Studencka Pracownia Inżynierii Oprogramowania

Instytut Informatyki Uniwersytetu Wrocławskiego

Michał Sobecki, Artur Jankowski

# FAZA KONSTRUKCJI

Wrocław, 5 stycznia 2021

Wersja 0.5

| Data | Numer wersji | Opis | Autor |
|---|---|---|---|
| 2020-12-15 | 0.1 | Utworzenie Dokumentu | Artur Jankowski |
| 2020-12-16 | 0.2 | Dodanie nowej zawartości | Michał Sobecki |
| 2020-12-17 | 0.3 | Korekta dokumentu | Artur Jankowski |
| 2021-01-04 | 0.4 | Dodanie nowej zawartości | Artur Jankowski |
| 2021-01-05 | 0.5 | Korekta dokumentu | Michał Sobecki |

# Spis treści

1. Testy funkcjonalne....5
1.1 Przypadek 1....5
1.1.1 Opis przypadku....5
1.1.2 Kroki testu....6
1.2 Przypadek 2....7
1.2.1 Opis przypadku....7
1.2.2 Kroki testu....8
1.3 Przypadek 3....9
1.3.1 Opis przypadku....9
1.3.2 Kroki testu....10
2. Pomiary spełnienia wymagań niefunkcjonalnych....11
2.1 Prosty interfejs....11
2.2 Przyjemny w użyciu interfejs....12
2.3 Dostępność aplikacji....12
2.4 Testowalność systemu....13
2.5 Skalowalność systemu....13
2.6 Trwałość baz danych....14
2.7 Modularność i reużywalność systemu....14
2.8 Raportowanie błędów i logi....15
2.9 Bezpieczeństwo danych....15
2.10 Dokładność i skuteczność algorytmu oceny....16
2.11 Dostępność online....16
2.12 Bezawaryjność....16
2.13 Dokumentacja i pomoc....17
2.14 Szybkość działania aplikacji....18
3. Plan testowania....19
3.1 Alfa testy....19

3.2 Beta testy....19
3.3 Fazy testowania....20
3.4 Testy najważniejszych funkcji aplikacji....20
3.4.1 Testy dodania nowej oferty sprzedaży produktu....20
3.4.2 Testy dodania nowego produktu do prywatnego spisu....20
3.4.3 Testy dodania nowej oferty promocyjnej restauracji....21
3.4.4 Testy wyszukiwania informacji o ofercie sprzedaży....21
3.4.5 Testy wyszukiwania informacji o produkcie w prywatnym spisie....21
3.4.6 Testy wyszukiwania informacji o ofercie promocyjnej....21
4. Plan zarządzania jakością oprogramowania....22
4.1 Zasady....22
4.2 Realizacja....22
5. Dokładniejszy planu wykonania produktu....23
5.1 Prace wstępne....23
5.2 Iteracje Scrum....23
5.3 Testy i marketing....24
5.4 Ocena pracochłonności....24
6. Ocena zgodności wykonanych prac z wizją systemu i specyfikacją wymagań....24

# 1. Testy funkcjonalne

## 1.1 Przypadek 1

### 1.1.1 Opis przypadku

| ID | FUT-UC-1 |
|---|---|
| Tytuł | Dodanie oferty sprzedaży |
| Opis | Użytkownik chce dodać nową ofertę sprzedaży produktu |
| Aktor | Użytkownik aplikacji |
| Warunek początkowy | Użytkownik jest zalogowany do systemu |
| Warunek końcowy | Użytkownik dodał ofertę sprzedaży |
| Główny scenariusz | 1. Użytkownik przechodzi do panelu „Oferty”<br>2. Użytkownik wybiera opcję „Dodaj nową ofertę” znajdującą się na górze ekranu<br>3. System wyświetla formularz z polami do uzupełnienia<br>4. Użytkownik wypełnia pola:<br>- nazwa (max. 30 znaków alfanumerycznych)<br>- opis (max. 300 znaków alfanumerycznych)<br>- zdjęcia (max. 5, max. 5 MB każde)<br>- data ważności (data format: DD-MM-RRRR)<br>- lokalizacja (ulica: max. 50 znaków alfanumerycznych, współrzędne wpisane lub zaznaczone na mapie: max. 10 znaków alfanumerycznych)<br>- termin odbioru (wybiera dwie godziny:  dwa pola po 4 cyfry)<br>- cena (max. 10 cyfr)<br>5. Użytkownik wybiera opcję „Dodaj ofertę”<br>6. System prosi o potwierdzenie czynności<br>7. Użytkownik potwierdza dodanie oferty do publicznej listy |

### 1.1.2 Kroki testu

| Krok | Opis | Oczekiwany rezultat |
|---|---|---|
| 1 | Przejście do panelu „Oferty”. | System przechodzi do ekranu, w którym dostępne są wszystkie oferty sprzedaży produktów zapisane w bazie danych. Pole do wyszukiwania produktu jest puste. |
| 2 | Przejście do formularza dodania nowej oferty. | System po wybraniu opcji „Dodaj nową ofertę” przechodzi do formularza z polami do uzupełnienia: nazwa, opis, zdjęcia, data ważności, lokalizacja, termin odbioru, cena oraz przyciskiem do zatwierdzenia dodania oferty. |
| 3 | Uzupełnienie pól. | Użytkownik podaje następujące dane:<br>*Nazwa: Makowiec*<br>*Opis: Świeżo upieczone ciasto, bardzo pyszne.*<br>*Zdjęcia: (Trzy zdjęcia ciasta)*<br>*Data ważności: 16-01-2021*<br>*Lokalizacja: Wrocławska 56, 59°N, 67°E*<br>*Termin odbioru: 15:00 - 18:00*<br>*Cena: 150* |
| 4 | Zapisanie oferty. | Użytkownik wybiera opcję „Dodaj ofertę”, po czym system wyświetla komunikat proszący o potwierdzenie czynności. Użytkownik może potwierdzić dodanie oferty lub wrócić do uzupełniania danych. |
| 5 | Zapisanie danych | Użytkownik wybrał opcję potwierdzenia dodania oferty.<br>Nowe rekordy w tabelach bazy danych:<br>**Produkt wystawiony**: ${id}, ${id_sprzedającego}, 'Makowiec', 'Świeżo upieczone ciasto, bardzo pyszne.', ${zdjęcia}, 16-01-2021, ${id_lokalizacji}, 15:00 - 18:00, 150<br>Kolumny: ID, ID_sprzedającego, nazwa, opis, zdjęcie, data ważności, ID lokalizacji, termin odbioru, cena<br>**Lokalizacja**: ${id}, ${id_użytkownika}, 'Wrocławska 56', '67', '59'<br>Kolumny: ID, ID_użytkownika/firmy, adres, szerokość geograficzna, wysokość geograficzna |

## 1.2 Przypadek 2

### 1.2.1 Opis przypadku

| ID | FUT-UC-2 |
|---|---|
| Tytuł | Dodanie oferty promocji restauracji |
| Aktor | Właściciel restauracji |
| Warunek początkowy | Użytkownik jest zalogowany do systemu jako właściciel restauracji |
| Warunek końcowy | Użytkownik dodał ofertę promocyjną informującą o zniżkach w jego restauracji |
| Główny scenariusz | 1. Użytkownik przechodzi do panelu „Panel dla firm”<br>2. Użytkownik wybiera opcję „Dodaj nową ofertę” znajdującą się na górze ekranu<br>3. System wyświetla formularz z polami do uzupełnienia<br>4. Użytkownik wypełnia pola:<br>- typ zniżki (max. 50 znaków alfanumerycznych)<br>- dania (max. 300 znaków alfanumerycznych)<br>- okres zniżki (dwie daty: data i dwie godziny: dwa pola po 4 cyfry)<br>5. Użytkownik wybiera opcję „Dodaj ofertę”<br>6. System prosi o potwierdzenie czynności<br>7. Użytkownik potwierdza dodanie oferty do publicznej listy |

### 1.2.2 Kroki testu

| Krok | Opis | Oczekiwany rezultat |
|---|---|---|
| 1 | Przejście do panelu „Panel dla firm”. | System przechodzi do ekranu, w którym dostępne są opcje przejścia do ekranów związanych z zarządzaniem danych i ustawień dla firm oraz wyświetla wszystkie oferty promocyjne upublicznione przez firmę zachowane w bazie danych, wraz z opcją dodania nowej oferty. |
| 2 | Przejście do formularza dodania nowej oferty. | System po wybraniu opcji „Dodaj nową ofertę” przechodzi do formularza z polami do uzupełnienia: typ zniżki, dania, okres zniżki oraz przyciskiem do zatwierdzenia dodania nowej oferty. |
| 3 | Uzupełnienie pól. | *Oczekiwany rezultat: Użytkownik podaje następujące dane:*<br>*Typ zniżki: Zniżka 50% na wybrane produkty*<br>*Dania: Mięsne*<br>*Data zniżki: 05-01-2021 : 15-01-2021*<br>*Godziny zniżki: 17:00 - 19:00* |
| 4 | Zapisanie oferty. | Użytkownik wybiera opcję „Dodaj ofertę”, po czym system wyświetla komunikat proszący o potwierdzenie czynności. Użytkownik może potwierdzić dodanie oferty lub wrócić do uzupełniania danych. |
| 5 | Zapisanie danych | Użytkownik wybrał opcję potwierdzenia dodania oferty.<br>Nowe rekordy w tabelach bazy danych:<br>**Ogłoszenie**: ${id}, ${id_firmy}, ${link_do_menu}, 'Zniżka 50% na wybrane produkty', 'Mięsne', '05-01-2021 : 15-01-2021', '17:00 - 19:00'<br>Kolumny: ID, ID_firmy, menu_link, typ_zniżki, dania, data_zniżki, godziny_zniżki |

## 1.3 Przypadek 3

### 1.3.1 Opis przypadku

| ID | FUT-UC-3 |
|---|---|
| Tytuł | Dodanie produktu do prywatnego spisu |
| Aktor | Użytkownik aplikacji |
| Warunek początkowy | Użytkownik jest zalogowany do systemu |
| Warunek końcowy | Użytkownik dodał produkt do spisu |
| Główny scenariusz | 1. Użytkownik przechodzi do panelu „Produkty”<br>2. Użytkownik wybiera opcję „Dodaj nowy produkt” znajdującą się na górze ekranu<br>3. System wyświetla formularz z polami do uzupełnienia<br>4. Użytkownik wypełnia pola:<br>- nazwa (max. 30 znaków alfanumerycznych)<br>- opis (max. 300 znaków alfanumerycznych)<br>- zdjęcia (max. 5, max. 5 MB każde)<br>- data ważności (data format: DD-MM-RRRR)<br>5. Użytkownik wybiera opcję „Dodaj produkt”<br>6. System prosi o potwierdzenie czynności<br>7. Użytkownik potwierdza dodanie produktu do prywatnej listy |

### 1.3.2 Kroki testu

| Krok | Opis | Oczekiwany rezultat |
|---|---|---|
| 1 | Przejście do panelu „Produkty”. | System przechodzi do ekranu, w którym dostępne są wszystkie produkty użytkownika zapisane w bazie danych. Pole do wyszukiwania produktu jest puste. |
| 2 | Przejście do formularza dodania nowego produktu. | System po wybraniu opcji „Dodaj nowy produkt” przechodzi do formularza z polami do uzupełnienia: nazwa, opis, zdjęcia, data ważności oraz przyciskiem do zatwierdzenia dodania produktu. |
| 3 | Uzupełnienie pól. | *Użytkownik podaje następujące dane:*<br>*Nazwa: Banany*<br>*Opis: Świeże banany, prosto z Afryki.*<br>*Zdjęcia: (Cztery zdjęcia bananów)*<br>*Data ważności: 12-01-2021* |
| 4 | Zapisanie produktu. | Użytkownik wybiera opcję „Dodaj produkt”, po czym system wyświetla komunikat proszący o potwierdzenie czynności. Użytkownik może potwierdzić dodanie produktu lub wrócić do uzupełniania danych. |
| 5 | Zapisanie danych | Użytkownik wybrał opcję potwierdzenia dodania produktu.<br>Nowe rekordy w tabelach bazy danych:<br>**Produkt własny**: ${id}, ${id_posiadacza}, 'Banany', 'Świeże banany, prosto z Afryki.', ${zdjęcia}, '12-01-2021'<br>Kolumny: ID, ID_posiadacza, nazwa, opis, zdjęcie, data ważności |

# 2. Pomiary spełnienia wymagań niefunkcjonalnych

Wybraliśmy odpowiednie metryki dotyczące wymagań niefunkcjonalnych na podstawie normy ISO/IEC 9126 oraz ISO 25000. Nie są to wszystkie metryki (bo np. niektóre odnoszą się do innych dokumentów takich jak: RODO, specyfikacja dokumentacji itp.)

## 2.1 Prosty interfejs

### Understandable Input and Output

*Czy użytkownikowi łatwo zrozumieć jakiego rodzaju i jak musi wprowadzić dane oraz jaki wynik zwraca system?*

***Sposób:*** Testy przeprowadzanego na obserwowanym potencjalnym użytkowniku.

**Pomiar:** X = A / B

**Dane:**

A -> Liczba wejść i wyjść, które użytkownik jest w stanie zrozumieć w krótkim czasie w trakcie testów

B -> Liczba wejść i wyjść, które użytkownik próbuje zrozumieć podczas testów

**Ograniczenia:** 0 <= X <= 1

**Cel:** Im bliżej 1.0 tym lepiej

### Time Between Human Error Operations

*Ile czasu użytkownik traci ze względu na nieintuicyjność systemu?*

***Pomiar:*** X = T / N

**Dane:**

T -> czas działania

N -> liczba błędów użytkownika

**Ograniczenia:** 0 < X

**Cel:** Im więcej tym lepiej

## 2.2 Przyjemny w użyciu interfejs

### Attractive interaction

***Sposób:*** Ankieta skierowana do użytkowników na etapie prototypowania i na kolejnych iteracjach interfejsu.

### User Interface appearance customisability

*Jaka proporcja elementów interfejsu użytkownika może być zmieniana przez użytkownika pod względem wyglądu?*

***Pomiar:*** X = A / B

**Dane:**

A -> Liczba typów elementów interfejsu które można zmieniać.

B -> Całkowita liczba typów elementów interfejsu.

**Ograniczenia:** 0 <= X <= 1

***Cel:*** *Im bliżej 1.0 tym lepiej*

## 2.3 Dostępność aplikacji

### UAC-G-1 Physical accessibility

*Jaką cześć funkcji jest dostępna dla użytkowników z niepełnosprawnością?*

***Pomiar:*** X = A / B

**Dane:**

A -> liczba funkcji dostępna dla użytkowników z niepełnosprawnością.

B -> całkowita zaimplementowana liczba funkcji.

**Ograniczenia:** 0 <= X <= 1

***Cel:*** *Im bliżej 1.0 tym lepiej*

## 2.4 Testowalność systemu

### Autonomy of testability

*Jak niezależnie od siebie można testować system?*

**Sposób:** Policz liczbę zależności z innymi systemami na potrzeby testowania.

***Pomiar:*** X = A / B

**Dane:**

A -> Liczba zależności z innymi systemami zastąpiona „atrapami” (mockupy).

B -> Całkowita liczba zależności od innych systemów.

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.5 Skalowalność systemu

### PCAG-1 (Max.)No. of online requests

*Ile zapytań system potrafi przetworzyć na jednostkę czasu?*

***Pomiar:*** X = B / A

**Dane:**

A -> czas działania

B -> liczba przetworzonych zapytań

### PCAG-2 (Max.) No. of simultaneous accesses

*Ile użytkowników może korzystać jednocześnie z systemu?*

***Pomiar:*** X = B / A

**Dane:**

A -> czas działania

B -> liczba jednoczesnych dostępów

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.6 Trwałość baz danych

### Data corruption prevention

***Pomiar:***

X = 1 – A / N

Y = 1 – B / N

**Dane:**

A -> liczba pomniejszych uszkodzeń danych

B -> liczba poważnych uszkodzeń danych

N -> łączna ilość testów

**Ograniczenia:** 0 <= X, Y <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.7 Modularność i reużywalność systemu

### System software environmental adaptability (to the OS)

***Pomiar:*** X = A / B

**Dane:**

A -> liczba funkcji które są adaptowalne pomiędzy systemami IOS i Android.

B -> całkowita liczba funkcji, które działają na obu systemach.

### MMO-G-1 Condensability

***Pomiar:*** X = A / B

***Dane:***

A -> liczba komponentów na które, w znaczący sposób, nie wpływają zmiany w innych komponentach

B -> całkowita liczba komponentów

### MRE-G-1 Execution of reusability

*Jak wygląda użycie reużywalnych zasobów?*

***Pomiar:*** X = A / B

**Dane:**

A = liczba zasobów użytych ponownie

B = całkowita liczba zasobów która mogła zostać użyta ponownie

**Do wszystkich:**

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.8 Raportowanie błędów i logi

### Activity recording

*Jak skrupulatne jest zapisywanie statusu systemu?*

***Pomiar:*** X = A / B

**Dane:**

A -> liczba zaimplementowanych punktów zapisywania logów

B -> całkowita liczba zmiennych do raportowania

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.9 Bezpieczeństwo danych

### Data Encryption

***Pomiar:*** X = A / B

**Dane:**

A -> ilość danych podlegających szyfrowaniu

B -> łączna ilość danych podlegających ochronie

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.10 Dokładność i skuteczność algorytmu oceny

### Precision

***Pomiar:*** X = A / B

**Dane:**

A -> liczba funkcji spełniających kryteria

B -> liczba funkcji opisanych w specyfikacji

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.11 Dostępność online

### Availability

***Pomiar:*** X = { To / (To + Tr) }

**Dane:**

To -> czas działania

Tr → czas napraw

**Ograniczenia:** 0 < X < 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.12 Bezawaryjność

### Failure density (Fault density)

*Natężenie występowania awarii*

***Pomiar:***

X= NFAI / SIZE

Y= NFAU / SIZE

**Dane:**

NFAI -> liczba wykrytych defektów

NFAU -> liczba wykrytych awarii

SIZE -> rozmiar projektu

**Ograniczenia:** 0 < X, Y

***Cel:*** Początkowo większe wartości są lepsze, na koniec rozwoju systemu powinno dążyć do zera

### Mean time between failures (MTBF)

***Pomiar:***

X = TOPT / NAFI

Y = TSIB / NAFI

**Dane:**

TOPT -> czas działania

TSIB -> suma interwałów miedzy awariami

NAFI -> łączna liczba zaobserwowanych awarii

**Ograniczenia:** 0 < X, Y

***Cel:*** Im więcej tym lepiej

## 2.13 Dokumentacja i pomoc

### Completeness of user documentation and/or help facility

*Kompletność dokumentacji i ekranów pomocy.*

***Pomiar:*** X = A / B

**Dane:**

A -> Liczba opisanych funkcji

B -> całkowita liczba funkcji

**Ograniczenia:** 0 <= X <= 1

***Cel:*** Im bliżej 1.0 tym lepiej

## 2.14 Szybkość działania aplikacji

### Throughput time

**Pomiar:** X = A / T

**Dane:**

A -> liczba wykonanych zadań

T -> łączny czas testu

**Ograniczenia:** 0 < X

***Cel:*** Im więcej tym lepiej

### Worst case response time ratio

**Pomiar:** X= Tmax / Rmax

**Dane:**

Tmax -> MAX(Ti)

Rmax -> oczekiwane górne ograniczenie

MAX(Ti) -> maksymalny czas odpowiedzi w przeprowadzonych testach

**Ograniczenia:** 0 < X

***Cel:*** Im bliżej 1.0 tym lepiej

# 3. Plan testowania

W celu testowania aplikacji zamierzamy wyodrębnić osobne fazy, podczas których damy aplikację do przetestowania testerom, potencjalnym użytkownikom oraz restauratorom.

## 3.1 Alfa testy

Programiści piszący kod powinni w miarę możliwości pisać testy jednostkowe. Testowaniem integracji, manualnym oraz brakującymi unit testami zajmą się zatrudnieni przez nas testerzy. Testy te będą trwały przez cały okres wszystkich faz testowych, tak by kolejne iteracje nie psuły już istniejących i działających dobrze części systemu. Ich zadaniem w ostatniej fazie będzie sprawdzanie krytycznych części systemu takich jak bezpieczeństwo danych, w czym pomoże im audyt w wykonaniu zewnętrznej firmy.

## 3.2 Beta testy

1. Testowanie aplikacji na wybranej grupie potencjalnych użytkowników. Dostarczą nam oni informacji, jakie poprawki należałoby nanieść, aby użytkownik mógł w pełni cieszyć się dostarczonym produktem. Zależy nam na opinii tej grupy co do przyjemności użytkowania naszego interfejsu. Uczestnikom testów będziemy oferować specjalny status beta-testera oraz dodatkowe punkty w momencie startu aplikacji. Część tej grupy będzie też poddana odpłatnym testom pod okiem obserwatora, który może notować momenty zamyślenia, problemy użytkownika itp.

2. Testowanie aplikacji na wybranej grupie restauratorów. Dzięki temu będziemy wiedzieli, czy aplikacja spełnia ich wymagania oraz w razie wystąpienia uwag, przeanalizujemy je i wprowadzimy stosowne poprawki do systemu.

## 3.3 Fazy testowania

**alfa-testy:** cały czas trwania rozwoju aplikacji

**beta-testy:**

| Faza | Czas trwania | Opis |
|---|---|---|
| I faza | w trakcie trwania implementacji | pojedyncze osoby, na potrzeby przetestowania, pod okiem obserwatora |
| II faza | 2 tygodnie | zamknięte testy na mniejszej grupie (~50 użytkowników), w stałym kontakcie z zespołem, będziemy też aranżować transakcje i obserwować ich przebieg. |
| III faza | 2 tygodnie | otwarte testy pozwalające sprawdzić stabilność serwerów i bezpieczeństwo oraz ewentualne błędy wynikające z różnych konfiguracji urządzeń, z których korzystają testujący |

## 3.4 Testy najważniejszych funkcji aplikacji

### 3.4.1 Testy dodania nowej oferty sprzedaży produktu

**Przygotowanie:** Poprawne zalogowanie się do systemu.

**Opis:** Użytkownik dodaje nową ofertę, która poprawnie zapisuje się w bazie danych.

**Oczekiwanie:** Po wypełnieniu wymaganych pól oraz zatwierdzeniu dodania nowej oferty, system powinien zapisać w bazie danych informacje o nowej ofercie do tabeli „Produkt wystawiony” oraz informacji o jej lokalizacji do tabeli „Lokalizacja”.

### 3.4.2 Testy dodania nowego produktu do prywatnego spisu

**Przygotowanie:** Poprawne zalogowanie się do systemu.

**Opis:** Użytkownik dodaje nowy produkt, który poprawnie zapisuje się w bazie danych.

**Oczekiwanie:** Po wypełnieniu wymaganych pól oraz zatwierdzeniu dodania nowego produktu, system powinien zapisać w bazie danych informacje o nowym produkcie do tabeli „Produkt własny”.

### 3.4.3 Testy dodania nowej oferty promocyjnej restauracji

**Przygotowanie:** Poprawne zalogowanie się do systemu jako konto właściciela restauracji.

**Opis:** Użytkownik dodaje nową ofertę promocyjną, która poprawnie zapisuje się w bazie danych.

**Oczekiwanie:** Po wypełnieniu wymaganych pól oraz zatwierdzeniu dodania nowej oferty, system powinien zapisać w bazie danych informacje o nowej ofercie do tabeli „Ogłoszenie”.

### 3.4.4 Testy wyszukiwania informacji o ofercie sprzedaży

**Przygotowanie:** Poprawne zalogowanie się do systemu.

**Opis:** Użytkownik wyszukuje produkty o podanej przez niego nazwie w polu znajdującym się na górze ekranu oraz wybiera interesującą go ofertę, aby przejrzeć znajdujące się tam informacje z nią związane.

**Oczekiwanie:** Po wybraniu interesującej oferty system powinien wyświetlić okno z informacjami z nią związanymi.

### 3.4.5 Testy wyszukiwania informacji o produkcie w prywatnym spisie

**Przygotowanie:** Poprawne zalogowanie się do systemu.

**Opis:** Użytkownik wyszukuje produkty o podanej przez niego nazwie w polu znajdującym się na górze ekranu oraz wybiera interesujący go produkt, aby przejrzeć znajdujące się tam informacje z nim związane.

**Oczekiwanie:** Po wybraniu interesującego produktu system powinien wyświetlić okno z informacjami z nim związanymi.

### 3.4.6 Testy wyszukiwania informacji o ofercie promocyjnej

**Przygotowanie:** Poprawne zalogowanie się do systemu.

**Opis:** Użytkownik wyszukuje oferty o podanej przez niego nazwie w polu znajdującym się na górze ekranu oraz wybiera interesującą go ofertę, aby przejrzeć znajdujące się tam informacje z nią związane.

**Oczekiwanie:** Po wybraniu interesującej oferty system powinien wyświetlić okno z informacjami z nią związanymi.

# 4. Plan zarządzania jakością oprogramowania

## 4.1 Zasady

Tworząc plan zarządzania jakością oprogramowania, będziemy kierować się takimi zasadami jak:

1. Ukierunkowanie na klienta (również klient wewnętrzny)

2. Przywództwo (budowa wizji, identyfikacja wartości)

3. Zaangażowanie ludzi (satysfakcja, motywacja, szkolenia)

4. Podejście procesowe (koncentracja na poszczególnych krokach procesu i relacjach pomiędzy tymi krokami, pomiary)

5. Podejście systemowe (całe otoczenie procesu wytwórczego - wewnątrz i na zewnątrz firmy - dostawcy)

6. Ciągłe doskonalenie (doskonalenie stanu obecnego, ewolucja, a nie rewolucja)

7. Rzetelna informacja (zbieranie i zabezpieczanie danych do podejmowania obiektywnych decyzji)

8. Partnerstwo dla jakości (bliskie związki producentów z klientami)

## 4.2 Realizacja

Przykłady praktycznych rozwiązań dotyczące zasad zarządzania jakością:

- Przed rozpoczęciem tworzenia oprogramowania chcemy jak najszybciej przeszkolić zatrudnione osoby z technologii, które mogą być dla nich nowe, z założeniem, że ich nauka nie potrwa dłużej niż 3 tygodnie,

- Zamierzamy zbudować wizję i zidentyfikować wartości, którymi będziemy kierować się podczas tworzenia aplikacji,

- Tworząc kod oprogramowania, programistów będą obowiązywać główne zasady kodowania, określone w koncepcji wykonania systemu, dzięki czemu zachowamy spójność oraz kod ten będzie zrozumiały dla pracowników, którzy będą aktualizować system w przyszłości,

- Każda zmiana w kodzie programu będzie przed wypuszczeniem testowana oraz recenzowana przez przeznaczoną do tego odpowiednią osobę,

- Zespół będzie raportował postępy prac podczas codziennego standupu,

- Menadżer projektu czuwa nad trzymaniem terminów oraz zadaniami w backlogu,

- Będziemy stosować narzędzia do statycznej analizy kodu pod względem błędów oraz przyjętych konwencji pisania kodu,

- Testy systemu będą powstawać na bieżąco, a ich przejście będzie wymagane przy dołączaniu (integracji) kodu w kolejnych iteracjach (sprintach),

- Określimy przyjęte  standardy kodowania, komentowania, projektowania oraz dokumentacji,

- W trakcie, jak i po wykonaniu oprogramowania, planujemy nieustannie słuchać uwag naszych użytkowników, aby móc stale rozwijać system i wprowadzać należyte poprawki,

- Będziemy utrzymywać stały kontakt z firmami restauracyjnymi, dzięki czemu będziemy mogli udoskonalać elementy systemu przeznaczone dla nich, przez co również mamy nadzieję na promowanie w ich restauracjach naszego produktu oraz chęć wykupienia reklamy restauracji w aplikacji.

# 5. Dokładniejszy planu wykonania produktu

## 5.1 Prace wstępne

| Data | Czas trwania (w dniach roboczych) | Opis |
|---|---|---|
| 01.01.2021 – 15.01.2021 | 9 dni | Stworzenie konceptu aplikacji: określenie celów, wizji, identyfikacja wartości, opracowanie tablicy koncepcyjnej |
| 09.01.2021 – 29.01.2021 | 15 dni | Wywiad środowiskowy, rozmowy z potencjalnymi odbiorcami oraz prawnikami i restauracjami |
| 11.01.2021 – 18.01.2021 | 6 dni | Stworzenie historyjek użytkownika |
| 27.01.2021 – 29.01.2021 | 3 dni | Zakup sprzętu |
| 01.02.2021 – 08.02.2021 | 6 dni | Określenie wymagań |
| 08.02.2021 – 22.02.2021 | 11 dni | Wstępne opracowanie architektury systemu |
| 15.02.2021 – 28.02.2021 | 10 dni | Opracowanie wstępnego projektu interfejsu |
| 01.03.2021 – 03.03.2021 | 3 dni | Dyskusja na temat interfejsu, testy |
| 04.03.2021 – 11.03.2021 | 6 dni | Poprawienie wstępnej wersji interfejsu na podstawie zebranych informacji |

## 5.2 Iteracje Scrum

| | | |
|---|---|---|
| 15.03.2021 – 11.06.2021 | 62 dni (12 sprintów) | Implementacja backendu i frontendu |
| 12.05.2021 – 24.05.2021 | 10 dni | Zaprojektowanie bazy danych |
| 14.06.2021 – 24.09.2021 | 75 dni (15 sprintów) | Praca programistów na podstawie informacji zwrotnych od testerów |

## 5.3 Testy i marketing

| | | |
|---|---|---|
| 14.06.2021 – 9.08.2021 | 41 dni | Testowanie przez testerów |
| 12.07.2021 – 9.08.2021 | 21 dni | Beta testy |
| 19.07.2021 – 9.08.2021 | 16 dni | Testy bezpieczeństwa |
| 9.08.2021 – 16.08.2021 | 6 dni | Audyt bezpieczeństwa i jakości |
| 16.08.2021 – 17.09.2021 | 25 dni | Poprawki błędów |
| 20.09.2021 – 24.09.2021 | 5 dni | Wypuszczenie aplikacji na rynek |
| 30.08.2021 – 29.11.2021 | 64 dni | Akcja marketingowa |

## 5.4 Ocena pracochłonności

Przewidujemy, że prace od zaprojektowania do wypuszczenia aplikacji będą trwać od 01.01.2021 do 24.09.2021, czyli około 10 miesięcy (tj. 266 dni roboczych).

# 6. Ocena zgodności wykonanych prac z wizją systemu i specyfikacją wymagań

Najważniejszą różnicą pomiędzy poprzednimi dokumentami a obecnym jest wydłużenie się przewidywanego czasu na wykonanie systemu. Wydłużył się on o około 3 miesiące, co podyktowane jest poświęceniem większej ilości czasu na działania wstępne przed rozpoczęciem implementacji oraz wydłużeniem czasu testowania, by w momencie wypuszczenia produkt spotkał się on z pozytywnym odbiorem (nauczył nas tego CD Projekt Red :)).

Dodaliśmy parę dodatkowych metryk, które nie znalazły się w specyfikacji wymagań, mających sprawdzać wymagania niefunkcjonalne.

Po wykonaniu wszystkich potrzebnych dokumentacji zauważyliśmy, że w naszej bazie danych w tabeli „Ogłoszenie” powinniśmy dodać kolumnę „godziny_zniżki”, aby móc przechowywać informacje o godzinach, w których obowiązuje zniżka na wybrane dania.

Z mniejszych poprawek można wymienić zwiększenie maksymalnej ilości znaków dla opisów ofert/produktów (120 -> 300).

Poza tymi poprawkami nie zauważyliśmy większych nieprawidłowości związanych z poprzednimi założeniami.