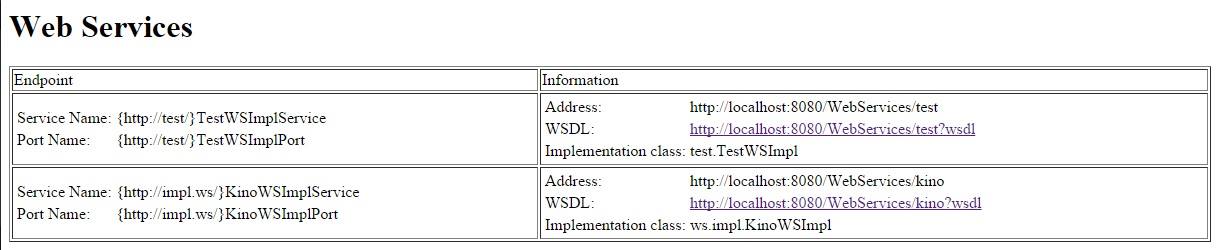
|  |  |  |
| --- | --- | --- |
| ***Systemy równoległe i rozproszone*** | | |
| *Imię i nazwisko:*  Dawid Skóbel  Tomasz Machna  Maciej Nawara | *Temat projektu:*  Kino | *Data oddania projektu:* |
| *Kierunek i wydział:*  WIMiIP Informatyka Stosowana | *Ocena:* |

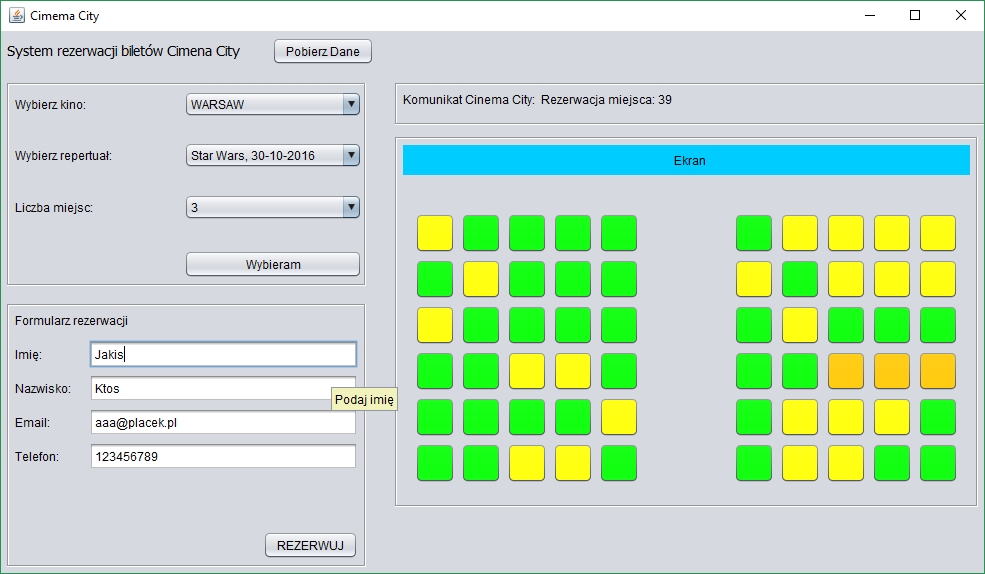
**1. Opis projektu.**

Głównym celem projektu było udostępnienie użytkownikowi możliwości zamówienia biletów do kina oraz wybrania najbardziej odpowiedniego miejsca w zależności od swoich upodobań. Stworzony projekt jest systemem rozproszonym wykorzystującym technologie Web Services. Cały projekt został napisany w technologii Java z użyciem serwera Tomcat oraz bazy danych MySQL. Projekt dzieli sie na dwie części:

**- Część po stronie serwera** - jest to część programu niewidoczna z punktu widzenia użytkownika. Ma ona za zadanie opublikowanie odpowiednich Web Service’ów potrzebnych do komunikacji z aplikacją kliencką oraz odpowiada za połączenie z bazą danych oraz determinuje i udostępnia cały system realizujący rezerwacje biletu.



**- Część po stronie klienta** - część programu dzięki, której występuje interakcja z użytkownikiem. Użytkownik za pomocą tej aplikacji może wybrać odpowiadający mu film i miejsca, mając do dyspozycji pełna wizualizacje miejsc które są dostępne w danej chwili dla danego seansu. GUI programu również zostało zaimplementowane za pomocą Javy, jest ona udostępniona w postaci aplikacji Java.



W przypadku wybrania interesującego nas seansu, użytkownik musi uzupełniając podstawowe dane, które pozwolą mu zarezerwować bilety, w przypadku błędu zostanie poinformowany odpowiednim komunikatem w celu poprawy danych, w przeciwnym wypadku otrzymujemy komunikat o poprawnej rezerwacji miejsc.

**2. Etapy realizacji projektu.**

1. **zebranie wymagań oraz rozłożenie prac projektowych:**

Po opracowaniu wymagań projektowych poszczególne etapy realizacji projektu zostały rozdzielone pośród Studentów.

1. **założenie repozytorium do przechowywania kodu:**

Kolejnym etapem było założenie repozytorium do przechowywania, wersjonowania oraz synchronizacji kodu. W tym celu posłużono się repozytorium w serwisie GitHub.

1. **utworzenie bazy danych:**

W ramach projektu stworzono bazę danych MySQL, która ma przechowywać dane każdego seansu oraz rezerwację i zajęte już miejsca. Do tego celu stworzono osobną bazę danych i dwie tabele. Połączenie z bazą posiada Aplikacja serwera, która ma za zadanie wystawić Web Service’y.

1. **założenie repertuaru dla poszczególnych kin:**

Założono osobny plik konfiguracyjny, który został umieszczony na serwerze w którym przechowujemy informacje o aktualnym repertuarze. Zawiera on informacje o dostępnych kinach oraz filmach, na które możemy rezerwować miejsca. Aplikacja kliencka przy starcie pobiera te informacje w celu uzupełnienia menu użytkownika. Część tych informacji (id) jest zsynchronizowana z baza danych.

1. **implementacja Web Service’ów:**

Kolejnym etapem była implementacja aplikacji po stronie serwera, która ma za zadanie wykonywanie wszystkich operacji na bazie danych, synchronizacje rezerwacji i wolnych miejsc na dany seans. Komunikuje się za pomocą wystawionych Web Service’ów.

1. **implementacja Aplikacji klienta:**

Druga aplikacją, która klient/użytkownik komunikuje się z serwerem jest aplikacja kliencka, która udostępnia menu, gdzie mamy do wyboru interesujący nas film i po wybraniu wszystkich interesujących klienta miejsc i wpisaniu danych ich rezerwacja.

1. **synchronizacja projektu:**

W tym etapie wybrano sposób komunikacji klienta i serwera oraz ustandaryzowano sposób przesyłania i odbierania danych po obu stronach. Komunikacja odbywa się za pomocą języka XML, poprzez wygenerowania wiadomości w standardzie SOAP oraz przesłanie wraz z danymi odpowiednich informacji. Wszystko jest możliwe poprzez udostępnienie przez serwer odpowiedniego adresu WSDL, za którym wykonywana jest odpowiednia operacja po stronie serwera oraz zwracana odpowiedź.

1. **wykonanie testów i sprawdzenie poprawności działania, ewentualne poprawki:**

W ramach projektu wykonano testy poprawności działania aplikacji.

1. **dokumentacja oraz opis projektu:**

Sporządzono również dokumentację w formacie Doxygen, oraz opis projektu. Dołączono również diagramy: klas UML oraz diagram Use-Case UML.

**3. Cechy systemów rozproszonych zawarte w projekcie.**

Opisywany system posiada następujące cechy systemu rozproszonego:

**a) Dzielenie zasobów (ang. resource sharing)** - cecha ta zapewnia nam następująca rzecz: wielu użytkowników systemu może korzystając z danego zasobu.

W naszym programie zasoby, które są dzielone przez użytkowników to:

- dostęp do bazy danych - jest realizowany w celu pobrania informacji o zajętych miejscach po wybraniu danego seansu oraz w celi rezerwacji i walidacji dla danego zgłoszenia o rezerwacji.

- pobranie repertuaru - po uruchomieniu aplikacji klienckiej następuje pobranie z serwera informacji o dostępnych kinach oraz filmach, na które możemy zarezerwować bilety.

**b) Otwartość (ang. openness)** – podatność na rozszerzenia, możliwość rozbudowy systemu zarówno pod względem sprzętowym, jak i oprogramowania. Podstawowym elementem do rozbudowy jest tutaj kwestia oprogramowania, gdzie proponowanymi elementami do przyszłej rozbudowy są:

-dodanie wysyłania maila z potwierdzeniem,

-panel administracyjny,

-logowanie i kupno biletów,

-przeniesienie/udostępnienie aplikacji na urządzenia mobilne,

-ogólne poprawki.

**c) Współbieżność (ang. concurrency)** – zdolność do przetwarzania wielu zadań jednocześnie. Program jest przystosowany do odbierania żądań od wielu użytkowników co za tym idzie, w tej samej chwili wielu klientów może rezerwować bilety.

**d) Skalowalność (ang. scalability)** – cecha systemu umożliwiająca zachowanie podobnej wydajności systemu przy zwiększaniu skali systemu (np. liczby procesów, komputerów, itp.). W naszym programie jest to zachowane ponieważ komunikacja odbywa się za pomocą Web Service a sam czas komunikacji nie jest wielki. Komunikacja z serwerem odbywa się tu za pomocą wysłania z klienta koperty SOAP’owej oraz odebranie wiadomości zwrotnej.

**e) Tolerowanie awarii (ang. fault tolerance)** – System jest odporny na awarię, w testowanych przypadkach zachowuje się zgodnie z założeniem. Wykrywając awarie dostajemy komunikat, który wyświetla się po stronie klienta, jednak sama aplikacja działa dalej poprawnie.

**f) Przeźroczystość (ang. transparency)** – właściwość systemu powodująca postrzeganie systemu przez użytkownika jako całości, a nie poszczególnych składowych. System jest w pełni przeźroczysty dla użytkownika, wszystkie operację walidacji oraz pobierania danych czy też rezerwacji wykonują się za pośrednictwem Web Service, które są umieszczone na serwerze razem z bazą danych, czego klient nie jest w stanie zauważyć posługując się aplikacją.

**4. Diagramy UML.**

Diagram Use-Case

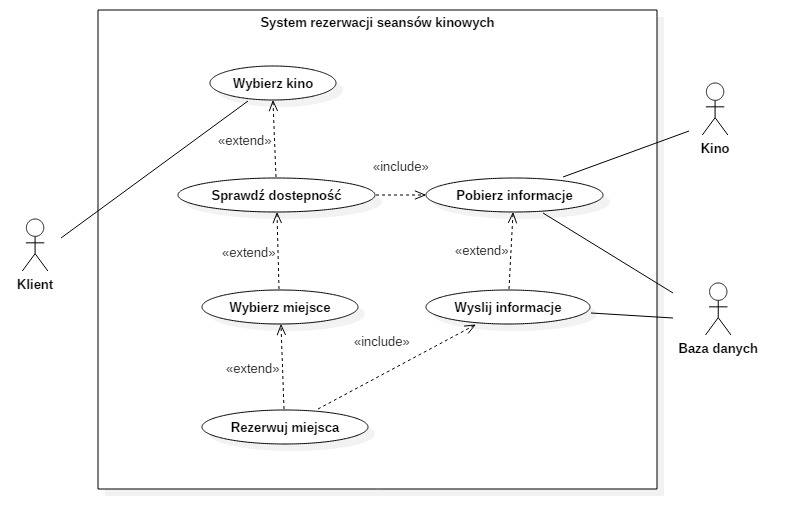
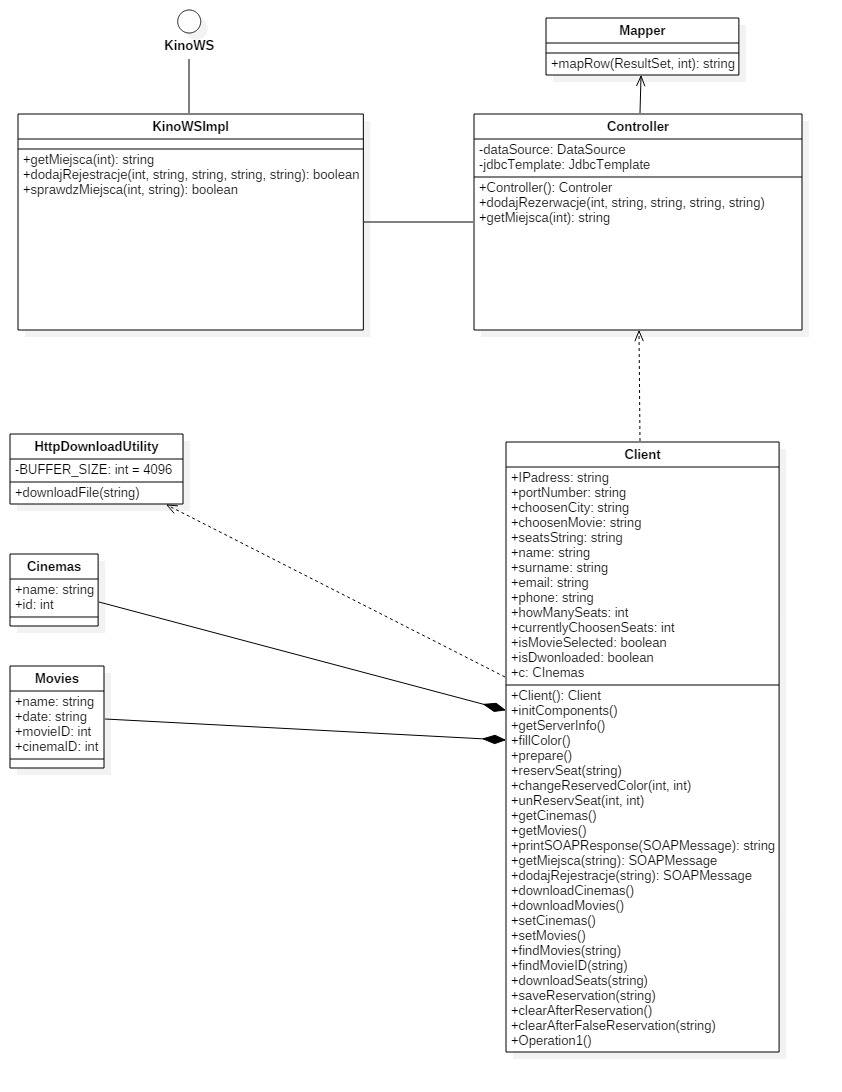
****

Diagram klas

****

**5. Podsumowanie.**

Zgodnie z założeniami projektu utworzono system umożliwiający użytkownikowi zamawianie biletów do kina za pośrednictwem aplikacji klienckiej. Wszelkie postawione w pierwszym etapie cele zostały wykonane i program działa poprawnie umożliwiając wykonywanie wszystkich zaimplementowanych operacji. Równocześnie system został napisany w taki sposób że posiada wiele rozwiązań rozwojowych, które w przyszłości mogą usprawnić działanie stworzonej aplikacji. Zagadnienia te zostały opisane przy okazji omawiania cech systemu rozproszonego. Bardzo dużym polem do rozwoju jest tutaj udostępnienie aplikacji klienckiej dla systemu Android. Ponieważ aplikacja została napisana w języku Java kompatybilność jest duża i nakład pracy przy implementacji ograniczy się do implementacji części odpowiadającej za interakcję z użytkownikiem i menu dla użytkownika oraz obsługi zdarzeń. System ten został przetestowany oraz jest gotowy dla przyszłych użytkowników.