

Dokumentacja

Tytuł: Obsługa sklepu rowerowego

Przedmiot: Zaawansowane techniki internetowe

Wykonawca: Krzysztof Żelazny

1. Wprowadzenie

Projekt dotyczący obsługi sklepu rowerowego, jego celem jest uproszczenie pracy i wprowadzenie i pozwolenie na tworzenie zamówień przez klientów. Możliwości aplikacji:

- Rejestracja i logowanie się do konta
- Dodawanie nowych przedmiotów do bazy danych
- Dodawanie nowych zamówień i możliwość zmiany ich statusu
- Możliwość zobaczenie wszystkich obecnych i wcześniej istniejących przedmiotów w bazie

2. Technologie

Java

Java to uniwersalny język programowania o wysokim poziomie, znany z niezależności platformowej, bezpieczeństwa, wydajności oraz dużej liczby bibliotek i wsparcia społeczności. Z powyższych powodów Java została wybrana jako główny język pisania backendu aplikacji.

Spring Boot

Java Spring Framework (Spring Framework) to popularna platforma typu open source na poziomie przedsiębiorstwa do tworzenia autonomicznych aplikacji klasy produkcyjnej, które działają na wirtualnej maszynie Java (JVM).

Java Spring Boot (Spring Boot) to narzędzie, które sprawia, że tworzenie aplikacji internetowych i mikro usług za pomocą Spring Framework jest szybsze i łatwiejsze dzięki trzem podstawowym możliwościom:

Automatyczna konfiguracja

Uparte podejście do konfiguracji

Możliwość tworzenia samodzielnych aplikacji

Te funkcje współpracują ze sobą, aby zapewnić narzędzie do konfigurowania aplikacji opartej na platformie Spring przy minimalnej konfiguracji i konfiguracji. Aplikacje Spring Boot można również optymalizować i uruchamiać za pomocą środowiska uruchomieniowego Open Liberty.

Vue.js

Vue to framework JavaScript do tworzenia interfejsów użytkownika. Opiera się na standardowym HTML, CSS i JavaScript i zapewnia deklaratywny, oparty na komponentach model programowania, który pomaga wydajnie rozwijać interfejsy użytkownika o dowolnej złożoności.

Tailwind

Tailwind CSS działa poprzez skanowanie wszystkich plików HTML, komponentów JavaScript i wszelkich innych szablonów w poszukiwaniu nazw klas, generowanie odpowiednich stylów, a następnie zapisywanie ich w statycznym pliku CSS.

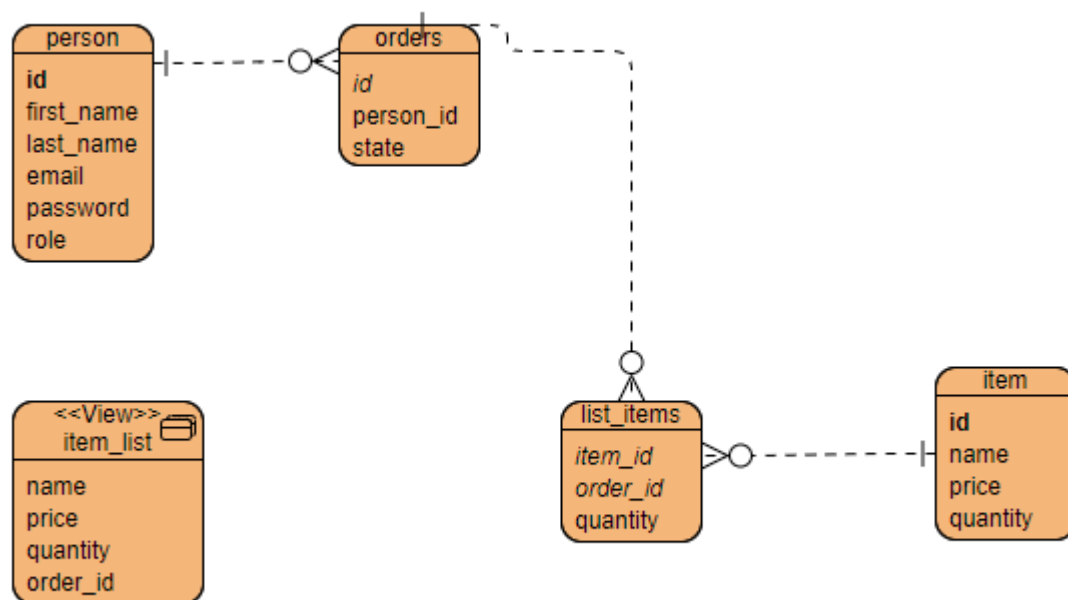
Jest szybka, elastyczna i niezawodna — bez czasu pracy.

Postgres

PostgreSQL zawiera wiele funkcji, które mają pomóc programistom w tworzeniu aplikacji, administratorom w ochronie integralności danych i tworzeniu środowisk odpornych na uszkodzenia, a także w zarządzaniu danymi bez względu na to, jak duży lub mały jest zestaw danych. Oprócz tego, że jest darmowy i open source, PostgreSQL jest wysoce rozszerzalny. Na przykład możesz definiować własne typy danych, budować niestandardowe funkcje, a nawet pisać kod z różnych języków programowania bez ponownej kompilacji bazy danych!

3. Baza danych

Baza danych składa się z 4 tabel i jednego widoku.



Rysunek 1) Wykres ERD stworzony w VisualParadigm Online

Kod inicjalizujący bazę danych:

```
create table person
(
    id          varchar(100) NOT NULL primary key,
    first_name  varchar(30)  NOT NULL,
    last_name   varchar(30),
    email       varchar(50)  NOT NULL,
    password    text         NOT NULL,
    role        varchar(20)  NOT NULL
);

create table item
(
    id          varchar(30)      NOT NULL primary key,
    name        varchar(30)      NOT NULL,
    price       double precision NOT NULL,
```

```

        quantity integer          NOT NULL
    );

create table orders
(
    id          varchar(30) NOT NULL primary key,
    person_id   varchar(30) NOT NULL,
    state       integer      NOT NULL,
    FOREIGN KEY (person_id) REFERENCES person (id)
);

CREATE TABLE list_items
(
    item_id   varchar(30) NOT NULL,
    order_id  varchar(30) NOT NULL,
    quantity  integer      NOT NULL,
    FOREIGN KEY (item_id) REFERENCES item (id),
    FOREIGN KEY (order_id) REFERENCES orders (id)
);

CREATE VIEW item_list AS
SELECT
    i.name AS name,
    i.price AS price,
    il.quantity AS quantity,
    il.order_id AS order_id
FROM
    list_items AS il
    LEFT JOIN item AS i ON i.id = il.item_id;

```

4. Struktura i Architektura

MVC

Zastosowana Struktura Model -View-Controller (MVC) to wzorzec architektoniczny/projektowy, który dzieli aplikację na trzy główne komponenty logiczne: Model , View i Controller . Każdy komponent architektoniczny jest zbudowany tak, aby obsługiwać określone aspekty rozwoju aplikacji. Izoluje od siebie warstwę logiki biznesowej i prezentacyjną.

Pliki backendu znajdują się w folderze backend/src/main.

Backend zajmuje się Modelem i Kontrolerem w projekcie Kontrolery znajdują się backend/src/main/jav/com.example.zti/controller

Kontroler jest komponentem, który umożliwia połączenie między widokami i modelem, więc działa jako pośrednik. Kontroler nie musi się martwić o obsługę logiki danych, po prostu mówi modelowi, co ma robić. Przetwarza całą logikę biznesową i przychodzące żądania, manipuluje danymi za pomocą komponentu Model i wchodzi w interakcję z View , aby renderować ostateczny wynik.

Service są to klasy odpowiadające za logikę biznesową, w której wykonywane są operacje.

Modele znajdują się w poszczególnych folderach ../sql gdzie uzyskujemy modele danych takie same jak w bazie danych. Komponent Model odpowiada całej logice związanej z danymi, z którą

pracuje użytkownik. Może on reprezentować dane, które są przesyłane między komponentami View i Controller lub dowolne inne dane związane z logiką biznesową. Może dodawać lub pobierać dane z bazy danych. Odpowiada na żądanie kontrolera, ponieważ kontroler nie może sam wchodzić w interakcję z bazą danych. Model wchodzi w interakcję z bazą danych i zwraca wymagane dane kontrolerowi.

View odwołuje się do Komponentów, a Komponenty odwołują się do Serwisów, a Serwisy odwołują się do Modelu, a Model do bazy danych.

Pliki frontendu znajdują się w folderze frontendv2.

Komponent View jest używany do całej logiki UI aplikacji. Generuje interfejs użytkownika dla użytkownika. Widoki są tworzone przez dane, które są zbierane przez komponent modelu, ale te dane nie są pobierane bezpośrednio, ale przez kontroler. Oddziałuje on tylko z kontrolerem.

W tej aplikacji do stworzenia komponentu View zastosowano vue.js. Utworzono za jego pomocą aplikację działającą w sposób SPA(Single Page Application).

Vue składa się z 3 głównych komponentów, którymi są komponenty, widoki, router.

Komponenty są to elementy, które pozwalają na budowanie modułów wielokrotnego użycia dla interfejsu użytkownika. Języki używane w komponentach to HTML jako szablon, JavaScript jako logika i style CSS.

Widoki są odpowiedzialne za renderowanie danych i interakcje z użytkownikiem. W Vue.js widoki są reprezentowane przez komponenty, które mogą być dynamicznie ładowane i renderowane w zależności od stanu aplikacji. Najczęściej, w kontekście routingu w Vue.js, widok jest związany z komponentem, który odpowiada konkretnej trasie w aplikacji.

Vue Router jest to biblioteka, która umożliwia tworzenia SPA, dzięki niej można definiować ścieżki URL i mapować je na konkretne widoki.

Autoryzacja w projekcie opiera się o role przetrzymywane w bazie danych następnie aby uzyskać dostęp do wybranych funkcji należy posiadać odpowiednią rolę. Rola jest przechowywana po zalogowaniu w local storage aby dostęp był ułatwiony.

Autentykacja jest stworzona na podstawie systemu logowania oznacza to, że użytkownik musi podać mail i hasło aby uzyskać dostęp do swojego konta. Jest to stworzone jako porównywanie parametrów podanych przez użytkownika z danymi znajdującymi się w bazie danych. Jeśli dane się zgadzają użytkownik zostaje uwierzytelniony.

Rest jest to styl architektoniczny używany do tworzenia usług API. Kluczowe zasady REST obejmują używanie standardowych metod HTTP (takich jak GET, POST, PUT, DELETE). REST promuje statelessness, co oznacza, że każda interakcja między klientem a serwerem jest niezależna i nie przechowuje stanu.

5. Przygotowanie do tworzenia

Potrzebne instalatory:

Java: java 21 sdk

Tailwind :
npm install --save-dev tailwindcss postcss autoprefixer
npx tailwindcss init -p

Vue.js:

npm install
npm init vue@3
npm install axios

6. API

Endpoint zwracający przedmioty, których ilość jest większa niż 0 w bazie

URL: api/items
Metoda : Get
Request: api/items
Response:

```
[  
  {  
    "id": "i002",  
    "name": "Smartphone",  
    "price": 500.0,  
    "quantity": 20  
  },  
]
```

Endpoint zwracający wszystkie przedmioty

URL: api/items/ itemExist
Metoda : Get
Request: api/items/ itemExist
Response:

```
[  
  {  
    "id": "i012",  
    "name": "MoterraSLLAB71",  
    "price": 15000.0,  
    "quantity": 0  
  },  
]
```

]

Endpoint modyfikujący ilość przedmiotów

URL: api/items/i001/modifyItem/1

Metoda : Get

Request: api/items/i001/modifyItem/1

Endpoint tworzący nowy przedmiot

URL: api/items

Metoda : Post

Request: api/items

Body: {

 "name": "LapierreSensium300",

 "price": 5000.0,

 "quantity": 2

}

Endpoint modyfikujący ilość przedmiotów

URL: api/items

Metoda : Put

Request: api/items

Body: {

 "id": "LapierreSensium300",

 "quantity": 1

}

Endpoint tworzący token

URL: api/token

Metoda : Post

Request: api/token

Response:

Endpoint tworzący order

URL: api/order

Metoda : Post

Request: api/order

Endpoint pobierający wszystkie order

URL: api/order

Metoda : Get

Request: api/order

Response: [

```
{
  "personId": "p001",
  "state": false,
  "itemsList": [
    {
      "id": null,
      "name": "Laptop",
      "price": 1000.0,
      "quantity": 1
    },
    {
      "id": null,
      "name": "Smartphone",
      "price": 500.0,
      "quantity": 2
    }
  ],
  "value": 2000.0
},
]
```

Endpoint pobierający order po id

URL: api/order/o002

Metoda : Get

Request: api/order/o002

Response: {

"personId": "p002",

"state": false,

"itemsList": [

{

"id": null,


```
        "name": "Headphones",
        "price": 100.0,
        "quantity": 3
    }
],
"value": 300.0
}
```

Endpoint pobierający wszystkie orderS

URL: api/orders

Metoda : Get

Request: api/orders

Response:

```
[
  {
    "id": "o002",
    "personId": "p002",
    "state": false
  },
  {
    "id": "o001",
    "personId": "p001",
    "state": false
  }
]
```

Endpoint pobierający wszystkie orders

URL: api/orders/o001

Metoda : Get

Request: api/orders/o001

Response:

```
{
  "id": "o001",
  "personId": "p001",
  "state": false
}
```

```
}
```

Endpoint modyfikujący stan orders

URL: api/orders/o001/ modifyState

Metoda : Put

Request: api/orders/o001/modifyState

Response:

Endpoint pobierający wszystkich user

URL: api/users/user

Metoda : Get

Request: api/users/user

Response:

Endpoint tworzący user

URL: api/users/register

Metoda : Post

Request: api/users/register

Body:

```
{
  "firsstName": "Jan",
  "lastName": "Kowalski",
  "email": "JanKowalski@gmail.com",
  "password": "password",
  "role": "user"
}
```

Endpoint tworzący ListItem

URL: api/listItem

Metoda : Post

Request: api/listItem

Body:

```
{
  "item_id": "i001",
  "person_order_id": "o001",
  "quantity": "1"
}
```

Response:

Endpoint zwracający ListItem

URL: api/listItem

Metoda : Get

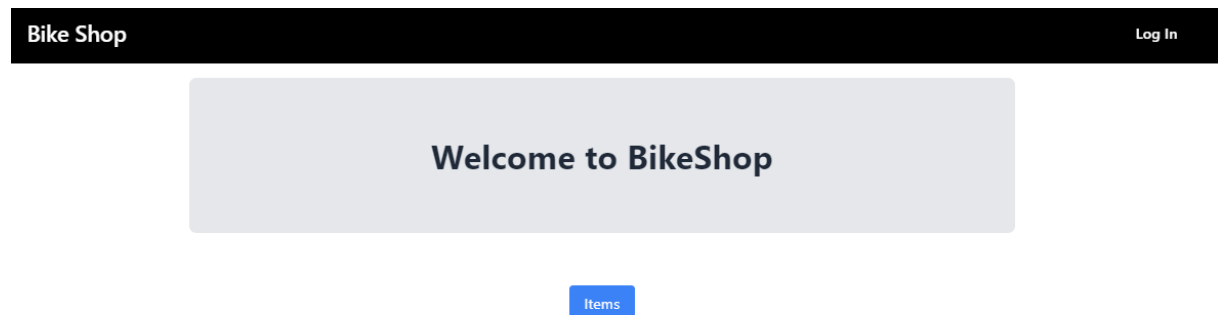
Request: api/listItem

Response:

```
{  
  "item_id": "smartwatch",  
  "person_order_id": "admin.mkxd.id.Dbbt.id",  
  "quantity": 2  
}
```

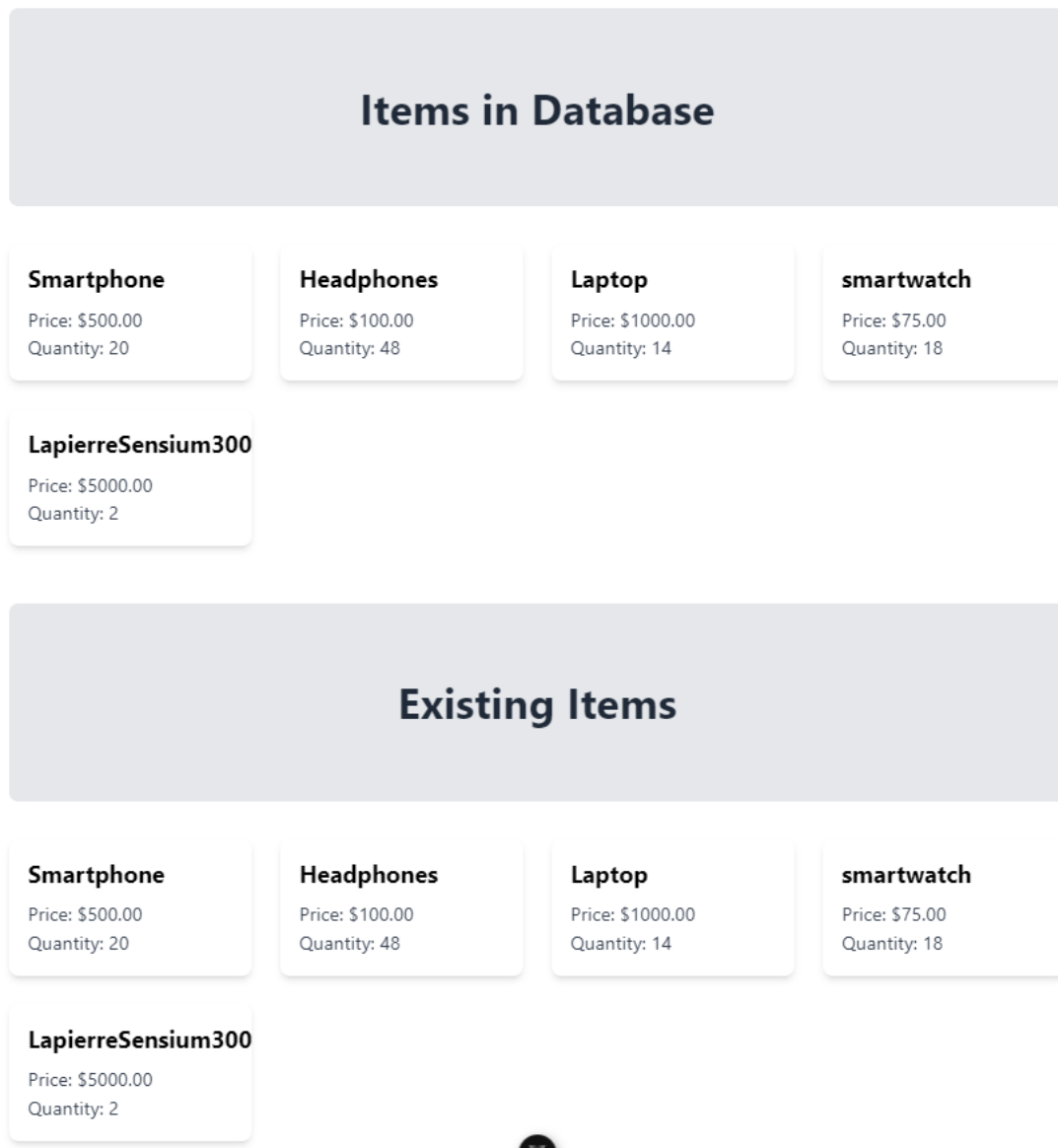
7. Prezentacja działania

Strona startowa



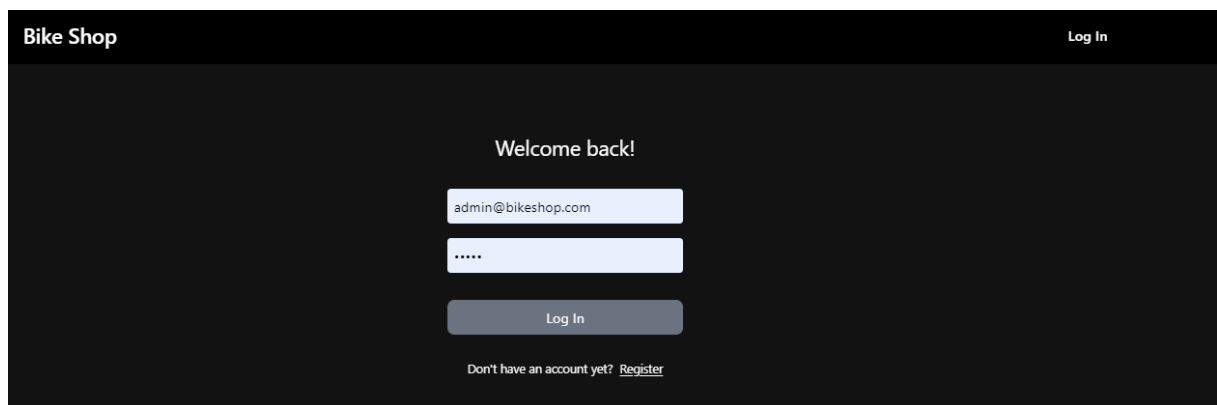
Rysunek 2) Strona startowa

Każdy odwiedzający stronę może sprawdzić jakie przedmioty znajdują się w sklepie aby mógł dokonać wybory. Należy kliknąć na Items.



Rysunek 3) Strona Items

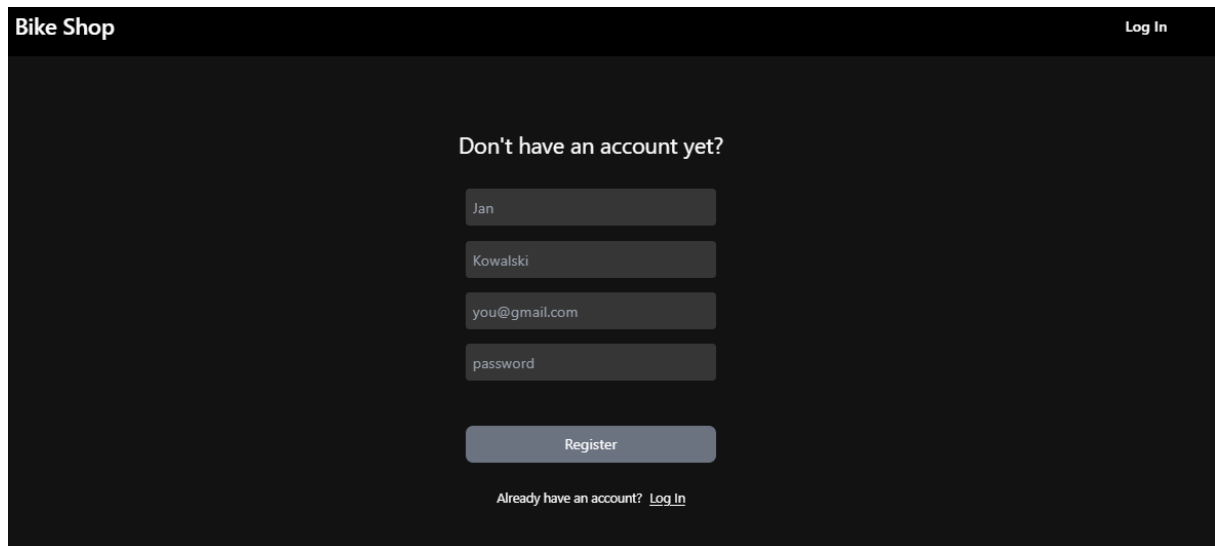
Aby zobaczyć więcej funkcjonalności należy się zalogować. W prawym górnym rogu znajduje się Log in, które należy kliknąć. Następnie zostaniemy przekierowani na stronę.



Rysunek 4) Strona logowania

Login do admina: admin@gmail.com
Hasło: admin

Dla użytkowników nie posiadających konta należy utworzyć konto klikając w podlinkowany napis „Register” przekieruje on na stronę rejestracji.



Bike Shop Log In

Don't have an account yet?

Jan

Kowalski

you@gmail.com

password

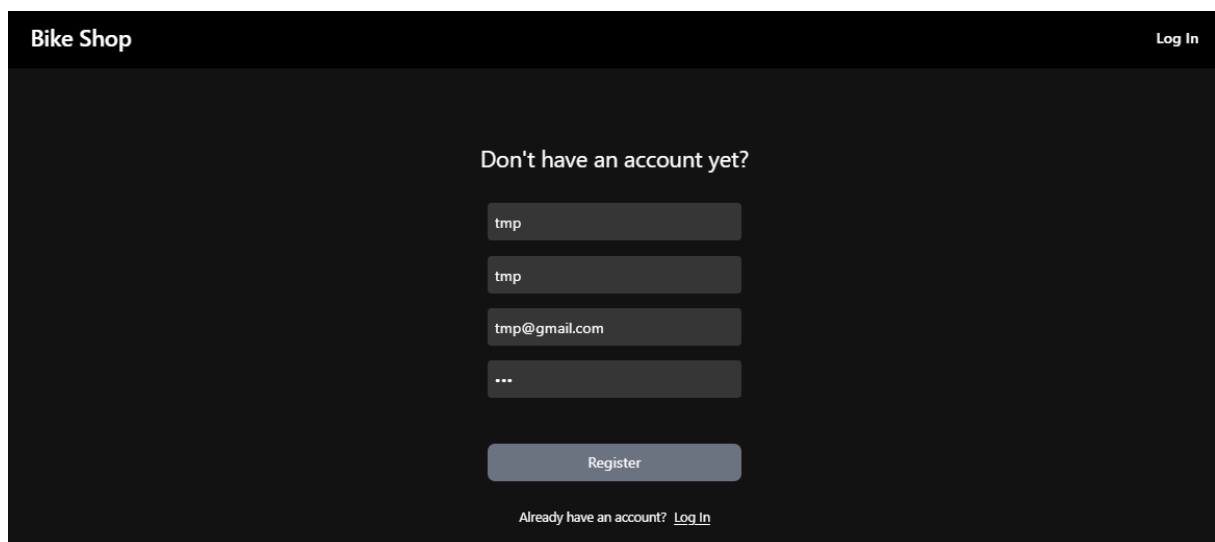
Register

Already have an account? [Log In](#)

Rysunek 5) Strona rejestracji

Zostanie utworzony nowy użytkownik z danymi:

Login: tmp@gmail.com
Hasło: tmp



Bike Shop Log In

Don't have an account yet?

tmp

tmp

tmp@gmail.com

...

Register

Already have an account? [Log In](#)

Rysunek 6) Utworzenie użytkownika tmp

Po wypełnieniu formularza należy kliknąć przycisk register, który spowoduje utworzenie konta.

Następnie należy się zalogować na to konto.

Jeżeli strona nie została przeładowana należy zrobić to ręcznie.

Po zalogowaniu strona startowa wygląda:

Welcome to BikeShop

[Items](#)[Order](#)

Rysunek 7) Strona startowa po zalogowaniu

Pojawiło się nowe okno order.

Create New Order

Items

Select an item

▼

1

Remove

[Add Item](#)

Wybierz element z listy.

[Create Order](#)

Rysunek 8 Okno Order

W tym oknie zalogowany użytkownik może stworzyć zamówienie.

Create New Order

Items

smartwatch - \$75.00

▼

2

Remove

LapierreSensium300 - \$5000.00

▼

1

Remove

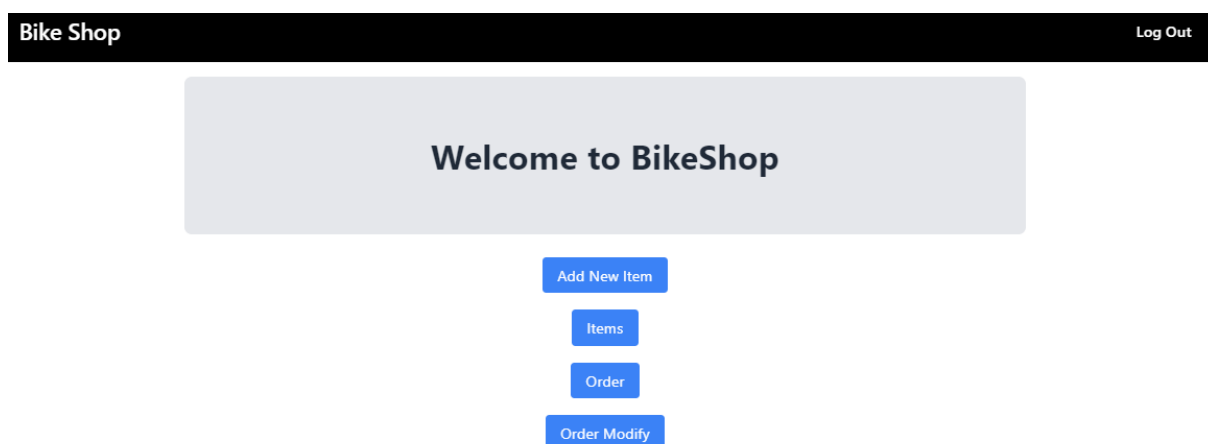
[Add Item](#)[Create Order](#)

Rysunek 9 Przykładowe zamówienie

Po kliknięciu create order zostanie utworzone zamówienie. Zamówienia może sprawdzać jedynie admin dlatego teraz należy się przelogować na konto admin.

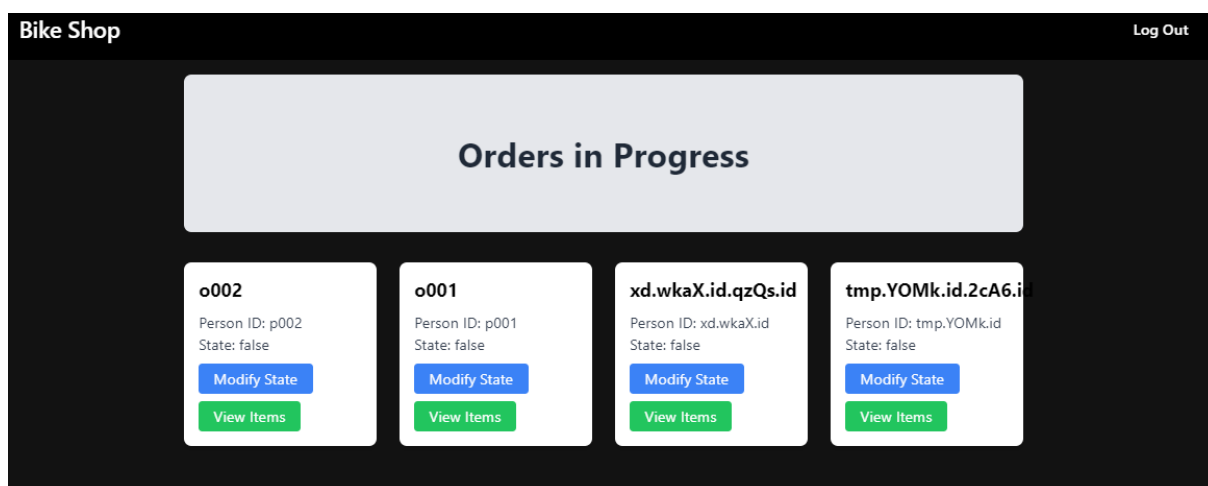
W prawym górnym rogu napis LogOut wylogowuje użytkownika.

Po zalogowaniu do konta admina strona startowa wygląda:



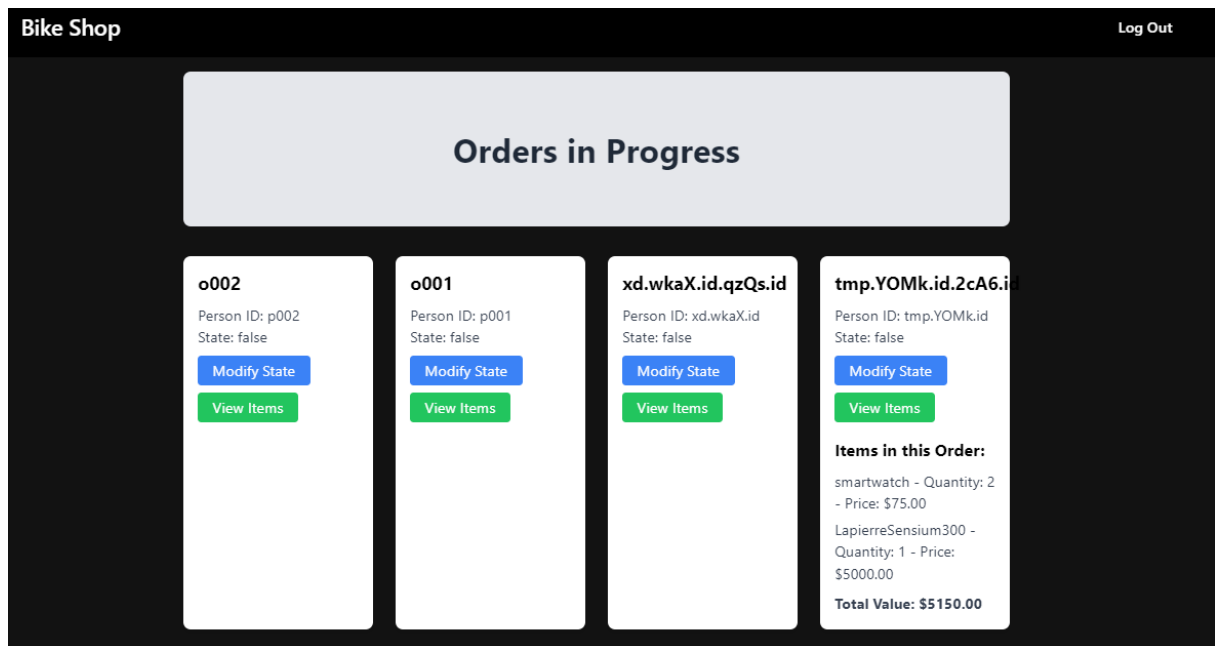
Rysunek 10 Strona startowa dla admina

Klikamy w guzik Order Modify i zostaniemy przeniesieni do okna obsługi zamówień.



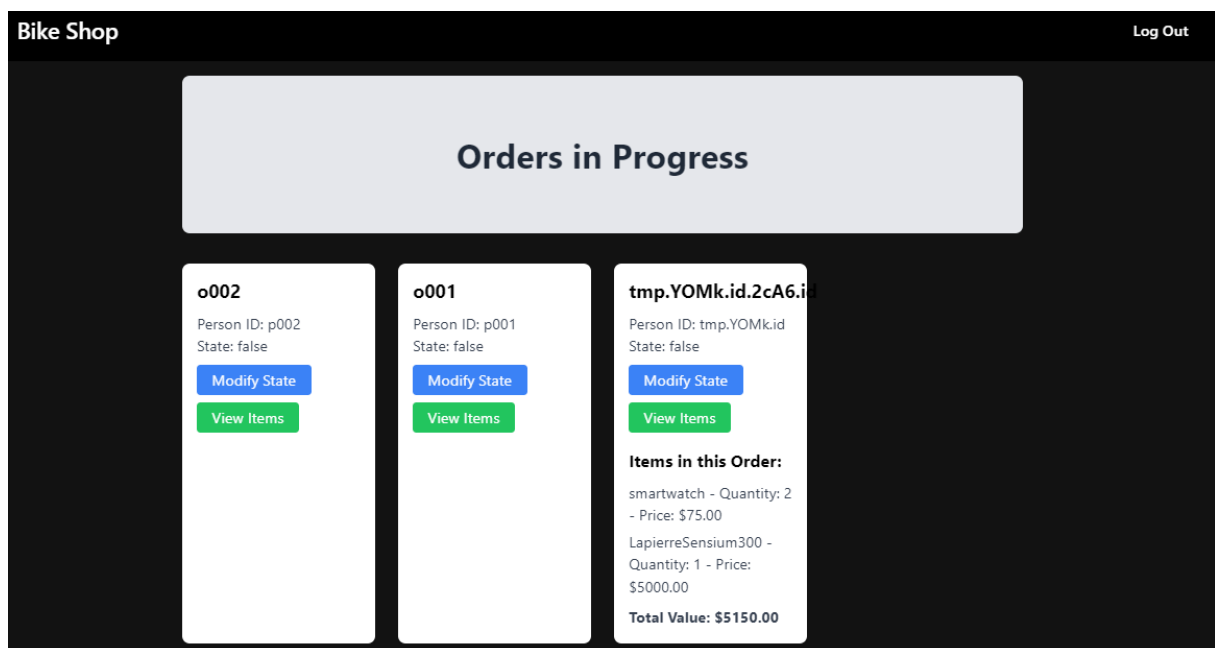
Rysunek 11 Okno obsługi zamówień

Klikając zielony guzik admin może zobaczyć jakie są przedmioty i jaka jest łączna kwota.



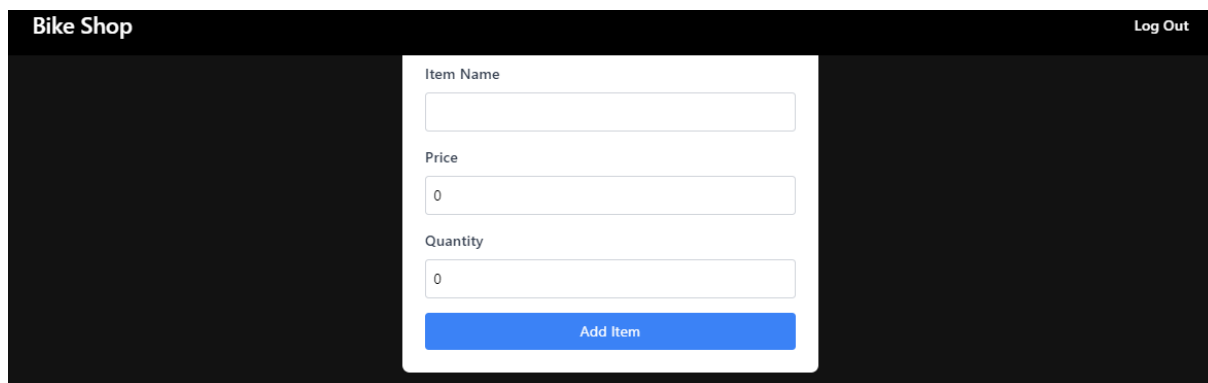
Rysunek 12 Widok po kliknięciu view items

Admin może również zmienić status zamówienia aby uznać je jako wykonane i kliknąć modify state. Dla przykładu zostanie wykonane trzecie zamówienie.



Rysunek 13 Widok po wykonaniu zamówienia

Do obsługi zostało jeszcze jedno okno dodawania przedmiotów. Należy kliknąć w oknie startowym Add new Item.



Bike Shop Log Out

Item Name

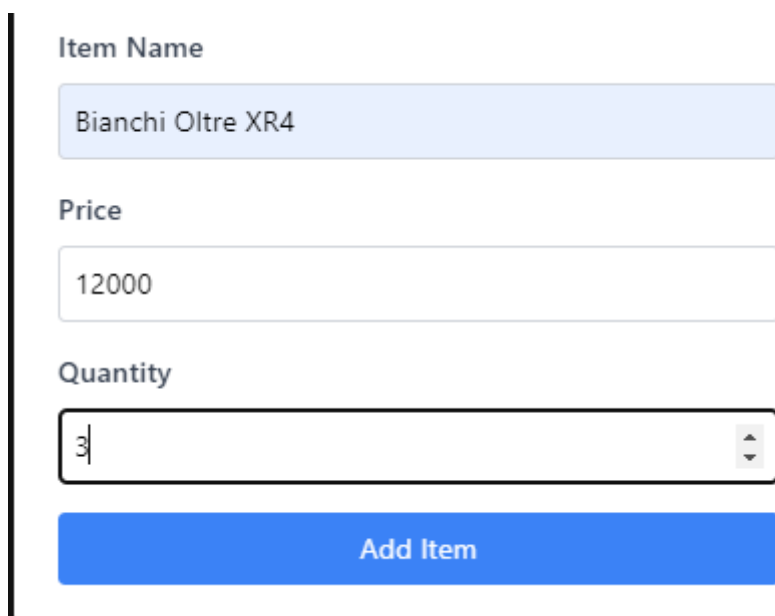
Price

Quantity

Add Item

Rysunek 14 Okno dodawania przedmiotów

Dodanie przykładowego przedmiotu:



Item Name

Price

Quantity

Add Item

Rysunek 15 Dane przykładowego przedmiotu

Po kliknięciu Add Item przedmiot powinien pojawić się w widoku przedmiotu.

Items in Database

Smartphone

Price: \$500.00
Quantity: 20

Headphones

Price: \$100.00
Quantity: 48

Laptop

Price: \$1000.00
Quantity: 14

smartwatch

Price: \$75.00
Quantity: 16

LapierreSensium300

Price: \$5000.00
Quantity: 1

Bianchi Oltre XR4

Price: \$12000.00
Quantity: 3

Existing Items

Smartphone

Price: \$500.00
Quantity: 20

Headphones

Price: \$100.00
Quantity: 48

Laptop

Price: \$1000.00
Quantity: 14

smartwatch

Price: \$75.00
Quantity: 16

LapierreSensium300

Price: \$5000.00
Quantity: 1

Bianchi Oltre XR4

Price: \$12000.00
Quantity: 3

Rysunek 16 Widok przedmiotów po dodaniu przedmiotu

8. Przyszłość

- poprawa UI
- dodanie obsługi płatności
- dodanie pracowników
- zestawienia czasowe
- export do csv

9. Bibliografia

<https://vuejs.org/guide/introduction.html>

<https://www.ibm.com/topics/java-spring-boot>

<https://tailwindcss.com/docs/installation>

<https://www.geeksforgeeks.org/mvc-framework-introduction/>