POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)

SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

PRACA DYPLOMOWA

INŻYNIERSKA

System zarządzania restauracją

Restaurant managment system

AUTOR:

Olaf Krawczyk

PROWADZĄCY PRACĘ:

dr inż. Roman Ptak, W4/K-9

OCENA PRACY:

WROCŁAW, 2017

**Spis treści**

[Spis rysunków 4](#_Toc499489347)

[Spis tabel 5](#_Toc499489348)

[Spis listingów 6](#_Toc499489349)

[Skróty 7](#_Toc499489350)

[1. Wstęp 8](#_Toc499489351)

[1.1. Wprowadzenie 8](#_Toc499489352)

[1.2. Cel i zakres pracy 8](#_Toc499489353)

[1.3. Aplikacje typu SPA 9](#_Toc499489354)

[1.4. Aplikacja mobilna 9](#_Toc499489355)

[2. Istniejące rozwiązania 11](#_Toc499489356)

[2.1. OpenTable 11](#_Toc499489357)

[2.2. Zomato 12](#_Toc499489358)

[3. Projekt systemu 14](#_Toc499489359)

[3.1. Użytkownicy systemu 14](#_Toc499489360)

[3.1.1. Klient 14](#_Toc499489361)

[3.1.2. Właściciel 14](#_Toc499489362)

[3.2. Wymagania funkcjonalne 14](#_Toc499489363)

[3.2.1. Identyfikacja użytkowników 14](#_Toc499489364)

[3.2.2. Wyszukiwanie restauracji 15](#_Toc499489365)

[3.2.3. Dokonanie rezerwacji 15](#_Toc499489366)

[3.2.4. Statusy rezerwacji 15](#_Toc499489367)

[3.2.5. Dodawanie restauracji 16](#_Toc499489368)

[3.2.6. Zarządzanie restauracjami 16](#_Toc499489369)

[3.2.7. Obsługa rezerwacji 16](#_Toc499489370)

[3.3. Wymagania niefunkcjonalne 16](#_Toc499489371)

[3.3.1. Dostęp do serwisu 16](#_Toc499489372)

[3.3.2. Komunikacja pomiędzy składowymi systemu 16](#_Toc499489373)

[3.3.3. Zabezpieczenia 17](#_Toc499489374)

[3.3.4. Wykorzystane technologie 17](#_Toc499489375)

[3.4. Przypadki użycia 18](#_Toc499489376)

[3.4.1. Przypadki użycia dla użytkownika typu Klient 18](#_Toc499489377)

[3.4.2. Przypadki użycia dla użytkownika typu Właściciel 22](#_Toc499489378)

[3.5. Architektura systemu 23](#_Toc499489379)

[3.6. Kolejna sekcja poziomu 1 24](#_Toc499489380)

[3.6.1. Kolejna sekcja poziomu 2 24](#_Toc499489381)

[4. Podsumowanie i wnioski 26](#_Toc499489382)

[Literatura 27](#_Toc499489383)

[Dodatek A 28](#_Toc499489384)

# Spis rysunków

[Rys. 1. Przykład podpisu rysunku 22](#_Toc499481703)

# Spis tabel

[Tab. 1. Przykład podpisu tabeli 21](#_Toc499481730)

# Spis listingów

[Listing. 1. Początkowe żadanie HTTP 3](#_Toc465685644)

# Skróty

**JSON** (ang. JavaScript Object Notation)

**HTTP** (ang. *eXtensible Markup Language*)

**REST** (ang. *Simple Object Access Protocol*)

**SPA** (ang. *Web Services Description Language*)

**WS** (ang. *Geographical Information System*)

**WK** (ang. *Spatial Data Infrastructure*)

**API** (ang. *International Standards Organization*)

**RFC** (ang. *Web Map Service*)

**MVC** (ang. *Web Feature Service*)

**JPA** (ang. *Web Processing Service*)

**ORM** (ang. *Geography Markup Language*)

**SRG** (ang. *Seeded Region Growing*)

**SOA** (ang. *Service Oriented Architecture*)

**IT** (ang. *Information Technology*)

1. Wstęp

Niniejsza praca dyplomowa jest opisem realizacji systemu aplikacji, którego celem jest dostarczenie rozwiązania ułatwiającego rezerwację stolików w restauracjach. System ten składa się z trzech aplikacji – mobilnej, internetowej oraz serwera dostarczającego interfejs komunikacji.

* 1. Wprowadzenie

Na rynku znajdziemy wiele serwisów pozwalających zamówić jedzenie online z dowozem do klienta. Niewiele jest jednak takich, które pozwalają na składanie rezerwacji w restauracjach. W ramach realizacji pracy dyplomowej stworzony zostanie serwis ułatwiający rezerwację stolików w restauracjach.

* 1. Cel i zakres pracy

Celem pracy będzie stworzenie serwisu, którego głównym zadaniem będzie umożliwienie jego użytkownikom dokonywanie rezerwacji stolików w restauracjach. Na serwis ten składały będą się aplikacje klienckie pozwalające na dostęp do serwisu za pomocą przeglądarki internetowej oraz aplikacji mobilnej. Aplikacja internetowa będzie pozwalała klientom na rezerwację stolików oraz zarządzanie restauracjami przez ich właścicieli. Mobilna część serwisu dedykowana będzie wyłącznie klientom. W skład serwisu będzie wchodził serwer dostarczający interfejs komunikacji pomiędzy aplikacjami klienckimi, a bazą danych.

Zakres prac obejmował obejmuje następujące czynności:

* Sporządzenie analizy wymagań użytkowników oraz specyfikacji funkcjonalności dostarczanych przez serwis
* Zaprojektowanie oraz implementację części serwerowej
* Stworzenie aplikacji klienckich, internetowej oraz mobilnej
* Wybór oraz wdrożenie odpowiednich zabezpieczeń w serwisie
* Sporządzenie dokumentacji projektu
  1. Aplikacje typu SPA

Technologia umożliwiająca tworzenie aplikacji typu Single Page Applicaion powstała, aby zmniejszyć ilość zapytań HTTP wysyłanych do serwerów oraz dać ich użytkownikom wrażenie korzystania z aplikacji desktopowej. Aplikacja te ładowane są w całości do pamięci przeglądarki przy odwiedzeniu strony. Przy korzystaniu z serwisów stworzonych z wykorzystaniem technologii SPA użytkownik nie doświadcza przeładowania strony. Zmianom podlegają tylko wybrane elementy modelu DOM, np. określone znaczniki div. Nowe dane pobierane są wyłącznie, gdy zaistnieje taka potrzeba, a ich rozmiar można ograniczyć przez przesyłanie tylko istotnych informacji. Wiąże się to z oszczędnościami transferu oraz pozytywnie wpływa na odbiór strony przez użytkownika Popularnym serwisem opierającym się o tę technologię jest aplikacja internetowa gmail.com

Do tworzenia aplikacji SPA wykorzystywane są powszechnie znane technologie takie jak JavaScript (oraz jego odmiany), HTML i CSS. Wymagają one jednak frameworków takich jak Angular czy ReactJS.

Rozwiązania SPA mają wiele zalet, jednak posiadają również wady. Z uwagi na fakt, że aplikacja jest ładowana w całości, podczas pierwszego połączenia przesyłana jest stosunkowo duża ilość informacji.

* 1. Aplikacja mobilna

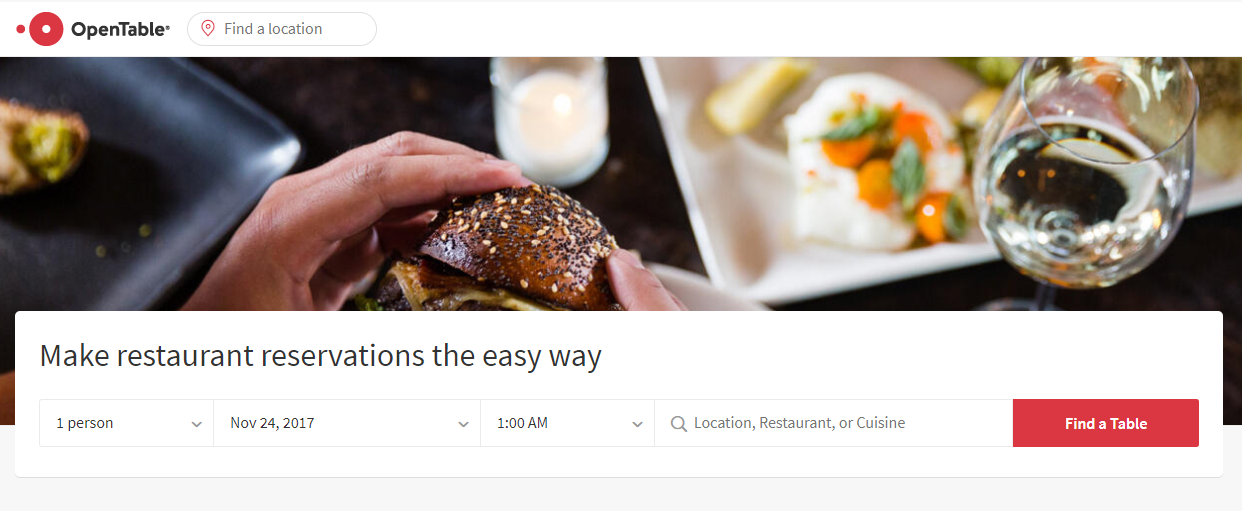
Aplikacje mobilne to aplikacje, które można uruchomić na urządzeniach mobilnych takich jak smartfony oraz tablety. Aplikacje te mogą być tworzone z wykorzystaniem technologii dostarczanych przez twórców systemu operacyjnego lub takich które pozwalają na tworzenie wieloplatformowych aplikacji.

Jednym z takich rozwiązań jest Ionic, działający w oparciu o Angular oraz Apatche Cordova. Framework ten zostanie wykorzystany do stworzenia aplikacji mobilnej na potrzeby niniejszej pracy.

1. Istniejące rozwiązania
   1. OpenTable

Amerykański serwis OpenTable daje możliwość rezerwacji miejsc w restauracjach na całym świecie.

Użytkownicy mogą dokonać rezerwacji za pośrednictwem strony internetowej lub aplikacji mobilnej. Aby zarezerwować stolik należy wypełnić formularz znajdujący się na stronie głównej serwisu.



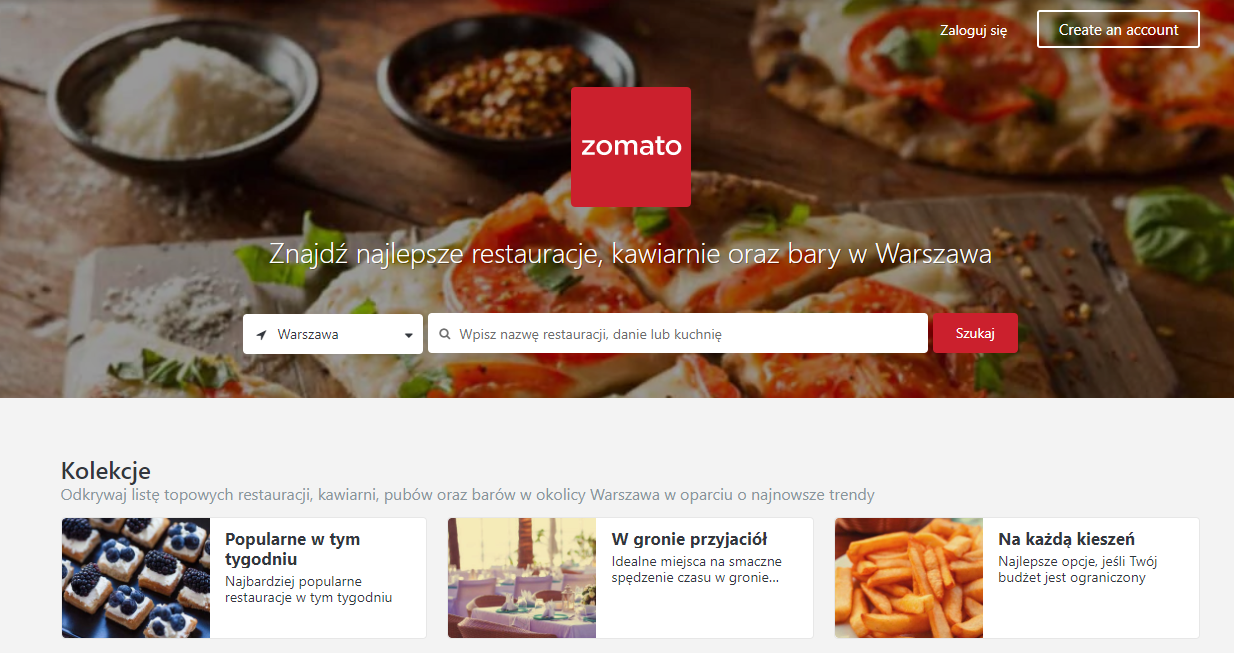
Rysunek 1 Fragment widoku strony głównej OpenTable

Jak widać na powyższym rysunku dane jakie należy podać to liczba osób dla jakiej ma zostać zarezerwowany stolik, datę rezerwacji, godzinę oraz lokalizację. Rezerwacji stolika można dokonać również poprzez aplikację mobilną.

Serwis poza podstawowymi informacjami takimi jak nazwa restauracji oraz adres udostępnia użytkownikom aktualnie serwowane menu oraz galeria restauracji.

Restauratorzy mają możliwość zarządzania restauracją za pomocą aplikacji na urządzenia z systemem iOS dostarczanej przez OpenTable. Aplikacja ta pozwala między innymi na wprowadzenie stolików znajdujących się w restauracji do serwisu oraz zarządzanie przychodzącymi rezerwacjami.

* 1. Zomato



Rysunek 2 Widok strony głównej serwisu Zomato

Użytkownicy aplikacji mają dostęp do galerii, menu restauracji, wskazówek dojazdu oraz danych kontaktowych. Serwis ten jest bardzo podobny do OpenTable, różni się on jednak podejściem do rezerwacji online. Użytkownicy korzystający z Zomato mogą zarezerwować za jego pośrednictwem stolik tylko w wybranych restauracjach.

Podobnie jak OpenTable Zomato udostępnia swoim użytkownikom aplikacje mobilne. Aplikacja ta dostarcza takich samych funkcjonalności jak serwis internetowy.

Porównując OpenTable oraz Zomato można dostrzec wiele cech wspólnych. Oba serwisy udostępniają użytkownikom możliwość wyszukiwania restauracji w wybranym mieście. Informacje o restauracjach dostarczane przez serwisy również są do siebie zbliżone. Jak wiele popularnych serwisów udostępniają klientom aplikacje mobilne pozwalające na korzystanie z serwisu za pomocą urządzeń przenośnych.

1. Projekt systemu
   1. Użytkownicy systemu

System opisany w tej pracy będzie dostarczał funkcjonalności dwu grupom użytkowników. Będą to klienci restauracji oraz właściciele restauracji. Zarówno właściciel jak i klient będą identyfikowani za pomocą danych osobowych oraz adresu email.

* + 1. Klient

Klient to osoba wykorzystująca serwis do znalezienia restauracji znajdującej się w okolicy oraz zarezerwowania stolika. Każdy klient posiadał będzie indywidualne konto, za pośrednictwem którego będzie mógł dokonać nowych rezerwacji oraz będzie miał dostęp do historii swoich rezerwacji.

* + 1. Właściciel

Właściciel restauracji będzie miał możliwość dodawania nowych restauracji do serwisu oraz zarządzania przychodzącymi rezerwacjami.

* 1. Wymagania funkcjonalne
     1. Identyfikacja użytkowników

Każdy z użytkowników niezależnie od roli posiadać będzie indywidualne konto. Aby utworzyć konto użytkownik będzie musiał podać imię, nazwisko, telefon, adres email oraz hasło. Użytkownik posiadający konto w serwisie będzie mógł się do niego zalogować podając adres email oraz hasło.

* + 1. Wyszukiwanie restauracji

Każdy klient będzie miał możliwość wyszukania restauracji w serwisie. Aby wyświetlić listę restauracji należy podać liczbę osób dla jakiej zarezerwowany ma zostać stolik, datę oraz godzinę rezerwacji i miasto z jakiego mają zostać wyświetlone dostępne restauracje.

* + 1. Dokonanie rezerwacji

Klient będzie miał możliwość wysłania prośby o rezerwację stolika w wybranej restauracji. Rezerwacja może zostać anulowana przez klienta. Klient będzie mógł wyświetlić wszystkie swoje rezerwacje, niezależnie od ich statusu. Dla uproszczenia przyjmujemy, że godzina, na którą zostaje dokonana rezerwacja może być wyłącznie pełna, a czas trwania rezerwacji jest stały i wynosi 2 godziny.

* + 1. Statusy rezerwacji

Każda rezerwacja w serwisie w zależności od wykonanych na niej operacji musi posiadać status odzwierciedlający jej stan. Status może przyjmować jedną z poniższych wartości:

Tabela 1 Statusy rezerwacji

|  |  |  |
| --- | --- | --- |
| Lp. | Status | Opis |
| 1 | PENDING | Rezerwacja została wysłana przez użytkownika,  oczekuje na akcję ze strony właściciela restauracji |
| 2 | ACCEPTED | Rezerwacja została zaakceptowana przez  właściciela restauracji |
| 3 | REJECTED | Rezerwacja została odrzucona przez właściciela  restauracji |
| 4 | CANCELED | Rezerwacja została anulowana przez klienta |
| 5 | CLOSED | Zaakceptowana rezerwacja została automatycznie  zamknięta po upłynięciu terminu rezerwacji |

* + 1. Dodawanie restauracji

Użytkownik zalogowany jako właściciel może dodawać nowe restauracje do serwisu. Dodając restauracje podaje dane takie jak nazwa, adres, opis, menu restauracji oraz stoliki w niej dostępne. Dodana restauracja przypisana jest do konta właściciela. Właściciel może mieć wiele restauracji.

* + 1. Zarządzanie restauracjami

Właściciel powinien mieć możliwość zmiany danych restauracji, jej menu oraz stolików.Wszystkie zmiany powinny być realizowane za pomocą interfejsu właściciela dostępnego z poziomu przeglądarki internetowej.

* + 1. Obsługa rezerwacji

Właściciel restauracji powinien mieć możliwość akceptacji bądź odrzucenia rezerwacji złożonych przez klientów. Powinien mieć również możliwość wyświetlenia wszystkich rezerwacji dla każdej z restauracji przypisanych do jego konta.

* 1. Wymagania niefunkcjonalne
     1. Dostęp do serwisu

Dostęp do serwisu dla klienta oraz właściciela powinien być możliwy za pomocą przeglądarki internetowej. Aplikacja internetowa powinna mieć intuicyjny oraz przyjazny użytkowników interfejs. Funkcjonalności dedykowane klientom powinny być również dostępne w aplikacji mobilnej.

* + 1. Komunikacja pomiędzy elementami systemu

Ponieważ opisywany system będzie składał się z trzech aplikacji oraz bazy danych, komunikacja pomiędzy poszczególnymi aplikacjami powinna być realizowana za pomocą protokołu HTTP z wykorzystaniem architektury REST.

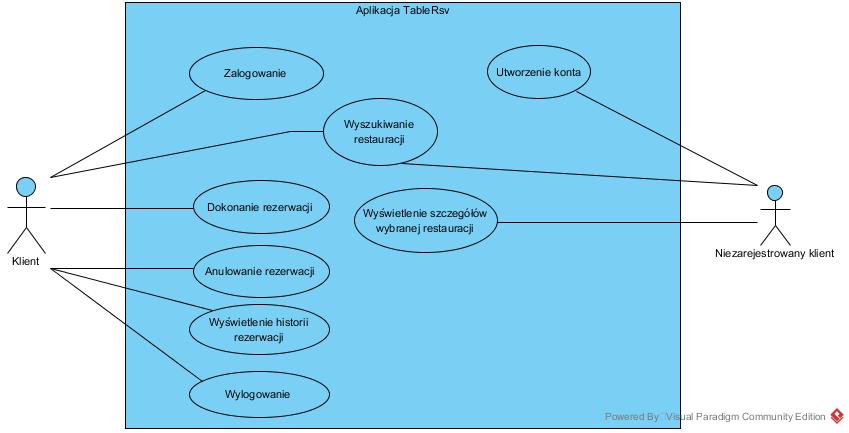
* + 1. Zabezpieczenia

Aby uzyskać dostęp do konta użytkownik musi zalogować się za pomocą adresu email oraz hasła podanego przy rejestracji. Wszystkie zapytania do serwera udostępniającego REST API powinny być zabezpieczone tokenem potwierdzającym, że użytkownik ma dostęp do żądanych informacji lub może wykonać żądaną akcję. Aby ułatwić użytkownikom korzystanie z serwisu token powinien być zapamiętany w pamięci przeglądarki. Czas ważności tokena powinien wynosić 10 dni.

* + 1. Technologie

Serwis powinien zostać stworzony z wykorzystaniem następujących technologii:

* Spring Boot
* Spring Security
* JSON Web Token
* Angular
* HTML5, CSS3
* Ionic
* PostgreSQL
* Bootstrap
  1. Przypadki użycia
     1. Przypadki użycia dla użytkownika typu Klient



Rysunek 3 Diagram przypadków użycia użytkownika typu Klient

Opis przypadków użycia widocznych na powyższym diagramie znajduje się w dalszej części podrozdziału. Powyższy diagram dotyczy klienta korzystającego z serwisu za pomocą przeglądarki internetowej. Klient korzystający z aplikacji mobilnej ma dostęp do tych samych funkcjonalności, z tą różnicą, że nie ma możliwości rejestracji nowego konta.

* + - 1. PU Utworzenie konta klienta

Przypadek ten opisuje utworzenie nowego konta użytkownika. Funkcjonalność ta jest dostępna jedynie w aplikacji internetowej.

Tabela 2 PU Utworzenie konta klienta

|  |  |
| --- | --- |
| WS | Otwarcie strony WWW serwisu |
| WK | Utworzenie nowego konta użytkownika lub komunikat o błędzie |
| Cel | Rejestracja nowego użytkownika typu Klient |
| Przebieg | 1. Przejście do formularza rejestracji klienta 2. Podanie danych osobowych 3. Podanie adresu email oraz hasła 4. Weryfikacja czy wszystkie pola zostały wypełnione 5. Weryfikacja czy w serwisie nie istnieje już klient   o tym samym adresie email   1. Utworzenie nowego konta klienta lub komunikat   o błędzie |

* + - 1. PU Zalogowanie

Przypadek użycia zalogowanie opisuje zachowanie aplikacji podczas logowania się klienta do serwisu zarówno za pomocą przeglądarki internetowej, jak i aplikacji mobilnej. Mechanizm autoryzacji tokenem zostanie opisany w części omawiającej implementację systemu.

Tabela 3 PU Zalogowanie klienta

|  |  |
| --- | --- |
| WS | Otwarcie strony WWW serwisu lub aplikacji mobilnej |
| WK | Zalogowanie klienta lub komunikat o błędzie |
| Cel | Zalogowanie użytkownika oraz utworzenie tokena sesji |
| Przebieg | 1. Przejście do formularza logowania klienta 2. Podanie adresu email oraz hasła 3. Weryfikacja czy klient o podanym adresie   email istnieje w serwisie oraz czy skrót hasła  zapisanego w bazie jest zgodny ze skrótem hasła  wprowadzonego podczas logowania   1. Jeżeli podane dane nie są poprawne wyświetlenie   komunikatu o błędzie, w przeciwnym przypadku  wygenerowanie tokena.   1. Wysłanie odpowiedzi z tokenem do aplikacji   klienckiej |

* + - 1. PU Wylogowanie

Wylogowanie jest możliwe wyłącznie w podczas korzystania z internetowej części serwisu, ponieważ tylko w tej w niej token użytkownika zostaje zapisany w pamięci urządzenia.

Tabela 4 PU Wylogowanie klienta

|  |  |
| --- | --- |
| WS | PU Zalogowanie |
| WK | Usunięcie tokena z pamięci przeglądarki |
| Cel | Wylogowanie klienta poprzez usunięcie tokena sesji |
| Przebieg | 1. Naciśnięcie przycisku „Logout” w menu 2. Usunięcie tokena z pamięci przeglądarki 3. Przekierowanie użytkownika na stronę główną |

* + - 1. PU Wyszukiwanie restauracji

Przebieg przypadku użycia „Wyszukiwanie restauracji” jest identyczny dla serwisu internetowego i aplikacji mobilnej. Dla aplikacji mobilnej warunkiem startowym jest „PU Zalogowanie”.

Tabela 5 PU Wyszukiwanie restauracji

|  |  |
| --- | --- |
| WS | Otwarcie strony WWW serwisu lub aplikacji mobilnej |
| WK | Zwrócenie listy restauracji spełniających podane kryteria  lub wyświetlenie komunikatu o błędzie |
| Cel | Wyświetlenie listy restauracji w których klient może  zarezerwować stolik |
| Przebieg | 1. Przejście do głównej strony 2. Wybór miasta 3. Wybór liczby gości 4. Wybór daty oraz godziny 5. Wybór rodzaju kuchni 6. Wyświetlenie restauracji spełniających kryteria   lub komunikatu z informacją, że żadna z restauracji  nie spełnia podanych kryteriów. |

* + - 1. PU Wyświetlenie szczegółów restauracji

Ten przypadek użycia jest identyczny dla aplikacji mobilnej oraz serwisu internetowego.

Tabela 6 PU Wyświetlenie szczegółów restauracji

|  |  |
| --- | --- |
| WS | PU Wyszukiwanie restauracji |
| WK | Wyświetlenie widoku szczegółów restauracji |
| Cel | Umożliwienie klientowi zapoznanie z ofertą restauracji |
| Przebieg | 1. Klient wybiera restaurację z listy otrzymanej podczas   wyszukiwania   1. Przekierowanie użytkownika do widoku szczegółów   restauracji |

* + - 1. PU Dokonanie rezerwacji

Tabela 7 PU Dokonanie rezerwacji

|  |  |
| --- | --- |
| WS | PU Wyświetlenie szczegółów restauracji |
| WK | Utworzenie nowej rezerwacji w wybranej restauracji |
| Cel | Zarezerwowanie stolika |
| Przebieg | 1. Należy kliknąć przycisk „View tables” 2. Wybranie jednego ze stolików z listy dostępnych   poprzez naciśnięcie przycisku „Book table”   1. Utworzenie nowej rezerwacji w statusie PENDING 2. Przekierowanie klienta do widoku historii jego   rezerwacji |

* + - 1. PU Wyświetlenie historii rezerwacji

Tabela 8 Wyświetlenie historii rezerwacji

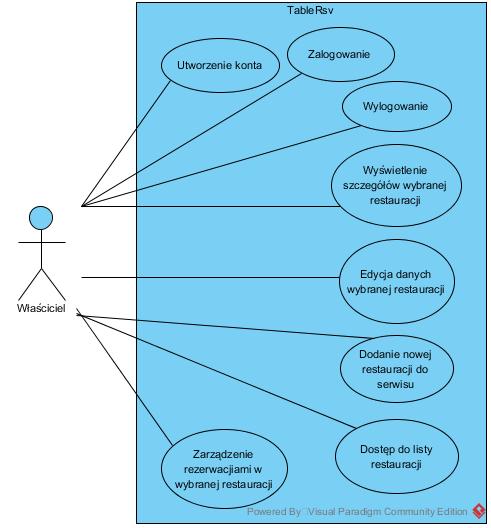
|  |  |
| --- | --- |
| WS | PU Zalogowanie |
| WK | Przekierowanie użytkownika na stronę zawierającą  historię jego rezerwacji |
| Cel | Wyświetlenie wszystkich rezerwacji  zalogowanego klienta |
| Przebieg | 1. Wybór widoku „Reservations” 2. Pobranie rezerwacji z bazy danych 3. Przekierowanie użytkownika do widoku   pokazującego dokonane przez niego rezerwacje |

* + - 1. PU Anulowanie rezerwacji

Tabela 9 Wyświetlenie historii rezerwacji

|  |  |
| --- | --- |
| WS | PU Wyświetlenie historii rezerwacji |
| WK | Zmiana statusu wybranej rezerwacji na CANCELED |
| Cel | Anulowanie wybranej przez klienta rezerwacji |
| Przebieg | 1. Należy kliknąć przycisk „Cancel”   znajdujący się przy wybranej rezerwacji   1. Zmiana statusu rezerwacji w bazie danych 2. Odświeżenie widoku rezerwacji |

* + 1. Przypadki użycia dla użytkownika typu Właściciel



Rysunek 4 Diagram przypadków użycia – właściciel

Funkcjonalności serwisu dedykowane właścicielom restauracji są dostępne jedynie poprzez aplikację internetową.

* + - 1. PU Utworzenie konta

Tabela 10 PU Utworzenie konta właściela

|  |  |
| --- | --- |
| WS | Otwarcie strony WWW serwisu |
| WK | Utworzenie nowego konta właściciela lub komunikat o błędzie |
| Cel | Rejestracja nowego użytkownika typu Właściciel |
| Przebieg | 1. Przejście do formularza rejestracji właściciela 2. Podanie danych osobowych 3. Podanie adresu email oraz hasła 4. Weryfikacja czy wszystkie pola zostały wypełnione 5. Weryfikacja czy w serwisie nie istnieje już właściciel   o tym samym adresie email   1. Utworzenie nowego konta właściciela lub komunikat   o błędzie |

* + - 1. PU Zalogowanie

Tabela 11 PU Logowanie właściciela

|  |  |
| --- | --- |
| WS | Otwarcie strony WWW serwisu |
| WK | Zalogowanie właściciela lub komunikat o błędzie |
| Cel | Zalogowanie właściela oraz utworzenie tokena sesji |
| Przebieg | 1. Przejście do formularza logowania właściciela 2. Podanie adresu email oraz hasła 3. Weryfikacja czy właściciel o podanym adresie   email istnieje w serwisie oraz czy skrót hasła  zapisanego w bazie zgadza się ze skrótem hasła  wprowadzonego podczas logowania   1. Jeżeli podane dane nie są poprawne wyświetlenie   komunikatu o błędzie, w przeciwnym przypadku  wygenerowanie tokena.   1. Zapisanie tokena w pamięci przeglądarki 2. Ustawienie praw właściciela dla zalogowanego   użytkownika   1. Wyświetlenie widoku panelu właściciela |

* + - 1. PU Wylogowanie właściciela

Tabela 12 PU Wylogowanie właściciela

|  |  |
| --- | --- |
| WS | PU Zalogowanie właściciela |
| WK | Usunięcie tokena z pamięci przeglądarki, usunięcie  praw właściciela dla aktywnej sesji |
| Cel | Wylogowanie właściciela |
| Przebieg | 1. Naciśnięcie przycisku „Logout” z menu 2. Usunięcie tokena z pamięci przeglądarki 3. Usunięcie uprawnień właściciela 4. Przekierowanie użytkownika na stronę główną |

* + - 1. PU Dostęp do listy restauracji

Tabela 13 Dostęp do listy restauracji

|  |  |
| --- | --- |
| WS | PU Zalogowanie |
| WK | Wyświetlenie widoku panelu właściciela |
| Cel | Wyświetlenie listy restauracji danego właściciela |
| Przebieg | 1. Naciśnięcie przycisku „Dashboard” w menu 2. Wyświetlenie widoku panelu właściciela |

* + - 1. PU Dodanie nowej restauracji do serwisu

Tabela 14 PU Dodanie nowej restauracji do serwisu

|  |  |
| --- | --- |
| WS | PU Dostęp do listy restauracji |
| WK | Utworzenie rekordu restauracji w bazie danych serwisu |
| Cel | Dodanie nowej restauracji do serwisu |
| Przebieg | 1. Wyświetlenie formularza dodawania   nowej restauracji   1. Podanie podstawowych informacji o restauracji:    * nazwa restauracji    * adres    * telefon, adres email    * rodzaj serwowanej kuchni    * godziny otwarcia 2. Podanie opisu restauracji 3. Podanie informacji na temat karty dań:    * nazwa dania    * opis    * cena 4. Weryfikacja czy wszystkie pola zostały wypełnione 5. Zapisanie restauracji w bazie danych 6. Wyświetlenie widoku restauracji |

* + - 1. PU Wyświetlenie szczegółów wybranej restauracji

Tabela 15 PU Wyświetlenie szczegółów wybranej restauracji

|  |  |
| --- | --- |
| WS | PU Dostęp do listy restauracji |
| WK | Wyświetlenie szczegółowego widoku wybranej restauracji |
| Cel | Dostęp do szczegółów restauracji |
| Przebieg | 1. Wybranie restauracji z listy restauracji właściciela 2. Pobranie informacji o restauracji z bazy danych 3. Wyświetlenie widoku szczegółów restauracji |

* + - 1. PU Zarządzanie rezerwacjami restauracji

Tabela 15 PU Zarządzanie rezerwacjami restauracji

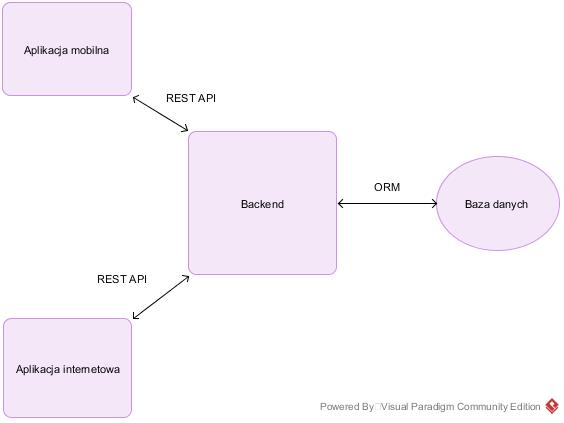
|  |  |
| --- | --- |
| WS | PU Wyświetlenie szczegółów wybranej restauracji |
| WK | Wyświetlenie listy wszystkich rezerwacji w systemie powiązanych  z wybraną restauracją |
| Cel | Zarządzanie rezerwacjami |
| Przebieg | 1. Wciśnięcie przycisku „All reservations” w widoku   szczegółów restauracji   1. Wyświetlenie widoku listy rezerwacji powiązanych z daną   restauracją posortowanych według daty utworzenia oraz  przyciskami umożliwiającymi zmianę statusu rezerwacji |

* + - 1. PU Edycja danych wybranej restauracji

Tabela 17 PU Edycja danych wybranej restauracji

|  |  |
| --- | --- |
| WS | PU Wyświetlenie szczegółów wybranej restauracji |
| WK | Aktualizacja danych restauracji |
| Cel | Zarządzanie danymi restauracji |
| Przebieg | 1. Naciśnięcie przycisku „Edit” w widoku szczegółów   restauracji   1. Wyświetlenie formularza edycji restauracji 2. Wprowadzenie zmian w danych restauracji 3. Weryfikacja czy wszystkie pola zostały poprawnie   wypełnione   1. Naciśnięcie przycisku „Save” 2. Aktualizacja danych restauracji 3. Przekierowanie użytkownika do widoku   szczegółów restauarcji |

* 1. Architektura systemu

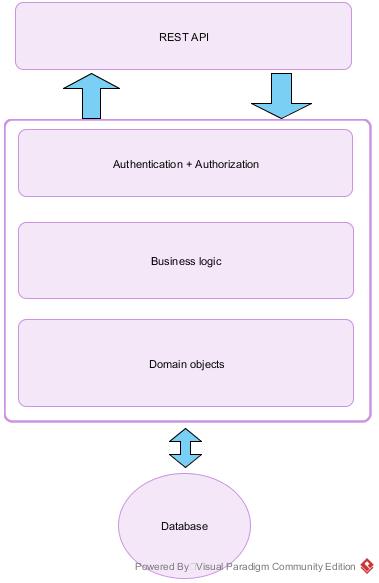


Rysunek 5 Schemat logiczny systemu

Powyższy rysunek przedstawia schemat logiczny systemu. W jego centrum jest aplikacja nazwana Beckend będąca częścią serwerową dla aplikacji klienckich. Najważniejszymi zadaniem tej aplikacji test dostarczanie dostępu do bazy danych oraz zapewnienie logiki biznesowej.

Aplikacjami klienckimi w tym systemie jest aplikacja internetowa SPA oraz aplikacja mobilna. Obie aplikacje wykorzystują do komunikacji z serwerem protokół HTTP, a dokładniej interfejs REST.

* + 1. Backend



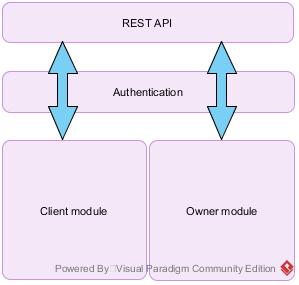
Rysunek 6 Schemat logiczny aplikacji Backend

Rysunek 6 przedstawia zarys architektury aplikacji dostarczającej REST API. Interfejs zaznaczony jako REST API dostępny będzie dla aplikacji klienckich. Będzie składał się on z kontrolerów obsługujących zapytania HTTP. Warstwa ta ma docelowo być dostępna w Internecie, dlatego musi zostać zabezpieczona.

Część zaznaczona jako Authentication + Authorization odpowiedzialna będzie za kontrolę dostępu do aplikacji. W niej generowane będą nowe tokeny użytkowników. Logikę biznesową aplikacji symbolizuje fragment Buisiness logic.

Wykorzystanie technologii ORM wymaga utworzenia klas encji w aplikacji. Zbiór tych obiektów znajdował się będzie w części Domain. Na podstawie pól tych obiektów zostanie wygenerowany schemat bazy danych.

* + 1. Aplikacja internetowa SPA

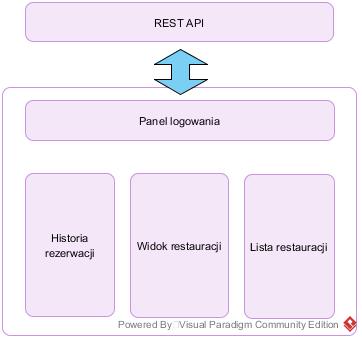


Rysunek 7 Aplikacja SPA schemat logiczny

Aplikacja internetowa do komunikacji z serwerem będzie wykorzystywała REST API. Moduł Authentication będzie odpowiedzialny za bezpieczeństwo aplikacji, zarządzanie tokenami użytkowników oraz tworzenie nagłówków zapytań HTTP z danymi autoryzacyjnymi.

Aplikacja internetowa będzie dostarczała funkcjonalności opisanych w przypadkach użycia dla użytkownika typu klient oraz właściciel. Moduł klienta będzie dostarczał funkcjonalności określonych w przypadkach użycia klienta, a właściciel odpowiednio właściciela.

* + 1. Aplikacja mobilna



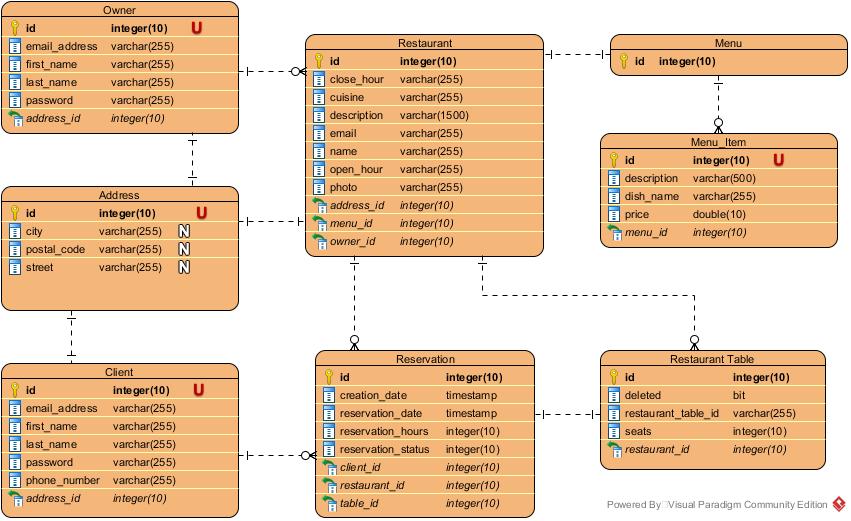
Rysunek 8 Schemat logiczny aplikacji mobilnej

Funkcjonalności aplikacji mobilnej będą pokrywały się z funkcjonalnościami dostarczonymi przez moduł klienta aplikacji internetowej.

Powyższy rysunek pokazuje moduły na jakie aplikacja zostanie podzielona. Podobnie jak w aplikacji internetowej komunikacja z serwerem będzie realizowana za pomocą protokołu HTTP. Moduł logowania ma odpowiadać za logowanie użytkownika oraz przechowywanie tokena sesji.

W module Historia rezerwacji dostępne będą wszystkie rezerwacje klienta, w module widok restauracji będzie można zobaczyć szczegóły wybranej restauracji. Na dostęp do listy restauracji w serwisie pozwalał będzie moduł Lista restauracji.

* + 1. Baza danych



Rysunek 9 Schemat bazy danych

Powyższy rysunek przedstawia schemat ideowy bazy danych. Prezentuje tabele jakie znajdą się w bazie danych oraz powiązania między nimi. Rzeczywisty schemat zostanie wygenerowany za pomocą narzędzia ORM i w zależności od wybranej bazy danych może różnić się w niewielkim stopniu np. w typach pól tabel ich lub nazwach.

* + 1. Zabezpieczenie komunikacji w systemie

Komunikacja pomiędzy aplikacjami przebiegać będzie za pomocą zapytań HTTP, z tego względu potrzebne będzie odpowiednie zabezpieczenie przed nieautoryzowanymi użytkownikami. W tym celu wykorzystana zostanie technologia JSON Web Token. Podczas wysyłania zapytań HTTP do nagłówka żądania zostanie dołączony token, którego poprawność jest sprawdzana w aplikacji serwera. Opis tego rozwiązania znajduje się w następnym rozdziale.

1. Wykorzystane technologie
   1. Spring Boot

Spring Boot jest jednym z projektów wchodzących w skład rodziny frameworka Spring. Został on zaprojektowany z myślą o tworzeniu aplikacji internetowych. W skład standardowego pakietu Spring Boot wchodzi serwer WWW Apatche Tomcat, co znacznie ułatwia wdrażanie aplikacji internetowych.

* + 1. Spring Boot Web Starter

Spring Boot Web Starter jest podstawowym pakietem do tworzenia aplikacji internetowych w Spring Boot. To właśnie w jego skład wchodzi wbudowany serwer Tomcat oraz pakiet Spring Web MVC pozwalający na tworzenie aplikacji z wykorzystujących Model View Controller. Poza tworzeniem aplikacji MVC Spring Web umożliwia tworzenie REST API.

* + 1. Spring Data

Pakiet Spring Data dostarcza narzędzi potrzebnych do pracy z bazą danych. W jego skład wchodzi między innymi Hibernate, czyli framework ORM zapewniający translację obiektów aplikacji Java na rekordy w bazie danych i odwrotnie.

* + 1. Spring Security

Spring Security to zbiór narzędzi ułatwiających wdrożenie kontroli dostępu w aplikacji wykorzystującej framework Spring. Dostarcza on mechanizmy potrzebne do uwierzytelniania użytkowników oraz ich autoryzacji. Pozwala on na integrację z wieloma standardami autoryzacji takimi jak OAuth, Kerberos czy wykorzystany w tej pracy JSON Web Token. W skład Spring Security wchodzą również narzędzia do zarządzania rolami użytkowników w aplikacji oraz mechanizmy hashowania haseł.

* + 1. Inne narzędzia do tworzenia aplikacji internetowych

Spring Boot nie jest jedynym pakietem na rynku, za pomocą którego można stworzyć aplikacje internetowe. ASP.NET MVC firmy Microsoft dostarcza narzędzi do tworzenia aplikacji internetowych opartych o model MVC w języku C#. Podobne efekty można uzyskać wykorzystując Django – framework języka Python.

* 1. Angular

Angular to platforma do tworzenia aplikacji internetowych typu SPA stworzona przez firmę Google. Framework ten rozwijany jest na licencji MIT, tak więc wykorzystywanie go w celach komercyjnych jak i niekomercyjnych nie wiąże się dodatkowymi opłatami.

Jako najważniejsze cechy frameworka Angular należy wymienić:

* Podział aplikacji na komponenty. Komponent w Angularze jest najmniejszą składową aplikacji np. formularz, tabela. Komponenty są tworzone w taki sposób, aby ich wielokrotne użycie było możliwie proste
* Serwisy, które są klasami dostarczającymi funkcjonalności dla komponentów w aplikacji. Takim serwisem może być klasa realizująca dostęp do REST API. Dzięki wstrzykiwaniu zależności zostanie stworzony tylko jeden obiekt takiego serwisu, a referencja do niego zostanie automatycznie przekazana do komponentów, które wymagają go do poprawnego działania.
* Mechanizm szablonów HTML. Każdy komponent posiada własny szablon HTML, który stanowi widok danego komponentu. Szablony tworzone są w języku HTML, jednak dzięki składni Angulara istnieje możliwość dynamicznego wstawiania do nich informacji z kontrolera. Pozwala to na wyświetlanie zawartości zmiennych kontrolera w szablone HTML.
  1. Ionic

Ionic jest zestawem narzędzi programistycznych pozwalającym na tworzenie aplikacji mobilnych z wykorzystaniem technologii JavaScript, HTML oraz CSS. Ionic wykorzystuje technologie Angular, co pozwala na przeniesienie części kodu odpowiedzialnego za aplikację internetową do aplikacji mobilnej.

Aplikacje tworzone z wykorzystaniem Ionic budowane są za pomocą komponentów udostępnianych wraz z platformą. W bibliotece Ionic można znaleźć komponenty oraz wtyczki odpowiedzialne za pobieranie danych od użytkownika np. pola tekstowe lub wyboru daty. Zawiera ona również elementy takie jak gotowe listy, schematy widoków poszczególnych stron oraz wiele innych.

Budowanie aplikacji na urządzeniach mobilnych możliwe jest dzięki wykorzystaniu technologii Apatche Cordova. Jest ona podstawą działania platformy Ionic. Cordova dostarcza aplikację opakowującą kod HTML oraz JavaScript w taki sposób, żeby użytkownik miał wrażenie korzystania z natywnej aplikacji na jego urządzenie. Ponadto Cordova dostarcza mechanizmów dostępu do zasobów urządzeń takich jak aparat fotograficzny, odbiornik GPS i wiele innych.

* 1. PostgreSQL

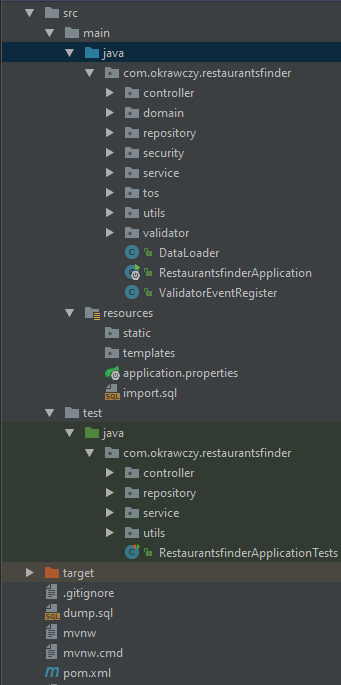
PostgreSQL jest jedną z najpopularniejszych relacyjnych baz danych. Dostępna jest na licencji MIT, dzięki czemu wykorzystywanie jej w celach komercyjnych, jak i prywatnych jest darmowe. PostgreSQL posiada wiele zalet między innymi bogaty zbiór obsługiwanych typów danych. Pozwala między innymi na przechowywanie oraz wykonywanie operacji na obiektach typu JSON. Ponadto użytkownik korzystający z tej bazy ma możliwość definiowania własnych typów danych.

Należy wspomnieć, że w tej pracy tej zostanie wykorzystany mechanizm ORM, tak więc sam wybór bazy danych nie pełni kluczowej roli.

1. Implementacja

Rozdział ten opisuje implementację serwisu z podziałem na poszczególne aplikacje. Zostaną w nim omówione wyłącznie najbardziej istotne fragmenty aplikacji w szczególności te odpowiedzialne za realizację logiki biznesowej, komunikacji z innymi systemami oraz mechanizmy zabezpieczeń.

* 1. Backend – aplikacja Spring Boot



Rysunek 10 Pakiety aplikacji Backend

Rysunek 10 pokazuje podział aplikacji na pakiety. W katalogu „main” znajdują się pliki źródłowe aplikacji, a w katalogu „tests” testy.

* + 1. Pakiet „controller”

Pakiet ten zawiera kontrolery odpowiedzialne za obsługę zapytań HTTP wysyłanych do aplikacji. Pliki w tym pakiecie zostały podzielone na klasy zgodnie z ich rodzajem realizowanych przez nie funkcjonalności. Przychodzące zapytania obsługiwane są przez metody tych klas, to w nich realizowana jest logika biznesowa.

W skład pakietu „controllers” wchodzą klasy:

* ClientController – obsługuje zapytania związane z użytkownikami typu klient. Pozwala na rejestrowanie nowych użytkowników oraz pobranie danych użytkowników znajdujących się w bazie.
* OwnerController – realizuje funkcjonalności dotyczące właścicieli, zakładanie nowych kont, pobranie danych właściciela oraz pobranie listy restauracji właściciela na podstawie jego adresu email.
* ReservationController – tutaj obsługiwane są zapytanie dotyczące obsługi rezerwacji. Między innymi tworzenie nowych rezerwacji, pobieranie listy rezerwacji dla danej restauracji lub użytkownika oraz zmiany statusów
* RestaurantController – w klasie tej znajdują się metody pozwalające na obsługę restauracji, dodawanie nowych, edycję danych istniejących restauracji oraz wyszukiwanie ich na podstawie nazwy
* RestaurantTableController – zarządzanie stolikami w restauracji, pobieranie stolików wybranej restauracji, usuwanie oraz dodawanie nowych. Ważnym zadaniem, które realizuje ta klasa jest sprawdzanie czy usuwany stolik nie ma aktywnych rezerwacji.

Listing. 1. Metoda kontrolera obsługującego pobieranie listy rezerwacji

@CrossOrigin

@GetMapping("/reservations/restaurant/{id}")

public ResponseEntity<?> getReservationsByRestaurant(

@PathVariable(value = "id") Long id) {

List<Reservation> reservations =

reservationRepository.findReservationsByRestaurant\_Id(id);

List<ReservationTO> result = reservations.stream()

.map(p -> reservationTOConverter.convertToTO(p))

.collect(Collectors.toList());

return ResponseEntity.ok(result);

}

Powyższy listing pokazuje jedną z metod klasy ReservationController. Rodzaj obsługiwanych zapytań oraz ich adres definiowany jest poprzez adnotacje języka Java. W tym przypadku zdefiniowany jest kontroler obsługujący zapytania GET wysyłane na adres „/reservations/restaurant/”. Jako parametr zapytania musi przekazany zostać numer id restauracji. Dla przykładu zapytanie GET zwracające listę rezerwacji przypisanych do restauracji o id 1 będzie miało postać „/reservations/restaurant/1”. Metoda przeszukuje bazę danych wykorzystując repozytoria JPA i zwraca wynik w formacie JSON.

* + 1. Pakiet „domain”

W tym pakiecie znajdują się klasy obiektów jakie będą przechowywane w bazie danych. Z tych obiektów mechanizm ORM stworzy tabele w bazie danych. Wiersze odczytane z bazy danych będą konwertowane na obiekty klas znajdujących się tym pakiecie. Cały proces translacji obiektów przebiega automatycznie. Wszystkie klasy, które mają zostać przechowane w bazie muszą mieć adnotację @Entity, tabele w bazie danych będą nazywały się tak jak klasy. Istotne są również typy pól klasy, ponieważ na ich podstawie definiowane są typy kolumn w bazie danych.

Klasy wchodzące w skład tego pakietu to między innymi:

* Owner - klasa użytkownika typu właściciel
* Restaurant – klasa opisujaca restaurację
* Client – klasa definiująca klienta

Listing. 2. Klasa Client

@Entity

public class Client {

@Id

private Long id;

@Column(nullable = false)

private String firstName;

@Column(nullable = false)

private String lastName;

@Column(nullable = false, unique = true)

private String emailAddress;

@Column(nullable = false)

private String password;

@OneToOne

private Address address;

@Column

private String phoneNumber;

@OneToMany(mappedBy = "client")

private Collection<Reservation> reservations;

Powyższy listing pokazuje implementację obiektu Client wraz z konfiguracją potrzebą do działania ORM Hibernate. Widać na nim również w jaki sposób tworzone relacje pomiędzy tabelami oraz jak zdefiniowane są parametry kolumn. @Id wskazuje, że to pole ma być kluczem głównym, @Column(nullable = false) powoduje, że kolumna oznaczona zostanie jako NOT NULL, @OneToMany definiuje relację pomiędzy obiektami, a w szczególności tabelami w bazie danych. Na listingu nie zostały wymienione metody dostępu do zmiennych, które są wykorzystywane przez ORM.

* + 1. Pakiet „repository”

W pakiecie „repository” znajdują się interfejsy repozytoriów JPA. Repozytoria te odpowiadają za pobieranie oraz zapisywanie obiektów do bazy danych. Każdy obiekt w pakiecie „domain” ma odpowiadające sobie repozytorium. Nazwy metod zawarte w interfejsach definiują jakie operacje zostaną przez nie wykonane.

Listing. 3. Repozytorium ClientRepository

public interface ClientRepository extends PagingAndSortingRepository<Client, Long> {

Client findClientById(Long id);

Client findClientByEmailAddressIgnoreCase(String emailAddress);

List<Client> findByEmailAddress(String emailAddress);

}

* + 1. Pakiet „security”

W pakiecie „security” znajdują się klasy konfigurujące działanie pakietu Spring Security. Najważniejszymi klasami w tym pakiecie są filtry uwierzytelniania JWTAuthenticationFilter oraz JWTAuthenticationFilterOwner. Obsługują one logowanie użytkowników oraz generację tokentów JWT. Filtry te odbierają zapytania przesyłane na adresy logowania użytkowników. Jako dane takiego zapytania w formacie JSON przesyłany jest adres email użytkownika oraz hasło. Z bazy danych pobierany jest obiekt użytkownika w tym skrót jego hasła. Skrót ten następnie jest porównywany ze skrótem odebranego hasła. Jeżeli są zgodne następuje wygenerowanie tokena. W tokenie JWT zapisany jest adres email użytkownika, data wygaśnięcia tokena oraz uprawnienia jakie posiada użytkownik. Utworzony token wysyłany do użytkownika.

Działanie JWT zostało opisane w dokumencie RFC-7519. Przykładowy token ma następującą postać:

Listing. 4. Token wygenerowany dla jednego z użytkowników

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJwYXdlbEBnbWFpbC5jb20iLCJvd25lciI6ImZhbHNlIiwiZXhwIjoxNTExNDQ3Njc4fQ.vjppV-4lNdCpP\_HZuJ\_6UkDuN04jS7\_R1d796KoKkHybOJNeHm0iser4KKYgNuLkiwWLOoD6L8RqspWeeWt2bA

Składa się on z trzech części oddzielonych kropkami. Pierwsza część to nagłówek definiujący rodzaj algorytmu jaki został użyty do wygenerowania podpisu, w drugiej zawarte są dane które dostarcza użytkownik. Zarówno nagłówek jak i dane zakodowane są algorytmem base64. Trzecia część to podpis tokenu. Podpis wygenerowany jest za pomocą funkcji skrótu, której parametrami są nagłówek, dane oraz sekretny klucz podany podczas konfiguracji pakietu.

Powyższy token w sekcji danych przechowuje następujące informacje:

{

"sub": "pawel@gmail.com",

"owner": "false",

"exp": 1511447678

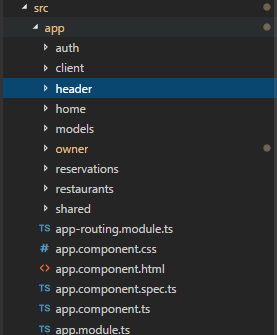
}

Warto zauważyć, że w tokenie nie mogą być przesyłane newralgiczne informacje, ponieważ do ich zakodowania używany jest algorytm base64. Istotą działania tego mechanizmu jest podpis, którego poprawność jest sprawdzana przy każdym zapytaniu przesłanym do serwera. Dokładną specyfikację działania JWT opisuje dokument RFC-7519.

* + 1. Pozostałe pakiety

Wymienione powyżej pakiety są podstawą działania aplikacji serwerowej, realizują najważniejsze funkcjonalności takie jak dostarczenie REST API, dostęp do bazy danych czy zapewnienie bezpieczeństwa. Pozostałe pakiety zostaną opisane poniżej.

* „tos” – pakiet zawiera definicję klas typu Transfer Objects. Obiekty z klasy domen są konwertowane do obiektów TO przed wysłaniem ich do użytkownika. Ma to na celu zredukowanie ilości przesyłanych informacji oraz zmniejszenie liczby powiązań pomiędzy obiektami wysyłanymi do klientów. Na przykład obiekt OwnerTO pozbawiony jest pola „password” przechowującego informacje o haśle użytkownika.
* „service” – zawiera w sobie klasy pomocnicze dla mechanizmu uwierzytelniania użytkowników, między innymi ich wyszukiwanie na podstawie adresu email. Znajduje się w nim również klasa wyszukująca dostępne stoliki.
* „utils” – zawiera klasy pozwalające na konwersję klas z pakietu „domain” na Transfer Objects. W tym pakiecie znajduje się również klasa automatycznie zmieniająca status rezerwacji na „CLOSED” po jej upłynięciu.
  1. Aplikacja internetowa



Rysunek 11 Moduły aplikacji internetowej

Aplikacja internetowa napisana z wykorzystaniem frameworka Angular podzielona jest na foldery oraz moduły. Znajdują się w nich komponenty, szablony HTML, arkusze stylów oraz w serwisy udostępniające dane dla komponentów. Wszystkie moduły oraz komponenty dołączane są do aplikacji w pliku app.module.ts, który jest korzeniem aplikacji. Poniżej zostaną opisane poszczególne moduły oraz komponenty.

* + 1. Moduł auth

Wszystkie komponenty, serwisy oraz formularze związane z logowaniem oraz rejestracją użytkowników są zawarte w module auth.

Moduł ten zawiera klasę AuthService dostarczającą serwis dostępny w całej aplikacji. Jego zadaniem jest przechowywanie informacji o aktualnie zalogowanym użytkowniku i jego uprawnieniach. Serwis ten odpowiada również za generowanie nagłówków HTTP z zapisanym tokenem autoryzacji.

Poza logowaniem klientów oraz właścicieli moduł auth definiuje formularze rejestracji nowych użytkowników oraz jest odpowiedzialny za odczyt tokenu zapisanego w pamięci przeglądarki.

W folderze guards znajduje się implementacja serwisu przyznająca dostęp do poszczególnych sekcji aplikacji. Dzięki temu serwisowi tylko zalogowani klienci mają dostęp do listy rezerwacji, mogą dokonywać rezerwacji i nie mają dostępu do panelu właściciela.

* + 1. Moduł owner

Moduł owner zawiera wszystkie komponenty tworzące interfejs właściciela. Wchodzący w jego skład komponent DashboardComponent jest panelem właściciela służącym do zarządzania restauracjami. Składają się na niego: formularz edycji oraz dodawania nowej restauracji, widok szczegółów restauracji, widok listy rezerwacji oraz pasek boczny z listą restauracji właściciela.

* + 1. Folder restaurants

Folder ten zawiera komponent wyświetlający szczegóły restauracji dostępne dla użytkownika. Dostęp do niego jest możliwy z listy wyników wyszukiwania bądź bezpośrednio po przejściu na adres „/restaurant/[id restauracji]”. Wyświetla on podstawowe informacje o restauracji takie jak nazwa, adres, godziny otwarcia oraz opis. Ponadto zawiera kartę dań restauracji oraz formularz wyszukiwana stolików. Z poziomu tego komponentu możliwa jest rezerwacja stolików.

* + 1. Models

Folder models zawiera modele niektórych obiektów wykorzystywanych w aplikacji. Znajduje się w nim między innymi model User przechowujący dane aktualnie zalogowanego użytkownika.

Listing. 10. Model User aplikacji internetowej

export class User {

id: number;

firstName: string;

lastName: string;

emailAddress: string;

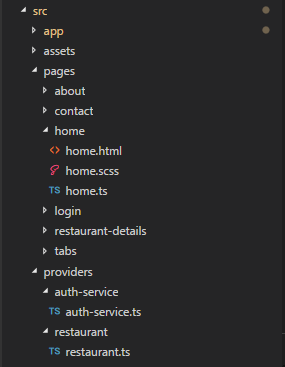
phoneNumber: string;

}

* + 1. Reservations

W folderze tym znajduje się implementacja serwisu dostarczającego funkcjonalności zarządzania rezerwacjami.

* 1. Aplikacja mobilna



Rysunek 14 Struktura aplikacji mobilnej

Do tworzenia aplikacji mobilnych z wykorzystaniem platformy Ionic wykorzystuje się framework Angular. Z tego powodu struktura projektu aplikacji mobilnej jest podobna do przedstawionej wcześniej aplikacji internetowej.

W folderze „pages” znajdują się strony opisujące ekrany, do jakich użytkownik będzie miał dostęp. Podobnie jak w aplikacjach internetowych składają się one z komponentów.

* + 1. Strona login

Aby było możliwe korzystanie z aplikacji mobilnej, należy się do niej zalogować. Pierwszą stroną wyświetlaną użytkownikowi jest strona login, która zawiera formularz logowania. Dane wprowadzone przez użytkownika są przesyłane do serwisu AuthProvier, które z buduje zapytanie do serwera REST. Jeżeli wprowadzone dane są poprawne serwis ten zapisuje otrzymany token JWT w pamięci aplikacji.

* + 1. Strona about

W tym module znajdują znajduje się implementacja strony pozwalającej użytkownikowi na wyświetlenie listy jego rezerwacji oraz zarządzanie nimi. Lista rezerwacji pobierana jest z serwera za pośrednictwem serwisu RestaurantProvider.

* + 1. Strona home

Strona home umożliwia użytkownikowi wyszukiwanie restauracji. Podobnie jak w przypadku strony about, dane pobrane w formularzu przekazywane do jednej z metod RestauratProvier, która wysyła zapytanie o listę restauracji spełniających podane kryteria.

* + 1. Strona restaurat details

Restaurant details wyświetla szczegóły wybranej restauracji, takie jak menu, opis czy adres. Na tej stronie można również zarezerwować jeden z dostępnych stolików.

Listing. 12. Metoda RestaurantProvider pobierająca dostępne stoliki

getAvailableSlots(date, restaurantId, seats) {

let params = new URLSearchParams();

params.set('seats', String(seats));

params.set('restaurantId', String(restaurantId));

params.set('date', date);

console.log(params);

return this.http.get(baseURL + 'reservations/availableSlots', { params: params, headers: this.authService.getHeaders() });

}

Powyższa metoda pokazuje w jaki sposób pobierane są dostępne stoliki z serwera aplikacji.

Listing. 13. Metoda RestaurantDetailsPage

onReserveTable(table) {

this.restaurantService.makeReservation(table.reservationDate,

table.restaurantTable.restaurantId, table.restaurantTable.id).

subscribe(

data => {

console.log(data);

this.getAvailableTables();

this.navCtrl.parent.select(1);

});}

Metoda onReserveTable wysyła zapytanie do serwera, które tworzy nową rezerwację w serwisie. Po dokonaniu rezerwacji użytkownik przenoszony jest na widok pokazujący jego wszystkie rezerwacje.

1. Prezentacja działania serwisu
   1. Aplikacja serwerowa

Głównym zadaniem części serwerowej jest dostarczanie interfejsu REST. Opis dostępnych adresów interfejsu został wygenerowany z pomocą narzędzia Swagger i dostępny pod adresem „/swagger-ui.html”.

Poniżej zostaną opisane interfejsy udostępniane przez poszczególne kontrolery. Obiekty przesyłane do serwera oraz zwracane przez serwer serializowane są do formatu JSON.

* + 1. Client Controller

Tabela 16 Interfejs ClientController

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /clients/getByEmail | Zapytanie to wyszukuje klienta według adresu email podanego w parametrze, a następnie zwraca jego dane |
| POST | /clients/new | Tworzy nowego użytkownika na postawie obiektu przesłanego w ciele zapytania. |

* + 1. Helper Controller

Tabela 17 Interfejs HelperController

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /availableCuisines | Zwraca listę rodzajów kuchni w serwisie. Dane te potrzebne są do list wyboru aplikacji |
| GET | /restaurantsCities | Zwraca listę miast w dostępne są restauracje |

* + 1. Owner Controller

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /owners/getByEmail | Zapytanie to wyszukuje właściciela według adresu email podanego w parametrze, a następnie zwraca jego dane |
| GET | /owners/{email}/  restaurants/stubs | Zwraca listę obiektów zawierających podstawowe informacje o restauracjach właściciela takie jak id oraz nazwa. W miejsce {email} należy podać adres email właściciela |
| POST | /owners/new | Tworzy nowego właściciela na postawie obiektu przesłanego w ciele zapytania. |

* + 1. Reservation Controller

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /reservations/  availableSlots | Zwraca listę wolnych stolików dla wybranej restauracji. Jako parametry należy podać id restauracji, datę oraz liczbę miejsc |
| GET | /reservations/client/{id} | Zwraca listę rezerwacji użytkownika o id podanym w miejscu {id} |
| GET | /reservations  /pendingReservations/{id} | Zwraca listę rezerwacji w statusie PENDING dla restauracji, której id przesłane jest w adresie |
| GET | /reservations  /restaurant/{id} | Zwraca listę wszystkich rezerwacji restauracji o numerze id przekazanym w miejscu {id} |
| POST | /reservations/accept | Zmienia status rezerwacji na ACCEPTED.W parametrze zapytania należy przesłać id rezerwacji |
| POST | /reservations/cancel | Zmienia status rezerwacji na CANCELED.W parametrze zapytania należy przesłać id rezerwacji |
| POST | /reservations/reject | Zmienia status rezerwacji na REJECTED.W parametrze zapytania należy przesłać id rezerwacji |
| POST | /reservations  /makeReservation | Tworzy nową rezerwację. W ciele należy przesłać obiekt zawierający adres email klienta, id restauracji, id stolika oraz datę rezerwacji |

* + 1. Restaurant Controller

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /restaurants/{id} | Zwraca obiekt zawierający informacje o restauracji. W miejsce {id} należy wpisać id restauracji |
| POST | /restaurants  /findByParameters | W odpowiedzi zwraca listę restauracji spełniających kryteria podane w przekazanym obiekcie. Ten kontroler odpowiedzialny jest za wyszukiwanie restauracji w których znajdują się wolne stoliki. Jako parametry należy podać miasto, w którym restauracje mają być wyszukiwane, liczbę gości, rodzaj kuchni oraz datę. |
| POST | /restaurants/new | Dodaje nową restaurację do bazy. W ciele zapytania należy przesłać obiekt restauracji. |
| POST | /restaurants/{id} | Aktualizuje dane restauracji o id przekazanym w miejscu {id}. W zapytaniu należy przesłać obiekt restauracji po wprowadzonych zmianach |

* + 1. Restaurant Table Controller

|  |  |  |
| --- | --- | --- |
| Typ zapytania | Adres | Opis |
| GET | /tables/restaurant/{id} | Zwraca listę stolików restauracji, której id podane jest w miejscu {id} |
| POST | /tables/delete/{id} | Usuwa stolik z id podanym w adresie |
| POST | /tables/add | Dodaje nowy stolik do bazy. W zapytaniu należy przesłać dane stolika takie jak: liczba miejsc, id restauracji oraz id stolika w restauracji |

Zapytania do wszystkich wymienionych powyżej kontrolerów za wyjątkiem opisanych w tabeli Helper Controllers wymagają autoryzacji.

Przewidziane zostały również dwa kontrolery dzięki którym użytkownicy mogą zalogować się do serwisu. Dla klienta jest to „/login”, dla właściciela „/owners/login. Obsługują one zapytania typu POST, a w sekcji danych należy podać obiekt przechowujący adres email (emailAddress) oraz hasło (password).

* 1. Aplikacja internetowa
     + 1. Sekcja poziomu 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Tab. 1. Przykład podpisu tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

* 1. Kolejna sekcja poziomu 1
     1. Kolejna sekcja poziomu 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Rys. 1. Przykład podpisu rysunku

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Listing. 1. Początkowe żądanie HTTP

GET /script/Articles/Latest.aspx HTTP/1.1  
Host: www.codeproject.com  
Connection: keep –alive  
Cache -Control: max-age=0  
Accept: text/html ,application/xhtml+xml,application/xml|  
User -Agent: Mozilla/5.0 ...  
Accept -Encoding: gzip ,deflate ,sdch  
Accept -Language: en-US...  
Accept -Charset: windows -1251,utf -8...

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus.

Przykład listy numerowanej:

1. Podsumowanie i wnioski

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dapibus enim. Etiam lobortis pulvinar enim in maximus. Aliquam erat volutpat. Integer maximus est turpis, ut bibendum ligula accumsan et. Ut eget vestibulum libero. Aliquam erat volutpat. Nullam placerat mauris a lectus tincidunt, et aliquet turpis aliquam. Etiam in malesuada lacus. Proin dignissim augue sit amet auctor elementum. Suspendisse potenti. Vivamus suscipit vulputate massa ac molestie. Suspendisse a justo porttitor, commodo mi at, placerat risus. Integer lobortis augue ac neque suscipit, vel sodales lacus fringilla.

# Literatura

[1] M. Bickley, C. Slominski. A MySQL-based data archiver: preliminary results. Proceedings of ICALEPCS07, Paz. 2007. http://www.osti.gov/scitech/servlets/purl/922267 [dostęp dnia 20 czerwca 2015].

[2] J. Jędrzejczyk, B. Sródka. Segmentacja obrazów metodą drzew decyzyjnych. Raport instytutowy, Politechnika Wrocławska, Wydział Elektroniki, 2007.

# Dodatek A