PIS - MikroBiblioteka

Projekt **MikroBiblioteka** to aplikacja webowa umożliwiająca użytkownikom przesyłanie, przechowywanie, przeglądanie, pobieranie i usuwanie plików.

System opiera się na architekturze wielowarstwowej z podziałem na frontend, backend i warstwę baz danych.

Całość została uruchomiona w kontenerach Dockera i wdrożona na serwerze Oracle.

Proces wytwarzania i testowania oprogramowania wspierają narzędzia **GitHub**, **Jenkins**, **Nexus**, **JaCoCo** oraz **Jira**.

Stos technologiczny

1. Frontend

- **Technologia:** Angular
- **Język:** TypeScript, HTML, CSS
- **Funkcjonalność:**
    - interfejs użytkownika (upload, pobieranie, przeglądanie, usuwanie plików),
    - komunikacja z backendem przez REST API,
    - walidacja danych po stronie klienta.

2. Backend

- **Technologia:** Spring Framework (Spring Boot)
- **Język:** Java
- **Funkcjonalność:**
    - obsługa logiki aplikacji,
    - zarządzanie metadanymi plików,

    - komunikacja z bazami danych (PostgreSQL i MongoDB),
    - udostępnienie REST API dla frontendu.

3. Baza danych

- **PostgreSQL** – relacyjna baza danych przechowująca **metadane** o plikach (np. nazwa, data dodania, rozmiar pliku).
- **MongoDB** – nie relacyjna baza danych przechowująca **same pliki (binary data)**.

4. Konteneryzacja i infrastruktura

- **Docker** – konteneryzacja każdej warstwy systemu (frontend, backend, PostgreSQL, MongoDB).
- **Docker Compose** – orkiestracja czterech kontenerów i konfiguracja sieci między nimi.
- **Serwer Oracle** – środowisko uruchomieniowe hostujące aplikację i narzędzie Jenkins.

5. System kontroli wersji

- **GitHub** – repozytorium kodu źródłowego, zarządzanie wersjami, pull requestami i współpracą zespołową.

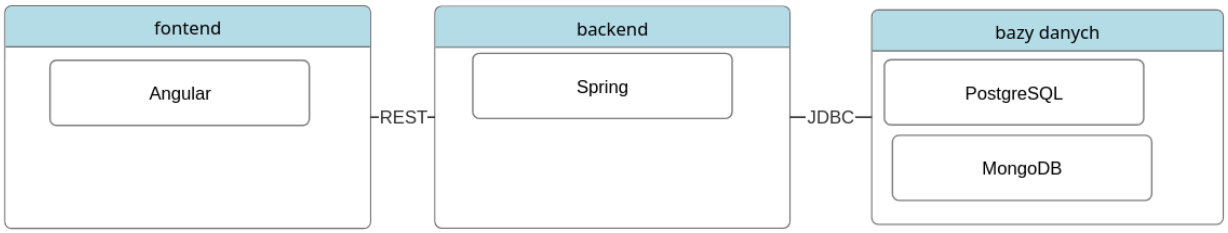
6. CI/CD, testy i repozytorium artefaktów

- **Jenkins** – narzędzie CI/CD do automatyzacji budowania, testowania i wdrażania aplikacji.
- **JaCoCo** – narzędzie zintegrowane z Jenkins do pomiaru **pokrycia kodu testami** w Javie.
- **Nexus Repository Manager** – repozytorium artefaktów (np. plików `.jar`, `.war`), wykorzystywane do przechowywania wersji aplikacji po procesie budowania.

7. Zarządzanie projektem

- **Jira** – system do zarządzania zadaniami, epikami i iteracjami; wspiera planowanie i monitorowanie postępów w projekcie.

Architektura aplikacji



Architektura aplikacji składa się z:

1. **Warstwy prezentacji (frontend)** – zrealizowanej w technologii Angular. Odpowiada za interfejs użytkownika, umożliwia przesyłanie, pobieranie i przeglądanie plików. Komunikuje się z backendem za pośrednictwem REST API.

2. **Warstwy logiki biznesowej (backend)** – zaimplementowanej w technologii Spring Boot. Odpowiada za przetwarzanie danych, walidację, logikę biznesową oraz komunikację z warstwą baz danych.

3. **Warstwy danych** – składającej się z dwóch baz: relacyjnej PostgreSQL (metadane plików) i nierelacyjnej MongoDB (przechowywanie plików binarnych).

Komunikacja między warstwami odbywa się w modelu klient–serwer z wykorzystaniem protokołu HTTP oraz interfejsów REST API.
Aplikacja została skonteneryzowana w środowisku Docker, co umożliwia łatwe wdrożenie i skalowanie systemu.