Sprawozdanie część 1

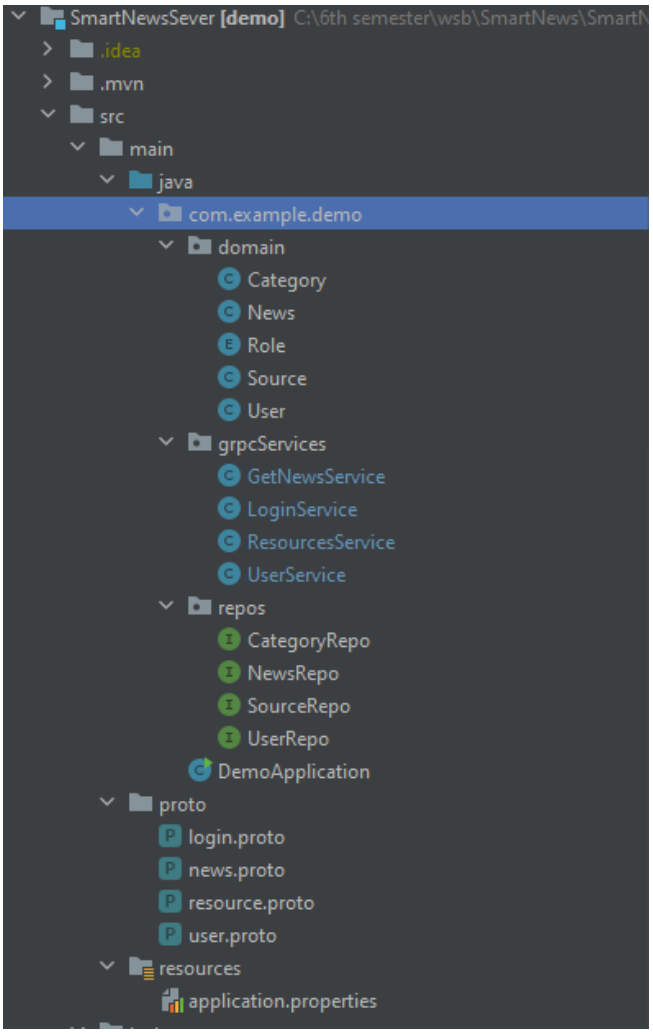
Artem Romanenko 38185

W pierwszej części sprawozdania opiszę swoją aplikację back-end która udostępnia API i wykonuje pracę z bazą danych.

Technologie użyte podczas pisania programu:

- Java
- Spring
- GRpc (Remote Procedure Calls)
- Hibernate
- PostgreSql server

Struktura projektu



Projekt składa się z :

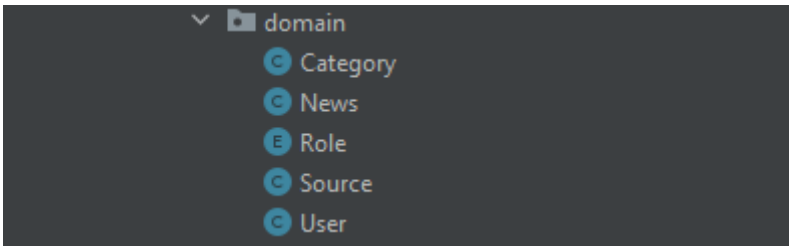
teczki domain w której jest opisana struktura bazy danych

teczki repos która zawiera interfejsy JpaRepository za ich pomocą komunikuję z bazą danych

teczki proto która zawiera .proto pliki potrzebne do opisu danych przesyłanych za pomocą GRpc.

teczki GrpcServices która zawiera implementację logiki i przesyła potrzebne dane za pomocą GRpc.

Struktura bazy danych



Baza danych składa się z 5ciu tablic, każda klasa definiuje jedną tablicę.

Klasa User

```
@Entity
@Table(name = "userok")
public class User implements UserDetails {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;
   private String username;
   private String password;
   private String email;
   private String token;

   @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
   @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
   @Enumerated(EnumType.STRING)
   private Set<Role> roles = new TreeSet<>();

   public User(String username, String password, String email){
      this.username = username;
      this.password= password;
      this.email = email;
   }

   public User() {

   }

 //Getter and Setters, Equals and HashCode override
```

Jak można zrozumieć z powyższego kodu tabela User zawiera kolumny id, username, password, email, token(potrzebny dla autentykacji)

Enum Role

```
public enum Role implements GrantedAuthority {
   ADMIN, SUPERADMIN;

   @Override
   public String getAuthority() {
      return name();
   }

  public static Role parseRoleFromString(String role) throws Exception {
      switch (role){
         case("SUPERADMIN") : return Role.SUPERADMIN;
         case("ADMIN") : return Role.ADMIN;
         default: throw new Exception("Can't parse user's role");
      }
   }
}
```

Enum Role definiuje tablicę role która jest związana z tablicą user i w tej tablice przechowywuję informację o poziomie dostępu użytkownika.

Klasa Category

```
@Entity
public class Category {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;
   private String name;

   @OneToMany(mappedBy="category", fetch = FetchType.EAGER)
   private List<News> news;

   public Category(String name) {
      this.name = name;
   }

   public Category() {
   }
// getters setter equals and hashcode here
```

Klasa Category zawiera tylko swoją nazwę i listę obiektów news

Klasa News

```
@Entity
public class News {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long id;
  private String heading;

  @OneToMany(mappedBy="news", fetch = FetchType.EAGER)
  private List<Source> sources;

  @ManyToOne
  @JoinColumn(name="category_id")
  private Category category;

  public News() {
  }

  public News(String heading, Category category) {
     this.heading = heading;
     this.category = category;
  }
```

Klasa News zawiera nagłówek, obiekt Category z którym jest związana i listę obiektów Source

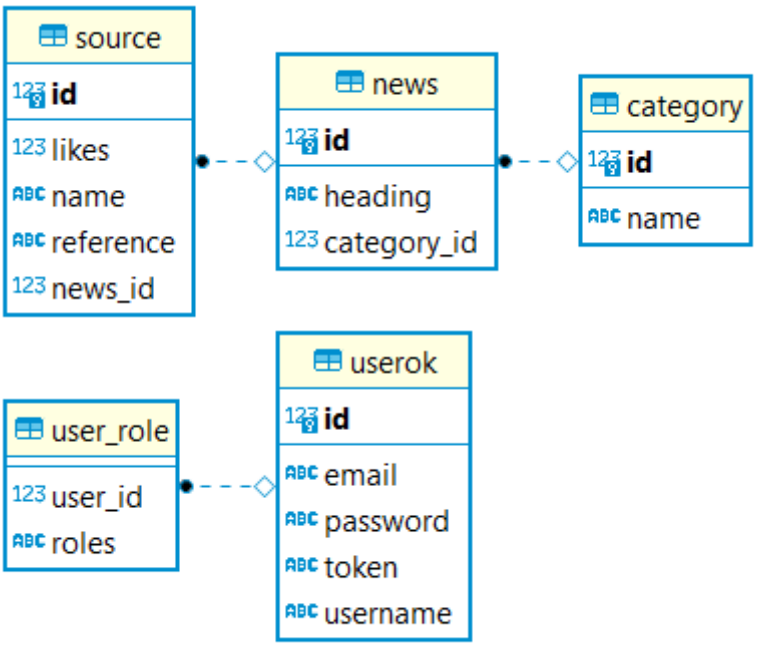
Klasa Source

```
@Entity
public class Source {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long id;

  private String name;
  private String reference;
  private Integer likes;
  @ManyToOne
  @JoinColumn(name="news_id")
  private News news;

  public Source() {
  }

  public Source(String name, String reference, News news) {
     this.name = name;
     this.reference = reference;
```

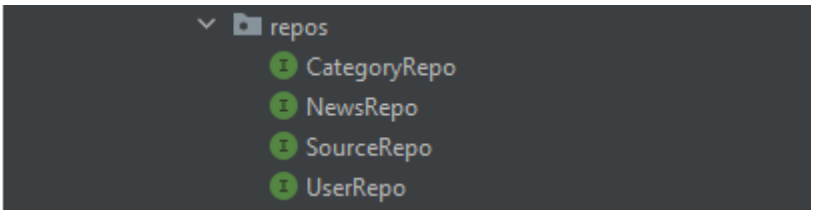
```
        this.news = news;
    }
```
// getters, setter, equals and hashcode here

Klasa Source zawiera pola id, name, likes oraz obiekt news z którym jest związana

Diagram bazy danych



Repositories



CategoryRepo

```
public interface CategoryRepo extends JpaRepository<Category, Long> {
   Category findByName(String name);
}
```

Odpowiada za komunikację z tablicą category

NewsRepo

```
public interface NewsRepo extends JpaRepository<News, Long> {
}
```

Odpowiada za komunikację z tablicą news

SurceRepo

```
public interface SourceRepo extends JpaRepository<Source, Long> {
}
```

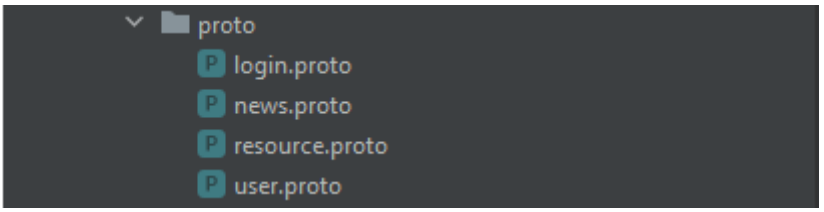
Odpowiada za komunikację z tablicą source

UserRepo

```
public interface UserRepo extends JpaRepository<User, Long> {
}
```

Odpowiada za komunikację z tablicą user

Pliki .proto



Pliki .proto używana technologiją Grpc dla zdefiniowania wszystkich możliwych żądań do serwera i odpowiedzi od serwera.

login.proto

```
syntax = "proto3";

// options for Java generated sources
option java_multiple_files = true;
option java_package = "com.thinhda.spring.grpc.core.model";


package sms.core;

message LoginRequest{
 string username = 1;
 string password = 2;
}

message LoginResponse{
 string token = 1;
}

//login service receives request that contains username and password and returns response with token,
// you should save token to the android session, you should add this token to each request, otherwise you will get 403 unauthenticated
service LoginService{
 rpc login (LoginRequest) returns (LoginResponse);
}
```

login.proto plik deklaruje że klient może wywołać na serwerze metodę login która przejmuje LoginRequest i zwraca LoginResponce.

user.proto

```
syntax = "proto3";

// options for Java generated sources
option java_multiple_files = true;
option java_package = "com.thinhda.spring.grpc.core.model";


package sms.core;

message User{
 string name =1;
 string password=2;
 string token = 3;
}
message AllUsersResponse{
 repeated User user = 1;
}
message GetUsersRequest{
 string token=1;
}

service UserService{
 rpc getAllUsers(GetUsersRequest) returns(AllUsersResponse);
}
```

user.proto deklaruje że klient może wywołać na serwerze metodę getAllUsers która przyjmuje obiekt GetAllUsersRequest i odpowiada obiektem AllUsersResponce.

resources.proto

```
syntax = "proto3";

// options for Java generated sources
option java_multiple_files = true;
option java_package = "com.thinhda.spring.grpc.core.model";


package sms.core;

message Resource{
  string name = 1;
  string reference = 2;
}
message AllResources{
  repeated Resource resource = 1;
```

```
}
message GetResourceByIdRequest{
  string id =1;
  string token =2;
}
message GetAllResourcesRequest{
  string token= 1;
}
message CreateResourceRequest{
  string name=1;
  string reference =2;
}
message CreateResourceResponse{
  bool created =1;
}

message EditResourceRequest{
  string id =1;
  string name = 2;
  string reference = 3;
}
message EditResourceResponse{
  bool updated =1;
}

service ResourcesService{
  rpc getAllResources(GetAllResourcesRequest) returns(AllResources);
  rpc getResourceById(GetResourceByIdRequest) returns(Resource);
  rpc createNewResource(CreateResourceRequest) returns(CreateResourceResponse);
  rpc editResource(EditResourceRequest) returns(EditResourceResponse);
}
```

resources.proto deklaruje że klient może wywołać metody getAllResources, getResourceById, createNewResource, editResource.

news.proto

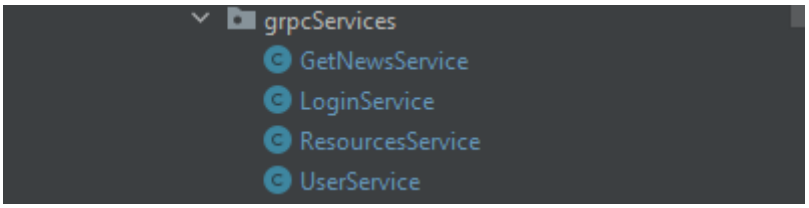
// messages declaration omitted to save space

```
service NewsService{
 rpc getNewsByCategory(GetNewsByCategoryRequest) returns(MultipleNewsResponse);
 rpc getNewsById(GetNewsByIdRequest) returns(SingleNewsResponse);
 rpc getAllNews(GetAllNewsRequest) returns(MultipleNewsResponse);
 rpc getAllCategories(GetCategoriesRequest) returns(MultipleCategoriesResponse);
 rpc deleteCategory(DeleteCategoryRequest) returns(DeleteResponse);
 rpc deleteSource(DeleteSourceRequest) returns(DeleteResponse);
```

```
 rpc deleteNews(DeleteNewsRequest) returns(DeleteResponse);
 rpc getSourcesByNews(GetSourcesByNewsRequest) returns(MultipleSourcesResponse);
 rpc createCategory(CreateCategoryRequest) returns (CreateResponse);
 rpc createNews(CreateNewsRequest) returns (CreateResponse);
 rpc createSource(CreateSourceRequest) returns(CreateResponse);
 rpc editCategory(EditCategoryRequest) returns(EditResponse);
 rpc editSource(EditSourceRequest) returns(EditResponse);
 rpc editNews(EditNewsRequest) returns(EditResponse);
}
```

deklaruje metody które klient może wywołać na serwerze, metody operują z obiektami news, sources oraz category.

Grpc services



NewsService implementuje metody zdefiniowane w news.proto

Ta klasa jest duże więc żeby oszczędzić miejsce pokaże implementację tylko jednej metody. Całą klasę można zobaczyć w kodzie dołączonym do sprawozdania.

```
@Override
public void getNewsByCategory(GetNewsByCategoryRequest request,
StreamObserver<MultipleNewsResponse> responseObserver) {
   String categoryName = request.getCategory();
   Category category = categoryRepo.findByName(categoryName);

   List<News> news = category.getNews();

   MultipleNewsResponse.Builder responseBuilder = MultipleNewsResponse.newBuilder();
```

```
  for(int i=0; i<news.size(); i++){
     com.thinhda.spring.grpc.core.model.News.Builder newsGrpcBuilder =
com.thinhda.spring.grpc.core.model.News.newBuilder()
           .setId(String.valueOf(news.get(i).getId()))
           .setHeading(news.get(i).getHeading());
     for(int j=0; j< news.get(i).getSources().size(); j++){
        com.example.demo.domain.Source source = news.get(i).getSources().get(j);
        com.thinhda.spring.grpc.core.model.Source sourceGrpc = Source.newBuilder()
              .setId(String.valueOf(source.getId()))
              .setName(source.getName())
              .setReference(source.getReference())
              .setLikes(source.getLikes())
              .build();
        newsGrpcBuilder.addSources(j, sourceGrpc);
     }
     com.thinhda.spring.grpc.core.model.News newsGrpc = newsGrpcBuilder.build();
     responseBuilder.addNews(i, newsGrpc);
  }
  MultipleNewsResponse response = responseBuilder.build();
  responseObserver.onNext(response);
  responseObserver.onCompleted();
}
```

Powyższa metoda przyjmuje obiekt GetNewsByKategoryRequest, “wyciąga” z niego nazwę kategorii, znajduje wszystkie obiekty news związane z tą kategorią i wysyła do klienta za pomocą metody onNext().
Pozostałe metody są podobne do tej.

LoginService

```
@GRpcService
public class LoginService extends LoginServiceGrpc.LoginServiceImplBase {
   private final UserRepo userRepo;

   public LoginService(UserRepo userRepo) {
       this.userRepo = userRepo;
   }

   @Override
   public void login(LoginRequest request, StreamObserver<LoginResponse>
responseObserver) {
       User user = userRepo.findByUsername(request.getUsername());
       LoginResponse.Builder responseBuilder = LoginResponse.newBuilder();
       if (user!=null&& user.getPassword().equals(request.getPassword())) {
           String uuid = UUID.randomUUID().toString();
           user.setToken(uuid);
           userRepo.save(user);
```

```
            responseBuilder.setToken(uuid);
        }else{
            responseBuilder.setToken("denied");
        }
        responseObserver.onNext(responseBuilder.build());
        responseObserver.onCompleted();
    }
}
```

Ta klasa zawiera metodę login która sprawdza przesłane username i password i wysyła klientowi token potrzebny dla dalszej autentykacji. Token będzie przechowywany w sesji użytkownika I dostępny do następnego logowania.

ResourcesService

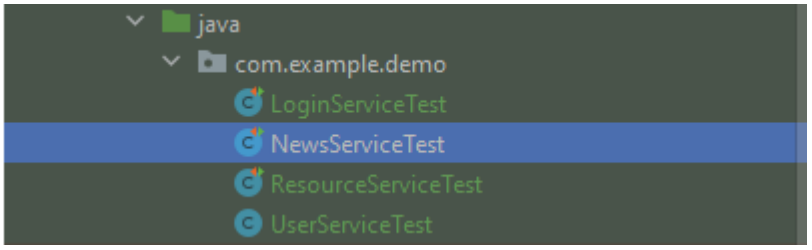
```
@GRpcService
public class ResourcesService extends ResourcesServiceGrpc.ResourcesServiceImplBase {
    private final ResourceRepo resourceRepo;

    public ResourcesService(ResourceRepo resourceRepo) {
        this.resourceRepo = resourceRepo;
    }

    @Override
    public void getAllResources(GetAllResourcesRequest request, StreamObserver<AllResources>
responseObserver) {
        List<com.example.demo.domain.Resource> resources = resourceRepo.findAll();
        AllResources.Builder response = AllResources.newBuilder();
        for (int i=0; i<resources.size(); i++){
            Resource resource = Resource.newBuilder()
                    .setName(resources.get(i).getName())
                    .setReference(resources.get(i).getReference())
                    .build();
            response.addResource(i, resource);
        }
        responseObserver.onNext(response.build());
        responseObserver.onCompleted();
    }

    @Override
    public void getResourceById(GetResourceByIdRequest request, StreamObserver<Resource>
responseObserver) {
        Long id = Long.valueOf(request.getId());
        com.example.demo.domain.Resource resource = resourceRepo.findById(id).get();
        if(resource ==null) {
            responseObserver.onError(new StatusException(Status.NOT_FOUND));
            return;
        }
```

```
        Resource response = Resource.newBuilder()
                .setName(resource.getName())
                .setReference(resource.getReference())
                .build();
        responseObserver.onNext(response);
        responseObserver.onCompleted();
    }

    @Override
    public void createNewResource(CreateResourceRequest request,
StreamObserver<CreateResourceResponse> responseObserver) {
        com.example.demo.domain.Resource resource = new
com.example.demo.domain.Resource(request.getName(), request.getReference());
        resourceRepo.save(resource);
        CreateResourceResponse response = CreateResourceResponse.newBuilder()
                .setCreated(true)
                .build();
        responseObserver.onNext(response);
        responseObserver.onCompleted();
    }

    @Override
    public void editResource(EditResourceRequest request,
StreamObserver<EditResourceResponse> responseObserver) {
        com.example.demo.domain.Resource resource  =
resourceRepo.findById(Long.parseLong(request.getId())).get();
        if (resource==null){
            responseObserver.onError(new StatusException(Status.NOT_FOUND));
            return;
        }else {
            resource.setName(request.getName());
            resource.setReference(request.getReference());
            resourceRepo.save(resource);
            EditResourceResponse response = EditResourceResponse.newBuilder()
                    .setUpdated(true)
                    .build();
            responseObserver.onNext(response);
            responseObserver.onCompleted();
        }
    }
```

Implementuje CRUD metody związane z obiektami Resource, editResource, crateResource, getById, getAll. Dane są przesyłane za pomocą technologii grpc.

Testy

Żeby sprawdzić działanie programu napisałem 4 klasy z testami.



Klasa NewsServiceTest zawiera testy wszystkich metod Klasy NewsService

przykład :

```
@Test
public void getNewsByCategory(){
  GetNewsByCategoryRequest request = GetNewsByCategoryRequest.newBuilder()
        .setToken("token")
        .setCategory("Sport")
        .build();
  ManagedChannel channel;
  NewsServiceGrpc.NewsServiceBlockingStub stub;
  channel = ManagedChannelBuilder.forTarget("localhost:6565").usePlaintext().build();
  stub = NewsServiceGrpc.newBlockingStub(channel);
  MultipleNewsResponse response = stub.getNewsByCategory(request);

Assert.assertTrue(response.getNewsCount()==categoryRepo.findByName("Sport").getNews().size());
}
@Test(expected = StatusRuntimeException.class)
public void getNewsByCategory_CategoryNameDoNotExist(){
  GetNewsByCategoryRequest request = GetNewsByCategoryRequest.newBuilder()
        .setToken("token")
        .setCategory("Sportttt")
        .build();
  ManagedChannel channel;
  NewsServiceGrpc.NewsServiceBlockingStub stub;
  channel = ManagedChannelBuilder.forTarget("localhost:6565").usePlaintext().build();
  stub = NewsServiceGrpc.newBlockingStub(channel);
  MultipleNewsResponse response = stub.getNewsByCategory(request);
```

```
}
@Test(expected = StatusRuntimeException.class)
public void getNewsByCategory_ExistedToken_ExceptionReturned(){
  GetNewsByCategoryRequest request = GetNewsByCategoryRequest.newBuilder()
        .setToken("t")
        .setCategory("Sport")
        .build();
  ManagedChannel channel;
  NewsServiceGrpc.NewsServiceBlockingStub stub;
  channel = ManagedChannelBuilder.forTarget("localhost:6565").usePlaintext().build();
  stub = NewsServiceGrpc.newBlockingStub(channel);
  MultipleNewsResponse response = stub.getNewsByCategory(request);
}
```

Wszystkie testy można przeczytać w kodzie źródłowym.

W następnym sprawozdaniu opiszę część frontend mojego projektu i proces przesyłania danych między backendem a frontendem.

Dziękuję za przeczytanie.