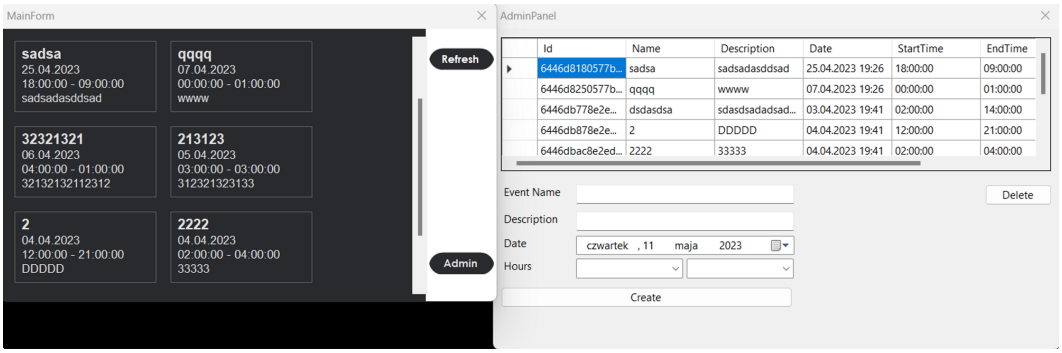
Dokumentacja terminarz



Zadania postawione systemowi

Funkcjonalności programu

- Przeglądanie harmonogramu
- Dodawanie wydarzeń
- Usuwanie wydarzeń
- Autoryzacja

Projekt systemu

Realizacja systemu

- Interfejs użytkownika
- Warstwa biznesowa
- Baza danych

Rozbieżności

Zadania postawione systemowi

Terminarz pozwalający użytkownikowi na przeglądanie i zarządzanie swoim harmonogramem poprzez tworzenie i modyfikowanie listy zadań.
Ogólne wymagania klienta:

- Aplikacja powinna umożliwiać dodawanie, usuwanie i edycje wydarzeń
- Użytkownik ma możliwość przeglądania swoich wydarzeń w formie kalendarza
- Użytkownik może ustawiać przypomnienia do poszczególnych wydarzeń
- Aplikacja powinna umożliwiać zmienianie statusu wydarzeń na

wykonane, oczekujące lub przesunięte na inny termin

Funkcjonalności programu

Terminarz zawiera następujące funkcje:

Przeglądanie harmonogramu

Użytkownik może przeglądać terminarz według dni, tygodni i miesięcy. Program zapewnia wizualną reprezentacje harmonogramu co ułatwia nawigacje.

Dodawanie wydarzeń

Użytkownik może dodawać wydarzenia do swojego terminarza przez wprowadzenie daty, czasu i opisu wydarzenia. Może ustawić również przypomnienie dla zbliżających się wydarzeń.

fragment kody odpowiadający za dodawanie wydarzeń do bazy:

```
// Dodawanie eventu public async Task CreateEventAsync(Event
eventObj) { await _events.InsertOneAsync(eventObj);
```

używa stworzonego wcześniej obiektu “event”

```
// Obiekt Eventu public class Event { public ObjectId Id { get;
set; } public string Name { get; set; } public string
Description { get; set; } public DateTime Date { get; set; }
public TimeSpan StartTime { get; set; } public TimeSpan EndTime
{ get; set; } public string AdministratorEmail { get; set; } }
```

tworzenie obiektu event:

```
// Utwórz obiekt eventu var newEvent = new Event { Name =
eventName, Description = description, Date = eventDate,
```

```
StartTime = TimeSpan.Parse(fromHour), EndTime =
TimeSpan.Parse(toHour), AdministratorEmail =
_administrator.Email }
```

Usuwanie wydarzeń

Użytkownik może usuwać wydarzenia z terminarza, gdy przestają być potrzebne

fragment kodu odpowiadający za usuwanie wydarzeń:

```
// Usuwanie eventu public async Task DeleteEventAsync(ObjectId
id) { await _events.DeleteOneAsync(e => e.Id == id); }
```

Autoryzacja

Program wymaga logowania przed przydzieleniem dostępu do harmonogramu. Dzięki temu tylko autoryzowani użytkownicy mogą zarządzać harmonogramem. Autoryzacja działa na zasadzie unikalnego tokenu, który przy pomocy narzędzia Sendinblue jest wysyłany na adres e-mail podany przez użytkownika

Generowanie tokenu

Przebieg autoryzacji/tworzenie użytkownika

Projekt systemu

Projekt napisany językiem programowania C# z wykorzystaniem platformy .NET Framework

Użyte narzędzia i biblioteki:

- MongoDB - baza danych do przechowywania użytkowników i wydarzeń
- MongoDB.Driver - Umożliwia wykonywanie operacji na bazie danych w aplikacji C#
- Visual Studio - zintegrowane środowisko projektowe

- Visual Studio - zintegrowane środowisko projektowe
- Windows Forms - interfejsy graficzne

Realizacja systemu

projekt składa się z poszczególnych warstw

Interfejs użytkownika

interfejs graficzny właściciela terminarza zawiera:

- przedstawienie graficzne zaplanowanych wydarzeń razem z opisami
- wybór interesującego nas zakresu godzin wydarzeń
- przycisk dodawania wydarzeń
- przycisk usuwania wydarzeń
- przycisk “refresh”, który ładuje wydarzenia od nowa

Warstwa biznesowa

Warstwa biznesowa zawiera klasę Event, która służy do reprezentowania wydarzenia i przechowywania jego atrybutów, takich jak nazwa, opis, data i godzina rozpoczęcia i zakończenia.

```
private async void btnCreateEvent_Click(object sender, EventArgs
e) { string eventName = txtEventName.Text; string description =
txtDescription.Text; DateTime eventDate = dtpEventDate.Value;
string fromHour = cbFromHour.SelectedItem?.ToString(); string
toHour = cbToHour.SelectedItem?.ToString();
```

Wciskając przycisk dodawania wydarzenia aplikacja sprawdza czy wszysktie potrzebne dane zostały odpowiednio wprowadzone

```
if (string.IsNullOrEmpty(eventName) ||
string.IsNullOrEmpty(description) || fromHour == null || toHour
== null) { MessageBox.Show("Uzupełnij wszystkie pola!", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error); return;
```

funkcja dodawania wydarzenia:

```
public async Task CreateEventAsync(Event eventObj) { await
_events.InsertOneAsync(eventObj); }
```

po dodaniu wydarzenia widok terminarza zostaje zauktualizowany

```
await LoadEventsAsync();
```

Chcąc usunąć wydarzenie należy zaznaczyć jego pole i kliknąć przycisk usuwania. Program sprawdza czy pole zostało zaznaczone, następnie po ponownym potwierdzeniu usuwa wydarzenie z bazy danych i aktualizuje wygląd formularza

```
private async void btnDeleteEvent_Click(object sender, EventArgs e
czy cokolwiek zaznaczyłeś if (dataGridViewEvents.SelectedRows.Coun
MessageBox.Show("Zaznacz pola do usunięcia!", "Error", MessageBoxBu
MessageBoxIcon.Error); return; } DialogResult result = MessageBox.
pewny że chcesz usunać ten event?", "Confirmation", MessageBoxButt
MessageBoxIcon.Warning); // Jeśli tak to usuń if (result == Dialog
// Pobierz id zaznaczonego eventu var eventId =
ObjectId.Parse(dataGridViewEvents.SelectedRows[0].Cells["Id"].Valu
// Użyj gotowej funkcji do usunięcia await
_eventService.DeleteEventAsync(eventId); // zaktualizuj widok awai
LoadEventsAsync(); }
```

funkcja usuwania wydarzenia:

```
public async Task DeleteEventAsync(ObjectId id) { await
_events.DeleteOneAsync(e => e.Id == id); }
```

funckja pobierająca wszystkie wydarzenia z bazy danych:

```
public async Task<IEnumerable<Event>> GetAllEventsAsync() {
return await _events.Find(e => true).ToListAsync();
```

Baza danych

Warstwa bazy danych jest odpowiedzialna za przechowywanie danych związanych z użytkownikami oraz wydarzeniami. W projekcie wykorzystano bazę danych MongoDB.

W bazie danych przechowywane są kolekcje "Users" oraz "Events". Kolekcja "Users" zawiera informacje o zarejestrowanych użytkownikach - adres e-mail, przypisany im token oraz informację czy są właścicielami terminarza. Kolekcja "Events" przechowuje dane o utworzonych wydarzeniach, w tym nazwę, opis, datę, godzinę rozpoczęcia i zakończenia, a także adres email administratora, który utworzył wydarzenie.

Przykładowy fragment kodu odpowiedzialny za połączenie z bazą danych:

```
public UserService(string connectionString, string databaseName,
string collectionName) { var client = new
MongoClient(connectionString); var database =
client.GetDatabase(databaseName); _users =
database.GetCollection<User>(collectionName); }
```

Rozbieżności

Funkcjonalności, które ostatecznie nie zostały zaimplementowane:

- ustawianie przypomnień do poszczególnych wydarzeń
- edycja istniejących wydarzeń
- połączenie z API

Projekt udało się ukończyć i okazać w określonym terminie 15.05.2023.