Wymagania funkcjonalne:

1. **Rezerwacja stolika:**
   - Użytkownik może dokonać rezerwacji stolika na określoną datę, godzinę i liczbę osób.
   - System powinien przypisywać dostępny stolik zgodnie z preferencjami użytkownika.
2. **Potwierdzenie rezerwacji:**
   - Po złożeniu rezerwacji użytkownik powinien otrzymać potwierdzenie na stronie oraz drogą e-mailową.
3. **Edycja rezerwacji:**
   - Użytkownik może edytować datę, godzinę i liczbę osób rezerwacji przed datą rezerwacji.
4. **Anulowanie rezerwacji:**
   - Użytkownik może anulować rezerwację przed datą rezerwacji.
5. **Powiadomienia:**
   - Administrator systemu otrzymuje powiadomienia o nowych rezerwacjach oraz ich edycji i anulacjach.
6. **Historia rezerwacji:**
   - Użytkownik może przeglądać historię swoich rezerwacji.

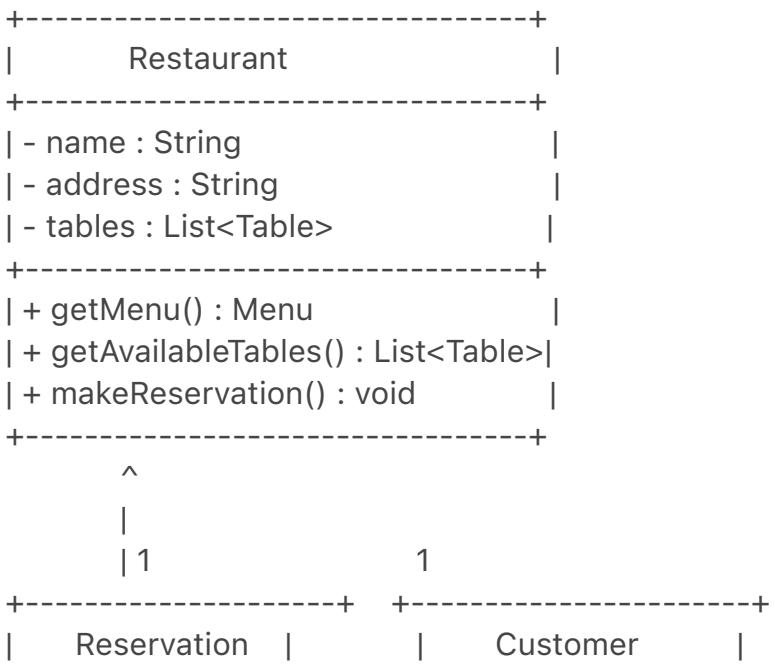
Wymagania niefunkcjonalne:

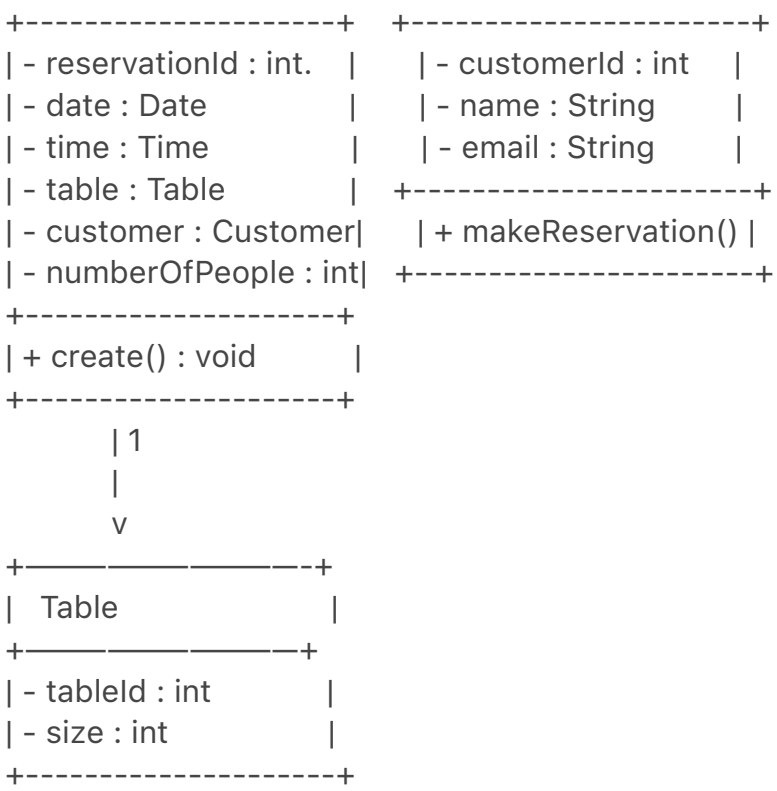
1. **Wydajność:**
   - System powinien być wydajny i responsywny, aby obsłużyć wielu użytkowników jednocześnie, minimalizując czas odpowiedzi.
2. **Bezpieczeństwo:**
   - Wszelkie dane użytkowników, w tym dane osobowe i informacje o płatnościach, powinny być przechowywane i przetwarzane zgodnie z obowiązującymi przepisami dotyczącymi prywatności.
3. **Dostępność:**
   - System powinien być dostępny przez całą dobę, 7 dni w tygodniu, aby użytkownicy mogli dokonywać rezerwacji w dogodnym dla siebie czasie.
4. **Łatwość obsługi:**
   - Interfejs użytkownika powinien być intuicyjny i łatwy w obsłudze, aby użytkownicy mogli szybko i łatwo dokonywać rezerwacji bez zbędnych trudności.
5. **Monitorowanie i raportowanie:**
   - System powinien umożliwiać monitorowanie aktywności użytkowników oraz generowanie raportów dotyczących liczby rezerwacji, popularności danego terminu itp.
6. **Skalowalność:**
   - System powinien być skalowalny, aby móc obsłużyć wzrost liczby użytkowników i rezerwacji w miarę rozwoju biznesu restauracji.

7. **Integralność danych:**
    - Wszystkie operacje na danych, takie jak dodawanie, edycja i usuwanie rezerwacji, powinny być przeprowadzane w sposób bezpieczny, aby zapewnić integralność danych i uniknąć utraty informacji.
8. **Kompatybilność przeglądarek:**
    - System powinien być kompatybilny z różnymi przeglądarkami internetowymi, zapewniając spójne działanie na różnych platformach i urządzeniach.

Opis aktorów:

1. **Klient**
    - **Opis**: Klient jest głównym użytkownikiem systemu rezerwacji stolików. Może przeglądać menu, wybierać stoliki, dokonywać rezerwacji, anulować rezerwacje i przeglądać listę zarezerwowanych stolików.
2. **System**
    - **Opis**: System reprezentuje wszelkie automatyczne procesy, które mogą być zautomatyzowane w systemie rezerwacji stolików. Może to obejmować wysyłanie powiadomień, generowanie raportów, zarządzanie rezerwacjami itp.
3. **Administrator**
    - **Opis**: Administrator jest odpowiedzialny za zarządzanie systemem rezerwacji stolików. Może dodawać, edytować i usuwać stoliki, zarządzać rezerwacjami, przeglądać raporty i monitorować działanie systemu.





Opis klas:

1. **Restaurant**
   - **Opis**: Klasa reprezentująca restaurację. Zawiera informacje takie jak nazwa, adres i lista dostępnych stolików.
   - **Metody**:
     - **getMenu()**: Zwraca menu restauracji.
     - **getAvailableTables()**: Zwraca listę dostępnych stolików.
     - **makeReservation()**: Pozwala na dokonanie rezerwacji stolika.
2. **Reservation**
   - **Opis**: Klasa reprezentująca pojedynczą rezerwację stolika. Zawiera informacje takie jak identyfikator rezerwacji, data, godzina, stolik i klient.
   - **Metody**:
     - **create()**: Tworzy nową rezerwację.
3. **Customer**
   - **Opis**: Klasa reprezentująca klienta. Zawiera informacje takie jak identyfikator klienta, imię, adres e-mail.
   - **Metody**:
     - **makeReservation()**: Pozwala klientowi dokonać rezerwacji.
4. **Table**
   - **Opis**: Klasa reprezentująca pojedynczy stolik w restauracji. Zawiera informacje takie jak identyfikator stolika, liczba miejsc oraz flagę informującą o dostępności.

Relacje:

- **Restaurant** może mieć wiele **Reservation** (1 do wielu) poprzez metody **getAvailableTables()** i **makeReservation()**.
- **Reservation** należy do jednego **Restaurant** (1 do 1).
- **Reservation** jest przypisana do jednego **Customer** (1 do 1).
- **Customer** może mieć wiele **Reservation** (1 do wielu) poprzez metodę **makeReservation()**.
- **Reservation** jest przypisana do jednego **Table** (1 do 1).
- **Table** może być przypisany do jednego **Reservation** (1 do 1).