Fase 2: Evoluzione Architetturale (Microservizi & NoSQL)

1. Introduzione e Contesto

Il sistema monolitico "CIAMS" sviluppato nella Fase 1 è funzionale, ma l'azienda prevede un aumento del carico di utenti e la necessità di integrare nuove applicazioni esterne. Per supportare questa crescita, è stato deciso di avviare una **modernizzazione dell'architettura** passando da un approccio Monolitico a uno a **Microservizi**.

L'obiettivo di questa fase è **disaccoppiare** la logica di autenticazione dal dominio di business (Inventario), implementando il pattern architetturale *Strangler Fig* per estrarre funzionalità dal monolite.

2. Requisiti di Studio e Analisi (Research Task)

Prima di procedere con lo sviluppo, il team dovrà approfondire i concetti teorici sottostanti. È richiesta la preparazione di una **presentazione tecnica** da esporre al team, coprendo i seguenti argomenti:

1. **Architetture:** Differenze tra Monolite e Microservizi (Pro e Contro).
2. **Database:** Differenze tra Relazionale (SQL) e Documentale (NoSQL). Quando usare MongoDB rispetto a PostgreSQL?
3. **Sicurezza:** Differenza tra Autenticazione Stateful (Sessioni/Cookie) e Stateless (JWT).
4. **Design Patterns:**
    a. *Strangler Fig Pattern:* Strategie di migrazione graduale.
    b. *CQRS (Command Query Responsibility Segregation):* Concetti base, studiarlo e basta senza applicarlo al progetto.

3. Nuova Architettura di Sistema

Il sistema sarà composto da due servizi distinti che comunicano tramite protocollo HTTP e Token JWT.

3.1. Auth Service (Nuovo Microservizio)

Sarà l'unico responsabile della gestione delle identità (Identity Provider).

- **Database:** MongoDB
- **Responsabilità:** Registrazione utenti e rilascio dei Token.

3.2. Inventory Service (Refactoring)

È l'evoluzione del progetto attuale.

- **Database:** PostgreSQL (Relazionale).
- **Responsabilità:** Gestione Asset, Uffici e Licenze.
- **Modifica:** Non gestirà più la tabella utenti. Si fiderà dei Token emessi dall'Auth Service.

4. Specifiche Tecniche: Auth Service

Dovrà essere creato un nuovo progetto Spring Boot.

4.1. Stack Tecnologico

- **Database:** MongoDB (eseguito su container Podman/Docker).
- **Libreria DB:** Spring Data MongoDB.
- **Security:** Spring Security + Libreria JWT (es. jjwt o nimbus).

4.2. Modello Dati (Documento Mongo)

Non esistono tabelle, ma *Collection*. Il documento User dovrà contenere almeno:

- username (univoco).
- password (hashata tramite BCrypt).
- roles (array di stringhe, es. ["USER", "ADMIN"]).

4.3. API Endpoint

- POST /auth/register: Accetta username/password e crea l'utente su Mongo.
- POST /auth/login:
    - Accetta username/password.
    - Verifica le credenziali su Mongo.
    - Se valide, genera un **JWT (JSON Web Token)** firmato con una chiave segreta.
    - Il token deve contenere nel *Payload* lo username e i ruoli.

5. Specifiche Tecniche: Inventory Service (Refactoring)

Il progetto esistente deve subire le seguenti modifiche strutturali:

5.1. Pulizia del Dominio

- Eliminare (se presenti) le Entity User e le relative tabelle su PostgreSQL.
- Rimuovere qualsiasi logica di registrazione/login interna.

5.2. Integrazione Sicurezza JWT

- Implementare un **JWT Authentication Filter** che intercetti ogni richiesta HTTP.
- Il filtro deve:
    - Leggere l'header Authorization: Bearer <token>.
    - Validare la firma del token (usando la stessa chiave segreta dell'Auth Service).
    - Estrarre l'utente e i ruoli dal token.
    - Autenticare l'utente nel contesto di sicurezza di Spring per la durata della singola richiesta (Stateless).

6. Roadmap Operativa Fase 2

Step 1: Infrastruttura NoSQL

- Avviare un container MongoDB: podman run --name auth-db -p 27017:27017 -d mongo
- Installare un client per visualizzare i dati (es. MongoDB Compass).

Step 2: Sviluppo Auth Service

- Creare il progetto auth-service.

- Implementare la connessione a Mongo.
- Implementare la logica di generazione JWT.

Step 3: Refactoring Inventory Service

- Implementare la validazione del token.
- Testare che le chiamate senza token ricevano 401 Unauthorized.
- Testare che le chiamate con token valido (generato dall'altro servizio) ricevano 200 OK.

7. Deliverables Aggiornati

- **Collection Postman Integrata:** Uno scenario che esegua il Login sul primo servizio, salvi il token in una variabile, e lo usi per chiamare il secondo servizio.

Impostazione della gerarchia su GIT: Struttura del Monorepo (Gradle)

Il progetto deve essere consegnato in un **unico repository Git**, organizzato a cartelle separate per mantenere l'isolamento tra i servizi.

/ciams-project-root (La root del tuo Repository Git)

```
│
├── /auth-service         <-- Microservizio Auth (Progetto Gradle autonomo)
│   ├── src/
│   ├── build.gradle      <-- Configurazione specifica di Auth
│   ├── settings.gradle
│   └── Dockerfile
│
├── /inventory-service    <-- Microservizio Inventory (Progetto Gradle autonomo)
│   ├── src/
│   ├── build.gradle      <-- Configurazione specifica di Inventory
│   ├── settings.gradle
│   └── Dockerfile
│
├── /infrastructure       <-- Orchestrazione
│   └── docker-compose.yml  <-- Avvia Mongo, Postgres e le app
│
```

└── README.md <-- Istruzioni di avvio