## 1 Tytuł projektu

* Construction management
* “Construction management” to system zarządzania projektami budowlanymi, umożliwiający tworzenie, organizację i monitorowanie projektów, zadań zespołów oraz budżetów w celu efektywnego zarządzania procesami budowlanymi

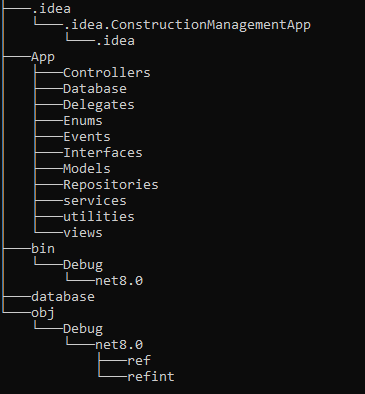
## 2 Opis projektu

* Cel projektu: Stworzenie kompleksowego narzędzia do zarządzania projektami budowlanymi, narzędzia umożliwiającego efektowne planowanie i komunikację między zespołami oraz klientami
* Aplikacja umożliwia zarządzanie użytkownikami, zespołami, projektami, zadaniami, sprzętem, budżetem, materiałami budowlanymi, pozwala wysyłać zgłoszenia, raporty, obsługuje wiadomości między użytkownikami,
* Aplikacja jest przeznaczona dla przedsiębiorstw budowlanych, które chcą posiadać elektroniczny system zarządzania swoimi pracownikami, projektami

## 3 Technologie jakich wykorzystano do projektu

* Język użyty do implementacji aplikacji: C# 12.0
* Środowisko: .NET 8.0
* IDE: VS code oraz VS studio 2022
* Entity Framework Core: ORM do zarządzania bazą danych za pomocą klas

## 4 Struktura katalogów



App – zawiera wszystkie pod foldery związane z aplikacją

* Controllers – logika kontrolerów, zarządzająca nimi
* Database – w tym folderze znajduje się kontekst do bazy danych
* Delegates – delegaty używane w aplikacji
* Enums – typy wyliczeniowe, które pomagają w zarządzaniu aplikacją są tam np. Role użytkowników oraz ich uprawnienia w aplikacji
* Events - obsługa zdarzeń w aplikacji
* Interfaces – Interfejsy używane w aplikacji
* Models – Głównie modele odzwierciedlające strukturę w bazie danych
* Repositories – Klasy odpowiadzialne za operacje na obiektach w bazie danych
* Services - zawiera klasy odpowiedzialne za logikę biznesową i operacje wspierające, takie jak uwierzytelnianie, kontrola dostępu (RBAC) oraz inne usługi pomocnicze używane w aplikacji.
* Utilities – zawiera klasę hashującą hasła oraz klasę odpowiedzialną za suskrybcję zdarzeń
* Views – zawiera klasy odpowiadające za prezentację aplikację użytkownikowi

## 5 Instrukcja uruchomienia:

* [instalacja krok po kroku](https://github.com/cool-white-leaders-gang/ConstructionManagementApp/blob/main/README.md)

## 6 Opis działania aplikacji:

Funkcjonalności:

1. **Zarządzanie projektami**:
   1. Tworzenie, edytowanie i usuwanie projektów.
   2. Przypisywanie zespołów i budżetów do projektów.
   3. Monitorowanie postępów realizacji projektów.
2. **Zarządzanie zadaniami**:
   1. Dodawanie, edytowanie i usuwanie zadań w ramach projektów.
   2. Przypisywanie priorytetów i statusów zadań (np. "Nowe", "W trakcie", "Ukończone").
   3. Przypisywanie użytkowników do zadań oraz usuwanie ich z przypisań.
   4. Wyświetlanie listy zadań przypisanych do użytkownika lub projektu.
3. **Zarządzanie użytkownikami i uprawnieniami**:
   1. Obsługa logowania i sesji użytkowników.
   2. Kontrola dostępu oparta na rolach (RBAC), np. menedżerowie projektów mogą zarządzać zadaniami, a pracownicy mogą jedynie realizować przypisane zadania.
4. **Zarządzanie budżetami**:
   1. Tworzenie, edytowanie i usuwanie budżetów.
   2. Monitorowanie wydatków w ramach budżetów projektów.
5. **Logowanie zdarzeń**:
   1. Rejestrowanie kluczowych działań, takich jak dodawanie, edytowanie czy usuwanie obiektów w bazie danych, w celu zapewnienia przejrzystości działań użytkowników.
6. **Interfejs użytkownika**:
   1. Aplikacja zawierać widoki umożliwiające użytkownikom interakcję z systemem, takie jak przeglądanie projektów, zadań i raportów.

## 7 Wygląd aplikacji:

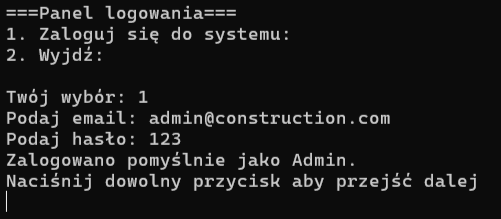
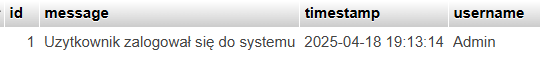
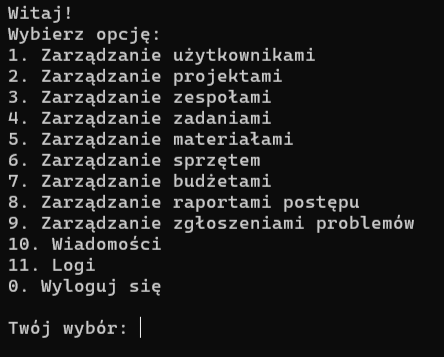
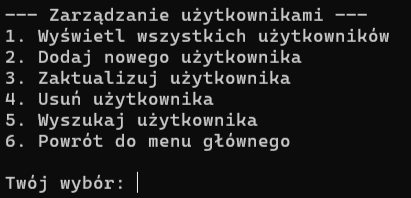
Ekran logowania:  


Tabela logów po zalogowaniu użytkownika

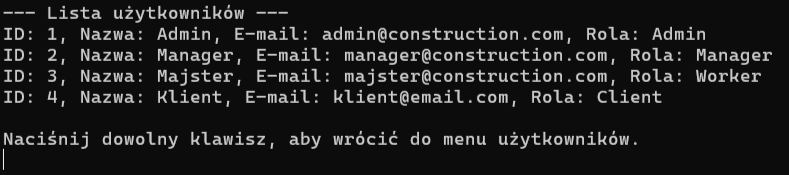


Ekran główny aplikacji po zalogowaniu



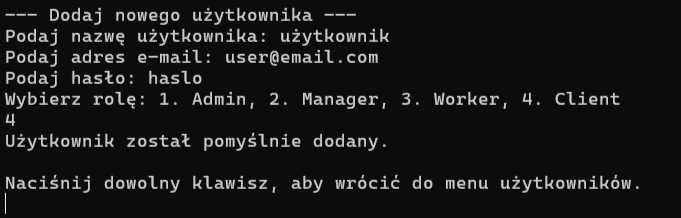
Przykładowy ekran zarządzania   
  


Przykładowy ekran wyświetlania zawartości tabeli



Ekrany operacji CRUD

Dodawanie

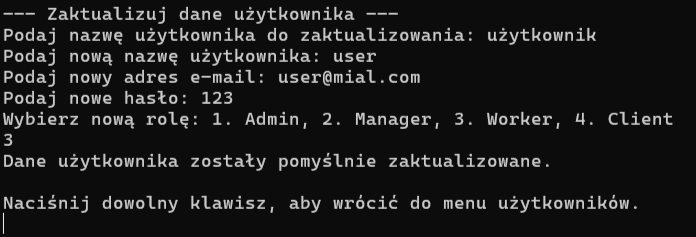




Log po dodaniu użytkownika



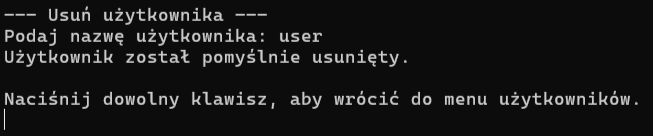
Aktualizacja

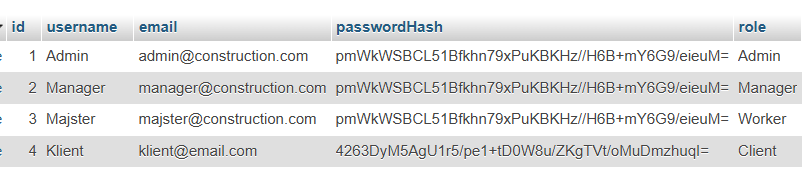


 Log po aktualizacji



Usuwanie

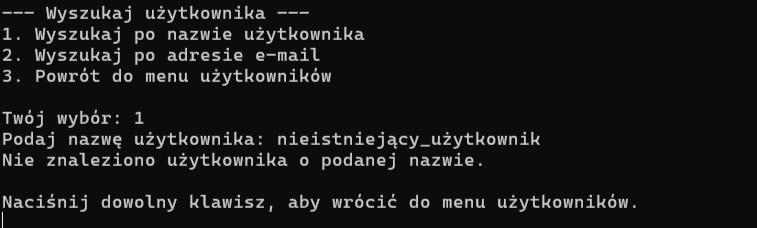


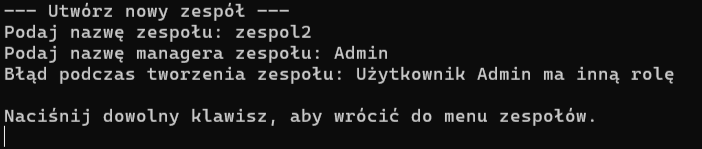


log po usunięciu użytkownika



Przykładowy ekran po wpisaniu nieprawidłowych lub nieistniejacyh danych

Przykładowy ekran po wpisaniu danych niezgodnych z logiką aplikacji (zespołami może zarządzać użytkownik, który ma rolę managera)



## 8 Jak korzystać z aplikacji

Aby sterować menu należy wpisywać liczby odpowiadające danym funkcjom w aplikacji, po wpisaniu liczby nacisnąć “enter”, aplikacja instruuje użytkownika jak się po niej poruszać. Staraliśmy się, żeby UI aplikacji było jak najbardziej intuicyjne.

## 9 Struktury danych i klasy

AppDbContext

* Klasa odpowiedzialna za zarządzanie połączeniem z bazą danych.
* Definiuje zestawy danych (DbSet), takie jak Users, Tasks, Projects, Budgets, itp., które reprezentują tabele w bazie danych.
* Umożliwia wykonywanie operacji CRUD na danych oraz konfigurację modelu danych za pomocą metody OnModelCreating.

AuthenticationService

* Klasa obsługująca logowanie i zarządzanie sesjami użytkowników.
* Metody:
  + Login: Weryfikuje dane logowania użytkownika.
  + Logout: Kończy sesję użytkownika.
  + GetCurrentUser: Zwraca aktualnie zalogowanego użytkownika.
  + Wykorzystuje UserRepository do pobierania danych użytkownika i weryfikacji hasła.

LogService

* Klasa odpowiedzialna za rejestrowanie zdarzeń w systemie.
* Metody:
  + Zapisuje logi do bazy danych (za pomocą tabeli Logs w AppDbContext).
  + Używana do monitorowania działań użytkowników w aplikacji.

RBACService

* Klasa implementująca kontrolę dostępu opartą na rolach (Role-Based Access Control).
* Metody:
  + HasPermission: Sprawdza, czy użytkownik ma określone uprawnienia.
  + IsProjectManager: Weryfikuje, czy użytkownik jest menedżerem projektu.
  + IsWorkerInProjectTeam: Sprawdza, czy użytkownik należy do zespołu projektu.
  + GetProjectsForUserOrManagedBy: Zwraca listę projektów przypisanych do użytkownika lub zarządzanych przez niego.
  + Wspiera bezpieczeństwo aplikacji, ograniczając dostęp do funkcji na podstawie ról użytkowników.

Klasy widoków (Views)

* Odpowiadają za prezentację danych użytkownikowi.
* Przykłady:
* Widoki listy projektów, zadań, użytkowników.
* Formularze do tworzenia i edycji danych.
* Metody:
  + ShowView: Wyświetla menu i obsługuje wybory użytkowników
  + DisplayAll... : Wyświetla wszystko co jest w danej tabeli odpowiadającej widokowi
  + Create... : Tworzy nowy obiekt zawireający wpisane przez użytkownika dane i wpisuje go do bazy danych
  + Delete... : Usuwa obiekt z bazy danych o podanym ID lub nazwie (zależnie od klasy)
  + Update... : Aktualizuje obiekt, który użytkownik wybiera po nazwie lub ID (zależnie od klasy)
  + ReturnToMenu: Po wywołaniu metody użytkownik wraca do menu głównego

Klasy repozytoriów (Repositories):

* Odpowiadają za dostęp do danych w bazie danych.
* Implementują operacje CRUD (Create, Read, Update, Delete).
* Metody:
  + Create... : Przyjmują obiekt danej klasy i wpisują go do bazy danych
  + Update... : Przyjmują zaktualizowany obiekt danej klasy i aktualizują go również w bazie
  + Delete... : Przyjmują id lub nazwę (zależnie od klasy) obiektu danej klasy po czym usuwa go z tabeli
  + Get..ById: Przyjmuje ID danego obiektu w tabeli i zwraca cały obiekt o tym id
  + Get..ByName: Przyjmuje nazwę danego obiektu w tabeli i zwraca cały obiekt o tej nazwie
  + GetAll... : Zwraca listę wszystkich obiektów z danej tabeli

Klasy Kontrolerów (Controllers):

* Obsługują logikę aplikacji i interakcje użytkownika.
* Wywołują metody repozytoriów i serwisów w odpowiedzi na akcje użytkownika.
* Mają zaimplementowaną logikę walidacji danych
* Metody:
  + Ma podobne metody co Repositories, tylko jest w nich obsuga błędów, oraz wykorzystanie konstruktorów z Models

Klasy modeli (Models)

* Reprezentują dane w bazie danch połączonej z aplikacją.
* Zawieraja konstruktory wykorzystywane przez kontrolery
* Metody:
  + ToString: Zwraca obiekt jako łańcuch znaków

## 10 Obsługa błędów

Przewidziane błędy:

KeyNotFoundException: Wyjątek który jest wyrzucany wtedy kiedy format danych jest poprawny, ale podana wartość nie istnieje w tabeli



UnauthorizedAccessException: Wyjątek wyrzucany, kiedy użytkownik niemający uprawnień , próbuje przejść do którejś z funkcjonalności aplikacji

InvalidOperationException: Wyjątek zostanie wyrzucony w sytuacji np. , gdy użytkownik aktualnie zalogowany w systemie nie jest przypisany do żadnych projektów, a chce wyświetlić listę projektów do których jest przypisany lub kiedy próbuje wejść do aplikacji kiedy nie można połączyć się z bazą.



ArgumentNullException: Wyjątek ten zostanie wyrzucony, np. jeśli ktoś spróbuje utworzyć obiekt LogService, przekazując null jako argument do konstruktora.





ArgumentException: Jest wyrzucany kiedy użytkownik próbuje wpisać dane o nieprawidłowym formacie 

## 11 Testowanie

Aplikacja została przetestowana przy użyciu testów jednostkowych z wykorzystaniem frameworka xUnit oraz biblioteki Moq.

Testy jednostkowe obejmują kluczowe funkcjonalności aplikacji, takie jak:

**AuthenticationService**:

Logowanie:

Sprawdzanie poprawności logowania przy prawidłowych danych.

Walidacja zachowania, gdy użytkownik nie istnieje lub hasło jest nieprawidłowe.

Wylogowanie:

Weryfikacja, czy sesja użytkownika jest poprawnie czyszczona.

Pobieranie aktualnie zalogowanego użytkownika:

Sprawdzanie, czy zwracany jest użytkownik w sesji lub null, jeśli nikt nie jest zalogowany.

**UserRepository**:

Tworzenie użytkownika:

Weryfikacja dodania użytkownika do bazy danych.

Sprawdzanie, czy zgłaszany jest wyjątek w przypadku próby utworzenia użytkownika z istniejącym adresem e-mail.

Pobieranie użytkownika po ID:

Sprawdzanie, czy metoda zwraca poprawnego użytkownika, jeśli istnieje w bazie danych.

**Do testów wykorzystano**:

xUnit jako framework do pisania testów jednostkowych.

Moq do tworzenia mocków w celu izolacji testowanych komponentów.

Mockowane obiekty bazy danych (AppDbContext).

Testy pokrywają najważniejsze elementy systemu, zapewniając poprawność logiki aplikacji.

Klasy były też testowane na bieżąco, bezpośrednio w program.cs bez użycia żadnych bibliotek

## 12 Problemy i organiczenia

Wydaje nam się, że aplikacja działa bardzo dobrze, program nie wywala się podczas korzystania z niego

Co można poprawić:

* Na pewno można poprawić aktualizację, że jak użytkownik chciałby zmienić tylko email, to musi wpisywać od nowa wszystkie właśćiwości, a nie tylko ten email
* Strukture kodu – Mimo naszych największych chęci, to w kodzie łatwo się pogubić

## 13 Plany rozwoju

* W przyszłości można dodać bardziej przyjazne dla oka UI
* Aby baza danych nie była tylko lokalna

## 14 Autorzy

* Filip Lubka: [github](https://github.com/lubkaf)
* Patryk Stafecki: [github](https://github.com/stafecki)