/get', methods=['GET'])
@role_required([OCROL.ADMIN, OCROL.ASSISTANT, OCROL.DOCTOR], SECRET_KEY)
def get_appointments():
    token = request.headers.get('Authorization')
    header = {
        'content-type': "application/json",
        "Authorization": token
    }
                    .
                    .
                    .
                    .
try:
    result = requests.get(f'{app.config['ADMIN_SVC']}/user/{OCENT.DOCTOR}/{id}', headers = request.headers)
    if result.status_code != HTTPStatus.OK:
        return json_error(HTTPStatus.INTERNAL_SERVER_ERROR, OCERR.ADMIN_SVC_ACCESS_ERROR)
```

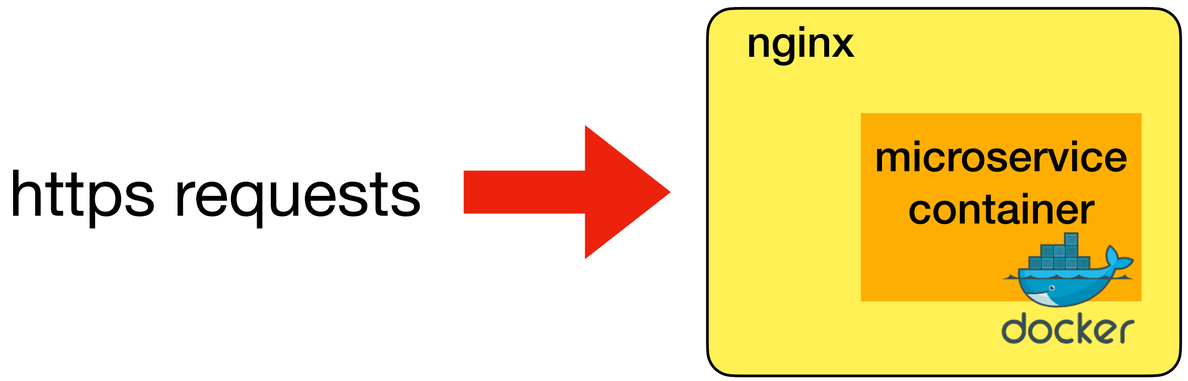
agenda-server.py

```python
@app.route('/get', methods=['GET'])
@token_required
def get_agenda():
    """…
```

Consideracions

- **Altres:**
  - Deployment: runtime Flask (no recomanable)
  - HTTPS: (no implementat). Recomanat per
    - Web Proxy (seguretat i capacitat) (apache2, nginx)

- Gunicorn
- Waitress
- mod_wsgi
- uWSGI
- gevent
- eventlet
- ASGI



DEMO (SQLite)

1. Iniciar els 4 contenidors (docker-compose)
2. Veure inicialización contenidors
3. Obrir 4 finestres LOGS
4. Login com usuari administrador (admin/admin)
5. Accedir a traves de l'API als serveis: AUTH, ADMIN i AGENDA
6. Canviar password
7. Importar (usuaris, cliniques i doctors).
8. Afegir (random) 100-usuaris, 20-doctors, 10-Clíniques.
9. Afegir una cita
   - I una amb conflicte(veure error).
10. Afegir 1000 cites (random)
11. Canviar a usuari "Assistent"
12. Error afegir nova cita (no permís)
13. Prova anul·lació cites
14. Prova filtre cites
15. Login metge
16. Prova consulta agenda personal

Pendents

Manca de Temps

- Test unitaris i test integració (no massa experiència en llibreria unittest o pytest) (funcions testejades a l'antiga)
- Documentació del codi no està totalment completa
- Ara.. o faria més ràpid i potser algunes funcionalitats més senzilles, però no hi ha temps.