****

**Praca projektowa programowanie obiektowe**

*Tytuł projektu[[1]](#footnote-1)*

Prowadzący: Autor:

pracownik UR *Imię Nazwisko*

nr albumu:

Kierunek: Wybierz element., grupa Wybierz element.

Rzeszów 20XX

Spis treści

[**1.** **Opis założeń projektu** 3](#_Toc155609263)

[**2.** **Specyfikacja wymagań** 5](#_Toc155609264)

[**3.** **Opis struktury projektu** 10](#_Toc155609265)

[**4.** **Harmonogram realizacji projektu** 11](#_Toc155609266)

[**5.** **Prezentacja warstwy użytkowej projektu** 12](#_Toc155609267)

[**6.** **Podsumowanie** 14](#_Toc155609268)

[**7.** **Literatura** 15](#_Toc155609269)

1. **Opis założeń projektu**

*Przykład wstępu:*

Na co dzień wielu współpracowników w biurach spotyka się z pewnymi niedociągnięciami pojawiającymi się na tle drobnych obowiązków domowych, z którymi nie chcieliby spotykać się jeszcze dodatkowo w pracy, a które wynikają z zaniedbań na tym tle innych pracowników. Do takich małych niedociągnięć i zaniedbań obowiązków należy dla przykładu pozostawianie na dłuższy czas naczyń w zlewozmywaku, nie wynoszenie na czas śmieci, które zalegają w koszu przez dłuższy czas. Albo też nieumyty po używaniu ekspres do kawy, niewymienione po wykorzystaniu pojemniki z tuszem w drukarce. Problemem istotnym, a niebędącym bezpośrednio związanym z przedmiotem działalności firmy, jest dbanie o takie szczegóły, a które mogą wpływać na samopoczucie pracowników i atmosferę w firmie. Dlatego, wbrew pozorom, ważne jest odpowiednie zmotywowanie pracowników aby o takie, mimo że drobne obowiązki, dbali na równi z własnym domem, dzięki czemu każdemu współpracownikowi będzie się lepiej pracowało. I tutaj pojawia rozwiązanie naszego zespołu – system, który jednocześnie w jakiś sposób rozwiązuje powyższe zagadnienie, a z drugiej strony jest także formą urozmaicenia i rozrywki w pracy.

*Przykłady wstępów do prac projektowych/naukowych/dyplomowych można znaleźć w artykułach/pracach naukowych/dyplomowych dostępnych w sieci (artykuły/prace naukowe) lub w repozytorium uczelni (biblioteka).*

**Cele i założenia projektu**

Tutaj powinien zostać umieszczony tekst zawierający następujące informacje:

* Jaki jest cel projektu?
* Jaki jest problem, który będzie rozwiązywany oraz proszę wskazać podstawowe źródło problemu?
* Dlaczego ten problem jest ważny oraz jakie są dowody potwierdzające jego istnienie?
* Co jest niezbędne, aby problem został rozwiązany przez Zespół i dlaczego?
* W jaki sposób problem zostanie rozwiązany? Jak krok po kroku będzie przebiegała realizacja projektu? Co będzie wynikiem prac. (Wynikiem może być np.: aplikacja, system, sposób, metoda, program komputerowy).

Należy podać założenia do projektu opis. UWAGA!!! Tekst piszemy bezosobowo np. zaprojektowano, stworzono, umieszczono itp.…

Pracownik linii lotniczych może dodawać do harmonogramu nowe loty. Każdy lot jest reprezentowany przez następujące dane: numer lotu, numer samolotu, lotnisko początkowe, lotnisko docelowe, data i przewidywany czas lotu. Klient może zarejestrować się na stronie podając swoje dane osobowe a następnie zakupić bilet lotniczy, wyszukując żądany lot w wyszukiwarce i wybierając dostępną taryfę. Klient przed zakupem będzie widział, jakie są koszty biletów i będzie mógł zakupić ich kilka. Pracownik linii lotniczych może dodatkowo modyfikować i usuwać dane o lotach i zakupionych biletach, oraz dane zarejestrowanych kont. Klient może przeglądać dostępne loty za pośrednictwem strony internetowej lub bezpośrednio za pomocą specjalnego programu. Zakłada się, że klientów jednocześnie przeglądających loty, oraz dokonujących zakupu biletów lotniczych może być około 100. Linie lotnicze mogą oferować kilkanaście różnych tras. Linie lotnicze oferują swoje usługi lotnicze w kilku lotniskach na świecie. Zaleca się język Python + framework Django.

1. **Specyfikacja wymagań**

W tym rozdziale należy zdefiniować wymagania funkcjonalne i niefunkcjonalne projektowanej aplikacji.

**Definicja wymagań funkcjonalnych i niefunkcjonalnych**

**Wymagania funkcjonalne**

* Opisują funkcje (czynności, operacje, usługi) wykonywane przez system.
* Często stosowany sposób opisu wymagań – język naturalny.
* Liczba wymagań funkcjonalnych może być bardzo duża; konieczne jest pewnego rodzaju uporządkowanie tych wymagań, które ułatwi pracę nad nimi (złożoność!).
* Opisują, jak funkcja powinna działać.
* Skupiają się na wyniku działania użytkownika.
* Definiuje wymagania użytkownika.
* Posiada funkcje uwzględnione w przypadkach użycia.
* Weryfikuje funkcjonalność system.

**Wymagania niefunkcjonalne**

* Opisują ograniczenia, przy zachowaniu których system powinien realizować swe funkcje.
* Opisują, jakie właściwości sprawią, że funkcja będzie działać.
* Skupiają się na uproszczeniu procesu i wykonania wyniku.
* Definiują oczekiwania i doświadczenia użytkownika działania użytkownika.
* Posiadają ograniczenia, które pomogą zredukować czas i koszty rozwoju.
* Weryfikują wydajność systemu.

**Wymagania funkcjonalne przykłady:**

Lista przykładów wymagań funkcjonalnych obejmuje każde zachowanie systemu IT, zmieniające się pod wpływem zastosowanej funkcji. Jeżeli wymagania funkcjonalne nie zostaną potwierdzone, system nie będzie działał:

* Reguły biznesowe.
* Poziomy autoryzacji.
* Śledzenie audytów.
* Interfejsy zewnętrzne.
* Funkcje administracyjne.
* Generowanie danych historycznych.
* Uwierzytelnianie użytkownika na żądanie.
* Logi serwera wszystkich istniejących danych.
* Generowanie raportów w określonym czasie.
* Definiowanie poziomów autoryzacji systemu.

**Wymagania niefunkcjonalne przykłady:**

Na liście wymagań niefunkcjonalnych znajdują się:

* Pojemność.
* Wydajność.
* Środowisko.
* Użyteczność.
* Skalowalność.
* Niezawodność.
* Odzyskiwalność.
* Bezpieczeństwo.
* Utrzymywalność.
* Interoperacyjność.
* Integralność danych.
* 2-poziomowe uwierzytelnianie.

**Rozwinięcie wymagań niefunkcjonalnych:**

* Aplikacja IT powinna mieć kolor tła wszystkich ekranów \#fffaaa.
* Aplikacja IT powinna przestrzegać wymagań regulatora.
* Aplikacja IT powinien rejestrować każdą nieudaną próbę logowania;
* Użytkownicy powinni zmienić hasło po pierwszym udanym logowaniu.
* Dashboard powinien pojawić się w ciągu 3 sekund po zalogowaniu użytkownika.
* Aplikacja IT powinien być w stanie obsłużyć XYZ liczbę użytkowników, zapewniając płynne działanie.

**Jak zdefiniować wymagania funkcjonalne?**

Jeśli twoje podejście do rozwoju oprogramowania jest zwinne (Agile), prawdopodobnie zdefiniujesz wymagania w dokumencie. Dokument wymagań funkcjonalnych będzie zawierał historie użytkowników, przypadki użycia, a także następujące sekcje.

* Cel: Ta sekcja będzie zawierała całe tło, definicje i przegląd systemu.
* Zakres aplikacji, oczekiwania i zasady biznesowe.
* Wymagania dotyczące bazy danych, atrybuty systemu i wymagania funkcjonalne.
* Przypadki użycia, czyli opisywać, w jaki sposób użytkownik będzie wchodził w interakcję z systemem. Zdefiniuj rolę każdego aktora biorącego udział w interakcji.
* Napisz jasno cel wdrożenia systemu IT.
* Wspomnij o użytkownikach aplikacji, którzy szczegółowo opiszą, jak krok po kroku będą się angażowali w tworzenie aplikacji.
* Opracuj klikalny prototyp aplikacji. To pomoże Ci reprezentować produkt w lepszy i przekonujący sposób dla interesariuszy. Możesz wybrać prototypy do wyrzucenia lub prototypy interaktywne dla swojego projektu.

**Jak zdefiniować wymaganie niefunkcjonalne?**

Teraz nadchodzi część, w której definiujesz oczekiwania jakościowe aplikacji dedykowanej. Te atrybuty opisują sposoby, w jakie oczekujesz, że aplikacja będzie się zachowywała.

* Zdefiniuj oczekiwania dotyczące użyteczności produktu.
* Opisz, do jakich praw i regulacji aplikacja powinna spełniać.
* Zdefiniuj dostępność aplikacji, czyli czy będzie ona funkcjonować 24/7/365?
* Określ wydajność systemu IT dla różnych funkcjonalności. To znaczy, w jakim czasie użytkownik powinien zobaczyć listę, jak długo użytkownik będzie połączony z aplikacją w przypadku braku połączenia z internetem, itp.
* Zdefiniuj wymagania dotyczące bezpieczeństwa systemu IT.
* Użyj narzędzi do automatycznego testowania, aby upewnić się co do wydajności aplikacji dedykowanej.

**Przekłady:**

Przykłady wymagań funkcjonalnych aplikacji webowej:

* Rejestracja i logowanie użytkowników.
* Bezpieczne uwierzytelnianie i autoryzacja użytkowników.
* Zarządzanie profilami użytkowników.
* Możliwość wyszukiwania treści w aplikacji.
* Funkcjonalność e-commerce, taka jak koszyk i proces kasowy.
* Treści generowane przez użytkowników, takie jak komentarze i oceny.
* Integracja z usługami stron trzecich, takimi jak media społecznościowe i bramki płatności.
* Dynamiczne aktualizacje treści i powiadomienia.
* Pulpit administracyjny do zarządzania aplikacją.

**Przykłady wymagań niefunkcjonalnych aplikacji webowej:**

* Użyteczność aplikacji i dostępność, takie jak responsywny design i dostępność klawiatury.
* Wydajność aplikacji i skalowalność, np. szybkie czasy ładowania i zdolność do obsługi dużej liczby użytkowników jednocześnie.
* Bezpieczeństwo aplikacji i prywatność, takie jak szyfrowanie wrażliwych danych i ochrona przed atakami.
* Niezawodność aplikacji i dostępność, np. kopie zapasowe i plany odzyskiwania danych po awarii.
* Zgodność aplikacji z wymogami prawnymi i regulacyjnymi, takimi jak GDPR i przepisy dotyczące dostępności.
* Interoperacyjność aplikacji, taka jak zgodność z różnymi przeglądarkami i systemami operacyjnymi.
* Utrzymanie i wsparcie aplikacji, takie jak łatwość aktualizacji i dokumentacja dla programistów.
* Efektywność kosztowa aplikacji, taka jak minimalizacja kosztów serwera i hostingu.

System zarządzania treścią (CMS) umożliwiający edycję i usuwanie treści.

**Przykłady wymagań funkcjonalnych aplikacji webowej:**

* Integracja aplikacji z zewnętrznymi API w celu wymiany danych lub rozszerzenia funkcjonalności.
* Funkcje optymalizacji aplikacji pod kątem wyszukiwarek (SEO) w celu poprawy widoczności w wyszukiwarkach internetowych.
* Obsługa wielu języków w aplikacji w celu dostosowania do użytkowników posługujących się różnymi językami.
* Narzędzia współpracy, takie jak czat w czasie rzeczywistym i udostępnianie plików dla zespołów.
* Narzędzia analizy danych do śledzenia zachowań użytkowników i wydajności aplikacji.

Projektowanie doświadczeń użytkownika (UX), takich jak łatwy w użyciu interfejs i przejrzysta nawigacja.

**Przykłady wymagań niefunkcjonalnych aplikacji webowej:**

* Optymalizacja mobilna aplikacji, np. responsywny design i podejście mobile-first.
* Integracja systemu, np. kompatybilność z dotychczasowymi systemami i narzędziami stron trzecich.
* Bezpieczeństwo aplikacji i prywatność danych, takie jak szyfrowanie danych, kopie zapasowe i kontrola dostępu.
* Zgodność aplikacji z normami branżowymi, takimi jak PCI-DSS dla aplikacji e-commerce.
* Wsparcie dla użytkowników aplikacji, takie jak help desk, podręczniki użytkownika i samouczki.
* Monitorowanie aplikacji i raportowanie, takie jak śledzenie błędów i metryki wydajności.

To tylko kilka przykładów funkcjonalnych i niefunkcjonalnych wymagań aplikacji internetowych. Konkretne wymagania będą zależały od celu i charakteru aplikacji internetowej, a także potrzeb użytkowników i zainteresowanych stron.

1. **Opis struktury projektu**

W tym rozdziale mają pojawić się informacje odnośnie zaprojektowanej struktury oraz jej opis wraz z opisem technicznym. Należy umieścić informacje odnośnie wykorzystywanego języka, narzędzi oraz minimalnych wymagań sprzętowych. Opisać zarządzanie danymi oraz BD, umieścić informacje odnośnie zaprojektowanej hierarchii klas wraz z krótkim opisem najważniejszych metod.

Np.:

* Środowisko programistyczne Javy: Java JDK Kit 7u79.
* Środowisko programistyczne: Android Studio 2.1.2.
* Urządzenie z system operacyjnym Android 4.4 KitKat – może być emulator.
* Ze względów na liczbę szczegółów aplikacja była projektowana na urządzenia typu tablet – w Samsung Tab Pro 10,1’.

Należy przedstawić diagram klas projektowanej aplikacji (diagram można wygenerować po zrealizowaniu projektu) oraz opisać strukturę przechowywanych danych w pliku/bazie.

Przykładowy diagram klas:

Obraz zawierający tekst, diagram, Plan, Rysunek techniczny

Opis wygenerowany automatycznie

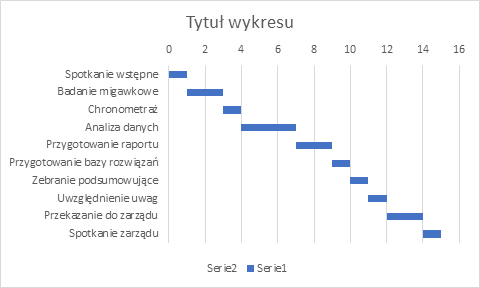
Rysunek 3. Diagram klas projektowanej aplikacji

1. **Harmonogram realizacji projektu**

W rozdziale tym należy umieścić harmonogram realizacji projektu – diagram Ganta. Rysunek oraz krótki opis do niego. Można napisać jakie problemy trudności wystąpiły w trakcie realizacji projektu. Rozdział ten musi zawierać informacje o repozytorium i systemie kontroli wersji.

**UWAGA!!!**

Należy pamiętać że pliki do projektu na repozytorium musza być dostępne przez rok od dnia złożenia końcowej pracy. W przypadku gdy ktoś chciałby usunąć pliki z repozytorium do projektu należy dołączyć załącznik z plikami źródłowymi.



Rysunek 2. Diagram Gantta

Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git, wszystkie pliki źródłowe projektu znajdują się pod adres: <https://github.com/> i będą dostępne do 31.01.2024. Na rysunku 6 przedstawiono zrzut ekranu pokazujący historię kosmitów.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

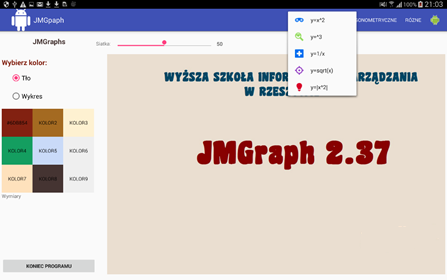
Rysunek 6. Historia komitów

1. **Prezentacja warstwy użytkowej projektu**

Należy przedstawić opis warstwy użytkowej projektu w tym celu należy umieścić opis aplikacji oraz PrtSc o których jest mowa. Np.:

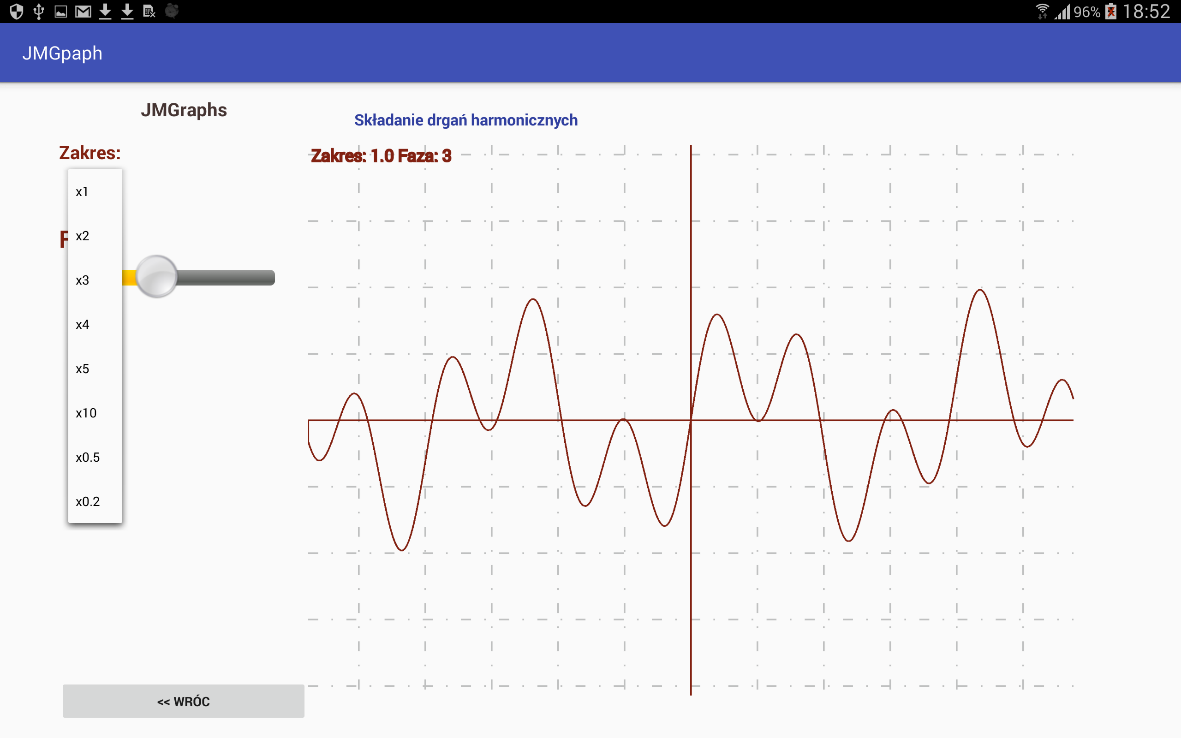
Na rysunku 1 przedstawiono główne okno aplikacji. Użytkownik po uruchomieniu aplikacji otrzymuje ekran startowy na którym może:

* Ustawić kolor tła i linii wykresu – domyślnie kolor tła jest czarny, a wykresu biały.
* Zobaczyć wymiary grafiki i obszaru rysowania.
* Z menu górnego wybiera grupy wykresów:
  + Proste wykresy: y=x2, y=x3, y=1/x, y=√(x,) y=|x2|.
  + Funkcje trygonometryczne: y=sin(x), y=cos(x), y=th(x), y=ctg(x).
  + Różne: 2 fraktale: Smok Heighwaya, Krzywe Kocha oraz superpozycja funkcji harmonicznych.
* Ostatnia pozycja z ikoną Androida wyświetla informację o autorze.
* Dodatkowo poniżej menu można ustawić skalę siatki wykresu.



Rysunek 4. Główne okno aplikacji

Krzywe Kocha oraz krzywe harmoniczne wyświetlane są w nowych aktyvity z własnym layoutem (patrz Rysunek 2). Generując krzywe Kocha możemy wybrać ich stopień: od 0 – 6. Ponieważ są to funkcje rekurencyjne generowanie wyższych stopni trwa zbyt długo. Tworząc funkcje harmoniczne możemy ustalić zakres wykresu – wielokrotność 3600 oraz różnicę faz między funkcjami – w płynnej skali względnej 0 – 10.

****

Rysunek 5. Przykład wygenerowanego wykresu dla zadanych parametrów

1. **Podsumowanie**

W rozdziale należy opisać zrealizowane prace oraz ewentualne planowane dalsze prace rozwojowe projektu.

1. **Literatura**

Należy podać literaturę książki/podręczniki/artykuły lub link do stron www.

1. Jacek Matulewski, Visual Studio 2017. Tworzenie aplikacji Windows w języku C#, 2018, Wyd. Helion.
2. Kurs WPF [www.przykaldowastrona.pl](http://www.przykaldowastrona.pl) (data dostępu: 05.06.2021).

1. Przykład poprawnie zdefiniowanego tytułu: System do raportowania przewinień w biurze, Elektroniczny system oceniania, itp. Przykład niepopranie zdefiniowanego tytułu: Kuchenka mikrofalowa, Samochód, itp. [↑](#footnote-ref-1)