

PIS-MikroBiblioteka

Alesia Filinkova
Weronika Maślana
Diana Pelin

Wstęp

Celem systemu MikroBiblioteka jest zapewnienie lekkiej, mikroserwisowej platformy do przechowywania i zarządzania plikami. Projekt umożliwia użytkownikom przesyłanie, pobieranie, filtrowanie oraz usuwanie zasobów, a także prezentuje metadane przechowywane w relacyjnej bazie danych oraz dane binarne składowane w bazie dokumentowej.

Realizacja projektu odbywała się w ramach przedmiotu Programowanie i Systemy Informacyjne i została podzielona na kilka iteracji odpowiadających kolejnym etapom rozwoju funkcjonalnego: od stworzenia minimalnego działającego prototypu (M1) po rozszerzoną wersję produkcyjną (M2). Każda iteracja obejmowała implementację funkcji, testy, integrację systemu oraz dostarczenie działającego wdrożenia w oparciu o pipeline CI/CD.

Zespół projektowy składał się z trzech osób, z których każda brała udział zarówno w implementacji backendu, frontendu, jak i konfiguracji środowiska oraz integracji (DevOps). Szczegółowy podział prac i odpowiedzialności został udokumentowany w systemie Jira, gdzie znajdują się powiązania między zadaniami, milestone'ami i osobami odpowiedzialnymi za konkretne elementy systemu.

Wymagania i backlog

System MikroBiblioteka został rozwinięty iteracyjnie w trzech etapach: M0 (setup infrastruktury), M1 (MVP) i M2 (rozszerzona funkcjonalność). Backlog był zarządzany w systemie Jira, gdzie każde wymaganie miało przypisany milestone, osoby odpowiedzialne oraz status.

Wymagania

M0 — Setup projektu i infrastruktury

- Konfiguracja code coverage
- Jenkins setup
- Nexus setup
- Konfiguracja Jira
- Utworzenie szkieletu aplikacji
- Utworzenie repozytorium GitHub
- Podstawowe testy jednostkowe
- Konfiguracja serwera VM
- Uzupełnienie funkcjonalności w Jira

- Konfiguracja Maven
- Automatyzacja pipeline CI/CD

M1 — Minimal Viable Product (MVP)

- Upload pliku z urządzenia
- Wyświetlanie listy plików z nazwą i rozmiarem
- Pobieranie pliku z listy

M2 — Rozszerzona funkcjonalność

- Usuwanie pliku
- Filtrowanie listy plików po nazwie
- Wyświetlanie liczby pobrań pliku

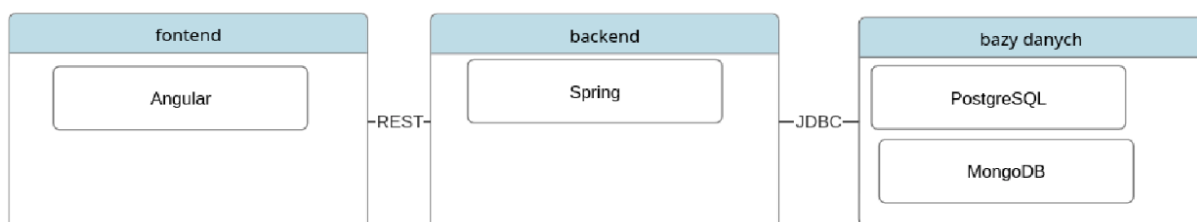
Architektura systemu

System MikroBiblioteka został zaprojektowany jako modułowa aplikacja oparta na architekturze trójwarstwowej, z wyraźnym rozdzieleniem logiki prezentacji, logiki biznesowej oraz warstwy danych. Całość działa w środowisku kontenerowym Docker, co umożliwia łatwą replikację środowiska oraz automatyzację procesu wytwórczego poprzez CI/CD. Aplikacja składa się z czterech głównych komponentów: frontend (Angular), backend (Spring Boot), baza danych PostgreSQL oraz baza danych MongoDB. Komunikacja między modułami odbywa się poprzez protokół HTTP (REST) oraz natywne sterowniki do baz danych (JDBC)

1. Warstwa prezentacji (Frontend – Angular) - interakcja z użytkownikiem
2. Warstwa biznesowa (Backend – Spring Boot) - logika aplikacji
3. Warstwa danych

PostgreSQL przechowuje metadane plików: nazwa, rozmiar, typ, licznik pobrań.

MongoDB przechowuje binarne treści plików



Stos technologiczny i środowisko pracy

Projekt wykorzystuje zestaw narzędzi i technologii, które wspierają implementację, kontrolę wersji, automatyzację procesu wytwórczego oraz wdrożenie aplikacji. Środowisko zostało oparte o kontenery Docker oraz pipeline CI/CD uruchamiany na maszynie wirtualnej w chmurze.

Narzędzia infrastrukturalne

- Docker & Docker Compose
 - Aplikacja działa w czterech kontenerach: frontend, backend, PostgreSQL, MongoDB
 - Docker Compose odpowiada za uruchomienie całego środowiska jednym poleceniem
 - Oddzielenie środowisk umożliwia powtarzalne wdrożenia i łatwe debugowanie
- VM w chmurze (Azure)
 - Na maszynie działa: Jenkins (orchestracja pipeline), Nexus Repository, środowisko uruchomieniowe (docker-compose)
- Jenkins
 - Automatyczny pipeline wywoływany po zmianach w repozytorium
 - Kroki pipeline:
 - Checkout z GitHub
 - Build projektu (Maven)
 - Testy jednostkowe
 - Analiza pokrycia kodu (JaCoCo)
 - Budowanie obrazu Dockera
 - Publikacja artefaktu do Nexusa
 - Wdrożenie na VM (docker-compose up)
- Nexus Repository Manager
 - Przechowywanie artefaktów buildów — w tym JAR backendu
 - Służy jako centralny rejestr artefaktów dla pipeline
- JaCoCo
 - Analiza pokrycia testami
 - Generowanie raportów HTML i XML dla Jenkinsa
- Jira
 - Backlog projektu z podziałem na milestoney M0/M1/M2
 - Każde zadanie przypisane do osoby i opatrzone statusem

Implementacja i główne scenariusze użycia

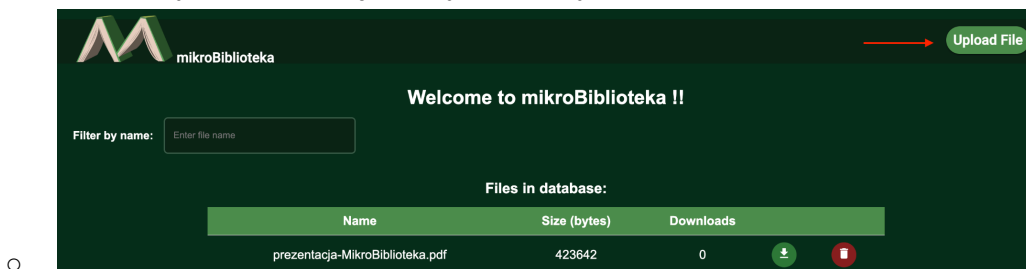
Implementacja systemu MikroBiblioteka została oparta na wyraźnym podziale warstw: REST → serwisy → repozytoria → bazy danych PostgreSQL/MongoDB.

Struktura backendu

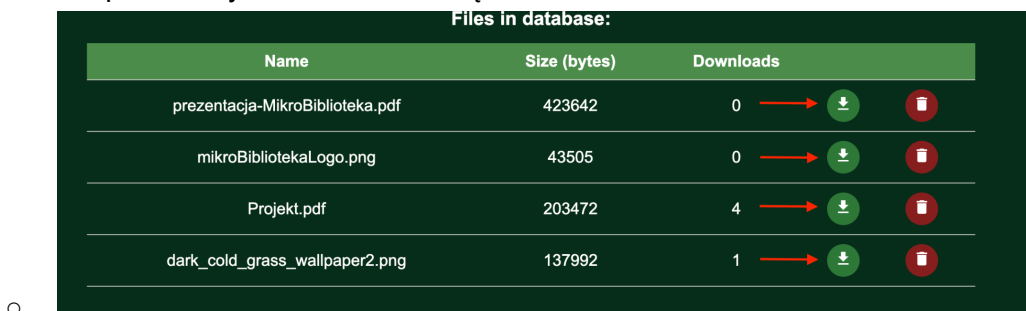
- Controller – ekspozycja REST API, walidacja wejścia
- Service – logika biznesowa
- Repository (JPA/Mongo) – dostęp do baz danych
- Model/DTO – definicje struktury danych

Kluczowe scenariusze użycia

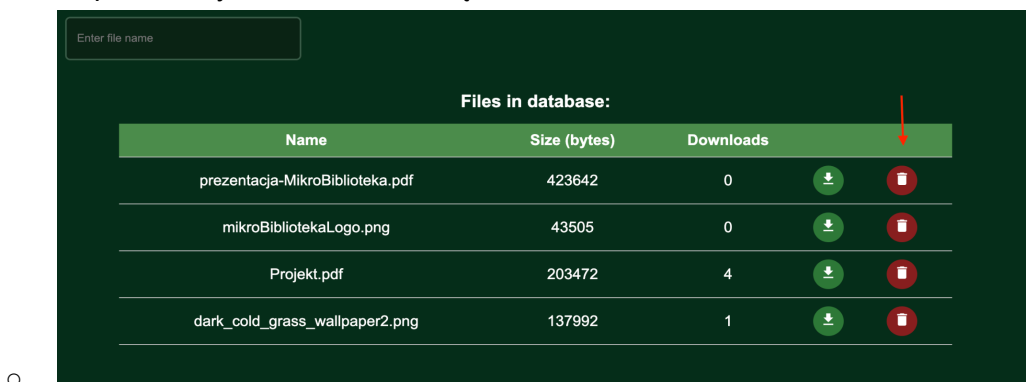
- Upload pliku - użytkownik dodaje nowy plik do systemu.



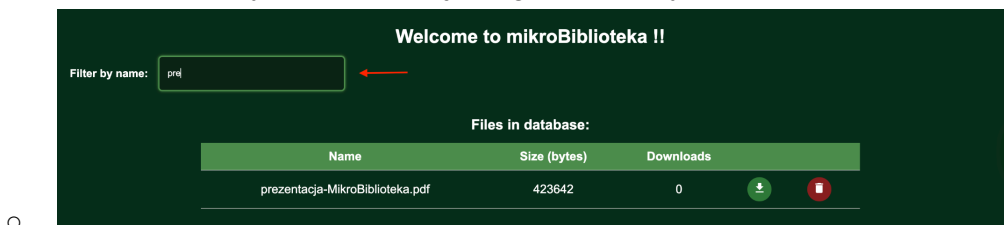
- Pobieranie pliku - Użytkownik klika ikonę „download” w UI



- Usuwanie pliku - Użytkownik klika ikonę kosza



- Filtrowanie plików - Użytkownik wpisuje fragment nazwy w polu search.



- Liczba pobrań - użytkownik może sprawdzić liczbę pobrań

Files in database:

| Name | Size (bytes) | Downloads | | |
|---------------------------------|--------------|-----------|--|--|
| prezentacja-MikroBiblioteka.pdf | 423642 | 0 | | |
| mikroBibliotekaLogo.png | 43505 | 0 | | |
| Projekt.pdf | 203472 | 4 | | |
| dark_cold_grass_wallpaper2.png | 137992 | 1 | | |

-
- Rozmiar pliku - użytkownik może sprawdzić rozmiar pliku

Files in database:

| Name | Size (bytes) | Downloads | | |
|---------------------------------|--------------|-----------|--|--|
| prezentacja-MikroBiblioteka.pdf | 423642 | 0 | | |
| mikroBibliotekaLogo.png | 43505 | 0 | | |
| Projekt.pdf | 203472 | 4 | | |
| dark_cold_grass_wallpaper2.png | 137992 | 1 | | |

Testy i jakość

Proces testowania w projekcie MikroBiblioteka został zaprojektowany tak, aby zapewnić stabilność, przewidywalność i powtarzalność działania systemu. Testy były wykonywane zarówno lokalnie, jak i automatycznie w pipeline CI/CD uruchamianym w Jenkinsie. Celem testów było zweryfikowanie poprawności logiki biznesowej, obsługi błędów oraz integracji warstw aplikacji.

Analiza pokrycia kodu została przeprowadzona za pomocą **JaCoCo**, które integruje się bezpośrednio z Mavenem i Jenkins Pipeline.







com.mikroBiblioteka.project.tests 28 ✓ 28 ⌚ Took 5.4 sec 📊 🔗

All Tests

| Class | Failed | Skipped | Passed | Total | Duration |
|-------------------------|--------|---------|-------------------|-------------------|----------|
| ApplicationTest | 0 | 0 | 2 | 2 | 2.7 sec |
| CorsConfigTest | 0 | 0 | 1 +1 | 1 +1 | 5 ms |
| FileServiceTest | 0 | 0 | 9 +9 | 9 +9 | 1.5 sec |
| MetaFileControllerTest | 0 | 0 | 9 | 9 | 1 sec |
| MetaFileIntegrationTest | 0 | 0 | 7 | 7 | 0.1 sec |

raport JaCoCo - pokrycie kodu

backend

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|--|---|------|---|------|--------|------|--------|-------|--------|---------|--------|---------|
| com.mikroBiblioteka.project.service |  | 100% |  | 100% | 0 | 8 | 0 | 38 | 0 | 7 | 0 | 1 |
| com.mikroBiblioteka.project.controller |  | 100% |  | 100% | 0 | 8 | 0 | 33 | 0 | 6 | 0 | 1 |
| com.mikroBiblioteka.project.config |  | 100% | | n/a | 0 | 4 | 0 | 8 | 0 | 4 | 0 | 2 |
| com.mikroBiblioteka.project |  | 100% | | n/a | 0 | 3 | 0 | 4 | 0 | 3 | 0 | 1 |
| Total | 0 of 338 | 100% | 0 of 6 | 100% | 0 | 23 | 0 | 83 | 0 | 20 | 0 | 5 |

CI/CD i wdrożenie

System MikroBiblioteka jest wdrażany automatycznie z wykorzystaniem pipeline'u CI/CD w Jenkinsie, działającym na maszynie wirtualnej w chmurze Azure. Pipeline jest uruchamiany po pushu do głównego repozytorium w GitHub i realizuje pełny proces od kodu do działającej aplikacji w kontenerach.

Pipeline CI/CD (Jenkins)

Typowy przebieg pipeline:

- Checkout kodu z GitHub (branch main).
- Build i testy – mvn clean test, uruchomienie testów jednostkowych
- Generowanie raportu pokrycia – mvn jacoco:report, publikacja raportów w Jenkinsie.
- Budowa artefaktu – JAR backendu oraz build frontendu (Angular).
- Budowa obrazów Dockera dla frontendu i backendu.
- Publikacja artefaktów do Nexusa (JAR) oraz rejestru obrazów (lokalnie lub zewnętrznie).
- Wdrożenie na VM – wykonanie docker-compose pull/build oraz docker-compose up -d.

Raport z realizacji projektu

Realizacja projektu MikroBiblioteka została podzielona na trzy główne fazy: przygotowanie środowiska, implementacja podstawowych funkcji (M1) oraz implementacja funkcjonalności rozszerzających (M2).

Kamienie milowe (milestones)

| Milestone | Zakres |
|-------------------------|---|
| M0 - Setup | Jenkins, Nexus, JaCoCo, Docker Compose, skeleton projektu |
| M1 - Main functionality | upload, list, download |
| M2 - System finalny | delete, filtering, download counter |

| | | | | | | | | | | |
|--------------------------|--|-----|------------------|---------------------|--------|-------|------------|--------------------|--------------------|------|
| <input type="checkbox"/> | ▼ KAN-24 M0 | ⌵ + | Unassigned | Weronika Maslana... | Medium | TO DO | Unresolved | 10 Nov 2025, 13:42 | 10 Nov 2025, 13:42 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-26 automated Deployment Pipeline | | Diana | Diana | Medium | DONE | Done | 15 Nov 2025, 08:34 | 15 Nov 2025, 08:35 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-25 Maven setup | | Alesia Filinkova | Alesia Filinkova | Medium | DONE | Done | 10 Nov 2025, 22:43 | 11 Nov 2025, 19:29 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-17 add functionalities to Jira - seperated for 2 ... | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:38 | 10 Nov 2025, 14:18 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-11 Oracle server setup | | Diana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:23 | 11 Nov 2025, 07:52 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-8 add basic unit tests | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:14 | 10 Nov 2025, 14:09 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-7 make Github repo | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:12 | 10 Nov 2025, 13:43 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-6 test_Jira | | Alesia Filinkova | Alesia Filinkova | Medium | DONE | Done | 09 Nov 2025, 17:22 | 20 Nov 2025, 21:20 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-5 make app skeleton | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 07 Nov 2025, 16:09 | 10 Nov 2025, 14:09 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-4 Jira setup | | Alesia Filinkova | Alesia Filinkova | Medium | DONE | Done | 03 Nov 2025, 17:03 | 10 Nov 2025, 14:09 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-3 nexus setup | | Alesia Filinkova | Alesia Filinkova | Medium | DONE | Done | 03 Nov 2025, 17:02 | 11 Nov 2025, 19:29 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-2 Jenkins setup | | Diana | Diana | Medium | DONE | Done | 03 Nov 2025, 17:01 | 11 Nov 2025, 07:52 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-1 code coverage setup | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 02 Nov 2025, 21:42 | 10 Nov 2025, 14:09 | None |
| <input type="checkbox"/> | ▼ KAN-23 M2 | | Unassigned | Weronika Maslana... | Medium | TO DO | Unresolved | 10 Nov 2025, 13:31 | 10 Nov 2025, 13:32 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-20 func: show how many times a file was dow... | | Alesia Filinkova | Weronika Maslana... | Medium | DONE | Done | 10 Nov 2025, 00:07 | 25 Nov 2025, 00:12 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-19 func: filter files by name | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 10 Nov 2025, 00:06 | 28 Nov 2025, 00:12 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-16 func: delete file | | Diana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:35 | 23 Nov 2025, 11:46 | None |
| <input type="checkbox"/> | ▼ KAN-22 M1 | | Unassigned | Weronika Maslana... | Medium | TO DO | Unresolved | 10 Nov 2025, 13:31 | 10 Nov 2025, 13:31 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-18 func: download file from list | | Diana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:35 | 23 Nov 2025, 09:35 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-14 func: show a list of files with name and size | | Alesia Filinkova | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:34 | 22 Nov 2025, 14:16 | None |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> KAN-12 func: add file from device | | Weronika Maslana | Weronika Maslana... | Medium | DONE | Done | 09 Nov 2025, 23:33 | 16 Nov 2025, 18:46 | None |

Metryki per osoba

Liczba commitów

| | |
|----------|----|
| Alesia | 31 |
| Weronika | 33 |
| Diana | 16 |

Liczba wykonanych zadań w Jira

| | M0 | M1 | M2 |
|----------|----|----|----|
| Alesia | 4 | 1 | 1 |
| Weronika | 5 | 1 | 1 |
| Diana | 3 | 1 | 1 |

Doświadczenia i wnioski

Realizacja projektu MikroBiblioteka pozwoliła zespołowi zdobyć praktyczne doświadczenie zarówno w warstwie programistycznej, jak i DevOps.

Problemy napotkane w projekcie

1. Konfiguracja Dockera na maszynie wirtualnej
Odmienne ustawienia systemowe powodowały błędy przy montowaniu wolumenów oraz uruchamianiu obrazów z poziomu Jenkinsa.

2. Pipeline wdrożeniowy CI/CD
Budowanie obrazów Dockera i jednocześnie publikowanie artefaktów do Nexusa wymagało dostosowania uprawnień i ścieżek.
3. Maszyna wirtualna
Była potrzeba zmiany maszyny wirtualnej z Oracle cloud na Azure ze względu na ograniczenie systemowe i wydajność
4. Spójność danych podczas usuwania plików
Konieczne było opracowanie logicznej transakcyjności między Postgres i MongoDB, aby unikać pozostawiania „sierot”