Wydział Informatyki, Elektroniki i Telekomunikacji

Katedra Informatyki

Zegar do gier – przypominacz/budzik dla wielu zdarzeń o charakterze cyklicznym

Koncepcja projektu

Damian Kudas, Michał Mrowczyk

Wersja: 0.2

Spis Treści

1. Wstęp 2
2. Słownik pojęć 2
3. Zarys architekturalny produktu 3
3.1 Architektura aplikacji klienckiej. 3
4. Wykorzystywane technologie (rozwiązania) 3
4.1 Narzędzia wspierające tworzenie produktu....3
4.2 Stosowane biblioteki do realizacji GUI 4
4.3 Inne technologie....4
5. Możliwe rozszerzenia 4

1. Wstęp

Niniejszy dokument zawiera ogólną koncepcję architektury tworzonej aplikacji. Wprowadza słownik stosowanych pojęć oraz charakteryzuje pokrótce używane w aplikacji technologie.

2. Słownik pojęć

aplikacja mobilna - wersja aplikacji na urządzenia mobilne. Posiada bardziej kompaktowy interfejs i mniejsze wymagania sprzętowe, przyjazny interfejs graficzny.

aplikacja stacjonarna - wersja aplikacji przeznaczona do zastosowania w grach na komputery stacjonarne.

klient mobilny - osoba korzystająca z aplikacji mobilnej

budziki synchroniczne – alarmy, które są ustawione na konkretną godzinę, nienakładające się na siebie w czasie. Informujące użytkownika o nadejściu danego zdarzenia sygnałem dźwiękowym.

3. Zarys architekturalny produktu

Naszym celem jest skonstruowanie produktu (artefaktu) oprogramowania, nie zaś całego systemu. Mając to na uwadze naszym celem będzie stworzenie modularnej aplikacji w Javie o architekturze możliwie zgodnej ze wzorcem projektowym (architekturalnym) MVC (Model – View – Controller).

Model - model danych, struktury danych (klasy), które w naszym projekcie będą przechowywały informacje o zdefiniowanych alarmach, ich właściwościach oraz różnego rodzaju dane konfiguracyjne aplikacji.

View - widok, interfejs użytkownika stworzony w ramach projektu. Stanowi odzwierciedlenie modelu danych, umożliwia interakcję na modelu danych użytkownikowi (za pośrednictwem kontrolera)

Controller – kontroler, zmienia zawartość modelu pod wpływem zdarzeń wykonanych na widoku.

3.1 Architektura aplikacji klienckiej.

Klienci będą mieli możliwość zarządzania swoją listą budzików. Wyświetlania ich wszystkich, modyfikacji czasów oraz ich dźwięków. Warstwa prezentacji za pomocą przystępnego GUI będzie wyświetlała użytkownikowi czas do rozpoczęcia alarmu przy śledzeniu konkretnego budzika bądź zdarzenia. Warstwa prezentacji dostarczy przystępnej opcji modulacji głośności dźwięku alarmu. Zakładamy możliwość stworzenia dowolnej liczby budzików, których alarmy będą mogły pojawiać się cyklicznie, by nie było konieczności ponownego ich ustawiania.

4. Wykorzystywane technologie (rozwiązania)

4.1 Narzędzia wspierające tworzenie produktu

- **AndroidSDK-** narzędzie ułatwiające uruchamianie aplikacji na Androida pod eclipsem. Dostarcza wymagane sterowniki, oraz biblioteki, pozwalając na symulację działania aplikacji.
  Więcej: http://developer.android.com/sdk/index.html
- **MAVEN** – narzędzie ułatwiające (automatyzujące) budowanie, testowanie i deployowanie produktu. Umożliwia również wygodne zarządzanie dependencjami.

Więcej: http://maven.apache.org/

- **ECLIPSE JUNO 4.2** – Zintegrowane środowisko programistyczne(IDE) umożliwiające sprawne tworzenie kodu w Javie oraz innych językach programowania np. C, C++, Python itp. Jest to inteligentne narzędzie umożliwiające łatwe tworzenie oprogramowania.
  Więcej: http://www.eclipse.org/
- **JAVADOC** – narzędzie służące do generowania dokumentacji do kodu. Dokumentacja generowana jest w postaci dokumentów html.
  Przykład: http://docs.oracle.com/javase/6/docs/api/
  Więcej:
  http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

4.2 Stosowane biblioteki do realizacji GUI

- **AndroidSDK-** Proste narzędzie do interfejsu graficznego dostarczane przez bibliotekę.
- **SWING** – prosta biblioteka do tworzenia GUI
  Więcej: http://pl.wikipedia.org/wiki/Swing_%28Java%29
- **SWT (Standard Widget Toolkit)** – alternatywna biblioteka do tworzenia GUI w Javie. Więcej: http://pl.wikipedia.org/wiki/Standard_Widget_Toolkit

4.3 Inne technologie

Poza wymienionymi wyżej możemy wykorzystać inne technologie, które ułatwią nam proces kodowania projektu. Decyzje o ich stosowaniu będą podejmowane na etapie implementacji projektu (dynamicznie).

5. Możliwe rozszerzenia

Na obecną chwilę nie planujemy implementacji na inne systemy niż Android. Związane jest to z naszym ograniczonym czasem i trudnością w przenośności aplikacji Androida na dekstopową. Jeśli jednak wystarczy nam czasu i chęci, to po ostatecznej implementacji na Androida możemy podjąć się realizowania i tego zadania w trybie dodatkowym.