# Spis Tresci

## 1. Dziedzina problemowa.

Ten projekt może być stosowany w systemach gastronomicznych, restauracjach typu fast food i innych zautomatyzowanych restauracjach.

## 2. Cel.

Postanowiłem skorzystać z tego systemu, ponieważ tak wiele osób codziennie odwiedza fast foody, a czasami restauracje po prostu nie są w stanie poradzić sobie z ogromnym napływem klientów. Personelowi konserwacji będzie łatwiej koordynować i zarządzać znacznie szybciej.

## 3. Zakres odpowiedzialności systemu.

- System powinien usuwać stare zamówienia co miesiąc.

- Liczyc wynagrodzenie dla każdego pracownika.

- Rejstrowac nowych klientow

- Twórzyć zamówienia online

- Ustawic typ platnosci

Pracując nad implementacja dodam jeszcze kilka metod.

## 4. Użytkownicy systemu.

- Klient

- Kasjer

- Manager

- Dostawca

- Pakowacz

## 5. Wymagania użytkownika.

W systemie powinny byc przechowywane dane klientow, ktore są zarejestrowane na stronie. Dla osob zarejstrowanych trzeba pamietac e-mail (dla online zamowien), adres I opcjonalne login.

W systeme powinny byc przechowany dane pracownikow: dane osobowe, opcjonalne numer telefonu, pesel, data zatrudnienia, ustalona stawka za godzinu pracy I minimalna stawka za godzinu pracy.

Pracowniki to manager, kasjer, pakowacz I dostawca. Dla managera naleza pamietac staz, dla kasjera rang od 0 do 5, dla pakowacza naleza pamietac predkosc pakowania I minimalnu przedkosc pakowania. Kasjer jednoczesnie moze byc pakowaczem. Dla dostawca naleza pamietac firme. Kasjer moze byc pakowaczem.

Raz na miesiac trzeba placic wynagrodzenie dla pracownikow. Wynagrodzenie managerow zaleza, od stazu pracy, kasjerow od rangu, pakowaczow od przedkosci pakowania. Wynagrodzenie dostawcy zaleze, od firmy w ktorej pracuje.

Klient moze skladac zamowienie. Dla zamowien na miesce naleza pamietac nr.zamowienia, date, status("zamowione", "oplacone", "odstapienie"), cena, nr. tabeli, ilosc wolnych miesc I max. ilosc miejsc. Jeden kasjer moze obslugowac jednoczesnej tylko jedno zamowienie.

W trakcie realizacji zamówienia klient może mieć problemy. Dla problemu musisz pamiętać typ problemu, datę zgloszenia i numer zamówienia. Menedżer powinien pomóc klientowi rozwiązać problem.

Klient musze oplacic zamowienie. Dla klasy platnosc naleza przychowac unikatowy nr.platnosci, date I typ platnosci ( “Gotowka” czy “karta”). Dla kazdego zamowienia musi byc jedna platnosc.

System oferuje dostawu dla online zamowien. Dla online zamowienia naleza pamietac godzinu dostawy oraz ilosc klientow. Dla dostawy naleza pamietac nr.dostawy, date dostawu(termin) I status dostawy ("w trakcie przygotowywania", "w drogie", "dostawione"). Jeden dostawca moze dostarczyc wiele dostaw.

**Funkcjonalność systemu:**

- System powinen liczyc wynagrodzenie dla wszystkich pracownikow. (Manager)

- System powinie pokazywac rang wszystkich kasjerow. (Manager)

- System musze przypisywac zamowienia do kasjerow. (Kasjer)

- System powinien rejstrowac nowych klientow dla online zamowien. (Klient)

- Rejstracja nowych zamowienia, wyberanie typu platnosci I oplata zamowienia. (Klient)

- Usuwanie starych zamowien co miesiac. (Automatyczne na paczatku miesiaca)

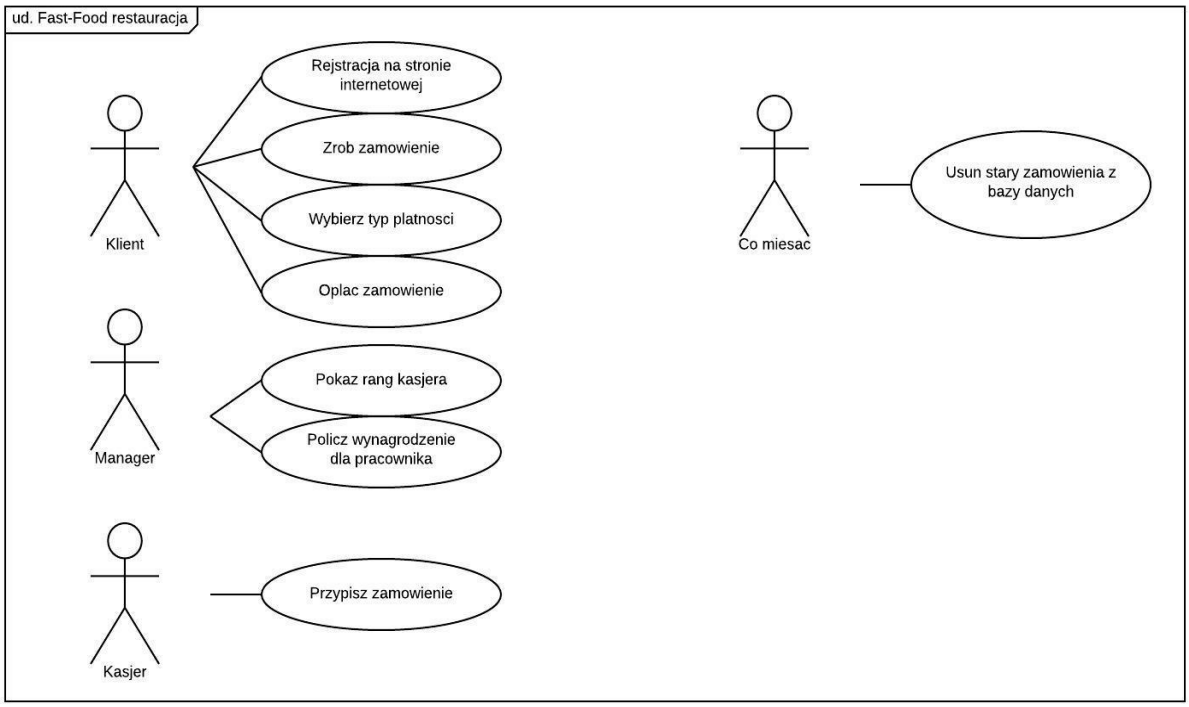
**Ograniczenia:**

- Stawka za godzinu pracy nie moze byc mniej niz minimalna stawka za godzinu pracy.

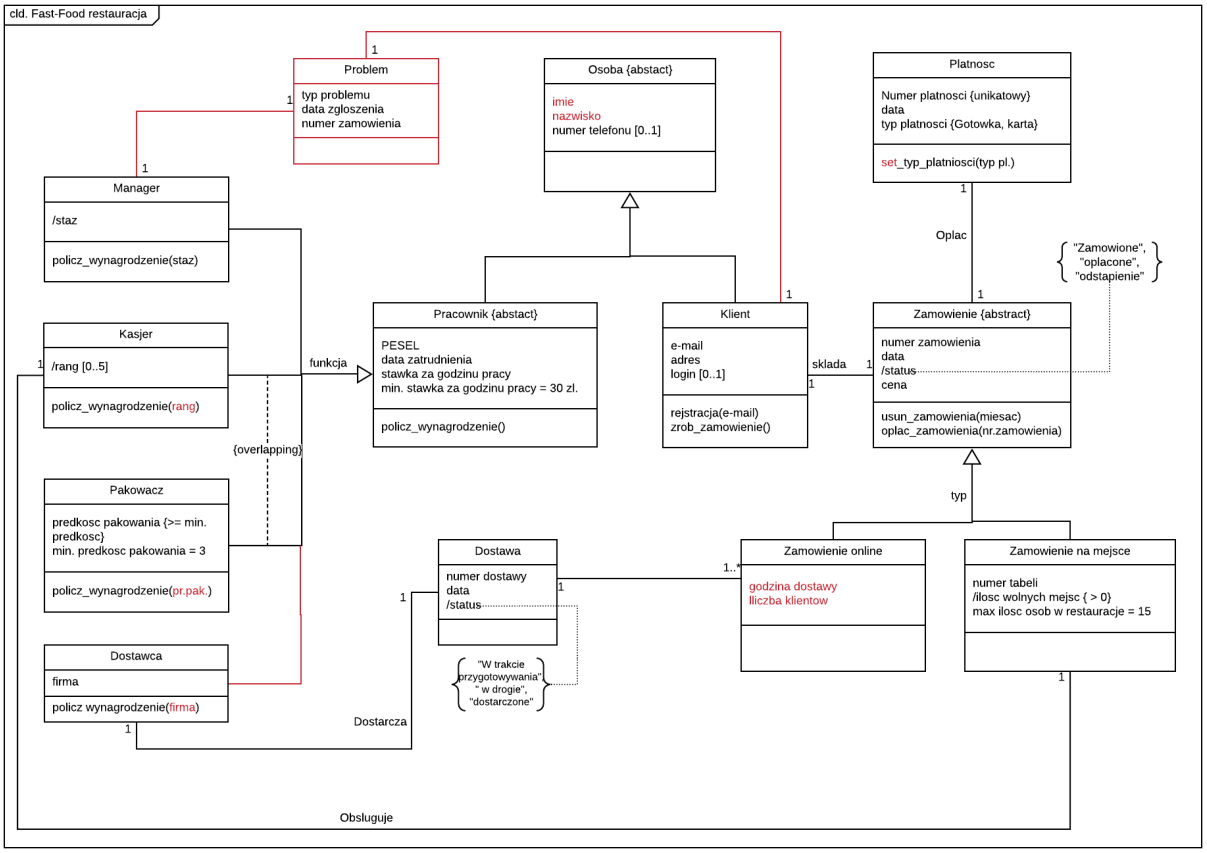
- Przedkosc pakowania nie moze byc mniej niz minimalna przedkosc pakowania.

- Liczba osób w restauracji nie może przekraczać 15 osob

## 6. Wymagania funkcjonalne.




## 7. Opis struktury systemu (schemat pojęciowy).



## 8. Przypadki użycia.

**Scenariusz dla przypadku uzycia "Tworzenie zamówień na miejscu”:**

**Stan początkowy**: Klient wchodzi do restauracji.

**Główny przeplyw zdarzen:**

1. Aktor klient uruchamia przypadek użycia.

2. Aktor kasjer pyta o rodzaje zamowienia. Aktor wybiera na miejscu.

3. Aktor klient zaczyna wybierać pozycje z menu.

4. Aktor kasjer sprawdza dostępność towaru. Jeśli towar jest dostępny, przekazuje informację do pakującego.

5. Aktor klient płaci za zamówienie.

6. Aktor pakujący przygotowuje zamówienie. Aktor wydaje zamówienie klientowi.

7. Aktor klient otrzymuje zamówienie i siada przy stole.

**Alternatywny przeplyw zdarzen:**

2a. Aktor klient wybiera dostawę online.

2aa. Aktor klient płaci za zamówienie

4a. Brak wyrobów gotowych. Wydłużenie czasu oczekiwania na zamówienie.

5a. Klient odmawia zapłaty.

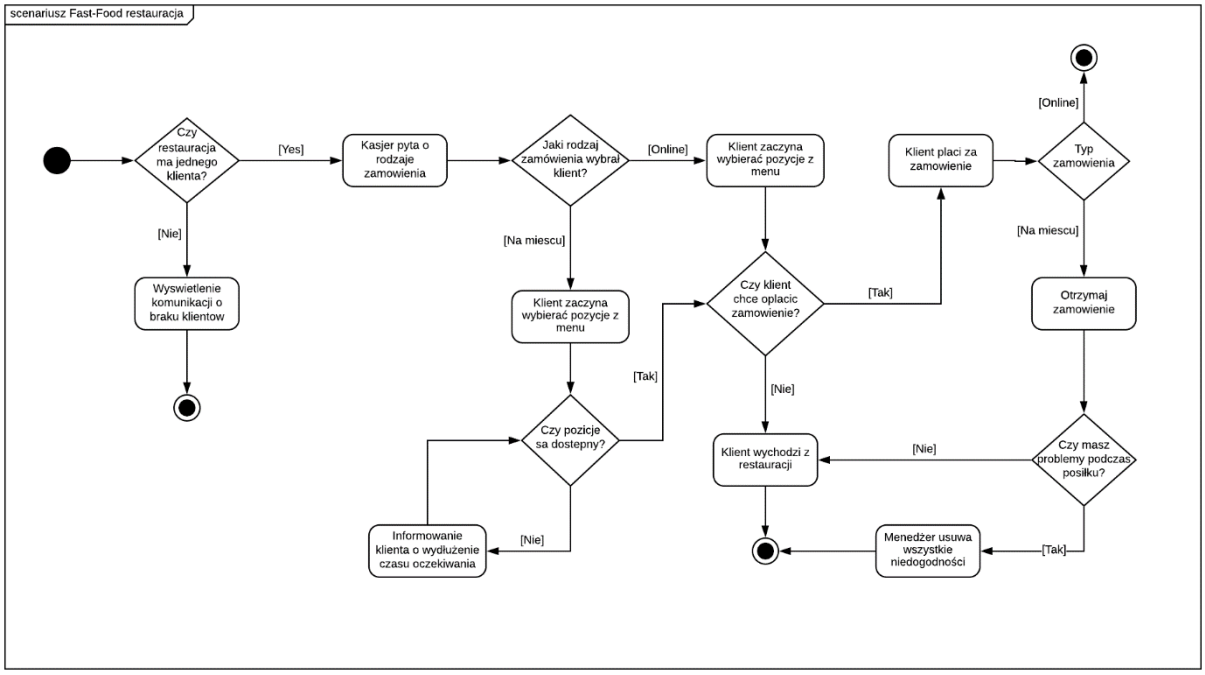
7a. Klient ma problemy podczas posiłku

7aa. Aktor menedżer usuwa wszystkie niedogodności.

**Stan końcowy:**

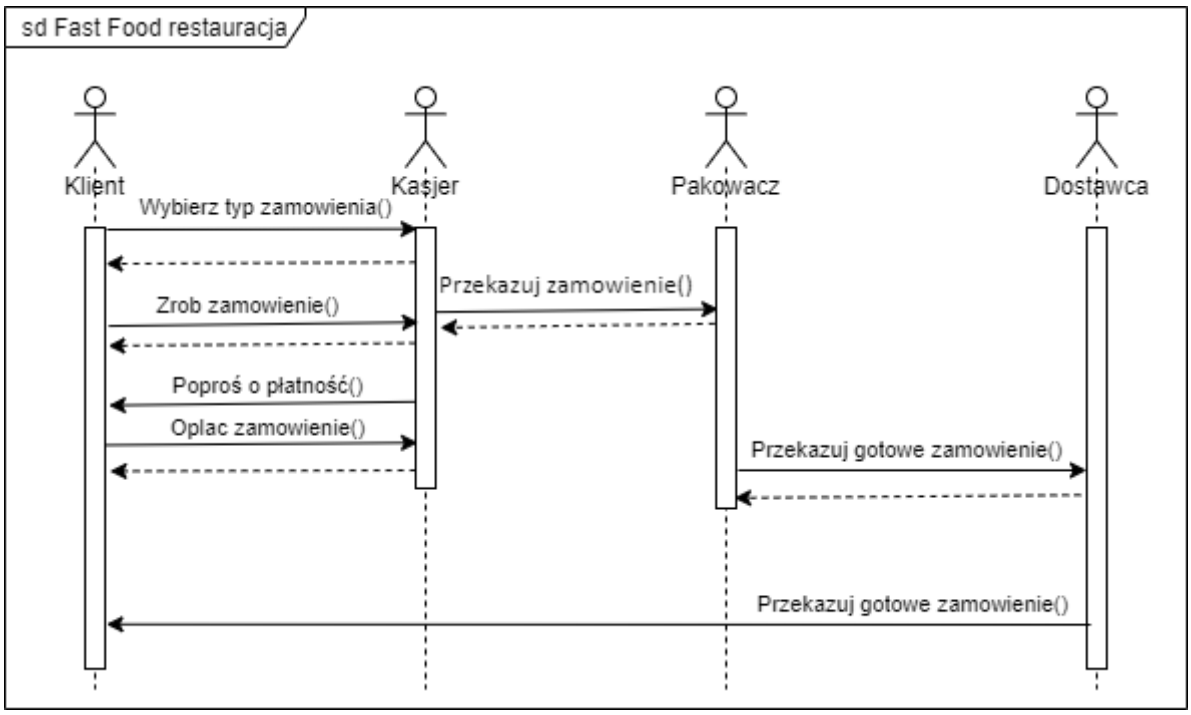
Klient wychodzi z restauracji

**Diagram:**




## 9. Analiza dynamiczna.

**Diagram sekwencji:**




Diagramy sekwencji to moja słabość...

**Diagram aktywnosci:**

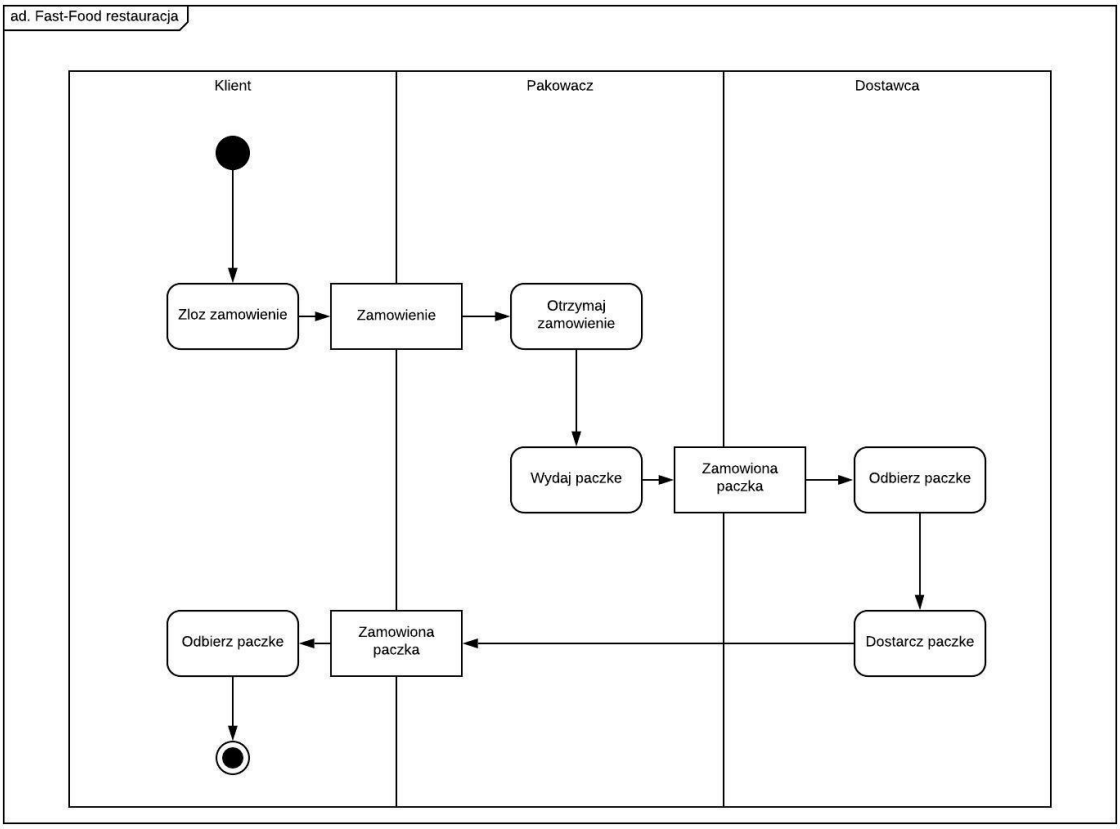
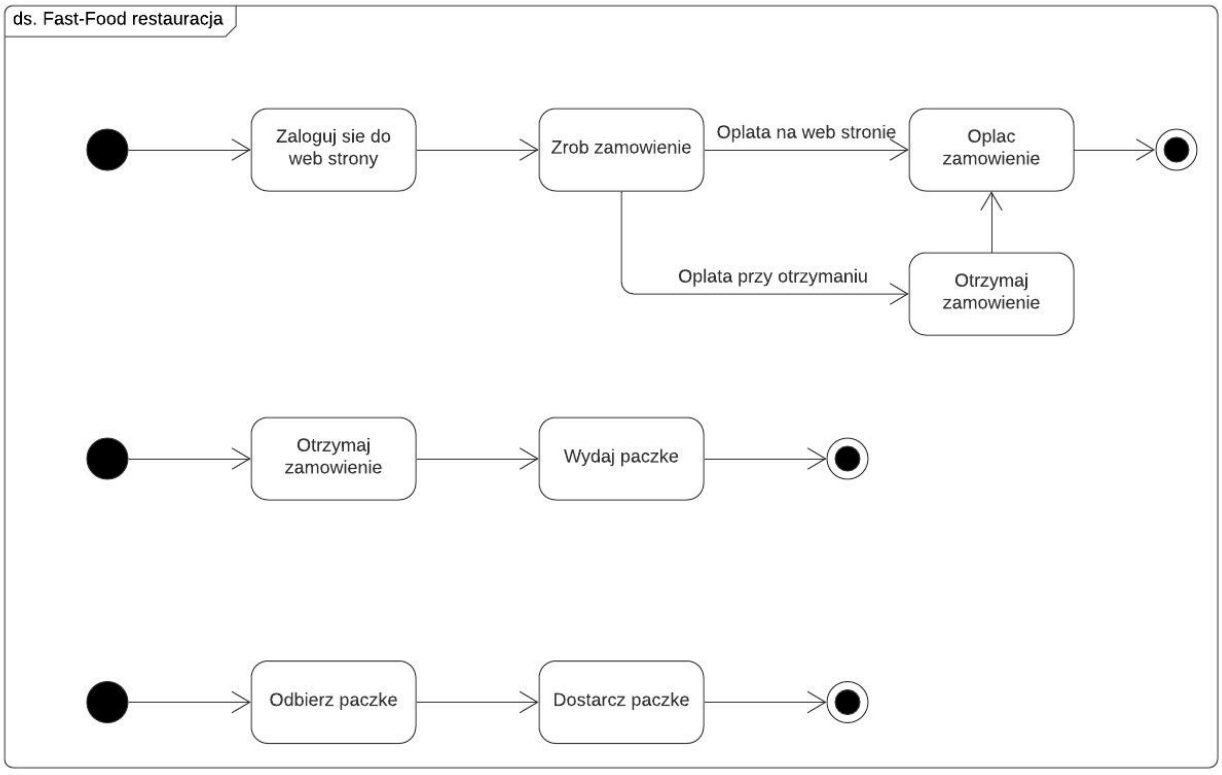
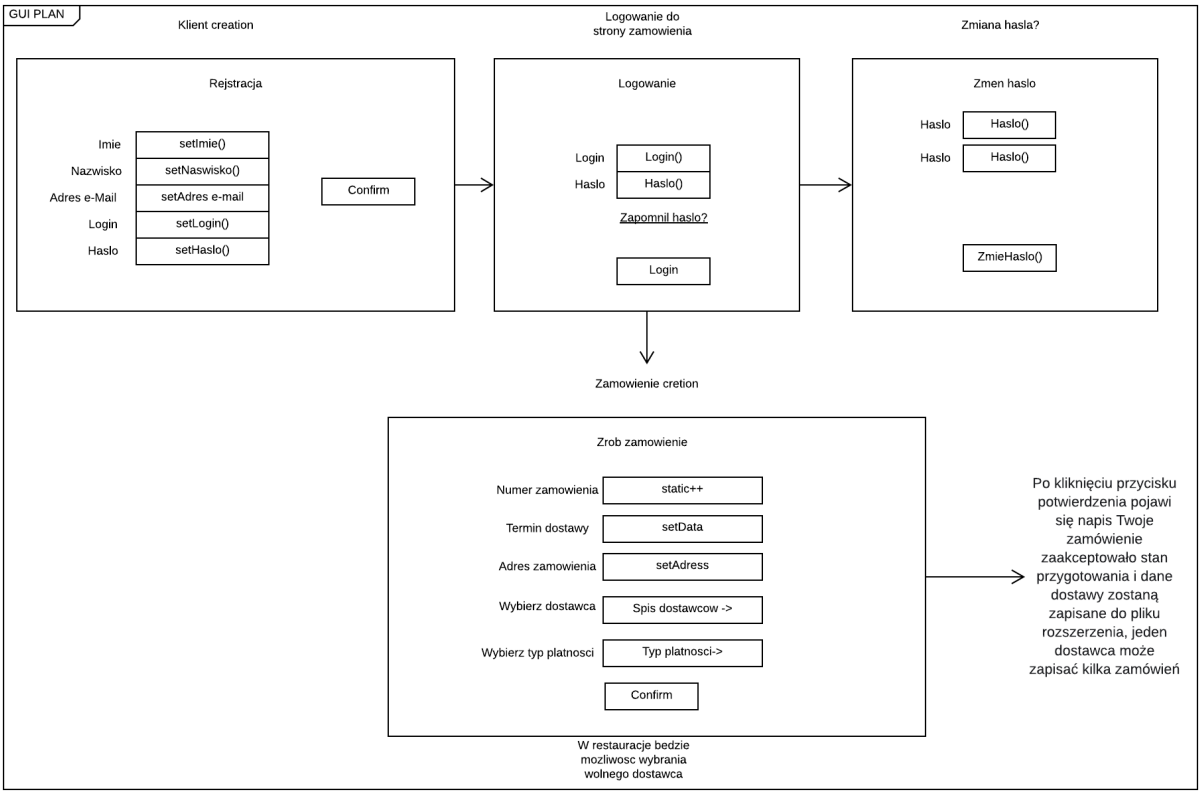



**Diagram stanow:**



## 10. Project GUI.




## 11. Podstawę do implementacji.

**Klasa abstrakcyjna**: tworzona poprzez dodanie słowa abstract: public abstract class Osoba{}, Pracownik{}, Zamowienie{}

**Dziedziczenie**: tworzone poprzez dodanie extend i klasy: class Pracownik extends class Osoba

**Polimorfizm metod**: polega na różnym wykonaniu metody o takiej samej nazwie w zależności, od klasy obiektu. Implementacja polega na różnym zdefiniowaniu konkretnej metody w wielu klasach.

Policz wynagrodzenie() -> Policz wynagrodzenie(staz), Policz wynagrodzenie(rank) i t.d.

**Overlapping**: Uzyskiwany poprzez stworzenie jednej klasy, która będzie zawierała w sobie atrybuty z kilku klas (Packowacz i Kasjer w jednej klasie Pracownik) + enum.

**Atrybut unikalny**: Należy sprawdzać czy taka wartość już nie istnieje. Wykorzystam do tego HashMap oraz metody findZamowienia(numer).

**Atrybut opcjonalny**: dla złożonych atrybutów przypisany jako null, dla prostych klasy opakowujące.
Dodam if w metodzie ToString.

**Asocjacja binarna**: asocjacja 1 do 1. Dodam tablicę arraylist do obu klas i odpowiadające im metody, które będą ze sobą współdziałać.

**Asocjacja jeden do wielu**: Dodam Vector oraz HashMap do klasy, gdy bedzie jeden oraz obiekt klasy do klasy, gdy bedzie wielu.

**Ograniczanie atrybutu**, które nie mogą być mniejsze niż pewna ustawiona wartość, zrobię, dodając osobną metodę.
**Atrybuty dynamiczne**, takie jak stasus, zrobię przez enum

## 12. Wymagania niefunkcjonalne.

- Stawka za godzinu pracy musze byc wieksza niz minimalna stawka za godzinu pracy (zl).

- Przedkosc pakowania nie moze byc mniej niz minimalna przedkosc pakowania (zamowienie/3 min).

- Liczba osób w restauracji nie może przekraczać 15 z powodu COVID-19 (liczba osob)