POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INS

PROJEKT INŻYNIERSKI

Budowa graficznego interfejsu użytkownika aplikacji sieciowej z wykorzystaniem Java2Script

Development of a web application's GUI with the use of Java2Script

AUTOR:

Mateusz Adamiak

PROWADZĄCY PRACĘ:

dr inż. Tomasz Kubik

OCENA PRACY:

WROCŁAW, 2012

Spis treści

Skróty

XSD (ang. *XML Schema Definition*)

XML (ang. *eXtensible Markup Language*)

JS (ang. *JavaScript*)

J2S (*Java2Script*)

MSWiA (*Ministerstwo Spraw Wewnętrznych i Administracji*)

OOP (ang. *Object-Oriented Programming*)

Rozdział 1

Wstęp

W dzisiejszych czasach Internet to główne źródło wiedzy i informacji. Dziennie przeglądamy dziesiątki stron WWW, pobieramy wiele plików, kontaktujemy się z innymi użytkownikami, używamy bankowości internetowej. Nie sposób wymienić dzisiaj wszystkich funkcji, jakie oferuje nam sieć.

Jednym z podstawowych elementów działania sieci są serwisy internetowe. To dzięki nim możemy korzystać z dobrodziejstw, jakie daje nam Internet. Przeglądanie towarów w sklepie internetowym, skrzynka pocztowa, czytanie najnowszych wiadomości ze świata, oglądanie filmów online – takie aktywności wymagają stron internetowych lub aplikacji sieciowych.

Stajemy tu przed problemem komunikacji użytkownika z serwisem lub aplikacją sieciową. Człowiek, używając interfejsu użytkownika, ma możliwość interakcji z komputerem i korzystania z funkcji programu czy też strony internetowej.

Historia interfejsu użytkownika jest tak długa, jak długa jest historia komputera. Od czasu powstania maszyny liczącej, projektant stawał przed zadaniem skonstruowania odpowiedniego zestawu przyrządów, które ułatwią korzystającemu z maszyny interakcję z urządzeniem.

Rozróżniamy trzy rodzaje interfejsu użytkownika: tekstowy, graficzny i interfejs strony internetowej. Najuboższy jest tryb tekstowy, oferujący nam urządzenia wejścia-wyjścia w postaci klawiatury i wyświetlacza lub drukarki znakowej.

Jednak jako ludzie, jesteśmy nauczeni wskazywać, jest to dla nas naturalne, stąd drugi typ interfejsu, jakim jest interfejs graficzny. Aby umożliwić łatwiejszą interakcję z komputerem, wymyślono urządzenie, które popularnie nazywamy myszką. Pozwala nam ona wskazywać na elementy wyświetlacza i wchodzić w bardziej naturalną dla nas interakcję z komputerem. Postęp techniki w tym względzie prowadzi nas w stronę ekranów dotykowych, co ułatwia nam komunikację z maszynami. Niepotrzebne są wtedy żadne urządzenia zewnętrzne, takie jak myszka.

Ostatnim rodzajem interfejsu jest strona internetowa. To rozwiązanie powstało na potrzeby rozwoju sieci. Przeglądarka internetowa wyświetla nam pożądaną stronę, po której następnie nawigujemy i wchodzimy w interakcję z systemem, który jest po stronie serwera.

Przyszłość interfejsu użytkownika czeka jeszcze długa droga, by komunikacja z maszyną odbywała się w środowisku naturalniejszym dla człowieka. Prowadzone są badania nad rzeczywistością wirtualną, czy interfejsami mózg-maszyna, które mogą ułatwić użytkownikowi współpracę z komputerem.

Ten projekt inżynierski skupia się właśnie na implementacji interfejsu użytkownika. Stworzona aplikacja posiada graficzny interfejs użytkownika, tworzący komunikację między człowiekiem a maszyną i umożliwia użytkownikowi na korzystanie z zaimplementowanych funkcji systemu.

1.1. Cel

Celem niniejszej pracy jest implementacja interfejsu graficznego dla użytkownika aplikacji sieciowej, realizującej funkcje wybranego systemu.

1.2. Zakres

Zakres powstałego projektu obejmuje:

1. Implementacja logiki biznesowej w języku Java,
2. Implementacja interfejsu graficznego dla użytkownika aplikacji, używając biblioteki graficznej SWT,
3. Wygenerowanie strony HTML – z zawartym skryptem Javascript – przy użyciu wtyczki do Eclipse'a – Java2Script,
4. Przetestowanie działania serwisu,
5. Napisanie dokumentacji projektowej wraz z instrukcją wdrożenia.

Rozdział 2

Wymagania projektowe

Zrealizowany projekt został wykonany w oparciu o serwis EDAP na potrzeby pracy inżynierskiej na kierunku Informatyka na Politechnice Wrocławskiej. Projekt nosi nazwę AtEditor. Aplikacja stanowi narzędzie, mające na celu usprawnienie pisania aktów prawnych zgodnie ze schematem narzuconym przez ministerstwo.

Serwis EDAP – elektroniczna forma aktów prawnych, został przygotowany przez MSWiA w związku z wejściem w życie rozporządzenia w sprawie wymagań technicznych dokumentów elektronicznych zawierających akty normatywne i inne akty prawne, elektronicznej formy dzienników urzędowych oraz środków komunikacji elektronicznej i informatycznych nośników danych (Dz. U. 2008 Nr 75, poz 451 z późn. zm.).

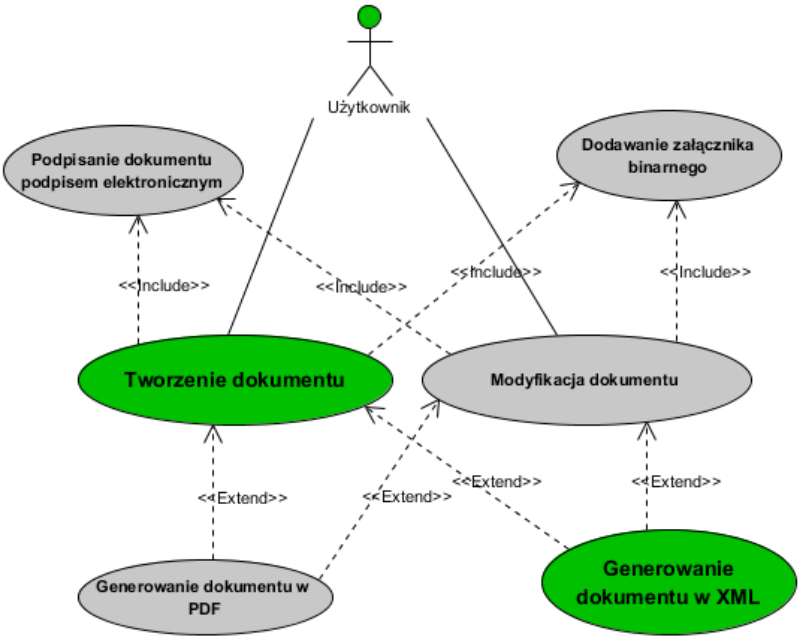
Aplikacja służy do tworzenia oraz edycji aktów prawnych, wspomaga także zarządzanie oraz przechowywanie stworzonych dokumentów.

Akty prawne są generowane w postaci pliku z rozszerzeniem .xml oraz .pdf w oparciu o elektroniczny schemat aktów prawnych, przygotowany przez MSWiA w postaci pliku schema.xsd.

2.1. Wymagania funkcjonalne

Aplikacja ActEditor ma prezentować użytkownikowi przyjazny interfejs, który ułatwi mu korzystanie z takiego narzędzia, jak generator aktów prawnych. Istniejący serwis EDAP bywa pod tym względem mało intuicyjny. Korzystanie z Java2Script powinno umożliwić korzystanie ze stworzonej aplikacji za pomocą interfejsu webowego.

Na rys. 2.1 został diagram przypadków użycia serwisu EDAP. Na zielono wyróżnione zostały funkcje systemu, które zostały zaimplementowane w ActEditor, a na szaro które nie zostały zrealizowane, m.in. ze względu na ograniczenia użytych technologii.



Rys. 2.1: Diagram przypadków użycia dla użytkownika serwisu EDAP

2.2. Wymagania niefunkcjonalne

Poniżej zostały przedstawione języki programowania, języki znaczników, narzędzia wspomagające oraz technologie użyte w projekcie.

Java
Główny język programowania, użyty w projekcie. Oprogramowana jest w nim cała logika biznesowa, opierająca się na wczytaniu danych z pliku schema.xsd, wyświetleniu i obsłudze danych w oknie aplikacji, generowaniu rezultatu, jakim jest plik XML.

JavaScript

HTML

XML

XSD

Eclipse IDE

Java2Script

Github, EGit

. . . //TODO: uzupełnić

2.3. Cykl życia projektu

Cykl życia projektu reprezentuje model przyrostowy (iteracyjny). W stosunku do modelu kaskadowego daje on o wiele większą elastyczność, dzięki powrotom do wstępnych faz projektu i pozwala reagować na opóźnienia, czyli przyspieszać prace nad innymi częściami projektu. Daje też elastyczność pod względem definiowania wymagań – możemy początkowo zdefiniować wstępne wymagania, podczas gdy końcowa specyfikacja może powstać dopiero na etapie testowania.

1. Definiowanie projektu

   a) Etap inicjowania projektu

      Inicjowanie projektu wymaga odpowiedzi na pytania: jaka jest tematyka przedsięwzięcia? Jakie są główne problemy zagadnienia? Jakie problemy ma rozwiązać projekt?

   b) Etap definiowania celu

      Definiowanie zagadnienia polega na wyspecyfikowaniu kluczowego problemu, który formułujemy jako cel główny. Identyfikuje się tutaj również cele cząstkowe, które muszą zostać zrealizowane, by osiągnąć cel końcowy, a także prognozuje się potencjalne przeszkody, które mogą wpłynąć na sukces projektu.

   c) Etap weryfikacji

      W tym etapie ocenia się możliwość wykonania postanowionych wymagań oraz weryfikuje szanse na osiągnięcie sukcesu projektu.

2. Planowanie

   Podsumowanie tego, co musi być wykonane w ramach projektu, aby osiągnąć cel. Tworzy się tu podział na podzadania, które są łatwe do zorganizowania i zarządzania.

3. Wykonanie planu projektu

a) Organizacja kamieni milowych

Ustalenie kolejności wykonywanych podzadań wraz z rozmieszczeniem ich w czasie. Specyfikowana jest tutaj kolejność wykonywania zadań oraz ich przewidywany czas wykonania.

b) Wprowadzenie planu w życie

Wykonanie wyspecyfikowanych zadań i dążenie do ich realizacji przed upływem ich szacowanego czasu trwania.

c) Przetestowanie aplikacji

Testowanie implementowanych części systemu. Najczęściej jest to etap iteracyjny, który jest wykonywany po zaimplementowaniu konkretnej funkcji w systemie.

4. Zamykanie projektu

a) Ocena projektu

Następuje tu ocena odpowiedniości, efektywności, skuteczności projektu. Jest to przegląd osiągnięć realizowanego projektu w stosunku do początkowych oczekiwań. Odpowiada na pytania: czy rezultaty projektu były zgodne z oczekiwaniami? Czy prace zostały wykonane zgodnie z planem?

b) Wnioski

Wyciągnięcie wniosków z realizacji projektu. Opis doświadczeń zdobytych w trakcie wykonywania zadania, które mają służyć poprawie przyszłych projektów i programów.

Rozdział 3

Java2Script

3.1. Analiza technologii

Java2Script (J2S) Pacemaker jest wtyczką typu open-source do środowiska Eclipse, która pozwala na konwertowanie klas języka Java do JavaScript. Wtyczka opiera się na zorientowanym obiektowo symulatorze JavaScript, który został nazwany Java2Script (J2S) Clazz. Zapewnia on pełne wsparcie OOP, włączając dziedziczenie i polimorfizm.

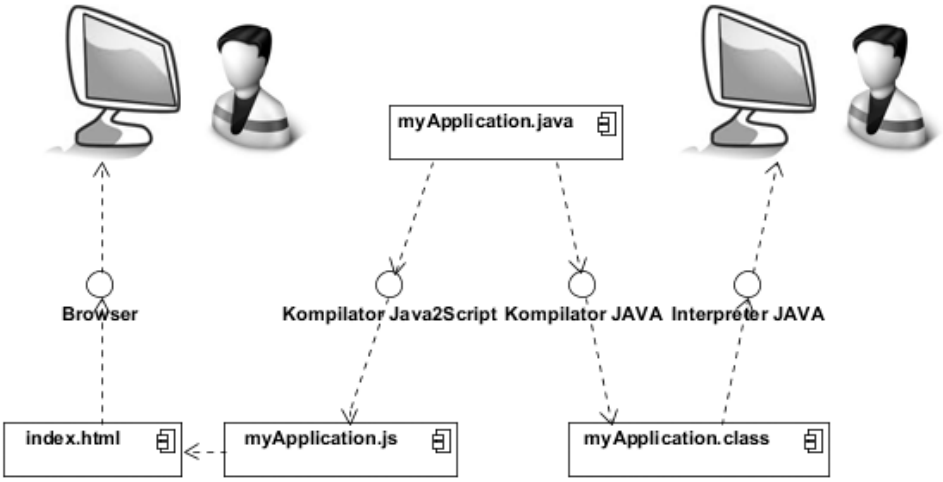
Głównym pomysłodawcą i konstruktorem wtyczki jest Zhou Renjian, który od 2005 roku starał się stworzyć kompilator, który mógłby skonwertować kod języka Java do JavaScript. Pomysł zrodził się dlatego, że praca z plikami źródłowymi JS bywa koszmarem, gdy osiągają one większy rozmiar, a także trudno nimi zarządzać, ze względu na brak profesjonalnych narzędzi, w porównaniu do Eclipse i Visual Studio. Brak jest też zaawansowanych narzędzi do testowania JavaScriptu, jak i jego debuggowania.

Główne funkcje i właściwości J2S:

- Polimorfizm,
- Dziedziczenie,
- Implementacja interfejsów,
- Zapewnia generowanie klas anonimowych,
- Zapewnia poprawne działanie zmiennych finalnych,
- Zapewnia poprawne działanie typów klas.

Zasadę działania wtyczki w porównaniu do normalnego kompilatora Java pokazuje rys. 3.1 Po zainstalowaniu J2S możemy wybrać, czy chcemy skompilować projekt używając standardowego kompilatora, czy kompilatora Java2Script. Gdy wybierzemy tę drugą opcję, J2S utworzy do każdej klasy `.java` plik `.js`. Zostanie też wygenerowany plik `.html` do każdej klasy,

posiadającej metodę `main()`. Do tego pliku dołączony zostaje powstały plik `.js`, który w swoim ciele może już odwoływać się do innych powstałych plików JavaScript. Plik w formacie HTML otwieramy w przeglądarce internetowej i korzystamy z funkcjonalnej aplikacji sieciowej jak ze strony internetowej.



Rys. 3.1: Zasada działania Java2Script

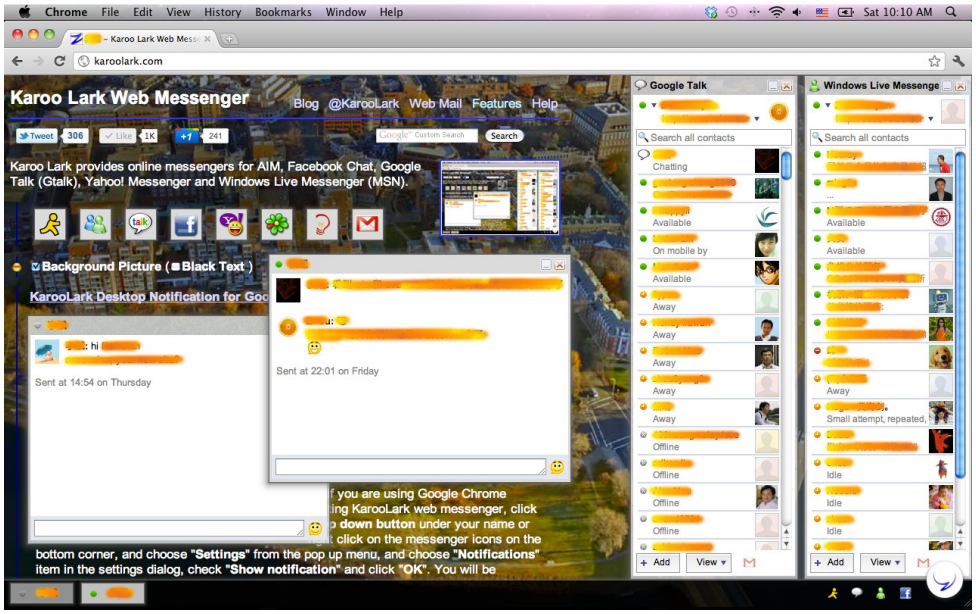
Twórca wtyczki ocenia, że dzięki środowisku Eclipse z dołączoną wtyczką J2S można poprawnie przetłumaczyć nawet 90% plików źródłowych Java. Niestety występują pewne rozbieżności, między tymi dwoma językami, których nie da się rozwiązać. Użytkownicy wtyczki muszą pamiętać, że pola statyczne klasy mogą nie kompilować się poprawnie, typy prymitywne: `char`, `byte`, `short`, `int`, `long`, `float`, `double` są traktowane jako `Number`, a także brak jest wsparcia dla wątków czy asynchroniczności.

J2S wspiera klasy wyłącznie z pakietów: `java.io`, `java.util`, `java.lang`, `java.lang.reflect`, `junit`, `org.eclipse.swt`. Trzeba jednak pamiętać, że niektóre klasy z wymienionych pakietów nie zostały zaimplementowane w Java2Script i nie można z nich skorzystać (TODO: odnośnik do pełnej listy wspieranych klas).

3.2. Analiza istniejących przypadków użycia

Karoo Lark Web Messenger

Serwis udostępnia wersje online słynnych komunikatorów, takich jak Facebook Chat, Google Talk czy Windows Live Messenger (MSN). Głównymi zaletami takiego rozwiązania są: łatwość wyboru komunikatorów, których chcemy używać oraz łatwe przełączanie się między nimi. Jednak aplikacja posiada również inną zaletę. Wyobraźmy sobie sytuację, gdy jesteśmy uczniem lub studentem i korzystamy z komputera należącego do



Rys. 3.2: Interfejs użytkownika aplikacji Karoo Lark Web Messenger

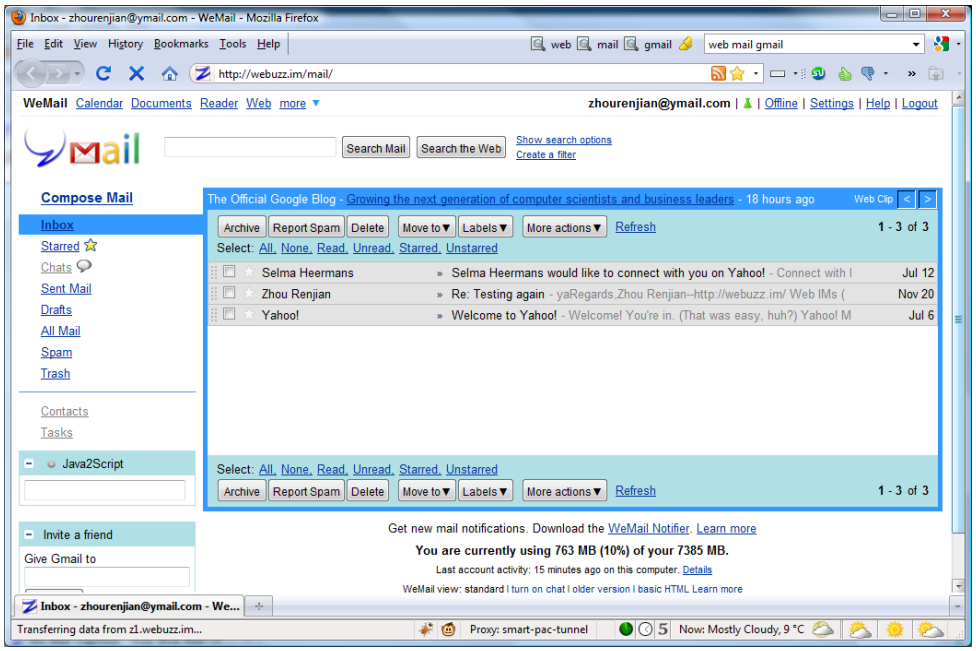
jednostki edukacyjnej. Niektóre z nich blokują usługi komunikatorów przy użyciu zapór ogniowych. Jednak zazwyczaj nie blokują one stron internetowych, takich jak karoolark.com. Dzięki temu serwisowi możemy bez obaw zalogować się do interesujących nas komunikatorów.

Lemon Dove Web Mail Client

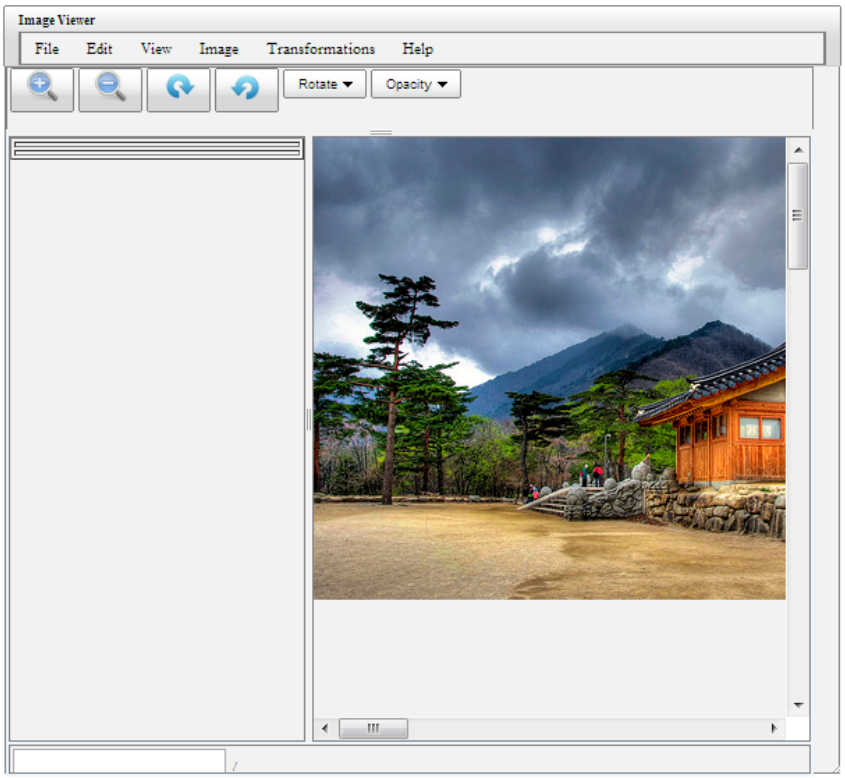
Serwis ten jest darmowym klientem poczty elektronicznej. Zapewnia połączenie z najpopularniejszymi skrzynkami pocztowymi, takimi jak Google Mail, Hotmail, Yahoo! Mail, czy AOL/AIM Mail. Główną zaletą tego rozwiązania jest to, że możemy zalogować się do konkretnej skrzynki pomimo zapory ogniowej, która może blokować taką usługę. Serwis zaprojektowany jest z myślą o uruchamianiu go na różnych systemach operacyjnych i w różnych sytuacjach - czy w domu, czy w biurze z włączoną zaporą. Zapewnia też dość intuicyjne korzystanie z aplikacji dla użytkowników poczty Gmail, ze względu na podobieństwo interfejsu.

Advanced Image Viewer

Aplikacja pozwala użytkownikowi załadować obraz z Internetu, obejrzeć i manipulować nim online. Wspierane są takie operacje, jak obrót, zbliżenie, import, eksport, zaawansowane filtry (np. rozmycie), wykrywanie krawędzi i wiele innych. Aplikacja przeznaczona jest dla użytkowników, którzy chcą w sposób zaawansowany manipulować danym obrazem w pełni po stronie klienta serwisu. W odróżnieniu do większości projektów z użyciem J2S, nie używa biblioteki SWT do wyświetlenia interfejsu. Używa natomiast YUI z frameworkiem Raphaël.



Rys. 3.3: Interfejs użytkownika aplikacji Lemon Dove Web Mail Client



Rys. 3.4: Interfejs użytkownika aplikacji Advanced Image Viewer

Rozdział 4

Opis implementacji

4.1. Pierwsza iteracja

1. Definiowanie projektu

   a) Etap inicjowania projektu

      Pierwsza wersja aplikacji miała dotyczyć implementacji graficznego interfejsu użytkownika aplikacji sieciowej, jaką był serwis semWeb, którego tematyką było wykorzystanie metadanych i sieci semantycznych do wyszukiwania, integracji i weryfikacji urzędowych danych posiadających aspekt czasowo-przestrzenny.

   b) Etap definiowania celu

      Celem projektu miało być stworzenie interfejsu aplikacji sieciowej, która posiadała już pełną funkcjonalność. Interfejs miał łączyć się z bazowym systemem, jakim był semWeb. Dostarczał on endpoint, umożliwiający zarządzanie dziedzinową bazą wiedzy. Serwis bazowy był oparty na fasadowym wzorcu projektowym, co było jednym z powodów, dla którego został on wybrany jako potencjalna baza, ze względu na łatwość integracji modułu interfejsu z resztą aplikacji.

      Potencjalnymi zagrożeniami były problemy z importem plików o rozszerzeniu `.jar` do projektu. Powodem tego jest obecna implementacja wtyczki Java2Script, która opiera się na wczytywaniu skompilowanych klas środowiska Java, ale tylko tych, które umieszczone są bezpośrednio w projekcie, lub należą do konkretnych, wbudowanych bibliotek, jak `java.lang` (//TODO: odnośnik do strony J2S).

   c) Etap weryfikacji

      Z uwagi na to, że nie dało się dołączyć plików `.jar` do projektu w taki sposób, by kompilator Java2Script mógł prawidłowo załadować klasy oraz skonwertować

je do plików z rozszeżeniem `.js`, projekt nie przeszedł pozytywnie przez fazę weryfikacji i został odrzucony.

Brak możliwości dołączania zewnętrznych archiwów .jar oraz wbudowanych bibliotek, takich jak `java.sql`, uniemożliwiło połączenie się z bazą danych. Konsekwencją tego był brak możliwości zbudowania aplikacji, która spełniałaby minimalne wymagania zawarte w etapie definiowania projektu. W związku z tym kolejnym krokiem w cyklu życia projektu było przejście do kolejnej iteracji i zdefiniowanie nowego problemu.

4.2. Druga iteracja

1. Definiowanie projektu

   a) Etap inicjowania projektu

   Druga wersja aplikacji miała dotyczyć implementacji graficznego interfejsu użytkownika aplikacji sieciowej, jaką był serwis EDAP, którego tematyką było tworzenie dokumentów prawnych, ich edycja oraz zarządzanie już istniejącymi dokumentami.

   Projekt ułatwia skonstruowanie wybranego dokumentu prawnego zgodnie ze schematem narzuconym przez MSWiA. Dokument generowany jest w postaci pliku o formacie XML, a wszelkie obostrzenia i zasady jego utworzenia zawiera plik `schema.xsd`. Używając serwisu EDAP, użytkownik może także wczytać już istniejący plik z rozszerzeniem `.xml` i zmodyfikować go.

   b) Etap definiowania celu

   Celem projektu było stworzenie aplikacji sieciowej, zawierającej główną funkcję serwisu EDAP, czyli konstruowanie dokumentu prawnego w postaci pliku XML na podstawie schematu zawartego w pliku `schema.xsd`.

   Aby osiągnąć cel główny, należało wykonać 3 podstawowe kroki: odpowiednio wczytać zawartość pliku ze schematem XML, następnie stworzyć interfejs użytkownika, który wyświetlałby dostępne pozycje do wypełnienia przez użytkownika, z których potem generowany byłby plik XML.

   Głównymi potencjalnymi problemami były operacje na plikach, takie jak wczytywanie i generowanie, które nie są wspierane przez Java2Script. Oprócz tego problemem mogło się okazać odpowiednie wczytanie zawartości schematu XML, gdyż nie można było skorzystać z zewnętrznych bibliotek, które by to ułatwiały. Powodem tego był brak możliwości dołączenia takich bibliotek w taki sposób, by Java2Script mógł je załadować.

c) Etap weryfikacji

Projekt przeszedł pozytywnie etap weryfikacji z uwagi na realną możliwość stworzenia aplikacji, której celem byłoby skonstruowanie dokumentu prawnego w postaci pliku XML.

Projekt zakłada jedynie generowanie takiego pliku, nie jego modyfikację. Kolejnym założeniem jest brak generowania samego pliku XML, a jedynie wyświetlenie jego zawartości w aplikacji. Oba założenia wynikają z braku możliwości operacji na plikach, które nie są wspierane przez wtyczkę Java2Script.

2. Planowanie

3. Wykonanie planu projektu

a) Organizacja kamieni milowych

b) Wprowadzenie planu w życie

c) Przetestowanie aplikacji

4. Zamykanie projektu

a) Ocena projektu

b) Wnioski

Rozdział 5

Testowanie

Rozdział 6

Podsumowanie

Literatura

Dodatek A

Instrukcja wdrożenia

Dodatek B

Instrukcja użytkownika