SYSTEM OBSŁUGI PRZYSTANI ŻEGLARSKIEJ

MARINEYE

Autor projektu: **Piotr Kupczyk**

Grupa: **K80**

Numer albumu:

Data: **01.11.2024**

Repozytorium: **https://github.com/piotrku91/MARINEYE**

Wersja dokumentacji: **0.2**

Wersja aplikacji: **0.1**

Spis treści

1. Wykorzystane technologie

1.1. Krótki opis

System służy do zarządzania przystanią żeglarską w której czarterowane są łodzie dla użytkowników zewnętrznych a także dla działającego w jej wewnętrznych strukturach klubu żeglarskiego.

Przy pomocy **MARINEYE** każdy użytkownik może łatwo zarezerwować łódź na poszczególny dzień lub dni a po zatwierdzeniu przez bosmana klubu dokonać operacji wypożyczenia sprzętu wodnego.

Pozwala na śledzenie aktualnych rezerwacji, statusu floty, składek klubowiczów, uregulowanych opłat za czarter przez użytkowników zewnętrznych.

Zapewnia też zarządzanie aktualną flotą, przestrzeganiem terminów konserwacji oraz zgłaszania usterek.

Dodatkową funkcjonalnością jest prowadzenie statystyk w celu określenia obłożenia floty żeglarskiej w każdym sezonie co pozwala w przyszłości na rozwój i kupno sprzętu ukierunkowanego na potrzeby klientów zewnętrznych lub klubowiczów.

Aplikacja kliencka dostępna jest w przeglądarce internetowej po stronie klienta co ułatwia korzystanie z systemu z każdego miejsca na świecie.

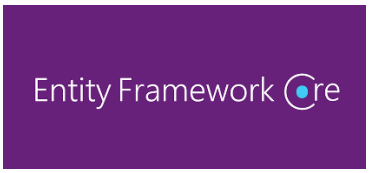
2. Wykorzystane technologie

Aplikacja serwera opiera się na popularnym wzorcu architektonicznym MVC (ang. Model-View-Controller), który pozwala na klarowną separację warstw odpowiedzialnych za dane, logikę biznesową oraz interfejs użytkownika.

Główna część aplikacji bazuje na technologii **Microsoft .NET 8.0**, z zastosowaniem **Entity Framework** jako technologii ORM (ang. Object-Relational Mapping), która umożliwia płynną integrację między relacyjną bazą danych a aplikacją zorientowaną obiektowo.

Kod aplikacji napisano w języku C#, a interfejs użytkownika opiera się na plikach HTML i CSS oraz innych komponentach wspierających funkcjonalność i estetykę aplikacji. Dane przechowywane są w bazie **MS SQL Server**, co zapewnia stabilne i wydajne zarządzanie informacjami w systemie.

Entity Framework umożliwia wygodną i bezpieczną obsługę interakcji z bazą, co znacząco upraszcza zarządzanie danymi w kontekście aplikacji.



3. Instrukcje uruchomienia projektu

- Pobierz repozytorium za pomocą polecenia:
  git clone https://github.com/piotrku91/MARINEYE.git

- Upewnij się że na twoim komputerze jest zainstalowane:
  - Środowisko uruchomieniowe NET8.0,
  - W menedżerze pakietów NUGET (Pakiety najwyższego poziomu)
    - Microsoft.AspNetCore.Diagnostics.EntityFrameWorkCore (8.0.11)
    - Microsoft.AspNetCore.Identity.EntityFrameWorkCore (8.0.11)
    - Microsoft AspNetCore.Identity.UI (8.0.11)
    - Microsoft.EntityFrameworkCore.Sqlite (8.0.11)
    - Microsoft.EntityFrameworkCore.SqlServer (8.0.11)
    - Microsoft.EntityFrameworkCore.Tools (8.0.11)
    - Microsoft.VisualStudio.Web.CodeGeneration.Design (8.0.7)

- Wejdź do pobranego folderu MARINEYE i otwórz rozwiązanie (MARINEYE.sln) w Visual Studio

- W konsoli menedżera pakietów NUGET wydaj polecenie:
  Update-Database -Migration ClearInitial

- Wyczyść rozwiązanie oraz uruchom w wersji **Release**

- Po uruchomieniu powinna zostać wyświetlona strona główna wraz z tekstem powitalnym oraz powinien być dostępny pasek nawigacyjny z możliwością przeglądu floty, rejestracją oraz możliwością logowania do systemu

- Zarejestruj pierwsze konto (Pierwsze konto w systemie zostaje głównym administratorem, każde kolejne otrzymuje najniższy poziom uprawnień)

- Po rejestracji i zalogowaniu system jest gotowy do dalszej pracy

4. Opis struktury projektu

Kontrolery	• BoatCalendarEvents • Boats • ClubDues • DueTransactions • Home • Statistics • User
Modele	• MARINEYEUser (dziedziczony z domyślnego usera Identity) • User (DTO) • BoatCalendarEvent • BoatCalendarEvent (DTO) • Boat • CharterDueTransaction • ClubDueTransaction • ClosedDueTransaction • ClubDue • ErrorView • Statistics.MostUsedBoat (DTO)
Widoki	• BoatCalendarEvents • Boats • ClubDues • DueTransactions • Home • Statistics • User

Inne	• Constants (Globalne stałe pomocnicze do definiowania ról w projekcie) • Utilities.BoatState (Pomocniczy słownik stanu łodzi) • Utilities.BoatCalendarEventState (Pomocniczy słownik statusu rezerwacji) • Utilities.BoatCalendarEventType (Pomocniczy słownik typu rezerwacji) • Transactions (Pomocniczy plik wykonujący z funkcjami zarządzającymi transakcjami) • MARINEYEContext (Główny plik kontekstu bazy danych)

5. Szczegółowa specyfikacja

5.1. Modele

- MARINEYEUser (dziedziczony z domyślnego usera Identity) – Główny model użytkownika używającego systemu (Przechowywany w bazie danych)

```
[PersonalData]
[Display(Name = "Imię")]
public string? FirstName { get; set; }
[PersonalData]
[Display(Name = "Nazwisko")]
public string? LastName { get; set; }
[PersonalData]
[Display(Name = "Data urodzenia")]
public DateTime DOB { get; set; }
[Display(Name = "Data rejestracji")]
public DateTime RegistrationDate { get; set; }

// Personal account operations
[PersonalData]
[Display(Name = "Stan konta")]
public int CashAmount { get; set; }

public void Deposit(int amount) {
    CashAmount += amount;
}

public bool Withdraw(int amount) {
    if (amount > CashAmount) {
        return false;
    }

    CashAmount -= amount;
    return true;
}
public int GetCashAmount() {
    return CashAmount;
}
```

- User (DTO) – Model używany do edycji wybranych ustawień użytkownika (w tym roli). (Nieprzechowywany w bazie danych)

```
[Display(Name = "Identyfikator")]
public string Id { get; set; }

[Display(Name = "Adres e-mail")]
public string Email { get; set; }

[Display(Name = "Imię")]
public string FirstName { get; set; }

[Display(Name = "Nazwisko")]
public string LastName { get; set; }

[Display(Name = "Rola")]
public string Role { get; set; }

[Display(Name = "Data rejestracji")]
public DateTime RegistrationDate { get; set; }

[Display(Name = "Opłaty")]
public bool AllDuesPaid { get; set; }
```

- BoatCalendarEvent – Model rezerwacji jednostki w danym terminie (Przechowywany w bazie danych)

```
[Display(Name = "Identyfikator")]
public int Id { get; set; }

[Required]
[Display(Name = "Data rozpoczęcia")]
public DateTime BeginDate { get; set; }

[Required]
[Display(Name = "Data zakończenia")]
public DateTime EndDate { get; set; }

public string UserId { get; set; }

[ForeignKey("UserId")]
public virtual MARINEYEUser User { get; set; }

[Required]
public int BoatId { get; set; }

[ForeignKey("BoatId")]
public virtual BoatModel Boat { get; set; }

[Display(Name = "Status")]
public BoatCalendarEventState EventState { get; set; }

[Display(Name = "Typ")]
public BoatCalendarEventType EventType { get; set; }
```

- BoatCalendarEvent (DTO) – Model do tworzenia i edycji rezerwacji w danym terminie (Nieprzechowywany w bazie danych)

```
    [Display(Name = "Identyfikator")]
    public int Id { get; set; }

    [Display(Name = "Data rozpoczęcia")]
    public DateTime BeginDate { get; set; }

    [Display(Name = "Data zakończenia")]
    public DateTime EndDate { get; set; }

    [Display(Name = "Jednostka")]
    public int BoatId { get; set; }
```

- Boat – Model jednostki która może być wypożyczona (Przechowywany w bazie danych)

```
    public int Id { get; set; }

    [Display(Name = "Nazwa jednostki")]
    public string Name { get; set; }

    [Display(Name = "Opis")]
    public string Description { get; set; }

    [Display(Name = "Długość jednostki")]
    [Range(1, 100, ErrorMessage = "Długość musi być w zakresie od 1 do
    100 m")]
    public int Length { get; set; }

    [Display(Name = "Rok produkcji")]
    [Range(1900, 3000, ErrorMessage = "Rok produkcji musi być w
    zakresie od 1900 do 3000")]
    public int Year { get; set; }

    [Display(Name = "Klasa jednostki")]
    public string Class { get; set; }

    [Display(Name = "Stan")]
    public BoatState State { get; set; }

    [Display(Name = "Zdjęcie")]
    public string ImageName { get; set; }

    [Display(Name = "Koszt czarteru (1 dzień)")]
    public int OneDayCharterCost { get; set; }
```

- CharterDueTransaction – Model transakcji dla czarteru jednostki (Przechowywany w bazie danych)

```
    [Key]
     [Display(Name = "Identyfikator")]
     public int Id { get; set; }

     [Display(Name = "Identyfikator rezerwacji")]
     public int BoatCalendarEventId { get; set; }

     [ForeignKey("BoatCalendarEventId")]
     public virtual BoatCalendarEvent BoatCalendarEvent { get; set; }
```

```
    [Display(Name = "Zapłacona kwota")]
    public int AmountPaid { get; set; }

    [Display(Name = "Data płatności")]
    public DateTime PaymentDate { get; set; }

    public bool Closed { get; set; }
```

- ClubDueTransaction – Model transakcji dla opłat członkowskich (Przechowywany w bazie danych)

```
    [Key]
    [Display(Name = "Identyfikator")]
    public int Id { get; set; }

    public string UserId { get; set; }

    [ForeignKey("UserId")]
    [Display(Name = "Użytkownik")]
    public virtual MARINEYEUser User { get; set; }

    [Display(Name = "Identyfikator opłaty klubowej")]
    public int ClubDueId { get; set; }

    [ForeignKey("ClubDueId")]
    public virtual ClubDueModel ClubDue { get; set; }

    [Display(Name = "Zapłacona kwota")]
    public int AmountPaid { get; set; }

    [Display(Name = "Data płatności")]
    public DateTime PaymentDate { get; set; }

    public bool Closed { get; set; }
```

- ClosedDueTransaction – Model transakcji zamkniętych (historycznych). Jeden model dla obu typów transakcji. (Przechowywany w bazie danych)

```
    [Key]
    [Display(Name = "Identyfikator")]
    public int Id { get; set; }

    [Display(Name = "Id transakcji (historyczne)")]
    public int HistoricalDueTransactionId;

    [Display(Name = "Zapłacona kwota")]
    public int AmountPaid { get; set; }

    [Display(Name = "Data płatności")]
    public DateTime PaymentDate { get; set; }

    [Display(Name = "Data zamknięcia")]
    public DateTime ClosedDate { get; set; }

    [Display(Name = "Opis")]
    public string Description { get; set; }

    [Display(Name = "Płacący")]
    public string FullName { get; set; }
```

```
    [Display(Name = "Login zamykającego")]
    public string CloserUserName { get; set; }
```

- ClubDue – Model opłaty członkowskiej (Przechowywany w bazie danych)

```
    [Display(Name = "Identyfikator")]
     public int Id { get; set; }

     [Display(Name = "Data początku okresu")]
     public DateTime PeriodBegin { get; set; }

     [Display(Name = "Data końca okresu")]
     public DateTime PeriodEnd { get; set; }

     [Display(Name = "Opis")]
     public string? Description { get; set; }

     [Display(Name = "Kwota")]
     public int Amount { get; set; }
```

- Statistics.MostUsedBoat (DTO) – Model wyjściowy statystyk dla najbardziej używanej łodzi / największej ilości dni (Nieprzechowywany w bazie danych)

```
     public int BoatId { get; set; }

      [ForeignKey("BoatId")]
      public virtual BoatModel Boat { get; set; }

      public int SumOfUseTimes;
```

5.2. Kontrolery z metodami

- BoatCalendarEvents
  - public async Task<IActionResult> Index()
    Metoda pobierająca rezerwację z bazy danych których termin jeszcze nie minął.

  - public IActionResult Create()
    Metoda wyświetlająca widok do tworzenia nowej rezerwacji

  - public async Task<IActionResult> Confirm(int? id)
    Metoda potwierdzająca rezerwację i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id rezerwacji na której ma być wykonana operacja potwierdzenia. Metoda waliduje czy wybrany sprzęt nie jest uszkodzony i czy użytkownik ma opłacone wszystkie składki.

  - [HttpPost] public async Task<IActionResult> Create([Bind("Id,BeginDate,EndDate,BoatId")] BoatCalendarEventDTO boatCalendarEventDTO)

Metoda tworząca nową rezerwacje i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje model BoatCalendarEventDTO który zawiera niezbędne informacje wejściowe do stworzenia rezerwacji. Reszta danych jest pobierana z aktualnej bazy danych, systemu. Jeżeli to rezerwacja czarterowa automatycznie następuje próba pobrania opłaty (dzień x stawka za dzień czarteru jednostki) i walidacja powodzenia tej operacji . Rezerwacja zostaje dodana jako rekord bazy danych (jako model BoatCalendarEvent).

- public async Task<IActionResult> Edit(int? id)
  Metoda wyświetlająca widok do edycji rezerwacji. Jako argument przyjmuje Id rezerwacji do edycji.

- [HttpPost] public async Task<IActionResult> Edit(int id, [Bind("Id,BeginDate,EndDate,BoatId")] BoatCalendarEventDTO boatCalendarEventDTO)
  Metoda edytująca rezerwacje i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id rezerwacji do edycji oraz model BoatCalendarEventDTO który zawiera niezbędne informacje wejściowe do edycji rezerwacji. Reszta danych jest pobierana z aktualnej bazy danych, systemu. Metoda waliduje czy wydarzenie się już nie rozpoczęło.

  public async Task<IActionResult> Delete(int? id)
  Metoda wyświetlająca widok do usuwania rezerwacji. Jako argument przyjmuje Id rezerwacji do usunięcia.

- [HttpPost, ActionName("Delete")] public async Task<IActionResult> DeleteConfirmed(int id)
  Metoda usuwająca rezerwacje i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id rezerwacji do usunięcia. Metoda waliduje czy wydarzenie się już nie rozpoczęło i czy nie ma powiązanych z nim transakcji.

- Boats
  - public async Task<IActionResult> Index()
    Metoda pobierająca jednostki z bazy danych.

  - public IActionResult Create()
    Metoda wyświetlająca widok do tworzenia nowej jednostki

    public async Task<IActionResult> Report(int? id)
    Metoda zmieniająca status jednostki na uszkodzony (Naprawa) i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id jednostki do zgłoszenia problemu.

- [HttpPost] public async Task<IActionResult> Create([Bind("Id,Name,Description,Length,Year,Class,State,ImageName,OneDayCharterCost")] BoatModel boatModel))

  Metoda tworząca nową jednostkę i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje model BoatModel który zawiera dane o jednostce.

- public async Task<IActionResult> Edit(int? id)
  Metoda wyświetlająca widok do edycji jednostki. Jako argument przyjmuje Id jednostki do edycji.

- [HttpPost] public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description,Length,Year,Class,State,ImageName,OneDayCharterCost")] BoatModel boatModel)
  Metoda edytująca jednostkę i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id jednostki do edycji.

  public async Task<IActionResult> Delete(int? id)
  Metoda wyświetlająca widok do usuwania jednostki. Jako argument przyjmuje Id jednostki do usunięcia.

- [HttpPost, ActionName("Delete")] public async Task<IActionResult> DeleteConfirmed(int id)
  Metoda usuwająca rezerwacje i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id jednostki do usunięcia. Metoda waliduje czy łódź jest aktualnie zarezerwowana.

- ClubDues
  - public async Task<IActionResult> Index()
    Metoda pobierająca z bazy danych wszystkie opłaty członkowskie które zostały stworzone po lub w dzień rejestracji aktualnego użytkownika.

  - public IActionResult Create()
    Metoda wyświetlająca widok do tworzenia nowej opłaty członkowskiej

  - public Task<IActionResult> Create([Bind("Id,PeriodBegin,PeriodEnd,Description,Amount")] ClubDueModel clubDueModel)
    Metoda tworząca nową opłatę członkowską i zapisująca zmiany w bazie danych.

  - public async Task<IActionResult> Pay(int? id) **Metoda opłacająca opłatę członkowską i zapisująca zmiany w bazie danych. Metoda waliduje czy transakcja się powiodła.**

- public async Task<IActionResult> Edit(int id, [Bind("Id,PeriodBegin,PeriodEnd,Description,Amount")] ClubDueModel clubDueModel)
  Metoda edytująca opłatę członkowską i zapisująca zmiany w bazie danych.

- public async Task<IActionResult> Delete(int? id)
  Metoda wyświetlająca widok do usuwania opłaty członkowskiej. Jako argument przyjmuje Id opłaty członkowskiej do usunięcia.

- [HttpPost, ActionName("Delete")] public async Task<IActionResult> DeleteConfirmed(int id)
  Metoda usuwająca opłatę członkowską i zapisująca zmiany w bazie danych. Jako argument wejściowy przyjmuje id opłaty członkowskiej do usunięcia. Metoda waliduje czy z opłatą członkowską są powiązane jakieś niezamknięte transakcje.

- DueTransactions
  - public async Task<IActionResult> Index()
    Metoda pobierająca z bazy danych wszystkie niezamknięte transakcje (klubowe/czarter), saldo konta użytkownika oraz saldo klubowe w niezamkniętych transakcjach.

  - public async Task<IActionResult> TopUpAccount ()
    Metoda zasilająca konto użytkownika. UWAGA! To jest tylko tymczasowa metoda dla przypadków testowych. Realnie należało by zaimplementować połączenie z systemem bankowym lub innym interfejsem płatności.

  - public async Task<IActionResult> RollbackClubDue(int? id)
    Metoda wyświetlająca widok do cofnięcia transakcji klubowej. Jako argument wejściowy przyjmuje id transakcji do cofnięcia.

  - public async Task<IActionResult> RollbackClubDueConfirmed(int id)
    Metoda cofająca transakcję klubową. Jako argument wejściowy przyjmuje id transakcji do cofnięcia. Cofnięcie transakcji zwraca kwotę na konto użytkownika który ją wykonał. Zmienia się też status opłaty członkowskiej ponieważ płatność zostaje wycofana.

  - public async Task<IActionResult> RollbackCharterDue(int? id)
    Metoda wyświetlająca widok do cofnięcia transakcji czarterowej. Jako argument wejściowy przyjmuje id transakcji do cofnięcia.

  - public async Task<IActionResult> RollbackCharterDueConfirmed(int id)
    Metoda cofająca transakcję czarterową. Jako argument wejściowy przyjmuje id transakcji do cofnięcia. Cofnięcie transakcji zwraca kwotę na konto użytkownika który ją wykonał.

  - public async Task<IActionResult> CloseClubDue(int? id)

Metoda wyświetlająca widok do zamknięcia transakcji klubowej. Jako argument wejściowy przyjmuje id transakcji do zamknięcia.

- public async Task<IActionResult> CloseClubDueConfirmed(int id)
  Metoda zamykająca transakcję klubową. Jako argument wejściowy przyjmuje id transakcji do zamknięcia. Zamknięcie transakcji powoduje nieodwracalność tej transakcji. Zwrot opłaty klubowej jest od tej pory niemożliwy.

- public async Task<IActionResult> CloseCharterDue(int? id)
  Metoda wyświetlająca widok do zamknięcia transakcji czarterowej. Jako argument wejściowy przyjmuje id transakcji do zamknięcia. Zamknięcie transakcji powoduje nieodwracalność tej transakcji.

- public async Task<IActionResult> CloseClubDueConfirmed(int id)
  Metoda Metoda zamykająca transakcję czarterową. Jako argument wejściowy przyjmuje id transakcji do zamknięcia. Zamknięcie transakcji powoduje nieodwracalność tej transakcji. Zwrot opłaty za czarter jest od tej pory niemożliwy.

- Home
  - public async Task<IActionResult> Index()
    Metoda wyświetlająca widok strony powitalnej

  - public IActionResult Error()
    Metoda wyświetlająca widok błędu

- Statistics
  - public async Task<IActionResult> Index()
    Metoda wyświetlająca widok tabel ze statystykami. Metoda pobiera z bazy danych informacje o użytkowaniu łodzi sumuje je i wyświetla tabelarycznie

- User
  - public async Task<IActionResult> Index()
    Metoda pobiera z bazy danych informacje o użytkownikach i sprawdza status dla każdego z klubowiczów czy wszystkie składki zostały opłacone. Rola „klient" zwraca zawsze status „NIE DOTYCZY". Sprawdzenie następuje tylko dla składek stworzonych po lub w dzień daty rejestracji użytkownika

  - public async Task<IActionResult> Edit(int? id)
    Metoda wyświetlająca widok do edycji użytkownika. Jako argument przyjmuje Id użytkownika do edycji.

  - [HttpPost] public async Task<IActionResult> Edit(UserModelDTO model)
    Metoda wyświetlająca widok do edycji użytkownika. Jako argument przyjmuje UserModelDTO **czyli model przechowujące najważniejsze informacje o użytkowniku (do edycji).**

6. Opis systemu użytkowników

6.1. Dostępne role w systemie

Rola	**Krótki opis**
Admin	Najwyższy poziom uprawnień. Pierwszy użytkownik w systemie zostaję przypisany do tej roli automatycznie.
Bosman	Poziom uprawnień dla zarządzającego przystanią żeglarską
Członek	Poziom uprawnień dla członków klubu żeglarskiego którzy uiszczają cykliczne opłaty klubowe
Klient	Poziom uprawnień dla klientów zewnętrznych którzy mogą czarterować jednostki pływające a opłatę uiszczają przy każdej rezerwacji. Każdy nowo zarejestrowany użytkownik otrzymuję tę role jako domyślną.

6.2. Uprawnienia poszczególnych ról

Rola	**Uprawnienia**
Admin	- Może zmieniać uprawnienia innych użytkowników (wszystkie role) - Może wyświetlać, tworzyć, edytować i usuwać jednostki pływające - Może wyświetlać, tworzyć, edytować i usuwać opłaty członkowskie - Może przeglądać listę użytkowników - Może wyświetlać swoje saldo i zasilać je środkami - Może wyświetlać saldo klubowe (w otwartych transakcjach) - Może wyświetlać listę transakcji klubowiczów na konto klubu (wszystkich użytkowników) - Może wyświetlać listę transakcji czarterów na konto klubu (wszystkich użytkowników) - Może cofać transakcję klubowiczów - Może cofać transakcje czarterów - Może zamykać permanentnie transakcje klubowiczów - Może zamykać permanentnie transakcje czarterów - Może wyświetlać saldo klubowe (transakcje zamknięte) - Może przeglądać historie wszystkich transakcji zamkniętych - Może przeglądać statystyki - Może opłacać swoje składki członkowskie - Może wyświetlać, tworzyć (niepotwierdzone), edytować, usuwać i potwierdzać rezerwacje w wypożyczalni - Może zgłaszać problemy odnośnie stanu sprzętu
Bosman	- Może wyświetlać, tworzyć, edytować i usuwać jednostki pływające - Może wyświetlać, tworzyć, edytować i usuwać opłaty członkowskie - Może przeglądać listę użytkowników - Może wyświetlać swoje saldo i zasilać je środkami - Może wyświetlać saldo klubowe (w otwartych transakcjach) - Może wyświetlać listę transakcji klubowiczów na konto klubu (wszystkich użytkowników) - Może wyświetlać listę transakcji czarterów na konto klubu (wszystkich użytkowników)

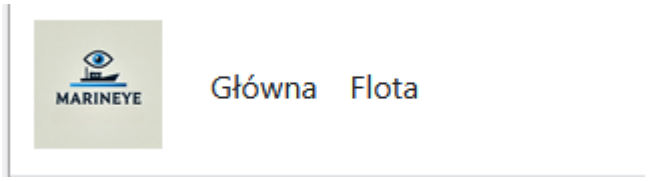
	- Może cofać transakcję klubowiczów - Może cofać transakcje czarterów - Może zamykać permanentnie transakcje klubowiczów - Może zamykać permanentnie transakcje czarterów - Może wyświetlać saldo klubowe (transakcje zamknięte) - Może przeglądać historie wszystkich transakcji zamkniętych - Może przeglądać statystyki - Może opłacać swoje składki członkowskie - Może wyświetlać, tworzyć (niepotwierdzone), edytować, usuwać i potwierdzać rezerwacje w wypożyczalni - Może zgłaszać problemy odnośnie stanu sprzętu
Członek	- Może wyświetlać jednostki pływające - Może wyświetlać opłaty członkowskie - Może przeglądać listę użytkowników - Może wyświetlać swoje saldo i zasilać je środkami - Może wyświetlać listę transakcji klubowiczów na konto klubu (tylko swoje) - Może przeglądać statystyki - Może opłacać swoje składki członkowskie - Może wyświetlać, tworzyć rezerwacje (niepotwierdzone) w wypożyczalni - Może zgłaszać problemy odnośnie stanu sprzętu
Klient	- Może wyświetlać jednostki pływające - Może wyświetlać swoje saldo i zasilać je środkami - Może wyświetlać listę transakcji czarterów na konto klubu (tylko swoje) - Może przeglądać statystyki - Może wyświetlać, tworzyć rezerwacje (niepotwierdzone) w wypożyczalni

6.3. Widoczność informacji dla użytkowników niezalogowanych

Niezalogowani użytkownicy mają dostęp jedynie do listy dostępnej floty, co oznacza, że mogą przeglądać podstawowe informacje o jednostkach pływających oferowanych przez przystań. W tym trybie wyświetlane są kluczowe dane, takie jak nazwa łodzi, klasa, długość, rok produkcji, aktualny stan oraz koszt czarteru na jeden dzień.

Dostęp do innych funkcji, takich jak rezerwacja łodzi, zarządzanie kontem, przegląd historii transakcji czy dostęp do szczegółowych informacji technicznych, jest zarezerwowany wyłącznie dla zalogowanych użytkowników.

Funkcja przeglądania listy floty bez logowania została wprowadzona, aby umożliwić potencjalnym użytkownikom zapoznanie się z ofertą przystani i podjęcie decyzji o ewentualnym założeniu konta lub zalogowaniu w celu skorzystania z pełnej funkcjonalności systemu



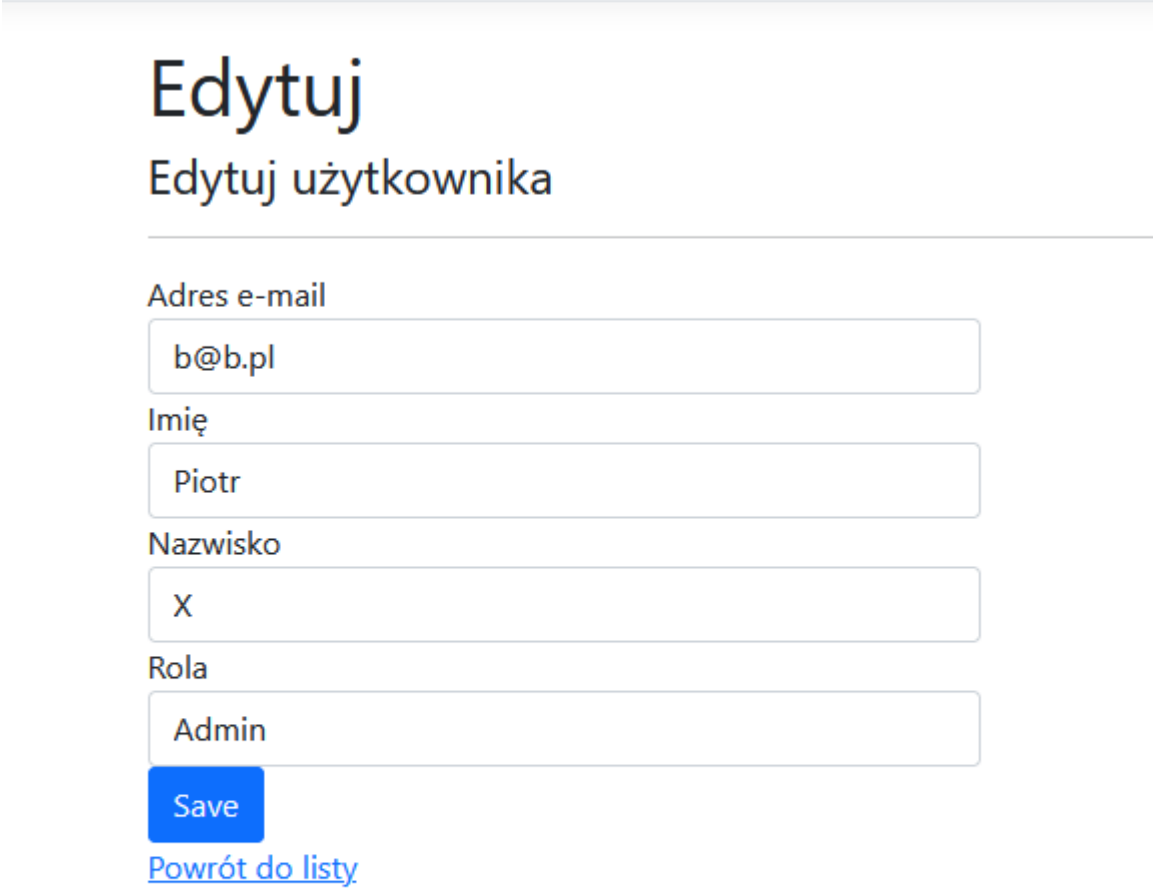
6.4. Nadawanie ról użytkownikowi (nadawanie uprawnień)

- Pierwszy użytkownik w systemie automatycznie otrzymuje rolę administratora (Admin)

- Każdy następny użytkownik w systemie automatycznie otrzymuje domyślnie najniższą rolę czyli Klient

- Zdefiniowano że w systemie użytkownik może mieć co najwyżej jedną rolę

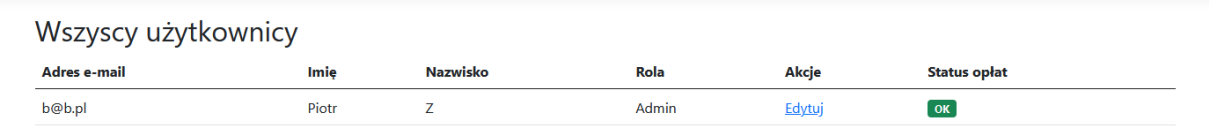
- Aby zmienić rolę poszczególnego użytkownika, należy zalogować się na konto administratora, z listy użytkowników wybrać edycję i zmienić rolę na pożądaną.



7. Najciekawsze funkcjonalności

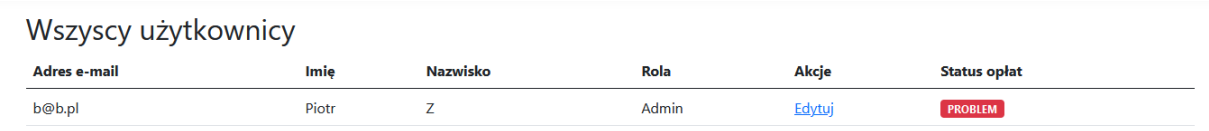
- System pokazuje status płatności opłat członkowskich i uniemożliwia potwierdzenie rezerwacji jeżeli wszystkie składki nie zostały opłacone

Krok 1 - Brak opłat członkowskich w systemie

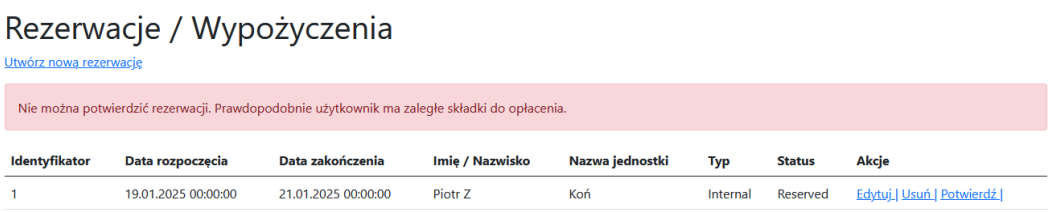


Krok 2 - Opłaty członkowskie dodane do systemu

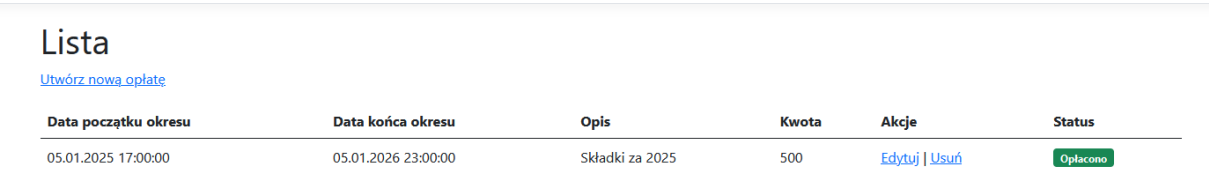
Krok 3 – Użytkownik nie opłacił opłaty członkowskiej więc ma status PROBLEM



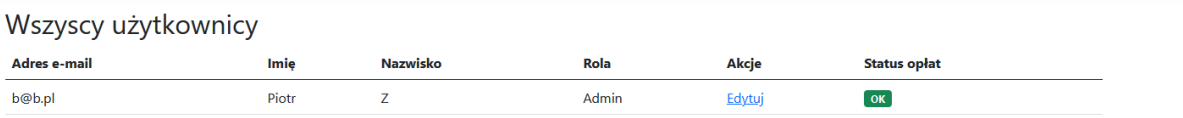
Krok 4 – Przy próbie potwierdzenia rezerwacji występuje błąd



Krok 5 – Użytkownik opłacił składkę



Krok 6 – Użytkownik opłacił opłatę członkowską więc ma status OK



Krok 7 – Rezerwacja przebiegła pomyślnie