



Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych

Ekosystem dla obsługi zakupów on-line

Proseminarium

Prowadzący: **mgr inż. Marcin Ochab**

Członkowie zespołu (grupa laboratoryjna 3):

- Marcin Wielgos
- Karol Lech
- Przemysław Pleśniarski
- Krzysztof Podkulski

Rok akademicki 2020/2021, semestr zimowy

Spis treści

Spis treści	2
Opis projektu	4
Funkcjonalności	4
Struktura i uruchomienie projektu	4
Mikroserwis magazynów	6
Warstwa frontend	6
Uruchomienie nowej instancji magazynu	8
Zapytania	10
Mikroserwis produktów	11
Opis mikroserwisu produktów	11
Aplikacji mikro serwisu produktów	12
API mikro serwisu produktów	15
Dodawanie produktu	15
Edycja produktu	16
Jeden produkt	16
Usuwanie produktu	16
Lista wszystkich produktów	17
Struktura plików mikro serwisu	18
Przedstawienie poszczególnych kodów źródłowych odpowiadających za działanie mikro serwisu	19
Usuwanie produktu przy pomocy technologii AJAX	19
Przykładowy plik PUG do przedstawiania stron internetowych	19
Definiowanie ścieżek do obsługi stron WWW	20
Przykładowe fragmenty kodu kontrolerów	20
Tworzenie migracji i modelu odpowiedzialnych za komunikację z bazą danych	23
Mikroserwis sklepów	24
Opis	24
Warstwa frontend	26
Zawartość pliku docker-compose.yaml:	26
Mikroserwis autoryzacji	30
Opis	30
Prezentacja zapytania	31
Zawartość pliku docker-compose.yaml jest następująca:	31

Opis projektu

Celem projektu jest stworzenie systemu sieci magazynów ze sklepami. Będzie możliwe utworzenie dowolnej ilości instancji magazynów. Użytkownik będzie miał możliwość wyboru produktów z listy (wszystko ze wszystkich magazynów). Podczas zamawiania administrator wybiera, z którego magazynu zostanie złożone zamówienie.

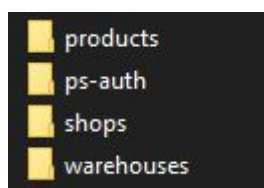
Funkcjonalności

- budowanie dowolnej ilości instancji magazynów
- przechowywanie informacji o ilościach stanów magazynowych
- możliwość dodawania do koszyka produktów pochodzących z różnych magazynów
- sumowanie kosztów za produkty
- autoryzacja użytkownika

Struktura i uruchomienie projektu

Stabilna wersja aplikacji znajduje się w branchu **main**.

Aby uruchomić każdy mikroserwis przechodzimy do każdego folderu i budujemy kontenery Dockera poprzez *docker-compose up --build*

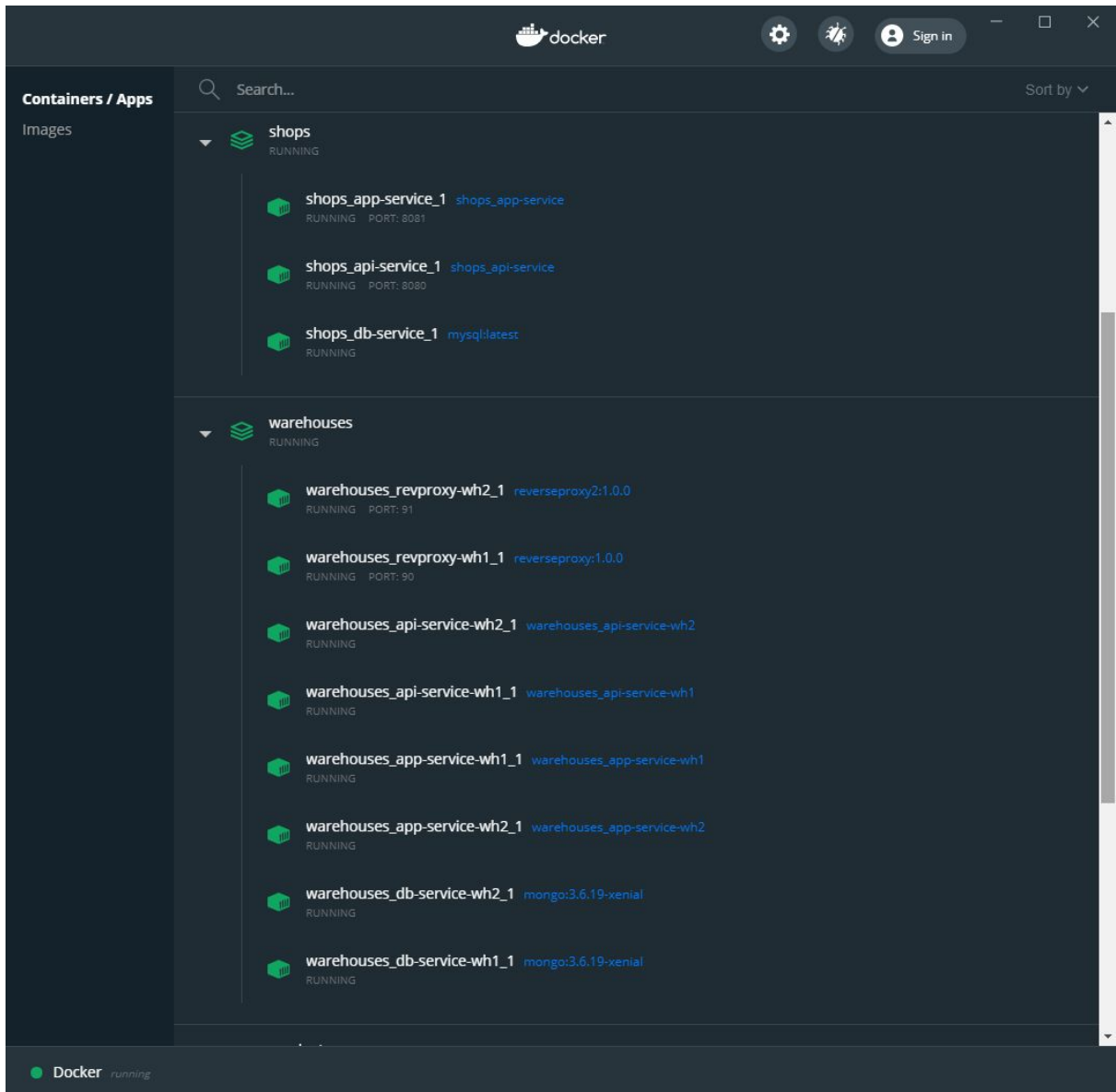


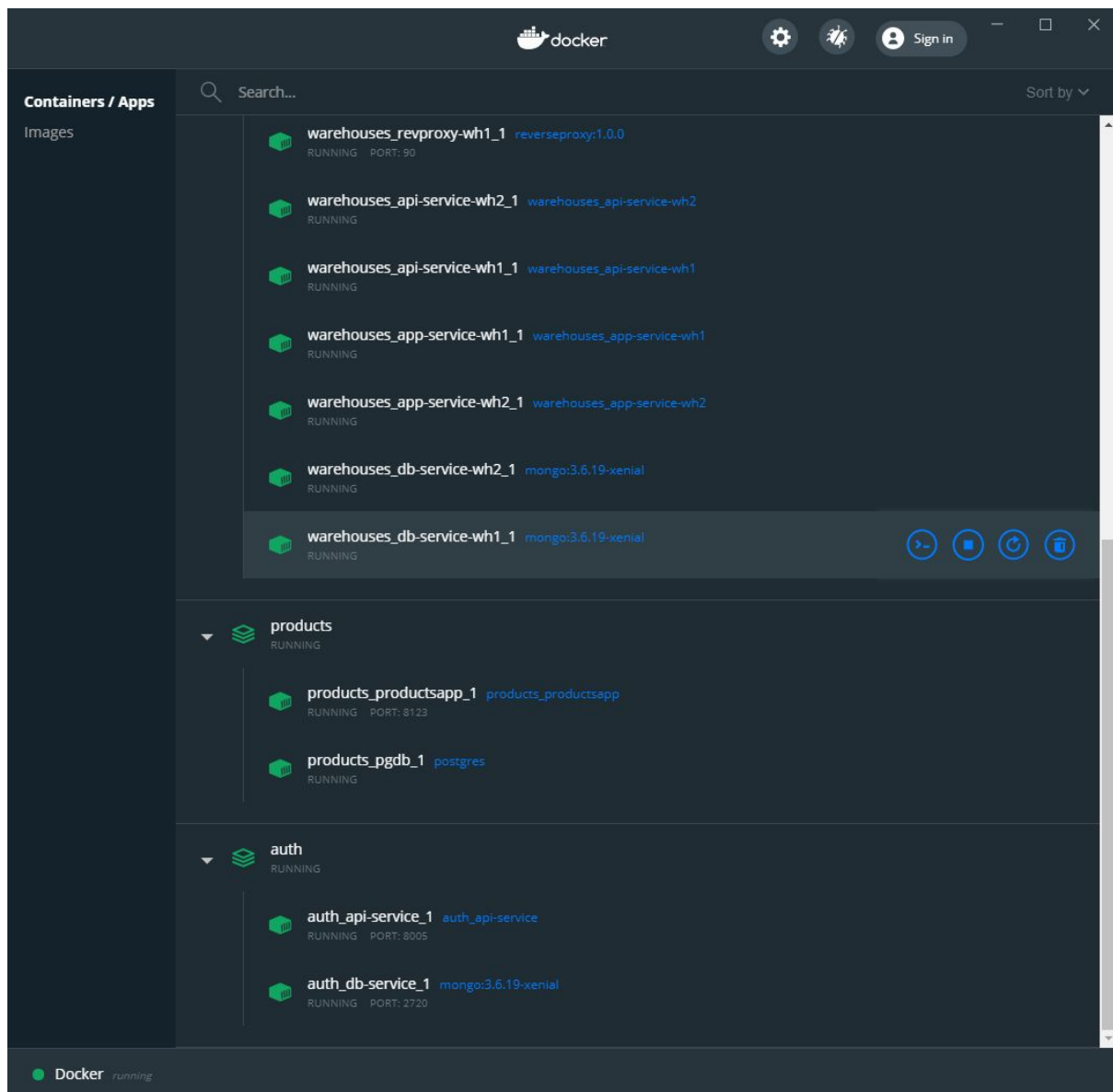
Pierwszy magazyn dostępny jest pod adresem <http://localhost:90/>
API magazynu <http://localhost:90/wh1/api>

Dla drugiego magazynu analogicznie
<http://localhost:91/>
<http://localhost:91/wh2/api>

Mikroserwis produktów pod adresem <http://localhost:90/products>

Mikroserwis sklepów <http://localhost:8081> z API na porcie 8080





Mikroserwis magazynów

Warstwa frontend: ReactJS

Warstwa backend: ExpressJS

Baza danych: MongoDB

Obsługa ruchu sieciowego: nginx z *reverse-proxy*

Warstwa frontend

Po dodaniu przykładowych produktów w mikroserwisie produktów, jesteśmy w stanie odczytać te informacje w mikroserwisie magazynów i przypisać odpowiedni status dostępności (ilość na magazynie).

Produkty

Nazwa	Opis	Cena	Wymiary (dł. x szer. x wys)	EAN	Kategoria	Akcje	<button>Dodaj</button>
Żelazko BOSCH	Opis testowy	200.99 zł	10.0x30.0x20.0 (mm)	99293847	AGD	<button>Wiecej</button>	<button>Usun</button>
Żelazko Tefal	Opis testowy	180.99 zł	10.0x34.0x20.0 (mm)	99293123	AGD	<button>Wiecej</button>	<button>Usun</button>

Magazyn 2 Poznań

Produkt:

Wybór produktu

99293847 Żelazko BOSCH

99293123 Żelazko Tefal

Ilość (szt.):

4

Zapisz

Usuń w

Aktualny stan magazynowy

GUI w mikroserwisie magazynów pozwala na zarządzanie stanem magazynowym.

Magazyn 2 Poznań

Produkt:

Wybór produktu

Ilość (szt.):

4

Zapisz

Usuń wszystko

Aktualny stan magazynowy

Produkt	Dostępna ilość	Dodaj ilość
Żelazko BOSCH	252	<div>1</div> <div><button>Dodaj</button></div>
Żelazko Tefal	190	<div>1</div> <div><button>Dodaj</button></div>

<

1

>

System przewiduje możliwość utworzenia wielu odrębnych instancji magazynów z odrębnym statusem zamówień.

W ramach testu utworzono dwie instancje (dwa magazyny).

- Magazyn 1 Rzeszów
- Magazyn 2 Poznań

localhost:90

Magazyn 1 Rzeszów

Produkt: 99293847 Żelazko BOS... Ilość (szt.): 300 Zapisz

Usuń wszystko

Aktualny stan magazynowy

Produkt	Dostępna ilość	Dodaj ilość
Żelazko BOSCH	300	<input type="text" value="1"/> Dodaj

localhost:91

Magazyn 2 Poznań

Produkt: Wybór produktu Ilość (szt.): 4 Zapisz

Usuń wszystko

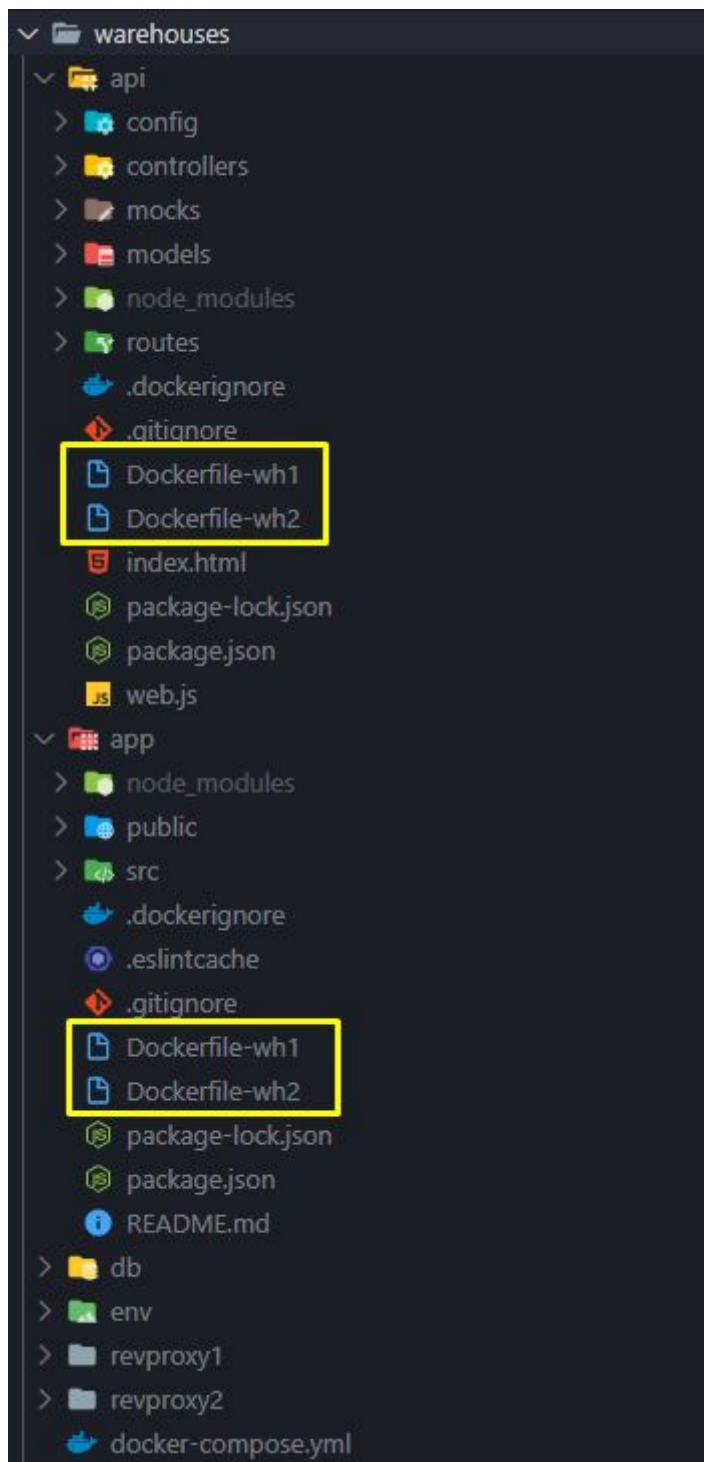
Aktualny stan magazynowy

Produkt	Dostępna ilość	Dodaj ilość
Żelazko BOSCH	252	<input type="text" value="1"/> Dodaj

W magazynie możemy przechowywać dowolne produkty, których informacje są przechowywane w mikrosерwisie produktów, jednak ich ilości magazynowe są odrębne dla każdej instancji magazynu.

Uruchomienie nowej instancji magazynu

Uruchomienie nowej instancji wiąże się z wykonaniem kilku kroków:



1. Ustawienie odpowiednich plików Dockerfile dla /api i /app pamiętając o zmiennych środowiskowych dla portu i nazwy:
ENV REACT_APP_PORT=90
ENV REACT_APP_WHKEY=wh1
ENV REACT_APP_WHNAME="Magazyn 1 Rzeszów"
2. Ustawienie odpowiedniej konfiguracji kontenera nginx dla ruchu sieciowego z odpowiednim portem (revproxy).


```
server {
    listen 90;

    location /wh1/api/ {
        proxy_pass          http://apiservice:8005/;
        proxy_redirect      off;
        proxy_set_header    Host $host;
        proxy_set_header    X-Real-IP $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Host $server_name;
    }
}
```

Oraz w pliku Dockerfile dla nginx

```
# dockerfile
FROM nginx:1.16.0-alpine
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/nginx.conf
EXPOSE 90
CMD ["nginx", "-g", "daemon off;"]
```

Zapytania

Sklep internetowy może pobrać aktualną dostępność produktów przez zapytanie GET
<http://localhost:91/wh2/api/stock/>

Przykładowy wynik:

```
{
  "stock": [
    {
      "_id": "60256e85435d12e212732365",
      "productId": "1",
      "__v": 0,
      "availableQuantity": 300,
      "createdAt": "2021-02-11T17:51:01.887Z",
      "updatedAt": "2021-02-11T17:51:01.887Z"
    },
    {
      "_id": "60256e8b435d12e21273236b",
      "productId": "2",
      "__v": 0,

```

```

        "availableQuantity": 200,
        "createdAt": "2021-02-11T17:51:07.150Z",
        "updatedAt": "2021-02-11T17:51:07.150Z"
      }
    ]
  }
}

```

Sklep może także złożyć dyspozycję zmniejszenia stanu magazynowego w przypadku zakupu poprzez zapytanie POST `http://localhost:91/wh2/api/stock/make` z przykładowym body:

```

{
  "products": [
    {
      "id": 1,
      "quantity": 50
    },
    {
      "id": 2,
      "quantity": 10
    }
  ]
}

```

Wysłanie zapytania spowoduje zmniejszenie ilości stanu magazynowego produktu o ID 1 o 50 sztuk, a produktu z ID 2 o 10 sztuk.

Mikroserwis produktów

Opis mikroserwisu produktów

Aplikacja została napisana za pomocą:

FrontEnd - engine view = pug

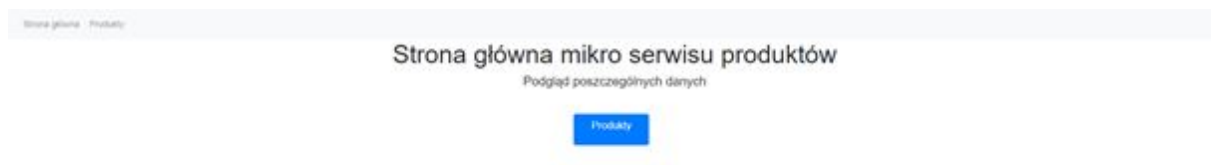
BackEnd - Node.js, express.js

Baza Danych: PostgreSQL

Aplikacja przedstawia mikro serwis produktów. Został napisany w node.js, z wykorzystaniem express.js oraz silnika PUG, który umożliwia tworzenie stron internetowych bez ingerencji zewnętrznych frameworków takich jak REACT bądź ANGULAR.

Aplikacji mikro serwisu produktów

Na głównej stronie znajduje się informacja co to za serwis oraz odnośnik do listy produktów.



Lista produktów przedstawia wszystkie najważniejsze informacje dotyczące produktów, również jest możliwość stworzenia nowego produktu za pomocą przycisku dodaj.

Produkty

Nazwa	Opis	Cena	Wymiary (dł. x szer. x wys)	EAN	Kategoria	Akcje
-------	------	------	--------------------------------	-----	-----------	-------

[Dodaj](#)

Dzięki odpowiedniemu formularzu jest możliwe dodanie produktu. Jest aktywna walidacja, gdzie produkt nie zostanie wprowadzony do listy, jeżeli nie zostały spełnione wszystkie warunki walidacji jak odpowiednio zapisana cena np. (123.00) bądź rozmiar. Wszystkie pola muszą zostać wypełnione.

Dodaj Produkt

[Potwierdz](#)[Powrót](#)

Dodaj Produkt

Potwierdz

Powrót

Po zatwierdzeniu, produkt zostanie wypisany na liście produktów. Jest możliwe usunięcie produktu za pomocą przycisku usuń, który działa w technologii AJAX, bez potrzeby przeładowania strony produkt zostanie usunięty. Przycisk więcej przenosi nas do podstrony o produkcie.

Produkt 1	Opis 1	152.01 zł	200.1x300.2x400.1 (mm)	12345678	Kategoria 1	Więcej	Usuń
-----------	--------	-----------	------------------------	----------	-------------	--------	------

Dokładny opis produktu oraz możliwość edytowania produktu oraz usunięcie produktu. Przy zmianie danych produktów również występuje walidacja, która musi zostać spełniona.

Produkt 1							
Nazwa	Opis	Cena	Wymiary (dł. x szer. x wys)	EAN	Kategoria	Akcje	
Produkt 1	Opis 1	152.01 zł	200.1x300.2x400.1 (mm)	12345678	Kategoria 1	Edytuj	Usuń

Edytuj Produkt

Potwierdz

Powrót

Produkt 1							
Nazwa	Opis	Cena	Wymiary (dl. x szer. x wys)	EAN	Kategoria	Akcje	PowerBI
Produkt 1	Opis 1	152.014 zł	200.1x300.2x400.1 (mm)	12345678	Kategoria 1	Edycja	Usun

Produkty							
Nazwa	Opis	Cena	Wymiary (dl. x szer. x wys)	EAN	Kategoria	Akcje	Dodaj
Produkt 1	Opis 1	152.014 zł	200.1x300.2x400.1 (mm)	12345678	Kategoria 1	Widok	Usun
Produkt 2	Opis 2	152.01 zł	123.1x1.1x123.4 (mm)	12345677	Kategoria 2	Widok	Usun

Produkty							
Nazwa	Opis	Cena	Wymiary (dl. x szer. x wys)	EAN	Kategoria	Akcje	Dodaj
Produkt 2	Opis 2	152.01 zł	123.1x1.1x123.4 (mm)	12345677	Kategoria 2	Widok	Usun

API mikro serwisu produktów

Do komunikacji między mikro serwisami jest wymagane api, które zostało przygotowane w pełnej konfiguracji.

Dodawanie produktu

POST

http://localhost:3000/api/produkty

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

```

{
  "nazwa": "Produkt 3",
  "opis": "Opis 18",
  "cena": 289.01,
  "wymiary": "123.5x1.1x123.4",
  "ean": 12345667,
  "kategoria": "Kategoria 3"
}

```

Body

Cookies (1)

Headers (8)

Test Results

Status: 201 Created

Time: 61 ms

Size: 578 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

```

{
  "id": 14,
  "nazwa": "Produkt 3",
  "opis": "Opis 18",
  "cena": 289.01,
  "wymiary": "123.5x1.1x123.4",
  "ean": 12345667,
  "kategoria": "Kategoria 3",
  "updatedAt": "2021-02-11T20:36:36.971Z",
  "createdAt": "2021-02-11T20:36:36.971Z"
}

```

Edycja produktu

PUT http://localhost:3000/api/produkty/14 Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "nazwa": "Produkt 6",
3   "opis": "Opis 10",
4   "cena": 200.01,
5   "wymiar": "123.5x1.1x123.4",
6   "ean": 12345667,
7   "kategoria": "Kategoria 3"
8 }
```

Body Cookies (1) Headers (8) Test Results Status: 200 OK Time: 70 ms Size: 567 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 14,
3   "nazwa": "Produkt 6",
4   "opis": "Opis 10",
5   "cena": 200.01,
6   "wymiar": "123.5x1.1x123.4",
7   "ean": 12345667,
8   "kategoria": "Kategoria 3",
9   "createdAt": "2021-02-11T20:36:36.971Z",
10  "updatedAt": "2021-02-11T20:37:46.056Z"
11 }
```

Jeden produkt

PUT http://localhost:30... GET http://localhost:30... GET http://localhost:30... GET http://localhost:30... +

No Environment

Save

GET http://localhost:3000/api/produkty/13 Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (7) Test Results Status: 200 OK Time: 11 ms Size: 436 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 13,
3   "nazwa": "Produkt 3",
4   "opis": "Opis 10",
5   "cena": 200.01,
6   "wymiar": "123.5x1.1x123.4",
7   "ean": 8,
8   "kategoria": "Kategoria 3",
9   "createdAt": "2021-02-11T20:35:41.447Z",
10  "updatedAt": "2021-02-11T20:35:41.447Z"
11 }
```

Usuwanie produktu

DELETE http://localhost:3000/api/produkty/13 Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (4) Test Results Status: 204 No Content Time: 85 ms Size: 134 B Save Response

Pretty Raw Preview Visualize Text

```
1
```

Lista wszystkich produktów

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/api/produkty`. The response status is 200 OK, with a time of 98 ms and a size of 653 B. The response body is displayed in JSON format, showing a list of two products.

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (7) Test Results

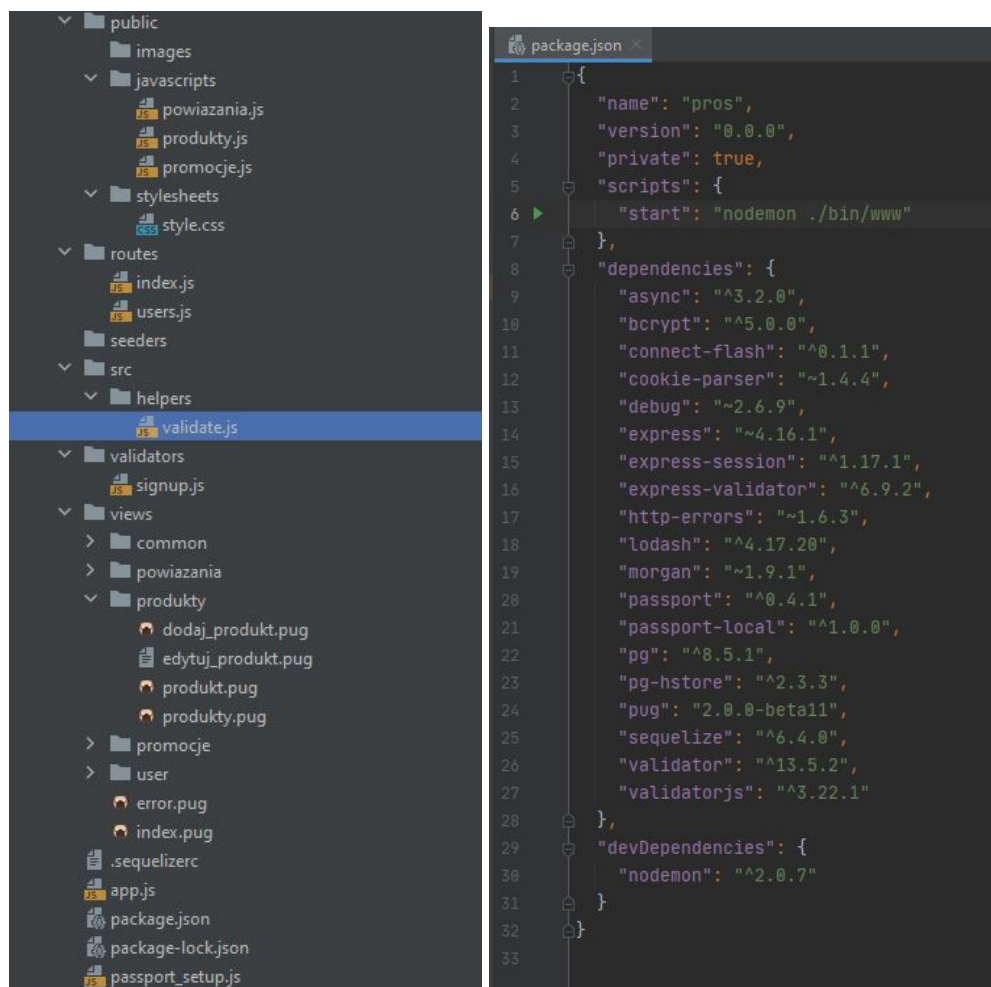
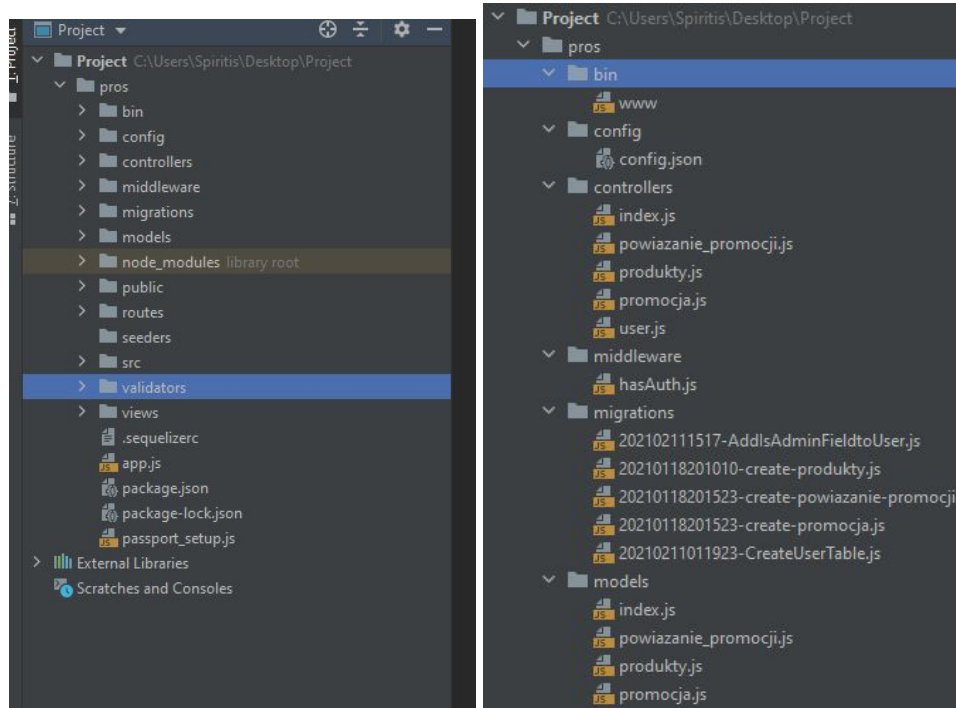
Status: 200 OK Time: 98 ms Size: 653 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 12,
4     "nazwa": "Produkt 2",
5     "opis": "Opis 2",
6     "cena": 152.01,
7     "wymiar": "123.1x1.1x123.4",
8     "ean": 12345677,
9     "kategoria": "Kategoria 2",
10    "createdAt": "2021-02-11T19:30:33.703Z",
11    "updatedAt": "2021-02-11T19:30:33.703Z"
12  },
13  {
14    "id": 14,
15    "nazwa": "Produkt 6",
16    "opis": "Opis 10",
17    "cena": 200.01,
18    "wymiar": "123.5x1.1x123.4",
19    "ean": 12345667,
20    "kategoria": "Kategoria 3",
21    "createdAt": "2021-02-11T20:36:36.971Z",
22    "updatedAt": "2021-02-11T20:37:46.056Z"
23  }
24 }
```

Bootcamp Runner Trash

Struktura plików mikro serwisu



Przedstawienie poszczególnych kodów źródłowych odpowiadających za działanie mikro serwisu

Usuwanie produktu przy pomocy technologii AJAX

```
function usunProdukt(id) {  
  $.ajax({  
    url: '/products/produkty/' + id + '/delete-json',  
    contentType: 'application/json; charset=utf-8',  
    dataType: 'json',  
    data: JSON.stringify({value: {id}}),  
    type: 'POST',  
    success: ((res) => {  
      // Replace follow button with unfollow.  
      console.log("Result: ", res)  
      $("#"+id).remove();  
    }),  
    error: ((error) => {  
      console.log("Error:", error);  
    })  
  });  
}
```

Przykładowy plik PUG do przedstawiania stron internetowych

```
doctype html  
html(lang='en')  
  include ../common/head.pug  
  +head("Dodaj Produkt")  
  body  
    header  
      include ../common/navbar.pug  
      +navbar()  
    .container  
      .row.align-items-center  
        .col-md-6.order-md-1.text-center.text-md-left.pr-md-5  
          h1 Dodaj Produkt  
          form(action="/products/produkt", method="POST")  
            .input-group.mb-3  
              input(name="nazwa", type="text", placeholder="Podaj nazwę produktu: ").form-control  
            .input-group.mb-3  
              input(name="opis", type="text", placeholder="Podaj opis produktu: ").form-control  
            .input-group.mb-3  
              input(name="cena", type="text", placeholder="Podaj cenę produktu: np(123.02)").form-control  
            .input-group.mb-3  
              input(name="wymiar", type="text", placeholder="Podaj wymiary produktu: (mm) (1.1x1.1x1.1) (dł x szer x wys)").form-control  
            .input-group.mb-3  
              input(name="ean", type="text", placeholder="Podaj EAN produktu: (8 cyfr)").form-control  
            .input-group.mb-3  
              input(name="kategoria", type="text", placeholder="Podaj kategorię produktu: ").form-control  
            .input-group.mb-3  
              button(type="submit").btn.btn-primary Potwierdź  
          a(href="/products/produkty").check3.btn.btn-primary Powrót  
      include ../common/footer.pug  
      +footer()
```

Definiowanie ścieżek do obsługi stron WWW

```
router.get( path: '/products/produkty',produktyKontroller.list2);
router.get('/products/produkt',produktyKontroller.pre_add2);
router.post('/products/produkt',produktyKontroller.add2);
router.get( path: '/products/produkt/:id', produktyKontroller.getById2);
router.get( path: '/products/produkt/:id/edit', produktyKontroller.pre_update2);
router.post('/products/produkt/:id/edit',produktyKontroller.update2);
router.post( path: '/products/produkt/:id/delete', produktyKontroller.delete2);
router.post( path: '/products/produkty/:id/delete-json', produktyKontroller.usun_produkt_json);
```

Przykładowe fragmenty kodu kontrolerów

```
list(req, res) {
  return Produkty
    .findAll( options: {
    }) Promise<Model[]>
    .then((produkties : Model[] ) => res.status(200).send(produkties)) Promise<Model[]>
    .catch((error) => { res.status(400).send(error); });
},
getById(req, res) {
  return Produkty
    .findByPk(req.params.id, options: {
    }) Promise<Model>
    .then((produkties : Model ) => {
      if (!produkties) {
        return res.status(404).send({
          message: 'Produkt Not Found',
        });
      }
      return res.status(200).send(produkties);
    }) Promise<Model>
    .catch((error) => {
      console.log(error);
      res.status(400).send(error);
    });
},
```

```

update(req, res) {
  const validationRule={
    "nazwa": "required|string",
    "opis": "required|string",
    "cena": "required:regex:/^[0-9]{1,10}[.]{1}[0-9]{1,2}$/",
    "wymiar": "required:regex:/^[0-9]{1,10}[.]{1}[0-9]{1,2}[x,X]{1}[0-9]{1,10}[.]{1}[0-9]{1,2}[x,X]{1}[0-9]{1,10}[.]{1}[0-9]{1,2}$/",
    "ean": "required|min:8",
    "kategoria": "required"
  }
  validator(req.body, validationRule, {customMessages: {}}, {callback: (err, status) => {
    if (!status) {
      res.status(412)
      .send({
        success: false,
        message: 'Validation failed',
        data: err
      });
    } else {
      return Produkt
      .findById(req.params.id, {options: {}}, Promise<Model>)
      .then(produkty => {
        if (!produkty) {
          return res.status(404).send({
            message: 'coo? Not Found',
          });
        }
        return produkty
          .update({
            nazwa: req.body.nazwa || produkty.nazwa,
            opis: req.body.opis || produkty.opis,
            cena: req.body.cena || produkty.cena,
            wymiar: req.body.wymiar || produkty.wymiar,
            ean: req.body.ean || produkty.ean,
            kategoria: req.body.kategoria || produkty.kategoria,
          })
          .then(() => res.status(200).send(produkty))
          .catch((error) => res.status(400).send(error));
      }) Promise<Model>
      .catch((error) => res.status(400).send(error));
    }
  });
},

```

```

add2(req, res) {
  const validationRule={
    "nazwa": "required|string",
    "opis": "required|string",
    "cena": "required:regex:/^[0-9]{1,10}[.]{1}[0-9]{1,2}$/",
    "wymiar": "required:regex:/^[0-9]{1,10}[.]{1}[0-9]{1,2}[x,X]{1}[0-9]{1,10}[.]{1}[0-9]{1,2}[x,X]{1}[0-9]{1,10}[.]{1}[0-9]{1,2}$/",
    "ean": "required|size:8",
    "kategoria": "required",
  }
  validator(req.body, validationRule, {customMessages: {}}, {callback: (err, status) => {
    if (!status) {
      res.render('produkty/dodaj-produkt');
      //res.render('produkty/dodaj-produkt');
    } else {
      return Produkt
      .create({ values: {
        nazwa: req.body.nazwa,
        opis: req.body.opis,
        cena: req.body.cena,
        wymiar: req.body.wymiar,
        ean: req.body.ean,
        kategoria: req.body.kategoria,
      }} Promise<Model>)
      // .then((produkty) => res.redirect('./produkty'))
      .then((produkty : Model) => res.redirect('/products/produkty')) Promise<Model>
      .catch((error) => res.status(400).send(error));
    }
  });
}

```

```

delete2(req, res) {
  return Produkty
    .findByPk(req.params.id) Promise<Model>
    .then(produkties => {
      if (!produkties) {
        return res.status(400).send({
          message: 'Product Not Found',
        });
      }
      return produkties
        .destroy() Promise<void>
        // .then(() => res.redirect('./produkty'))
        .then(() => res.redirect('/products/produkty')) Promise<void>
        // .catch((error) => res.redirect('./produkt/' + req.params.id));
        .catch((error) => res.redirect('/products/produkt/' + req.params.id));
    }) Promise<Model>
    // .catch((error) => res.redirect('./produkt/' + req.params.id));
    .catch((error) => res.redirect('/products/produkt/' + req.params.id));
},

usun_produkt_json(req, res, next) {
  return Produkty.destroy( options: {
    where: {
      id: req.params.id
    }
  }).then(result => {
    res.send({ msg: "Success" });
  })
},

```

Tworzenie migracji i modelu odpowiedzialnych za komunikację z bazą danych

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable( tableName: 'Produkties', attributes: {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nazwa: {
        type: Sequelize.STRING
      },
      opis: {
        type: Sequelize.STRING
      },
      cena: {
        type: Sequelize.FLOAT
      },
      wymiary: {
        type: Sequelize.STRING
      },
      ean: {
        type: Sequelize.INTEGER(8)
      },
      kategoria: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable( tableName: 'Produkties');
  }
};
```



```

'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Produkty extends Model {
    static associate(models) {
      Produkty.hasMany(models.Powiazanie_promocji, { options: {
        foreignKey: 'id_produktu',
        as: 'produkty-powiazania',
      }});
    }
  }
  Produkty.init({
    nazwa: DataTypes.STRING,
    opis: DataTypes.STRING,
    cena: DataTypes.FLOAT,
    wymiary: DataTypes.STRING,
    ean: DataTypes.INTEGER,
    kategoria: DataTypes.STRING
  }, {
    options: {
      sequelize,
      modelName: 'Produkty',
    }
  });
  return Produkty;
};

```

Mikroserwis sklepów

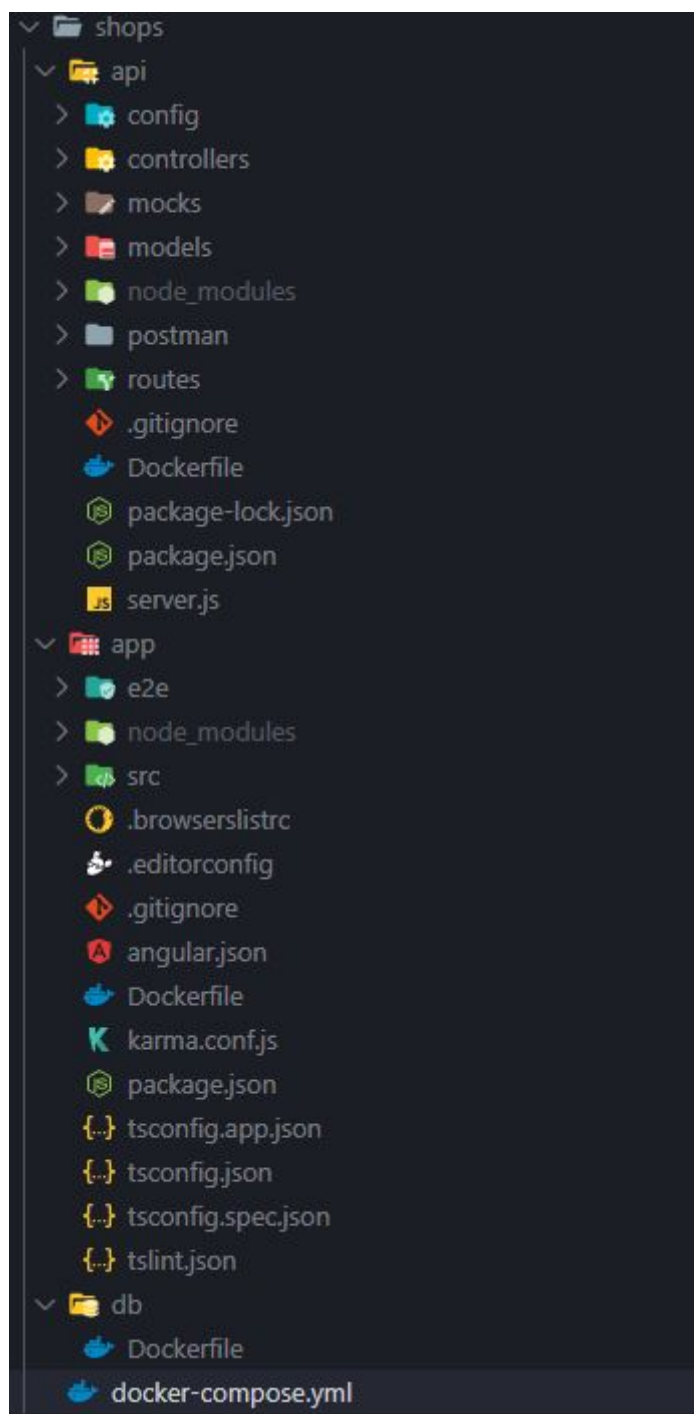
Warstwa frontend: AngularJS

Warstwa backend: ExpressJS

Baza danych: MySQL

Opis

Mikroserwis pobiera dane o produktach i wyświetla je po stronie sklepów. Poprzez GUI możemy dodać produkty do koszyka i złożyć zamówienie.



Mikroserwis sklepów korzysta ze swojej osobnej kolekcji kontenerów z osobną bazą danych.

Warstwa frontend

[Zarządzanie zamówieniami](#) [Produkty](#) [Historia zamówień](#) [Koszyk \(PLN652.84\)](#)

Produkty

wewqe
PLN12.23
asdasdasd
12334567

[Do koszyka](#)

test1
PLN123.23
opis
54362314

[Do koszyka](#)

sadasda123
PLN223.23
sadasd123
12321332

[Do koszyka](#)

[«](#) [1](#) [»](#)

© Przemysław Fleśniński, Karol Lech, Marcin Wielgos, Krzysztof Podkulski

[Zarządzanie zamówieniami](#) [Produkty](#) [Historia zamówień](#) [Koszyk \(PLN652.84\)](#)

Koszyk

Nazwa	Cena	Ilość	Suma
wewqe	PLN12.23	3	PLN36.69
test1	PLN123.23	5	PLN616.15

[Złóż zamówienie](#)

© Przemysław Fleśniński, Karol Lech, Marcin Wielgos, Krzysztof Podkulski

Po dodaniu produktów do koszyka obliczana jest ich wartość zależnie od posiadanej w koszyku ilości.

Zawartość pliku docker-compose.yaml:

```
version:
"3"
```

```
services:

  db-service:

    image: mysql:latest

    environment:

      MYSQL_ROOT_PASSWORD:
password

      MYSQL_DATABASE: shop

    # ports:

    #   - "3306:3306"

  networks:

    - sh1-network

  volumes:

    - db-data:/data/db

  api-service:
```

```
    build: "./api"

    volumes:

        - ./api:/usr/src/api

        - /usr/src/api/node_modules

    ports:

        - "8080:8080"

    networks:

        - sh1-network

        - wspolna

    depends_on:

        - db-service

app-service:

    build: "./app"

    volumes:
```

```

- ./app:/usr/src/app

-
/usr/src/app/node_modules

ports:

- "8081:8081"

networks:

sh1-network:

wspolna:

                                ipv4_address:
"10.1.0.200"

networks:

sh1-network:

driver: bridge

wspolna:

name: wspolna

ipam:

```

```
    driver: default

    config:

      - subnet: 10.1.0.0/24

  volumes:

    db-data:

      driver: local
```

Mikroserwis autoryzacji

API: ExpressJS

Baza danych: MongoDB

Opis

Mikroserwis autoryzacji pozwala użytkownikom na rejestrację, logowanie i zmianę swoich danych. Na tej podstawie można złożyć zamówienie i przypisać je do konkretnego użytkownika.

Logowanie

Email *

Hasło *

Zaloguj się

Zmiana danych

Użytkownik

Email *

Hasło *

Powtórz hasło *

Imię *

Nazwisko *

Telefon *

Adres *

Zapisz

Mikroserwis posiada swoją odrębną kolekcję kontenerów z odrębną bazą danych.

Prezentacja zapytań

TYPE

OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to

Request Headers

Configure New Token

Token Name

user

Grant Type

Authorization Code

Callback URL

http://localhost:8005

Auth URL

http://localhost:8005/oauth/authenticate

Access Token URL

http://localhost:8005/oauth/access_token

Client ID

kj34n5jk43nkj34nkj34nkj34n

Client Secret

a5828e9d6052d-c3b14a93e07a5932dd9

Scope

user

State

fwef

Client Authentication

Send as Basic Auth header

Get New Access Token

Mikroserwis autoryzacji

File Edit View Help

Mikroserwis autoryzacji Rejestracja Zmiana danych

Logowanie

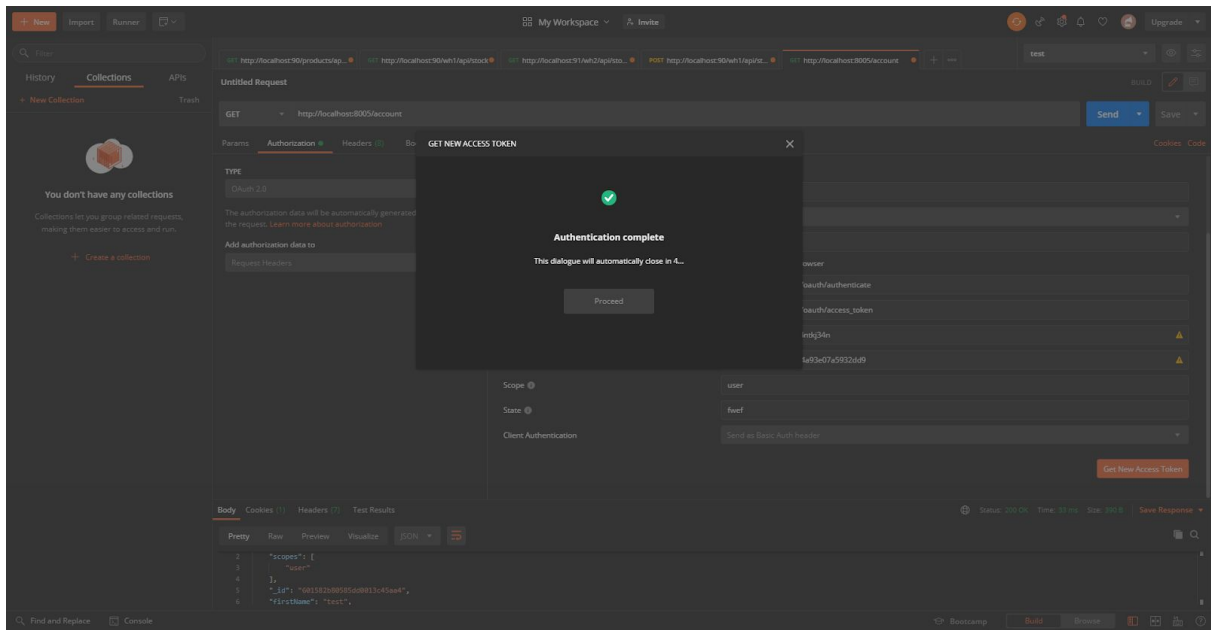
Email *

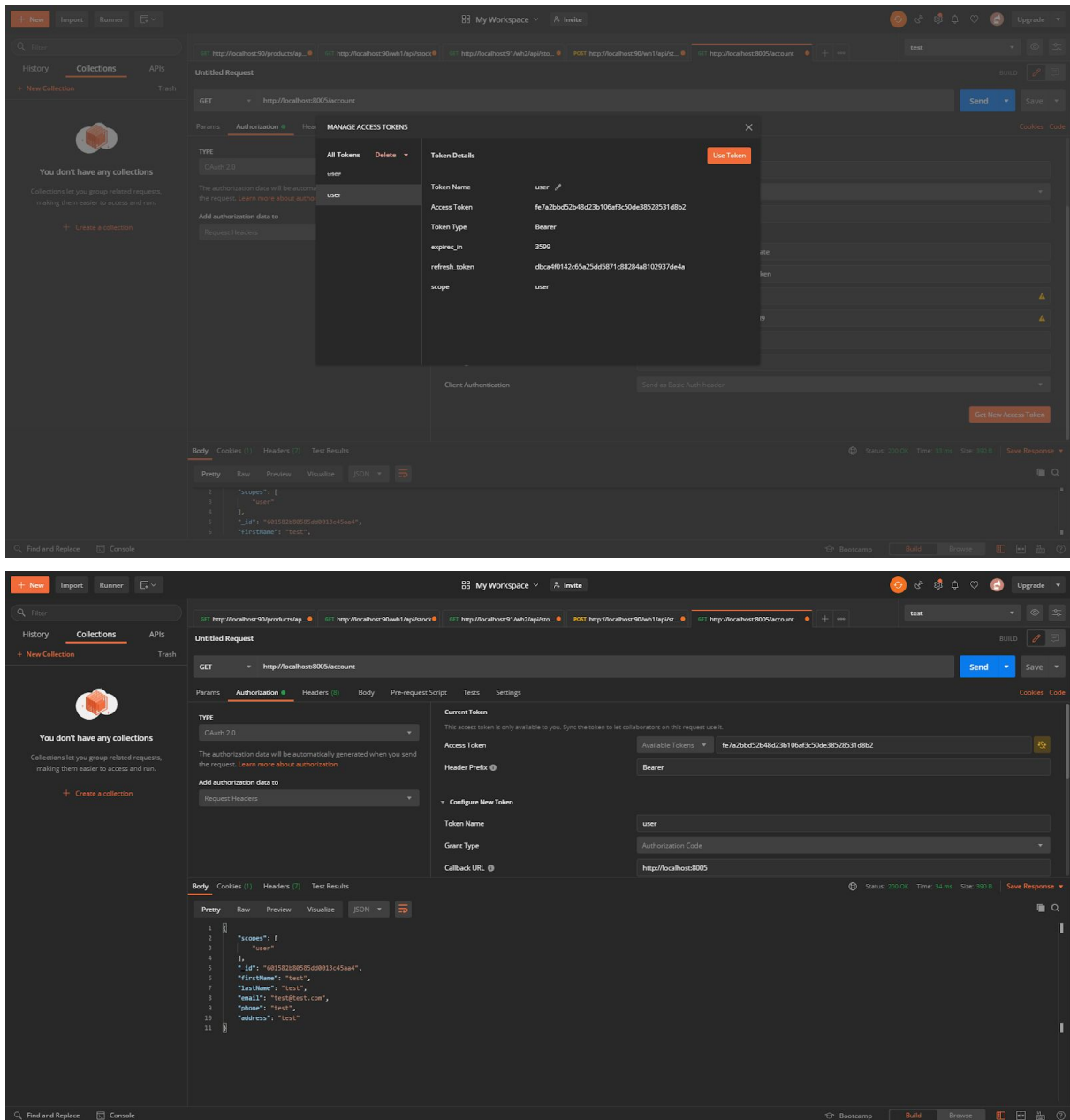
test@test.com

Hasło *

....

Zaloguj się





Zawartość pliku docker-compose.yaml:

```

version:
  "3"

services:

  api-service:

    build: "./api"

    volumes:

```

```

- ./api:/usr/src/api
-
/usr/src/api/node_modules
ports:
- "8005:8005"
networks:
  auth-network:
  wspolna:
    ipv4_address:
"10.1.0.212"
  depends_on:
    - db-service

db-service:
  image: mongo:3.6.19-xenial
  ports:
    - "2720:27017"
  networks:
    - auth-network
  volumes:
    - db-data:/data/db

networks:
  auth-network:
    driver: bridge
  wspolna:
    name: wspolna
    ipam:
      driver: default
      config:
        - subnet: 10.1.0.0/24
  volumes:
    db-data:
      driver: local

```