`

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie oraz nie korzystałem ze źródeł innych niż wymienione w pracy.

…….

(czytelny podpis)

Spis treści

1. Wstęp

Już od najdawniejszych czasów ludzie interesowali się różnego rodzaju zmaganiami sportowymi. Organizowano rozmaite zawody sportowe, igrzyska, turnieje a na początku XX wieku powstały cyklicznie rozgrywki ligowe. Do najpopularniejszych rozgrywek można zaliczyć: piłkę nożną, koszykówkę, hokej, baseball czy piłkę siatkową. Kibice obok ekscytacji związanej z oglądaniem spotkań sportowych zaczęli interesować się również statystykami prowadzonymi podczas rozgrywanych spotkań. Zaczęto odnotowywać liczbę zdobytych punktów, bramek czy asyst na mecz przez danego zawodnika. Wylicza się różnego rodzaju średnie, które pozwalają wybrać najlepszego zawodnika ligi, stworzenie najlepszej drużyny sezonu czy ustalenie która drużyna ma najlepsza obronę lub atak. Zaczęły też powstawać rożnego rodzaju gry planszowe czy karciane, które poprzez opis parametrów danego zawodnika czy drużyny umożliwiały rozegrać kilku osobom rozgrywki zbliżonej do tej rzeczywistej.

Pod koniec XX wieku rozwój komputerów i oprogramowania pozwolił na tworzenie aplikacji, które miały na celu symulować dane rozgrywki. Powstały rożnego rodzaju gry symulujące rozgrywki piłkarskie czy innych sportów drużynowych. Odkąd pamiętam zawsze interesowałem się piłką nożną. Oglądałem wszystkie turnieje mistrzostw świata i Europy, śledziłem różne ligi europejskie i puchary. Jednocześnie byłem miłośnikiem symulatorów rozgrywek piłkarskich tak zwanych futbol menadżerów. Grałem w nie od czasów Commodore c-64 po dzień dzisiejszy. Zawsze zastanawiałem się jak działają i jak są skonstruowane. Tak narodziła się pasja do programowania, która w połączeniu z pasją do piłki nożnej pozwoliła mi stworzyć symulator rozgrywek piłkarskich.

2. Cel Pracy

Celem niniejszej pracy jest stworzenie aplikacji, która umożliwiła by symulowanie rozgrywek ligowych w tym przypadku rozgrywek piłki nożnej.

3. Wybór technologii

W tym rozdziale skupię się na przeglądzie dostępnych technologii na rynku, które umożliwiają wytwarzanie oprogramowania. Przedstawię podział języków ze względu na kryterium przydatności oraz uargumentuję dlaczego jedna grupa jest lepsza od innej do wykonania tego projektu. Dodatkowo postaram się opisać biblioteki graficzne dedykowane do poszczególnych języków, które umożliwiają w krótkim czasie wytwarzanie aplikacji okienkowych.

Języki programowania możemy podzielić na kilka grup. Najpopularniejsze to:

- funkcyjne
- skryptowe
- proceduralne
- obiektowe

Przedstawicielami języków funkcyjnych jest Haskell, Lisp, Erlang czy F#. Programy napisane w językach funkcyjnych składają się jedynie z funkcji. Główny program jest funkcją, która przyjmuje argumenty i zwraca obliczoną wartość. Może składać się on z innych funkcji, które mogą wywoływać jeszcze inne funkcje. Funkcja może przyjmować jako parametr funkcję jak również zwracać inna funkcję. W programowaniu funkcyjnym zamiast zmiennych występują stałe a zamiast pętli stosowana jest rekurencję. Najważniejszą cechą tych języków jest to, iż nie pozwalają na żadne efekty uboczne. W czystym programowaniu funkcyjnym, raz zdefiniowana funkcja dla tych samych danych wejściowych zawsze zwróci ten sam wynik. Języki funkcyjne zostały zaprojektowane z myślą o zastosowaniach współbieżnych (Erlang) oraz pod kątem tworzenia systemów rozproszonych wymagających długotrwałej pracy oraz odporności na awarie. Jak również często stosowane są do tworzenia sztucznej inteligencji (Lisp).

Drugą grupą są języki skryptowe. Jak sama nazwa wskazuje są to języki, które obsługują skrypty. Najpopularniejsze to JavaScript, Pyton, PHP, Ruby on Rails, Perl czy Node.JS. Grupa ta odróżnia się od pozostałych tym, że kod napisany w tych językach może być tworzony nawet w notatniku i nie musi być kompilowany do pliku wykonywalnego - zrozumiałego dla komputera. Proces kompilacji w językach skryptowych odbywa się za każdym razem w momencie uruchomienia programu. Zaletą takiego programowania jest szybsze pisanie aplikacji a programy tworzone w tych technologiach mają zwykle małą objętość. Języki skryptowe często stosuje się do automatyzowania różnego rodzaju zadań najczęściej administracyjnych. Skrypty wykorzystywane są również do sterownia przebiegiem gry, kontrolują wtedy fabułę, dialogi czy sterują zachowaniem wirtualnych postaci. Języki takie jak PHP czy JavaScript wykorzystywane są w technologiach internetowych (HTML) do dynamicznej interakcji z użytkownikiem poprzez reagowanie na zdarzenia, zapewnieniu walidacji danych z formularzy czy dynamicznym manipulowaniu obiektami na stronach internetowych.

Kolejną grupą reprezentowaną przez takie języki jak: C, Pascal czy Fortran są języki proceduralne. Głównym założeniem tych technologii jest to, że program główny składa się z wielu procedur (podprogramów) lub funkcji realizujących określone zadania które mogą być wywoływane wile razy. Zadaniem procedur jest modyfikacja zmiennych globalnych, albo też

modyfikacja wartości lub struktury przekazanej w parametrze, natomiast funkcja oprócz modyfikacji parametrów może je również zwracać. Możliwe jest również zagnieżdżanie odwołań do procedur dzięki czemu można korzystać z rekurencji.

Na rynku programistycznym istnieje wiele różnych technologii za pomocą których można by zrealizować aplikację do symulowania rzeczywistych rozgrywek sportowych. Zarówno języki proceduralne-strukturalne (C, Fortran, Pascal) jak i obiektowe (Smaltalk, Python, C++, Java, C#) nadają się do tworzenia aplikacji czy też skomplikowanych systemów symulujących rozmaite zdarzenia występujące w świecie rzeczywistym jak i wirtualnym. Jednakże popularność w ostatnich latach języków obiektowych oraz dostępność darmowych i rozbudowanych bibliotek, które ułatwiają i przyśpieszają znacząco tworzenie kodu przemawia na korzyść właśnie tych języków. Dużym atutem i przewagą języków obiektowych nad językami proceduralnymi jest fakt, że taki sposób programowania jest bardziej zgodny rzeczywistością a ludzki mózg jest lepiej przystosowany do takiego podejścia podczas przetwarzania informacji. Dodatkowo podstawowe założenia paradygmatu obiektowego takie jak abstrakcja czy enkapsulacja przyczyniają się do zwiększenia czytelności kodu natomiast dziedziczenie i polimorfizm są zwykle używane w celu redukcji zbędnego kodu.

3.1 C++ i Qt

3.2 Java i Swing

3.3 C# WPF