Politechnika Wrocławska

Zastosowanie informatyki w gospodarce

Aplikacja do rezerwacji miejsc w restauracjach

Hubert Duś
Jędrzej Kozal
Eliza Mocek
Piotr Montewka

prowadzący
Dr inż. Marek Woda

12 maja 2018

1 Wstęp

1.1 Cel projektu

Celem projektu realizowanego w ramach kursu, jest stworzenie aplikacji biznesowej, umożliwiającej rezerwacje miejsc w wybranych restauracjach. Zakłada się, że tworzona aplikacja będzie umożliwiała rezerwację miejsc w restauracji w porozumieniu z właścicielem i obsługą. Analogicznym pomysłem może być rezerwacja miejsc w kinie, która najczęściej odbywa się drogą elektroniczną. Tworzona aplikacja ma ułatwić pracę restauratorom i pozwolić na lepsze zarządzanie dostępnym miejscem oraz pośrednio zaopatrzeniem i personelem.

Ponadto istotnym celem projektu, jest zapoznanie się z realiami pracy nad dużym projektem informatycznym oraz analiza i próba rozwiązania podstawowych problemów jakie są związane z tym zagadnieniem. Projekt może być wymagający na poziomie technicznym, jak i komunikacyjnym. Przed przystąpieniem do aktywności zawodowej nie jest łatwo zdobyć doświadczenie w zakresie pracy w większym zespole inżynierskim.

1.2 Zakres projektu

Podstawowy zakres funkcjonalności można rozważać z perspektywy klienta, chcącego zamówić miejsce w restauracji, oraz właściciela i obsługi rezerwacji. Klient dzięki aplikacji powinien mieć zdolność zarezerwowania miejsca w wybranej przez siebie restauracji. Restauratorzy powinni mieć zdolność dodawania własnych rezerwacji oraz potwierdzania rezerwacji danych użytkowników. Aplikacja ma ułatwić i zautomatyzować komunikację między klientami a restauracjami. Warto zaznaczyć, że aplikacja nie udostępnia narzędzi umożliwiających zarządzanie restauracjami. W celu osiągnięcia przedstawionego celu należy stworzyć stronę internetową umożliwiającą dostęp do wybranych funkcjonalności, połączoną z aplikacją webową z dostępem do bazy danych.

2 Analiza wymagań

2.1 Analiza rynkowa

Głównym celem projektu jest ożywienie branży gastronomicznej oraz usprawnienie i automatyzację niektórych procesów w restauracjach. Aplikacja ma na celu częstsze odwiedzanie lokalu przez ludzi, a także do odciążenie właścicieli restauracji i eliminację ksiąg rezerwacyjnych z rezerwacją przez telefon. Aplikacja pozwoli na promocję restauracji poprzez opinie użytkowników czy ilość rezerwacji. Na podstawie ilości rezerwacji w ciągu dnia system może prognozować zapełnienie restauracji, a tym samym dostosować odpowiednią ilość pracowników obsługi restauracji w zależności od dnia tygodnia czy pory dnia.

2.1.1 Dostępne rozwiązania

Na rynku jest dostępnych kilka aplikacji oferujących zbliżony zakres funkcjonalności do przedstawionego. Poniżej przedstawiono pobieżną analizę dostępnych rozwiązań.

gastrobooking.pl Popularny w polsce serwis do rezerwacji miejsc w restauracjach. W Polsce umożliwia rezerwację stolików jedynie w Krakowie.

quandoo.com

zomato.com

opentable.com Jest to aplikacja posiadająca najwięszką bazę restauracji (40 000) w 14 krajach. Na Polskim rynku dostępne są jedynie dwie restauracje.

TODO: @Hubert zdjecia tabelka porownujaca, wnioski

Rysunek 1: Firmy konkurencyjne.

	gastrobooking.pl	quandoo.com	zomato.com	opentable.com
Popularne	W Polsce	W 12 krajach	W 24 krajach	W 14 krajach
Rezerwacja stolików w polskich restauracjach	Kraków	Niedostępna	Niedostępna	2 restauracje
Recenzje i ocena	✓	✓	✓	✓
Menu	✓	✓	✓	✓
Lokalizacja	✓	✓	✓	✓
Galeria zdjęć	✓	✓	✓	✓
Wskaźnik popularności w danym tygodniu	✗	✗	✓	✓
Wersja mobilna	✗	✓	✓	✓
Ilość zarejestrowanych restauracji	77	17 000	-	40 000

Rysunek 2: Porównanie firm konkurencyjnych.

2.1.2 Analiza wymagań biznesowych

Wymienione w poprzednim paragrafie serwisy nie występują w Polsce, lub są słabo rozpowszechnione. W porównaniu do konkurencji podstawową zaletą aplikacji ma być jej niska cena, oraz prostota. Zwiększa to zakres firm, który mogłyby sobie pozwolić na wdrożenie naszej aplikacji, co na dłuższą metę może stanowić o większej popularności.

Główną grupą docelową naszego produktu są restauratorzy oraz obsługa restauracji z dużą liczbą rezerwacji. Dzięki przygotowanej aplikacji mogą skorzystać z najnowszych rozwiązań technicznych, aby lepiej zarządzać swoją dostępnymi miejscami, oraz personelem. Zintegrowanie opracowanego systemu z innymi systemami umożliwiającymi zarządzanie personelem, kosztami czy zamówieniami znacząco ułatwiałoby zarządzanie i podejmowanie właściwych decyzji na poziomie managerskim.

Ryzyka projektu

W projektcie przyjęto oszacowanie ryzyka na podstawie określinia poziomu ryzyka jako iloczynu prawdopodobieństwa wystąpienia i stopnia ryzyka. Wyróżniono następujące prawdopodobieństwa wystąpienia:

1. prakycznie niemożliwe
2. bardzo mało prawdopodobne
3. zdarza się rzadko
4. może się zdarzyć
5. duże prawdopodobieństwo zdarzenia

Wyróżniono następujące stopnie ryzyka:

1. bardzo mały
2. mały
3. średni
4. duży
5. bardzo duży

Stosując opisaną metodykę, wybrano kila najważniejszych czynników i oszacowano poziom ryzyka.

Tabela 1: oszacowanie poziomu ryzyka

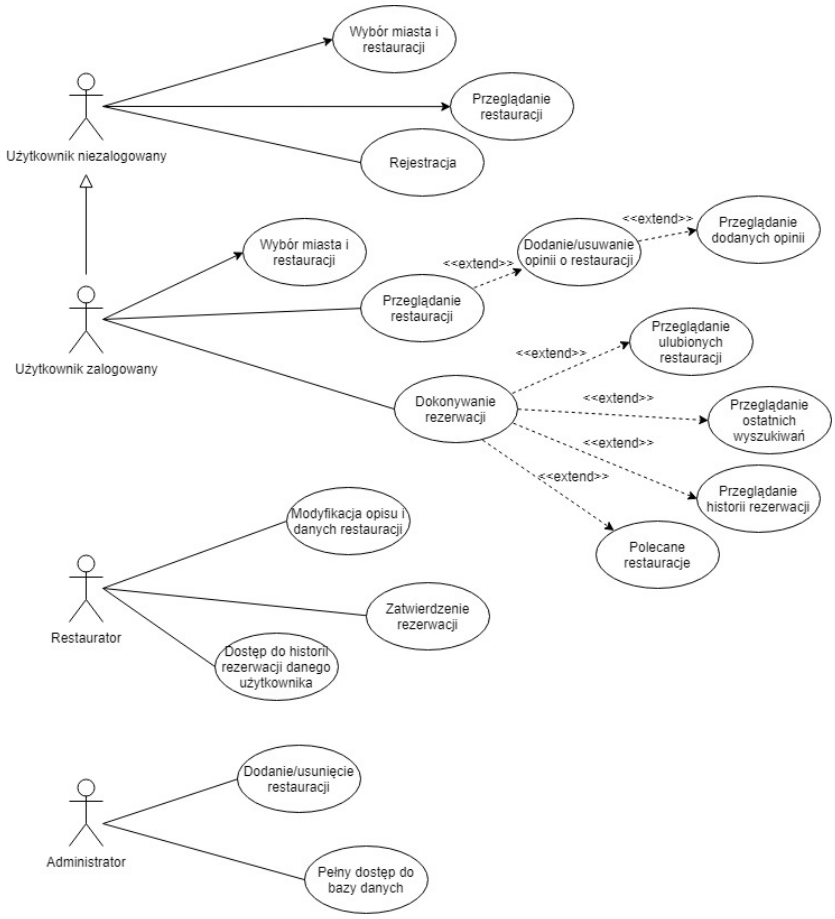
Ryzyko	P(w)	Stopień ryzyka	poziom ryzyka
Nie zaimplementowanie funkcji ze względu na napotkane problemy	4	4	16
Złe oszacowanie harmonogramu projektu (diagram Gantta)	3	4	12
Brak współpracy w zespole , Problem z komunikacją	4	4	16
Awaria sprzętu	3	4	12
Zdarzenia przy projekcie (Utrata danych, złośliwe oprogramowanie , brak zasilania)	2	4	8
Źle dobrane technologie	2	4	8
Przekroczenie budżetu	4	4	16
Złe decyzje projektowe podjęte na początku, mające duży wpływ na ostatnią fazę projektu i powodujące zmiany w systemie	4	5	20
Niedostateczne zabezpieczenie danych	4	4	16
Kłopoty związane z regulacjami prawnymi	2	4	8

2.2 Wymagania funkcjonalne

2.2.1 Podstawowe przypadki użycia

Użytkowników aplikacji można podzielić na 3 grupy. Każda z grup posiada własne przypadki użycia.

1. klient restauracji, niezalogowany użytkownik
   (a) przeglądanie profilu restauracji
   Niezalogowany użytkownik może wyświetlać profil restauracji i ma dostęp do wszystkich informacji zawartych w profilu.
   (b) rejestracja
   Większość przypadków użycia jest dostępnych po rejestracji i zalogowaniu.
2. klient restauracji, zalogowany użytkownik



Rysunek 3: Schemat przypadków użycia systemu.

(a) złożenie rezerwacji
Klient ma możliwość wyboru daty i godziny rezerwacji. Na podstawie informacji o dostępności stolików system daje informację zwrotną, czy rezerwacja w wybranych godzinach jest możliwa. Klient dokonuje rezerwacji, która następnie musi być zatwierdzona przez restaurację.

(b) dodawanie opinii o restauracji
Użytkownik ma możliwość dodatnia opinii, która będzie widoczna dla wszystkich użytkowników w profilu wybranej restauracji.

(c) przeglądanie dodanych opinii
Klient ma dostęp do wcześniej dodanych opinii w swoim profilu.

(d) przeglądanie ulubionych restauracji
Na podstawie historii rezerwacji prezentowana jest lista ulubionych restauracji, która jest prezentowana użytkownikowi po zalogowaniu.

(e) przeglądanie historii rezerwacji
Użytkownik ma dostęp do historii własnych rezerwacji.

(f) wysyłanie wiadomości do restauracji
Klient może się komunikować bezpośrednio z restauracją przez stronę internetową.

(g) polecane restauracje
Na postawie historii rezerwacji, oraz preferencji użytkowników o podobnych cechach prezentowana jest lista restauracji, które mogą się spodobać użytkownikowi. Do wyznaczania listy stosuje się algorytm kNN.

3. restaurator

(a) zmiana informacji odnośnie restauracji
Restaurator posiada możliwość zmiany danych prezentowanych na profilu restauracji. Zaliczają się do nich informacje o nazwie, adresie i zdjęcia. Restaurator nie ma wpływu na opinie widoczne na profilu jego restauracji.

(b) potwierdzenie rezerwacji
Po złożeniu rezerwacji przez użytkownika, musi zostać ona zatwierdzona przez restaurację. W trakcie ewaluacji danej rezerwacji pomocne może być przeglądanie profilu klienta.

(c) wysłanie wiadomości do klienta Opcja bezpośredniej komunikacji przez stronę internetową.

(d) przeglądanie profilu klienta Restaurator ma dostęp do profilu klienta.

4. administrator

(a) dodanie restauracji Dodawanie restauracji powinno być kontrolowane, aby uniknąć udostępniania nieprawdziwych bądź niewłaściwych informacji użytkownikom. W celu ułatwienia procesu weryfikacji danych, tylko administrator ma możliwość dodania restauracji do systemu.

(b) pełny dostęp do bazy danych Administrator może dodawać, usuwać, modyfikować i odczytywać wszystkie tabele w bazie.

2.3 Wymagania niefunkcjonalne

2.3.1 Wykorzystane technologie i narzędzia

Projekt został zrealizowany z wykorzystaniem języka C# oraz frameworka ASP.NET MVC. Do realizacji frontendu zostały wykorzystane HTML5, CSS 3.0 oraz JavaScript. Jako system zarządzania bazą danych wykorzystano Microsoft SQL Server.

Rysunek 4: Wykorzystane technologie.

- **Język programowania C#**

  C# jest obiektowym językiem programowania, który oficjalnie przedstawiony został w 2000 roku. Powstał na zlecenie firmy Microsoft, aby ułatwić tworzenie oprogramowania dla systemu operacyjnego Windows. W początkowej fazie swojego istnienia język był wspierany wyłącznie przez Microsoft, dlatego też rozpatrując język nieodzownym elementem było platforma programistyczna .NET. Jednak jego dobrze przemyślana struktura sprawiła, że język stał się popularny i powstało dla niego kilka open source standardów (m.in. Mono). Od 2016 roku Microsoft zmieniło swoją politykę i stworzyło .Net Core, które stało się praktycznie w pełni open source projektem.

- **Platforma .NET**

  Do roku 2016 .NET rozwijały się dwa podejścia co do środowisk uruchomieniowych, czyli oficjalną wersję opublikowaną przez Microsoft, jak i tworzoną przez pasjonatów wersja open source. Jednak wraz z publikacją .NET Core 1.0 rozwój platformy został połączony w jedno. Wprowadzono ujednolicony standard platformy (.NET Standard Library). Dzięki temu tworzone od teraz biblioteki są możliwe w użyciu również dla .NET Core, Xamarin i .NET Framework. Platforma .NET wspiera nie tylko język C# ale również takie języki jak m.in. F#, Visual Basic, C++. .NET Framework udostępnia specyficzne dla systemu operacyjnego Windows API takie jak Windows Forms i WPF (tworzenie aplikacji okienkowych). .NET Core zakłada całkiem odmienne podejście do przeznaczenia od wcześniejszego standardu. Głównym założeniem jest wieloplatformowość. Przykładem

API są: ASP.NET Core (aplikacje webowe) oraz Universal Windows Platform (docelowo reużywalne oprogramowanie działające zarówno na komputerach, urządzeniach mobilnych, jak i dla aplikacji wspierający Internetu rzeczy itd.). Mono for Xamarin w tej chwili wersja .NET dla aplikacji działających pod iOS oraz Android. Historycznie opensource'owa wersja standardu .NET Framework.

- **ASP.NET Core 2.0**

  ASP.NET to API umożliwiające tworzenie aplikacji webowych[**?**], czyli takie, które do swojego działania potrzebują jedynie przeglądarki internetowej. Nie wymagają od użytkownika instalowania zewnętrznego oprogramowania. Dzięki temu stworzone aplikację działają na wielu platformach jednocześnie. Dodatkowo wszelkie zmiany wprowadzane są wyłącznie po stronie serwera (nie angażują użytkownika). Do głównych elementów API zaliczamy: Web Forms, MVC (MVC + Web Page + Web API) oraz SignalR. Wprowadzenie standardu .NET Core uporządkowało kwestie związane z różnym podejściem tworzenia aplikacji webowych. Web Forms oraz Web Page praktycznie przestały istnieć.

- **Entity Framework Core**

  Entity Framework Core to narzędzie pozwalające przetłumaczyć relacyjne bazy danych na obiekty. Narzędzia te nazywamy ORM (Object Relational Mapping). Dzięki temu podejściu można tworzyć bazy danych bez użycia języka SQL. Ułatwia to zarządzanie danymi oraz przyspiesza pracę z nimi. Ponadto biblioteka ta zapewnia wsparcie nie tylko na poziomie tworzenia zapytań, ale również dzięki podejściu Code First możemy zaprojektować relacyjną bazę danych z poziomu języka obiektowego. Entity Framework pozwala także tworzyć automatycznie obiekty na podstawie już istniejącej bazy. Ostatnim wspieranym podejściem jest tworzenie bazy przy pomocy podejście Model first polegające na stworzeniu modelu w ADO.NET Entity Data Model Designer, a następnie Entity Framework tłumaczy go na obiekty oraz relacyjną bazę danych.

- **Bootstrap**

  Biblioteka zapewniająca głównie style CSS, ale również określa ich zachowanie (wykorzystuje JavaScript). Główną zaletą Bootstrap jest to, że wspomaga tworzenie responsywnych aplikacji webowych, to znaczy, że powinny dobrze wyglądać nie tylko na monitorze, ale również na tablecie, czy telefonie komórkowym. Pomaga w kontrolowaniu zachowania gotowych komponentów HTML.

- **jQuery**

  JQuery jest biblioteką JavaScript ułatwiająca manipulacją drzewa DOM

(Document Object Mode). Wykorzystanie jQuery eliminuję problem interpretacji kodu JS przez różne przeglądarki. Jego niewątpliwą zaletą jest wbudowanie AJAX. Sprawia, że opisane rozwiązanie jest czytelniejsze oraz wprowadza uproszczone odwołanie się do elementów strony, oraz wykorzystania AJAX.

- **SignalR**

  SignalR jest biblioteką Microsoft ASP.NET, która pozwala na wysyłanie asynchronicznych powiadomień do aplikacji po stronie klienta. Biblioteka zawiera komponenty JavaScript po stronie serwera i po stronie klienta. Umożliwia dodanie funkcjonalności do aplikacji w czasie rzeczywistym.

2.3.2 Wykorzystane dobre praktyki

W trakcie projektu przyjęto metodologię SOLID, oraz fragmenty metodologii Clean Code. Pozwala to na stworzenie łatwo rozszerzalnego kodu, który jest utrzymywalny i dobrze zorganizowany.

3 Projekt systemu

3.1 Architektura systemu

W niniejszym rozdziale przedstawiono ogólny przegląd architektury całej aplikacji, opartej na zalecanych sposobach pracy z wybranymi narzędziami.

3.1.1 ASP.NET MVC

Projekty wykonywane w frameworku ASP.NET MVC wymuszają pewną organizację projektu, która zwiększa uporządkowanie oraz wymusza korzystanie z dobrych praktyk. W projekcie można wyróżnić 4 zasadnicze części: Views, Controllers, Models i Router. Poniżej przedstawiono krótki opis poszczególnych części.

Model Modele są odpowiedzialne za przechowywanie informacji domenowej i stanu aplikacji. Najczęściej są implementowane jako Plain Old CLR Object (POCO) i służą do modelowania danych z bazy. Obiekty te są niezależne od frameworków, systemu zarządzania bazą danych czy mapowań ORM. Modele mogą być łączone w ViewModele, aby ułatwić grupowanie i przekazywanie informacji w aplikacji.

View Jest to frontendowa część projektu, która determinuje wygląd strony pokazywanej użytkownikowi. Technologie wykorzystywane w tym obszarze to HTML, CSS oraz JavaScript. Dodatkowo do widoku są przesyłane dane z innych komponentów w postaci modeli lub viewmodeli. ASP.NET MVC umożli-

wia dodawanie fragmentów kodu w C# do ułatwienia wykorzystywania danych z modeli.

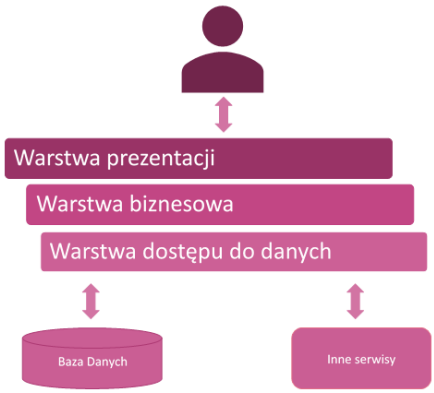
Controler Najczęściej jest to element implementujący wybraną część logiki biznesowej. Wykorzystuje dane z Modeli, aby wytworzyć i przekierować użytkownika do odpowiedniego Widoku. Każdy controler jest odpowiedzialny za obsługę HTTP Request. Taka organizacja wymusza podział projektu na mniejsze klasy z dobrze zdefiniowanym zakresem odpowiedzialności.

Router Jest to komponent, który odpowiada za mapowanie zapytań HTTP na poszczególne akcje w kontrolerach.

Pierwsze trzy komponenty są dobrze znane ze wzorca projektowego na Model-View-Controler, natomiast ostatni jest często spotykanym dodatkiem w wielu frameworkach sieciowych, umożliwiający łatwe mapowanie. Konstrukcja frameworku MVC pozwala na łatwe rozdzielenie odpowiedzialności komponentów i klas oraz wstrzykiwanie zależności do poszczególnych klas.

3.1.2 Architektura warstwowa

Wzorzec programowania oparty na architekturze warstwowej (ang. tier architectrue) jest powszechny w procesie tworzenia aplikacji internetowych. Podejście te rozdziela logikę systemu od interfejsu użytkownika. W podejściu architektury trzywarstwowej wyróżniamy następujące warstwy:



Rysunek 5: Warstwy w architekturze wielowarstwowej.

- **Warstwa prezentacji (ang. presentation tier)** - odpowiedzialna za przedstawienie funkcjonalności użytkownikowi, umożliwiająca wyświetlanie oraz wprowadzanie danych. W tej warstwie ASP.NET używa się oprócz języka platformy (np. C#) języków do opisów wyglądu strony (CSS, HTML, JavaScript).

- **Warstwa biznesowa (ang. business tier)** - w miejscu tym umieszczona jest logika aplikacji. Dane pozyskane z warstwy danych są przetwarzane by były w odpowiedni sposób przygotowane do wysyłania do warstwy prezentacji, jak również pozyskane od użytkownika przekształcane są w oczekiwanej formie, transportowane są do warstwy danych.
- **Warstwa danych (ang. persistance tier)** - odpowiedzialna za komunikacji z zewnętrznymi serwisami zarządzającymi danymi, np. z bazą danych.

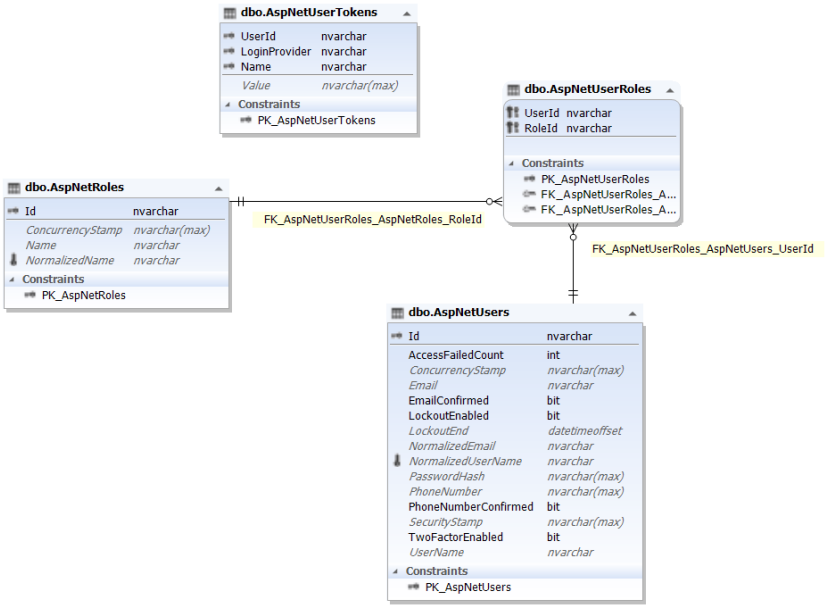
Dzięki zastosowaniu tego podejściu zyskujemy łatwość modyfikacji poszczególnych funkcjonalności, ponieważ zmiana w jednej warstwie nie wymusza zmian w całym systemie. Inną zaletą jest fakt, iż tworzone rozwiązania stają się bardziej przejrzyste. Za używaniem tego sposobu projektowania aplikacji przemawia to, że tę samą logikę możemy wykorzystać zarówno przy tworzeniu aplikacji internetowych jak i mobilnych. Wykorzystanie architektury warstwowej pozwala na reużywalność zaimplementowanych komponentów oraz ogranicza liczbę błędów (poprzez użycie wstrzykiwania zależności lub interfejsów), poza bieżącą warstwą nie ma dostępu do wewnętrznych metod. W stworzonym systemie oprócz przedstawionego podziału wprowadzono dodatkowy element przytrzymujący modele wykorzystywane między warstwami. Ich szczególnym przykładem są modele DAO, które używane są wyłącznie przez warstwę danych do komunikacji z bazą danych (mapowane są one przy użyciu Automappera). Dla realizacji założeń każda warstwa zaimplementowana została w osobnym projekcie.

3.2 Diagram elementów systemu

TODO: @Jędrzej

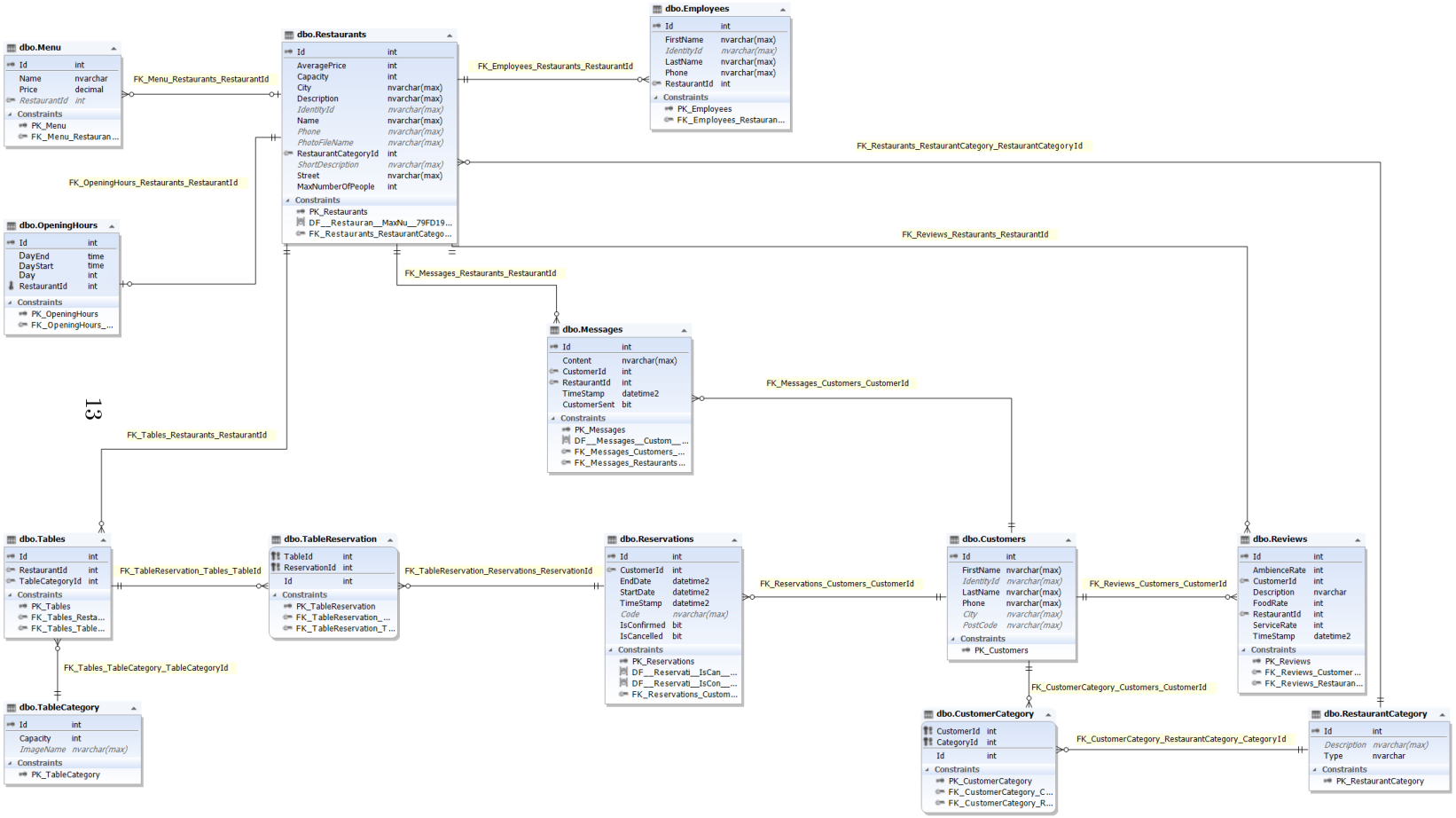
3.3 Projekt bazy danych

Aplikacja wykorzystuje dwie bazy danych. Jedna baza danych (rysunek 5) jest używana do zarządzania użytkownikami - dane logowania, ich role i inne dane potrzebne do weryfikacji. Tabele, jak i relacje w tej bazie danych zostały utworzone przy użyciu biblioteki ASP.NET Core Identity ze względu na szereg udostępnianych przez nią rozwiązań m.in. do logowania, rejestracji, przypominania czy resetowania haseł.



Rysunek 6: Schemat bazy danych zarządzającej użytkownikami.

Drugą bazą danych jest baza zawierająca tabele potrzebne do obsługi aplikacji. Baza danych 7 jest powiązana przez unikalne pole *IdentityId* w tabeli *Customers/Restaurants* z polem Email w bazie danych 6.

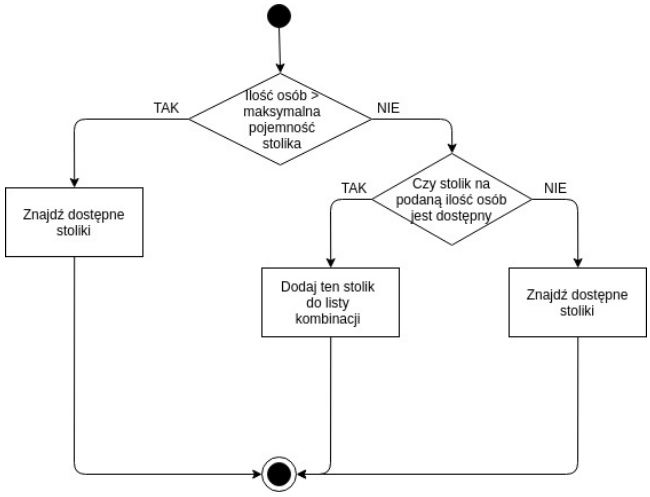


Rysunek 7: Schemat bazy danych aplikacji.

3.4 Algorytm wyszukiwania stolików

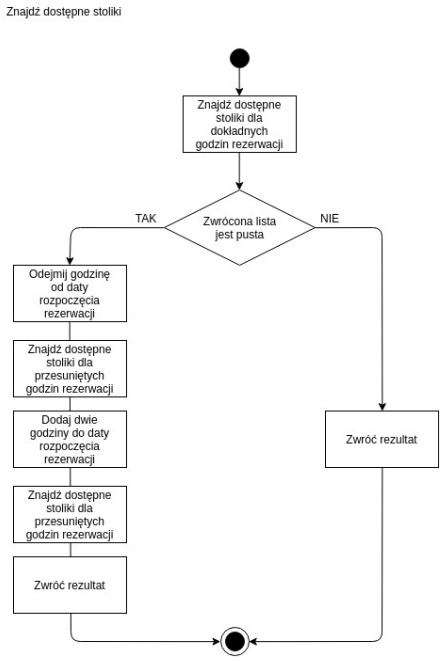
W trakcie składania zamówienia użytkownik aplikacji podaje liczbę osób, dla których mają zostać zarezerwowane miejsca. W celu określenia czy złożenie zamówienia w wybranych godzinach jest możliwe opracowano algorytm, tworzący kombinację stolików dostępnych w wybranym czasie. Wejście algorytmu stanowią: idRestaracji, ilość osób, dla których ma zostać zrealizowana rezerwacja, godzina rozpoczęcia rezerwacji oraz czas trwania rezerwacji. Wyjście algorytmu stanowi lista kombinacji stolików dostępnych w wybranym czasie mogących pomieścić wszystkie osoby, dla których została zrealizowana rezerwacja.

Przy projektowaniu algorytmu wykorzystano technikę dziel i rządź. Na każdym etapie dokonywano odcięć w celu podziału zadania na mniejsze i lepiej określone problemy.



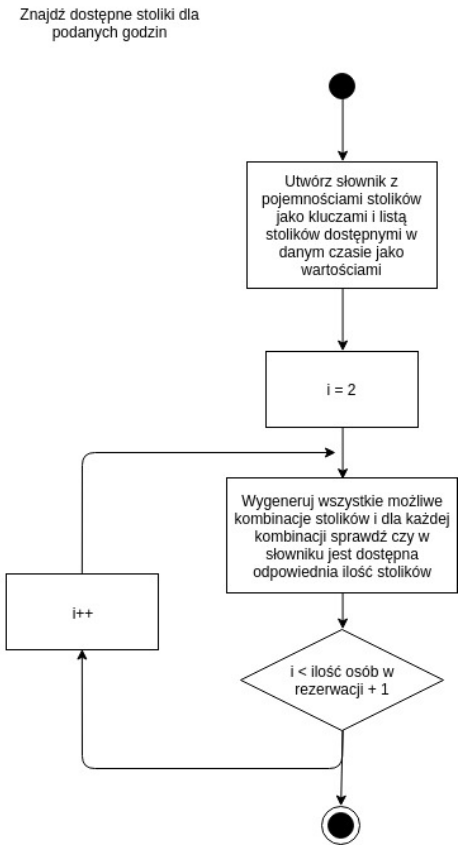
Rysunek 8: Pierwszy krok działania algorytmu

Algorytm w pierwszym kroku (rys. 8) określa czy podana w rezerwacji ilość osób może zostać posadzona przy jednym stoliku. Jeśli tak, sprawdzane jest czy stoliki o pojemności określonej równej ilości osób z rezerwacji są dostępne w wybranym czasie. Jeśli są dostępne algorytm kończy działanie i zwraca znaleziony stolik. Jeśli stolik nie jest dostępny, możliwe jest uzyskanie odpowiedniej ilości miejsc poprzez zajęcie mniejszych stolików (przykładem może być zarezerwowanie dwóch dwuosobowych stolików dla czterech osób). Odpowiedzialna jest za to procedura *Znajdź dostępne stoliki* opisana poniżej. Procedura ta jest także wywoływana w przypadku gdy ilość osób w zamówieniu jest większa od stolika z największą pojemnością.



Rysunek 9: Znajdź dostępne stoliki

Procedura *Znajdź dostępne stoliki* określa czy są dostępne kombinacje stolików w podanych przez użytkownika godziny (rys. 9). Jeśli zostanie znaleziona dowolna kombinacja będąca w stanie pomieścić ilość osób z rezerwacji procedura kończy działanie. W przeciwnym wypadku procedura modyfikuje godziny rozpoczęcia rezerwacji - dodaje i odejmuje godzinę od oryginalnej godziny rozpoczęcia rezerwacji i szuka kombinacji dla zmodyfikowanych dat.



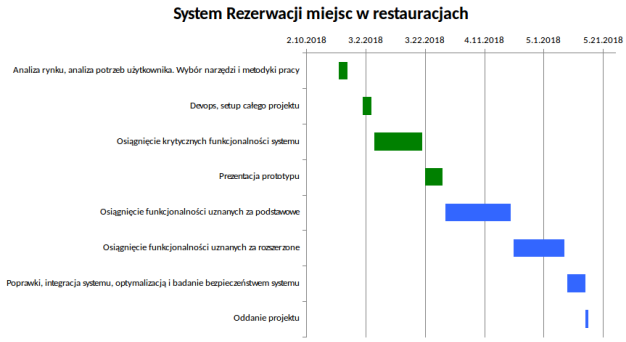
Rysunek 10: Znajdź dostępne stoliki dla podanych godzin

Faktyczne szukanie kombinacji (rys. 10) odbywa się z wykorzystaniem słownika mapującego pojemność stolików na listę stolików dostępnych w danym czasie o danej pojemności. Następnie, iterując po ilości stolików wyznacza się wszystkie możliwe kombinacje stolików (np. realizując zamówienie dla 10 osób przy użyciu dwóch stolików następujące kombinacje mogą zostać użyte dla zrealizowania zamówienia: {{8,2},{7,3},{6,4},{5,5}}). Dla wyznaczonej kombinacji sprawdza się, czy słownik zawiera odpowiednią ilość stolików dostępnych w danym czasie. Jeśli istnieją wolne stoliki potrzebne do utworzenia danej kombinacji w słowniku, kombinacja zostaje dodana do listy rozwiązań.

4 Kamienie milowe i plan projektu

4.1 Wykres Gantta

Na rys. 11 przedstawiono wykres Gantta opracowany przed przystąpieniem do pracy nad projektem. Niestety przy opracowaniu tego wykresu poczyniono zbyt



Rysunek 11: Wykres Gantta.

optymistyczne założenia, przez co nie udało się zrealizować planu przestawionego na wykresie.

4.2 Rzeczywista kalkulacja kosztów i nakładów pracy

TODO: @Jędrzej

5 Wdrożenie projektu

Aplikacja sieci Web jak i baza danych została wdrożona z użyciem usług Microsoft Azure. Utworzony został serwer SQL zawierający dwie bazy danych (pierwszy zawierający bazę danych użytkowników oraz ich uprawnienia, drugi zawierający pozostałe tabele) oraz serwer aplikacji. Od docelowego użytkownika wymagana jest przeglądarka z dostępem do internetu. Aplikacja webowa została przetestowana w przeglądarce Google Chrome wersji 65 oraz Microsoft Edge 41.

5.1 Adres URL

Aplikacja jest dostępna pod adresem:

https://zarezerwujstolik.azurewebsites.net

5.2 Konta testowe

Przygotowane zostały dwa konta testowe:

Konto restauracji:

Nazwa użytkownika: restaurant@email.com Hasło: Password1!

Konto klienta:

Nazwa użytkownika: client@email.com Hasło: Password1!

6 Podsumowanie i wnioski

Literatura

[1] Oficjalna strona ASP.NET MVC
https://www.asp.net/mvc

[2] Oficjalna dokumentacja ASP.NET MVC
https://docs.microsoft.com/plpl/aspnet/#pivotc̄ore&panelc̄ore_overview

[3] Mosh Hamedani *Should you split your ASP.NET MVC project into multiple projects?*.
https://programmingwithmosh.com/csharp/should-you-split-your-asp-net-mvc-project-into-multiple-projects/