



Politechnika Łódzka

Institut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja społecznościowa dla miłośników sportów samochodowych

A social app for car sports enthusiasts

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Promotor: dr inż. Łukasz Chomątek

Dyplomant: Kamil Kowalewski

Nr albumu: 216806

Kierunek: Informatyka

Specjalność: Inżynieria Oprogramowania i Analiza Danych

Łódź, luty 2021



Institut Informatyki

90-924 Łódź, ul. Wólczańska 215, *budynek B9*

tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Spis treści

Streszczenie	4
1 Wstęp	5
1.1 Problematyka	5
1.2 Cel i założenia projektu	6
1.3 Układ pracy	6
2 Wprowadzenie do świata sportów samochodowych	7
2.1 Sporty samochodowe	7
2.1.1 Podział sportów samochodowych	8
2.1.2 Najbardziej rozpoznawalne tory wyścigowe w Europie	8
2.2 Amatorskie starty w zawodach	9
2.3 Analiza dostępnych aplikacji społecznościowe i systemy informatyczne .	10
2.4 Uzasadnienie tworzenia aplikacji <i>Social Racetrack</i>	12
3 Wykorzystane technologie w aplikacji <i>Social Racetrack</i>	13
3.1 Uzasadnienie wyboru technologii	13
3.2 Środowisko uruchomieniowe Node.js	14
3.2.1 Silnik V8	14
3.2.2 Protokół HTTP w Node.js	15
3.2.3 Menadżer pakietów Yarn	15
3.3 Serwis hostingowy GitHub	16
3.3.1 System kontroli wersji Git	16
3.3.2 Platforma GitHub	17
3.4 Framework Bootstrap	17

3.4.1	RWD	18
3.4.2	Responsywna siatka	18
3.4.3	Komponenty stylizowane	20
3.4.4	Projekty rozszerzające funkcjonalności Bootstrap	20
3.4.4.1	Material design	20
3.5	Biblioteka do tworzenia interfejsu użytkownika React.js	21
3.5.1	Technologie wspierające koncept SPA	21
3.5.2	Założenia biblioteki React.js	22
3.5.3	Składnia JSX oraz idea komponentów	22
3.5.4	Rodzaje komponentów	24
3.5.5	React Hooks	27
3.6	Plaforma chmurowa Firebase	29
3.6.1	Authentication	30
3.6.2	Cloud Firestore	31
3.6.3	Storage	33
3.6.4	Hosting	33
3.6.5	Functions	34
4	Dokumentacja techniczna aplikacji <i>Social Racetrack</i>	36
4.1	Wymagania funkcjonalne i нефункционалне	36
4.1.1	Wymagania funkcjonalne	36
4.1.2	Wymagania нефункционалне	39
4.2	Architektura aplikacji	40
4.3	Implementacja	41
4.3.1	Model danych	41
4.3.2	Warstwa obsługi interfejsu programistycznego aplikacji API	43
4.3.3	Struktura stron	44
4.3.3.1	Komponenty dostępne publicznie	45
4.3.3.2	Komponenty dostępne dla zalogowanego użytkownika	45
4.3.3.3	Komponenty dostępne dla administratora	46
4.3.4	Struktura komponentów i ich użycie	46
4.3.5	Responsywność interfejsu użytkownika	48

4.4	Instalacja	50
5	Dokumentacja użytkownika aplikacji <i>Social Racetrack</i>	52
5.1	Wprowadzenie do obsługi interfejsu aplikacji	52
5.2	Obsługa aplikacji dla użytkownika niezalogowanego	54
5.3	Logowanie i rejestracja	56
5.4	Obsługa funkcjonalności dostępnych dla użytkownika zalogowanego . . .	57
5.4.1	Obsługa konta użytkownika	57
5.4.1.1	Dodawanie i usuwanie floty pojazdów	58
5.4.1.2	Dodawanie i usuwanie zdobytych nagród	59
5.4.1.3	Edycja danych osobowych	59
5.4.2	Przeglądanie, dołączanie do wydarzeń i rezygnacja z udziału . . .	60
5.4.3	Tworzenie wydarzeń	61
5.4.4	Przeglądanie kont innych użytkowników	62
5.4.5	Wylogowanie	63
5.5	Obsługa funkcjonalności dostępnych dla administratora	63
5.5.1	Obsługa panelu administratora	64
5.5.2	Usuwanie wydarzeń	64
5.5.3	Dodawanie i usuwanie torów wyścigowych	65
6	Podsumowanie	66
	Spis rysunków	68
	Spis listingów	70
	Spis tabel	72
	Bibliografia	73

Streszczenie

Celem niniejszej pracy inżynierskiej było stworzenie aplikacji społecznościowej dla miłośników sportów samochodowych. W obecnym czasie jest dostępnych wiele portali społecznościowych o zastosowaniach ogólnych natomiast można zaobserwować potężną lukę w portalach społecznościowych ukierunkowanych na pewne grupy społeczne takie jak miłośnicy sportów samochodowych. Wspomniane wcześniej portale społecznościowe o zastosowaniach ogólnych nie zapewniają dedykowanych funkcjonalności i wsparcia dla określonych grup społecznych. W wyniku dogłębnej analizy potrzeb społeczności mocno związanej ze sportami samochodowymi oraz rozwiązań obecnych na rynku została stworzona aplikacja internetowa *Social Racetrack*.

Rozdział 1

Wstęp

W erze potężnego rozwoju cyfrowego świata portale społecznościowe stały się nieodłączną częścią życia ogromnego odsetka społeczeństwa na naszym globie. Ich cele i założenia są różne, jednak wspólnym celem jest łącznie określonych grup społecznych. Aplikacja społecznościowa *Social Racetrack* ma za zadanie spajać i przyciągać osoby zainteresowane sportami samochodowymi.

1.1 Problematyka

Problematyka związana z tworzeniem społeczności jest bardzo rozległa i zdecydowanie nie można o niej powiedzieć, że nie jest trywialna. Jej podwaliny można było zaobserwować już setki, jeśli nie tysiące lat temu w pierwotnych ludach zamieszkujących nasz glob. Z biegiem czasu oraz rozwojem kulturowym zmieniały się relacje, zależności oraz tematyka czy też cel skupiający dane grupy ludzi.

Aktualnie poprzez bardzo szybki rozwój technologii a w szczególności niesamowity postęp w informatyzacji społeczności zaczęły się przenosić do globalnej sieci Internet, która poprzez portale internetowe, często potocznie określane jako strony internetowe, takie jak Facebook[1] czy Twitter[2], łączy ogromne liczby osób w niesamowitej, kiedyś nie do pomyślenia liczbie społeczności. Społeczności te są zgromadzone wokół bardzo różnych dziedzin takich jak np. nauka czy sport stąd mają bardzo zróżnicowane potrzeby. Wcześniej wymienione portale starają się sprostać tym wymaganiom lecz poprzez dużą rozpiętość dziedzin jest to niezwykle trudne.

Dosyć wyjątkową niszą są grupy zrzeszające miłośników sportów samochodowych, gdyż zazwyczaj są to osoby o wysokim statusie społecznym, mające relatywnie wysokie wymagania. Kolejnym aspektem jest fakt, iż przepisy związane z sportami samochodowymi są mocno rygorystyczne więc dostosowanie ogólnych portali społecznościowych do tych potrzeb mogłoby wpłynąć negatywnie na inne grupy społeczne co najprawdopodobniej skutkowałoby rezygnacją z użycia przez coraz większe liczby ludzi a finalnie mogłoby się skończyć na potężnej utracie popularności danego portalu i jego likwidacji gdyż nie byłby on rentowny.

1.2 Cel i założenia projektu

Celem niniejszej pracy inżynierskiej jest zaimplementowanie aplikacji społecznościowej oraz udostępnienie jej w formie nieodpłatnej osobom związanym ze sportami samochodowymi oraz tym osobom, które chcą rozpocząć przygodę z tym sportem. W zakres pracy można włączyć zarówno analizy aktualnie dostępnych rozwiązań na rynku, jak i przygotowanie aplikacji serwerowej z wykorzystaniem platformy chmurowej Firebase oraz aplikacji dla klienta z użyciem technologii Node.js oraz React.

1.3 Układ pracy

Rozdział pierwszy stanowi wstęp do problemu zaobserwowanego przez autora pracy. Drugi rozdział prezentuje sporty samochodowe, aktualnie dostępne aplikacje społecznościowe i ich analizę oraz uzasadnienie stworzenia aplikacji *Social Racetrack*. Kolejnym rozdziałem jest rozdział trzeci, którego zadaniem jest przedstawienie technologii wykorzystanych w aplikacji. Rozdział czwarty skupia się na dokumentacji technicznej składającej się z wymagań funkcjonalnych oraz нефункциональных, architektury aplikacji, jej implementacji oraz procesu instalacji. Następnym, piątym z kolei rozdziałem jest dokumentacja użytkownika tworzonej aplikacji. Ostatni rozdział podsumowuje całą pracę i jest zwieńczeniem przedstawionego problemu oraz jego rozwiązania.

Rozdział 2

Wprowadzenie do świata sportów samochodowych

Z powodu odwiecznej chęci rywalizacji między ludźmi na wielu płaszczyznach oraz coraz szybszego rozwoju technologii pojawiły się sporty samochodowe. Dzięki wybitnym konstruktorom, prężnym gałęziom przemysłu oraz genialnym naukowcom, powstawały coraz bardziej doskonałe pojazdy rozpalające jeszcze większą chęć rywalizacji w bardzo zróżnicowanych warunkach. To właśnie dzięki tym pokoleniom ludzi możemy być teraz świadkami heroicznych zmagania kierowców.

2.1 Sporty samochodowe

Sporty samochodowe są dyscypliną sportową skupiającą się współzawodnictwie, z wykorzystaniem samochodów. Parametrem, poprzez który jest określany zwycięzca oraz kolejność innych uczestników jest zazwyczaj czas przejazdu określonego liczby okrążeń lub dojechanie do wyznaczonego punktu, który często jest trudny do osiągnięcia.

Jako początek tej dyscypliny uznaje się rok 1894, kiedy został zorganizowany pierwszy rajd Paryż-Rouen. Dziesięć lat później czyli w roku 1904 powstała FIA (fr. Fédération Internationale de l'Automobile) czyli międzynarodowa federacja sportów samochodowych. Cel z jakim została stworzona to łączenie narodowych federacji odpowiedzialnych za sporty samochodowe.

Ze względu na fakt, że w wieku XX nastąpił bardzo duży rozwój nauki i techniki same

sporty samochodowe zaczęły bardzo się zmieniać i specjalizować na kategorie zależnie od otoczenia, dystansu, reguł określających zarówno dozwolone i niedozwolone zachowania jak i sposób wyłaniania zwycięzcy. Stopień zaawansowania pojazdów i ich części składowych takich jak silniki czy napędy znacznie wzrastały aby same zawody mogły się odbywać w coraz to innych bardziej ciekawych warunkach wykraczających poza wyobrażenia osób biorących w nich udział dwadzieścia lat wcześniej.

2.1.1 Podział sportów samochodowych

Aktualnie po ponad 120 latach od pierwszego rajdu wspomnianego powyżej sporty samochodowe można podzielić na dwie główne kategorie:

- Wyścigi na torach
- Rajdy na otwartej przestrzeni

Do wyścigów na torach można zaliczyć serie takie jak:

- Formuły F1, F2, IndyCar, które odbywają się w pojazdach jednomiejscowych
- Długodystansowe - ELMS (ang. European Le Mans Series), WEC (ang. World Endurance Championship)
- Turystyczne - DTM, TCR
- Amatorskie - imprezy jednodniowe (ang. Track day), przejazdy turystyczne na przykład na torze Nurburgring Nordschleife

Do rajdów na otwartej przestrzeni należą:

- Rajdy terenowe np. Rajd Dakar
- Cykliczne rajdy organizowane przez FIA - WRC (ang. World Rally Championship), ERC (ang. European Rally Championship)
- Amatorskie - Imprezy jednodniowe (ang. Track day)

2.1.2 Najbardziej rozpoznawalne tory wyścigowe w Europie

W Europie jak i na całym świecie od początku XX wieku powstało wiele torów wyścigowych oraz obiektów ogólnego użytku, które są wykorzystywane w sportach samochodowych. W Europie większość państw o wysokim produkcie krajowym brutto PKB posiada tory wyścigowe, jedne mniejsze znane lokalnie oraz większe, na których odby-

wają się na przykład wyścigi Formuły 1. Wiele z tych większych obiektów zasłynęło ze względu na to, że wymagają bardzo wysokich umiejętności od kierowców czy też bardzo ciekawą i unikalną charakterystyką niespotykaną nigdzie indziej na świecie. Poniżej zostały przedstawione wybrane z nich znajdujące się w Europie oraz wybrane tory zlokalizowane w Polsce:

- Niemcy
 - Nurburgring Nordschleife
 - Nurburgring GP
- Wielka Brytania
 - Silverstone Circuit
- Hiszpania
 - Circuito Ascari
- Belgia
 - Circuit de Spa-Francorchamps (pot. Spa)
- Węgry
 - Hungaroring
- Polska
 - Tor Poznań
 - Tor Łódź
 - Silesia Ring

2.2 Amatorskie starty w zawodach

W związku ze zwiększaniem się dostępności do pojazdów oraz coraz większym zainteresowaniem, od wielu lat są organizowane amatorskie imprezy sportowe dla miłośników sportów samochodowych. Biorą w nich udział zarówno osoby, które w przeszłości brały udział w profesjonalnych rajdach oraz wyścigach, jak i osoby posiadające podstawowe umiejętności jazdy samochodem, które chcą zwiększać poziom swoich umiejętności oraz świadomość podczas jazdy na drodze publicznej. Główną przyczyną brania przez te osoby udziału w takich wydarzeniach jest realizacja swojego hobby i spędzanie wolnego czasu w gronie osób dzielące ich zamiłowania. Ze względu na fakt, iż sporty samochodowe wymagają wysokich nakładów finansowych, zazwyczaj osoby biorące w nich udział po-

siadają średni lub wysoki stan majątkowy. Jest to przeciwieństwo profesjonalnego udziału w zawodach, gdzie większość wydatków na sprzęt pokrywają sponsorzy. Same koszty wynikają z tego, że wynajem czy też budowa obiektu sportowego jest droga, ponieważ musi on spełniać wymogi bezpieczeństwa danego państwa w, którym się znajduje oraz musi być zatrudniona określona liczba osób do jego obsługi. Co więcej sam zakup pojazdu, jego modyfikacje oraz dostosowanie go do wymogów bezpieczeństwa poprzez na przykład instalację klatki bezpieczeństwa jest kosztowny. Często sprawą są też awarie i konieczność wymiany zespołów poddawanych wysokim obciążeniom takie jak sprzęgła czy też turbosprężarki. Wiele osób, które bardzo angażują się w ten sport posiadają własną ekipę serwisową, która jeździ z tą osobą i w razie awarii czy też niezbyt poważnego wypadku jest w stanie przywrócić sprawność pojazdu w relatywnie krótkim czasie.

Same zawody są zazwyczaj organizowane w weekendy lub dni wolne od pracy, dzięki czemu więcej osób ma zapewnioną możliwość wzięcia udziału w danym wydarzeniu sportowym. Organizatorem może być podmiot gospodarczy, który jest właścicielem danego obiektu lub też osoba, która w porozumieniu z właścicielem danego obiektu organizuje tam dane wydarzenie. Sam fakt organizacji danego wydarzenia i chęci zaproszenia zainteresowanych osób ogłasza zazwyczaj w internecie poprzez na przykład portale społecznościowe opisane w sekcji 2.3.

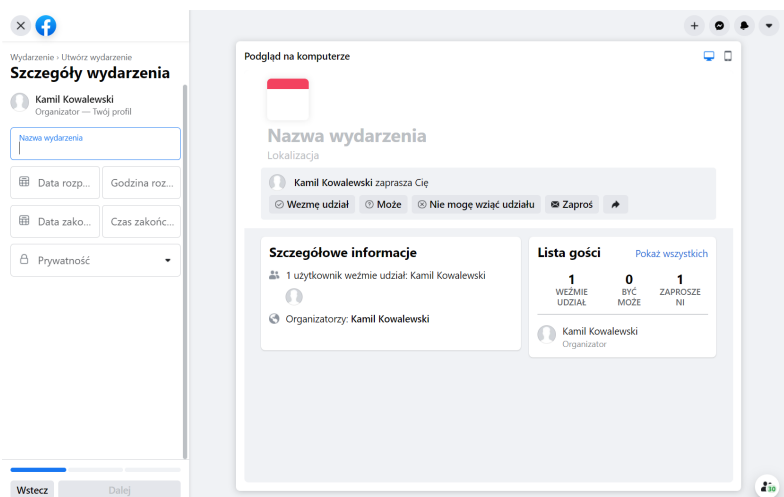
2.3 Analiza dostępnych aplikacji społecznościowe i systemy informatyczne

Do jednych z najpopularniejszych serwisów społecznościowych bez wątpienia należą Facebook[1], Twitter[2] oraz Instagram[3]. Zapewniają one możliwość tworzenia konta bez, którego dostęp do funkcjonalności aplikacji jest mocno ograniczony. Po uwierzytelnieniu użytkownik otrzymuje możliwość tworzenia sieci kontaktów poprzez propozycje znajomości z innymi osobami, dodanie zdjęć i opisów do nich w celu zaprezentowania odwiedzonych miejsc czy też przedstawienia swoich wrażeń na temat wybranego lokalu czy też usługi. Co więcej istnieje również możliwość tworzenia wydarzeń, do których mogą dołączać osoby deklarujące chęć wzięcia udziału. Co ciekawe, przedstawione powyżej portale internetowe Facebook oraz Twitter skupiają się na treściach tekstowych połączo-

nych z sieciami kontaktów oraz zdjęciami w celu lepszej wizualizacji natomiast twórcy portalu Instagram główną wagę przywiązują do zdjęć, poprzez które ma być przekazywana większość informacji a pozostała ich część za pomocą opisu pod zamieszczanym zdjęciem.

Bardzo ważnym aspektem jest to, że portale nie wymagają opłat natomiast muszą one być rentowne aby był możliwy zwrot zainwestowanych w ich stworzenie pieniędzy oraz dalszy rozwój. W tym celu na portalach są publikowane reklamy, które są dostosowywane do preferencji użytkownika. Same preferencje są wyłuskiwane z działań wykonywanych przez użytkownika, które są przechowywane przez serwery danego portalu społecznościowego. Co warto wspomnieć same dane te są przechowywane, analizowane oraz odsprzedawane reklamodawcom w celu lepszego profilowania danego użytkownika co jest dosyć sporną kwestią w kontekście prywatności danych.

Co ważne, same portale są projektowane aby zapewnić funkcjonalności dla statystycznego mieszkańca naszego globu. Są więc one mocno uśrednione aby trafić w potrzeby jak największej liczby odbiorców a co za tym idzie wygenerować jak największe zyski z reklam. Brak w nich możliwości ukierunkowania na określone grupy społeczne, na przykład miłośników sportów samochodowych. Świetnym przykładem jest panel tworzenia wydarzeń przedstawiony na rysunku 2.1. Posiada on bardzo podstawowe możliwości, takie jak nadanie nazwy czy też daty, natomiast brak jest niezwykle wymaganych w społeczności sportów samochodowych parametrów związanych z wymogami bezpieczeństwa sportów samochodowych, takich jak maksymalna dopuszczalna głośność wydechu, prędkość maksymalna czy też prześwit zawieszenia na danym obiekcie sportowym. Analizując najnowsze trendy ważna też jest personalizacja pod kątem profilu użytkownika poprzez zapewnienie możliwości dodanie na przykład informacji o zdobytych w świecie sportów samochodowych osiągnięciach czy też przedstawienie posiadanej floty pojazdów zgromadzonej jako dorobek życiowego zamiłowania.



Rysunek 2.1: Panel tworzenia wydarzenia na portalu Facebook

2.4 Uzasadnienie tworzenia aplikacji *Social Racetrack*

W związku z widoczną luką na rynku specjalistycznych aplikacji społecznościowych została stworzona aplikacja *Social Racetrack*. Jej głównym celem jest wyeliminowanie wad popularnych portali społecznościowych przedstawionych w sekcji 2.3 oraz wypełnienie wcześniej wspomnianej luki. Sama aplikacja ułatwia dogodny sposób komunikacji między miłośnikami sportów samochodowych. Zapewnia dostęp do takich funkcji jak:

- Tworzenie wydarzeń na torach wyścigowych
- Podpowiedzi torów
- Uwzględnianie przepisów bezpieczeństwa
- Możliwość personalizacji konta poprzez dodawanie floty pojazdów oraz zdobytych osiągnięć sportowych
- Możliwość przeglądanie kont innych użytkowników w celu poznania ich dorobku sportowego i posiadanego sprzętu
- System łatwego dołączania do wydarzenia oraz rezygnacji z udziału

Co warto wspomnieć aplikację *Social Racetrack* cechuje:

- Przejrzysty i intuicyjny interfejs użytkownika
- Pełna prywatność danych
- Brak reklam i natrętnych komunikatów
- Szybkość i płynność działania

Rozdział 3

Wykorzystane technologie w aplikacji

Social Racetrack

Projekt stworzony w ramach niniejszej pracy inżynierskiej składa się z dwóch części. Pierwszą z nich jest aplikacja po stronie klienta, która została przygotowana w języku JavaScript[4] uruchamianym w środowisku Node.js[5]. W celu utworzenia aplikacji o większym poziomie rozszerzalności została użyta biblioteka React[6]. Drugą częścią pracy inżynierskiej jest aplikacja po stronie serwera. W tym przypadku przyszła z pomocą platforma chmurowa Firebase[7]. Komunikacja między dwiema częściami aplikacji jest zapewniona poprzez protokół HTTP[8].

3.1 Uzasadnienie wyboru technologii

Przez ostatnie paręnaście lat nie trudno zauważyć ogromny rozwój aplikacji internetowych, co za tym idzie bardzo rozwinęły się języki programowania, frameworki oraz środowiska uruchomieniowe. Przykładami takiego oprogramowania jest chociażby język programowania Java[9] w połączeniu z Spring Framework[10], język programowania C#[11] połączony z frameworkiem ASP.NET Core[12] uruchamianym na platformie .NET Core[13] lub wcześniej już wspomnianym i wykorzystanym w aplikacji środowisku uruchomieniowym Node.js[5] z wykorzystaniem języka programowania JavaScript[4]. Jego bardzo dużymi zaletami są:

- Dystrybucja na zasadach otwartego oprogramowania (ang. open source)

- Możliwość programowania w różnych generacjach JavaScript oraz TypeScript[14]
- Bardzo duża społeczność korzystająca z tej technologii
- Duża liczba bibliotek i frameworków
- Mnogość poradników dzięki, którym tworzenie oprogramowania jest jeszcze szybsze
- Wysoka wydajność dzięki silnikowi V8 oraz procesowi kompilacji kodu zamiast jego interpretacji

3.2 Środowisko uruchomieniowe Node.js

Node.js[5] jest asynchronicznym, sterowanym zdarzeniami środowiskiem uruchomieniowym języka JavaScript. Został zaprojektowany do budowy skalowalnych aplikacji sieciowych. Jest to środowisko, które nie udostępnia programiście możliwości korzystania z wielu wątków natomiast samo w tle korzysta z wielu wątków w celu wykonywania asynchronicznych fragmentów kodu programu. Jest to odmienna filozofia w stosunku do języka Java, gdzie programista może sam tworzyć współbieżne oprogramowanie i w tym celu są wykorzystywane wątki systemu operacyjnego. Ogromną zaletą podejścia jedno-wątkowego w aplikacjach sieciowych jest wzrost wydajności oraz łatwość w użyciu. Co więcej, same funkcje w Node.js praktycznie nie przeprowadzają operacji wejścia / wyjścia oraz poprzez brak blokad zostaje wyeliminowany problem zakleszczenia (ang. deadlock). Wszystkie wyżej wymienione zalety Node.js powodują, że wybór tego środowiska do tworzenia nowoczesnych, skalowalnych i łatwych w rozbudowie systemów jest bardzo dobrą decyzją. Warto podkreślić, że brak wielowątkowości nie oznacza braku możliwości korzystania z wielu rdzeni danego procesora w maszynie. Aby tego dokonać, zostało stworzone *child_process.fork API*, zapewnia ono tworzenie procesów dzieci (ang. child processes) i łatwą komunikację między tymi procesami.

3.2.1 Silnik V8

Sercem środowiska uruchomieniowego Node.js jest silnik V8[15] (ang. V8 engine). Jest on odpowiedzialny za parsowanie i wykonywanie kodu języka JavaScript. Jego au-

torem jest Google Open Source. Jest wykorzystywany przede wszystkim w przeglądarce Chrome[16] oraz w jej otwartoźródłowej wersji Chromium[17]. Jego twórcy położyli ogromny nacisk na wydajność, przez co został stworzony w języku C++ i jest ciągle udoskonalany. Z zasady JavaScript jest językiem interpretowanym przez przeglądarkę natomiast dzięki silnikowi V8 jest on kompilowany w procesie JIT (ang. Just-In-Time compilation). Zapewnia to diametralne przyspieszenie wykonywania programów a co za tym idzie jego zastosowanie są znacznie szersze.

3.2.2 Protokół HTTP w Node.js

Jedną z podstawowych składowych środowiska jest protokół HTTP[8], dzięki temu Node.js staje się serwerem HTTP. Sam protokół jest bezpołączeniowy, zapewnia niskie opóźnienia oraz został stworzony z myślą o przesyłaniu strumieniowym.

3.2.3 Menadżer pakietów Yarn

W sekcji 3.1 została wspomniana zaleta dotycząca bardzo dużej społeczności a co za tym idzie rozwoju tej technologii i powstawania coraz większej liczby bibliotek i pakietów. Aby mieć możliwość skutecznego udostępniania i pobierania do użytku przez innych programistów wspomnianych wcześniej bibliotek oraz pakietów Node.js oferuje NPM[18] (ang. Node Package Manager). Jest to domyślny menadżer pakietów dla języka JavaScript w środowisku Node.js, który powstał w 2010 roku. W jego skład wchodzi CLI (ang. Command Line Interface) oraz zdalne repozytorium zawierające pakiety. Niestety jego wydajność pozostawia wiele do życzenia. Z tego powodu w 2016 roku powstał Yarn[19]. Nie posiada on wprawdzie własnego repozytorium z pakietami tylko korzysta z innych repozytoriów takich jak na przykład NPM. Poprzez odmienny sposób działania zapewnia dużo wydajniejsze instalowanie pakietów. Sam plik typu lockfile zawiera dokładne informacje o wymaganych zależnościach aby ich kod źródłowy mógł zostać pobrany i umieszczony w folderze `node_modules` po wykonaniu komendy `npm install` lub `yarn install`. W czasie wykonywania tych komend jest zapisywana informacja o używanych pakietach w cache w celu szybszego działania gdy ten sam pakiet będzie jeszcze pobierany. Poniżej została przedstawiona tabela porównująca wybrane parametry i czas wykonania dla obydwu menedżerów przy instalowaniu pakietów poleceniami `npm install`

oraz *yarn install* odpowiednio dla NPM oraz Yarn. Przedstawione wartości zostały oparte o rezultatach uzyskanych w artykule autorstwa pnpm[20].

Tabela 3.1: Porównanie szybkości działania menadżerów NPM oraz Yarn

	NPM	Yarn
Brak cache, lockfile i katalogu node_modules	43.2s	43.2s
Brak lockfile i katalogu node_modules	30.4s	27.0s
Brak cache	8.5s	811ms
Brak katalogu node_modules	21.9s	12.9s

3.3 Serwis hostingowy GitHub

3.3.1 System kontroli wersji Git

Git[21] jest systemem kontroli wersji stworzonym przez Linusa Torvaldsa, twórcę systemu Linux, w 2005 roku. W kolejnych latach był on rozwijany przez innych programistów, gdyż jest to projekt otwartoźródłowy. Jego celem jest kontrolowanie zmian w czasie tworzenia oprogramowania. Możliwości tego oprogramowania są dużo szersze, gdyż może on kontrolować dowolne pliki. Jest głównie używany przez deweloperów tworzących oprogramowanie. Można powiedzieć, że podejście zastosowane w systemie Git było przełomowe w zakresie wersjonowania plików. Poprzednie systemy kontroli wersji były mocno zależne od zdalnego serwera. Gdy programista zdecydował się na zatwierdzenie pewnej zmiany w plikach, musiał je od razu przesłać na serwer poprzez wykonanie odpowiednich komend. Z drugiej strony, każde repozytorium Git, które znajdowało czy też znajduje się na komputerze zawiera, w przeciwieństwie do starszych systemów, pełną historię operacji oraz wszystkie funkcjonalności. Co więcej, Git jest w pełni niezależny od zdalnego serwera, repozytorium może istnieć bez zdalnej kopii. Z doświadczenia autora wynika, że system kontroli Git jest bardzo często wybierany w czasie przygotowania i rozwijania nowych projektów informatycznych.

3.3.2 Platforma GitHub

GitHub jest produktem firmy GitHub, Inc. zapewniającej hosting systemu kontroli wersji Git oraz wspiera tworzenie oprogramowania. Powstał on w 2008 roku do tego momentu zgromadził ponad 50 milionów użytkowników. Wszystkie funkcje udostępnione przez rozproszony system kontroli wersji Git są zapewnia a ponadto GitHub zapewnia dodatkowe funkcjonalności takie jak publiczne i prywatne repozytoria, statystyki aktywność, możliwość współpracy z wieloma osobami, dzielenie się kodem źródłowym, GitHub Issues czy Pull Requests oraz mnóstwo innych. Jedynym warunkiem aby mieć do tego dostęp jest założenie darmowego konta. Istnieją również usługi płatne jeszcze bardziej zwiększające możliwości platformy. Co ważne, studenci mają dostęp do płatnych funkcjonalności całkowicie za darmo, wymagane jest jedynie poświadczenie, iż jest się aktualnie studentem danej uczelni. Tak jak zostało już wspomniane, została utworzona gigantyczna społeczność, która tworzy i rozwija projekty mają wpływ na światowe trendy w informatyce i przemyśle z nim związanym. Takim właśnie projektem jest .NET Core wspomniany w sekcji 3.1. Z racji na bardzo dużą popularność portalu GitHub oraz jego bardzo użyteczne funkcje kod źródłowy aplikacji *Social Racetrack* jest właśnie tam przechowywany.

3.4 Framework Bootstrap

Razem z rozwojem urządzeń mobilnych ze zróżnicowaną rozdzielczością oraz proporcjami ekranu pojawił się problem z tworzeniem stron internetowych, które wyglądałyby dobrze zarówno gdy korzysta się z komputera z relatywnie dużym wyświetlaczem jak i z telefonu z małym ekranem. W poprzedniej dekadzie tego stulecia częstym rozwiązaniem było tworzenie mobilnych wersji stron internetowych. Idea ta miała wiele wad, takich jak potrzeba tworzenia danej witryny całkowicie od nowa co drastycznie zwiększało koszty danego przedsięwzięcia. Co więcej, same wersje mobilne nie zawierały wszystkich funkcjonalności danej strony a często kluczowe funkcje były pomijane gdyż ich stworzenie mogłyby wygenerować znaczące koszty. Rozwiązaniem okazał się stworzony w 2011 roku przez programistów portalu Twitter, framework Bootstrap[22]. Bazuje on na technologiach CSS3 oraz HTML5. Zapewnia on bardzo łatwe użycie tych technologii, dzięki

czemu nawet osoba bez dużego doświadczenia jest w stanie szybko nauczyć się tworzenia witryn internetowych stosując ideę RWD (ang. Responsive Web Design).

3.4.1 RWD

Responsive web design został stworzony w celu uzyskania dobrego wyglądu stron internetowych na wszystkich urządzeniach. Jego zadaniem jest automatyczne dostosowywanie rozłożenia komponentów na stronie w zależności od wysokości oraz szerokości ekranu danego urządzenia. Wcześniej wspomniane zmienne rozłożenie komponentów jest często realizowane poprzez zmianę wymiarów danego elementu, zmiana stanu na niewidoczny lub zwiększanie i zmniejszanie wybranych komponentów w celu uzyskaniażądanego efektu spełniającego aktualne standardy projektowania warstwy widoku stron internetowych. Realizacja tego konceptu może być zaimplementowana poprzez zastosowanie Media Queries w języku CSS (ang. Cascading Style Sheets). W przykładzie przedstawiony poniżej, elementy stylizowane przy pomocy klas *.left*, *.main*, *.right* do szerokości ekranu 800px będą zajmowały 100% dostępne miejsca.

```
1 @media screen and (max-width: 800px) {  
2   .left, .main, .right {  
3     width: 100%;  
4   }  
5 }
```

Listing 3.1: Przykład użycia Media Queries do tworzenia RWD

Drugim rozwiązaniem jest skorzystanie z responsywnej siatki Bootstrap, która zapewnia stosowne style zmieniające pozycję komponentów w zależności od wymiarów ekranu.

3.4.2 Responsywna siatka

Jedną z kluczowych elementów wchodzących w skład technologii Bootstrap jest siatka (ang. grid). W celu zapewnienia maksimum możliwości wykorzystuje ona kontenery, rzędy oraz kolumny. Ma to na celu układanie i wyrównywanie zawartości strony internetowej. Jest to możliwe dzięki zastosowaniu technologii występującej w CSS o nazwie flexbox. Sama siatka składa się z 12 kolumn. Liczba 12 nie została wybrana przypadkowo

gdyż zawiera stosunkowo dużo dzielników, dzięki czemu możliwy jest bardzo dokładny podział strony zgodnie z wymaganiami założonymi w projekcie. Aby móc korzystać z tej siatki musi zostać zachowana ściśle określona hierarchia klas. Na samej górze musi znajdować się komponent stylizowany poprzez klasę *container* lub *container-fluid*, następnie musi zostać użyta klasa *row* w celu wyznaczeniu liczby rzędów a na samym końcu musi zostać określona liczba zajmowanych przez element kolumn oraz wartość w pikselach od jakiej obowiązuje. Jest to uzyskiwane przy pomocy klasy na przykład *col-sm-6*. Przykładowa hierarchia klas na podstawie oficjalnej dokumentacji technologii została przedstawiona poniżej. Zapewnia ona, że od szerokości ekranu 576px element o zawartości *first column* będzie zajmował 2 z 12 kolumn, element o zawartości *second column* będzie zajmował 3 z 12 kolumn natomiast element o zawartości *third column* będzie zajmował 7 z 12 kolumn

```
1 <div class="container">
2   <div class="row">
3     <div class="col-sm-2">
4       first column
5     </div>
6     <div class="col-sm-3">
7       second column
8     </div>
9     <div class="col-sm-7">
10      third column
11    </div>
12  </div>
13 </div>
```

Listing 3.2: Hierarchia klas elementów

Bardzo ważny jest podział klas ze względu na szerokość ekranu w pikselach, poniżej zostało to przedstawione w formie tabeli:

Tabela 3.2: Minimalne szerokości ekranu oraz nazwy odpowiadających im klas

Rodzaj ekranu	Minimalna szerokość	Nazwa klasy
Bardzo mały	Brak	<i>.col</i>
Mały	576px	<i>.col-sm</i>
Średni	768px	<i>.col-md</i>
Duży	992px	<i>.col-lg</i>
Bardzo duży	1200px	<i>.col-xl</i>

3.4.3 Komponenty stylizowane

Plik CSS wydawany przez autorów technologii Bootstrap nie zawiera tylko klas do tworzenia responsywnych witryn lecz zapewnia zdecydowanie więcej możliwości. Korzystając z przygotowanych klas w łatwy i szybki sposób można dokonać stylizacji przycisków, formularzy czy nawet głównych pasków nawigacyjnych (ang. Navigation bar).

3.4.4 Projekty rozszerzające funkcjonalności Bootstrap

Ze względu na to, że Framework ten jest udostępniany jako oprogramowanie otwartoźródłowe, możliwe jest rozwijanie przez innych autorów. Jednym z nich jest zespół MDBBootstrap tworzący projekt o tej samej nazwie[23], który jest rozwinięciem technologii Bootstrap poprzez dodanie *Material design*, który został opisany w sekcji 3.4.4.1. Zespół ten zapewnia znakomitą większość komponentów dostępnych w Bootstrap ze zmienionym wyglądem oraz dużą pulę nowych komponentów, które są płatne. Wszystkie te komponenty są dostępne do użycia w technologiach React[6], Angular[24], Vue[25] oraz jQuery[26] co zapewnia możliwość korzystania z nich bez względu na technologię jaka została użyta do stworzenia projektu.

3.4.4.1 Material design

Material Design[27] jest to styl wyglądu stworzony przez Google w 2014 roku. Jest on używany we flagowych produktach tej firmy takich jak Gmail, YouTube czy też Google Drive. Głównymi zasadami tego wzornictwa są:

- Elementy takie jak przyciski powinny sprawiać wrażenie rzeczywistych, co uzyskuje się poprzez korzystanie z cieni aby podkreślić zapewnić wrażenie trójwymiarowości
- Kolorystyka powinna być barwna i odważna oraz tworzyć przejrzyste wrażenia na przykład poprzez negatywną przestrzeń. Same elementy powinny być duże i zdecydowane
- Animacja i ruch to podstawa, dzięki nim możemy przekazywać użytkownikom informacje zwrotne i nakierowywać ich na ustalone akcje w celu ułatwienia obsługi.

Projekt tego wyglądu, ze względu na swój sukces rynkowy jest rozwijany i wykorzystywany w coraz to większej liczbie projektów informatycznych w, których ważne są odczucia użytkownika (ang. User Experience).

3.5 Biblioteka do tworzenia interfejsu użytkownika React.js

Koncept aplikacji typu SPA (ang. Single Page Application) został zapoczątkowany tuż po rozpoczęciu nowego tysiąclecia. Polega on na tym, że dana aplikacja ma tylko jeden plik o rozszerzeniu HTML i wszystkie niezbędne pliki CSS oraz JavaScript są pobierane przy pierwszym załadowaniu strony natomiast zmiany widoczne przez użytkownika są jedynie przeładowywaniem komponentów zamiast ładowania całych stron od nowa. Celem konceptu jest zapewnienie płynniejszego działania oraz odwzorowanie odczuć użytkownika jak przy korzystania z aplikacji desktopowych.

3.5.1 Technologie wspierające koncept SPA

Jedną z pierwszych technologii implementujących koncept SPA był AngularJS[24]. Powstał on w 2009 roku i jego twórcą była firma Google. W podobnym okresie czasu zostało wydane środowisko uruchomieniowe Node.js, którego opis został umieszczony w sekcji 3.2. Od tego czasu z roku na rok rozwój języka programowania JavaScript bardzo wzrastał. W 2013 roku powstała biblioteka React.js[6] a rok później został zaprezentowany framework Vue.js[25]. Wymienione powyżej technologie są aktualnie jednymi z

najbardziej popularnych i najczęściej używanych zarówno w projektach otwartoźródłowych jak i komercyjnych.

3.5.2 Założenia biblioteki React.js

Głównym założeniem biblioteki React jest ułatwienie tworzenia interaktywnego interfejsu użytkownika z łatwą obsługą stanu aplikacji i wydajnym przeładowywaniem komponentów. Same widoki dzięki tej technologii są tworzone w sposób deklaratywny przez co kod jest bardziej przewidywalny i łatwiejsze jest do usuwania ukrytych błędów. Podział na izolowane komponenty, które zarządzają własnym stanem oraz są zdadne do ponownego użycia jest jednym z głównych fundamentów tej technologii. Co więcej, sama logika komponentów jest tworzona w języku JavaScript zamiast w szablonach a zatem przekazywanie zaawansowanych struktur danych staje się znacząco łatwiejsze. Co więcej, jednym z przełomowych elementów biblioteki React jest wirtualny DOM[28] (ang. Document Object Model), który jest koncepcją programistyczną. Ma ona w zamyśle przechowywanie wirtualnej reprezentacji interfejsu użytkownika w pamięci oraz jej synchronizację z modelem DOM obecnym w przeglądarce. Przykładem biblioteki implementującej tą koncepcję jest ReactDOM[29]. Jest to realizowane poprzez wykorzystanie deklaratywnego interfejsu, który zapewnia automatyczną obsługę zdarzeń, manipulację atrybutami oraz zwalnia z obowiązku ręcznej aktualizacji modelu DOM.

3.5.3 Składnia JSX oraz idea komponentów

Składnia JSX jest jedną ze składowych, dzięki której biblioteka React zyskała bardzo dużą popularność. Jest to rozszerzenie składni języka JavaScript, który łudzaco przypomina HTML[30]. Dzięki temu oprócz możliwości tworzenie zmiennych, stałych, funkcji, klas czy też obiektów istnieje możliwość wstawiania szablonów. Z tego powodu w łatwy sposób możliwe jest wykorzystanie wybranych komponentów w zależności na przykład od działań użytkownika, poniżej został przedstawiony przykład:


```

1 export const conditionalRedirect = (redirectPath, lastLocation) => {
2   if (redirectPath !== "/") {
3     return (<Redirect to={redirectPath}/>);
4   } else {
5     return (<Redirect to={lastLocation}/>);
6   }
7 };

```

Listing 3.3: Wyświetlanie warunkowe z wykorzystaniem składni JSX

Jednym z ważnych założeń przyjętych przez twórców biblioteki React jest to, że zarówno logika biznesowa jak i interfejs użytkownika z jego logiką są powiązane ze sobą i nie powinny być rozdzielane. Zamiast tego wykorzystywany jest koncept podziału na komponenty, które mogą być wielokrotnie wykorzystywane. Sama składnia nie jest obowiązkowa natomiast jest bardzo popularna i wykorzystywana przez społeczność. Poniżej zostały przedstawione dwa analogiczne fragmenty kodu, pierwsza z nich z wykorzystaniem składni JSX natomiast druga bez jej wykorzystania:

```

1 const element = (
2   <h1 className="header">
3     Przykładowy tekst
4   </h1>
5 );

```

Listing 3.4: Przykład komponentu z wykorzystaniem JSX

```

1 const element = React.createElement(
2   'h1',
3   {className: 'header'},
4   'Przykładowy tekst'
5 );

```

Listing 3.5: Przykład komponentu bez wykorzystania JSX

W celu wyświetlenia stworzonych powyżej komponentów należy skorzystać z biblioteki ReactDOM przedstawionej w sekcji 3.5.2. Samo użycie zostało przedstawione poniżej:

```
1 ReactDOM.render(element, document.getElementById('root'));
```

Listing 3.6: Wyświetlanie stworzonych komponentów

Warto dodać, że w większości aplikacji przedstawiony powyżej sposób jest używany tylko raz w celu wyświetlenia głównego komponentu, który zawiera określoną liczbę podkomponentów. Przykład takiego wykorzystania został przedstawiony poniżej:

```
1 ReactDOM.render(<App/>, document.getElementById("root"));
```

Listing 3.7: Wyświetlanie głównego komponentu aplikacji

Składnia ta również pozwala na przekazywania wartości do komponentów przez co możliwa jest ich parametryzacja co zapewnia zwiększenie ich uniwersalności w projekcie. Poniżej zostało przedstawione użycie komponentu z parametrem oraz jego definicja:

```
1 <ErrorMessage title={"Przykładowy tekst"}/>
```

Listing 3.8: Użycie komponentu z parametrem

```
1 <div className="container">
2   <div className="custom-margin-text text-center">
3     <h1 className="font-weight-bold custom-font-size-7">
4       {props.title}
5     </h1>
6   </div>
7 </div>
```

Listing 3.9: Definicja komponentu z parametrem

3.5.4 Rodzaje komponentów

Pisząc oprogramowanie z użyciem biblioteki React korzystamy głównie ze składni języka JavaScript. Skutkuje to tym, że komponenty mogą być tworzone na podstawie klas oraz funkcji. Są one nazywane kolejno komponentami klasowymi oraz komponentami funkcyjnymi. Do czasu powstania React Hooks, które zostały opisane w sekcji

3.5.5 znaczenie komponentów funkcyjnych było marginalne i sprowadzało się tworzenia do tworzenia komponentów o niskim stopniu złożoności, które nie posiadały własnego stanu. Skutkiem czego większość komponentów było tworzone jako komponenty klasowe. Charakteryzują się one tym, że klasa w języku JavaScript dziedziczy po klasie *React.Component*, dzięki czemu są dostępne określone metody cyklu życia (ang. Lifecycle methods) oraz wymagana metoda *render()* odpowiadająca za wyświetlanie wybranych komponentów. Jednymi z najczęściej używanych metod cyklu życia są

1. `componentDidMount()`
2. `componentDidUpdate()`
3. `componentWillUnmount()`

Pierwsza z nich jest uruchamiana w czasie montowania w drzewie DOM i jest uruchamiana tylko raz, dzięki czemu jest to świetne miejsce do pobrania danych na przykład z REST API. Druga z nich jest uruchamiana gdy w danym komponencie zachodzą zmiany w drzewie DOM. Ostatnia z tych metod jest uruchamiana w czasie usuwania komponentu z drzewa DOM. Jest to miejsce, w którym możemy dokonać na przykład oczyszczania po komponencie czy też zamknięcia połączenia.

W technologii React Hooks komponenty funkcyjne zyskały możliwość przechowywania stanu, dzięki czemu mogą zapewniać analogiczne możliwości co komponenty klasowe. Ponadto użycie ich jest znacząco łatwiejsze oraz została zmniejszona nadmiarowość powtarzającego się kodu, które nic nie wnosi (ang. boilerplate code). Poniżej zostały przedstawione dwa komponenty o identycznym działaniu, pierwszy z nich na listingu 3.10 jest to komponentem klasowych natomiast drugi z nich na listingu 3.11 jest to komponentem funkcyjnym. Warto zauważyć jak wiele linii kodu zostało zaoszczędzone dzięki zastosowaniu komponentów opartych na funkcjach, które są zalecane przez twórców biblioteki React. Co więcej warto dodać, że nie ma planów usunięcia komponentów klasowych więc została pozostawiona dowolność wyboru rodzaju komponentu oraz możliwość ich łączenia.

```

1  import React, {Component} from "react";
2  import propTypes from "prop-types";
3  import Avatar from "react-avatar";
4
5  const colorsArray = [
6    "#ffeb3b", "#ffc107", "#ff9800", "#ff5722", "#f44336", "#e91e63",
7    "#9c27b0", "#673ab7",
8  ];
9
10 export class LetterAvatar extends Component {
11   constructor(props) {
12     super(props);
13     this.state = {
14       color: ""
15     };
16   }
17
18   componentDidMount() {
19     this.setState({color: this.getRandomColor()});
20   }
21
22   getRandomNumber(begin, end) {
23     const min = Math.ceil(begin);
24     const max = Math.floor(end);
25     return Math.floor(Math.random() * (max - min + 1)) + min;
26   }
27
28   getRandomColor() {
29     return colorsArray[this.getRandomNumber(0, colorsArray.length)];
30   }
31
32   render() {
33     return (
34       <Avatar color={this.state.color} name={this.props.fullname}
35         size={this.props.fontSize} round={true}
36       />
37     );
38   }
39 }

```

Listing 3.10: Przykład komponentu klasowego

```

1  import React, {useEffect, useState} from "react";
2  import propTypes from "prop-types";
3  import Avatar from "react-avatar";
4
5  export const LetterAvatar = (props) => {
6    const [color, setColor] = useState("");
7    const colorsArray = [
8      "#ffeb3b", "#ffc107", "#ff9800", "#ff5722", "#f44336", "#e91e63",
9      "#9c27b0", "#673ab7",
10   ];
11
12   const getRandomNumber = (begin, end) => {
13     const min = Math.ceil(begin);
14     const max = Math.floor(end);
15     return Math.floor(Math.random() * (max - min + 1)) + min;
16   };
17
18   const getRandomColor = () => {
19     return colorsArray[getRandomNumber(0, colorsArray.length)];
20   };
21
22   useEffect(() => {
23     setColor(getRandomColor());
24   }, []);
25
26   return (
27     <Avatar color={color} name={props.fullname}
28       size={props.fontSize} round={true}
29     />
30   );
31 };

```

Listing 3.11: Przykład komponentu funkcyjnego

3.5.5 React Hooks

Jedną z przełomowych nowości zaprezentowanych w 2019 roku przez twórców technologii React są React Hooks. Są to funkcje, które pozwalają na wydzielanie logiki i korzystanie z niej w wielu komponentach. Istnieje możliwość tworzenia własnych React Hooks oraz korzystania z zapewnionych przez twórców technologii.

Hook *useState()* zapewnia możliwość deklarowania zmiennej stanu, dzięki czemu w przypadku gdy komponent potrzebuje przetrzymywać w sobie stan nie musimy zmieniać

typu komponentu z funkcyjnego na klasowy. Zmienną stanową możemy zainicjalizować poprzez przekazanie wybranej wartości jako parametr do funkcji *useState()*, domyślnie jest to wartość *undefined*. Sama funkcja zwraca zmienną stanową oraz funkcję do jej zmiany. Jest ona wymagana z powodu tego, że zmienianie zawartości zmiennej stanowej poprzez bezpośrednie przypisanie do niej wartości jest zabronione, gdyż może powodować nieprzewidywalne skutki takie jak brak przeładowania komponentu. Poniżej został przedstawiony przykład dla zmiennej stanowej *color*:

```
1 const [color, setColor] = useState("");
```

Listing 3.12: Przykład użycia *useState()*

Hook *useEffect()* zapewnia funkcjonalności metod cyklu życia. Jako parametr przyjmuje ona funkcję, której zawartość ma zostać wykonana oraz opcjonalny drugi parametr od, którego zależy zachowanie tego hooka. Gdy nie zostanie przekazany drugi parametr, hook zachowuje się jak metoda *componentDidUpdate()*, gdy zostanie przekazana pusta tablica jego zachowanie jest analogiczne do metody *componentDidMount()* natomiast gdy chcemy aby przekazana funkcja została wykonana gdy zmienna stanowa ulegnie aktualizacji wtedy należy przekazać tę zmienną umieszczoną w tablicy. Sprzątanie, czyli analogia do metody *componentWillUnmount()* jest realizowana poprzez słowo kluczowe *return* na końcu *useEffect()*. Wszystkie powyżej wspomniane metody zostały opisane w sekcji 3.5.4. Poniżej zostały przedstawione wszystkie warianty użycia tego hooka:

```
1 useEffect(() => { console.log("componentDidUpdate"); });
2
3 useEffect(() => { console.log("componentDidMount"); }, []);
4
5 useEffect(() => {
6     console.log("Wywołanie podczas aktualizacji wybranej zmiennej stanowej");
7 }, [color]);
8
9 useEffect(() => { return () => { console.log("componentWillUnmount"); }; }, []);
```

Listing 3.13: Przykład użycia *useEffect()*

Hook *useContext()* zapewnia możliwość korzystania z kontekstu aplikacji czy zmiennych lub stałych globalnych dostępnych w określonych komponentach. Poniżej zostało

przedstawione jego użycie:

```
1  const AppContext = React.createContext(null);
2
3  function App() {
4      const context = useContext(AppContext);
5      return (
6          <button>
7              {context.day}
8          </button>
9      );
10 }
11
12 ReactDOM.render(
13     <AppContext.Provider value={{day: "monday"}}>
14         <App/>
15     </AppContext.Provider>,
16     document.getElementById("root")
17 );
```

Listing 3.14: Przykład użycia *useContext()*

W niniejszym podrozdziale zostały przedstawione podstawowe Hooki, natomiast istnieją ich wyspecjalizowane wersje na przykład dla *useState()* istnieje hook *useMemo()* w, którym należy przechowywać dane wymagające dużego nakładu obliczeniowego komputera.

3.6 Plaforma chmurowa Firebase

Firebase[7] jest projektem zapoczątkowanym w 2011 roku przez startup założony przez Jamesa Tamplina oraz Andrew Lee. Trzy lata później, w roku 2014 został on odkupiony przez firmę Google, Inc. Firebase jest platformą chmurową typu SaaS (ang. Software as a Service). Skutkuje to tym, że zapewnia ona kompleksowy pakiet oprogramowania w chmurze. W przeciwieństwie do IaaS (ang. Infrastructure as a Service) czy też PaaS (ang. Platform as a Service) zazwyczaj spełnia potrzeby użytkownika końcowego a co za tym idzie ułatwia korzystanie, zmniejsza potrzebną ilość wiedzy technicznej do zarządzania systemem co skutkuje przyspieszeniem rozwoju aplikacji oraz zmniejszeniem jego kosztów poprzez brak potrzeby zatrudniania specjalistów z danych dziedzin.

Sama platforma Firebase składa się z kilkunastu produktów. Najpopularniejsze i najbardziej przełomowe zostały przedstawione poniżej w formie poszczególnych rozdziałów. Produkty te w łatwy sposób można kontrolować poprzez narzędzie Firebase Console. Zapewnia ono możliwości zarządzania danym projektem.

3.6.1 Authentication

Authentication jest modułem zapewniającym funkcje rejestracji, logowania, przywracania hasła oraz określania różnych stopni dostępu co w bardziej rozbudowanych systemach jest niesamowicie potrzebne. Samo logowanie może odbywać się poprzez login i hasło oraz przez przesyłanie na dany adres email linku autentykacyjnego (ang. email link Authentication). Dodatkowo jest zapewnione logowanie poprzez popularne portale, dzięki temu użytkownik posiadający konto na danym portalu może zalogować się podając dane z wcześniej wspomnianego portalu. Odbywa się to oczywiście przez przekierowanie na wybrany portal w celu zapewnienia maksymalnego bezpieczeństwa. Poniżej została przedstawiona lista oferowanych portali:

- Google
- Facebook
- Apple
- Twitter
- GitHub
- Microsoft
- Yahoo

Możliwe jest też logowanie przez numer telefonu. Odbywa się to poprzez przesłanie na wskazany adres wiadomości SMS z jednorazowym kodem do logowania.

Sama konsola administratora zapewnia możliwości:

- Ręcznego dodawanie konta użytkowników

- Resetowania hasła
- Wyłączenia konta
- Usuwania konta

3.6.2 Cloud Firestore

Cloud Firestore jest nierelacyjną bazą danych (ang. NoSQL), która jest hostowana w chmurze Firebase (ang. cloud-hosted). Na wiele popularnych technologii takich jak Node.js, Java oraz Python platform jest zapewnione SDK (ang. Software Development Kit) oraz komunikacja poprzez REST[31]. Zgodnie z regułami obowiązującymi w nierelacyjnych bazach danych, przechowywanie danych ma miejsce w dokumentach zawierających pola mapowane na wartości. Same dokumenty są przechowywane w kolekcjach, na których mogą być tworzone zapytania. Typy dokumentów mogą być bardzo zróżnicowane zaczynając od łańcuchów znaków a kończąc na zagnieżdżonych obiektach. Istnieje możliwość tworzenia podkolekcji w dokumentach a co za tym idzie budowania struktur danych o charakterze hierarchicznym. Zapytania tworzone w Cloud Firestore zapewniają możliwość płytkich zapytań. Skutkuje to brakiem pobierania całej kolekcji lub zagnieżdżonych podkolekcji przez co są wydajne i elastyczne. Zapewniona jest możliwość filtrowania, sortowania, ograniczania liczby zwracanych dokumentów oraz podział uzyskanych danych na strony. Bardzo przydatną funkcjonalnością są zasady dostępu do danych (ang. Security Rules), dzięki nim wybrane dane mogą być dostępne dla określonej grupy użytkowników. Przykład takich zasad został przedstawiony poniżej:

```
1 rules_version = '2';
2 service cloud.firestore {
3   match /databases/{database}/documents {
4     match /events/{event} {
5       allow read, create, update: if request.auth != null;
6       allow delete: if request.auth.token.admin == true;
7     }
8   }
9 }
```

Listing 3.15: Zasady dostępu do kolekcji w Cloud Firestore

Samo użycie wcześniej wspomnianego SDK jest wyjątkowo proste a jednocześnie daje ogromne możliwości. Kolejno na listingach 3.16 3.17 3.18 3.19 zostały przedstawione operacje zapisu, odczytu, aktualizacji oraz usuwania. Przykłady te zostały przeniesione bezpośrednio z oficjalnej dokumentacji technicznej Cloud Firestore[7].

```
1  const data = {
2    stringExample: 'Hello, World!',
3    booleanExample: true,
4    numberExample: 3.14159265,
5    dateExample: admin.firestore.Timestamp.fromDate(new Date('December 10, 1815')),
6    arrayExample: [5, true, 'hello'],
7    nullExample: null,
8    objectExample: {
9      a: 5,
10     b: true
11   }
12 };
13
14 const res = await db.collection('data').doc('one').set(data);
```

Listing 3.16: Przykładowy zapis danych do bazy danych

```
1  const cityRef = db.collection('cities').doc('SF');
2  const doc = await cityRef.get();
3  if (!doc.exists) {
4    console.log('No such document!');
5  } else {
6    console.log('Document data:', doc.data());
7  }
```

Listing 3.17: Przykładowy odczyt danych z bazy danych

```
1  const res = await db.collection('cities').doc('DC').delete();
```

Listing 3.18: Przykładowa aktualizacja danych w bazie danych

```
1  const res = await db.collection('cities').doc('DC').delete();
```

Listing 3.19: Przykładowe usuwanie danych z bazy danych

3.6.3 Storage

Storage jest kolejnym produktem wchodzącym w skład platformy Firebase. Jego zadaniem jest zapewnienie możliwości przechowywania plików binarnych takich jak zdjęcia czy też filmy video. Co więcej, same dane są przechowywane bezpośrednio w zasobniku Google Cloud Storage. Rozwiązanie to zapewnia możliwość przechowywania obiektów z dużą dostępnością oraz globalną dostępnością aż do skali eksabajta. Tak jak w przypadku Cloud Firestore opisanej w sekcji 3.6.2 zapewnione jest SDK dzięki, któremu możliwe jest szybkie i łatwe dodawanie, pobierania oraz usuwanie plików. Kolejną zaletą jest funkcjonalność kontroli pobierania i dodawania plików, dzięki której przy problemach z dostępnością sieci przesyłanie zostanie automatycznie przerwane przy utracie połączenia. Zostanie wznowione tuż po ponownym uzyskaniu dostępu do sieci. Co ważne, każdy plik posiada unikalną ścieżkę dostępu, którą można przechowywać w bazie danych Cloud Firestore w celu zapewniania na przykład zdjęcia profilowego dla użytkowników systemu informatycznego. Analogicznie jak w przypadku bazy danych dostępnej w platformie Firebase, zapewnione zostały reguły dostępowe, dzięki którym jest możliwość określenia grup użytkowników, dla których dane treści są dostępne. Przykład takich zasad został przedstawiony na listingu 3.20

```
1 service firebase.storage {
2   match /b/{bucket}/o {
3     match /{allPaths=**} {
4       allow read: if true;
5       allow write: if request.auth.token.admin == true
6     }
7   }
8 }
```

Listing 3.20: Zasady dostępu do plików w Storage

3.6.4 Hosting

Firebase Hosting jest to usługa zapewniająca możliwość łatwego udostępnienia statycznej strony czy też aplikacji internetowej. Ciekawą nowością jest opcja tworzenia i rozmieszczania mikroserwisów (ang. microservices). Bardzo dużą zaletą jest darmowy certyfikat SSL[32] nie wymagający żadnej konfiguracji, który zapewnia bezpieczeństwo

witryny. Co ważne Firebase Hosting zapewnia ważną i dosyć unikalną funkcjonalność jaką jest optymalizacja do obsługi aplikacji typu SPA (ang. Single Page Application) oraz statycznych stron internetowych. Samo dostarczanie zasobów statycznych takich jak HTML, CSS lub JavaScript odbywa się poprzez globalną sieć CDN rozproszoną po wszystkich głównych lokalizacjach na świecie oraz przy pomocy pamięci masowej SSD. Jedną z kluczowych spraw jest domena, która zapewnia ułatwienie dla człowieka poprzez zapewnienie nazwy zamiast konieczności zapamiętywania adresu IP. Sam produkt zapewnia domyślną domenę natomiast możliwe jest też ustawienie własnej nazwy domenowej. Warto dodać, że szczególnie w aplikacjach biznesowych chwytliwa nazwa domeny może znacznie zwiększyć ruch sieciowy na danej stronie co najprawdopodobniej będzie skutkować zwiększeniem zysków.

3.6.5 Functions

Cloud Functions daje możliwość tworzenia funkcji uruchamianych bezpośrednio w chmurze Firebase, zapewniając automatycznie działający kodu po stronie serwera. Dane funkcje mogą być wywoływane poprzez żądanie HTTPS. Same funkcje mogą być tworzone zarówno w języku programowania JavaScript jak i TypeScript. W planach jest również rozszerzenie o możliwość tworzenia wcześniej wspomnianych funkcji w języku Python. W przypadku rozwoju aplikacji i zwiększenia mocy obliczeniowej oczywiście nie jest to problemem. Sposób tworzenia funkcji jak i jej udostępniania nie przysparza problemów gdyż jest bardzo intuicyjny a sama dokumentacja techniczna jest niesamowicie pomocna i bardzo zrozumiała. Na listingu 3.21 został przedstawiony przykład funkcji, która jest odpowiedzialna za przyznawanie uprawnień administratora z wykorzystaniem Firebase Custom Claims. Sam kod został podzielony na dwie części, pierwsza z nich jest funkcja wywoływana poprzez żądanie HTTPS a w niej są sprawdzane stosowne warunki logiczne i jeżeli są one zgodne to wywoływana jest funkcja, która odpowiada za przyznanie wcześniej wspomnianych uprawnień. Sam kod jest czytelny, prosty do zrozumienia oraz nie wymaga dużej ilości czasu aby go napisać i przetestować.

```

1  exports.grantAdmin = functions.https.onCall((data, context) => {
2      if (context.auth?.token.admin !== true) {
3          return {error: "Request Not Authorized - Only admin can call this function"}
4      }
5
6      const email = data.email;
7
8      return grantAdminRole(email).then(() => {
9          return {result: `Request succeeded - ${email} is an admin`}
10     });
11 });
12
13 async function grantAdminRole(email: string): Promise<void> {
14     const user = await admin.auth().getUserByEmail(email);
15
16     if (user.customClaims && (user.customClaims as any).admin === true) {
17         return;
18     }
19
20     return admin.auth().setCustomUserClaims(user.uid, {admin: true});
21 }

```

Listing 3.21: Przykład funkcji zapewniającej przyznawanie uprawnień administratora

Rozdział 4

Dokumentacja techniczna aplikacji

Social Racetrack

Niniejszy rozdział zawiera dokumentację techniczną aplikacji. Jej celem jest przedstawienie wszelkich niezbędnych informacji o produkcie, dzięki czemu osoba rozpoczynająca rozbudowę lub modyfikację projektu ma możliwość poznania wszystkich potrzebnych szczegółów.

4.1 Wymagania funkcjonalne i нефункционалне

4.1.1 Wymagania funkcjonalne

Wymagania funkcjonalne określają, jakie funkcjonalności ma oferować dany system informatyczny, jak ma się zachowywać w pewnych sytuacjach oraz jak ma reagować na charakterystyczne dane wejściowe. Poniżej zostały one przedstawione dla aplikacji *Social Racetrack*.

Wyświetlanie głównej strony

Strona główna zapewnia użytkownikowi możliwość łatwego dostępu do paneli logowania oraz rejestracji użytkownika. Co więcej, zapewniona jest informacja o samym portalu aby zachęcić potencjalnych użytkowników do założenia konta.

Zmiana barw aplikacji

Użytkownik ma zapewnioną możliwość zmiany barw aplikacji na jasną lub ciemną.

Otwarcie bocznego panelu nawigacyjnego

Użytkownik ma zapewnioną możliwość otwarcia bocznego panelu nawigacyjnego dzięki czemu widoczne są pełne nazwy zamiast samych ikon.

Rejestracja użytkownika

Formularz rejestracji użytkownika zapewnia możliwość rejestracji z uprawnieniami zwykłego użytkownika. Minimalny wiek posiadacza konta to 15 lat.

Logowanie użytkownika

Formularz logowania zapewnia użytkownikowi możliwość weryfikacji tożsamości, dostępu do konta oraz do funkcjonalności dostępnych dla zalogowanych użytkowników, które zostały opisane poniżej.

Przywracanie hasła użytkownika

Formularz przywracania hasła zapewnia użytkownikowi anulowanie starego hasła i utworzenie nowego. Opcja ta jest dostępna tylko dla użytkowników posiadających konto w aplikacji oraz mających dostęp do poczty email przypisanej do konta, gdyż na tą pocztę jest wysyłany link do opisanej powyżej czynności.

Panel obiektów sportowych

Panel obiektów sportowych jest dostępny dla użytkownika niezalogowanego, zalogowanego oraz administratora. Użytkownik ten ma dostęp do wszystkich aktualnie dostępnych obiektów sportowych w aplikacji oraz może je wyszukiwać po nazwie.

Dodawanie obiektu sportowego

Tylko administrator ma dostępną możliwość dodawania nowego obiektu sportowego.

Podgląd szczegółów obiektu sportowego

Podgląd szczegółów obiektu sportowego jest dostępny dla użytkownika niezalogowanego, zalogowanego oraz administratora. Panel ten zapewnia możliwość wyświetlenia użytkownikowi dodatkowych informacji, do których należy opis, liczba zakrętów, maksymalna głośność wydechu w pojeździe oraz minimalny prześwit zawieszenia.

Usuwanie obiektu sportowego

Tylko administrator ma dostępną możliwość usuwania istniejącego obiektu sportowego.

Panel konta użytkownika

Panel konta użytkownika jest dostępny dla danej zalogowanej osoby lub administratora. Zapewnia on użytkownikowi możliwość sprawdzenia takich informacji jak imię, na-

zwisko, data ostatniego logowania, adres email do tego konta. Użytkownik może obejrzeć flotę pojazdów oraz zdobyte osiągnięcia sportowe. Dostępne jest tam również przycisk, dzięki któremu można przejść do panelu ustawień konta użytkownika.

Usuwanie wybranych pojazdów z floty

Użytkownik może usunąć ze swojego profilu wybrane przez siebie pojazdy z floty.

Usuwanie wybranych osiągnięć sportowych

Użytkownik może usunąć ze swojego profilu wybrane przez siebie osiągnięcia sportowe.

Panel ustawień konta użytkownika

Panel ustawień konta użytkownika jest dostępny dla danej zalogowanej osoby lub administratora. Zapewnia on trzy możliwości przedstawione poniżej.

Dodawanie nowych pojazdów do floty

Użytkownik zalogowany na swoje konto może dodać nowy pojazd do swojej floty.

Dodawanie nowych osiągnięć sportowych

Użytkownik zalogowany na swoje konto może dodać nowe osiągnięcia sportowe.

Edycja danych osobowych użytkownika

Użytkownik zalogowany na swoje konto może edytować swoje dane osobowe.

Usuwanie konta użytkownika

Usuwanie konta użytkownika w aplikacji jest niemożliwe.

Tworzenie nowego wydarzenia

Użytkownik tworzący dane wydarzenie jednocześnie bierze w nim udział. Nie ma możliwości stworzenia wydarzenia i rezygnacji z udziału w nim. Tylko administrator ma możliwość usunięcia wydarzenia. Możliwe jest utworzenie wydarzenia, którego data jest wcześniejsza niż data utworzenia wydarzenia. Skutkuje to automatycznym jego przeniesieniem do wydarzeń przeszłych.

Przeglądanie przyszłych wydarzeń

Zapewniona jest możliwość przeglądania przyszłych wydarzeń tylko dla użytkowników zalogowanych lub ze statusem administratora. Dodatkowo użytkownik może je wyszukiwać po nazwie. Gdy do rozpoczęcia wydarzenia zostało mniej niż 24 godziny zostanie ono przeniesione do panelu przeszłych wydarzeń.

Przeglądanie przeszłych wydarzeń

Użytkownik zalogowany lub z uprawnieniami administratora może przeglądać przeszłe wydarzenia. Analogicznie jak w przypadku przyszłych wydarzeń ma on możliwość wyszukiwania ich po nazwie.

Podgląd szczegółów wydarzenia

Użytkownik zalogowany lub z uprawnieniami administratora ma możliwość przeglądania szczegółów wszystkich dostępnych wydarzeń.

Dołączanie do wydarzenia

Użytkownik, który jest zalogowany lub jest administratorem może wziąć udział w wydarzeniu jeżeli nie jest na liście osób biorących udział.

Rezygnacja z udziału w wydarzeniu

Użytkownik, który jest zalogowany lub jest administratorem może zrezygnować z udziału w wydarzeniu jeżeli dołączył wcześniej do wydarzenia.

Usuwanie wydarzenia

Administrator ma możliwość usunięcia wydarzenia. Skutkuje to jego usunięciem oraz usunięciem wydarzeń, w których brał lub będzie brał dany użytkownik.

Przeglądanie kont innych użytkowników

Zapewniona jest możliwość przeglądania kont innych użytkowników tylko dla użytkowników zalogowanych lub ze statusem administratora. Dodatkowo istnieje możliwość aby użytkownik wyszukiwał konta innych użytkowników aplikacji po nazwie.

Przeglądanie szczegółów kont innych użytkowników

Użytkownik zalogowany lub z uprawnieniami administratora ma możliwość przeglądania szczegółów wybranego przez niego konta użytkownika.

Nadawanie uprawnień administratora

Tylko administrator może nadać uprawnienia administracyjne. Wymagane jest aby osoba ta posiadała konta w aplikacji oraz podała adres email, który został podany w czasie procesu rejestracji. Ten adres email jest również wyświetlanych w panelu konta użytkownika.

4.1.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają ograniczenia przy jakich dany system informatyczny ma spełniać swoje funkcje. Poniżej zostały one przedstawione dla aplikacji *Social*

Racetrack.

Wysoki stopień bezpieczeństwa

Zapewnienie wysokiego poziomu bezpieczeństwa w celu zapewnienia poufności danych użytkownika oraz ograniczenie możliwości podszywania w celu wykradnięcia danych lub ich zmiany.

Łatwość użycia

Umożliwienie prostego korzystania z aplikacji, intuicyjnej nauki dla osób nie korzystających z niej wcześniej oraz schludny i estetyczny wygląd.

Wysoka wydajność

Aplikacja powinna działać tak samo dobrze na różnych rodzajach urządzeń o zróżnicowanej wydajności. W przypadku urządzeń zaawansowanych technologicznie powinna ona zużywać minimalne ilości zasobów obliczeniowych.

Modularna architektura aplikacji

Podział aplikacji na odseparowane moduły, które nie są zależne od siebie. Istnieje możliwość podmiany dowolnego z nich.

Kompatybilność z popularnymi technologiami

Aplikacja powinna mieć możliwość działania w technologiach, które są popularne i wykorzystywane przez większość użytkowników sieci Internet.

4.2 Architektura aplikacji

Aplikacja *Social Racetrack* została stworzona w koncepcie monolitu. Cała logika i zachowanie aplikacji jest w jednym miejscu z podziałem na stosowne warstwy. Jest to przeciwieństwo architektury mikroserwisów gdzie aplikacja składa się w wielu małych blocków czyli małych aplikacji o zazwyczaj jednym zadaniu czy też odpowiedzialności. Przykładem takiego zadania jest autentykacja użytkownika. W przypadku architektury mikroserwisów, poszczególne mikroserwisy bardzo często współpracują ze sobą, wymieniając między sobą dane.

Ze względu na dosyć nowatorskie rozwiązanie w platformie Firebase opisanej w sekcji 3.6 została zastosowana architektura monolitu razem ze wzorcem projektowym MVC (ang. Model-View-Controller). Polega on na rozdzieleniu aplikacji na trzy warstwy *modelu*, *widoku* oraz *kontrolera*, dzięki czemu możliwa jest podmiana komponentu umiesz-

czonego w dowolnej z trzech warstw na inny kompatybilny komponent. Co więcej, baza danych Cloud Firestore, które została opisana w sekcji 3.6.2 oraz jej wykorzystanie w aplikacji zostało przedstawione w sekcji 4.3.1, ma dosyć nowatorskie podejście gdyż oprócz składowania danych, zapewnia również metody CRUD (ang. Create, Read, Update, Delete) do operowania na niej. Z tego powodu w aplikacji po stronie klienta została stworzona warstwa opakowująca wspomniany zestaw metody. Został on dokładnie opisany w sekcji 4.3.2.

Utworzona dodatkowo warstwa po stronie klienta ma za zadanie ułatwienie korzystania z metod operujących na bazie danych oraz przeniesienie na nią część logiki z warstwy widoku. Ma to na celu zmniejszenie ilości kodu i tak już mocno rozbudowanej warstwy widoku. W samej warstwie widoku została użyta biblioteka *React* opisana w sekcji 3.5. Dodatkowo została ona podzielona na komponenty ułożone hierarchicznie, dzięki czemu zmniejszył się stopień powtarzalności kodu źródłowego. Podział ten został przedstawiony w sekcji 4.3.4. Warto dodać, że warstwa widoku jest zgodna z ideą RWD opisaną w sekcji 3.4.1, a jej dokładne użycie w projekcie zostało opisane w sekcji 4.3.5.

4.3 Implementacja

4.3.1 Model danych

Odwzorowaniem realnego świata jest model kolekcji w bazie danych Cloud Firestore opisanej w sekcji 3.6.2. Z tego powodu powstały trzy kolekcje, pierwszą z nich jest reprezentująca obiekty sportowe o nazwie *Racetracks*, kolejną przedstawiającą użytkownika aplikacji jest kolekcja *Members* natomiast ostatnią jest kolekcja *Events*, której celem jest reprezentacja wydarzeń. Same kolekcje zawierać mogą w sobie wiele dokumentów stąd nazwą ich w języku angielskim w liczbie mnogiej. Na rysunku 4.1 znajdują się właściwości, które posiada dokument danej kolekcji. Mając na celu zachowanie spójnego nazewnictwa, nad właściwościami danego dokumentu została użyta nazwa danej kolekcji.

Racetracks	Members	Events
id name country city lengthInMeters turnsNumber maximumExhaustLoudnessInDecibels minimumRideHeightInMillimeters description imageUrl	id firstName lastName birthDate country city email carsArray receivedAwardsArray eventsRefPathArray	id name racetrackRefPath eventCreatorRefPath membersRefPathArray eventDate

Rysunek 4.1: Kolekcje istniejące w bazie danych Cloud Firestore

Warto dodać, że w projekcie zostały przyjęte pewne konwencje w nazewnictwie wcześniej wspomnianych właściwości dokumentu w danej kolekcji. Przedstawione w dokumencie znajdujący się w kolekcji zawiera właściwości *carsArray* oraz *receivedAwardsArray*. Są to tablice zawierające obiekty klas *Car* oraz *Award*, których zawartości strukturalne zostały przedstawione na rysunku 4.2.

Car	Award
id brand model productionYear mileageInKilometers carType engineType enginePowerInHorsepower driveTrainType	id description year

Rysunek 4.2: Zawartości strukturalne klas *Car* oraz *Award*

Kolejną sprawą jest łączenie naturalnie powiązanych ze sobą dokumentów z różnych kolekcji. Zgodnie z zaprezentowanymi w oficjalnej dokumentacji technicznej bazy danych Cloud Firestore technikami zostały stworzone właściwości w dokumentach wybranych kolekcji odpowiadające za przechowywania odniesień do dokumentów z innych kolekcji. W zależności od tego czy przechowują jedno odniesienie czy też wiele posiadają sufiks odpowiednio *RefPath* lub *RefPathArray*. Przykładem zawartość właściwości przechowującej odniesienie *eventCreatorRefPath* do innego dokumentu z innej kolekcji jest przedstawiony ciąg tekstowy dla `/members/j1VVqMXuoNefURvUVjyE530MtFm1`. Odnosi się ona do dokumentu z kolekcji `Members` o ID `j1VVqMXuoNefURvUVjyE530MtFm1`.

Do przechowywania plików binarnych, takich jak zdjęcia jest wykorzystywane Fire-store Storage. Moduł ten został opisany w sekcji 3.6.3. Sama baza danych przechowuje jedynie URL do danych plików i pobranie go w przypadku potrzeby jest wykonywane poprzez Storage SDK, które zostało opisane w sekcji 3.6.3. Zgodnie z przyjętą konwencją właściwość dokumentu w danej kolekcji jest oznaczana poprzez sufiks *Url* i przykładem takiej właściwości jest *imageUrl* w dokumentach kolekcji *Racetracks*.

4.3.2 Warstwa obsługi interfejsu programistycznego aplikacji API

Ze względu na dosyć nowatorskie rozwiązanie obsługi bazy danych poprzez zapewnienie metod znajdujących się w SDK poszczególnych produktów wchodzących w skład platformy Firebase, które zostały opisane w sekcjach 3.6.2, 3.6.3, 3.6.5 w aplikacji została stworzona warstwa logiki, która znajduje się w folderze *logic*. Jej zadaniem jest opakowanie metod oraz funkcji zapewnione przez SDK. Pierwszą podwarstwą jest warstwa ogólna zawierająca wszystkie metody dostępne w SDK natomiast kolejną podwarstwą są skonkretyzowane dla danej kolekcji metody. Ułatwia to użycie, zwiększa czytelność kodu oraz zmniejsza ilość nadmiarowego kodu gdyż nie jest on nigdzie powtórzony a jedynie są wywołania funkcji czy też metod w wielu komponentach biblioteki React, która została opisana w sekcji 3.5. Aby podążać w sposób konsekwentny w nazewnictwie każda taka klasa zawierająca takie metody czy też plik z funkcjami posiada sufiks *Controller* natomiast skonkretyzowane użycie zawiera w nazwie prefiks kolekcji w formie pojedynczej na przykład *Racetrack*. Mało skomplikowanym choć obrazującym w bardzo dobry sposób jest przykład użycia Firebase Storage, *FirebaseStorageController*, który należy do warstwy ogólnej oraz *RacetrackFirebaseStorageController*, który z kolei należy do skonkretyzowanej warstwy. Tak jak zostało to przedstawione na listingu 4.1, jest to opakowanie we własne metody, funkcji zapewnionych przez SDK. Na listingu 4.2 zostały przedstawione konkretne ich użycia uwzględniające założenia logiczne aplikacji oraz przenoszące część odpowiedzialności na te metody a nie bezpośrednio na ich wywołanie. Przykładem takiego zabiegu jest chociażby konkretyzacja ścieżki (ang. path) oraz typu przesyłanych danych. Warto dodać, że ze względu na mnogość dostępnych metod w SDK zostały przedstawione niepełne listingi omówionych klas.

```

1 export class FirebaseStorageController {
2   uploadFile = async (path, data, metadata, errorFunction) => {
3     try { return await config.storage().ref(path).put(data, metadata); }
4     catch (err) { errorFunction(); }
5   };
6
7   downloadFile = async (path, errorFunction) => {
8     try { return await config.storage().ref(path).getDownloadURL(); }
9     catch (err) { errorFunction(); }
10  };
11  //...
12 }

```

Listing 4.1: Kod źródłowy klasy FirebaseStorageController

```

1 export class RacetrackFirebaseStorageController {
2   _firebaseStorageController = new FirebaseStorageController();
3
4   uploadRacetrackImage = async (filename, data, errorFunction) => {
5     await this._firebaseStorageController.uploadFile(
6       PATH_STORAGE_RACETRACK_IMAGE + filename, data, METADATA_IMAGE, errorFunction);
7     return await this._firebaseStorageController
8       .downloadFile(PATH_STORAGE_RACETRACK_IMAGE + filename, errorFunction);
9   };
10  //...
11 }

```

Listing 4.2: Kod źródłowy klasy RacetrackFirebaseStorageController

4.3.3 Struktura stron

Ze względu na to, że aplikacja *Social Racetrack* została stworzona w koncepcji SPA poprzez użycie biblioteki React nie przeładowywuje ona stron. Cały ten koncept i technologia zostały przedstawione w sekcji 3.5. Z pomocą w przypadku potrzeby posiadania wielu stron przychodzi biblioteka *React Router Dom*. Zapewnia ona możliwość ładowania określonych komponentów ze względu na adres URL, co zapewnia symulację podziału aplikacji internetowej na strony. Poniżej zostały przedstawione adresy URL i odpowiadające mu komponenty. Zostały one podzielone ze względu na poziom wymaganych uprawnień aby mieć do nich dostęp.

4.3.3.1 Komponenty dostępne publicznie

<https://domena/login>

Komponent *LoginPage*, którego zadaniem jest wyświetlanie panelu logowania. Został on przedstawiony na rysunku 5.13.

<https://domena/reset-password>

Komponent *ResetPasswordPage*, którego zadaniem jest wyświetlanie panelu przywracania hasła. Został on przedstawiony na rysunku 5.14.

<https://domena/register>

Komponent *RegisterPage*, którego zadaniem jest wyświetlanie panelu rejestracji użytkownika. Został on przedstawiony na rysunku 5.12.

<https://domena/>

Komponent *HomePage*, którego zadaniem jest wyświetlanie głównej strony. Został on przedstawiony na rysunku 5.1.

<https://domena/racetracks>

Komponent *RacetracksPage*, którego zadaniem jest wyświetlanie obiektów sportowych. Został on przedstawiony na rysunku 5.10.

<https://domena/racetrack-details>

Komponent *RacetrackDetailsPage*, którego zadaniem jest wyświetlanie szczegółów obiektów sportowych. Został on przedstawiony na rysunku 5.11.

4.3.3.2 Komponenty dostępne dla zalogowanego użytkownika

<https://domena/account>

Komponent *AccountPage*, którego zadaniem jest wyświetlanie profilu użytkownika. Został on przedstawiony na rysunku 5.15.

<https://domena/account-settings>

Komponent *AccountSettingsPage*, którego zadaniem jest zapewnienie możliwości dodawania pojazdów, osiągnięć sportowych oraz edycja danych osobistych użytkownika. Wszystkie te możliwości zostały przedstawione na rysunkach 5.17, 5.18, 5.19.

<https://domena/create-event>

Komponent *CreateEventPage*, którego zadaniem jest wyświetlanie panelu tworzenia wydarzeń. Został on przedstawiony na rysunku 5.22.

<https://domena/future-events>

Komponent *FutureEventsPage*, którego zadaniem jest wyświetlanie przyszłych wydarzeń. Został on przedstawiony na rysunku 5.20.

<https://domena/past-events>

Komponent *PastEventsPage*, którego zadaniem jest wyświetlanie przeszłych wydarzeń. Jego wygląd jest identyczny w stosunku do komponentu odpowiadającego za wyświetlanie przyszłych wydarzeń z wyjątkiem braku możliwości tworzenia wydarzenia.

<https://domena/event-details>

Komponent *EventDetailsPage*, którego zadaniem jest wyświetlanie szczegółów wybranego wydarzenia. Został on przedstawiony na rysunku 5.21.

<https://domena/members>

Komponent *MembersPage*, którego zadaniem jest wyświetlanie kont użytkowników aplikacji. Został on przedstawiony na rysunku 5.23.

<https://domena/member-details>

Komponent *MemberDetailsPage*, którego zadaniem jest wyświetlanie szczegółów wybranego konta użytkownika aplikacji. Został on przedstawiony na rysunku 5.24.

4.3.3.3 Komponenty dostępne dla administratora

<https://domena/admin-panel>

Komponent *AdminPanelPage*, którego zadaniem jest wyświetlanie panelu administratora. Został on przedstawiony na rysunku 5.27.

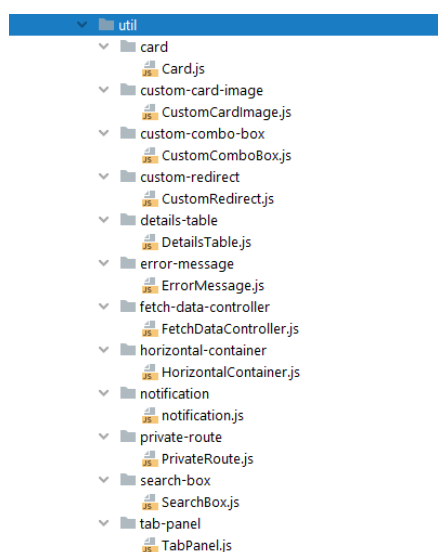
<https://domena/create-racetrack>

Komponent *CreateRacetrackPage*, którego zadaniem jest wyświetlanie panelu dodawania obiektu sportowego. Został on przedstawiony na rysunku 5.30.

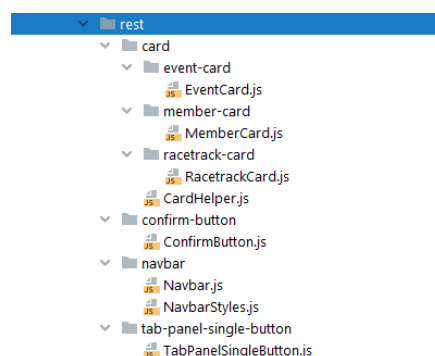
4.3.4 Struktura komponentów i ich użycie

Zgodnie z dobrymi praktykami sugerowanymi przez twórców biblioteki React przedstawionej w sekcji 3.5 aplikacja została podzielona na komponenty wyłącznie funkcyjne z użyciem React Hooks. Sam podział na komponenty został wykonany w zgodzie z zasadą *"Jeden komponent, jedna odpowiedzialność"* a co za tym idzie cała hierarchia użycia od najbardziej bazowych komponentów poprzez konkretyzujące bazowe do określonych

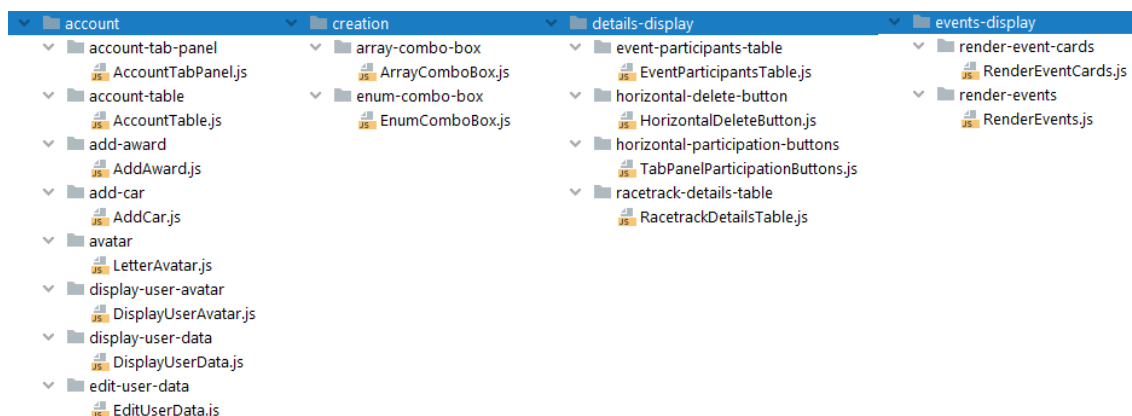
zadań po takie składających się z wielu komponentów niżej w hierarchii. Jedną z najważniejszych zasad przyświecających autorowi było unikanie powtarzania kodu. Bardzo ważne było też tworzenie atomowych komponentów o prostej odpowiedzialności oraz budowanie całości z małych fragmentów w celu uzyskania skalowalnej i łatwej w rozbudowie aplikacji. Struktura hierarchii komponentów bazowych została przedstawiona na rysunku 4.3. Następnie na rysunku 4.5. zostały przedstawione komponenty użyte na wybranych strona, których struktura została przedstawiona w sekcji 4.3.3. Komponenty, które były używane w wielu komponentach stojących wyżej w hierarchii oraz zróżnicowanych odpowiedzialnościach zostały przedstawione na rysunku 4.4.



Rysunek 4.3: Lista komponentów bazowych



Rysunek 4.4: Lista pozostałych komponentów



Rysunek 4.5: Lista komponentów wykorzystywanych na wybranych stronach

4.3.5 Responsywność interfejsu użytkownika

Ze względu na bardzo szeroką gamę urządzeń z różnymi wymiarami i proporcjami ekranów aplikacja *Social Racetrack* została przygotowana z wykorzystaniem idei RWD, technologii Bootstrap oraz projektów rozszerzających jego możliwości, które zostały opisane w sekcji 3.4.

Świetnym przykładem obrazującym jest strona główna aplikacji przedstawiona na rysunku 5.1. Komponent został podzielony na dwie części, lewą oraz prawą. Ich wyświetlanie zostało przedstawione na listingu 4.5 natomiast fragmenty kodu źródłowego lewej i prawej części, odpowiednio funkcjami *renderLeftSide* oraz *renderRightSide*, zostały przedstawione na listingach 4.4, 4.3. Warto dodać, że fragmenty bezpośrednio nie ilustrujące omawianej tematyki w tej sekcji zostały zastąpione poprzez *// ...*. Dokładny opis responsywnej siatki został przedstawiony w sekcji 3.4.2. Sam podział miejsca na stronie został zrealizowany przy wykorzystaniu klasy *col-sm-6*. Oznacza to, że do szerokości ekranu *sm* lewa oraz prawa strona będzie zajmować cały dostępny ekran. Co do samego ułożenia to najpierw będzie wyświetlona lewa część a pod nią prawa część. Powyżej szerokości *sm* będą one zajmować połowę ekranu i będą umieszczone obok siebie. Definicje rozmiarów zostały przedstawione w tabeli 3.2.

Co więcej, rozmiar obrazka oraz opisu umieszczonego nad nim zmienia się wraz z rozmiarem ekranu. Zostało użyte połączenie trzech klas *col-md-12*, *col-lg-10*, *col-xl-8*. Skutkuje to tym, że do szerokość ekranu mniejszej niż *lg* obrazek i opis zajmuje całą dostępną przestrzeń, gdy szerokość ekranu jest z przedziału klas *lg* oraz *xl* to zajmuje 10/12 szerokości ekranu natomiast od szerokości *xl* zajmuje 8/12 szerokości ekranu.

```
1  const renderRightSide = () => {
2    const renderDescription = () => { //... };
3    const renderButtons = () => { //... };
4
5    return ( <div className="col-sm-6 text-center font-weight-bold mb-4">
6      {renderDescription()}
7      {renderButtons()}
8    </div> );
9  };
```

Listing 4.3: Kod komponentu *HomePage* odpowiedzialny za wyświetlanie jej prawej części

```

1  const renderLeftSide = () => {
2      const renderDescription = () => {
3          return ( <div className="container-fluid">
4              <div className="row justify-content-center">
5                  <div className="col-md-12 col-lg-10 col-xl-8 custom-font-size-1 text-justify
6                      font-weight-bold mb-3">
7                      <span className={globalStyles.materialBlueFont}>
8                          {strings.homePage.socialRacetrack + " "}
9                      </span>
10                     <span>
11                         {strings.homePage.siteIntroFirst + " "}
12                     </span>
13                     <span>
14                         {strings.homePage.siteIntroSecond}
15                     </span>
16                 </div>
17             </div> );
18     };
19
20     const renderImage = () => {
21         return ( <div className="container-fluid">
22             <div className="row justify-content-center">
23                 <div className="col-md-12 col-lg-10 col-xl-8">
24                     <Card variant="outlined">
25                         <CardMedia component="img" src={porscheImage}/>
26                     </Card>
27                 </div>
28             </div>
29             </div> );
30     };
31
32     return ( <div className="col-sm-6 text-center align-self-center mb-4">
33         {renderDescription()}
34         {renderImage()}
35     </div> );
36 };

```

Listing 4.4: Kod komponentu HomePage odpowiedzialny za wyświetlanie jej lewej części

```

1   return ( <div className="container-fluid custom-page-small-margin">
2       <div className="row justify-content-center">
3           {renderLeftSide()}
4           {renderRightSide()}
5       </div>
6   </div> );

```

Listing 4.5: Sekcja odpowiedzialna za łączenie lewej i prawej części komponentu
HomePage

4.4 Instalacja

W celu dokonania procesu instalacji należy na lokalnej maszynie posiadać zainstalowane środowisko uruchomieniowe Node.js, które zostało opisane w sekcji 3.2, menadżer pakietów Yarn przedstawiony w sekcji 3.2.3 oraz należy posiadać zaktualizowaną przeglądarkę internetową. Zalecanymi są Chrome[16] lub Firefox[33] ze względu na dużą popularności oraz wsparcie dla nowoczesnych technologii. Kluczowy jest również dostęp do sieci Internet, bez którego instalacja nie zakończy się sukcesem.

Pierwszym krokiem jest uzyskanie dostępu do kodu źródłowego aplikacji i umieszczenie go w wybranej lokalizacji na wcześniej wspomnianej lokalnej maszynie. Kolejnym krokiem jest uruchomienie przeglądarki i uruchomienie strony pod adresem <https://console.firebase.google.com/> . Następnie należy utworzyć lub zalogować się na konto *Google* i w konsoli platformy chmurowej Firebase opisanej w sekcji 3.6 stworzyć nowy projekt poprzez przejście konfiguratora projektu i nadanie mu nazwy. Po wykonaniu tej czynności powinien być dostępny projekt ze wszystkimi darmowymi usługami.

Następnym ważnym krokiem, który należy wykonać na maszynie lokalnej jest otwarcie konsoli i przejście do lokalizacji w której jest umieszczony kod źródłowy oprogramowania. Następnie należy wykonać komendę `yarn install` , później trzeba przejść do katalogu *functions* korzystając z poleceń charakterystycznych dla danego systemu operacyjnego oraz wykonać polecenie `npm install` . Następny z kolei krokiem jest powrót do głównego katalogu w kodzie źródłowym aplikacji i wykonanie komend `npm install -g firebase-tools` oraz `firebase login` odpowiadających kolejno za globalną instalację

pakietu *firebase-tools* oraz za zalogowanie do konta Google korzystając adresu email oraz hasła z wcześniej utworzonego konta. Kolejnym zadaniem jest powrót do otwartej przeglądarki z Firebase Console i przejście do sekcji ustawień a następnie do karty *Ogólne* i przewinięcie strony do jej końca w celu pozyskania kluczy projektu, które należy zgodnie z przedstawioną instrukcją na stronie umieścić w pliku *.env*. Zapewnienie błędnych kluczy lub pomyłka w ich kolejności skutkują niepowodzeniem w instalacji.

Kolejnym krokiem jest określenie sposobu autentykacji użytkowników aplikacji. Należy wybrać z bocznego panelu kartę *Authentication*, przejść do karty *Sign-in method* a następnie aktywować autentykację poprzez *email/hasło*.

Następnie należy uruchomić polecenie, którego zadaniem jest udostępnienie zasad dostępu do bazy danych Cloud Firestore opisanej w sekcji 3.6.2 poprzez polecenie `firebase deploy --only firestore:rules` lub skorzystanie ze skryptu uruchamianego poleceniem `yarn fb:db`.

Następnym krokiem jest udostępnienie zasad dostępu do Firebase Storage opisanego w sekcji 3.6.3. Należy tego dokonać poprzez wykonanie polecenia `firebase deploy --only storage:rules` lub użycie skryptu uruchamianego poleceniem `yarn fb:stor`.

W celu udostępnienia funkcji serwerowych określonych mianem Cloud Functions przedstawionych w sekcji 3.6.5 należy wykonać polecenie `firebase deploy --only functions` lub skorzystać ze skryptu uruchamianego poleceniem `yarn fb:fun`.

Ostatecznym krokiem jest udostępnienie aplikacji w sieci Internet. Aby tego dokonać należy skorzystać z poleceń `yarn build` oraz `firebase deploy --only hosting` lub użyć skryptu uruchamianego poleceniem `yarn fb:host`.

Rozdział 5

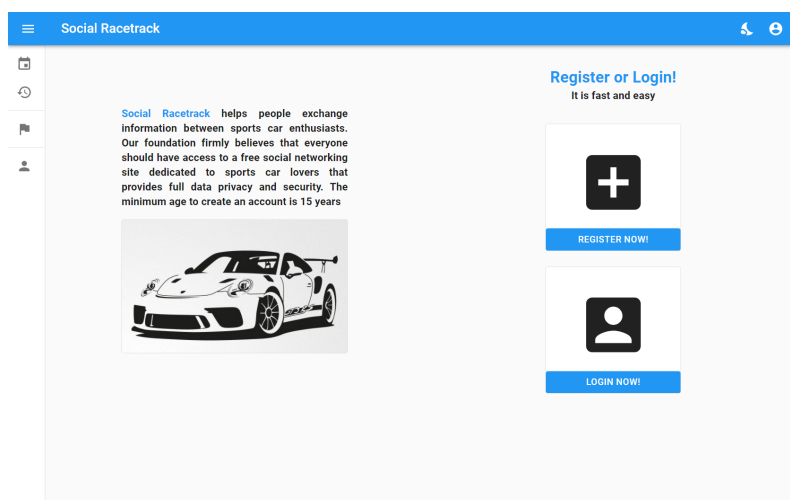
Dokumentacja użytkownika aplikacji

Social Racetrack

Niniejszy rozdział zawiera pełną dokumentację użytkownika aplikacji. Jego celem jest omówienie wszystkich dostępnych funkcji z punktu widzenia osoby korzystającej z aplikacji.

5.1 Wprowadzenie do obsługi interfejsu aplikacji

Interfejs aplikacji w swoim zamyśle był tworzony zgodnie z zasadami ergonomii, prostoty oraz intuicyjnej obsługi. Na rysunku 5.1 został przedstawiony ekran powitalny aplikacji, warto dodać, że jest on wyświetlany gdy użytkownik nie jest zalogowany.



Rysunek 5.1: Ekran powitalny aplikacji

Posiada on opcję zmiany barw aplikacji, dostępne opcje to jasna i ciemna. Pierwsza z nich została przedstawiona na rysunku 5.1. Aby dokonać zmiany należy nacisnąć przycisk w prawym górnym rogu. Na rysunkach 5.2 oraz 5.3 zostały przedstawione dwie różne ikony zależnie w jakim trybie znajduje się aplikacja.



Rysunek 5.2: Przycisk do przełączania
do trybu ciemnego



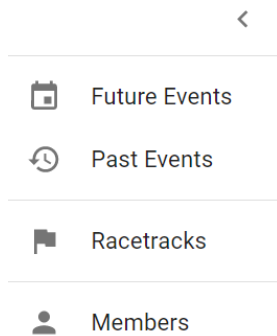
Rysunek 5.3: Przycisk do przełączania
do trybu jasnego

W celu otwarcia bocznego panelu znajdującego się po lewej stronie należy nacisnąć przycisk przedstawiony na rysunku 5.4.



Rysunek 5.4: Przycisk do otwierania bocznego panelu

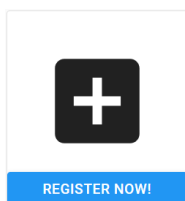
Akcja ta zapewnia otwarcie bocznego panelu i wyświetlenie opisu poszczególnych przycisków. Przycisk *Future Events* odpowiada za wyświetlanie wydarzeń, które odbędą się w przyszłości. Przycisk *Past Events* ma za zadanie wyświetlanie wydarzeń, które już się odbyły. Kolejny przycisk *Racetracks* dokonuje prezentacji dostępnych obiektów sportowych. Ostatni dostępny przycisk *Members* zapewnia możliwość przedstawienia użytkowników aplikacji. W celu schowania bocznego paska należy nacisnąć przycisk przedstawiony na rysunku 5.6.



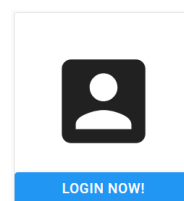
Rysunek 5.5: Boczny panel

Rysunek 5.6: Przycisk do zamykania bocznego panelu

W celu logowania lub rejestracji użytkownika należy skorzystać z przedstawionych na rysunkach 5.7 oraz 5.8 przycisków.



Rysunek 5.7: Przycisk do rejestracji użytkownika



Rysunek 5.8: Przycisk do logowania użytkownika

Inną możliwością jest skorzystania z przycisku konta użytkownika, przedstawionego na rysunku 5.9, który przenosi do okna logowania analogicznie jak przycisk przedstawiony na rysunku 5.8.

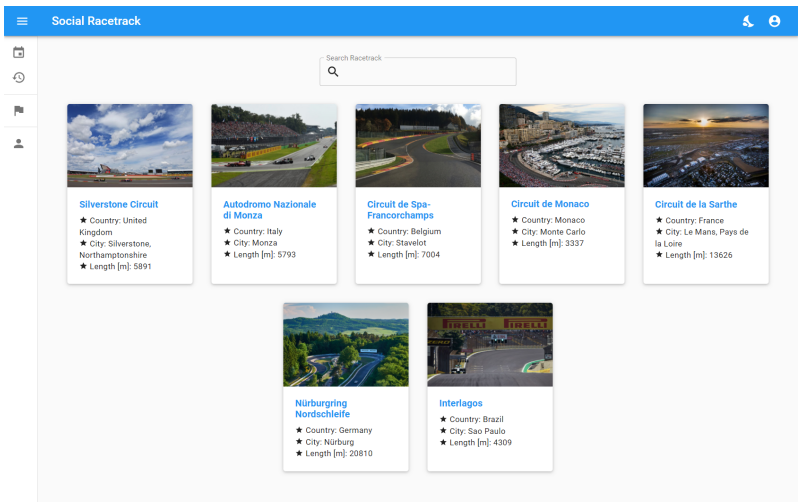


Rysunek 5.9: Przycisk konta użytkownika

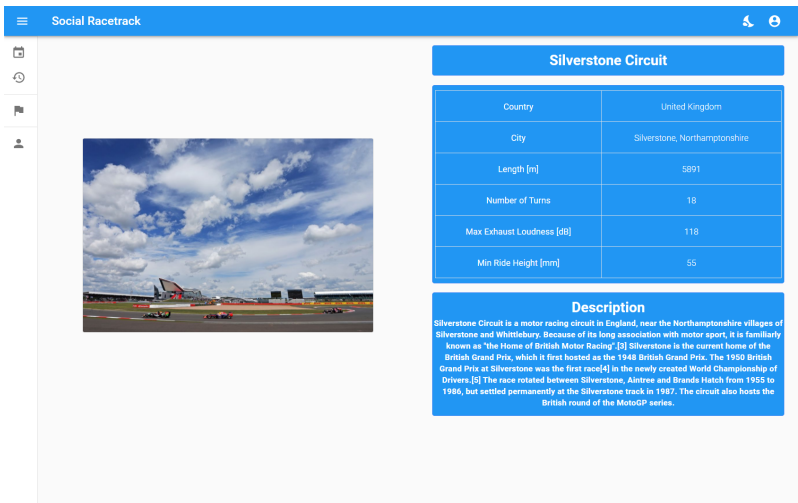
5.2 Obsługa aplikacji dla użytkownika niezalogowanego

Zgodnie z wymaganiami funkcjonalnymi i нефункциональными określonymi w sekcji 4.1 akcje dostępne dla użytkownika niezalogowanego są mocno ograniczone. Dla takiego

użytkownika dostępny jest ekran powitalny przedstawiony na rysunku 5.1 oraz możliwość wyświetlania obiektów sportowych oraz szczegółów na ich temat. Na rysunkach 5.10 oraz 5.11 zostały przedstawione strony wyświetlające wszystkie obiekty sportowe oraz szczegóły na temat tego, który został wybrany przez użytkownika. Aby wyświetlić szczegóły danego obiektu sportowego należy nacisnąć myszą na wybraną kartę z torem. W celu powrotu do ekranu przedstawionego na rysunku 5.10 należy skorzystać z przycisku powrotu w przeglądarce.



Rysunek 5.10: Panel wyświetlający wszystkie obiekty sportowe

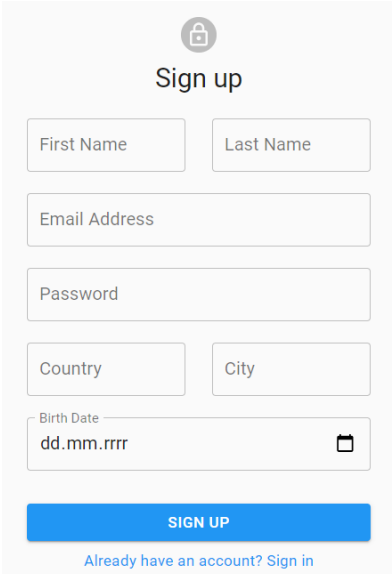


Rysunek 5.11: Panel wyświetlający szczegóły na temat wybranego obiektu sportowego

5.3 Logowanie i rejestracja

Logowanie lub rejestracji odbywa się na specjalnie wydzielonej podstronie, aby uzyskać do niej dostęp należy skorzystać z przycisków przedstawionych na rysunkach 5.7, 5.8 lub 5.9. Opis ich użycia został przedstawiony w sekcji 5.1.

Po wybraniu przycisku do rejestracji użytkownika zostanie wyświetlony panel przedstawiony na rysunku 5.12. Należy go wypełnić zgodnie z opisem każdego z pól w formularzu a następnie nacisnąć przycisk *SIGN UP*. Po wykonaniu tej akcji zostanie przeprowadzone automatyczne zalogowanie na nowo utworzone konto a następnie użytkownik zostanie poinformowany o konieczności wylogowania. Po zakończeniu tych operacji użytkownik może się zalogować do swojego konta i korzystać z jego wszystkich funkcjonalności. W przypadku posiadania już konta w aplikacji i otwarciu okna rejestracji możliwe jest przejście do panelu logowania poprzez naciśnięcie przycisku *Already have an account? Sign in*.

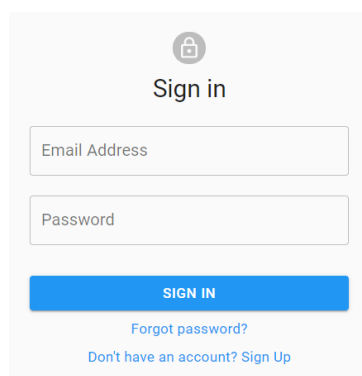
The image shows a 'Sign up' form with a lock icon at the top. The form contains several input fields: 'First Name' and 'Last Name' (side-by-side), 'Email Address', 'Password', 'Country' and 'City' (side-by-side), and 'Birth Date' (with a date format 'dd.mm.rrrr' and a calendar icon). At the bottom, there is a blue 'SIGN UP' button and a link that says 'Already have an account? Sign in'.

Rysunek 5.12: Panel rejestracji użytkownika

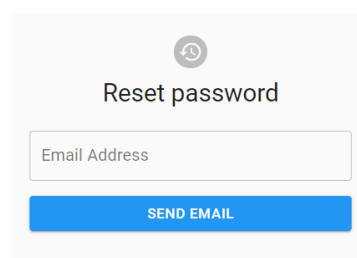
Po wybraniu przycisku do logowania lub przycisku konta użytkownika zostanie wyświetlony panel przedstawiony na rysunku 5.13. Należy go wypełnić zgodnie z opisem każdego z pól w formularzu a następnie nacisnąć przycisk *SIGN IN*. Po tej akcji aplikacja wyświetli panel konta użytkownika. W przypadku nie posiadania konta w aplikacji i otwarcia okna logowania możliwe jest przejście do panelu rejestracji użytkownika po-

przez naciśnięcie przycisku *Don't have an account? Sign Up*.

W panelu logowania istnieje możliwość przywracania hasła, w tym celu należy nacisnąć przycisk *Forgot password?*. Po tej akcji zostanie otwarty panel przywracania hasła przedstawiony na rysunku 5.14, należy go wypełnić zgodnie z opisem przedstawionym w formularzu i nacisnąć przycisk *SEND EMAIL*. Na wskazany adres email zostanie wysłany link do zmiany hasła, należy go otworzyć i postępować zgodnie ze wskazówkami w nim przedstawionymi.

The image shows a 'Sign in' form. At the top is a lock icon and the text 'Sign in'. Below are two input fields: 'Email Address' and 'Password'. A blue button labeled 'SIGN IN' is positioned below the fields. At the bottom, there are two links: 'Forgot password?' and 'Don't have an account? Sign Up'.

Rysunek 5.13: Panel logowania
użytkownika

The image shows a 'Reset password' form. At the top is a clock icon and the text 'Reset password'. Below is an input field for 'Email Address'. A blue button labeled 'SEND EMAIL' is at the bottom.

Rysunek 5.14: Panel przywracania
hasła

5.4 Obsługa funkcjonalności dostępnych dla użytkownika zalogowanego

5.4.1 Obsługa konta użytkownika

Aby mieć możliwość dostępu do strony przedstawionej na rysunku 5.15 należy być zalogowanym i należy nacisnąć przycisk pokazany na rysunku 5.9. Po wykonaniu tej akcji zostanie załadowany panel użytkownika. Składa się on z kilku części ułożonych w postaci poziomych pasów.

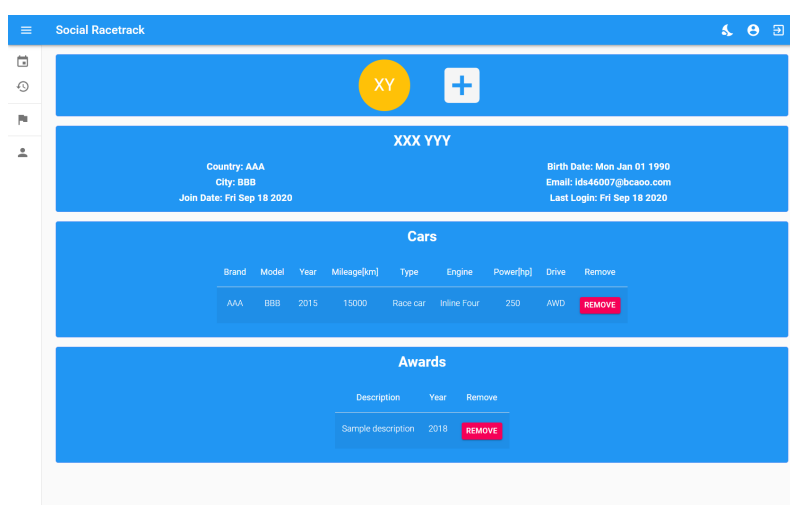
Pierwszy z nich zawiera awatar z inicjałami użytkownika, obok niego jest przycisk, przedstawiony na rysunku 5.16, służący do dodawania pojazdów i osiągnięć oraz edycji danych osobowych. Funkcjonalności te zostały opisane w sekcjach 5.4.1.1, 5.4.1.2 oraz 5.4.1.3, aby uzyskać do nich dostęp należy nacisnąć przycisk pokazany na rysunku 5.16.

Drugi z poziomych pasów zawiera dane osobowe użytkownika takie jak kraj, data

urodzenia czy też email. Trzeci z kolei przedstawia flotę pojazdów danego użytkownika oraz zapewnia opcje usuwania pojazdu, szczegółowo zostało to opisane w sekcji 5.4.1.1.

Trzeci z poziomych pasów przedstawia listę zdobytych nagród przez użytkownika, analogicznie jak w przypadku elementu wyświetlającego flotę pojazdów zapewniony jest przycisk do usuwania, które szczegółowo został opisany w sekcji 5.4.1.2.

Ostatni poziomy pas, nie przedstawiony na rysunku 5.15 jest fragmentem odpowiedzialnym za wyświetlanie wydarzeń, w których brał lub będzie brał udział dany użytkownik. Wyświetlane są karty z wydarzeniami analogicznie jak w panelu przyszłych wydarzeń przedstawionym na rysunku 5.20.



Rysunek 5.15: Panel konta użytkownika



Rysunek 5.16: Przycisk dodawania i edycji panelu użytkownika

5.4.1.1 Dodawanie i usuwanie floty pojazdów

Aby mieć możliwość dodania nowego pojazdu, należy nacisnąć przycisk *ADD CAR* a następnie wypełnić formularz zgodnie z opisami jego pól i zatwierdzić przyciskiem *CONFIRM*. Sam formularz został przedstawiony na rysunku 5.17. W celu usunięcia wybranego pojazdu należy w panelu użytkownika przedstawionego na rysunku 5.15 nacisnąć przycisk *REMOVE* przy wybranym pojeździe.

The screenshot shows the 'Social Racetrack' application with a sidebar menu on the left. The main content area displays the 'ADD CAR' form, which is highlighted. Above the form are three buttons: 'ADD CAR', 'ADD AWARD', and 'EDIT USER DATA'. The form fields are arranged in two columns: Brand and Model, Production Year and Mileage [km], Car Type and Engine Type, and Engine Power [hp] and Drive Train Type. A 'CONFIRM' button is located at the bottom center of the form.

Rysunek 5.17: Panel dodawania floty pojazdów użytkownika

5.4.1.2 Dodawanie i usuwanie zdobytych nagród

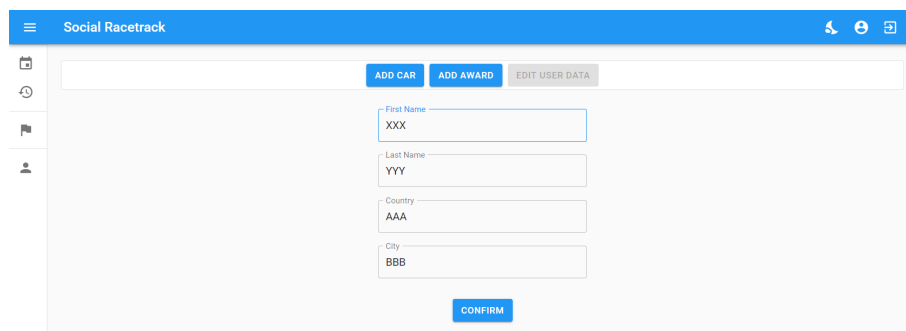
Aby mieć możliwość dodawania nowych osiągnięć, należy nacisnąć przycisk *ADD AWARD* a następnie wypełnić formularz zgodnie z opisami jego pól i zatwierdzić przyciskiem *CONFIRM*. Sam formularz został przedstawiony na rysunku 5.18. W celu usunięcia wybranego osiągnięcia sportowego należy w panelu użytkownika przedstawionego na rysunku 5.15 nacisnąć przycisk *REMOVE* przy wybranym osiągnięciu.

The screenshot shows the 'Social Racetrack' application with the 'ADD AWARD' form highlighted. Above the form are three buttons: 'ADD CAR', 'ADD AWARD', and 'EDIT USER DATA'. The form fields are 'Description' and 'Year'. A 'CONFIRM' button is located at the bottom center of the form.

Rysunek 5.18: Panel dodawania osiągnięć sportowych użytkownika

5.4.1.3 Edycja danych osobowych

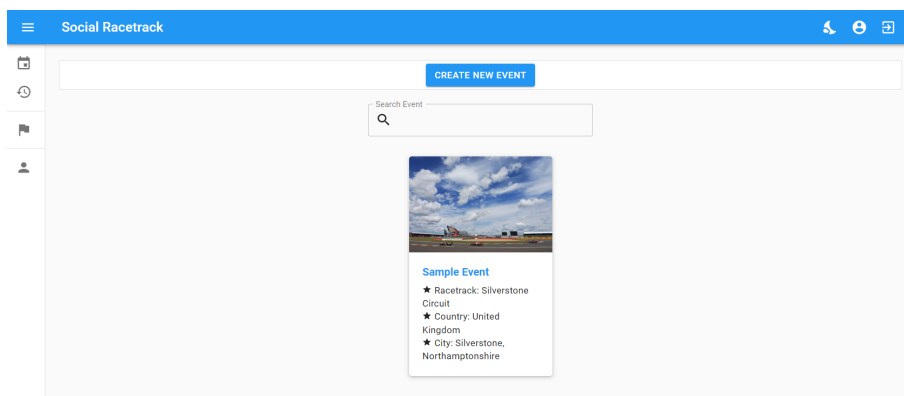
Aby dokonać edycji danych osobowych należy nacisnąć przycisk *EDIT USER DATA* a następnie dokonać wybranych zmian w formularzu i zatwierdzić przyciskiem *CONFIRM*. Sam formularz został przedstawiony na rysunku 5.19.



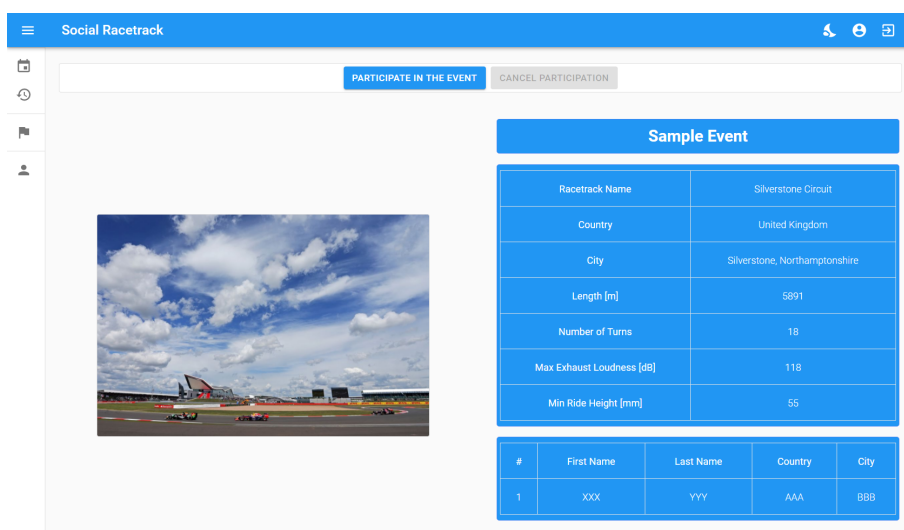
Rysunek 5.19: Panel edycji danych osobowych użytkownika

5.4.2 Przeglądanie, dołączanie do wydarzeń i rezygnacja z udziału

Wydarzenia w aplikacji są podzielone ze względu na datę, kiedy się odbyły lub się odbędą. Aby uzyskać dostęp do przeszłych wydarzeń należy nacisnąć przycisk *Past Events* natomiast dla przyszłych wydarzeń należy nacisnąć przycisk *Future Events*, obydwa przyciski znajdują się w bocznym panelu a ich użycie zostało opisane w sekcji 5.1. Wyświetlanie i przeglądanie jest identyczne dla dwóch rodzajów wydarzeń, mając uruchomiony panel czy to przeszłych czy to przyszłych wydarzeń, są wyświetlane one w formie kart. Panel ten został przedstawiony na rysunku 5.20. Aby uzyskać więcej informacji na jego temat należy nacisnąć myszką na wybrane wydarzenie i zostanie pokazany panel szczegółów wydarzenia, który został przedstawiony na rysunku 5.21. Dołączanie do wydarzenia jest możliwe poprzez naciśnięcie przycisku *PARTICIPATE IN THE EVENT* i jest on dostępny tylko wtedy, gdy dany użytkownik nie bierze w nim udziału. Rezygnacja z udziału odbywa się analogicznie do dołączenia czyli poprzez naciśnięcie przycisku *CANCEL PARTICIPATION*. Jest on dostępny tylko wtedy, gdy dany użytkownik bierze udział w wydarzeniu.



Rysunek 5.20: Panel przyszłych wydarzeń



Rysunek 5.21: Panel szczegółów wydarzenia

5.4.3 Tworzenie wydarzeń

W celu stworzenia wydarzenia należy uruchomić panel przyszłych wydarzeń, który został pokazany na rysunku 5.20. Następnie należy nacisnąć przycisk *CREATE NEW EVENT* i wypełnić formularz zgodnie z opisem przy każdym polu oraz nacisnąć przycisk *CONFIRM*. Formularz został przedstawiony na rysunku 5.22.

Event Name

Racetrack

Date dd.mm.mrr

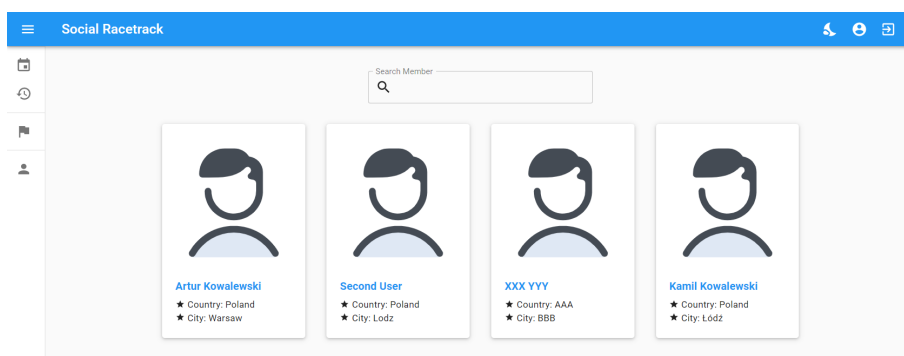
Time --:--

CONFIRM

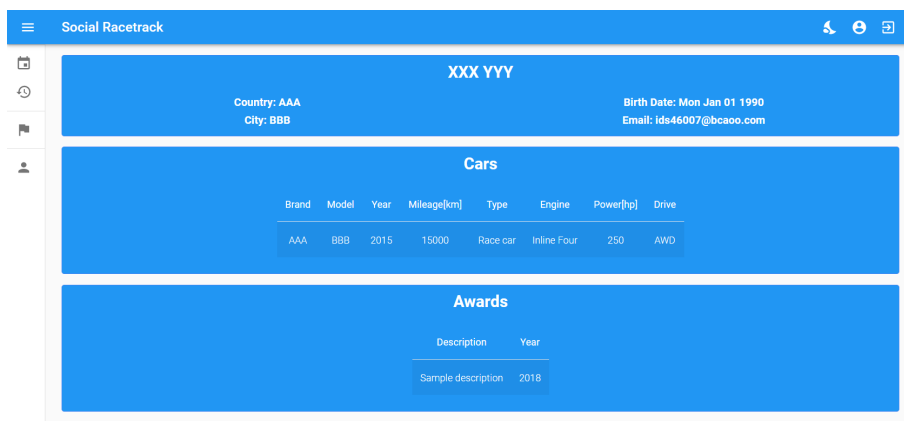
Rysunek 5.22: Panel tworzenia wydarzeń

5.4.4 Przeglądanie kont innych użytkowników

Aby mieć możliwość przeglądania kont innych użytkowników aplikacji należy nacisnąć przycisk *Members* znajdujący się w bocznym panelu a jego użycie zostało opisane w sekcji 5.1. Konta użytkowników są wyświetlane w formie kart w panelu użytkowników aplikacji, który został przedstawiony na rysunku 5.23. Aby uzyskać więcej informacji na jego temat należy nacisnąć myszką na wybrane konto użytkownika i zostanie pokazany panel szczegółów użytkownika aplikacji, który został przedstawiony na rysunku 5.24.



Rysunek 5.23: Panel użytkowników aplikacji



Rysunek 5.24: Panel szczegółów innych użytkowników aplikacji

5.4.5 Wylogowanie

Aby wylogować się z aplikacji należy nacisnąć przycisk przedstawiony na rysunku 5.25, który znajduje się w prawym górnym rogu.



Rysunek 5.25: Przycisk służący do wylogowywania

5.5 Obsługa funkcjonalności dostępnych dla administratora

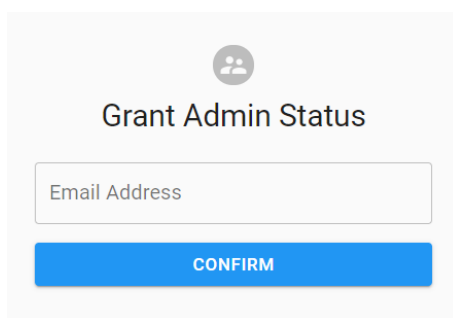
Aby mieć możliwości, które zostały przedstawione poniżej należy posiadać uprawnienia administratora. W tym celu osoba posiadająca konto z takimi uprawnieniami musi nadać osobie, która się o nie stara aby była również administratorem. W innym przypadku możliwości te nie są dostępne dla standardowego użytkownika. Sama informacja o posiadaniu takich uprawnień jest wyświetlana w drugiej sekcji konta użytkownika w postaci komunikatu *Member has admin privileges* i jest to widoczne tylko przez niego oraz w bocznym panelu w postaci przycisku przedstawionego na rysunku 5.26. Sekcje konta użytkownika zostały opisane w sekcji 5.4.1.



Rysunek 5.26: Przycisk otwierający panel administratora

5.5.1 Obsługa panelu administratora

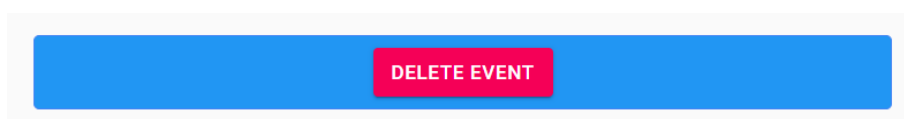
Funkcją dostępną w tym panelu jest nadawanie uprawnień administratora. Aby tego dokonać należy nacisnąć przycisk przedstawionego na rysunku 5.26. Po tej akcji wyświetli się nowa strona z formularzem na, której należy podać adres email konta, które ma mieć uprawnienia administratora oraz zatwierdzić poprzez naciśnięcie przycisku *CONFIRM*. Formularz ten został przedstawiony na rysunku 5.27.



Rysunek 5.27: Panel nadawania uprawnień administratora

5.5.2 Usuwanie wydarzeń

Aby usunąć wydarzenie należy wyświetlić jego szczegóły przedstawione na rysunku 5.21 a następnie nacisnąć dodatkowy przycisk *DELETE EVENT*, który jest dostępny dla administratora przedstawiony na rysunku 5.28. Usuwanie wydarzeń przez administratora jest możliwe zarówno dla przyszłych jak i przeszłych. Usuwanie wydarzeń, które się odbyły jest niezalecane.

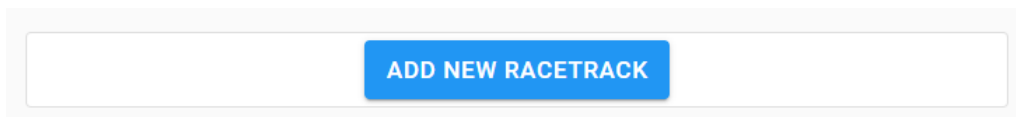


Rysunek 5.28: Przycisk usuwanie wydarzenia

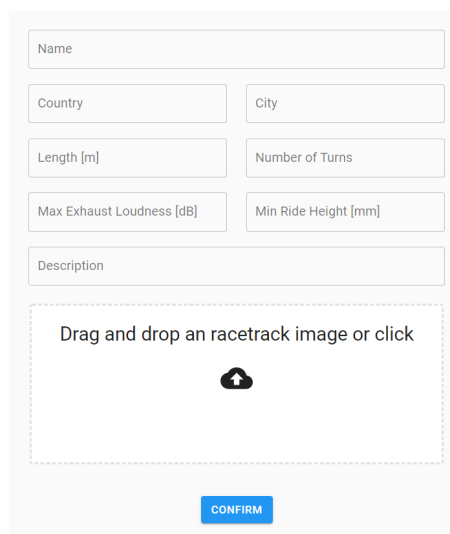
5.5.3 Dodawanie i usuwanie torów wyścigowych

W celu dodanie nowego obiektu sportowego należy w bocznym panelu należy nacisnąć przycisk *Racetracks*. Po tej akcji zostanie wyświetlony panel przedstawiony na rysunku 5.10 z dodatkowym przyciskiem *ADD NEW RACETRACK* przedstawionym na rysunku 5.29. Należy go nacisnąć a następnie zostanie wyświetlony formularz przedstawiony na na rysunku 5.30, który należy wypełnić zgodnie z opisem przy każdym polu oraz nacisnąć przycisk *CONFIRM*.

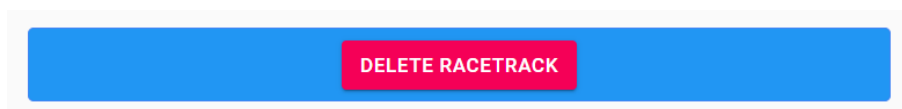
Aby usunąć wybrany obiekt sportowy należy wyświetlić szczegóły obiektu sportowego, pokazanego na rysunku 5.11 a następnie nacisnąć przycisk *DELETE RACETRACK*, który został przedstawiony na rysunku 5.31.



Rysunek 5.29: Przycisk dodawania nowego obiektu sportowego



Rysunek 5.30: Panel dodawania nowego obiektu sportowego



Rysunek 5.31: Przycisk usuwania obiektu sportowego

Rozdział 6

Podsumowanie

Zadaniem pierwszej części niniejszej pracy inżynierskiej było wprowadzenie do tematyki związanej z sportami samochodowymi, w której została przedstawiona ich historia i rozwój na przestrzeni lat. Następnie został zaprezentowany podział na kategorie w tej dziedzinie sportu oraz zostały przedstawione bardzo znane obiekty sportowe znajdujące się w Europie oraz amatorskie starty w zawodach. W kolejnych sekcjach zostały przedstawione wady aktualnie dostępnych aplikacji społecznościowych, które pozostawiły lukę wypełnioną przez aplikację *Social Racetrack*.

W kolejnej części zostały przedstawione technologie użyte w czasie tworzenia aplikacji. Dobór ich nie był pozostawiony przypadkowi. Były one wybrane ze względu na dużą popularność, wsparcie oraz kompatybilność wsteczną, dzięki czemu aplikacja będzie mogła być rozszerzana o kolejne funkcjonalności bez potrzeby jej gruntownej przebudowy. Samo jej utrzymanie nie będzie stanowić problemu.

Trzecia już z kolei część zawiera dokumentację techniczną aplikacji. Zostały tam przedstawione wymagania funkcjonalne i нефункционалне oraz sam opis implementacji i procesu instalacji. Tak jak zostało wcześniej wspomniane aplikacja została stworzona z myślą o możliwości rozszerzania jej funkcjonalności. Została ona zbudowana w sposób modularny przy pomocy podziału na komponenty w bibliotece React. Sam podział na warstwy jest bardzo ważny. Zapewnia on czytelny i łatwy przepływ danych przez aplikację oraz elementy odpowiedzialne za określone czynności są zgrupowane w wybranych miejscach. Sam proces instalacji został maksymalnie uproszczony poprzez zastosowanie wszystkich dostępnych narzędzi platformy chmurowej Firebase oraz lokalnych skryptów.

Ostatnia część ukazuje pełną dokumentację użytkownika, dzięki której rozpoczęcie korzystania z aplikacji jest niezwykle proste. Dodatkowo minimalistyczny interfejs użytkownika z wykorzystaniem Material Design wspiera szybką naukę obsługi aplikacji.

Wedle szacunków dotyczących rozwoju portali społecznościowych, aplikacja *Social Racetrack* w przeciągu najbliższych kilku miesięcy powinna zgromadzić wiele jej aktywnych użytkowników, którzy są miłośnikami sportów samochodowych i wypełnić lukę w portalach społecznościowych zauważoną przez autora pracy dyplomowej.

Spis rysunków

2.1	Panel tworzenia wydarzenia na portalu Facebook	12
4.1	Kolekcje istniejące w bazie danych Cloud Firestore	42
4.2	Zawartości strukturalne klas <i>Car</i> oraz <i>Award</i>	42
4.3	Lista komponentów bazowych	47
4.4	Lista pozostałych komponentów	47
4.5	Lista komponentów wykorzystywanych na wybranych stronach	47
5.1	Ekran powitalny aplikacji	52
5.2	Przycisk do przełączania do trybu ciemnego	53
5.3	Przycisk do przełączania do trybu jasnego	53
5.4	Przycisk do otwierania bocznego panelu	53
5.5	Boczny panel	54
5.6	Przycisk do zamykania bocznego panelu	54
5.7	Przycisk do rejestracji użytkownika	54
5.8	Przycisk do logowania użytkownika	54
5.9	Przycisk konta użytkownika	54
5.10	Panel wyświetlający wszystkie obiekty sportowe	55
5.11	Panel wyświetlający szczegóły na temat wybranego obiektu sportowego	55
5.12	Panel rejestracji użytkownika	56
5.13	Panel logowania użytkownika	57
5.14	Panel przywracania hasła	57
5.15	Panel konta użytkownika	58
5.16	Przycisk dodawania i edycji panelu użytkownika	58
5.17	Panel dodawania floty pojazdów użytkownika	59

5.18	Panel dodawania osiągnięć sportowych użytkownika	59
5.19	Panel edycji danych osobowych użytkownika	60
5.20	Panel przyszłych wydarzeń	61
5.21	Panel szczegółów wydarzenia	61
5.22	Panel tworzenia wydarzeń	62
5.23	Panel użytkowników aplikacji	62
5.24	Panel szczegółów innych użytkowników aplikacji	63
5.25	Przycisk służący do wylogowywania	63
5.26	Przycisk otwierający panel administratora	64
5.27	Panel nadawania uprawnień administratora	64
5.28	Przycisk usuwanie wydarzenia	64
5.29	Przycisk dodawania nowego obiektu sportowego	65
5.30	Panel dodawania nowego obiektu sportowego	65
5.31	Przycisk usuwania obiektu sportowego	65

Spis listingów

3.1	Przykład użycia Media Queries do tworzenia RWD	18
3.2	Hierarchia klas elementów	19
3.3	Wyświetlanie warunkowe z wykorzystaniem składni JSX	23
3.4	Przykład komponentu z wykorzystaniem JSX	23
3.5	Przykład komponentu bez wykorzystania JSX	23
3.6	Wyświetlanie stworzonych komponentów	24
3.7	Wyświetlanie głównego komponentu aplikacji	24
3.8	Użycie komponentu z parametrem	24
3.9	Definicja komponentu z parametrem	24
3.10	Przykład komponentu klasowego	26
3.11	Przykład komponentu funkcyjnego	27
3.12	Przykład użycia <i>useState()</i>	28
3.13	Przykład użycia <i>useEffect()</i>	28
3.14	Przykład użycia <i>useContext()</i>	29
3.15	Zasady dostępu do kolekcji w Cloud Firestore	31
3.16	Przykładowy zapis danych do bazy danych	32
3.17	Przykładowy odczyt danych z bazy danych	32
3.18	Przykładowa aktualizacja danych w bazie danych	32
3.19	Przykładowe usuwanie danych z bazy danych	32
3.20	Zasady dostępu do plików w Storage	33
3.21	Przykład funkcji zapewniającej przyznawanie uprawnień administratora	35
4.1	Kod źródłowy klasy <i>FirestoreStorageController</i>	44
4.2	Kod źródłowy klasy <i>RacetrackFirestoreStorageController</i>	44

4.3	Kod komponentu HomePage odpowiedzialny za wyświetlanie jej prawej części	48
4.4	Kod komponentu HomePage odpowiedzialny za wyświetlanie jej lewej części	49
4.5	Sekcja odpowiedzialna za łączenie lewej i prawej części komponentu HomePage	50

Spis tabel

3.1	Porównanie szybkości działania menadżerów NPM oraz Yarn	16
3.2	Minimalne szerokości ekranu oraz nazwy odpowiadających im klas	20

Bibliografia

- [1] Facebook Inc. *Facebook*. [online]. [dostęp: 10.08.2020]. Dostępny w Internecie, URL: <https://www.facebook.com/>.
- [2] Twitter Inc. *Twitter*. [online]. [dostęp: 10.08.2020]. Dostępny w Internecie, URL: <https://twitter.com/>.
- [3] Facebook Inc. *Instagram*. [online]. [dostęp: 02.10.2020]. Dostępny w Internecie, URL: <https://www.instagram.com/>.
- [4] Mozilla. *JavaScript MDN web docs*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/pl/docs/Web/JavaScript>.
- [5] OpenJS Foundation. *Node.JS Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://nodejs.org/en/docs/>.
- [6] Facebook Open Source. *React Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://en.reactjs.org/docs/getting-started.html>.
- [7] Google LLC. *Firebase Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://firebase.google.com/docs>.
- [8] Mozilla. *HTTP MDN web docs*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/pl/docs/Web/HTTP>.
- [9] Oracle Corporation. *Java Documentation*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://www.oracle.com/java/>.
- [10] Pivotal Software. *Spring Framework Documentation*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://spring.io/>.

- [11] Microsoft. *C# Documentation*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://docs.microsoft.com/en-us/dotnet/csharp/>.
- [12] Microsoft. *ASP.NET Core Documentation*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1>.
- [13] Microsoft. *.NET Core Platform*. [online]. [dostęp:]. Dostępny w Internecie, URL: <https://docs.microsoft.com/pl-pl/dotnet/core/>.
- [14] Microsoft. *TypeScript Documentation*. [online]. [dostęp: 20.08.2020]. Dostępny w Internecie, URL: <https://www.typescriptlang.org/>.
- [15] Google Open Source. *V8 Engine Documentation*. [online]. [dostęp: 03.09.2020]. Dostępny w Internecie, URL: <https://v8.dev/docs>.
- [16] Google LLC. *Google Chrome Documentation*. [online]. [dostęp: 03.09.2020]. Dostępny w Internecie, URL: <https://developer.chrome.com>.
- [17] Google LLC. *Chromium Documentation*. [online]. [dostęp: 03.09.2020]. Dostępny w Internecie, URL: <https://www.chromium.org/Home>.
- [18] Inc. npm. *Node Package Manager Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://docs.npmjs.com/>.
- [19] Yarn. *Yarn Package Manager Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://classic.yarnpkg.com/en/docs>.
- [20] pnpm. *Benchmarks of JavaScript Package Managers*. [online]. [dostęp: 02.09.2020]. Dostępny w Internecie, URL: <https://github.com/pnpm/benchmarks-of-javascript-package-managers/blob/master/README.md>.
- [21] Git. *Git Documentation*. [online]. [dostęp: 05.09.2020]. Dostępny w Internecie, URL: <https://git-scm.com/doc>.
- [22] Thornton Jason Otto Mark. *Bootstrap Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>.
- [23] MDBBootstrap. *Material Design for Bootstrap Documentation*. [online]. [dostęp: 31.07.2020]. Dostępny w Internecie, URL: <https://mdbootstrap.com/>.

- [24] Google LLC. *AngularJS Documentation*. [online]. [dostęp: 10.09.2020]. Dostępny w Internecie, URL: <https://angular.io/docs>.
- [25] Evan You. *Vue.JS Guide*. [online]. [dostęp: 10.09.2020]. Dostępny w Internecie, URL: <https://vuejs.org/v2/guide/>.
- [26] John Resig. *jQuery Documentation*. [online]. [dostęp: 10.09.2020]. Dostępny w Internecie, URL: <https://api.jquery.com/>.
- [27] Google LLC. *Material Design*. [online]. [dostęp: 10.09.2020]. Dostępny w Internecie, URL: <https://material.io/design>.
- [28] Mozilla. *DOM MDN web docs*. [online]. [dostęp: 14.09.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/pl/docs/DOM>.
- [29] Facebook Open Source. *ReactDOM Documentation*. [online]. [dostęp: 14.09.2020]. Dostępny w Internecie, URL: <https://en.reactjs.org/docs/react-dom.html>.
- [30] Mozilla. *HTML MDN web docs*. [online]. [dostęp: 15.09.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/pl/docs/Web/HTML>.
- [31] Mozilla. *REST MDN web docs*. [online]. [dostęp: 06.09.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/en-US/docs/Glossary/REST>.
- [32] Mozilla. *SSL MDN web docs*. [online]. [dostęp: 09.09.2020]. Dostępny w Internecie, URL: https://developer.mozilla.org/en-US/docs/Archive/Security/Introduction_to_SSL.
- [33] Mozilla. *Mozilla Firefox Documentation*. [online]. [dostęp: 19.09.2020]. Dostępny w Internecie, URL: <https://developer.mozilla.org/pl/docs/Mozilla/Firefox>.