**WYDZIAŁ MATEMATYKI I INFORMATYKI UNIWERSYTETU ŁÓDZKIEGO**

Kierunek studiów: Informatyka

Specjalizacja: Grafika Komputerowa i Projektowanie Gier

**Aplikacja internetowa „System zgłaszania błędów”**

Mikołaj Szymański

Numer albumu: 359892

Praca licencjacka napisana  
w Katedrze Funkcji Rzeczywistych  
pod kierunkiem  
dr Sebastiana Lindnera

**Łódź 2018**

Spis treści

[Wstęp 2](#_Toc511770720)

[1. Wzorce architektury używane w projektowaniu aplikacji internetowych. 3](#_Toc511770721)

[**1.1.** Opis wzorca architektury MVC 3](#_Toc511770722)

[**1.2.** Opis wzorca architektury MVP 3](#_Toc511770723)

[**1.3.** Opis wzorca architektury MVVM 3](#_Toc511770724)

[**1.4.** Który model jest odpowiedni dla aplikacji internetowej? 3](#_Toc511770725)

[2. Dokumentacja użytkownika 3](#_Toc511770726)

[**2.1.** Do czego służy aplikacja 3](#_Toc511770727)

[**2.2.** Instrukcja obsługi aplikacji 3](#_Toc511770728)

[**2.2.1.** Tworzenie zgłoszenia 3](#_Toc511770729)

[**2.2.2.** Przeglądanie zgłoszeń 3](#_Toc511770730)

[**2.2.3.** Edycja zgłoszeń 3](#_Toc511770731)

[3. Dokumentacja programisty 3](#_Toc511770732)

[**3.1.** Użyte narzędzia programistyczne 3](#_Toc511770733)

[**3.1.1.** Instalacja 3](#_Toc511770734)

[**3.1.2.** Konfiguracja 3](#_Toc511770735)

[**3.2.** Opis działania kodu 3](#_Toc511770736)

[**3.3.** Klasy 3](#_Toc511770737)

[**3.3.1.** Klasa Account 3](#_Toc511770738)

[**3.3.2.** Klasa Ticket 3](#_Toc511770739)

[**3.3.3.** Klasa FileUploadHelperExtension 3](#_Toc511770740)

[**3.4.** Opis działania wybranych funkcji 3](#_Toc511770741)

[**3.4.1.** Funkcja DownloadFile 3](#_Toc511770742)

[**3.5.** Graficzna reprezentacja schematu bazy danych 3](#_Toc511770743)

[4. Podsumowanie 3](#_Toc511770744)

[5. Bibliografia 3](#_Toc511770745)

# Wstęp

Tu będzie wstęp.

# Wzorce architektury używane w projektowaniu aplikacji.

## Opis wzorca architektury MVC

Wzorzec Model-View-Controller jest przeznaczony do projektowania aplikacji internetowych. **Model** reprezentuje naszą warstwę biznesową, czyli nie tylko klasy mapujące dane pobrane z bazy danych. To również funkcjonalności potrzebne do działania naszej aplikacji . **View** jest odpowiedzialny za prezentowanie danych użytkownikowi, pobieranych z Modelu, który jest dla niego tylko do odczytu. **Controller** przekierowuje wszelkie żądania użytkownika na wywołanie metod konkretnego Modelu, oraz przygotowuje dane dla Widoku i żądaniem jego wygenerowania. Jeden kontroler może obsługiwać kilka widoków.

## Opis wzorca architektury MVP

Wzorzec Model-View-Presenter używany jest głównie do projektowania aplikacji mobilnych. **Presenter** w tym wzorcu ma podobne zastosowanie co kontroler w MVC, lecz dodatkowo posiada on w sobie logikę biznesową, która w MVC jest pod Modelem. Tutaj dane nie są przekazywane bezpośrednio z modelu na widok, ale prezenter wykonuje zapytania do modelu o pewne wartości, które później są przetwarzane i wysyłane na widok. **Model** jest zwykłą reprezentacją danych, zawierająca powiązania i struktury danych. **View** ma to samo zadanie co w architekturze MVC, wyświetlić nasze żądania. W tej architekturze, presenter odnosi się do jednego widoku, nie może obsługiwać wielu.

## Opis wzorca architektury MVVM

Wzorzec Model-View-ViewModel to wzorzec wykorzystywany głównie do aplikacji graficznych, w których dane są mocno ze sobą powiązane. Powstał na bazie architektury MVC, i stąd **Model** jest niezmienny w swoich zastosowaniach. **View** jest interfejsem użytkownika, który nie ma pojęciao **ViewModel-u**. **ViewModel** pobiera dane z Modelu i przygotowuje je do wyświetlenia na możliwie wiele Widoków. Ten wzorzec nie ogranicza się do ilości obsługiwanych widoków.

## Który wzorzec jest odpowiedni dla mojej aplikacji?

Każdy z opisanych wzorców ma swoje wady i zalety. Zastosowanie **MVP** ogranicza nas do obsługi tylko jednego widoku, przy użyciu **MVC** bądź **MVVM** nie musimy się martwić o limit obsługi widoków. Jednak w porównaniu do **MVVM**, popularność wśród aplikacji internetowych architektury **MVC** jest zauważalnie większa**.** Użycie tej architektury pomaga programistom współpracować ze sobą nad jednym kodem. Dzięki modułowej budowie (Model – View- Controller) modyfikacja kodu jest prosta a sam kod jest przejrzystszy.

# Dokumentacja użytkownika

## Do czego służy aplikacja

Aplikacja została stworzona do użytku przez firmy informatyczne, które mogą kontrolować ilość błędów w programach tworzone, bądź użytkowane przez nie. Użytkownik, czyli osoba posiadająca do własnego użytku konkretny program, po zarejestrowaniu się na stronę, posiada możliwość zgłoszenia błędu wynikłego podczas korzystania z programu komputerowego. Podając szczegóły, wrzucając zdjęcia obrazujące powstały ambaras, użytkownik zaznajamia serwisantów. Serwisanci po przeanalizowaniu problemu, podejmują po swojej stronie odpowiednie kroki, żeby zażegnać kłopot.

## Instrukcja obsługi aplikacji

* + 1. Rejestracja użytkownika

### Tworzenie zgłoszenia

Aby móc stworzyć zgłoszenie, użytkownik musi być zarejestrowany

### Przeglądanie zgłoszeń

### Edycja zgłoszeń

# Dokumentacja programisty

## Użyte narzędzia programistyczne

### Instalacja

### Konfiguracja

## Opis działania kodu

## Klasy

## Klasa Account

#### Controller

#### Model

* + - 1. View

### Klasa Ticket

#### Controller

#### Model

* + - 1. View
    1. Klasa FileUploadHelperExtensions

## Opis działania wybranych funkcji

### Funkcja DownloadFile

## Graficzna reprezentacja schematu bazy danych

# Podsumowanie

A tu będzie podsumowanie

# Bibliografia