

System biblioteczny

Autorzy: Michał Goławski, Maciej Dobrowolski, Marcin Bagnowski, Marcin Sikorski


Opis tematu: System do zarządzania biblioteką, jej księgozbiorem, wypożyczeniami oraz czytelnikami i pracownikami.

Budowa projektu

Projekt składa się z następujących mikroservisów znajdujących się w poszczególnych repozytoriach:

- Frontend - <https://github.com/Kapturz0ny/pis-frontend>
- Gateway Api - <https://github.com/Kapturz0ny/pis-gateway-api>
- Auth Api - <https://github.com/Kapturz0ny/pis-auth-api>
- User Management Api - <https://github.com/Kapturz0ny/pis-user-management-api>
- Books Api - <https://github.com/Kapturz0ny/pis-books-api>
- Book Rent Api - <https://github.com/Kapturz0ny/pis-book-rent-api>

Dodatkowo przebieg prac był planowany z wykorzystaniem serwisu JIRA.

 Atlassian Cloud

Uruchomienie

1. Należy pobrać kod z każdego repozytorium.
2. Frontend należy uruchomić z poziomu głównego katalogu frontendu wykorzystując polecenie
`"docker-compose up --build"`
3. Następnie należy uruchomić kolejne api w podanej kolejności, gdzie dla każdego z nich należy wykonać 2 polecenia (oba z poziomu głównego katalogu danego api)
 - a. Polecenie budujące projekt:
`"mvn clean package -DskipTests"`
 - b. Polecenie uruchamiające kontenery:
`"docker-compose up --build"`
 - c. Kolejność uruchamiania api:
 - i. Gateway Api
 - ii. Auth Api
 - iii. User Management Api
 - iv. Books Api
 - v. Book Rent Api
4. Początkowo bazy danych są puste dlatego możemy wprowadzić przykładowe dane do bazy książek z wykorzystaniem skryptu `book_populate.sql` w Books Api.
5. Teraz trzeba utworzyć połączenie z serwerem poprzez ssh z przekierowaniem odpowiednich portów. Na serwerze uruchomione są usługi Jenkins oraz Nexus. Bez tego połączenia można korzystać z aplikacji jednak spowoduje to brak dostępu do tych dwóch usług. Dostęp do serwera jest za pośrednictwem serwera pośredniego `mion.elka.pw.edu.pl`
 - a. Połączenie na serwer pośredni

- ssh username@mion.elka.pw.edu.pl -L 8081:localhost:<X> -L 8080:localhost:<Y>
- b. Połączenie na serwer docelowy
ssh username@192.168.162.223 -L <X>:localhost:8081 -L <Y>:localhost:8080
 - c. X i Y to dowolne wolne porty na serwerze pośredniczącym np 42000 i 42001
6. Od tego momentu są dostępne następujące usługi:
- a. Frontend - <http://localhost:80>
 - b. Jenkins - <http://localhost:8080>
 - c. Nexus - <http://localhost:8081>

Podział prac w zespole

Praca w zespole była podzielona, jednak każdy z członków zespołu miał swój udział w większości komponentów systemu - współpraca opierała się na wzajemnym wsparciu.

- Michał Goławski - Book Api, Integracja Kafki, Frontend, Gateway Api, Auth Api
- Marcin Sikorski - Frontend, Gateway Api, Auth Api, Jenkins
- Marcin Bagnowski - User Management Api, Kafka
- Maciej Dobrowolski - Book Rent Api, Frontend, Jenkins

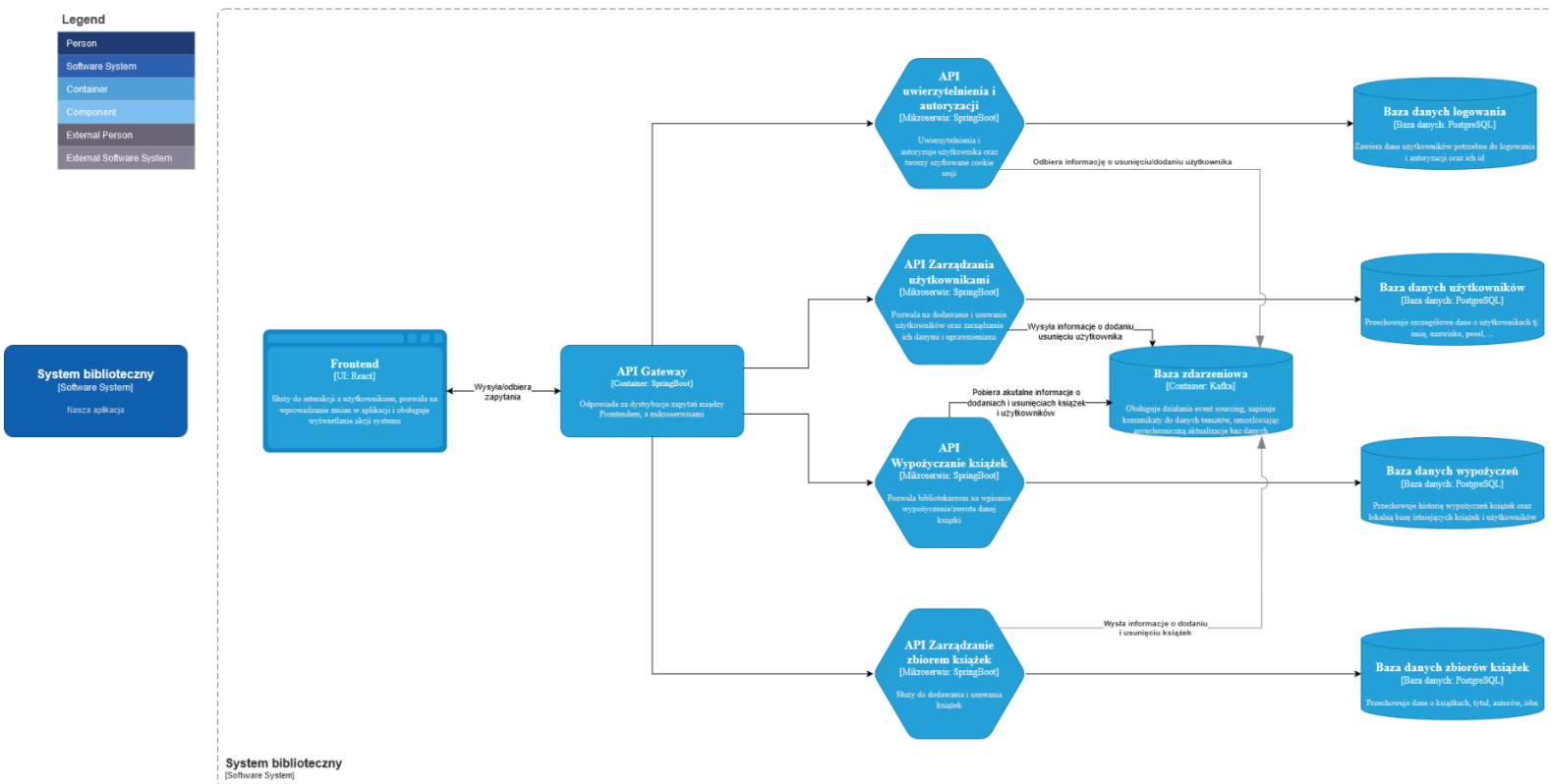
Jenkins

Mamy zdefiniowane przepływy, które sprawdzają poprawność budowania każdego projektu. Są one połączone z repozytoriami na platformie GitHub. Dla każdego repozytorium zostały zdefiniowane następujące przepływy:

S	W	Name	Last Success	Last Failure	Last Duration	
✓	☁	auth-api	1 hr 36 min #31	2 hr 2 min #28	32 sec	▶
✓	☀	book-api	1 hr 44 min #1	N/A	37 sec	▶
✓	☁	book-rent-api	12 min #4	2 hr 18 min #1	17 sec	▶
✓	☀	frontend	46 min #94	1 hr 49 min #89	1 min 20 sec	▶
✓	☀	gateway-api	12 min #34	24 days #26	12 sec	▶
✓	☀	system-biblioteczny	1 mo 27 days #20	2 mo 3 days #15	11 sec	▶

Architektura systemu

System oparty jest na architekturze mikroserwisów. Każdy z nich ma swoją lokalną bazę danych oraz komunikują się ze sobą z wykorzystaniem bazy zdarzeniowej Kafka. Dostęp do aplikacji z perspektywy użytkownika jest za pomocą frontendu, który to komunikuje się z odpowiednim mikroserwisem za pośrednictwem Api Gateway, które przekierowuje zapytania do odpowiedniego mikroserwisu. Poniższe zdjęcie prezentuje drugą warstwę modelu C4 przedstawiając architekturę komponentów systemu.



Schemat bazy danych

