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)

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

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

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

**UDDI** (ang. *Universal Description Discovery and Integration*)

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

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

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

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

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

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

**GML** (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 aplikacji dostarczającej interfejs dostępu do bazy danych.

* 1. Wprowadzenie

Znaczna część serwisów internetowych oraz aplikacji mobilnych powstaje, aby ułatwić użytkownikom codzienne czynności. Dla przykładu można wymienić aplikacje takie jak Any.do pozwalającą na tworzenie przejrzystych list zadań. Istnieją również aplikacje pozwalające w kilku kliknięciach zamówić taksówkę. Powstają również serwisy pozwalające zamówić jedzenie online. Podczas realizacji pracy dyplomowej stworzę serwis internetowy ułatwiający rezerwację stolików klientom restauracji za pośrednictwem przeglądarki internetowej oraz aplikacji mobilnej. Serwis ten będzie również wspierał właścicieli restauracji pozwalając im na dodawanie ich restauracji oraz zarządzanie pojawiającymi się rezerwacjami.

* 1. Cel i zakres pracy

Celem tej pracy jest zaprojektowanie oraz implementacja systemu składającego się z bazy danych, aplikacji mobilnej, aplikacji internetowej typu Single Page Application oraz aplikacji dostarczającej interfejs do komunikacji z bazą danych. Aplikacja mobilna zostanie stworzona z wykorzystaniem narzędzia Ionic, które pozwala na tworzenie aplikacji działających na różnych platformach mobilnych. Mobilna części serwisu będzie dedykowana klientom restauracji, za jej pomocą będą mogli dokonać rezerwacji stolika. Aplikacja internetowa będzie dostarczała interfejs klienta oraz właściciela restauracji. Ta część systemu będzie udostępniała funkcjonalności aplikacji mobilnej oraz interfejs dla właściciela restauracji. Ponieważ wymienione wyżej części składowe systemu uruchamiane będą na wielu różnych urządzeniach istotny jest interfejs komunikacji pomiędzy nimi, a bazą danych. Aby go zrealizować stworzę aplikację dostarczającą interfejs programistyczny wykorzystujący protokół http oraz REST API. Aplikacja ta zostanie stworzona z wykorzystaniem frameworka Spring. Ponieważ w serwisie będą przechowywane dane osobowe użytkowników komunikacja w systemie musi zostać zabezpieczona w taki sposób, aby do.

* 1. Aplikacje typu SPA

Aplikacje Single Page Applicaion powstały, aby zmniejszyć objętość zapytań http wykonywanych przez przeglądarki internetowe. Aplikacja tego typu jest ładowana do pamięci przeglądarki przy odwiedzeniu strony. W praktyce odwiedzający pobiera tylko raz plik index.html, wszelkie zmiany treści oraz wyglądu strony są realizowane w wyniku działań użytkownika. Przy korzystaniu z serwisów stworzonych z wykorzystaniem technologii SPA użytkownik nie doświadcza przeładowania strony. Zmianom podlegają tylko wybrane elementy modelu DOM, np. określone znaczniki. Nowe dane pobierane są wyłącznie, gdy istnieje taka potrzeba, a ich rozmiar można maksymalnie ograniczyć przez przesyłanie tylko istotnych informacji, bez dodatkowych danych takich jak strony HTML. Wiąże się to z oszczędnościami transferu oraz pozytywnym wpływem na odbiór strony przez użytkownika, gdy ładowane są nowe elementy strony użytkownik cały czas może korzystać z tych które są już załadowane. Znanym 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 oraz CSS. Ponadto wymagają one frameworków takich jak Angular czy ReactJS.

Rozwiązania SPA mają wiele zalet, jednak posiadają również wady. Z uwagi na fakt iż, strona jest ładowana w całości wyłącznie raz podczas tego połączenia przesyłana jest duża ilość informacji – skompilowany plik JavaScript z aplikacją, pomocnicze biblioteki oraz arkusze stylów CSS. Do wad można również zaliczyć stopień skomplikowania frameworków do tworzenia tego typu stron, gdyż aplikacje tworzone za ich pomocą muszą mieć ściśle określoną architekturę, która nie zawsze jest intuicyjna. Należy również zauważyć, że witryny SPA ze względu na swój dynamiczny charakter nie zawsze są poprawnie indeksowane przez crawlery wyszukiwarek internetowych, chociaż w ostatnim czasie uległo to znacznej poprawie.

* 1. Aplikacja mobilna

Aplikacje mobilne to aplikacje, które można uruchomić na szeroko rozumianych urządzeniach mobilnych takich jak smartfony, tablety oraz smartwatche. Aplikacje te instalowane są bezpośrednio na urządzeniu oraz mają możliwość korzystania z jego zasobów np. kamery, głośników czy mikrofonu. Obecnie istnieje wiele sposobów tworzenia aplikacji mobilnych. Aplikacje te mogą być tworzone z wykorzystaniem technologii dostarczanych przez twórców systemu operacyjnego, jednak korzystanie z nich będzie możliwe jedynie na urządzeniu z dedykowanym systemem. Takie podejście wymaga napisania osobnych aplikacji na urządzenia z systemem Android oraz osobnej na urządzenia z systemem iOS. Istnieją również narzędzia pozwalające na tworzenie aplikacji z wykorzystaniem technologii znanych z rozwiązań internetowych (JavaScript, HTML, CSS), która następnie może zostać skompilowana do aplikacji dedykowanej na konkretny system. W praktyce programista pisze kod aplikacji raz, a następnie za pomocą odpowiednich komend framework buduje aplikacje na wybrane platformy. Jednym z takich rozwiązań jest Ionic, działający w oparciu o Angular oraz Apatche Cordova. Framework ten wykorzystałem do stworzenia aplikacji mobilnej na potrzeby niniejszej pracy. Główną zaletą pakietu Ionic jest ponowne wykorzystanie kodu. Posiadając działającą aplikację Ionic możemy za pomocą interfejsu dostarczonego wraz z pakietem zbudować aplikacje na urządzenia iOS, Android oraz Windows. Oczywiście rozwiązanie to nie jest idealne, aplikacje napisane w ten sposób mają ograniczony dostęp do natywnych funkcji systemu oraz działają zauważalnie wolniej od aplikacji dedykowanych na poszczególne platformy. Pomimo wad rozwiązanie to bardzo dobrze sprawdza się w przypadku w aplikacjach realizujących bardzo proste zadania takie jak dostęp do danych z zewnętrznej bazy danych.

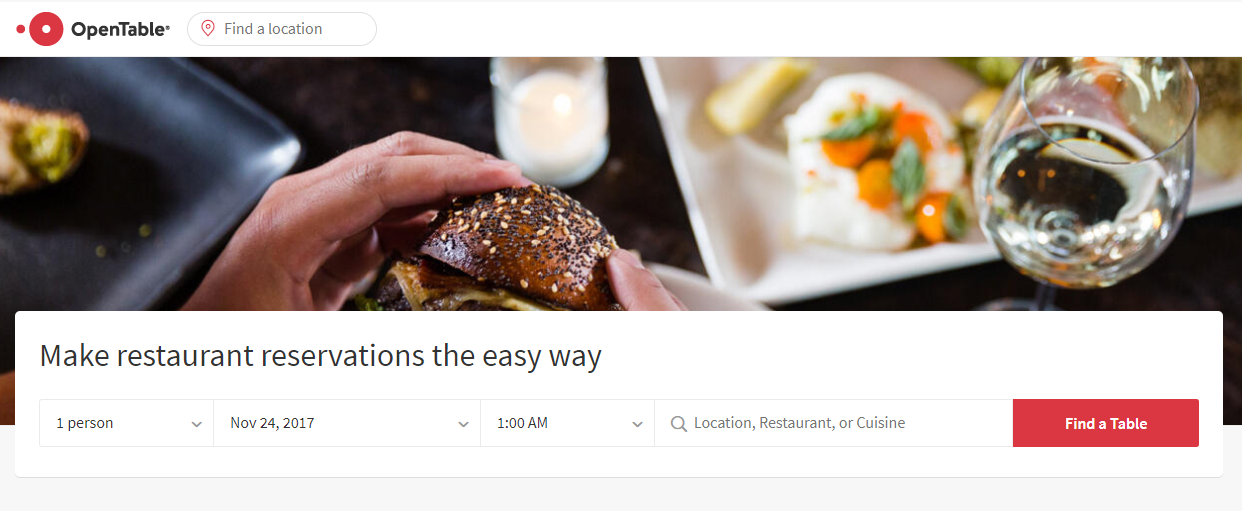
1. Istniejące rozwiązania

Na światowym rynku istnieje wiele rozwiązań pozwalających na rezerwację stolików online. Na polskim rynku jednak nie są one jeszcze powszechne.

* 1. OpenTable

Amerykański serwis OpenTable założony w 1998 roku daje możliwość rezerwacji miejsc w restauracjach na całym świecie. Według danych dostępnych w Wikipedii na dzień pisana tej pracy w serwisie znajduje się 40 tysięcy restauracji w wielu krajach świata. Niestety serwis nie działa w Polsce.

Użytkownicy mogą dokonać rezerwacji za pośrednictwem strony internetowej <http://opentable.com> lub aplikacji mobilnych. Aby zarezerwować stolik należy wypełnić formularz znajdujący się na stronie głównej serwisu.



Rysunek 1 Widok części strony głównej OpenTable

Jak widać na powyższym rysunku dane jakie należy podać to ilość osób dla jakiej ma zostać zarezerwowany stolik, datę rezerwacji, godzinę oraz lokalizację. Po wysłaniu formularza zostaje wyświetlona lista restauracji spełniających podane kryteria oraz możliwe godziny rezerwacji. Po wybraniu restauracji oraz godziny użytkownik zostaje przekierowany na stronę finalizującą proces w na której należy podać dane osobowe oraz numer telefonu kontaktowego. Rezerwacji stolika można dokonać również poprzez aplikację mobilną.

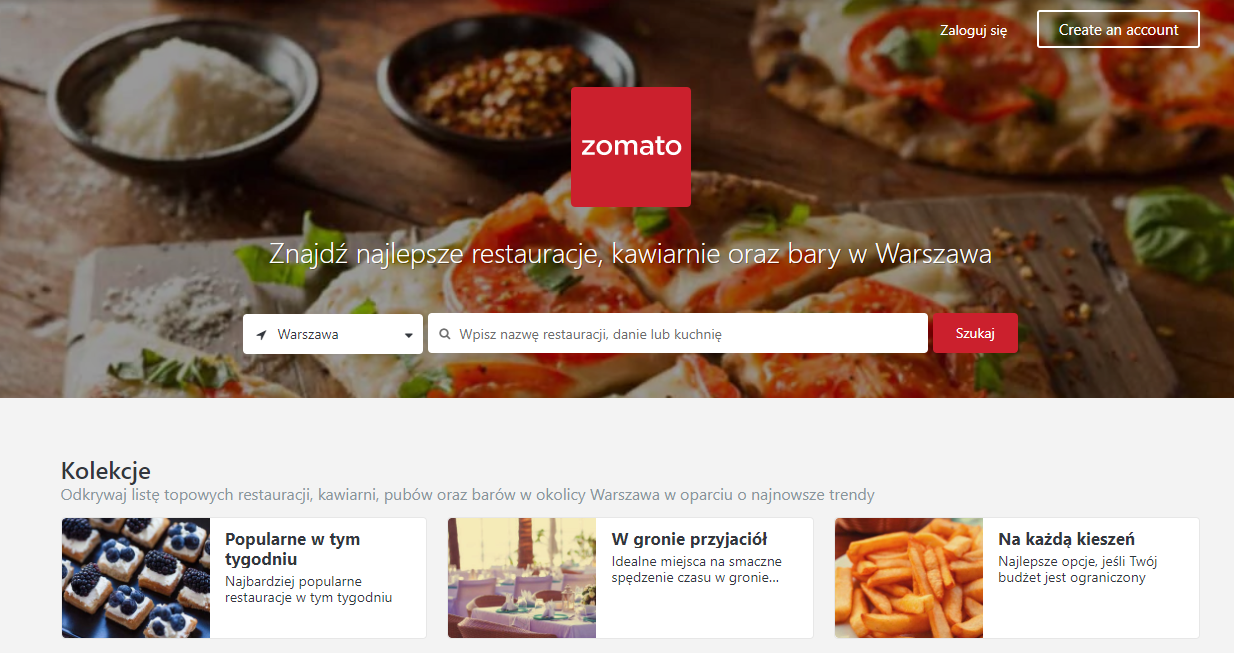
Serwis poza podstawowymi informacjami takimi jak nazwa restauracji oraz adres udostępnia użytkownikom inne dane pozwalające pomagające w wyborze restauracji. Są to na przykład aktualnie serwowane menu, galeria zdjęć prezentująca wystrój restauracji oraz przedział cenowy w jakim znajduje się restauracja.

OpenTable posiada również część społecznościową. Użytkownicy serwisu mogą umieszczać w nim opinie o odwiedzonych restauracjach oraz oceniać je przyznając punkty w postaci gwiazdek.

Restauratorzy mają możliwość zarządzania restauracją za pomocą aplikacji na urządzenia z systemem iOS dostarczanej przez OpenTable. Aplikacja ta pozwala na wprowadzenie stolików znajdujących się w restauracji oraz zarządzania nimi z jej poziomu. Możliwe jest również sprawdzanie statystyk restauracji oraz bieżących rezerwacji.

* 1. Zomato

Serwis Zomato (zomato.com) podobnie jak OpenTable jest bazą restauracji. Działa on w ponad 150 miastach na świecie. Pozwala na wyświetlenie restauracji znajdujących się w wybranej okolicy.



Rysunek 2 Widok strony głównej serwisu Zomato

Użytkownicy aplikacji mają dostęp do galerii, menu restauracji, wskazówek dojazdu oraz danych kontaktowych. Klienci mogą oceniać restauracje na podstawie skali gwiazdek oraz dodawać recenzje. 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 na urządzenia działające pod kontrolą popularnych systemów operacyjnych. Aplikacja ta dostarcza takich samych funkcjonalności jak serwis internetowy.

Patrząc na ofertę serwisów OpenTable oraz Zomato możemy dostrzec wspólne cechy tych rozwiązań. Oba serwisy udostępniają użytkownikom możliwość znajdowania restauracji w wybranym mieście, dostarczają dane kontaktowe oraz informacje o serwowanym menu. Jak wiele obecnych na rynku usług udostępniają dostarczają aplikacje mobilne pozwalające na korzystanie z serwisu za pomocą urządzeń przenośnych. Tworząc system w ramach niniejszej pracy chciałbym odwzorować wymienione powyżej cechy oraz architekturę systemu.

1. Projekt systemu
   1. Użytkownicy systemu

System opisany w tej pracy będzie dostarczał funkcjonalności dwóm grupom użytkowników. Będą to osoby indywidualne nazywane, klienci restauracji nazywane dalej klientami oraz właściciele restauracji lub osoby przez nie upoważnione nazywane dalej właścicielami. 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 dokonywał nowych rezerwacji w restauracjach oraz będzie mógł przejrzeć ich historię.

* + 1. Właściciel

Jako właściciela w serwisie rozumie się osobę, która jest przedstawicielem restauracji. Konto właściciela pozwala na dodawanie nowych restauracji do serwisu oraz zarządzanie rezerwacjami.

* 1. Wymagania funkcjonalne
     1. Identyfikacja użytkowników

Każdy z użytkowników niezależnie od roli posiadać będzie indywidualne konto. Konto zostaje utworzone przez użytkownika przez podanie danych – imię, nazwisko, numer telefonu oraz hasło. Aby zalogować się do serwisu użytkownik musi podać adres email oraz hasło.

* + 1. Wyszukiwanie restauracji

Każdy klient będzie miał możliwość wyszukania restauracji spełniających sprecyzowane przez niego kryteria.

* + 1. Dokonanie rezerwacji

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

* + 1. Statusy rezerwacji

Każda rezerwacja w serwisie w zależności od wykonanych na niej operacji powinna posiadać swój 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 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 użytkownik podaje dane takie jak nazwa, adres, opis, definiuje 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 za pośrednictwem przeglądarki internetowej.

* + 1. Obsługa rezerwacji

Właściciel restauracji powinien mieć możliwość akceptacji bądź odrzucenia rezerwacji wysłanych przez klientów. Właściciel powinien mieć możliwość wyświetlenia wszystkich rezerwacji dotyczących jego restauracji.

* 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 za pomocą aplikacji mobilnej działającej na urządzeniach z systemem Android.

* + 1. Komunikacja pomiędzy składowymi 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. Implementacja takiego sposobu komunikacji pozwoli również na rozszerzanie systemu o kolejne aplikacje np. aplikację działającą na komputerach osobistych.

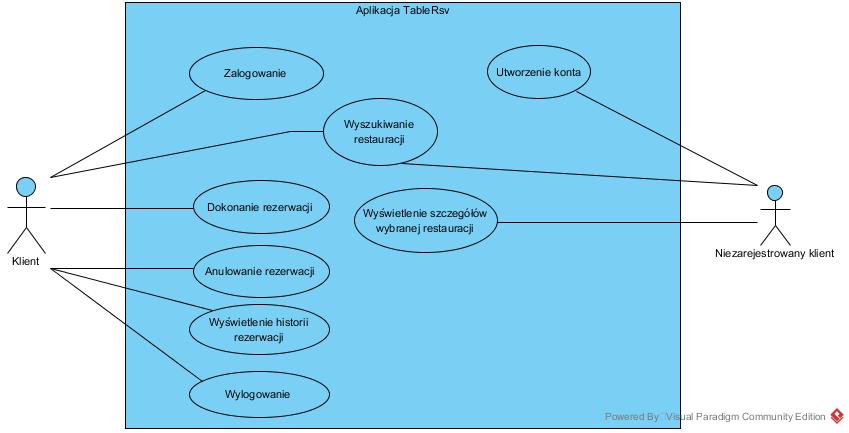
* + 1. Zabezpieczenia

Aby uzyskać dostęp do konta użytkownik musi zalogować się za pomocą adresu email oraz hasła podanego przy rejestracji, niezależnie od typu konta. Wszystkie zapytania do serwera udostępniającego REST API powinny być zabezpieczone tokenem potwierdzającym, że użytkownik ma dostęp do danych. Aby ułatwić użytkownikom korzystanie z serwisu token powinien być zapamiętany w pamięci przeglądarki.

* + 1. Wykorzystane 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 wymienionych przypadków użycia wymienionych 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 utworzenia nowego konta przez aplikację mobilną. Aplikacja mobilna daje dostęp do wymienionych funkcjonalności po uprzednim zalogowaniu się.

* + - 1. PU Utworzenie konta

Przypadek ten opisuje utworzenie nowego konta użytkownika. Funkcjonalność ta jest dostępna jedynie za pośrednictwem aplikacji WWW.

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 wygenerowanym poniżej 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  autoryzacji |
| 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 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. Wysłanie odpowiedzi z tokenem |

* + - 1. PU Wylogowanie

Wylogowanie jest możliwe wyłącznie w podczas korzystania z internetowej części serwisu, ponieważ tylko w tej wersji 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” z 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 oraz 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 na wybraną godzinę |
| 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 | Przeniesienie użytkownika do widoku szczegółów restauracji |
| Cel | Umożliwienie klientowi zapoznanie się z ofertą restauracji |
| Przebieg | 1. Klient wybiera restaurację z listy otrzymanej podczas   wyszukiwania   1. Przeniesienie użytkownika do widoku szczegółów   restauracji |

* + - 1. PU Dokonanie rezerwacji

Tabela 7 PU Dokonanie rezerwacji

|  |  |
| --- | --- |
| WS | PU Wyświetlenie szczegółów restauracji, PU Zalogowanie |
| WK | Utworzenie nowej rezerwacji dla wybranej restauracji |
| Cel | Rejestracja nowego użytkownika typu Klient |
| Przebieg | 1. Należy kliknąć przycisk „View tables” 2. Wybór jednego ze stolików z listy dostępnych   poprzez naciśnięcie przycisku „Book table”   1. Utworzenie nowej rezerwacji o statusie PENDING 2. Przekierowanie klienta do widoku historii jego   rezerwacji |

* + - 1. PU Wyświetlenie historii rezerwacji

Tabela 8 Wyświetlenie historii rezerwacji

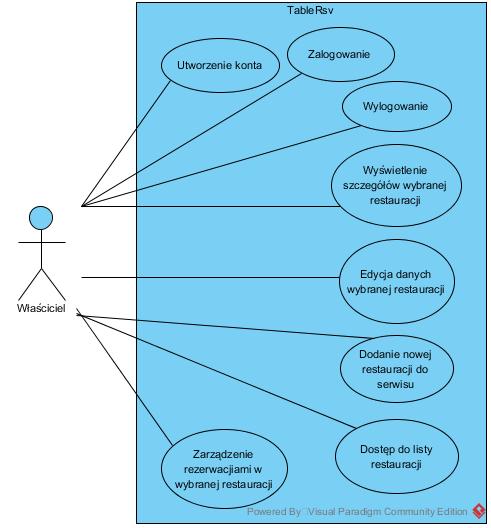
|  |  |
| --- | --- |
| WS | PU Zalogowanie |
| WK | Wyświetlenie wszystkich rezerwacji klienta |
| Cel | Wyświetlenie wszystkich rezerwacji użytkownika |
| Przebieg | 1. Wybór widoku „Reservations” 2. Pobranie rezerwacji z bazy danych 3. Przekierowanie użytkownika do widoku   pokazującego dokonane przez 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 z aplikacji internetowej. Wszystkie opisane poniżej przypadki użycia zakładają, że użytkownik korzysta z aplikacji internetowej.

* + - 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 do  autoryzacji |
| 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 4 PU Wylogowanie klienta

|  |  |
| --- | --- |
| 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 Wylogowanie klienta

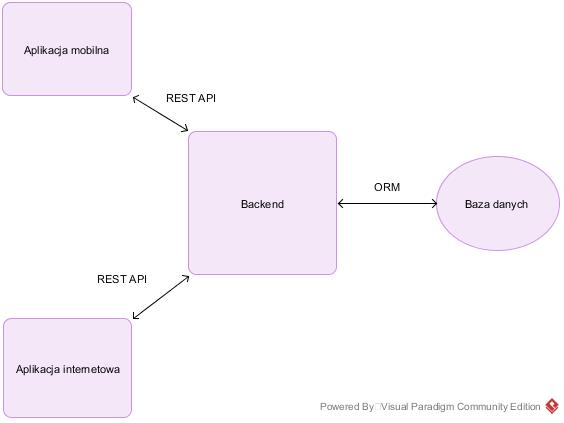
|  |  |
| --- | --- |
| WS | PU Wyświetlenie szczegółów wybranej restauracji |
| WK | Wyświetlenie listy wszystkich rezerwacji w systemie powiązanych  z wybraną restauracją |
| Cel | Zarządzanie rezerwacjami |
| Przebieg | 1. Wciśnięcie przycisku „All reservations” w widoku   szczegółów restauracji   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 4 PU Wylogowanie klienta

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

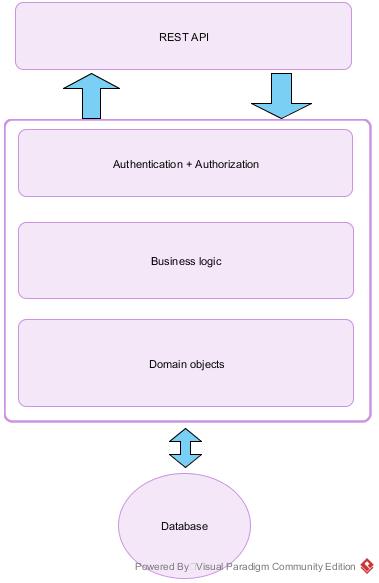


Rysunek 5 Schemat logiczny systemu

Powyższy rysunek przedstawia schemat logiczny systemu. W jego centrum jest aplikacja nazwana „Backend” jest ona warstwą dostępu do bazy danych dla aplikacji klienckich. Najważniejszymi zadaniem tej aplikacji test dostarczanie dostępu do bazy danych z którą jest połączona.

Aplikacjami klienckimi w tym systemie jest aplikacja internetowa SPA oraz aplikacja mobilna. Obie aplikacje wykorzystują do komunikacji z aplikacją Backend protokół HTTP, a dokładniej interfejs REST. W celu pobierania danych czy wysyłania danych z aplikacji Backend wysyłane są zapytania na odpowiednie adresy dostarczane przez Backend.

* + 1. Backend

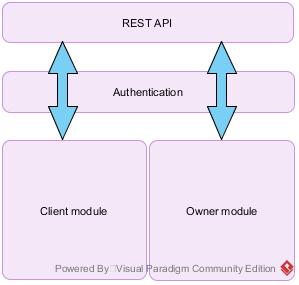


Rysunek 6 Schemat logiczny aplikacji Backend

Rysunek 6 pokazuje zarys architektury aplikacji dostarczającej REST API oraz komunikującej się z bazą danych. W tym modelu warstwa REST API jest interfejsem dla aplikacji klienckich. ??? Relizowana jest za pomocą kontrolerów http. Warstwa ta ma docelowo być dostępna w Internecie, dlatego musi zostać zabezpieczona. Tutaj główną rolę odgrywa wartstwa druga, której zadaniem jest sprawdzanie czy w nagłówkach zapytań przesyłane zostają poprawne tokeny.

Logikę biznesową aplikacji symbolizuje warstwa trzecia. Jej zadaniem będzie zarządzanie obiektami aplikacji Wykorzystanie technologii ORM wymaga utworzenia klas encji w aplikacji, zbiór tych obiektów reprezentuje warstwa Domain objects. ???

* + 1. Aplikacja internetowa SPA

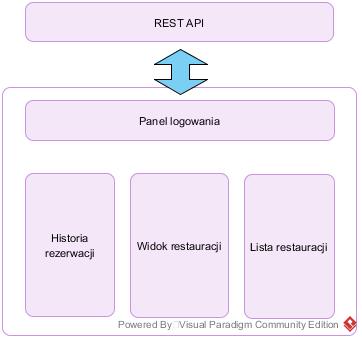


Rysunek 7 Aplikacja SPA schemat logiczny

Aplikacja internetowa do komunikacji z „Backend” będzie wykorzystywała REST API. Zabezpieczenia realizowane będą przez warstwę Authentication, w do zapytań dodawane będą odpowiednie nagłówki.

Ta część systemu będzie dostarczała funkcjonalności opisanych w przypadkach użycia dla użytkownika typu klient oraz właściciel. Na rysunku te kompetencje zostały zaznaczone jako osobne moduły. Moduł klienta będzie dostarczał funkcjonalności określonych w przypadkach użycia klienta, a właściciel odpowiednio właściciela.

* + 1. Aplikacja mobilna

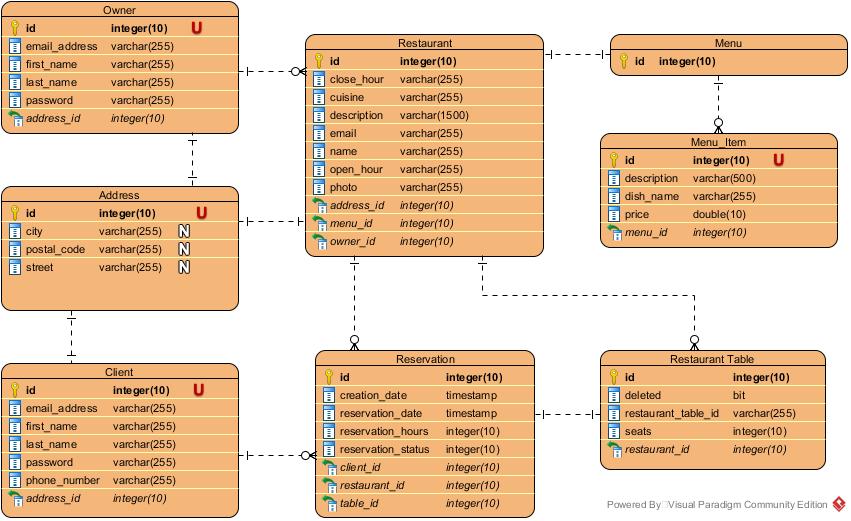


Rysunek 8 Schemat logiczny aplikacji mobilnej

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

Powyższy rysunek pokazuje moduły na jakie aplikacja zostanie podzielona. Moduły będą komunikowały się z aplikacją „Backend” za pomocą wymienionego wcześniej REST API. Moduł logowania ma odpowiadać za logowanie użytkownika. 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 najważniejsze tabele jakie znajdą się w projekcie 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 metoda JSON Web Token. Podczas wysyłania zapytań HTTP do nagłówka żądania musi zostać dołączony token, którego poprawność jest sprawdzana w aplikacji dostarczającej REST API. 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. W pakiecie znajduje się również Spring Core, którego głównym celem jest dostarczenie mechanizmu wstrzykiwania zależności (ang. Dependency Injection). Pusty projekt aplikacji można wygenerować za pomocą narzędzia Spring Initializr, który pozwala na dodanie dodatkowych modułów.

* + 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 wykorzystaniem wzorca Model View Controller jak również 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 zapewniający translację danych pomiędzy relacyjną bazą danych, a obiektami języka Java. Warto również wymienić Spring Data JPA wchodzący w skład tego pakietu, który zapewnia użytkownikom kolejną warstwę abstrakcji dostępu do bazy danych. Spring Data JPA, pozwala na tworzenie repozytoriów JPA za pomocą których możemy tworzyć zapytania do bazy w formie metod. Metody realizujące dostęp do bazy muszą mieć ściśle określoną nazwę opisaną w dokumentacji Spring Data JPA.

* + 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. Podobne efekty można uzyskać wykorzystując na przykład framework Django.

* 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 mogą być 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 serwisu, a referencja do niego zostanie automatycznie przekazana do komponentów go wymagają.
* 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.
  + 1. TypeScript

Do tworzenia aplikacji bazujących na platformie Angular zalecane jest używanie języka TypeScript stworzonego przez firmę Microsoft. TypeScript jest implementacją skryptowego języka ECMAScript, a dokładnie nadzbiorem języka JavaScript. Rozszerza on JavaScript o możliwości takie jak typowanie zmiennych, argumentów funkcji oraz funkcji. Pozwala na tworzenie klas, interfejsów oraz typów wyliczeniowych. Do uruchomienia programów napisanych w języku TypeScipt w przeglądarkach internetowych niezbędna jest kompilacja do plików „.js”. To właśnie na poziome sprawdzana jest poprawność typów.

* + 1. AngularCLI

AngularCLI jest narzędziem wspomagającym tworzenie aplikacji internetowych na platformie Angular. Dostarcza on interfejs wiersza poleceń pozwalający na tworzenie nowych projektów oraz zarządzanie istniejącymi. W skład tego pakietu wchodzi kompilator języka TypeScript, serwer deweloperski oraz narzędzia wspomagające tworzenia komponentów oraz testowanie aplikacji.

* 1. Ionic

Ionic jest zestawem narzędzi programistycznych pozwalającym na tworzenie aplikacji mobilnych z wykorzystaniem technologii JavaScript, HTML oraz CSS. Ionic wspiera dodatkowo 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 do wprowadzania teksu, hasła lub wyboru daty.

Uruchamianie aplikacji na urządzeniach mobilnych możliwe jest dzięki wykorzystaniu technologii Apatche Cordova. Jest ona podstawą działania platformy Ionic. Apatche Cordova dostarcza aplikacji opakowujących kod HTML oraz JavaScript w taki sposób, żeby użytkownik miał wrażenie korzystania z natywnej aplikacji na jego platformę. Ponadto Cordova dostarcza mechanizmów dostępu do zasobów urządzenia takich jak aparat fotograficzny, odbiornik GPS, czytnik linii papilarnych oraz wiele innych.

* 1. Bootstrap

Bootstrap jest biblioteką JavaScript oraz CSS ułatwiającą tworzenie reseponsywnych stron internetowych, czyli takich które dostosowują się do ekranu użytkownika przez odpowiednie skalowanie elementów na stronie lub zmianę ich pozycji. Ponadto Bootstrap dostarcza gotowe schematy kolorów elementów strony, tak aby była ona spójna wizualnie. Użytkownicy tej biblioteki mają możliwość wykorzystania na stronie gotowych przycisków, powiadomień czy formularzy.

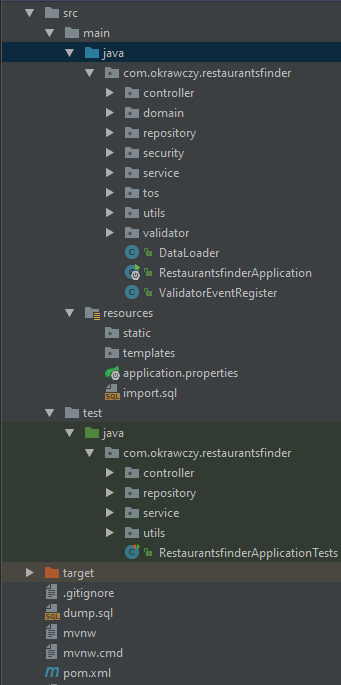
* 1. PostgreSQL

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 na przechowywanie oraz wykonywanie operacji na obiektach typu JSON. Ponadto użytkownik ma możliwość definiowania własnych typów danych.

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. Do budowania aplikacji zastosowane zostało narzędzie Maven, które poza automatycznym budowaniem aplikacji oraz uruchamianiem testów pozwala na pobieranie zależności projektu z zewnętrznych repozytoriów. Konfiguracja tego narzędzia znajduje się w pliku „pom.xml”.

Warto również zwrócić uwagę na pliki „application.properties” oraz „import.sql” znajdujące się w katalogu „resources”. Pierwszy z nich zawiera konfigurację frameworka Spring Boot. To w nim umieszczone są dane logowania do serwera bazy danych, port na jakim ma działać aplikacja oraz wiele innych. Plik „import.sql” zawiera przykładowe dane, którymi wypełniana jest baza danych podczas uruchomienia w trybie deloperskim”

* + 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 czy pobranie listy restauracji właściciela na podstawie jego adresu email.
* ReservationController – tutaj obsługiwane są zapytanie dotyczące obsługi rezerwacji. Tworzenie nowych rezerwacji, pobieranie listy rezerwacji dla danej restauracji lub użytkownika oraz zmiany statusów
* RestaurantController – obsługa restauracji, dodawanie nowych, edycja danych istniejących oraz wyszukiwanie na podstawie nazwy restauracji
* RestaurantTableController – 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 baze danych będą nazywały się tak jak klasy. Istotne są również typy pól klasy, ponieważ na ich podstawie definiowane są typy 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 tworzone relacje oraz zdefiniowane parametry kolumn. @Id wskazuje, że to pole ma być kluczem głównym, @Columnt(nullable = false) powoduje, że kolumna nie może być pusa, @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);

}

Repozytoria rozszerzają interfejs CRUDRepository lub RagingAndSortingRepository. Metoda findClientById pobierze z bazy danych obiekt użytkownika, którego id pokrywa się z id podanym jako parametr.

* + 1. Pakiet „security”

W pakiecie „security” znajdują się klasy konfigurujące działanie 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 typu klient oraz właściciel. 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 dane się pokrywają następuje generowanie tokena. W tokenie JWT zapisana jest nazwa użytkownika, data jego wygaśnięcia oraz uprawnienia jakie posiada użytkownik. Token ten na końcu jest wysyłany do użytkownika.

Działanie JWT zostało opisane w dokumecie 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 kropami. 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

}

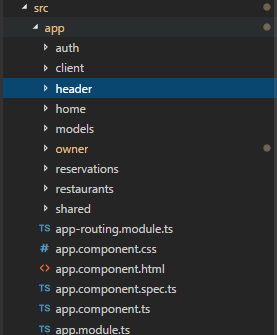
Warto zauważyć, że w tokenie nie mogą być przesyłane newralgiczne informacje, ponieważ do ich zakodowania używany jest algorytm base64, który nie zapewnia bezpieczeństwa. Istotą działania tego mechanizmu jest podpis, którego poprawność jest sprawdzana przy każdym zapytaniu przesłanym do serwera.

Dokładną specyfikację działania JWT opisuje dokument RFC-7519.

* + 1. Pozostałe pakiety

Wymienione powyżej pakiety stanowią podstawę działania aplikacji, realizują najważniejsze funkcjonalności takie jak dostarczenie REST API, dostęp do bazy danych czy zapewnienie bezpieczeństwa. Pozostałe pakiety zostaną w skrócie opisane poniżej.

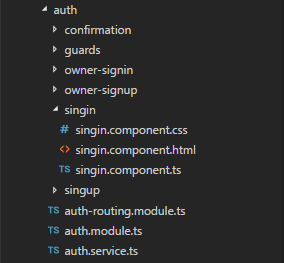
* „tos” – pakiet zawiera obiekty typu Transfer Objects. Obiekty z klasy domen są spłaszczane do obiektów TO przed wysłaniem ich do użytkownika. Ma to na celu zredukowanie ilości informacji oraz powiązań pomiędzy obiektami wysyłanymi za pomocą protokołu HTTP. Na przykład obiekt OwnerTO pozbawiony jest 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” – obejmuje klasy konwerterów obiektów 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 napisana z wykorzystaniem frameworka Angular podzielona jest na foldery oraz moduły. Znajdują się w nich komponenty, szablony HTML, arkusze stylów oraz w miarę potrzeby serwisy udostępniające dane dla komponentów. Wszystkie moduły oraz komponenty dołączane są do aplikacji w pliku app.module.ts, który jest korzeniem aplikacji. Zewnętrze moduly takie jak NgDatePicker zawierający kalendarz również muszą zostać zaimportowane w pliku app.modue.ts. Poniżej zostaną opisane poszczególne moduły oraz komponenty.

* + 1. Moduł „auth”



Rysunek 12 Moduł "auth"

Wszystkie komponenty serwisy oraz formularze związane z logowaniem oraz rejestracją użytkowników są zawarte a module „auth”.

Klasa „auth.service.ts” zapewnia serwis dostępny w całej aplikacji, którego zadaniem jest dostarczanie informacji o aktualnie zalogowanym użytkowniku, jego uprawnieniach oraz generowanie nagłówków HTTP z zapamiętanym tokenem autoryzacji.

W aplikacjach typu SPA przechodzenie pomiędzy stronami realizowane jest przez wbudowany mechanizm przekierowań. Użytkownik klikając łącze wywołuje kod JavaScript zmieniający aktualnie wyświetlany komponent. Powiązania pomiędzy łączami, a komponentami zawarte są najczęściej w plikach routingu. W tym przypadku jest to plik „auth-routing.module.ts”. Tablica przekierowań module tym jest definiowana następująco:

Listing. 5. Tablica przekierowań modułu auth

const authRoutes: Routes = [

{path: 'signup', component: SingupComponent},

{path: 'signin', component: SinginComponent},

{path: 'owner\_signup', component: OwnerSignupComponent},

{path: 'owner\_signin', component: OwnerSigninComponent},

{path: 'confirmation', component: ConfirmationComponent}

];

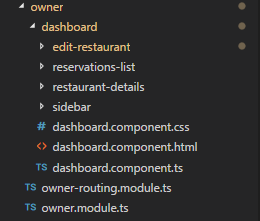
Komponent, w którym zawarty jest formularz logowania dostępny jest po przejściu pod adres „/signiup”.

Na rysunku pokazującym moduł auth widać również strukturę komponentu. Zawiera on arkusz stylów, szablon HTML oraz plik, w którym zdefiniowane jest zachowanie komponentu oraz odpowiadające mu metody. W plikach .html zdefiniowana jest struktura widoku komponetu oraz jego wygląd. W pokazanym na rysunku pliku „signin.component.html” znajduje się definicja formularza logowania. Plik z rozszerzeniem „.ts” zawiera metody opisujące zachowanie komponentu np. gdy użytkownik kliknie przycisk „Submit”. Pliki „.ts” komponentów mogą również przechowywać dane, które widoczne będą w szablonach. W przypadku komponentu logowania dane pobrane od użytkownika czyli adres email oraz hasło pobierane są z szablonu, a następnie wysyłane przez do aplikacji dostarczającej API przez metodę signInClient serwisu „auth.service.ts”.

Poza logowaniem klientów oraz właścicieli moduł „auth” definiuje formularze rejestracji nowych użytkowników oraz jest odpowiedzialny za odczytywanie tokenów zapisanych w pamięci przeglądarki.

W folderze „guards” znajduje się implementacja serwisu przyznająca dostęp do poszczególnych adresów zdefiniowanych w aplikacji. Dzięki serwisowi „guard” tylko zalogowani klienci mają dostęp do listy rezerwacji, mogą dokonywać rezerwacji i nie mają dostępu do panelu właściciela.

* + 1. Moduł „owner”



Rysunek 13 Moduł owner

Moduł „owner” zawiera wszystkie komponenty tworzące interfejs właściciela. Zawiera on definicję powiązań adresów z komponentami w tym module oraz komponent „dashboard”, który jest panelem zarządzania restauracjami. Na „dashboard” składają się formularz edycji oraz dodawania nowej restauracji, widok szczegółów restauracji, widok listy rezerwacji oraz pasek boczny z listą restauracji właściciela.

Listing. 6. Tablica przekierowań modułu auth

const authRoutes: Routes = [

{path: 'dashboard', component: DashboardComponent, canActivate: [OwnerGuard], children:

[{path: 'edit/:id', component: EditRestaurantComponent},

{path: 'edit', component: EditRestaurantComponent},

{path: 'details/:id', component: RestaurantDetailsComponent},

{path: 'reservations/:id', component: ReservationsListComponent}]}

];

Zmienne id w definicji ścieżek odnoszą się do id restauracji. Są one odczytywane wewnątrz komponentów, a na ich podstawie pobierane tworzone jest zapytanie REST typu GET do aplikacji Backend, które zwraca szczegóły restauracji w formacie JSON.

Listing. 7. Pobranie id z adresu w restaurant-details.component.ts

ngOnInit() {

this.activatedRoute.params.subscribe(

(params: Params) => {

this.restaurantId = params['id'];

this.getRestaurantData();

});}

Listing. 8. Wysłanie zapytania o szczegóły restauracji

getRestaurantData() {

this.restaurantService.getRestaurantInfo(

Number(this.restaurantId)).subscribe(

(data) => this.restaurant = data.json(),

(error) => console.log(error)

);}

Dane o restauracji odebrane z aplikacji dostarczającej REST API zapisywane są w zmiennej restaurant. Informacje zawarte w tej zmiennej odczytywane są w szablonie HTML.

Listing. 9. Fragment szablonu HTML restaurant-details.component.html

<div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">

<h2>{{ restaurant.name }}</h2>

<p>

<span class="label label-default">{{ restaurant.cuisine }}</span>

<br> {{ restaurant.openHour }} - {{ restaurant.closeHour }}

<br> {{ restaurant.address.street }}

<br> {{ restaurant.address.postalCode }} {{ restaurant.address.city}}

<br> {{ restaurant.phone }}

<br> {{ restaurant.email }}

</p>

<p>{{ restaurant.description }}</p>

</div>

Na powyższym listingu widać działanie mechanizmu wiązania danych pomiędzy szablonem, a kontrolerem. W trakcie działania aplikacji, gdy szablon będzie renderowany w miejsce {{ restaurant.name }} zostanie wstawiona nazwa restauracji.

* + 1. Foldery header oraz home

Pakiety te zawierają kolejno implementację komponentu wyświetlającego menu na górze strony oraz strony głównej z formularzem wyszukiwania restauracji.

* + 1. Folder restaurants

Folder ten zawiera komponent wyświetlający szczegóły restauracji dostępne dla użytkownika. Dostęp do niego jest możliwy przez listę restauracji wyświetlaną w wynikach 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 wyświetla menu dostępne w restauracji oraz formularz wyszukiwana stolików. Po wypełnieniu formularza użytkownik ma możliwość rezerwacji wybranego stolika.

* + 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. Serwis ReservationService udostępnia metody pozwalające na składanie nowych rezerwacji, pobieranie rezerwacji użytkownika lub restauracj oraz zmianę statusu rezerwacji.

Listing. 11. Metoda makeReservation serwisu ReservationService

makeReservation(date: Date, restauranId: number, tableId: number) {

const user = this.authService.getUser();

return this.http.post(baseURL + '/reservations/makeReservation',

{reservationDateISO: date.toISOString(),

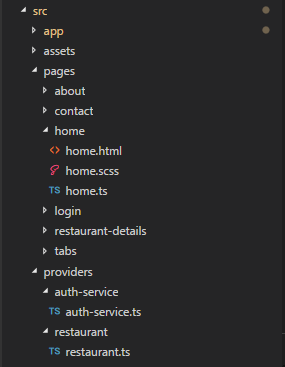
restaurantId: restauranId,

tableId: tableId, clientEmailAddress: user.emailAddress},

{ headers: this.authService.getHeaders() }}

Metoda pokazana na Listinug 11 pokazuje sposób w jaki wysyłane są zapytania do serwera Backend. Jako parametry tej metody przekazywane są id restauracji, id stolika oraz data rezerwacji. Dane użytkownika, który dokonuje rezerwacji pobierane są z AuthService. Następnie kierowane jest zapytanie na adres „/reservations/makeReservation” w ciele zapytania przesyłany jest obiekt JOSN przechowujący pobrane w parametrach informacje. Do zapytania dodawany jest również nagłówek z tokenem JWT potrzebnym do autoryzacji użytkownika w aplikacji Backend.

* 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 jest zbliżona do opisanej wcześniej aplikację mobilną. W folderze „pages” znajdują się strony definiujące ekrany, dostępne dla użytkownika. Podobnie jak w aplikacjach internetowych składają się one z komponentów. Zastosowanie w aplikacji mobilnej znajdują również serwisy przedstawione wcześniej. Znajdują się one w folderze „providers”, a ich zadaniem jest dostarczenie informacji o zalogowanym użytkowniku oraz restauracjach.

* + 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 logowanai. Dane pobrane od 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 komponencie znajdują znajduje się implementacja strony pozwalającej użytkownikowi na wyświetlenie listy jego rezerwacji oraz zarządzanie nimi. Lista rezerwacji pobierana jest za pośrednictwem serwisu RestaurantProvider, który wysyła zapytanie REST do serwera o listę rezerwacji, a odpowiedź zwraca do strony about.

* + 1. Strona home

Strona home umożliwia użytkownikowi wyszukiwanie restauracji na postawie wypełnionego formularza. Podobnie jak w przypadku strony about, dane pobrane w formularzu przekazywane są jako parametr metody RestauratProvier, która wysyła zapytanie o listę restauracji spełniających podane kryteria. Odpowiedź w postaci listy jest również wyświetlana na stronie home.

* + 1. Strona restaurat details

Użytkownik może przejść do widoku tej strony z home. Restaurant details wyświetla szczegóły wybranej restauracji, takie jak menu, opis czy adres. Na tej stronie można również dostępny stolik.

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 tworząca nową rezerwację

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 pobiera jako argument stolik oraz wysyła zapytanie do serwera, tworzące nową rezerwację w serwisie. Po dokonaniu rezerwacji użytkownik przenoszony jest na widok pokazujący jego wszystkie rezerwacje.

* + - 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