

**Wydział fizyki i informatyki stosowanej**

Przedmiot: Zaawansowane Technologie Internetowe

Temat: Aplikacja do zarządzania magazynem

Autor: *Mateusz Niepokój*

Kraków, 2023

Spis treści

[1. Cel projektu 3](#_Toc139140397)

[2. Temat projektu 3](#_Toc139140398)

[3. Architektura aplikacji 3](#_Toc139140399)

[4. Serwer 3](#_Toc139140400)

[5. Klient 6](#_Toc139140401)

[6. Baza danych 10](#_Toc139140402)

[7. Informacje uruchomieniowe 10](#_Toc139140403)

[8. Wnioski 11](#_Toc139140404)

# Cel projektu

Celem projektu było napisanie aplikacji klient serwer zrealizowanej zgodnie z wzorcem RESTful (Java API for RESTful Web Services JAX-RS). Część serwerowa powinna wykorzystywać serwer bazy danych (relacyjny bądź NoSQL). Klient może być zrealizowany z wykorzystaniem serwisu WWW i języka Javascript lub może to być aplikacja mobilna. Dodatkowo aplikacja powinna wykorzystywać programowanie Aspektowe.

# Temat projektu

Tematem projektu była aplikacja webowa służąca do zarządzania magazynem. Użytkownik dzięki niej będzie miał możliwość monitorowania stanu magazynu, zgłaszania prośby o dostawę nowych produktów oraz śledzenia statusu realizowanego zamówienia.

# Architektura aplikacji

Architekturę aplikacji można podzielić na 3 części:

* Serwer aplikaci
* Aplikację kliencką
* Serwer bazodanowy

Klient aplikacji korzysta z protokołu HTTP, aby pobierać informacje z serwera. Dane na serwerze są przechowywane w oddzielnym serwerze baz danych. Przesyłane dane są serializowane do formatu JSON i przesyłane między serwerem a klientem. Ważną rolę w aplikacji odgrywa także technologia WebSocket. Zalogowani użytkownicy są połączeni z serwerem za pomocą WebSocket, umożliwiając bieżącą komunikację. W przypadku, gdy jeden z użytkowników dokonuje modyfikacji w bazie danych, serwer wysyła informację do wszystkich klientów aplikacji, aby zaktualizować dane.

# Serwer

Aplikacja została przygotowana z wykorzystaniem języka Java w wersji 11. Platformą programistyczną aplikacji została wybrana Jakarta EE. Do przygotowania usług sieciowych typu REST został wykorzystany zbiór narzędzi JAX-RS. Aplikacja została umieszczona na serwerze TomEE w wersji 10.0.27.

* 1. Logika

Pod względem logiki, aplikację można podzielić na 4 części:

* Użytkownik(‘user’) – jest to część pozwalająca na autoryzację użytkownika
* Przechowywane(‘stored product’) – elementy obejmują przechowywanie produktów w magazynie
* Zamówienia (‘ordered product’)– obejmuje wysyłanie ogłoszeń o zapotrzebowaniu na dany produkt
* Dostawy (‘delivered product’) – obejmuje dostawę produktu pomiędzy dwoma magazynami
  1. Struktura projektu

Pod względem struktury serwer można podzielić na:

* Controller – zawiera zbiór entpointów co całą logiką aplikacji, co pozwala na wykonywanie zapytań restowych na serwer Obraz zawierający tekst, zrzut ekranu, Czcionka

  Opis wygenerowany automatycznie

Rysunek 1 Endpoint wysyłający klientowi listę dostaw

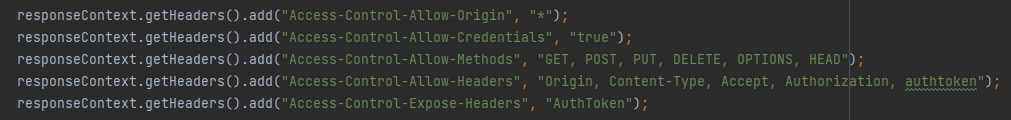
* Model – zawiera zbiór obiektów, wchodzących w skład zapytań oraz odpowiedzi serwera
* Database – jest to zbiór metod bazy danych odpowiadających za wykonywanie. Do komunikacji z bazą danych został wykorzystany interfejs jdbc.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

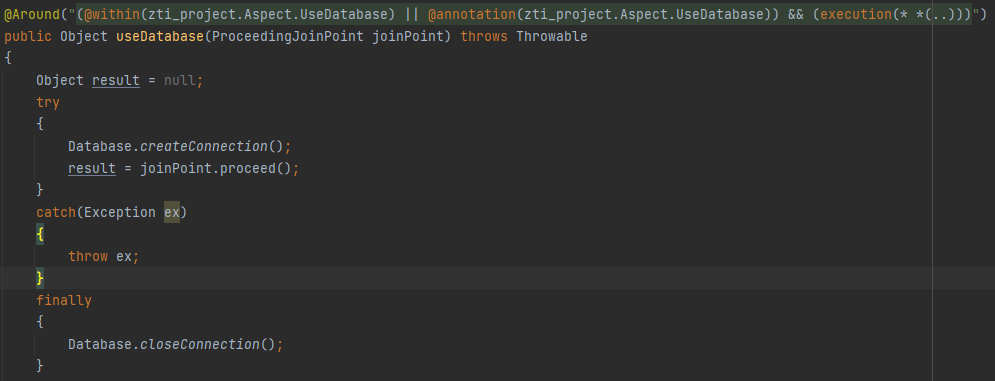
Rysunek 2 Diagram klas aplikacji serwerowej

* 1. Interesujące elementy
* Auth token – w aplikacji został zaimplementowany token, pozwalający na autoryzację użytkonika. Po zalogowaniu serwer przesyła w odpowiedzi klientowi token autoryzacyjny. Klient musi następnie dołączać owy token do każdego kolejnego zapytania. Token składa się z trzech części: nazwy użytkownika, daty wygaśnięciu oraz losowo wygenerowanej wartości tokenu. Po zalogowaniu wartość tokenu jest zapisywana do bazy danych i porównywana przy każdym zapytaniu
* CorsFilter – w celu poprawnego działania aplikacji został zdefiniowany cors filter.



Rysunek 3 Implementacja cors filter

* AspectJ – w aplikacji został dodatkowo wykorzystane programowanie aspektowe. Do skonfigurowania aspektu został wykorzystany plik konfiguracyjny mavena pom.xml oraz plugin ‘AspectJ’, znajdujący się w Intelij Ultimate marketplace. Aspekty zostały wykorzystane do obsługi błędów oraz do otwierania oraz zamykania połączenia z bazą danych



Rysunek 4 Implementacja aspektu służącego na obsługi połączenia z bazą danych

* Websocket – w celu konfiguracji websocketów aplikacja została podłączona do serwera GlassFish. Implementacja tego rozwiązania, spowodowała konieczność samodzielnego zdefiniowania klasy do obsługi JsonProvidera. W tym celu, do singletonu aplikacji została dodana klasa JacksonJsonProvider, będąca częścią pakietu GlassFish. Moduł websocketowy został wykorzystany do informowaniu klienta o zmianie bazy danych.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Rysunek 5 Metoda klasy typu WebSocket informujące klienta o zmianie przechowywanych produktów

# Klient

Klient aplikacji został napisany przy pomocy biblioteki React. Aplikcja składa się z hierarchicznego szeregu componentów wyświetlających kolejne części logiki aplikacji. Całość została zrealizowana w konwencji SPA.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Rysunek 6 Metoda renderująca komponentu ProductManager. Renderowane wewnątrz niej komponenty stanowią część główną frontu aplikacji.

Zapytania do serwera, umożliwiające odbiór i modyfikację tabel, zostały zaimplementowane z wykorzystaniem metody fetch

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Rysunek 7 Implementacja dodawania nowej kategorii produktów na serwer

Aktualizacja danych następuje wtedy, gdy zmienią się dane na serwerze. Do tego celu zostały wykorzystane web sockety.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 8 Metoda pobierająca z serwera nowe dane na podstawie otrzymanego komunikatu

Obraz zawierający tekst, zrzut ekranu, diagram, design

Opis wygenerowany automatycznie

Rysunek 9 Fragment widoku aplikacji z pokazaną listą produktów w magazynie

Aplikacja pozwala na dodawanie i usuwywanie produktów znajdujących się w magazynie. Możliwe, jest także dodanie nowego typu produktów.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

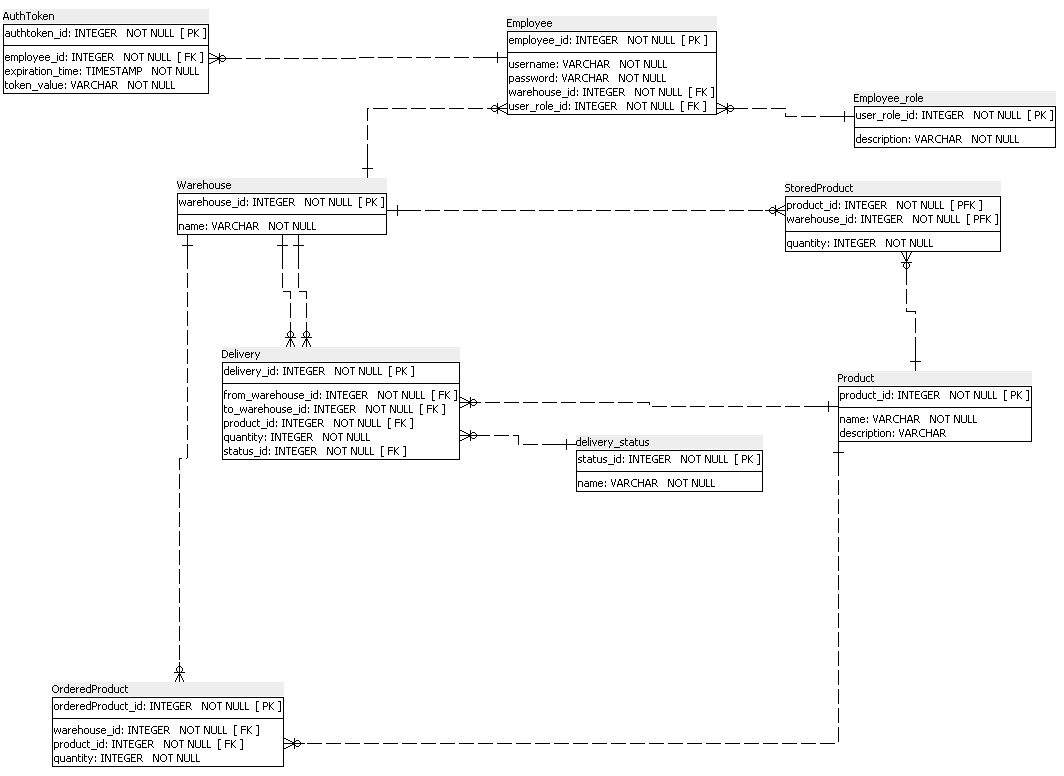
Opis wygenerowany automatycznie

Rysunek 10 Zrzut ekranu pokazujący wygląd aplikacji w uwzględnieniem zamówień i ich realizacji.

W Aplikacji istnieje możliwość dodawania nowego zamówienia. Po jego dodaniu, u wszystkich użytkowników pozostałych magazynów wyświetla się rekord z zamówieniem, z możliwością zaakceptowania zamówienia. Po zaakceptowaniu dane zamówienie przechodzi na listę dostaw. Użytkownik wystawiający ogłoszenie ma wtedy możliwość potwierdzenia dostawy produktu, natomiast użytkownik akceptujący zamówienie może anulować dostawę. Po zakończeniu transakcji, należy potwierdzić wyświetlenie statusu co będzie skutkować usunięciem rekordu z widoku użytkownika.

# Baza danych

Jako bazę danych została wykorzystana baza PostgresSQl. Została ona hostowana na stronie https://www.elephantsql.com. Widoczne na kontach testowych dane zostały dodane przez terminal dostępny w elephant sql w przypadku dodawania użytkowników, bądź przez api restowe jak w przypadku tabel powiązanych z produktami. Wykorzystywane tabele zostały wygenerowane w programie SQL Power Architect.



Rysunek 11 Widok tabel wykorzystywanych w logice aplikacji

# Informacje uruchomieniowe

Serwer:

Krok 1: Otworzyć backend/zti\_projct w IDE Intelij Idea Ultimate

Krok 2: Należy pobrać plugin 'AspectJ' z market place Intelij Ultimate

Krok 3: Skonfigurować projekt przy pomocy mavena

Krok 4: Skonfigurować serwer aplikacji

- Ustawić port: Http 8086

- ustawić URL: http://localhost:8086/ZTI\_project\_war\_exploded/

- dodać do Deployment ZTI\_project:war exploded

- preferowana wersja serwera TomEE 10.0.27

Krok 5: Uruchomić serwer

Klient:

Krok 1: Należy upewnić się, że pobrany został manager pakietów npm

Krok 2: Wejść poprzez terminal do folderu frontend\warehouse-app

Krok 3: uruchomić komendę 'npm install'

Krok 4: uruchomić komendę 'npm start'

Krok 5: Przejść w przeglądarce na URL: <http://localhost:3000/>

Dane do logowanie:

Użytkownik 1:

* user1
* pass1

Użytkownik 2:

* user2
* pass2

# Wnioski

Głównym wnioskiem jest to, że Jakarta EE i JAX-RS są potężnymi narzędziami do tworzenia skalowalnych i bezpiecznych aplikacji internetowych. Dzięki ich zaawansowanym funkcjom, takim jak obsługa transakcji, bezpieczeństwo uwierzytelniania i autoryzacji, oraz zarządzanie zasobami, można budować solidne i niezawodne systemy. Napotkane błędy, takie jak "Internal Server Error", były nieuniknione, ale po ich skutecznym rozwiązaniu cały proces stawał się łatwy i przyjemny. Ważne jest, aby umiejętnie korzystać z narzędzi diagnostycznych, takich jak logi i debugowanie, aby identyfikować i rozwiązywać problemy. JAX-RS dostarczał wygodne narzędzia do definiowania ścieżek, parametrów i formatów danych, co umożliwiło tworzenie spójnych i intuicyjnych interfejsów API. Mimo że warstwa front-end nie była głównym celem projektu, jej umiarkowane opisanie jest istotne. Wymagane było tworzenie prostych formularzy i interfejsów użytkownika, które integrowały się z back-endem. Dodatkowym istotnym wnioskiem z projektu na zaawansowane techniki internetowe jest wykorzystanie AspectJ, które dostarcza wiele nowych możliwości, niedostępnych w czystym języku Java. AspectJ to rozszerzenie języka, które umożliwia programistom definiowanie aspektów, czyli modułów kodu, które przechwytują i modyfikują działanie innych komponentów.

Podsumowując, projekt na zaawansowane techniki internetowe z wykorzystaniem Jakarta EE i JAX-RS dostarczył cennych wniosków. Pomimo możliwości napotkania błędów, które jednak można skutecznie rozwiązać, proces tworzenia aplikacji był łatwy i przyjemny. Również zrozumienie zasad RESTful API oraz umiarkowane zaangażowanie w warstwę front-end przyczyniły się do osiągnięcia satysfakcjonujących rezultatów.