# APT-Market

## Dokumentacja techniczna projektu

**Autorzy projektu:**
**Jakub Stroiński**
**Szymon Szymkowiak**

# Spis treści

# 1. Opis działania projektu

## Krótki ogólny opis działania

Witryna pozwala wystawiać ogłoszenia na wynajem mieszkania przez wynajmujących. Mają oni również dostęp do paneli, w których widzą zgłoszenia serwisowe od najemców, mogą sprawdzać terminy płatności za swoje lokale oraz wysyłać powiadomienia dla najemców. Mogą również edytować umowę najmu oraz generować raporty z ogłoszeń. Najemcy natomiast mają możliwość przeglądania ogłoszeń, wynajmowania przez stronę, zgłaszania zleceń serwisowych dla wynajmowanych lokali oraz sprawdzać terminy płatności za wynajęte apartamenty.

## Zastosowana baza danych

Witryna używa bazy MS SQL i przechowuje dane na temat użytkowników, apartamentów, zgłoszeń serwisowych, płatności oraz umów najmu.

## Rejestracja i logowanie użytkowników

Witryna pozwala utworzyć konto użytkownika. Każdy użytkownik może zdecydować czy rejestruje się jako najemca lub wynajmujący. Podczas rejestracji użytkownik wpisuje dane do formularza takie jak: imię, nazwisko, numer telefonu, adres email oraz hasło (opcjonalnie NIP firmy). Wynajmujący może dodawać ogłoszenia na swoje apartamenty, obsługiwać zlecenia serwisowe, zarządzać płatnościami i wynajmami. Najemca może przeglądać wystawione przez najemców ogłoszenia. Istnieje również opcja wylogowania się z witryny.

## Role użytkowników

W aplikacji użytkownik może mieć jedną z trzech ról:

1. **Administrator**: posiada dodatkowe możliwości edycji użytkowników, usuwania ich z systemu oraz tworzenia nowych użytkowników. Ma również wgląd we wszystkie ogłoszenia najemców.
2. **Wynajmujący**: ma dostęp do widoków wystawiania nowych ogłoszeń, panel do zarządzania zgłoszeniami serwisowymi oraz ma wgląd w terminy płatności za dane lokale i może wysyłać powiadomienia.
3. **Najemca**: ma dostęp do przeglądu ogłoszeń oraz do panelu, w którym może podejrzeć swoje wynajmy i terminy ich opłacenia. Ma również widok na utworzenie zlecenia serwisowego.

## Podstrony publiczne i prywatne

1. **Podstrony publiczne**: są to podstrony dostępne dla każdego użytkownika takie jak: strona główna, przeglądanie ogłoszeń czy strona z rejestracją.
2. **Podstrony prywatne**: widoczne są tylko dla zalogowanych użytkowników w zależności od ich ról.

# 2. Specyfikacja wykorzystanych technologii

- **Backend**: C# .NET 8 – to najnowsza wersja platformy .NET, która umożliwia tworzenie nowoczesnych, skalowalnych aplikacji wieloplatformowych. Platforma wspiera programowanie obiektowe, reaktywne oraz asynchroniczne, co pozwala na wydajną obsługę aplikacji klient-serwer. .NET 8 integruje najnowsze rozwiązania technologiczne, takie jak lepsza obsługa minimalnych API i wydajniejsze podejście do przetwarzania danych.

- **Framework**: ASP.NET Core MVC - to framework oparty na wzorcu projektowym Model-View-Controller (MVC), który rozdziela logikę aplikacji, dane oraz interfejs użytkownika. Dzięki temu kod jest bardziej przejrzysty, łatwiejszy w utrzymaniu i skalowalny.

- **Baza danych**: MS SQL Server - Zaawansowany system zarządzania relacyjnymi bazami danych (RDBMS) stworzony przez Microsoft. Oferuje szeroki zakres funkcji, takich jak transakcje, widoki, procedury składowane i pełne wsparcie dla ORM, np. Entity Framework.

- **ORM**: Entity Framework Core - to nowoczesny, wieloplatformowy ORM (Object-Relational Mapper), który umożliwia interakcję z bazą danych przy użyciu obiektów w języku C#. EF Core automatycznie mapuje klasy na tabele w bazie danych i generuje zapytania SQL.

- **Autoryzacja**: ASP.NET Identity do zarządzania użytkownikami i rolami - ASP.NET Identity to framework do zarządzania uwierzytelnianiem użytkowników oraz rolami w aplikacjach ASP.NET. Umożliwia rejestrację, logowanie, zarządzanie hasłami i przydzielanie ról użytkownikom.

- **Razor Pages** - Razor Pages to technologia ASP.NET Core, która pozwala na tworzenie dynamicznych widoków za pomocą języka Razor. Pozwala na łączenie logiki C# z HTML w intuicyjny sposób, co ułatwia tworzenie wydajnych i czytelnych interfejsów użytkownika.

# 3. Instrukcje pierwszego uruchomienia

1. Sklonuj repozytorium projektu.
2. Otwórz projekt w środowisku Rider (lub Visual Studio).
3. Wybierz bazę danych:
    - Jeśli używasz MS SQL Server, skonfiguruj połączenie w appsettings.json.
    - Jeśli preferujesz SQLite, ustaw odpowiednią konfigurację i wskaż ścieżkę bazy danych.
4. Zainicjuj bazę danych, wykonując migrację poprzez wykonanie polecenia "*Update-Database* " w konsoli menadżera pakietów lub wybierając ją przy pomocy panelu do zarządzania narzędziami
5. Uruchom aplikację, aby rozpocząć korzystanie z systemu.

# 4. Opis struktury projektu

## Controllers – Kontrolery zarządzające logiką aplikacji

Kontrolery w projekcie obsługują logikę związaną z żądaniami HTTP, przekierowując je do odpowiednich usług oraz widoków. Każdy kontroler odpowiada za jeden obszar funkcjonalny, np. zarządzanie nieruchomościami czy umowami najmu.
W projekcie znajdą się kontrolery takie jak:

- **PropertyController**: Zarządza operacjami CRUD dla nieruchomości.
- **TenantController**: Obsługuje operacje związane z najemcami.
- **RentalAgreementController**: Obsługuje tworzenie i zarządzanie umowami najmu.
- **MaintenanceRequestController**: Zarządza zgłoszeniami serwisowymi.
- **PaymentController**: Obsługuje płatności najemców.

Wszystkie kontrolery będą miały wbudowaną walidację wejściową oraz obsługę wyjątków, co pozwala zapewnić bezpieczeństwo i spójność aplikacji.

## Models – Modele danych reprezentujące struktury tabel w bazie

Modele definiują strukturę danych przechowywanych w bazie. Wykorzystywany jest Entity Framework Core, dzięki czemu każdy model jest automatycznie odwzorowywany na odpowiednią tabelę w bazie danych.
Każdy model zawiera:

- **Atrybuty walidacyjne**: np. [Required], [StringLength], [Range] itp.
- **Relacje**: np. Property może mieć wiele powiązanych umów RentalAgreement (relacja jeden-do-wielu).

# Views – Widoki odpowiadające za interfejs użytkownika

Widoki w projekcie korzystają z technologii Razor Pages, która umożliwia dynamiczne generowanie HTML na podstawie danych z kontrolerów.
Każdy widok jest powiązany z określonym kontrolerem i akcją, np. widok Index.cshtml w kontrolerze PropertyController odpowiada za listę nieruchomości.
Widoki są odpowiedzialne za:

- Prezentację danych użytkownikowi.
- Generowanie formularzy (np. do tworzenia lub edycji nieruchomości).

# Services – Klasy usług odpowiedzialne za logikę biznesową

Usługi oddzielają logikę biznesową od kontrolerów, co ułatwia utrzymanie i testowanie aplikacji. W projekcie znajdą się usługi takie jak:

- **PropertyService**: Zarządza operacjami związanymi z nieruchomościami (np. filtrowanie po lokalizacji lub cenie).
- **TenantService**: Obsługuje logikę związaną z najemcami, np. sprawdzanie czy najemca ma aktywną umowę.

# Data – Pliki konfiguracji bazy danych oraz zarządzania użytkownikami

Folder Data przechowuje:

- **ApplicationDbContext**: Klasa kontekstu bazy danych, która definiuje tabele i relacje między nimi.
- **Migrations**: Automatycznie generowane migracje bazy danych.
- **Seed Data**: Dane inicjalne, które są ładowane do bazy przy pierwszym uruchomieniu aplikacji.

# 5. Wylistowane wszystkie modele

## Property

Reprezentuje nieruchomość, która utworzona wcześniej przez wynajmujących lub administratora jest widoczna na liście nieruchomości. Za każdą nieruchomość można ustalić cenę, podać dane kontaktowe oraz adres danej nieruchomości.

- **Id**: Identyfikator nieruchomości.
- **Address**: Adres nieruchomości, wymagane.
- **RentPrice**: Cena najmu, wymagane, wartość dodatnia.
- **Size**: Rozmiar nieruchomości (w metrach kwadratowych), wartość dodatnia.

## Tenant

Reprezentuje najemcę. Każdy najemca musi podać swoje dane osobiste, numer telefonu oraz email kontaktowy, na który będzie otrzymywał powiadomienia.

- **Id**: Identyfikator najemcy.
- **FullName**: Imię i nazwisko najemcy, wymagane.
- **ContactNumber**: Numer kontaktowy, wymagany, format walidowany.
- **Email**: Adres e-mail, wymagany, format walidowany.

## RentalAgreement

Reprezentuje umowę najmu. Posiada wszystkie atrybuty, które ma *Property* oraz zawiera daty rozpoczęcia oraz zakończenia najmu. Ma również pole, w którym można odczytać cenę wysokość czynszu.

- **Id**: Identyfikator umowy.
- **PropertyId**: Identyfikator nieruchomości, wymagane.
- **TenantId**: Identyfikator najemcy, wymagane.
- **StartDate**: Data rozpoczęcia, wymagane, nie może być wcześniejsza od bieżącej.
- **EndDate**: Data zakończenia, opcjonalne.
- **MonthlyRent**: Miesięczny czynsz, wymagane, wartość dodatnia.

## Payment

Reprezentuje płatności najemcy. Przechowuje dane o identyfikatorze umowy do której dana płatność się odnosi, datę do której płatność powinna zostać zrealizowana oraz kwotę danej opłaty. Naturalnie posiada również klucz główny w postaci ID płatności.

- **Id**: Identyfikator płatności.
- **RentalAgreementId**: Identyfikator umowy najmu, wymagane.
- **Amount**: Kwota płatności, wymagane, wartość dodatnia.
- **PaymentDate**: Data płatności, wymagane.

## MaintenanceRequest

Reprezentuje zgłoszenia serwisowe, które mogą zostać utworzone przez najemców. Pozwala podejrzeć statusy zgłoszeń oraz to do jakiej nieruchomości odnosi się zgłoszenie. Statusy zmieniane są z pozycji najemcy lub administratora.

- **Id**: Identyfikator zgłoszenia.
- **PropertyId**: Identyfikator nieruchomości, wymagane.
- **Description**: Opis problemu, wymagane.
- **Status**: Status zgłoszenia (np. „Oczekujące”, „Zakończone”), wymagane.

# 6. Wylistowane kontrolery

## PropertyController

Zarządza operacjami CRUD związanymi z nieruchomościami.

- **Index():**
  - HTTP Method: GET
  - Parametry: Brak
  - Wyświetla listę wszystkich nieruchomości w systemie.
  - Zwracane dane: Widok z listą nieruchomości.
- **Details(int id):**
  - HTTP Method: GET
  - Parametry: id (int) – Identyfikator nieruchomości.
  - Wyświetla szczegóły nieruchomości na podstawie podanego identyfikatora.
  - Zwracane dane: Widok z informacjami o konkretnej nieruchomości.
- **Create():**
  - HTTP Method: GET, POST
  - Parametry (GET): Brak.
  - Parametry (POST): Property (model zawierający dane nowej nieruchomości).
    - GET: Wyświetla formularz tworzenia nowej nieruchomości.
    - POST: Odbiera dane z formularza i zapisuje nową nieruchomość w bazie.
  - Zwracane dane (GET): Widok formularza.
  - Zwracane dane (POST): Przekierowanie na listę nieruchomości.
- **Delete(int id):**
  - HTTP Method: POST
  - Parametry: id (int) – Identyfikator nieruchomości do usunięcia.
  - Usuwa nieruchomość na podstawie jej identyfikatora.
  - Zwracane dane: Przekierowanie na listę nieruchomości.

# TenantController

Zarządza operacjami związanymi z najemcami.

- **Index():**
  - HTTP Method: GET
  - Parametry: Brak
  - Wyświetla listę wszystkich najemców.
  - Zwracane dane: Widok z listą najemców.
- **Create():**
  - HTTP Method: GET, POST
  - Parametry (GET): Brak.
  - Parametry (POST): Tenant (model zawierający dane nowego najemcy).
    - GET: Wyświetla formularz tworzenia nowego najemcy.
    - POST: Dodaje nowego najemcę na podstawie danych przesłanych z formularza.
  - Zwracane dane (GET): Widok formularza.
  - Zwracane dane (POST): Przekierowanie na listę najemców.

# RentalAgreementController

Zarządza operacjami związanymi z umowami najmu.

- **Create():**
  - HTTP Method: GET, POST
  - Parametry (GET): Brak.
  - Parametry (POST): RentalAgreement (model zawierający dane nowej umowy najmu).
    - GET: Wyświetla formularz tworzenia nowej umowy najmu.
    - POST: Dodaje nową umowę najmu na podstawie danych przesłanych z formularza.
  - Zwracane dane (GET): Widok formularza.
  - Zwracane dane (POST): Przekierowanie na listę umów.
- **Details(int id):**
  - HTTP Method: GET
  - Parametry: id (int) – Identyfikator umowy najmu.
  - Wyświetla szczegóły umowy najmu.
  - Zwracane dane: Widok z informacjami o konkretnej umowie.

# PaymentController

Obsługuje płatności powiązane z umowami najmu.

- **Index(int agreementId):**
  - HTTP Method: GET
  - Parametry: agreementId (int) – Identyfikator umowy najmu.

- Wyświetla listę płatności związanych z podaną umową najmu.
- Zwracane dane: Widok z listą płatności.

- **Create():**
  - HTTP Method: GET, POST
  - Parametry (GET): Brak.
  - Parametry (POST): Payment (model zawierający dane nowej płatności).
    - GET: Wyświetla formularz dodawania nowej płatności.
    - POST: Zapisuje nową płatność do bazy danych.
  - Zwracane dane (GET): Widok formularza.
  - Zwracane dane (POST): Przekierowanie na listę płatności.

## MaintenanceRequestController

Obsługuje zgłoszenia serwisowe dla nieruchomości.

- **Create():**
  - HTTP Method: GET, POST
  - Parametry (GET): Brak.
  - Parametry (POST): MaintenanceRequest (model zgłoszenia serwisowego).
    - GET: Wyświetla formularz zgłaszania problemu.
    - POST: Dodaje nowe zgłoszenie serwisowe na podstawie przesłanych danych.
  - Zwracane dane (GET): Widok formularza.
  - Zwracane dane (POST): Przekierowanie na listę zgłoszeń.
- **UpdateStatus(int id, string status):**
  - HTTP Method: POST
  - Parametry:
    - id (int) – Identyfikator zgłoszenia.
    - status (string) – Nowy status zgłoszenia (np. "Zakończone").
  - Aktualizuje status zgłoszenia serwisowego.
  - Zwracane dane: Przekierowanie na listę zgłoszeń lub widok szczegółów zgłoszenia.

# 7. Opis systemu użytkowników

## Role w Systemie

W systemie zarządzania wynajmem nieruchomości istnieją trzy główne role użytkowników:

1. **Administrator**

**Możliwości:**

- Zarządzanie użytkownikami (tworzenie nowych kont, przypisywanie ról).
- Dodawanie, edycja i usuwanie nieruchomości, najemców oraz umów najmu.

- Zarządzanie płatnościami i zgłoszeniami serwisowymi.
- Przeglądanie raportów oraz statystyk.

Rola jest przypisywana przy tworzeniu konta przez innego administratora lub w procesie inicjalnym aplikacji**.**

2. **Manager**

**Możliwości:**

- Zarządzanie umowami najmu, płatnościami oraz zgłoszeniami serwisowymi.
- Dodawanie i edycja danych o najemcach.
- Przeglądanie listy nieruchomości, jednak bez możliwości ich edycji czy usuwania.

Rola przypisywana przez administratora lub w procesie rejestracji.

3. **Tenant (Najemca)**

**Możliwości:**

- Przeglądanie swoich płatności oraz zgłoszeń serwisowych.
- Dodawanie nowych zgłoszeń serwisowych (np. problemów z wynajmowaną nieruchomością).
- Nie ma dostępu do danych globalnych ani możliwości zarządzania nieruchomościami, najemcami, czy umowami.

# Rola użytkownika jest zarządzana za pomocą ASP.NET Identity:

- Administrator może edytować rolę użytkownikowi w panelu administracyjnym aplikacji oraz jest ona przydzielana podczas rejestracji.
- Rola jest zapisywana w bazie danych w tabeli AspNetUserRoles.
- Dla przypisania roli w kodzie można użyć serwisu UserManager

# Możliwości Użytkowników Zalogowanych vs Gości

1. **Goście (niezalogowani użytkownicy):**
    - Mogą przeglądać publiczne informacje o nieruchomościach dostępnych na wynajem.
    - Nie mają dostępu do żadnych operacji modyfikujących dane.
    - Brak możliwości dodawania zgłoszeń, przeglądania płatności czy zarządzania danymi.
2. **Zalogowani użytkownicy:**
   Mają dostęp do funkcji zależnych od ich ról:
        - **Administratorzy**: Dostęp do wszystkich funkcjonalności.
        - **Wynajmujący**: Ograniczone zarządzanie danymi (bez kontroli nad użytkownikami i nieruchomościami).

- **Najemcy**: Dostęp do własnych danych, takich jak płatności, umowy i zgłoszenia serwisowe.

## Powiązane Informacje

**Informacje powiązane z użytkownikiem:**

Dla **najemcy**:
- Lista jego zgłoszeń serwisowych.
- Historia płatności oraz aktywne umowy najmu.

Dla **administratora**:
- Pełna historia operacji, które wykonali w systemie (np. logi edycji danych).

Dla **wynamującego**:
- Lista wynajętych apartamentów wraz z płatnościami za nie.
- Widok zgłoszeń serwisowych i podgląd ich statusów.

**Informacje globalne:**
- Lista dostępnych nieruchomości.
- Raporty finansowe i statystyki (dostępne tylko dla administratorów).
- Zgłoszenia serwisowe niezwiązane z konkretnym najemcą, np. dotyczące nieruchomości pustych.

# 8. Charakterystyka najciekawszych funkcjonalności

## Śledzenie Statusu Zgłoszeń Serwisowych

Funkcja umożliwia zgłaszanie usterek przez najemców oraz śledzenie statusu ich naprawy. Administratorzy lub menedżerowie mogą zmieniać status zgłoszeń (np. „Oczekujące”, „W realizacji”, „Zakończone”), a także dodawać notatki o podjętych działaniach**.**

- **Działanie:**
  - Najemca może zgłosić problem z wynajmowaną nieruchomością poprzez prosty formularz (np. „Awaria ogrzewania”).
  - Po zgłoszeniu, administrator otrzymuje powiadomienie o nowym zgłoszeniu.
  - Status zgłoszenia zmienia się w miarę postępu naprawy.

- **Zalety:**
  - Poprawa komunikacji między wynajmującym a najemcą.
  - Automatyczne porządkowanie zgłoszeń według statusów i priorytetów.

- **Przykład użycia:**
  - Najemca zgłasza brak ciepłej wody w nieruchomości, zgłoszenie trafia do administratora, który przypisuje je technikowi. Status zmienia się na „W realizacji", a po zakończeniu naprawy – na „Zakończone".

# System Powiadomień dla Najemców

System automatycznie wysyła powiadomienia e-mail lub SMS do najemców. Przykładowe powiadomienia obejmują przypomnienia o:

- Zbliżających się terminach płatności.
- Kończących się umowach najmu.
- Nowych wiadomościach lub aktualizacjach w systemie.

- **Działanie:**
  - W przypadku zaległości w płatnościach system wysyła przypomnienia co określony czas (np. co tydzień).
  - Automatyczna generacja wiadomości oparta na danych z bazy (np. data końca umowy).
- **Zalety:**
  - Zmniejszenie ryzyka opóźnień w płatnościach.
  - Lepsza komunikacja z najemcami bez angażowania administratorów.
- **Przykład użycia:**
  - Najemca otrzymuje powiadomienie: „Przypominamy, że termin płatności za listopad wynosi 20.11.2024. Prosimy o dokonanie przelewu."

# Statystyki i Raporty Wynajmu

Funkcja generuje szczegółowe raporty i statystyki, które umożliwiają analizowanie danych związanych z wynajmowanymi nieruchomościami. Administratorzy mogą przeglądać:

- Przychody z najmu w wybranym okresie.
- Liczbę aktywnych i zakończonych umów.
- Statystyki dotyczące zgłoszeń serwisowych (np. liczba zgłoszeń zrealizowanych w danym miesiącu).

- **Działanie:**
  - Raporty są generowane dynamicznie na podstawie danych w bazie.
  - Użytkownik wybiera zakres czasu i typ raportu (np. „Przychody z wynajmu").

- **Zalety:**
  - Szybki wgląd w stan finansowy i operacyjny firmy.
  - Możliwość analizy trendów, co ułatwia podejmowanie decyzji biznesowych.
- **Przykład użycia:**
  - Administrator generuje raport: „Przychody z najmu w Q4 2024 wyniosły 50 000 PLN".

# Intuicyjne Zarządzanie Umowami Najmu

Funkcja umożliwia administratorom i menedżerom łatwe dodawanie, edycję i przeglądanie umów najmu. Formularz umowy zawiera wszystkie niezbędne dane, takie jak:

- Identyfikator nieruchomości.
- Dane najemcy.
- Daty rozpoczęcia i zakończenia najmu.
- Ustalony czynsz miesięczny.

- **Działanie:**
  - System automatycznie waliduje dane wprowadzone w formularzu (np. poprawność daty zakończenia).
  - Możliwość generowania dokumentów umów najmu w formacie PDF.
- **Zalety:**
  - Uproszczony proces tworzenia i zarządzania umowami.
  - Mniej błędów dzięki wbudowanej walidacji.
- **Przykład użycia:**
  - Menedżer dodaje nową umowę dla nieruchomości przy ulicy „Głównej 10" z miesięcznym czynszem 2500 PLN.

# Dynamiczne Powiadomienia o Statusie Płatności

System wyświetla dynamiczne powiadomienia o stanie płatności dla administratorów i menedżerów. Najemcy widzą informacje o swoich zaległościach w panelu użytkownika.

- **Działanie:**
  - Status płatności jest automatycznie aktualizowany na podstawie danych w bazie (np. „Opłacone", „Zaległość 7 dni").
  - System generuje alerty w przypadku braku płatności przez określony czas.
- **Zalety:**
  - Umożliwia administratorom szybką reakcję na zaległości.
  - Najemcy mają pełną przejrzystość dotycząca płatności.
- **Przykład użycia:**
  - Menedżer widzi na swoim panelu komunikat: „Nieruchomość: Główna 10 – Zaległość 14 dni (2500 PLN)".