III FAZA KONSTRUKCJI - GRUDZIEŃ

Alicja Danilczuk

Karolina Szlęk

Zadanie 1

Opracowanie przypadków testowych

Przypadek Testowy 1.1	
Nazwa	Wyszukiwanie połączeń
Opis	**Jako** użytkownik **chcę** sprawdzić dostępne połączenia pomiędzy dwoma miastami **by** kupić najbardziej odpowiadające mi bilety.
Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Użytkownik uzupełnia formularz wyszukiwania- stację wyjazdu, przyjazdu oraz preferowany termin.
Warunki końcowe	Po wybraniu połączenia użytkownik zostaje przekierowany na stronę kupna biletu na wybrane połączenie.
Oczekiwany rezultat	Użytkownik widzi listę dostępnych połączeń na podanej trasie.
Scenariusz (kroki do wykonania)	1. Wejście na stronę główną. 2. Wybranie przycisku Wyszukiwarka. 3. Uzupełnienie formularza wyszukiwania połączeń: Stacja Początkowa - Wrocław Stacja Końcowa - Katowice Przewoźnicy - pociągi i autobusy Wyjazd przed – 16:05 Data wyjazdu – 18 luty 2020 4. Wybranie przycisku Wyszukaj. 5. Wybranie z listy dostępnych najbardziej pasującego połączenia. 6. Przejście do strony kupna biletu u przewoźnika.

Przypadek Testowy 1.2	

Nazwa	Wybór połączeń - brak dostępnych przejazdów w podanym terminie/na podanej trasie
Opis	**Jako** użytkownik **chcę** sprawdzić dostępne połączenia pomiędzy dwoma miastami **by** kupić najbardziej odpowiadające mi bilety.
Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Użytkownik uzupełnia formularz wyszukiwania- stację wyjazdu, przyjazdu oraz preferowany termin.

Warunki końcowe	Stan aplikacji nie uległ zmianie.
Oczekiwany rezultat	Użytkownik widzi komunikat “Brak dostępnych połączeń.”
Scenariusz (kroki do wykonania)	1. Wejście na stronę główną. 2. Wybranie przycisku Wyszukiwarka. 7. Uzupełnienie formularza wyszukiwania połączeń: Stacja Początkowa - Wrocław Stacja Końcowa - Katowice Przewoźnicy - pociągi i autobusy Wyjazd przed – 16:05 Data wyjazdu – 18 luty 2013 3. Wybranie przycisku Wyszukaj. 4. Pojawienie się komunikatu o braku połączeń.

Przypadek Testowy 2.1	
Nazwa	Wgląd w Moje Bilety

Opis	**Jako** użytkownik **chcę** wejść w zakładkę Moje Bilety **by** zobaczyć zakupiony kiedyś bilet.
Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Użytkownik posiada konto w naszej aplikacji. W systemie istnieje użytkownik o adresie email: “login@g.com” oraz haśle “haslo”.
Warunki końcowe	Użytkownik otworzył interesujący go bilet.
Oczekiwany rezultat	Otwarcie pliku zawierającego wybrany bilet.
Scenariusz	1. Wejście na stronę główną. 2. Wybranie przycisku Zaloguj.

	3. Zalogowanie się do aplikacji adresem email: “login@g.com” oraz hasłem: “haslo”. 4. Przekierowanie do strony Moje Konto. 5. Wybranie odnośnika Wyświetl więcej. 6. Wybranie biletu z listy zakupionych biletów.

Przypadek Testowy 2.2	
Nazwa	Brak konta w aplikacji
Opis	**Jako** użytkownik **chcę** wejść w zakładkę Moje Bilety **by** zobaczyć zakupiony kiedyś bilet.
Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Brak
Warunki końcowe	Przekierowanie do strony z rejestracją.
Oczekiwany rezultat	Użytkownik otrzymuje powiadomienie od aplikacji o braku konta powiązanego z podanym adresem email.
Scenariusz	1. Wejście na stronę główną. 2. Wybranie przycisku Zaloguj. 3. Próba zalogowania do aplikacji adresem: “abc@g.com” oraz hasłem: “haslo”. 4. Wyświetlenie komunikatu “Nie ma takiego konta” oraz przycisku Załóż Konto. 5. Przekierowanie do strony Rejestracji.

Przypadek Testowy 3.1	
Nazwa	Dodanie tras do ulubionych
Opis	**Jako** zalogowany użytkownik **chcę** dodać trasę do ulubionych, **by** w przyszłości móc szybciej znajdować połączenia na często uczęszczanych trasach.

Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Użytkownik posiada konto w naszej aplikacji. W systemie istnieje konto o adresie email: “login@g.com” oraz haśle: “haslo”. Użytkownik jest zalogowany.

Warunki końcowe	Pojawienie się wybranej trasy na liście Ulubione
Oczekiwany rezultat	Użytkownik otrzymuje komunikat potwierdzający dodanie trasy do ulubionej.
Scenariusz (kroki do wykonania)	1. Wejście na stronę główną. 2. Wejście na stronę Moje Konto. 3. Wybranie odnośnika Dodaj trasę/przejazd. 4. Zaznaczenie opcji Dodaj trasę. 5. Uzupełnienie zmodyfikowanego formularza wyszukiwania połączeń: Stacja Początkowa - Wrocław Stacja Końcowa - Katowice Przewoźnicy - pociągi i autobusy 6. Wybranie przycisku Dodaj do ulubionych.

Przypadek Testowy 3.2	
Nazwa	Dodanie połączeń do ulubionych
Opis	**Jako** zalogowany użytkownik **chcę** dodać połączenie do ulubionych, **by** w przyszłości móc szybciej przechodzić do strony z podobnymi połączeniami.
Aktorzy	Aplikacja, Użytkownik
Warunki wstępne	Użytkownik posiada konto w naszej aplikacji. W systemie istnieje konto o adresie email: “login@g.com” oraz haśle: “haslo”. Użytkownik jest zalogowany.
Warunki końcowe	Pojawienie się wybranego połączenia na liście Ulubione.
Oczekiwany rezultat	Użytkownik otrzymuje komunikat potwierdzający dodanego połączenia do Ulubionych.
Scenariusz (kroki do wykonania)	1. Wejście na stronę główną. 2. Wejście na stronę Moje Konto. 3. Wybranie odnośnika Dodaj trasę/przejazd. 4. Zaznaczenie opcji Dodaj połączenie 5. Uzupełnienie zmodyfikowanego formularza wyszukiwania połączeń: Stacja Początkowa - Wrocław Stacja Końcowa - Katowice Przewoźnicy - pociągi i autobusy Wyjazd przed – 16:05

	6. Wybranie przycisku Wyszukaj. 7. Zaznaczenie serduszka przy wybranym połączeniu.

Zadanie 2

ISO/IEC 9126 – międzynarodowa norma oceny jakości oprogramowania utworzony w 1991r. przez Międzynarodową Organizację Normalizacyjną.

Norma składa się z czterech części:

- **Model jakości** – opisuje cechy charakterystyczne systemu. Skupia się zarówno na standardzie, w którym program powinien być napisany oraz na badaniu oczekiwań użytkowników;
- **Metryki zewnętrzne** – składa się z metryk, dzięki którym można mierzyć
- charakterystyki oprogramowania, by ustalić zachowania systemu np. W fazie testowania czy początków użytkowania programu;
- **Metryki wewnętrzne** – umożliwiają one pomiar charakterystyk systemu w fazie wczesnej implementacji;
- **Wykorzystanie metryk jakości** – raport techniczny zawierający zestaw przykładowych metryk opracowanych dla charakterystyk programu w cyklu życia oprogramowania.

Wymagania niefunkcjonalne:

- **Intuicyjny interfejs** – interfejs powinien prowadzić użytkownika po wszystkich dostępnych opcjach ukazując możliwe do wybrania funkcjonalności.
- **Bezpieczeństwo i anonimowość** - użytkownik ma pozostać anonimowy, chociaż baza danych zachowuje pdf z biletem użytkownika to jego dane (imię, nazwisko, adres) nie są zapisywane w bazie bezpośrednio w tabelach.

Korzystamy z miar:

- Data corruption Prevention

$$X = 1 - A / N$$

$$Y = 1 - B / N$$

Legenda:

A - liczba pomniejszych uszkodzeń

B - liczba poważnych uszkodzeń danych

N - łączna ilość testów

- Data encryption

X = A/B

Legenda:

A - ilość danych podlegających szyfrowaniu

B - łączna ilość danych podlegających ochronie

- **Bezawaryjność systemu.**

Korzystamy z miary:

Data exchangeability (User's success attempt based)

Sprawdzimy jak często komunikacja ze stronami przewoźników nie zawodzi.

Y= 1 – (A / B)

Legenda:

A -> liczba napotkanych niepowodzeń w komunikacji

B -> łączna liczba podjętych prób

- Możliwa **migracja systemu** na inny serwer.
- **Testowalność i łatwość konserwacji** - kod i infrastruktura muszą spełniać najwyższe standardy.

Korzystamy z miary:

Failure density (Fault density)

Dzieląc wartości przez ilość testów możemy ocenić skuteczność testów

X= NFAI / SIZE

Y= NFAU / SIZE

Legenda:

NFAI -> liczba wykrytych defektów

NFAU -> liczba wykrytych awarii

SIZE -> rozmiar projektu

Zadanie 3

Opracowanie planu beta testowania.

Aplikacja będzie na bieżąco testowana podczas procesu powstawania.

Testowanie będzie odbywać się zgodnie z piramidą testów.

Pierwszą fazą ostatnich testów rozpoczną nasi znajomi i rodzina których

poprosimy o użycie naszej aplikacji i konstruktywną krytykę. Wcielimy ich w role

użytkowników. Zalecimy testy na więcej niż jednym urządzeniu. Pomoże nam to wyeliminować potencjalne błędy jakie mogą się pojawiać na starszych urządzeniach lub takich z małą rozdzielczością ekranu. Z ich opinii skupimy się na czynniku user-friendly, jeśli oni nie będą w stanie zrozumieć interfejsu oznacza to, że należy go poprawić.

Następnie zwrócimy się do firmy zewnętrznej o pomoc w przeprowadzeniu testów, których wyniki uwzględnimy podczas przygotowań wersji beta.

Zanim wprowadzimy naszą finalną wersję do użytku publicznego wersję beta opublikujemy na serwerze i na popularnych serwisach społecznościowych poinformujemy o jej powstaniu. Poprosimy użytkowników o ich opinie. Na ich podstawie wprowadzimy poprawki i będziemy kontynuować testy do uzyskania oprogramowania nas satysfakcjonującego.

Zadanie 4

Sporządzenie planu zarządzania ryzykiem.

Potencjalne ryzyka:

1.Biznesowe

a. Podstawowym ryzykiem biznesowym jest brak zainteresowania ze strony użytkowników.
b. Istnieje również możliwość powstania konkurencyjnej aplikacji z lepszą, bogatszą ofertą (np. Rozszerzoną o połączenia samolotowe).
c. Możliwe niskie zainteresowanie reklamodawców.

2.Techniczne

a. Częste przeciążenia serwerów (zbyt mała wydajność serwerów).
b. Możliwe zmiany w interfejsie stron przewoźników uniemożliwiające nam wprowadzanie danych w ich wyszukiwarki.
c. Zawodność systemu (awaryjność).
d. Nieintuicyjny, skomplikowany interface – przeszkoda dla użytkowników.

Plan działania w razie wystąpienia:

1.Biznesowe

Skonsultujemy się ze specjalistami do spraw marketingu.

2.Techniczne

a. Rozważymy zmianę serwera na wydajniejszy, jeśli sytuacja będzie tego wymagała i dochód ze strony będzie wystarczająco duży.
b. Będziemy na bieżąco analizowali strony przewoźników, dodamy przycisk zgłoszenia błędu przy przejściu do zakupu biletu.
c. Proces wytwarzania oprogramowania powinien wyeliminować większość błędów, jednak na wypadek wystąpienia błędu w gotowym produkcie dostarczymy użytkownikom odpowiedni formularz połączony z issuetrackerem.
d. Przeprowadzimy testy zrozumienia przez użytkowników interfejsu naszej aplikacji.

Zadanie 5

Napisanie planu zarządzania jakością wytwarzania oprogramowania

Skorzystamy z metody zwinnej polegającej na pisaniu automatycznych testów przed stworzeniem docelowego kodu, czyli **Test Driven Development**.

Ponieważ postanowiliśmy utworzyć nasz projekt przy pomocy programowania obiektowego będziemy kierować się podstawowymi założeniami tegoż sposobu, czyli **SOLID**.

Chcemy umożliwić modelowanie systemów informatycznych ekspertom, którzy znają specyfikę problemu, lecz nie muszą znać się na projektowaniu architektury systemów informatycznych więc skorzystamy z techniki **DDD - Domaine Driven Design.**

Celem efektywnej pracy, zamierzamy wyznaczać i dostarczać małe części systemu w krótkich odstępach czasowych. Aby tego dokonać skorzystamy z popularnej metody znanej jako **SCRUM**.
Po wypuszczeniu naszej aplikacji na rynek planujemy jednak zmienić system pracy na korzystanie z oprogramowania **Jira.** Dzięki temu na bieżąco będziemy mogli ustalać plan rozwiązywania pojawiających się kolejnych problemów.
Jira to zamknięte oprogramowanie do śledzenia błędów oraz zarządzania projektami. Dzięki niej będziemy mogli weryfikować dokumenty, procesy, przepływ danych i zadania pracowników. Pozwoli im również w dogodny sposób przedstawiać nowe pomysły oraz dzielić się swoją wiedzą z innymi członkami zespołu.

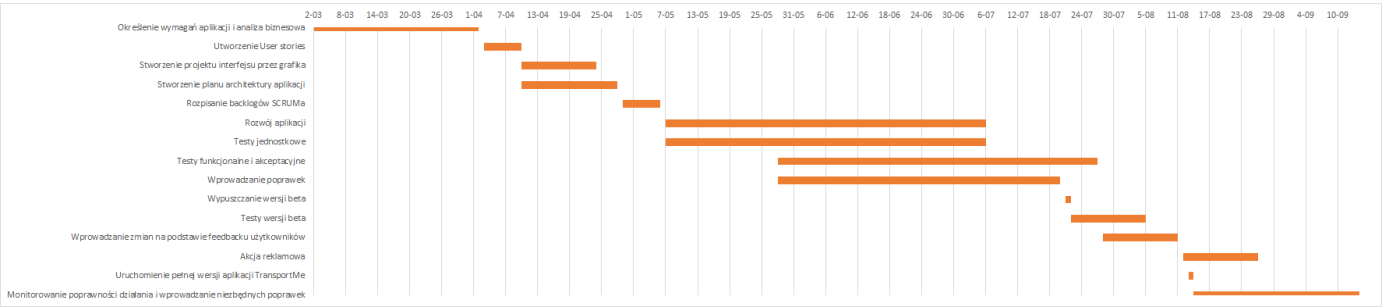
Aby zachować estetykę kodu zalecimy **code review**, wtedy przed przesłaniem Merge Request do głównego brancha deweloperskiego będzie musiał on zostać zaakceptowany przez dwóch programistów z zespołu, którzy ocenią kod pod względem czytelności, wydajności i ogólnej jego architektury.

Skorzystamy z Git jako system kontroli wersji oraz GitHub jako jego repozytorium.

Rozważamy przeprowadzanie **Kwartalnych audytów aplikacji** przez konsultanta spoza zespołu sprawdzającego pokrycie wymagań funkcjonalnych i niefunkcjonalnych.
A także po zakończeniu prac nad stroną, przed wypuszczeniem jej na rynek zlecimy zewnętrznej firmie przeprowadzenie **audytu bezpieczeństwa.**

Zadanie 6

Sporządzenie dokładniejszego planu wykonania produktu, dokładniejsza ocena pracochłonności, dokładniejszego harmonogramu.



https://imgur.com/a/gvt7ytZ

Zadanie 7

Ocena zgodności wykonanych prac z wizją systemu i specyfikacją wymagań.

Estymacja czasu nie zmieniła się. Rozbudowaliśmy jednak metodologie, którymi będziemy się kierować oraz poszerzyliśmy zasób narzędzi z których zamierzamy skorzystać przy wytwarzaniu aplikacji.