Skład zespołu:

* Ratajczak Jędrzej 246991
* Michał Kaja, 246763
* Daria Hornik 246700
* Chrząstek Maciej 247004

System wspierający zarządzanie testami funkcjonalnymi

* Rejestracja błędów (utworzenie + zmiana stanu)
* Raportowanie i analityka błędów -> wyciagniecie wiedzy z rejestracji błędów, przetworzenie błędów i ich wizualizacja albo wykonaniem testów przez testerów (wysiłek związany z procesem testowania - po 3h znaleziono x błędów)
* Planowanie procesu testowania - procedury testowe, przypadki testowe, ludzie
* Wykonanie testów -> rejestracja błędów, śladowanie błędów do testów (powiązane z rejestracją), powiązanie między testami a błędami, po kolei procedury odhaczane checkboxem, w którym miejscu procedury testowej wyszedł ewentualny błąd, pobranie kroków testowych i stworzenie z nich checkboxów
* Zarządzanie wymaganiami -> powiązana z planowaniem testów (plan testu na podstawie wymagań)

Maciej Chrząstek: Apka mobilna: zgłaszanie błędów || wykonywanie testów; wyświetlanie błędów, przypadków, procedur, raportów

Notatki: użytkownik może być w tylko w jednym zespole, w programie są predefiniowane role

**Dopisać zakresy!**

Temat 1: Warstwa prezentacji aplikacji webowej modułu planowania testów funkcjonalnych

Aplikacja webowa do planowania testów funkcjonalnych

-> Michał Kaja:

* CRUD planu testów -> Przy dodawaniu na pewno będzie cel (w tym pokrycie funkcjonalności), dodawanie na zasadzie wybierania lub tworzenia nowych przypadków testowych, użytkownicy potrzebni do przeprowadzenia testu
* CRUD przypadków testowych (przypisanie do planu testów)
* Rejestracja (Create) przypadków testowych -> Id, Nazwa przypadków testowych, Powiązane procedury, Warunki wstępne, Dane testowe, Rezultat
* -> Dane testowe -> Pole edycyjne do tworzenia tabel (wielowymiarowe tabele), szablon do wyboru, z którego można stworzyć dane testowe
* Przypisywanie procedur testowych do przypadków testowych -> id, nazwa, cel procedury, zależny przypadek, kroki (co trzeba zrobić + punkt kontrolny) (? samodzielny byt)

Plan testu - przypadki i testerzy

Przypadek - procedury

Temat 2: Warstwa prezentacji aplikacji webowej modułu rejestracji wykonania i raportowania testów funkcjonalnych

Aplikacja webowa do rejestracji wykonania i raportowania testów funkcjonalnych

-> Jędrzej Ratajczak:

* Rejestrowanie nowego błędu (z od razu wybranym stanem początkowym i datą zgłoszenia, wpisuje: nazwę, deadline, priorytet/wpływ, opis, wybór przypadku testowego, wybór ludzi) - z możliwością edycji, które pola chcemy mieć w błędzie
* Usuwanie konkretnego błędu (z modalem do potwierdzenia)
* Edycja konkretnego błędu (z użyciem formularza od rejestracji, z wypełnionymi już polami danymi tego błędu)
* Zmiana stanu błędu (lista rozwijana stanów, do których możemy przejść) - z momentem przejścia na ostatni stan wypełnia się data zakończenia
* Przypisanie recenzenta/recenzentów (z ustawieniem progu zaakceptowania)
* Wyświetlanie konkretnego błędu
* Raportowanie dla całego projektu, biorące pod uwagę wszystkie testy i błędy - jakie raporty? - przeprowadzic ankiete jakie raporty są potrzebne (top 5)
* W procedurze można kliknąć wykonywanie, który przenosi do checklisty z wykonywaniem, przy każdym kroku można zgłosić błąd - tworzenie powiązanego błędu, po wykonanej procedurze użytkownik może oznaczyć procedurę dla siebie jako wykonaną

Temat 3: Warstwa prezentacji aplikacji mobilnej modułu rejestracji wykonania i raportowania testów funkcjonalnych

Aplikacja mobilna do rejestracji wykonania i raportowania testów funkcjonalnych

-> Maciej Chrząstek:

* Rejestrowanie nowego błędu (z od razu wybranym stanem początkowym i datą zgłoszenia, wpisuje: nazwę, deadline, priorytet/wpływ, opis, wybór przypadku testowego, wybór ludzi) - z możliwością edycji, które pola chcemy mieć w błędzie
* Usuwanie konkretnego błędu (z modalem do potwierdzenia)
* Edycja konkretnego błędu (z użyciem formularza od rejestracji, z wypełnionymi już polami danymi tego błędu)
* Zmiana stanu błędu (lista rozwijana stanów, do których możemy przejść) - z momentem przejścia na ostatni stan wypełnia się data zakończenia
* Przypisanie recenzenta/recenzentów (z ustawieniem progu zaakceptowania)
* Wyświetlanie konkretnego błędu
* Raportowanie dla całego projektu, biorące pod uwagę wszystkie testy i błędy - jakie raporty?
* W procedurze można kliknąć wykonywanie, który przenosi do checklisty z wykonywaniem, przy każdym kroku można zgłosić błąd - tworzenie powiązanego błędu, po wykonanej procedurze użytkownik może oznaczyć procedurę dla siebie jako wykonaną

Temat 4: Warstwa logiki biznesowej aplikacji wspierającej proces zarządzania testami funkcjonalnymi

Warstwa biznesowa (i baza danych) aplikacji wspierającej proces zarządzania testami funkcjonalnymi

-> Daria Hornik:

* Zaplanowanie bazy danych
* Logowanie, rejestracja (jwt)
* Obsługa CRUD’u dla błędów
* Wystawienie API dla danych potrzebnych do stworzenia raportów (?)
* Obsługa CRUD’u dla procesu testowania: procedur testowych, przypadków testowych
* Wykonanie testów połączone z rejestracją błędów

Zarejestrowany błąd:

**- kod (typu <nazwaprojektu><id> / DEF\_<numer>)**

**- nazwa**

**- status (in progress, oczekuje na akceptacje, rozwiązany -> maszyna stanów - cykl życia błędów, kontrola z jakiego stanu na jaki może przechodzić) - dopisać gotowe statusy**

**- wpływ błędu na projekt (priorytet)**

- opis (detale, po rozwinięciu)

**- deadline**

- data zgłoszenia

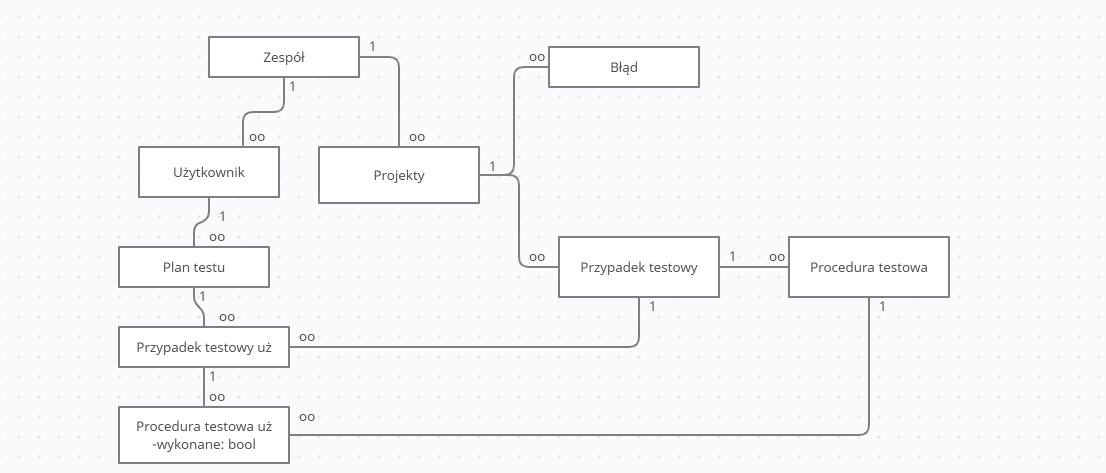
- data zakończenia (rozwiązania buga)

- czas rozwiązywania (wpisywany z palca, widget z wyznaczaniem czasu)

- potwierdzenia rozwiązania od wyznaczonych ludzi z teamu

- powiązany test

**Baza danych:**



Wybrać najlepsze cechy z kilku narzędzi:

<https://theqalead.com/tools/best-test-management-tools/>

1. PractiTest (<https://www.practitest.com/>)

- integracja z Jira itp., dostarcza api do integracji

- dashboard’y i raporty z danymi (wykresy kołowe, słupkowe, tabele), grafy z progresem rozwiązywania problemów

- pozwala używać ponownie istniejące już testy

- pozwala dostosować każdy widok (kafelki), tworzyć widoku, zmieniać uprawnienia do wglądu do nich

- dostęp tylko do zasobów w chmurze

- automatyczne backupy

- trochę przestarzały design

- $39/użytkownika/miesiąc

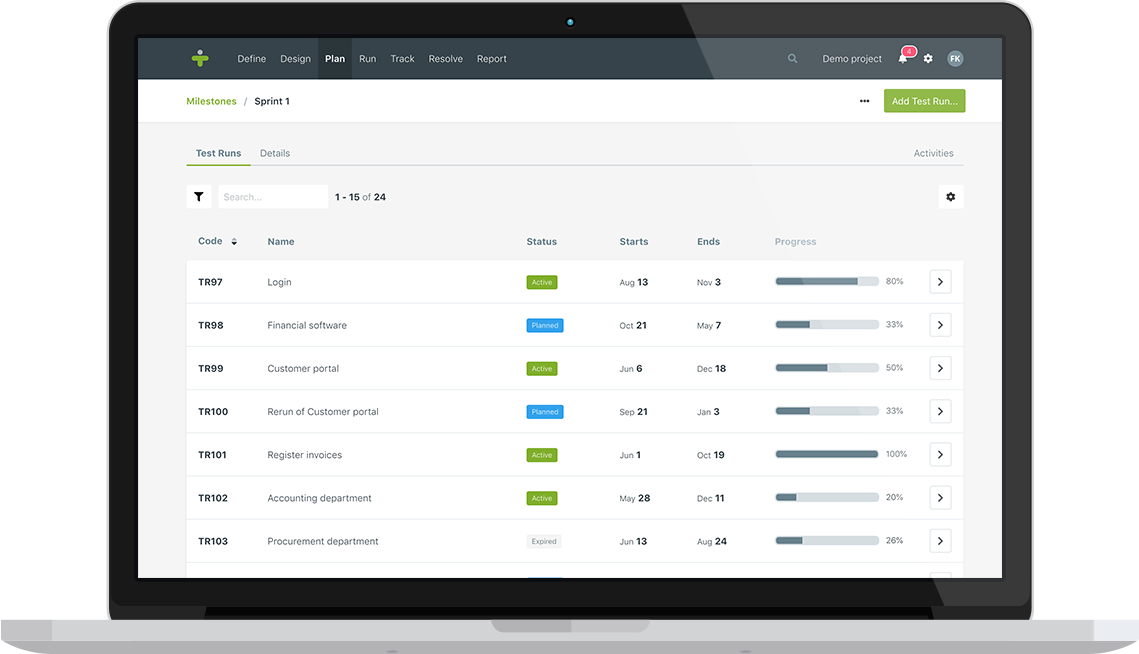
2. TestMonitor (<https://www.testmonitor.com/>) - możemy wzorować się na tym

- zarządzanie błędami, wymaganiami i ryzykiem/wpływem na projekt z nimi związanymi

- śledzenie wyników i raportowanie z różnymi filtrami i wizualizacjami

- całkiem dobry i prosty design, intuicyjny interfejs

- $19/użytkownika/miesiąc



3. Kualitee

- Integracja aplikacji mobilnej i webowej.

- W aplikacji mobilnej:

\*zarządzanie bugami - przeglądanie listy zgłoszonych, edytowanie, dodawanie

\*Dostęp do list przypadków testowych, scenariuszy i bugów

\*raportowanie przebiegu testów bardzo rozbudowane

\*powiadomienia o postępach

- Zarządzanie wymaganiami i przypisywanie im przypadków testowych jakich wymagają do spełnienia

- Zespół z podziałem na role - zarządzanie z poziomu admina

- Integracja z Jirą

- Ogólnie spoko bo robi wszystko to co chcemy - zarządzanie wykrytymi bugami, testami (przypadkami i scenariuszami), wykonaniem testów i wymaganiami

4. Testpad

-Mi się nie podoba - ma taką podstawową funkcjonalność bardzo i tylko w zasadzie tworzenie przypadków i scenariuszy testowych oraz zgłaszanie bugów w trakcie testowania

- Apka mobilna do przebiegu manualnych testów i zgłaszania błędów

-bezpieczna połączeniowo

5. TestRail

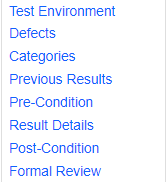
* Kamienie milowe - każda ważna rzecz w projekcie, którą chcemy śledzić (np. wersja testowa), można do niego przypisywać przypadki testowe, dodatkowo posiada pole rozpoczęcia i zakończenia
* Możemy podawać rezultat dla każdego kroku



* Możemy przeglądać historię aktywności
* Możemy wykonywać testy pojedynczo lub wiele na raz
* Przewodnik po aplikacji, jakie są funkcje itp (nie do funkcjonalności , ale fajnie by było mieć coś takiego w naszej aplikacji )
* $34 / użytkownik / miesiąc

6. IBM Rational Quality Manager

* Test Enviroments - zawiera listę różnych środowisk obsługiwanych i testowanych w ramach tego planu testów. Można dodać różne platformy, takie jak przeglądarki, bazy danych, systemy operacyjne i inne elementy.
* Importowanie planów testów
* Wykonywanie testu składa się z następujących sekcji:



7. [Microfocus Silk Central](https://www.microfocus.com/en-us/products/silk-central/overview?r=tmt)

* Pozwala na zunifikowanie wszystkich podejść?/rodzajów testów, żeby zyskać kontrolę nad przejrzystością apki,
* Zunifikowane zarządzanie zarówno testami manualnymi jak i automatycznymi,
* Zarządzanie testami tradycyjnymi jak i w metodologii Agile,
* Aplikacja webowa
* Integracja z wieloma oprogramowaniami
* 
* Płatność po wysłaniu żądania, brak ustalonej z góry stawki
* IMO Interfejs apki z okolic ~2010

8. TestLink

* Strona wydawcy jak typowa strona SCAMowa,
* Pozwala użytkownikom na tworzenie przypadków testowych i planów testowych, zarządzanie użytkownikami, wizualizowanie rezultatów testów i postępów poprzez raportowanie i statystyki.
* Eksportowanie/Importowanie przypadków testowych jest proste i intuicyjne
* Oferuje łatwe metody do przypisywanie przypadków testowych do wielu użytkowników
* Interfejs z czasów wspierania Java Server Faces
* Open Source,
* Darmowe