Aplikacja webowa do obsługi rozgrywek tenisowych

(Web application for managing tennis tournaments)

Marcin Wróbel

Praca inżynierska

Promotor: dr hab. Dariusz Biernacki

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

13 stycznia 2024

Streszczenie

Celem pracy jest zaprojektowanie i stworzenie aplikacji webowej umożliwiającej organizację rozgrywek tenisa ziemnego. Projekt obejmuje opracowanie responsywnego interfejsu użytkownika wraz z zapleczem. Każdy element aplikacji został skonteneryzowany i skonfigurowany do współpracy z resztą projektu.

The aim of the project is to design and create web application for organizing tennis tournaments. The project includes developing responsive user interface along with the backend. Each component of the application has been contenerized and configured to work with the rest of the project.

Spis treści

Rozdział 1.

Wprowadzenie

Obsługa amatorskich lig tenisa ziemnego bardzo często polega na wykorzystaniu notesu i długopisu. W ramach tych lig zawodnicy są zazwyczaj zobowiązani do własnej organizacji meczów i pozyskiwania danych kontaktowych innych uczestników w celu ustalenia terminów spotkań. Pojawiła się potrzeba stworzenia nowoczesnego narzędzia do organizacji amatorskich rozgrywek ligowych. W poniższej pracy przedstawiono aplikację webową, która odpowiada na tę potrzebę.

Aplikacja umożliwia łatwe wprowadzanie wyników meczów, weryfikuje je oraz uwzględnia przypadki szczególne, takie jak walkover, czy krecz. Na podstawie wyników meczów sytuacja w grupie i drabince turniejowej jest obliczana automatycznie. Zawodnicy mają dostęp do danych kontaktowych osób, z którymi mają rozegrać mecz i są powiadamiani o zakończonych meczach i zmianach w tabeli. Aplikacja udostępnia także bogatą parametryzację, pozwalającą dostosować system do wymagań danej rozgrywki.

Aplikacja została stworzona z wykorzystaniem nowoczesnych i popularnych technologii. Warstwa frontendowa została napisana z użyciem frameworka Vue.js, który pozwala na tworzenie dynamicznego interfesju użytkownika z komponentami wielokrotnego użytku. Z kolei backend został napisany w frameworku Spring Boot, który zapewnia niezawodność i wydajność. Dane, niezbędne do sprawnego funkcjonowania aplikacji, są przechowywane w bazie danych PostgreSQL. Całość została skonteneryzowana za pomocą technologii Docker.

Co istotne, dzięki responsywnemu interfejsowi użytkownika, aplikacja jest przystosowana zarówno do urządzeń mobilnych, jak i stacjonarnych, Dzięki temu zawodnicy mają możliwość korzystania z systemu nie tylko w domu, ale również w przerwach w pracy, czy na korcie tenisowym. Aplikacja webowa została opublikowana i jest dostępna na stronie `www.rozgrywkitenisa.pl`.

Rozdział 2.

Specyfikacja zagadnienia

W tym rozdziale zostanie opisana specyfikacja projektu będącego przedmiotem tej pracy. Specyfikacja obejmuje opis wymaganych funkcjonalności, algorytmy odpowiadające za organizację rozgrywki, oraz przypadki użycia.

2.1. Funkcjonalności

Poniższa lista przedstawia listę wymaganych funkcjonalności, wszystkie zostały zrealizowane:

- Aplikacja umożliwia tworzenie rozgrywek. Rozgrywka może być dwuczęściowa: na początku zawodnicy rywalizują w fazie grupowej, a następnie najlepsi awansują do fazy pucharowej. Rozgrywka może być też jednoczęściowa: faza grupowa jest pominięta i wszyscy zawodnicy zaczynają od fazy pucharowej.
- Liczba grup i zawodników w fazie pucharowej jest konfigurowalna. W pojedynczej rozgrywce liczba zawodników została ograniczona do 128, a liczba grup została ograniczona do 24. Limity zostały wybrane arbitralnie i mogą zostać zmienione. Niestety limity są konieczne ponieważ nie można pozwolić użytkownikowi na stworzenie rozgrywki o bardzo dużej (np. 10000) liczbie grup, lub zawodników. Przeglądarka nie potrafiłaby ich wyrenderować.
- Podczas tworzenia rozgrywki skład grup jest losowany. Losowanie jest wspomagane rankingiem. W przypadku rozgrywek bez fazy grupowej rozstawienie zawodników w fazie pucharowej ustalane jest na podstawie rankingu.
- Zawodnicy mają dostęp do danych kontaktowych osób, z którymi będą rozgrywać mecz.
- Wynik meczu może wpisywać organizator/sędzia, lub zawodnicy.
- Wynik meczu powinien obsługiwać przypadki szczególne:

  - walkower - w tabeli wynik liczony jest tak jakby zawodnik poddający się przegrał wszystkie gemy, aż do zwycięstwa w meczu drugiego zawodnika,
  - krecz - w tabeli wynik liczony jest tak jakby zawodnik poddający się przegrał wszystkie gemy od momentu wycofania się, aż do zwycięstwa w meczu drugiego zawodnika.

- Rejestracja użytkowników. Hasło można zresetować za pomocą emaila podenego podczas rejestracji.

- Obliczanie rankingu na podstawie wyników zakończonych rozgrywek.

2.2. Generowanie grup

Podczas generowania grup zawodnicy są dzieleni na koszyki według rankingu. Dla rozgrywki o N zawodnikach i M grupach w pierwszym koszyku znajduje się M zawodników o najlepszym rankingu, w drugim koszyku kolejne M zawodników, itd. Wyjątkiem jest ostatni koszyk. Jeżeli $N \not\equiv 0 \pmod{M}$, wtedy liczba zawodników w ostatnim koszyku jest mniejsza, wynosi $N \bmod M$.

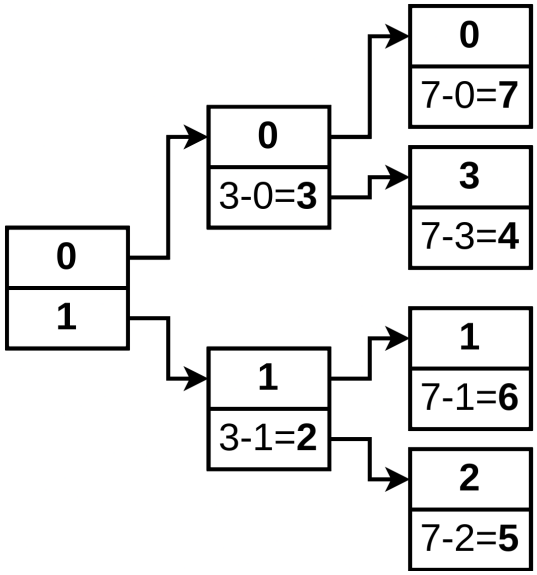
2.3. Generowanie drabinki fazy pucharowej

Na początku generowania drabinki fazy pucharowej zawodnicy są sortowani. Jeżeli rozgrywka posiada fazę grupową, to kolejność zależy od wyników w grupie. Największe znaczenie ma pozycja w grupie, potem bilans meczów, bilans setów, a na końcu bilans gemów. Jeżeli rozgrywka składa się tylko z fazy pucharowej to kolejność zawodników zależy od pozycji w rankingu ogólnym.

Posortowanym zawodnikom przypisane zostają numery startowe od 0 do $N-1$, gdzie N to ilość zawodników. Numery startowe są następnie przypisywane do drabinki według poniższego algorytmu. Jego wizualizacja znajduje się na rysunku 2.1.

```
numeryStartowe=[0,1]
dopoki (numeryStartowe.rozmiar() < N) {
    poprzednieNumeryStartowe = numeryStartowe
    numeryStartowe = []
    najwiekszyNumer = poprzednieNumeryStartowe.rozmiar() * 2 - 1
    dla kazdego (ns w poprzednieNumeryStartowe) {
        numeryStartowe.dodaj(ns)
        numeryStartowe.dodaj(najwiekszyNumer - ns)
    }
}
zwroc numeryStartowe
```

Kolejne pary w zwróconej liście odpowiadają kolejnym parom w drabince fazy pucharowej. Przykładowo dla $N = 8$ zostanie zwrócona lista $[0, 7, 3, 4, 1, 6, 2, 5]$. Odpowiada to parom $[0, 7]$, $[3, 4]$, $[1, 6]$, $[2, 5]$.



Rysunek 2.1: Schemat generowania drabinki fazy pucharowej

2.4. Obliczanie rankingu

TODO ...

2.5. Przypadki użycia

W tej sekcji zostaną opisane przypadki użycia aplikacji webowej będącej przedmiotem tej pracy. Ich celem jest zademonstrowanie działania programu.

2.5.1. Utworzenie rozgrywki

Nazwa	Tworzenie nowej rozgrywki
Aktor	Organizator rozgrywki
Scenariusz	1. Użytkownik znajduje się na stronie rozgrywek i klika utwórz rozgrywkę. 2. Użytkownik uzupełnia wszystkie pola konfiguracyjne: nazwę, czas trwania, liczbę zawodników, liczbę grup, liczbę finalistów, liczbę setów do wygranej, punkty do zdobycia za poszczególne etapy 3. Użytkownik wyszukuje zawodników używając pola wyszukiwania i dodaje ich 4. Użytkownik klika przycisk "Wylosuj skład grup"i aplikacja tworzy losuje grupy 5. Użytkownik klika przycisk "Utwórz", aplikacja tworzy rozgrywkę i przekierowuje użytkownika do strony z nowo utworzoną rozgrywką

2.5.2. Dodawanie wyniku meczu

Nazwa	Dodawanie wyniku meczu
Aktor	Zawodnik meczu, który się zakończył
Scenariusz	1. Użytkownik znajduje się na stronie rozgrywek i wpisuje w polu wyszukiwania nazwę rozgrywki w ramach, której odbył się mecz. 2. Użytkownik klika rozgrywkę, którą wyświetliła mu aplikacja. 3. Użytkownik klika grupę, w której się znajduje. 4. Użytkownik klika mecz, którego wynik chce dodać. 5. Użytkownik klika ikonę edycji i wprowadza wynik meczu. 6. Użytkownik zatwierdza wynik meczu klikając ikonę zapisu. 7. Aplikacja wysyła email do wszystkich zawodników w grupie informując o nowym wyniku.

Rozdział 3.

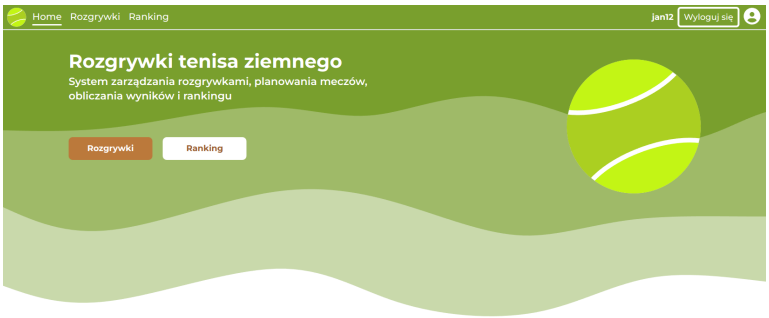
Opis aplikacji

Ten rozdział opisuje interfejs aplikacji pozwalając użytkownikowi zapoznać się z nim. Następnie przedstawione zostały kroki pozwalające instalację aplikacji lokalnie.

3.1. Interfejs użytkownika

W tej sekcji zostaną omówione elementy interfejsu użytkownika.

3.1.1. Strona główna



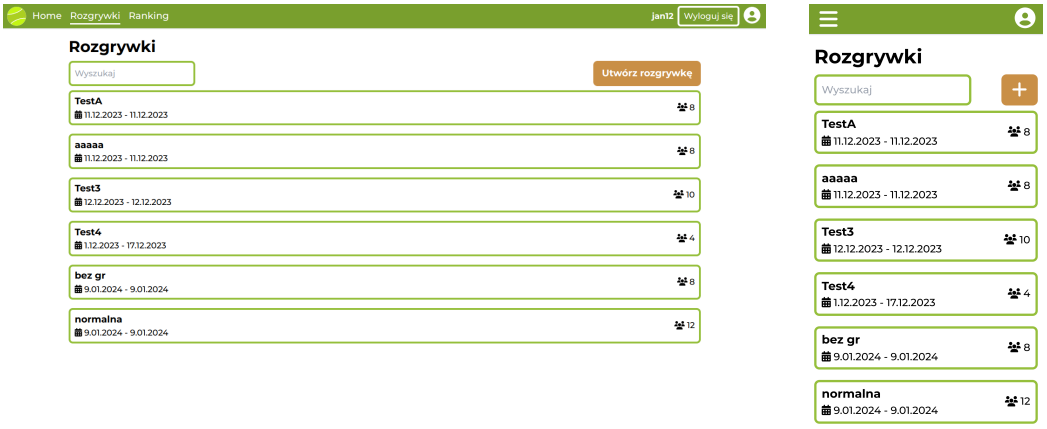
(a) Wersja na komputery

(b) Wersja na telefony

Rysunek 3.1: Strona główna

3.1.2. Strona rozgrywek

Strona rozgrywek zawiera listę rozgrywek, oraz pozwala na wyszukiwanie konkretnej rozgrywki po nazwie. Każda rozgrywka jest opisana podstawowymi informacjami, a po kliknięciu użytkownik jest przenoszony na stronę ze szczegółami. Dodatkowo strona zawiera przycisk "Utwórz rozgrywkę"przenoszący do widoku tworzenia rozgrywki.

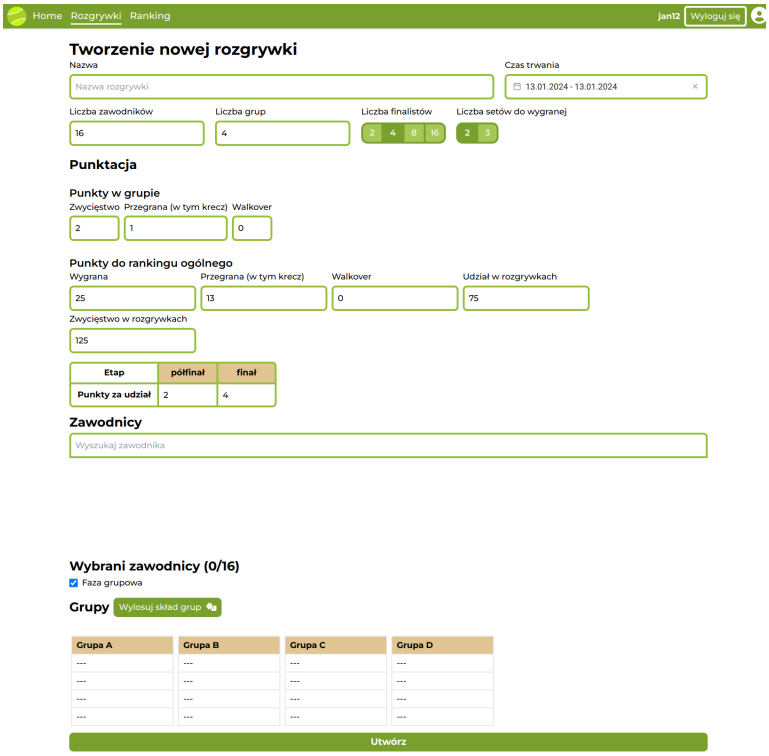


(a) Wersja na komputery (b) Wersja na telefony

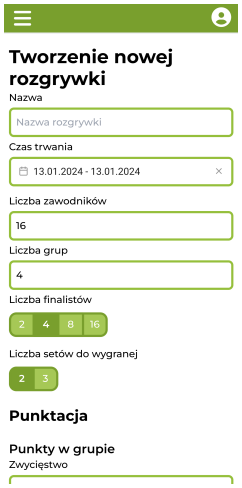
Rysunek 3.2: Strona rozgrywek

3.1.3. Strona tworzenia rozgrywki

Dostęp do tej strony mają tylko zalogowani użytkownicy z uprawnieniami do tworzenia rozgrywek. Użytkownik może dostosować wszystkie widoczne parametry. Dostępne opcje parametru liczba finalistów, są zależne od liczby zawodników. Liczba finalistów musi być potęgą dwójki i nie może być większa od liczby zawodników. Pole wyszukiwania zawodników pozwala na dodanie ich rozgrywki, a przycisk "Wylosuj skład grup"tworzy grupy zgodnie z podziałem na koszyki.



(a) Wersja na komputery

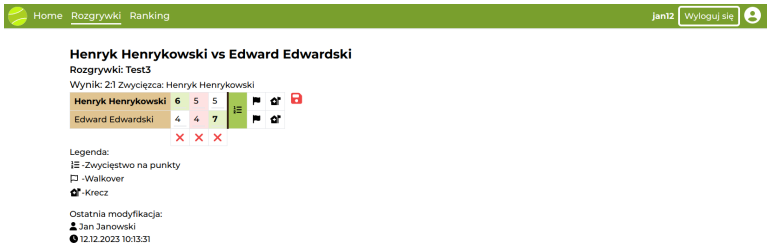


(b) Wersja na telefony

Rysunek 3.3: Strona rozgrywek

3.1.4. Edycja meczu

Wynik meczu może wprowadzić/edytować organizator rozgrywki, lub jego uczestnicy. Aplikacja weryfikuje, czy wynik jest poprawny, i zaznacza na czerwono błędy. Z prawej stronie wyniku użytkownik może wygrać zakończenie meczu zgodnie z legendą, która znajduje się po niżej. Następnie użytkownik może zapisać zmiany klikając ikonę dyskietki.



Rysunek 3.4: Strona edycji wyniku meczu

3.1.5. Logowanie i rejestracja

Strona rejestracji pozwala na stworzenie konta. Nowe konta nie mają uprawnień do organizowania turnieju. W celu uzyskania uprawnień administrator musi dodać odpowiednie uprawnienia użytkownikowi.

Rysunek 3.5: Strona rejestracji

Rysunek 3.6: Strona logowania

3.1.6. Weryfikacja wejścia

Opisane w tej sekcji strony zawierają wiele pól, które pozwalają na wpisywanie danych. Nie można zakładać, że użytkownik zawsze wpisze wszystkie dane poprawnie. W związku z tym wszystkie pola, które akceptują dane użytkownika są weryfikowane. W przypadku błędu aplikacja wyświetla odpowiedni komunikat.

Rysunek 3.7: Weryfikacja liczby zawodników i graczy w widoku tworzenia rozgrywki

3.2. Instalacja

Aplikacja została przygotowana do uruchomienia w dwóch środowiskach:

- lokalnym - do testowania na urządzeniu użytkownika,
- produkcyjnym - gdy aplikacja jest publicznie dostępna

Instrukcje instalacji dla środowiska lokalnego i produkcyjnego różnią się, różnice są oznaczone.

3.2.1. Zmienne środowiskowe

Należy utworzyć pliki zawierające zmienne środowiskowe (`.env`). Pliki `.env.example` zawierają przykładowe wartości, które pozwalają na uruchomienie w środowisku lo-

kalnym.

- `cp .env.example .env`

- `cd frontend`
  `cp .env.example .env`

Następnie można dostosować wartości do własnej konfiguracji.

3.2.2. Instalacja zależności

Aby zbudować i uruchomić projekt należy zainstalować `Docker` zgodnie z instrukcjami zawartymi w oficjalnej dokumentacji
`https://docs.docker.com/`
`https://docs.docker.com/engine/install/`
Użytkownik musi należeć do grupy `docker`

```
sudo usermod -aG docker nazwa_uzytkownika
```

3.2.3. Budowanie

Aby zbudować projekt należy wykonać poniższe polecenie:

```
# srodowisko produkcyjne
docker compose build
# srodowisko lokalne
docker compose -f docker-compose-localhost.yml build
```

3.2.4. Uruchamianie

```
# srodowisko produkcyjne
docker compose up -d
# srodowisko lokalne
docker compose -f docker-compose-localhost.yml up -d
```

Flaga `-d` nie jest wymagana, uruchamia projekt w tle.

3.2.5. Wyłączanie

```
docker compose down
```

3.2.6. Uwagi do instalacji

Dane aplikacji są przechowywane w folderze `data/`. Aby oczyścić system z wszystkich kontenerów, obrazów, cache i innych plików stworzonych przez Dockera należy w wykonać poniższe polecenie [2]. Należy wykonywać to polecenie z rozwagą, ponieważ usuwa ono także kontenery, obrazy i pliki z innych projektów dockera.

```
docker system prune
```

Przed uruchomieniem należy upewnić się, że żaden inny proces nie używa portu 80, lub 443. Przykładowo domyślna instalacja KDE Neon i innych dystrybucji Linuxa opartych na Ubuntu uruchamia apache2, który używa port 80. Aby zatrzymać apache2 należy wykonać polecenie:

```
sudo systemctl stop apache2
```

3.3. Porównanie z innymi znanymi rozwiązaniami

Rozdział 4.

Architektura aplikacji

Opis architektury aplikacji składa się z ogólnego zarysu struktury, oraz bardziej szczegółowego opisu każdej części.

4.1. Struktura aplikacji

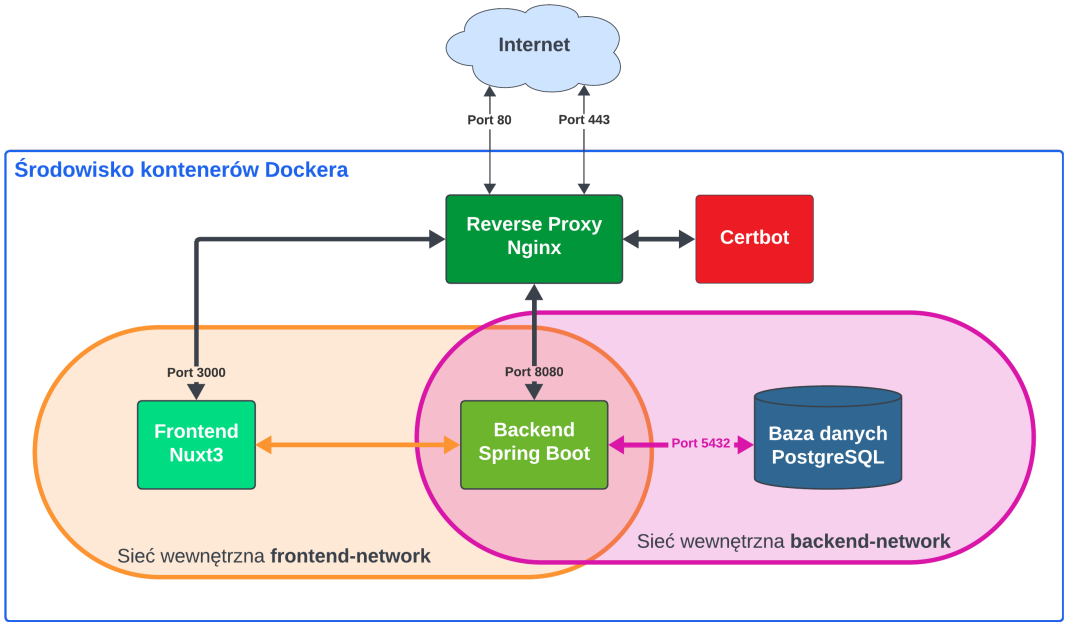
Aplikacja została podzielona na 5 głównych części:

- **frontend** - cześć aplikacji widoczna dla użytkownika
- **backend** - odpowiada za przekazywanie i odbieranie danych od frontendu, odczytuje i zapisuje dane do bazy danych
- **baza danych** - przechowuje dane aplikacji
- **reverse proxy** - wysyła, odbiera, przekazuje zapytania do odpowiednich części aplikacji
- **narzędzie do odnawiania certyfikatów SSL/TLS** - automatyzuje proces odnawiania certyfikatów SSL/TLS

Każda część aplikacji jest opisana w następnej sekcji.

4.2. Kontenery Dockera

W celu utrzymania dobrej współpracy pomiędzy wszystkimi elementami aplikacji zdecydowałem się użyć Dockera[1]. Każda z 5 części aplikacji jest opakowywana w osobny kontener, czyli lekką maszynę wirtualną, która zawiera tylko potrzebne do jej działania zależności. Konfiguracja całego środowiska znajduje się w pliku `docker-compose.yml`. Najważniejsze elementy zostały pokazane na rysunku 4.1.



Rysunek 4.1: Struktura kontenerów Dockera

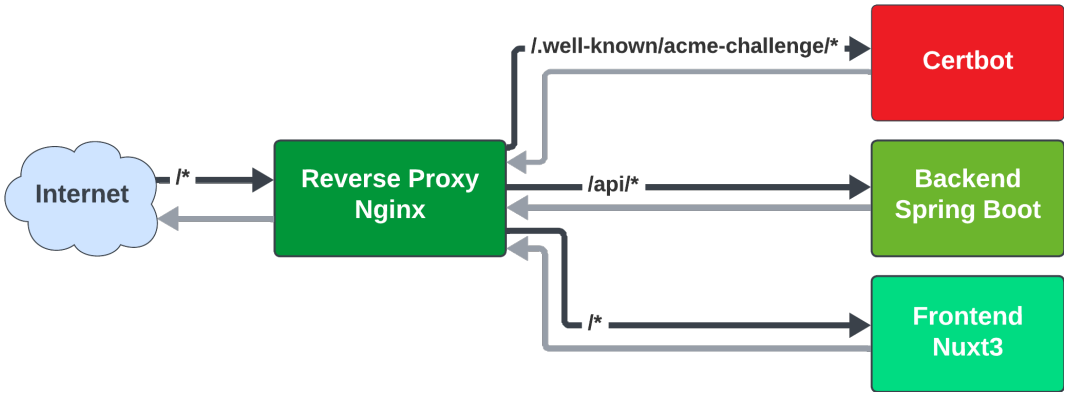
4.2.1. Reverse Proxy - Nginx

Nginx[3] działa jako Reverse Proxy i jest odpowiedzialny za przekierowywanie zapytań http i https do odpowiednich części aplikacji. W folderze `nginx/config/` znajdują się 3 pliki konfiguracyjne. Podczas działania aplikacji używany jest dokładnie jeden z nich.

- `nginx_init_ssl_certificate.conf` jest używany podczas pierwszego uruchomienia w środowisku produkcyjnym, obsługuje tylko zapytania http i pozwala kontenerowi Certbot(4.2.5.) na pobranie certyfikatu SSL/TLS. Jest potrzebny, ponieważ nie jest możliwe obsługiwanie zapytań https bez poprawnego certyfikatu SSL/TLS.

- `nginx.conf` jest używany w środowisku produkcyjnym. Definiuje przekierowywanie zapytań do odpowiednich kontenerów. W tej konfiguracji nieszyfrowana komunikacja za pomocą http jest przekierowywana do szyfrowanej komunikacji za pomocą https.

- `nginx_localhost.conf` jest używany w środowisku lokalnym.



Rysunek 4.2: Przepływ zapytań https zdefiniowanych w pliku `nginx.conf`

Na rysunku 4.2 pokazano schemat przepływu zapytań https dla pliku `nginx.conf`. W przypadku zapytania http reverse proxy nie przekazuje zapytania do kolejnego kontenera, a odpowiada klientowi od razu przekierowaniem do https. Dla zapytań https zdefiniowane są następujące reguły:

1. Jeżeli url zaczyna się od `/.well-known/acme-challenge/` przekaż zapytanie do kontenera Certbot.

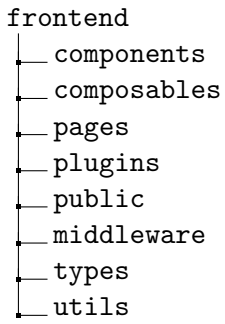
2. Jeżeli url zaczyna się od `/api` przekaż zapytanie do kontenera backend.

3. W przeciwnym przypadku przekaż zapytanie do kontenera frontend.

Wszystkie odpowiedzi na zapytania przekazywane są do reverse proxy, a reverse proxy przekazuje je dalej do nadawcy zapytania.

4.2.2. Frontend - Nuxt3, Vue.js

Frontend został napisany we frameworku Nuxt3 [4], który oparty jest na Vue.js. Użyłem języka TypeScript, ponieważ wprowadza on statyczne typowanie. TypeScript w porównaniu do JavaScriptu ułatwił pisanie kodu, dzięki lepszym podpowiedziom środowiska programistycznego oraz pozwolił zapobiegać błędom, szybciej ostrzegając o problemach podczas kompilacji. Do utrzymania jednolitego formatowania kodu źródłowego użyłem narzędzia Prettier [5]. Kod źródłowy frontendu znajduje się w folderze `frontend/`. W projekcie struktura plików frontendu oparta jest na standardach zdefiniowanych w dokumentacji Nuxt3.



Rysunek 4.3: Zarys struktury plików

Zawartość tych folderów jest następująca:

- `components/` zawiera elementy interfejsu użytkownika (np. tabela wyników, element tablicy wyników, drabinka turniejowa)
- `composables/` zawiera elementy logiki, które posiadają swój stan i są używane w wielu miejscach. Moja aplikacja definiuje `AuthStatus`, który jest odpowiedzialny za przechowywanie informacji o tym, czy użytkownik jest zalogowany, czy nie.
- `pages/` struktura adresów url odpowiada plikom w tym folderze (np. `rozgrywkitenisa.pl/rozgrywki/123` $\longrightarrow$ `pages/rozgrywki/[id].vue`)
- `plugins/` zawiera konfigurację dwóch pluginów (FontAwesome [7] (dostarcza ikony aplikacji), VueDatePicker [8] (Dostarcza narzędzie do wybierania daty)
- `public/` zawiera statyczne pliki, które nie są zmieniane podczas kompilacji i są przekazywane jako niezmienione klientowi
- `middleware/` zawiera middleware, które odpowiedzialne są za modyfikowanie zapytań przed przekazaniem ich dalej. Moja aplikacja używa tylko 1 middleware `logged.in.ts`, który przekierowuje niezalogowanych użytkowników do strony logowania
- `types/` zawiera pliki definiujące typy języka TypeScript
- `utils/` zawiera funkcje i stałe używane w co najmniej 2 plikach aplikacji

Oprócz wyżej wymienionych technologii zastosowałem bibliotekę TailwindCSS [9] do stylizacji aplikacji. W pliku `tailwind.config.js` znajduje się konfiguracja tej biblioteki, wraz z dedykowaną dla tego projektu paletą kolorów. Wielokrotnie używałem prefiksów dla stylów oferowanych przez TailwindCSS, które aktywują/dezaktywują dany styl w zależności od wielkości ekranu. Dzięki temu aplikacja jest responsywna i dostosowana do komputerów i urządzeń mobilnych.

4.2.3. Backend - Spring Boot

Backend napisałem używając frameworka Spring Boot. Do zarządzania zależnościami i do automatyzacji budowania użyłem narzędzia Maven.
Kod backendu został podzielony na warstwy:

- **kontrolery** - odpowiedzialne za przetworzenie parametrów żądania, wysłanie odpowiedzi
- **serwisy** - odpowiedzialne za logikę biznesową backendu
- **repozytoria** - odpowiedzialne za komunikację z bazą danych

Dodatkowo każdy obiekt przechowywany w bazie danych posiada klasy odpowiedzialne za ich działanie:

- **encje** - definiują pola, cechy obiektu przechowywanego w bazie danych
- **obiekty** transferu danych (DTO) - definiują obiekty przesyłane z, lub do serwera
- **maper** - klasy służące do konwertowania obiektów między encjami, a obiektami transferu danych

Przykładowo dla obiektu odpowiedzialnego za rozgrywkę (`Tournament`) zdefiniowane są następujące klasy:

- `TournamentController` - kontroler
- `TournamentMapper` - maper
- `TournamentRepository` - repozytorium
- `TournamentService` - serwis
- `Tournament` - encja
- `TournamentCreateDto` - obiekt transferu danych wysyłany przez frontend podczas żądania stworzenia rozgrywki
- `TournamentBasicDto` - obiekt transferu danych zawierający tylko podstawowe informacje o rozgrywce, jest używany podczas wysyłania listy wszystkich rozgrywek

4.2.4. Baza danych - PostgreSQL

Do przechowywania danych wybrałem PostgreSQL. Zdecydowałem się wybrać tą technologię ze względu na to, że chciałem użyć relacyjnej bazy danych, która jest wydajna, niezawodna i aktywnie rozwijana. PostgreSQL spełnia wszystkie te wymagania.

Konfiguracja bazy danych jest zarządzana przez backend i narzędzie Flyway [11]. W folderze `backend/src/main/resources/db/migration` znajdują się pliki z poleceniami SQL definiujące migracje pomiędzy wersjami. Gdy schemat bazy danych był zmieniany dokładałem kolejny plik odpowiedzialny za migrację. Przy uruchomieniu backendu Flyway automatycznie sprawdza, czy zostały dodane jakieś pliki migracyjne. Jeżeli tak, to dokonuje migracji. Dzięki temu baza danych używana lokalnie oraz baza danych na serwerze w wersji produkcyjnej utrzymują ten sam schemat bazy danych.

4.2.5. Narzędzie do odnawiania certyfikatów SSL/TLS - Certbot

Dla bezpieczeństwa użytkowników połączenie z stroną internetową powinno być szyfrowane. Do nawiązania szyfrowanego połączenia używane są certyfikaty SSL/TLS, które zawierają klucz publiczny i inne dane, takie jak okres ważności, dane właściciela, wystawcy. Każdy ma możliwość wystawić własny certyfikat, ale przeglądarka będzie ufać tylko certyfikatom wydanym przez zaufane (według twórców przeglądarki) instytucje. Świadomy użytkownik znając wystawcę certyfikatu może ręcznie dodać go do listy zaufanych certyfikatów. Takie rozwiązanie jest właściwe tylko wtedy, gdy nie chcemy powierzać generowania kluczy zewnętrznej instytucji i użytkownicy mogą potwierdzić jego prawdziwość (np. gdy wystawiamy certyfikat aplikacji webowej do użytku wewnętrznego firmy). W związku z tym, że aplikacja webowa będąca tematem pracy będzie publicznie dostępna, generowanie certyfikatów SSL/TLS powierzyłem organizacji Let's Encrypt [13]. Let's Encrypt wystawia darmowe certyfikaty SSL/TLS oraz umożliwia automatyzację tego procesu. Kontener Certbot oparty jest na narzędziu o tej samej nazwie [12]. Odpowiada on za automatyczne odnawianie certyfikatów poprzez wysyłanie zapytania do Let's Encrypt i odpowiadanie serwerom tej organizacji na zapytania weryfikujące - czy prośbę o wydanie certyfikatu wysłał właściciel domeny rozgrywkitenisa.pl.

Bibliografia

[1] Dokumentacja Dockera https://docs.docker.com/

[2] Dokumentacja polecenia docker system prune https://docs.docker.com/engine/reference/commandline/system_prune/

[3] Dokumentacja Nginx https://docs.nginx.com/

[4] Nuxt3 https://nuxt.com/

[5] Prettier https://prettier.io/

[6] Nuxt3 - dokumentacja struktury plików w folderze pages/, https://nuxt.com/docs/guide/directory-structure/pages

[7] FontAwesome https://fontawesome.com/docs

[8] VueDatePicker https://vue3datepicker.com/

[9] TailwindCSS https://tailwindcss.com/

[10] SpringBoot https://spring.io/projects/spring-boot/

[11] Flyway https://flywaydb.org/

[12] Certbot https://certbot.eff.org/

[13] LetsEncrypt https://letsencrypt.org/pl/