

Boğaziçi University

bespoke Web Application

SWE574



Cihangir Özmüş - 2017719102
Çağrı Altuğ – 2017719003
Gülsah Çoşkun – 2018719033
Serhat Uzunçavdar – 2017719120

12-23-2019

Table of Contents

WHAT IS BESPOKE?	1
WHAT IS ACTIVITY STREAM 2.0?	2
EXAMPLE OF ACTIVITY	3
IMPLEMENTATIONS.....	3
WHAT IS W3C WEB ANNOTATION DATA MODEL?	4
MODEL	4
EXAMPLE OF WEB ANNOTATION MODELS	5
RELATED WORK	7
TECHNOLOGY STACK.....	8
FRONTEND.....	8
BACKEND.....	8
DEPLOYMENT	8
DATABASE	8
TEST	8
IDEA	8
HOW TO START PROJECT IN A LOCAL MACHINE	9
DATABASE CREATION	9
HOW TO START FRONTEND	9
HOW TO START BACKEND	10
HOW TO START ANNOTATION SERVER LOCALLY	10
<i>Prerequisites</i>	<i>10</i>
<i>Technology Stack:</i>	<i>10</i>
<i>Configuration</i>	<i>11</i>
<i>In order to run Elucidate Server correctly some properties should be configured. Those are;</i>	<i>11</i>
<i>Database and Security</i>	<i>11</i>
<i>Building</i>	<i>11</i>
<i>Deployment.....</i>	<i>12</i>
<i>Tomcat should be configured in the IDE.</i>	<i>12</i>
HOW TO DEPLOY PROJECT	14
INSTALLATION.....	14
BUILD BACKEND	14
DATABASE CONFIGURATION	14
RUN WITHOUT DOCKER	15
<i>Development environment:</i>	<i>15</i>
BUILD AND RUN WITH DOCKER	16
RUNNING INSTANCES	16
CONTINUOUS INTEGRATION & CONTINUOUS DEPLOYMENT.....	18
STATIC CODE ANALYSIS	19
USER MANUAL.....	20
REGISTER / LOGIN	20
CREATING NEW TOPIC WITH WIKIDATA	22
CREATING NEW CONTENT.....	24

CREATING QUESTIONS.....	26
ENROLL TO TOPIC.....	27
FOLLOWING A USER.....	29
MAKING ANNOTATIONS	29
PROJECT REQUIREMENTS.....	31
GLOSSARY.....	31
REQUIREMENTS	32
1. <i>Non-Functional Requirements</i>	32
2. <i>Functional Requirements</i>	32
3. <i>Topic Requirements</i>	32
4. <i>API Requirements</i>	33
5. <i>Quiz Requirements</i>	33
6. <i>User Requirement</i>	33
7. <i>Activity Requirements</i>	34
8. <i>Annotation Requirements</i>	34
9. <i>Recommendation System</i>	34
SEQUENCE DIAGRAMS	35
CREATE ANNOTATION	35
GET ANNOTATIONS	35
DELETE ANNOTATION	36
SAVE ACTIVITY.....	36
GET NOTIFICATIONS	37
GET RECOMMENDATIONS	37
ER DIAGRAMS	38
CLASS DIAGRAMS	39
FOLDER STRUCTURE.....	51
CONCLUSION	52
APPENDIX	53
RESEARCH: GIT AND GitHub.....	53
RESEARCH: REST	54
OVERVIEW OF SECURITY MECHANISM	55
How To Use POSTMAN WITH JWT TOKEN	55

Table of Figures

Figure 1 - Activity Example	3
Figure 2 - Annotation Data Model	4
Figure 3 - Basic Annotation Model	5
Figure 4 - Typing of Body and Target.....	5
Figure 5 - Choice	5
Figure 6 - Text Position Selector	6
Figure 7 - CSS Selector	6
Figure 8 - Multiple Bodies or Targets.....	6
Figure 9 - Textual Body	7
Figure 10 - bespoke in localserver	9
Figure 11 - bespoke backend	10
Figure 12 - Annotation Server Tomcat Configuration.....	12
Figure 13 - Annotation Tomcat Server.....	12

Figure 14 - AWS EC2 for Backend	16
Figure 15 - AWS S3 for Frontend	16
Figure 16 - AWS RDS for Database.....	17
Figure 17 - Annotation Database on AWS	17
Figure 18 - Circle CI	18
Figure 19 - Full Pipeline Report.....	18
Figure 20 - Sonar Cloud.....	19
Figure 21 - Sequence Diagram for Create Annotation.....	35
Figure 22 - Sequence Diagram for Get Annotations	35
Figure 23 - Sequence Diagram for Delete Annotation.....	36
Figure 24 - Sequence Diagram for Save Activity	36
Figure 25 - Sequence Diagram for Get Notifications	37
Figure 26 - Sequence Diagram for Get Recommendations.....	37
Figure 27 - Entity Diagram	38
Figure 28 - client package	39
Figure 29 - client/request package	40
Figure 30 - client/response package	41
Figure 31 - config package	42
Figure 32 - controller package	43
Figure 33 - controller/dto/request package	44
Figure 34 - controller/dto/response	45
Figure 35 - converter package	46
Figure 36 - exception package	46
Figure 37 - persistence package	47
Figure 38 - persistence/model package	48
Figure 39 - security package	49
Figure 40 - service/implementation package	50
Figure 41 - Frontend Folder Structure	51
Figure 42 - Backend Folder Structure	51

What is bespoke?

Bespoke is about creating a learning space for learners and teachers. It focused on query-based approach. Users will be able to sign up to the platform.

User can create topics with at least one basic level including at least one question and add more difficult levels to the topic. All topics will be listed in a glossary.

Learners can search through the topics and enroll. Topic shall have at least one basic level with at least one question.

Application will create activity streams and recommends related topics to the users via wikidata semantic tagging system.

Project will be deployed online and to be able to accessible via browser.

All project related files can be found at <https://github.com/altugcagri/boun-swe-574>.

What is Activity Stream 2.0?

An Activity Stream can be defined as a list of recent activities performed by user on a single website. For example, Facebook's news stream is an activity stream. Other major websites have their similar implementation.

With the development of social media, the activity stream has become a common way to present this type of aggregated information to users.

Tools like the Stream API allow developers to easily build activity streams into their projects and platforms. These resources make it easy to update the user according to what other people are doing on various platforms, or what is happening elsewhere on the Web.

An "Activity" is a semantic description of an action. Goal of Activity Stream 2.0 is to specify a JSON-based syntax which can express metadata about activities in a human friendly version but machine-processable.

Activity Stream 2.0 is suitable as a social data syntax.

The first version was published in May 2011. Some of the differences from the first version is listed below for the current version (2.0):

- Multi-lingual representation of activities
- Unification of "verb" and "objectType" to "type"
- Removal of activity types and object types that weren't core to social use cases
- Introduction of the Link type for richly described links
- Incorporation of audience targeting into the core spec
- The generalized "Undo" activity type, so all activities can be undone
- Consistent collection and paging representation
- Formalizing the namespace of the base vocabulary of object types and activity types
- Extensibility framework for other types and properties
- Compatibility with JSON-LD

Activity Streams 2.0 documents must be serialized using the UTF-8 character encoding. An Activity Streams Document is a JSON document whose root value is an Activity Streams Object of any type, including a Collection, and whose MIME media type is "application/activity+json".

JSON-LD uses the special @context property to define the processing context. The value of the @context property is defined by the [JSON-LD] specification. Implementations producing Activity Streams 2.0 documents should include a @context property with a value that includes a reference to the normative Activity Streams 2.0 JSON-LD @context definition using the URL "<https://www.w3.org/ns/activitystreams>". Implementations may use the alternative URL "<http://www.w3.org/ns/activitystreams>" instead. This can be done using a string, object, or array.

Example of Activity

```
{  
  "@context": "https://www.w3.org/ns/activitystreams",  
  "summary": "Joe liked a note",  
  "type": "Like",  
  "id": "http://www.test.example/activity/1",  
  "actor": "http://example.org/profiles/joe",  
  "object": "http://example.com/notes/1",  
  "published": "2014-09-30T12:34:56Z"  
}
```

Figure 1 - Activity Example

Implementations

Conforming implementations are software that publish, store, analyze, consume or otherwise process conforming documents. The two main kinds of implementations are publishers and consumers.

A non-exhaustive list of example publishers includes:

- A social network
- A personal web sites
- A document publishing system
- A bridge from a non-conforming social network
- A document converter from similar document types such as RSS or Atom

A non-exhaustive list of example consumers includes:

- A social network
- A search engine
- A feed reader
- A document validator
- A feed aggregator
- A statistical analyzer

What is W3C Web Annotation Data Model?

An annotation is a web resource which provides association of meta-information to resources. Traditionally, it's a practice among readers to underline text, highlight sections or write comments in the margins. Readers may explain their opinions via such annotations. Since authors or publishers of content cannot censor people, they can express their thoughts freely.

Web Annotation is a standard for creating similar annotations on the web content. It's standardized by the W3C Web Annotation Working Group. It reuses concepts and tools from Semantic Web. W3C web annotation data model (WADL)'s objective is to bring a general format to web annotation so that they can be interoperable.

The specification od W3C provides a specific JSON format for ease of creation and consumption of annotations. An annotation is a rooted directed graph that establishes relationships among resources. A resource can be either a body or a target. Typically, an annotation has a single body, which is a comment or other descriptive resource, and a single target that the Body is somehow "about". The Annotation likely also has additional descriptive properties.

Example Use Case: Ali has written a post that makes a comment about a particular web page. His client creates an annotation with the post as the body resource, and the web page as the target resource.

Since resources are distributed on the web, they are identified using IRI (Internationalized Resource Identifier). Users often want to select part of a resource and not the entire content at the IRI. This is called Segment (of Interest). A Selector is used to extract the segment from a resource. For example, selectors are available to select some region of an image; an exact quote; content that matches a CSS or XPath rule; some text by its start and end positions; and so on.

Model

Term	Type	Description
@context	Property	The context that determines the meaning of the JSON as an Annotation. The Annotation must have 1 or more @context values and http://www.w3.org/ns/anno.jsonld must be one of them. If there is only one value, then it must be provided as a string.
id	Property	The identity of the Annotation. An Annotation must have exactly 1 IRI that identifies it.
type	Relationship	The type of the Annotation. An Annotation must have 1 or more types, and the Annotation class must be one of them.
Annotation	Class	The class for Web Annotations. The Annotation class must be associated with an Annotation using type.
body	Relationship	The relationship between an Annotation and its Body. There should be 1 or more body relationships associated with an Annotation but there may be 0.
target	Relationship	The relationship between an Annotation and its Target. There must be 1 or more target relationships associated with an Annotation.

Figure 2 - Annotation Data Model

Example of Web Annotation Models

```
EXAMPLE 1: Basic Annotation Model
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno1",
  "type": "Annotation",
  "body": "http://example.org/post1",
  "target": "http://example.com/page1"
}
```

Figure 3 - Basic Annotation Model

```
EXAMPLE 2: Typing of Body and Target
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno3",
  "type": "Annotation",
  "body": {
    "id": "http://example.org/video1",
    "type": "Video"
  },
  "target": {
    "id": "http://example.org/website1",
    "type": "Text"
  }
}
```

Figure 4 - Typing of Body and Target

```
EXAMPLE 3: Choice
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno10",
  "type": "Annotation",
  "body": {
    "type": "Choice",
    "items": [
      {
        "id": "http://example.org/note1",
        "language": "en"
      },
      {
        "id": "http://example.org/note2",
        "language": "fr"
      }
    ]
  },
  "target": "http://example.org/website1"
}
```

Figure 5 - Choice

```

EXAMPLE 4: Text Position Selector
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno24",
  "type": "Annotation",
  "body": "http://example.org/review1",
  "target": {
    "source": "http://example.org/ebook1",
    "selector": {
      "type": "TextPositionSelector",
      "start": 412,
      "end": 795
    }
  }
}

```

Figure 6 - Text Position Selector

```

EXAMPLE 5: CSS Selector
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno21",
  "type": "Annotation",
  "body": "http://example.org/note1",
  "target": {
    "source": "http://example.org/page1.html",
    "selector": {
      "type": "CssSelector",
      "value": "#elemid > .elemclass + p"
    }
  }
}

```

Figure 7 - CSS Selector

```

EXAMPLE 6: Multiple Bodies or Targets
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno9",
  "type": "Annotation",
  "body": [
    "http://example.org/description1",
    {
      "type": "TextualBody",
      "value": "tag1"
    }
  ],
  "target": [
    "http://example.org/image1",
    "http://example.org/image2"
  ]
}

```

Figure 8 - Multiple Bodies or Targets

```

EXAMPLE 7: Textual Body
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno5",
  "type": "Annotation",
  "body": {
    "type" : "TextualBody",
    "value" : "<p>j'adore !</p>",
    "format" : "text/html",
    "language" : "fr"
  },
  "target": "http://example.org/photo1"
}

```

Figure 9 - Textual Body

Related Work

Hypothesis - Annotation of the web, with anyone, anywhere.

Europeana Labs - Transform cultural heritage of europe and make it easier to use.

Apache Annotator - Provides annotation enabling code for browsers, servers, humans.

Open Annotation - Previous data model that lead to W3C WADL.

Annotorious - One such plugin for image annotation.

Pundit Annotator - Based on AngularJS and comes with open source client code which can be downloadable.

Annotator is being used in many projects: Hypothes.is, Harvard's Open Video Annotation Project, EdX, MIT's Annotation Studio, WritingPod, Crunched Book and many more.

Technology Stack

Project is designed as a REST API. For these purpose different technologies are used together. Details can be seen below:

Frontend

- Npm for package management
- Node as javascript engine
- React for UI/UX
- HTML and CSS for design



Backend

- Gradle for dependency management
- Java for Object Oriented Programming
- Spring Boot as Web Framework



Deployment

- Amazon EC2 as REDHAT machine
- Amazon RDS for PostgreSQL database



Database

- PostgreSQL as Database
- Hibernate for ORM



Test

- JUnit for unit testing
- Postman for endpoint tests



Idea

- IntelliJ IDEA for backend
- WebStorm IDEA for frontend



How to Start Project in a Local Machine

Clone repository to local machine.

```
git clone https://github.com/altugcagri/boun-swe-574
```

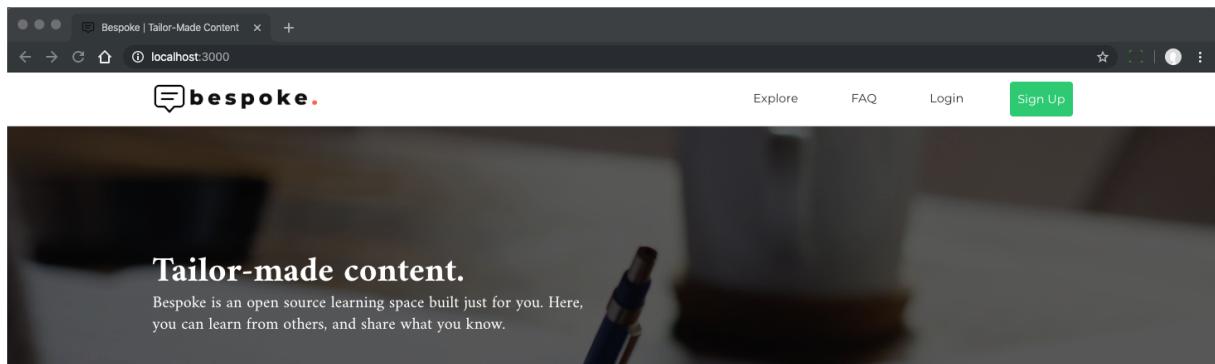
Database Creation

- Install PostgreSQL
- Create "POSTGRESQL" database from terminal with root user
- Or use dockerfile for Postgresql image

How to Start Frontend

- Go in to "\boun-swe-574\frontend" folder
- npm install
- npm audit fix
- npm start

Frontend will be started in development mode locally. Bespoke web application will be opened in default browser automatically.



Latest.



First Topic

20-10-2019 • by • @tallyye

Lorem ipsum dolor sit amet consectetur adipisicing elit.

dsa

[Explore all topics](#)

You might be interested.

Figure 10 - bespoke in localserver

How to Start Backend

- Open "\boun-swe-574\backend" folder with IntelliJ
- Run the application

You will see “Started BespokeApplication in ...” logs.

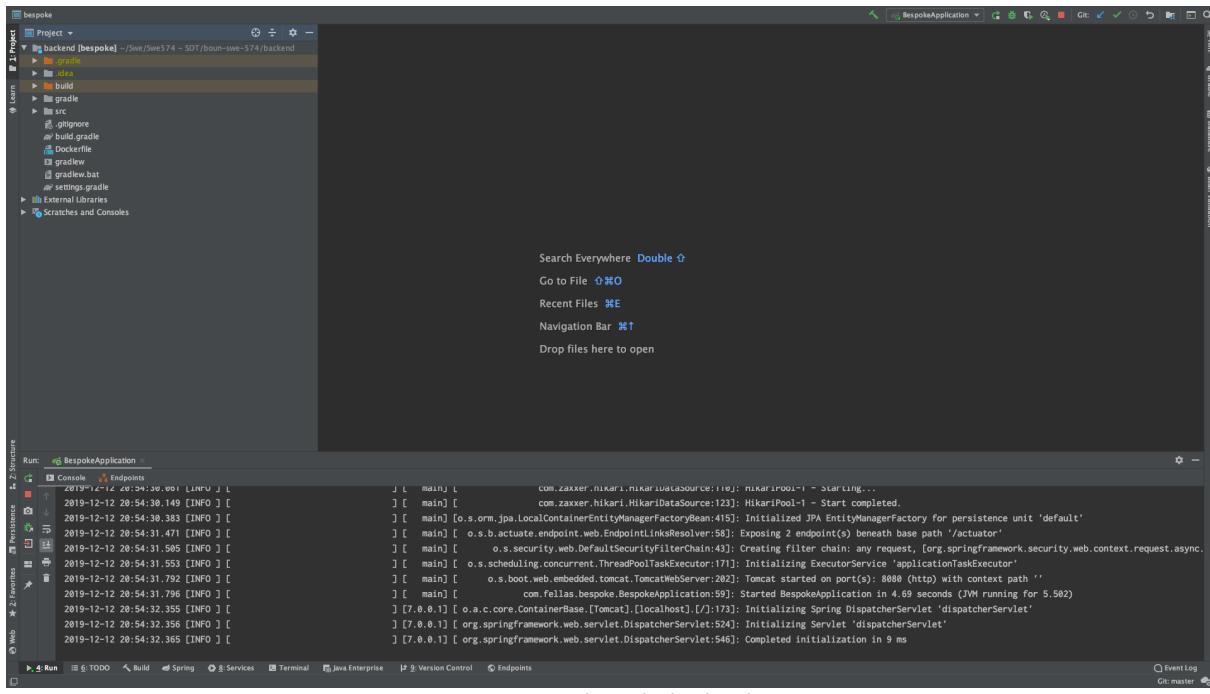


Figure 11 - bespoke backend

How to Start Annotation Server Locally

In Bespoke, an open source annotation server project has been chosen. The name of the project is “elucidate-server”. The source code of the project and all the related information can be accessed from the following Github link. → <https://github.com/dlcs/elucidate-server>

Elucidate is a web annotation server which is complaint with W3C Web Annotation Data Model and its protocol.

Prerequisites

- Java 8
- Apache Tomcat 8
- PostgreSQL 9.4
- Maven 3.6

Technology Stack:

- Spring Framework
- Jackson
- JSONLD-JAVA

- JSON Schema Validator
- Liquibase

Configuration

In order to run Elucidate Server correctly some properties should be configured. Those are;

- **db.user** → required to authenticate database
- **db.url** → JDBC url required to connect to database
- **base.scheme** → uri scheme to be used by annotation IRIs
- **base.host** → the hostname
- **base.port** → the port number
- **base.path** → the path prefix
- **auth.token.verifierType** → whether the JWT verification key is a shared secret or an RSA public key.
- **auth.token.verifierKey** → The secret or public key used to verify signed tokens.
- **auth.token.uidProperties** → The name of the JWT property that represents a unique user ID

Database and Security

Elucidate Server has been built and tested against PostgreSQL 9.4+

A [Liquibase](#) changelog contains the SQL scripts required to create the Elucidate Server schema. On first connection to a JDBC URI the changes will be applied, and a changelog table created in the database for any subsequent runs.

Elucidate Server supports user authentication and authorization using detached JWTs as credentials. Authentication can be enabled or disabled by changing the [auth.enabled](#) property. Those tokens can be verified using either a shared secret key or RSA public key.

Building

There are 4 main modules in Elucidate project. Those are as follows;

- elucidate-parent
 - Parent Maven project that defines dependency library version numbers and common dependencies amongst all Elucidate projects.
- elucidate-common-lib
 - Contains common classes that are used by similar projects.
- elucidate-converter
 - Simple library that allows for conversion between a W3C Web Annotation and OA Web Annotation.
- elucidate-server
 - Main application where all the annotation creation and retrieval processes are handled with controllers and services

Each module including elucidate-server should be built using Maven using following command;

```
mvn clean package install -U
```

Deployment

Tomcat should be configured in the IDE.

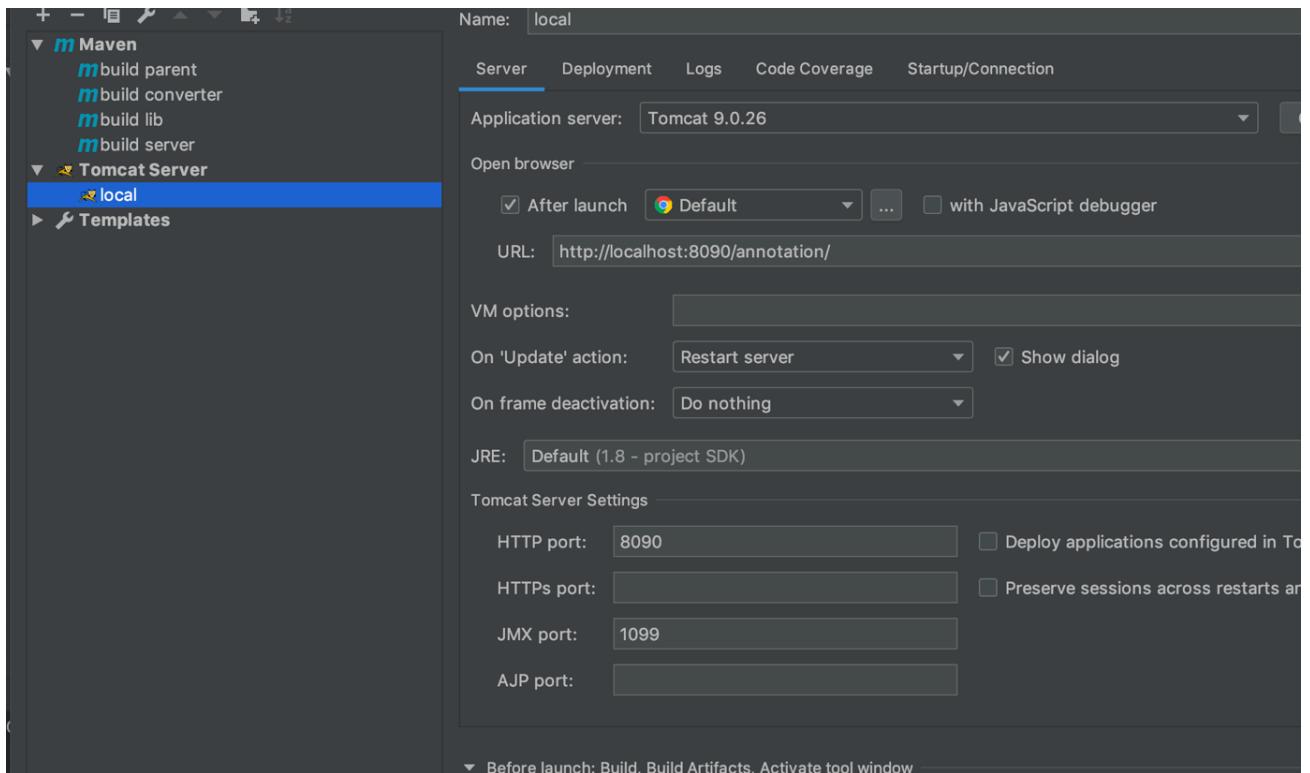


Figure 12 - Annotation Server Tomcat Configuration

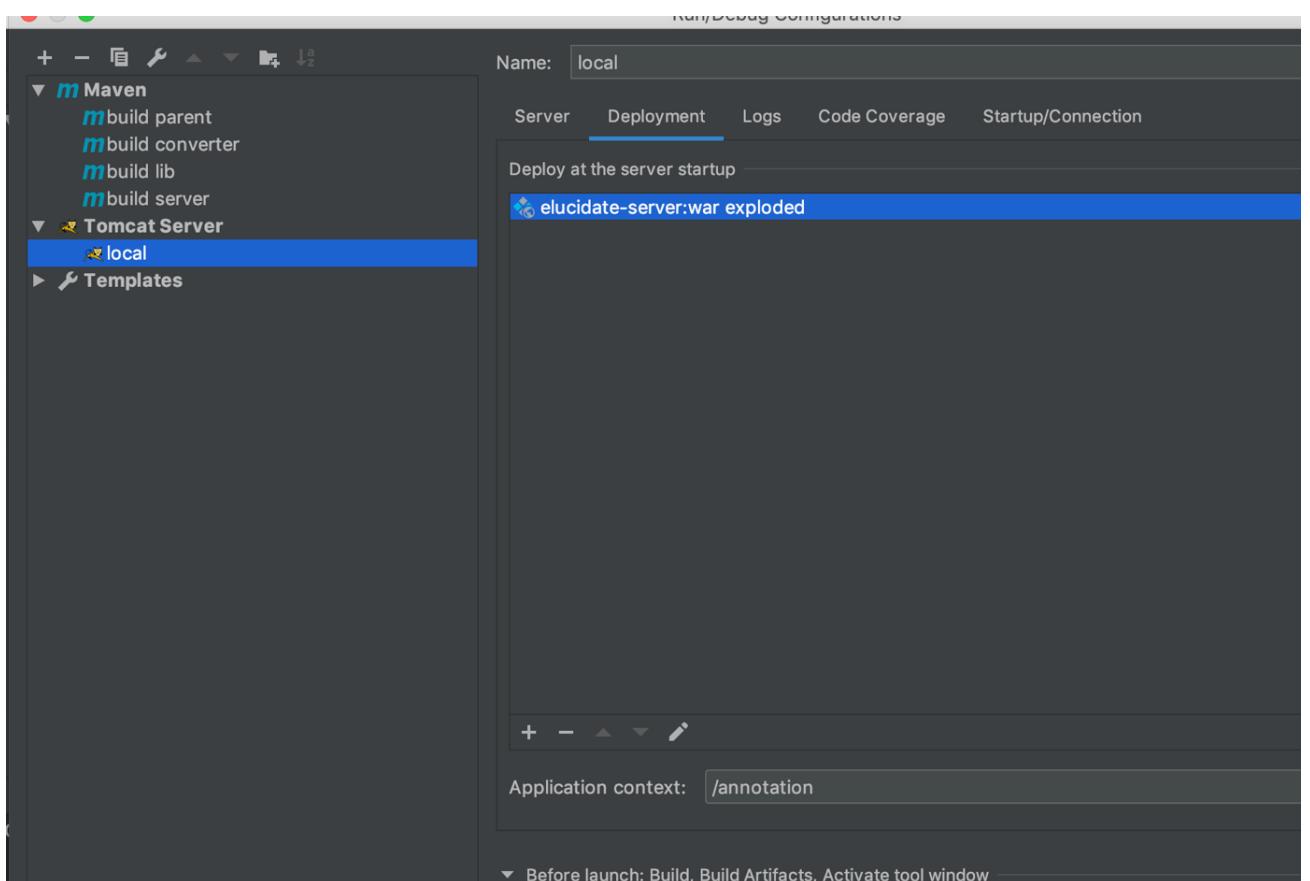


Figure 13 - Annotation Tomcat Server

After building the modules with Maven, generated war file of annotation.war can be found under target folder. The war should be selected for the deployment using Tomcat.

When project is run using this Tomcat configuration, war file will be deployed, and the application can be reached using following url:

→ localhost:8090/annotation/

How to Deploy Project

Installation

In order to install and run the backend project Java Jdk v11+, Gradle and Postgresql database must be installed.

Note that: Postgresql database is not mandatory. Any other database can be used. However database driver dependencies must be added to build.gradle file and database configurations must be added to application.properties file.

Database configuration explained below.

In order to install and run the frontend project Node must be installed

Set JAVA_HOME like "C:\Program Files\Java\jdk-11.0.2", "bin" folder is not required. Otherwise gradle tasks can fail.

In order to run the projects on docker environment Docker must be installed.

After installation of Java, Gradle and Docker, you can clone the project from the repository into your workspace.

Build BackEnd

After cloning the project go to workspace and run the gradle command:

\$.\backend\gradle build

in order to run tests, run the gradle command:

\$.\backend\gradle test

Builds the backend for production to the backend/build/libs/folder as com.fellas.bespoke-0.0.1-SNAPSHOT.jar.

Database Configuration

In order to make database configurations for local environment, you should open backend\src\main\resources\applicaion-db-dev.properties file and change the parameters below. After making changes you should run the application with dev profile. Running with profiles is explained under running the backend sections

```
spring.datasource.url= jdbc:postgresql://{{DATABASE_URL}}/{{DATABASE_NAME}}
spring.datasource.username= {{USERNAME_NAME}}
spring.datasource.password= {{PASSWORD}}
```

{DATABASE_URL} is the url of the database. it is mostly localhost:5432 for the local environments
{DATABASE_NAME} is the name of the database that you created on database server for the application. If you did not, first create a database on the server.

{USERNAME_NAME} is the username of your database.

{PASSWORD} is the password of your database.

In order to see database configurations for prod environment, you should open **backend\src\main\resources\applicaion-db-prod.properties** file. You should not change anything in this file. The parameters defined below should be defined as environment variable on the server.

You should run the application with prod profile. Running with profiles is explained under running the backend sections

```
spring.datasource.url=${DATABASE_HOST}
spring.datasource.username=${DATABASE_USER_NAME}
spring.datasource.password=${DATABASE_PASS}
```

`${DATABASE_HOST}` is the datasource url of the database. it must be defined as environment variable on the server named DATABASE_HOST. It is constructed like `jdbc:postgresql://${DATABASE_URL}/${DATABASE_NAME}`. If you did not create database, first create a database on the server.

`${DATABASE_USER_NAME}` is the username of your database. it must be defined as environment variable on the server named DATABASE_USER_NAME

`${DATABASE_PASS}` is the password of your database. it must be defined as environment variable on the server named DATABASE_PASS

Please see to create environment variables on the server. <https://www.geeksforgeeks.org/environment-variables-in-linux-unix/>

If you like to use another database please see <https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-sql>

Run Without Docker

After building the project running the project requires the steps below;

Development environment:

```
$ java -jar -Dspring.profiles.active=dev {Jar_File_Location}/com.altugcagri.smep-0.0.1-SNAPSHOT.jar
```

{Jar_File_Location} is the directory where you keep your jar file. Please change it according to your settings

It runs the backend project in dev profile. It means that application will read `backend\src\main\resources\application-db-dev.properties` file for connecting to the your development database (most probably local database).

Production environment:

```
$ nohup java -jar -Dspring.profiles.active=prod {Jar_File_Location}/com.altugcagri.smep-0.0.1-SNAPSHOT.jar &
```

{Jar_File_Location} is the directory where you keep your jar file. Please change it according to your settings

It runs the backend project in prod profile. It means that application will read `backend\src\main\resources\application-db-prod.properties` file for connecting to production database.

Application writes the log to `nohup.out` file which is placed where the jar is located.

Build and Run with Docker

Note That: Docker files is constructed for production mode. You cannot run docker files in your local environment. You have to set database environment variables explained above before building image of backend and running containers.

Build image

```
$ docker build -f Dockerfile --tag={ContainerName} .
```

Run Container

```
$ docker run -p 8080:8080 {ContainerName}
```

Running Instances

You can see all the running instances on AWS below;

The screenshot shows the AWS EC2 Instances page. A single instance is listed: 'bespoke-instance' (ID: i-0cfb15a36e63be57). It is an 't2.micro' instance type, located in 'eu-central-1b' availability zone, and is currently 'running'. Its public DNS is ec2-3-124-181-240.eu-central-1.compute.amazonaws.com, and its private IP is 172.31.47.14. The instance was launched on November 2, 2019, at 9:50:07 PM UTC+1 (1173 hours).

Figure 14 - AWS EC2 for Backend

Application and Annotation instance on EC2 (in the same machine)

The screenshot shows the AWS S3 'bespoke-frontend' bucket. It contains several files and folders: assets, dummy, static, asset-manifest.json, fav.ico, index.html, manifest.json, precache-manifest.067798ed8ddff178ffba5dc015cc3a53.js, precache-manifest.15abe7c6e9aab19571579214a5d24ce0.js, precache-manifest.3f3d6f244b13f586eea5243a756b7beb.js, and precache-manifest.6f8ce73f596ef95a44f2ca96d8f5a56.js. The files were last modified on December 16, 2019, at 5:16:56 PM GMT+0100, and their sizes range from 1.1 KB to 306.0 B. The storage class is Standard.

Figure 15 - AWS S3 for Frontend

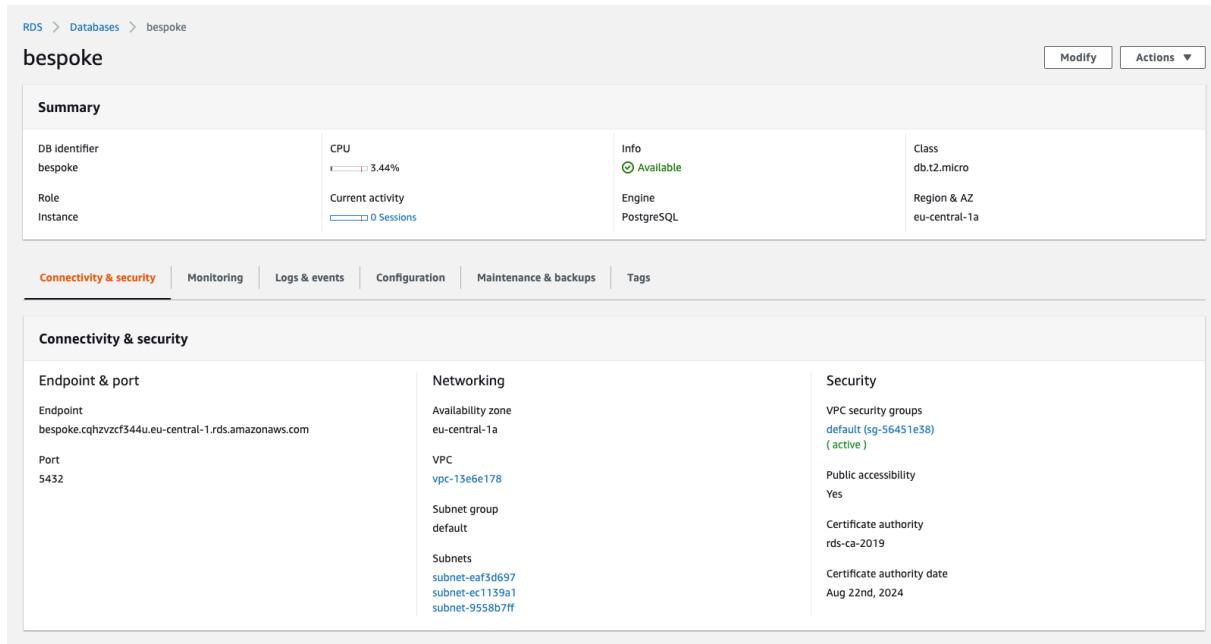


Figure 16 - AWS RDS for Database

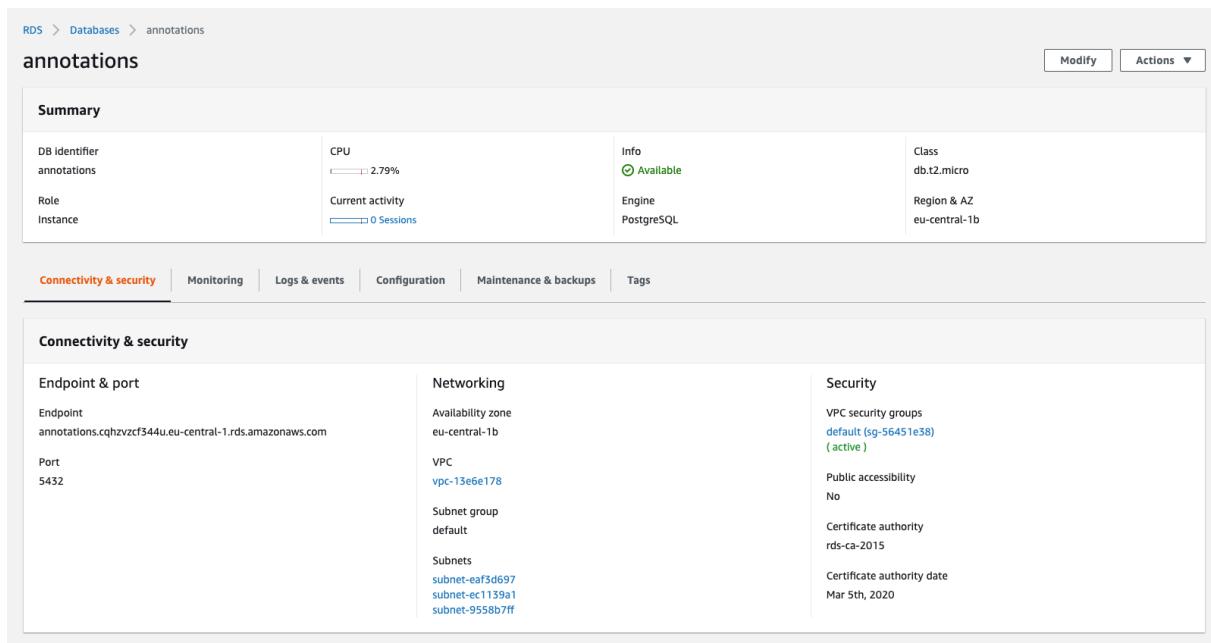


Figure 17 - Annotation Database on AWS

Continuous Integration & Continuous Deployment

CircleCI is used for the continuous integration and continuous delivery. There are three pipeline integrations on the project. I created one pipeline for each branch on github. For the sonarcloud integration I used dev branch pipeline. For backend implementation I used backend-master branch pipeline. Finally, I used the web-master branch for frontend pipeline.

Backend Pipeline

For the back-end pipeline there are three jobs on the workflow. In the first Job it runs the tests. If the test job gets results as success, second job starts. In the second job it runs build commands. It creates a jar file and puts it in a docker image. After that, it pushes the image to AWS repositories and starts a new container from pushed image on the AWS environment.

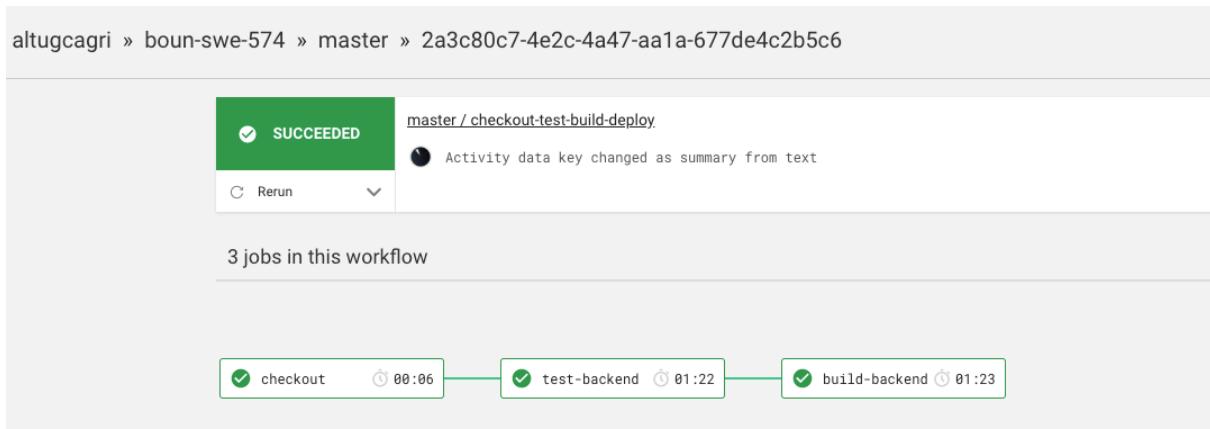


Figure 18 - Circle CI

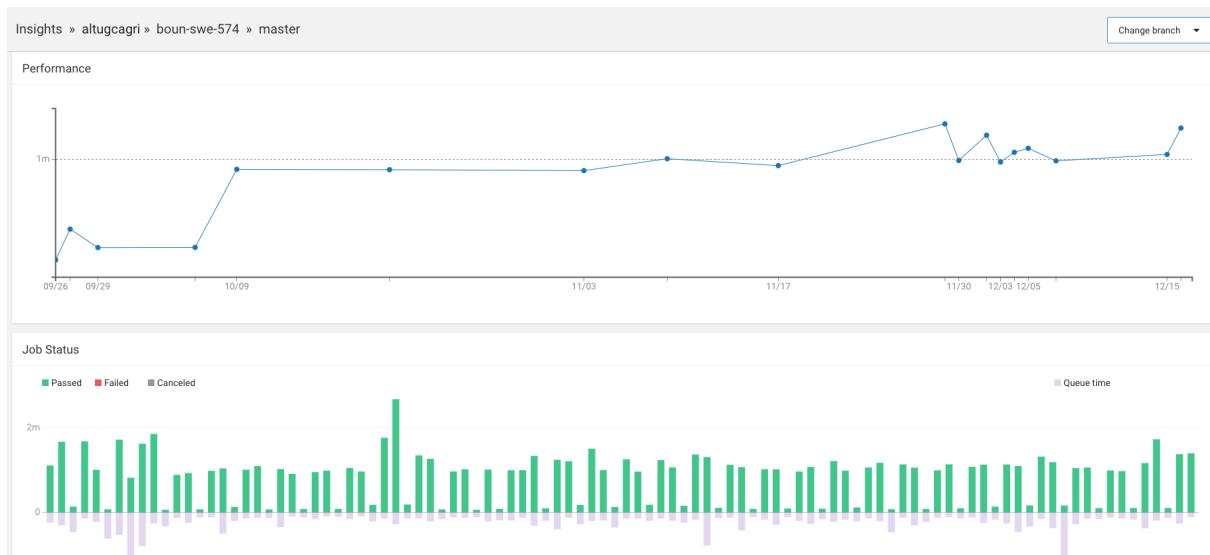


Figure 19 - Full Pipeline Report

Static Code Analysis

Static code analysis results are listed below. According to results project is passing the quality gate with total degree A.

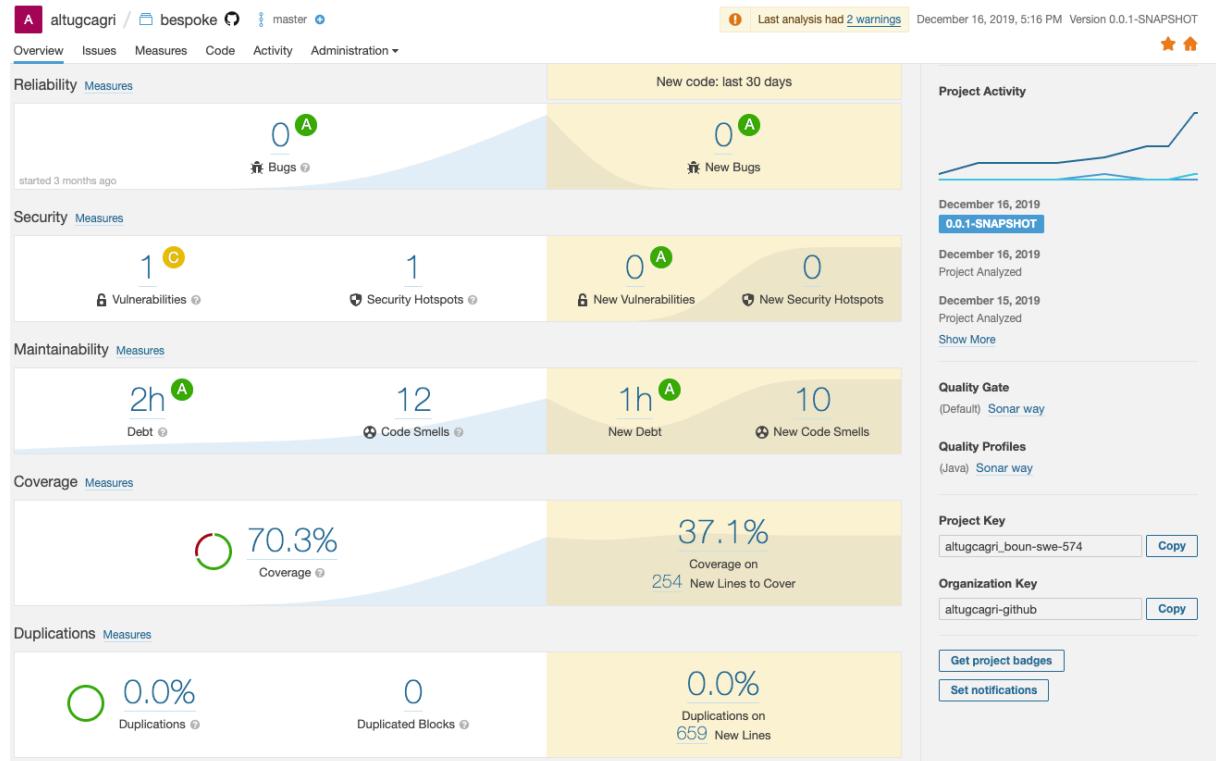


Figure 20 - Sonar Cloud

Reliability

According to results, there is no bug exists on the code. This makes reliability A as degree.

Security

According to results, there is one security vulnerability and 1 Security hot spot, which is not real vulnerability. They should be just check whether there is a potential gap or not. In the project we checked and did not see any potential gap on those hot spots. Security degree is A.

Maintainability

Code Smells issues are zero. Also, there is no depth issue on the code, which make maintainability degree as A.

Code Coverage

Unit tests are covering 70.3% of total code. There is some missing branch converge. For further improvements those branches can be covered. Despite the missing coverage, code coverage degree is A.

Duplication

There is no line or block duplication. This make 0 code duplication which is A.

User Manual

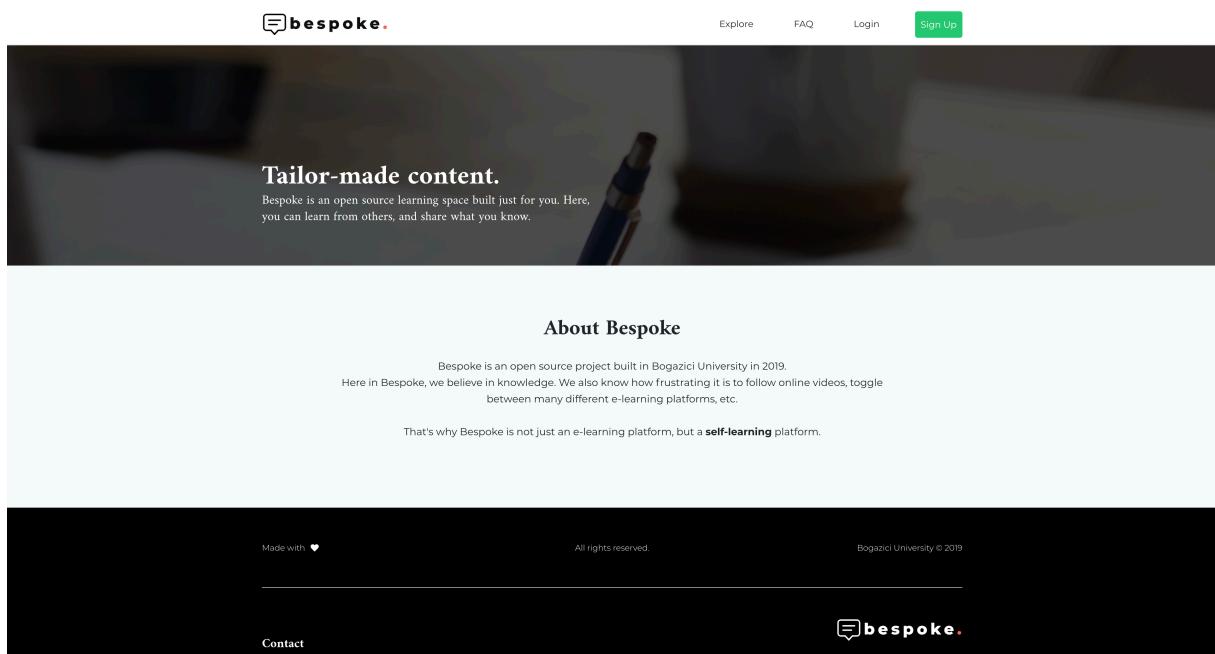
On this section, we'll briefly describe the using the Bespoke project. We'll focus on:

- Register / Login
- Creating New Topic with WikiData
- Creating Learning Path Content
- Creating Questions
- Enroll to Topic
- Following User
- Making Annotations

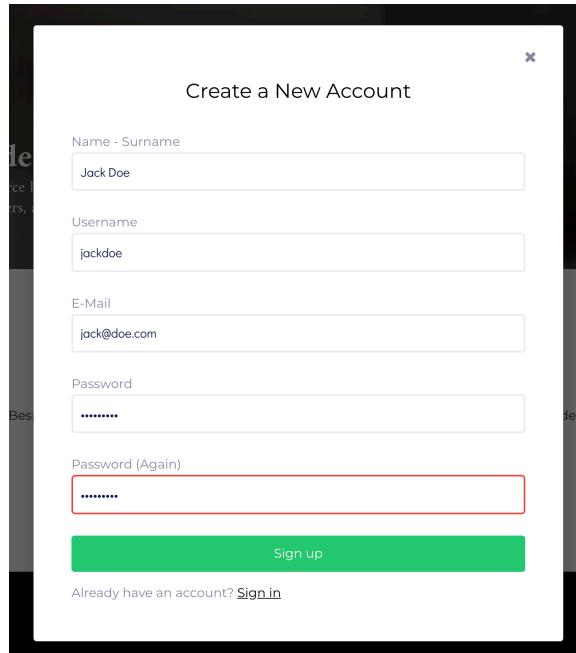
Register / Login

In order to use the Bespoke, sign-in is required. Follow the steps below described in the screenshots to create & login to your account.

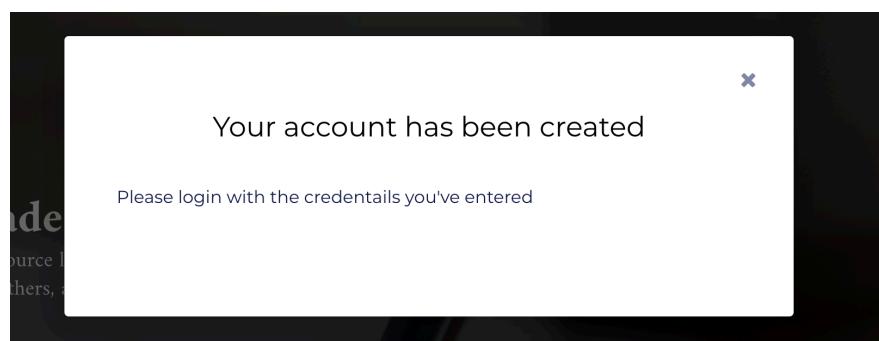
Step-1: On the welcome screen, click on the green “Sign Up” button at the top-right corner of the page.



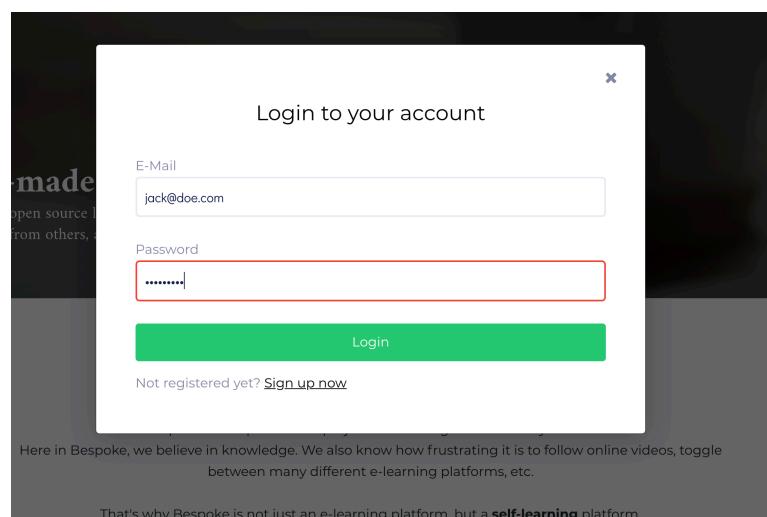
Step-2: Fill the form in the opened dialog with your credentials.



Step-3: Now you are ready to log in to your account with the credentials you've entered.

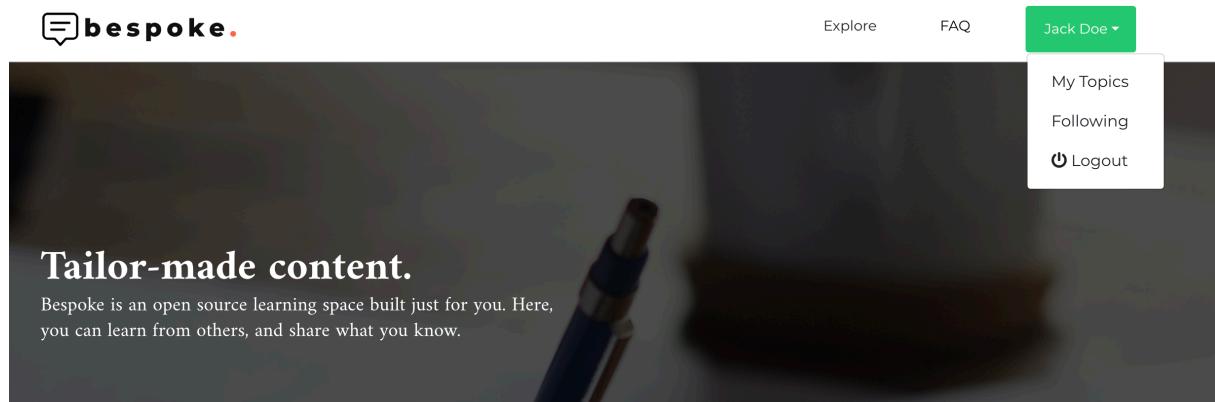


Step-4: On the header, click on “Login” link and provide your registered email and password to the form on the dialog.

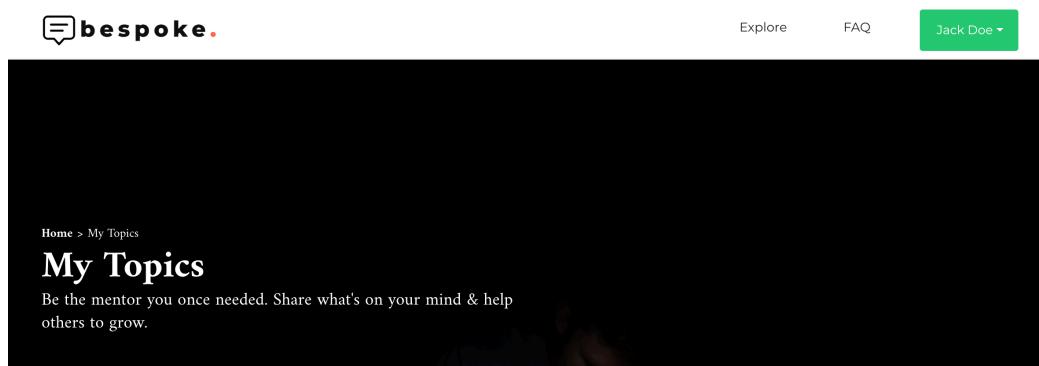


Creating New Topic with WikiData

Step-1: On the header, click on the green button with your name. And on the opened dropdown, click on “My Topics”



Step-2: On the page opened, click on “Create a Topic”



Step-3: Fill the topic form and search for wiki for each keyword you want to tag.

Title of your Topic

Hiking in Norway

Main Image Url

<https://images.unsplash.com/photo-1506701160839-34cfdecaf53c?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&a>

Description

Hiking is an awesome activity, and Norway is an excellent place for such activity. Bit cold maybe, but whenever you get tired, you can feast your eyes with the spectacular Scenery of Norway!.

WikiData

hiking

<input type="checkbox"/>	walking as a hobby, sport, or leisure activity	Visit
<input type="checkbox"/>	sailing action	Visit
<input type="checkbox"/>	path for hiking in a natural environment	Visit
<input type="checkbox"/>	scientific article published on 3 April 2008	Visit
<input type="checkbox"/>	sturdy lace-up ankle boot for outdoor wear	Visit
<input type="checkbox"/>	scientific article published in May 2000	Visit
<input type="checkbox"/>	important component of Israeli culture and tourism	Visit
<input type="checkbox"/>	equipment taken on outdoor walking trips	Visit
<input type="checkbox"/>	scientific article	Visit
<input type="checkbox"/>	scientific article	Visit

Create Topic

Step-4: Check all the wiki keywords you would like to include and search for new terms until you are done. Once you are done, click “Create Topic” and save the topic you’ve just created.

Your Wikis:

hiking - walking as a hobby, sport, or leisure activity [Remove](#)

fjord - long, narrow inlet with steep sides or cliffs, created in a valley carved by glacial activity [Remove](#)

Norway - constitutional monarchy in Northern Europe [Remove](#)

WikiData

norway

<input checked="" type="checkbox"/>	constitutional monarchy in Northern Europe	Visit
<input type="checkbox"/>	town in Maine, USA	Visit
<input type="checkbox"/>	city in Michigan, USA	Visit
<input type="checkbox"/>	city in Iowa, United States	Visit

Step-5: Now the topic you've created will be visible under your topics.

The screenshot shows a dark-themed interface for managing topics. At the top, there's a breadcrumb navigation: "Home > My Topics". Below it is a large title "My Topics" in bold. A subtitle reads: "Be the mentor you once needed. Share what's on your mind & help others to grow." In the center, there's a green button labeled "+ Create a Topic". Below this is a single topic card for "Hiking in Norway". The card features a small thumbnail image of a fjord at sunset. The title "Hiking in Norway" is in bold. A brief description follows: "Hiking is an awesome activity, and Norway is an excellente place for such". Below the description are three black buttons with white text: "# hiking", "# Norway", and "# fjord". At the bottom of the card is an orange "Details" button.

Creating New Content

Step-1: Click on “Details” button on the topic-card which you want to add content to. Click on the green plus button next to the “Learning Path”

This screenshot shows the detailed view of the "Hiking in Norway" topic. At the top, there's a dark header with the topic name and a tip: "Tip: a topic is just a start. It is in your own hands to master a topic. If you have another perspective, share it!". Below the header is a large image of a mountainous landscape. The main content area starts with a section titled "About Hiking in Norway". It contains a short description: "Hiking is an awesome activity, and Norway is an excellente place for such activity. Bit cold maybe, but whenever you get tired, you can feast your eyes with the spectacular Scenery of Norway!". Underneath this are two buttons: "Edit This Topic" and "Publish This Topic". To the right of the main content is a sidebar titled "Wiki" with three hashtags: "# fjord", "# hiking", and "# Norway". At the bottom of the page is a decorative wavy line graphic.

Learning Path +

Step-2: Provide the information you want to share on the form and click on “Save”.

Title of this material

Choosing the Right Backpack

Body of the material

B I S U { } [] " " % % [] Normal ↻ ↺ ↻ ↺

Once you are there in the wild, what you'll need a solid and reliable backpack. Make sure that your backpack is:

- Waterproof
- Modular
- Has a room for external mattress



Save

Step-3: Now you'll see this content under the topic you have created.

Learning Path +

1 - Choosing the Right B... ➔

Choosing the Right Backpack + Question 🗑️ 📁

Once you are there in the wild, what you'll need a solid and reliable backpack. Make sure that your backpack is:

Waterproof
Modular
Has a room for external mattress



Creating Questions

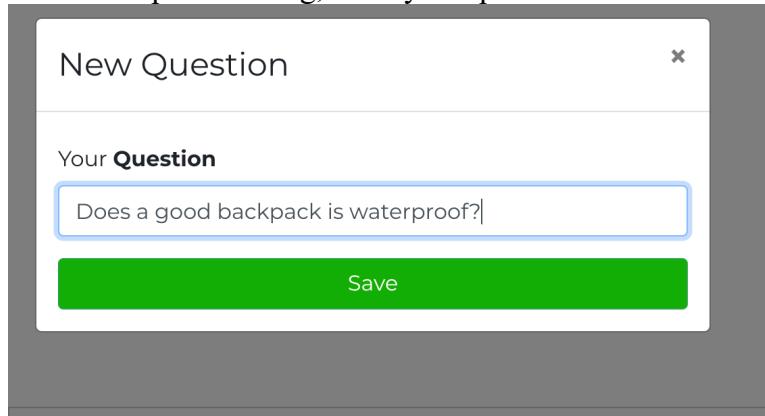
Step-1: You can add questions for every content. On the Topic detail page, find the content that you want to add question to, and click on the green “+ Question” button.

Choosing the Right Backpack + Question trash edit

Once you are there in the wild, what you'll need a solid and reliable backpack. Make sure that your backpack is:

- Waterproof
- Modular
- Has a room for external mattress

Step-2: On the form in the opened dialog, enter your question and click on “Save”

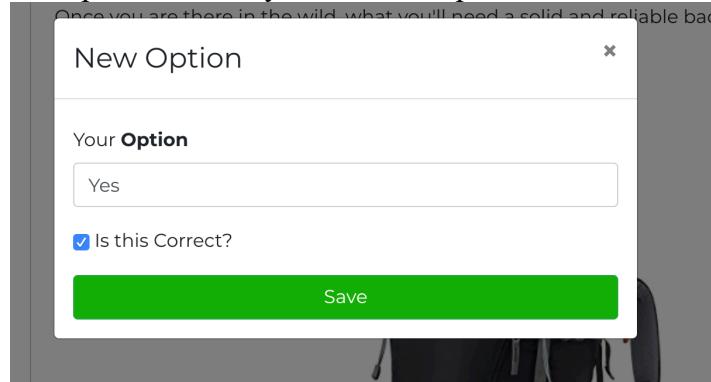


Step-3: Now your question will be visible under the content. Click on “+ Option” to add options to the question.



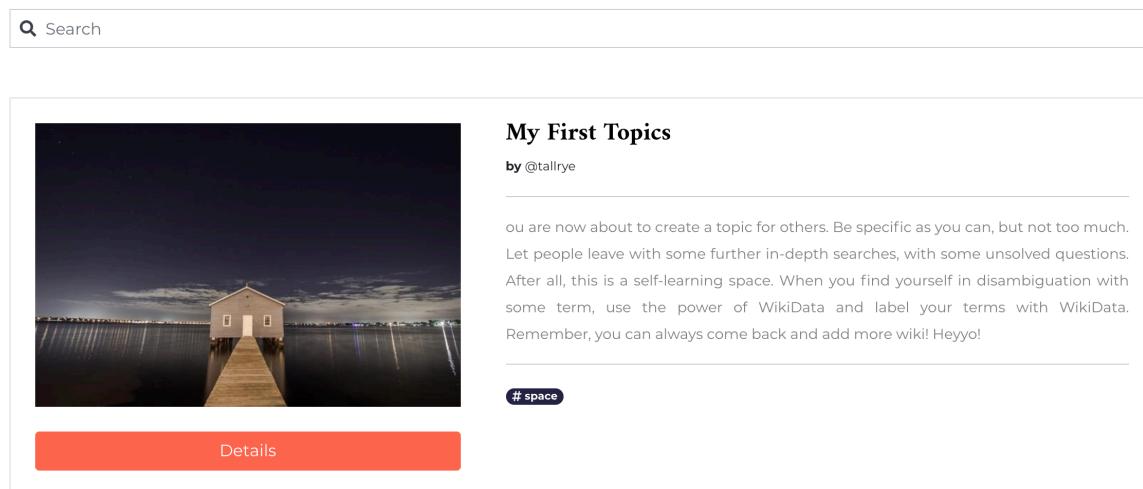
Q1: Does a good backpack is waterproof? + Option trash

Step-4: Enter your option, and mark as “correct” if the option is true. You can add as many options as you want. But please add only one correct option.



Enroll to Topic

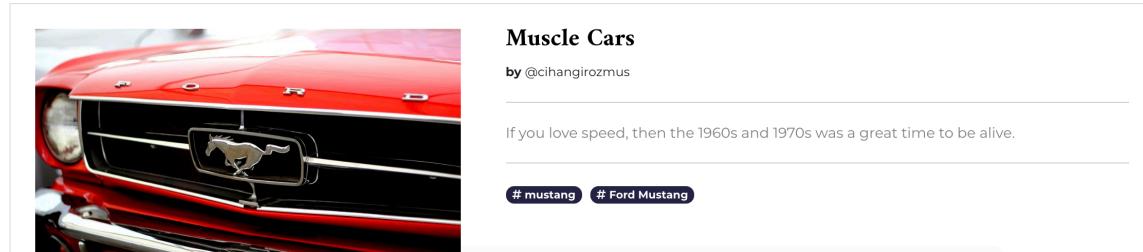
Step-1: Visit “Explore” page and find a topic that you might be interested in.



My First Topics
by @tallrye

You are now about to create a topic for others. Be specific as you can, but not too much. Let people leave with some further in-depth searches, with some unsolved questions. After all, this is a self-learning space. When you find yourself in disambiguation with some term, use the power of WikiData and label your terms with WikiData. Remember, you can always come back and add more wiki! Heyyo!

space



Muscle Cars
by @cihangirozmus

If you love speed, then the 1960s and 1970s was a great time to be alive.

mustang # Ford Mustang

Step-2: Click on the topic you are interested and see the details. On this page, you can register to this topic by simply clicking on “Enroll to This Topic” button.

The screenshot shows a topic page titled "Muscle Cars". At the top left is a navigation bar: "Home > Explore > Muscle Cars". Below the title is a tip: "Tip: a topic is just a start. It is in your own hands to master a topic. If you have another perspective, share it!" To the right of the tip is a "Wiki" section with hashtags: "#mustang" and "#Ford Mustang". A green button at the bottom left says "Enroll To This Topic".



Learning Path

- 1 - History of Mustang... >

Step-3: Once you are enrolled to a topic, you can explore the content with the questions until you are finished with the topic.

The screenshot shows a page titled "History of Mustang". At the top left is a navigation bar: "Home > Muscle Cars > History of Mustang". Below the title is a tip: "Tip: learning is something you do on your own. Do not settle with this material, go ahead and explore it further!".

History of Mustang

The legendary 1965 Mustang Shelby GT350 were serious high- performance machines. In fact, some buyers that very first year felt these cars were a little too hardcore and at the same time Shelby was on a rampage to cut costs. So, for 1966, Shelby replaced, deleted or made optional some of the car's signature high performance features like the adjustable Koni shocks, the fiberglass hood, free-flowing (and loud) side exhaust outlets and that fully locking Detroit Locker rear differential.

> Start Material Quiz

The screenshot shows a page titled "Content Quiz". At the top left is a navigation bar: "Home > Muscle Cars > Content Quiz". Below the title is a tip: "Tip: learning is something you do on your own. Do not settle with this material, go ahead and explore it further!".

Quiz: History of Mustang

Q1: What is production year of GT350?

- 1991
- 1960
- 1956
- 1965

[Answer]

[Back to Home]

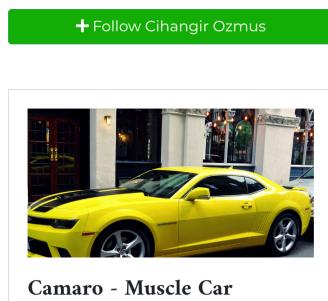
Following a User

Step-1: Click on a username on the topic you're interested in.

Muscle Cars

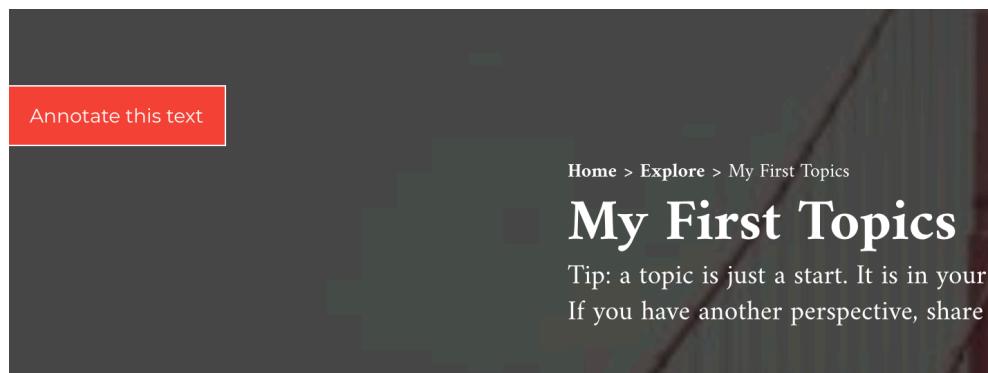
by @cihangirozmus

Step-2: Once you get to their profile, click on “Follow Username”



Making Annotations

Step-1: Select a piece of text on the screen that you want to annotate. Once you do this, an orange button will appear on the left-side of the screen. Click on this button to create your annotation.



About My First Topics

You are now about to create a topic for people leave with some further in-depth this is a self-learning space. When you

Step-2: Enter your comment along with your motivation and click on “Annotate”

You are annotating the following text:
"create"

Your comment
Annotations are awesome.

Your motivation
commenting

Annotate

When you find yourself in disambiguation with some term, use

A screenshot of a web-based annotation tool. At the top, it says "You are annotating the following text: 'create'". Below that is a "Your comment" field containing "Annotations are awesome.". Underneath is a "Your motivation" field containing "commenting", which is highlighted with a red border. At the bottom is a large green "Annotate" button. At the very bottom of the interface, there is a note: "When you find yourself in disambiguation with some term, use".

Project Requirements

Requirement study is completed through the project description.

Glossary

Visitor: A person who enters to site

User: Visitor who signed up and logged in

Glossary: List of Topics

Topic: Consists of topic title, topic description, topic content and label(s)

Topic Content: Contains topic image(s) and topic text

Topic Statistic: Consist of enrolled user number, accuracy of the quiz answers

Learning Path: A consecutive collection of topic content and related quiz

Quiz: A collection of questions

Question: Related with specific Content and created by users as a step of a Learning-Path for others to explore

User Profile: Consists of user email, created topic list, enrolled topic list, enrolled topic statistics

Activity: Action of a user defined by the W3C in <https://www.w3.org/TR/activitystreams-core/>

Activity Stream: Collection of activities defined in JSON-LD format

Annotation: A piece of comment in JSON-LD format created by a User on a page in the system related to an image or a text

Annotation-Server: A stand-alone, separate server that contains and serves all annotations

Label: Wiki tag related to content which is created by the owner of the content

Requirements

1. Non-Functional Requirements

1.1. Back-end shall use the following database system: PostgreSQL

1.2. Back-end shall be written in Java

1.3 Developed application should be deployed to web

1.4 System language shall be English

1.5 System shall have continuous integration / continuous development implementation through CircleCI

1.6 Static Code analysis shall be done through Sonar Cloud

1.7 As an acceptance criterion there should be 80 percent of code coverage on Back-end codes

1.8 System development shall be followed via Github

1.9. All Annotations should be stored in a stand-alone Annotation-Server

1.10. All Annotations should be retrieved from a stand-alone Annotation-Server

2. Functional Requirements

2.1.1 Visitor shall be able to see the landing, glossary, signup and login pages.

2.1.2 Visitor and user shall be able to search glossary.

2.1.3 Visitor shall be able to sign up as a user.

2.1.4 Email and Password shall be required for sign up.

2.1.5 Email shall be unique.

2.1.6 Email and password shall be required for log in.

3. Topic Requirements

2.2.1 User shall be able to create topic.

2.2.2 Topic shall have a title.

2.2.3 Topic shall have at least one tag.

2.2.4 Topic shall have a description.

2.2.5 Topic shall be seen by users.

2.2.6 Topics shall be listed in glossary.

2.2.7 Topic shall have more than one learning path.

2.2.8 Topic shall have zero or many quiz.

2.2.9 Topic statistic shall be seen by topic owner.

4. API Requirements

2.3.1 User shall be able to search api with keyword(s).

2.3.2 User shall be able to select zero or many result from api search.

2.3.3 User shall use selected api result(s) in his/her topic.

5. Quiz Requirements

2.4.1 Learning path shall have at least one quiz.

2.4.2 A quiz shall have at least one question.

2.4.3 Question shall be related with content.

2.4.4 Question shall have five options with only one correct answer.

6. User Requirement

2.5.1 User shall be able to update the owned topic.

2.5.2 User shall be able to update the learning path which belongs to owned topic.

2.5.3 User shall be able to update the quiz which belongs to owned topic.

2.5.4 User shall be able to see quiz when the quiz is started.

2.5.5 User shall be able to continue the learning path after answering the all questions correctly.

2.5.6 User shall be able to restart the quiz.

2.5.7 User shall see his/her own user profile.

2.5.8 User statistics shall be stored with enrolled topic(s) and quiz answers.

2.5.9 User statistics shall be displayed in user profile.

2.5.10 Users should be able to follow another Users

2.5.11 Users should be able to follow Labels

7. Activity Requirements

2.6.1 Users should be able to see the Activities of the Users which they are following

2.6.2 Activity Streams should be generated by the system.

8. Annotation Requirements

2.7.1 Users should be able to create an Annotation for an image on a page

2.7.2 Users should be able to create an Annotation for a selected text on a page

2.7.3 Users should be able to see all the Annotations created on a page in the platform

9. Recommendation System

2.8.1 Users should be able to see Recommendations based on the Labels and Users that they are following

Sequence Diagrams

Create Annotation

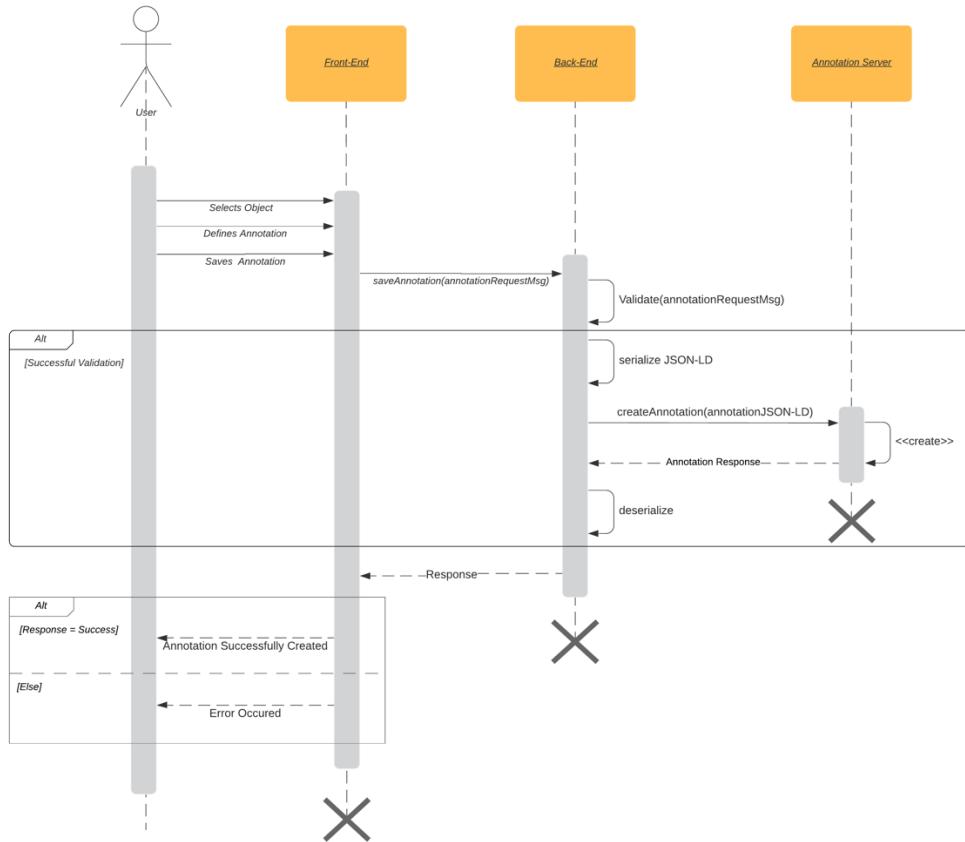


Figure 21 - Sequence Diagram for Create Annotation

Get Annotations

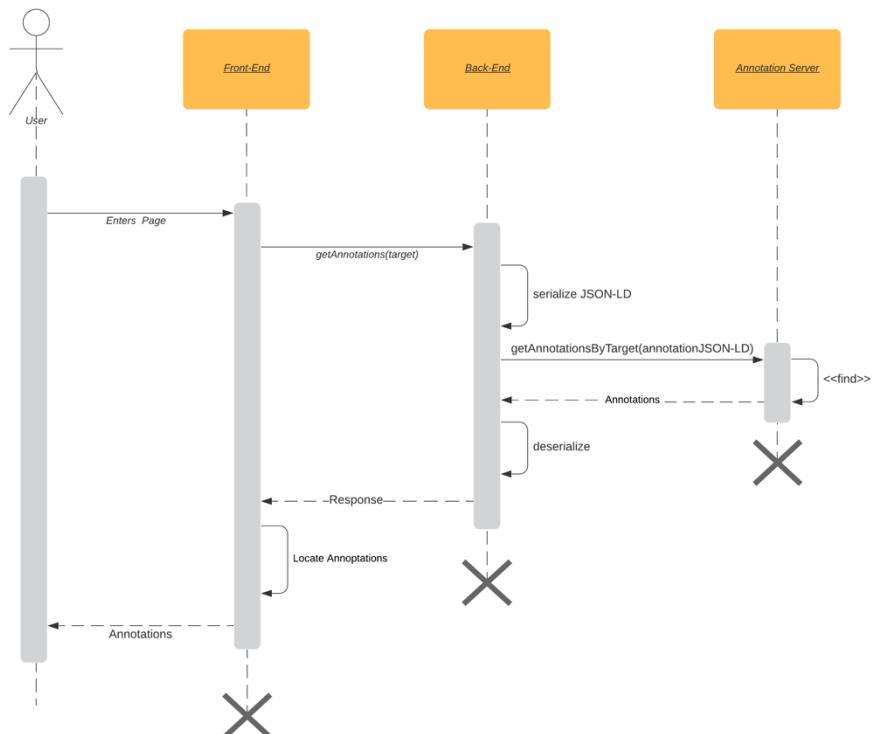


Figure 22 - Sequence Diagram for Get Annotations

Delete Annotation

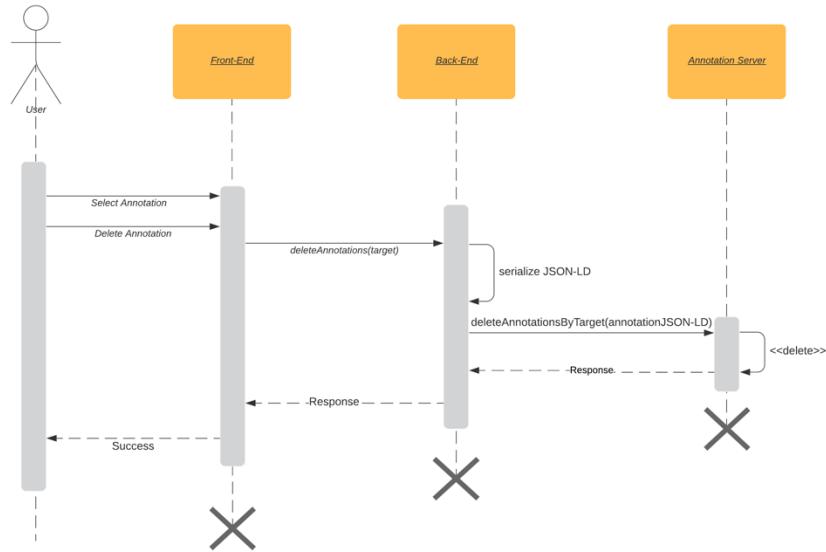


Figure 23 - Sequence Diagram for Delete Annotation

Save Activity

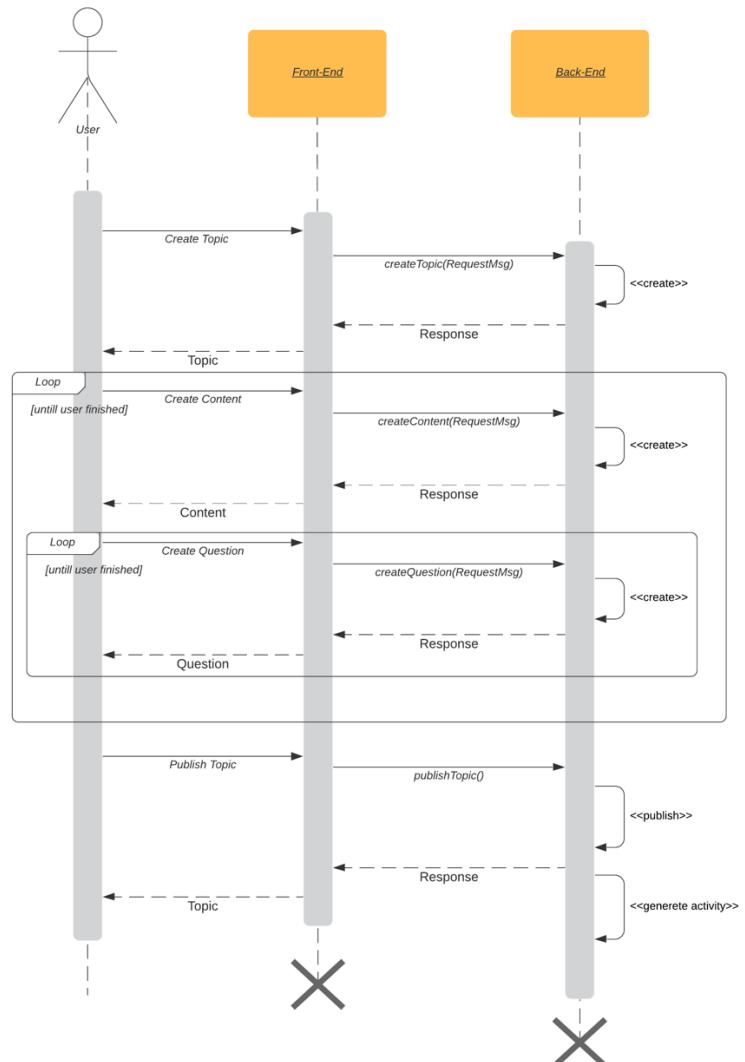


Figure 24 - Sequence Diagram for Save Activity

Get Notifications

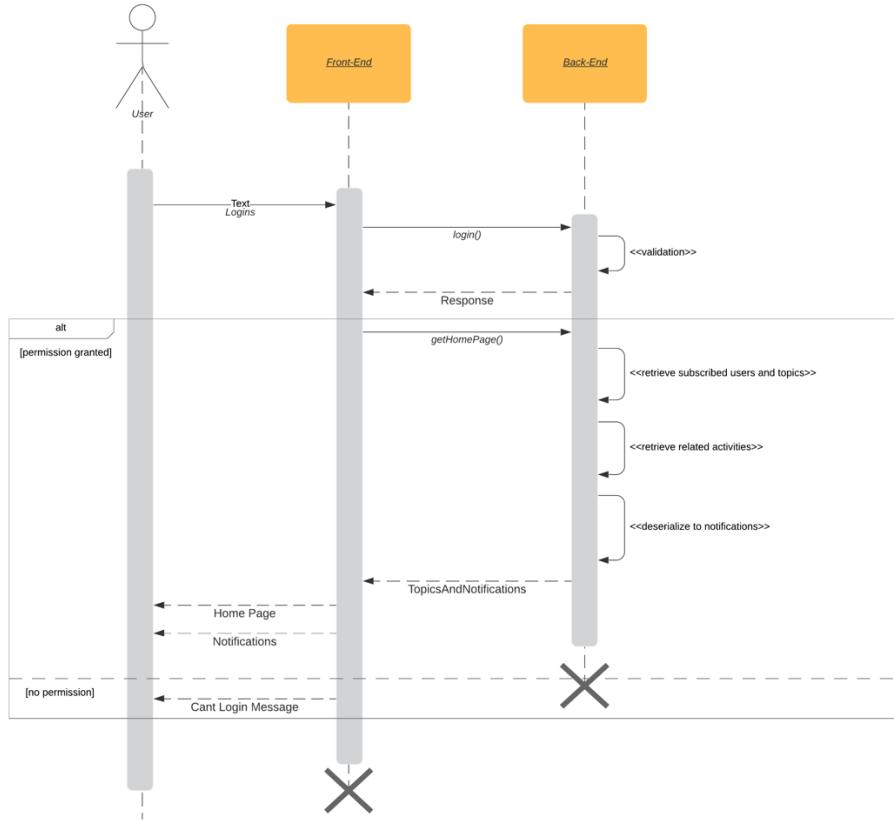


Figure 25 - Sequence Diagram for Get Notifications

Get Recommendations

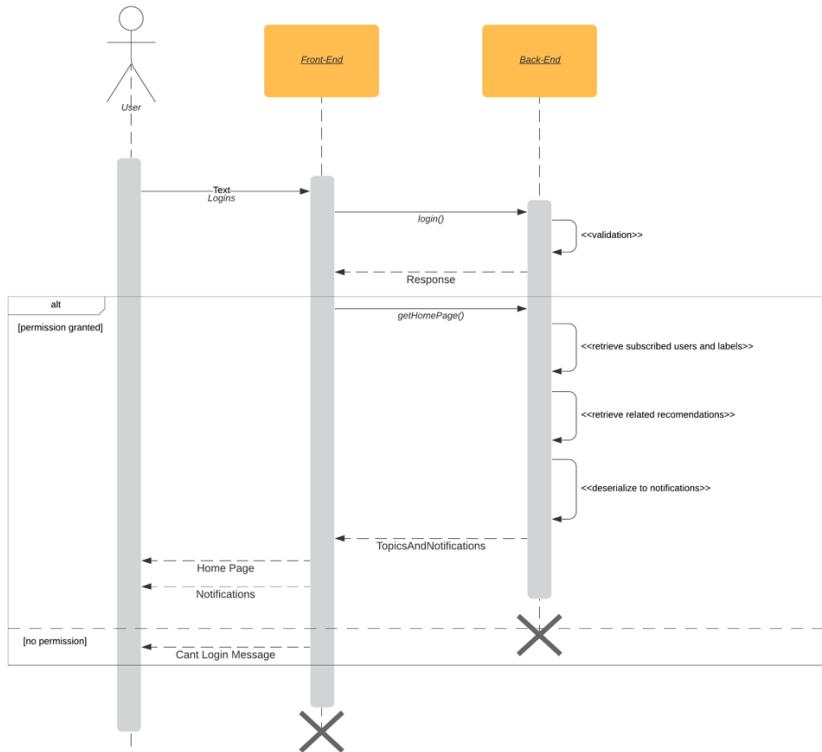


Figure 26 - Sequence Diagram for Get Recommendations

ER Diagrams

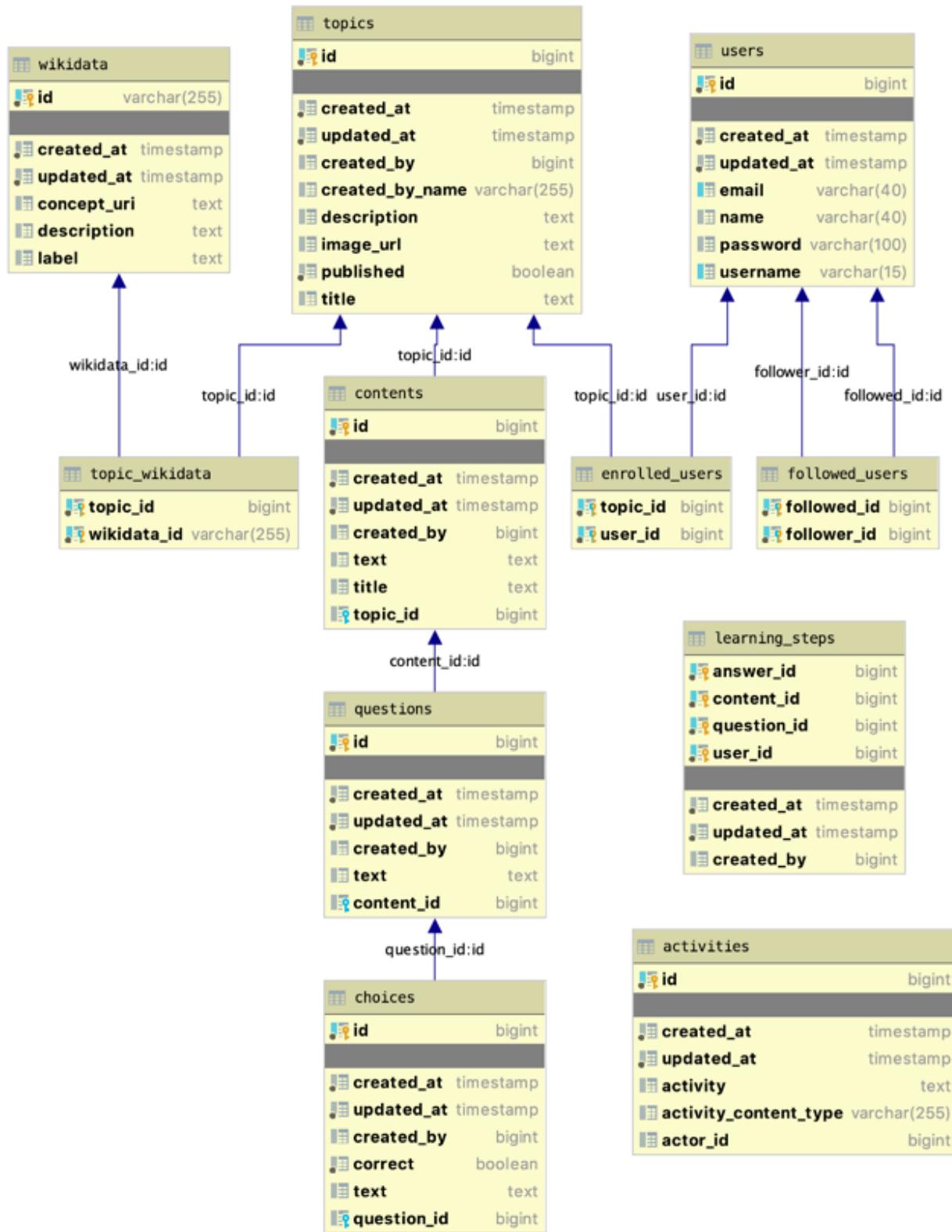


Figure 27 - Entity Diagram

Class Diagrams

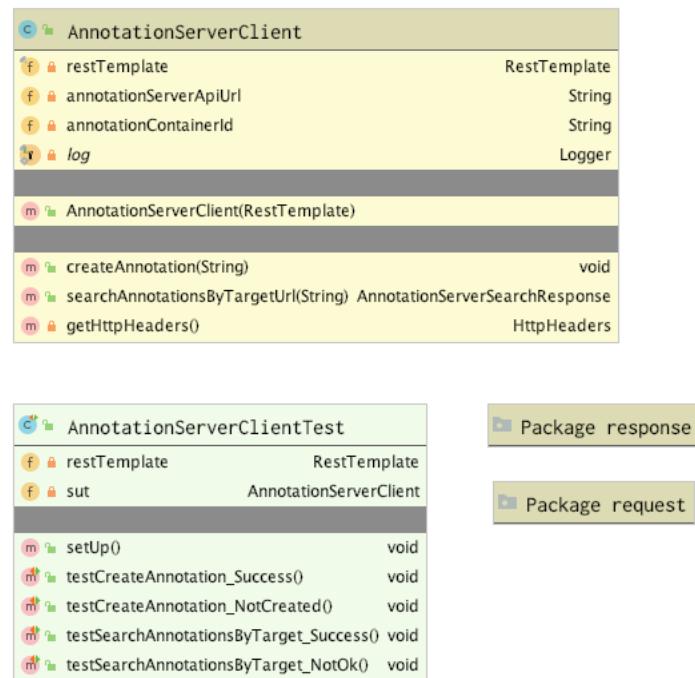


Figure 28 - client package

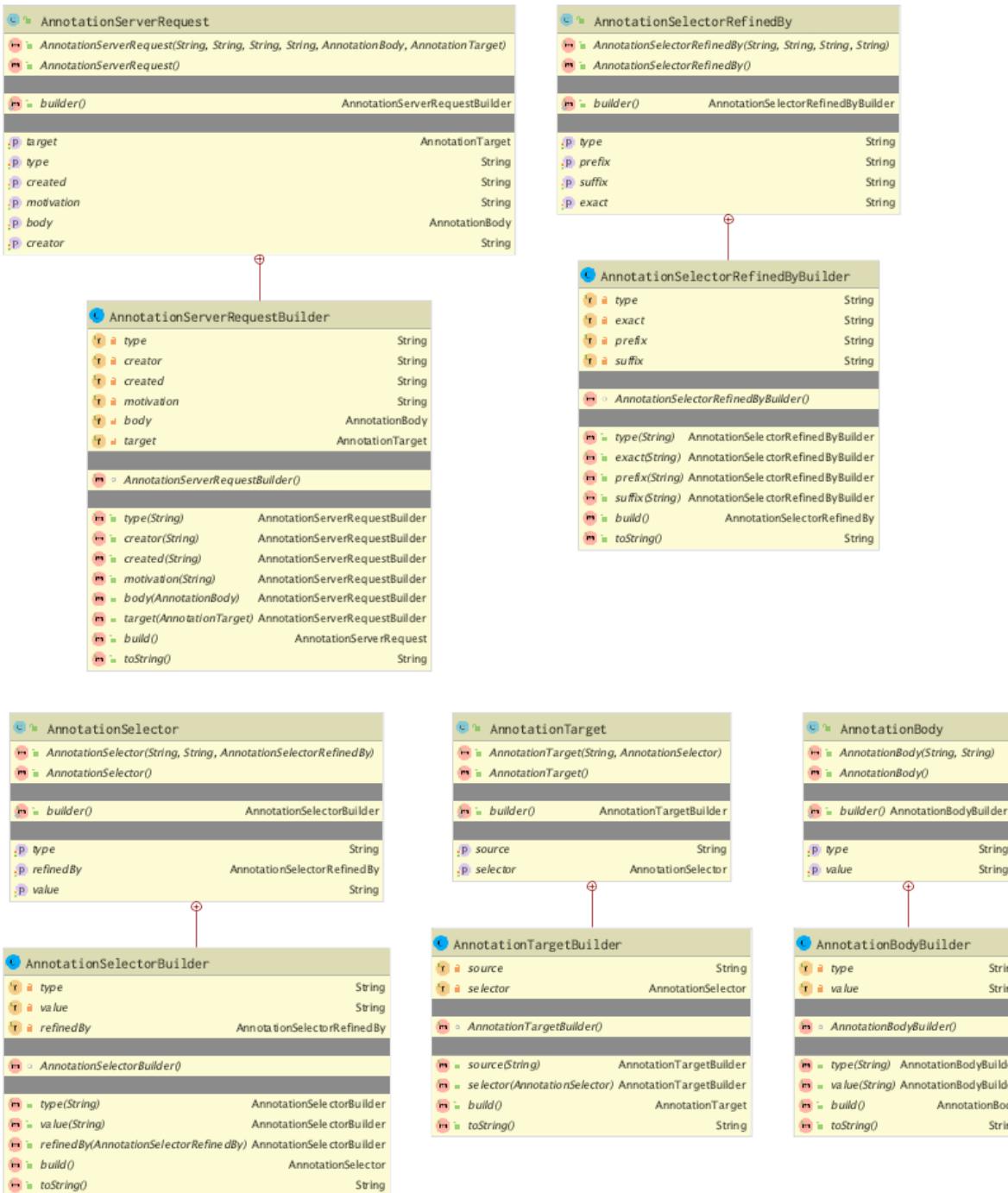


Figure 29 - client/request package

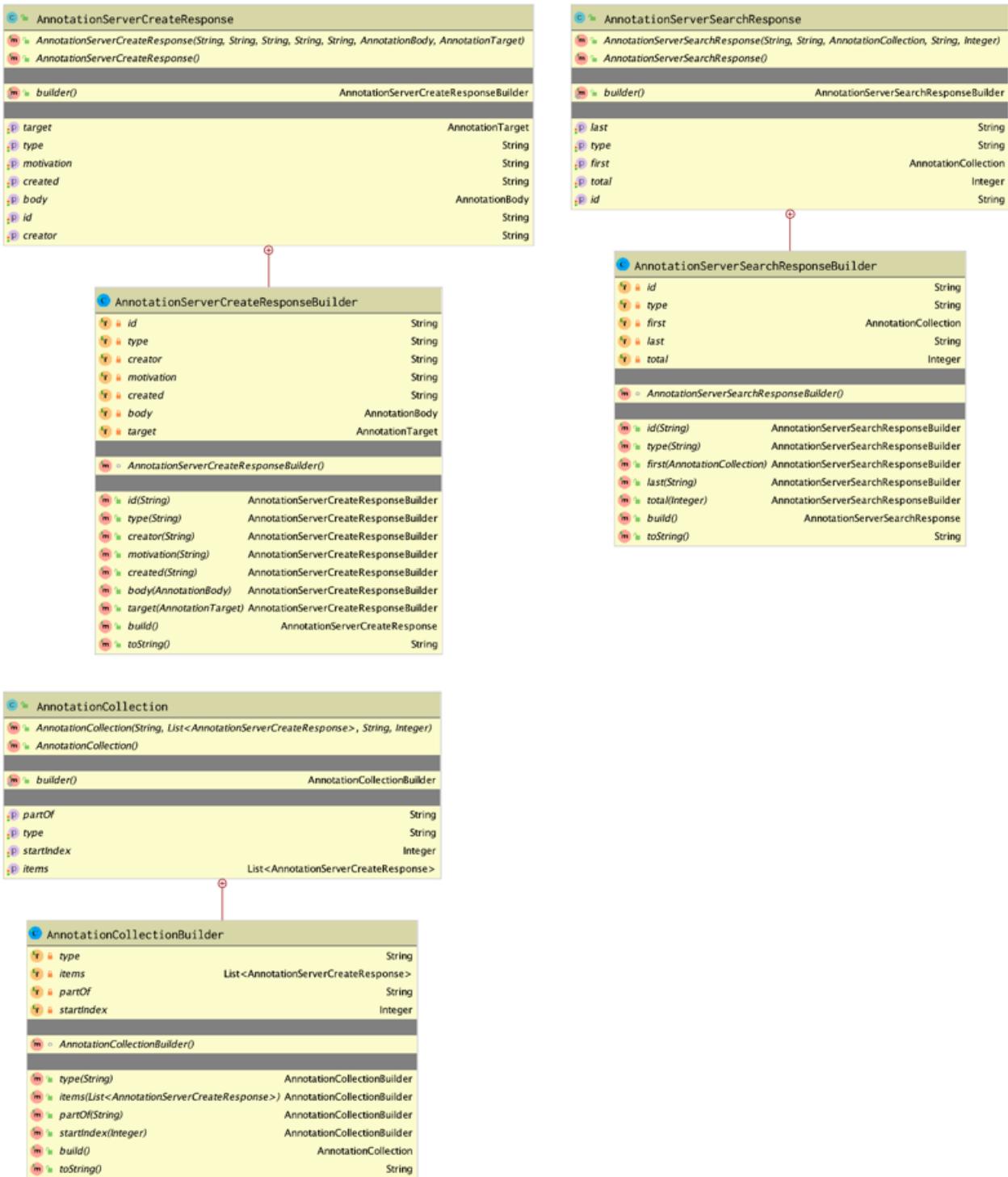


Figure 30 - client/response package

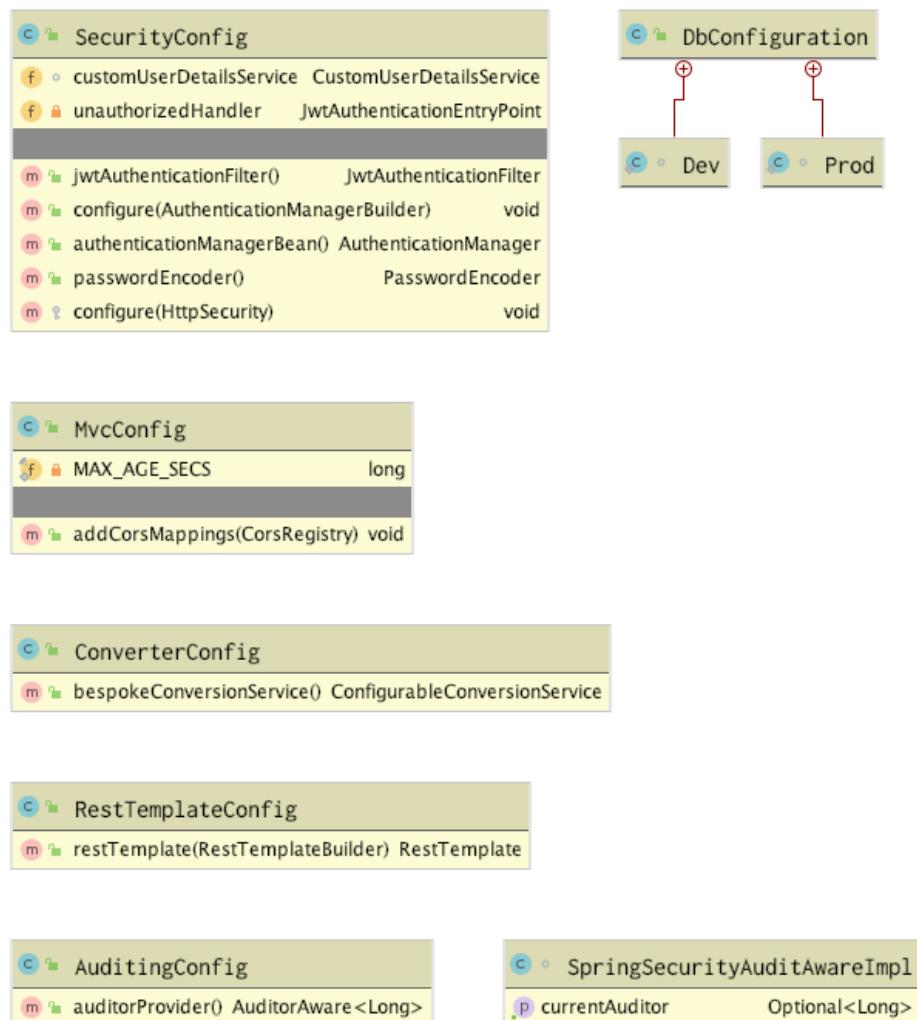


Figure 31 - config package

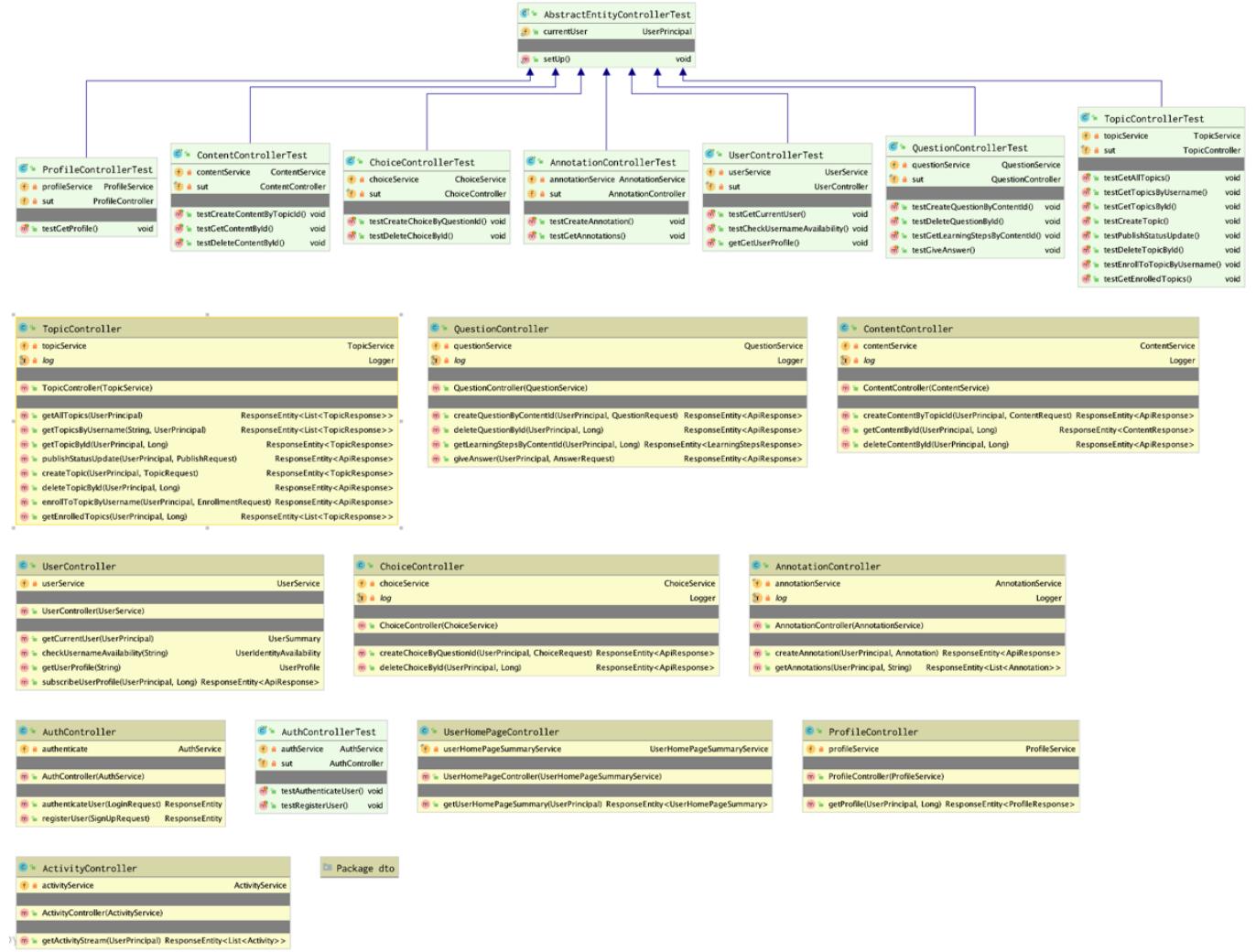


Figure 32 - controller package

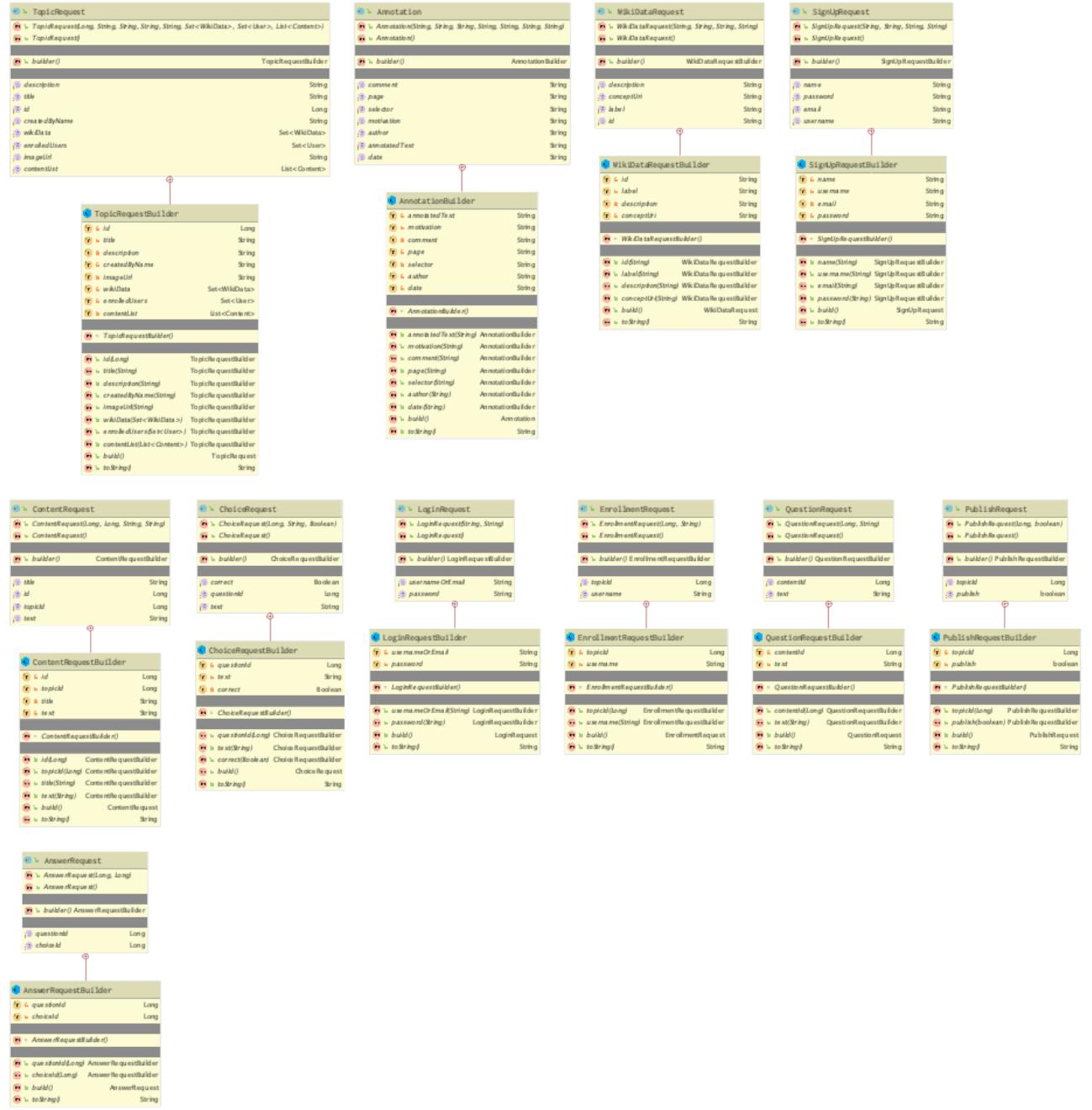


Figure 33 - controller/dto/request package



Figure 34 - controller/dto/response

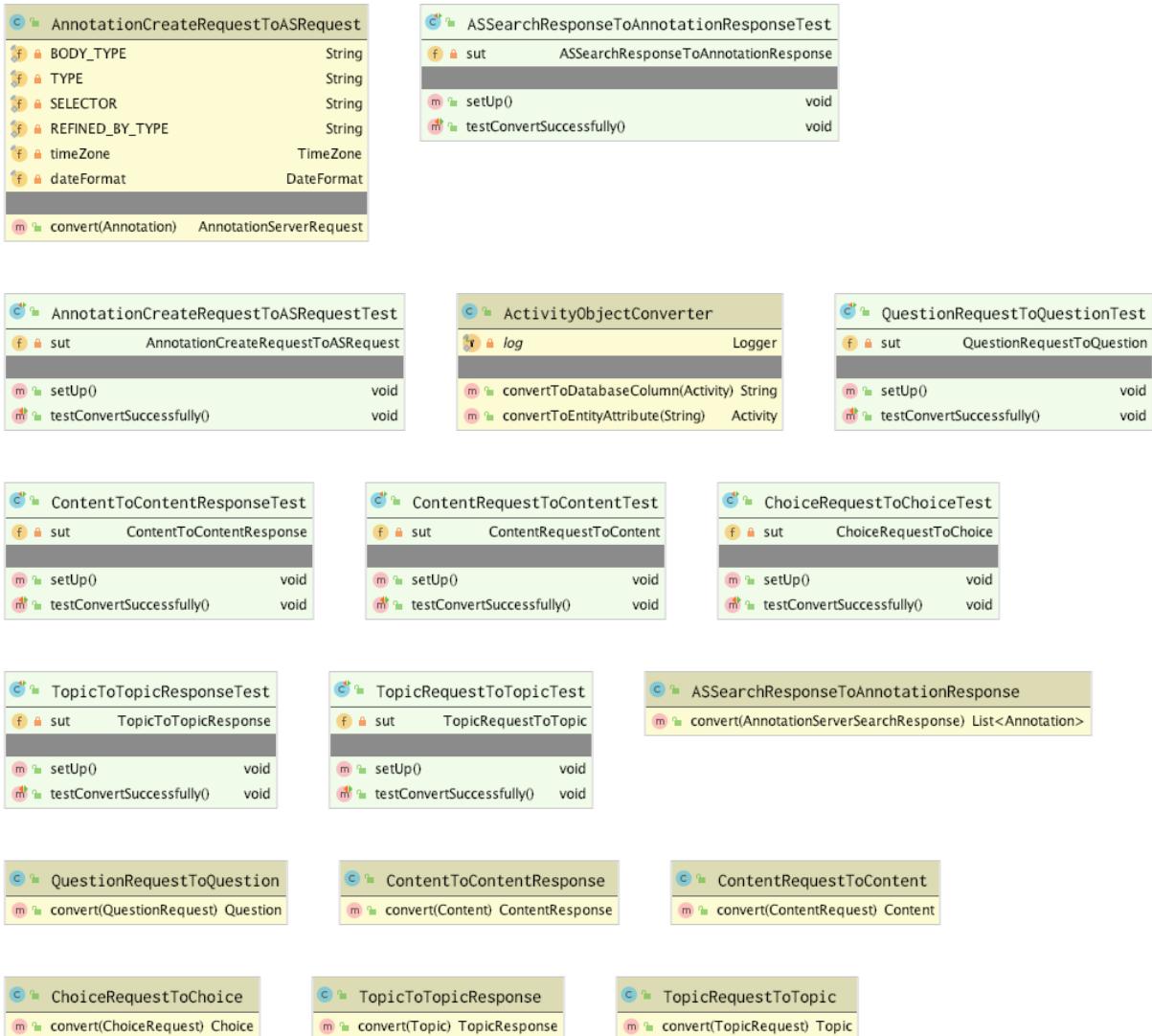


Figure 35 - converter package

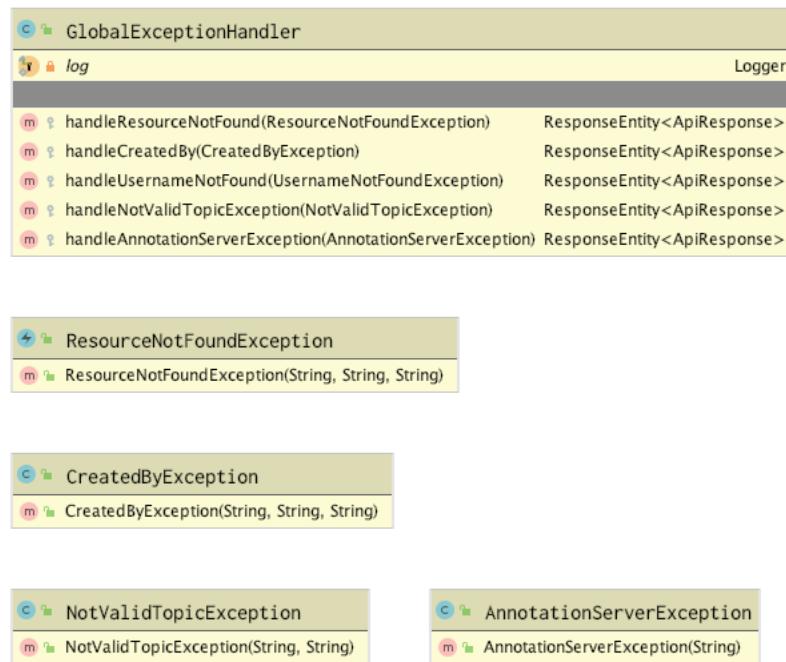


Figure 36 - exception package



Figure 37 - persistence package

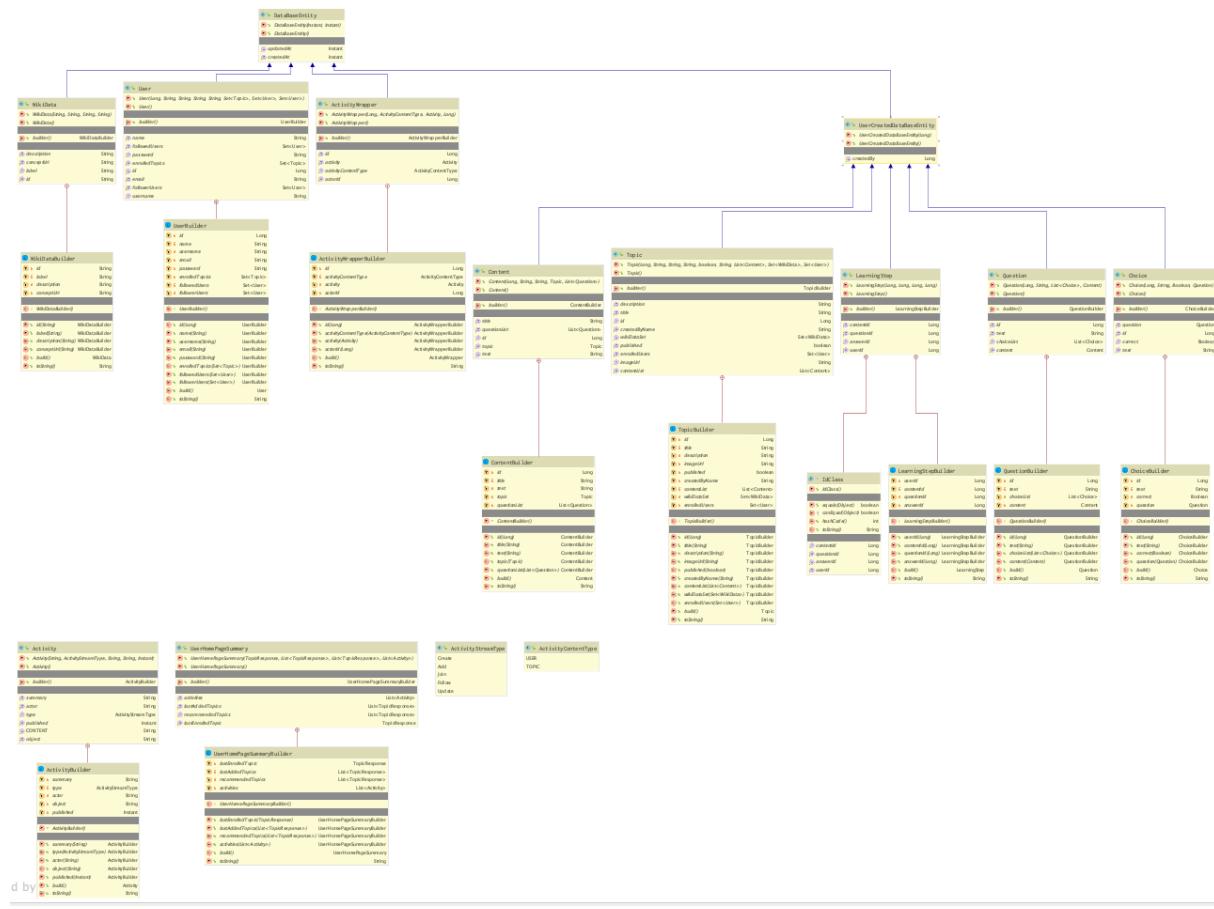


Figure 38 - persistence/model package

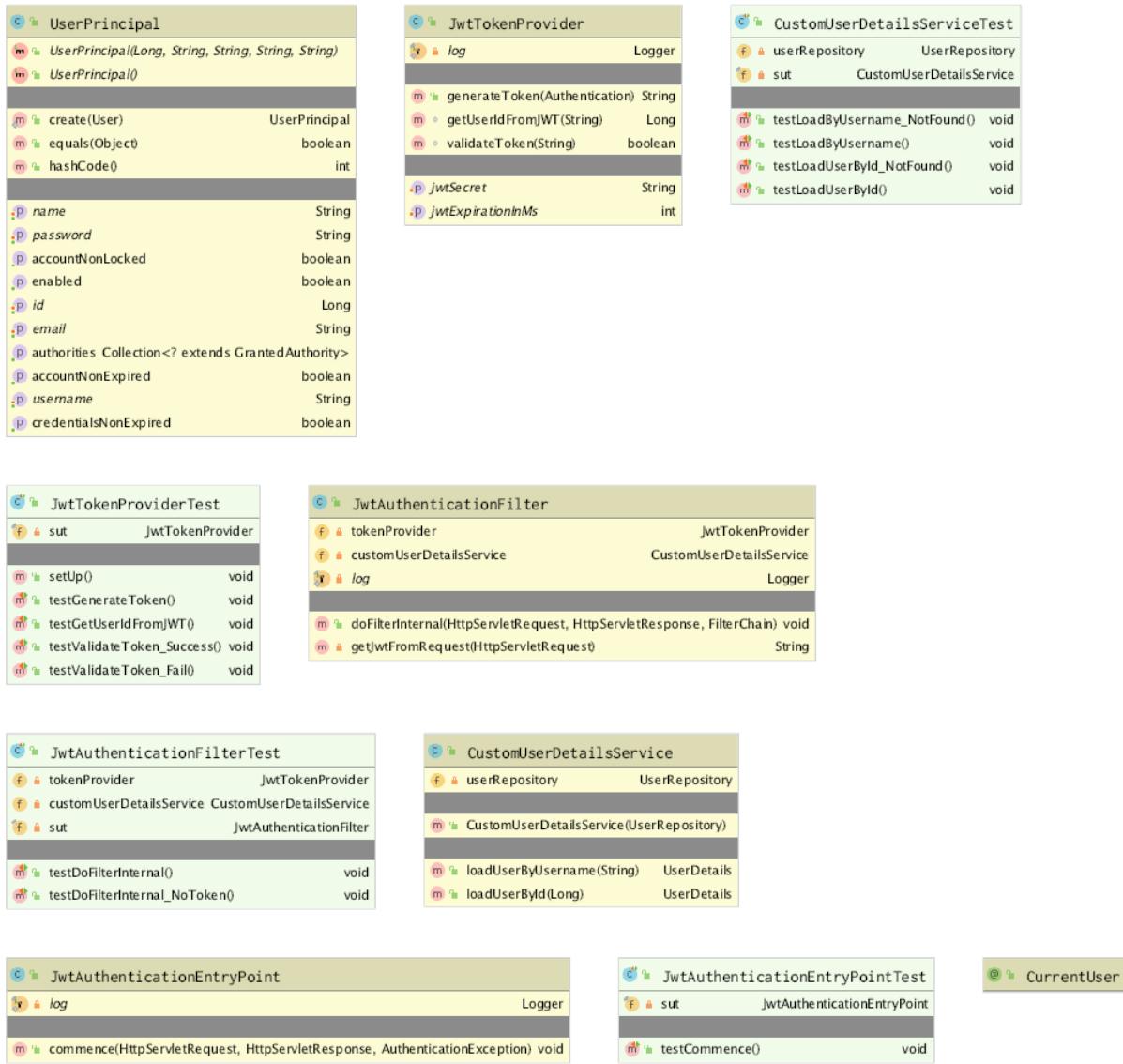


Figure 39 - security package

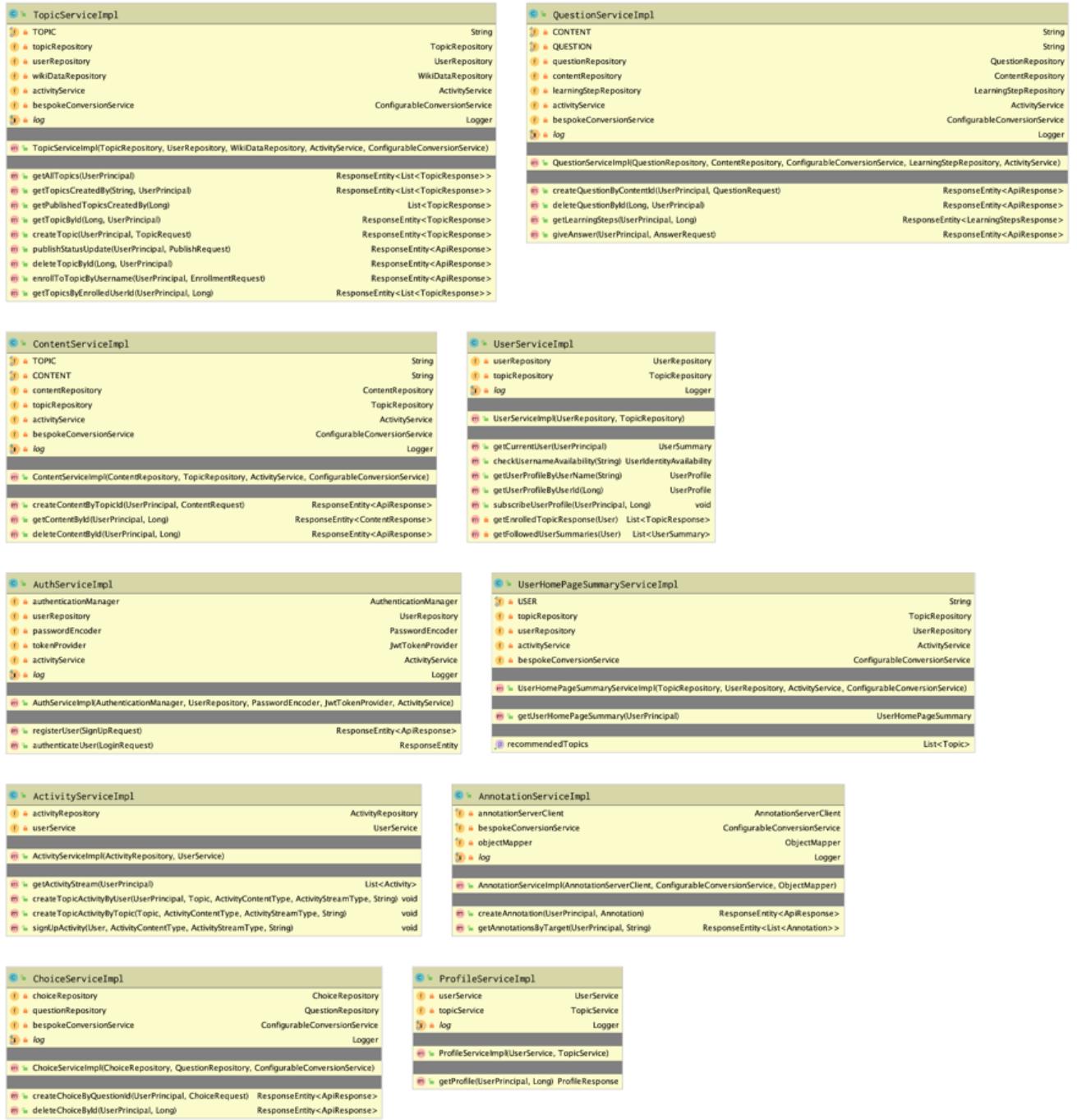


Figure 40 - service/implementation package

Folder Structure

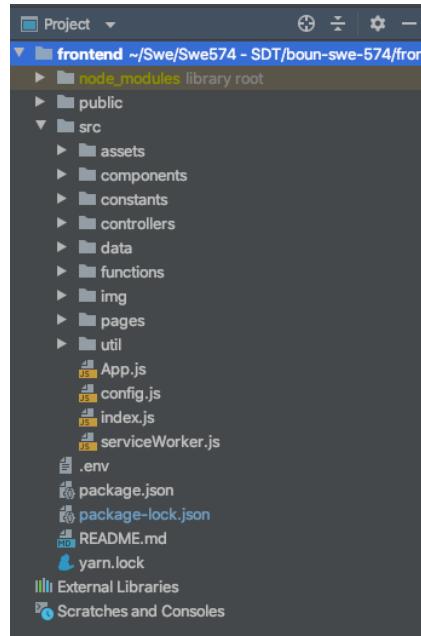


Figure 41 - Frontend Folder Structure

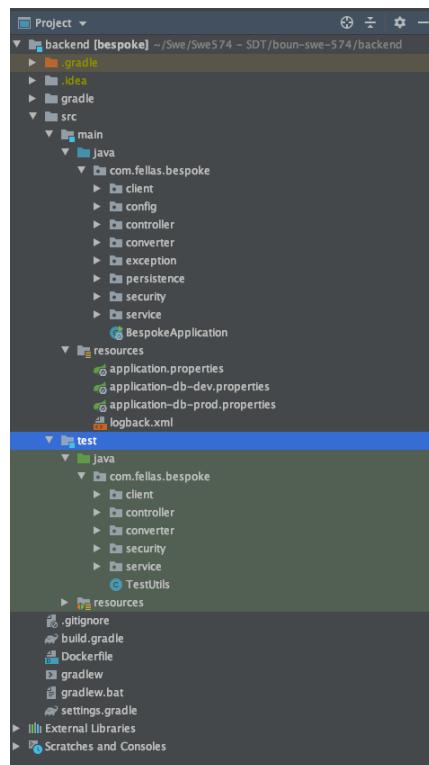


Figure 42 - Backend Folder Structure

Conclusion

Spring Rest API architecture is used for Bespoke Web Application project. Frontend and Backend are designed as separate projects. Database entities are designed with Hibernate ORM. Amazon EC2 service is used as a cloud server and Amazon RDS selected as a database server.

Activity Streams are hardcoded by our own solution.

Elucidate Server is used for annotations as a third-party application.

We are thankful for the project to give us a chance to examine a team experience.

Appendix

Research: Git and GitHub

Git is a free and open source distributed version control system. Can be downloaded from <https://git-scm.com/download/win>.

Version control is a system that records changes to a file or set of files over time so that we can recall specific versions later.

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features such as issue tracking, labels, milestones and project management. <https://github.com/>

Especially we do not need GitHub to use Git. On the other hand GitHub is useful for code sharing and collaboration. Bitbucket, gitlab can also be the alternatives for GitHub. There are also gui tools to visualize version control workflow such as SourceTree. <https://www.sourcetreeapp.com/>

Benefits of GitHub

- Sharing our repositories.
- Accessing other user's repositories.
- Storing our local repositories in GitHub cloud as remote repository.

Git Commands

These commands can be used in the terminal after installing Git. For details please refer to documentation <https://git-scm.com/doc>.

```
git --version  
git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com  
git config --list  
git init  
git status  
git add -A  
git add .  
git commit -m "message"  
git push  
git log  
git clone  
git branch  
git merge
```

Research: Rest

Rest stands for Representational state transfer. GET, PUT, POST, DELETE are used as verbs.

- Messages(payload): JSON/XML
- URI: Uniform resource identifier (a unique string identifying a resource)
- URL: Uniform resource locator (a URI with network identification - <http://www.example.com>)
- Idempotence: Certain operation applied multiple times without changing the result. GET, PUT and DELETE are idempotence. POST is the only operation which is not idempotence.
- Stateless: Service does not maintain any client state.
- HATEOAS: Hypermedia As The Engine of Application. A rest client should then be able to use server-provided links dynamically to discover all the available actions and resources it needs. As access proceeds, the server responds with text that includes hyperlinks to other actions that are currently available.

Richardson Maturity Model

A model used to describe the maturity and quality of RESTful services.

There is no formal specification for REST.

RMM Levels

Level 3: Hypermedia Controls, makes API more self-documenting

Level 2: HTTP Verbs (most common)

Level 1: Resources

Level 0: The swap of POX (plain old XML)

HTTP Request Methods

GET, get information

POST, send information

PUT, update existing information

DELETE

HTTP Status Codes

100 - informational

200 - successful

300 - redirections

400 - client errors, bad request

500 - server errors

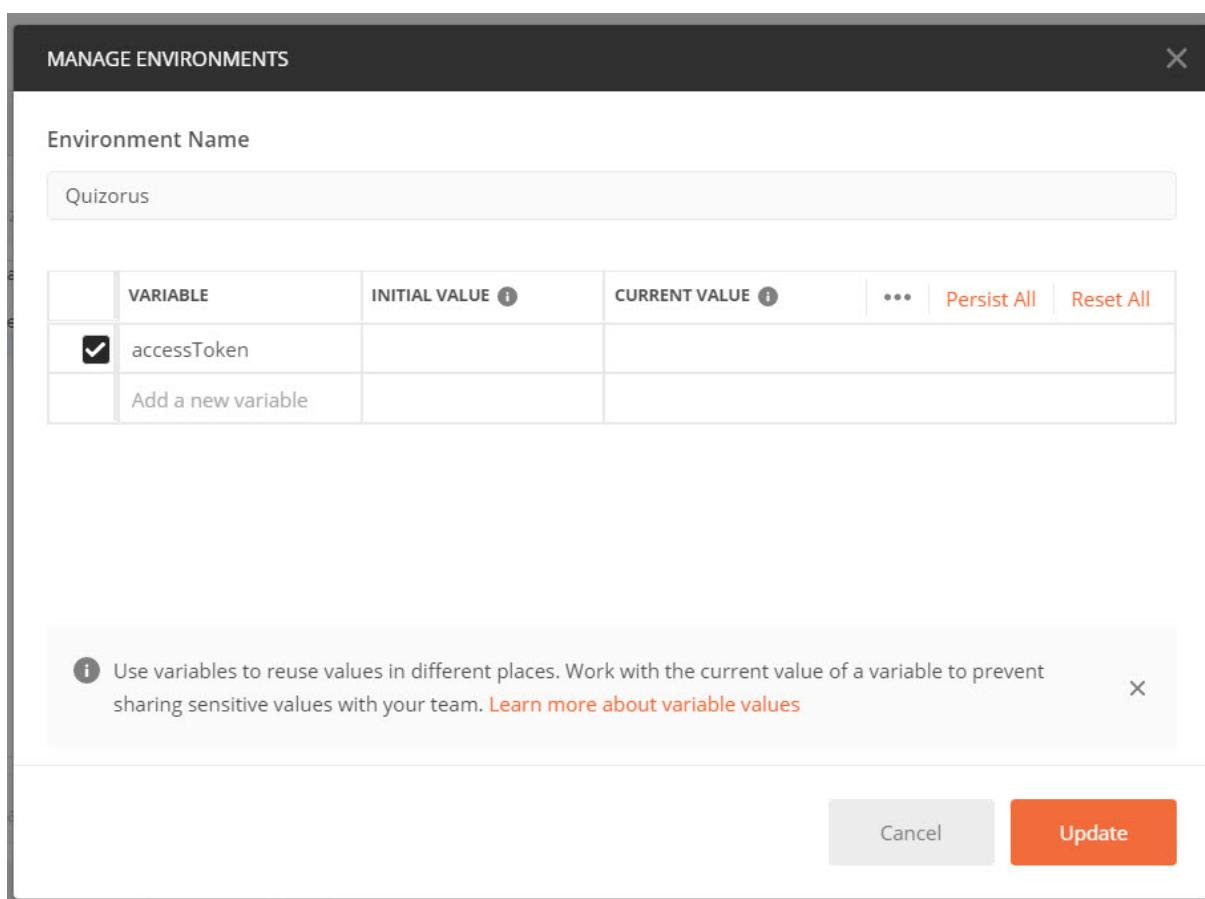
Overview of Security Mechanism

- API to register new users with their full name, username, email and password.
- API to let users log in using their username/email and password. After validating the user's credentials, the API should generate JWT token and return the token in the response. The client will use that token in the Authentication header of all requests to access any protected resources.
- Configure Spring Security to restrict access to protected resources and also will throw 401 for unauthorized access.
- Configure Role-based Authorization to protect resources on the server.

How To Use Postman with JWT Token

Note: Postman v7.0.9 is used for the project.

- Add accessToken variable to postman environmental variable.



- Then use Tests tab to write javascript.

```
let jsonData = pm.response.json();
let token = jsonData.accessToken;
pm.environment.set('accessToken', token);
```

The screenshot shows the Postman interface with a 'POST login' collection. Under the 'POST /login' request, the 'Tests' tab is selected. It contains the following JavaScript code:

```
1 let jsonData = pm.response.json();
2 let token = JSON.stringify(jsonData.accessToken);
3 pm.environment.set('accessToken', token);
4
```

- If you are using the current version "Bearer Token" type is under Authorization tab.

The screenshot shows the Postman interface with a 'POST /topics' request. The 'Authorization' tab is selected. The 'TYPE' dropdown is set to 'Bearer Token'. The 'Token' field contains the placeholder {{accessToken}}. A warning message is displayed: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.' A 'Preview Request' button is visible at the bottom left.