Dokumentacja techniczna

Aplikacja do zarządzania ogrzewaniem gazowym.

Opis: Aplikacja będzie służyć do zarządzania ogrzewaniem gazowym w domu/mieszkaniu. Użytkownik po pierwszym zalogowaniu będzie musiał wypełnić informacje na temat swojego mieszkania, po czym będzie mógł tworzyć harmonogramy oraz sprawdzić ich szacowany koszt. Użytkownicy będą mieli również możliwość współdzielenia dostępu do zarządzenia ogrzewaniem.

Wykorzystane technologie:

- .net 8
- Entity Framework - do stworzenia bazy danych

Podczas pierwszego uruchomienia należy wykonać polecenie Update-Database

Modele

Model Mieszkanie i jego atrybuty:

- **Int Id** - Unikalny identyfikator mieszkania (klucz główny).
- **Int LiczbaOkien** - Liczba okien w mieszkaniu (zakres: 0-5).
- **Int LiczbaPokoi** - Liczba pokoi w mieszkaniu (zakres: 1-5).
- **Int BazowaTemperatura** - Bazowa temperatura mieszkania (zakres: 10-17°C).
- **String Uzytkownik** - Identyfikator użytkownika przypisanego do mieszkania.
- **IdentityUser User** - Obiekt typu IdentityUser, który reprezentuje użytkownika w systemie.
- **String Uzytkownik2** - Identyfikator drugiego użytkownika przypisanego do mieszkania .

Model Harmonogram i jego atrybuty:

Int Id - Unikalny identyfikator harmonogramu.

String Nazwa - Nazwa harmonogramu.

Int Start - Godzina rozpoczęcia (zakres: 0-23).

Int End - Godzina zakończenia (zakres: 1-24).

Int DocelowaTemperatura - Docelowa temperatura do osiągnięcia w przedziale godzinowym (zakres: 18-35°C).

Int MieszkanieId - Identyfikator mieszkania, do którego należy harmonogram (klucz obcy).

virtual Mieszkanie - Nawigacyjne odniesienie do obiektu Mieszkanie.

Walidacja:

Godzina Start musi być mniejsza niż godzina End.

```
Odwołania: 12
public int Id { get; set; }
Odwołania: 12
public string Nazwa { get; set; }
Odwołania: 15
[Range(0, 23)] public int Start { get; set; }
Odwołania: 15
[Range(1, 24)] public int End { get; set; }
Odwołania: 11
[Range(18, 35)] public int DocelowaTemperatura  { get; set; }
Odwołania: 3
public int MieszkanieId { get; set; }
1 odwołanie
public virtual Mieszkanie? Mieszkanie { get; set; }
Odwołania: 0
public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
{
    if (Start >= End)
    {
        yield return new ValidationResult(
            "Wartość 'Start' musi być mniejsza niż 'End'.",
            new[] { nameof(Start), nameof(End) });
    }
}
```

4. Kontrolery

HarmonogramController

MieszkaniesController

4.1 MieszkaniesController

Kontroler MieszkaniesController zarządza operacjami związanymi z mieszkanami.

Metody:

Index() - Ta metoda zwraca widok z listą mieszkań przypisanych do zalogowanego użytkownika. Metoda filtruje dane, wyświetlając tylko mieszkania, do których użytkownik jest przypisany.

- **Przyjmowane parametry**: Brak
- Pobiera identyfikator aktualnie zalogowanego użytkownika.
- Sprawdza, czy użytkownik jest przypisany do jakiegokolwiek mieszkania. Jeśli nie, zwraca odpowiedź "Forbidden" (403).
- Jeśli użytkownik jest przypisany do mieszkania, zwraca widok z listą mieszkań użytkownika.
- Dodakowo wysyła dane do widoku z informacją czy użytkownik jest już przypisany do mieszkania na których podstawie wyświetlane są dostępne dla

użytkownika możliwości albo tylko dodania, albo do usuwania dodawania drugiego użytkownika itd.

Details() - pozwala na edytowanie danych mieszkania. Kontroler sprawdza, czy użytkownik jest właścicielem mieszkania przed pozwoleniem na edycję.

- **Przyjmowane parametry**: int? id (ID mieszkania)
- Sprawdza, czy ID mieszkania zostało przekazane. Jeśli nie, zwraca "Not Found" (404).
- Pobiera szczegóły mieszkania o danym ID.
- Sprawdza, czy mieszkanie istnieje i czy użytkownik ma do niego dostęp. Jeśli nie, zwraca "Forbidden" (403).

Create() - umożliwia tworzenie nowego mieszkania. Przed zapisaniem mieszkania w bazie danych, kontroler sprawdza, czy użytkownik już nie ma przypisanego innego mieszkania.

- **Przyjmowane parametry**: int? id (ID mieszkania)
- Sprawdza, czy ID mieszkania zostało przekazane. Jeśli nie, zwraca "Not Found" (404).
- Pobiera szczegóły mieszkania o danym ID.
- Sprawdza, czy użytkownik już ma przypisane mieszkanie, jeśli tak, zwraca błąd.
- Sprawdza, czy mieszkanie istnieje i czy użytkownik ma do niego dostęp. Jeśli nie, zwraca "Forbidden" (403).

Edit() - pozwala na edytowanie danych mieszkania. Kontroler sprawdza, czy użytkownik jest właścicielem mieszkania przed pozwoleniem na edycję. Dodany urzytkownik nie ma tej możliwości.

- **Przyjmowane parametry**: int? id (ID mieszkania)
- **Opis działania**:
- Sprawdza, czy ID mieszkania zostało przekazane. Jeśli nie, zwraca "Not Found" (404).
- Pobiera mieszkanie o danym ID.
- Sprawdza, czy mieszkanie należy do aktualnie zalogowanego użytkownika (jeśli nie, zwraca "Forbidden").
- Jeśli zalogowany użytkownik **jest** Uzytkownik2, to dostęp do edycji jest zabroniony, i zwracana jest odpowiedź 403 Forbidden.

Delete() - Usuwa mieszkanie. Podobnie jak w edycji drugi użytkownik nie ma tej możliwości

- **Przyjmowane parametry**: int? id (ID mieszkania)
- **Opis działania**:
- Sprawdza, czy ID mieszkania zostało przekazane. Jeśli nie, zwraca "Not Found" (404).
- Pobiera mieszkanie o danym ID.
- Sprawdza, czy mieszkanie należy do aktualnie zalogowanego użytkownika (jeśli nie, zwraca "Forbidden").
- Jeśli zalogowany użytkownik **jest** Uzytkownik2, to dostęp do edycji jest zabroniony, i zwracana jest odpowiedź 403 Forbidden.

Dodaj() - Ta metoda pozwala na przypisanie drugiego użytkownika do mieszkania. Użytkownik może wybrać dostępnych użytkowników i dodać ich do swojego mieszkania. Również dodany już użytkownik ma ą funkcje zablokowaną

- **Przyjmowane parametry**: int? id (ID mieszkania) string Uzytkownik2 (ID mieszkania i ID drugiego użytkownika)
- Sprawdza, czy ID mieszkania zostało przekazane. Jeśli nie, zwraca "Not Found" (404).
- Pobiera mieszkanie o danym ID.
- Pobiera listę użytkowników, którzy nie są jeszcze przypisani do tego mieszkania.
- Pokazuje formularz do przypisania drugiego użytkownika.
- Aktualizuje mieszkanie, przypisując drugiego użytkownika (Uzytkownik2).
- Jeśli zalogowany użytkownik **jest** Uzytkownik2, to dostęp do edycji jest zabroniony, i zwracana jest odpowiedź 403 Forbidden.

```
var przypisaniUzytkownicy = _context.Mieszkanie
.Select(d => d.Uzytkownik)
.ToList();
var filteredUsers = await _context.Users
.Where(user => user.Id != uzytkownikAktualny && !przypisaniUzytkownicy.Contains(user.Id))
.ToListAsync();

ViewData["Uzytkownik2"] = new SelectList(filteredUsers, "Id", "UserName");

return View(mieszkanie);
```

Wszystkie metody są zabezpieczone przed wejściem

4.2HarmonogramController

Kontroler HarmonogramController obsługuje harmonogramy temperatury w mieszkaniu.

Metody:

Index() - Metoda **Index** wyświetla listę harmonogramów przypisanych do aktualnie zalogowanego użytkownika. Tylko harmonogramy związane z mieszkaniami przypisanymi do użytkownika będą wyświetlane.

- Parametry: Brak
- Sprawdza, czy użytkownik jest zalogowany (przez ClaimTypes.NameIdentifier). Jeśli użytkownik nie jest zalogowany, zwraca odpowiedź 403 Forbidden.
- Zwraca widok z listą wszystkich harmonogramów.

Create() - umożliwia tworzenie nowego harmonogramu dla mieszkania przypisanego do zalogowanego użytkownika.

- Parametry: Brak
- Sprawdza, czy model jest poprawny (np. czy wszystkie wymagane dane są obecne).
- Pobiera identyfikator użytkownika i sprawdza, czy jest zalogowany. Jeśli nie, zwraca 401 Unauthorized.
- Pobiera mieszkanie przypisane do użytkownika z bazy danych i przypisuje identyfikator mieszkania do tworzonego harmonogramu

Edit() - Funkcja pozwala na edycje harmonogramu

- Parametry: Identyfikator harmonogramu.
- Sprawdza, czy identyfikator id jest przekazany. Jeśli nie, zwraca 404 Not Found.
- Pobiera harmonogram na podstawie identyfikatora id.
- Sprawdza, czy użytkownik jest zalogowany. Jeśli nie, zwraca 403 Forbidden.
- Zwraca formularz edycji harmonogramu, w tym dane mieszkania (ale mieszkania nie można edytować).

Delete() - Usuwanie harmonogramu

- Parametry: Identyfikator harmonogramu.
- Sprawdza, czy identyfikator id jest przekazany. Jeśli nie, zwraca 404 Not Found.
- Pobiera harmonogram na podstawie identyfikatora id.
- Sprawdza, czy użytkownik jest zalogowany. Jeśli nie, zwraca 403 Forbidden.
- Zwraca formularz potwierdzenia usunięcia harmonogramu

Details() - Pozwala na podgląd danych na danego harmonogramu

- Parametry: id (int?): Identyfikator harmonogramu
- Sprawdza, czy identyfikator id jest przekazany. Jeśli nie, zwraca 404 Not Found.
- Sprawdza, czy użytkownik jest zalogowany. Jeśli nie, zwraca 403 Forbidden.
- Pobiera szczegóły harmonogramu z bazy danych na podstawie identyfikatora id.
- Zwraca widok z danymi harmonogramu.

Licznik() - Pobiera Informacje na z mieszkania oraz harmonogramu, i na ich podstawie wylicza szacowany kosz dzienny, tygodniowy oraz miesięczny korzystania z danego harmonogramu.

Parametry: Identyfikator harmonogramu .

- Sprawdza, czy identyfikator id jest przekazany. Jeśli nie, zwraca 404 Not Found.
- Sprawdza, czy użytkownik jest zalogowany. Jeśli nie, zwraca 403 Forbidden.
- Pobiera mieszkanie powiązane z użytkownikiem.
- Pobiera harmonogram na podstawie id.
- Oblicza dzienną, tygodniową i miesięczną cenę na podstawie danych o mieszkaniu i harmonogramie.
- Przekazuje obliczone wartości do widoku.

```
// Oblicz cenę dzienną
decimal kurs = 0.1m;
decimal liczbaOkien = (decimal)mieszkanie.LiczbaOkien;
decimal liczbaPokoi = (decimal)mieszkanie.LiczbaPokoi;
decimal bazowaTemperatura = (decimal)mieszkanie.BazowaTemperatura;
decimal docelowaTemperatura = (decimal)harmonogram.DocelowaTemperatura;
decimal czasTrwania = (decimal)harmonogram.End - (decimal)harmonogram.Start;

decimal cenaDzienna = Math.Round(
    (liczbaOkien * 1.2m)
    * (liczbaPokoi * 1.5m)
    * czasTrwania
    * ((docelowaTemperatura - bazowaTemperatura))
    * kurs,
    2,
    MidpointRounding.AwayFromZero
);

decimal cenatygodniowa = Math.Round(cenaDzienna * 7, 2, MidpointRounding.AwayFromZero);
decimal cenamiesieczna = Math.Round(cenaDzienna * 30, 2, MidpointRounding.AwayFromZero);
ViewBag.CenaDzienna = cenaDzienna;
ViewBag.cenatygodniowa = cenatygodniowa;
ViewBag.cenamiesieczna = cenamiesieczna;
return View();
```

W systemie użytkowników nie ma zdefiniowanych ról, użytkownicy mogą zostać powiązaniu z tylko jednym mieszkaniem. Użytkownik zalogowany ma dostęp do tworzenia mieszkania i harmonogramów oraz funkcji z nimi związanych. Użytkownik nie zalogowany ma jedynie dostęp do strony z opisem działania strony internetowej.

Ciekawe funkcje

Wyświetlanie użytkownikowi szacunkowych koszów harmonogramu na podstawie danych z harmonogramu oraz danych z przypisanego do niego mieszkania

Autorzy:

Cezary Szymonik

Paweł Zielonka

Grupa: Grupa 4