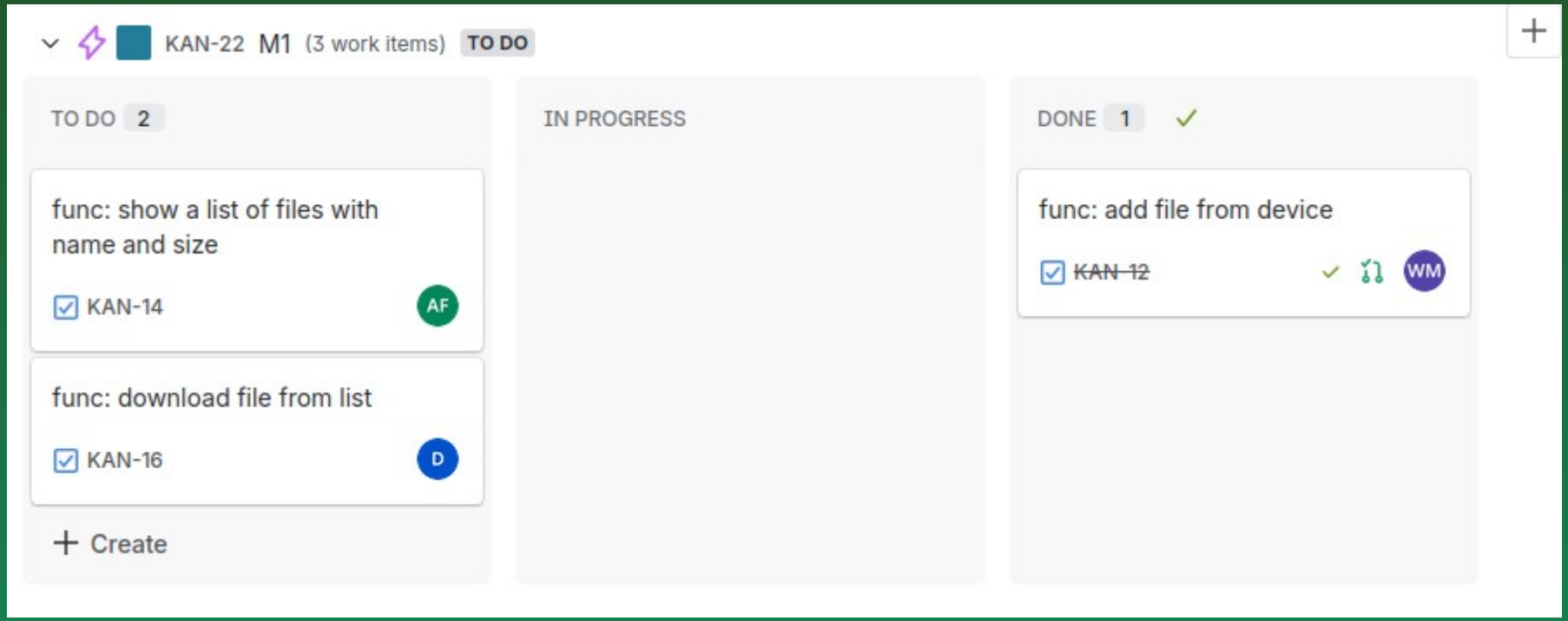
MikroBiblioteka

System do archiwizowania plików

Zespół: Alesia Filinkova, Diana Pelin, Weronika Maślana

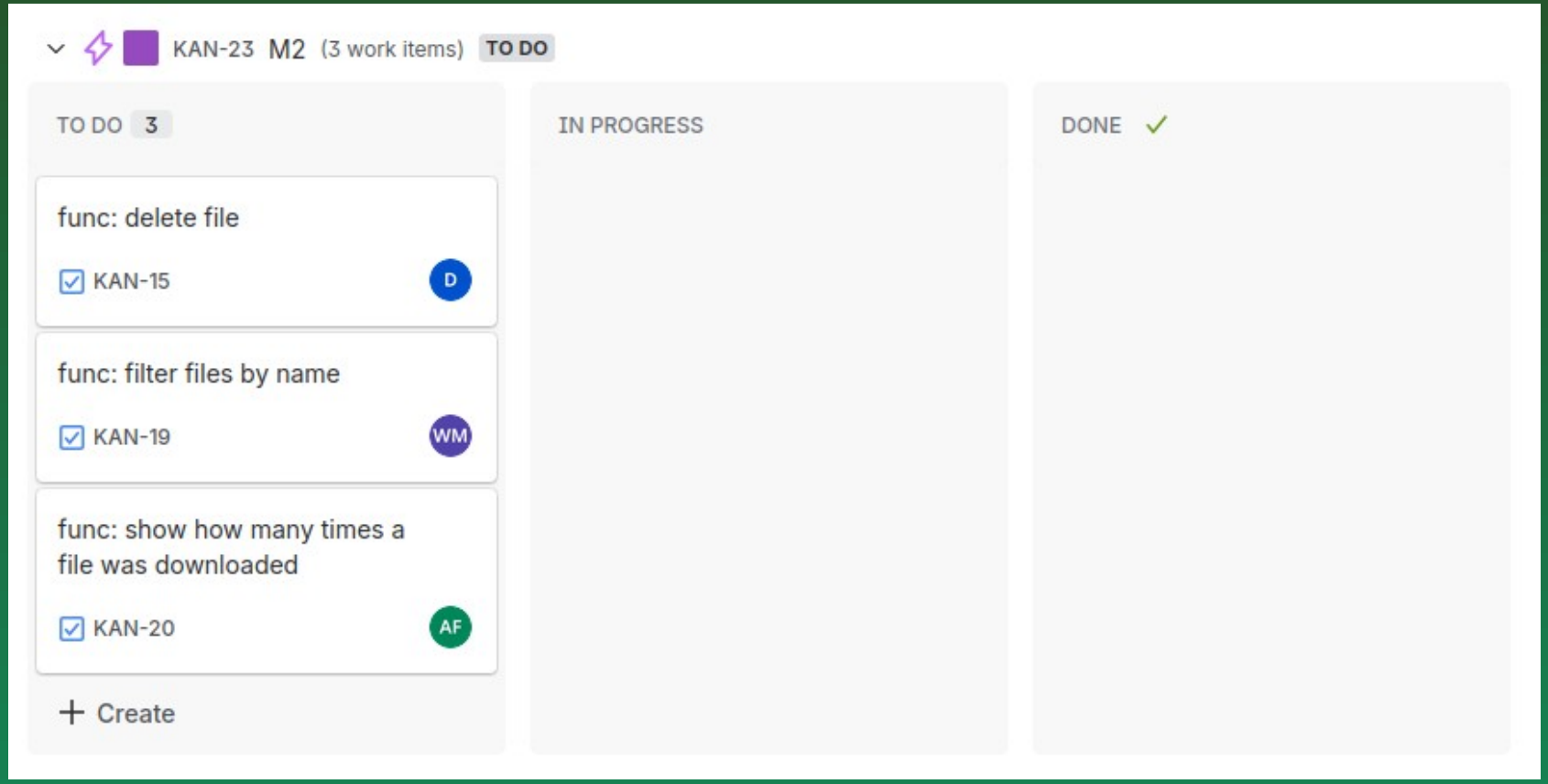
Wymagania funkcjonalne

1 milestone (M1 z Jiry): MVP – Minimum Viable Product

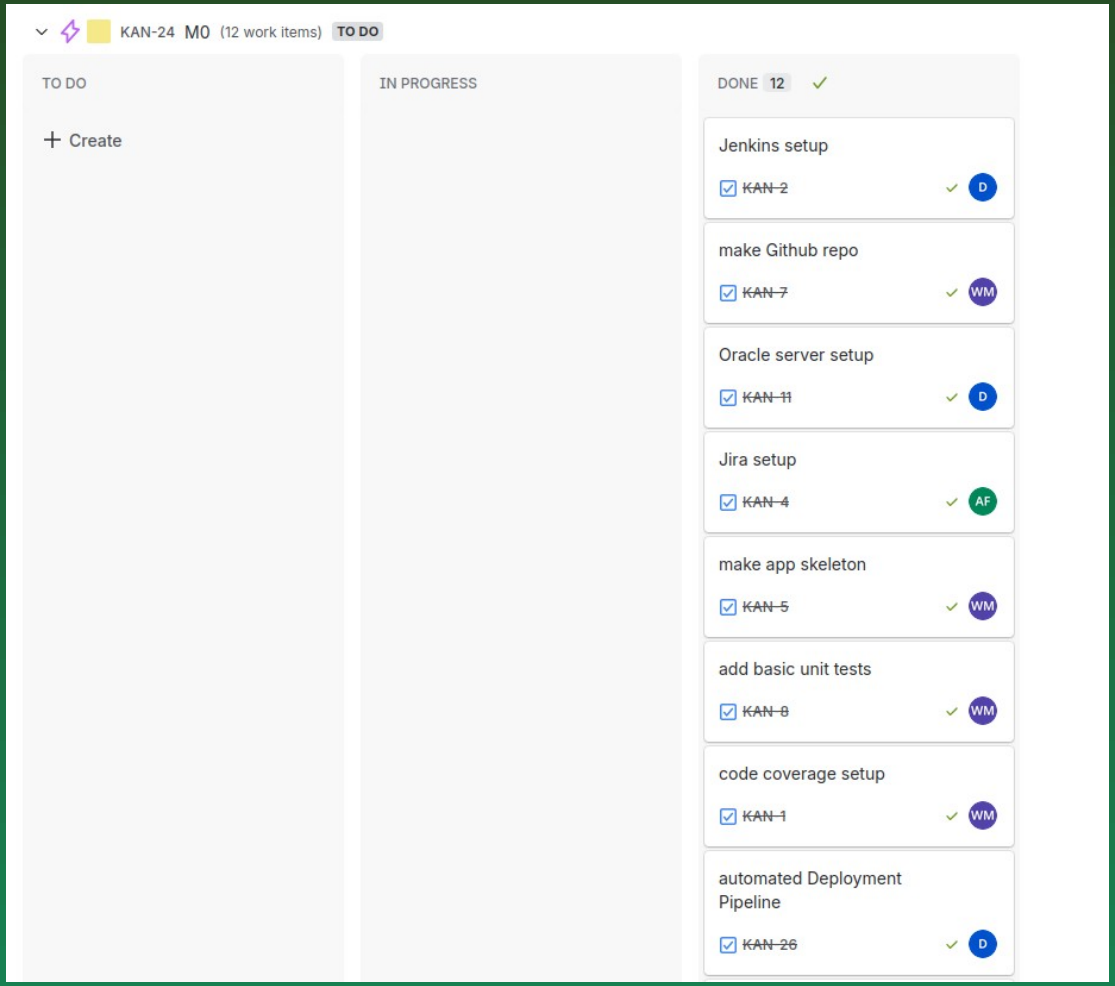


Wymagania funkcjonalne

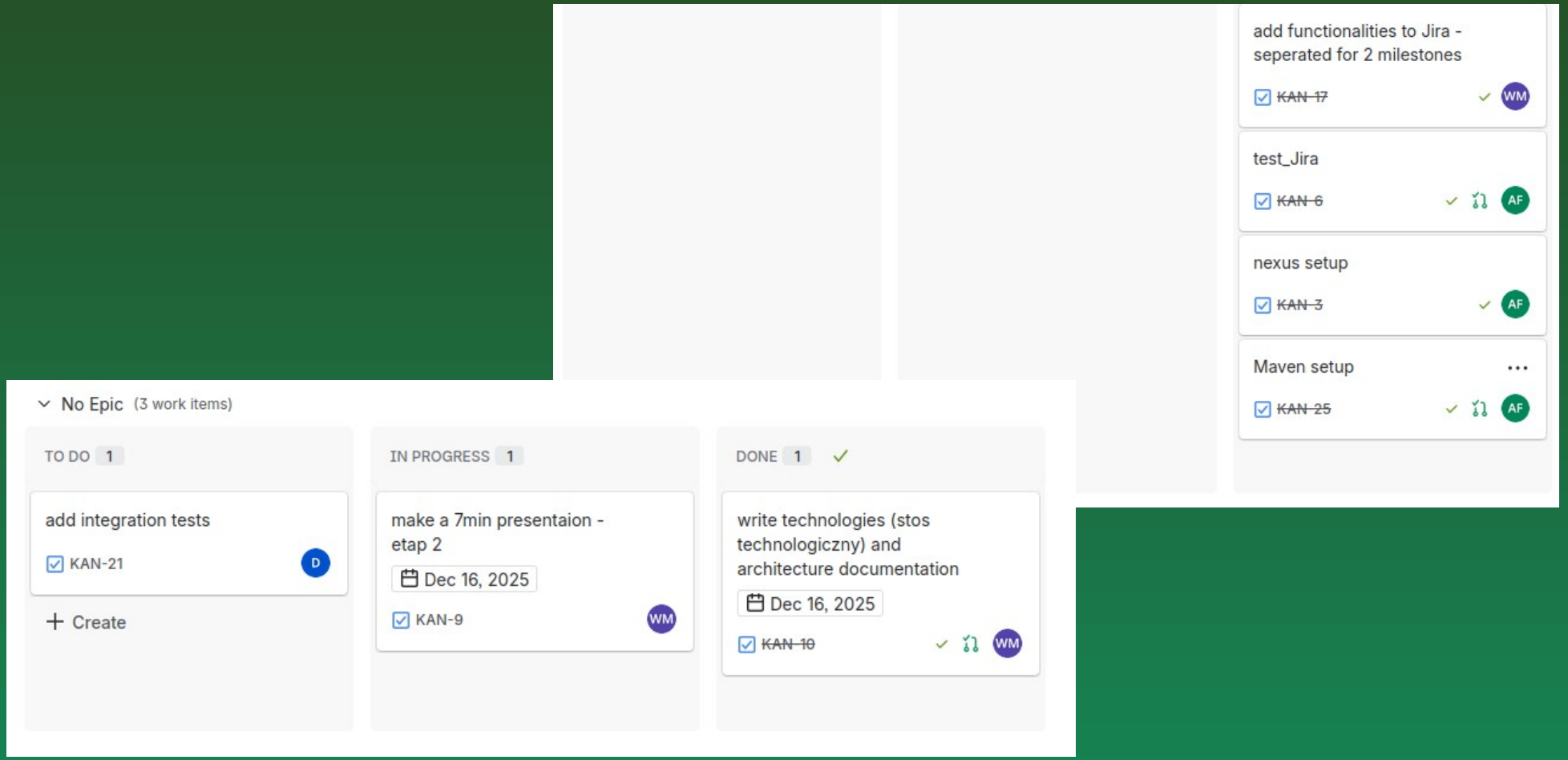
2 milestone (M2 z Jiry): Management & UI

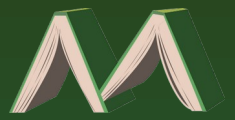


Raport z pozostałych zadań



Raport z pozostałych zadań





Stos Technologiczny - opis

- 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 maszynie wirtualnej w chmurze Azure.
- Proces wytwarzania i testowania oprogramowania wspierają narzędzia GitHub, Jenkins, Nexus, JaCoCo oraz Jira.

Stos Technologiczny - frontend

- **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.

Stos Technologiczny - backend

- **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.

Stos Technologiczny – bazy danych i infrastruktura

- **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.
  - ● Wirtualna maszyna w chmurze Azure – środowisko uruchomieniowe hostujące aplikację, narzędzia Jenkins oraz Nexus.

Stos Technologiczny – c.d.

- **5. System kontroli wersji**
  - ● GitHub – repozytorium kodu źródłowego, zarządzanie wersjami, pull requestami i współpracą zespołową.
- **6. Zarządzanie projektem**
  - ● Jira – system do zarządzania zadaniami i iteracjami; wspiera planowanie i monitorowanie postępów w projekcie.

Stos Technologiczny – c.d.

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.

Architektura aplikacji

Diagram warstwowy

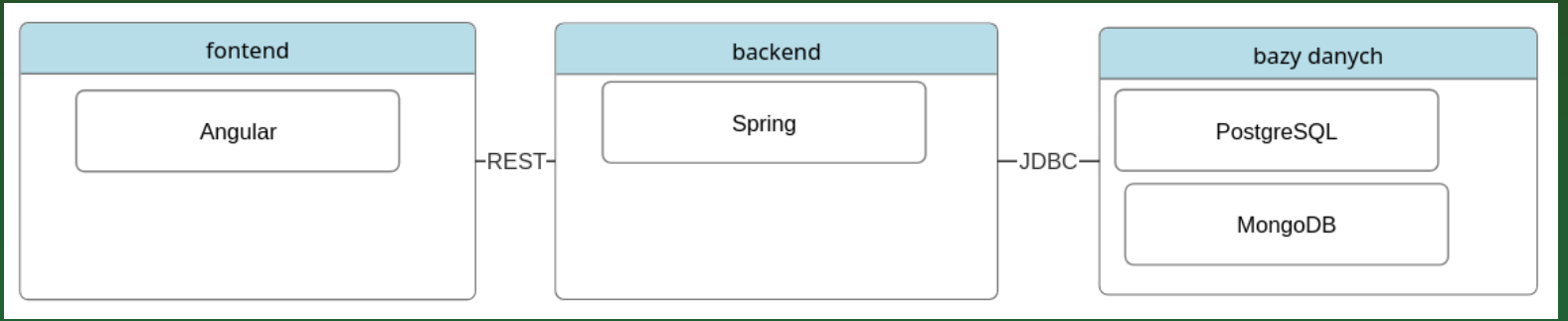
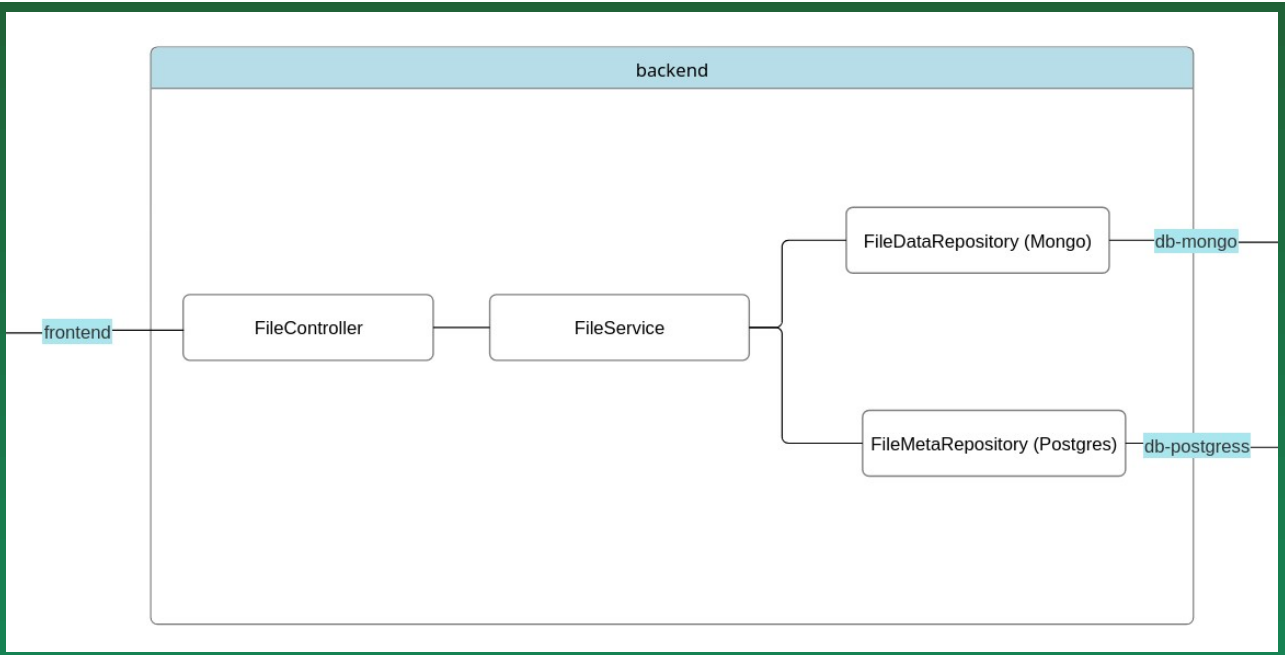


Diagram komponentów



Architektura aplikacji - opis

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).

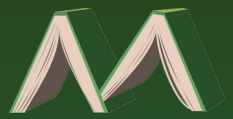
Architektura aplikacji - opis

- **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.

Zrąb projektu

Zaprezentujemy:

- Implementację jednej funkcjonalności w docelowych technologiach – upload file
- Wszystkie warstwy
- Testy



Lista materiałów szkoleniowych

- Oficjalna dokumentacja Spring Boot
- MongoDB University courses
- Angular.io
- Docker docs
- Baeldung (Spring)
- YouTube – Amigoscode, Java Brains

Instrukcja budowania projektu

Zawarta w README na Githubie:

Uruchomienie

z głównego katalogu zawołaj w terminalu:

```
docker-compose build
docker-compose up
```

(można dopisać frontend lub backend lub db-postgres by uruchomić tylko dany obraz z dockera) Program korzysta z biblioteki Material UI, zainstaluj ją komendą:

```
npm install @mui/material
npm install @hello-pangea/dnd
npm install @angular/core @angular/common @angular/platform-browser @angular/c
```

Dziękujemy za uwagę :)