PLATFORMY PROGRAMISTYCZNE – PROJEKT

SPRAWOZDANIE

Kowalczyk Emilia 272539

Rojek Julia 272529

SPIS TREŚCI

1. WSTĘP

1.1 Cel aplikacji

Celem aplikacji Rezerwacje Sal jest umożliwienie użytkownikom zarządzania rezerwacjami salek konferencyjnych w sposób intuicyjny i efektywny. Aplikacja pozwala na przeglądanie dostępnych sal, ich edycję, a także dodawanie nowych rezerwacji. Dzięki integracji z Google Maps API użytkownicy mogą w prosty sposób zlokalizować wybraną salę.

Aplikacja została zaprojektowana jako system **klient-serwer**, gdzie frontend działa jako aplikacja **WPF na Windows**, a backend funkcjonuje na **serwerze Linux**, napisany w **Node.js** i obsługujący bazę danych **MariaDB**. System zapewnia łatwość użytkowania, szybką komunikację z serwerem oraz przejrzysty interfejs użytkownika.

1.2 Opis projektu

Projekt składa się z dwóch głównych części:

- **Aplikacja kliencka (frontend)** – stworzona w technologii **Windows Presentation Foundation (WPF)**, umożliwia użytkownikom interakcję z systemem poprzez wygodny interfejs graficzny.
- **Serwer aplikacyjny (backend)** – napisany w **Node.js**, działa na serwerze Linux i obsługuje wszystkie zapytania klientów. Backend komunikuje się z bazą danych, przechowując informacje o salach oraz rezerwacjach.

Główne funkcjonalności aplikacji:

✅ Przeglądanie dostępnych sal wraz z ich szczegółami (nazwa, adres, pojemność, opis).

✅ Edycja istniejących sal (zmiana nazwy, lokalizacji, liczby miejsc itp.).

✅ Dodawanie nowych rezerwacji.

✅ Integracja z Google Maps API – umożliwia podgląd lokalizacji sali oraz edycję jej położenia na mapie.

✅ Komunikacja z backendem poprzez REST API oparte na Node.js.

✅ Obsługa i walidacja danych użytkownika.

✅ Instalacja aplikacji na systemie Windows za pomocą Inno Setup.

Aplikacja została zaprojektowana tak, aby była łatwa w użytkowaniu i mogła być rozszerzana o dodatkowe funkcjonalności w przyszłości.

1.3 Technologie użyte w projekcie

W projekcie wykorzystano następujące technologie:

Frontend (aplikacja kliencka)

- **Język:** C#
- **Framework:** Windows Presentation Foundation (**WPF**)
- **Obsługa map: WebView2** + **Google Maps API**
- **Instalacja aplikacji: Inno Setup**

Backend (serwer aplikacyjny)

- **Język:** JavaScript (**Node.js**)
- **Framework:** Express.js
- **Baza danych:** MariaDB
- **Obsługa żądań HTTP:** REST API

Dodatkowe narzędzia

- **Postman** – testowanie API backendu
- **Git** – system kontroli wersji
- **Visual Studio** – środowisko programistyczne
- **Swagger + Doxygen** – dokumentacja techniczna
- **Inno Setup Compiler** – tworzenie instalatora

2. OPIS APLIKACJI

2.1 Architektura aplikacji

Aplikacja **Rezerwacje Sal** została zaprojektowana w modelu **klient-serwer**, gdzie część kliencka działa jako aplikacja desktopowa **WPF**, a backend jako **REST API** uruchomione na serwerze **Linux**. Komunikacja pomiędzy frontendem a backendem odbywa się za pomocą protokołu **HTTP**, a dane są przechowywane w bazie danych **MariaDB**.

2.1.1 Frontend

Aplikacja kliencka została stworzona w technologii **Windows Presentation Foundation (WPF)** w języku **C#**. Jest odpowiedzialna za interakcję użytkownika z systemem oraz obsługę rezerwacji.

Kluczowe elementy frontend-u:

- **WPF** – interfejs użytkownika, formularze, walidacja danych.
- **WebView2** – do integracji z **Google Maps API**.
- **Komunikacja z backendem** – poprzez **HTTP REST API**.
- **Obsługa błędów** – komunikaty dla użytkownika w przypadku błędów API.
- **Instalacja aplikacji** – realizowana za pomocą **Inno Setup**.

2.1.2 Backend

Backend został napisany w języku **JavaScript** przy użyciu **Node.js** oraz frameworka **Express.js**. Odpowiada za logikę biznesową, obsługę żądań HTTP oraz komunikację z bazą danych.

Technologie backendu:

- **Node.js + Express.js** – obsługa REST API.
- **Baza danych: MariaDB** – przechowywanie informacji o salach i rezerwacjach.
- **Biblioteki do komunikacji z bazą:** mysql2 (obsługa MariaDB w Node.js).

- **Obsługa zapytań HTTP** – axios, express.json().

2.2 Funkcjonalności aplikacji

Aplikacja **Rezerwacje Sal** oferuje następujące funkcjonalności:

✅ **Przeglądanie listy sal** – użytkownik może zobaczyć wszystkie dostępne sale, ich nazwy, adresy oraz pojemność.
✅ **Edycja sali** – możliwość zmiany nazwy, lokalizacji (za pomocą **Google Maps API**), liczby miejsc oraz opisu.
✅ **Dodawanie nowych rezerwacji** – użytkownik może zarezerwować wybraną salę na określony termin.
✅ **Obsługa bazy danych** – dane są przechowywane w **MariaDB**, a aplikacja komunikuje się z serwerem przez REST API.
✅ **Integracja z Google Maps** – użytkownik może zobaczyć lokalizację sali na mapie oraz edytować jej położenie poprzez przesunięcie znacznika.
✅ **Obsługa błędów i walidacja danych** – aplikacja sprawdza poprawność wprowadzonych danych oraz obsługuje błędy połączenia z serwerem.
✅ **Instalacja aplikacji** – aplikacja jest dostarczana w formie instalatora utworzonego przy pomocy **Inno Setup**.

3. INTERFEJS UŻYTKOWNIKA

3.1 Technologie UI

Aplikacja kliencka została zbudowana w technologii **WPF (Windows Presentation Foundation)**, co umożliwia dynamiczne zarządzanie interfejsem użytkownika oraz integrację z dodatkowymi komponentami, takimi jak **WebView2** do obsługi Google Maps.

Wykorzystane technologie i biblioteki:

- **WPF (Windows Presentation Foundation)** – interfejs użytkownika.
- **XAML** – język deklaratywny do definiowania układu UI.
- **MVVM (Model-View-ViewModel)** – wzorzec architektoniczny używany do separacji logiki biznesowej od interfejsu.
- **WebView2** – kontrolka do osadzenia przeglądarki **Microsoft Edge** w aplikacji WPF.
- **Google Maps API** – do wyświetlania mapy i edycji lokalizacji sal.

3.2 Obsługa formularzy i walidacji danych

- **Walidacja po stronie klienta:** Wprowadzone dane są sprawdzane przed wysłaniem do serwera (np. poprawność formatu e-maila, wymagane pola).
- **Walidacja po stronie serwera:** Express Validator sprawdza dane przed zapisaniem do bazy (np. unikalność e-maila, poprawność współrzędnych geograficznych).

3.3 Obsługa mapy Google Maps w WebView2

Aplikacja wykorzystuje **WebView2** do wyświetlania map Google Maps, co umożliwia użytkownikowi interaktywną edycję lokalizacji sali.

Jak działa integracja?

1. Aplikacja pobiera klucz API Google Maps z pliku konfiguracyjnego App.config.

2. Po otwarciu okna edycji sali, WebView2 wczytuje stronę HTML zawierającą kod **Google Maps API**.
3. Na mapie pojawia się znacznik (marker) reprezentujący lokalizację sali.
4. Użytkownik może przesuwać znacznik, a nowe współrzędne są wysyłane do aplikacji za pomocą window.chrome.webview.postMessage().
5. Współrzędne są aktualizowane w polach formularza w aplikacji WPF.

Dzięki tej implementacji użytkownik może **interaktywnie ustawić lokalizację sali**, co znacząco ułatwia zarządzanie danymi.

4. BACKEND I BAZA DANYCH

4.1 Technologie backendowe

Backend aplikacji został zbudowany przy użyciu **Node.js** oraz frameworka **Express.js**, który zapewnia obsługę REST API. Do przechowywania danych wykorzystano bazę **MariaDB**, która umożliwia efektywne zarządzanie informacjami o użytkownikach, salach oraz rezerwacjach.

Zastosowane technologie backendowe:

- **Node.js** – środowisko uruchomieniowe dla języka JavaScript, wykorzystywane do obsługi serwera.
- **Express.js** – lekki framework webowy ułatwiający tworzenie REST API.
- **MariaDB** – relacyjna baza danych przechowująca informacje o użytkownikach, salach i rezerwacjach.
- **bcrypt** – szyfrowanie haseł użytkowników.
- **Swagger UI** – dokumentacja API dostępna pod /api-docs.

4.2 Struktura API i obsługiwane endpointy

Backend udostępnia **REST API**, które pozwala na wykonywanie operacji CRUD na użytkownikach, salach oraz rezerwacjach. Dokumentacja API jest generowana automatycznie przez **Swagger UI** i dostępna pod /api-docs.

Poniżej znajduje się tabela z obsługiwanymi endpointami:

AUTORYZACJA I UŻYTKOWNICY

Metoda	Endpoint	Opis
POST	`/api/auth/register`	Rejestracja nowego użytkownika
POST	`/api/auth/login`	Logowanie użytkownika
GET	`/api/users`	Pobranie listy użytkowników (admin)
DELETE	`/api/users/:id`	Usunięcie użytkownika (admin)

ZARZĄDZANIE SALAMI

Metoda	Endpoint	Opis
GET	`/api/rooms`	Pobranie listy wszystkich sal
GET	`/api/rooms/:id`	Pobranie szczegółów konkretnej sali
POST	`/api/rooms`	Dodanie nowej sali
PUT	`/api/rooms/:id`	Edycja informacji o sali
DELETE	`/api/rooms/:id`	Usunięcie sali

ZARZĄDZANIE REZERWACJAMI

Metoda	Endpoint	Opis
GET	`/api/reservations/occupied/:room_id`	Pobranie zajętych terminów dla danej sali
GET	`/api/reservations/user/:email`	Pobranie rezerwacji użytkownika
POST	`/api/reservations`	Tworzenie nowej rezerwacji
DELETE	`/api/reservations/:id`	Usunięcie rezerwacji

4.3 Opis bazy danych

Baza danych została zaprojektowana w **MariaDB** i składa się z trzech głównych tabel: **users**, **rooms** oraz **reservations**.

TABELA USERS – PRZECHOWUJE DANE UŻYTKOWNIKÓW

Kolumna	Typ danych	Opis
`id`	INT (PK, AUTO_INCREMENT)	Unikalny identyfikator użytkownika
`email`	VARCHAR(255) UNIQUE	Adres e-mail użytkownika
`password`	VARCHAR(255)	Zaszyfrowane hasło
`role`	ENUM('user', 'admin')	Rola użytkownika

TABELA ROOMS – PRZECHOWUJE INFORMACJE O SALACH KONFERENCYJNYCH

Kolumna	Typ danych	Opis
`id`	INT (PK, AUTO_INCREMENT)	Unikalny identyfikator sali
`name`	VARCHAR(100)	Nazwa sali

Kolumna	Typ danych	Opis
address	TEXT	Adres sali
seats	INT	Liczba miejsc
latitude	DECIMAL(9,6)	Szerokość geograficzna
longitude	DECIMAL(9,6)	Długość geograficzna

TABELA RESERVATIONS – PRZECHOWUJE REZERWACJE UŻYTKOWNIKÓW

Kolumna	Typ danych	Opis
id	INT (PK, AUTO_INCREMENT)	Unikalny identyfikator rezerwacji
user_id	INT (FK → users.id)	Identyfikator użytkownika
room_id	INT (FK → rooms.id)	Identyfikator sali
date	DATE	Data rezerwacji
time	TIME	Godzina rezerwacji

5. TESTOWANIE API

Testowanie API zostało przeprowadzone za pomocą narzędzia **Postman**, które umożliwia ręczne wysyłanie zapytań HTTP do serwera oraz analizę odpowiedzi. Celem testów było sprawdzenie poprawności działania endpointów backendu, obsługi błędów, walidacji danych oraz odpowiednich kodów statusu HTTP. Przetestowano wszystkie najważniejsze endpointy aplikacji.

Wnioski z testowania

- Wszystkie endpointy działają poprawnie przy podaniu prawidłowych danych.
- Błędy użytkownika (np. brak wymaganych pól) są poprawnie wychwytywane przez walidację i skutkują odpowiednimi kodami statusu HTTP (np. 400, 404).
- Odpowiedzi serwera są w formacie JSON i zawierają potrzebne dane lub komunikaty błędów.
- Testy potwierdziły, że backend jest gotowy do integracji z aplikacją kliencką.

6. GENEROWANIE DOKUMENTACJI

6.1 Dokumentacja API backendu

Dokumentacja API backendu została wygenerowana przy pomocy **Swagger UI**. Zawiera opis wszystkich dostępnych endpointów, metod HTTP oraz schematy danych, które są przesyłane i zwracane przez serwer.

http://95.215.232.175:5001/api-docs/

6.2 Dokumentacja projektu w C#

Dokumentacja projektu aplikacji klienckiej WPF została wygenerowana przy użyciu komentarzy XML oraz narzędzi do przetwarzania dokumentacji.

\docs\html\index.html

7. INSTALACJA I KONFIGURACJA

Aby umożliwić łatwą instalację aplikacji na komputerach użytkowników, przygotowano **instalator** przy użyciu **Inno Setup**. Instalator automatyzuje proces kopiowania plików aplikacji, tworzenia skrótów oraz konfiguracji wymaganych zależności.

8. PODSUMOWANIE

Aplikacja **Rezerwacje Sal** została zaprojektowana jako system do zarządzania rezerwacjami sal konferencyjnych. Składa się z aplikacji klienckiej **WPF** oraz backendu działającego na **Node.js** i korzystającego z **MariaDB**.

8.1 Osiągnięcia projektu

✅ **Stworzenie aplikacji WPF** umożliwiającej zarządzanie salami i rezerwacjami.
✅ **Backend w Node.js + Express.js** zapewniający REST API do obsługi żądań.
✅ **Baza danych MariaDB**, przechowująca informacje o użytkownikach, salach i rezerwacjach.
✅ **Dokumentacja API w Swagger UI** (/api-docs/).
✅ **Integracja z Google Maps API** przy użyciu WebView2.
✅ **Instalator Inno Setup**, ułatwiający wdrażanie aplikacji na komputerach użytkowników.

8.2 Możliwe kierunki rozwoju

🔷 **Obsługa rezerwacji cyklicznych** – możliwość powtarzających się rezerwacji (np. co tydzień).
🔷 **Powiadomienia e-mail/SMS** – informowanie użytkowników o nadchodzących rezerwacjach.
🔷 **Dodatkowe opcje filtrowania sal** – np. według dostępności sprzętu multimedialnego.
🔷 **Mobilna wersja aplikacji** – stworzenie aplikacji na Android/iOS.

8.3 Wyzwania i napotkane problemy

Instalacja WebView2 – początkowe problemy z jego obsługą w WPF.
Obsługa błędów w API – zapewnienie poprawnych kodów odpowiedzi HTTP i walidacji danych.

Dzięki wdrożonym rozwiązaniom aplikacja jest w pełni funkcjonalna i gotowa do wdrożenia w środowisku produkcyjnym.

9. ZAŁĄCZNIKI

Kod źródłowy aplikacji znajduje się w repozytorium projektu:

- **Frontend (WPF, C#)**: RezerwacjeSal/
- **Instalator:** installer.exe
- Link do repozytorium Git (jeśli dostępne):
  GitHub: https://github.com/afra50/System-zarzadzania-rezerwacjami-sal
- **Dokumentacja API backendu** (Swagger UI) http://95.215.232.175:5001/api-docs/

- **Dokumentacja XML dla C#** (automatycznie wygenerowana w Visual Studio): RezerwacjeSal/bin/Release/net8.0-windows/RezerwacjeSal.xml
- **Dokumentacja projektu** (HTML): docs/html/index.html