# Kontekst testowania

1. Terminologia i definicje
    a. Testowanie modułowe (Unit tests) - Testowanie modułowe (zwane także testowaniem jednostkowym, testowaniem komponentów lub testowaniem programu) skupia się na modułach, które można przetestować oddzielnie. Cele tego typu testowania to między innymi: zmniejszanie ryzyka; weryfikacja zgodności zachowań funkcjonalnych i niefunkcjonalnych modułu z projektem i specyfikacjami

    b. Testowanie integracyjne (Integration tests) - Testowanie integracyjne skupia się na interakcjach między modułami lub systemami. Cele testowania integracyjnego to: zmniejszanie ryzyka;  weryfikacja zgodności zachowań funkcjonalnych i niefunkcjonalnych z projektem i specyfikacjami; budowanie zaufania do jakości interfejsów; wykrywanie defektów (które mogą występować w samych interfejsach lub w modułach/systemach); zapobieganie przedostawaniu się defektów na wyższe poziomy testowania

    c. Testowanie systemowe (end to end tests) - skupia się na zachowaniu i możliwościach całego systemu lub produktu, często z uwzględnieniem całokształtu zadań, jakie może on wykonywać oraz zachowań niefunkcjonalnych, jakie można stwierdzić podczas wykonywania tych zadań. Cele testowania systemowego to między innymi: zmniejszanie ryzyka; werfikacja zgodności zachowań funkcjonalnych i niefunkcjonalnych systemu z projektem i specyfikacjami; walidacja, czy system jest kompletny i działa zgodnie z oczekiwaniami; budowanie zaufania do jakości systemu jako całości; wykrywanie defektów; zapobieganie przedostawaniu się defektów na wyższe poziomy testowania lub na produkcję

2. Podprocesy projektu i podprocesy testowe:
    a. testy integracyjne (baza danych zewnętrzne API, komunikacja backend - frontend)
    b. testy E2E (odwzorowanie wszystkich przypadków użycia opisanych w [1])
    c. unit testy (testy wszystkich modułów zawierających implementację logiki biznesowej)
    d. testy manualne (testy efektów wizualnych interfejsów użytkownika i poprawności interakcji użytkownika końcowego z systemem)

3. Elementy testowe
    a. Integracja:
        i. z bazą danych
        ii. z zewnętrznymi API
        iii. z serwisem uwierzytelniającym
    b. Logika biznesowa
        i. zgodność z wymaganiami funkcjonalnymi
        ii. poprawność zaimplementowanych funkcjonalności
    c. Interfejs użytkownika
        i. poprawne odwzorowanie przypadków użycia (use case)

ii. poprawna integracja z backend-em
iii. efekty wizualne

d. Bezpieczeństwo
i. implementacji autoryzacji i uwierzytelnienia
ii. połączenia z bazą danych, zewnętrznymi API
iii. przechowywanie kluczy i haseł

4. Zakres testów
a. czas dostępu do bazy danych i zewnętrznych API
b. stabilność połączeń z API i bazą danych
c. poprawność implementacji logiki biznesowej
d. przystępność interfejsu użytkownika
e. bezpieczeństwo przechowywanych kluczy i haseł
f. zgodność integracji:
i. backend - frontend
ii. backend - baza danych
iii. backend - zewnętrzne API
iv. frontend - serwis uwierzytelniający

5. Założenia i ograniczenia
a. wszystkie rodzaje testów są wykonywane:
i. podczas budowania projektu
ii. przed mergowaniem do głównej gałęzi
iii. podczas CI/CD pipeline'u
b. czas wykonania CI/CD pipeline jest krótszy niż 30 minut
c. pokrycie kodu wynosi co najmniej 90%
d. nowe moduły zawierające logikę biznesową muszą posiadać testy jednostkowe przed zmergowaniem do głównej gałęzi projektu
e. manualne testy efektów wizualnych są wykonywane przed dodaniem nowych funkcjonalności interfejsu użytkownika

6. Interesariusze (lista interesariuszy i ich powiązania z procesem testowym).
a. Członkowie zespołu developerskiego (niezaangażowani w proces testowania produktu)
b. Project Manager Krzysztof Stępień - ocena stopnia postępu testowania, tworzenie harmonogramu prac, ocena gotowości produktu

# Rejestr ryzyka. Identyfikacja ryzyka, którym odpowiadać będzie testowanie wynikłe z planu.

1. Ryzyka projektowe
a. Nie zaimplementowanie wszystkich funkcjonalności w zadanym czasie
I. Harmonogram określający kolejne kroki oraz uwzględniający zapas czasowy na ewentualne opóźnienia
b. Problemy z integracją funkcjonalności rozwiniętych przez różne grupy projektowe

I. Przeprowadzanie testów integracyjnych oraz end-to-end sprawdzających działanie zarówno nowo dodanych funkcjonalności jak i tych zaimplementowanych wcześniej

c. Niedobór bądź brak wymaganych umiejętności wśród członków zespołu do odpowiedniej realizacji funkcjonalności/aplikacji

I. Odpowiednio wczesne rozpoznanie problemu oraz nauka wymaganych technologii/zdobycie potrzebnej wiedzy/reorganizacja zadań

2. Ryzyka produktowe (jakościowe)

a. Nieuwzględnienie wszystkich możliwych przypadków w testowaniu i wynikające z tego błędy występujące u użytkownika końcowego produktu

I. Testy end-to-end wszystkich funkcjonalności na końcowym produkcie przeprowadzone przez więcej niż jedną osobę

b. Końcowe oprogramowanie nie spełnia wszystkich założonych funkcjonalności

I. Lista wszystkich wymaganych funkcjonalności oraz bieżąca analiza zrealizowanych oraz oczekujących na zrealizowanie funkcjonalności

II. Testowanie każdej zaimplementowanej funkcjonalności

c. Nieintuicyjny design UI utrudniający użytkownikom korzystanie z programu

I. Pozyskanie zewnętrznych opinii w oparciu o zarys/prototyp na wczesnym stadium pracy nad UI

d. Niewłaściwe uprawnienia przyznane różnym rodzajom użytkowników

I. Sprawdzenie osobno dla każdego rodzaju użytkownika zarówno dostępności danej funkcji jak i także braku możliwości wykonania danej czynności (sprawdzając zarówno możliwość skorzystania z danej funkcji jak i upewniając się o braku uprawnień niwelujemy ryzyko pominięcia danego aspektu w przypadku któregoś rodzaju użytkownika)

e. Istnienie luk bezpieczeństwa

I. Stworzenie listy potencjalnych słabych punktów i zapobieganie ich powstawaniu

II. Dostosowane przypadków testowych aby uwzględniały weryfikację działania funkcjonalności/aplikacji również w aspekcie bezpieczeństwa

# Strategia testowa. Opisuje podejście testowe do konkretnego projektu.

1. Produkty testowania
    a. Produkty planowania
        i. Poniższy dokument
        ii. Readme.md
        iii. Tablica Kanban
    b. Produkty monitorowania i nadzoru testów
        i. Raporty generowane przez Cypress, Vitest
        ii. CI/CD
        iii. Tablica Kanban
    c. Produkty analizy
        i. Raport pokrycia kodu

        ii. Scenariusze testowania
    d. Produkty projektowania testów
        i. Pliki z testami
        ii. Scenariusze testowania
        iii. Stworzenie środowiska testowego
    e. Produkty implementacji testów
        i. Procedury testowe i kolejność ich wykonywania
        ii. Zestawy testowe
        iii. Harmonogram wykonywania testów
    f. Produkty wykonywania testów
        i. Raporty generowane przez narzędzia testowe
        ii. Raport o wykazanych przez testy defektach
    g. Produkty ukończenia testów
        i. Lista poprawek eliminujących wykryte w testach defekty
2. Narzędzia

    Manualne sprawdzenie funkcjonalności wprowadzone jest jako jeden z etapów kontroli jakości przy integracji funkcjonalności z główną gałęzią

    a. Testowanie manualneaplikacji.
    b. Testowanie automatyczne
        i. Testy automatyczne jednostkowe
            1. Biblioteka Vitest wraz z React Testing Library
                a. Vitest to środowisko uruchomieniowe testów, odpowiedzialne za uruchamianie, automatyzację i sterowanie środowiskiem testowym (tj. implementacją
                    1. React Testing Library zapewnia narzędzia pozwalające na manipulację zawartością treści DOM w przeglądarce oraz budowanie asercji.
                b. Testy automatyczne end-to-end (E2E)
                c. Biblioteka Cypress wraz z Cypress Testing Libraryprzeglądarki internetowej).
                d. Cypress to narzędzie pozwalające na tworzenie i uruchamianie testów polegających na wykonywaniu czynności na stronie w sposób automatyczny poprzez udostępniane metody pozwalające na interakcje bezpośrednio z uruchomioną instancją przeglądarki internetowej
    c. Jarzmo testowe
        i. happy-dom - implementacja przeglądarki zrealizowana w języku JavaScript, kompatybilna z biblioteką Vitest, umożliwiająca uruchamianie testów jednostkowych
        ii. pełne implementacje przeglądarek internetowych (w przypadku testów end-to-end)
        iii. okrojona, pozbawiona interfejsu graficznego implementacja przeglądarki internetowej oparta o projekt chromium (w przypadku testów end-to-end)

    d. Zarządzanie danymi
        i. W celu pokrycia testami funkcjonalności związanych z danymi dynamicznymi, zastosowano narzędzie do generacji danych losowych, aby odtworzyć model interakcji z użytkownikiem.
3. Techniki projektowania testów
    a. Czarnoskrzynkowe
        i. Testy end-to-end
        ii. Integracyjne
    b. Białoskrzynkowe
        i. Testy jednostkowe

Scenariusze przypadków testowych tworzone są w oparciu o doświadczenie, w szczególności za pomocą techniki zgadywania błędów.

4. Kryteria zakończenia testów
    a. Osiągnięcie odpowiedniego procentowego poziomu pokrycia
    b. Przeprowadzenie testów na wszystkich uwzględnianych w procesie testowania komponentach
    c. Niska gęstość defektów zapewniona przez pozytywny rezultat wszystkich zaprojektowanych testów
5. Metryki
    a. Procentowe pokrycie testowe
    b. Liczba wykrytych i usuniętych defektów
    c. Postęp projektu uwidoczniony na tablicy Kanban
6. Wymagania co do danych testowych
    a. Głównym wymaganiem danych testowych jest odpowiedni dobór do warunków i założeń testowych. Przykładowo, testując formularz logowania, pożądane dane do przypadków weryfikujących poprawność badanego scenariusza powinny spełniać warunki testu tj. testy poprawnego logowania powinny wykorzystywać dane spełniające warunki walidacji, testy odrzucenia logowania zaś wykorzystywać dane niespełniające walidacji.
7. Wymagania co do środowiska testowego.
    a. Baza danych wypełniona przykładowymi danymi spełniającymi wymagania narzucone przez warunki testowe (patrz: punkt 6).
    b. Wymagania techniczne pokrywające się z wymaganiami pełnej wersji aplikacji.
    c. Środowisko testowe możliwie blisko odzwierciedlające specyfikę środowiska produkcyjnego.
    d. Zaślepienie elementów niepodlegających badanemu scenariuszowi testowemu w sposób odzwierciedlający ich funkcjonowanie w pełnej wersji aplikacji.
8. Testy regresji i retesty.
    a. Pełen zestaw testów wykonywany jest każdorazowo po dołączeniu nowej funkcjonalności, tym samym uwzględniając regresję i retesty w procesie testowania.
    b. Integralność procesu będzie zachowana poprzez narzędzia CI/CD oraz bibliotekę Husky, która wykona proces testowy przy każdym commicie.

9. Kryteria zawieszenia i wznowienia testów oraz osoby odpowiedzialne za decyzję odnośnie ich przeprowadzenia (np. warunki pełnej akceptacji, warunkowej akceptacji lub odrzucenia).
    a. Warunki pełnej akceptacji
        i. Testy spełniają warunki zakończenia (patrz: punkt 4) oraz kod zaakceptowany w procesie code review.
    b. Warunkowa akceptacja
        i. Testy spełniają warunki zakończenia, kod niezaakceptowany w procesie code review.
    c. Warunki odrzucania
        i. Testy nie spełniają warunków zakończenia.

# Personel

1. Role, zadania i odpowiedzialności (opisują, jakie role występują w procesie testowym oraz kto i za co jest odpowiedzialny).
    a. Rozróżniamy dwie główne role: kierownika testów oraz testera, którym przypisujemy następujące zadania i odpowiedzialności:
        i. Kierownik testów:
            1. jest to jeden z liderów zespołu developerskiego
            2. planuje cele testów oraz terminy ich wykonania
            3. na bieżąco aktualizuje plan testów dostosowując go do wydajności czasowej zespołu
            4. koordynuje pracę testerów, pomaga w ich integracji
            5. sprawdza jakość i pokrycie testów przez testerów (wprowadza miary oceniające jakość testów oraz ich pokrycie)
            6. monitoruje i przygotowuje raporty dotyczące postępu zespołu testującego
            7. podejmuje główną decyzję dotyczącą stosowanych bibliotek/środowisk do testów
            8. służy pomocą i radą - jest mentorem zespołu
            9. reprezentuje zespół

        ii. Tester:
            1. może być to członek zespołu developerskiego, który tworzy funkcjonalności w projekcie
            2. tworzy testy jednostkowe oraz integracyjne dla stworzonej przez siebie (lub developera, w wypadku, gdy nie jest członkiem zespołu developerskiego) funkcjonalności
            3. uczestniczy w opracowywaniu planów testów
            4. analizuje dane funkcjonalności pod kątem ich testowalności (wymagania, scenariusze, kryteria akceptacji, przypadki brzegowe)
            5. przygotowuje dane testowe

6. członek zespołu developerskiego ma za zadanie obowiązkowo pisać testy jednostkowe oraz integracyjne do swoich funkcjonalności
7. tworzy testy E2E
8. korzysta z narzędzi ustalonych przez kierownika testów
9. przegląda testy innych członków zespołu pod kątem ich jakości
10. dokumentuje odchylenia od oczekiwanych rezultatów po wykonaniu testów oraz informuje zespół developerów o konieczności poprawy funkcjonalności
11. ewaluuje charakterystyki niefunkcjonalne, przede wszystkim: wydajność, niezawodność

   b. W strukturze naszego personelu występują **pary testujące**, są to zespoły 2-osobowe, złożone z dwóch testerów, którzy wykonują swoje zadania i odpowiedzialności w ramach przypisanych parze funkcjonalności aplikacji.
2. Potrzeby związane z zatrudnieniem (jeśli np. brakuje w zespole osób określonych umiejętnościach).
   a. w zespole developersko-testowym każdy członek potrafi na podstawie dostarczonych materiałów poprawnie testować wytwarzane oprogramowanie, w związku z czym, nie są spodziewane żadne braki kadrowe;
   b. dodatkowo przez wzgląd na naukowo-badawczy charakter projektu nie byłoby możliwe zatrudnienie do zespołu żadnej osoby zewnętrznej;
3. Potrzeby związane ze szkoleniami.
   a. braki w umiejętnościach i posługiwaniu się danym językiem programowania/biblioteką lub narzędziem do testowania będą uzupełniane poprzez współpracę i coaching wewnątrz zespołu;

# Czynności testowe

1. **Planowanie testów:** Planowanie testów skupia się na zdefiniowaniu celów testowych oraz określenia podejścia do osiągnięcia ich. Obejmuje to następujące działania:
   a. Określenie zakresu oraz celu testów, a także związany z nimi ryzyka
   b. Wybranie ogólnego podejście do testowania
   c. Określenie jakie części aplikacji trzeba przetestować i określić jakie zasoby ludzkie oraz inne będą do tego potrzebne
   d. Zaplanowanie procesu analizy, projektowania, implementacji, wykonywania i oceny testów poprzez określenie konkretnych terminów
   e. Wybór miar do monitorowania i nadzorowania testów
   f. Określenie szczegółowości i schematu dokumentacji testów.
2. **Monitorowanie testów i nadzór nad testami:** Monitorowanie testów polega na trwającym cały czas porównywaniu rzeczywistego z zaplanowanym postępem testowania przy użyciu miar specjalnie w tym celu zdefiniowanych w planie testów natomiast nadzór nad testami polega na podejmowaniu działań, które są niezbędne do osiągnięcia celów wyznaczonych w planie testów. Głównymi czynnościami, które zostaną wykonane to:
   a. sprawdzenie rezultatów testów określonych kryteriów pokrycia
   b. oszacowanie poziomu jakości systemu na podstawie rezultatów testów

    c. ustalenie, czy są konieczne dalsze testy
3. **Analiza testów:** analiza testów służy do zdefiniowania testowalnych cech oraz związanych z nimi warunków testowych:
    a. Analizy podstawy testów na podstawie wymagań biznesowych, funkcjonalnych i systemowych, przypadków użycia, raportów analizy ryzyka, implementacji aplikacji.
    b. dokonywanie oceny testowalności podstawy testów i elementów testowych w celu zidentyfikowania często występujących typów defektów, które mogą powodować problemy z testowalnością.
    c. identyfikowanie cech i zbiorów cech, które mają zostać przetestowane
    d. definiowanie warunków testowych w odniesieniu do poszczególnych cech oraz określenie ich priorytetów na podstawie analizy podstawy testów — z uwzględnieniem parametrów funkcjonalnych, niefunkcjonalnych i strukturalnych, innych czynników biznesowych i technicznych oraz poziomów ryzyka
4. **Projektowanie testów:** Projektowanie testów służy do ustalenia jak należy przetestować dany system, gdzie głównymi czynnościami, które zostaną wykonane to:
    a. projektowanie przypadków testowych i zbiorów przypadków testowych oraz określenie ich priorytetów
    b. identyfikowanie danych testowych niezbędnych do obsługi warunków testowych i przypadków testowych
    c. projektowanie środowiska testowego oraz zidentyfikowanie wszelkich niezbędnych narzędzi i elementów infrastruktury
    d. tworzenie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi i procedurami testowymi
5. **Implementacja testów:** implementacja testów polega na napisaniu testów według wcześniej stworzonego planu oraz sprawdzeniu czy wszystko jest gotowe do uruchomienia. Głównymi czynnościami, które zostaną wykonane to:
    a. opracowanie procedur testowych i określenie ich priorytetów oraz potencjalnie, utworzenie skryptów testów automatycznych
    b. utworzenie zestawów testowych oraz skryptów testów automatycznych
    c. uporządkowanie zestawów testowych w harmonogram wykonywania testów w sposób zapewniający efektywny przebieg całego procesu
    d. zbudowanie środowiska testowego oraz sprawdzenie, czy zostało ono poprawnie skonfigurowane
    e. przygotowanie danych testowych i sprawdzenie, czy zostały poprawnie załadowane do środowiska testowego
    f. zweryfikowanie i zaktualizowanie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi, procedurami testowymi i zestawami testowymi
6. **Wykonywanie testów:** Uruchamiamy zestawy testowe, zgodnie z harmonogramem wykonania. Główne czynności przeprowadzane w ramach wykonywania testów to:
    a. zarejestrowanie danych identyfikacyjnych i wersji elementów testowych bądź przedmiotu testów, narzędzi testowych i testaliów

    b. wykonywanie testów ręcznie lub przy użyciu narzędzi do wykonywania testów
    c. porównanie rzeczywistych wyników testów z oczekiwanymi
    d. przeanalizowanie anomalii w celu ustalenia ich prawdopodobnych przyczyn
    e. raportowanie defektów oparte na obserwowanych awariach
    f. zarejestrowanie wyniku wykonania testów
    g. powtórzenie czynności testowych w wyniku działań podjętych w związku z wystąpieniem anomalii albo w ramach zaplanowanego testowania
    h. zweryfikowanie i zaktualizowanie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi, procedurami testowymi i wynikami testów
7. **Ukończenie testów:** Zbieramy dane pochodzące z wykonanych czynności testowych w celu skonsolidowania zdobytych doświadczeń, testaliów oraz innych stosownych informacji. Główne czynności przeprowadzane w ramach wykonywania testów to:
    a. sprawdzenie, czy wszystkie raporty o defektach są zamknięte oraz wprowadzenie żądań zmian lub pozycji do rejestru produktu w odniesieniu do wszelkich defektów, które nie zostały rozwiązane do momentu zakończenia wykonywania testów
    b. utworzenie sumarycznego raportu z testów
    c. dla wersji końcowych: zarchiwizowanie środowiska testowego, danych testowych, infrastruktury testowej oraz innych testaliów, do ponownego wykorzystania w przyszłości
    d. przeanalizowanie zdobytych doświadczeń omawianych na wszystkich etapach projektu w celu ustalenia, jakie zmiany będą konieczne w przypadku przyszłych iteracji, wydań i projektów
    e. wykorzystanie zebranych informacji do zwiększenia dojrzałości procesu testowego

# Harmonogram

1. **Kamień milowy**
    - Zadanie: Przygotowanie Planu Testów
    - Termin: 17.10.2022
2. **Kamień milowy**
    - Zadanie: Zaprojektowanie scenariuszy testowych
        i. Warunek wstępny: zidentyfikowanie cech aplikacji, nadających się do przetestowania
    - Termin: 31.10.2022
3. **Kamień milowy**
    - Zadanie: Przygotowanie Przypadków Testowych
    - Termin: 14.11.2022
4. **Kamień milowy**
    - Zadanie:
        i. Implementacja testów

ii. Wykonywanie testów
iii. Monitorowanie i nadzorowanie testów
iv. Ukończenie testów

Efekt finalny przedstawić w formie raportu z wykonanych testów

- Termin: 28.11.2022

# Komunikacja

W tym projekcie zespół testerów składa się głównie z developerów, którzy zajmują się również testowaniem zaimplementowanych przez siebie funkcjonalności. Oprócz tego jest też kilku testerów, którzy nie są developerami.

Informacjami, które będą przekazywane pomiędzy testerami a developerami są:

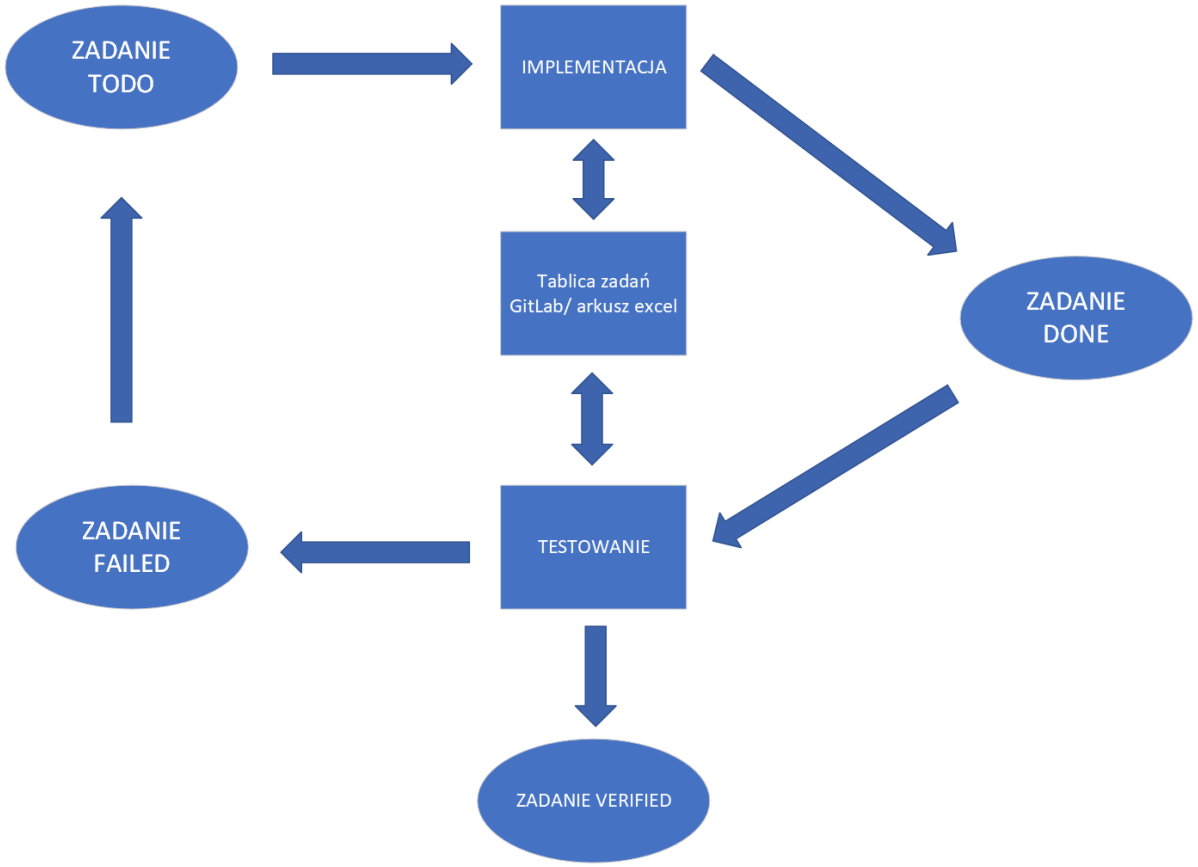
- statusy implementacji funkcjonalności - to czy dana funkcjonalność została w pełni zaimplementowana przez developera
- statusy wykonanych testów przez testera oraz w przypadku niepowodzenia testu, informację dla developera, co należy poprawić

Główny kanał komunikacji, za pomocą którego te informacje będą przekazywane, to **tablica zadań** w repozytorium projektu na portalu GitLab. Drugorzędnym w tej kwestii kanałem komunikacji jest **arkusz excel** z wypisanymi funkcjonalnościami i przyporządkowanymi do nich zespołami. Jednakże ten kanał komunikacji dotyczy bardziej developerów, gdyż jego głównym zadaniem jest przyporządkowanie funkcjonalności do zespołu. Developerzy po zakończeniu pracy nad zadaniem powinni zmienić jego status na ***DONE***.

Jest to znak dla testerów, że zadanie jest gotowe i można przystąpić do testowania. Jeżeli testy wykażą nieprawidłowości, status zadania powinien zostać zmieniony na ***FAILED***, a w opisie powinien znaleźć się raport z testu oraz opis sposobu odtworzenia błędu. Takie zadanie trafia z powrotem do programisty, a po naprawieniu błędu do testera. Jeżeli tester nie wykryje żadnych nieprawidłowości to zadanie otrzymuje status ***VERIFIED***.

Wykorzystywanie tablicy zadań w repozytorium projektu na portalu GitLab oraz arkusza excel pozwala na monitorowanie postępu prac. W łatwy sposób każdy członek zespołu może zweryfikować postępy, a tym samym zaobserwować ile zadań zostało zweryfikowanych, a w ilu zadaniach wykryto nieprawidłowości.

Dodatkowym kanałem komunikacji dla całej grupy projektowej, pozwalający w szybki i sprawny sposób poinformować testera czy developera o postępie procesu wykonywania czy weryfikowania zadania jest specjalnie stworzona konwersacja grupowa w aplikacji Messenger. Wymienione wyżej kanały komunikacji są jednak priorytetowe. To one są jasną i przejrzystą formą weryfikacji postępu testowania, a także przechowują informacje o testach w łatwo dostępny sposób.


ZADANIE
TODO
IMPLEMENTACJA
Tablica zadań
GitLab/ arkusz excel
ZADANIE
DONE
ZADANIE
FAILED
TESTOWANIE
ZADANIE VERIFIED