Dokumentacja

System XYZ

Specyfikacja wymagań

Informacje podstawowe

Autorzy

- Marek Wyrzykowski

Abstrakt

Dokument zawiera specyfikację wymagań dla systemu XYZ składającą się z wizji systemu, wymagań użytkownika oraz wymagań oprogramowania dla pierwszych iteracji projektu.

Historia zmian

Wersja	Data	Kto	Opis
1.0.0	12.2.2025	Marek Wyrzykowski	Wersja startowa
1.0.1	14.2.2025	Marek Wyrzykowski	Drobne poprawki
1.1.0	15.2.2025	Marek Wyrzykowski	Dodano procesy biznesowe:

1. Wprowadzenie

1.1 Cel dokumentu

Tutaj znajdzie się opis celu dokumentu... - uzupełnić pod koniec projektu

1.2 Streszczenie dla kierownictwa

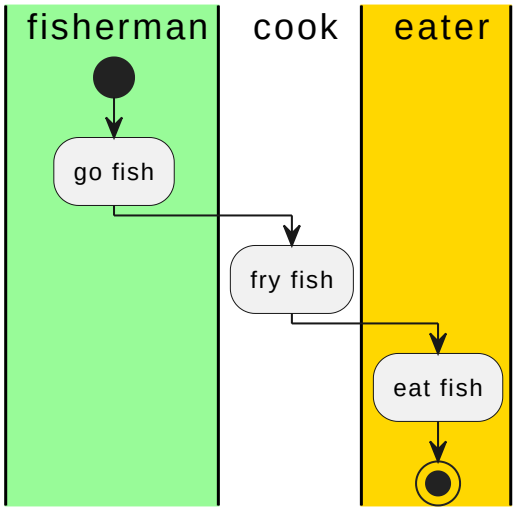
Tutaj znajdzie się streszczenie dla kierownictwa... - uzupełnić pod koniec projektu

2. Opis biznesu

2.1 Procesy biznesowe

(PB0001) Droga ryby na talerz

Diagram



Opis

Na podstawie diagramu PlantUML, można opisać proces biznesowy związany z **przygotowaniem i konsumpcją ryby**. Proces ten składa się z trzech głównych etapów, w których uczestniczą trzy różne role:

1. **Łowienie ryb (Go Fish)**:
   - Rola: Rybak (fisherman)
   - Opis: Rybak łowi ryby. Jest to początkowy etap procesu, w którym surowiec (ryba) jest pozyskiwany.
   - Kolor: `#palegreen`
2. **Smażenie ryb (Fry Fish)**:
   - Rola: Kucharz (cook)
   - Opis: Kucharz smaży ryby przygotowane przez rybaka. Na tym etapie surowiec jest przetwarzany w gotowy produkt.
   - Oznaczenie: `c`
3. **Jedzenie ryb (Eat Fish)**:
   - Rola: Konsument (eater)
   - Opis: Konsument zjada usmażoną rybę. Jest to końcowy etap procesu, w którym gotowy produkt jest konsumowany.
   - Kolor: `#gold`
   - Oznaczenie: `e`

Proces biznesowy polega na przekształceniu surowej ryby w gotowy posiłek, który jest konsumowany przez konsumenta. Każdy etap jest wykonywany przez inną osobę, co pokazuje podział odpowiedzialności w procesie.

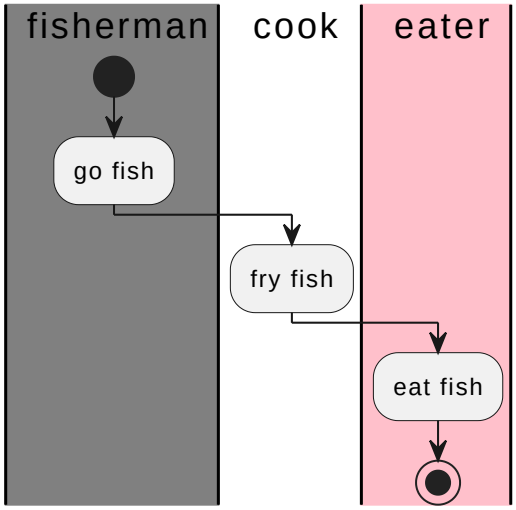
Kod diagramu

```
@startuml
|#palegreen|f| fisherman
|c| cook
|#gold|e| eater
|f|
```

```
start
:go fish;
|c|
:fry fish;
|e|
:eat fish;
stop
@enduml
```

(PB0002) Proces B

Diagram



Opis

opis procesu .....

Kod diagramu

```
@startuml
|#grey|f| fisherman
|c| cook
|#pink|e| eater
|f|
start
:go fish;
|c|
:fry fish;
|e|
:eat fish;
stop
@enduml
```

3. Wizja systemu.

Motto

Motto dla tego systemu

3.1 Opisy problemów

Problemy

(P0001) Problem czasu realizacji zamówienia

Priorytet	*wysoki*
Trudność	*niska*

Opis

Problem polegający na ..., dotyczący ..., którego rezultatem jest ..., można rozwiązać, budując system ..., co spowoduje (korzyści) ...

(P0002) Problem obsługi klientów

Priorytet	*wysoki*
Trudność	*niska*

Opis

Problem polegający na ..., dotyczący ..., którego rezultatem jest ..., można rozwiązać, budując system ..., co spowoduje (korzyści) ...

3.2 Interesariusze

Administrator

Nazwa i typ

Administrator systemu/Administrator danych publicznych. Jest to użytkownik systemu, który ma uprawnienia do zarządzania jego zasobami

```
Pozostałe cechy interesariusza odszukać w podręczniku
[...]

##### ???
```

3.3 Cechy funkcjonalne

(CF0001) Nazwa opisowa

Priorytet	*kluczowy*
Trudność	*średnia*

Opis

System musi umożliwiać administratorowi dodawanie nowych użytkowników.

```
##### Opis
System musi/powinien umożliwiać [użytkownikowi/aktorowi] [wykonanie
konkretnej akcji/osiągnięcie konkretnego celu]
```

3.4 Cechy jakościowe

Wydajność

(CJW0001) Nazwa opisowa

Typ	*wydajnościowy*
Priorytet	*kluczowy*
Trudność	*średnia*

Opis

System musi/powinien umożliwiać [użytkownikowi/aktorowi] [wykonanie konkretnej akcji/osiągnięcie konkretnego celu]

np. *System powinien być wydajny w przetwarzaniu dużych ilości danych.*

Sposób pomiaru

Możliwe wyniki pomiaru

Akceptowalne wartości pomiaru

Użyteczność

```
Pozostałe typy cech jakościowych odszukać w podręczniku
[...]
```

```
#### ???
```

3.5 Słownik

Administrator

Opis

Osoba lub rola w systemie, posiadająca uprawnienia do zarządzania i konfigurowania systemu, w tym do zarządzania użytkownikami, danymi oraz parametrami systemu

Przykłady użycia

- CF0001

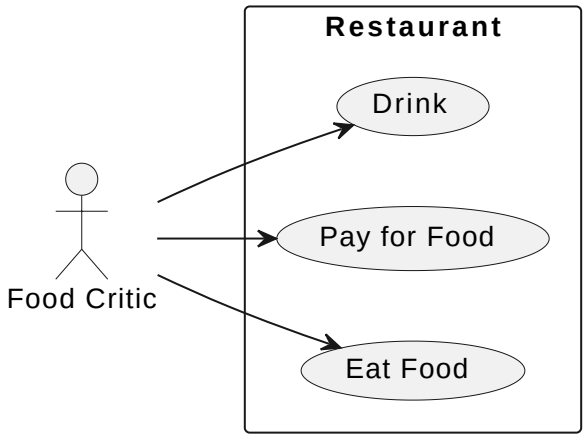
---

4. Wymagania użytkownika

4.1 Wymagania funkcjonalne

(PU0001) Wizytowanie restauracji przez krytyka kulinarnego

Diagram



Opis

Diagram przypadków użycia przedstawia interakcje między krytykiem kulinarnym a restauracją, uwzględniając różne aspekty korzystania z usług restauracji przez krytyka. Poniżej znajduje się szczegółowy opis każdego przypadku użycia:

- **Jedzenie (Eat Food)**: Ten przypadek użycia odnosi się do sytuacji, w której **krytyk kulinarny spożywa posiłki oferowane przez restaurację**. Jest to kluczowy element pracy krytyka, ponieważ na podstawie smaku, jakości składników, sposobu podania i ogólnego wrażenia z dania, krytyk formułuje swoją opinię na temat restauracji.

- **Płacenie za jedzenie (Pay for Food)**: Krytyk kulinarny **reguluje należność za spożyte posiłki**. Ten przypadek użycia, choć wydaje się oczywisty, jest ważny, ponieważ pokazuje, że krytyk kulinarny, tak jak każdy inny klient, ponosi koszty związane z wizytą w restauracji.
- **Picíe (Drink)**: Krytyk kulinarny **spożywa napoje oferowane przez restaurację**. Obejmuje to zarówno napoje bezalkoholowe, jak i alkoholowe, które są dostępne w menu restauracji. Wybór i jakość napojów również wpływają na ogólną ocenę restauracji.

Kod diagramu

```
@startuml
left to right direction
actor "Food Critic" as fc
rectangle Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
@enduml
```

4.2 Wymagania jakościowe i ograniczenia

(J0001) Maksymalna liczba kliknięć w celu otwarcia formularza awarii

Typ	*Użyteczność*
Priorytet	*przydatne*
Trudność	*średnia*
Sposób pomiaru	Przeprowadzenie serii nawigacji do ekranu formularza awarii z różnych miejsc w interfejsie użytkownika zgodnie ze scenariuszem TS112
Oczekiwane wartość	liczba kliknięć dla 90% sytuacji wynosi mniej niż 3; dla wszystkich sytuacji wynosi mniej niż 5

4.3 Słownik

4.3.1 Aktorzy

Użytkownik

Opis

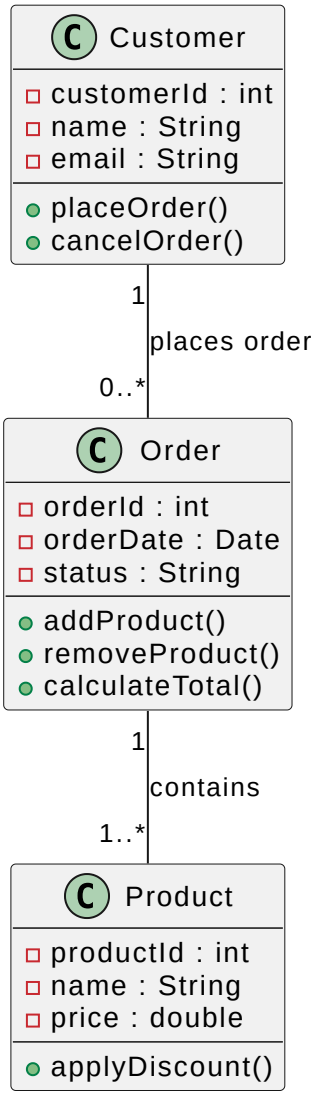
użytkownik systemu to osoba korzystająca z aplikacji webowej lub mobilnej w celu przeglądania, wyszukiwania, oceniania i komentowania udostępnionych zbiorów danych. Użytkownik może mieć różne

poziomy dostępu i uprawnień w systemie.

4.3.2 Słownik dziedziny

diagram klasy 1

Diagram



Opis

Diagram przedstawia trzy główne klasy, które modelują elementy systemu zamówień:

- **Customer (Klient)**
  - Posiada atrybuty: `customerId`, `name` oraz `email` (wszystkie prywatne).
  - Może wykonywać operacje: `placeOrder()` (składanie zamówienia) oraz `cancelOrder()` (anulacja zamówienia).
- **Order (Zamówienie)**
  - Zawiera atrybuty: `orderId`, `orderDate` oraz `status` (wszystkie prywatne).

    - Umożliwia operacje: `addProduct()` (dodanie produktu), `removeProduct()` (usunięcie produktu) oraz `calculateTotal()` (obliczenie łącznej wartości zamówienia).

- **Product (Produkt)**

    - Ma atrybuty: `productId`, `name` oraz `price` (wszystkie prywatne).
    - Umożliwia operację: `applyDiscount()` (zastosowanie rabatu).

Relacje między klasami ilustrują następujące zależności:

- **Relacja między Customer a Order:**
  Każdy klient (Customer) może mieć powiązane zero lub więcej zamówień (Order). Relacja ta jest oznaczona przez "1" przy Customer oraz "0..*" przy Order, co wskazuje na relację jeden-do-wielu.

- **Relacja między Order a Product:**
  Każde zamówienie (Order) zawiera jeden lub więcej produktów (Product). Relacja "1" przy Order oraz "1..*" przy Product oznacza, że każde zamówienie musi zawierać co najmniej jeden produkt.

Diagram ten pomaga zrozumieć strukturę systemu zamówień oraz zależności między klientem, zamówieniem i produktami, co jest kluczowe przy projektowaniu systemu opartego na tych encjach.

Kod diagramu

```
@startuml
class Customer {
    -customerId : int
    -name : String
    -email : String
    +placeOrder()
    +cancelOrder()
}

class Order {
    -orderId : int
    -orderDate : Date
    -status : String
    +addProduct()
    +removeProduct()
    +calculateTotal()
}

class Product {
    -productId : int
    -name : String
    -price : double
    +applyDiscount()
}

Customer "1" -- "0..*" Order : "places order"
Order "1" -- "1..*" Product : "contains"
@enduml
```