

# Koncepcja wykonania systemu

## Aplikacja myRestaurant

### 1. Przypadki użycia

#### a. Kelner chce obsłużyć zamówienie z danego stolika

- i. Zaznacza w aplikacji że to on zajmuje się tym stolikiem
- ii. Przyjmuje zamówienie i podchodzi do urządzenia z aplikacją
- iii. Zaznacza swój stolik
- iv. Wyskakuje okienko w którym dodaje zamówienie
- v. Zatwierdza, koszt zamówienia dolicza się do paragonu

#### b. Manager chciałby mieć zgromadzone informacje na temat pracowników

- i. Manager włącza aplikację z uprawnieniami kierownika
- ii. Znajduje w bazie danych odpowiedniego pracownika
- iii. Otrzymuje odpowiednie informacje, godziny pracy i tym podobne

#### c. Klient chciałby zamówić stolik w swojej ulubionej restauracji

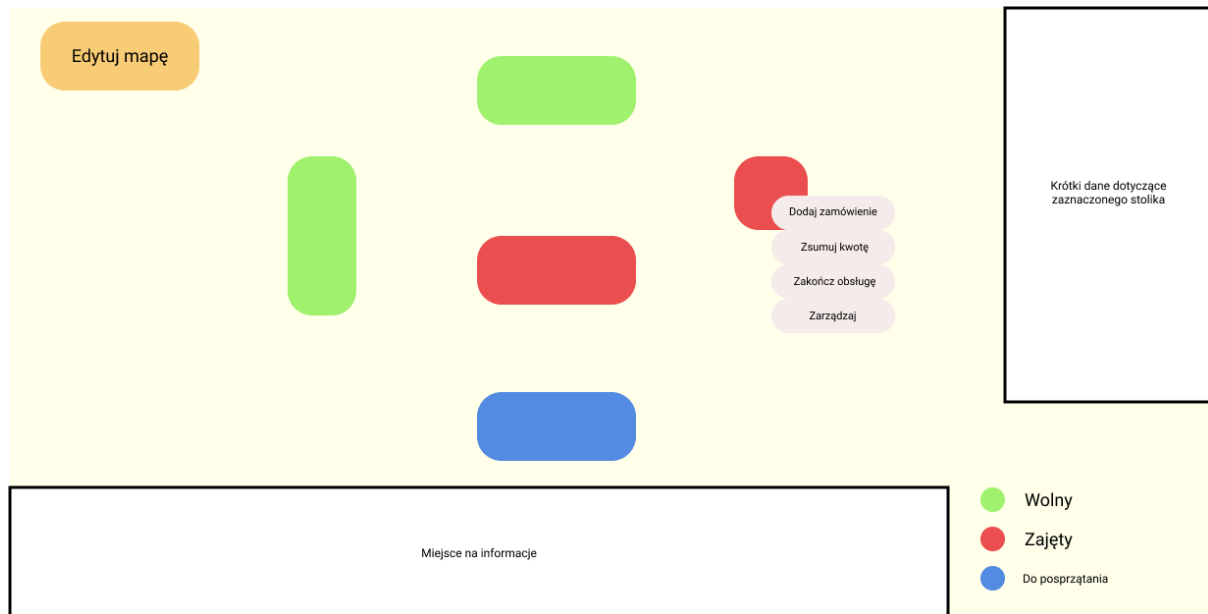
- i. Otwiera stronę restauracji
- ii. Wchodzi w ekran rezerwacji stolika, wyświetla mu się mapa lokalu
- iii. Zaznacza odpowiedni stolik
- iv. Wyskakuje okno modalne, w którym zaznacza datę, godzinę.
- v. Potwierdza to swoim imieniem, nazwiskiem, adresem mailowym i numerem telefonu.
- vi. Rezerwacja wysyłana do systemu

#### d. Kelner/Manager chce edytować mapę lokalu

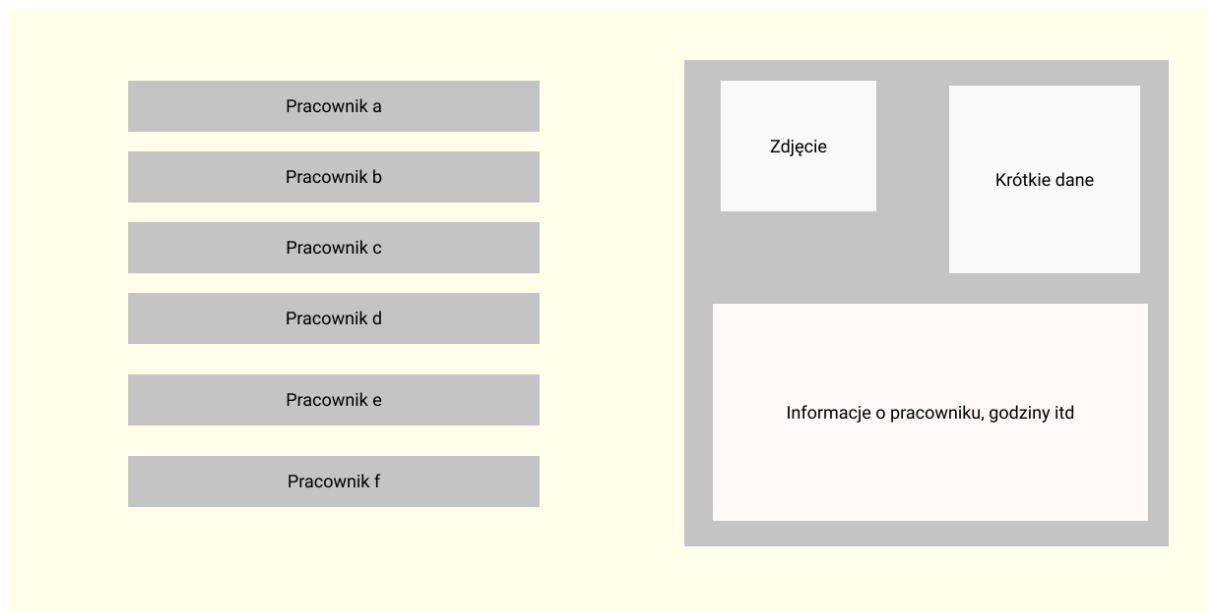
- i. Na ekranie głównym wybiera opcję "edytuj"
- ii. Mapa przechodzi w tryb edycji, stolikami można przesuwąć
- iii. Po odpowiednim ułożeniu użytkownik klika przycisk "Zapisz"
- iv. Od teraz mapa zawiera nowe ustawienie

## 2. Koncept interfejsu

### Obsługa mapy przez pracownika

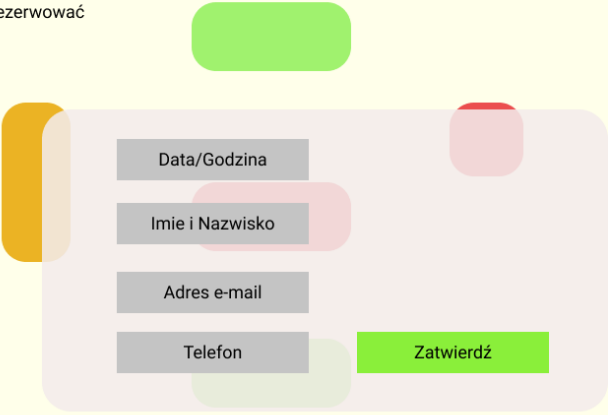


### Panel Managera



## Panel rezerwacji stolika

Wybierz stół który chcesz zarezerwować



Data/Godzina

Imię i Nazwisko

Adres e-mail

Telefon

Zatwierdź

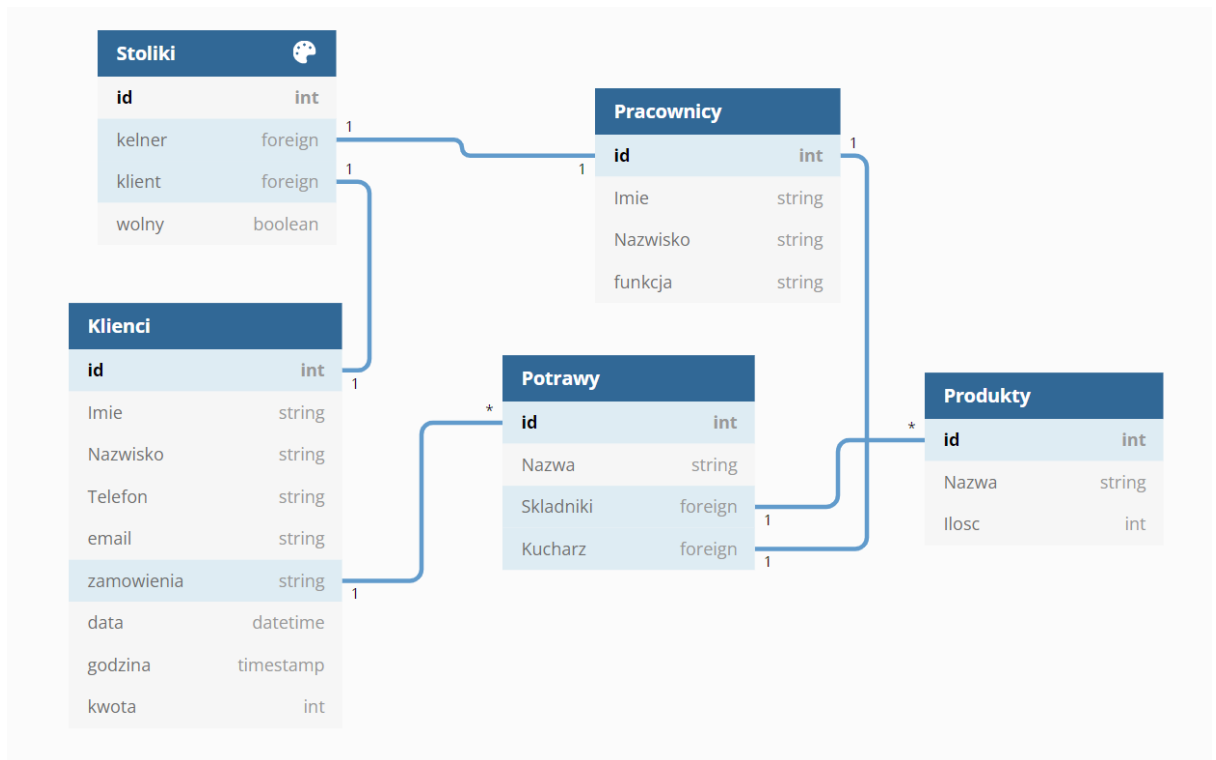
● Wolny  
● Zajęty  
● Wybrany

## Spis zasobów

składnik	ilość	składnik	ilość	składnik	ilość
składnik	ilość	składnik	ilość	składnik	ilość
składnik	ilość	składnik	ilość	składnik	ilość
składnik	ilość	składnik	ilość	składnik	ilość
składnik	ilość	składnik	ilość	składnik	ilość

Stawiamy w naszym interfejsie na prostotę i przejrzystość. Z aplikacji tej będą korzystać osoby o różnej biegłości w obsługiwaniu urządzeń, tak więc musimy zapewnić płynną obsługę dla jak najszerszego grona użytkowników.

### 3. Diagram bazy danych



### 4. Projekt architektury systemu:

- Frontend: TypeScript v4.5 & Angular.js 12.0
- Backend: Java Spring Boot v2.6, warstwa bazodanowa Spring JPA/Hibernate, protokoły bezpieczeństwa za pomocą Spring Security
- Baza danych: MySQL
- Repozytorium projektu na GitHub
- Testowanie za pomocą SpringBootTest, DataJpaTest i WebMVCTest
- Serwis na systemie Linux
- Chmura AWS

## 5. Główne zasady kodowania

- a. Frontend
  - i. Strona frontendowa będzie napisana w Visual Studio Code.
  - ii. Projekty interfejsów wykonane w programie Figma
- b. Backend
  - i. Warstwa serwerowa zostanie napisana przy użyciu narzędzia IntelliJ IDEA
  - ii. Trzymamy się standardów programowania w Javie
- c. Git
  - i. Poszczególne funkcjonalności wykonujemy w innych gałęziach niż master w formacie `/feature/{ID_FEATURE'A}/{NAZWA_FUNKCJONALNOŚCI}`
  - ii. Zmiany dodane na poszczególnych gałęziach będą podlegały code review innych programistów projektu za pomocą strony Github.com
  - iii. Używamy "git rebase" przed scalaniem gałęzi do master'a, aby utrzymać liniową historię commit'ów co zapewni przejrzystość gałęzi master
  - iv. "Squashujemy" commity na innych gałęziach
- d. SQL
  - i. Zapytania muszą być optymalne i proste w zrozumieniu
- e. Testowanie
  - i. testy jednostkowe będą pisane w JUNIT oraz Mockito, natomiast testy E2E w Selenium
- f. Instancje naszego serwera będziemy przechowywać w chmurze AWS.

## 6. Identyfikacja zagrożeń i zasady zarządzania rynkiem

- a. Awaryjność serwera
- b. Niedokładnie przetestowany kod, co może skutkować wieloma błędami oraz podatnościami m.in. możliwością kradzieży danych użytkowników jak i kodu aplikacji

Planujemy aby nasi testerzy przeprowadzili testy penetracyjne, co zidentyfikuje jakiegokolwiek podatności w naszym kodzie.

Nasz serwer będzie w chmurze AWS uruchomiony na trzech instancjach w trzech innych regionach, jeśli którakolwiek z nich padnie mechanizm auto-scaling-group na jej miejsce postawi nową. Szansa, że serwery ulegną awarii w trzech różnych miejscach na świecie jest bardzo znikoma, dzięki temu narzędziu.

## 7. Zmiany w porównaniu do tablicy koncepcyjnej

Wszystkie przedstawione wyżej są zgodne z naszą tablicą koncepcyjną, niektóre idee zostały bardziej rozwinięte (np. stack technologiczny warstwy serwerowej, serwer w chmurze AWS). Nic nie zostało pominięte lub dodane względem pierwotnego pomysłu.