Warehouse Management System (WMS) dla Firmy Budowlanej

Dokumentacja Techniczna

Spis treści

1.Autorzy Projektu

- **Adam Jóźwiak**
- **Michał Jaworski**

2.Krótki opis projektu

Warehouse Management System (WMS) dla Firmy Budowlanej to aplikacja umożliwiająca efektywne zarządzanie magazynami materiałów budowlanych. Aplikacja zrealizowana w ramach tego projektu będzie mogła służyć małym i średnim przedsiębiorcom do zarządzania ich magazynami, śledzenia stanów produktów i zapisywania wydań oraz przyjęć na magazyn. Aplikacja będzie zrealizowana w formie aplikacji webowej, przy użyciu .NET 8 oraz języka C# . System pozwali na zarządzanie magazynami, przyjmowanie do nich produktów oraz wydawanie tych produktów do podwykonawców. Aplikacja pozwoli adminowi na wprowadzanie i śledzenie wszystkich wydarzeń związanych z zarządzanymi przez niego magazynami.

3. Instrukcje pierwszego uruchomienia projektu

1. **Pobierz repozytorium**:
    a. Sklonuj repozytorium z systemu kontroli wersji (np. Git) za pomocą polecenia:

```
git clone https://github.com/AdusAdzik/WarehouseManagment.git
```

2. **Skonfiguruj bazę danych**:
    a. Jeśli chcesz korzystać z własnej, niestandardowej bazy otwórz plik `appsettings.json` i zaktualizuj sekcję

b. `Wykonaj migrację update-database`

3. **Przygotowanie migracji**:
    a. Otwórz terminal w katalogu projektu i wykonaj polecenie:
    b. To polecenie utworzy i skonfiguruje schemat bazy danych zgodnie z definicjami w migracjach.
4. **Uruchom projekt**

    a. **Dostęp do aplikacji**:
    b. Zarejestruj nowe konto użytkownika w aplikacji, klikając register w prawym górnym rogu.
    c. Po zalogowaniu uzyskasz dostęp do pełnej funkcjonalności systemu.

4. Opis struktury projektu

Struktura projektu jest zorganizowana zgodnie z konwencjami **ASP.NET MVC**. Główne foldery projektu to:

- **Controllers**: Zawiera klasy kontrolerów odpowiadających za logikę biznesową aplikacji.
- **Models**: Przechowuje klasy modeli danych, w tym encje oraz klasy pomocnicze (np. `ViewModel`).
- **Views**: Zawiera pliki widoków (`.cshtml`) do renderowania interfejsu użytkownika.
- **wwwroot**: Katalog publiczny, w którym znajdują się zasoby statyczne, takie jak pliki CSS, JS i obrazy.
- **Migrations**: Przechowuje pliki migracji dla Entity Framework Core, które definiują zmiany w schemacie bazy danych.
- **Services**: Przechowuje dodatkowe pliki pomocnicze, w naszym przypadku helper do zarządzania sesją koszyka

5.Główne Funkcjonalności

1. Moduł Zarządzania Magazynami

- Magazyny są przypisywane do konkretnych użytkowników (zalogowany użytkownik widzi swoje magazyny) i umożliwiają:
- **CRUD** operacje na magazynach.
- Śledzenie logów operacji na magazynach.

- Obsługę stanów magazynowych produktów.
- Przyjmowanie i wydawanie produktów
- Rejestrowanie wydarzeń takich jak przyjęcie i wydanie produktów.

2. Moduł Zarządzania Produktami

System obsługuje produkty i pozwala na:

- Produkty są widoczne globalnie dla wszystkich użytkowników, można wyświetlić produkty bez zalogowania.
- Dodawanie, edytowanie i usuwanie produktów (soft delete), możliwe dla zalogowanych użytkowników.
- Śledzenie logów operacji na produktach.

3. Moduł Podwykonawców (Subcontractors)

Każdy Podwykonawca jest przypisany do użytkownika i umożliwia:

- **CRUD** operacje na podwykonawcach.
- Rejestrowanie wydarzeń związanych z magazynami i podwykonawców.
- Przeglądanie historii wydarzeń związanych z konkretnym podwykonawcą.

4. Logowanie Operacji

Każda operacja (tworzenie, edycja, usuwanie) na magazynach, produktach i podwykonawcach jest logowana, co pozwala na pełną transparentność działań.

6. Najciekawsze funkcjonalności:

1. Przyjmowanie produktów na magazyn

Opis procesu:

1. Użytkownik wybiera magazyn, do którego chce dodać produkty.
2. Klika przycisk “Receive” i wyświetlana jest lista dostępnych produktów.
3. Użytkownik dodaje produkty do koszyka, określając ilość.
4. Po zatwierdzeniu produkty są:
    a. Dodawane do stanu magazynowego (`WarehouseInventory`).

b. Tworzony jest odpowiedni `WarehouseEvent` z typem "Receive".

Walidacja:

- Ilość dodawana do koszyka nie może być mniejsza niż 0.
- Produkty muszą istnieć w systemie.

2. Wyświetlanie stanów magazynowych

Opis procesu:

1. Po kliknięciu “Inventory” dla wybranego magazynu wyświetlane są wszystkie produkty i ich ilości.
2. Użytkownik może zobaczyć historię wydarzeń związanych z magazynem, w tym szczegóły takie jak:
    a. Typ wydarzenia.
    b. Ilość produktów.
    c. Użytkownik wykonujący operację.
    d. Przyjęty produkt w określonej ilości.
    e. Wydany produkt danemu podwykonawcy w określonej ilości.
    f. Datę wydarzenia

3. Wydawanie produktów do podwykonawcy

Opis procesu:

1. Użytkownik przyciskiem “Inventory” wybiera magazyn, z którego chce wydać produkty.
2. Wyświetlane są produkty dostępne w magazynie (stan magazynowy).
3. Użytkownik dodaje produkty do koszyka, określając ilość.
4. Użytkownik wybiera podwykonawcę, do którego produkty mają zostać wydane.
5. Po zatwierdzeniu:
    a. Ilość produktów w magazynie (`WarehouseInventory`) jest aktualizowana.
    b. Tworzony jest `WarehouseEvent` z typem "Issue".

Walidacja:

- Ilość wydawana nie może przekraczać ilości dostępnej w magazynie.
- Podwykonawca musi istnieć w systemie i należeć do danego użytkownika.

7. Modele:

1. Model `Warehouse`

Model reprezentuje magazyn w systemie.

Pola:

- **`Id`**: Klucz główny magazynu.
- **`Name`**: Nazwa magazynu.
  - **Walidacja**:
    - Pole wymagane.
    - Maksymalna długość: 100 znaków.
- **`Address`**: Adres magazynu.
  - **Walidacja**:
    - Maksymalna długość: 200 znaków.
- **`City`**: Miasto, w którym znajduje się magazyn.
  - **Walidacja**:
    - Pole wymagane.
- **`Country`**: Kraj, w którym znajduje się magazyn.
  - **Walidacja**:
    - Pole wymagane.
- **`PostalCode`**: Kod pocztowy.
  - **Walidacja**:
    - Maksymalna długość: 10 znaków.
- **`Email`**: Adres e-mail.
  - **Walidacja**:
    - Poprawny format adresu e-mail.
- **`Phone`**: Numer telefonu.
  - **Walidacja**:
    - Poprawny format numeru telefonu.
- **`Capacity`**: Pojemność magazynu.
  - **Walidacja**:
    - Liczba całkowita większa lub równa zero.
- **`IsActive`**: Informacja, czy magazyn jest aktywny.
- **`UserId`**: Identyfikator użytkownika przypisanego do magazynu.

Relacje:

- **`WarehouseLogs`**: Logi operacji związanych z magazynem.
- **`WarehouseInventories`**: Produkty znajdujące się w magazynie.
- **`WarehouseEvents`**: Wydarzenia magazynowe (np. przyjęcia, wydania).

2. Model `WarehouseLog`

Reprezentuje logi operacji wykonywanych na magazynie.

Pola:

- **`Id`**: Klucz główny logu.
- **`WarehouseId`**: Identyfikator magazynu, do którego log się odnosi.
- **`Action`**: Akcja wykonana na magazynie (np. "Create", "Edit", "Delete").
- **`Changes`**: Zmiany dokonane w magazynie (w formacie JSON).
- **`Timestamp`**: Data i czas wykonania akcji.
- **`UserId`**: Identyfikator użytkownika, który wykonał akcję.

Relacje:

- **`Warehouse`**: Nawigacja do magazynu, którego dotyczy log.
- **`User`**: Użytkownik, który wykonał akcję.

3. Model `WarehouseEvent`

Model przechowuje szczegóły wydarzeń magazynowych, takich jak przyjęcia i wydania produktów.

Pola:

- **`Id`**: Klucz główny wydarzenia.
- **`WarehouseId`**: Identyfikator magazynu, gdzie wydarzenie miało miejsce.
- **`ProductId`**: Identyfikator produktu uczestniczącego w wydarzeniu.
- **`Quantity`**: Ilość produktów w wydarzeniu.
- **`EventType`**: Typ wydarzenia (np. "Receive", "Issue").
- **`EventDate`**: Data i czas wydarzenia (domyślnie: `DateTime.UtcNow`).
- **`UserId`**: Użytkownik, który zainicjował wydarzenie.
- **`WarehouseInventoryId`**: Opcjonalne powiązanie z zapasami magazynowymi.
- **`SubcontractorId`**: Identyfikator podwykonawcy, jeśli dotyczy wydarzenia.

Relacje:

- **`Warehouse`**: Magazyn, w którym odbyło się wydarzenie.
- **`Product`**: Produkt uczestniczący w wydarzeniu.
- **`WarehouseInventory`**: Powiązane zapasy magazynowe.
- **`Subcontractor`**: Powiązany podwykonawca, jeśli dotyczy.

4. Model `WarehouseInventory`

Model reprezentuje ilości produktów w magazynie.

Pola:

- **`Id`**: Klucz główny zapasu.
- **`WarehouseId`**: Identyfikator magazynu.
- **`ProductId`**: Identyfikator produktu.
- **`Quantity`**: Ilość produktu w magazynie.
    - **Walidacja**:
        - Liczba większa lub równa 0.

Relacje:

- **`Warehouse`**: Magazyn, w którym przechowywany jest produkt.
- **`Product`**: Produkt przechowywany w magazynie.

5.Model Product

Reprezentuje produkt w systemie.

Pola:

- **Id**: Klucz główny produktu.
- **Number**: Unikalny numer produktu.
    - Walidacja: Pole wymagane, maksymalna długość 50 znaków.
- **Name**: Nazwa produktu.
    - Walidacja: Pole wymagane, maksymalna długość 100 znaków.
- **Unit**: Jednostka miary.
    - Walidacja: Pole wymagane, maksymalna długość 10 znaków.
- **Description**: Opis produktu.
    - Walidacja: Maksymalna długość 50 znaków.
- **CubicVolume**: Objętość produktu w metrach sześciennych.
    - Walidacja: Liczba większa lub równa 0.
- **CreatedBy**: Identyfikator użytkownika, który stworzył produkt.
- **CreatedAt**: Data utworzenia produktu.
- **DeletedAt**: Data usunięcia produktu (soft delete).

Relacje:

- **ProductLogs**: Logi zmian w produktach.
- **WarehouseInventories**: Stan magazynowy produktu.
- **WarehouseEvents**: Wydarzenia związane z produktem.

6.Model ProductLog

Rejestruje zmiany w modelu `Product`.

Pola:

- **Id**: Klucz główny logu.
- **ProductId**: Klucz obcy produktu.
- **Action**: Typ akcji (np. "Create", "Edit", "Delete").
- **Changes**: Szczegóły zmian.
- **Timestamp**: Data i czas operacji.
- **UserId**: Identyfikator użytkownika, który wykonał operację.

Relacje:

- **Product**: Produkt, do którego odnosi się log.
- **User**: Użytkownik, który wykonał operację.

7.Model Subcontractor

Reprezentuje podwykonawcę w systemie.

Pola:

- **Id**: Klucz główny podwykonawcy.
- **Name**: Nazwa podwykonawcy.
    - Walidacja: Pole wymagane, maksymalna długość 100 znaków.
- **Email**: Adres e-mail podwykonawcy.
    - Walidacja: Poprawny format e-mail.
- **Phone**: Numer telefonu podwykonawcy.
    - Walidacja: Poprawny format numeru telefonu.
- **Description**: Opis podwykonawcy.
    - Walidacja: Maksymalna długość 500 znaków.
- **UserId**: Identyfikator użytkownika zarządzającego podwykonawcą.

Relacje:

- **SubcontractorLogs**: Logi zmian w podwykonawcy.
- **WarehouseEvents**: Wydarzenia związane z podwykonawcą.

8.Model SubcontractorLog

Rejestruje zmiany w modelu `Subcontractor`.

Pola:

- **Id**: Klucz główny logu.

- **SubcontractorId**: Klucz obcy podwykonawcy.
- **Action**: Typ akcji (np. "Create", "Edit", "Delete").
- **Changes**: Szczegóły zmian.
- **Timestamp**: Data i czas operacji.
- **UserId**: Identyfikator użytkownika, który wykonał operację.

Relacje:

- **Subcontractor**: Podwykonawca, do którego odnosi się log.
- **User**: Użytkownik, który wykonał operację.

9.Model CartItem

Reprezentuje element w koszyku operacji magazynowych.

Pola:

- **WarehouseId**: Klucz obcy magazynu.
- **ProductId**: Klucz obcy produktu.
- **Quantity**: Ilość produktu w koszyku.

10.Model DashboardViewModel

Używany do generowania danych na potrzeby panelu głównego.

Pola:

- **WarehouseCount**: Liczba magazynów.
- **ProductCount**: Liczba produktów.
- **SubcontractorCount**: Liczba podwykonawców.
- **WarehouseEventCount**: Liczba wydarzeń magazynowych.

8.Opis systemu użytkowników

W naszym projekcie każdy użytkownik posiada jedno unikalne konto, które jest zarządzane za pomocą systemu **ASP.NET Identity**, zapewniającego bezpieczne uwierzytelnianie i autoryzację. Użytkownicy niezalogowani mają dostęp do publicznych sekcji systemu, takich jak przeglądanie listy produktów, ale nie mogą wykonywać żadnych operacji CRUD ani przeglądać szczegółów magazynów czy podwykonawców. Po zalogowaniu użytkownicy widzą zasoby przypisane do swojego konta: swoje magazyny, podwykonawców oraz historię działań w formie logów operacji. Każda operacja, taka jak tworzenie, edycja czy usunięcie magazynu, produktu lub podwykonawcy, jest rejestrowana i dostępna w logach, co pozwala użytkownikom na pełną kontrolę nad ich zasobami. Dzięki przypisaniu magazynów i podwykonawców do konkretnych

użytkowników oraz ścisłemu filtrowaniu dostępu, system gwarantuje przejrzystość i ogranicza widoczność danych wyłącznie do właściciela konta.

9. Specyfikacja wykorzystanych technologii

Projekt został zbudowany z użyciem najnowszej stabilnej wersji platformy **.NET 8**. Kluczowe technologie i narzędzia użyte w projekcie to:

- **.NET 8**: Framework do tworzenia aplikacji webowych w architekturze MVC (Model-View-Controller).
- **Entity Framework Core**: ORM (Object-Relational Mapping) do obsługi bazy danych.
- **SQL Server**: Relacyjna baza danych przechowująca dane aplikacji.
- **ASP.NET Identity**: Mechanizm uwierzytelniania i autoryzacji użytkowników.
- **Bootstrap**: Framework CSS do tworzenia responsywnych interfejsów użytkownika.
- **SweetAlert2**: Biblioteka JavaScript do dynamicznych powiadomień i formularzy modalnych.
- **jQuery**: Biblioteka JavaScript do manipulacji DOM oraz obsługi zapytań AJAX używana przy zarządzaniu koszykiem produktów.

```
STRUKTURA PROJEKTU
|  appsettings.Development.json
|  appsettings.json
|  Program.cs
|  README.md
|  WarehouseManagment.csproj
|  WarehouseManagment.csproj.user
|  WarehouseManagment.sln
|
+---Areas
|  \---Identity
|      \---Pages
|              _ViewStart.cshtml
|
+---bin
|  \---Debug
|
+---Controllers
|      HomeController.cs
|      ProductsController.cs
|      SubcontractorsController.cs
|      WarehouseInventoriesController.cs
|      WarehousesController.cs
```

```
|
+---Data
|  |  ApplicationDbContext.cs
|  |
|  \---Migrations
|        [ALL MIGRATION FILES]
|        ApplicationDbContextModelSnapshot.cs
|
+---Models
|     CartItem.cs
|     DashboardViewModel.cs
|     ErrorViewModel.cs
|     Product.cs
|     ProductLog.cs
|     Subcontractor.cs
|     SubcontractorLog.cs
|     Warehouse.cs
|     WarehouseEvent.cs
|     WarehouseInventory.cs
|     WarehouseLog.cs
|
+---obj
|  |
|  \---Debug
|
+---Properties
|     launchSettings.json
|     serviceDependencies.json
|     serviceDependencies.local.json
|     serviceDependencies.local.json.user
|
+---Services
|     SessionExtensions.cs
|
+---Views
|  |  _ViewImports.cshtml
|  |  _ViewStart.cshtml
|  |
|  +---Home
|  |     Index.cshtml
|  |     Privacy.cshtml
|  |
|  +---Products
|  |     Create.cshtml
|  |     Delete.cshtml
|  |     Details.cshtml
|  |     Edit.cshtml
|  |     Index.cshtml
```

```
|  |
|  +---Shared
|  |      Error.cshtml
|  |      _Layout.cshtml
|  |      _Layout.cshtml.css
|  |      _LoginPartial.cshtml
|  |      _LogListPartial.cshtml
|  |      _ValidationScriptsPartial.cshtml
|  |
|  +---Subcontractors
|  |      Create.cshtml
|  |      Delete.cshtml
|  |      Details.cshtml
|  |      Edit.cshtml
|  |      Index.cshtml
|  |
|  +---WarehouseInventories
|  |      CartSummary.cshtml
|  |      ProductsForWarehouse.cshtml
|  |
|  \---Warehouses
|         Create.cshtml
|         Delete.cshtml
|         Details.cshtml
|         Edit.cshtml
|         Index.cshtml
|         Inventory.cshtml
|
\---wwwroot
    |  favicon.ico
    |
    +---css
    |      sidebar.css
    |      site.css
    |
    +---js
    |      site.js
    |
    \---lib
        +---bootstrap
        +---jquery
        +---jquery-validation
        +---jquery-validation-unobtrusive
        ---sweetalert
```