**Dokumentacja projektu zaliczeniowego**

Przedmiot: Inżynieria oprogramowania

Temat: **Bugtracker i TaskManager dla firm zajmujących się tworzeniem gier komputerowych.**

Autorzy: **Jakub Kościołowski, Maciej Lewicki**

Grupa: I1-20B

Kierunek: informatyka

Rok akademicki: 2

Poziom i semestr: I/4

Tryb studiów: niestacjonarne

**1. Spis treści**

2. Odnośniki do innych źródeł

3 .Słownik pojęć

4. Wprowadzenie

4.1 Cel dokumentacji

4.2 Przeznaczenie dokumentacji

4.3 Opis organizacji lub analiza rynku

5. Specyfikacja wymagań

5.1 Charakterystyka ogólna

5.2 Wymagania funkcjonalne

5.3 Wymagania niefunkcjonalne

6. Zarządzanie projektem

6.1 Zasoby ludzkie

6.2 Harmonogram prac

6.3 Etapy/kamienie milowe projektu

7. Zarządzanie ryzykiem

7.1 Lista czynników ryzyka

7.2 Ocena ryzyka

7.3 Plan reakcji na ryzyko

8. Zarządzanie jakością

8.1 Scenariusze i przypadki testowe

9. Projekt techniczny

9.1 Opis architektury systemu

9.2 Technologie implementacji systemu

9.3 Diagramy UML

9.4 Charakterystyka zastosowanych wzorców projektowych

9.5 Projekt bazy danych

9.6 Projekt interfejsu użytkownika

9.7 Procedura wdrożenia

10. Podsumowanie

11.1 Szczegółowe nakłady projektowe członków zespołu

**2. Odnośniki:**

Login:

Hasło:

**3. Słownik pojęć:**

Task- Zadanie do wykonania przez konkretne osoby

Bug – Błąd lub awaria

BugTracker- System do śledzenia bugów

API -  Zbiór reguł ściśle opisujący, w jaki sposób programy lub podprogramy komunikują się ze sobą

UI – Projekt interfejsu użytkownika

**4.Wprowadzenie**

**4.1 Cel dokumentacji**

Celem dokumentacji jest wprowadzenie użytkownika w funkcjonalność systemu, którego zadaniem jest przechowywanie, katalogowanie i zbieranie informacji na temat błędów  w aplikacjach stworzonych przez użytkowników.  Dodatkową funkcją jest obsługa tasków przez kierowników i pracowników.

**4.2. Przeznaczenie dokumentacji**

Dokumentacja jest napisana dla użytkowników aplikacji(menedżerów i pracowników) małych i średnich przedsiębiorstw.

**4.3. Opis organizacji lub analiza rynku:**

System udostępniany jest na masowy rynek, jednakże aktualnie jest przeznaczony do użytku przez maksymalnie 100 firm. Organizacje, dla których wykonana jest aplikacja powinny składać się z maksymalnie 500 użytkowników. Każda z nich musi się skupiać w obrębie tworzenia gier komputerowych. W najbliższej przyszłości planowane jest rozszerzenie liczby użytkowników w taki sposób, by aplikacja była atrakcyjna dla większych firm.

**5. Specyfikacja wymagań:**

**5.1 Charakterystyka ogólna**

**5.1.1. Definicja produktu:**

Aplikacja służy do zgłaszania i przechowywania danych o błędach oraz do przypisywania tasków pracownikom.

**5.1.2. Podstawowe założenia**

Aplikacja jest przeznaczona do użytku przez deweloperów gier komputerowych. System jest przeznaczony dla firm, których problemem jest słaba organizacja pracy oraz potrzeba dokumentowania, zarządzania i przypisywania bugów, by łatwiej można było je rozwiązać.

System służy do zarządzania procesem wykrywania błędów i nieprawidłowości podczas tworzenia gier komputerowych. Pozwala zarządzać wszelkimi nieścisłościami w powstałym projekcie oraz pomaga w szybki sposób przypisać dane taski do określonych pracowników.

**5.1.3. Cel biznesowy:**

Organizacja docelowa, wdrażając system, chce osiągnąć przyspieszenie pracy nad projektem. Dodatkowo szybkość zarządzania bugami pozwala na większą wydajność pracowników, co daje lepsze wyniki finansowe i biznesowe. Możliwość przypisania tasków pracownikom pozwala na polepszenie organizacji pracy. Firma pracująca z pomocą bugtrackera statystycznie dużo lepiej  prosperuje, co sprawia, że jest ona bardziej atrakcyjna dla potencjalnych klientów i inwestorów.

**5.1.4. Użytkownicy**

- Managerowie

- Deweloperzy

- Testerzy

- Graficy

- Klienci

**5.1.5. Korzyści z systemu:**

1. Managerowie:

a - łatwiejsza kontrola nad działaniami firmy i nad samym projektem

b - przejrzysty widok działań i napotkanych problemów

c -  obserwacja dotychczasowych problemów

d - opcja zarządzania zgłoszonymi bugami i nadawania im wagi

e - nadawanie wagi taskom

f  - możliwość przypisywania konktretnych tasków poszczególnym pracownikom

g - obserwacja pracy poszczególnych pracowników

2. Deweloperzy:

a - Dodawanie i usuwanie bugów

b - Możliwość ustawienia statusu tasków i bugów

c - Opcja dodawania notatek do poszczególnych bugów

d. Możliwość zlecenia testu.

e. Możliwość wysyłania zapytań

3. Testerzy:

a - opcja dodawania notatek do poszczególnych bugów

b - Dodawanie bugów

c - Opcja zatwierdzenia testów i dodania ich statusu

4. Graficy:

a - Dodawanie i usuwanie bugów

b - Możliwość ustawienia statusu tasków i bugów

c - opcja dodawania notatek do poszczególnych bugów

5. Klienci

a – Możliwość zgłoszenia bugu.

**5.1.6 Ograniczenia projektowe i wdrożeniowe**:

System powinien być kompatybilny z systemami operacyjnymi:

- Windows 7/8/10,

- Mac OSX 10.12

**5.1.7 Lista wymagań:**

1.Klient zgłasza problem z grą.

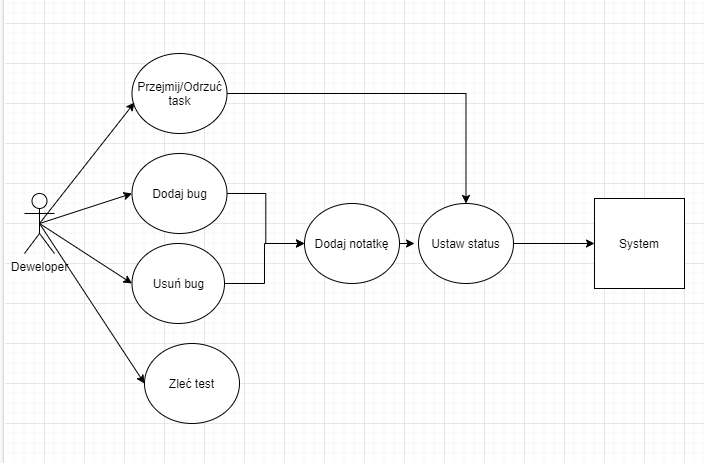
2. Deweloper zgłasza bug.

3. Manager dodaje task.

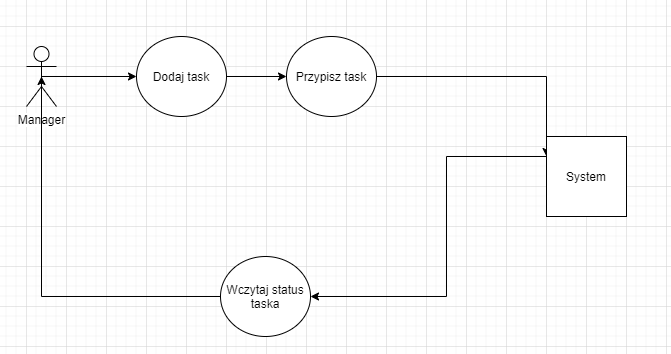
4. Pracownik modyfikuje projekt.

5. Deweloper zleca test.

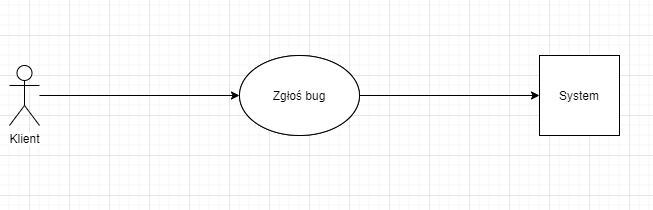
**5.1.8 Diagramy przypadków użycia**:

Deweloper: 

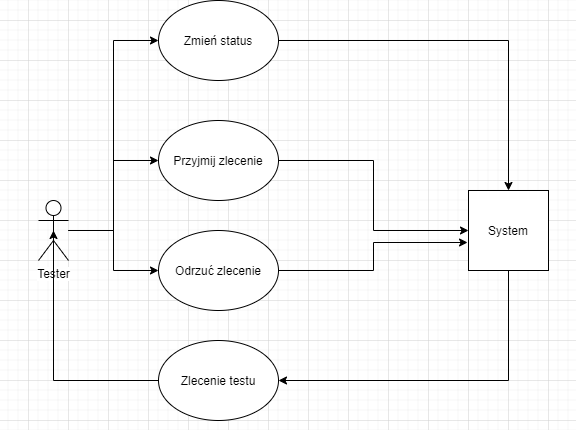
Manager:



Klient:



Tester:



**5.2. Wymagania funkcjonalne**

**5.1.9. Szczegółowy opis wymagań**

**Numer :1**

Nazwa: Klient zgłasza problem

Uzasadnienie biznesowe : patrz punkt 5.1.5(punkt 5.a)

Użytkownicy:

-klient,

-manager

-deweloper

-tester

**Scenariusz 1: Klient zgłasza istniejący błąd**

Warunki początkowe:

Klient widzi, że gra nie działa poprawnie. Zgłasza powstały błąd deweloperom.

Przebieg działań:

1.Zgłoszenie błędu - klient

2.Zatwierdzenie błędu - manager

3.Przypisanie taska – manager

4. Przyjęcie taska – deweloper

5. Naprawienie błędu - deweloper

6. Testowanie gry- tester

Efekty:

Bug powinien zostać wyeliminowany, a gra działać bez zarzutów.

Wymagania niefunkcjonalne:

Klient powinien zauważyć naprawienie błędu.

Częstotliwość: 2

Istotność: 5

**Scenariusz 2: Klient zgłasza błąd, który nie istnieje.**

Warunki początkowe:

Klient widzi, że gra nie działa poprawnie. Zgłasza powstały błąd deweloperom.

Przebieg działań:

1.Zgłoszenie błędu - klient

2.Odrzucenie błędu - manager

3.Wysłanie wiadomości wyjaśniającej odrzucenie błędu - manager

Efekty:

Klient został uświadomiony o braku błędu.

Wymagania niefunkcjonalne:

Klient pomylił błąd przez brak znajomości gry.

Częstotliwość: 5 Istotność: 1

**Numer :2**

Nazwa: Deweloper dodaje bug

Uzasadnienie biznesowe : patrz punkt 5.1.5(punkt 2.a)

Użytkownicy:

-manager

-deweloper

- tester

Warunki początkowe:

Deweloper znajduje bug.

Przebieg działań:

1.Zgłoszenie błędu - deweloper

2.Zatwierdzenie błędu - manager

3.Przypisanie taska – manager

4. Przyjęcie taska – deweloper

5. Naprawienie błędu - deweloper

6. Testowanie gry- tester

7. Zmiana statusu taska na wykonany

8. Przyjęcie wykonanego taska przez managera

Efekty:

Bug powinien zostać wyeliminowany, a gra działać bez zarzutów.

Wymagania niefunkcjonalne:

Błąd powinien być naprawiony w ciągu 48h.

Dane nie powinny zostać utracone.

Częstotliwość: 3

Istotność: 5

**Numer : 3**

Nazwa: Manager dodaje task

Uzasadnienie biznesowe : patrz punkt 5.1.5(punkt 1.f)

Użytkownicy:

-manager

-deweloper

-tester

**Scenariusz 1: Task zostaje wykonany**

Warunki początkowe:

Manager dodaje task.

Przebieg działań:

1. Dodanie taska - manager

2. Przypisanie taska deweloperom, testerom lub grafikom - manager

3. Przyjęcie taska – deweloper/tester

4. Wykonanie taska – deweloper/tester/grafik

5. Testowanie - tester

6. Zmiana statusu taska na wykonany

7. Zgłoszenie o wykonanym tasku – tester

Efekty:

Task został wykonany, a gra zyskała na jakości.

Wymagania niefunkcjonalne:

Task powinien zostać wykonany w ciągu 72h.

Deweloper powinien oznaczyć status swojej pracy.

Częstotliwość: 5

Istotność: 3

**Scenariusz 2: Task nie zostaje zrealizowany**

Warunki początkowe:

Manager dodaje task.

Przebieg działań:

1. Dodanie taska - manager

2. Przypisanie taska deweloperom, testerom lub grafikom - manager

3. Przyjęcie taska – deweloper/tester

4. Task nie zostaje wykonany

6. Zmiana statusu taska na odrzucony

7. Generowanie raportu - manager

Efekty:

Task nie został wykonany. Wymagana głębsza analiza problemu i ponowna próba wykonania.

Wymagania niefunkcjonalne:

Task powinien zostać ponownie rozpatrzony

Częstotliwość: 1

Istotność: 5

**Numer : 4**

Nazwa: Pracownik modyfikuje projekt

Uzasadnienie biznesowe : patrz punkt 5.1.5(punkt 2.e)

Użytkownicy:

-manager

-deweloper

-tester

-grafik

**Scenariusz 1**: Zgoda na podjęcie działań

Warunki początkowe:

Pracownik chce zmodyfikować projekt. Otrzymuje zgodę na podjęcie działań.

 Przebieg działań:

1. Zapytanie do managera o możliwość wprowadzenia zmian- deweloper/grafik

2.Zatwierdzenie zapytania - manager

3.Wprowadzenie zmian – deweloper/grafik

4. Testowanie zmian - tester

5. Prezentacja zmian – deweloper/grafik

6. Zaakceptowanie/odrzucenie zmian – manager

Efekty:

Projekt został zmieniony.

Wymagania niefunkcjonalne:

Aplikacja jest monitorowana i aktualizowana.

Częstotliwość: 5

Istotność: 2

**Scenariusz 2**: Brak zgody na podjęcie działań.

Warunki początkowe:

Pracownik chce zmodyfikować projekt. Nie otrzymuje zgody na podjęcie działań.

Przebieg działań:

1. Zapytanie do managera o możliwość wprowadzenia zmian- deweloper/grafik

2.Odrzucenie zapytania - manager

Efekty

Projekt nie został zmieniony

Częstotliwość: 2

Istotność: 1

**Numer: 5**

Nazwa: Deweloper zleca test.

Uzasadnienie biznesowe: patrz punkt 5.1.5(punkt 2.d)

Użytkownicy:

-deweloper

-tester

-manager

**Scenariusz 1:** Test zakończony rezultatem pozytywnym.

Warunki początkowe:

Deweloper zleca testerom przetestowanie systemu.

Przebieg działań:

1.Deweloper zleca test.

2.Tester przyjmuje zlecenie.

3.Test zakończony rezultatem pozytywnym.

4.Tester przekazuje wynik managerowi.

Efekt:

Pozytywny rezultat.

Częstotliwość: 5

Istotność 3

**Scenariusz 2**: Test zakończony rezultatem negatywnym

Warunki początkowe:

Deweloper zleca testerom przetestowanie systemu.

Przebieg działań:

1.Deweloper zleca test.

2.Tester przyjmuje zlecenie.

3.Test zakończony rezultatem negatywnym.

4.Tester przekazuje raport deweloperowi.

5.Deweloper wprowadza poprawki

Efekt:

Negatywny rezultat testów, Wymagane poprawki.

Częstotliwość:5

Istotność 3

**5.3.Wymagania niefunkcjonalne:**

System powinien być dostępny dla klientów przez minimum 99.9%.

Aplikacja powinna utrzymać 10000 użytkowników.

Dane powinny być zapisywane na serwerach należących do naszej firmy w formie kopii zapasowej. Dodatkowo powinniśmy udostępniać klientom klucz, dzięki któremu mają do nich dostęp w razie ich utraty, bądź zniszczenia .

Aplikacja nie powinna ingerować w działanie innych systemów, jeżeli jednak doszłoby do takiego wypadku, nasza firma ponosi wszelka odpowiedzialność za to i ponosi koszty naprawy.

System jest monitorowany i chroniony przed atakami Denial of Service (DOS).

Blokowanie adresów IP, z których następuję częsta prośba o połączenie.

Bezpieczeństwo fizyczne powinno być realizowane na poziomie uprawnień dostępu do plików.

Inne cechy jakości:

  - niska awaryjność – przeprowadzone testy

- wysoka użyteczność

- wysoka testowalność

- wysoka dostępność – aplikacja dostępna dla deweloperów i klientów przez 99.99% czasu

**6.Zarządzenie projektem**

**6.1. Zasoby ludzkie**

Przy realizacji projektu pracowały 2 grupy projektowe w składzie:

3 deweloperów

2 grafików

1 manager

2 testerów

**6.2 Etapy/kamienie milowe projektu:**

1.Planowanie działań

2. Spotkanie organizacyjne

3.Szkolenia obejmujące wykorzystywane technologie

4.Dokładne omówienie projektu

5. Przypisanie zadań

6. Rozpoczęcie pracy

7. Rozwiązywanie napotkanych problemów

8. Spotkania kontrolne

9. Testowanie rozwiązań

10. Ewentualne poprawki

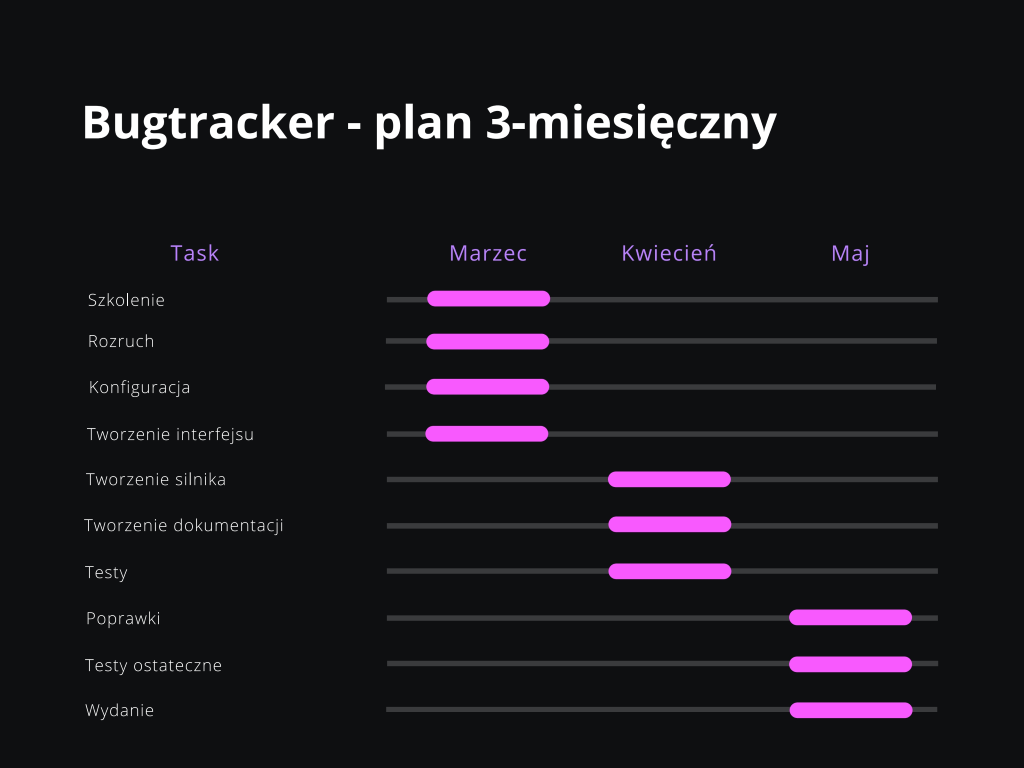
11. Doskonalenie aplikacji

12. Przygotowanie do wydania z ostatecznymi poprawkami

13. Wydanie aplikacji

14. Aktualizacja

**6.3 Harmonogram prac:**



**7. Zarządzanie ryzykiem:**

**7.1 Lista czynników ryzyka i ocena prawdopodobieństwa:**

wymagania - mało prawdopodobne

wydajność - prawdopodobne

jakość – prawdopodobne

niezawodność - prawdopodobne

prawo i regulaminy - prawdopodobne

odbiorca - mało prawdopodobne

finansowanie - mało prawdopodobne

kontrola –  mało prawdopodobne

bezpieczeństwo – mało prawdopodobne

**7.2 Plan reakcji na ryzyko**:

- Częste przeprowadzanie testów

- Kontrola jakości

- Aktualizowanie systemu

- Dostosowywanie się do zmian prawnych

- Zbieranie opinii klientów

- Zawieranie umów

- Monitorowanie aplikacji

- Zabezpieczenie danych

- Zawieranie umów o zachowaniu poufności z pracownikami

**8.Zarządzanie jakością**

**8.1. Scenariusze i przypadki testowe:**

**Scenariusz 1:**

* ID: 1
* Nazwa scenariusza: Testy wymagań biznesowych
* Kategoria: Testy funkcjonalne
* Opis: Sprawdzenie czy aplikacja koncentruje się na wymaganiach biznesowych, które ma spełniać dana aplikacja.
* Tester: tester
* Termin: 03.03.2020r.
* Narzędzia wspomagające: QtCreator
* Środowisko: QtCreator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Założenia | Warunki | Dane wejściowe | Kroki |
| 1 | System pozwala na przechowywanie, katalogowanie i zbieranie informacji na temat błędów  w aplikacjach stworzonych przez użytkowników oraz przypisywanie tasków do poszczególnych pracowników | Test zakończony sukcesem, wszystkie funkcjonalności zwracają prawidłowy wynik. | Dodanie błędów, dodanie tasków do każdego pracownika | Sprawdzenie czy błąd został dodany.  Sprawdzenie czy status został nadany.  Sprawdzenie czy taski zostały dodane |

**Scenariusz 2:**

* ID: 2
* Nazwa scenariusza: Testy UI
* Kategoria: Testy integracyjne
* Opis: Sprawdza czy interfejs graficzny jest napisany zgodnie ze specyfikacją.
* Tester: tester
* Termin: 07.04.2020r.
* Narzędzia wspomagające: QtCreator
* Środowisko: QtCreator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Założenia | Warunki | Dane wejściowe | Kroki |
| 2 | Interfejs graficzny jest wykonany zgodnie ze specyfikacją. | Test powinien być zakończony sukcesem, UI zawiera wszystkie wymagane funkcjonalności. | Dodanie buga, taska, pracownika, statusu, daty | Sprawdzenie czy wyświetlane są bugi, taski, pracownicy, status oraz data.  Sprawdzenie czy wyświetlane są przyciski umożliwiające dodanie tasku, i dodanie buga |

**Scenariusz 3:**

* ID: 3
* Nazwa scenariusza: Testy API
* Kategoria: Testy integracyjne
* Opis: Sprawdza czy wszystkie komponenty programu ze sobą współpracują.
* Tester: tester
* Termin: 12.04.2020r.
* Narzędzia wspomagające: QtCreator
* Środowisko: QtCreator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Założenia | Warunki | Dane wejściowe | Kroki |
| 3 | Wszystkie komponenty programu powinny ze sobą współpracować bez zarzutu | Task powinien być prawidłowo dodany do osoby.  Bug powinien być prawidłowo obsługiwany. | Dodanie buga,  dodanie taska, zmiana statusu, usunięcie buga, usunięcie taska | Dodanie taska.  Zmiana statusu taska.  Usunięcie taska.  Dodanie buga.  Zmiana statusu buga.  Usunięcie buga. |

**Scenariusz 4:**

* ID: 4
* Nazwa scenariusza: Testy bazy danych
* Kategoria: Testy funkcjonalne
* Opis: Uzyskanie określonych wartości z bazy danych, zgodnie z wymaganiami produktu..
* Tester: tester
* Termin: 25.04.2020r.
* Narzędzia wspomagające: QtCreator
* Środowisko: QtCreator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Założenia | Warunki | Dane wejściowe | Kroki |
| 3 | Uzyskanie wartości z bazy danych zgodnych z zapytaniami i wymaganiami produktu. | Baza danych powinna przechowywać bugi, taski, daty, statusy, dane logowania, id błędów i użytkowników, pozwolenia. | Login, hasło, bug, status, task, status | Zalogowanie się.  Dodanie buga i taska wraz z datą i statusem.  Sprawdzenie czy wyświetliła się data.  Usunięcie taska i buga.  Sprawdzenie czy po usunięciu buga i taska znikają one z bazy danych. |

**Scenariusz 5:**

* ID: 5
* Nazwa scenariusza: Testy przypadków użycia
* Kategoria: Testy jednostkowe
* Opis: Sprawdzenie czy poszczególne komponenty działają
* Tester: tester
* Termin: 15.05.2020r.
* Narzędzia wspomagające: QtCreator
* Środowisko: QtCreator

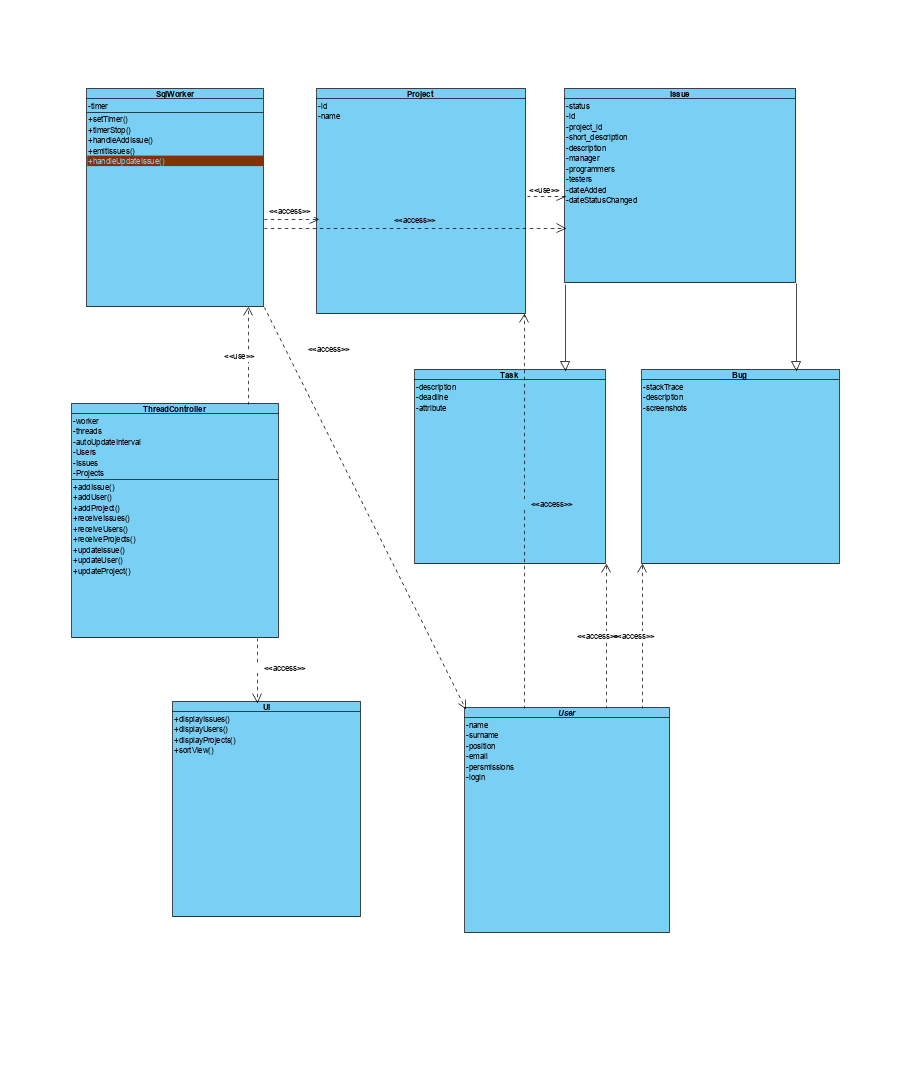
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Założenia | Warunki | Dane wejściowe | Kroki |
| 5 | Wszystkie komponenty aplikacji prawidłowo działają. | Dodanie taska, dodanie buga,, zmiana statusu, przekazanie taska, usunięcie taska, usunięcie buga, logowanie, otrzymanie zgłoszenia działają prawidłowo. | Login, hasło, bug, status, task, status, report a bug, | Zalogowanie się.  Zgłoszenie buga.  Dodanie buga i taska wraz z datą i statusem.  Przekazanie buga i taska.  Usunięcie taska i buga.  Generowanie raportu |

**9.Projekt techniczny**

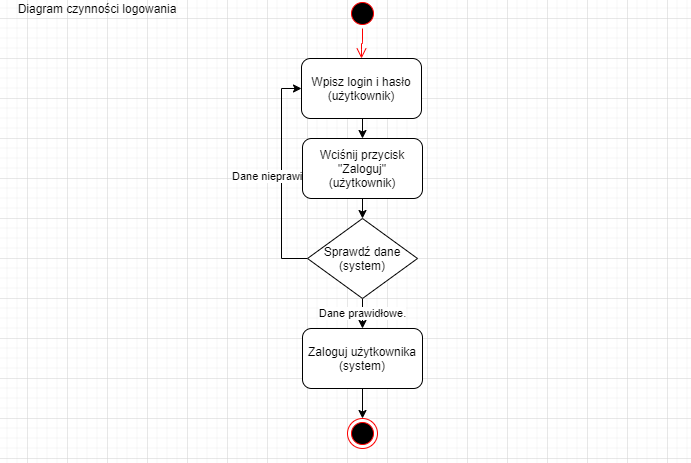
**9.1 Technologie implementacji systemu**:

|  |  |
| --- | --- |
| C++ | Wydajny niskopoziomowy język. |
| SQL Server | Przystępna i szybka baza danych |
| Biblioteka Qt | Pozwala na tworzenie aplikacji graficznych na wiele platform. |
| Qt-Frameless-Window | Tworzenie ładnego okna aplikacji |
| Qt DarkStyle | Ciemny wygląd okien aplikacji |
| hashlibpp | Biblioteka do szyfrowania danych. |

**9.3 Diagramy UML:**

**9.2.1 Diagramy klas:** 

**9.3.1. Diagramy czynności:**



**9.3.2. Diagramy sekwencji**

**Diagram sekwencji dodawania buga:**

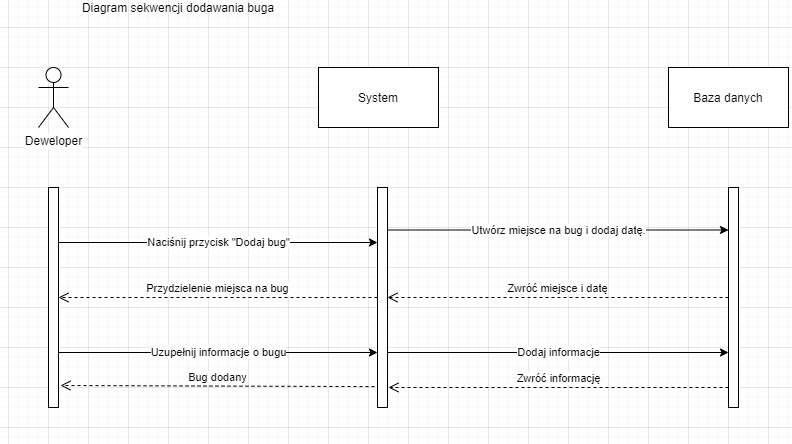
****

Diagram sekwencji usuwania buga:

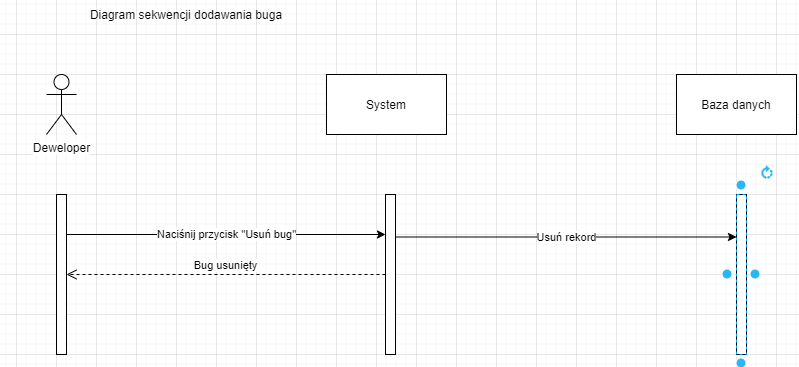


Diagram sekwencji dodania taska:

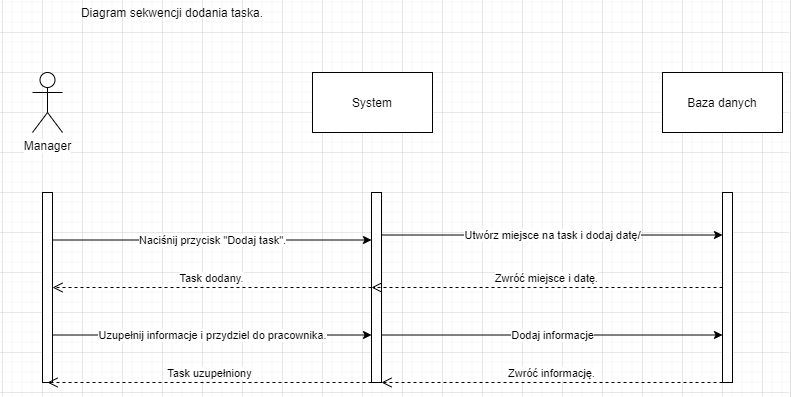


Diagram sekwencji usunięcia taska:

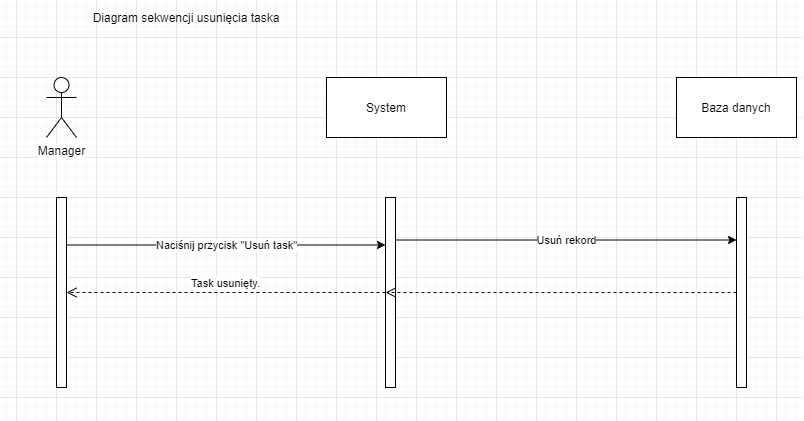
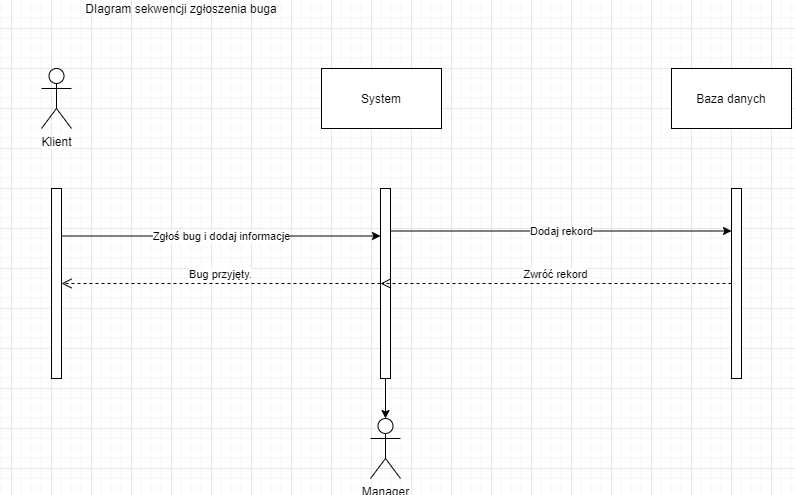
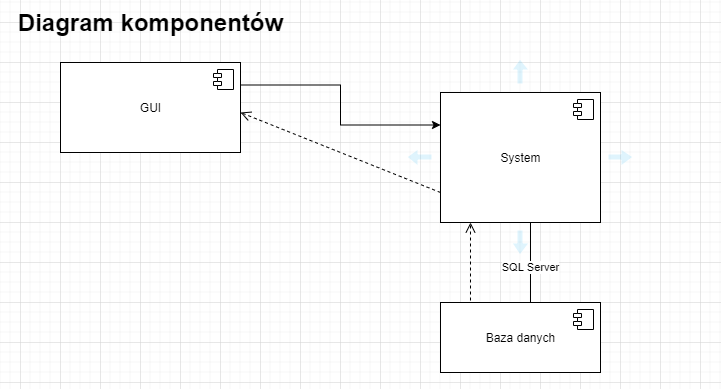
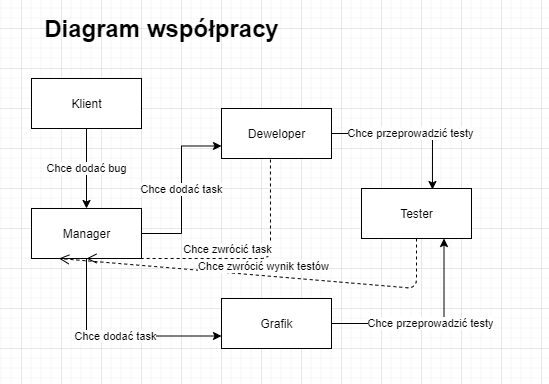


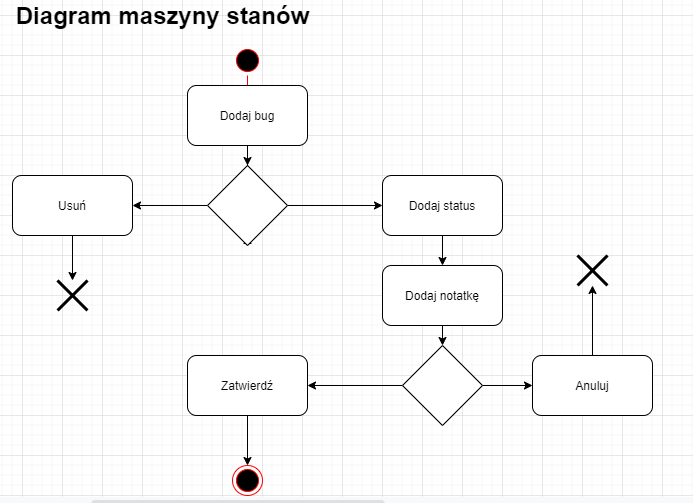
Diagram sekwencji zgłoszenia buga:



**9.3.3.Diagramy komponentów, rozmieszczenia, maszyny startowej:**

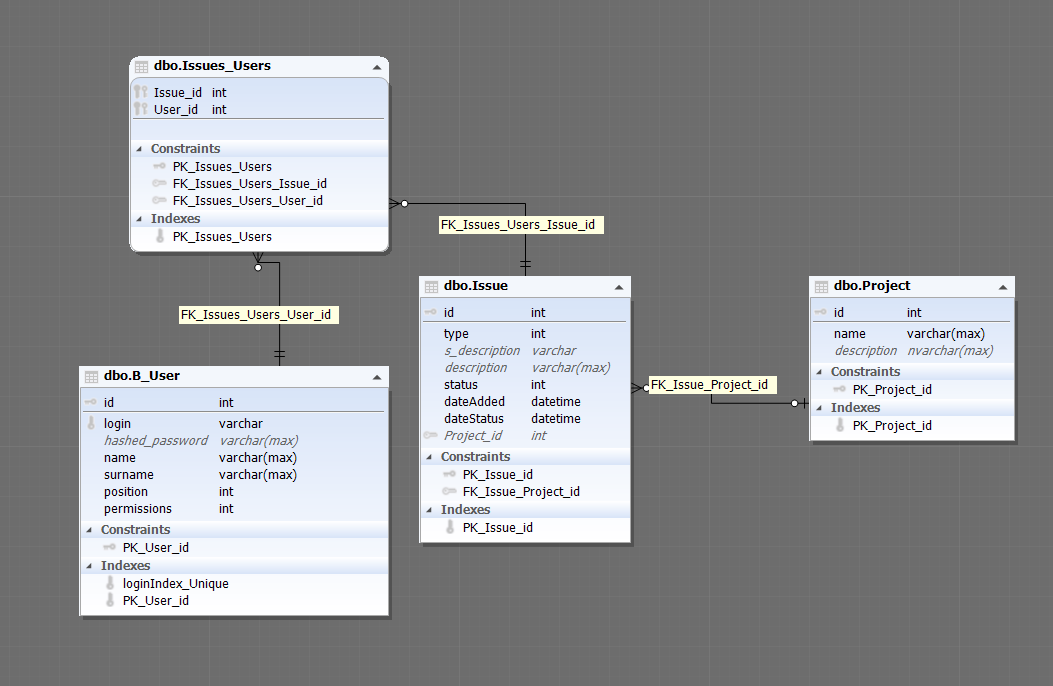




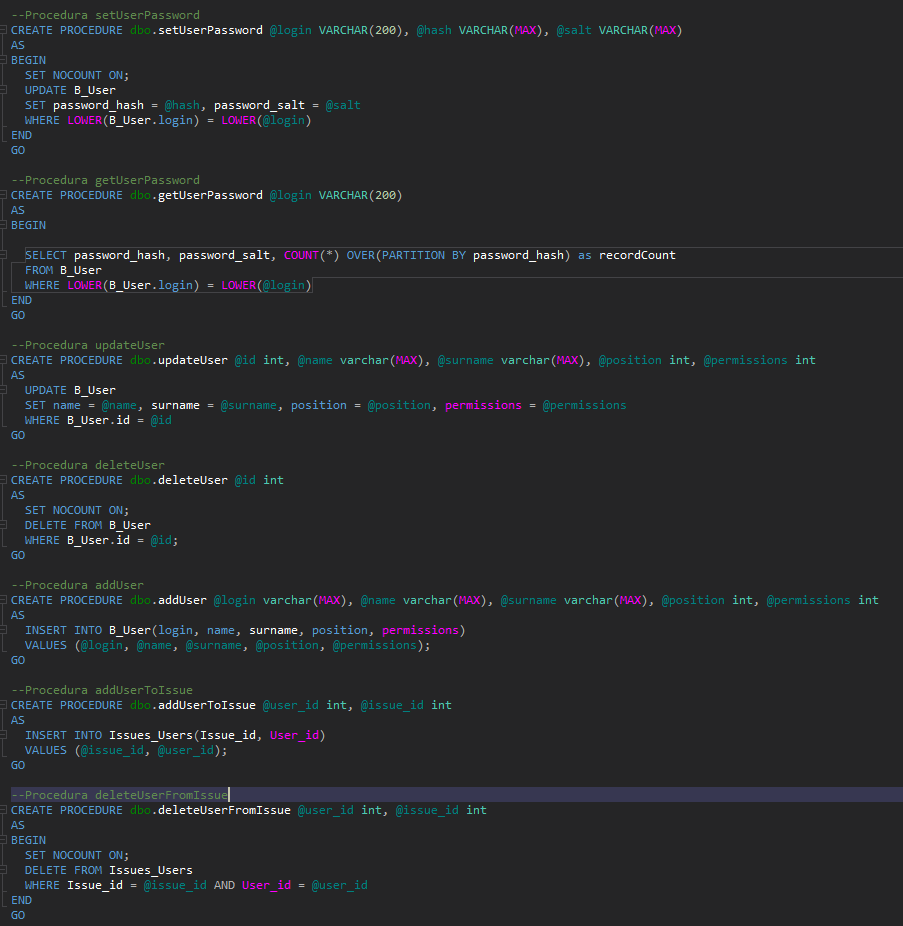


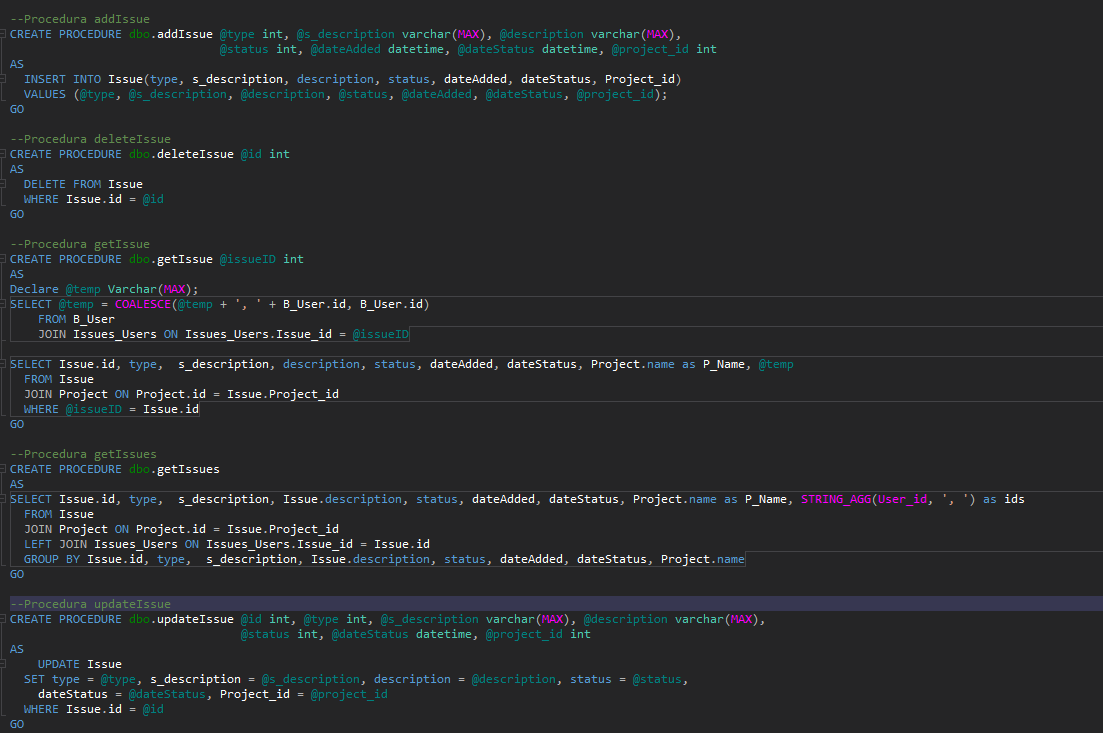
**9.4. Charakterystyka zastosowanych wzorców projektowych:**

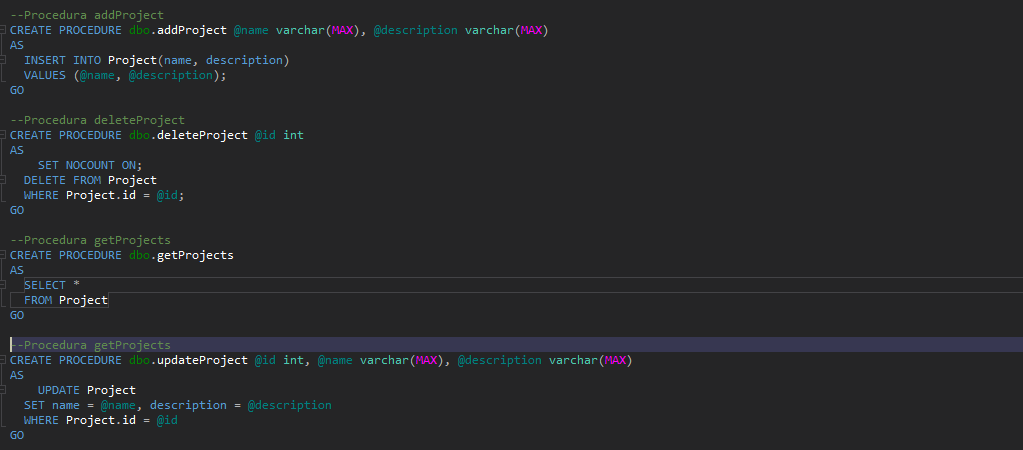
W projekcie nie zostały świadomie wykorzystane wzorce projektowe.

**9.5 .Projekt bazy danych**

**Procedury bazy danych:**

**-User:**

**-Issue:**

**Project:**

**9.6. Projekt szczegółowy tabel:**

Tabela User:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Typ | Klucz | Zasady |
| Id | Int | Tak | Nie może być nullem |
| Login | Varchar | Nie | Nie może być nullem |
| Hashed\_password | Varchar | Nie | Może być nullem. |
| Name | Varchar | Nie | Nie może być nullem |
| Surname | Varchar | Nie | Nie może być nullem |
| Position | Int | Nie | Nie może być nullem |
| Permissions | Int | Nie | Nie może być nullem |
| PK\_User\_ID | Index + ograniczenie |  |  |
| LoginIndex\_Unique | index |  |  |

Tabela Issues\_Users:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Typ | Klucz | Zasady |
| Issue\_id | Int | Tak | Nie może być nullem |
| User\_id | int | Tak | Nie może być nullem |
| PK\_Issues\_Users | index |  |  |
| PK\_Issues\_Users\_Issue\_Id | ograniczenie |  |  |
| PK\_Issues\_Users\_User\_Id | ograniczenie |  |  |

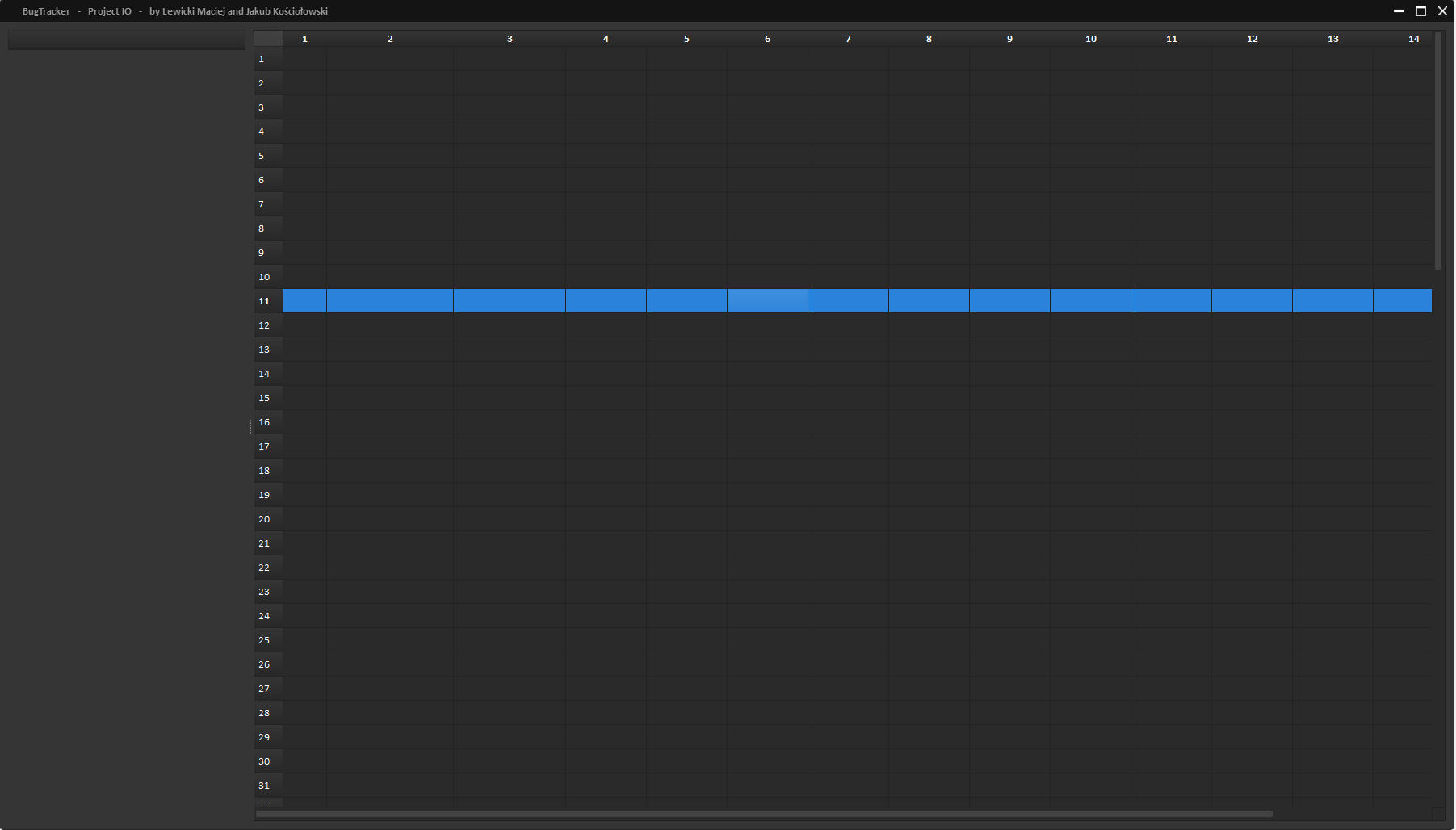
Tabela Issue:

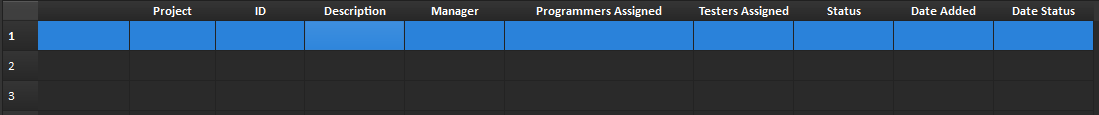
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Typ | Klucz | Zasady |
| ID | Int | Tak | Nie może być nullem |
| Type | Int | Nie | Nie może być nullem |
| S\_description | Varchar | Nie | Może być nullem. |
| Description | Varchar | Nie | Może być nullem. |
| Status | Int | Nie | Nie może być nullem |
| Date\_added | Datetime | Nie | Nie może być nullem |
| Date\_status | Datetime | Nie | Nie może być nullem |
| Project\_id | Int | Tak | Może być nullem. |
| PK\_Issue\_Id | Indeks+ograniczenie |  |  |
| PK\_Issue\_Project\_Id | ograniczenie |  |  |

Tabela Project:

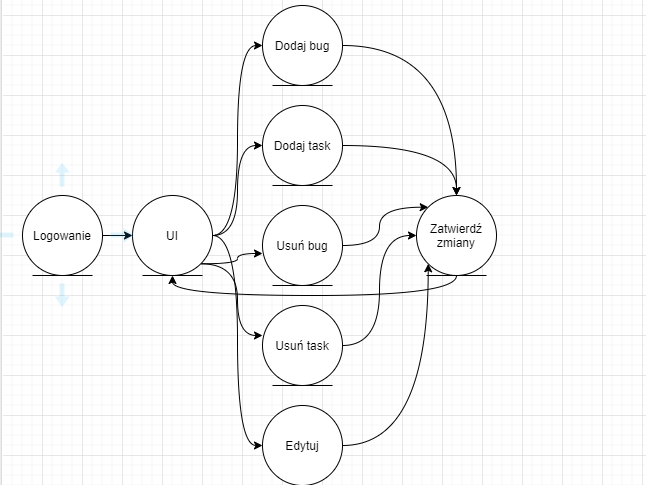
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Typ | Klucz | Zasady |
| Id | Int | Tak | Nie może być nullem |
| Name | Varchar | Tak | Nie może być nullem |
| Description | nvarchar | Nie | Może być nullem. |
| PK\_Project\_id | Indeks+ograniczenie |  |  |

**9.6. Projekt interfejsu użytkownika:**





**9.7. Przejścia między głównymi elementami:**



**9.8. Lista głównych elementów interfejsu:**

1. Okno logowania

2. Zakładka ‘Projects’

3. Zakładka ‘Users’

4. Zakładka ‘Issues’

4. Przycisk ‘Add Item’

5. Przycisk ‘Edit selection’

6. Przycisk ‘Delete selection’

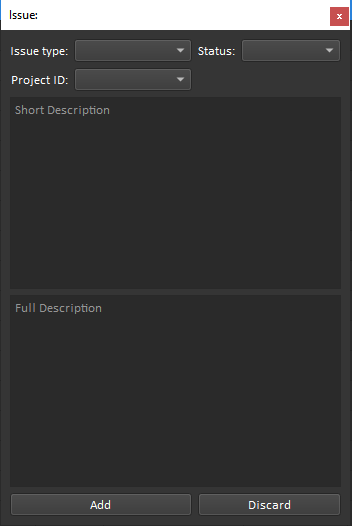
7. Przycisk ‘Add/Remove resources’

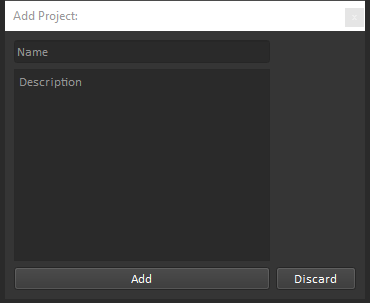
8. Zakładki ID, Project, Short description, Description, Menagers, Programmers assigned, Testers assigned, Status, Date added, Date status

**9.9 Projekty szczegółowe poszczególnych elementów:**

**ID:1**

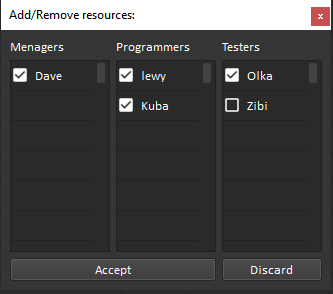
**Nazwa: ‘Add Item’**





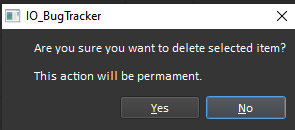
**ID: 2**

**Nazwa ‘Add/remove resources’**



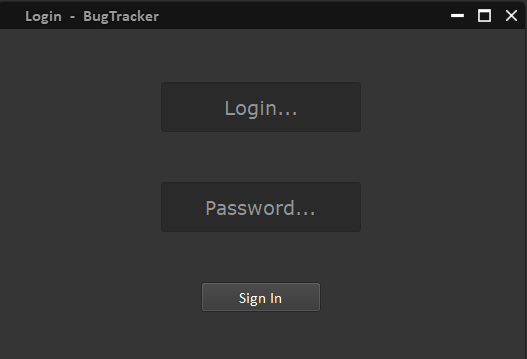
**ID: 3**

**Nazwa: ‘Delete selection’**



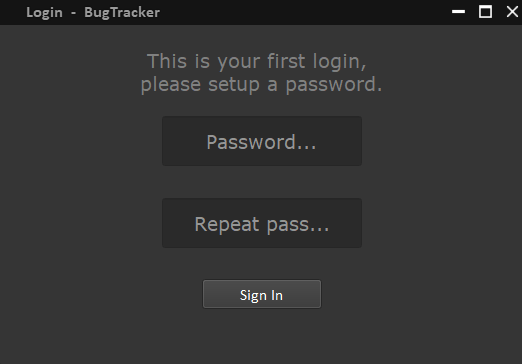
**ID: 4**

**Nazwa: ‘Login’**

****

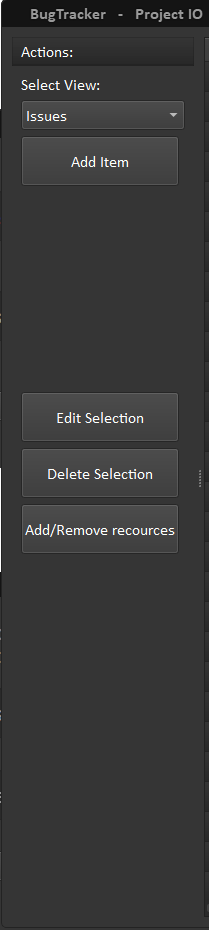
**ID: 5**

**Nazwa: ‘PasswordSetup’**

****

**ID: 5**

**Nazwa: ‘ActionBar'**

****

**9.10. Procedury wdrożenia:**

- Postawienie bazy danych u klienta

-Instalacja systemu u klientów

-Implementacja API w projektach klienta

-Testowanie systemu

-Szkolenie pracowników

**11.Podsumowanie:**

|  |  |  |
| --- | --- | --- |
|  | **Maciej Lewicki** | **Jakub Kościołowski** |
| **Odnośniki do innych źródeł** | 0% | 100% |
| **Słownik pojęć** | 20% | 80% |
| **Wprowadzenie** | 0% | 100% |
| **Specyfikacja wymagań** | 0% | 100% |
| **Zarządzanie projektem** | 0% | 100% |
| **Zarządzanie ryzykiem** | 0% | 100% |
| **Zarządzanie jakością** | 0% | 100% |
| **Projekt techniczny** | 50% | 50% |
| **Projekt bazy danych** | 50% | 50% |
| **Implementacja** | 100% | 0% |
| **Readme** | 100% | 0% |

**11.1 Szczegółowy nakład członków zespołu:**