DOKUMENTACJA PROJEKTU ALLEDRO

1. Uruchamianie aplikacji

Aby uruchomić aplikację:

- cd app
- npm install
- ng serve

Następnie:

- cd MongoClient
- npm install
- node server.js

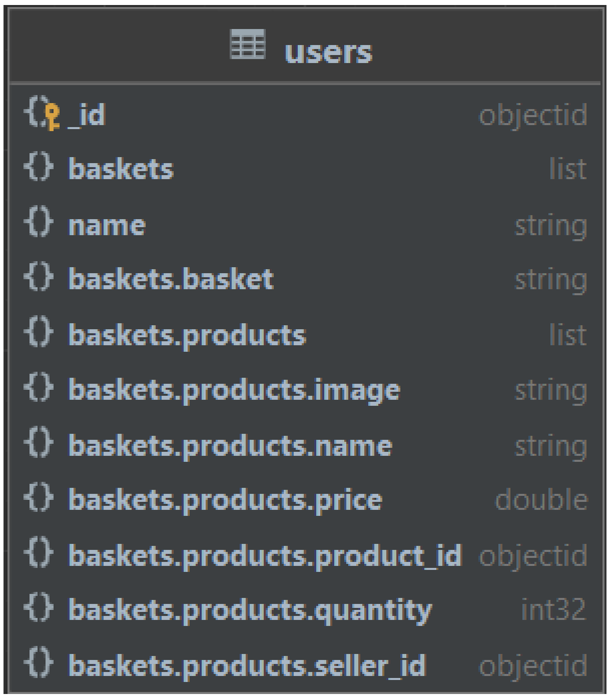
2. Baza danych:

Aplikacja korzysta z bazy noSQL, MongoDB.

Baza danych posiada 3 kolekcje:

- Users:

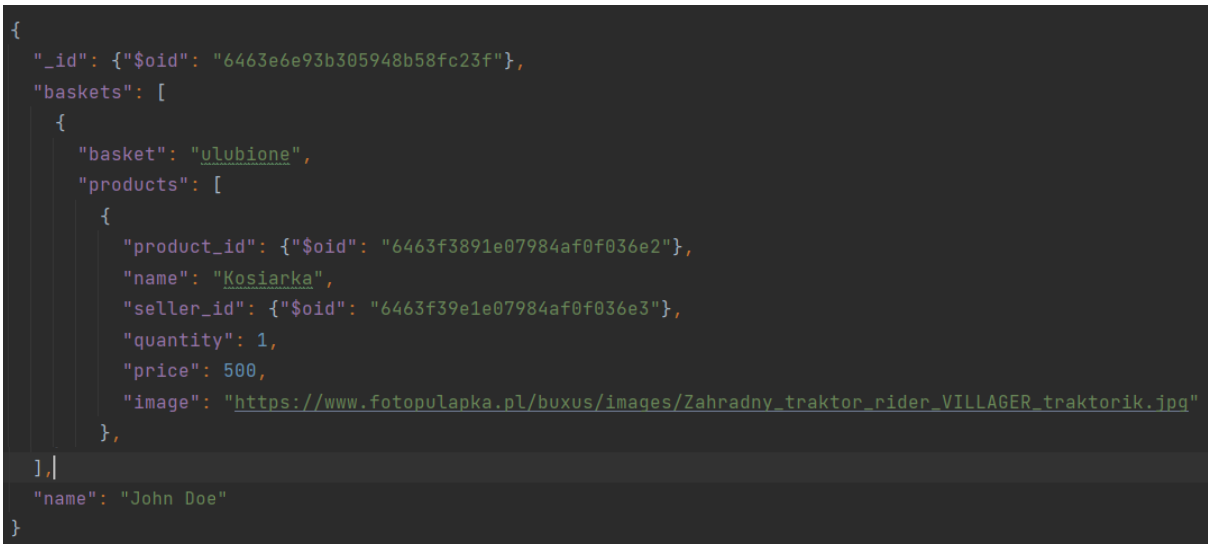
```
{
 name: string;
 baskets: Basket{
   basket: string;
   products: Product{
     _id: ProductID;
     category: string;
     description: string;
     image: Url;
     name: string;
     price: number;
     quantity: number;
     seller_id: string;
   };
 };
};
```



users	
_id	objectid
baskets	list
name	string
baskets.basket	string
baskets.products	list
baskets.products.image	string
baskets.products.name	string
baskets.products.price	double
baskets.products.product_id	objectid
baskets.products.quantity	int32
baskets.products.seller_id	objectid

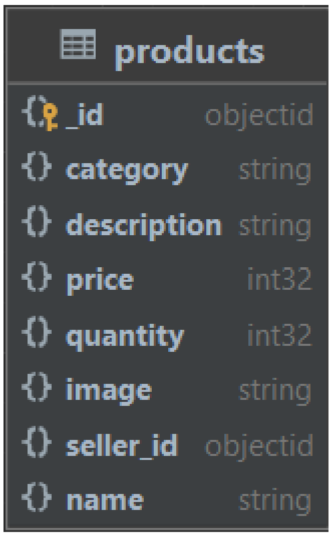
Służy do przechowywania informacji o użytkowniku oraz jego koszykach.

Przykładowe dane:



```
{
  "_id": {"$oid": "6463e6e93b305948b58fc23f"},
  "baskets": [
    {
      "basket": "ulubione",
      "products": [
        {
          "product_id": {"$oid": "6463f3891e07984af0f036e2"},
          "name": "Kosiarka",
          "seller_id": {"$oid": "6463f39e1e07984af0f036e3"},
          "quantity": 1,
          "price": 500,
          "image": "https://www.fotopulapka.pl/buxus/images/Zahradny_traktor_rider_VILLAGER_traktorik.jpg"
        },
  ],
  "name": "John Doe"
}
```

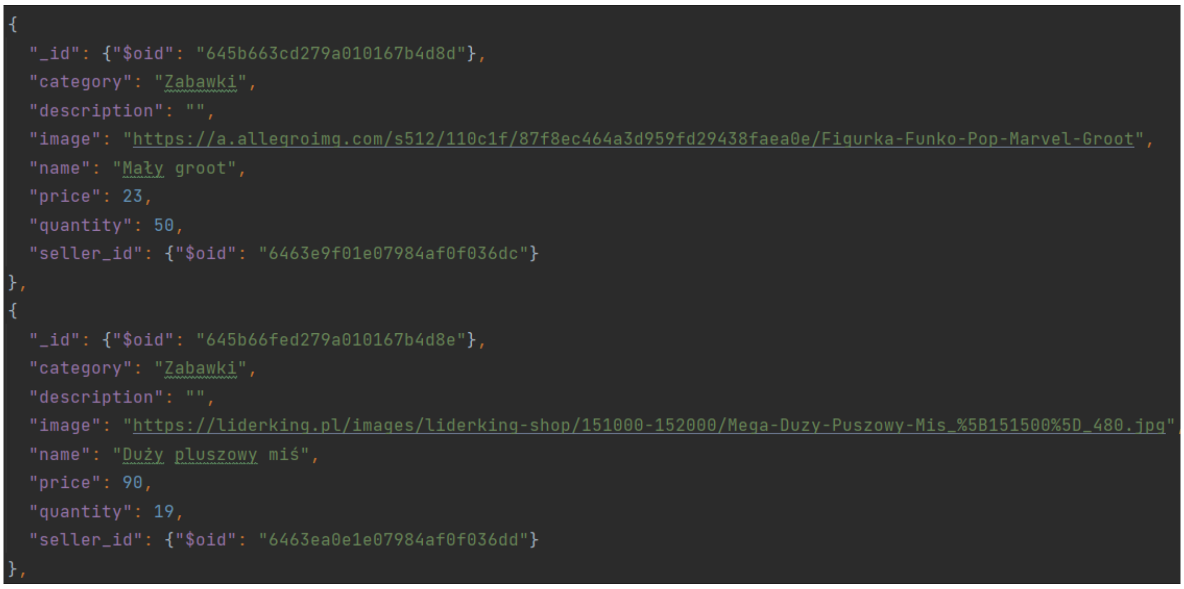
- Products:
  {
  category: string;
  description: string;
  image: Url;
  name: string;
  price: number;
  quantity: number;
  seller_id: string;
  }



products	
_id	objectid
category	string
description	string
price	int32
quantity	int32
image	string
seller_id	objectid
name	string

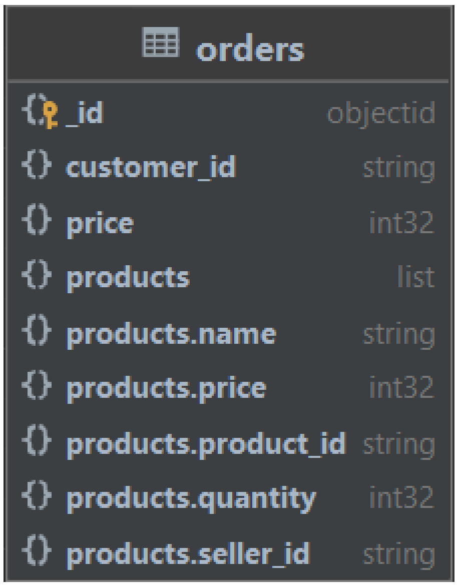
Przechowuje informacje o produktach.

Przykładowe dane:



```
{
  "_id": {"$oid": "645b663cd279a010167b4d8d"},
  "category": "Zabawki",
  "description": "",
  "image": "https://a.allegroimg.com/s512/110c1f/87f8ec464a3d959fd29438faea0e/Figurka-Funko-Pop-Marvel-Groot",
  "name": "Mały groot",
  "price": 23,
  "quantity": 50,
  "seller_id": {"$oid": "6463e9f01e07984af0f036dc"}
},
{
  "_id": {"$oid": "645b66fed279a010167b4d8e"},
  "category": "Zabawki",
  "description": "",
  "image": "https://liderking.pl/images/liderking-shop/151000-152000/Mega-Duzy-Puszowy-Mis_%5B151500%5D_480.jpg"
  "name": "Duży pluszowy miś",
  "price": 90,
  "quantity": 19,
  "seller_id": {"$oid": "6463ea0e1e07984af0f036dd"}
},
```

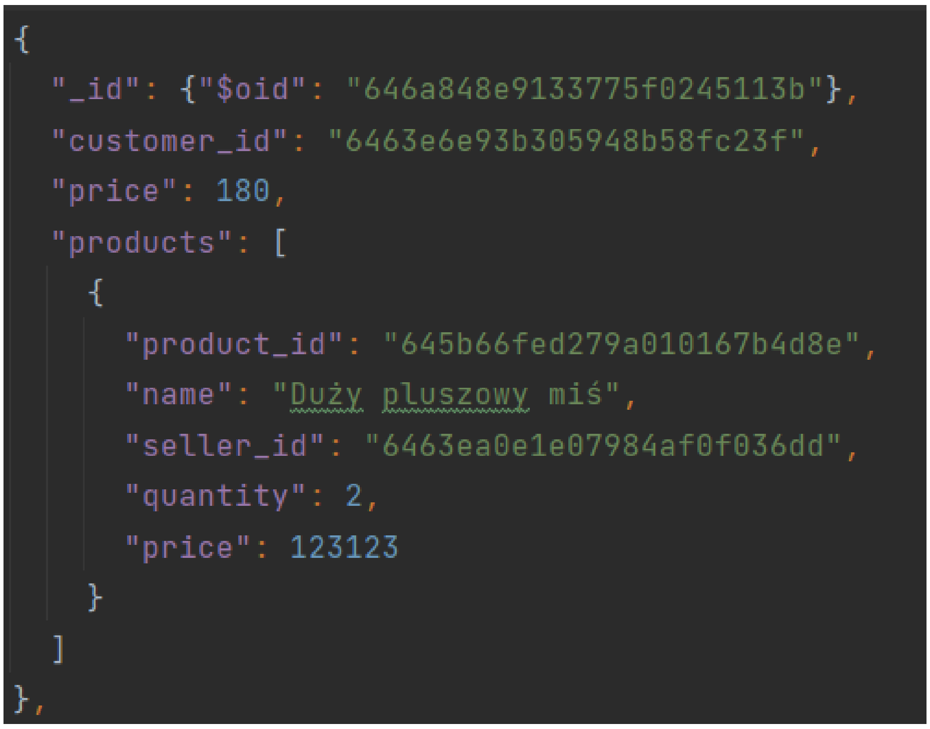
- Orders:
```
{
{
   customer_id: string,
   price: number,
   products: [
     {
       product_id: string,
       name: string,
       seller_id: string,
       quantity: number,
       price: number
     }
   ]
 },
}
```



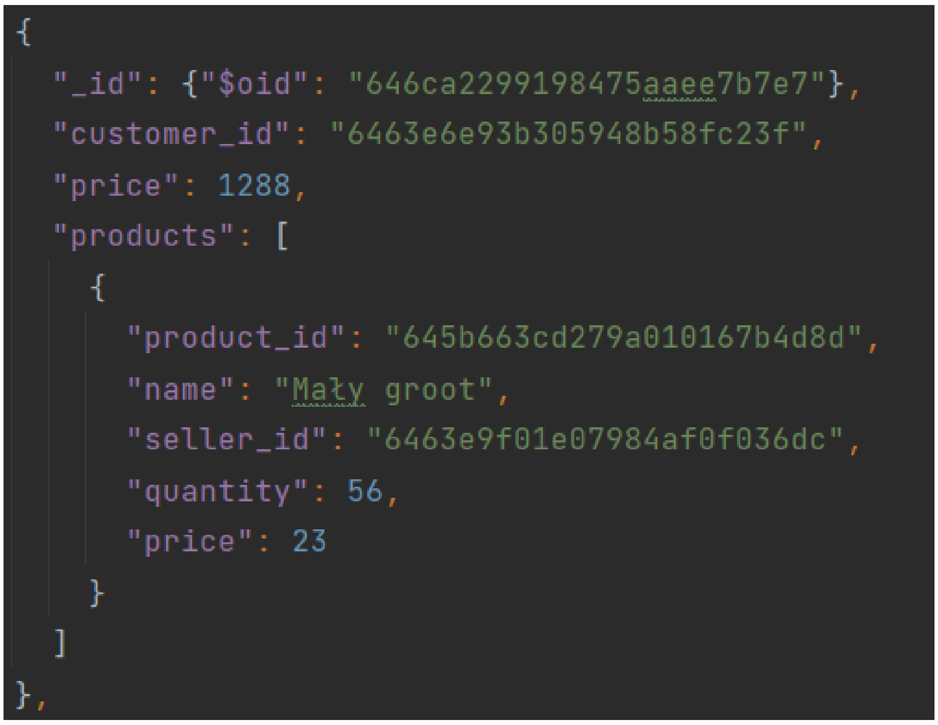
orders	
_id	objectid
customer_id	string
price	int32
products	list
products.name	string
products.price	int32
products.product_id	string
products.quantity	int32
products.seller_id	string

Przechowuje informacje o zamówieniach

Przykładowe dane:



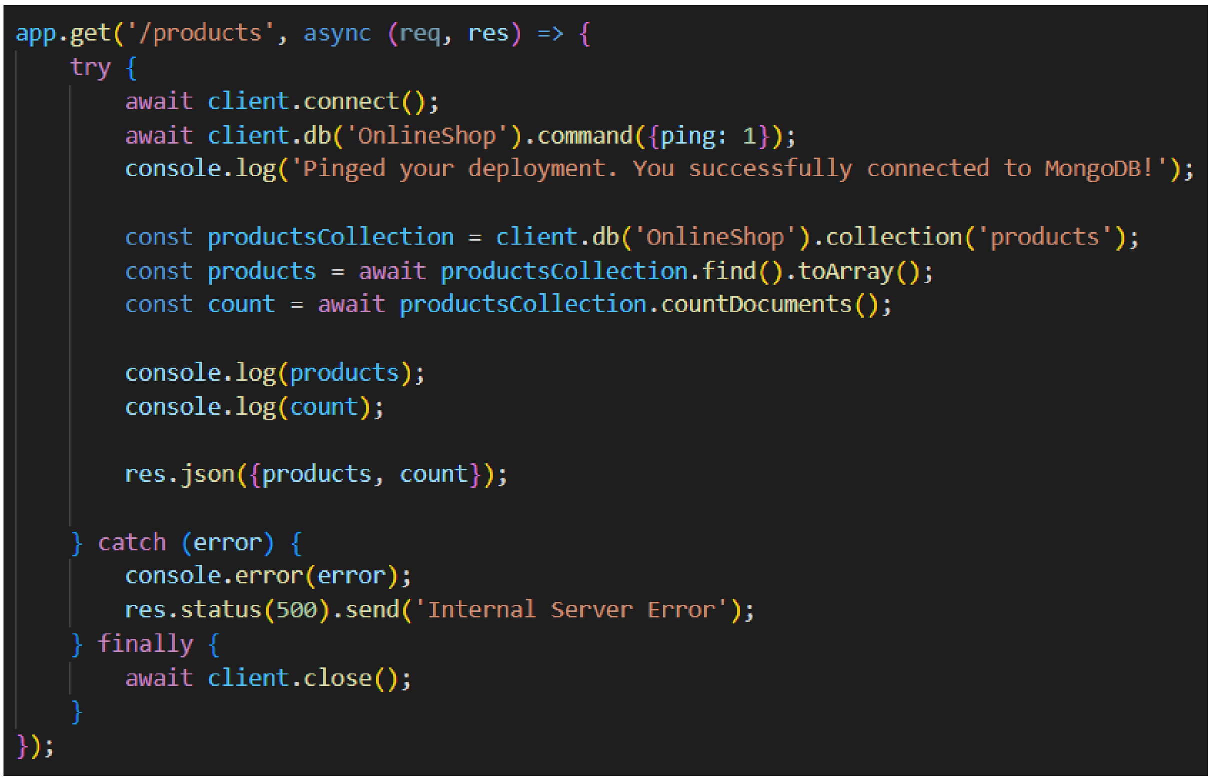
```
{
  "_id": {"$oid": "646a848e9133775f0245113b"},
  "customer_id": "6463e6e93b305948b58fc23f",
  "price": 180,
  "products": [
    {
      "product_id": "645b66fed279a010167b4d8e",
      "name": "Duży pluszowy miś",
      "seller_id": "6463ea0e1e07984af0f036dd",
      "quantity": 2,
      "price": 123123
    }
  ]
},
```



```
{
  "_id": {"$oid": "646ca2299198475aaee7b7e7"},
  "customer_id": "6463e6e93b305948b58fc23f",
  "price": 1288,
  "products": [
    {
      "product_id": "645b663cd279a010167b4d8d",
      "name": "Mały groot",
      "seller_id": "6463e9f01e07984af0f036dc",
      "quantity": 56,
      "price": 23
    }
  ]
},
```

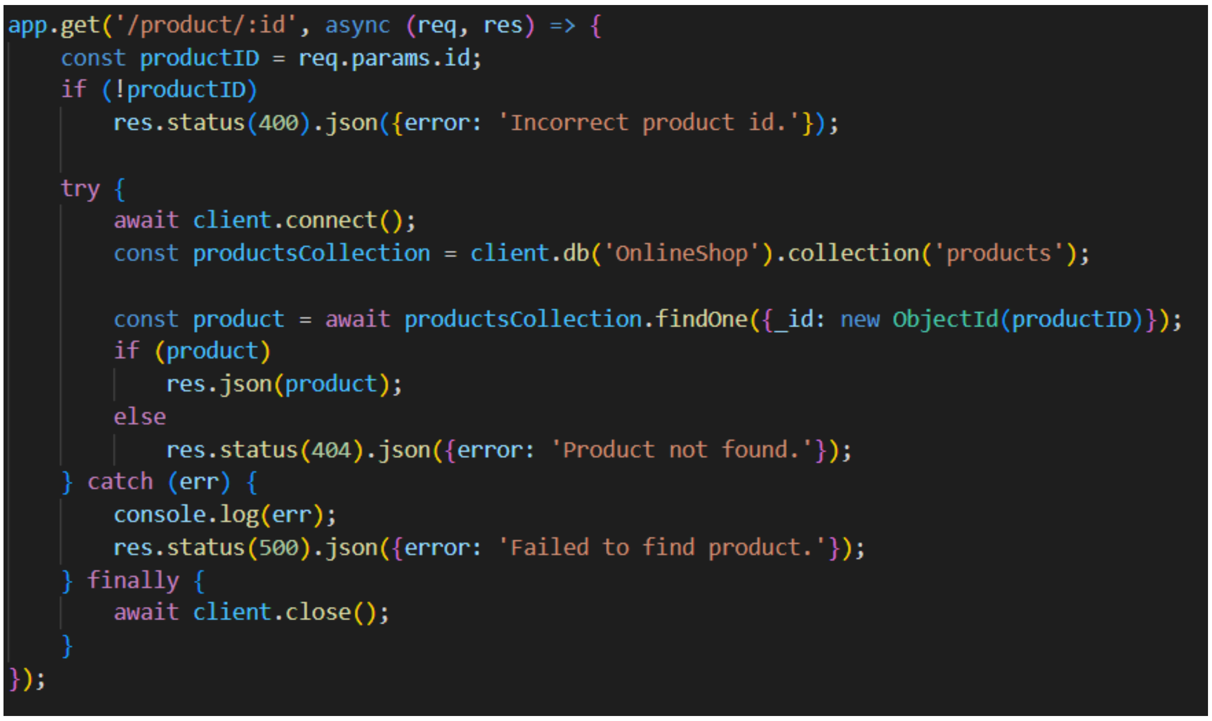
3. Zapytania do bazy:

- Pozyskanie kolekcji Products:



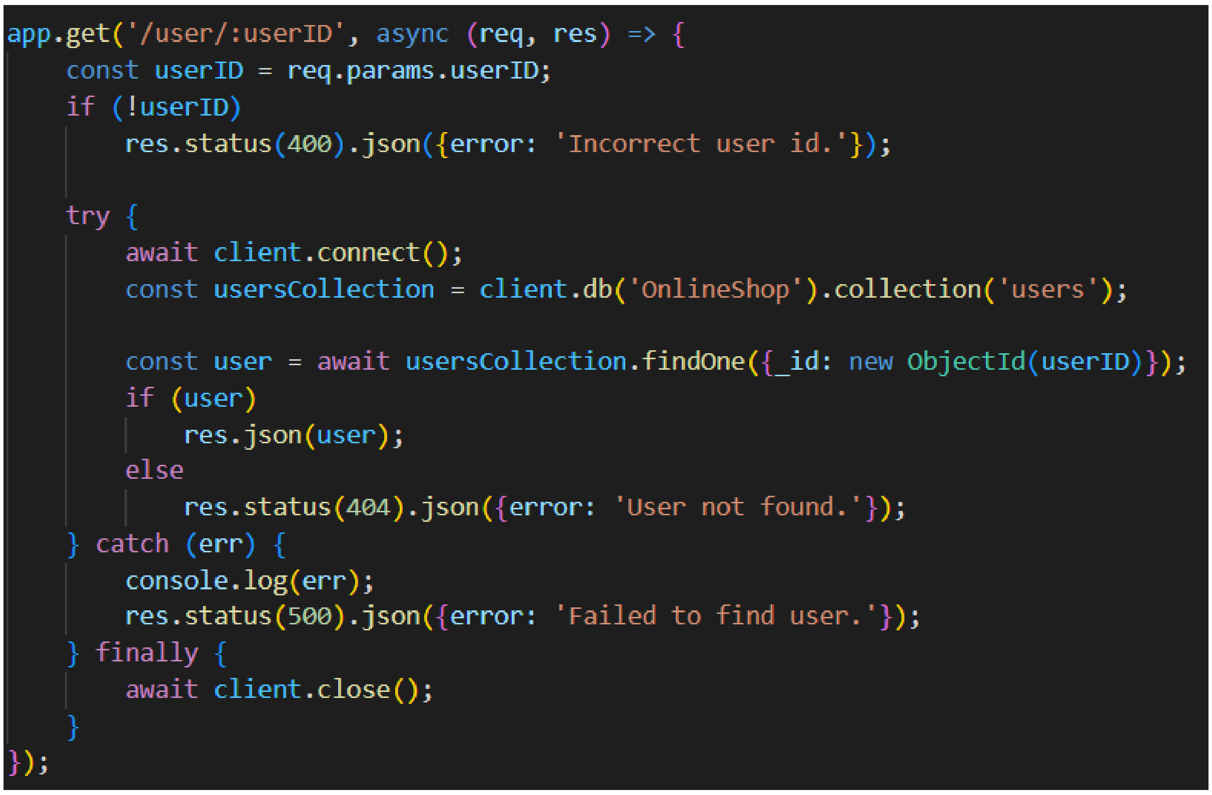
```javascript
app.get('/products', async (req, res) => {
    try {
        await client.connect();
        await client.db('OnlineShop').command({ping: 1});
        console.log('Pinged your deployment. You successfully connected to MongoDB!');

        const productsCollection = client.db('OnlineShop').collection('products');
        const products = await productsCollection.find().toArray();
        const count = await productsCollection.countDocuments();

        console.log(products);
        console.log(count);

        res.json({products, count});

    } catch (error) {
        console.error(error);
        res.status(500).send('Internal Server Error');
    } finally {
        await client.close();
    }
});
```

- Pozyskanie konkretnego produktu:



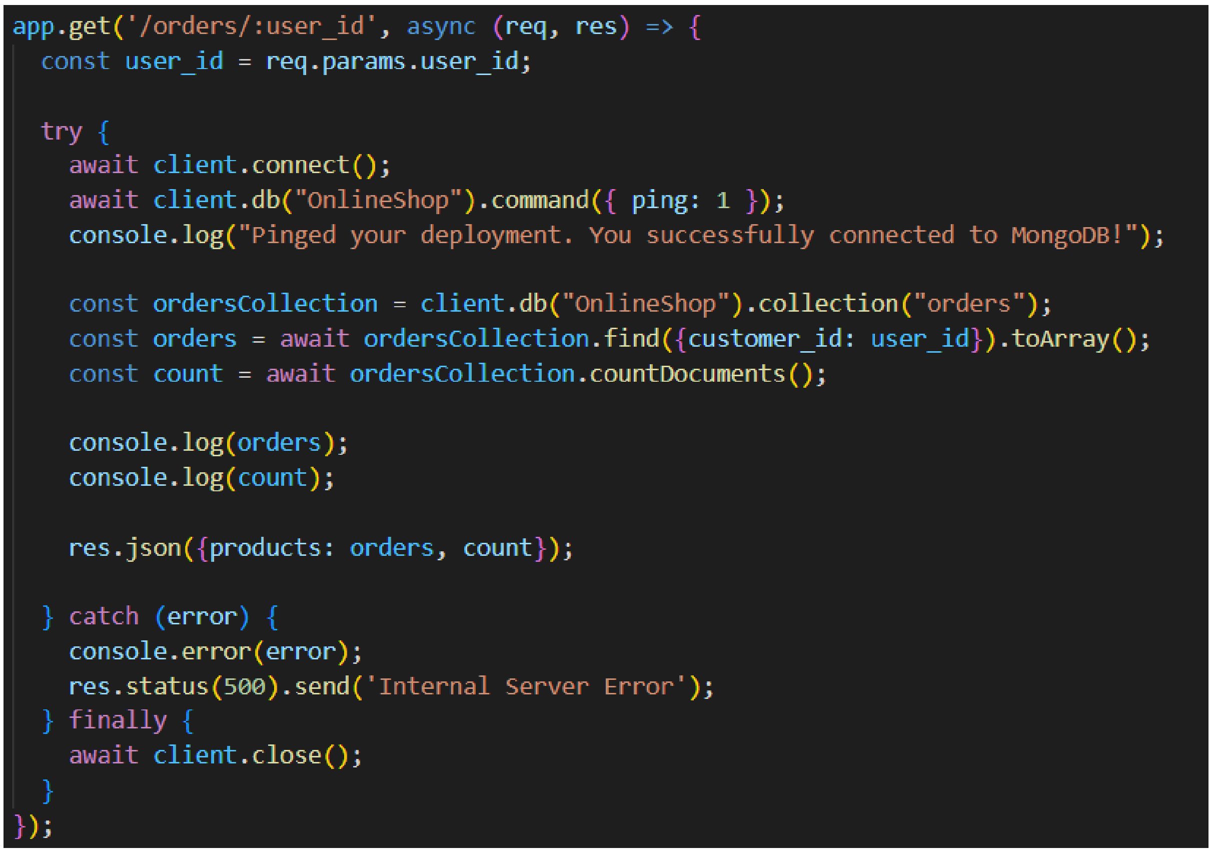
```javascript
app.get('/product/:id', async (req, res) => {
    const productID = req.params.id;
    if (!productID)
        res.status(400).json({error: 'Incorrect product id.'});

    try {
        await client.connect();
        const productsCollection = client.db('OnlineShop').collection('products');

        const product = await productsCollection.findOne({_id: new ObjectId(productID)});
        if (product)
            res.json(product);
        else
            res.status(404).json({error: 'Product not found.'});
    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to find product.'});
    } finally {
        await client.close();
    }
});
```

- Pozyskanie konkretnego użytkownika:



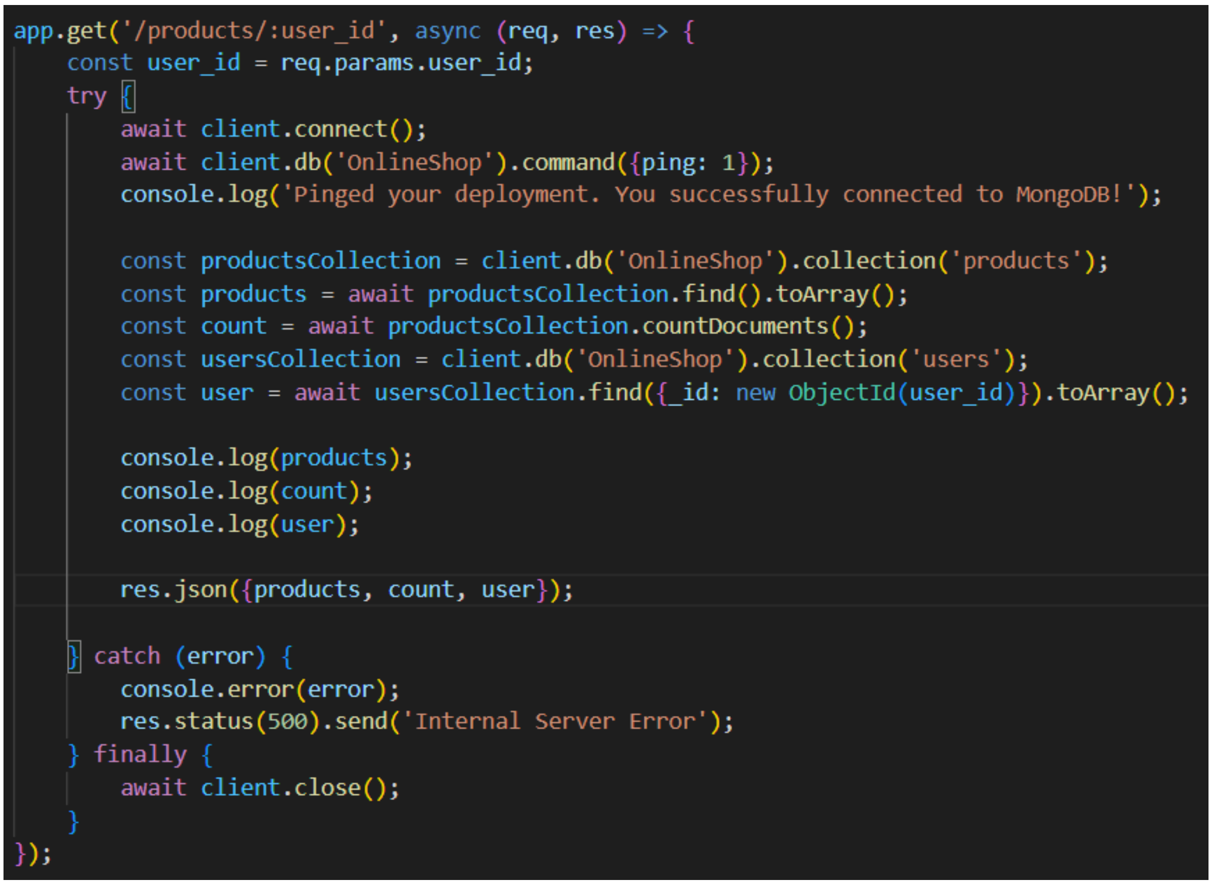
```
app.get('/user/:userID', async (req, res) => {
    const userID = req.params.userID;
    if (!userID)
        res.status(400).json({error: 'Incorrect user id.'});

    try {
        await client.connect();
        const usersCollection = client.db('OnlineShop').collection('users');

        const user = await usersCollection.findOne({_id: new ObjectId(userID)});
        if (user)
            res.json(user);
        else
            res.status(404).json({error: 'User not found.'});
    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to find user.'});
    } finally {
        await client.close();
    }
});
```

- Pozyskanie zamówień użytkownika:



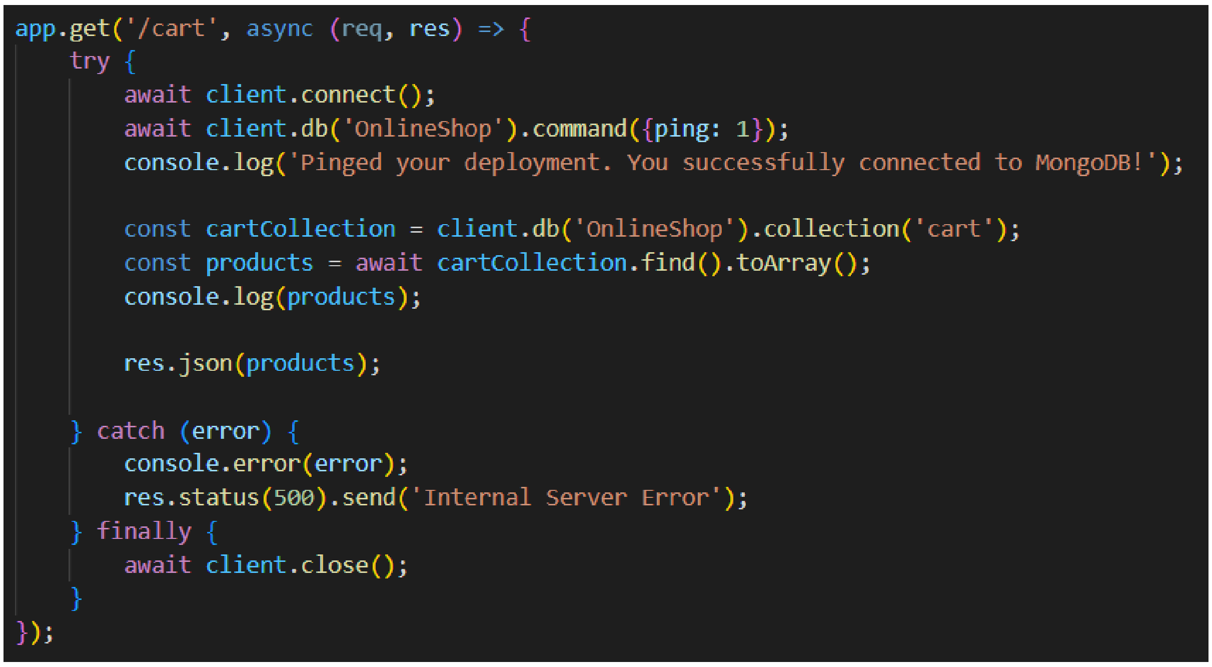
```
app.get('/orders/:user_id', async (req, res) => {
  const user_id = req.params.user_id;

  try {
    await client.connect();
    await client.db("OnlineShop").command({ ping: 1 });
    console.log("Pinged your deployment. You successfully connected to MongoDB!");

    const ordersCollection = client.db("OnlineShop").collection("orders");
    const orders = await ordersCollection.find({customer_id: user_id}).toArray();
    const count = await ordersCollection.countDocuments();

    console.log(orders);
    console.log(count);

    res.json({products: orders, count});

  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  } finally {
    await client.close();
  }
});
```

- Pozyskanie produktów sprzedawanych przez użytkownika:



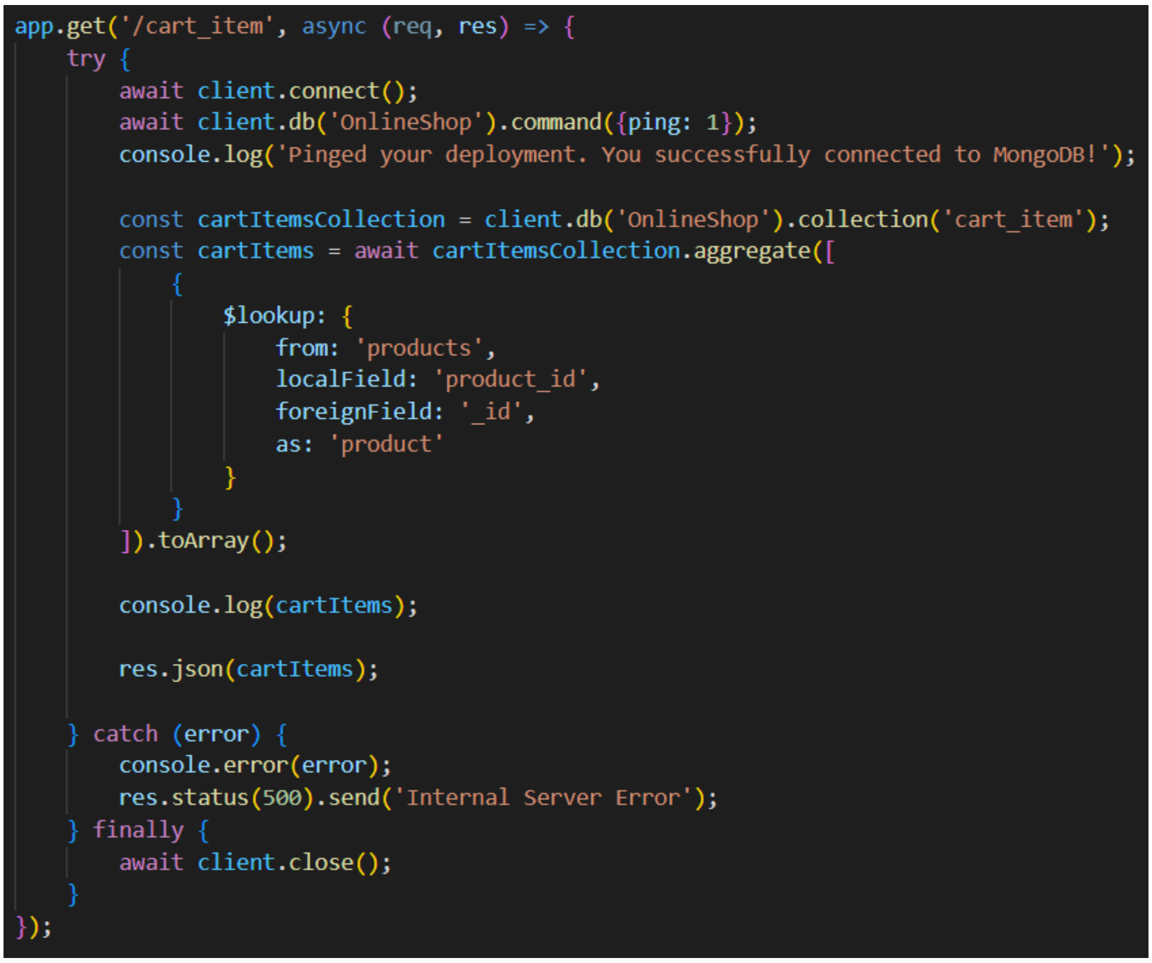
```
app.get('/products/:user_id', async (req, res) => {
    const user_id = req.params.user_id;
    try {
        await client.connect();
        await client.db('OnlineShop').command({ping: 1});
        console.log('Pinged your deployment. You successfully connected to MongoDB!');

        const productsCollection = client.db('OnlineShop').collection('products');
        const products = await productsCollection.find().toArray();
        const count = await productsCollection.countDocuments();
        const usersCollection = client.db('OnlineShop').collection('users');
        const user = await usersCollection.find({_id: new ObjectId(user_id)}).toArray();

        console.log(products);
        console.log(count);
        console.log(user);

        res.json({products, count, user});

    } catch (error) {
        console.error(error);
        res.status(500).send('Internal Server Error');
    } finally {
        await client.close();
    }
});
```

- Pozyskanie koszyka:



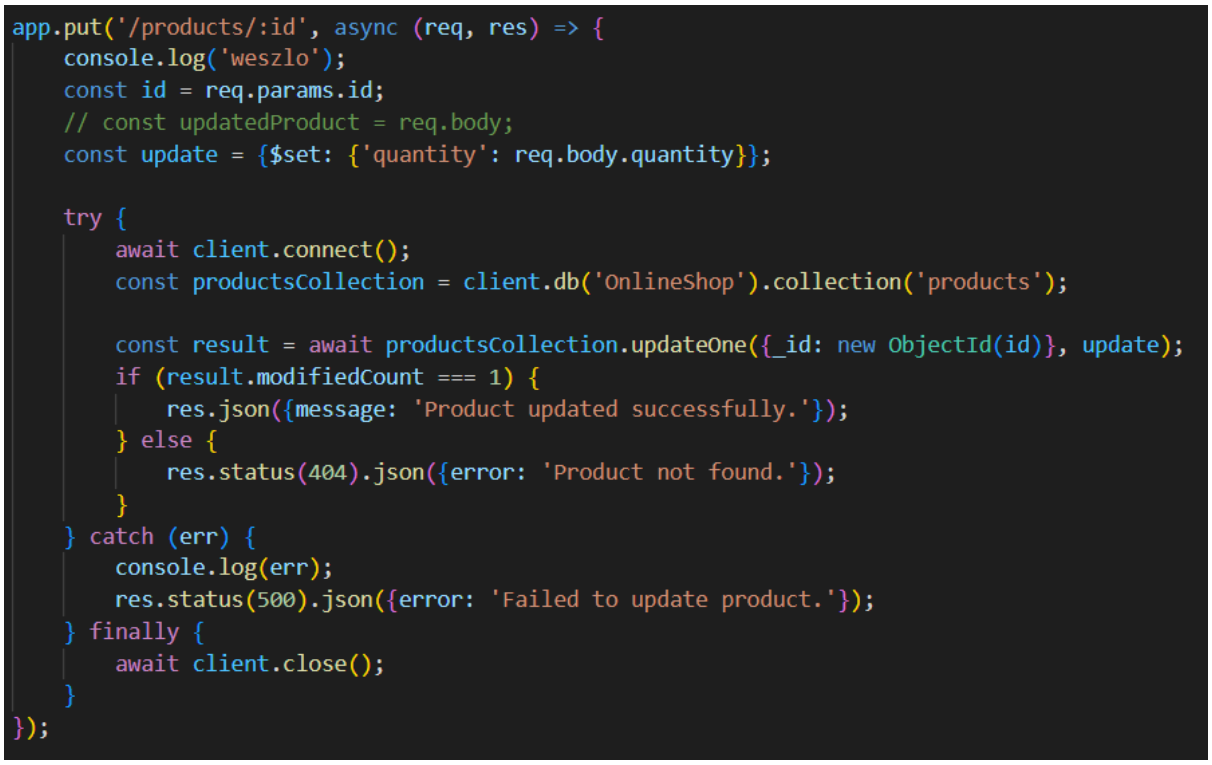
```
app.get('/cart', async (req, res) => {
    try {
        await client.connect();
        await client.db('OnlineShop').command({ping: 1});
        console.log('Pinged your deployment. You successfully connected to MongoDB!');

        const cartCollection = client.db('OnlineShop').collection('cart');
        const products = await cartCollection.find().toArray();
        console.log(products);

        res.json(products);

    } catch (error) {
        console.error(error);
        res.status(500).send('Internal Server Error');
    } finally {
        await client.close();
    }
});
```

- Pozyskanie produktów z koszyka:



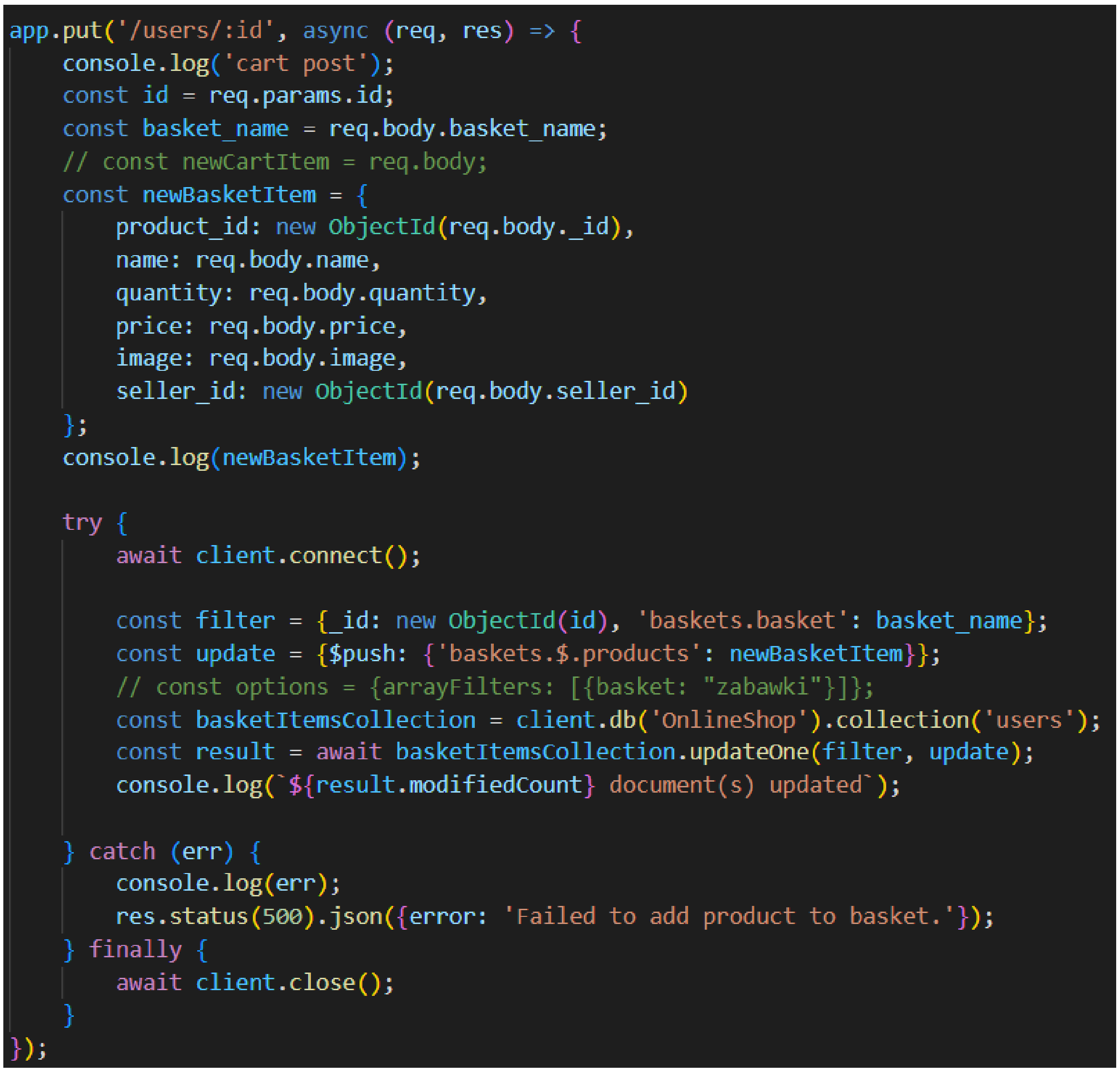
```javascript
app.get('/cart_item', async (req, res) => {
    try {
        await client.connect();
        await client.db('OnlineShop').command({ping: 1});
        console.log('Pinged your deployment. You successfully connected to MongoDB!');

        const cartItemsCollection = client.db('OnlineShop').collection('cart_item');
        const cartItems = await cartItemsCollection.aggregate([
            {
                $lookup: {
                    from: 'products',
                    localField: 'product_id',
                    foreignField: '_id',
                    as: 'product'
                }
            }
        ]).toArray();

        console.log(cartItems);

        res.json(cartItems);

    } catch (error) {
        console.error(error);
        res.status(500).send('Internal Server Error');
    } finally {
        await client.close();
    }
});
```

- Aktualizacja konkretnego produktu (zmiana dostępnej ilości):



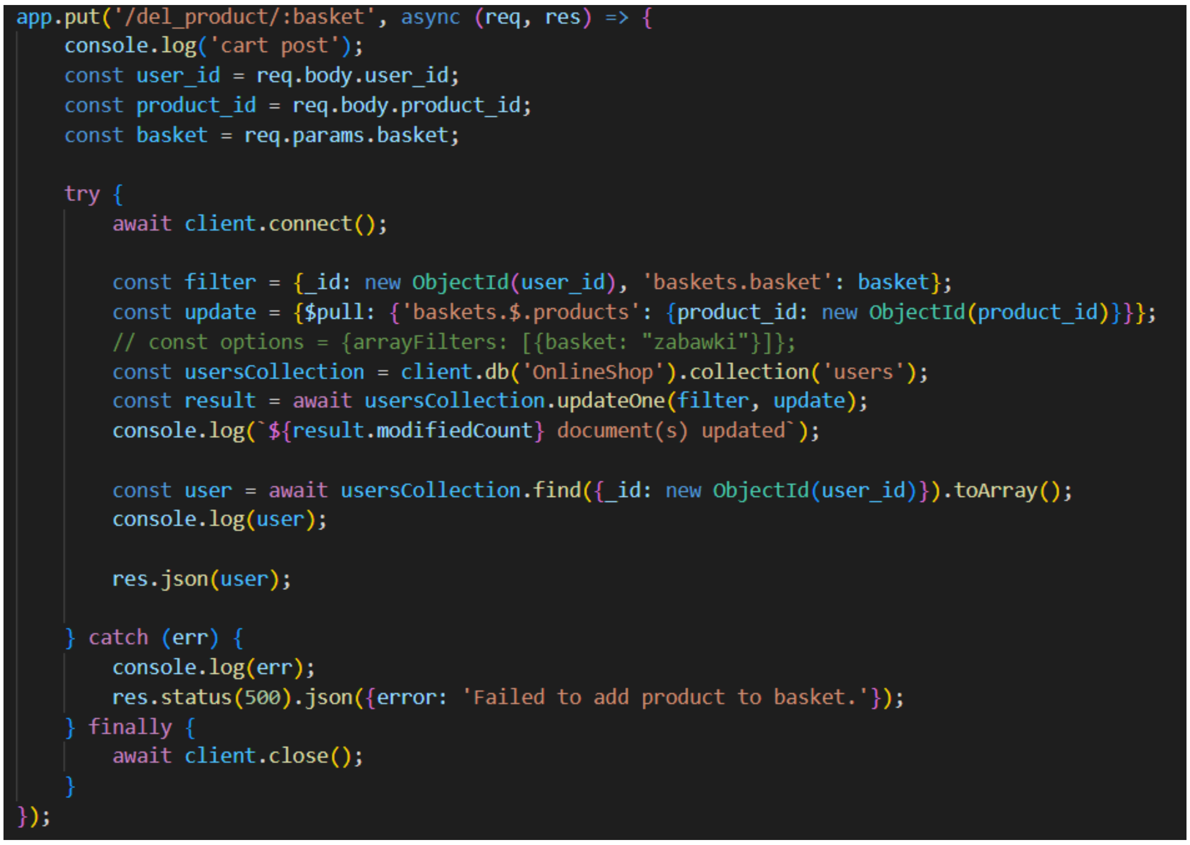
```javascript
app.put('/products/:id', async (req, res) => {
    console.log('weszlo');
    const id = req.params.id;
    // const updatedProduct = req.body;
    const update = {$set: {'quantity': req.body.quantity}};

    try {
        await client.connect();
        const productsCollection = client.db('OnlineShop').collection('products');

        const result = await productsCollection.updateOne({_id: new ObjectId(id)}, update);
        if (result.modifiedCount === 1) {
            res.json({message: 'Product updated successfully.'});
        } else {
            res.status(404).json({error: 'Product not found.'});
        }
    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to update product.'});
    } finally {
        await client.close();
    }
});
```

- Dodawanie produktu do koszyka:



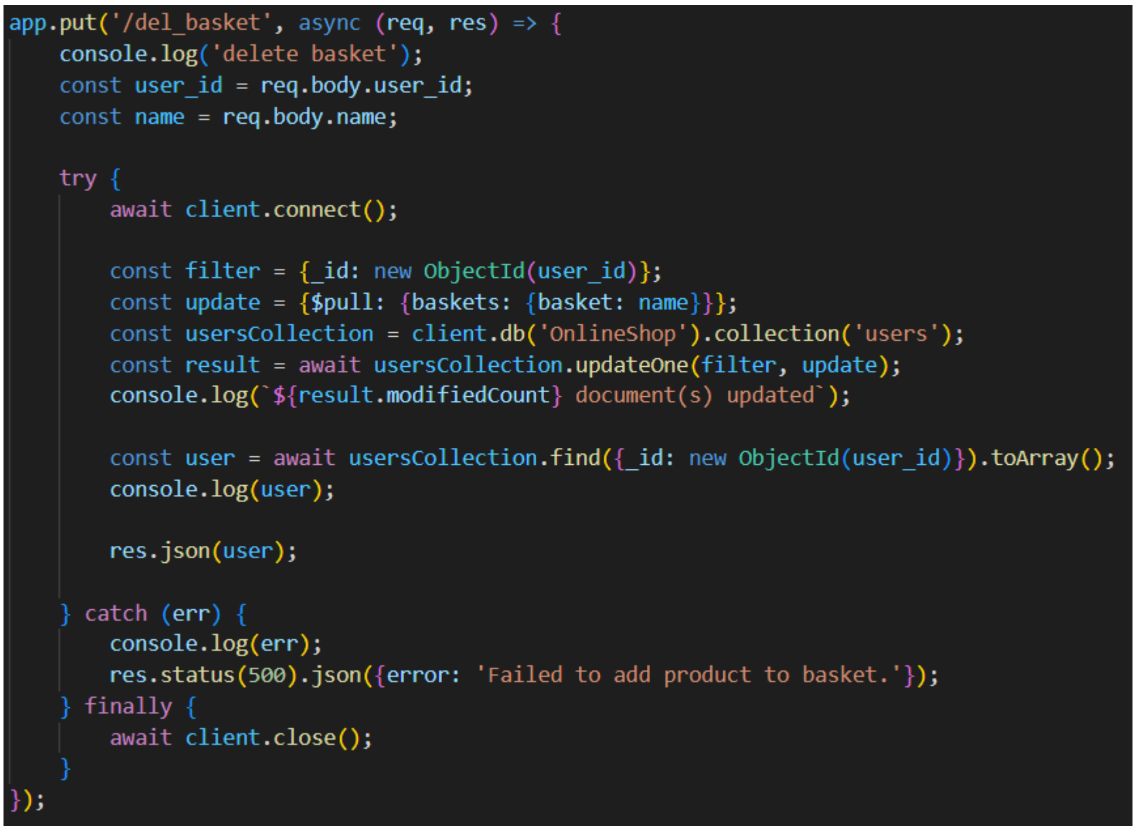
```javascript
app.put('/users/:id', async (req, res) => {
    console.log('cart post');
    const id = req.params.id;
    const basket_name = req.body.basket_name;
    // const newCartItem = req.body;
    const newBasketItem = {
        product_id: new ObjectId(req.body._id),
        name: req.body.name,
        quantity: req.body.quantity,
        price: req.body.price,
        image: req.body.image,
        seller_id: new ObjectId(req.body.seller_id)
    };
    console.log(newBasketItem);

    try {
        await client.connect();

        const filter = {_id: new ObjectId(id), 'baskets.basket': basket_name};
        const update = {$push: {'baskets.$.products': newBasketItem}};
        // const options = {arrayFilters: [{basket: "zabawki"}]};
        const basketItemsCollection = client.db('OnlineShop').collection('users');
        const result = await basketItemsCollection.updateOne(filter, update);
        console.log(`${result.modifiedCount} document(s) updated`);

    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to add product to basket.'});
    } finally {
        await client.close();
    }
});
```

- Usuwanie produktu z koszyka:



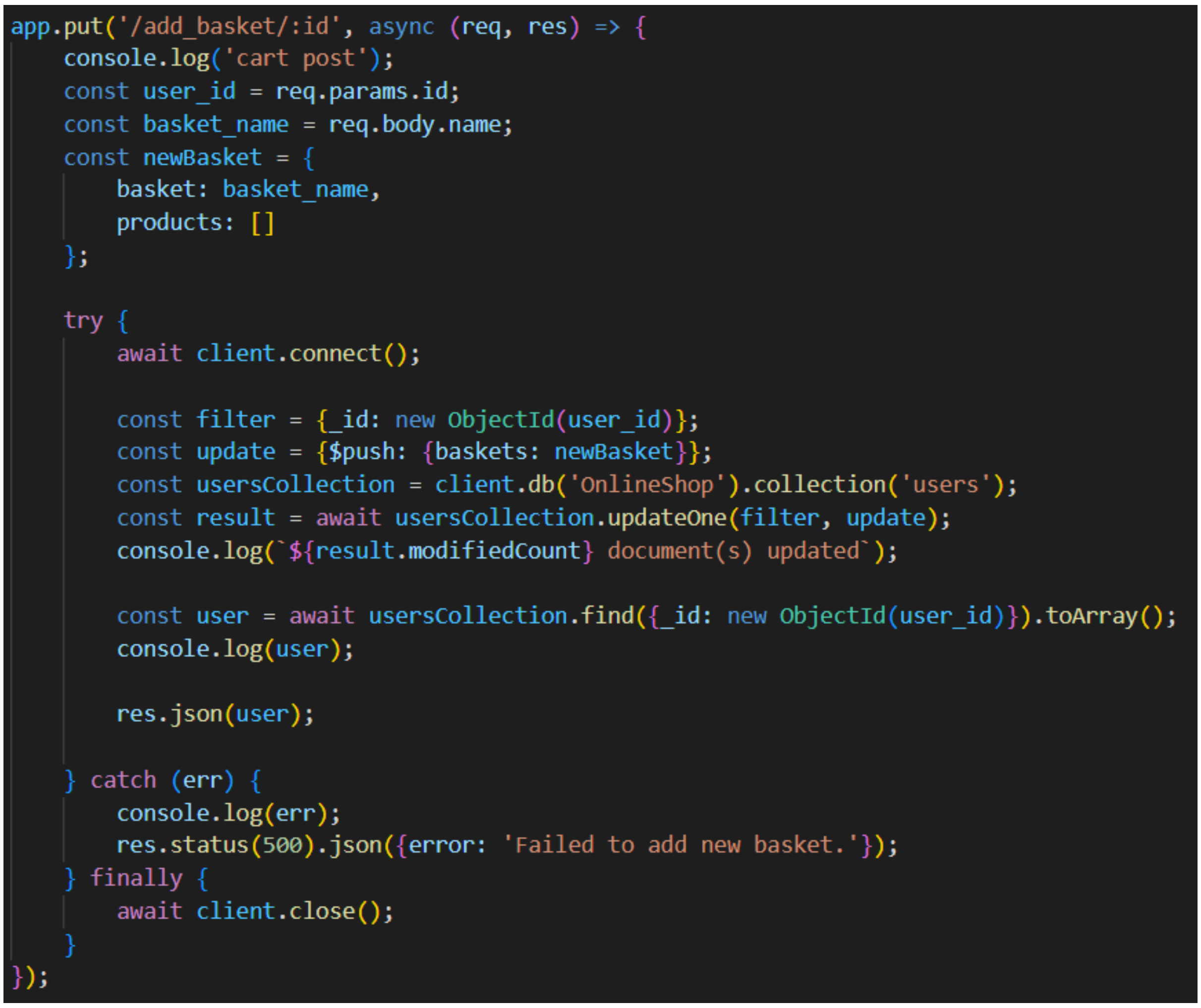
```javascript
app.put('/del_product/:basket', async (req, res) => {
    console.log('cart post');
    const user_id = req.body.user_id;
    const product_id = req.body.product_id;
    const basket = req.params.basket;

    try {
        await client.connect();

        const filter = {_id: new ObjectId(user_id), 'baskets.basket': basket};
        const update = {$pull: {'baskets.$.products': {product_id: new ObjectId(product_id)}}};
        // const options = {arrayFilters: [{basket: "zabawki"}]};
        const usersCollection = client.db('OnlineShop').collection('users');
        const result = await usersCollection.updateOne(filter, update);
        console.log(`${result.modifiedCount} document(s) updated`);

        const user = await usersCollection.find({_id: new ObjectId(user_id)}).toArray();
        console.log(user);

        res.json(user);

    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to add product to basket.'});
    } finally {
        await client.close();
    }
});
```

- Usuwanie koszyka:



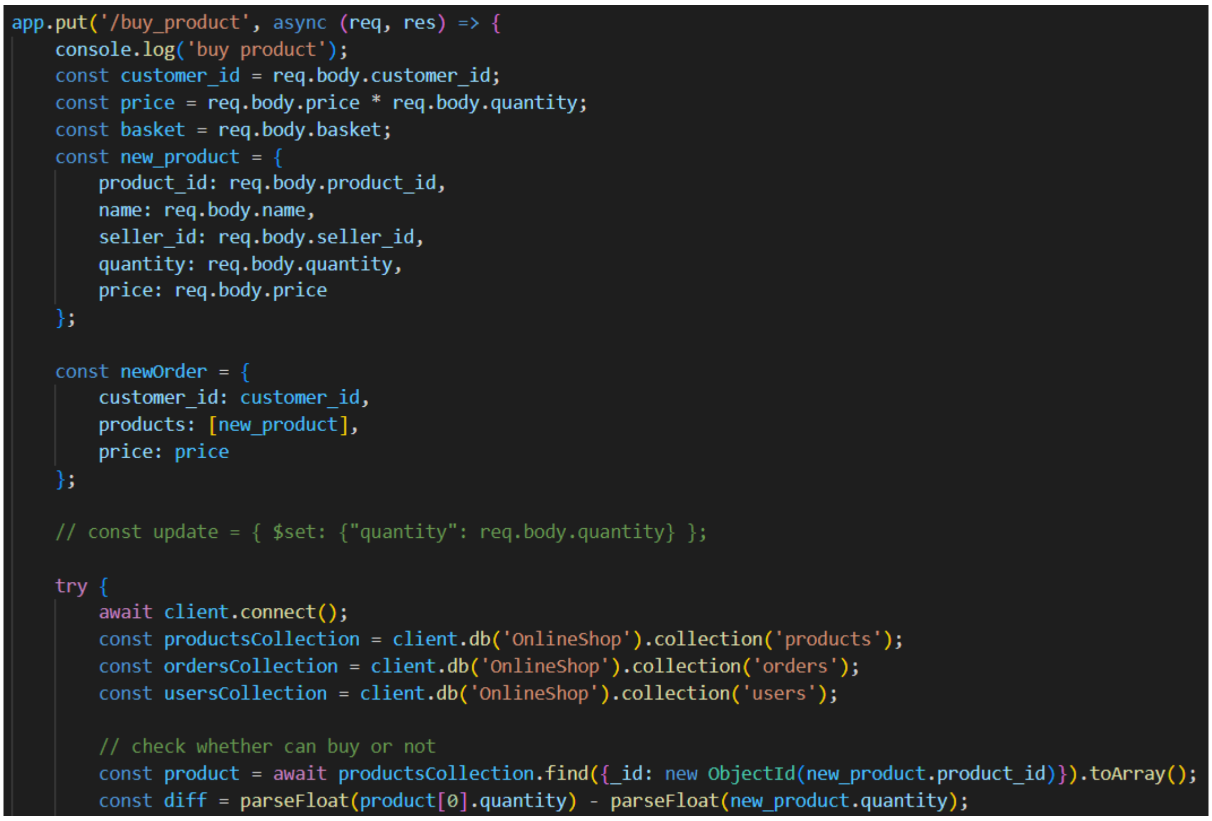
```javascript
app.put('/del_basket', async (req, res) => {
    console.log('delete basket');
    const user_id = req.body.user_id;
    const name = req.body.name;

    try {
        await client.connect();

        const filter = {_id: new ObjectId(user_id)};
        const update = {$pull: {baskets: {basket: name}}};
        const usersCollection = client.db('OnlineShop').collection('users');
        const result = await usersCollection.updateOne(filter, update);
        console.log(`${result.modifiedCount} document(s) updated`);

        const user = await usersCollection.find({_id: new ObjectId(user_id)}).toArray();
        console.log(user);

        res.json(user);

    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to add product to basket.'});
    } finally {
        await client.close();
    }
});
```

- Dodawanie koszyka

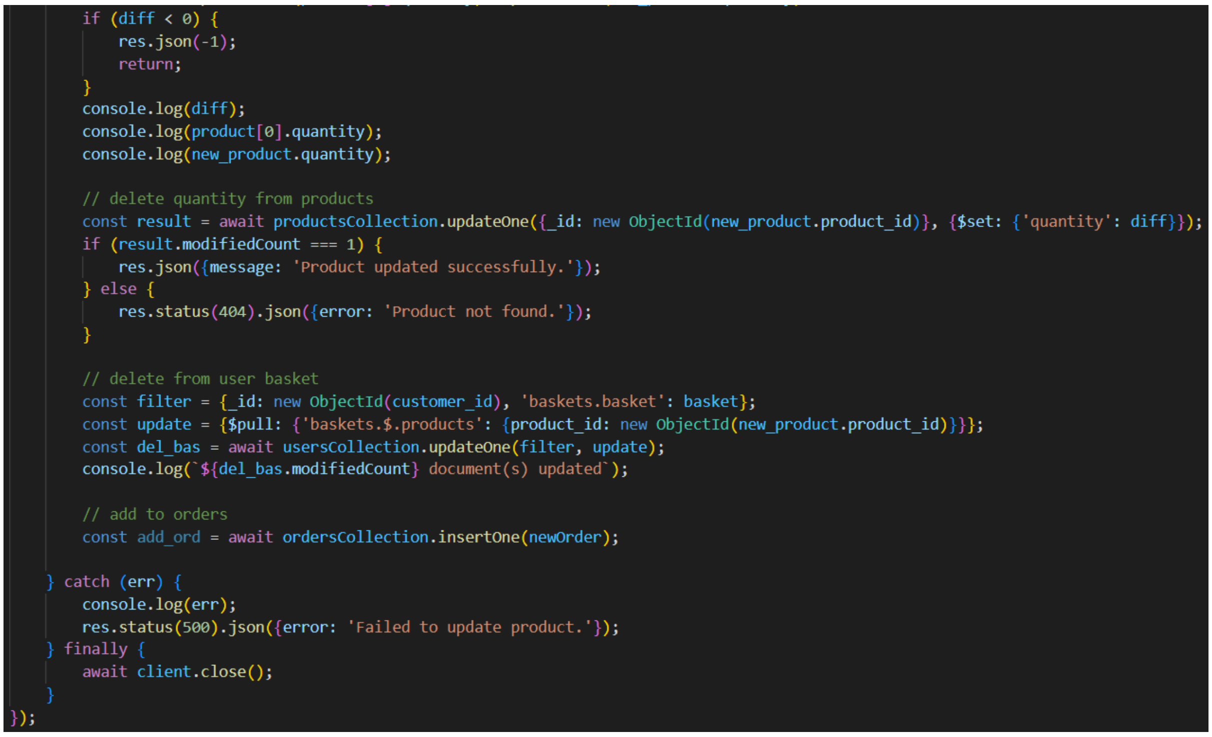


```javascript
app.put('/add_basket/:id', async (req, res) => {
    console.log('cart post');
    const user_id = req.params.id;
    const basket_name = req.body.name;
    const newBasket = {
        basket: basket_name,
        products: []
    };

    try {
        await client.connect();

        const filter = {_id: new ObjectId(user_id)};
        const update = {$push: {baskets: newBasket}};
        const usersCollection = client.db('OnlineShop').collection('users');
        const result = await usersCollection.updateOne(filter, update);
        console.log(`${result.modifiedCount} document(s) updated`);

        const user = await usersCollection.find({_id: new ObjectId(user_id)}).toArray();
        console.log(user);

        res.json(user);

    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to add new basket.'});
    } finally {
        await client.close();
    }
});
```

- Kupowanie produktu:

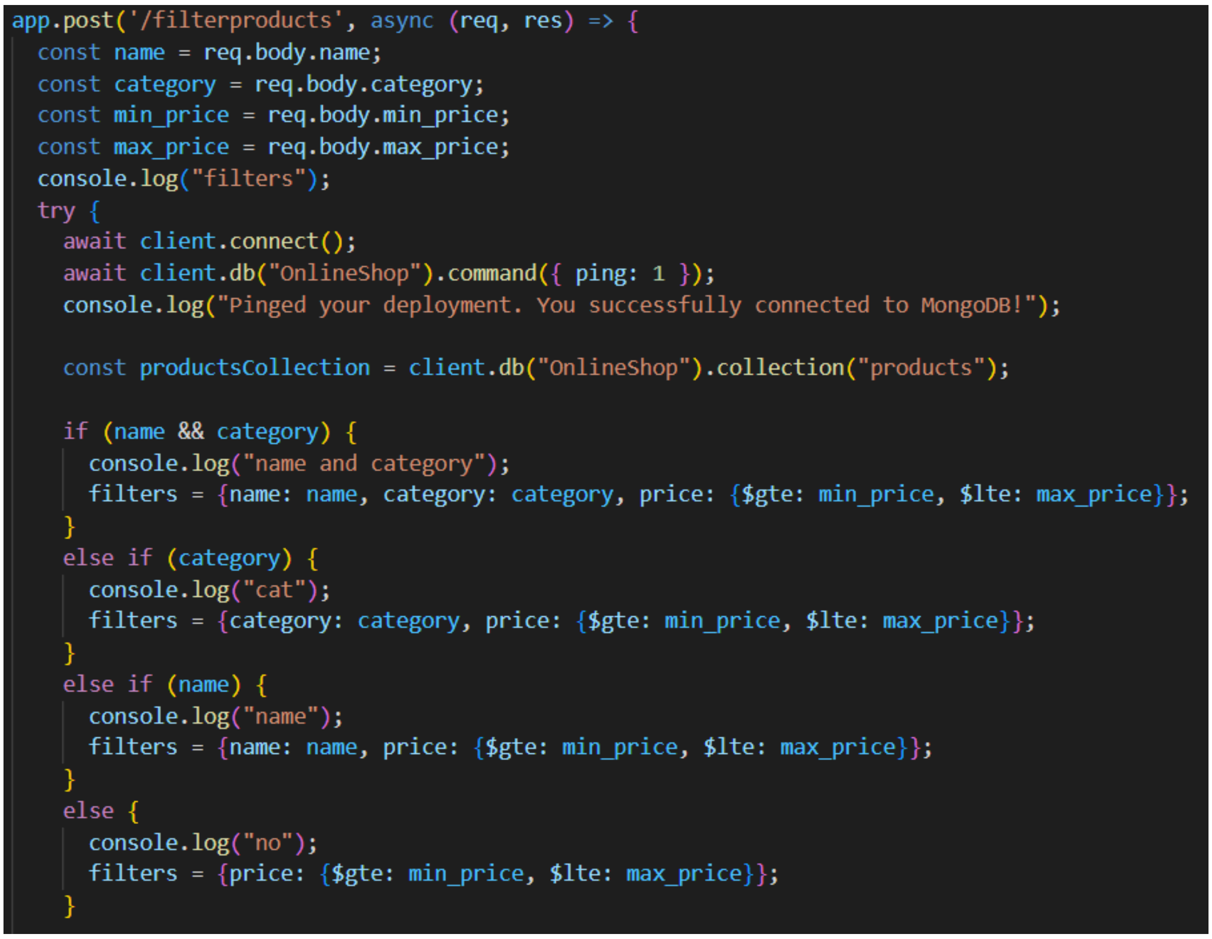


```
app.put('/buy_product', async (req, res) => {
    console.log('buy product');
    const customer_id = req.body.customer_id;
    const price = req.body.price * req.body.quantity;
    const basket = req.body.basket;
    const new_product = {
        product_id: req.body.product_id,
        name: req.body.name,
        seller_id: req.body.seller_id,
        quantity: req.body.quantity,
        price: req.body.price
    };

    const newOrder = {
        customer_id: customer_id,
        products: [new_product],
        price: price
    };

    // const update = { $set: {"quantity": req.body.quantity} };

    try {
        await client.connect();
        const productsCollection = client.db('OnlineShop').collection('products');
        const ordersCollection = client.db('OnlineShop').collection('orders');
        const usersCollection = client.db('OnlineShop').collection('users');

        // check whether can buy or not
        const product = await productsCollection.find({_id: new ObjectId(new_product.product_id)}).toArray();
        const diff = parseFloat(product[0].quantity) - parseFloat(new_product.quantity);
```

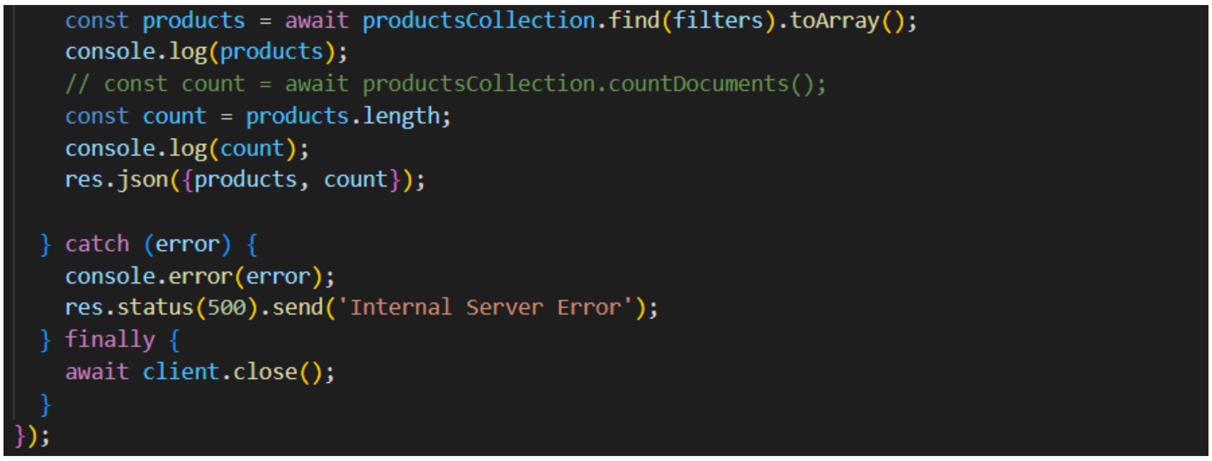


```
        if (diff < 0) {
            res.json(-1);
            return;
        }
        console.log(diff);
        console.log(product[0].quantity);
        console.log(new_product.quantity);

        // delete quantity from products
        const result = await productsCollection.updateOne({_id: new ObjectId(new_product.product_id)}, {$set: {'quantity': diff}});
        if (result.modifiedCount === 1) {
            res.json({message: 'Product updated successfully.'});
        } else {
            res.status(404).json({error: 'Product not found.'});
        }

        // delete from user basket
        const filter = {_id: new ObjectId(customer_id), 'baskets.basket': basket};
        const update = {$pull: {'baskets.$.products': {product_id: new ObjectId(new_product.product_id)}}};
        const del_bas = await usersCollection.updateOne(filter, update);
        console.log(`${del_bas.modifiedCount} document(s) updated`);

        // add to orders
        const add_ord = await ordersCollection.insertOne(newOrder);

    } catch (err) {
        console.log(err);
        res.status(500).json({error: 'Failed to update product.'});
    } finally {
        await client.close();
    }
});
```

- Filtrowanie produktów:



```
app.post('/filterproducts', async (req, res) => {
  const name = req.body.name;
  const category = req.body.category;
  const min_price = req.body.min_price;
  const max_price = req.body.max_price;
  console.log("filters");
  try {
    await client.connect();
    await client.db("OnlineShop").command({ ping: 1 });
    console.log("Pinged your deployment. You successfully connected to MongoDB!");

    const productsCollection = client.db("OnlineShop").collection("products");

    if (name && category) {
      console.log("name and category");
      filters = {name: name, category: category, price: {$gte: min_price, $lte: max_price}};
    }
    else if (category) {
      console.log("cat");
      filters = {category: category, price: {$gte: min_price, $lte: max_price}};
    }
    else if (name) {
      console.log("name");
      filters = {name: name, price: {$gte: min_price, $lte: max_price}};
    }
    else {
      console.log("no");
      filters = {price: {$gte: min_price, $lte: max_price}};
    }
```

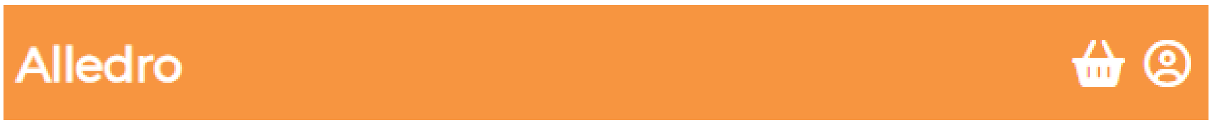


```
    const products = await productsCollection.find(filters).toArray();
    console.log(products);
    // const count = await productsCollection.countDocuments();
    const count = products.length;
    console.log(count);
    res.json({products, count});

  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  } finally {
    await client.close();
  }
});
```

4. Warstwa wizualna:

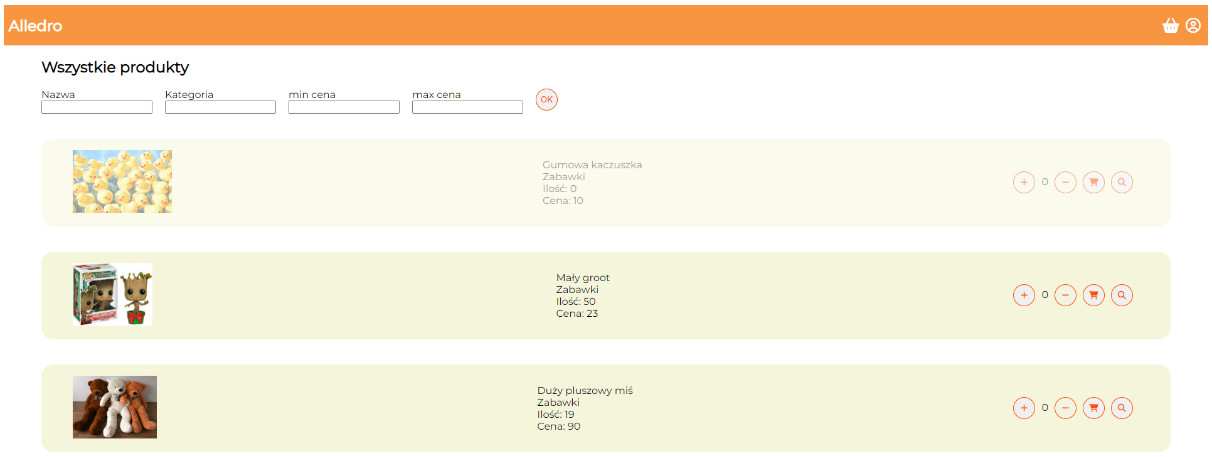
Warstwa wizualna składa się z komponentów:

- Menu-bar



Z którego można przejść do ekranu głównego (napis Alledro), koszyków użytkownika (koszyk) oraz informacji o użytkowniku (kółko z ludzikiem)

- All-products



Wszystkie produkty

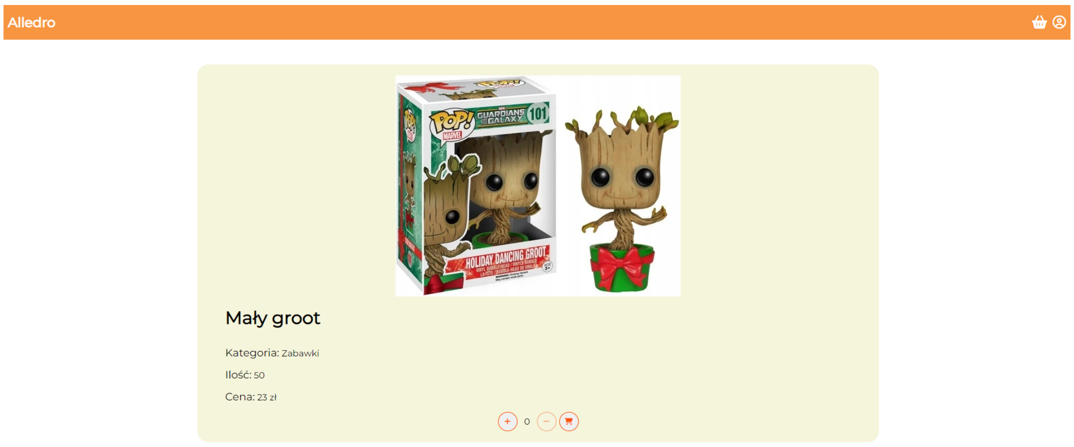
Gumowa kaczuszka
Zabawki
Ilość: 0
Cena: 10

Mały groot
Zabawki
Ilość: 50
Cena: 23

Duży pluszowy miś
Zabawki
Ilość: 19
Cena: 90

Jest równocześnie stroną startową. Z poziomu tej strony można dodawać produkty do koszyka, filtrować po odpowiednich polach oraz przejść do widoku indywidualnego produktu.

- Product-info



Mały groot

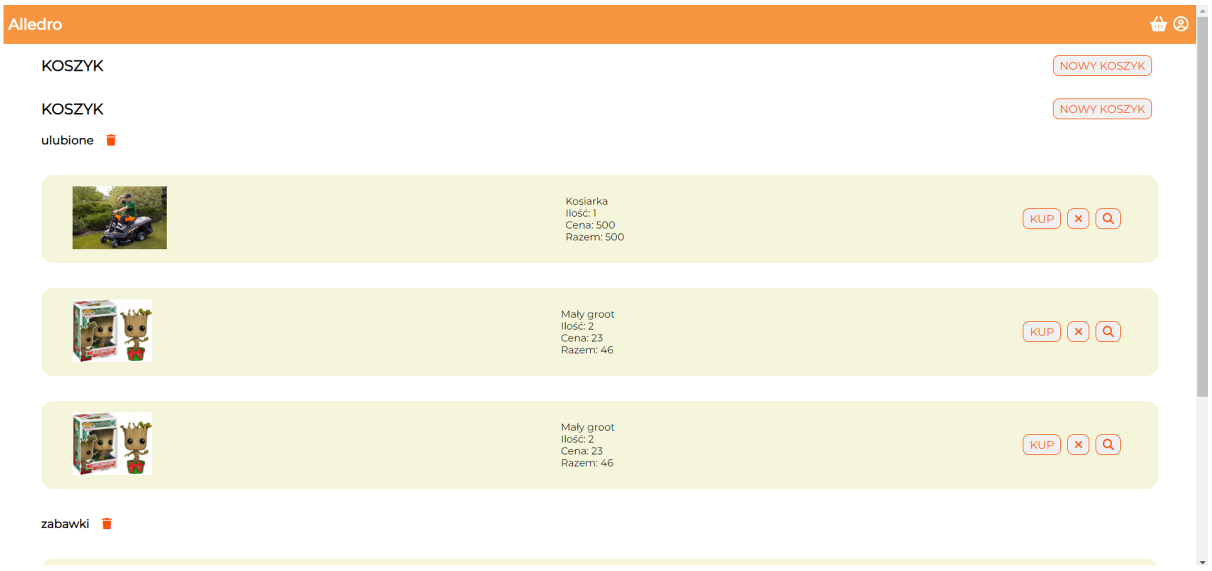
Kategoria: Zabawki

Ilość: 50

Cena: 23 zł

Strona dedykowana do informacji o danym produkcie.

- Basket



Alledro

KOSZYK NOWY KOSZYK

KOSZYK NOWY KOSZYK

ulubione

Kosiarka
Ilość: 1
Cena: 500
Razem: 500
KUP

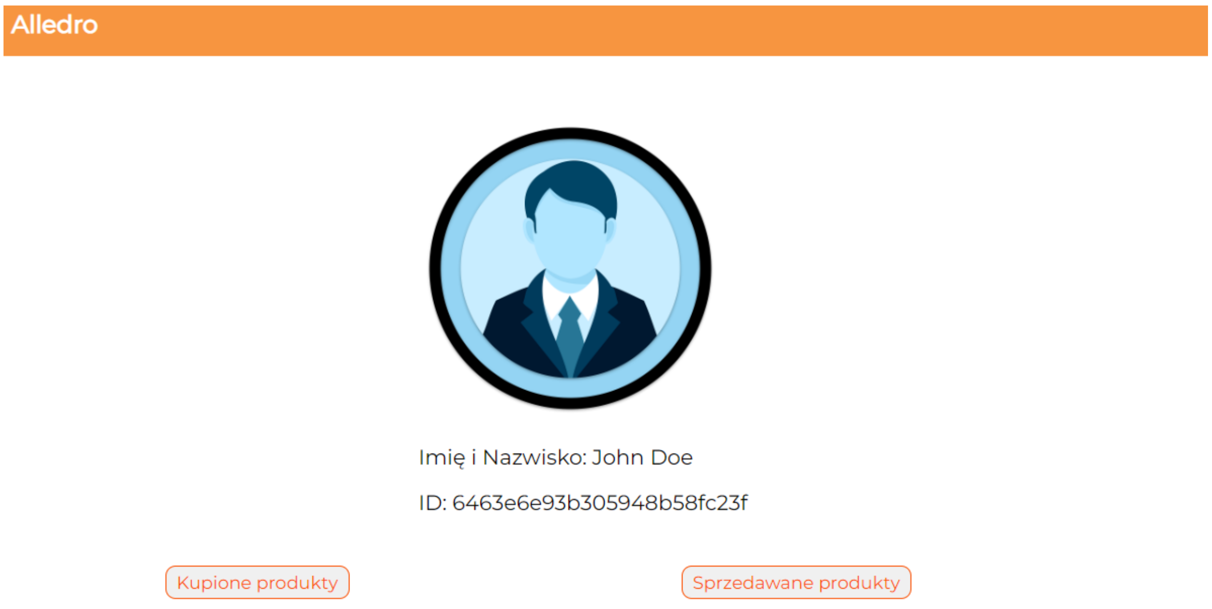
Mały groot
Ilość: 2
Cena: 23
Razem: 46
KUP

Mały groot
Ilość: 2
Cena: 23
Razem: 46
KUP

zabawki

Pokazuje stan koszyków użytkownika.

- User-info



Alledro

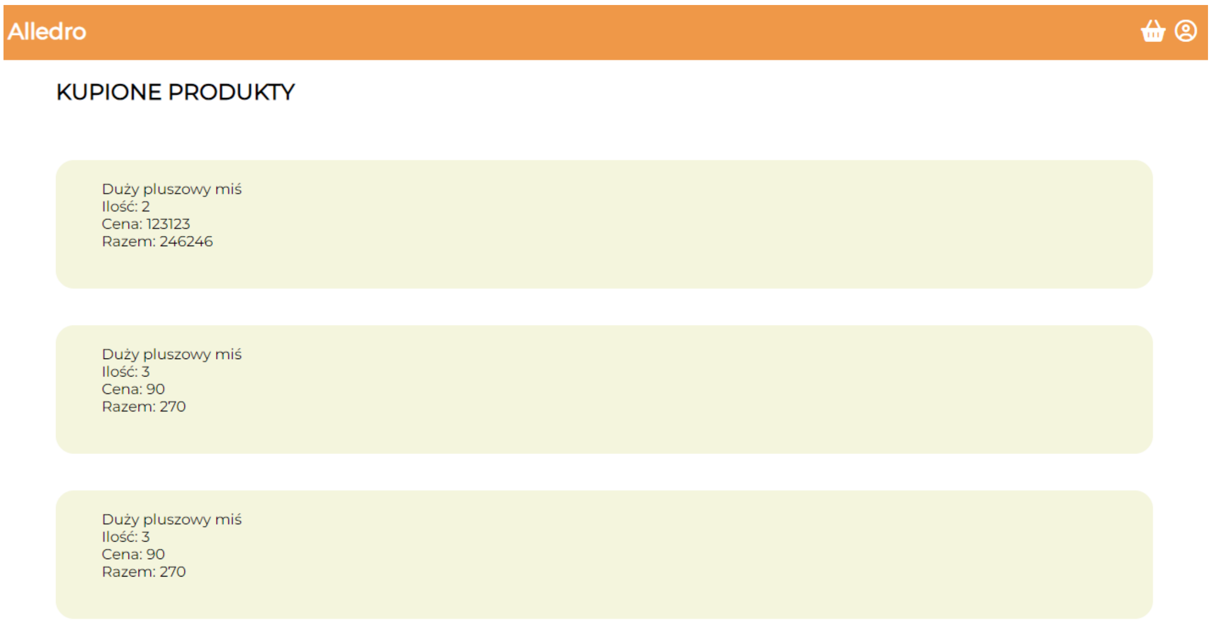
Imię i Nazwisko: John Doe

ID: 6463e6e93b305948b58fc23f

Kupione produkty Sprzedawane produkty

Widok informacji o użytkowniku. Z tego poziomu można przejść do zakupionych, oraz sprzedawanych produktów użytkownika.

- Purchased-product-list



KUPIONE PRODUKTY

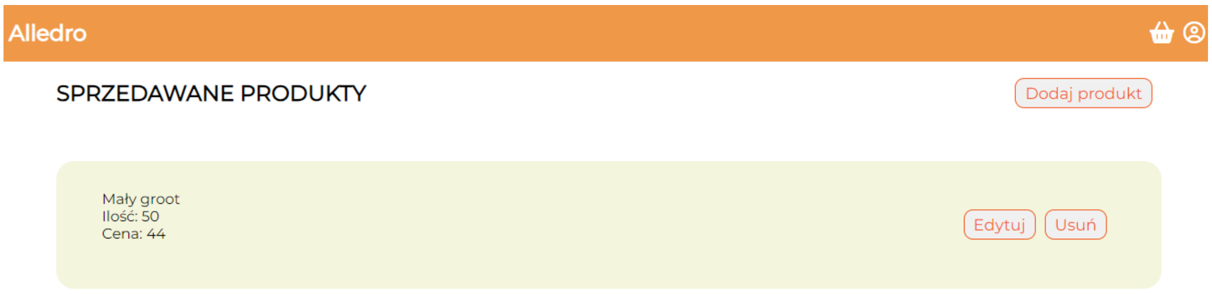
Duży pluszowy miś
Ilość: 2
Cena: 123123
Razem: 246246

Duży pluszowy miś
Ilość: 3
Cena: 90
Razem: 270

Duży pluszowy miś
Ilość: 3
Cena: 90
Razem: 270

Widok ten posiada informacje o zakupionych produktach przez użytkownika.

- Sold-product-list

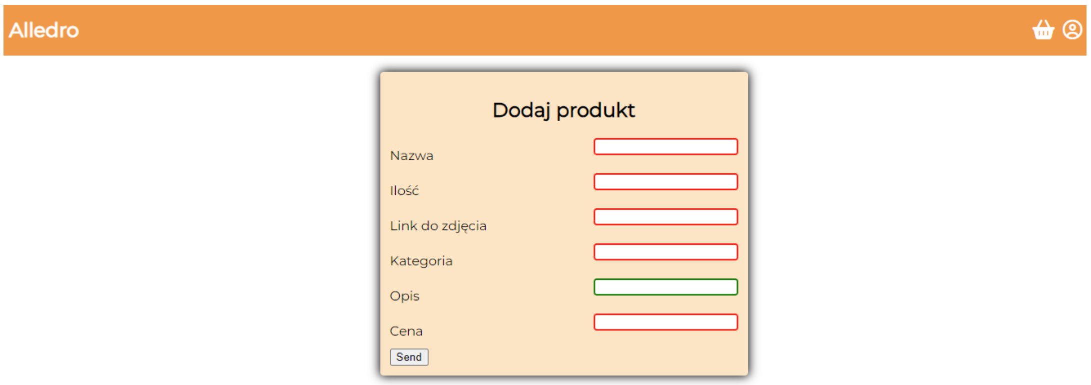


SPRZEDAWANE PRODUKTY

Mały groot
Ilość: 50
Cena: 44

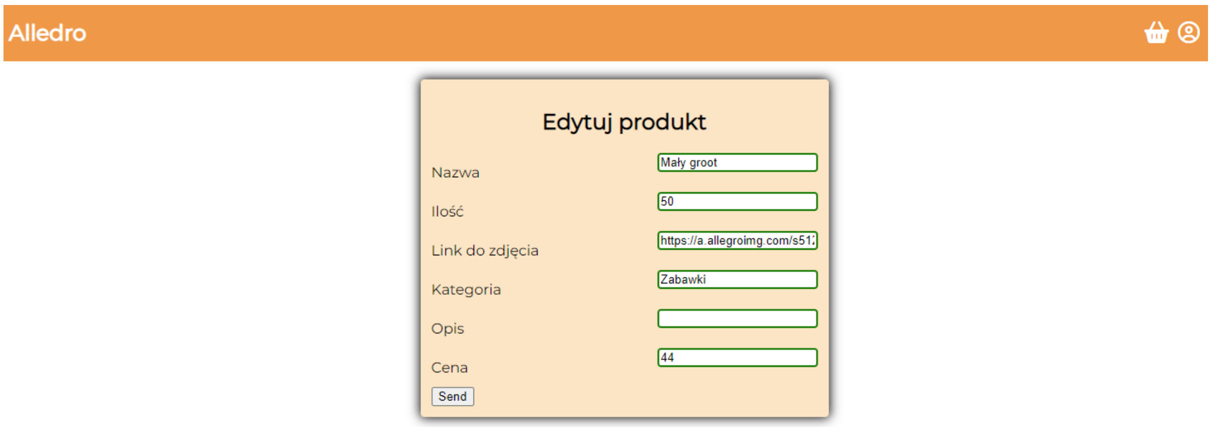
Widok ten posiada informacje o produktach sprzedawanych przez użytkownika

- Add-product



Widok ten posiada możliwość dodania nowego produktu

- Edit-product

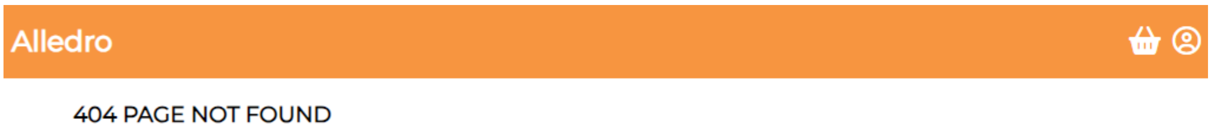

Alledro

Edytuj produkt

Nazwa	Mały groot
Ilość	50
Link do zdjęcia	https://a.allegroimg.com/s51
Kategoria	Zabawki
Opis	
Cena	44

Widok ten posiada możliwość modyfikacji produktu

- Page-not-found


Alledro

404 PAGE NOT FOUND

Widok ten wyskoczy gdy nastąpi próba przejścia do nieistniejącej strony w naszej domenie.