F.S.O Strefa Wydarzeń F.S.O

Strefa Wydarzeń to moduł służący do organizacji i zarządzania wydarzeniami, zaprojektowany jako część większej platformy dla mieszkańców osiedla F.S.O Park w Warszawie.

Krótki opis projektu

Osiedle to w przyszłości stanie się domem dla około 25 tysięcy osób, a celem platformy jest unowocześnienie codziennych czynności mieszkańców. Moduł wydarzeń wspiera integrację mieszkańców oraz ułatwia organizowanie aktywności w ramach społeczności osiedlowej.
Jego główne funkcje obejmują:

Tworzenie wydarzeń lokalnych: mieszkańcy mogą organizować różnorodne aktywności, takie jak sąsiedzkie grille, warsztaty, zajęcia sportowe czy spotkania tematyczne,

Planowanie wydarzeń ogólnych: umożliwia inicjowanie wspólnych wyjść na większe wydarzenia miejskie, takie jak koncerty lub Juwenalia,

Rekomendacje: moduł dostarcza personalizowane sugestie wydarzeń na podstawie poprzednich wydarzeń. Cała aplikacja ma za zadanie finalnie zastąpić widoczne nawet na nowych osiedlach tablice korkowe służące za główny punkt informacji.

Ciekawe rozwiązanie

Naszym celem było poprawne zaimplementowanie systemu do oceny wydarzeń, na które użytkownik reagował, następnie na podstawie tych danych, wyświetlanie mu wydarzeń od tych, które lubi najbardziej po wydarzenia, które są mu obojętne, kończąc na tych, których nie lubi. System na podstawie wciśnięcia przycisku w panelu wydarzeń dodaje bądź modyfikuje wpis w bazie danych, następnie na podstawie wszystkich zebranych o użytkowniku danych, tworzy słownik z punktacją danych tagów. Następuje pobranie danych o wydarzeniach oraz zliczenie ich punktacji na podstawie słownika. Wydarzenia następnie są sortowane od najbardziej punktowanego, do tego najmniej. Operacje wykonywane są za każdym uruchomieniem panelu wydarzeń, dlatego zareagowanie na jakiekolwiek wydarzenie, zmienia jego punktacje oraz o ile zaszła różnica, miejsce w liście wyświetlanej.

Specyfikacja wykorzystanych technologii

Projekt został napisany w języku C#, oparto go na frameworku ASP.NET Core 8.0. ORM to Entity Framework, za przekazywanie danych do do frontendu odpowiada Razor Pages. Za bazę danych służy SQL Server od Microsoft.

Opis ról użytkowników

Obecnie w projekcie są zaimplementowane dwie role, użytkownik oraz administrator. Użytkownikiem jest każdy zalogowany user, rola jest dodawana automatycznie w procesie rejestracji. Drugą rolą jest administrator systemowy, który ma dostęp do wszystkich funkcjonalności, dodawanie roli nie jest zaimplementowane, administrator jest jeden, każdorazowe uruchomienie projektu wywołuje weryfikację istnienia konta oraz w przypadku braku, dodanie go z odpowiednimi uprawnieniami. Informacja o haśle oraz adresie mailowym dostępna jest w program.cs. Jest to rozwiązanie zaimplementowane w celu pokazania funkcjonalności i w rozwiązaniu rozbudowanym o kolejne moduły, planowane jest dodanie systemu umożliwiającego nadanie użytkownikom roli administratora z poziomu aplikacji.

Instrukcja pierwszego uruchomienia projektu

W celu uruchomienia projektu należy projekt pobrać z repozytorium Git, następnie pliki umieścić w folderze /sources/repos/*nazwa folderu*. W Visual Studio należy wybrać opcję służącą do otwarcia istniejącego już projektu i wskazać mu plik FSO.sln. Po uruchomieniu należy uruchomić NuGet'a i zainicjować migrację bazy danych poleceniem Update-Database.

Opis struktury projektu

Areas: Zawiera ukryte domyślnie pliki obsługujące autentykację użytkowników, odkryte w celu przeprowadzenia spolszczenia,

Attributes: Zawiera specyficzne atrybuty do walidacji danych w bazie danych,

Controllers: Zawiera logikę kontrolerów odpowiadających za komunikację,

Data: Zawiera klasę kontekstu bazy danych,

Migrations: Przechowuje migracje bazy danych,

Models: Zawiera logikę biznesową oraz walidacji bazy danych,

Properties: Zawiera konfiguracje aplikacji,

Views: Zawiera widoki Razor Pages do obsługi interfejsu użytkownika.

Wylistowane modele

ErrorViewModel:
Domyślny model do obsługi wystąpień błędów

Event:
Reprezentuje wydarzenie w systemie

Pola:
Id: klucz główny,
Name: Tytuł wydarzenia z walidacją w postaci 5-50 znaków oraz wymaganym wypełnieniem,

Description: Opis wydarzenia z walidacją w postaci 30-250 znaków oraz wymaganym wypełnieniem,
isPublic: Zmienna boolowska obsługująca wskazanie, czy wydarzenie jest publiczne czy prywatne,
EventTypeId: Klucz obcy do obsługi rodzajów wydarzeń,
LocationId: Klucz obcy do obsługi rodzajów wydarzeń,
Start: Data początkowa wydarzenia,
End: Data końcowa wydarzenia, walidacja obu zmiennych na podstawie atrybutu TimeDifference, który dba o to, aby wydarzenie nie trwało dłużej niż tydzień, aby data startowa i końcowa były prawidłowo ustawione oraz aby nie umieszczać zbyt starych wydarzeń,
CreatorId: Zapisanie osoby, która wydarzenie dodała.

EventTag:
Model pomocniczy do obsługi relacji wiele do wielu między wydarzeniami, a tagami.

Pola:
Id: Klucz główny,
EventId: Klucz obcy dla wydarzeń,
TagId: Klucz obcy dla tagów.

EventType:
Model obsługujący strukturę kategorii danych wydarzeń.

Pola:
Id: Klucz główny,
Name: Nazwa kategorii.

EventView:
Model pomocniczy, którego zadaniem jest obsługa formularza dodawania wydarzeń, w którym to dodajemy tagi. Przez skorzystanie z relacji wiele do wielu, podłączenie właściwości inicjalizujących do modelu Event nie było w naszym odczuciu przejrzyste i łatwiejszym finalnie okazało się stworzenie modelu asystującego w kontekście tworzenia wydarzeń.

Pola:
Id: Klucz główny,
Event: Obiekt modelu reprezentujący wydarzenie,
AvailableTags: Lista dostępnych tagów,
SelectedTagIds: Lista wybranych przez użytkownika tagów z walidacją w postaci minimalnie 1 wybranego tagu oraz samej konieczności jego wyboru.

Location:
Model reprezentujący lokalizacje dla wydarzeń.

Pola:
Id: Klucz główny,
Name: Nazwa główna lokalizacji z walidacją w postaci 5-50 znaków oraz wymaganym wypełnieniem,
Position: Szczegóły dla lokalizacji z walidacją w postaci 5-50 znaków oraz wymaganym wypełnieniem,
lat: Szerokość geograficzna dla lokalizacji ze szczególnym typem tj. decimal dla poprawnej obsługi danych, wymaganym wypełnieniem oraz zakresem dla wyboru wydarzeń tylko i wyłącznie w zakresie miasta Warszawy,
lon: Długość geograficzna dla lokalizacji ze szczególnym typem tj. decimal dla poprawnej obsługi danych, wymaganym wypełnieniem oraz zakresem dla wyboru wydarzeń tylko i wyłącznie w zakresie miasta Warszawy.

Participants:
Model zawierający informacje na temat wyborów użytkowników w kontekście obecności na wydarzeniach. Nie zawiera walidacji, gdyż użytkownik bezpośrednio nie ma dostępu do wprowadzania danych innych niż przyciski dostępne przy wydarzeniach.

Pola:
Id: Klucz główny,
EventId: Klucz obcy dla wydarzeń,
UserID: Zapisanie użytkownika reagującego,
Status: Wybrany rodzaj reakcji na dane wydarzenie.

Tag:
Model obsługujący tagi dla wydarzeń.

Pola:
Id: Klucz główny,
Name: Nazwa kategorii z walidacją w postaci zakresu 5-50 znaków oraz wymaganiem uzupełnienia.

Wylistowane kontrolery

Zgodnie z dobrymi praktykami, kontrolery kończą się na „Controller", dla przejrzystości opisu ta część nazwy zostanie pominięta.

Admin:
Bardzo prosty i intuicyjny kontroler, którego zadaniem jest przekazanie widoku dla użytkownika będącego administratorem. Widok ten zawiera nawigację do poszczególnych elementów w aplikacji.

Metody:
Index() (GET) – Służy do przekazania widoku.

Events:
Kontroler obsługujący kluczową funkcję programu jakimi są wydarzenia. Kontroler jest zabezpieczony przed dostępem dla osób niezalogowanych.

Metody:
Index() (GET) – Metoda zwracająca użytkownikowi posortowane według jego upodobań wydarzenia, które to następnie przekazywane są do widoku. Metoda na podstawie Id użytkownika pobiera dane o wszystkich wyborach użytkownika, następnie w kolejnym zapytaniu pobiera dla odpowiednich wydarzeń wszystkie tagi. Zebrane dane są sumowane w słownik zawierający tagi, co do których użytkownik ma jakąkolwiek relację oraz ich punktację. Kolejnym krokiem metody jest pobranie wszystkich wydarzeń, które nie zostały jeszcze zakończone, a następnie zgodnie z punktacją są one sortowane w liście i przekazywane do widoku.

Details(int? Id) (GET) – Metoda zwracająca użytkownikowi szczegółowe dane na temat przesłanego do kontrolera parametru Id. Metoda ta pobiera z bazy danych także dodatkowe informacje szczegółowe na temat lokalizacji, które są wykorzystywane do pokazania wydarzenia na interaktywnej mapie. Metoda zwraca widok oraz dodatkowo ustawiane są wartości dla ViewData dla kluczy Lat oraz Lon przechowywujące współrzędne geograficzne.

Create() (GET) – Metoda zwracająca widok z modelu EventView służąca do obsługi wyświetlania formularza. Pomocniczo wykorzystuje również ViewData dla kluczy EventTypeId oraz LocationId, aby obsłużyć w odpowiedni sposób wyświetlenie tagów wraz z nazwami w liście rozwijanej.

Create(EventView model) (POST) – Metoda obsługująca przesłanie danych użytkownika do bazy danych. Jego działanie jest bliskie standardowej strukturze z dodanymi instrukcjami warunkowymi, które ustalają, jaką wiadomość zwrotną wyświetlić użytkownikowi. Dodatkowo poza tym zdecydowaliśmy się na procesowanie prawidłowo walidowanego modelu w instrukcji warunkowej, ponieważ w przypadku niepowodzenia, tj. wystąpienia błędu ze strony użytkownika, ponownie przekazujemy listę tagów, kategorii oraz lokalizacji.

Edit(int? id) (GET) – Metoda z blokadą dostępu dla użytkowników w innej roli niż administrator. Służy do wyświetlenia danych w formularzu do edycji wydarzenia. Nie zawiera żadnych dodatkowych nietypowych funkcjonalności.

Edit(int? Id, [Bind("Id,Name,Description,EventTypeId,LocationId,Start,End,CreatorId")] Event @event) (POST) – Metoda obsługująca formularz edycji. Sposób działania jest identyczny do domyślnego, bez dodatkowych funkcjonalności.

Delete(int? id) (GET) – Metoda wyświetlająca formularz mający na celu finalne usunięcie wpisu, dostępna jedynie dla administratora. Nie jest ona zmodyfikowana w żaden sposób.

DeleteConfirmed(int id) (POST) – Metoda wykonująca operację usunięcia na podstawie int id rekordu z bazy danych, nie została zmodyfikowana w żaden sposób.

ReactToEvent(int eventId, int status) (POST) – Metoda wywoływana z widoku index event przyjmująca przekazaną przez użytkownika reakcję. Informacje zebrane są przekazywane do metody AddOrUpdateParticipants, która jest opisana niżej, następnie zwracany do użytkownika jest widok wraz z TempData dla klucza ConfirmationMessage informujący o dodaniu reakcji.

AddOrUpdateParticipants(int eventId, Guid userId, int status) – Metoda prywatna, dostęp z metody ReactToEvent, wybór przekazany przez użytkownika zostaje przekazany do bazy danych. Przed dodaniem następuje weryfikacja, czy użytkownik nie reaguje ponownie, tak aby nie pozwolić na podwojenie punktacji z jednego wydarzenia.

EventTags:

Kontroler obsługujący tylko i wyłącznie wyświetlanie eventu i przypisanego mu taga. Reszta operacji domyślnych została zablokowana przy użyciu return forbid(); stąd nie zostaną one wymienione.

Metody:

Index() (GET) – Metoda obsługująca wyświetlanie połączonej listy wydarzeń oraz odpowiadających im tagów. Forma metody nie została dodatkowo zmodyfikowana z racji na brak takiej konieczności

EventTypes:
Kontroler obsługujący operacje związane z rodzajami wydarzeń. Posiada tylko podstawowe metody kontrolera. Niektóre metody zostały wyłączone jak w poprzednim przypadku poprzez zwrócenie forbid(); i zostaną one pominięte w dokumentacji.

Metody:
Index() (GET) – Metoda obsługująca wyświetlanie kategorii dla wydarzeń. Forma metody nie została dodatkowo zmodyfikowana z racji na brak takiej konieczności

Create() (GET) – Metoda zwraca podstawowy widok formularza, nie została zmodyfikowana w żaden sposób.

Create([Bind("Id,Name")] EventType eventType) (POST) – Metoda obsługująca proces dodawania kategorii do bazy danych, forma została zachowana w pierwotnej postaci, bez modyfikacji.

Edit(int? Id) (GET) - Metoda służy do wyświetlenia danych w formularzu do edycji wydarzenia. Dostęp jest autoryzowany tylko dla administratora. Nie zawiera żadnych dodatkowych nietypowych funkcjonalności.

Edit(int id, [Bind("Id,Name")] EventType eventType) (POST) - Metoda obsługująca proces edycji kategorii w bazie danych, forma została zachowana w pierwotnej postaci, bez modyfikacji.

Delete(int? Id) (GET) – Metoda, której zadaniem jest przekazanie danych formularza do potwierdzenia usunięcia rekordu z bazy danych, metoda pozostała w formie niezmienionej. Metoda posiada ograniczenie dostępu pozwalającej na użycie jej jedynie dla administratora.

DeleteConfirmed(int id) (POST) – Metoda wykonująca operację usunięcia na podstawie int id rekordu z bazy danych, nie została zmodyfikowana w żaden sposób.

Home:
Kontroler domyślny do obsługi strony startowej frameworka ASP.NET.

Metody:
Index() (GET) – Metoda wyświetlająca stronę główną projektu. Do widoku przekazywana jest lista z pięcioma ostatnimi wydarzeniami publicznymi, które trwają, bądź są zaplanowane w przyszłości.

Privacy() (GET) – Metoda wyświetlająca widok strony z polityką prywatności, nie jest obecnie w użyciu.

Error() (GET) – Domyślna metoda zaimplementowana do obsługi błędów, forma nie została w żadnym stopniu zmieniona.

Locations:
Kontroler obsługujący żądania związane z lokalizacjami. Posiada on metody zwracające forbid(); z racji na brak ich wykorzystania, których to nie opiszę poniżej.

Metody:
Index() (GET) – Metoda do wyświetlania wszystkich dostępnych lokalizacji, metoda ta nie została zmodyfikowana.

Create() (GET) – Metoda wyświetlająca podstawowy formularz do dodania lokalizacji. Nie została ona w żadnym stopniu zmodyfikowana.

Create([Bind("Id,Name,Position,lat,lon")] Location location) (POST) – Metoda obsługująca formularz do dodawania lokalizacji do bazy danych. Został on zmodyfikowany o weryfikację dodania oraz położenia pinezki wskazującej lokalizacje. Walidacja ta jest przekazywana do TempDaty w kluczu ErrorMessage i wyświetlana użytkownikowi w widoku.

Edit(int? Id) (GET) – Metoda obsługująca edycję wybranej lokalizacji, poza domyślną strukturą metody, przekazywane do widoku są także w ViewData w kluczach Lat oraz Lon koordynaty, które to następnie zostają przekazane do skryptu JavaScript w celu ustawienia znacznika w miejscu, które jest wskazane w lokalizacji.

Edit(int id, [Bind("Id,Name,Position,lat,lon")]Location location) (POST) – Metoda edytująca wpis w bazie danych na podstawie przekazanych w formularzu danych. Nie jest ona dodatkowo zmodyfikowana.

Delete(int? Id) (GET) – Metoda przekazująca dane dot. Wpisu lokalizacji do usunięcia, jest to podstawowa metoda w jej naturalnej, niezmienionej formie. Posiada ona zabezpieczenie dostępu tylko i wyłącznie dla administratora.

DeleteConfirmed(int id) (POST) – Metoda wykonująca operację usunięcia na podstawie int id rekordu z bazy danych, nie została zmodyfikowana w żaden sposób.

Participants:
Kontroler obsługujący wybory użytkowników dotyczące stosunku do danych wydarzeń, jest to kontroler, do której metod może dostać się jedynie administrator. Niektóre metody są wyłączone z użytku, gdyż nie ma istotnych potrzeb ku trzymaniu ich uruchomionych, zwracają one forbid() i nie będą uwzględnione poniżej.

Metody:
Index() (GET) – Metoda do wyświetlania wszystkich wyborów wszystkich użytkowników. Metoda poza samym przekazaniem widoku, wykonuje także operację join na dwóch tabelach, tj. swojej, participants oraz events w celu poprawnego wyświetlenia nazw eventów. Metoda ta jest dostępna dla administratora oraz użytkownika i przekazuje odpowiednio na podstawie roli odpowiednie dane dla widoku. Administrator dostaje listę wszystkich użytkowników oraz ich wyborów, natomiast użytkownik widzi tylko i wyłącznie swoje wybory.

Create([Bind("Id,EventId,UserId,Status")] Participants participants) (POST) – Metoda służąca do dodania nowej wartości do bazy danych, nie jest ona w żaden sposób dodatkowo zmodyfikowana.

Tags:
Kontroler obsługujący żądania związane z tagami. Posiada on metody zwracające forbid(); z racji na brak ich wykorzystania, których to nie opiszę poniżej.

Metody:
Index() (GET) – Metoda do wyświetlania wszystkich dostępnych lokalizacji, metoda ta nie została zmodyfikowana.

Create() (GET) – Metoda wyświetlająca podstawowy formularz do dodania taga. Nie została ona w żadnym stopniu zmodyfikowana.

Create([Bind("Id,Name")] Tag tag) (POST) – Metoda obsługująca proces dodawania taga do bazy danych, forma została zachowana w pierwotnej postaci, bez modyfikacji.

Edit(int? Id) (GET) - Metoda służy do wyświetlenia danych w formularzu do edycji taga. Dostęp jest autoryzowany tylko dla administratora. Nie zawiera żadnych dodatkowych nietypowych funkcjonalności.

Edit(int id, [Bind("Id,Name")] Tag tag) (POST) - Metoda obsługująca proces edycji taga w bazie danych, forma została zachowana w pierwotnej postaci, bez modyfikacji.

Delete(int? Id) (GET) – Metoda, której zadaniem jest przekazanie danych formularza do potwierdzenia usunięcia rekordu z bazy danych, metoda pozostała w formie niezmienionej. Metoda posiada ograniczenie dostępu pozwalającej na użycie jej jedynie dla administratora.

DeleteConfirmed(int id) (POST) – Metoda wykonująca operację usunięcia na podstawie int id rekordu z bazy danych, nie została zmodyfikowana w żaden sposób.

Autorzy: Patryk Biazik, Jakub Brylczak