# Dokumentacja projektu

Projekt “Proof of Concept” portalu społecznościowego wykorzystującego grafową bazę danych Neo4j

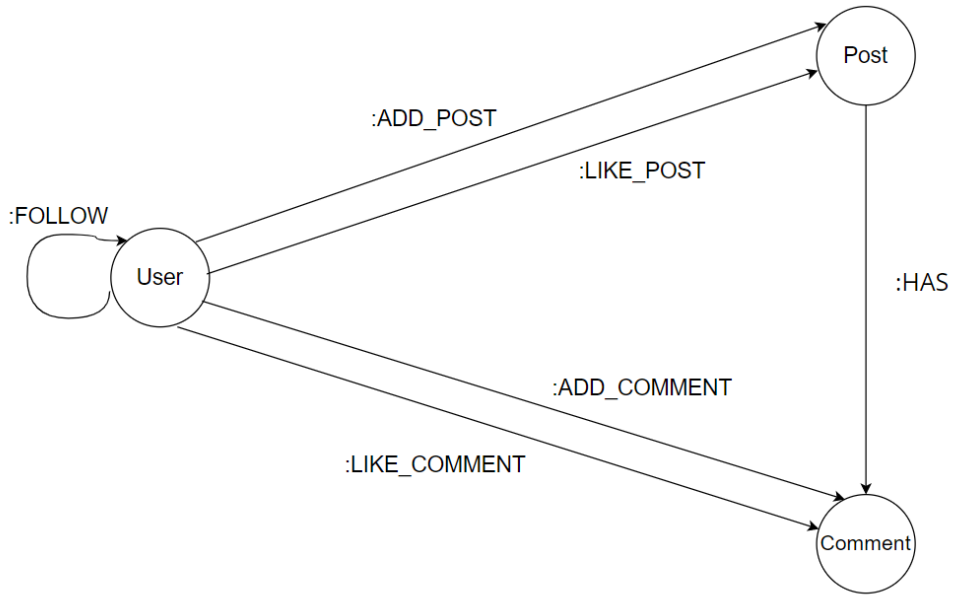
Jan Wojdylak, 20.12.2022

# 1. Opis projektu oraz założenia początkowe

Celem projektu było stworzenie prostego portalu społecznościowego opierającego się na grafowej bazie danych. Aplikacja pozwala użytkownikom na dodawanie nowych postów, przeglądanie postów innych użytkowników, obserwowanie użytkowników, dodawanie polubień do postów, dodawanie komentarzy do postów oraz dodawanie polubień do komentarzy.

# 2. Implementacja

## 2.1. Schemat bazy danych




Rys 1. Schemat bazy

Schemat bazy powstał po uwzględnieniu wstępnych założeń i spełnia on przedstawione wymagania.
Użytkownik może:
- obserwować innego użytkownika dodając go do relacji FOLLOW.

- tworzyć posty  dodając je do relacji ADD POST.
- polubić post każdego uczestnika dodając go do relacji LIKE_POST,
- dodać komentarz pod postem dodajac relacje ADD_COMMENT. Komentarz jest przypisany do autora, ale również do postu relacją :HAS.
- polubić komentarze pod postami

## 2.2. Backend

Obsługa serwera aplikacji została zaimplementowana w javie z wykorzystaniem spring framework. Do obsługi grafowej bazy danych neo4j został wykorzystany moduł Spring Data Neo4. Jest to biblioteka oparta na Spring Data, która udostępnia abstrakcje do pracy z bazami danych grafowymi opartymi na Neo4j. Spring Data Neo4j umożliwia łatwe mapowanie obiektów javy na węzły i relacje w grafie Neo4j oraz zapewnia łatwy dostęp do danych za pomocą Neo4j repository. Dzięki temu możliwe jest tworzenie aplikacji z wykorzystaniem baz danych grafowych bez konieczności ręcznego tworzenia zapytań Cypher i mapowania ich na obiekty w Javie, lecz jeśli istnieje potrzeba zdefiniowania własnego zapytania, można to zrobić w bardzo prosty i intuicyjny sposób.

Aplikacja backendowa składa się z 3 głównych modułów.
Pierwszy z nich to moduł modeli, zdefiniowane zostały w nim klasy odwzorowujące modele oraz relacje pomiędzy nimi.W celu prawidłowego odwzorowania obiektów z bazy danych, w klasach modelujących zastosowane adnotacje ze spring neo4j. Adnotacja @Node oznacza, że klasa jest mapowana na węzeł w grafie Neo4j. Wszystkie pola oznaczone adnotacjami @Id i @GeneratedValue są mapowane na pole klucza głównego, a pola oznaczone adnotacjami @Relationship są mapowane na relacje z innymi węzłami w grafie.

```
@Node
public class Comment {
    @Id
    @GeneratedValue
    private Long id;

    private String content;

    @Relationship(type = "ADD_COMMENT", direction =
Relationship.Direction.INCOMING)
    @JsonIgnoreProperties(value = {"posts", "follows", "likedPosts"},
allowSetters = true)
    private User author;

    @Relationship(type = "LIKE_COMMENT", direction =
Relationship.Direction.INCOMING)
    @JsonIgnoreProperties(value = {"posts", "follows", "likedPosts"},
```

```
allowSetters = true)
    private Set<User> likes;
}
```

Listing 1. Przykład klasy Comment modelujący węzeł z bazy

Osobny podmoduł stanowią DTO czyli Data Transfer Objects, sa ta obiekty, które przenoszą dane pomiędzy warstwami aplikacji, w projekcie konkretnie jest to komunikacja pomiędzy angularem, a serwerem.

Warstwę dostępu do danych stanowi moduł repository. Zadeklarowane są w nim Neo4j Repository, interfejsy z Spring Data Neo4j, które umożliwiają łatwe tworzenie zapytań do bazy danych grafowe. Interfejsy rozszerzają interfejs Repository z Spring Data, który umożliwia tworzenie podstawowych operacji CRUD (Create, Read, Update, Delete) w bazie danych. W sytuacji gdy podstawowe operacje nie są wystarczające można zdefiniować własne zapytanie w adnotacji @Query.

```
    @Query("CREATE (comment:Comment {content:$content})\n" +
            "WITH comment\n" +
            "MATCH (user:User)\n" +
            "MATCH (post:Post)\n" +
            "WHERE id(user)=$authorId AND id(post) = $postId\n" +
            "CREATE (user)-[:ADD_COMMENT]->(comment)\n" +
            "CREATE (post)-[:HAS]->(comment)")
    void addComment(String content, Long postId, Long authorId);

    @Query("MATCH (user:User)\n" +
            "MATCH (comment:Comment)\n" +
            "WHERE id(user)=$userId AND id(comment) = $commentId\n" +
            "CREATE (user)-[:LIKE_COMMENT]->(comment)\n" +
            "RETURN comment")
    Optional<Comment> likeComment(Long commentId, Long userId);
```

Listing 2. Metody w repozytorium komentarzy.

Trzeci moduł to klasy kontrolerów. Klasa kontrolera jest kontrolerem REST w aplikacji opartej na Spring Boot. Kontroler ten umożliwia obsługę żądań HTTP przesłanych do aplikacji poprzez adnotację na metodach kontrolera, takie jak @GetMapping, @PostMapping, @PutMapping i @DeleteMapping.

```
@RestController
@RequestMapping("/api")
public class PostController {
    private final PostRepository postRepository;
```

```
    public PostController(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    @GetMapping("/posts")
    public List<Post> getAllPosts(){
        return this.postRepository.findAll();
    }

    @PostMapping("/posts")
    public List<Post> createPost(@RequestBody PostDto post){
         this.postRepository.addPost(post.getTitle(), post.getContent(),
post.getAuthorId());
         return this.postRepository.findAll();
    }
}
```

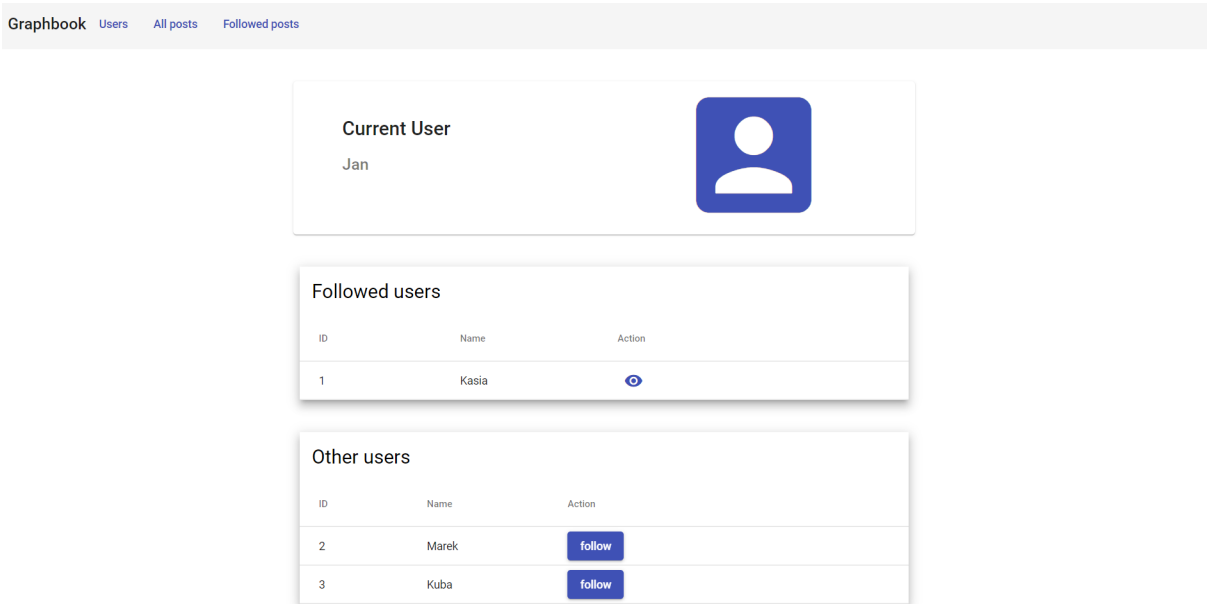
Listing 3. Fragment klasy kontrolera dla postów

## 2.3. Frontend

Cześć frontendowa to aplikacja SPA (Single Page Application) napisana w języku TypeScript przy użyciu frameworku Angular. Jedną z głównych cech angulara jest architektura oparta na komponentach, która pozwala tworzyć modułowe fragmenty kodu wielokrotnego użytku, które można łączyć w celu tworzenia złożonych aplikacji. Dlatego struktura aplikacji składa się wielu komponentów, które razem tworzą cala aplikacje.
Do symulowania przechodzenia po stronach został użyty routing angulara, który pozwala na wyświetlanie odpowiednich komponentów w zależności od ścieżki uri.
Do budowania widoków aplikacji, wykorzystana została biblioteka angular material, która wspiera tworzenie i projektowanie nowoczesnych widoków użytkownika.

# 3. Działanie aplikacji

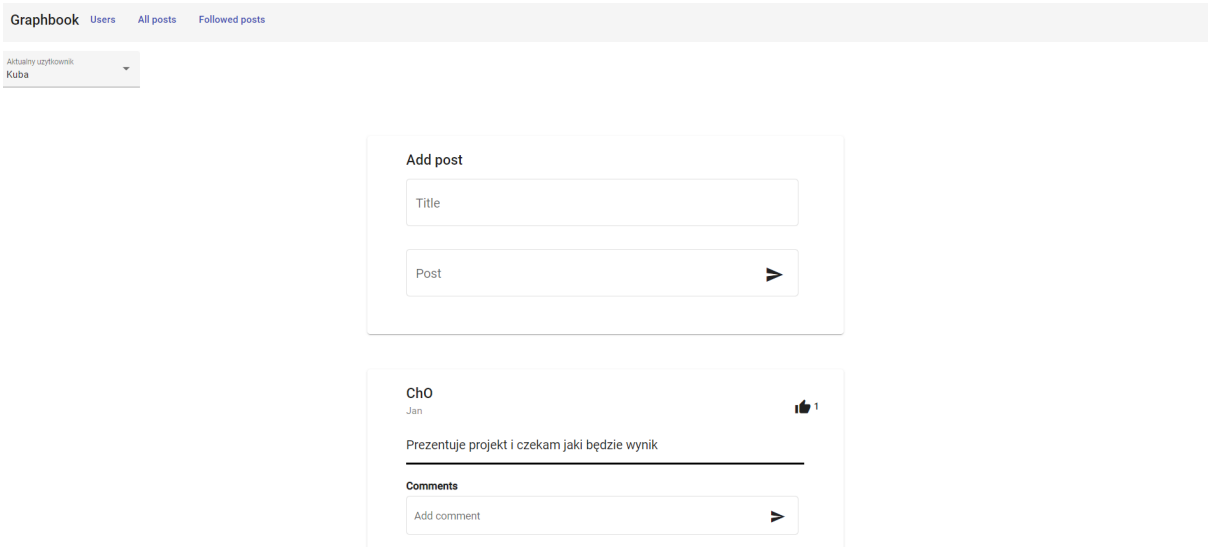
## 3.1. Widok użytkowników




Rys. 2 Widok użytkowników

Pierwszy element  na rysunku 2. to informacja o aktualnym użytkowniku korzystającym z serwisu. Następnie znajduje się lista zaobserwowanych użytkowników, a pod spodem lista pozostałych użytkowników zarejestrowanych w systemie. Po zaobserwowaniu użytkownika, lista się odświeża i trafia on listy zaobserwowanych użytkowników.

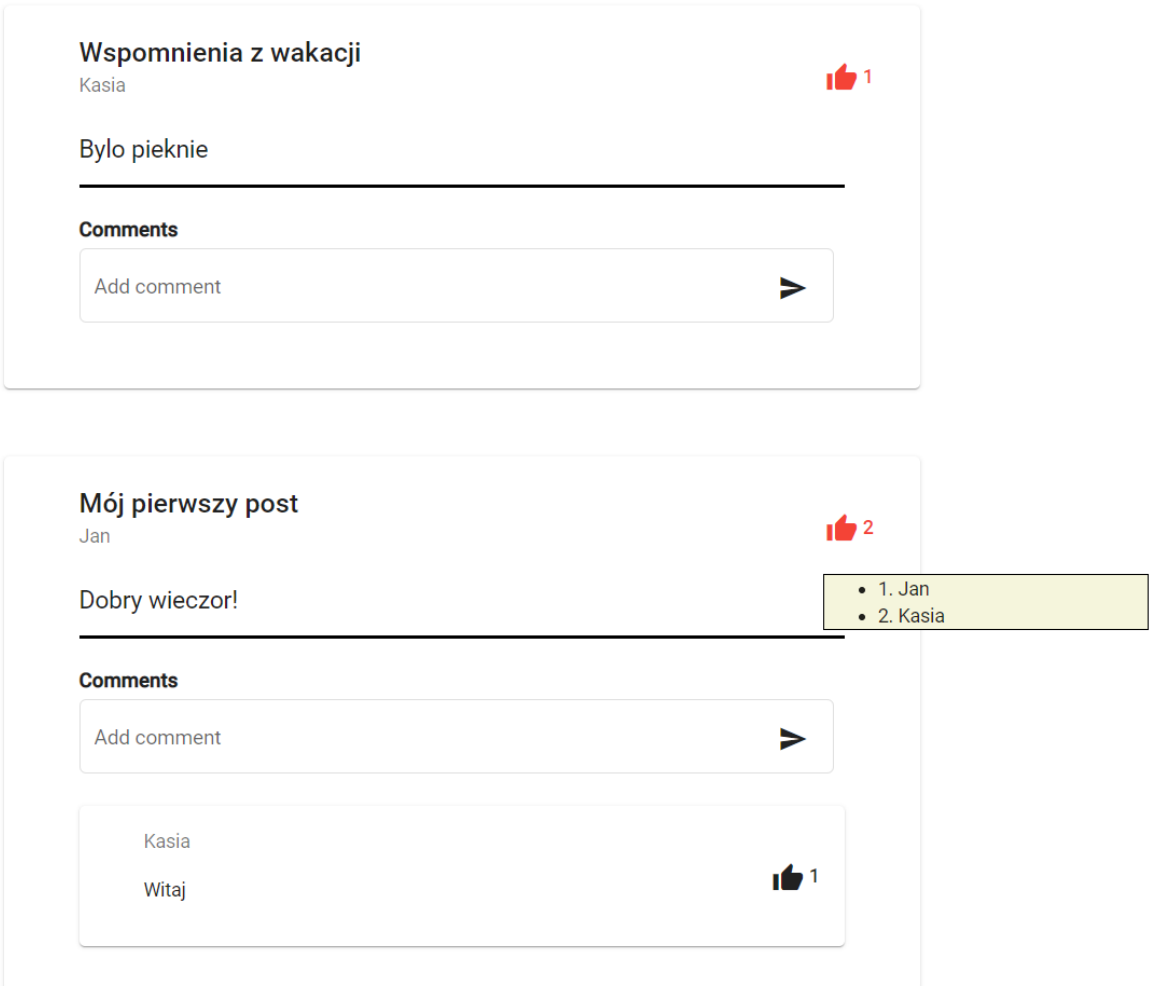
Drugi podstawowy widok to panel postów. W celu możliwości przeglądania portalu z poziomu różnych użytkowników,do aplikacji została dodana opcja wyboru aktualnego użytkownika, znajdująca się w lewym górnym rogu widoku postów.




Rys 3. Widok postów

Użytkownik może dodawać post dodając tytuł i treść postu. Użytkownik może przeglądać posty swoje i innych użytkowników. W zależności od wybranej zakładki, mogą być to posty wszystkich użytkowników, lub tylko tych których obserwuje.

Użytkownik moze dodawac komentarze do postu oraz polubienia postu i komentarzy. Po najechaniu na ikona like, wyswietla sie lista auzytkownikow, ktorzy dodali swoje polubienie. (rys 4.)




Rysunek 4. Widok listy polubień