**AKADEMIA MARYNARKI WOJENNEJ**

**im. BOHATERÓW WESTERPLATTE**

**Wydział Mechaniczno-Elektryczny**

Instytut Elektroniki i Automatyki Okrętowej

Katedra Informatyki

**PRACA DYPLOMOWA**

**INŻYNIERSKA**

Temat:

Wykonawca: Adam SZREIBER

Kierownik pracy

dr inż. Jan Masiejczyk

Kierownik katedry

kmdr dr hab. inż. Andrzej Żak

prof. nadzw. AMW

|  |  |  |
| --- | --- | --- |
| Ocena pracy dyplomowej | ………………..…………  słownie | ……………………...……….  data i podpis  Przewodniczącego Komisji Egzaminacyjnej |

Gdynia, dnia …………….…….. r.

Adam Szreiber

21662

**O Ś W I A D C Z E N I E**

Oświadczam, że przedłożoną do egzaminu pracę dyplomową pt. ………………………………………………………………………………………………  
……………………………………………….………………………………………………

kończącą studia I stopnia napisałem samodzielnie. Przy wykonywaniu pracy nie zlecałem jej opracowania ani żadnej jej części innym osobom, jak też nie skopiowałem cudzych opracowań   
i przestrzegałem postanowień Ustawy z dnia 4 lutego 1994 r. o prawie autorskim   
i prawach pokrewnych (Dz. U. z 2018 r. poz. 1191 z późn. zmianami).

Ponadto oświadczam, iż treści zaczerpnięte z literatury przedmiotu są oznaczone w tekście oraz w przypisach, w sposób ogólnie przyjęty dla prac naukowych.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, jestem świadomy(a) zasadności cofnięcia decyzji o wydaniu mi dyplomu.

Wyrażam zgodę na udostępnienie mojej pracy dyplomowej czytelnikom.

Przekazuję moją pracę dyplomową do Ogólnego Repozytorium Prac Dyplomowych.

…………………………………

podpis osoby składającej oświadczenie

Spis treści

[1. Wykaz skrótów i oznaczeń 5](#_Toc97063030)

[2. Wprowadzenie 6](#_Toc97063031)

[3. Dokumentacja projektu 7](#_Toc97063032)

[3.1 dqwedLorem ipsum 7](#_Toc97063033)

[3.2 Lorem ipsum dolor 7](#_Toc97063034)

[4. Implementacja aplikacji 8](#_Toc97063035)

[5. Podsumowanie 9](#_Toc97063036)

[6. Literatura / Załączniki 10](#_Toc97063037)

[7. Streszczenie 11](#_Toc97063038)

# Wykaz skrótów i oznaczeń

|  |  |
| --- | --- |
| JavaScript |  |
| TypeScript |  |
|  |  |

# Wprowadzenie

## Komercyjne aplikacje

Firmy informatyczne wytwarzające oprogramowanie pracują nad swoimi aplikacjami przez kilkadziesiąt lat. Ich code-base może liczyć wiele miloionów linii kodu, rozmieszczone w tysiącach plików. Korporacje starają się zapewnić funkcjonalności jakie oczekują od nich użytkownicy, usprawniają to co kiedyś zostało zrobione, wprowadzają nowe funkcje, tylko po to aby być konkurencyjnym na rynku, Pozyskać nowych użytkownik, utrzymać dotychczasowych dużych, stałych klientów. Na przestrzeni wielu lat struktura, organizacja i pracownicy firmy mogą się zmieniać, a jakość wytwarzanego kodu degraduje się. Aby temu zapobiec każda firma posiada swój styleguild – którego muszą przestrzegać pracownicy firmy. Pomaga to w zachowaniu ogólnego porządku w kodzie programu, ułatwia nowym pracownikom wdrożenie/zapoznanie się z nim, ułatwia proces code-review, ułatwia wyszukiwanie i naprawę błędów. Stosuje się również narzędzia typu SOLARLINT – który kontroluje jakość kodu, a w razie niedostosowania się do zadanych restrykcji informuje o tym np. poprzez podkreślenie zle napisanego kodu i wyświetlenie dokładnego błędu. Firmy stosują również pryncypia które narzucone przez sam język programowania w którym pracujemy. Dlatego tak ważne jest aby styl kodu dango programu był spójny, czytelny i otwarty na rozbudowę.

Napisać trochę o kontrolowaniu wersji… GIT

Wprowadzanie zmian i rozwój produktu jest kluczowym elementem aby zapewnić firmie byt na dynamicznie zmieniającym się rynku oprogramowania. Lecz nie należy to do najłatwiejszych rzeczy gdy pracujemy kodzie napisanym przez innego pracownika lub kodzie który jest bardzo przestarzały – napisany przed wprowadzeniem narzędzi kontrolujących jakość kodu / napisanym przed wprowadzeniem wewnętrznych firmowaych kryteriów kodowania. Aby wprowadzić jakiekolwiek zmiany, najpierw musimy poświęcić wiele godzin aby znaleźć odpowiednie miejsce w kodzie, następnie dokładnie przeanalizować dany fragment kodu, opracować plan wdroż enia nowej funkcjonalności, i ostatecznie przejść do implementacji. Tak złożony proces może przysporzyć wiele trudności, szczególnie mniej doświadczonym programistom.

## Cel pracy

*„Czasu nie cofniesz, ale błędy możesz naprawić.”*

Bezpieczeństow programu to sprawa najwyższje wagi. Firmy budują swoje zaufanie w sród klientów przez dziesiąki lat, jeden krytyczny błąd może spowodować katastrofalne skutki .

LEKKI OPIS TYPÓW BŁĘDÓW APLIKACJI. – BŁASZZCZYK.

Serwis Beheer (nid. „zarządzanie”) to projekt który ma wspomóc programisów bezpiecznie wprowadzać zmiany w kodzie progogramu, poprzez feature-flagi, automatyczne wyłączanie flag gdy wystąpi błąd wykonywania kodu, wykresy i statystyki. Wspołczesne programy są niezwykle rozbudowane i nie sposób prześledzić wszystkich ścierzek wykonywanego kodu, co za tym idzie, zmiany w jednym obszarze aplikacji mogą spowodować wystąpienie w innym module aplikacji. Szybkie reagowanie na takie zachowania mogą uratować firmę. Skótki wystąpnienia błędów krytycznych niosa za sobą poważne konsekwencje - w najgorszym przypadku spowodują zamknięcie firmy, sprawy karne, czy zadość uczynienie.. Z historii znamy już kilka takich przypadków:

PODAĆ TUTAJ KILKA KATASTROFALNYCH W SKUTKACH BŁĘDÓW PROGRAMÓW.

Czy wprowadznie narzędzi typu feature-toogle pozwoliłoby tego uniknąć? – Nie jestem w stanie na to odpwiedzeć, ale z pewnością pomogłoby szybciej i precyzyjniej zareagować na te błędy.

## Konkurencyjne rozwiązania na rynku

Feature toggle – launchdarkly

[Feature Flags: Faster software deployment and safer code releases | Split](https://www.split.io/product/feature-flags/)

[The Top 6 Feature Flag Management Tools | Harness](https://harness.io/blog/feature-flag-management-tools/)

[Do czego przydaje się technika Feature Flag (bulldogjob.pl)](https://bulldogjob.pl/news/871-do-czego-przydaje-sie-technika-feature-flag)

Zgłębiając temat Feature-flag można znaleźć wiele komercyjnych produktów na rynku. Niektóre z nich bardzo zaawansowane, zawierające […] złożone wykresy, możliwość serwowania różnych wartości flag na podstawie geolokalizacji, <<<<ITD>>>>

## Wady i zalety feature-flag

ZALETY:

Łatwość przywrocenia poprzedniej ścieżki kodu bez ingerencji w kod

Możliwość automatycznego wyłączania flagi gdy wystąpi błąd

Możliwość serwowania różnych typów flag – wartość string, boolean, intiger

Możliwość wprowadzania większej grupy zmian, kontrolowanej za pomocą 1 flagi

WADY:

Brak dostępu do internetu spowoduje wykonywanie starej ścieżki programu.

Spowolnienie wykonywania programu – komunikacja przez Ethernet jest stosunkowo wolna

Testowanie aplikacji staje się trudniejsze

Brak możliwości wykorzystania flag w high-performance apps

# Budowa Aplikacji

Proces wytwarzania aplikacji jest złożony, na początku ery komputerów, gdy stacje robocze były słabe/wolne i ograniczone przez zasoby, programy pisało się i konserwowało bez większych trudności. Na przełomnie lat 50/60 XIX wieku rozwój technologiczny przyspieszył do takiego stopnia, że zaczęto tworzyć coraz bardziej skomplikowane oprogramowanie, którego implementacja i utrzymanie zaczęło sprawiać trudności.

W latach 1968 i 1969 miały miejsce dwie konferencje w Rzymie i Garmich na których po raz pierwszy wprowadzono termin „INZYNIERIA OPROGRAMOWANIA” (ang. Software engineering, SE) dyscyplina informatyczna stosująca podejście inżynierskie do tworzenia oprogramowania: od analizy i określenia wymagań, przez projektowanie i wdrożenie, aż do ewolucji gotowego oprogramowania.”

Inzynieria oprogramowania zajmuje się metodami wytwarzania, oceniania i utrzymywania oprogramowania systemów komputerowych oraz metodami zarządzania realizacją projektów informatycznych. Celem stosowania tych metod jest zapewnienie wysokiej jakości oprogramowania oraz doprowadzenie do terminowej i zgodnej z budżetem realizacji projektu. Znaczenie metod inżynierii oprogramowania rośnie wraz z wielkością projektu

Dziedzina ta wprowadza wiele procesów (Proce s jest zbiorem aktywności wykonywanych z myślą o osiągnięciu pewnego celu) które należy dobrać indywidualnie do tworzonego projektu. Do realizacji niniejszego projektu użyto modelu kaskadowego (eng. Waterfall).



Jest to szeroko stosowany model charakteruzujacy się jasno uporządkowanym przebiegiem pracy, brakiem powtarzania aktywności oraz posiada wbudowany mechanizm weryfikacji wyników każdej z faz. Każda faza następuje po sobie a błąd w jakiejkolwiek faz jest bardzo kosztowne w usunięciu, ponieważ skutkuje powrotem do pierwszej fazy czyli określenia wymagań.

## Określenie wymagan

Cos tu będzie ………………………………………………………………………..

## Projektowanie

Cos tu będzie ………………………………………………………………………..

## Implementacja

Cos tu będzie ………………………………………………………………………..

## Eksploatacja

Cos tu będzie ………………………………………………………………………..

Cos tu będzie ………………………………………………………………………..

# Możliwe ścieżki rozwoju aplikacji

Roloout

String value instead of Boolean

Zaimplementowanie lepszego łapania błędów i wysyłanie ich do service

Wprowadznie pobierania danych w raz z błędami które wystąpiły

Przerobic clienta tak aby mogl bys kontrolowany za pomocą wartości true/false z plikow config

# Podsumowanie

Cos tu będzie ………………………………………………………………………..

# Literatura / Załączniki

Cos tu będzie ………………………………………………………………………..

# Streszczenie

bb Jakos to chyba dziala teraz.

Jakos to chyba dziala teraz.