Projekt zrealizowany w ramach przedmiotu Inżynieria Oprogramowania.
Autorzy: Szymon Fugas, Juliusz Ryłko, Miłosz Szwedo, Bartłomiej Zachariasz

SPIS TREŚCI:

ARCHITEKTURA SYSTEMU

link -
https://raw.githubusercontent.com/supsub/TramWaySimulator/master/interfaces2.png
(https://raw.githubusercontent.com/supsub/TramWaySimulator/master/interfaces2.png)

PRZYPADKI UŻYCIA

link -
https://raw.githubusercontent.com/supsub/TramWaySimulator/master/use_cases_final.png
(https://raw.githubusercontent.com/supsub/TramWaySimulator/master/use_cases_final.png)

LISTA WYMAGAŃ

Wymagania funkcjonalne:

I.Użytkownik

1.Zapewnienie rejestracji i logowania

2.Obsługa zaproszenia do gry

Użytkownik będąc w panelu widoku gry ma możliwość kliknięcia przycisku "Zaproś znajomych", wyszukania innych użytkowników i zaproszenia ich na daną grę. Wtedy system wysyła zaproszonym stosowne powiadomienie. Odbiorca po kliknięciu w powiadomienie może przejść do panelu gry i zapisać się na nią.

3.Dodanie obiektu

Użytkownik wybiera opcję "Dodaj obiekt". Wpisuje nazwę obiektu, adres oraz rodzaj obiektu(hala,orlik). Jeżeli wszystko się zgadza, klika opcję "Dodaj obiekt". Zostaje poinformowany że obiekt został dodany.

II.Szukający

1.Szukanie gry

Użytkownik wybiera jedną interesującą go dyscyplinę(np.piłka nożna), następnie może określić parametry gry takie jak: poziom gry(checkbox), **obszar przeszukiwania** (użytkownik podaje punkt na mapie i promień, można ręcznie dodawać i usuwać uwzględnione boiska), **przedział godzinowy gry**, datę, maksymalny koszt oraz minimalną ilość potrzebnych graczy. Użytkownik klika opcję "Szukaj". Zostają wyświetlone gry, które spełniają sprecyzowane parametry. Użytkownik może kliknąć dowolną z nich aby wyświetlić pełne szczegóły wraz z opcją zapisu na grę.

Uwaga: Różne sporty mogą mieć różne dodatkowe parametry, które należy uwzględnić, dla przykładu piłka nożna może mieć dodatkowe pole 'nawierzchnia', gdzie użytkownik precyzuje czy chce grać na hali czy może na sztucznej trawie.

2.Obsługa zapisów do gier

Będąc w widoku szczegółowym danej gry, użytkownik może kliknąć opcję "Zapisz mnie". Jeżeli gra posiada komplet graczy lub użytkownik nie łapie się na priorytetowy nabór, zamiast opcji "Zapisz mnie", wyświetlony zostaje stosunkowy komunikat oraz opcja "Zapisz mnie na rezerwę"(rezerwa działa na zasadzie kolejki). W momencie gdy zwolni się miejsce, lub gra zacznie akceptować graczy o mniejszym priorytecie, system **automatycznie zapisuje** tylu graczy z rezerwy, ilu jest potrzebnych aby wypełnić braki kadrowe. Zapisany staje się pełnoprawnym uczestnikiem spotkania, ma obowiązek się na nim pojawić.

Użytkownik może w każdej chwili wypisać się z gry do której się zapisał, jednak musi liczyć się z konsekwencjami, że jeżeli zrobi to za późno, osłabi swoją reputację na portalu.

III.Organizator

1.Utworzenie gry

Użytkownik wybiera typ dyscypliny, boisko, godzinę, poziom gry, koszt, ilość potrzebnych graczy.Te rzeczy nie mogą ulec późniejszej zmianie i zostają do końca takie jakie są. Następnie ma możliwość ustalenia pewnego priorytetu akceptacji.

Przy wybraniu opcji z priorytetem, organizator może sprecyzować konkretne role boiskowe(np. dla piłki nożnej będą to: bramkarz, obrońca, pomocnik, napastnik), które potrzebuje do swojej gry. Wybiera też ilość każdej z poszczególnych ról (domyślnie 0). Następnie, może wyznaczyć godzinę zaniknięcia priorytetu. Po tej godzinie, już wszyscy użytkownicy niezależnie od roli mają równy priorytet zapisu na grę.

Przed godziną zaniknięcia priorytetu użytkownicy, którzy nie łapią się tylko na pozycję "do przyjęcia" i tak mogą tę grę wyświetlać, a nawet na nią zapisywać. Zapisują się jednak wtedy na rezerwę.

Uwaga: Organizator może też wybrać opcję bez priorytetu.

2.Zarządzanie grą

Użytkownik w panelu własnej utworzonej gry ma możliwość usunięcia jej z systemu. W takiej sytuacji wszyscy zapisani na nią gracze (zarówno główny skład jak i rezerwa) zostają o tym fakcie powiadomieni. Istnieje możliwość dodania komentarza do tej operacji, który wyświetli się wraz z powiadomieniem.

3.Wysłanie komunikatu członkom spotkania

Użytkownik w panelu własnej utworzonej gry ma możliwość wysłania komunikatu do wszystkich członków spotkania. Można tam informować o rzeczach typu kolor koszulek, posiadanie znaczników itp. W takiej sytuacji wszyscy zapisani na nią gracze (zarówno główny skład jak i rezerwa) otrzymują ten komunikat.

Wymagania niefunkcjonalne:

- Maksymalny czas oczekiwania na odpowiedź serwera - 4 sekundy
- Maksymalna ilość klientów naraz - 100
- Zapewnione podstawowe bezpieczeństwo
- Ilość obsługiwanych transakcji na sekundę - 50
- Skalowalność na inne dziedziny sportowe

STOS TECHNOLOGICZNY

link - https://raw.githubusercontent.com/supsub/TramWaySimulator/master/stack.png
(https://raw.githubusercontent.com/supsub/TramWaySimulator/master/stack.png)

RAPORT Z TESTÓW

Testy jednostkowe i integracyjne.

W tej fazie przeprowadzano testy jednostkowe i integracyjne backendu aplikacji wykorzystując JUnit, Mockito, Mockmvc, AssertJ i Hamcrest. Testy sprawdzające komunikację z bazą danych wykonywano najpierw na bazie w pamięci H2, później na bazie danych MySQL środowiska testowego.

Testy systemowe.

W fazie testów systemowych korzystano ze zautomatyzowanych za pomocą frameworka Protractor testów funkcjonalnych mających na celu zweryfikowanie poprawności działania aplikacji przy wybranych scenariuszach testowych. Funkcjonalności których nie udało się przetestować w sposób automatyczny przetestowano manualnie.

Testy akceptacyjne.

Polegały zgodnie z projektem na testach manualnych wykonywanych przez wcześniej niezaangażowanych w testowanie członków zespołu ukierunkowanych na sprawdzenie zgodności aplikacji z wymaganiami funkcjonalnymi i niefunkcjonalnymi zdefiniowanymi na etapie projektowym.

Przy większych zmianach projektu uruchamiano obecne wtedy testy automatyczne w charakterze testów regresyjnych aby upewnić się o nie naruszeniu poprawności działania zaimplementowanych dotychczas funkcjonalności.

Uruchamianie testów

Testy backendu uruchamiano w IDE Intellij IDEA za pomocą polecenia Run All Test dostępnego z menu kontekstowego po kliknięciu PPM na katalog zawierający moduł testowy. Wszystkie zależności niezbędne do uruchomienia testów są wyszczególnione w pliku z zależnościami projektu z Mavena i zainstalują się wraz z resztą aplikacji po zastosowaniu się do opisanej niżej procedury instalacyjnej. Do uruchomienia automatycznych testów funkcjonalnych potrzebny jest Protractor, którego można zainstalować z npma. Automatyczne testy funkcjonalne uruchamia się wchodząc do katalogu e2e we frontendzie i wywołując `protractor app.e2e-spec.ts` w konsoli, po uprzednim uruchomieniu frontu i backendu aplikacji.

LISTA NARZĘDZI

W projekcie zostanie użyty system kontroli wersji - **git**. Repozytorium będzie hostowane na serwisie **GitHub** co umożliwi m. in. na issue tracking lub to-do list.

OPIS INTERFEJSÓW

Aplikacja będzie posiadała interfejs webowy. Użytkownik będzie miał możliwość utworzenia własnego konta. Po zalogowaniu i wyborze trybu szukania gry, aplikacja wyświetli mapę oraz listę dostępnych gier, spełniających wymagania podane przez użytkownika. W przypadku chęci założenia gry, wyświelony zostanie formularz z danymi niezbędnymi do utworzenia nowej grupy.

INSTRUKCJA OBSŁUGI

IOBackend

Importing project

After you unzip project file, switch to your IDE, and import this project as a Maven project. After that, update Maven project to be sure that every dependency was injected. Then, you are ready to setup database.

Setting up database

Application expect MySQL 8.0 database to work as intended. Next, it is necessary to create user assigned to localhost:

```
CREATE USER 'hbstudent'@'localhost' IDENTIFIED BY 'hbstudent';

GRANT ALL PRIVILEGES ON * . * TO 'hbstudent'@'localhost';
```

After that, you are ready to execute SQL script which generate database. Location of script: `io-project-02/sql_scripts/main_script.sql`

Running the application locally

To run spring boot application execute the `main` method in the `io-project-02/src/main/java/com/example/demo/SpringBootRestRegistrationApplication.java` class from your IDE.

IOFrontend

This project was generated with Angular CLI (https://github.com/angular/angular-cli) version 7.0.6.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically

reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory. Use the `--prod` flag for a production build.

Further help

To get more help on the Angular CLI use `ng help` or go check out the Angular CLI README (https://github.com/angular/angular-cli/blob/master/README.md) .

REST API

Dokumentacja REST API projektu dostępna jest pod linkiem:
`https://documenter.getpostman.com/view/6384574/RznJmGNB?fbclid=IwAR1Z-l-CBcIejxWmsg3ZkA-Q4KW7UYAjMF61VHi9BFRCCGMhRoU6CVnwQSQ`