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

Tu będzie wstęp.

1. Wzorce architektury używane w projektowaniu aplikacji.

1.1. 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.

1.2. 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.

1.3. 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 pojęcia o **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.

1.4. 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.

2. Dokumentacja użytkownika

2.1. 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.

2.2. Instrukcja obsługi aplikacji

2.2.1. Rejestracja użytkownika

2.2.2. Tworzenie zgłoszenia

Aby móc stworzyć zgłoszenie, użytkownik musi być zarejestrowany

2.2.3. Przeglądanie zgłoszeń

2.2.4. Edycja zgłoszeń

3. Dokumentacja programisty

3.2. Użyte narzędzia programistyczne

3.2.1. Instalacja

3.2.2. Konfiguracja

3.3. Opis działania kodu

3.4. Klasy

3.4.1. Klasa Account

3.4.1.1. Controller

3.4.1.2. Model

3.4.1.3. View

3.4.2. Klasa Ticket

3.4.2.1. Controller

3.4.2.2. Model

3.4.2.3. View

3.4.3. Klasa FileUploadHelperExtensions

3.5. Opis działania wybranych funkcji

3.5.1. Funkcja DownloadFile

3.6. Graficzna reprezentacja schematu bazy danych

4. Podsumowanie

A tu będzie podsumowanie

5. Bibliografia