TRAVEL BY BUS

"Web development using Spring Framework v5"

12.02.2020

Veronika Valentinova, 62073

1. Project description

Travel by bus (TBB) is online system for browsing and buying bus tickets. It supports different roles – Traveler, Bus Company and Admin. The system provides ability for travelers to search the best route, and for bus companies to manage their lines. In addition to that it allows users to register, and administrators to manage them.

The system is developed using Spring 5 Application Development Framework. The frontend uses Angular 8 to make Single Page Application (SPA). The backend is implemented as a REST/JSON API using JSON data serialization. Routing is done client-side. JWT is used for authorization.

The main user roles are:

- Bus Company
  - Create new bus lines with information for start point, end point, price, distance, duration, departure time, number of seats and working days
  - Edit/Delete bus line
  - View list of bus line details
- Traveler
  - Search route by start point, end point and travel date
  - Sort results by different criteria – duration, price, distance
  - Buy ticket
  - View list of all bought tickets
- Administrator
  - Manages users, bus lines and tickets
- All users
  - Edit personal profile data – password, first name, last name

1.1. Main Use Cases / Scenarios

Use case name	Brief description	Actors involved
Manage bus lines	Bus company can create new company lines, edit them, view list of details for existing lines and delete them	Bus company, Administrator

	Administrator can view, edit and delete bus lines of all companies	
Search route	Traveler can search route by start point, end point and travel date	Traveler
Buy ticket	When traveler has found the best route, he can buy ticket for it	Traveler
View tickets	Browse bought tickets	Traveler
Manage tickets	Administrator can view list of all tickets, details for ticket, edit and delete tickets	Administrator
Login	Login with username and password	Traveler, Bus company, Administrator
Register	Anonymous User can register in the system by providing a unique username, first and last name, and choosing password. By default, all new registered users have Traveler role. Administrator can register new by entering User Data and choosing a Role (Traveler, Bus company, Administrator).	Anonymous User, Administrator
Update profile	Registered user can update his own profile	Traveler, Bus company, Administrator

2. Technologies used

Backend:

- Java 1.8, Spring Boot 2.2, MongoDB 4.2, Gradle 6.0.1, jjwt 0.9.1

Frontend:

- Angular 8.3.21, NodeJS v12.14.0, Angular Material 8.2.3, Flex-layout 8.0.0-beta.27, moment.js 2.24.0

3. System Architecture

As described in the project summary, the application consists of 3 main components:

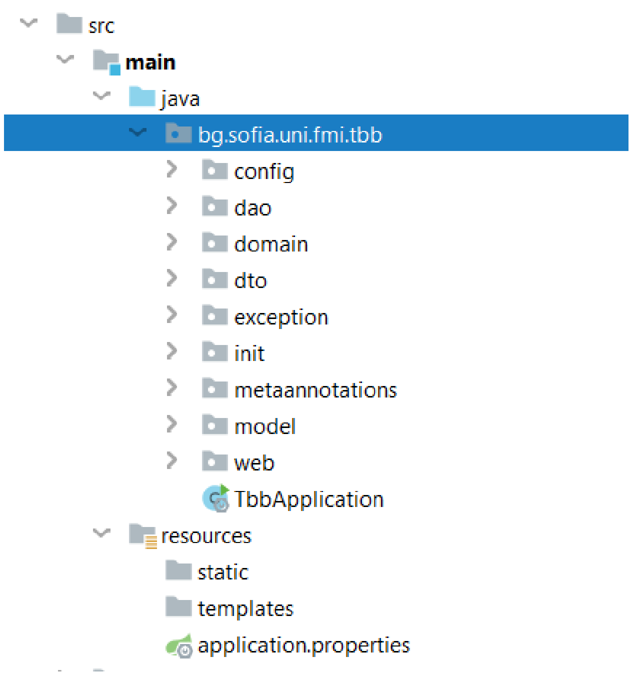
Backend application developed with Spring, frontend application developed with Angular and MongoDB database.

BE implements CRUD Web operations and exposing them through REST APIs so that UI clients can invoke these operations. It uses Domain Driven Design. Main layers are Domain, Repository and Controller.

Domain package contains anything related to TBB business logic and communicates with the database layer.

Dao package contains CRUD repositories to access the required data from the database.

Web package contains REST controllers for communication with TBB API. Global error handling is implemented here using ControllerAdvice.



```
src
  main
    java
      bg.sofia.uni.fmi.tbb
        config
        dao
        domain
        dto
        exception
        init
        metaannotations
        model
        web
        TbbApplication
    resources
      static
      templates
      application.properties
```

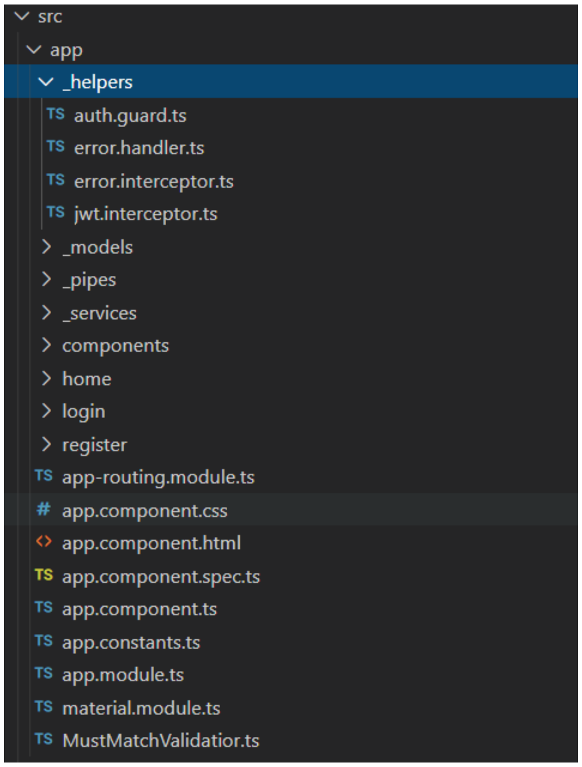
JSON Web Token (JWT) Is used for authorization. It defines a compact and self-contained way for securely transmitting information between parties as a JSON object - a stateless authentication mechanism as the user state is never saved in server memory.

Config package is used to store configuration classes – JWT configuration, Spring Web Security Configuration and CORS configuration.

Custom metaannotations are used to apply method security.

Model package contains the models for the database – User, Ticket, BusLine, Route, Stop. Stop represents cities in the transport network and all stops are retrieved in the frontend to search route. Relationship between BusLine and Route is represented as embedded document as route exists in the context of bus line.

FE application has only one module – the main AppModule as the project is not too big and does not need to be more complicated. Only authenticated users can access it.



```
src
  app
    _helpers
      auth.guard.ts
      error.handler.ts
      error.interceptor.ts
      jwt.interceptor.ts
    _models
    _pipes
    _services
    components
    home
    login
    register
    app-routing.module.ts
    app.component.css
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.constants.ts
    app.module.ts
    material.module.ts
    MustMatchValidatior.ts
```

This is the structure where _helpers folder provides core functionality services and classes:

○ JWT interceptor, which attaches jwt token to Authorization header for every request
○ AuthGuard which is Angular mechanism for protecting routes
○ Error handler which displays error message and logs the error (currently in the console)
○ Error interceptor to handle unauthorized or Forbidden responses

The components, pipes and models folders contains the components, pipes and models used across the application.

4. REST API and Main views

API Resources		
Name	**Brief Descriptions**	**URI**
Users	GET User Data for all users - Available only for Administrators. POST new User Data (Id is auto-filled and modified entity is returned as result from POST request) – used to register new user.	/api/users
User	GET, PUT, DELETE User Data for User with specified userId.	/api/users/{id}

Login	POST User Credentials (username and password) and receive a valid Security Token to use in subsequent API requests.	/api/login
Logout	POST a logout request for ending the active session and invalidating the issued Security Token.	/api/logout
Company line	GET, PUT, DELETE for company line with specified lineId. PUT/DELETE are available for Administrators and Bus Company.	/api/busLines/{id}
Bus lines	GET for all existing bus lines. POST for new line (Id is auto-filled and modified entity is returned as result from POST request) - Available for Administrators and Bus Company.	/api/busLines
Search route	GET for bus line results with specified start point, end point and travel date.	/api/busLines/{startPoint}/{endpoint}/{date}
User tickets	GET tickets for user, POST new Ticket Data for current user (ticketId is auto-filled and modified entity is returned as result from POST request). If there are no seats for this route, an exception is thrown	/api/tickets
Ticket	GET, PUT, DELETE for ticket with specified ticketId.	/api/tickets/{id}

Main Views (Frontend)		
View name	**Brief Descriptions**	**URI**
Home	Landing page after login. Home page for TBB	/
Login	Login with username and password	/login
Registration	Register in the system with username, first name, last name, password. Choose role – bus company or traveler	/register
Search routes	Provides ability for traveler to search route, view search results and buy ticket	/search

My Tickets	View list of bought tickets. Available for traveler.	/tickets
Company lines	Provides ability for bus company to view all lines, create new line, edit/delete exiting line	/company-lines
Profile	All users that are logged in can view and update personal data	/profile
Manage users	View list of users. Edit or delete them. Create new user with role Admin, Bus Company or Traveler	/manage-users

5. Configuration

5.1 Prerequisites:

In order to start the backend you shoud have Java 8+ (with JDK), Gradle and MongoDB configured:

https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html

https://docs.mongodb.com/manual/administration/install-community/

On Windows OS MongoDB can be installed as Windows Service.

The frontend application requires NodeJS and Angular CLI:

https://angular.io/guide/setup-local

5.2 Running backend application with Gradle

Navigate to project folder -> backend and open command prompt.

Enter *gradlew bootRun*

5.3 Running frontend application with Angular CLI:

Navigate to project folder/frontend and open command prompt.

Enter *npm install* to install dependencies.

Enter *ng serve –open* to launch the server from Angular CLI.

6. User documentation

Register or login In order to use the system.

6.1. Traveler

Default user:
Username: traveler
Password: Traveler123

After login traveler lands on home page. He can choose to view and update his profile from ”Profile”.
Traveler can search route from “Search route”. He has to specify start point, end point and travel date.
For example search route from **Sofia** to **Plovdiv**. Traveler sees table containing results of his search or message if no results were found.
In the table with results he can buy ticket. If bus company is out of seats for this route, an error message is displayed. Otherwise the ticket is bought and added to “My tickets”.

6.2. Bus company

Default user:
Username: company
Password: Company 123

After login bus company user lands on home page. BC view and update his profile from ”Profile”. “Company lines” provides ability for BC to manage his lines. To add new line, BC user has to specify all required fields – start point, end point, distance, duration, price, number of seats, working days and departure time.
BC user can edit line from the table with all company lines. He has to select the ‘edit’ button from the ‘Action’ column. He then sees dialog with all data preselected and can update all fields.
Or he can delete bus line as selecting ‘delete’ button from the ‘Action’ column.

6.3. Administrator

Default user:
Username: traveler
Password: Admin123

After login admin lands on home page. BC view and update his profile from ”Profile”.
From the tab “Manage users” he can update/delete existing user or create new user with role Admin, Traveler or Bus Company.

7. Conclusion

This project project helped me learn a lot and practice my analytical thinking and design. It can be improved in many areas – more flexibility for route search, integration of map to visualize the route, ability for user to choose seat. With more time it can be further extended to offer different kind of transport.

8. References

Publication	URL	
Introduction to Spring Method Security	https://www.baeldung.com/spring-security-prefilter-postfilter	Last modified: December 24, 2019, By Baeldung
Spring Security – @PreFilter and @PostFilter	https://www.baeldung.com/spring-security-prefilter-postfilter	Last modified: December 16, 2019 By baeldung
Error Handling for REST with Spring	https://www.baeldung.com/exception-handling-for-rest-with-spring	Last modified: February 4, 2020 By baeldung
Spring Boot Security Jwt Authentication	https://www.devglan.com/spring-security/spring-boot-jwt-auth#/projectstrct	By Dhiraj, 21 October, 2017
Model One-to-Many Relationships with Embedded Documents	https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/	MongoDB, Inc 2008-present.
Embedded documents with Spring Data and MongoDB	https://lankydan.dev/2017/05/29/embedded-documents-with-spring-data-and-mongodb	May 29, 2017
Angular Authentication: Using Route Guards	https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3	Ryan Chenkie, Jul 24, 2017
Angular Material Dialog: A Complete Example	https://blog.angular-university.io/angular-material-dialog/	Last Updated: 26 APRIL 2019
Angular Flex-Layout: Flexbox and Grid Layout for Angular Component	https://medium.com/angular-in-depth/angular-flex-layout-flexbox-and-grid-layout-for-angular-component-6e7c24457b63	Suguru Inatomi, Sep 28, 2018
Angular 7 - Reactive Forms Validation Example	https://jasonwatmore.com/post/2018/11/07/angular-7-reactive-forms-validation-example	JASON WATMORE, PUBLISHED: NOVEMBER 07 2018