Monika Wereszczyńska i Karolina Siwiak

I Informatyka

Magisterskie

Gr.3

Przedmiot: Multimedialne i obiektowe bazy danych

Projekt: Rezerwacja pokoi w hotelu

Rok akademicki 2020-21

Semestr letni

1. Koncepcja projektu:

Koncepcją naszego projektu jest serwis rezerwacji pokoju w hotelu.
Za jego pomocą będzie można sprawdzić:

- jakie pokoje są dostępne w konkretnym przedziale czasowym,
- dokonać rezerwacji pokoju,
- anulować rezerwację,
- dokonać edycji rezerwacji,
- sfinalizować rezerwację - zapłacić za wynajęcie pokoju przez gościa.

System będzie umożliwiał także wykonywanie operacji CRUD na poszczególnych pokojach w hotelu:

- dodanie pokoju,
- edycję danych pokoju,
- usunięcie pokoju z oferty hotelu.

Operacje CRUD będą mogły być przeprowadzane na poszczególnych rezerwacjach:

- zarezerwuj pokój,
- edytuj rezerwację,
- anuluj rezerwację,
- zapłać za rezerwację - tworzymy rachunek do danej rezerwacji (operacja CREATE obiekt Rachunek).

Użytkownikami mikrosystemu są pracownicy hotelu oraz klienci.
Pracownik hotelu będzie miał dostęp do wszystkich rachunków w bazie, rezerwacji, klientów i ich rezerwacji, a także do współpracowników i dokonanych przez nich rezerwacji.

2. Wymagania funkcjonalne

Nazwa wymagania	Aktorzy	Opis wymagania
Znajdź wolny pokój	Pracownik Klient	Sprawdzenie dostępności pokojów - zostanie zwrócona zostanie lista wolnych pokoi w określonym przedziale czasu.
Zarezerwuj pokój	Pracownik Klient	Rezerwacja pokoju na określony czas.
Edytuj rezerwację	Pracownik Klient	Edycja istniejącej rezerwacji - zmiana odstępu czasowego.
Anuluj rezerwację	Pracownik Klient	Anulowanie istniejącej rezerwacji.
Sfinalizuj rezerwację	Pracownik Klient	Zapłacenie za rezerwację - podsumowanie całego rachunku.

Lista pokoi	Pracownik Klient	Wyświetlanie wszystkich pokoi w systemie.
Dodaj pokój	Pracownik	Dodanie nowego pokoju do oferty hotelu.
Edytuj pokój	Pracownik	Edycja istniejącego pokoju.
Usuń pokój	Pracownik	Usunięcie istniejącego hotelu z oferty hotelu.
Twoje rezerwacje	Pracownik Klient	Zostaną wyświetlone wszystkie rezerwacje użytkownika. Pracownikowi zostanie wyświetlona lista rezerwacji, których dokonał osobiście w imieniu klienta.
Wszystkie rezerwacje w bazie	Pracownik	Zostaną wyświetlone wszystkie rezerwacje w systemie.
Wszystkie rachunki w bazie	Pracownik	Zostaną wyświetlone wszystkie rachunki w systemie.
Wszyscy klienci w bazie	Pracownik	Zostaną wyświetleni wszyscy klienci w systemie.
Wszyscy pracownicy w bazie	Pracownik	Zostaną wyświetleni wszyscy pracownicy w systemie.
Rezerwacje danego klienta	Pracownik	Zostaną wyświetlone wszystkie rezerwacje danego klienta.
Rezerwacje danego pracownika	Pracownik	Zostaną wyświetlone wszystkie rezerwacje, wykonane przez danego pracownika.
Rejestracja użytkownika	Przyszły klient	Tworzenie konta w systemie.

3. Wymagania niefunkcjonalne

Klient do rejestracji potrzebuje swojego niepowtarzalnego e-maila i hasła.
Logowanie do systemu za pomocą e-maila i hasła.
Interfejsem systemu jest strona www.
System jest dostępny w języku polskim.

4. Użyte technologie w projekcie

Baza danych	MongoDB
Pobieranie danych z bazy danych	C# .NET API
Interfejs	React
Protokół komunikacji	HTTP
Przewidywane żądania:	
GET	● Get(Pokoje), ● GetPokoj, ● getVacancy, ● getRachunek, ● getRezerwacja(All), ● getRezerwacje(Usera), ● GetRezerwacjeDetails(Usera), ● GetKlienci, ● GetPracownicy, ● GetUser.
POST	● Create(Pokoj), ● Create(Rezerwacja).
PUT	● Update(Pokoj), ● MakeBill(Rachunek), ● Cancel(Rezerwacja), ● Edit(Rezerwacja), ● Edit(User).
DELETE	● Delete(Pokoj).

5. Przewidywana struktura bazy danych

Nazwa tabeli	Pola w tabeli
Pokoje	● id, ● numer pokoju, ● nazwa pokoju, ● ile osób w pokoju, ● cena za noc.
Rezerwacje	● id, ● id pokoju, ● id klienta, ● id rezerwującego (może być sam klient lub przez pracownika), ● data rozpoczęcia pobytu, ● data zakończenia pobytu, ● przewidywany koszt, ● czy rezerwacja była edytowana, ● data ostatniej edycji, ● czy anulowana, ● czyOplacona.
Rachunki	● id, ● id rezerwacji, ● data zakończenia pobytu, ● koszt rezerwacji, ● czy uregulowany, ● data uregulowania rachunku.
Klienci	● id, ● nazwisko, ● imię, ● numer telefonu, ● e-mail.
Pracownicy	● id, ● nazwisko, ● imię, ● numer telefonu, ● e-mail.

6. Sposób logowania i zarządzania użytkownikami

Każdy klient będzie miał możliwość stworzenia swojego konta i wykonywania akcji, dostępnych dla klienta - przy wykonaniu 1 rezerwacji, jego dane zostaną zapisane w bazie MongoDB.
Każdy pracownik będzie miał możliwość utworzenia konta, ale musi on wybrać e-mail z bazy danych Pracownicy.

7. Podział prac

Czynność	Wykonawca
Wypełnienie baz danych	Wereszczyńska
Wygenerowanie projektu .NET API	Wereszczyńska
Wykonanie interfejsów w REACT	Siwiak
Integracja MongoDB, .NET API oraz REACT	Siwiak
Implementacja kontrolerów	Wereszyńska
Implementacja serwisów	Siwiak

8. Opis implementacji

Projekt został zaimplementowany w środowisku Visual Studio.
W pliku apsettings.json zostały określone źródła danych, wykorzystywanych przez system:

- Microsoft SQL Local Database - przechowuje dane użytkowników do logowania (linia 2 - 4),
- MongoDB (linia 5 - 12) - przechowuje informacje o :
    - klientach,
    - pracownikach,
    - rezerwacjach,
    - rachunkach,
    - pokojach.

Poniższy fragment kodu pochodzi z pliku, wspomnianego powyżej.

```
1. {
2.   "ConnectionStrings": {
3.     "DefaultConnection":
   "Server=(localdb)\\mssqllocaldb;Database=CoreDB;Trusted_Connectio
   n=True;MultipleActiveResultSets=true"
4.   },
5.   "MongoDBDatabaseSettings": {
6.     "PokojCollectionName": "pokoje",
```

```
    "RezerwacjaCollectionName": "rezerwacje",
    "RachunekCollectionName": "rachunki",
    "KlientCollectionName": "klienci",
      "PracownikCollectionName": "pracownicy",
      "ConnectionString": "mongodb://localhost:27017",
      "DatabaseName": "bazaDanych"
    }
```

Poniższa klasa MongoDBDatabaseSettings ułatwia pobieranie informacji do połączenia z bazą MongoDB i kolekcji, wewnątrz bazy.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProjectMongoDBReact.Models
{
    public class MongoDBDatabaseSettings : IMongoDBDatabaseSettings
    {
        public string PokojCollectionName { get; set; }
        public string RezerwacjaCollectionName { get; set; }
        public string RachunekCollectionName { get; set; }

        public string KlientCollectionName { get; set; }
        public string PracownikCollectionName { get; set; }

        public string ConnectionString { get; set; }
        public string DatabaseName { get; set; }
    }

    public interface IMongoDBDatabaseSettings
    {
        string PokojCollectionName { get; set; }
        string RezerwacjaCollectionName { get; set; }
        string RachunekCollectionName { get; set; }

        string KlientCollectionName { get; set; }
        string PracownikCollectionName { get; set; }

        string ConnectionString { get; set; }
        string DatabaseName { get; set; }
    }
}
```

Poniższy kod to fragment kodu klasy KlientService, której metody pobierają informacje z bazy danych MongoDB.

```
1.   private readonly IMongoCollection<Klient> _klient;
2.         private readonly IMongoCollection<Pracownik> _pracownik;
3.
4.         public KlientService(IMongoDBDatabaseSettings settings)
5.         {
6.             var client = new
   MongoClient(settings.ConnectionString);
7.             var database =
   client.GetDatabase(settings.DatabaseName);
8.
9.             _klient =
   database.GetCollection<Klient>(settings.KlientCollectionName);
10.               _pracownik =
   database.GetCollection<Pracownik>(settings.PracownikCollectionNam
   e);
11.           }
```

Poniższa metoda z klasy KlientService edytuje istniejące rekordy w bazie danych MongoDB w kolekcji “klienci”.

```
1. public void Edit(string id, Klient klient)
2. {
3.       _klient.ReplaceOne(p => p.Id == id, klient);
4. }
```

Poniższy kod przedstawia konstruktor w kontrolerze UserController - zadeklarowane zostają w nim obiekty KlientService i PracownikService, które poprzez właściwe metody tych obiektów, zwracają odpowiednie dane z MongoDB.

```
1. private readonly KlientService _klientService;
2.         private readonly PracownikService _pracownikService;
3.
4.         public UserController(KlientService klientService,
   PracownikService pracownikService)
5.         {
6.             _klientService = klientService;
7.             _pracownikService = pracownikService;
8.         }
```

Poniższy kod przedstawia metodę w kontrolerze UserController. Poprzez klasę _pracownikService oraz _klientService kontorler pobiera dane z MongoDB.

```
[HttpGet(Name = "GetUser")]
        [Route("GetUser/{name}")]
        public object GetUser(string name)
        {
            var p = _pracownikService.GetUser(name);

            if (p == null)
            {
                var k = _klientService.GetUser(name);

                    if (k == null)
                    {
                        return null;
                    }

                    return k;
                }

                return p;
            }
```

Do aplikacji REACT, żądania HTTP są wysyłane przez AXIOS i FETCH.

Poniższa metoda z pliku Pokoje.js - za pomocą FETCH - wysyła żądanie HTTP GET do kontrolera, aby otrzymać listę pokoi. Autoryzacja użytkownika odbywa się poprzez token.

```
async populatePokojData() {
    const token = await authService.getAccessToken();
    const response = await fetch('pokoj', {
      headers: !token ? {} : { 'Authorization': `Bearer ${token}`
  }
    });
    const data = await response.json();
      this.setState({ pokoje: data, loading: false });
    }
```

Poniższy fragment kodu z pliku Pokoje.js przedstawia wysyłanie żądania HTTP DELETE za pomocą AXIOS - ma ona doprowadzić do usunięcia pokoju z bazy danych. Autoryzacja użytkownika odbywa się poprzez token.

```
axios.delete(`/pokoj/Delete/` + idPokoj + '/' + user.name,
        {
            headers: {
                Authorization: `Bearer ${token}`
            },
```

```
6.              data: {
7.                  'id': idPokoj
8.              }
9.
10.           }).then(window.location.reload());
```