



Wyższa Szkoła Informatyki i Zarządzania
Kolegium Informatyki Stosowanej, Informatyka
Rafał Wątroba, w65575

System do zarządzania stanem magazynu z wykorzystaniem technologii RFID.

Prowadzący:

Dr. Inż. Arkadiusz Lewicki

Rzeszów, 23.06.2024

Spis treści

Opis Wybranego Tematu i Funkcjonalności Systemu	3
Temat.....	3
Główny Cel.....	3
Funkcjonalności Systemu:.....	3
Opis wybranego stosu technologicznego	3
Wymagania.....	4
Opis jak uruchomić aplikację	4
Diagram Baz Danych	7
Diagramy.....	8
Podsumowanie i wnioski	9

Opis Wybranego Tematu i Funkcjonalności Systemu

Temat

System do zarządzania stanem magazynu z wykorzystaniem technologii RFID

Główny Cel

System zarządzania magazynem w postaci aplikacji webowej, umożliwiającej usprawnienie i zautomatyzowanie procesów obsługi stanu magazynowego za pomocą technologii RFID. Wpłynie on także na efektywność zarządzania stanem magazynowymi.

Funkcjonalności Systemu:

- Zarządzanie użytkownikami
 - Autoryzacja użytkowników za pomocą loginu i hasła
 - Różne poziomy dostępu
- Zarządzanie stanem magazynu
 - Dodawanie produktów do systemu
 - Edycja danych produktów
 - Usuwanie produktów
- Zarządzanie RFID
 - Rejestracja tagów RFID poprzez dodanie ich oraz przypisanie do danego produktu
 - Odczyt tagów RFID – odczyt danych z tagu RFID
 - Usuwanie Tagów RFID
- Monitorowanie oraz raportowanie stanu magazynu
 - Przegląda aktualnych produktów w magazynie
 - Możliwość wyszukiwania produktów (RFID,ID,GUID)
 - Generowanie raportów aktualnego stanu magazynu
- Zarządzanie operacjami magazynowymi
 - Śledzenie historii operacji
- Bezpieczeństwo
 - Autoryzacja i autentykacja poprzez użycie tokenów JWT
 - Logi aktywności – Rejestrowanie aktywności w systemie

Opis wybranego stosu technologicznego

- Backend
 - Język Programowania: C#
 - Framework: ASP.NET Core do tworzenia API, Entity Framework Core do ORM
 - Bogus do sztucznego zapełnienia bazy danych
 - Newtonsoft.Json do konwersji na JSON
- Frontend
 - Swagger
- Baza Danych
 - System zarządzania bazą danych: SQL Server
- Infrastruktura
 - Docker do konteneryzacji aplikacji SQL Server

Wymagania

- .Net w wersji 7.0
- Docker Engine lub Docker Desktop
- Git
- Visual Studio 2022 Community Edition

Opis jak uruchomić aplikację

Kroki:

1. Sklonuj repozytorium projektu:

a. Klonowanie Repozytorium

- i. `git clone https://github.com/Nokijoto/Warehouse_API.git`
- ii. `cd Warehouse_API`

b. Uruchom serwer baz danych za pomocą Docker Compose:

- i. `docker-compose up -d`

2. Konfiguracja połączenia z bazą danych

- Domyślnie aplikacja jest przygotowana dla serwera sql z docker compose , jednakże w przypadku chęci ustawienia własnej bazy ,zmień w pliku appsettings.json w folderze zmień wartość Server na adres swojego serwera SQL Server.

```
"ConnectionStrings": {  
  
    "DefaultConnection": "Server=localhost,1433;Database=  
WarehouseDb;User=sa;Password=Pass@word "  
}
```

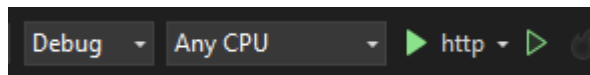
- Następnie uruchom migracje w konsoli mienadzera pakietów:

```
dotnet ef database update
```

3. W celu uruchomienia aplikacji należy wejść do folderu i uruchomić komendę:

```
dotnet run
```

lub uruchomić ją w programie Visual Studio 2022 Community Edition



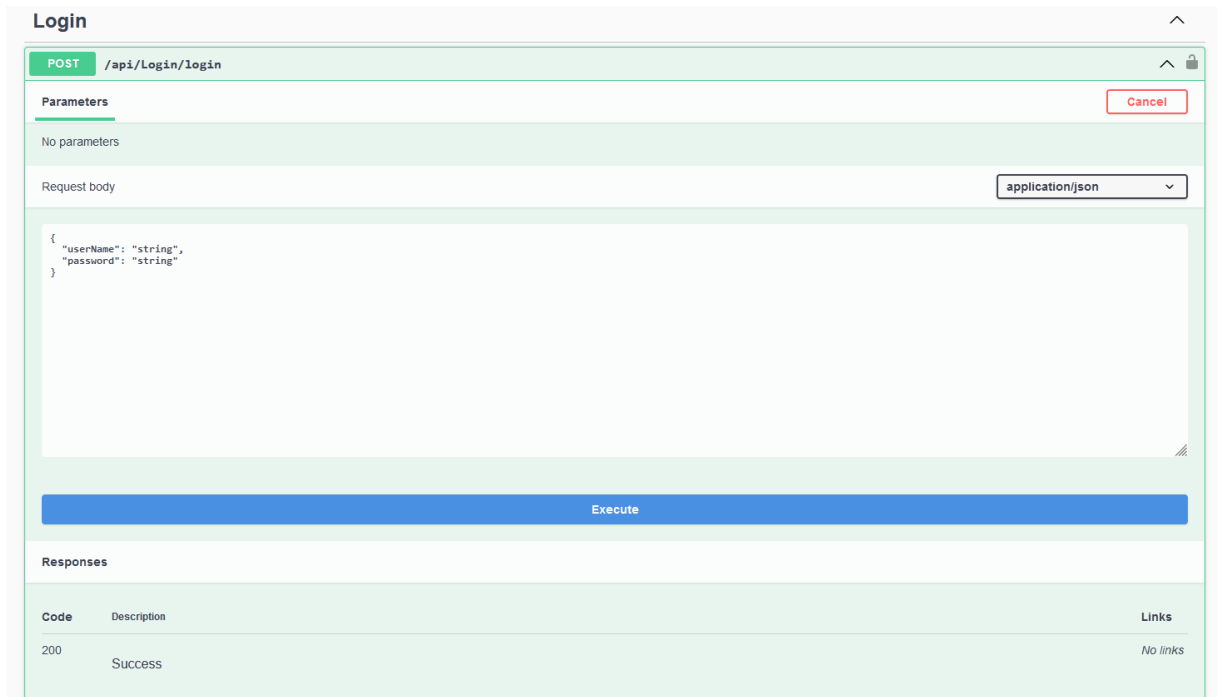
Rys. 1 Uruchomienie Aplikacji

4. Aplikacja będzie dostępna pod adresem `http://localhost:port`, gdzie port to port, na którym działa dana aplikacja. Domyślnie skonfigurowana aplikacja powinna być dostępna pod adresem:

- <http://localhost:5228/swagger/index.html>

5. Logowanie do systemu (login : hasło)

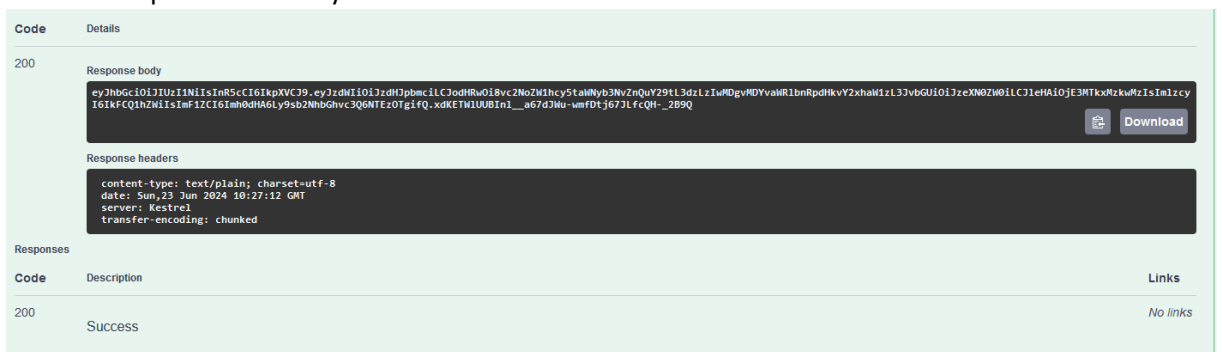
- W celu zalogowania do systemu należy pobrać token JWT poprzez podanie loginu i hasła i wywołanie endpointu `/api/Login/Login` a następnie skopiować otrzymany token



Rys. 2 Endpoint logowania

Dostępne Opcje Logowania

- System: `string : string`
 - Administrator: `admin : admin`
 - Hr: `hr : hr`
 - User: `user : user`
- Poprawnie zautoryzowanie



Rys. 3 Poprawna autoryzacja

- Następnie należy kliknąć przycisk authorize



Rys. 4 Przycisk autoryzacji

6. Autoryzacja poprzez podanie tokenu JWT

Available authorizations ✕

Bearer (http, Bearer)

Please enter a valid token

Value:

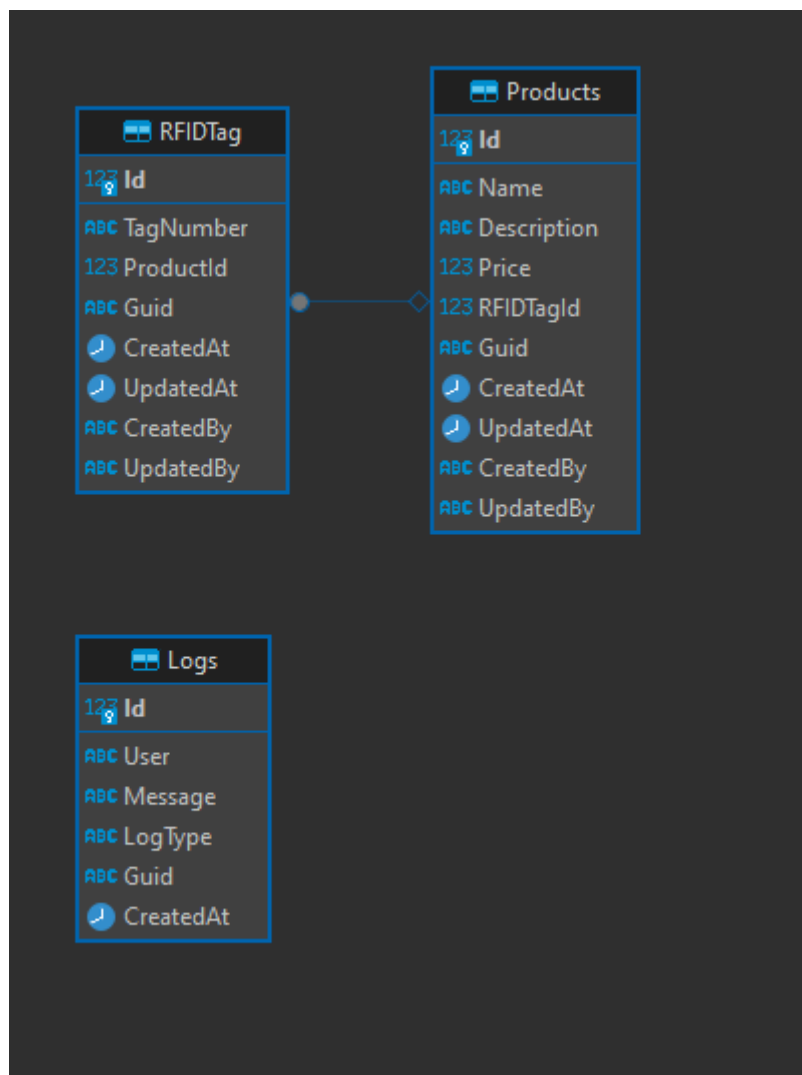
Authorize

Close

Rys. 5 Miejsce do podania tokenu JWT

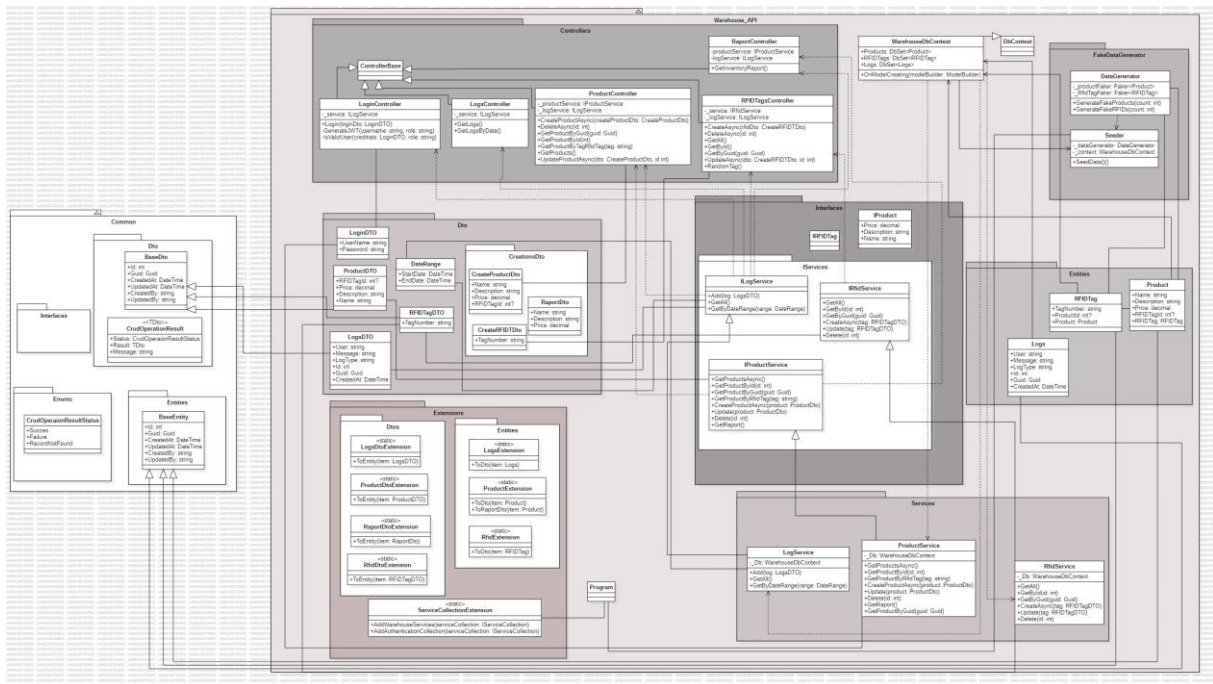
7. Aplikacja jest gotowa do użycia.

Diagram Baz Danych

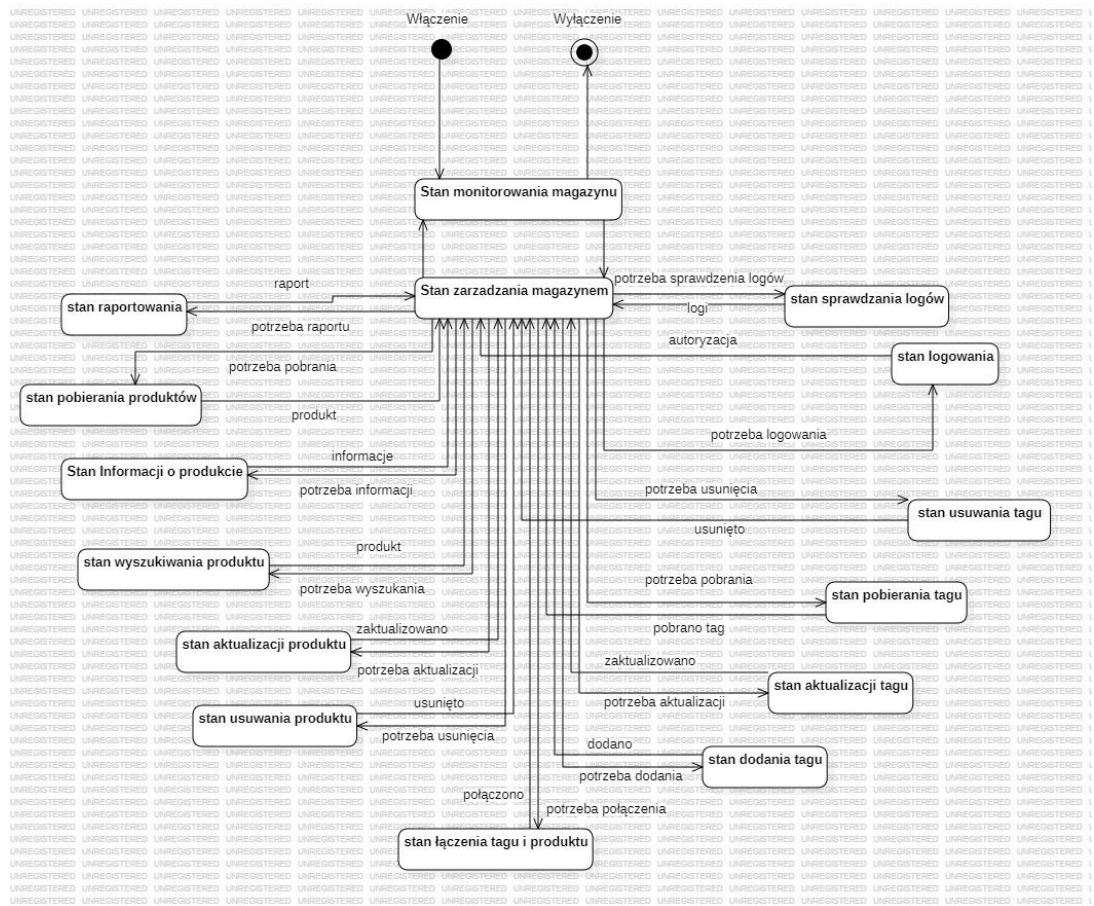


Rys. 6 Diagram bazy danych ERD

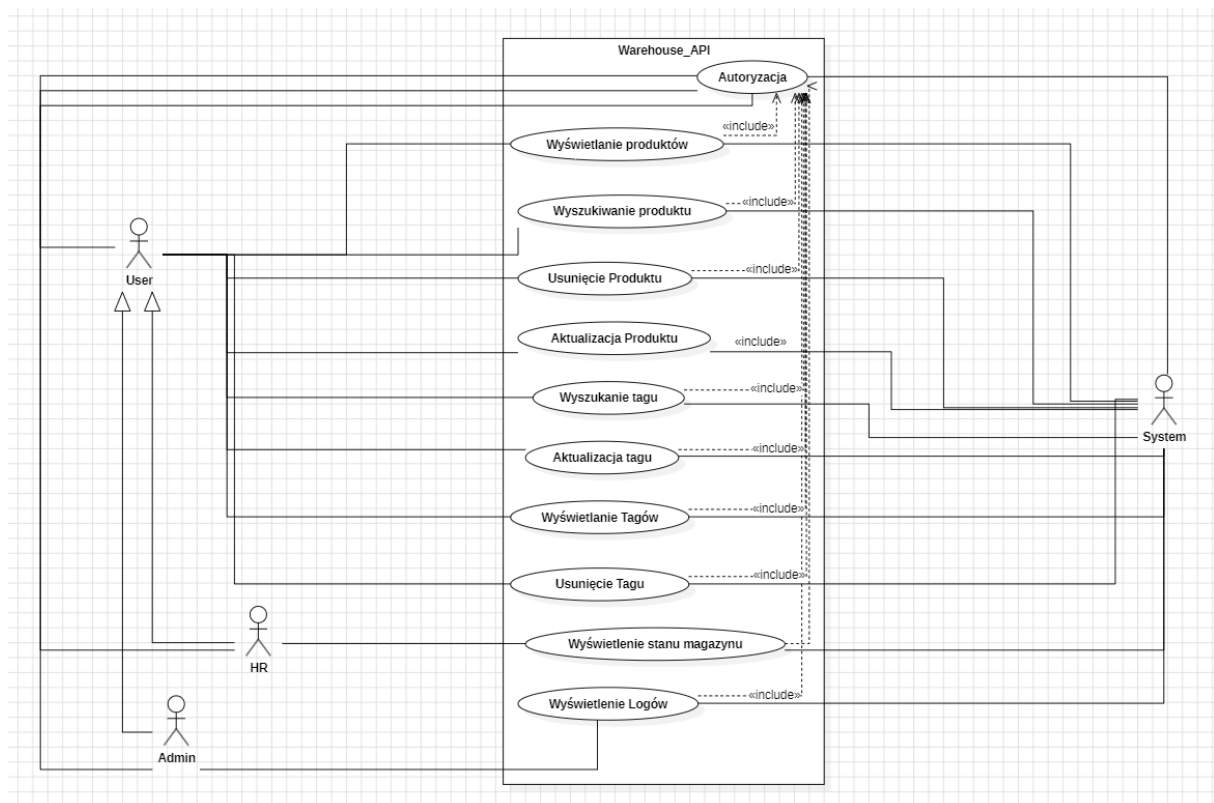
Diagramy



Rys. 7 Diagram Klas



Rys. 8 Diagram Stanów



Rys. 9 Diagram Przypadków Użycia

Podsumowanie i wnioski

Projekt realizowany w ramach zajęć z inżynierii oprogramowania miał na celu stworzenie minimalnie wartościowego projektu, pierwszą fazą było zaprezentowanie wizji systemu, w której należało określić przewidywane funkcjonalności, następnym krokiem było utworzenie 3 diagramów, diagram klas, który przedstawiał strukturę systemu poprzez reprezentacje klas oraz ich połączenia, kolejnym z diagramów był diagram stanów, w którym należało ukazać różne stany systemu, przez które można przechodzić, ostatnim z diagramów był diagram przypadków użycia, który prezentuje interakcje użytkownika podczas korzystania z określonej funkcjonalności. Ostatnią fazą było stworzenie minimalnie wartościowego projektu, który implementował założone funkcjonalności. Podczas realizacji projektu napotkałem problem z poprawnym połączeniem wszystkich elementów diagramów z sobą, przy zwiększającej się ilości klas łatwo było o pomyłkę przy łączeniu oraz podczas śledzenia połączenia. Dodatkowym problemem było odwzorowanie zależności z projektu w diagramach. Dzięki powolnej i dokładnej analizie udało mi się rozwiązać te problemy. Regularne weryfikowanie zgodności między diagramami oraz szczegółowa dokumentacja pozwoliły na skuteczne zarządzanie złożonością systemu i zapewnienie spójności modelu. Projekt zakończył się sukcesem oraz nabyciem nowych umiejętności z zakresu inżynierii oprogramowania, szczególnie w zakresie tworzenia diagramów oraz implementacji funkcjonalności.