

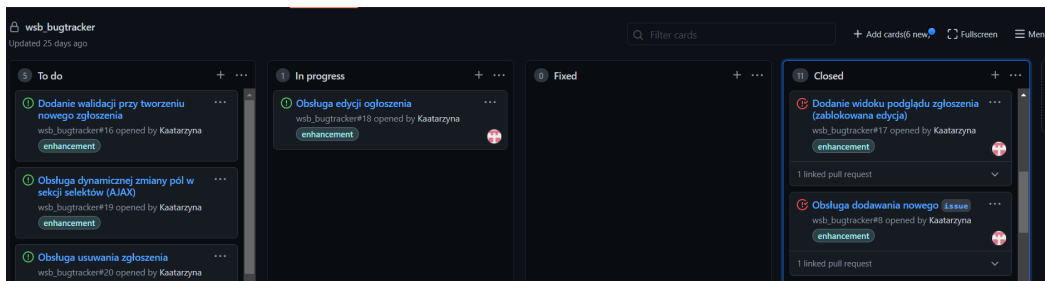
PROJEKT KOŃCOWY

Celem projektu jest stworzenie bugtrackera, czyli aplikacji, która pozwoli zgłaszać i zarządzać zadaniami oraz błędami napotkanymi w oprogramowaniu (podstawowe funkcjonalności będą podobne do tych oferowanych przez YouTrack czy sekcję `Issues` na GitHubie).

Przykładowe widoki w instrukcji należy potraktować jako inspirację - pochodzą one z różnych systemów (m. in. [YouTrack](#)) i mają na celu pomóc wyobrazić sobie, jak mniej więcej powinna wyglądać aplikacja. Wasza aplikacja nie musi być odwzorowaniem screenów - jeśli ktoś ma inny pomysł na rozmieszczenie elementów w interfejsie, nic nie stoi na przeszkodzie.

Wymagania podstawowe

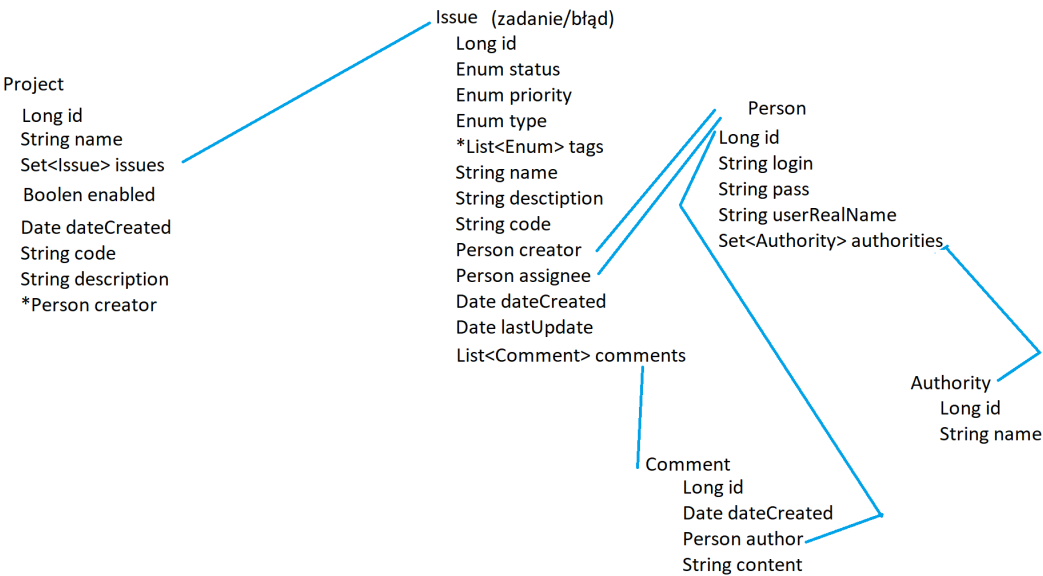
1. Kod tworzonej aplikacji musi znaleźć się w repozytorium na GitHubie.
2. Na GitHubie powinien także znaleźć się projekt, powiązany z repozytorium. Należy przygotować tablicę kanbanową, na której będzie można śledzić postępy prac:



3. Praca nad projektem powinna odbywać się w sposób zorganizowany - przed rozpoczęciem pracy należy zapisać odpowiednie zadanie, stworzyć branch roboczy, a po zakończeniu pracy domergować zmiany do głównego brancha.
4. Aplikacja powinna korzystać z relacyjnej bazy danych (Postgres).

Struktura danych

Przykładową strukturę danych (opracowaną już w ramach poprzednich zajęć) przedstawiono poniżej - oczywiście nie musi być to 1:1, strukturę można sobie modyfikować i rozbudowywać zależnie od własnych potrzeb i pomysłów, poniższy schemat można potraktować jako punkt wyjścia:

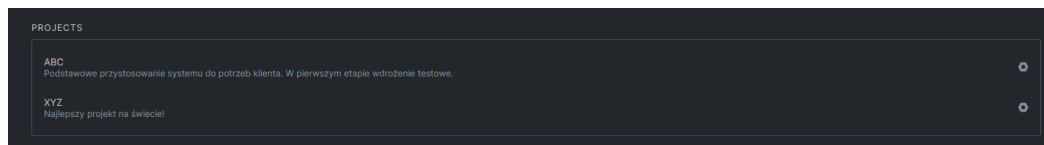


Wymagania funkcjonalne

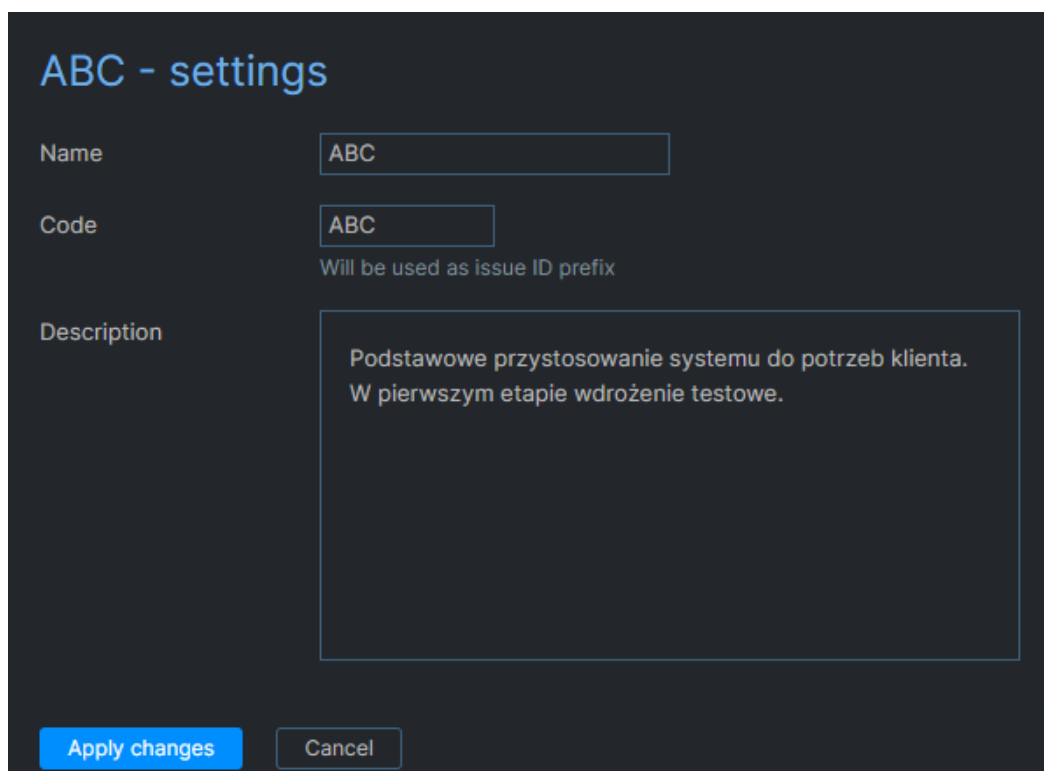
1. Projekty - wyświetlenie listy, dodawanie/edycja/usuwanie

Bugtracker powinien umożliwiać zapisywanie zadań i błędów do różnych projektów. Należy przygotować kod obsługujący podstawowe operacje CRUDowe na obiektach reprezentujących projekt.

Przykładowy widok listy:



Przykładowy widok formularza:

A screenshot of a web application showing a form for editing project settings. The title 'ABC - settings' is at the top. Below it, there are three input fields: 'Name' with the value 'ABC', 'Code' with the value 'ABC', and 'Description' with the value 'Podstawowe przystosowanie systemu do potrzeb klienta. W pierwszym etapie wdrożenie testowe.' The 'Code' field has a note below it: 'Will be used as issue ID prefix'. At the bottom, there are two buttons: 'Apply changes' and 'Cancel'.

2. Zgłoszenia - dodawanie/edycja/usuwanie

Zgłoszeniami są zadania lub błędy zapisane do wybranego projektu. Zgłoszenie powinno mieć:

- przypisany projekt, którego dotyczy
- stan (np. Otwarte, W trakcie, Zrobione, ...)
- typ (np. Zadanie, Błąd, ...)
- wykonawcę

Przykładowy widok szczegółów zgłoszenia:

WSB Bugtracker Zgłoszenia Projekty Użytkownicy Moje konto Nowe zgłoszenie

ABC-1 Nie ładuje się lista zgłoszeń

Po wejściu na listę zgłoszeń w logach aplikacji pojawia się wyjątek NullPointerException

Typ

Błąd

Priorytet

Krytyczne

Wykonawca

Nieprzydzielone

Status

Do zrobienia

3. Lista zgłoszeń

W aplikacji powinna być możliwość wyświetlenia listy zgłoszeń. Lista powinna umożliwiać wyfiltrowanie zgłoszeń co najmniej po 3 polach:

- po projekcie
- po stanie
- po wykonawcy

Przykładowy widok listy zgłoszeń:

WSB Bugtracker Zgłoszenia Projekty Użytkownicy Moje konto Nowe zgłoszenie

Projekt Status Wykonawca Szukaj Wyczyść

ABC-1 Nie ładuje się lista zgłoszeń

Po wejściu na listę zgłoszeń w logach aplikacji pojawia się wyjątek NullPointerException

Błąd Krytyczne Do zrobienia

XYZ-1 Aplikacja działa wolno

Wykonawca: Jan Kowalski

Czas oczekiwania na zapisanie zmian w formularzu transportu potrafi trwać nawet do 4 minut.

Wydajność Ważne W trakcie

XYZ-2 Brak tłumaczenia dla nagłówka na liście transportów

Wykonawca: Jan Kowalski

Pomimo ustawionego w przeglądarce języka polskiego, nagłówek listy transportów wyświetla się w języku angielskim

Użyteczność Drobne Zrobione

4. Walidacje

Jeśli w którymś formularzu dodawania/edycji jakiegoś obiektu pewne pola będą obowiązkowe, przed zapisaniem obiekty powinny być walidowane, a w przypadku niespełnionych warunków koniecznych do zapisania powinien zostać wyświetlony stosowny komunikat (obiekt wówczas nie powinien zostać zapisany w bazie).

Przykładowy widok:

XYZ

Podaj tytuł zadania

Treść nie może być pusta

Typ

Zadanie

Priorytet

Umiarkowane

Wykonawca

Nieprzydzielone

Status

Do zrobienia

Zapisz

5. Logowanie

Dostęp do funkcjonalności aplikacji powinien być tylko dla uwierzytelnionych użytkowników. Powinien zostać zaimplementowany mechanizm logowania. Niezalogowany użytkownik powinien mieć dostęp jedynie do strony logowania - nie powinien mieć możliwości wyświetlenia żadnych innych danych.

Jeśli chodzi o tworzenie nowych kont - mogą być tworzone przez użytkownika z rolą

`ROLE_MANAGE_USERS` w zakładce "Użytkownicy".

6. Moje konto



Zalogowany użytkownik, oprócz możliwości zarządzania projektami i zgłoszeniami, powinien mieć dostęp do swoich danych - możliwość zmiany loginu i hasła lub innych danych podanych podczas rejestracji.

7. Użytkownicy

Użytkownicy z odpowiednim uprawnieniem (patrz: `Zarządzanie uprawnieniami`) będą mieli dostęp do zakładki `Użytkownicy`. W zakładce tej będzie lista użytkowników systemu. Powinien tam istnieć także przycisk umożliwiający dodanie nowego użytkownika.

Przykładowy widok:

Użytkownicy

Imię i nazwisko	Edytuj
Logintegra	
Support Logintegra	

Po przejściu do szczegółów użytkownika będzie możliwość edycji jego danych, a także przypisania mu uprawnień.

Przykładowy widok:

Edycja użytkownika **super-admin**

Imię i nazwisko:	<input type="text" value="Logintegra"/>
e-mail:	<input type="text"/>
Telefon:	<input type="text"/>
Login:	<input type="text" value="super-admin"/>
Hasło:	<button>Zmień hasło</button>

Uprawnienia	<input checked="" type="checkbox"/> Zarządzanie projektami
	<input type="checkbox"/> Zarządzanie komentarzami
	<input checked="" type="checkbox"/> Dostęp do zakładki "Użytkownicy"
	<input checked="" type="checkbox"/> Zarządzanie użytkownikami

ZapiszUsuńAnuluj

8. i18n

Strona powinna posiadać dwie wersje językowe - polską i angielską (zależnie od ustawionego w przeglądarce języka).

9. Komunikacja mejlowa

Aplikacja powinna wysyłać co najmniej jednego mejla po wybranej akcji - np. mejl do twórcy zgłoszenia wysyłany w momencie gdy zgłoszenie zostanie zamknięte, albo mejl do osoby przypisanej do zgłoszenia wysyłany w momencie przydzielenia jej jakiegoś zadania.

10. Zarządzanie uprawnieniami

Część akcji w aplikacji powinna być zabezpieczona uprawnieniem - to znaczy, że tylko użytkownicy posiadający dane uprawnienie powinni móc wykonać daną akcję. Uprawnienia, które należy zaimplementować:

- `ROLE_MANAGE_PROJECT` - umożliwiającą zarządzanie projektami - tworzenie, edycję i usuwanie obiektów klasy `Project`,
- `ROLE_MANAGE_COMMENTS` - zarządzanie komentarzami; użytkownik z tą rolą będzie mógł edytować i usuwać wszystkie komentarze, użytkownicy bez tej roli będą mogli edytować i usuwać jedynie komentarze, których są autorami,
- `ROLE_USER_TAB` - dostęp do zakładki z danymi użytkowników systemu,
- `ROLE_MANAGE_USERS` - możliwość dodawania, edycji i usuwania użytkowników systemu, a także nadawania im uprawnień.

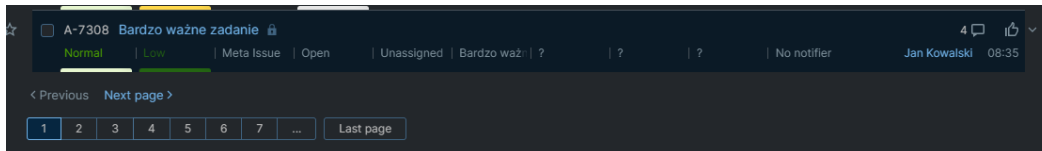
11. Testy jednostkowe

Należy napisać przynajmniej jeden test jednostkowy. Może on dotyczyć np. kontrolera, który renderuje listę zgłoszeń. Można wzorować się na [przykładzie z zajęć](#).

Wymagania dodatkowe (nieobowiązkowe)

1. Dodanie paginacji do list

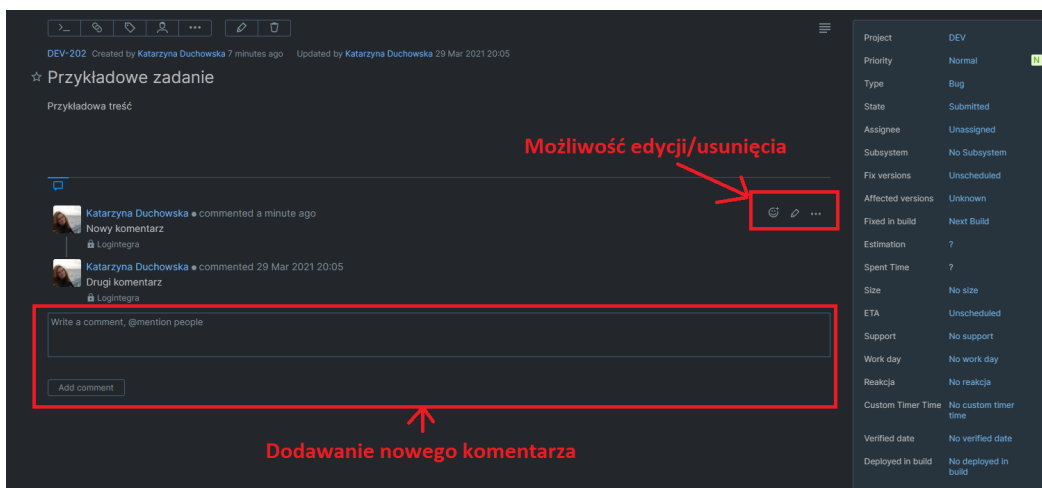
Lista zgłoszeń powinna być stronicowana - jeśli wyfiltrowanych zgłoszeń jest więcej niż 25, powinno się wyświetlać jedynie 25 pierwszych wyników, pozostałe powinny być widoczne na kolejnych stronach (licznik stron, umożliwiający nawigację, powinien się znaleźć pod listą zgłoszeń). Przykładowy widok:



2. Komentarze

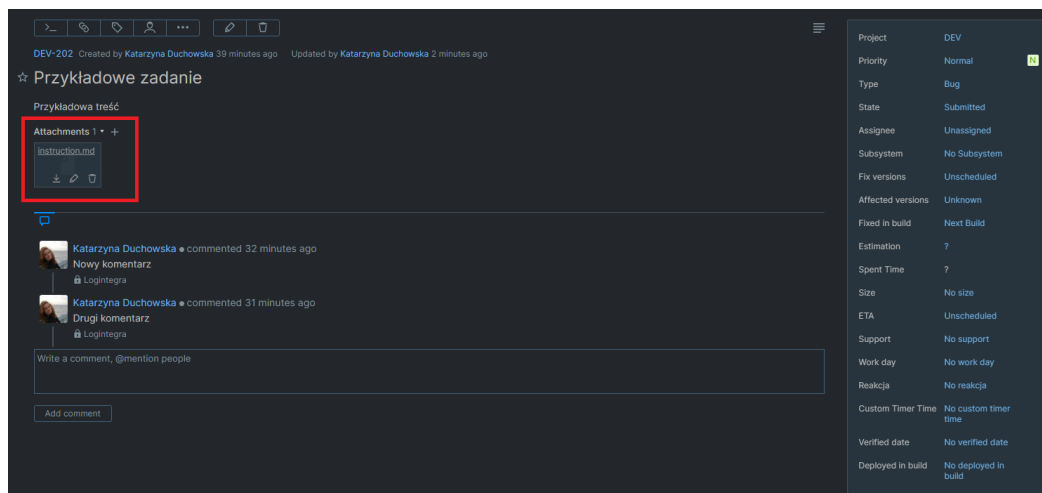
Powinna istnieć możliwość dodania komentarza do zgłoszenia. Komentarze powinny być wyświetlane w szczegółach zgłoszenia, pod jego treścią. Powinna być możliwość dodania, edycji oraz usunięcia komentarza.

Przykładowy widok:



3. Dodawanie załączników do zgłoszeń

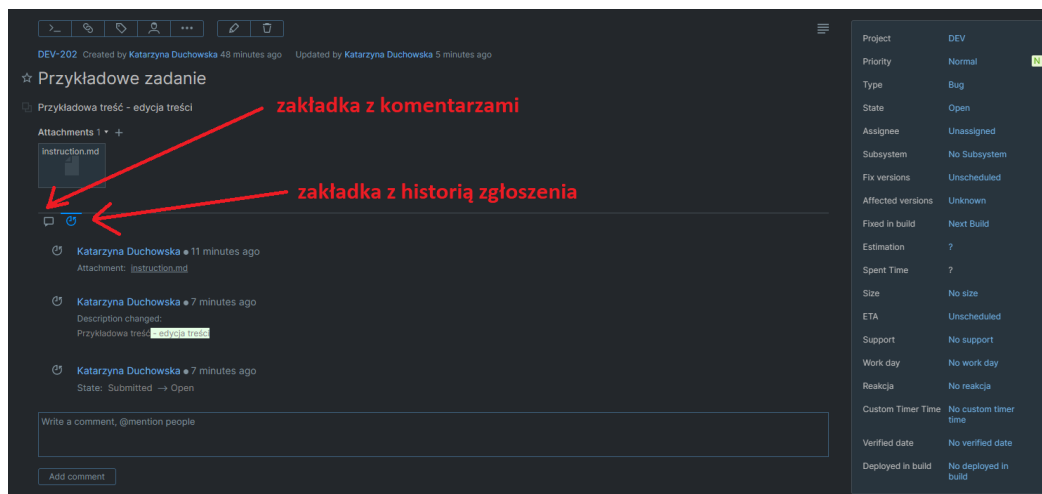
Powinna istnieć możliwość przesłania pliku do zgłoszenia. Załącznik powinno dać się pobrać oraz usunąć. Przykładowy widok:



4. Przechowywanie historii zgłoszenia (audit log)

Aplikacja powinna śledzić zmiany wykonywane na zgłoszeniu i zapisywać je w bazie. Powinien istnieć podgląd historii zgłoszenia - może to być jakiś przycisk w szczegółach zgłoszenia, który przeniesie użytkownika do nowej zakładki lub podstrony, w której będą chronologicznie wyświetlane zmiany w zgłoszeniu.

Przykładowy widok:



5. AJAX

Zmiany w wybranych polach w szczegółach zgłoszenia powinny być zapisywane asynchronicznie - np. po zmianie stanu zgłoszenia (przykładowo z `W trakcie` na `Zrobione`) od razu powinno iść żądanie do serwera, które zapisze nowy stan zgłoszenia, bez konieczności przeładowywania strony lub klikania dodatkowych przycisków.

6. Redis

Wykorzystanie Redisa jako bazy danych przechowującej cache - szczegółowe wymagania i przykłady zostaną omówione na majowym zjeździe na zajęciach z Redisa.