# daily manager –

# Aplikacja wspomagająca zarządzanie czasem

## Cele

Celem projektu jest stworzenie aplikacji, która jest odpowiedzią na zapotrzebowanie dzisiejszych czasów tj. zarządzanie czasem. Tylko zarządzając odpowiednio czasem, planując zadania na każdy miesiąc, tydzień, dzień można osiągnąć równowagę w życiu, osiągnąć sukces i uzyskać czas wolny od codziennych obowiązków by skupić się na rodzinie lub hobby.

Aplikacja nakierowana jest również na pomoc dla osób starszych, które mają problem z zapamiętywaniem codziennych czynności.

## Zakres projektu

Aplikacja umożliwia:

1. Tworzenie listy zadań, która w czytelny sposób obrazuje zakres aktywności, dając szerokie pole manewru przy dokładnym określaniu priorytetów.
2. Poszczególne zadania można dodawać, edytować oraz usuwać.
3. Istnieje możliwość planowania całego tygodnia oraz wyświetlania najważniejszych przedsięwzięć w danym tygodniu (zależy od podanego priorytetu na etapie wprowadzania zadania do programu).
4. Aplikacja wysyła przypomnienia, dając możliwość rezygnacji z tej opcji. Prowadzone będą statystyki informujące procentowo o zrealizowanych zadaniach, jak również pokazując ich liczbę.

## Zasoby

1. Zespół programistyczny
   * Adrian MALIK
   * Łukasz GUZEK
2. Zasoby technologiczne:
   * Licencja systemu operacyjnego MS Windows 10,
   * Licencja oprogramowania Enterprise Architect
   * Oprogramowanie Java,
   * Licencja na środowisko programistyczne IntelliJ IDEA.
3. Zasoby Informacyjne:
   * Znajomość wzorca projektowego MVC,
   * Umiejętność wykorzystywania Java i JavaFX.

## Budżet

|  |  |
| --- | --- |
| Wydatek | Koszt w zł |
| Godzina pracy programisty | 50 |
| Licencja Enterprise Architect professional | 940 |
| Licencja systemu operacyjnego MS Windows 10 | 260 |
| Licencja na środowisko programistyczne IntelliJ IDEA | 2275 |

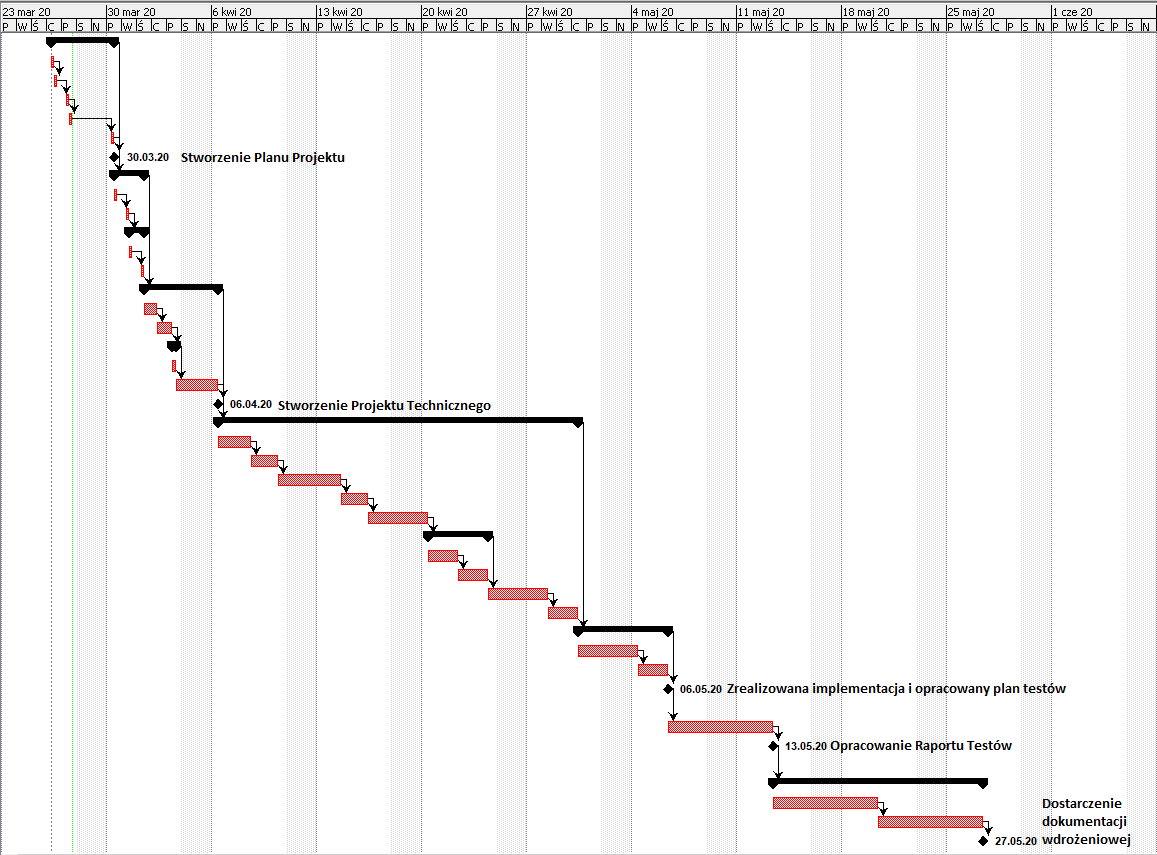
Przewidywana liczba godzin wymagana do ukończenia projektu – 406h

|  |  |
| --- | --- |
| Koszt wszystkich licencji na dwóch pracowników | 6950 |
| Koszt zespołu programistów | 40 600 |
| Przewidywany koszt projektu | 47 550 |

## Harmonogram:



## Wykres Gantta



## Produkty projektu:

1. Dokumentacja zawierająca:
   1. Plan Projektu na który się składają:
      1. Cel i zakres projektu
      2. Opis zasobów
      3. Oszacowany budżet
      4. Harmonogram
      5. Wykres Gantta
   2. Plan Techniczny na który się składa:
      1. Opisane użyte rozwiązanie
      2. Specyfikacja wymagań
      3. Model przypadków użycia
      4. Architektura rozwiązania
      5. Model Bazy Danych
      6. Diagram Klas
      7. Opis środowiska wdrożeniowego
2. Aplikacja, która posiada wymagane funkcjonalności
3. Plan Testów na który składają się:
   1. Scenariusze testowe
   2. Przygotowane Dane Testowe
4. Raport Testów
5. Dokumentacja wdrożeniowa zawierająca:
   1. Dokumentację użytkownika
   2. Dokumentację administratora

# Projekt Techniczny

## Opis rozwiązania

Przekrój społeczeństwa osób korzystających z aplikacji jest nieograniczony. Dzięki swej idei i intuicyjnej budowie z aplikacji mogą korzystać zarówno dzieci, dorośli i seniorzy. Używać jej będą ludzie, którzy potrzebują zarządzać swoim czasem w celu zwiększenia swojej produktywności czy znalezienia czasu w zatłoczonym dniu na realizowanie hobby czy pasji. Badanie zjawisk społecznych dzisiejszych czasów wskazuje na to, że takie zachowania pojawiają się w dużej większości społeczeństwa.

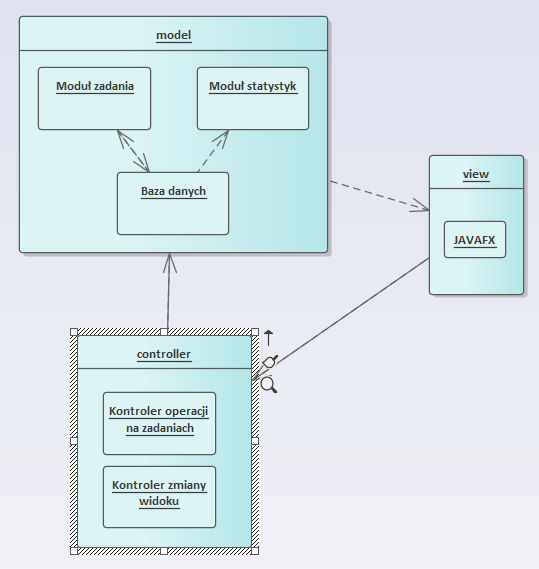
Zatem żeby dobrze zarządzać swoim czasem użytkownik musi kierować się zasadami planowania swojej pracy, które pomoże mu zobrazować aplikacji i sprawi, że będzie miał je cały czas pod ręką.

Użytkownik będzie miał możliwość tworzyć listę zadań, które zaplanował na dany dzień i zgodnie z zasadami planowania będzie mógł dodawać im priorytety. W razie gdy użytkownik nie zrealizuje jakiegoś zadania, a termin będzie się zbliżał zadanie te zostanie podkreślone dając sygnał użytkownikowi.

Po całym miesiącu użytkownik będzie miał dostęp do statystyk, które wskażą jego produktywność i dzięki temu będzie mógł próbować jaki rodzaj pracy daje mu najlepsze efekty.

Aplikacja zostanie wykonana w technologii JavaFX w wersji 11. Zastosowany zostanie wzorzec MVC (Model-View-Controller)

## Architektura rozwiązania



Opis diagramu:  
Zaprezentowane jest zastosowanie wzorca projektowego MVC (Model-View-Controller) w aplikacji:

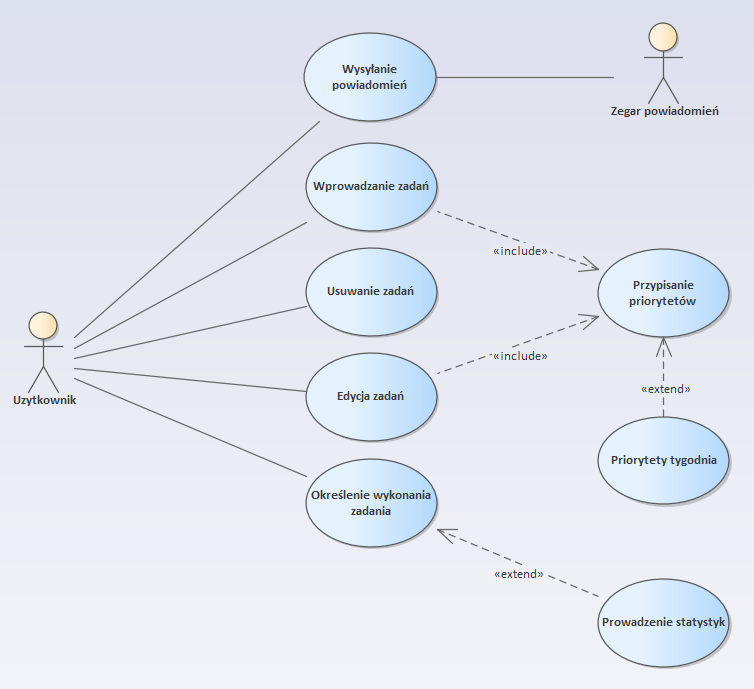
* W warstwie „controller” pokazane są jakie kontrolery są użyte w aplikacji. Odpowiada za wszelkie sterowanie aplikacją.
* Warstwa „view” odpowiada za wygląd graficzny aplikacji. Wykonywana jest w technologii JavaFX
* W warstwie „model” uwzględnione zostały główne moduły aplikacji. Uruchamia moduł zadania i moduł statystyk oraz odpowiada za połączenie z bazą danych.

## Specyfikacja wymagań

|  |  |
| --- | --- |
| Nazwa | Opis funkcji |
| Dodanie zadania | Aplikacja pozwala dodać planowaną aktywność do listy na dany dzień |
| Edycja zadania | Aplikacja pozwala na edycję wcześniej dodanej aktywności |
| Usunięcie zadania | Umożliwia użytkownikowi na pozbycie się wybranej aktywności z listy |
| Przypisanie priorytetów | Pozwala to na określenie ważności poszczególnych zadań |
| Określenie wykonania zadania | Użytkownik może zaznaczyć czy udało mu się wykonać zaplanowane działanie |
| Prowadzenie statystyk | Na podstawie zakończonych zadań aplikacja określa charakterystykę naszej proaktywności |
| Wysyłanie powiadomień o niewykonanych aktywnościach | Jeśli użytkownik nie określił wykonania zadania, aplikacja zakłada, że go nie wykonał i wysyła przypomnienie |
| Zgoda na powiadomienia | Użytkownik określa czy chce dostawać przypomnienia o niezrealizowanych przedsięwzięciach |
| Filtrowanie zadań | Aplikacja pozwala przełączać widok między zadaniami z poszczególnych dni, jak również wyświetla przedsięwzięcia o najwyższym priorytecie w danym tygodniu |
| Sortowanie zadań | Aplikacja umożliwia uporządkowanie listy zadań w danym dniu według priorytetów |

## Przypadki użycia

### Diagram przypadków użycia



#### Opis diagramu:

Diagram przedstawia możliwe funkcje naszej aplikacji w sposób obrazowy.

* Aplikacja może wysyłać powiadomienia (co łączy się z wykorzystaniem zegara powiadomień)
* Można wprowadzać zadania (obowiązkowe jest przypisanie priorytetów)
* Można usuwać zadania
* Można edytować zadania (również edytować priorytety)
* Można przeglądnąć listę wykonanych zadań (Dodatkowo można zobaczyć szczegółowe statystyki

### Specyfikacja przypadków użycia

|  |  |
| --- | --- |
| Nazwa | Wprowadzanie zadań |
| Warunki wstępne | Brak |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Użytkownik dodaje swoje pozycje do listy zadań |
| Scenariusz alternatywny | Odmowa wykonania polecenia z powodu zbyt dużej ilości znaków lub zadań na dany dzień |

|  |  |
| --- | --- |
| Nazwa | Usuwanie zadań |
| Warunki wstępne | Niepusta lista |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Użytkownik usuwa swoje pozycje z listy zadań |
| Scenariusz alternatywny | Odmowa wykonania polecenia z powodu braku zadań |

|  |  |
| --- | --- |
| Nazwa | Edycja zadań |
| Warunki wstępne | Niepusta lista |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Użytkownik edytuje swoje pozycje na liście zadań |
| Scenariusz alternatywny | Odmowa wykonania polecenia z powodu zbyt dużej ilości znaków lub pustej listy |

|  |  |
| --- | --- |
| Nazwa | Przypisanie priorytetów |
| Warunki wstępne | Otworzenie kreatora zadań |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Użytkownik określa lub zmienia ważność danego zadania |
| Scenariusz alternatywny | Ustawienie domyślnej wartości |

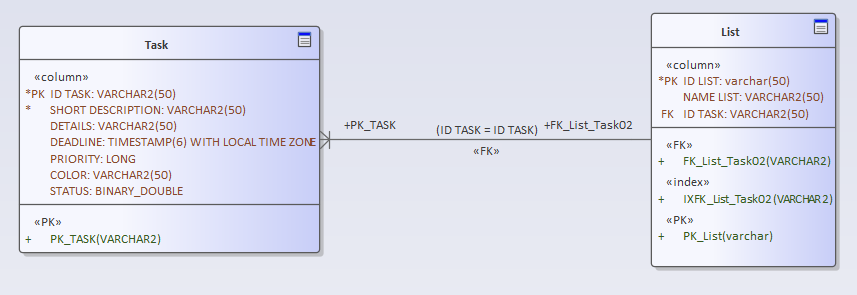
|  |  |
| --- | --- |
| Nazwa | Określenie wykonania zadania |
| Warunki wstępne | Niepusta lista |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Użytkownik określa czy wykonał zadnie |
| Scenariusz alternatywny | Uznanie zadania za niewykonane |

|  |  |
| --- | --- |
| Nazwa | Prowadzenie statystyk |
| Warunki wstępne | Określenie wykonania co najmniej jednego zadania |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | System podlicza podstawowe statystyki |
| Scenariusz alternatywny | Przyjmuje wartości zerowe |

|  |  |
| --- | --- |
| Nazwa | Wysyłanie powiadomień |
| Warunki wstępne | Wprowadzanie zadań do listy |
| Powiązani aktorzy | Zegar powiadomień, Użytkownik |
| Scenariusz główny | System wysyła użytkownikowi powiadomienia, gdy ten nie określił czy wykonał dane zadanie |
| Scenariusz alternatywny | Powiadomienia nie są wysyłane, gdy ta opcja zostanie wyłączona |

|  |  |
| --- | --- |
| Nazwa | Priorytety tygodnia |
| Warunki wstępne | Wprowadzanie zadań do listy |
| Powiązani aktorzy | Użytkownik |
| Scenariusz główny | Aplikacja wyświetla zadania o najwyższych priorytetach w danym tygodniu |
| Scenariusz alternatywny | Gdy jest zbyt dużo zadań o wysokim priorytecie aplikacja wyświetla pustą listę |

## Model bazy danych

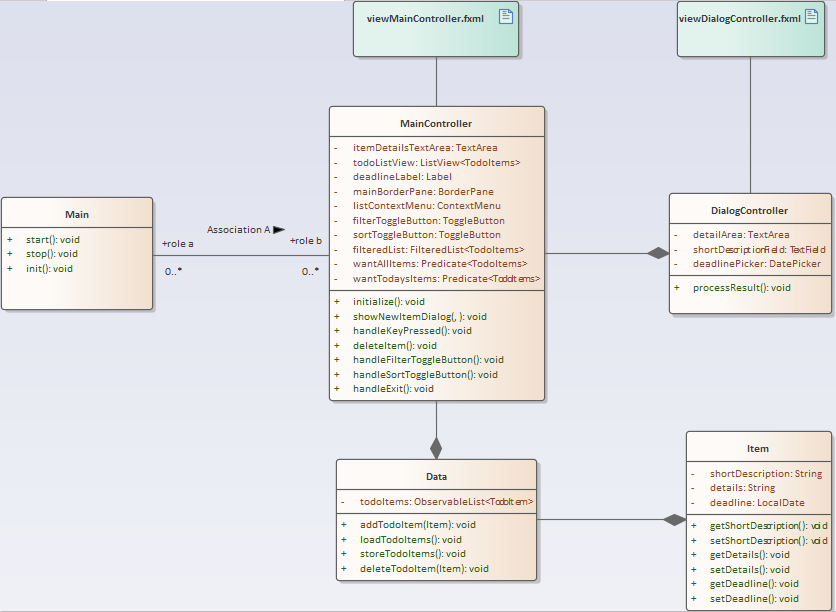


#### Opis diagramu:

Diagram bazy danych przedstawia model bazy danych aplikacji. Widoczna jest Tabela „Task” Przedstawiająca atrybuty zadań. Mamy również tabelę list z jej atrybutami, która może zawierać wiele zadań.

## Model aplikacji

### Diagram Klas

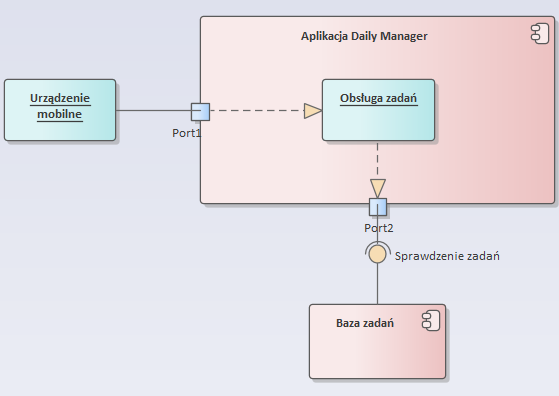


#### Opis diagramu:

Diagram klas przedstawia klasy, które zostaną zaimplementowane w aplikacji.

* W klasie Main będą główne metody uruchamiające aplikację.
* W klasie MainController będą atrybuty i metody sterowania głównym oknem dialogowym. Widać, że klasa współpracuje z plikiem viewMainController.fxml, który odpowiada za widok okna głównego. Wywołuje ona również DialogController. Korzysta ona z danych klasy Data.
* Dialog Controller pojawia się w przypadku dodawania czy edytowania zadań. Współpracuje z plikiem viewDialogController.fxml odpowiadającym za jego widok.
* Klasa Data zawiera dane na temat listy zadań i może zawierać wiele obiektów klasy Item.
* Klasa Item to klasa obiektu zadania. Przechowuje ona podstawowe dane na temat zadania to znaczy, krótki opis, dłuższy opis i termin wykonania zadania oraz metody umożliwiające manewrowaniem zadania.

### Diagram komponentów



Opis diagramu:

Diagram przedstawia komponenty, które składają się na aplikację.

Widzimy komponent baz danych zawierający bazę zadań. Można zauważyć główny komponent aplikacji Daily Manager, który odpowiada za szeroko rozumianą obsługę zadań, który współpracuje z urządzeniem mobilnym, na którym docelowo ma pracować nasza aplikacja.

## Środowisko wdrożeniowe

* Aplikacja będzie tworzona z myślą o urządzeniach mobilnych opartych na systemie android.
* Projekt będzie realizowany w technologii JavaFX w środowisku IntelliJ.
* Utworzona zostanie lokalna baza danych, w celu przyspieszenia działania aplikacji
* Aplikacja oparta będzie na architekturze Model-View-Controller
* Wykorzystany zostanie zestaw narzędzi Android SDK oraz plugin Gluon

# Plan Testow

## Wprowadzenie

### 1.1 Cel dokumentu

Celem jest określenie zakresu i specyfiki testów, które umożliwią sprawdzenie i rozpoczęcie ewentualnych poprawek.

### 1.2 Oczekiwania

Testowanie ma za zadanie udowodnić poprawność wszystkich aspektów aplikacji, zgodność z wymaganiami oraz bezawaryjność.

### 1.3 Załączniki

MALIK\_GUZEK\_Plan\_Projektu

MALIK\_GUZEK\_Projekt\_Techniczny

## Zakres i założenia

### 2.1 System zostanie sprawdzony pod kątem:

* Funkcjonalności
* Integralności
* Stabilności

### 2.2 Sprawdzenie poprawności działania funkcjonalności:

* Wprowadzanie zadań
* Usuwanie zadań
* Edycja zadań
* Określenie wykonania zadania
* Prowadzenie statystyk
* Wysyłanie powiadomień
* Sortowanie według priorytetów

### 2.3 Testowanie integralności

Sprawdzenie czy poszczególne elementy nie kolidują ze sobą oraz czy podczas komunikacji nie następuje utrata danych.

### 2.4 Testowanie stabilności

Poddanie aplikacji obciążeniom, sprawdzenie jak radzi sobie aplikacja działając na dużej ilości danych.

### 2.5 Kryteria

Aby uznać system za gotowy do wdrożenia powinien umożliwiać swobodne korzystanie oraz przejść testy funkcjonalności i integralności. Ewentualne braki w kwestii stabilności będą systematycznie patchowane.

# Wyniki testów

### 3.1 Rezultaty testów

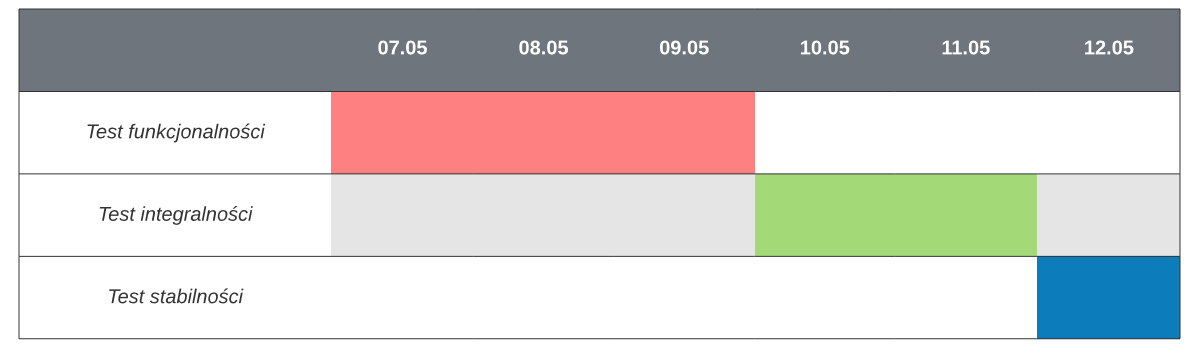
Po przeprowadzeniu testu sporządzana będzie odpowiednia dokumentacja zawierająca opis oraz przebieg testu.

### 3.2 Odnotowanie nieprawidłowości

Wszystkie słabości systemu będą zapisywane i naprawiane w kolejnych wersjach aplikacji.

# Testowanie

### 4.1 Harmonogram



# Zasoby

### 5.1 Zasoby ludzkie

* Adrian Malik
* Łukasz Guzek

### 5.2 Zasoby sprzętowe

Asus VivoBook 14 X412FL

* I5-8265U
* 8GB RAM
* 512GB SSD
* NVIDIA MX250

MSI GL-72

* I7-7700HQ
* 8GB RAM
* GTX1050

# Wymagane oprogramowanie

* MS Windows 10
* IntelliJ IDEA
* Java 13

# Scenariusze testowe

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lp.** | **Nazwa** | **Warunki wstępne** | **Kroki do wykonania** | **Oczekiwany rezultat** |
|  | Wprowadzanie zadań | Brak | 1. Naciśnięcie przycisku odpowiedzialnego za dodanie zadania 2. Wprowadzenie nazwy i opisu 3. Określenie daty i priorytetu 4. Zatwierdzenie | Zadanie zostało wprowadzone zgodnie z przyjętymi wartościami |
|  | Usuwanie zadań | Niepusta lista zadań | 1. Jednokrotne naciśnięcie prawym przyciskiem myszy na zadanie 2. Wybranie opcji Delete | Zadanie zostało usunięte |
|  | Edycja zadań | Niepusta lista zadań | 1. Jednokrotne naciśnięcie prawym przyciskiem myszy na zadanie 2. Wybranie opcji Edit 3. Edycja wybranych parametrów 4. Zatwierdzenie | Wprowadzenie zmian w zadaniu |
|  | Określenie wykonania zadania | Niepusta lista zadań | 1. Zaznaczenie zadania lewym przyciskiem myszy 2. Naciśnięcie przycisku odpowiedzialnego za uznanie zadania jako wykonane | Usunięcie zadania z listy i dodanie do statystyk |
|  | Prowadzenie statystyk | Niepusta lista zadań | 1. Tworzenie i zatwierdzanie ukończenia zadań 2. Sprawdzanie poprawności prowadzenia statystyk | Brak błędów w prowadzeniu statystyk |
|  | Wysyłanie powiadomień | Włączona aplikacja | 1. Naciśnięcie przycisku odpowiedzialnego za włączenie powiadomień (na potrzeby testów powiadomienia ukazują się co sekundę) 2. Ponowne naciśnięcie przycisku w celu wyłączenia powiadomień | Aplikacja prawidłowo reaguje na włączanie i wyłączanie powiadomień |
|  | Sortowanie według priorytetów | Niepusta lista zadań | 1. Naciśnięcie przycisku odpowiedzialnego za sortowanie 2. Ponowne naciśnięcie w celu cofnięcia sortowania | Prawidłowa reakcja na naciskanie przycisku. Zadania są sortowane zgodnie z priorytetem |
|  | Testowanie integralności | Brak | 1. Wielokrotne korzystanie z poszczególnych funkcjonalności 2. Używanie funkcji aplikacji w dowolnej kolejności | Brak utraty danych, prawidłowe działanie systemu |
|  | Testowanie stabilności | Brak | 1. Wpisywanie długich ciągów tekstowych 2. Zostawianie pustych miejsc w polach tekstowych 3. Doprowadzanie aplikacji do sytuacji skrajnych | Ciągłość w działaniu aplikacji, brak zawieszania |

# Raport z testów

## Cel:

Weryfikacja prawidłowości w działaniu aplikacji oraz sprawdzenie podatności na obciążenia.

## Środowisko testowe:

System operacyjny: Windows 10

## Zakres testów:

System zostanie sprawdzony pod kątem:

- Funkcjonalności

- Integralności

- Stabilności

Testy obejmują wykonanie scenariuszy testowych określonych w planie testów według określonego harmonogramu

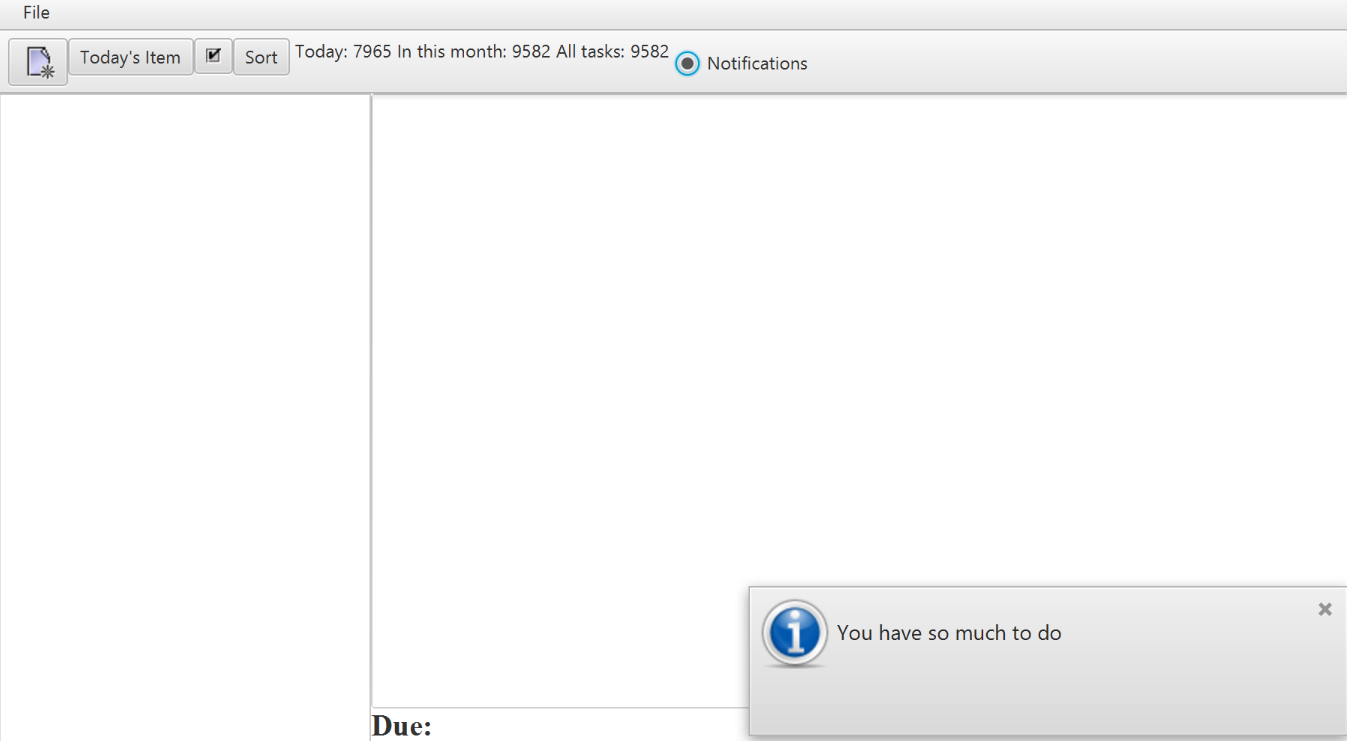
## Wykonane testy:

|  |  |  |
| --- | --- | --- |
| **Lp.** | **Nazwa** | **Czas** |
|  | Wprowadzanie zadań | 10 min |
|  | Usuwanie zadań | 10 min |
|  | Edycja zadań | 10 min |
|  | Określenie wykonania zadania | 10 min |
|  | Prowadzenie statystyk | 40 min |
|  | Wysyłanie powiadomień | 1 h |
|  | Sortowanie według priorytetów | 30 min |
|  | Testowanie integralności | 1,5 h |
|  | Testowanie stabilności | 1h |

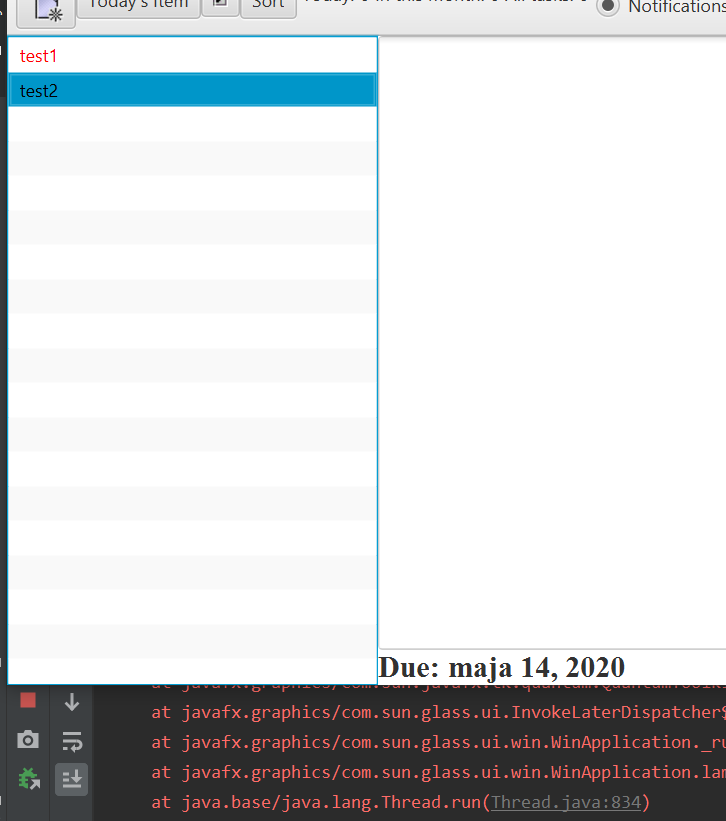
## Wykryte błędy

### Wysyłanie powiadomień

Brak kompatybilności z niektórymi wersjami javy. Nie powoduje to jednak błędów w działaniu aplikacji



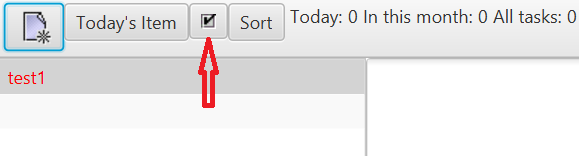
Na wersji javy 1.8 powiadomienia działają poprawnie.

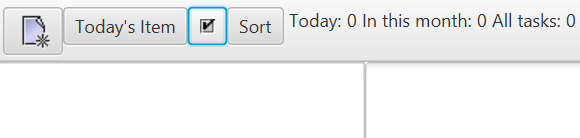


Na innych wersjach, z poziomu użytkownika nie widać żadnych komunikatów, natomiast w intelliJ cyklicznie (dokładnie w momentach, gdy powinno zostać wysłane powiadomienie) pojawiają się informacje o błędach.

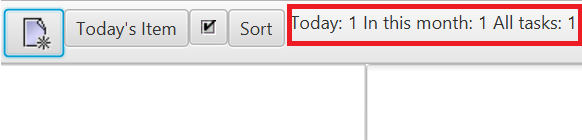
### Prowadzenie statystyk

Funkcja statystyk działa poprawnie natomiast dane zostają zaktualizowane dopiero po ponownym uruchomieniu aplikacji.





Następuje ponowne uruchomienie aplikacji:



### Przewidywany termin usunięcia wykrytych błędów:

Do 27.05.2020

## Wnioski:

Stan aplikacji można określić jako dobry. Brak błędów krytycznych, wykryte niedoskonałości działania systemu nie wpływają na ogólne użytkowanie. Aplikacja radzi sobie z setkami danych bez większych spadków płynności.