



A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INŻYNIERII METALI I INFORMATYKI PRZEMYSŁOWEJ

KATEDRA INFORMATYKI STOSOWANEJ I MODELOWANIA

Projekt dyplomowy

*Projekt i wykonanie systemu zarządzania wypożyczalnią oraz internetowej
obsługi klienta*

*Design and implementation of rental management system and online
customer service*

Autorzy: Grzegorz Studziński, Michał Stawarski

Kierunek studiów: Informatyka Stosowana

Opiekun pracy: dr hab. inż. Dorota Wilk-Kołodziejczyk

Kraków, 2022

Spis treści

1. Wstęp	3
1.1 Streszczenie	3
1.2 Cel	4
1.3 Dostępne technologie	4
2. Opis projektu	9
2.1 Projekt systemu	9
2.2 Podział pracy	10
2.3 Integracja danych	11
2.3.1 Integracja danych ze strony webowej	12
2.3.2 Integracja ze strony desktopowej	13
2.4 Kolekcje	15
2.5 Szyfrowanie danych	17
3. Aplikacja webowa	19
3.1. Wymagania funkcjonalne	19
3.2. Wykorzystane technologie	20
3.2.1 Spis technologii	20
3.2.2 Uzasadnienie doboru technologii	22
3.3 Wybrane aspekty implementacji	23
3.3.1 Statyczne typowanie na przykładzie komunikacji z API	24
3.3.2 Komponentowa struktura aplikacji	28
3.3.3 Zarządzanie stanem i optymalizacja kodu	30
3.4 Funkcjonalności	32
3.4.1. Możliwości rejestracji i uwierzytelniania	32
3.4.2. Obsługa wiadomości e-mail.	38
3.4.3. Edycja własnego profilu.	41
3.4.4. Przeglądanie produktów i rezerwacja	42
3.4.5. Podgląd aktualnych, anulowanych i archiwalnych usług	47
3.4.6. Responsywność i dostępność na urządzeniach mobilnych	49
3.4.7. Sesja użytkownika	50
3.4.8. Obsługa routingu	50
3.4.9. Elementy sterowane z aplikacji desktopowej	52
3.5. Testy	53
4. Aplikacja desktopowa	56
4.1 Wykorzystane technologie	56
4.2 Wymagania funkcjonalne	58
4.3 Struktura aplikacji	59
4.3.1 Wzorzec projektowy MVC	59

4.3.2 Testy	62
4.4 Funkcjonalności	63
4.4.1 Logowanie	63
4.4.2 Główny panel	64
4.4.3 Equipment (Sprzęt)	65
4.4.4 Settings (Ustawienia)	66
Ustawienia - edycja typu (kategorii) oraz cennika	67
Ustawienia - edycja pracowników	69
Ustawienia - masowe usuwanie klientów oraz sprzętu.	70
Ustawienia - zarządzanie podstawowymi informacjami dot. firmy	72
4.4.5 Clients (Klienci)	74
4.4.6 Reservations (Rezerwacje)	77
4.4.7 Rent (Wypożycz)	80
4.4.8 Return (Zwrot)	84
5. Podsumowanie	86
5.1 Wnioski	87
5.2 Możliwości rozwoju	88
Bibliografia	88

1. Wstęp

1.1 Streszczenie

W czasach coraz prężniej rozwijającego się pro-konsumenckiego podejścia firm usługowych, rozwoju Internetu, oraz idącymi za tym technologiami, firmy muszą dostosować swoje projekty do oczekiwów rynku, które razem z upływem czasu zmieniają się diametralnie. W niniejszej pracy przedstawiona została analiza dostępnych na rynku systemów do obsługi wypożyczeń. Za cel pracy postawiono dogłębną analizę technologii oraz wybór narzędzi, które w obecnym czasie, pozwalają na utworzenie w pełni zintegrowanego systemu desktopowego dla pracowników wypożyczalni i systemu webowego dla klientów. Uargumentowano wybór poszczególnych z nich. Począwszy od wybranego języka programowania, przez użycie wybranych bibliotek, wybór odpowiedniej bazy danych po zabezpieczenie danych. Dodatkowo w pracy ukazano projekt implementacji. Opisany został również dokładny podział na zadania z racji pracy grupy dwuosobowej, w której każda osoba miała swoje zadanie. Zaprezentowano tytułowy system obsługi wypożyczalni wraz z internetowym modułem obsługi klienta, zaimplementowane funkcjonalności obydwu modułów oraz szczegóły ich integracji. Przedstawione zostały również ważniejsze fragmenty kodu, wymagające analizy czy opisu. Po wykonaniu wszystkich czynności, podsumowano całą wykonaną pracę, w tym jej przebieg, nabycie umiejętności, wnioski ogólne oraz możliwości dalszego rozwoju.

1.2 Cel

Głównym celem było wykorzystanie nabytych umiejętności informatycznych na przestrzeni lat studiowania, do stworzenia projektu systemu obsługi wypożyczalni, przegląd rozwiązań programistycznych, umożliwiających jego poprawne działanie, w tym zapewnienie integracji pomiędzy zarówno modułem webowym jak i desktopowym, oraz jego późniejsza implementacja. Zintegrowany system posłuży zarówno pracownikom obsługującym punkt stacjonarny, jak i klientom chcącym dokonać rezerwacji przez Internet, w tym z urządzenia mobilnego. Użytkownicy będą mogli również sprawdzić stan obecnego wypożyczenia. Dzięki postawionym wymaganiom aplikacja stanie się w pełni dostosowana do wymagań obecnego rynku.

Praca ta ma na celu nie tylko stworzyć system służący do obsługi wypożyczalni, lecz przede wszystkim poszerzyć wiedzę oraz umiejętności, nabite na przestrzeni lat studiowania.

1.3 Dostępne technologie

Jest wiele dostępnych technologii i gotowych rozwiązań na rynku. Po głębszej analizie oraz przeprowadzonym badaniu rynku, obecne programy można sklasyfikować na dwie podgrupy:

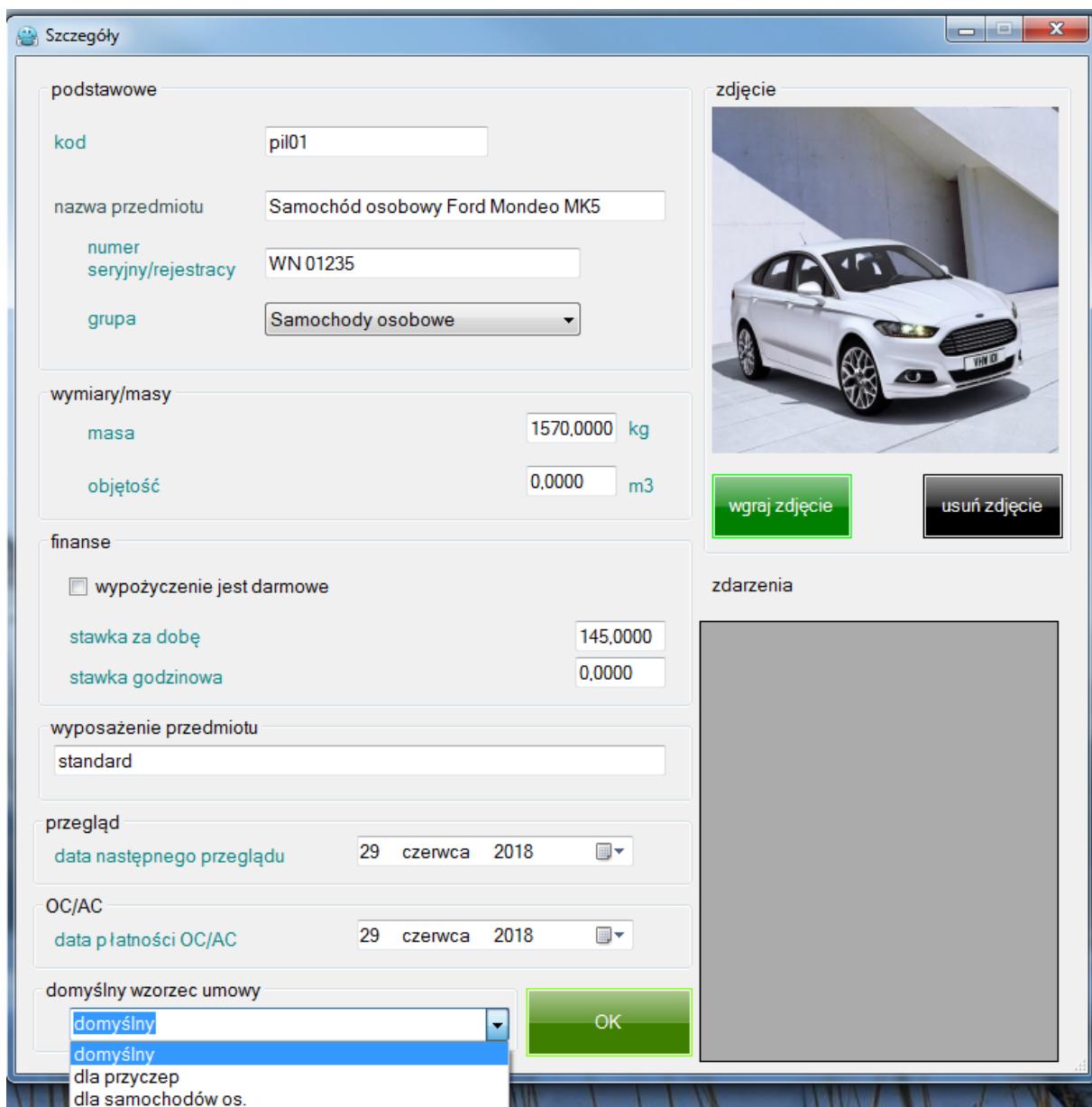
- a) Aplikacje tworzone przez małe firmy, często jednoosobowe działalności gospodarcze
- b) Aplikacje tworzone przez korporacje

W pierwszej podgrupie występuje wiele oprogramowania dostępnego na rynku, charakteryzującego się niską ceną, lecz również - idącą za tym - niedostateczną jakością. Sprawdzono wiele z występujących technologii, które posiadały wersję testową. Programy te często po pierwszej instalacji nie działały prawidłowo, a jednym z większych minusów było to, że nie posiadały modułów internetowych.

Druga podgrupa charakteryzowała się tym, że jest tworzona przez większe, doświadczone firmy, a sam system wypożyczeń jest rozbudowywany, często wychodzący za sam sektor wypożyczeń i dla większości mniejszych przedsiębiorstw jest po prostu zbędny. Cały proces wdrożenia systemu zajmuje kilka tygodni, a sama cena za subskrypcje lub abonament jest wysoka. Minusem jest to, że za dodatkowy moduł internetowy należy uiścić kolejną kwotę, a to w większości małych przedsiębiorstw może skutkować zrezygnowaniem z tej technologii.

- CerTech - program “Wypożyczalnia”

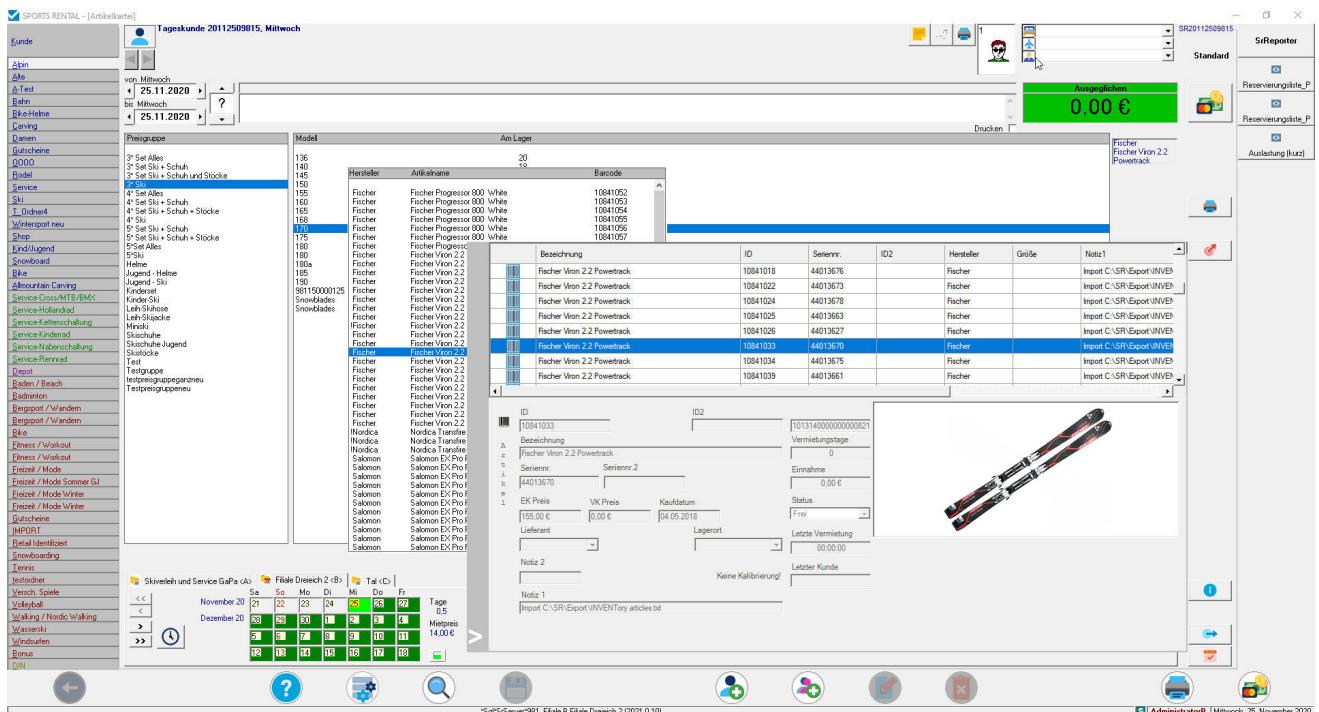
Aplikacja zawiera wszelkie funkcjonalności służące do obsługi wypożyczeń, w tym własny system fakturujący, to jest firma jest podpinana do kartoteki osobowej, a dane pobierane są automatycznie z GUS. Program posiada również wzorce umów, możliwość generowania raportów oraz obsługę czarnej listy. Wypożyczenie może odbywać się w trzech trybach: na dobę, na godziny oraz na czas nieokreślony. Aplikacja desktopowa dostarczana przez CerTech nie posiada modułu webowego, więc klient nie może samodzielnie wykonać rezerwacji sprzętu. Ta może zostać dokonana przez pracownika wypożyczalni, jednak w obecnych czasach najbardziej pożądanym rozwiązaniem jest zapewnienie możliwości dokonania rezerwacji przez samego klienta bezpośrednio przez internet.



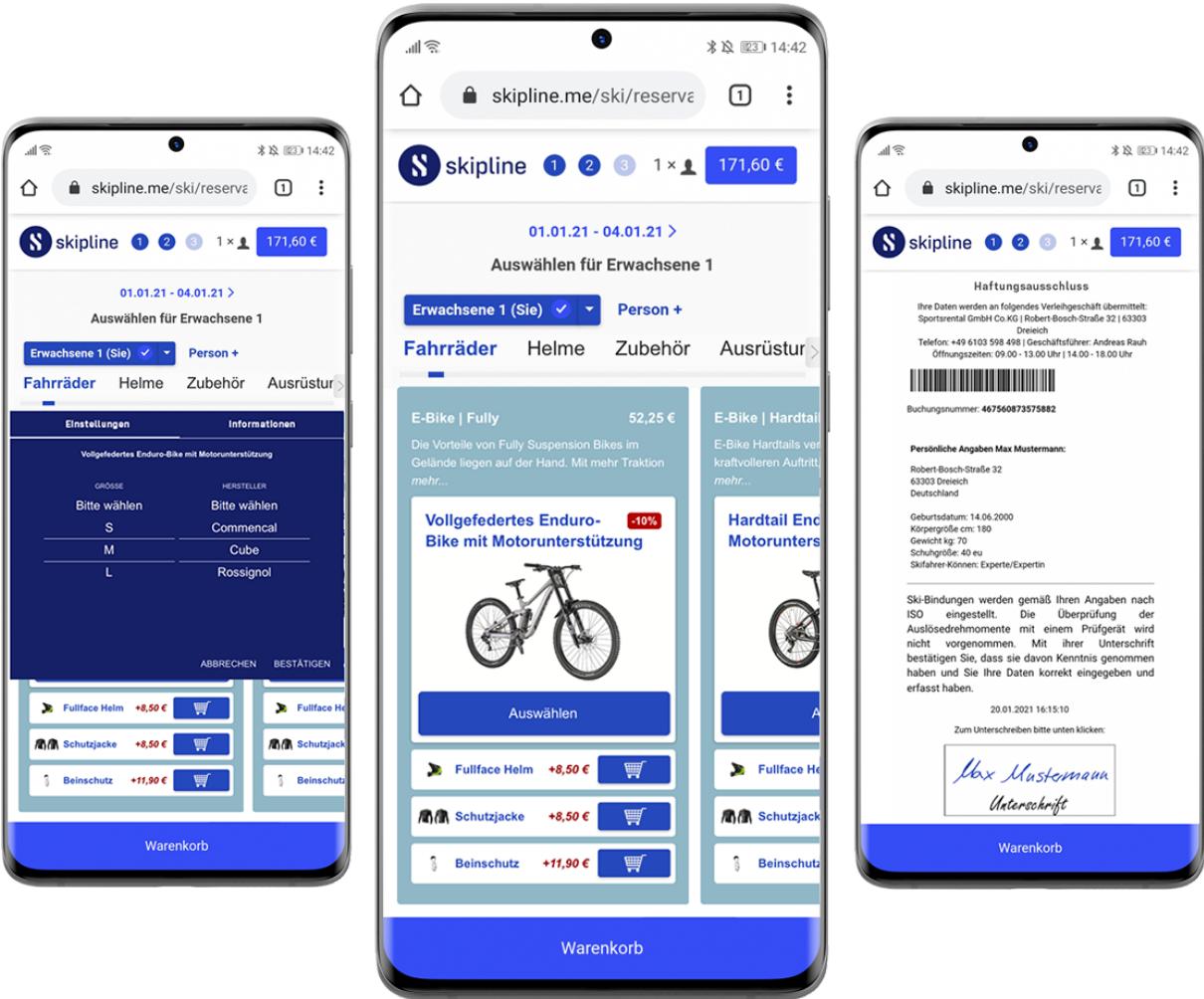
Zrzut ekranu 1. CerTech Wypożyczalnia. Źródło [1]

• S|R Software - program “S|R Rental”

Wiodące na rynku wypożyczeń sportowych oprogramowanie, powstałe w 1994 roku. Stworzone przez niemiecką firmę S|R Software. Dedykowany dla branż rowerowych, narciarskich i sportów wodnych takich jak surfing lub wypożyczalnia łodzi. Oprogramowanie zawiera większość potrzebnych funkcjonalności od wypożyczeń po zwroty oraz pełną obsługę magazynową. Program posiada również dedykowany system rezerwacji online. System jednak nie jest w pełni spójny. Moduł internetowy należy aktywować za dodatkową roczną opłatą abonentową, dostępną w 4 opcjach. Każdy z modułów jest optymalny dla wybranej ilości rezerwacji dokonywanych przez klientów w ciągu roku. Trzy pierwsze posiadają dodatkową opłatę (pomimo startowej), za każdą dokonaną przez klienta rezerwację. Moduły posiadają również ograniczenia w postaci dostępnych grup sportowych, w których dostępny jest sprzęt w zależności od wybranej opcji. Dedykowanymi językami są niemiecki i angielski, za wybranie innego należy uiścić dodatkową opłatę. Istnieje możliwość rozszerzenia modułu internetowego o dodatkowe, niekiedy bardziej spersonalizowane funkcje. Nie znaleziono informacji dotyczących podglądu wypożyczenia online, może być to jednak zawarte w dedykowanej aplikacji “skipline” (zrzut ekranu 3).



Zrzut ekranu 2. S|R Rental. Źródło: [2]



Zrzut ekranu 3. Skipline. Źródło [3]

• Ulisses Comotel Rental

Aplikacja desktopowa dostarczana przez firmę Ulisses jest rozwijana od ponad 15 lat, a ich rozwiązań - według producenta - używało ponad 150 firm. Producent zapewnia, że zakres jej działania pozwala na prowadzenie wynajmu w prawie każdej branży. Omawiana aplikacja jest jedną z dwóch programów do obsługi wypożyczalni oferowanych przez producenta, dostarcza również webową wersję Quinno Rental. Obie aplikacje nie mają ze sobą jednak powiązania, więc to klient musi zdecydować, którego programu ma zamiar używać. Aplikacja zawiera szereg funkcjonalności takich jak obsługa rezerwacji, wydań, zwrotów, pozwala również dokonać analizy rentowności oraz wygenerować raporty. Wynajem odbywa się w dwóch kategoriach: wynajem maszyn, w której to rozliczana jest każda sztuka według stawki czasowej oraz wynajem elementów, w której to klient rozliczany jest w innej jednostce, takiej jak metry kwadratowe czy procent od wartości. Program uruchamiany jest lokalnie na komputerze, lecz posiada dodatkowe webowe nakładki. Pierwsza z nich to Comotel WEB, która pozwala pracownikom na mobilny dostęp do podglądu wynajmu,

serwisu i CRM. Kolejną nakładką działającą w przeglądarce jest Comotel B2B, która umożliwia sprawdzenie stanu sprzętu oraz dokumentów magazynowych przez klientów wypożyczalni. Moduł webowy nie pozwala jednak na dokonanie samodzielnej rezerwacji przez klienta.

	Kod KTH	Nazwa KTH	Blokada	Umowa	P <small>I</small>	Data płatności	Wartość	Waluta	F
► 1	K0009	Comotex Sp. z o.o.		Up/1/3/20	P	01.09.2020	8 000,00	PLN	<input checked="" type="checkbox"/>
2	K0009	Comotex Sp. z o.o.		Up/2/3/20	P	01.09.2020	20 000,00	PLN	<input checked="" type="checkbox"/>
3	K0009	Comotex Sp. z o.o.		Up/2/6/20	P	01.09.2020	5 980,00	PLN	<input checked="" type="checkbox"/>
4	K0009	Comotex Sp. z o.o.		Up/3/8/20	P	01.09.2020	29 800,00	PLN	<input checked="" type="checkbox"/>
5	K0010	DRAMERS SPÓŁKA AKCYJNA		Up/3/3/20	P	01.09.2020	1 500,00	PLN	<input checked="" type="checkbox"/>
6	K0010	DRAMERS SPÓŁKA AKCYJNA		Up/3/6/20	P	01.09.2020	2 080,00	PLN	<input checked="" type="checkbox"/>
7	K0011	Fresh Food Company Spółka komandytowa	blokada\War	Up/1/4/20	P	01.09.2020	8 092,42	PLN	<input checked="" type="checkbox"/>
8	K0012	ADAMIX spółka z ograniczoną odpowiedzialnością spółk...		Up/1/5/20	P	01.09.2020	1 440,00	PLN	<input checked="" type="checkbox"/>
9	K0013	Nowicki & Nowicki Sp. z o.o.		Up/1/6/20	P	01.09.2020	2 772,00	PLN	<input checked="" type="checkbox"/>
10	K0013	Nowicki & Nowicki Sp. z o.o.		Up/1/7/20	P	01.09.2020	675,00	PLN	<input checked="" type="checkbox"/>
11	K0015	HAULOTTE POLSKA SPÓŁKA Z OGRANICZONĄ ODPO...		Up/1/8/20	P	18.08.2020	2 100,00	PLN	<input checked="" type="checkbox"/>

Data: Serie: Seria faktur: Data wyst. faktur: Data VAT: Przelicz Fakturuj

Waluta	Suma	Wybrane
PLN	82 439,42	82 439,42

Fakturuj wszystkich kontrahentów

Zrzut ekranu 4. Aplikacja Comotel Rental, Źródło: [4]

2. Opis projektu

2.1 Projekt systemu

Pierwszą czynnością, podczas koncepcyjnej części tworzenia projektu, było utworzenie diagramu UML. Ta część została utworzona, jeszcze przed wykonaniem szczegółowej analizy możliwości implementacyjnych systemu. Jak się okazało, podczas pisania samych aplikacji, wybrane elementy zostały zmienione. Główna część schematu została jednak zachowana i posłużyła do budowy programu. Zaktualizowany diagram przypadków użycia (ang. UML, Use Case Diagram) został przedstawiony jako Diagram 1.

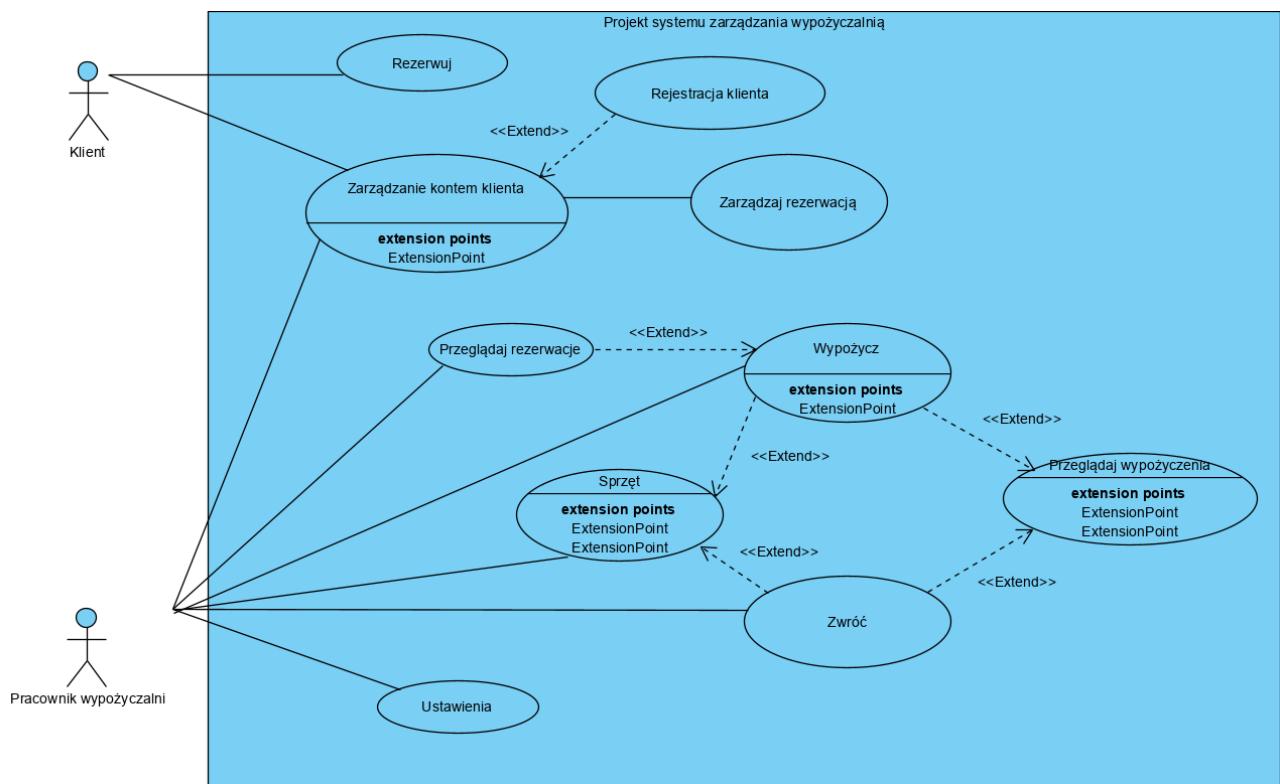


Diagram 1. Projekt systemu zarządzania wypożyczalnią. Źródło: opracowanie własne

Diagram przedstawia zależności pomiędzy danymi przypadkami użycia. Można zauważyć, jakie operacje może wykonać aktor. W systemie mamy dwóch aktorów, jakimi są Klient oraz Pracownik Wypożyczalni. Klient może zarządzać swoim kontem, w tym zarejestrować się w systemie, zarządzać rezerwacjami oraz wyszukać produkt i dokonać rezerwacji. Pracownik wypożyczalni odpowiedzialny jest z kolei za całe spektrum operacji po stronie wypożyczalni, takie jak zarządzanie jej danymi, sprzętem, wypożyczeniami i rezerwacjami. Obydwoje aktorów, ma zatem pewien zakres działań, którego obsługa jest dostarczona w formie oddzielnych aplikacji.

2.2 Podział pracy

Podział pracy w tytułowym systemie obsługi wypożyczalni wynika z racji współpracy dwóch osób, kooperujących ze sobą podczas tworzenia projektu inżynierskiego. Jest ona podzielona na dwa główne moduły:

1. Moduł webowy
2. Moduł desktopowy

Oprócz nich wystąpi również jeden mniejszy, a mianowicie moduł integracji nad którymi będą pracowali obydwa autorzy, w celu pełnego i prawidłowego połączenia obydwu głównych modułów. Należy również wspomnieć o samej koncepcji systemu, która również wymagała dyskusji współpracujących osób w zespole. Wspólna praca dwóch systemów wymaga pełnego omówienia oraz opisu, gdyż niedokładna jego realizacja może spowodować zatracenie całego sensu tworzenia wspólnego oprogramowania, na rzecz tworzenia dwóch odrębnych systemów, prac. Ostatecznie, należy podkreślić, że większość nakładu pracy zostało włożone w same aplikacje, które autorzy stworzyli indywidualnie, wykorzystując stworzony wcześniej koncept. Również podczas tworzenia aplikacji powstawało wiele nowych zagadnień, których poruszenie było nieuniknione, by stworzyć jeden spójny system. Jako przykład można podać różnice w wysyłanych danych z jednego systemu mogą spowodować braki w podstawowym działaniu drugiego. Integracja obu systemów, wiązała się więc z ciągłą komunikacją, przed oraz w trakcie implementacji.

Moduł webowy: jego głównym zadaniem jest obsługa klientów internetowych, tych którzy chcieliby skorzystać z usług danego przedsiębiorstwa, a dzięki łatwemu oraz intuicyjnemu

systemowi, będą mogli dokonać rezerwacji, jej edycji i sprawdzenia statusu. Oprócz tego aplikacja umożliwia klientom na podgląd swojego wypożyczenia.

Moduł desktopowy: skupia się głównie na obsłudze klientów stacjonarnych przez pracowników danego przedsiębiorstwa. Jednym z głównych funkcjonalności w systemie będzie możliwość wypożyczenia oraz zwrotu towaru, jego edycja oraz obsługa rezerwacji, które zostaną dokonane poprzez specjalny moduł internetowy.

Powyższy podział pracy jest tylko uproszczonym opisem. Pełny opis wszystkich modułów, technologii, realizacji dokonany zostanie w kolejnych rozdziałach.

Charakterystyka	Osoba odpowiedzialna	Język programowania
Projekt systemu	praca wspólna	nie dotyczy
Aplikacja webowa	Grzegorz Studziński	TypeScript
Aplikacja desktopowa	Michał Stawarski	Java

Tabela 1. Podział pracy w projekcie inżynierskim. Źródło: opracowanie własne

2.3 Integracja danych

Podstawowym wyzwaniem tytułowego projektu było zapewnienie szybkiego dostępu do informacji oraz ich edycji zarówno w aplikacji desktopowej, jak i webowej. Oznacza to bezwzględną potrzebę odświeżenia danych na bieżąco w obu aplikacjach. Jako przykład można podać sytuację gdy klient utworzy rezerwację w portalu internetowym, pracownik wypożyczalni potrzebuje mieć wgląd do tej rezerwacji od razu po jej dokonaniu. Z drugiej strony jeśli ten zdecyduje się ograniczyć godziny pracy wypożyczalni, klient korzystający z webowego interfejsu musi w tym samym momencie mieć ograniczenie wyboru godzin wypożyczenia na takie, które mieszczą się w tym przedziale.

Wraz z wzrostem możliwości jakie przyniósł Internet pojęcie oprogramowania jako usługi (ang. *software as a service, SaaS*) stawało się coraz to bardziej popularne. Wspomniane pojęcie w kontekście oprogramowania oznacza formę udostępniania przez firmę mocy obliczeniowej lub w

tem własnego oprogramowania, by umożliwić jej użytkownikowi dostęp do tych zasobów z dowolnego miejsca na świecie, z wykorzystaniem sieci. Autorzy postanowili poszukać rozwiązań bazodanowych działających właśnie w tym modelu. Usługa bazy danych w chmurze (ang. *database as a service, DBaaS*) to jedna z najszybciej rozwijających się usług chmurowych. Zaletą dla twórców oprogramowania jest możliwość korzystania z w pełni funkcjonalnej bazy danych, bez bycia zmuszonym do zarządzania i utrzymywania technologii bazodanowych. Zapewnia też bezpieczeństwo na wysokim poziomie nad którym czuwa dostawca usługi.

Głównym czynnikiem w wyborze dostawcy, oprócz łatwości w użytkowaniu, była również możliwość wykorzystania dzielonego klastra danych bez żadnych opłat. W projekcie został wykorzystany MongoDB Atlas, czyli nierelacyjny system zarządzania bazą danych w formie NoSQL. Autorzy tego rozwiązania zapewniają darmowy dostęp do bazy danych do 5 gigabajtów użytkowanej pamięci oraz dzieloną pamięć RAM.

2.3.1 Integracja danych ze strony webowej

Tradycyjny sposób łączenia się z bazą danych w aplikacji webowej to połączenie się z serwerem, na którym jest utworzone połączenie z bazą danych. MongoDB Atlas, jako usługa bazy danych w chmurze, zapewnia jednak rozwiązanie które umożliwia dostęp do danych bez potrzeby uruchamiania dodatkowego serwera. Udostępniane przez autorów API pozwala na czytanie i modyfikowanie danych zupełnie jak w tradycyjnym podejściu, lecz bez dodatkowych sterowników czy bibliotek. Połączenie to jest realizowane przez asynchroniczne wysyłanie zapytań HTTP i przetwarzanie odpowiedzi. Rozwiązanie to pozwala na łatwe śledzenie kodów błędów w przeglądarce oraz zapewnia bezpieczeństwo. Do każdego zapytania dodawany jest wygenerowany klucz API bez którego zapytanie nie będzie prawidłowe i zwrócony zostanie błąd. Pomimo, że zapytania są asynchroniczne, deweloper może wymóc odświeżenia części danych po odświeżeniu strony lub zmianie w kodzie. Szczegóły podstaw implementacyjnych w tym zakresie zostaną omówione w rozdziale 3.3, a sama implementacja w rozdziale 3.4. Zapytanie jest wysyłane również po utworzeniu rezerwacji czy użytkownika. Nie wymaga to jednak ciągłego obciążania sieci, podczas gdy wszystkie dane są pobrane. Należy wspomnieć, że format danych w bazie NoSQL to JSON, który stosowany jest bardzo często w technologiach webowych, szczególnie w języku JavaScript, który sam w sobie pozwala na edycję i czytanie danych. Do obsługi zapytań HTTP użyto biblioteki axios. Metoda została napisana w sposób generyczny, to jest znajduje się w jednym

pliku i pozwala na dowolne skonfigurowanie zapytania. Opis tej metody znajduje się w rozdziale 3.1, w którym opisano specyfikę języka TypeScript, ponieważ na jej przykładzie zostaną przedstawione jego cechy.

2.3.2 Integracja ze strony desktopowej

Dzięki dobrze rozbudowanemu API MongoDB Driver, połączenie bazy danych z aplikacją desktopową może zachodzić przez kilka sposobów. Każda z nich ma swoje plusy i minusy. Dwoma głównymi sposobami na integrację dwóch systemów jest:

1. Wysyłanie i otrzymywanie zapytań HTTP
2. Stałe połączenie obydwu aplikacji poprzez URI generowane w kliencie MongoDB.

Główną różnicą pomiędzy dwoma sposobami jest czas wykonywania oraz łatwość dostępu. Wysyłanie i otrzymywanie zapytań jest praktycznie natychmiastowe podczas tworzenia aplikacji webowej, w desktopowej operacja ta jest wykonywana zdecydowanie dłużej. Czas wykonywania takiej operacji, jest kluczowy, gdy zapytań będzie wysyłanych wiele. Aby wysłać takie zapytanie z poziomu aplikacji okienkowej, należało skorzystać z pomocy zewnętrznych bibliotek. OKHTTP3 [5] to technologia umożliwiająca wysyłanie zapytań do bazy danych. Przesyłanie to można podzielić na 3 czynności.

Najpierw należało utworzyć takie zapytanie oraz ustawić jako domyślny format danych plik JSON (ang. JavaScript Object Notation)..

```
OkHttpClient client = new OkHttpClient().newBuilder().build();
MediaType mediaType = MediaType.parse("application/json");
```

Fragment kodu 1 - tworzenie zapytań HTTP. Źródło: opracowanie własne.

Następnie utworzyć ciało pliku, w którym przetrzymywane będą podstawowe informacje potrzebne aby komunikować się z bazą danych.

```
RequestBody body = RequestBody.create(mediaType,
    content: "{\n      \"collection\": \"items\", \n      \" +\n      \"database\": \"rental-data\", \n      \" +\n      \"dataSource\": \"rental-system\", \n      \" +\n      \"document\": {\"type\": \"ski\"}\n    }"
);
```

Fragment kodu 2 - tworzenie ciała zapytania HTTP. Źródło: opracowanie własne.

Na samym końcu należy uzupełnić dane takie jak: adres URL, gdzie będzie wysyłane nasze zapytanie, z ustawieniem filtra insertOne, findOne, itp. Służą one do określenia jakie operację będą wykonywane. Następnie należy ustawić metodę obsługi zapytania w tym przypadku “POST”. Należy dodać nagłówek posiadający informacje takie jak: typ zawartości danych, ciała (ang “body”), należy również wskazać, które nagłówki będą używane podczas zapytania. Należy również podać klucz API, wygenerowany w kliencie bazy danych Mongo DB. Na samym końcu należy wykonać (execute) operację wysłania zapytania HTTP.

```
Request request = new Request.Builder()
    .url("https://data.mongodb-api.com/app/data-lasjp/endpoint/data/beta/action/insertOne") //findOne/insertOne
    .method( "POST", body)
    .addHeader( name: "Content-Type", value: "application/json")
    .addHeader( name: "Access-Control-Request-Headers", value: "*")
    .addHeader( name: "api-key", value: "6VPv7kgAnpG0Ge0PpLd0XAT0NkGoz7lzri7rve7KbbQ5wsLxeDXnSp4WtBtDEhQG")
    .build();
Response response = client.newCall(request).execute();
```

Fragment kodu 3 - tworzenie zapytań HTTP. Źródło: opracowanie własne.

Zwracany zostaje plik lub informacja w zależności od ustawień powyższego fragmentu kodu.

Pomimo poprawności działania całej operacji, czas wykonania, otrzymania lub wysłania zapytania zajmuje za dużo czasu, aby aplikacja mogła działać sprawnie. Sama obsługa tej technologii również nie należy do najprostszych i najszybszych możliwych rozwiązań dostępnych na rynku. Te dwa aspekty dyskwalifikują ten sposób integracji dwóch systemów - bazy danych oraz aplikacji desktopowej

Drugim, zdecydowanie szybszym sposobem jest stałe połączenie aplikacji desktopowej z bazą danych. Odbywa się dzięki udostępnionym API MongoDB Java Driver Sync [6]. Jedyne co użytkownik tej technologii musi zrobić to pobrać udostępnioną bibliotekę, a następnie utworzyć konto w , dzięki któremu będzie odbywała się komunikacja pomiędzy systemami. Należy wygenerowany w kliencie bazy danych link URI użyć do połączenia się z chmurą i wskazać, którą dokładnie bazę danych chcesz użyć.

```
static String uri = "mongodb+srv://";
static MongoClient mongoClient = MongoClients.create(uri);
static MongoDatabase database = mongoClient.getDatabase( s: "rental-data");
```

Fragment kodu 4 - tworzenie połączenia z bazą danych MongoDB, Java driver sync. Źródło własne.

Następnie wskazać, do której kolekcji użytkownik chce mieć dostęp. Jest ona wtedy natychmiast pobierana, a programista może swobodnie korzystać z pobranych danych. Do jego obsługi w języku Java należało jednak użyć specjalnej biblioteki BSON [7], która pozwoliła na tworzenie odpowiednich dokumentów.

```
MongoCollection<Document> collection = database.getCollection( "items");
```

Fragment kodu 5 - pobieranie kolekcji z bazy danych MongoDB. Źródło własne.

Dzięki udostępnionym w bibliotece funkcjom można wykonywać różne operacje na kolekcji, takie jak: dodawanie, usuwanie, edycję obiektów w bazie danych.

Sposób ten posiada również swoje wady. Wymaga stałego połączenia internetowego aplikacji desktopowej z bazą danych w chmurze. Podczas utraty połączenia aplikacja okienkowa nie będzie działać poprawnie, gdyż nie możliwy będzie dostęp do danych. Należy ten fakt mieć na uwadze wybierając ten sposób integracji danych między dwoma systemami. Podczas częstego dostępu do danych ten sposób jest zdecydowanie lepszy oraz szybszy, jednak gdy z danych w bazie użytkownik nie będzie korzystał bardzo często, wybór pierwszego sposobu może okazać się lepszy z racji braku potrzeby ciągłego dostępu do sieci internetowej.

2.4 Kolekcje

Dane przechowywane w nierelacyjnej bazie danych składają się z kolekcji. Każda składa się z dokumentów w określonym formacie. W projekcie utworzono 7 kolekcji.

1. Rezerwacje (ang. *reservations*) - zawiera numer rezerwacji, numer produktu, numer użytkownika, datę odbioru, datę zwrotu, cenę oraz status.
2. Ceny (ang. *prices*) - zawiera typ sprzętu oraz ceny zarówno za godzinę jak i za dobę wynajmu
3. Przedmioty (ang. *items*) - zawiera numer przedmiotu, jego typ, producenta, model oraz rozmiar.
4. Firma (ang. *company*) - zawiera numer telefonu, e-mail, nazwę wypożyczalni (pełniącej również rolę tytułu menu aplikacji internetowej wypożyczalni), godziny otwarcia i zamknięcia, adres oraz procent sprzętu przeznaczonego do rezerwacji on-line
5. Pracownik (ang. *employee*) - zawiera login i hasło pracownika, jego numer oraz imię i nazwisko.
6. Wypożyczenia (ang. *rentals*) - zawiera numer wypożyczenia, numer produktu, numer użytkownika, datę odbioru, datę zwrotu, cenę oraz status

7. Użytkownicy (ang. *users*) - zawiera numer, imię, nazwisko, e-mail, numer powiązania konta z pojedynczym logowaniem Google, telefon, hasło oraz numer dokumentu. Dane są zaszyfrowane.

Na zrzucie ekranu 5 przedstawiono opisywane kolekcje, wraz z wybranym, przykładowym dokumentem.

<pre> reservations _id: ObjectId("61d35d9f639e7eb85e90a451") productId: "3" userId: "61d24c299f5076cb5b581473" startDate: "1642174405000" finishDate: "1643207607000" price: "300" status: "confirmed" </pre>	<pre> prices _id: ObjectId("61ca219ca19b6956a389c44f") type: "ski" hour: "20" day: "70" </pre>
<pre> items _id: ObjectId("61d0ec7fb8b1812f19b0483e") type: "ski" producer: "Salomon" model: "Ski Runner x8" productId: "1" size: "150" </pre>	<pre> employee _id: ObjectId("61dcbe8ad46dff5c3868aa25") user: "506c709e6c2ee32b02b56f6419bb505aL2e10Gf password: "506c709e6c2ee32b02b56f6419bb505aG8t name: "506c709e6c2ee32b02b56f6419bb505aVqbjSKE surname: "506c709e6c2ee32b02b56f6419bb505ax5Li </pre>
users	
<pre> _id: ObjectId("61d602d3a2660ba02df3bf51") name: "U2FsdGVkX19Z9XoMTJLft5SIfqcXfQrPHRI502FJkCc=" surname: "U2FsdGVkX184mxp9m1ZkQBXZYb6oDW+VjshtFiBS98=" email: "U2FsdGVkX1/Rr60IDS5Wyc9k4A3lH8lqzN4gPVq437e8p9T+6ezUKqrt9/NetSxX" googleId: "U2FsdGVkX1/sKJLHVla2LXqPPelivzqWeX26viiGtqEDdxyj5MK47rNWhvhfZPl" phone: "U2FsdGVkX1+JSJ3IEdGeNcueCDwZAPVnVjXpNu8S7x4=" password: "" idCard: "" </pre>	
<pre> rentals _id: ObjectId("61d36ff29e1c0a3d0f1fe3c7") productId: "4" userId: "61d24c299f5076cb5b581473" startDate: "1641246706607" finishDate: "" price: "" status: "false" </pre>	<pre> company _id: ObjectId("61d0ae82e72f56076e46c1ef") phone: "555666777" email: "wypożyczalnia@wypożyczalnia.pl" title: "Wypożyczalnia "Projekt Inżynierski"" close: "20" open: "6" address: "Zakopane ul. Mickiewicza 20" percentage: "0" </pre>

Zrzut ekranu 5. Schemat bazy danych. Przykładowa wartość z każdej kolekcji. Źródło: opracowanie własne

W kolekcji przedmioty, numer elementu oprócz unikalnego identyfikatora generowanego automatycznie, jest również uzupełniony ręcznie przez pracownika wypożyczalni. Jest to związane ze specyfiką pracy wypożyczalni i możliwości wprowadzenia własnego kodu produktu np.: EAN. Jest to również rozwiązywanie które pozwala na podpięcie zewnętrznego skanera, uzupełniającego ten kod automatycznie i jeszcze bardziej ułatwiającego pracę wypożyczalni. Główną cechą bazy NoSQL jest elastyczność, więc pomimo, że kolekcje nie mają odgórnie przydzielonych relacji ze sobą, te zostały utworzone. Jako przykład można wskazać kolekcję rezerwacje, w której występuje relacja z przedmiotami oraz użytkownikami, tak więc konkretna rezerwacja jest przypisana do konkretnego użytkownika oraz konkretnego produktu.

2.5 Szyfrowanie danych

Podstawowym zabezpieczeniem podczas pracy z danymi osobowymi, podanymi przez klientów jest szyfrowanie. To właśnie ono odpowiada za bezpieczeństwo powierzonych przez osoby korzystające z wypożyczalni wrażliwych danych. Dane wysyłane przy pomocy ogólnodostępnej sieci, są bardzo narażone na złośliwe oprogramowanie, których celem jest odczytanie przesyłanych danych pomiędzy systemem a bazą danych. Samo połączenie jest jednak szyfrowane, ponieważ łączenie się z bazą danych następuje z wykorzystaniem protokołu HTTPS. Możliwości ataku przez użycie oprogramowania sniffującego jest więc ograniczone, jednak wciąż wskazane jest stosowanie kolejnych warstw bezpieczeństwa.

Wykorzystanym szyfrowaniem po obydwu modułach (webowym oraz desktopowym), jest AES (Advanced Encryption Standard, nazwa oryginalna: Rijndael, zmieniona po wyborze na międzynarodowy standard [8]), jest szyfrem symetrycznym. Do odszyfrowania i zaszyfrowania danych należy użyć tego samego tajnego klucza. Wybrany przez NIST (U.S. National Institute of Standards and Technology) w przeprowadzonym konkursie, jako standard szyfrowania danych 26 listopada 2001 roku [9]. Algorytm ten, w ostatnim czasie, został wybrany również przez NASK – Państwowy Instytut Badawczy, do zapewnienia bezpieczeństwa danych aplikacji mObywatel. Aplikacja wydana przez Kancelarię Prezesa Rady Ministrów ma na celu świadczyć obywatelom Rzeczypospolitej Polski podstawowe usługi potwierdzania tożsamości i uprawnień, poprzez zapewnienie dostępu do dokumentów, takich jak prawo jazdy, dowód osobisty czy unijny certyfikat covid. Jak opisuje portal gov.pl [10], zmiana na algorytm AES-256 została określona, jako użycie jeszcze silniejszego algorytmu szyfrowania danych. Brakuje informacji, jakiego algorytmu

zdecydowano się skorzystać jako poprzednik, jednak jako, że zmiana została opublikowana w grudniu 2021 roku, czyli w czasie pisania niniejszej pracy, można stwierdzić, że jest to technologia wciąż, nie tylko używana w obecnych czasach, ale i ciągle implementowana w nowych rozwiązańach.

W zbudowanym systemie, nie tylko hasło klienta jest szyfrowane, lecz wszelkie jego dane, w tym między innymi imię, nazwisko, telefon, numer dokumentu. Należy wskazać, że wszelkie z danych personalnych klienta są danymi wrażliwymi, i autorzy dołożyli wszelkich starań, aby dane te były bezpieczne. Po stronie webowej, autor zdecydował się skorzystać z pomocy biblioteki CryptoJS, czyli biblioteki zawierającej zbiór kryptograficznych algorytmów zaimplementowanych w języku JavaScript. Autor aplikacji desktopowej zdecydował się na użycie pakietów java Crypto [11] oraz Security [12], zapewniającej rozwiązania kryptograficzne, napisane w języku Java. Aby zapewnić spójność danych w bazie danych, należy, oprócz upewnienia się, że konfiguracja szyfrowania jest taka sama, zapisać na obu komputerach odpowiedni token, w miejscu do którego dostęp uzyskujemy w kodzie, jednak jest przechowywany lokalnie na komputerze programisty. Należy zwrócić uwagę, aby nie ujawnić tokenu podczas publikacji pracy w systemie kontroli wersji, jakim jest git.

3. Aplikacja webowa

Zgodnie z założeniami projektu aplikacja webowa powinna cechować się pięcioma cechami, jakimi są: intuicyjność, wygoda, responsywność, użyteczność i skalowalność. Intuicyjność jest wynikiem analizy upodobań użytkownika i odpowiednim rozmieszczeniu treści. Wygoda to zapewnienie takich rozwiązań jak logowanie się przy pomocy pojedynczego logowania (ang. *single sign-on, SSO*). Responsywność oznacza dostosowywanie się strony do każdego rozmiaru ekranu urządzenia. Przez użyteczność należy rozumieć pełny zakres obsługi klienta bezpośrednio przez aplikację, w tym kontakt z pracownikiem wypożyczalni. Skalowalność za to gwarantuje użycie zbioru najnowszych technologii i zapewnienie możliwości rozbudowy aplikacji w przyszłości.

3.1. Wymagania funkcjonalne

1. Rejestracja klienta z wykorzystaniem e-mail, w tym potwierdzenie poprawności poprzez wpisanie kodu wysłanego na skrzynkę, jak i poprzez hiperłącze w otrzymanej wiadomości.
2. Rejestracja klienta przy pomocy unikalnego kodu wygenerowanego przez pracownika wypożyczalni podczas wypożyczenia i zapewnienie możliwości zarządzania tym wypożyczeniem przez Internet. Użytkownik w tym przypadku musi uzupełnić jedynie dane do logowania.
3. Rejestracja klienta przy pomocy pojedynczego logowania. Klient musi uzupełnić jedynie dane osobowe.
4. Logowanie klienta do konta założonego w dowolny sposób, w tym wsparcie pojedynczego logowania.
5. Edycja danych osobowych klienta
6. Zmiana hasła oraz całkowite usunięcie konta klienta.
7. Przeglądanie dostępnych produktów na wynajem.
8. Filtrowanie produktów po wybranym czasie wynajmu i typie, w tym kalkulacja ceny wynajmu oraz ograniczenie godzin rezerwacji do godzin otwarcia wypożyczalni ustalanych w aplikacji desktopowej.
9. Rezerwacja produktu wraz z potwierdzeniem email.

10. Przegląd wszystkich dokonanych rezerwacji, zarówno aktualnych, odwołanych jak i archiwalnych
11. Anulowanie rezerwacji
12. Śledzenie stanu wypożyczenia, w tym pokazywanie naliczonej opłaty do momentu sprawdzenia, wraz z prezentacją sposobu naliczania opłat (godzinowy lub dobowy).
13. Wysłanie zapytania e-mail do pracownika wypożyczalni wprost z rezerwacji lub wypożyczenia, tak aby nie musieć podawać żadnych danych dotyczących usługi, jak i danych osobowych.
14. Zachowanie sesji użytkownika również po zamknięciu przeglądarki. Wylogowanie tylko z użyciem przycisku “Wyloguj”.
15. Przedstawienie danych kontaktowych do wypożyczalni, aktualizowanych na bieżąco z aplikacji desktopowej.
16. Responsywność i dostępność na urządzeniach mobilnych.
17. Obsługa routingu. Możliwość zapisania linku do wybranej usługi i przejście do strony tej usługi.

3.2. Wykorzystane technologie

3.2.1 Spis technologii

TypeScript - otwartoźródłowy (ang. *open source*) język programowania stworzony przez firmę Microsoft w 2012 roku. Jest nadzbiorem języka JavaScript i kompiluje się bezpośrednio do niego.

React - otwartoźródłowa, deklaratywna biblioteka języka JavaScript rozwijana przez Meta (dawniej Facebook) oraz społeczność. Pozwala na budowanie interfejsów użytkownika w oparciu o komponenty.

Material-ui - biblioteka zapewniając bazę komponentów korzystających z React. Pozwala na wykorzystanie gotowych komponentów oraz utworzenie całkowicie własnego systemu graficznego. Jest częścią specyfikacji wydanej przez Google podczas tworzenia Material Design, stosowanego w produktach tej firmy, w tym w systemie Android.

React Router - biblioteka zapewniająca routing, czyli umożliwiająca tworzenie adresów konkretnych stron, w tym dla konkretnej usługi.

EmailJS - biblioteka pozwalająca na połączenie z dowolnym klientem e-mail i wysyłanie wiadomości e-mail wprost z klienckiej części kodu JavaScript. Oznacza to, że nie jest potrzebny żaden serwer, aby wykonać tą czynność. Pozwala również na tworzenie szablonów wiadomości, które można personalizować przy użyciu wysyłanych parametrów w formie obiektu JSON.

Google OAuth Authentication - API od Google umożliwiające implementację pojedynczego logowania. Wymaga rejestracji aplikacji w portalu dla deweloperów Google [13] i przydzielenie numeru identyfikującego klienta.

Axios - klient HTTP działający w oparciu o obietnice (ang. *promises*). Obietnice w języku JavaScript to obiekty reprezentujące wynik asynchronicznych operacji i ich wartości.

npm - menedżer pakietów do języka JavaScript. Pozwala na dodawanie zarówno zewnętrznych bibliotek, jak i mniejszych komponentów. Tworzy największy na świecie rejestr oprogramowania (według danych z 2017 roku [14]).

HTML5 - język znaczników, służy do tworzenia struktury stron internetowych i prezentowania jej w sieci WWW

CSS - język kaskadowych arkuszy stylów, służy do opisywania prezentacji elementów napisanych w języku znaczników

Jest - otwartoźródłowa biblioteka do testowania kodu JavaScript, rozwijana przez Meta.

CryptoJS - biblioteka zapewniająca standardowe i bezpieczne algorytmy kryptograficzne zaimplementowane w języku JavaScript.

3.2.2 Uzasadnienie doboru technologii

Wybór Reacta jako podstawowej biblioteki frontendowej jest wynikiem obserwacji autora wzrostu jego popularności, aż do chwili obecnej, w której jest najczęściej używanym frameworkm frontendowym używanym na świecie. Łatwość w utrzymaniu kodu napisanego z jego pomocą jest powodem dla którego setki firm przepisuje od zera swój obecny kod na właśnie tą bibliotekę. Można to zauważyć na przykładzie znanego portalu społecznościowego Facebook, który zdecydował się w 2019 roku zrefaktoryzować całą aplikację webową, a pod koniec 2020 roku finalnie zakończyć testy i wdrożyć ją jako obowiązkową dla każdego użytkownika. W nowej wersji głównym używanym frameworkm jest właśnie React. Nowa szata graficzna została jednak uznana przez dziesiątki internautów, za dużo wolniejszą niż poprzednia oraz spotkała się z masowym niezadowoleniem. [15]. Autor pracy również, zauważył ten problem oraz w poszukiwaniu informacji przeszukał wiele stron i przeprowadził rozmowy z programistami pracującymi w zawodzie. Jako wniosek uznano, że prawdopodobnie głównym powodem jest właśnie łatwość w długoterminowym utrzymaniu kodu oraz łatwość wdrażania nowych funkcjonalności, która stoi w inżynierii oprogramowania wyżej niż doświadczenie użytkownika spowodowane wydajnością aplikacji. [16]. Utrzymanie i rozwijanie kodu jest bardzo kosztownym procesem dla firmy i minimalizacja czasu poświęcanego na tą czynność jest ważna z perspektywy biznesowej. Programiści również nie zawsze są chętni do utrzymywania kodu w starszych technologiach i posiadając w firmie nierozwijany kod zarówno współczynnik odejścia z firmy może być większy, jak i trudność w znalezieniu następcy. Komponentowa struktura plików w projekcie z wykorzystaniem React jest bardzo dobrze znana programistom i przypomina tworzenie stron internetowych w języku HTML, czyli technologii która jest właściwie najbardziej podstawowa w historii Internetu. Standard ten został zainicjowany 29 lat temu. Wykorzystujący podobny schemat programowania React powstał 8 lat temu, rewolucjonizując sposób w jaki piszemy aplikacje webowe. Powyższe fakty przyczyniły się do wyboru tej biblioteki jako głównej do częściowej do częściowej projektu.

Biblioteka React może być wykorzystana w projekcie używając zarówno języka JavaScript jak i TypeScript. Pierwszy został napisany w 1995 roku, jako język skryptowy, gdy zapotrzebowanie użytkowników internetu na wydajne aplikacje internetowe nie było tak wielkie jak jest w obecnych czasach. Powstały 9 lat temu TypeScript rozszerza jego możliwości nadając możliwość statycznego typowania. Jest to przydatne szczególnie podczas tworzenia obszernych aplikacji, lecz jego zalety można wykorzystać nawet tworząc małe projekty. Zapewnienie możliwości określania typów i interfejsów jasno określa oczekiwany typ danych w każdym miejscu

w kodzie, co minimalizuje ilość błędów w wywoływanych funkcjach oraz maksymalizuje zrozumienie pisanej kodu przez programistę. Wybór tego języka zamiast JavaScript został wybrany przez autora głównie poprzez chęć do samorozwoju i dążenia do tworzenia kodu o wysokiej jakości. Jego zalety zostaną również argumentowane na przykładach w kolejnych rozdziałach.

Pozostałe użyte biblioteki takie jak Material-ui jako biblioteka interfejsów użytkownika, React Router jako biblioteka zapewniająca routing oraz wiele innych zostały również wybrane ze względu na to, że są w grupie najpopularniejszych bibliotek stosowanych z Reactem. Posiadają one dokumentację na wysokim poziomie oraz użycie ich zapewnia większe możliwości rozwoju aplikacji w przyszłości

3.3 Wybrane aspekty implementacji

Aplikacja została w całości zaimplementowana w środowisku Visual Studio Code. Aplikacja od firmy Microsoft jest udostępniona na licencji MIT, czyli wolnego oprogramowania. Pozwala na wygodne tworzenie wysokiej jakości oprogramowania poprzez zapewnienie integracji z GitHub oraz dziesiątkami zewnętrznych dodatków, w tym wsparcia oprogramowania lintującego (ang. *linters*), czyli formatującego kod w ustalony w sposób. Zawiera również wiele dodatków wspierających ciągłe uruchamiania testów kodu.

W niniejszym rozdziale zostaną przedstawione najważniejsze aspekty implementacji, pozwalające czytelnikowi

- zrozumieć w jaki sposób statyczne typowanie wpływa na niezawodność pisanej kodu
- powiązać budowę aplikacji napisanej w React z prostą stroną napisaną w samym HTML
- przytoczyć zmiany jakie zostały wprowadzone do tej biblioteki w 2019 roku i wykorzystane podczas budowy aplikacji

Opis zostanie przeprowadzony na podstawie fragmentów kodu użytych w programie. Wprowadzone koncepcje zostaną również wykorzystane podczas szczegółowych opisów implementacji konkretnych funkcjonalności w rozdziale 3.4.

3.3.1 Statyczne typowanie na przykładzie komunikacji z API

Microsoft Visual Studio Code został pierwotnie napisany przy użyciu mieszanki JavaScript i TypeScript, ale - jak wspomina Benjamin Pasero, jeden z jego twórców - “szybko przyjęliśmy TypeScript dla całego naszego kodu i od razu przywiązałem się do jego ulepszonej obsługi narzędzi, takich jak statyczne sprawdzanie typów i refaktoryzacja.” [17]. Właśnie te atuty zdecydowały o wyborze tego języka do części webowej projektu. Każdy pojedynczy obiekt, który jest przesyłany w systemie ma utworzony interfejs definiujący jego strukturę. Na poniższym przykładzie widać interfejs pojedynczego elementu kolekcji *rezerwacje*.



```
export interface Reservation {
    productId: string;
    userId: string;
    startDate: string;
    finishDate: string;
    price: string;
    status: Status;
    _id?: string;
}

export enum Status {
    confirmed = "confirmed",
    cancelled = "cancelled",
}
```

A screenshot of a code editor showing two TypeScript files. On the left, an interface named 'Reservation' is defined with properties: productId, userId, startDate, finishDate, price, status, and an optional _id. The status property is typed as 'Status'. On the right, an enum named 'Status' is defined with two values: 'confirmed' and 'cancelled'. The code uses standard TypeScript syntax with export statements and curly braces for both the interface and the enum.

Fragment kodu 6. Interfejsy oraz enum użyty w języku TypeScript. Źródło: własne

Odpowiedź na zapytanie HTTP typu POST pod wygenerowany adres API z wybraną operacją *find* to właśnie tablica elementów o typie stworzonego interfejsu. Każda pojedyncza właściwość interfejsu może mieć określony własny typ. We fragmencie kodu 6, widać, że *status* może przyjmować jedynie dwie wartości określone przez typ enumeryczny *Status* jakimi są potwierdzona (ang. *confirmed*) i odwołana (ang. *cancelled*). Tak więc, w poprawnie zaimplementowanej aplikacji ze statycznym typowaniem niemożliwe jest ustalenie innego statusu niż dwa wcześniej wymienione. Jeśli w projekcie zostałby użyty język JavaScript, określenie oczekiwanej typu byłoby niemożliwe, ponieważ nawet jeśli określi się jego typ jako serię znaków, nie można wymagać jego weryfikacji pod kątem konkretnego ciągu znaków, tak jak to w języku TypeScript. Mając pod uwagę powyższe możemy dostrzec w jak dużym stopniu wykorzystanie tego języka zwiększa niezawodność pisanej części kodu, ale i ułatwia późniejsze jego zrozumienie przez innych programistów oraz ułatwia jego późniejszy rozwój.

Wykorzystanie statycznego typowania pozwala na tworzenie bezpiecznych funkcji, które można używać w wielu zastosowaniach. Za przykład może posłużyć funkcja *sendApiRequest*, która

zwraca obietnicę (ang. *promise*) o generycznym typie, który jest jednym z oczekiwanych interfejsów kolekcji pobieranych z API, na przykład wskazanym wcześniej interfejsie rezerwacji. Oznacza to, że jeśli jako odpowiedź dostaniemy inny typ danych zostanie zwrócony błąd.

```
export async function sendApiRequest({  
  collection,  
  operation,  
  body,  
  filter,  
  update,  
  setState,  
}: SendApiRequestProps): Promise<  
  Item[] | Reservation[] | User[] | Price[] |  
  CompanyInfo[] | Rental[] | string  
> {  
  
  interface SendApiRequestProps {  
    collection: string;  
    operation: CrudOperation;  
    body?: Reservation | User;  
    filter?: any;  
    update?: any;  
    setState?: (newState: any) => void;  
  }  
}
```

Fragment kodu 7. Nagłówek funkcji sendApiRequest. Źródło: opracowanie własne

Stanie się tak ponieważ w środku bloku zapewniającego się o powodzeniu operacji, to jest “try...catch”, zwracane jest “response.data.insertedId” lub “response.data.documents”. Aby zrozumieć dlaczego jest to oczekiwana odpowiedź należy poruszyć kwestię odpowiedzi na zapytanie HTTP. W przypadku zapytania z wybraną operacją *find* zwracany jest obiekt {“documents” : *Reservation*[]}. Z kolei w przypadku zapytania z operacją “insertOne” zwracany jest obiekt {“insertedId”: string}. Typ odpowiedzi jest regulowany przez MongoDB, jako twórców API. Jako deweloper korzystający z tego API należy dostosować swoją aplikację do tego formatu. Tak więc, jeśli zwrócony typ danych będzie inny niż tablica obiektów o typie *Reservation*, zwrócony zostanie błąd, analogiczna sytuacja będzie jeśli ustawiony zostanie zły typ akcji, na przykład *insertOne*, a zwrócony zostanie wcześniej wspomniany typ danych.

Headers	Payload	Preview	Response	Initiator	Timing
▼ {,...}	▼ documents: [{_id: "61d0ec7fb8b1812f19b0483e", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b0483e", length: 181, width: 8.5}, {_id: "61d0ec7fb8b1812f19b0483e", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b0483e", length: 181, width: 8.5}, {_id: "61d0ecbdb8b1812f19b04840", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b04840", length: 181, width: 8.5}, {_id: "61d0ecceb8b1812f19b04842", type: "snowboard", producer: "Salomon", model: "Snowboard Master 1812f19b04842", length: 181, width: 8.5}, {_id: "61d1010db8b1812f19b04843", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b04843", length: 181, width: 8.5}], status: 201, message: "Created", type: "object", subtype: "array", value: [{_id: "61d0ec7fb8b1812f19b0483e", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b0483e", length: 181, width: 8.5}, {_id: "61d0ec7fb8b1812f19b0483e", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b0483e", length: 181, width: 8.5}, {_id: "61d0ecbdb8b1812f19b04840", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b04840", length: 181, width: 8.5}, {_id: "61d0ecceb8b1812f19b04842", type: "snowboard", producer: "Salomon", model: "Snowboard Master 1812f19b04842", length: 181, width: 8.5}, {_id: "61d1010db8b1812f19b04843", type: "ski", producer: "Salomon", model: "Ski Master 1812f19b04843", length: 181, width: 8.5}]} <td></td> <td></td> <td></td> <td></td>				

Fragment kodu 8. Odpowiedź (ang. *response*) zapytania HTTP do kolekcji “items”. Widok w Chrome Developer Tools. Źródło: opracowanie własne

```

try {
  const response = await axios({
    method: "POST",
    url: `https://data.mongodb-api.com/app/data-lasjp/endpoint/data/beta/action/${operation}`,
    headers: {
      "Content-Type": "application/json",
      "api-key": apiKey,
    },
    data: JSON.stringify(requestData),
  });
  if (setState) {
    setState(true);
  }

  return operation === CrudOperation.CREATE
    ? response.data.insertedId
    : response.data.documents;
} catch (error) {
  console.log(error);

  if (setState) {
    setState(false);
  }
}
return [];

```

Fragment kodu 9. Blok “try...catch” funkcji sendApiRequest. Źródło: opracowanie własne

Niestandardową procedurą jaka jest zastosowana w MongoDB Data API jest również zalecenie wysyłania zapytań typu “POST”, który w tradycyjnym modelu “utwórz, odczytaj, aktualizuj, usuń” (ang. *Create, Delete, Update, Delete, CRUD*) powinien być używany tylko do zapytań w celu dodania rekordu do bazy. Decyzja ta jest związana z ograniczeniami w długości wartości oraz większym bezpieczeństwie danych. W metodzie *POST* dane są przesyłane w ciele zapytania *HTTP*, w metodzie *GET* są widoczne w linku *URL*. Maksymalna długość wartości w *GET* to 255 znaków, natomiast w *POST* nie ma żadnego limitu. W używanym API udostępniany jest link do bazy danych, więc aby wybrać akcję, należy ją dodać na końcu linku. Tutaj należy wskazać, że pomimo, że akcja to dowolny ciąg znaków, w zaimplementowanej funkcji operacja może mieć ścisłe określony typ, tak długo jak zgadza się z wyliczeniowym typem danych *CrudOperation*. Zastosowanie tego typu ogranicza możliwość popełnienia błędu podczas uzupełniania nazwy akcji przez programistę.

```
export enum CrudOperation {  
    CREATE = "insertOne",  
    READ = "find",  
    UPDATE = "updateOne",  
    DELETE = "deleteOne",  
    DELETE_MANY = "deleteMany",  
}
```

Fragment kodu 10. Enum CrudOperation. Źródło: opracowanie własne

Utworzona funkcja jest generyczna, więc dla każdego typu operacji modyfikuje dane przesyłane w zapytaniu. Jak widać na fragmencie kodu 7, argumenty przesyłane do funkcji zawierają operator opcjonalności, czyli znak zapytania przed dwukropkiem. Oznacza to, że w zależności od dokonywanej operacji deweloper może przesłać dowolny wybrany argument, jednak musi on odpowiadać rodzajowi wybranej akcji. Na fragmencie kodu 11 można zauważyć jak w zależności od wybranej akcji modyfikowany jest obiekt, który po przekonwertowaniu na ciąg znaków funkcją `JSON.stringify()`, wysyłany jest jako właściwość *data*, w obiekcie przesłanym do metody axios, w celu utworzenia zapytania HTTP (fragment kodu 8).

```
if (operation === CrudOperation.CREATE) {  
    requestData.document = body;  
}  
  
if (operation === CrudOperation.UPDATE) {  
    requestData.update = update;  
}  
  
if (  
    operation === CrudOperation.UPDATE ||  
    operation === CrudOperation.DELETE ||  
    operation === CrudOperation.DELETE_MANY  
) {  
    requestData.filter = filter;  
}
```

Fragment kodu 11. Dołączanie właściwości do zapytania HTTP. Źródło: opracowanie własne

3.3.2 Komponentowa struktura aplikacji

Tworzenie stron nazywanych w języku angielskim Single Page Application (SPA) jest wynikiem nowoczesnego podejścia do tworzenia aplikacji internetowych, które zaczęło powstawać około 2003 roku, a aktualnie stanowi główny model tworzenia witryn internetowych. Polega na operowaniu na tylko jednej podstawowej stronie bazowej i manipulowaniu jej elementami, które pojawiają się odpowiednio w warstwach. Aby zmienić wyświetlany konkretny element strony, wystarczy przeładować więc warstwę na której jest zawarty, a nie tak jak w przypadku tradycyjnych stron internetowych przeładowywaniu całej strony. Wiąże się to z dużo szybszym czasem działania oraz pozwala na pokazywanie symbolu ładowania treści, podczas gdy obsługujemy zapytania asynchronicznie i wynik nie jest zwracany natychmiast. Zaletą tego rozwiązania jest również możliwość zachowania statycznych elementów strony takich jak pasek nawigacyjny.

Do utworzenia aplikacji użyto jedną z bibliotek umożliwiających tworzenie aplikacji webowych w tym modelu. React, czyli biblioteka o której mowa, jest używana przez Facebooka, Twittera, Airbnb oraz wiele innych znanych stron. Zgodnie z dokumentacją biblioteki [13], w pierwszej z wymienionych, w użyciu jest ponad 50 tysięcy komponentów. W skład produktów Facebooka wchodzą również aplikacje pokrewne, jednak liczba ta pokazuje skalę, w jakiej aplikacje podzielone są na pojedyncze komponenty. Sam komponent w odniesieniu do Reacta jest to fragment kodu, zwykle w formie funkcji, który działa w izolacji i zwraca kod w formie znaczników HTML. Te ostatnie są to pewne słowa kluczowe, które użyte razem, tworzą treść i strukturę strony internetowej. Tagi mogą zawierać się w sobie i każdy tag będący rodzicem, definiuje sposób zachowania tagów będących jego dziećmi.

Żeby wyobrazić sobie samą logikę aplikacji działającej z użyciem tej biblioteki, można przytoczyć analogię tablicy korkowej, do której przypięta jest jedna wielka kartka A0, a wewnętrznych kartek kolejne, ale o coraz mniejszych formatach A1, A2, A3, A4 i tak dalej. Aby powiesić tą tablicę należy mieć do dyspozycji ścianę. Ta jest z kolei fragmentem pokoju, ten jest jedną ze składowych domu, który znajduje się w ogrodzie. W tym momencie można już odwołać się do analogii bazując na największych podstawach działania stron internetowych, które nie zmieniły się od początków sieci WWW. Plik tekstowy w formacie HTML, czyli element bezpośrednio przetwarzany przez przeglądarki internetowe może być porównany do ogrodu. Podstawowy plik html projektu nazywa się *index.html* i jest przedstawiony na fragmencie kodu 12.

```
<html lang="en">
  <head>
    ...
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Fragment kodu 12. Plik index.html. Źródło: opracowanie własne

Każdy typowy ogród posiada dom, tak jak każdy dokument HTML posiada znaczniki *html*. Wewnątrz znaczników *html* znajdują się różne znaczniki, na przykład *head* zawierający metadane strony, albo *body* który można porównać właśnie do omawianego pokoju. Będąc w pokoju można posiadać różne ściany, tak jak znacznik *body* może przyjąć różne znacznik. W tym miejscu widać różnice w stronie utworzonej z wykorzystaniem React. W tym wypadku, w odróżnieniu od tradycyjnych stron internetowych, tworzona jest jedynie jedna ściana, jako kontener w HTML (tag *div*) z oznaczeniem “korzeń” (ang. *root*). Nie zawiera on w sobie żadnej zawartości, ponieważ to zadaniem Reacta jest wkleić tam zawartość utworzoną przez dewelopera. Aby uzyskać do niej dostęp z pliku *index.tsx* (przedstawiony na fragmencie kodu 13) czyli pliku źródłowego całego katalogu komponentów, należy użyć metody biblioteki ReactDOM, która nazywa się *render*. Metoda ta renderuje główny komponent aplikacji do omawianego kontenera. Można założyć, że w omawianej analogii metoda *render* posłużyła jako metoda generująca tablicę korkową.

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Fragment kodu 13. Plik index.tsx. Źródło: opracowanie własne

W głównym komponencie możemy zawrzeć nieskończoną liczbę komponentów, w tym komponenty zawierające w sobie kolejne komponenty. Tutaj kończy się analogia do tablicy korkowej i coraz mniejszych przypiętych karteczek. Należy jednak mieć na uwadze, że liczba komponentów może być nawet pięciocyfrowa, jak to we wspomnianym Facebooku. W opisywanej aplikacji autor utworzył 25 własnych komponentów. Użyte biblioteki, w tym najliczniej, biblioteka interfejsu graficznego Material-ui, zawierają w sobie kolejne dziesiątki komponentów, tak więc

liczba, z której złożony jest cały projekt jest mimo wszystko kilku-kilkunastokrotnie razy większa. Liczba komponentów, nie definiuje jednak złożoności aplikacji, ponieważ struktura projektu zawsze zależy od wielu czynników. Największą zaletą Reacta jest możliwość tworzenia komponentów, które można używać ponownie. Tak więc, na przykładzie modułu uwierzytelniania, jaki zostanie przedstawiony w kolejnym rozdziale, komponent ten został użyty dwa razy w zupełnie innych miejscach. Często tworzenie wielu komponentów i rozdzielenie logiki na wiele plików upraszcza budowę aplikacji i zwiększa zrozumienie innych deweloperów, którzy czytają kod po raz pierwszy. Pozwala również na stosowanie jednolitych stylów we wszystkich panelach. W przedstawianej aplikacji utworzono *CustomContainer.tsx*, który stanowi podstawowy kontener treści strony. Po zmianie stylów dołączonych do tego komponentu zostanie zaktualizowana szata graficzna różnych, używających go, elementów. Został utworzony również komponent *AccessGuard.tsx*, który definiuje czy użytkownik na prawa dostępowe do otworzonej strony lub pokazujący efekt ładowania, gdy dane z API nie są jeszcze gotowe. Jest on przykładem komponentu, który jedynie dodaje pewną funkcjonalność do jego dziecka, więc trzeba mu przekazać całość kodu, który ma zabezpieczać jako właściwość nazwaną “dzieci” (ang. *children*) lub zdefiniować go na poziomie bliższym korzeniowi aplikacji. Ostatnim przykładem jest *CustomIcon.tsx*, czyli komponent zwracający ikonę odpowiadającą wskazanemu typowi sprzętu. Został on użyty 6 razy, więc stanowi najczęściej używany komponent wewnętrzny.

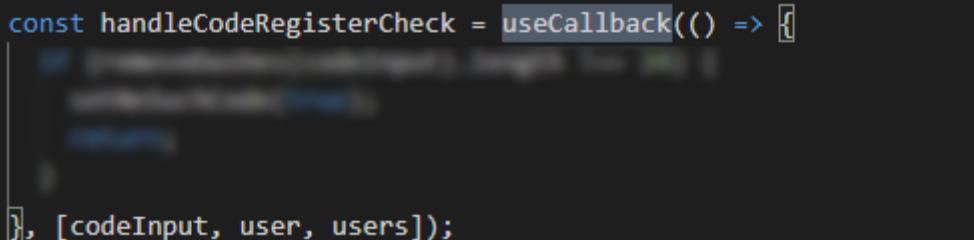
3.3.3 Zarządzanie stanem i optymalizacja kodu

React jest relatywnie nową biblioteką i nowe funkcjonalności są dodawane z każdym rokiem. Jak podaje dokumentacja [18] twórcy nie chcą dodawać zbędnego kodu biblioteki, który nie jest niezbędny i ograniczają się do elementów, bez których ludzie tworzyliby własne rozwiązania. Powstawałaby niespójność pomiędzy różnymi aplikacjami wykorzystującymi tą bibliotekę w tym samym celu. Nie inaczej stało się w ostatnim czasie, kiedy w roku 2019, wraz z wersją 16.8, wprowadzono hooki (ang. *hooks*). Całkowicie zreformowały one sposób w jaki pisze się kod z wykorzystaniem tej biblioteki, ponieważ dały możliwość zarządzania stanem i innymi funkcjonalnościami bez pisania klas, przy okazji dostarczając łatwy sposób pisania własnych hooków. Technicznie jest zwykłą funkcją, która zaczyna się od prefiku *use* i może używać innych hooków w swoim ciele oraz zawiera logikę, która może być używana w różnych komponentach.

Podczas tworzenia aplikacji skorzystano z wielu hooków. Najważniejsze z nich dostarczone są bezpośrednio od React, ale nie brakuje też hooków, które zostały wprowadzone po publikacji

omawianej wersji, przez pomniejsze biblioteki. Za przykład można podać *useParams* od biblioteki ReactRouter, pozwalający na przechwycenie parametrów trasy w ścieżce URL. W aplikacji oznacza to, że po przekierowaniu do strony *http://localhost:3000/reservation/id*, zostanie zwrócony wpisany id bezpośrednio do komponentu, który jest odpowiedzialny za tą trasę i pozwoli na wyświetlenie odpowiedniej treści. Najważniejszym hookiem, który jest używany w praktycznie każdym komponencie jest *useState* pozwalający na wprowadzenie lokalnego stanu do komponentu, który nie zmienia się podczas ponownego renderu tego komponentu. Zachowanie jest dokładnie przeciwnie, ponieważ właśnie gdy stan się zmieni, komponent jest ponownie renderowany. Pokazuje to jak ważne jest korzystanie z takich hooków jak *useCallback* oraz *useMemo*. Oba działają podobnie, lecz adresatem pierwszego są funkcje, a drugiego dowolne wartości. Zwracają zapamiętaną wersję funkcji bądź wartości tak dugo, jak wartości w tablicy zależności nie zmienią się. Działanie hooka zostało pokazane we fragmencie kodu 14. Oznacza to, dużą oszczędność w ilości wykonywanego kodu, a co za tym idzie, w większości przypadków, gwarantuje szybsze działanie programu.

```
const handleCodeRegisterCheck = useCallback(() => [
```



```
], [codeInput, user, users]);
```

Fragment kodu 14. Wykorzystanie *useCallback*. Źródło: opracowanie własne

Stan może zostać zmieniony w samym komponencie, lub funkcja odpowiadająca za zmianę stanu może zostać przekazana jako właściwość do komponentu dziecka. Ostatnim omawianym hookiem, który został użyty wiele razy w projekcie jest *useEffect*. Reguluje on akcje, jakie mają się wykonać jako efekt uboczny zmiany jednego z elementów w tabeli zależności. W projekcie został użyty między innymi do przekierowania do strony usług, jeśli asynchroniczna funkcja odpowiedzialna za kontakt z API zwróciła prawidłową odpowiedź, lub pokazania informacji o błędzie, jeśli operacja się nie powiedzia. Funkcja, której deklaracja została pokazana we fragmencie kodu X1, przyjmuje jako argument *setState*, czyli funkcję ustawiającą nowy stan, zdefiniowaną jako *(newState: any) => void*. Używając więc funkcji zmieniany jest stan podany jako argument w wywołaniu funkcji i możliwe jest oczekiwanie na zmianę tego stanu przy użyciu *useEffect*.

3.4 Funkcjonalności

Aplikacja spełnia wszystkie postawione wymagania funkcjonalne. Aby przybliżyć działanie programu, w rozdziale zostały opisane główne funkcjonalności. Nie brak również szczegółowych opisów implementacji, które wykorzystują opisane w poprzednim rozdziale koncepcje i pokazują wykorzystanie ich w praktyce.

3.4.1. Możliwości rejestracji i uwierzytelniania

We wstępie zaznaczone zostało, że głównymi celami projektu po stronie klienckiej jest zapewnienie intuicyjności, wygody, responsywności, skalowalności i użyteczności. Umożliwienie logowania za pomocą pojedynczego logowania niepodważalnie zwiększa drugi wyznaczony cel. Za jego pomocą użytkownik może zalogować się do systemu jednym kliknięciem i oszczędza czas na wpisywanie adresu e-mail i hasła, co bezpośrednio przyczynia się do odczuwalnej wygody podczas rezerwacji. Rozwiązań dostarczanych przez gigantów technologicznych z jednej strony mogą przyczynić się do zwiększenia bezpieczeństwa, ponieważ zwykle oferują uwierzytelnianie wielopoziomowe, takie jak potwierdzenie logowania na urządzeniu, czy też weryfikację kodu wysłanego drogą SMS oraz inne metody takie jak potwierdzenie tożsamości za pomocą twarzy przez FaceID. Z drugiej strony, jeśli użytkownik nie korzysta z tych metod weryfikacji mogą przyczynić się do osłabienia bezpieczeństwa aplikacji, ponieważ podczas wycieku hasła czy też tokenu dostępowego, oszust jest w stanie zalogować się do aplikacji i uzyskać dane osobowe klienta. Należy jednak zauważyc, że powszechnie jest jednak używanie przez użytkowników tego samego hasła w wielu portalach, więc w takim scenariuszu zagrożenie to jest zbliżone w obu sposobach logowania. Biorąc pod uwagę powyższe, zdecydowanie zasadne jest edukowanie użytkowników, aby ustawiali inne hasło dla każdej witryny, w zapamiętaniu którego może przyjść z pomocą menedżer haseł, lub też używali wielopoziomowej weryfikacji przy każdym logowaniu przez pojedyncze logowanie. Należy również zauważyc, że zawsze powinniśmy wymagać również wpisania adresu e-mail użytkownika, lub też potwierdzenia adresu pobranego z dostawcy usługi, tak aby umożliwić zmianę hasła, gdy użytkownik utraci dostęp do strony trzeciej obsługującej pojedyncze logowanie. Przykładowa sytuacja, która obejmuje taki przypadek, to założenie konta przez Facebook. Gdy użytkownik utraci dostęp do konta w serwisie społecznościowym musi wciąż mieć dostęp do zresetowania hasła przez e-mail podany przy rejestracji.

Panel uwierzytelniania został utworzony jako jeden główny komponent o nazwie *Auth* i znajduje się w pliku Auth.tsx. Został on użyty w dwóch miejscach. Sposób ich prezentacji został pokazany w Tabeli 2.

	akcja użytkownika	sposób prezentacji
1	wciśnięcie przycisku “Zaloguj” w prawej części paska nawigacyjnego	element rozwijany z paska nawigacyjnego
2	niezalogowany użytkownik wyszukał produkt i przeszedł do potwierdzenia rezerwacji	element pokazywany jest w komponencie potwierdzenia rezerwacji w pliku ReservationConfirmation.tsx (przedstawiony na zrzucie ekranu 6)

Tabela 2. Sposób prezentacji panelu uwierzytelniania. Źródło: opracowanie własne

Potwierdzenie rezerwacji



Snowboard

Salomon Snow Master X7

rozmiar: 1300

Odbiór: 14.01.2022 (16:21)

Koniec: 15.01.2022 (15:21)

25 zł

Cena za 1 dni wynajmu.(stawka dzienna)

KONTYNUJ Z UŻYCIEM KONTA GOOGLE

Uzupełnij dane aby utworzyć nowe konto

Masz już konto?

ZALOGUJ

Zarejestruj z użyciem kodu z wypożyczalni

ZAREJESTRUJ

ZAREJESTRUJ Z UŻYCIEM KODU

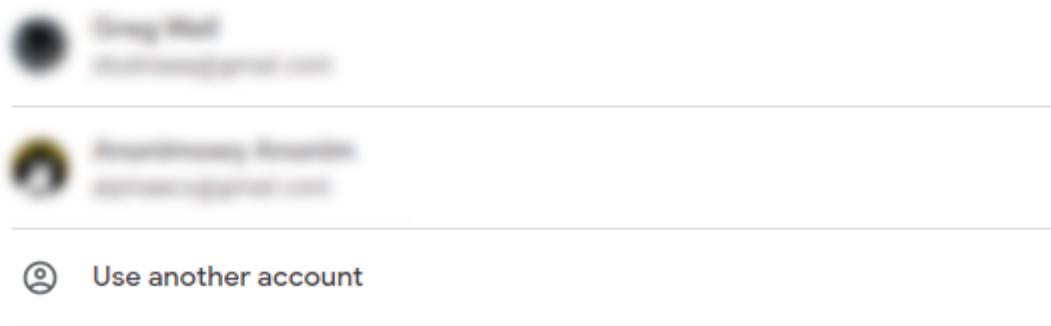
[WRÓĆ DO WYNIKÓW WYSZUKIWANIA](#)

Zrzut ekranu 6. Sposób prezentacji 2. Źródło: opracowanie własne

Najłatwiejszą metodą potwierdzenia tożsamości lub założenia konta w portalu jest pojedyncze logowania. Po naciśnięciu przycisku “Zaloguj przy pomocy konta Google” otworzy się nowe okno, którego treść jest dostarczana od tej firmy. Jeśli użytkownik nie będzie zalogowany do żadnego konta będzie musiał zweryfikować swoją tożsamość, w tym nierzadko użyć dwuetapowej weryfikacji. Jeśli użytkownik jest już zalogowany zostanie poproszony o wybór konta. Dialog wyboru konta został przedstawiony na Zrzucie Ekranu 7.

Choose an account

to continue to [System Obsługi Wypożyczalni](#)



Zrzut ekranu 7. Wybór konta Google. Źródło: opracowanie własne

Po poprawnym logowaniu przez Google, aplikacja odczyta numer identyfikujący konto Google i sprawdzi w bazie danych czy użytkownik powiązany z tym kontem już istnieje. Jeśli tak, zostanie zalogowany i będzie mógł korzystać z konta, w tym dokończyć rozpoczętą rezerwację. Jeśli jest to jednak pierwsze logowanie tego użytkownika wyświetli się formularz uzupełnienia danych osobowych. E-mail klienta, imię i nazwisko zostanie pobrane z odpowiedzi od Google. Dane osobiste wymagają potwierdzenia, ponieważ czasem ludzie używają nieprawidłowych danych w celu zachowania anonimowości w Internecie. Zostanie również pokazana prośba o uzupełnienie numeru telefonu. Może to być przydatne w przypadku nagłych sytuacji, w której pracownik wypożyczalni może chcieć się skontaktować z klientem. Nie jest wymagane jednak ustawianie hasła, a to czyni logowanie przez Google najbardziej przystępna wersją. Formularz uzupełnienia danych został pokazany na Zrzucie Ekranu 8.

Wygląda na to, że to Twoje pierwsze logowanie przy pomocy Google. Uzupełnij poniższe dane by kontynuować!

E-mail

Imię

Nazwisko

Telefon

ZAREJESTRUJ

Zrzut ekranu 8. Formularz uzupełnienia danych osobowych. Źródło: opracowanie własne

W obu scenariuszach aktualnie rozpoczęta rezerwacja nie będzie utracona po udanym uwierzytelnieniu. Sukces będzie oznaczał ukrycie panelu z pliku Auth.tsx, a pokazanie w zamian przycisku “Rezerwuję”. Zmieniony komponent został ukazany na Zrzucie Ekranu 9.

630 zł

Cena za 9 dni wynajmu.(stawka dzienna)

REZERWUJĘ

Zrzut ekranu 9. Przycisk “Rezerwuję” po udanym uwierzytelnianiu. Źródło: opracowanie własne

Kolejną metodą rejestracji jest wpisanie kodu z wypożyczalni. Każdy użytkownik utworzony w aplikacji desktopowej posiada swój unikalny identyfikator. Jeśli pracownik wypożyczalni przekaże mu ten identyfikator podczas wypożyczania sprzętu, użytkownik może skorzystać z tej metody logowania. Przykład kartki, z numerem do rejestracji, został pokazany na Zrzucie Ekranu 10. Z kolei na Zrzucie Ekranu 11 został przedstawiony formularz rejestracji, przy pomocy kodu, znajdujący się w komponencie *Auth*.

Dziękujemy za skorzystanie z naszych usług!

Stan swojego wypożyczenia możesz sprawdzić na naszej stronie <[link do strony](#)>

Użyj kodu 61d6fa - 762202 - 15e6d9 - 90ab11 aby się zalogować!

Zrzut ekranu 10. Przykład informacji o założonym koncie klienta, który można wręczyć klientowi.

Źródło: opracowanie własne

Zarejestruj z użyciem kodu z wypożyczalni

24-cyfrowy unikalny kod z wypożyczalni

61d6fa-762202-15e6d9-90ab11

ZAREJESTRUJ Z UŻYCIEM KODU

Zrzut ekranu 11. Fragment strony odpowiedzialny za rejestrację z użyciem kodu. Źródło: opracowanie własne

Klient zostanie w tym wypadku poproszony o uzupełnienie danych do logowania takich jak email i hasło. Wszelkie dane osobiste zostaną automatycznie pobrane z bazy danych, która została uzupełniona przez pracownika wypożyczalni. Formularz uzupełnienia danych został pokazany na Zrzucie Ekranu 12. Po udanym zalogowaniu klient będzie mógł sprawdzić stan swojego aktualnego wypożyczenia, w tym aktualnie nalecone opłaty oraz korzystać z konta w przyszłości by dokonywać rezerwacji.

Hej Jan Kowalski ! Utwórz hasło do konta aby kontynuować!

E-mail

Hasło

Powtórz hasło

ZAREJESTRUJ Z UŻYCIEM KODU

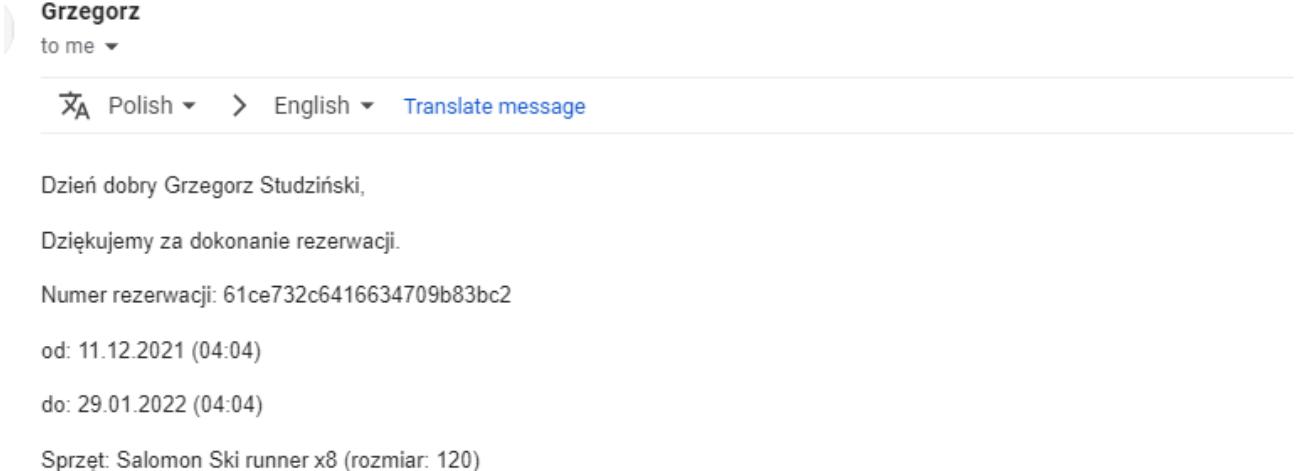
Zrzut ekranu 12. Formularz uzupełnienia danych do logowania przy rejestracji z użyciem kodu. Źródło: opracowanie własne

Na koniec należy wspomnieć o zapewnieniu możliwości logowania oraz rejestracji w tradycyjny sposób, czyli poprzez adres e-mail. W tym przypadku ważna jest jego poprawność, więc zapewnione zostało jego potwierdzenie. Więcej o tym sposobie weryfikacji zostanie przedstawione w opisie tej funkcjonalności.

3.4.2. Obsługa wiadomości e-mail.

Aplikacja zapewnia pełne wsparcie w dokumentowaniu operacji dokonanych przez klienta poprzez potwierdzenia w formie wiadomości e-mail. Na Zrzucie Ekranu 13 został przedstawiony przykładowy e-mail wysyłany na adres klienta, po utworzeniu rezerwacji w systemie.

Salomon Ski runner x8 (rozmiar: 120) - potwierdzenie rezerwacji  



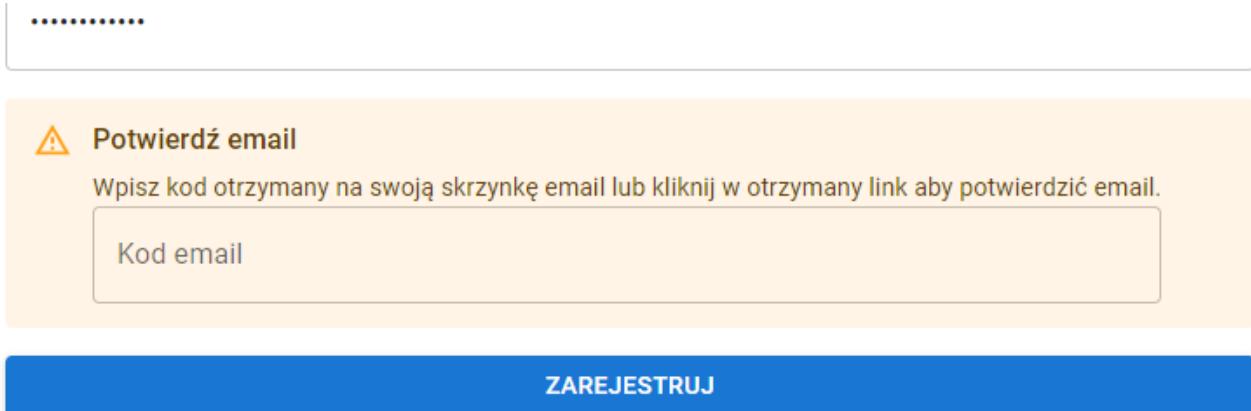
Grzegorz
to me ▾

Polish ▾ > English ▾ Translate message

Dzień dobry Grzegorz Studziński,
Dziękujemy za dokonanie rezerwacji.
Numer rezerwacji: 61ce732c6416634709b83bc2
od: 11.12.2021 (04:04)
do: 29.01.2022 (04:04)
Sprzęt: Salomon Ski runner x8 (rozmiar: 120)

Zrzut ekranu 13. Format wiadomości e-mail - potwierdzenie rezerwacji. Źródło: opracowanie własne

Jak zostało wspomniane w podrozdziale o uwierzytelnianiu, wymagane jest potwierdzenie adresu e-mail przy rejestracji użytkownika. Po uzupełnieniu wszystkich danych ukazywany jest monit o wymaganym wpisaniu kodu e-mail lub też kliknięciu w adres wysłany w wiadomości. Formularz manualnego wpisania kodu został przedstawiony na Zrzucie Ekranu 14. Z kolei na Zrzucie Ekranu 15 przedstawiono wiadomość e-mail, jaką użytkownik otrzymuje, po wpisaniu danych osobowych do konta.



Zrzut ekranu 14. Monit wymagający od użytkownika potwierdzenia e-mail. Źródło: opracowanie własne

Dzień dobry,

Aby potwierdzić email wpisz kod: 0728e9

Możesz też kliknąć na link: <http://localhost:3000/confirmEmail/U2FsdGVkX1c6cZB27TGJ8l6kUrVqli1vuZMmSz6jpJylUzb+tj5j7j5c9kPww+gFif8ENVFubs5X6jGXStemporarySolutionxaNfME3EGDShPJvWi8SRnxgQ9T7ApAF3Q0tjjrp82Z7T674HxyRlpfm4+AOsuyA==>

Jeśli nie zakładałeś konta w naszym portalu, zignoruj tą wiadomość.

Zrzut ekranu 15. Format wiadomości potwierdzenia e-mail. Źródło: opracowanie własne

Wygenerowany, sześciocyfrowy kod ustawiany jest jako stan lokalny komponentu. Jeśli użytkownik wprowadzi otrzymany kod ręcznie, komponent sprawdzi jego poprawność i pokaże odpowiedni komunikat. Link potwierdzający adres e-mail działa w odmienny sposób. Zawiera w sobie skonwertowany do ciągu znaków i zaszyfrowany obiekt JSON zawierający wprowadzone dane użytkownika. Po kliknięciu w podany link zostanie uruchomiony komponent *ConfirmEmail*, którego zadaniem jest deszyfracja obiektu. Jeśli obiekt będzie poprawny użytkownik zostanie utworzony w bazie danych, zostanie pokazany pozytywny komunikat (Zrzut ekranu 16) oraz pasek nawigacyjny zmieni się w menu panelu klienta. Jeśli w linku zostanie zmieniona choć jedna litera, wynik deszyfracji będzie negatywny, więc zostanie zwrócony komunikat o niepoprawnej operacji.

 Adres email został zweryfikowany.

Możesz już korzystać ze swojego konta.

Zrzut ekranu 16. Powiadomienie o poprawnej weryfikacji adresu email. Źródło: opracowanie własne

Jedną z dodatkowych możliwości jest zapewnienie możliwości kontaktu z pracownikiem wprost z dokonanej rezerwacji. Dzięki wprowadzonej opcji, klient nie musi szukać adresu email wypożyczalni oraz opisywać numeru czy terminu wypożyczenia aby uzyskać pomoc od pracownika wypożyczalni. Intuicyjność rozwiązania polega na tym, że klient wystarczy, że otworzy dokonaną rezerwację i od razu w tym samym miejscu ma możliwość wprowadzenia wiadomości. Po wpisaniu i wcisnięciu wyślij przez klienta, wysyłany jest e-mail na adres wypożyczalni, który można zmienić w ustawieniach aplikacji desktopowej. W treści wiadomości zawarte są najważniejsze informacje odnośnie rezerwacji, takie jak jej numer, id klienta, termin początku i końca, typ sprzętu i id sprzętu. Pracownik wypożyczalni może odpowiedzieć bezpośrednio na wiadomość e-mail, widząc treść zapytania od klienta. Zastosowanie takiej metodologii jest wygodne zarówno dla klienta, jak i dla pracownika wypożyczalni, ponieważ zarządzanie skrzynką e-mail jest bardzo uniwersalne i pracownik wypożyczalni może mieć dostęp do niej nawet z urządzenia mobilnego, co czyni odpowiadanie na prośby klientów wyjątkowo wygodne. Zarówno klient jak i pracownik będą mieli w tytule rozmowy typ sprzętu oraz datę rezerwacji więc łatwo będzie powiązać rozmowę z konkretną usługą. Format wiadomości, jaka trafia na adres wypożyczalni została przedstawiona na zrzucie ekranu 17.

Zapytanie klienta [28.01.2022 (17:57)] [Specialized Road Master 11 (rozmiar: 52)]



✉️ [Inbox](#) ✖️



Grzegorz

to me ▾

Sat, 1 Jan, 18:40 (5 days ago)



Klient wysłał następujące zapytanie:

Dzień dobry, Czy zamówienie zawiera kask

Numer rezerwacji: 61d0881a9f5076cb5b85ceab

Sprzęt: Specialized Road Master 11 (rozmiar: 52)

Daty: od 28.01.2022 (17:57) do 29.01.2022 (17:57)

Klient: Jane Anonim (id: 61d076dc639e7eb85e101afc)

Zrzut ekranu 17. Format wiadomości wysyłanej na adres wypożyczalni. Źródło: opracowanie własne

3.4.3. Edycja własnego profilu.

Klient po zalogowaniu może dokonać edycji swojego profilu. Z danych osobistych wyświetlane są imię, nazwisko, numer telefonu oraz adres e-mail. Operacja zmiany danych wysyła zapytanie do bazy danych ze znacznikiem akcji *updateOne*. Rezultat odpowiedzi z API decyduje o pokazanym użytkownikowi komunikacie. Oznacza to, że jeśli w czasie edycji danych utracone zostanie połączenie z Internetem, zostanie pokazany komunikat o negatywnym statusie operacji, a strona nie zmieni swojego interfejsu i nie wygaśnie. Jeśli użytkownik zapewni połączenie ponownie, kolejny komunikat będzie o pozytywnym zakończeniu operacji. Możliwa jest również zmiana hasła oraz całkowite usunięcie konta. Ostatnia akcja poprzedzona jest informacją, że operacji nie można cofnąć, dopiero ponowne naciśnięcie przycisku usuwa konto oraz wszystkie przypisane do niego rezerwacje w systemie.

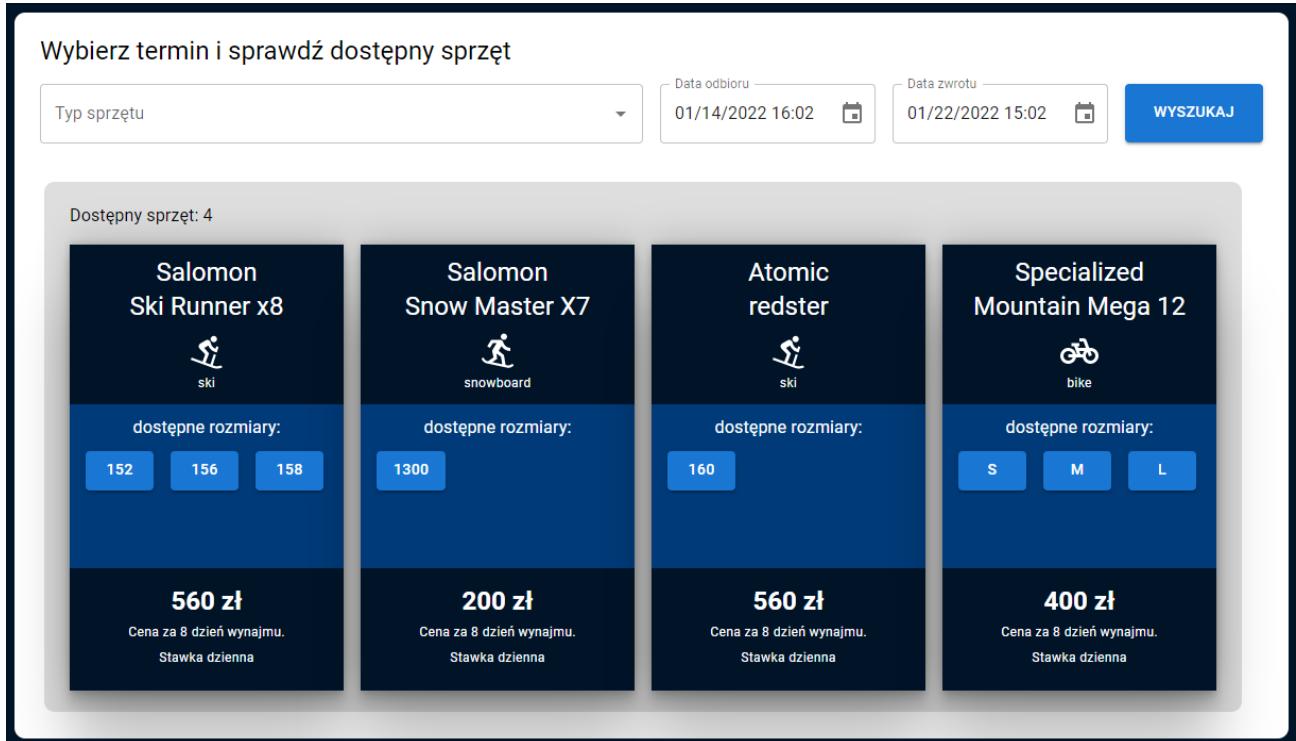
Edycja profilu

Dane osobowe		Zmiana hasła
Imię	Jan	Obecne hasło
Nazwisko	Kowalski	Nowe hasło
Numer telefonu	500200100	Powtórz hasło
Adres email	alpinaeco@gmail.com	ZMIEN HASŁO
AKTUALIZUJ		Tej akcji nie możesz cofnąć. Jesteś pewny/na?
		USUŃ KONTO

Zrzut ekranu 18. Interfejs graficzny komponentu *MyProfile*. Źródło: opracowanie własne

3.4.4. Przeglądanie produktów i rezerwacja

Jak zostało wspomniane we wcześniejszym podrozdziale, zarówno niezalogowany jak i zalogowany klient może wyszukać dostępne produkty. Zostały wdrożone pewne mechanizmy zapewniające wybór poprawnej daty przez użytkownika, takie jak możliwość wyboru wypożyczenia tylko w godzinach pracy wypożyczalni, czy wybór daty zwrotu po dacie odbioru. Główny komponent to *ReservationPanel*, a komponent obsługujący wybór daty to *ReservationDateTimePicker*, stanowiący odpowiednio dostosowany przez autora komponent z biblioteki Material-ui. Użytkownik może wybrać typ sprzętu, jaki go interesuje, a finalnie może wybierać z dostępnych rozmiarów. Cena jest kalkulowana w dwóch kategoriach. W trybie godzinnym oraz w trybie dobowym. Pokazywana i naliczana opłata jest równa bardziej opłacalnemu trybowi dla klienta. Omawiany panel został pokazany na Zrzucie Ekranu 19.



Zrzut ekranu 19. Interfejs graficzny komponentu *ReservationPanel*. Źródło: opracowanie własne

Aby w pełni umożliwić czytelnikowi zrozumienie, w jaki sposób dostępne produkty zostały wyświetlane w panelu rezerwacyjnym, a następnie w jaki sposób zostało wystawione potwierdzenie rezerwacji, szczegółowo opisano cały proces. Zasadne jest również zwrócenie uwagi na komponentową strukturę projektu, która została omówiona w rozdziale 3.3.2.

Na Diagramie 2 został przedstawiony uproszczony schemat zależności pomiędzy komponentami, zawartymi w opisie. Należy mieć na uwadze, że stanowią one tylko ułamek komponentów jakie są w nich zawarte. Na diagramie użyto również oznaczeń czarną strzałką, które oznaczają po kolei (patrząc od lewej):

- 1) zmianę stanu *isShowingReservationForm*, powodującą wyświetlenie drugiego komponentu wewnątrz *ReservationPanel*.
- 2) wywołanie funkcji *navigate*, biblioteki *ReactRouter* w celu przekierowania do trasy */services*.

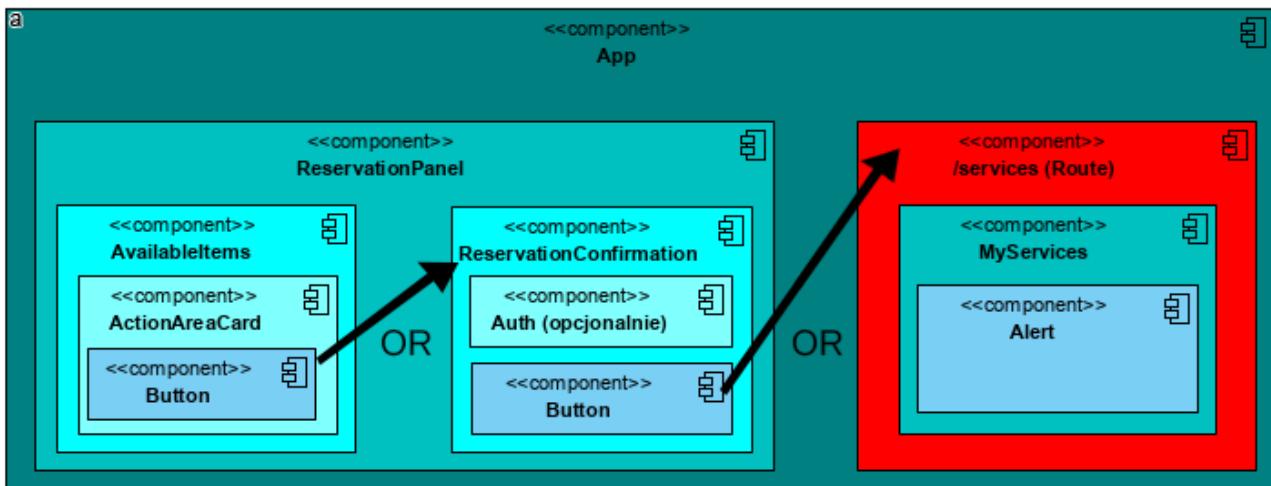


Diagram 2. Uproszczony schemat komponentów użytych w procesie. Źródło: opracowanie własne

Główny komponent *App*:

1. Po uruchomieniu strony wysyłane jest zapytanie POST do API z wybraną akcją *find* oraz kolekcją *items*. Używana jest funkcja *sendApiRequest* omówiona w rozdziale 3.3.1.
2. Zakładając, że odpowiedź z serwera będzie pozytywna, to jest, kod statusu HTTP powinien być równy 200, zostanie zwrócona tablica, w tym wypadku obiektów o interfejsie *items*.
3. Wynik wykorzystanej funkcji, będący obietnicą, przypisywany jest do nowej zmiennej *fetchedItems*. Gdy asynchroniczne zapytanie zostanie zakończone obietnica zmieni swój status na spełniony, a jej wartość będzie stanowić zwrócona tablica obiektów o interfejsie *Item[]*.
4. Hook *useState*, omówiony w podrozdziale 3.3.3 został uprzednio wykorzystany do przygotowania stanu dla kolekcji *items*. Stan ten nosi nazwę *items*, a funkcja ustawiająca ten stan to *setItems*. Używana jest więc funkcja *setItems*, z argumentem *fetchedItems*.
5. W wyniku działania funkcji *setItems*, która również jest asynchroniczną funkcją, lokalny stan komponentu *App*, w typ wypadku, *items* ma już wartość równą kolekcji *items* w bazie danych.

Komponent *ReservationPanel*

6. Odczytanie wartości *items*. React pozwala przesyłać wartości od komponentu rodzica do komponentu dziecka. Przesłany został więc, jako jeden z wielu, stan *items*, pobrany w poprzednich krokach.
7. Omówiony w podrozdziale 3.3.3. komponent *AccessGuard* odpowiedzialny jest za pokazywanie ikony ładowania, dopóki dane z API nie są gotowe. Między innymi właśnie

- zmiana wartości *items* przyczynia się do pokazania panelu rezerwacyjnego, pokazanego na zrzucie ekranu K2.
8. Stan *items* jest przypisywany do nowej zmiennej i całość jest grupowana po rozmiarach każdej unikalnej pary, składającej się z producenta i modelu.
 9. Pogrupowane obiekty są filtrowane i z każdego rozmiaru danego produktu, pozostawiany jest procent sprzętu, który został ustawiony w aplikacji desktopowej jako procent sprzętu przeznaczonego do rezerwacji on-line. Z pozostałych obiektów tworzona jest nowa tablica.
 10. Pozostałe obiekty są filtrowane, by usunąć produkty, które posiadają rezerwację w wybranym terminie.
 11. Ostatecznie, obiekty które są przeznaczone na wynajem są ponownie grupowane, lecz tym razem jedynie po samej parze producent i model, bez rozmiaru. Tak przygotowane obiekty trafiają do komponentu *AvailableItems*.
 12. Należy wspomnieć, że również w tym miejscu zostanie obliczona cena wynajmu danego modelu w wybranych danych. Wiąże się to z obsługą innego zapytania HTTP i pobraniem cennika, który ustawia pracownik wypożyczalni, z bazy danych. Aby uprościć, i tak już skomplikowany, proces dokonywania rezerwacji, wątek ten zostanie pominięty.

Komponent *AvailableItems*:

13. Tablica obiektów przekazana z poprzedniego komponentu jest mapowana po kluczu, to jest, dla każdej pary producent i model, zwracany jest komponent *ActionAreaCard*.
14. Jako właściwość każdego komponentu przekazywana jest tablica zawierająca różne rozmiary danego modelu. Technicznie, tablica jest filtrowana z każdego rozmiaru, który się w niej powtarza, zostawiając pierwszą znalezioną sztukę danego rozmiaru, z konkretnym numerem id.
15. W komponencie utworzony jest również lokalny stan definiujący aktualnie wybrany produkt oraz stan definiujący, czy panel potwierdzenia rezerwacji jest pokazywany. Funkcja zmieniająca oba stany jest również przesyłana do komponentu *ActionAreaCard*.

Komponent *ActionAreaCard*

16. Zwracany jest kontener z opisem danego produktu oraz dla każdego rozmiaru generowany jest przycisk. Kliknięcie go, wywołuje wspomnianą w punkcie 15 funkcję, efektem czego jest pokazanie komponentu *ReservationConfirmation*.

Komponent *ReservationConfirmation*

17. W uproszczeniu: jeśli użytkownik jest uwierzytelniony, kliknięcie przycisku “Rezerwuję” uruchomi funkcję, która wykona następujące czynności.

- Utworzy spójny z bazą danych obiekt JSON, bazując na stanach wybranego produktu, ceny i daty rezerwacji
- Zapisze w zmiennej wynik asynchronicznego wywołania funkcji *sendApiRequest*. Tym razem jako argumenty zostaną podane: kolekcja to *reservations*, operacja to *insertOne*, ciało (ang. *body*) równe sworzonemu plikowi oraz *setState*, jako *setNewReservationSuccess*. Tutaj należy odwołać się do funkcji omawianej w rozdziale 3.3.1. Pierwszą rzeczą jest dodanie ciała, które jest opcjonalne i należy je dodać jedynie gdy wykonujemy akcję *insertOne*. W rozdziale wspomniano, że jeśli do zapytania zostanie dodana właściwość nieodpowiednia do wybranej akcji, zostanie zwrócony błąd. Kolejną rzeczą jest ustawienie stanu definiującego sukces operacji. Właśnie do tego konkretnego przypadku, czyli dodawania rezerwacji autor odniósł się w opisie działania funkcji.
- Wynikiem wstawienia rezerwacji do bazy danych będzie *insertedId*, czyli numer id nowo utworzonej rezerwacji.
- Funkcja jest odpowiedzialna również, za ustawienie lokalnego stanu *currentReservation*, na utworzony obiekt JSON, po uprzednim dodaniu *insertedId* jako *_id*.

18. W tym momencie w praktyce zaobserwować można jaką funkcję pełni hook *useEffect*, opisany w rozdziale 3.3.3. Jego funkcja została opisana w następujący sposób: “reguluje on akcje, jakie mają się wykonać jako efekt uboczny zmiany jednego z elementów w tabeli zależności.”. W tym wypadku, jeśli zmieni się umieszczony w tabeli zależności stan *newReservationSuccess* na pozytywny zostaną wykonane operacje mające na celu potwierdzenie rezerwacji. Należy nadmienić, że za zmianę tego stanu jest odpowiedzialna funkcja *sendApiRequest*, która dostaje funkcję do zmiany tego stanu jako argument. Operacje jakie się wykonają to:

- przekierowanie użytkownika do podstrony “/services”, na której przedstawione są wszystkie rezerwacje.

- wysłanie e-maila potwierdzającego rezerwację, wraz ze szczegółami zapisanymi w stanie *currentReservation* oraz kilku innych stanach, których opis został pominięty w celu uproszczenia
- ustawienie lokalnego stanu *reservations*, będącego na poziomie głównego komponentu na wszystkie aktualne elementy tablicy z umieszczeniem na końcu nowo dodanej rezerwacji. Należy zauważyć, że jeśli akcja ta została pominięta, wymagane było wysłanie zapytania HTTP do API jeszcze raz, w celu pobrania wszystkich rezerwacji, w tym nowej. Jest to akcja zbyteczna, ponieważ można ją wykonać bez obciążania sieci, poprzez zarządzanie lokalnym stanem. Jest to też powód, dla którego lokalny stan każdej kolekcji jest trzymany w aplikacji.

Komponent *MyServices*

19. Komponent powiązany ze ścieżką względną “/services” ma za zadanie, oprócz pokazania wszystkich rezerwacji i wypożyczeń użytkownika, również pokazać komunikat o poprawnie dodanej rezerwacji. Może to zrobić dzięki, otrzymanemu jako właściwość, stanu *newReservationSuccess*, który został zdefiniowany na poziomie głównego komponentu *App*, a następnie zmieniony przez funkcję *sendApiRequest* w *ReservationConfirmation*.

Powyższy opis, po raz kolejny zwraca uwagę, na ułatwienia jakie zaoferowały hooki, wprowadzone w nowej wersji biblioteki, opublikowanej w 2019 roku.

3.4.5. Podgląd aktualnych, anulowanych i archiwalnych usług

W znajdującym się w menu pod nazwą “Zarządzaj i historia” komponencie *MyServices* zalogowany użytkownik może znaleźć bieżące wypożyczenia i przyszłe rezerwacje, w tym odwołane. Na samym końcu znajdują się archiwalne już usługi. Została przyjęta kolorystyka, w której zielony kolor oznacza wypożyczenie, niebieski przyszła rezerwację, szary - przyszłą, ale odwołaną rezerwację, czarny - archiwalne usługi. Klient ma wgląd to statusu wypożyczeń, w tym naliczonej do obecnego momentu opłaty, która pokazywana jest po prawej stronie. W analogicznym miejscu dla przyszłych rezerwacji pokazywane jest odliczanie do terminu odbioru.

Aktualne rezerwacje i wypożyczenia:

 Rezerwacja została dokonana pomyślnie!
Potwierdzenie możesz znaleźć w swojej skrzynce email.



Atomic redster (rozmiar: 160)

Rozpoczęto: 06.01.2022 (20:33)

140 PLN

NALICZONE DO TEJ PORY / ZA 2 D.



Specialized Mountain Mega 12 (rozmiar: S)

Rozpoczęto: 02.01.2022 (05:31)

300 PLN

NALICZONE DO TEJ PORY / ZA 6 D.



Salomon Ski Runner x8 (rozmiar: 154)

18.01.2022 (15:33) - 21.01.2022 (15:33)

280 zł

confirmed

za 0 mies. 10 dni



Specialized Mountain Mega 12 (rozmiar: S)

30.01.2022 (15:00) - 31.01.2022 (17:00)

100 zł

cancelled

Zrzut ekranu 20. Komponent MyServices. Źródło: opracowanie własne

Kliknięcie w wybraną rezerwację otwiera interfejs komponentu *ReservationFocus* lub *RentalFocus*, w zależności od wybranego typu usługi. W tym miejscu klient może utworzyć wiadomość do pracownika wypożyczalni, której szczegóły implementacyjne zostały opisane w rozdziale 3.4.2.



Salomon Ski Runner x8

Rozmiar: 154

Data odbioru: 18.01.2022 (15:33)

Data zwrotu: 21.01.2022 (15:33)

Cena: 280 zł

Status: confirmed

ANULUJ REZERWACJĘ

 Na tej stronie możesz wysłać zapytanie e-mail dotyczące tej rezerwacji.

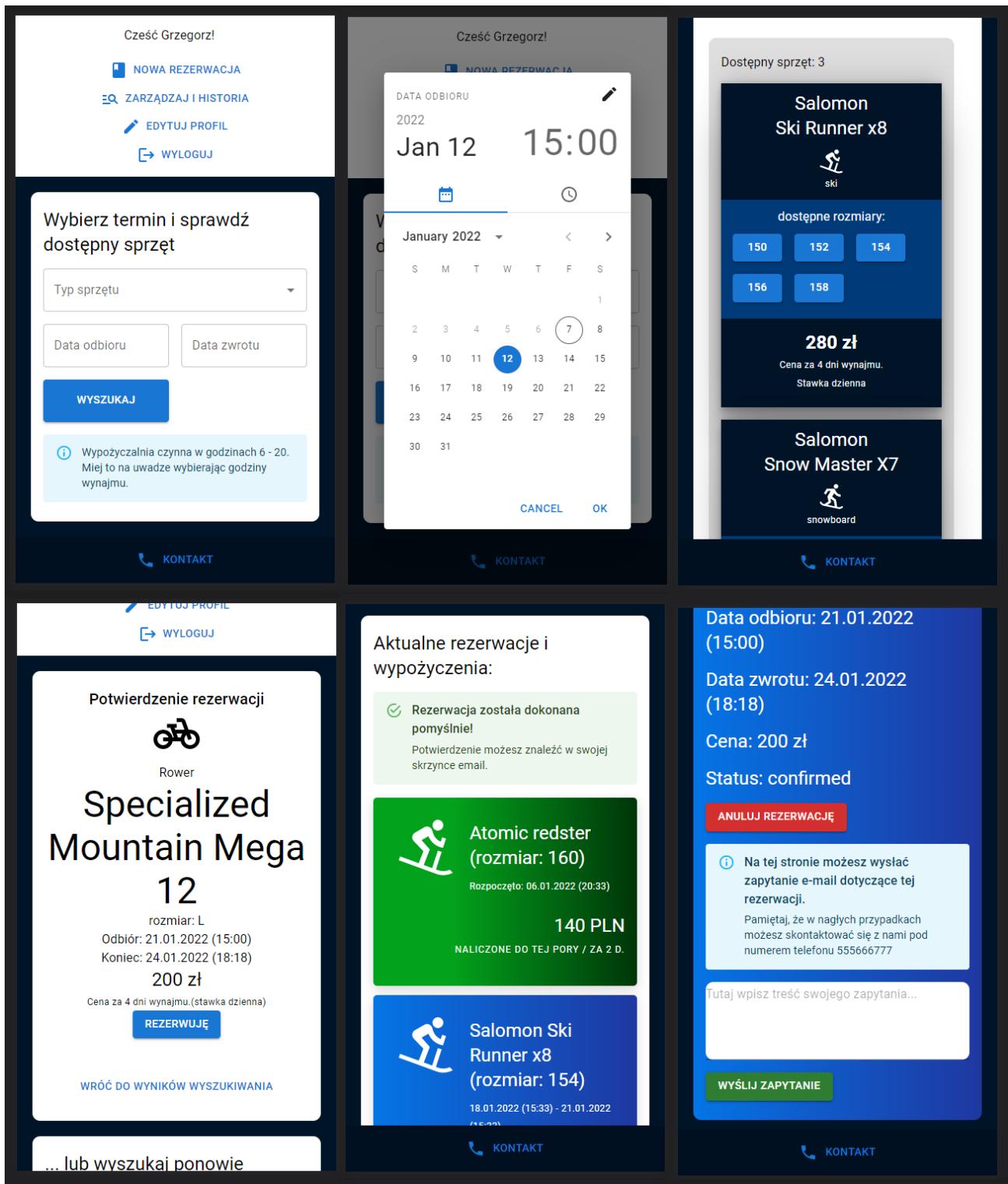
Pamiętaj, że w nagłych przypadkach możesz skontaktować się z nami pod numerem telefonu 555666777

Tuże wpisz treść swojego zapytania...

WYŚLIJ ZAPYTANIE

Zrzut ekranu 21. Komponent ReservationFocus. Źródło: opracowanie własne

3.4.6. Responsywność i dostępność na urządzeniach mobilnych



Zrzut ekranu 22. Pełny proces rezerwacji przeprowadzony na urządzeniu mobilnym. Źródło: opracowanie własne

Podczas tworzenia komponentów, ważnym punktem było zapewnienie wszystkich funkcjonalności na każdym powszechnie używanym urządzeniu. Ciężko bowiem wyobrazić sobie

nowoczesną stronę internetową, która nie zapewnia wsparcia dla użytkowników mobilnych. Na zrzucie ekranu 22 przedstawiono pełny proces rezerwacji sprzętu przez uprzednio zalogowanego użytkownika. Wymiary ekranu zostały ustawione na identyczne jak w smartfonie iPhone 6/7/8 Plus.

3.4.7. Sesja użytkownika

Aby wytlumaczyć zaimplementowane rozwiązanie, należy wpierw przybliżyć obiekty dostarczane przez Web Storage API. Pierwszy z nich to *localStorage*, który pozwala przechowywać obiekty w pamięci przeglądarki (po stronie klienta) bez daty wygaśnięcia. Drugi, nazwany *sessionStorage*, działa w podobny sposób, lecz obiekty są usuwane wraz z zamknięciem karty przeglądarki. W obu przypadkach dane są przechowywane w postaci par, składających się z klucza oraz wartości. Oba mechanizmy zostały wprowadzone wraz z publikacją HTML5, czyli nowej wersji tego języka. Autor zdecydował się na użycie pierwszego rozwiązania, aby użytkownik był uwierzytelniony, również po zamknięciu karty przeglądarki, co umożliwia mu szybki podgląd dokonanych rezerwacji. Sposób ten jest stosowany obecnie przez wiele stron internetowych. Zaleca się jednak aby użytkownik, zawsze wylogowywał się ze swojego konta, jeśli współdzieli komputer z innymi osobami.

Po poprawnym zalogowaniu lub rejestracji zaszyfrowany i przekonwertowany do ciągu znaków obiekt JSON z danymi użytkownika jest zapisywany w lokalnym stanie głównego komponentu *loggedUser*. Jeden z użytych w tym komponencie hooków *useEffect*, posiada ten stan w tablicy zależności. Po zmianie wartości stanu *loggedUser* wykonywana jest więc akcja. Wpierw sprawdzana jest jego zawartość. Jeśli obiekt nie jest pusty, wywoływana jest metoda *setItem* pochodząca z Web Storage API. Ustawia ona element pamięci lokalnej *user*, na wartość stanu *loggedUser*. Dzięki temu, po ponownym otwarciu strony, po odczytaniu tego elementu pamięci, wiadomo, który użytkownik jest zalogowany i można pominąć etap uwierzytelniania. Jeśli klient zdecyduje się wylogować, *loggedUser* jest ustawiany na pusty ciąg znaków, co skutkuje całkowitym usunięciem lokalnego użytkownika z pamięci.

3.4.8. Obsługa routingu

Każda zakładka z menu posiada osobną trasę (ang. *route*). Oznacza to, że otwarcie jej nie wymaga interakcji użytkownika z paskiem nawigacyjnym, a można ją otworzyć wpisując odpowiedni adres w przeglądarce. Utworzono zostało 7 osobnych tras, z czego dwie stanowią adresy dynamicznie tworzone, a jeden to trasa ogólna. Implementacja została wykonana, z użyciem

biblioteki React Router, która dostarcza, między innymi komponenty *Routes* oraz *Route*. W miejscu, w którym ma zostać wyświetlony komponent, do którego prowadzi trasa, należy użyć obu dostarczanych komponentów i wyszczególnić ścieżkę (ang. *path*) oraz element który ma się wyświetlić. Na diagramie 3 pokazano uproszczony diagram komponentów w odniesieniu do routingu. Można na nim zaobserwować, że pomimo zmiany trasy, komponent Navbar oraz Footer będzie wciąż wyświetlany

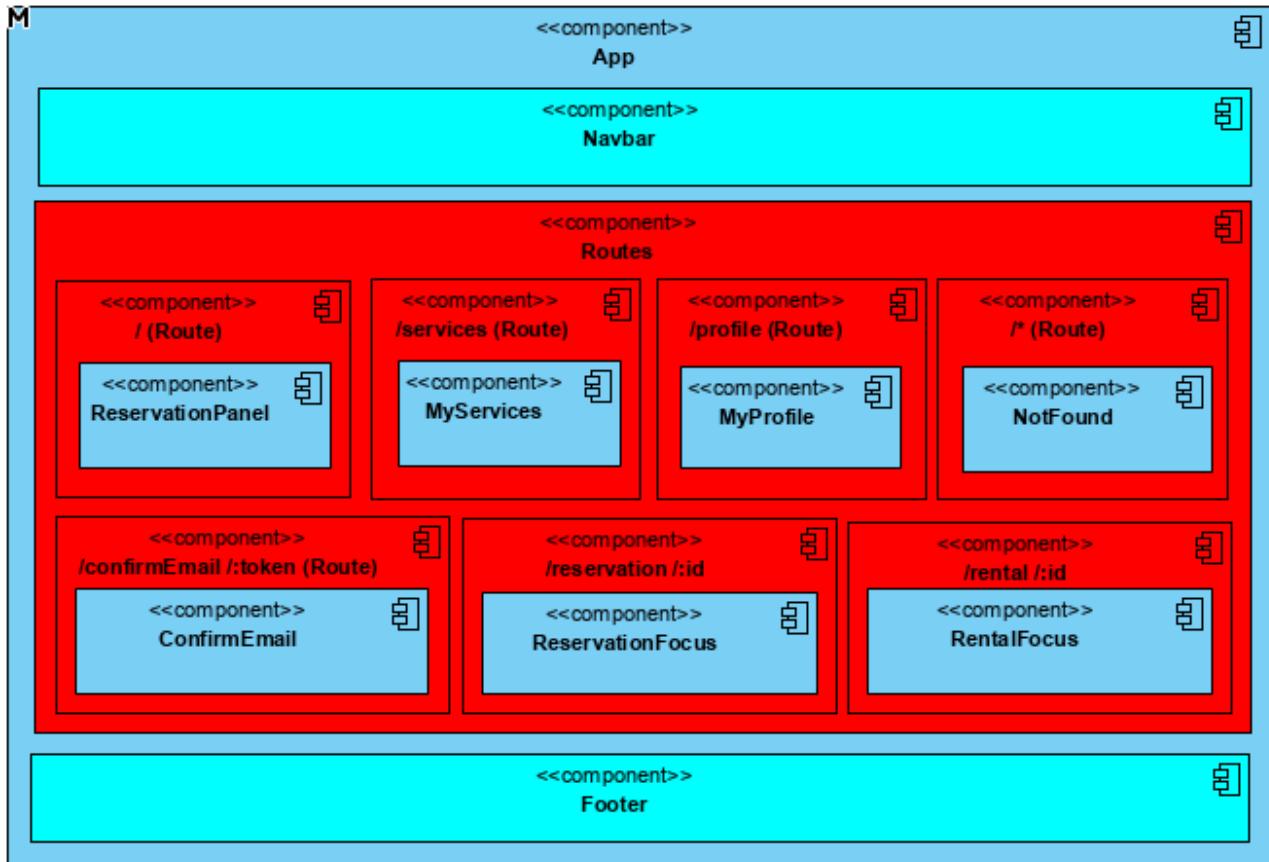
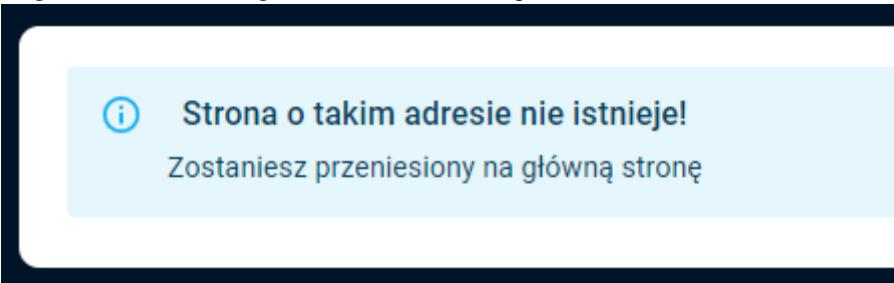


Diagram 3.Uproszczony układ strony w oparciu o routing. Źródło: opracowanie własne

Na fragmencie kodu 15 można również zauważyc, że ścieżka jest wybierana z interfejsu numerycznego Path, co oznacza, że jeśli deweloper zdecyduje się zmienić nazwę trasy, wszelkie odwołania do niej zostaną zmienione automatycznie. Jeśli użyty by został zwykły ciąg znaków, należałoby poprawić ścieżkę w każdym miejscu. Wspomniana wcześniej trasa ogólna jest oznaczona jako * (słownie: znak gwiazdy). Interpretuje ona każdą trasę, jaka nie została zdefiniowana i uruchamia komponent *NotFound*, wyświetlający informację o błędny adresie. (przedstawiony na zrzucie ekranu 23).

```
<Route
  path={`${Path.confirmEmail}/:token`}
  element={
    <ConfirmEmail
      setLoggedUser={setLoggedUser}
      users={users}
      apiDataInitialized={apiDataInitialized}
    />
  }
/>
```

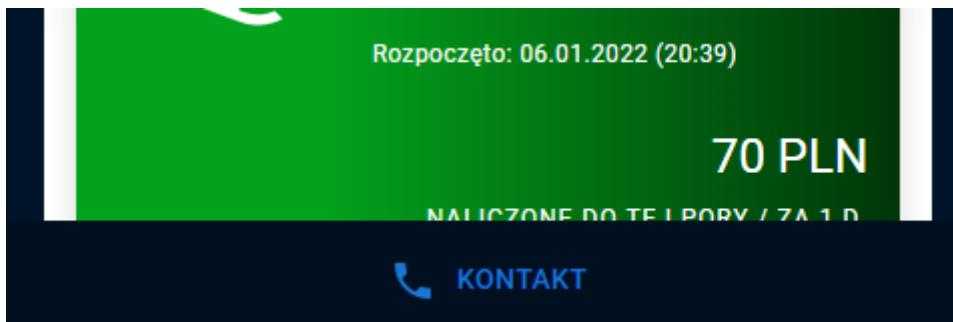
Fragment kodu 15. Komponent Route. Źródło: opracowanie własne



Zrzut ekranu 23. Komunikat o błędnym adresie strony. Źródło: opracowanie własne

3.4.9. Elementy sterowane z aplikacji desktopowej

Należy mieć na uwadze, że aplikacja bazuje głównie na danych, które są modyfikowane w aplikacji desktopowej. Aplikacja webowa nie jest w stanie funkcjonować, jeśli dane te nie zostaną dostarczone. Oznacza to, że podczas wdrożenia systemu do firmy, należy najpierw w pełni skonfigurować wszelkie informacje, według wiedzy pracownika wypożyczalni. W skład tego wchodzi dodanie wszystkich produktów, ich cen oraz wszystkich danych firmy takie jak adres e-mail, telefon kontaktowy, nazwa strony, godziny otwarcia czy procent sprzętu przeznaczonego do rezerwacji on-line. Dane te są szeroko wykorzystywane w cyklu życia aplikacji. Jak zostało wspomniane wcześniej, godziny otwarcia są wykorzystywane do ograniczenia wyboru godzin rezerwacji sprzętu. Inny przykład stanowi numer telefonu, który jest pokazywany wraz z informacją o błędzie, wynikającym z braku sieci. Pozwala na wyświetlenie numeru telefonu wraz z informacją o błędzie. Jest to możliwe ponieważ dane te są pobierane przy pierwszym uruchomieniu aplikacji, a następnie zapisywane w lokalnym stanie komponentu *App*, dokładniej *companyInfo*. Nie jest wymagane więc aktywne połączenie z siecią, aby uzyskać do nich dostęp. Podobnie jest z zakładką kontakt (przedstawiona na zrzucie ekranu 24 i 25), która zawiera wszelkie dane o wypożyczalni i pozwala na wgląd nawet bez połączenia z siecią. Zastosowane rozwiązanie zapewnia, więc niezawodność na wysokim poziomie.



Zrzut ekranu 24. Przycisk do otwarcia zakładki “Kontakt”. Źródło: opracowanie własne



Zrzut ekranu 25. Zakładka “Kontakt”. Źródło: opracowanie własne

3.5. Testy

Testowanie to ważny aspekt w zapewnieniu niezawodności komercyjnej aplikacji ponieważ każdy błąd lub niedostępność funkcjonalności może narazić wypożyczalnię na straty. Testowanie należy podzielić na kilka podgrup. Testy jednostkowe, czyli testowanie poprawności wykonywanych obliczeń, polegające na testowaniu wywołań funkcji z konkretnymi danymi i sprawdzaniu czy rezultat jest prawidłowy. Testowanie e2e polega na wcieleniu się w rolę końcowego użytkownika i wykonanie kompletu operacji, które użytkownik może wykonywać celowo lub niecelowo oraz kontrolowanie czy aplikacja zachowuje się w oczekiwany sposób. Ostatnią metodyką testowania jest testowanie manualne. Jest ono podobne do testów e2e, lecz nie jest zautomatyzowane, więc wymaga poświęcenia czasu przez programistę. Pisanie testów e2e jest więc szczególnie przydatne, gdy aplikacja jest rozwijana przez dłuższy czas i jest narażona na sytuacje, gdy jedna, nawet najmniejsza zmiana, może zmienić zachowanie kodu w innym miejscu, czego programista nie jest świadomym.

Autor zdecydował się napisać testy jednostkowe oraz przetestować aplikację manualnie. Pisanie testów e2e na tym poziomie zaawansowania aplikacji, nie jest wymagane. Doświadczenie

autora wskazuje, że testy e2e są pisane często dopiero po zaimplementowaniu danego etapu aplikacji, nie są pisanie na bieżąco. Zasadne jest więc napisanie tych testów, jeśli autor zdecyduje się rozwijać dalej aplikacje. Należy też zwrócić uwagę, że testy e2e są często pisane przez specjalistów na stanowiskach testerów, to znaczy, że bardzo często sam deweloper nie jest zaangażowany w ten proces. Doświadczenie pokazuje również, że testy e2e pisane przez inną osobę, niż deweloper odpowiedzialny za budowę kodu, mogą mieć lepszy efekt, ponieważ, osoba która jest autorem danych funkcjonalności, może nie przewidzieć wszystkich scenariuszy tak dobrze jak osoba, która nie była zaangażowana w proces tworzenia kodu i patrzy na funkcjonalności z perspektywy docelowego użytkownika aplikacji.

Testy jednostkowe napisane przy pomocy biblioteki Jest, mogą być uruchamiane lokalnie w środowisku deweloperskim programisty. Środowisko używane przez autora pozwala na ciągłe uruchamianie testów po każdej zmianie w kodzie, więc gwarantuje zauważalność błędów podczas pisania programu. Struktura aplikacji jest spójna i wszelkie funkcje pomocnicze każdego komponentu znajdują się w pliku *utils.ts* w katalogu danego komponentu. Testy integracyjne testują te funkcje i są zawarte w plikach *utils.spec.ts*. Jest to popularne nazewnictwo w projektach z wykorzystaniem biblioteki React, ponieważ termin specyfikacja (ang. *spec*) odnosi się do tego co dana funkcja powinna zwracać. Jeśli wynik funkcji jest zgodny, oznacza to, że funkcja spełnia swoją specyfikację. W pewnych sytuacjach, aby przeprowadzić testy potrzebne są atrapy obiektów (ang. *mock objects*). Służą one do wykonania testów na większych obiektach, takich jak wieloelementowe tablice, których użycie bezpośrednio w ciele funkcji testowej wiążałoby się z utratą jej czytelności. Obiekty te z reguły definiuje się na początku pliku z testami, zaraz pod importowanymi bibliotekami, lub w osobnych plikach, na tym samym poziomie co *utils.ts* oraz *utils.spec.ts*. Zwyczajowo nadaje się im nazwę równą testowanemu plikowi, z dopiskiem do rozszerzenia wyrazu *mock*. Autor zdecydował się na drugi sposób, a więc atrapy obiektów znajdują się w plikach *utils.mock.ts*.

Biblioteka Jest pozwala prowadzić różne strategie testowania, jednak najbardziej popularne jest otaczać testy funkcji danego komponentu w funkcję *describe*. Same testy przeprowadza się z użyciem funkcji *it*, gdzie należy słownie opisać oczekiwany rezultat testu oraz wykonać obliczenia. Wynik testu zostanie uznany za poprawny, jeśli, żadna z użytych funkcji *expect*, nie zwróci błędu. Funkcja ta pozwala używać asercji takich jak na przykład *toStrictEqual*, aby porównywać wartość oczekiwana i otrzymaną. Przykład testu z zastosowaniem omawianych funkcji został pokazany we fragmencie kodu 1. Testuje on poprawność działania funkcji *filterOutReservedItems*, filtrującej

tablicę produktów, by finalnie zwrócić tablicę produktów dostępnych w wybranym terminie. Argumenty jakie przyjmuje to data odbioru oraz data zwrotu typu *Date*, tablica obiektów typu *Reservation* oraz tablica obiektów typu *Item*. Po wykonaniu obliczeń, zwracana jest zaktualizowana tablica obiektów typu *Item*. Omówione zachowanie zostało opisane w opisie implementacji w podrozdziale 3.4.4, a dokładniej w punkcie 10, w zadaniach jakie wykonuje komponent *ReservationPanel*. Jest to jedno z zachowań, jakie należy wykonać, aby wyświetlić produkty możliwe do rezerwacji przez klienta. Należy zaznaczyć, że wynik testu integracyjnego może przyjmować tylko dwie formy, to jest, pozytywny lub negatywny. Aby uznać etap testowania za zrealizowany, wszystkie napisane testy muszą zakończyć się pozytywnie.

```
⑥ 11  ✓ describe("ReservationPanel utils.ts", () => {
  12   ✓ it("should filter out items that are reserved in provided dates", () => {
  13     ✓ expect(
  14       |   filterOutReservedItems(START_DATE, FINISH_DATE, RESERVATIONS, ITEMS)
  15     ).toStrictEqual(EXPECTED_FILTERED_ITEMS);
  16   });
})
```

Fragment kodu 16. Przykład testu z zastosowaniem atrap obiektów. Źródło: opracowanie własne

Testy manualne były wykonywane podczas tworzenia samej aplikacji, ponieważ wdrażając nowe funkcjonalności, jedynym sposobem na sprawdzenie ich poprawności, jest właśnie wykonanie szeregu operacji z ich pomocą. Na samym końcu, aplikacja została sprawdzona od początku do końca, aby upewnić się, że sposób jej działania jest zgodny z postawionymi wymaganiami.

4. Aplikacja desktopowa

Aplikacja komputerowa będzie obsługiwana głównie przez pracowników wypożyczalni. Podczas tworzenia bardzo ważnym aspektem będzie jego prostota, oraz intuicyjność. Nie można przy tym jednak zapomnieć o pełnej funkcjonalności systemu, tak aby sprostać wszelkim oczekiwaniom rynkowym. Dobór odpowiednich technologii, plan, oraz schemat działania również jest bardzo ważny, jak nie najważniejszy. To właśnie od tego zależy działanie całego systemu, dlatego ten aspekt zostanie rozwinięty jako pierwszy.

4.1 Wykorzystane technologie

Podczas rozpoczęcia pracy nad aplikacją desktopową, należało przeprowadzić przegląd dostępnych technologii i wybrać spośród nich odpowiednie. Głównym kryterium był wybór obiektowego języka programowania, ze względu na złożoność systemu. Należało również wybrać odpowiednią bazę danych tak, aby mogły z niej korzystać obydwa moduły (aplikacja desktopowa oraz webowa) tytułowej pracy.

Jako język programowania została wybrana Java 8 LTS. Został on wybrany głównie ze względu na obiektowość i przenośność aplikacji na system Windows oraz Linux. Wybór JDK w wersji 8 został uwarunkowany nabytym doświadczeniem w toku studiów podczas pracy nad różnymi projektami. Wersja LTS [19] (Long Term Support), jest skoncentrowana głównie na stabilności, długości wspierania oraz dostępnych kompatybilnych technologii.

JavaFX to kolejna użyta technologia. Framework używany do tworzenia aplikacji okienkowych, w skład którego wchodzi zestaw funkcjonalności, bibliotek służących do tworzenia cross platformowych GUI (ang. graphical user interface). JavaFX stanowi integralną część Java JDK 8, wchodząc w zestaw bibliotek podstawowych.

SceneBuilder - jedno z podstawowych narzędzi używanych do tworzenia graficznych interfejsów użytkownika przy użyciu framework JavaFX. Działa na zasadzie “drag and drop” (przeciągnij i upuść). Technologia ta umożliwia szybsze i łatwiejsze tworzenie szablonu aplikacji okienkowej, poprzez generowanie pliku z kodem XML, gdzie zawarty zostanie wygląd danej sceny, którą można dowolnie edytować. SceneBuilder wchodzi w skład Java JDK 8.

Apache Maven - to kolejna użytka technologia. Dzięki wbudowanym wtyczkom, podczas pierwszego użycia bibliotek lub frameworków zostają pobrane zależności (dependencies), wykorzystywane w programie. Technologia zapisuje informacje jakie biblioteki, rozszerzenia są aktualnie używane, aby móc pobrać je w razie potrzeby, podczas uruchamiania programu na innym środowisku.

Jedną z ważniejszych technologii użytych do stworzenia systemu jest odpowiednia baza danych. System składa się z dwóch modułów - aplikacji desktopowej oraz webowej, dlatego szukano technologii umożliwiającej wspólną pracę nad danymi. Wybór padł na bazę danych MongoDB. Działająca w chmurze, w oparciu o system NoSQL, nierelacyjna baza danych umożliwiła integrację dwóch systemów, a dzięki temu poprawne działanie aplikacji desktopowej.

MongoDB Java driver sync - jest to biblioteka, dzięki której odbywa się komunikacja z systemem bazodanowym w chmurze. Stałe, proste połączenie cechuje szybkość wykonywania operacji skracając do minimum czas oczekiwania na dostęp do danych, przez co sama aplikacja może działać szybciej.

JUnit - technologia wykorzystywana do tworzenia testów jednostkowych w środowisku Java. Dzięki nim programista może śledzić poprawność napisanej funkcjonalności, minimalizując przy tym ryzyko powstawania błędów w logice aplikacji. Wczesne wykrycie błędów przyspiesza pracę oraz poprawia jakość tworzonego oprogramowania.

IntelliJ - środowisko programistyczne, wykorzystane do stworzenia w całości części desktopowej tytułowej pracy. Dzięki szerokiemu wsparciu nowoczesnych technologii takich jak: Maven, GIT i wielu innych, bardzo często wybierane przez programistów. Projekt został stworzony w oparciu o studencką licencję.

GitHub - to kolejna technologia użytka do tworzenia aplikacji. Dzięki możliwości dodawania coraz to bardziej rozbudowanego kodu źródłowego do prywatnego repozytorium Git, programista zabezpiecza się na wypadek popełnienia poważnych błędów, które uniemożliwiłyby dalszą pracę dzięki możliwości powrotu do starszej wersji kodu. Technologia ta chroni również przed utratą danych umożliwiając pobranie kodu źródłowego z chmury na wybrany przez programistę język. Technologia przydaje się również podczas pracy wspólnej, dzięki niej każda z osób pracujących nad kodem może pobrać ostatnią dodaną przez współpracowników wersję, porównać ją oraz przesyłać wykonaną część kodu.

4.2 Wymagania funkcjonalne

W tym podrozdziale zostały wypunktowane wymagania funkcjonalne jakie powinien spełniać system obsługi wypożyczalni. Każdy z wymienionych podpunktów został zaimplementowany w aplikacji oraz opisany w dalszej części pracy.

1. Logowanie do panelu, weryfikacja danych użytkownika.

2. Obsługa sprzętu

- a. Przegląd
- b. Dodawanie
- c. Edycja
- d. Dodawanie podobnego
- e. Usuwanie

3. Ustawienia

- a. Dodawanie typu (kategorii) i cennika sprzętu
 - i. Przegląd istniejących typów i ich ceny
 - ii. Dodawanie typu
 - iii. Dodawanie cennika do danego typu
 - iv. Edycja istniejącego cennika i typu
 - v. Usuwanie istniejącego cennika i typu

b. Edycja pracowników

- i. Przegląd
 - ii. Dodawanie nowego pracownika
 - iii. Edycja istniejącego
 - iv. Usuwanie istniejącego pracownika
- c. Usuwanie wszystkich użytkowników
 - d. Usuwanie wszystkich dodanych sprzętów
 - e. Edycja podstawowych informacji o firmie

4. Klienci

- a. Przegląd zarejestrowanych
- b. Dodawanie nowych
- c. Edycja istniejących
- d. Usuwanie istniejących

5. Rezerwacje

- a. Przegląd rezerwacji potwierdzonych
- b. Możliwość wynajęcia danego zarezerwowanego sprzętu
- c. Edycja klienta o szczegółowe dane

6. Wypożyczenie

- a. Przegląd istniejących klientów
- b. Edycja istniejących klientów o szczegółowe dane
- c. Dodawanie nowego klienta
- d. Przegląd sprzętu wypożyczonego przez danego klienta
- e. Wypożyczenia sprzętu o danym ID
- f. Możliwość usunięcia wypożyczenia

7. Zwrot

- a. Zwrot sprzętu po ID
- b. Obliczenie sumy do zapłaty

4.3 Struktura aplikacji

4.3.1 Wzorzec projektowy MVC

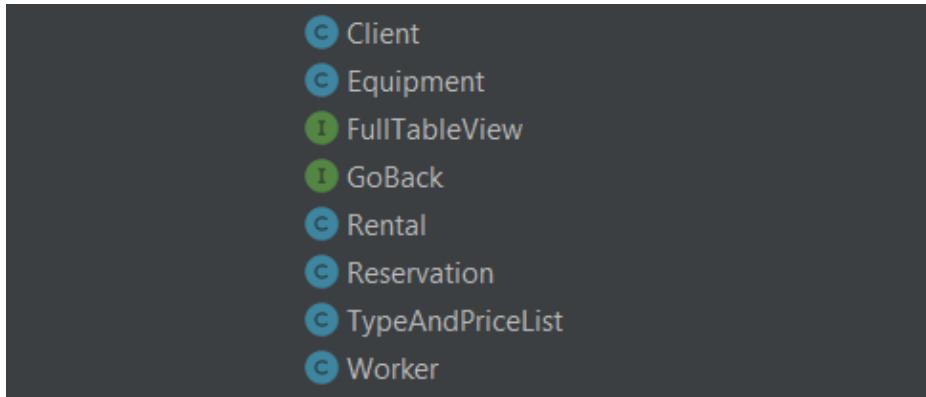
Moduł desktopowy tytułowego systemu został stworzony w oparciu o wzorzec architektonicznego MVC - Model View Controller. Charakteryzujący się jasnym podziałem struktury aplikacji na:

- model
- kontroler
- widok

Model

Po stronie modelu zostały utworzone pliki klas oraz interfejsów, wykorzystywane przez kontrolery. Model przechowuje dane, które następnie mogą zostać zmodyfikowane przez kontroler oraz wyświetcone przez widok. Stworzone klasy służą do generowania obiektów, które następnie znajdą zastosowanie w poprawnej obsłudze aplikacji oraz komunikowania się z bazą danych. Dodatkowo zostały zaimplementowane dwa interfejsy, w których znajdują się puste metody, które należy nadpisać podczas implementacji w klasach kontrolera. Służą one do obsługi takiej

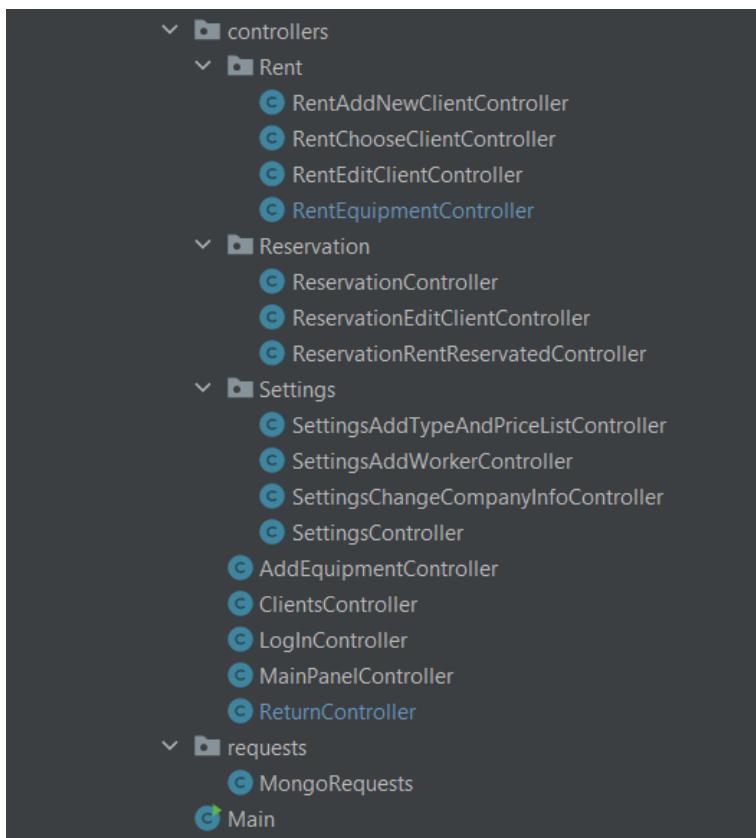
funkcjonalności jak: wracanie do sceny wcześniejszej oraz uzupełniania używanych tabel. Interfejsy te pomagają w czytelności kodu pracy nad aplikacją więcej niż jednej osobie.



Zrzut ekranu 26 - podział struktury aplikacji - MVC - model. Źródło: opracowanie własne

Kontroler

Kolejnym podziałem aplikacji jest strona kontrolera. Znajdujące się tutaj klasy odpowiedzialne są za modyfikację modelu oraz jego komunikację z widokiem. W klasach kontrolera zawarte zostały pełne funkcjonalności oraz logika aplikacji. Podczas występowania większej ilości scen obsługujących daną funkcjonalność, zostały utworzone pakiety, a w nich umieszczone klasy odpowiadające za daną scenę. Posłużyły one poprawie czytelności, zachowania przejrzystości struktury oraz ułatwieniu pracy i przeglądu.



Zrzut ekranu 27 - struktura aplikacji - MVC - kontroler. Źródło: opracowanie własne.

Dodatkową uwagę należy poświęcić klasie “MongoRequest”, w pakiecie “requests” (fragment kodu 17), w której zawarta jest cała logika oraz dane funkcjonalności odpowiadające za komunikację aplikacji okienkowej z systemem bazodanowym. Stworzone metody, wykorzystywane są przez większość klas kontrolera do komunikacji z bazą danych.

```
protected static void deleteObject(String collectionName, String _id) {...}

protected static boolean checkObjectFileterExists(String collectionName, String fieldname, String filter) {...}

protected static boolean checkObjectDoubleFilterExists(String collectionName, String fieldname, String filter, String secondFieldname) {...}

protected static boolean checkObjectFilter(String collectionName, String fieldname, String filter, String oldFilter) {...}

protected static ArrayList<Object> getCollection(String collectionName) {...}

protected static ArrayList<Object> getCollectionFilter(String collectionName, String fieldname, String filter) {...}

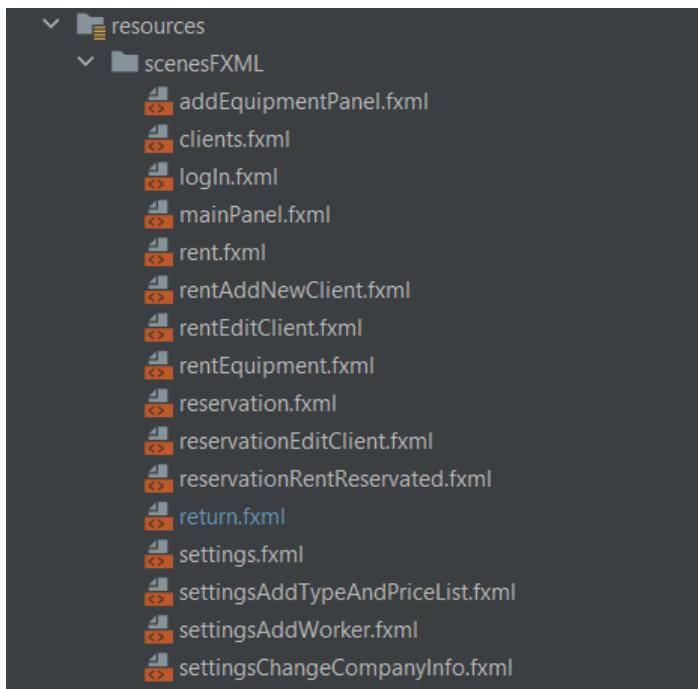
protected static Document getObjectFilter(String collectionName, String fieldName, String filter) {...}

protected static Document getObjectDoubleFilter(String collectionName, String fieldName, String filter, String secondFieldName, String secondFilter) {...}
```

Fragment kodu 17. Metody klasy “MongoRequest”, w pakiecie “requests”. Źródło: opracowanie własne.

Widok

Kolejnym i ostatnim podziałem struktury aplikacji ze względu na wzorzec architektoniczny MVC jest widok. Obsługuje on część odpowiedzialną za graficzny interfejs użytkownika. Zawarte w pakiecie “scenesFXML” pliki, przechowują rozmieszczenie komponentów biblioteki JavaFX w danej scenie. W części widoku umiejscowione zostały tylko odnośniki do danej funkcjonalności, zawartej w części kontrolera, oraz wyświetlane poszczególne elementy obiektów klas modelu.



Zrzut ekranu 28 - struktura aplikacji - MVC - widok. Źródło: opracowanie własne.

4.3.2 Testy

Podczas pisania obszernych aplikacji należy wykonać testy. Zapobiegają one powstawaniu ewentualnych błędów, przez fakt ciągłego nadzorowania poprawności kodu źródłowego, który może zmieniać się z czasem podczas edycji lub dodawania nowego kontentu. Autor aplikacji wykonał testy jednostkowe, sprawdzające poprawność danych funkcjonalności przy użyciu biblioteki zewnętrznej JUnit.

Aby aplikacja mogła przejść przez proces testowania, wymagane jest spełnienie wszystkich stworzonych testów. Podczas gdy przynajmniej jeden z nich nie zostanie wykonany pomyślnie, należy sprawdzić daną funkcję, tak aby znaleźć błąd. Dzięki temu programista w łatwy sposób może wyszukać błąd w kodzie, przeglądając tylko daną część, a nie przeglądając całości pracy, która przy obszerniejszych systemach pochłonęłaby bardzo dużo czasu.

Testy komunikacji z bazą danych

Podczas pracy z bazą danych umieszczoną w chmurze, bardzo ważnym aspektem jest sprawdzenie poprawności pobieranych oraz wysyłanych danych, tak aby nie dodawać pustych, błędnych oraz niepoprawnych treści. Stworzone testy miały na celu sprawdzenie operacji logicznych, wykluczających dane funkcjonalności. Dodatkowo sprawdzały, czy podane dane nie powodują powstawania wyjątków oraz czy są takowe wyłapywane.

Test Results		6 s 850 ms
✓	MongoRequestsTest	6 s 850 ms
✓	getEmployeeReturnFalse()	6 s 317 ms
✓	addEquipmentReturnFalse()	129 ms
✓	getEmployeeReturnTrue()	133 ms
✓	getClicentFalse()	132 ms
✓	deleteEmployeeCatchException()	139 ms

Zrzut ekranu 29 - pomyślne wykonanie testów łączenia z bazą danych. Źródło: opracowanie własne.

Testy szyfrowania i deszyfrowania danych

Szyfrowanie oraz deszyfrowanie jest bardzo ważną funkcją w programach pracujących na danych osobowych. Odpowiednie zabezpieczenie uniemożliwi osobom trzecim dostęp oraz ewentualny wyciek przechowywanych w bazie danych informacji o klientach wypożyczalni. Aby system mógł odpowiednio pracować, należy sprawdzić poprawność szyfrowanych i deszyfrowanych danych, tak aby informacje nie zostały zniekształcone co uniemożliwiłoby

odpowiednią pracę systemu. Dodatkowo źle zaszyfrowane dane zakłóciłyby działanie modułu webowego, gdyż wyświetlane informacje byłyby dla zarejestrowanych klientów niezrozumiałe.

✓	✓	Test Results	940 ms
✓	✓	CryptTest	940 ms
✓		checkDecrypted()	98 ms
✓		checkEncrypted()	842 ms

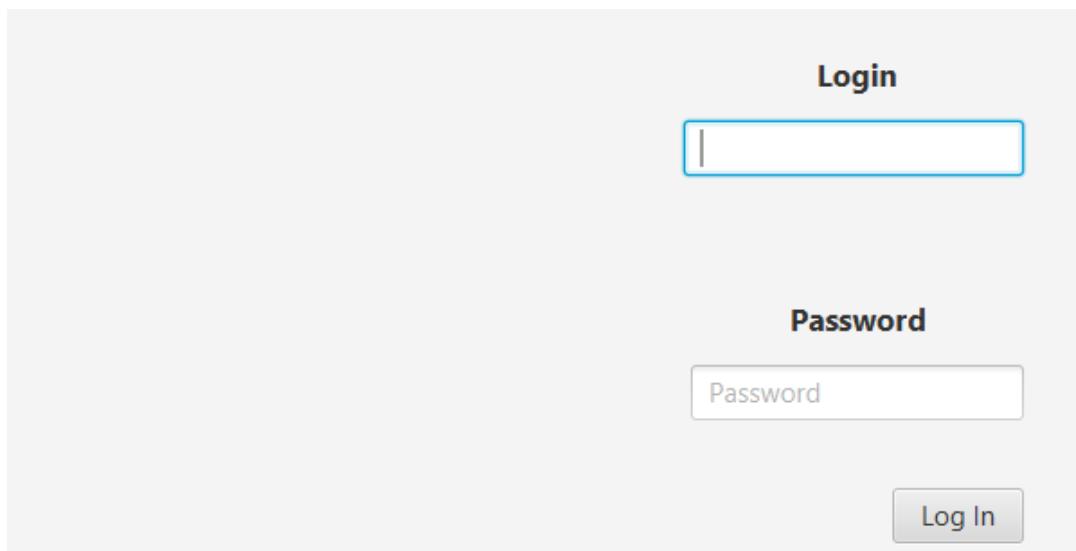
Zrzut ekranu 30 - testostawnie szyfrowania i deszyfrowania. Źródło: opracowanie własne.

4.4 Funkcjonalności

W tym podrozdziale zostały przedstawione stworzone, zaimplementowane funkcjonalności modułu desktopowego tytułowej pracy, wraz z zrzutami ekranu graficznego interfejsu użytkownika, tak aby w pełni wyjaśnić zasadę działania danej funkcjonalności.

4.4.1 Logowanie

Aby móc korzystać z tytułowej aplikacji należy wpierw przejść przez proces logowania. Domyślnym użytkownikiem jest “admin”. Po wpisaniu danych do logowania (login oraz hasło) oraz ich późniejszej weryfikacjach z bazą danych, następuje zalogowanie do głównego panelu, gdzie zachodzi obsługa i wybór przez użytkownika funkcjonalności. Dane przechowywane w bazie danych są w pełni zaszyfrowane, tak aby zachować maksimum bezpieczeństwa. Deszyfrowanie odbywa się podczas weryfikacji poprawności wpisanych danych.



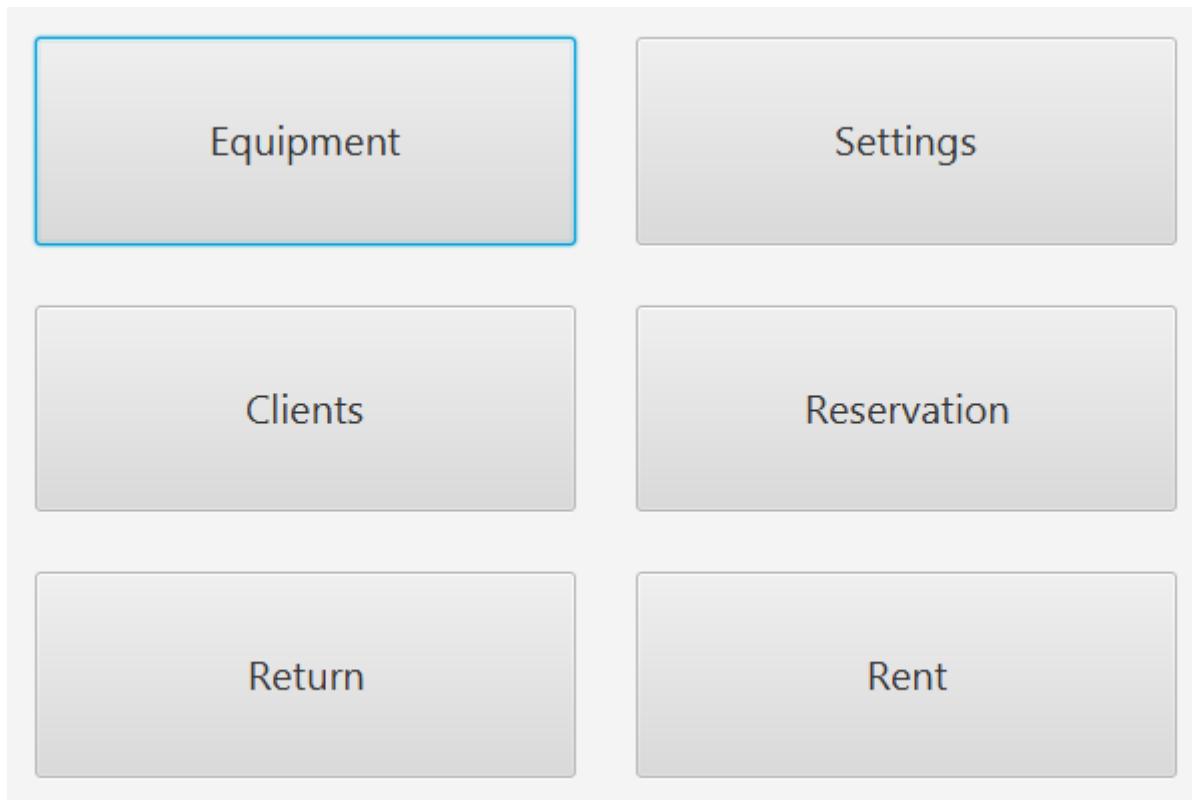
Zrzut ekranu 31 - Okno logowania do systemu obsługi wypożyczalni. Źródło: opracowanie własne.

4.4.2 Główny panel

Po przejściu procesu logowania, użytkownikowi aplikacji wyświetli się główny panel służący do obsługi wypożyczalni. Można tam znaleźć takie funkcjonalności jak:

- Equipment (Sprzęt)
- Settings (Ustawienia)
- Clients (Klienci)
- Reservation (Rezerwacja)
- Return (Zwróć)
- Rent (Wypożycz)

Każda z powyżej wymienionych jest konieczna do prawidłowego działania wypożyczalni. Pomimo faktu użycia nierelacyjnej bazy danych jaką jest MongoDB, praktycznie każda z funkcjonalności potrzebuje siebie nawzajem do pełni sukcesu.



Zrzut ekranu 32 - główny panel aplikacji desktopowej do obsługi wypożyczalni. Źródło: opracowanie własne.

4.4.3 Equipment (Sprzęt)

Po naciśnięciu pierwszego panelu “Equipment”, użytkownik zostaje przeniesiony do sceny, w której można zarządzać dostępnym sprzętem (rzut ekranu 33). W tabeli zostaje wyświetlony dostępny sprzęt. Użytkownik może sortować produkty po typie, producencie, modelu klikając w górną część tabeli, gdzie znajdują się odpowiednio nazwy: “Type”, “Producer”, “Model”.

Type	Producer	Model	Size	Product ID
ski	Super skis	Ski Runn...	152	2
bike	Super Bike	Mountain...	S	696962332...
bike	Super Bike	Mountain...	L	123544334...
car	Car manuf...	City 3	2.0	1235465
kayak	Sea Man	Far away	2 person	1231221
kayak	Sea Man	Far away	2 person	767676

[Back](#) [Delete](#) [Update](#) [Add Si...](#) [Add new](#)

Zrzut ekranu 33 - scena zarządzania sprzętem. Źródło: opracowanie własne.

Jedną z podstawowych funkcji jest dodanie nowego sprzętu, odbywa się to poprzez wybranie po prawej stronie z rozwijanej listy typu produktu. Jest ona pobierana prosto z ustawień, gdzie można dodać nowy typ (kategorię) wraz z pełnym jego cennikiem. Poniżej listy należy wpisać ręcznie producenta, model, rozmiar oraz podać kod identyfikujący dany produkt (Product ID). Następnie wybrać opcję “Add new”, z ang. dodaj nowy.

Występuje również opcja dodawania podobnego, istniejącego już produktu, lecz z innym kodem identyfikującym, wybierając produkt spośród tabeli oraz klikając “Add Similar”, z ang. dodaj podobny.

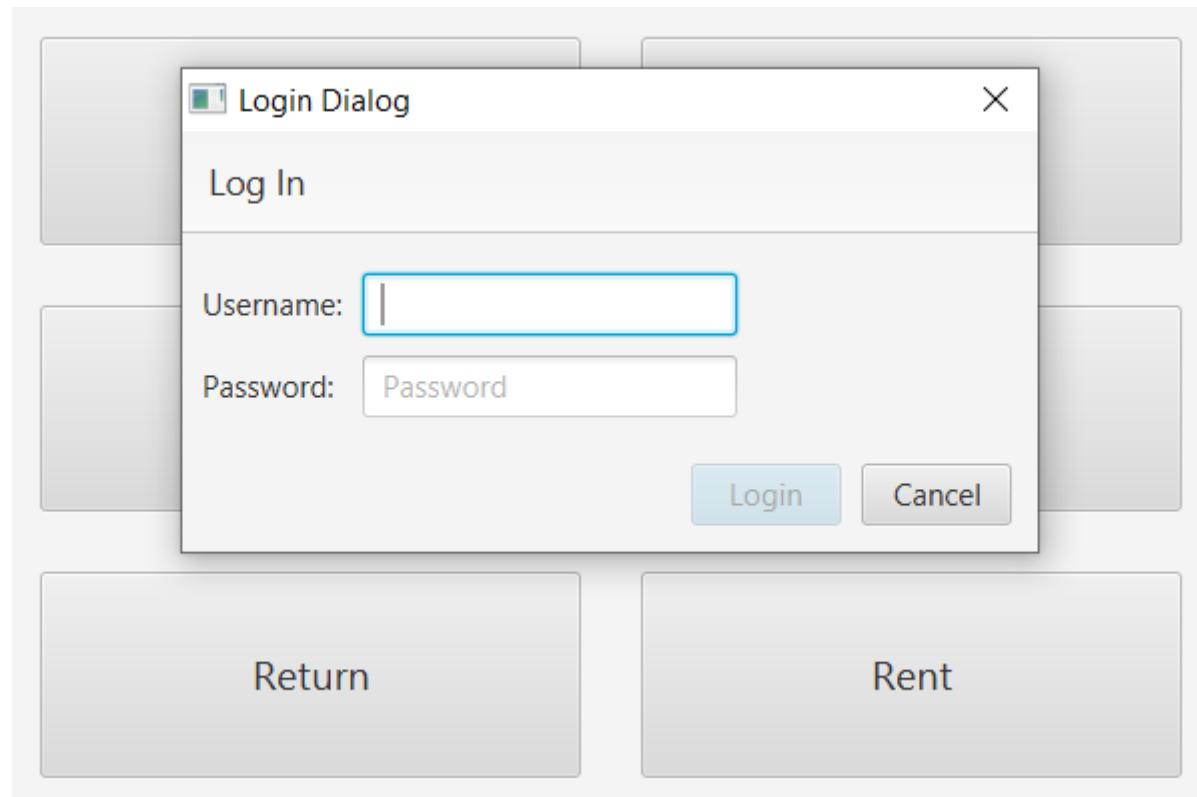
Po podwójnym kliknięciu na tabeli w interesujący nas obiekt, odsłania się opcja aktualizacji oraz usuwania obiektu (Update, Delete). Proces aktualizacji produktu przebiega bardzo prosto, po wykonaniu podwójnego kliknięcia, uzupełniają się pola tekstowe po prawej stronie. Należy

wprowadzić w nich zmiany, a następnie wybrać opcję “Update”. Zostanie zaktualizowana nie tylko baza danych produktów, lecz również istniejących rezerwacji oraz wypożyczeń, tak aby nie powstawały późniejsze błędy podczas zwracania towaru, lub wynajęcia sprzętu z rezerwacji.

Proces usuwania również jest bardzo intuicyjny. Należy wybrać produkt poprzez podwójne kliknięcie spośród występujących w tabeli, a następnie wybrać opcję “Delete”. Zostanie wtedy uruchomiona funkcja odpowiedzialna za usunięcie produktu z bazy danych oraz zmiany statusu rezerwacji na dany produkt jako anulowana. Usuwanie produktu nie przejdzie pomyślnie, jeżeli produkt, który chcemy usunąć jest obecnie wynajęty. Wyświetli się tekst z informacją, z jakiego powodu funkcja nie została wykonana.

4.4.4 Settings (Ustawienia)

Z panelu głównego można przejść do ustawień. Aby móc wejść do tej sceny, wymagane jest wpierw uwierzytelnienie danych. Program sprawdza czy użytkownik, który chce wejść do edycji ustawień to “admin”. Jeżeli nie, użytkownik nie będzie mógł wejść do tej sceny.

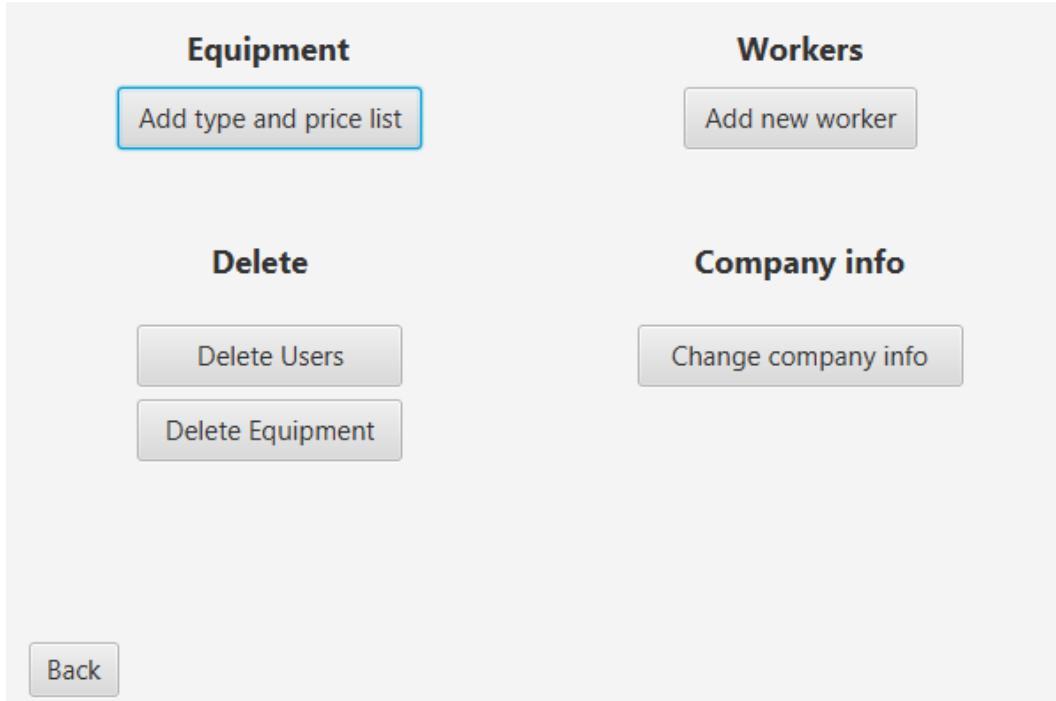


Zrzut ekranu 34 - uwierzytelnienie, dostęp do ustawień. Źródło: opracowanie własne.

Po wprowadzeniu poprawnych danych, przechodzi się do ustawień, można tutaj znaleźć takie funkcjonalności jak:

- Dodawanie typu (kategorii) oraz ustawień cennika

- Zarządzanie pracownikami (dodawanie, edycja, usuwanie)
- Masowe czyszczenie baz klientów, sprzętu
- Edycja podstawowych informacji o firmie



Zrzut ekranu 35- wygląd sceny Settings (ustawienia). Źródło: opracowanie własne.

Ustawienia - edycja typu (kategorii) oraz cennika

Po wyborze pierwszego z nich (Equipment - ang. Sprzęt), przechodzi się do ustawień, zarządzania typami (kategoriami) oraz cennikami.

The screenshot shows a table of equipment types with their rates per hour and day, and a form for adding new entries. The table includes rows for ski, bike, kayak, ręcznik, snowboard, and car. The form on the right has fields for Type, Hour, and Day, with corresponding input boxes. At the bottom are buttons for Delete, Update, ?, Add, and Back.

type	hour	day
ski	20	70
bike	10	50
kayak	8	25
ręcznik	5	25
snowboard	10	25
car	100	500

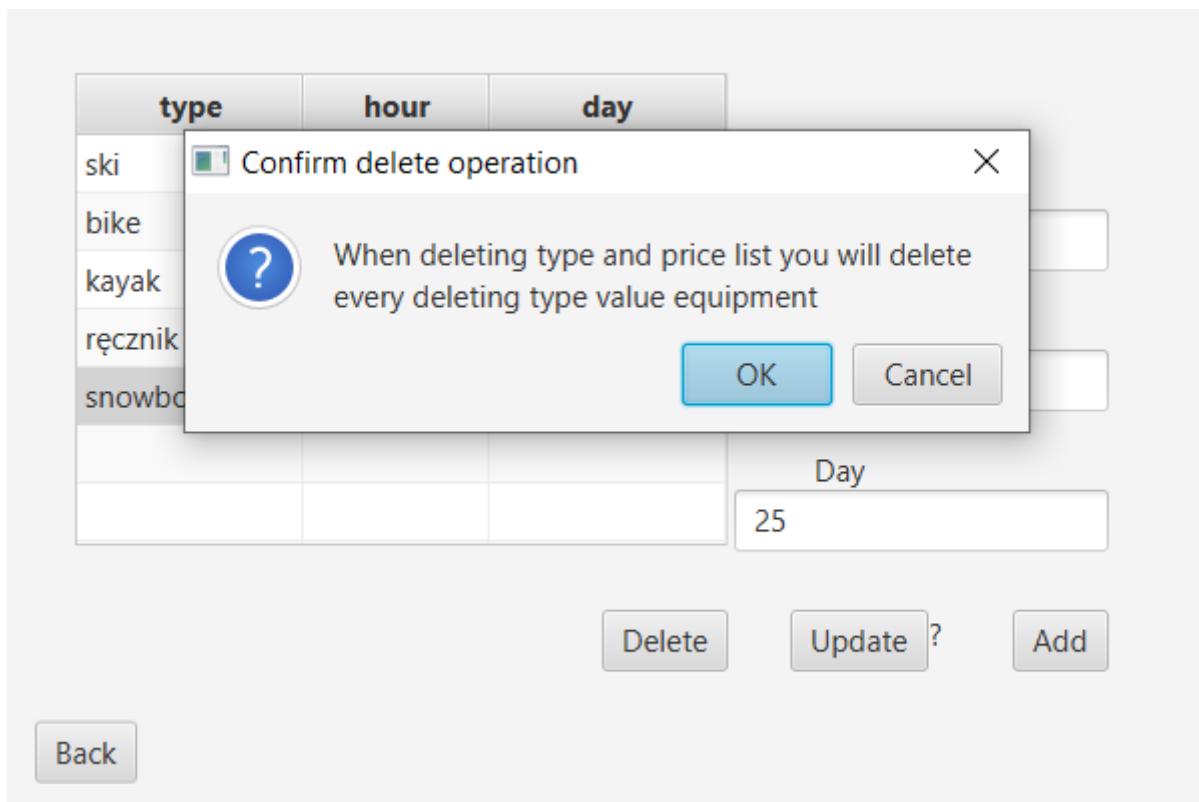
Zrzut ekranu 36 - ustawienia typu (kategorii) oraz cennika. Źródło: opracowanie własne.

Na Zrzucie Ekranu 36, po lewej stronie sceny znajduje się tabela, która wyświetla obecnie wprowadzone do systemu typy (kategorie) oraz odpowiadające im ceny godzinowe (hour), oraz za dzień (day) wypożyczenia. Dane z tej tabeli są pobierane bezpośrednio do funkcjonalności Equipment (Sprzęt), gdzie dodając nowy produkt, można wybrać typ jako jeden z dodanych tutaj obiektów. Dodatkowo obiekty z tej tabeli są pobierane do funkcjonalności "Return", z ang. zwrot, gdzie na ich podstawie jest wyliczana cena za wypożyczenie.

Po prawej stronie znajduje się kilka pól tekstowych, dzięki którym, po wypełnieniu i naciśnięciu "Add" z ang. dodaj, zostanie dodany nowy typ produktu oraz odpowiadające im ceny.

Gdy użytkownik naciśnie dwukrotnie na istniejący obiekt w tabeli, możliwa będzie edycja typu, ceny za godzinę oraz dzień, poprzez aktualizację danych, które wyświetla się w polach tekstowych oraz naciśnięciu przycisku "Update" (z ang. "aktualizuj"). Zaktualizowana nazwa obiektu, typu, zamieni wszelkie wartości nazwy typu istniejących już produktów w bazie danych.

Dodatkowo istnieje możliwość usunięcia takiego typu, spowoduje to również usunięciem wszystkich produktów w bazie danych, które są typu usuniętego obiektu. Podczas naciśnięcia przycisku "Delete", wyświetli się okno dialogowe, które ma na celu poinformować użytkownika o konsekwencjach danej czynności.



Zrzut ekranu 37 - okno dialogowe, usuwanie typu (kategorii) produktów. Źródło: opracowanie własne.

Ustawienia - edycja pracowników

Następną funkcjonalnością dostępną w ustawieniach jest zarządzanie pracownikami. Po przejściu w tę scenę, wyświetli się tabela z istniejącymi pracownikami oraz ich danymi (zrzut ekranu 38). Po prawej stronie znajdują się pola tekstowe, dzięki którym można zarządzać, edytować, usuwać, dodawać nowych pracowników wypożyczalni.

User	Name	Surname
admin	admin	admin
test	test	test

Name

Surname

User name

Password

Confirm Password

Zrzut ekranu 38 - ustawienia, zarządzanie pracownikami. Źródło: opracowanie własne.

Aby dodać pracownika, należy uzupełnić wszelkie pola tekstowe znajdujące się po prawej stronie sceny, następnie potwierdzić hasło (musi być jednakowe). Na sam koniec, po uzupełnieniu wszystkich danych należy nacisnąć przycisk “Add”. Nowy pracownik zostanie dodany do bazy danych. Od tej chwili pracownik może logować się nowo utworzonym kontem podczas uruchomienia aplikacji.

Aby edytować dane należy wybrać użytkownika poprzez podwójne naciśnięcie w tabeli, wtedy uzupełnią się wszelkie pola tekstowe, które można swobodnie edytować, oprócz hasła, gdzie należy wpisać istniejące, a następnie drugi raz wpisać poprawne hasło. Po naciśnięciu “Update” (będzie możliwe do naciśnięcia dopiero wtedy, gdy użytkownik dwukrotnie kliknie na obiekt w tabeli), zaktualizowane dane zostaną przesłane do bazy danych, a następnie tabela zostanie odświeżona, tak aby użytkownik mógł zobaczyć wprowadzone przez siebie dane.

Istnieje możliwość usunięcia użytkownika. Odbywa się to poprzez dwukrotne kliknięcie na pracownika z tabeli, a następnie przyciśnięciu guzika “Delete”. Wybrane konto zostanie usunięte na stałe z bazy danych.

Podczas gdy wprowadzone dane okazałyby się nieprawidłowe, niepełne, hasło potwierdzające nie będzie się zgadzało z wprowadzonym pierwotnie, zostanie wyświetlona informacja w górnej części ekranu, wskazująca gdzie znajduje się błąd.

Dodatkowo aby uniknąć problemu z dostępem do ustawień użytkownika “admin”, edycja tej pozycji została ograniczona do zmiany imienia i nazwiska, a usuwanie jest zablokowane. Podczas próby takiej operacji zostanie wyświetlona informacja o nielegalnej operacji, a ona sama nie zostanie wykonana (rzut ekranu 39).

The screenshot shows a user management interface. On the left is a table with columns: User, Name, and Surname. It contains two rows: one for 'admin' with values 'admin' in all three columns, and one for 'test' with values 'test' in all three columns. To the right of the table are several input fields:

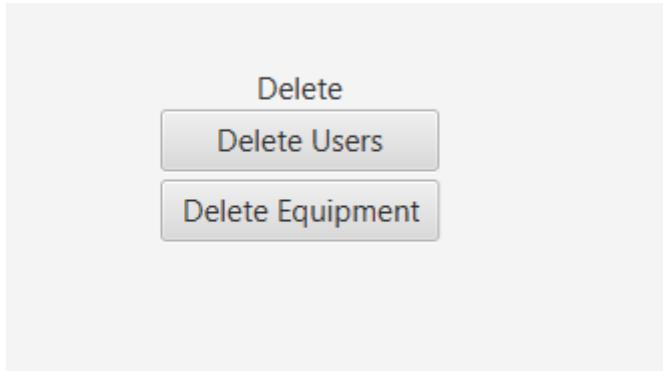
- Name: admin
- Surname: admin
- User name: admin123
- Password: (redacted)
- Confirm Password: (redacted)

At the bottom are buttons for Back, Delete, Update (which is highlighted in blue), and Add.

Zrzut ekranu 39 - próba nielegalnej zmiany nazwy użytkownika - admin. Źródło: opracowanie własne.

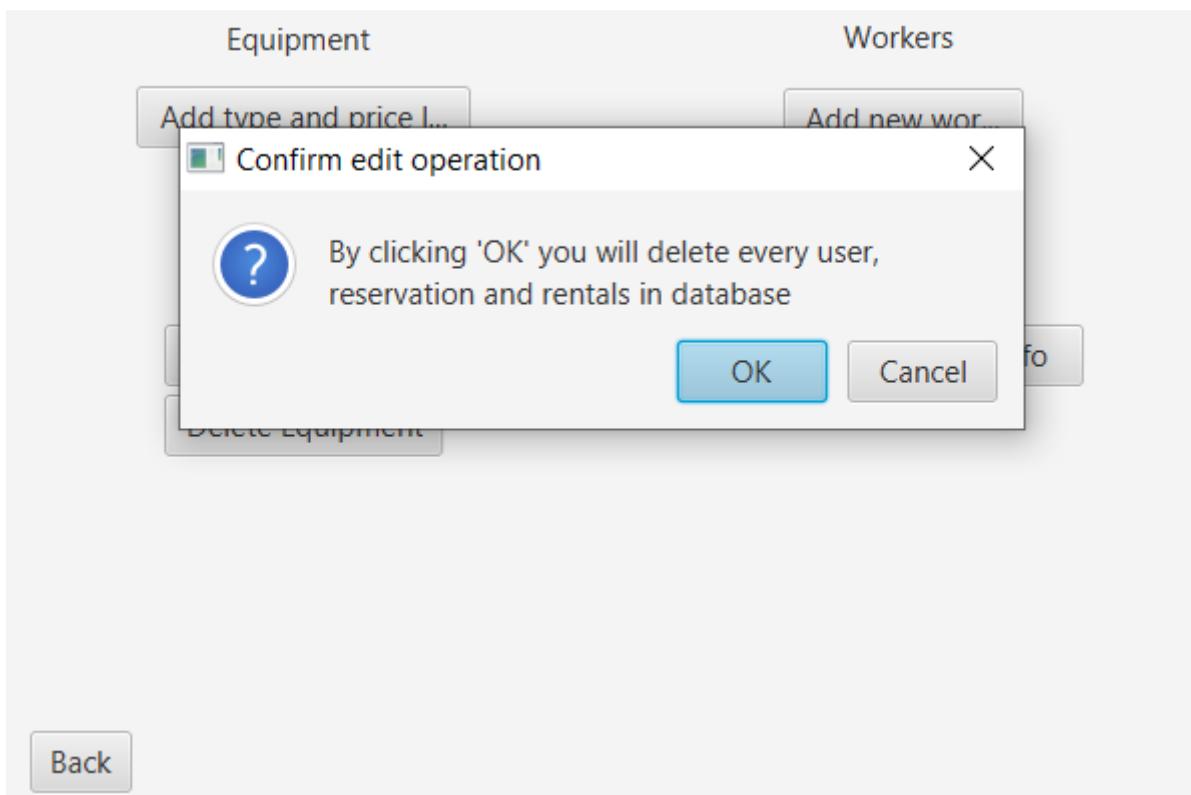
Ustawienia - masowe usuwanie klientów oraz sprzętu.

W celu czyszczenia bazy danych ze wszelkich produktów oraz klientów (może być to wykonywane cyklicznie po zakończeniu sezonu, roku, itp.) została wprowadzona opcja masowego czyszczenia bazy danych, kolekcji użytkownicy oraz sprzęt (rzut ekranu 40).



Zrzut ekranu 40 - masowe usuwanie klientów, sprzętu. Źródło: opracowanie własne.

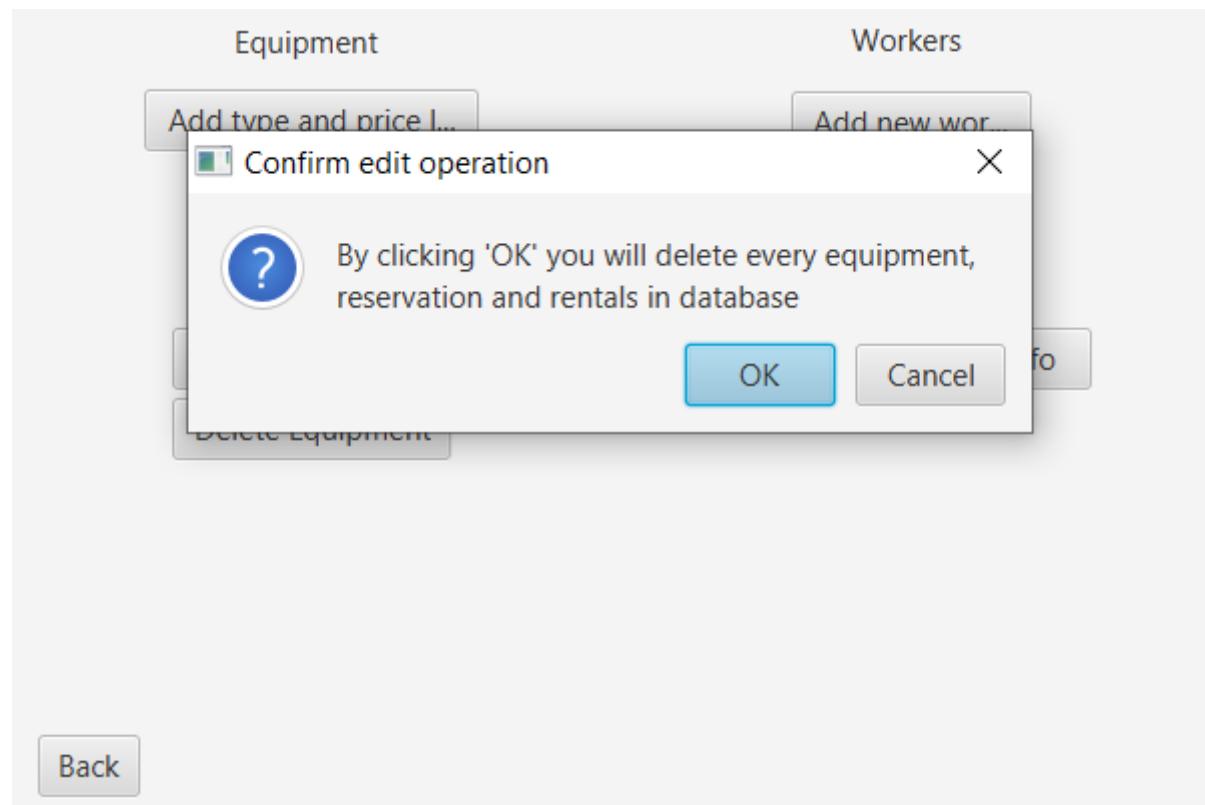
Podczas gdy użytkownik naciśnie przycisk, zostanie wyświetlone okno dialogowe, informujące o skutkach tego działania (zrzut ekranu 41). Użytkownik musi wybrać opcję “Ok”, gdy postanowi kontynuować, lub “Cancel” jeżeli zrezygnował.



Zrzut ekranu 41 - masowe usuwanie klientów. Źródło: opracowanie własne.

Operacji masowego usuwania klientów towarzyszy również czyszczenie bazy danych wszelkich rezerwacji oraz wypożyczeń. Jest to bezpośredni skutek usuwania klientów, gdyż rezerwacje oraz wypożyczenia zawierają dane klienta, dla którego świadczona jest taka usługa. Jest to wymagane działanie, aby program mógł działać poprawnie oraz aby zapobiec powstawaniu błędów w działaniu aplikacji.

Drugą masową operacją jest usuwanie sprzętu. Podczas gdy użytkownik naciśnie przycisk “Delete equipment”, zostanie również wyświetlone okno dialogowe, informujące o skutkach działania (zrzut ekranu 42).



Zrzut ekranu 42 - ustawienia, masowe usuwanie całego sprzętu. Źródło: opracowanie własne.

Ze sprzętem zostanie również usunięta baza wszystkich rezerwacji oraz wypożyczeń, gdyż te kolekcje przechowują informacje o zarezerwowanym oraz wypożyczonym sprzęcie. Jest to wymagane do poprawnego działania aplikacji, tak aby nie występowały błędy.

Ustawienia - zarządzanie podstawowymi informacjami dot. firmy

Podczas gdy użytkownik w głównym panelu ustawień wybierze opcję “Change company info”, zostanie przeniesiony do sceny zawierającej podstawowe informacje dot. firmy (zrzut ekranu 43).

Phone	Open Time
555666777	6
Email	Adress
wypożyczalnia@wypożyc:	Zakopane ul. Mickiewicza
Title	Percentage
Wypożyczalnia "Projekt Ir	0
Close Time	
20	
Back	Update

Zrzut ekranu 43 - ustawienia, zarządzanie podstawowymi informacjami dot. firmy. Źródło: opracowanie własne

Jak można zauważyć na zrzucie ekranu 43, edycja polega na zmianie wartości wpisanych w polach tekstowych, a następnie naciśnięciu “Update”.

Dane takie jak “Phone”, “Email”, “Title”, “Close Time”, “Open Time”, “Adress”, są wyświetlane klientom podczas użytkowania aplikacji webowej. Pola tekstowe “Close Time” oraz “Open Time”, umożliwiają dokonanie rezerwacji sprzętu po webowej stronie aplikacji, w godzinach otwarcia wypożyczalni. Jest to zabezpieczenie, które zapobiega wynajęcia sprzętu przez klienta przy użyciu aplikacji webowej, po godzinach zamknięcia wypożyczalni lub przed otwarciem. Zmienna, która określa procent sprzętu możliwy do zarezerwowania to “Percentage” (z. ang procent). Gdy występuje więcej niż kilka produktów o takim samym typie, tym samym modelu, producencie i rozmiarze, administrator może ustawić procent dostępnych produktów możliwych do rezerwacji online. Jest to zabezpieczenie wypożyczalni działających w dwóch modelach - modelu rezerwacji, oraz wypożyczenia sprzętu klientom na bieżąco. Użytkownik ustalając wartość na 0, lub 100 dać możliwość rezerwacji każdego, lub żadnego sprzętu, lub ustawić wartość pomiędzy tymi dwiema wartościami, a program sam podzieli ilość dostępnych produktów do rezerwacji.

Dodatkowo występują tutaj zabezpieczenia, które zapobiegają przed wprowadzeniem niepoprawnych danych np.: wprowadzenie tekstu w polu otwarcia wypożyczalni, lub wartości procentowej przekraczającej 100%. Podczas gdy program zauważy taki błąd, zostanie wyświetlona informacja w górnej części programu, a żądana operacja nie zostanie wykonana (zrzut ekranu 44).

percentage need to be set between 0 and 100

Phone 555666777	Open Time 6
Email wypożyczalnia@wypożyczalnia.pl	Address Zakopane ul. Mickiewicza
Title Wypożyczalnia "Projekt Ir"	Percentage 111
Close Time 20	

[Back](#)

[Update](#)

Zrzut ekranu 44 - ustawienia, zarządzanie podstawowymi informacjami dot. firmy, wyświetlanie błędu. Źródło: opracowanie własne.

4.4.5 Clients (Klienci)

Po naciśnięciu przycisku “Clients”, z głównego panelu aplikacji, użytkownik zostanie przeniesiony do sceny, w której będzie mógł przeglądać, edytować dane, lub dodawać i usuwać nowych lub istniejących już klientów (zrzut ekranu 45).

Name	Surname	Phone	ID Card	ID	
Jan	Greg	123123	ID_Card...	61dccce9fd...	
Janek	Jankowski	123123123	1111	61dccdbfa2...	

Name
Surname
Phone
ID Card

[Back](#)

[Delete](#)

[Update](#)

[Add](#)

Zrzut ekranu 45- funkcjonalność Clients (Klienci). Źródło: opracowanie własne.

W tabeli znajdują się istniejący klienci, których można sortować poprzez naciśnięcie górnej części danej kolumny w tabeli. Są tutaj wyświetlane wszelkie informacje, takie jak:

- imię
 - nazwisko
 - numer telefonu
 - numer dowodu tożsamości
 - ID klienta

Podczas gdy klient sam zarejestruje się w aplikacji webowej, zostanie również w tej tabeli wyświetlony, lecz bez informacji takich jak: numer telefonu, oraz numer dowodu tożsamości. Dane te są podawane pracownikowi wypożyczalni oraz weryfikowana ich poprawność podczas pierwszego przyjścia do stacjonarnego punktu. Informacje te są bardzo ważne, bo właśnie dzięki nim można bezpiecznie wypożyczyć sprzęt, a w przypadku kradzieży można zgłosić się na policję znając dane umożliwiające identyfikację osoby.

Zrzut ekranu 46 - Clients (Klienci) - osoba zarejestrowana online, bez uzupełnienia danych.

Źródło: opracowanie własne.

Pracownik dane wymagane do wypożyczenia może uzupełnić na dwa sposoby:

1. W funkcjonalności “Clients”
 2. Podczas wypożyczenia

Istnieje możliwość dodania nowego klienta w tym panelu poprzez wypełnienie danych w polach tekstowych oraz późniejszym naciśnięciu przycisku “Add”. Nowy klient zostanie dodany do bazy danych, a tabela automatycznie odświeży się, aby móc zobaczyć postęp naszej pracy. Istnieje również możliwość edycji danych, odbywa się to poprzez podwójne naciśnięcie osoby w tabeli, edycji danych w polach tekstowych, które uzupełnią się automatycznie, a następnie naciśnięciu “Update”.

Usuwanie klientów również jest możliwe, odbywa się to poprzez wybranie osoby z tabeli podwójnym kliknięciem, a następnie naciśnięcie przycisku “Delete”. Podczas tej operacji zostanie również usunięta każda rezerwacja klienta czy to potwierdzona, zakończona czy anulowana. Operacja będzie możliwa do wykonania tylko wtedy gdy dany klient nie posiada jakichkolwiek aktywnych wypożyczeń w momencie naciśnięcia przycisku “Delete”.

Podczas gdy wprowadzone dane są nieprawidłowe, numery identyfikujące są identyczne, lub klient posiada aktualne wypożyczenia, zostanie wyświetlona informacja w prawym górnym rogu, która wskaże błąd, a sama operacja nie zostanie wykonana (zrzut ekranu 47).

Zrzut ekranu 47- Clients, informacja o błędzie. Źródło: opracowanie własne.

4.4.6 Reservations (Rezerwacje)

Po przejściu z głównego panelu do funkcjonalności “Reservations” (Rezerwacje), wyświetlana zostanie tabela z obecnymi potwierdzonymi rezerwacjami.

Name	Surname	Product	Model	Product ID	Start	End	Status
Janek	Jankowski	bike	Mountain Mega 12	69696233232324	20/01/2022 16:22:30	21/01/2022 15:22:31	confirmed

Back

Rent Reserved

Zrzut ekranu 48 - Reservations, rezerwacje. Źródło: opracowanie własne.

Z podglądu, zaznaczając interesującą nas rezerwację można przejść prosto do wypożyczenia z rezerwacji poprzez podwójne kliknięcie interesującej użytkownika pozycji, lub pojedynczo zaznaczając daną rezerwację, a następnie naciskając przycisk “Rent Reserved”. Jeżeli dane użytkownika nie są w pełni uzupełnione, zostanie wyświetlona scena z polami tekstowymi, gdzie należy dokończyć uzupełnianie danych klienta.

Fill up empty client data

Name

Phone

Surname

ID Card

Back

Next

Zrzut ekranu 49- Reservation, rezerwacje - uzupełnianie niepełnych danych klienta. Źródło: opracowanie własne.

Po wypełnieniu wszystkich danych poprawnymi wartościami (w innym wypadku wyświetli błąd i operacja się nie wykona) oraz naciśnięciu przycisku “Next”, zostanie ukazana scena wypożyczenia.

Current client: **Janek Jankowski 1111**

Product ID	69696233232324	Rent		
Type	Model	Size	Product ID	Start date
No content in table				

Back

Zrzut ekranu 50 - Reservations, rezerwacje - wypożyczenie z rezerwacji. Źródło: opracowanie własne.

Jako argument przesłany został obiekt klienta, w górnej części wyświetlane są dane osoby, dla której jest obecnie wypożyczany sprzęt. Dodatkowo zostało wypełnione pole tekstowe, gdzie podajemy kod identyfikujący produktu. Aby wynająć ten dokładny sprzęt, należy nacisnąć “Rent” (zrzut ekranu 50).

Current client: **Janek Jankowski 1111**

Product ID	69696233232324	Rent		
Type	Model	Size	Product ID	Start date
bike	Mountain Mega 12	S	69696233232...	11/01/2022 01:48:38

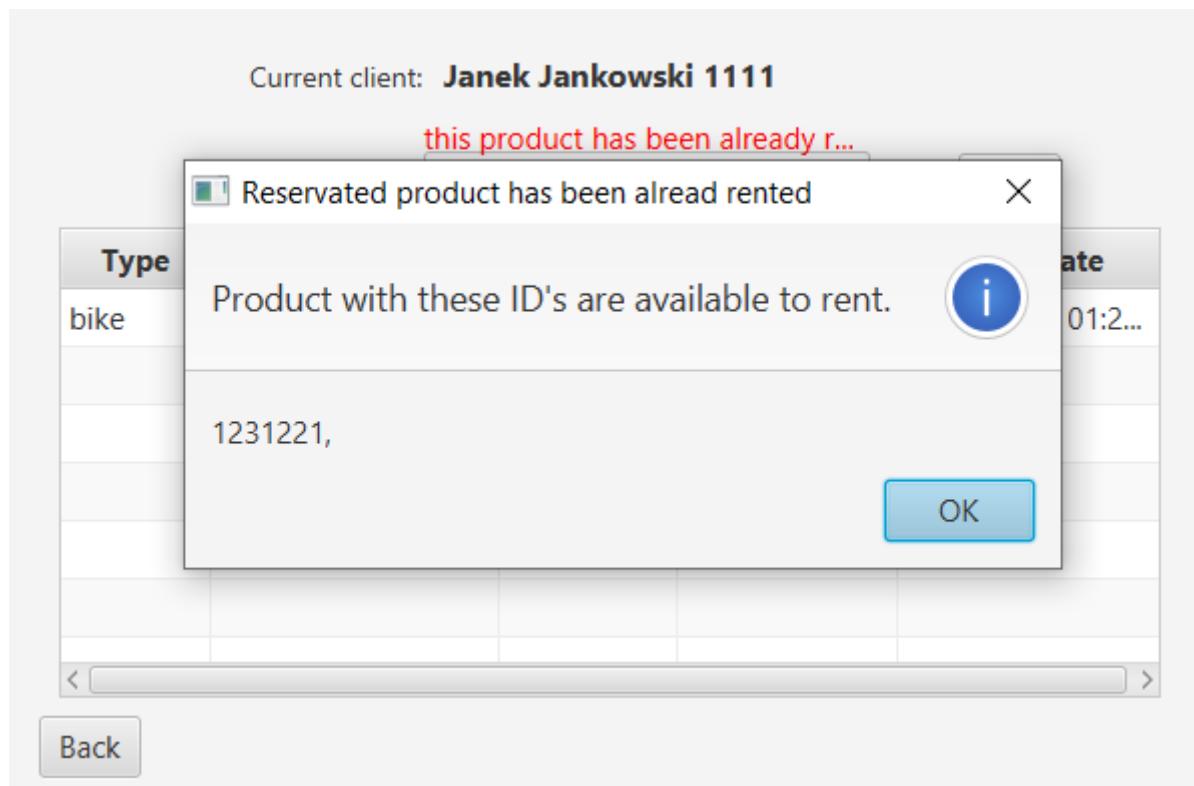
Back

Zrzut ekranu 51 - Reservations, rezerwacje - wypożyczenie z rezerwacji. Źródło: opracowanie własne.

Produkt o podanym numerze identyfikującym został wypożyczony (zrzut ekranu 51), w tabeli wyświetla się dokładne dane, takie jak:

- typ produktu
- model
- rozmiar
- numer identyfikujący
- data początkowa wraz z godziną

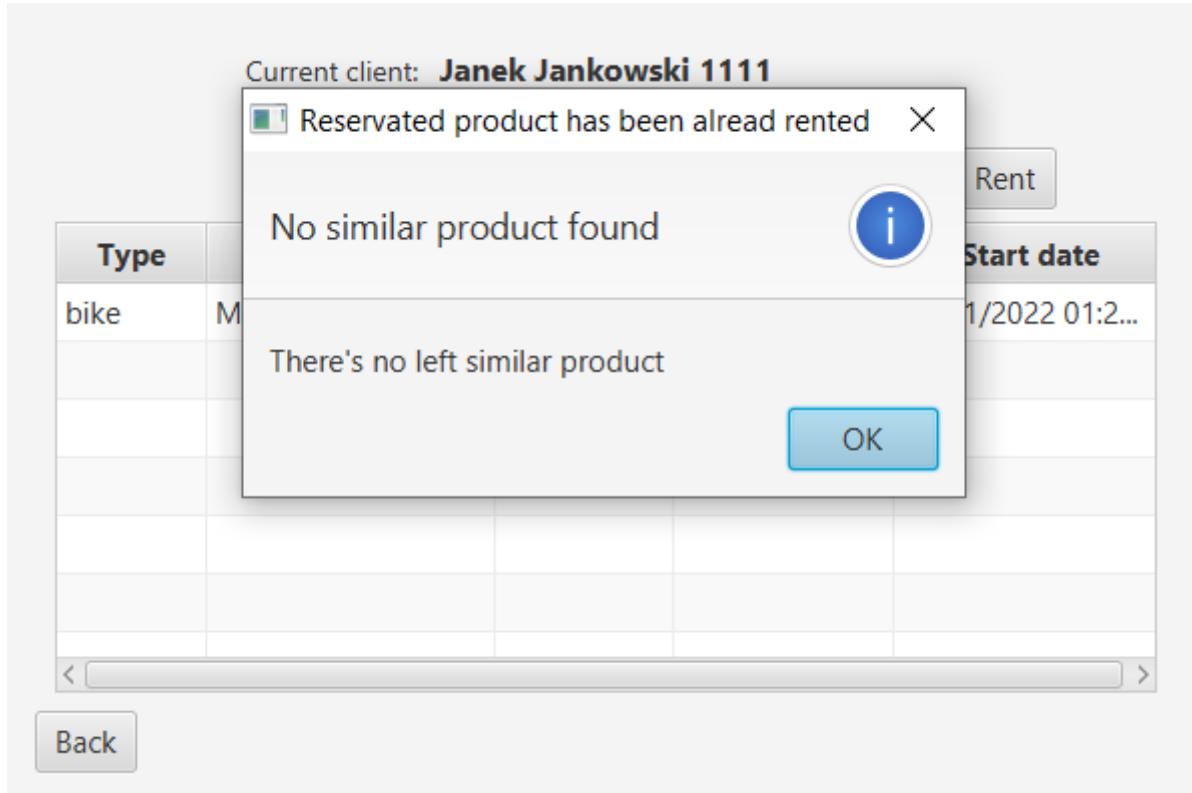
Do aplikacji została dodana również funkcjonalność sprawdzania podobnych produktów w bazie danych w przypadku gdy produkt zostanie już wcześniej wynajęty, lub klient nie zwróci na czas (zrzut ekranu 52).



Zrzut ekranu 52 - Reservations, rezerwacje - wypożyczenie z rezerwacji - okno dialogowe podobnych produktów.
 Źródło: opracowanie własne.

Produkt o podanym kodzie identyfikującym został już wcześniej wypożyczony, dlatego wyświetlane zostaje okno dialogowe z podanymi kodami identyfikującymi produktów podobnych (taki sam typ, model, producent, rozmiar, inne jedynie ID Produktu).

Jeżeli się okaże, że w systemie nie występuje więcej podobnych produktów, zostanie wyświetlone okno dialogowe informujące o danym stanie (zrzut ekranu 53).



Zrzut ekranu 53 - Reservations, reservation - wypożycz z rezerwacji - okno dialogowe - brak podobnego produktu.
Źródło: opracowanie własne.

Podeczas wykonania wypożyczenia, status danej rezerwacji jest zmieniany z “confirmed” (potwierdzony), na “done” (wykonany), zostaje zapisany w bazie danych dla informacji klienta, który może sprawdzić status archiwalnych rezerwacji i wypożyczeń w panelu aplikacji webowej. W tabeli, która wyświetla obecne potwierdzone rezerwacje, nie będzie ukazywana.

4.4.7 Rent (Wypożycz)

Jedną z podstawowych funkcjonalności, którą musi spełniać oprogramowanie służące do obsługi wypożyczalni, jest możliwość wypożyczenia sprzętu. Po naciśnięciu z głównego panelu przycisku “Rent”, użytkownik przenoszony zostaje do sceny wyboru klienta (zrzut ekranu 54).

Choose client

Name	Surname	Phone	ID Card	ID
Name	Surname	123123123	IDCard	61d72e2d80f5b...

< >

[Back](#) [Add new client](#) [Next](#)

Zrzut ekranu 54 - Rent, wypożyczenie - scena wyboru klienta. Źródło: opracowanie własne.

W głównej tabeli wyświetlanie są obecnie zarejestrowani klienci. Do pomocy szukania danego klienta z całej listy zaimplementowany został filter w górnej części sceny. Po wpisaniu danego imienia, nazwiska, numeru telefonu, kodu identyfikującego lub numeru dowodu tożsamości zostaną automatycznie wyświetleni szukani klienci.

Jeżeli klient nie jest zarejestrowany, należy wpierw dodać taką osobę do bazy danych, wykonając to można przy użyciu przycisku “Add new client” (zrzut ekranu 54).

Name

Surname

Phone

ID Card

[Back](#) [Next](#)

Zrzut ekranu 55 - Rent, wypożyczenie - dodawanie nowego klienta. Źródło: opracowanie własne.

Należy uzupełnić wszystkie dane klienta w pustych polach tekstowych, a następnie nacisnąć “Next” (zrzut ekranu 55).

The screenshot shows a software interface for managing client rentals. At the top, it displays the current client information: **NoweImie NoweNazwisko NoweIdCard**. Below this, there is a search bar labeled "Product ID" with an empty input field and a "Rent" button. A table header row is visible with columns: **Type**, **Model**, **Size**, **Product ID**, and **Start date**. The main area of the interface contains the message **No content in table**. At the bottom left, there is a blue-outlined "Back" button.

Zrzut ekranu 56 - Rent, wypożyczenie - wypożyczenie dla nowego klienta. Źródło: opracowanie własne.

Po uzupełnieniu wszystkich danych, użytkownik zostaje przeniesiony do sceny wypożyczenia (zrzut ekranu 56). Podając dany kod produktu (jeżeli jest wynajęty, lub nie istnieje wyświetli u góry błąd, a operacja się nie wykona), zostaje wypożyczony produkt.

Jeżeli pracownik wypożycza sprzęt klientowi, który ma niezupełnione dane (zrzut ekranu 57), zostanie przeniesiony do sceny, w której będzie musiał wpierw edytować wszelkie brakujące dane (zrzut ekranu 58).

Choose client

Name	Surname	Phone	ID Card	ID
Name	Surname	123123123	IDCard	61d72e2d80f5b...
TestName	TestSurname			61d74819749a...

< >

[Back](#) [Add new client](#) [Next](#)

Zrzut ekranu 57 - Rent, wypożyczenie - Edycja istniejącego klienta. Źródło: opracowanie własne.

Name

Surname

Phone

ID Card

[Back](#) [Next](#)

Zrzut ekranu 58 - Rent, wypożyczenie - edycja istniejącego klienta. Źródło: opracowanie własne.

Po przejściu do nowej sceny (zrzut ekranu 58), można zauważyć, że pola tekstowe, w których nie ma zawartości, są brakującymi danymi klienta. Należy je wypełnić, a następnie przy użyciu przycisku “Next”, wszelkie dane zostaną zapisane i otworzy się scena wypożyczenia (zrzut ekranu 59).

Current client: **TestName TestSurname TestIDCard**

Product ID	<input type="text"/>	Rent		
Type	Model	Size	Product ID	Start date
No content in table				

[Back](#)

Zrzut ekranu 59 - Rent, wypożyczenie - przejście ze sceny edycji istniejącego klienta, na scenę wypożyczenia. Źródło: opracowanie własne.

System obsługi wypożyczalni posiada również funkcję usuwania wypożyczenia pod warunkiem, że długość wynajmu jest krótsza niż godzina. Jest to zabezpieczenie przed ewentualnym błędem ze strony pracownika, lub zmianą decyzji klienta podczas wypożyczenia sprzętu. Aby tego dokonać należy dwukrotnie kliknąć na interesujący nas obiekt w tabeli, a produkt (jeżeli nie przekroczył czasu) zostanie zwrócony bez żadnych konsekwencji. W innym przypadku zostanie wyświetlona informacja dot. błędu.

4.4.8 Return (Zwrot)

Z głównego menu aplikacji można przejść do sceny zwrotu towaru, poprzez naciśnięcie przycisku “Return” (z ang. zwrot). Użytkownik zostanie przeniesiony do sceny, w której można zauważać dwie tabele oraz puste pole tekstowe (zrzut ekranu 60). Zwrot towaru następuje po wyszukaniu wypożyczonego produktu poprzez wpisanie kodu do pustego pola tekstowego i naciśnięciu “Return”. Obydwie tabele zaktualizują się i wyświetią dane. Góra tabela przedstawiona na zrzucie ekranu 60 przedstawia aktualne wypożyczenia danego klienta, z kolei druga poniżej aktualne produkty, które klient chce zwrócić.

Return by Product ID

Currenct client

Type	Model	Size	Product ID	Start date
No content in table				

Type	Model	Size	Product ID	Start date	Finish date
No content in table					

Zrzut ekranu 60 - Return, zwrot - scena zwrotu. Źródło: opracowanie własne.

Return by Product ID

Currenct client Jan Greg 123 this product has been already returned

Type	Model	Size	Product ID	Start date
ski	Ski Runner x8	152	2	11/01/2022 01:21:49
< [] >				

Type	Model	Size	Product ID	Start date	Finish date
bike	Mountain Mega 12	S	6969623323...	11/01/2022 01:30:42	11/01/2022 01:32:45
< [] >					

Price: 10

Zrzut ekranu 61 - Return, zwrot - zwrot towaru po wybraniu danego numeru identyfikującego produktu, wraz z informacją o obecnie zwróconym produkcie. Źródło: opracowanie własne.

Z górnej tabeli aktualnego wypożyczenia poprzez podwójne kliknięcie w interesujący nas obiekt można przenieść wypożyczony sprzęt do niższej tabeli zwrotu aktualnego wypożyczenia.

Podczas gdy produkt wyświetla się w dolnej tabeli zwrotu aktualnie wypożyczonego towaru, zostaje naliczona cena bazując na cenniku podanym w ustawieniach dla danego typu sprzętu oraz długości wypożyczenia. Jeżeli cena godzinowa przekracza stawkę dzienną, jest naliczana ta druga.

Po naciśnięciu przycisku “Finish” (z ang. zakończ), nastąpi zwrócenie towaru. Zostanie zaktualizowany stan wypożyczenia jako zwrócony (false), pracownik wypożyczalni nie będzie widział więcej tego wypożyczenia, lecz klient w panelu aplikacji webowej, będzie mógł zobaczyć archiwalne rezerwacje lub wypożyczenia.

Jeżeli podamy do pola tekstowego kod identyfikujący produktu, który nie jest wypożyczony, nie istnieje lub został już przeniesiony do tabeli zwrotu, zostanie wyświetlona informacja o błędzie, a sama operacja zwrócenia towaru nie wykona się (rzut ekranu 61).

5. Podsumowanie

Projekt systemu oraz wykonanie odrębnych aplikacji webowej oraz desktopowej zaprezentowany w niniejszej pracy ukazuje pewien podstawowy zakres dostosowania świadczonych usług do specyfiki pracy wypożyczalni. Jest również odpowiedzią na podstawowy problem klientów w XXI wieku, jakim wciąż jest niewielki procent wypożyczalni umożliwiających dokonanie rezerwacji przez Internet. Należy jednak mieć świadomość, że komercyjne oprogramowanie tworzone przez firmy jest rozwijane przez dziesiątki, a nawet setki programistów, którzy czuwają nad jakością kodu, bezpieczeństwem danych, wdrażaniem nowych funkcjonalności oraz modernizowaniu bazy kodu do najnowszych technologii. W przedstawionym projekcie został wykorzystany zbiór technologii używanych w przedsiębiorstwach do tworzenia kodu źródłowego. Nie zmienia to faktu, że technologie jakich się używa, z różnych względów, zmieniają się bardzo często. Jest to spowodowane kończącym się wsparciem dla używanych bibliotek, konkurencją na rynku która jest zmuszona wprowadzać nowe, innowacyjne rozwiązania aby osiągnąć zysk, ale również zmianą podejścia do programowania i dążeniem do łatwości w długoterminowym utrzymaniu kodu.

Dzięki dokładnemu przeglądowi dostępnych technologii oraz późniejszej dogłębnej analizie, poszerzono swoje umiejętności w tworzeniu oprogramowania zarówno modułu internetowego, jak i komputerowego oraz wszelkich działań potrzebnych do wykonania tytułowej pracy. Dzięki temu, iż

praca jest wykonywana w grupie dwuosobowej, rozwinięto również umiejętności współpracy oraz kooperacji.

5.1 Wnioski

Podczas tworzenia aplikacji zarówno webowej jak i desktopowej została wykonana praca wymagająca różnych, niekiedy odrębnych od siebie umiejętności nabyczych w toku studiowania. Nie tylko te były wykorzystywane. Do stworzenia owego systemu należało poszerzać swoje umiejętności i stale się rozwijać. Pokazuje to jak własny rozwój jest potrzebny w czasach zmieniających się technologii.

Praca w grupie, pomimo jasnego podziału obowiązków również jest ważnym aspektem pracy nad złożonymi aplikacjami. Podczas gdy brakowało komunikacji pomiędzy twórcami, powstawały trywialne błędy w komunikacji dwóch modułów, które po ustaleniu szczegółów oraz wspólnej pracy nad ich naprawą zostały wyeliminowane. Pokazuje, to jak ważna jest synchroniczna praca całego zespołu.

Dobór odpowiednich technologii przed rozpoczęciem pracy nad aplikacjami jest bardzo ważny. Należy przeanalizować dostępne technologie na rynku i wybrać najlepsze dla danego projektu, tak aby praca nad systemem przebiegła pomyślnie, a sam projekt służył jak najdłużej. Wybór odpowiednich technologii może bowiem zapewnić możliwości dalszego rozwoju oprogramowania, a początkowo, źle wybrana technologia wiąże się z poświęceniem dużej ilości czasu, aby skonwertować projekt do innej technologii w przyszłości. Z kolei nawet jeśli wybierzymy technologię, która jest nowoczesna, trzeba sprawdzić czy najnowsza wersja zapewnia wszelkie funkcje, jakich potrzebujemy. Okazuje się, że często lepszym rozwiązaniem jest wybraniem nieco starszej wersji, ale z zapewnieniem długoterminowego wsparcia i wieloma źródłami, z których można skorzystać w celu doszkalania się z ich obsługi.

Ostatnią rzeczą, jest elastyczność. Często, lepszym pomysłem jest odrzucenie wyboru, który się dokonało, nawet jeśli poświęcono sporo czasu na jego implementację. Zmiany należy dokonać, jeśli na przykład brakuje dostarczanych i ważnych funkcjonalności, a ich implementacja samemu jest zbyt czasochłonna, lub też nie pozwala na implementację, z takim współczynnikiem bezpieczeństwa jak zewnętrzne, komercyjne rozwiązania.

Mając powyższe rzeczy na uwadze, został wykonany projekt dyplomowy “Projekt i wykonanie systemu zarządzania wypożyczalnią oraz internetowej obsługi klienta”. Twórcy są

świadomi, że istnieje wiele możliwości rozwoju takiego oprogramowania, aby stawało się coraz lepsze i odpowiadało na ciągle rosnące potrzeby rynku.

5.2 Możliwości rozwoju

Dodając nowe funkcjonalności należy mieć na uwadze dalsze możliwości osób zaangażowanych w usprawnianie kodu. Proces tworzenia oprogramowania to złożony proces, który nie kończy się na napisaniu samego kodu źródłowego, a wymaga jego modernizowania i zapewnienia bezpieczeństwa jak i dostosowywaniu obecnych funkcjonalności do zmieniających się czasów. System można rozwijać pod wieloma względami. Jednym z nich jest praca nad polepszeniem wrażeń graficznych, szczególnie zadbać o wygląd i intuicyjność interfejsu desktopowego, a w aplikacji webowej atrakcyjność prezentacji produktów. Kolejną rzeczą jest dodawanie coraz to bardziej szczegółowych opcji konfiguracyjnych dla wypożyczalni, na przykład, zamiast udostępniać jedynie sztywny cennik za godzinę oraz za dobę można wprowadzić system kilkugodzinny lub tygodniowy. Ostatecznie, można wprowadzić pewne mechanizmy logiki rozmytej czy uczenia maszynowego, aby dostarczać rozwiązań inteligentnego kalkulowania ceny, w zależności od warunków, takich jak wysoki, czy niski sezon wynajmu, lub aktualną ilość wypożyczeń.

Bibliografia

[1] CerTech Wypożyczalnia,

<https://www.cer-tech.com/naszeprodukty/wypożyczalnia/35-wypożyczalnia>, [dostęp: 15.11.2021]

[2] S|R Rental, <https://sportsrental.de/products/sr-rental/?lang=en>, [dostęp: 15.11.2021]

[3] Skipline <https://skipline.info/en/>, [dostęp: 15.11.2021]

[4] Aplikacja Comotel Rental, <https://www.ulisses.pl/pl/rozwiazania/wynajem>, [dostęp: 15.11.2021]

[5] OKHTTP3,

<https://javadoc.io/doc/com.squareup.okhttp3/okhttp/3.14.9/okhttp3/package-summary.html>,

[dostęp: 3.12.2021]

[6] MongoDB Java driver sync, <https://mongodb.github.io/mongo-java-driver/>, [dostęp: 18.12.2021]

[7] BSON (MongoDB): <https://mvnrepository.com/artifact/org.mongodb/bson/4.4.0>, [dostęp: 16.12.2021]

[8] AES, Rijndael

,<https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1>, [dostęp: 16.12.2021]

[9] AES jako standard szyfrowania,

<https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf>, [dostęp: 13.12.2021]

[10] Zmiana algorytmu szyfrowania aplikacji mObywatel,

<https://www.gov.pl/web/cyfryzacja/200-tysiecy-testerow-nowego-mobywatela>, [dostęp: 29.12.2021]

[11] Package Java Crypto,

<https://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html>, [dostęp: 29.12.2021]

[12] Package Java Security

<https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html>, [dostęp: 29.12.2021]

[13], Panel dewelopera Google, <https://console.cloud.google.com/apis/dashboard>, [dostęp: 29.11.2021]

[14] Popularność npm,

<https://developers.slashdot.org/story/17/01/14/0222245/nodejss-npm-is-now-the-largest-package-registry-in-the-world>, [dostęp: 26.11.2021]

[15] Opinie użytkowników o nowej wersji Facebooka,

<https://www.thesun.co.uk/tech/12465795/facebook-new-design-permanent-september/>, [dostęp: 13.11.2021]

[16] Opinie środowiska deweloperskiego o nowej wersji Facebooka,

<https://stackoverflow.com/questions/49791865/why-is-facebook-using-react-instead-of-more-performant-native-js>, [dostęp: 13.11.2021]

[17] Wywiad z Benjaminem Pasero, jednym z twórców Microsoft Visual Code,

<https://www.git-tower.com/blog/developing-for-the-desktop-vscode/> [dostęp: 2.12.2021]

[18] Fragment dokumentacji biblioteki React, rozdział o wzorcach projektowych,

<https://reactjs.org/docs/design-principles.html> [dostęp: 4.01.2022]

[19] Java LTS, <https://blogs.oracle.com/javamagazine/post/java-long-term-support-lts-data> [dostęp: 05.01.2022]