

**Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie**

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

KATEDRA AUTOMATYKI



PRACA MAGISTERSKA

ŁUKASZ ZIEŃKOWSKI

INTERAKTYWNA MAPA ŚWIATA Z DODATKOWĄ OSIĄ CZASU

PROMOTOR:
dr inż. Grzegorz Rogus

Kraków 2013

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIESZCZONE W PRACY.

.....

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

DEPARTMENT OF AUTOMATICS



MASTER OF SCIENCE THESIS

ŁUKASZ ZIEŃKOWSKI

INTERACTIVE WORLD MAP WITH ADDITIONAL TIMELINE

SUPERVISOR:
Grzegorz Rogus Ph.D

Krakow 2013

Serdecznie dziękuję mojemu promotorowi bez którego niniejsza pracy nie powstały, za pomoc podczas jej tworzenia, trafne uwagi i spostrzeżenia.

Spis treści

1. Wstęp.....	7
1.1. Temat pracy	7
1.2. Podstawowe cele.....	7
1.3. Zawartość pracy.....	7
2. Rys historyczny	9
3. Dostępne rozwiązania	10
3.1. Google Earth.....	10
3.2. Aplikacje wykonane w technologii Flash.....	12
3.3. Wykorzystanie JavaScript.....	12
3.4. Wybrane technologie	13
4. Ogólny opis rozwiązania.....	14
4.1. Cel projektu	14
4.2. Problemy do rozwiązania	14
4.3. Użytkownicy.....	15
4.4. Granice systemu	15
4.5. Lista możliwości.....	15
4.6. Specyfikacja wymagań	17
4.6.1. Ogólny diagram przypadków użycia	17
4.6.2. Definicje przypadków użycia.....	18
4.6.3. Wymagania niefunkcjonalne	20
5. Opis rozwiązania	21
5.1. Wykorzystane technologie.....	21
5.1.1. JavaScript	21
5.1.2. Możliwości HTML5	22
5.1.3. Less	23
5.2. Przechowywanie i transmisja danych.....	24
5.2.1. GIS	24
5.2.2. Format zapisu	24

5.2.3. Format pliku	25
5.2.4. Pliki graficzne	26
5.2.5. Parser plików.....	26
5.2.6. Storage	27
5.2.7. Transmisja danych.....	28
5.3. Wizualizacja	30
5.3.1. Rodzaje map.....	30
5.3.2. Metody prezentacji.....	31
5.3.3. Przybliżanie zbiorów danych.....	36
5.3.4. Oś czasu	37
5.4. Bezpieczeństwo danych i aplikacji.....	39
5.4.1. Funkcje haszujące	39
5.5. Optymalizacja rozwiązania	41
5.5.1. Wydajność parsera	41
5.5.2. Reprezentacja graficzna	41
6. Implementacja.....	43
6.1. Struktura projektu	43
6.2. Minimalna konfiguracja.....	45
6.3. Interfery do map.....	46
6.4. Odczyt danych	46
6.4.1. Dodawanie filtrów	51
6.4.2. SVG.....	52
6.4.3. Zdjęcia.....	54
6.5. Krzywa Bezier-a	54
7. Przykład użycia	56
7.1. Historia USA	56
7.2. Granice Polski i Rzeczypospolitej Obojga Narodów	57
8. Podsumowanie	59
A. Dodatek A	61
A.1. Oryginalny plik kml.....	61
A.2. Zapis w pamięci sesyjnej.....	62
A.3. Wynik końcowy	63

1. Wstęp

1.1. Temat pracy

Celem pracy jest rozszerzenie tradycyjnych dwuwymiarowych obrazów kartograficznych, map o dodatkowy wymiar jakim jest czas. Statyczny obraz zostanie rozszerzony o możliwość zmiany prezentowanych informacji w zależności od wybranego okresu. Zabieg ten ma na celu dostarczenie możliwości obserwacji zachodzących zmian pozwalając na lepszą wizualizację i przyswajanie wiedzy.

Większość aktualnie dostępnych map stanowi stałą reprezentację określającą stan na danym terenie w ścisłe określonym czasie w historii . W tradycyjnej, papierowej wersji nie jest możliwa żadna interakcja z danymi. Dzięki użyciu komputerów osiągalne jest umieszczenie większej ilości danych przy jednoczesnym zwiększeniu ich dokładności na jednym obiekcie. Dotychczasowa mapa zawierająca zbiór kilku linii określających granice terytorialne kraju znane z lekcji historii, może teraz zamienić się w płynną animację ukazującą dokładne zmiany w każdym momencie dziejów.

1.2. Podstawowe cele

1. Analiza dostępnych rozwiązań interaktywnych map
2. Przegląd technologii i analiza problemów
 - (a) Wybór narzędzi
 - (b) Rozwiązywanie wyzwań stawianych przez:
 - Sposób zapisu i przechowywania danych
 - Wizualizację różnego typu danych
 - Bezpieczeństwo i wiarygoność danych
 - Optymalizację i zapewnienie płynności działania
3. Implementacja zaprojektowanej aplikacji

1.3. Zawartość pracy

Pracę rozpoczyna Wstęp, w którym to został zawarty ogólny cel projektu. Posiada on również listę zadań które zostały opracowane i zrealizowane w trakcie jego tworzenia.

Drugi rozdział - Rys historyczny - zawiera krótką analizę dziejów i ewolucji map, od prymitywnych rysunków na ścianach aż po czasy współczesne.

Następny rozdział - Dostępne Rozwiązań - jest próbą analizy gotowych rozwiązań dostarczających połączenie statycznego obrazu i osi czasu. Jego zadaniem jest zapoznanie się z technologiami wykorzystywanymi w tego typu projektach i wybór najbardziej adekwatnych.

Rozdział - Ogólny opis rozwiązania - ma na celu przedstawienie biznesowej analizy, ogólny opis funkcjonalności które ma spełniać końcowy program. Przedstawienie granic systemu, jakie funkcje zawierają się w jego zakresie i co można przy jego pomocy wykonać.

Główna część - Opis rozwiązania - jest sprawozdaniem z prac jakie zostały przeprowadzone w procesie rozwiązywania napotkanych problemów. Przedstawia m.in. sposób przechowywania danych umożliwiający na błyskawiczne wykorzystanie aplikacji, ich prezentację z zależnością od ich typu i ustawień programu.

Następnie zawarty rozdział - Implementacja - to fragmenty kodu źródłowego który został stworzony podczas fazy tworzenia końcowego programu. Zawarte sekcje odnoszą się do najbardziej istotnych fragmentów kodu wraz z ich omówieniem.

Przedostani - Przykład użycia - przedstawia końcowy wynik otrzymyany w procesie analizy danych wejściowych przez stworzony framework.

Ostatni rozdział - Podsumowanie - ma on na celu analizę dokonać wykonanych w trakcie tworzenia omawianej pracy.

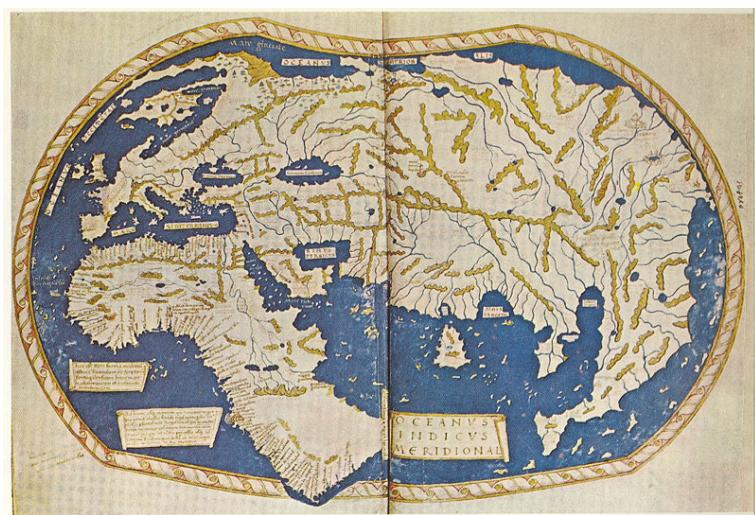
2. Rys historyczny

Pomimo że w obecnych czasach codziennie mamy kontakt z mapami i uznajemy je za normalny przedmiot codziennego użytku to nie zawsze tak było. W historii człowieka można wskazać okresy czasu kiedy kartografia była nieznana, a świadomość człowieka o otaczającym go świecie ograniczała się jedynie do tego co mógł zobaczyć przy pomocy swoich oczu.

Pierwsze malowidła które można uznać za graficzną reprezentację otoczenia archeolodzy natknęli w okolicach Pavloc (Czechy), datowaną są one na 25 wiek przed naszą erą [Lam09]. Na 14 wiek p.n.e datowane są wykopaliska w Navarre(Hiszpania) obejmujące rysunki na piaskowcu [Cho09].

Jeszcze w XIV w. kartografia była bardzo ograniczona, mapy którymi posługiwano się nie były zbyt dokładne. 12 października 1410 gdy Krzysztof Kolumb dotarł do brzegów wyspy San Salvador uznał ją za jedną z wysp japońskich [Irv28]. Aby lepiej zrozumieć powód tej pomyłki wystarczy spojrzeć na mapę stworzoną w XV wieku przez kartografa Henricusa Martellusa 2.1. Widzimy na niej że m.in. obie ameryki nie były znane ówczesnym ludziom.

Obecny rozwój technologiczny pozwala na stworzenie dokładnej mapy całego świata z jego prawidłowymi wymiarami nie tylko w postaci szkiców ale jako zbiór zdjęć dzięki czemu dostarcza ona znacząco więcej danych.



Rysunek 2.1: Mapa świata z 1489 roku.

3. Dostępne rozwiązania

Przed rozpoczęciem implementacji własnego rozwiązania postanowiono przeprowadzić badania mające na celu analizę aktualnie dostępnych rozwiązań spełniających przynajmniej w części założenia. Proces ten nie ma na celu znalezienie idealnego rozwiązania lecz poznanie różnych podejść do zagadnienia. Analiza różnych technologii które mogą być wykorzystane pozwoli na świadomie wybranie najlepszych.

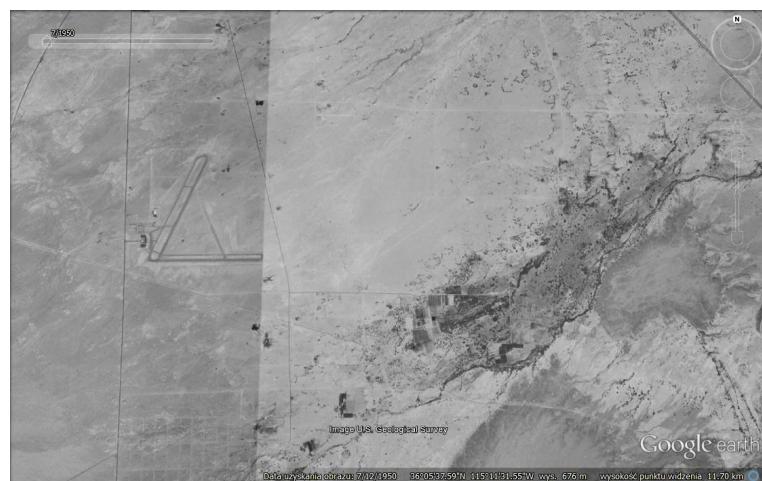
3.1. Google Earth

Funkcjonalność interaktywnych map można znaleźć w aplikacji stworzonej przez firmę Google. Program ten jest napisany przy użyciu języka C++, nie służy on do tworzenia aplikacji mobilnych, dlatego nie został on wybrany jako kandydat do tworzonej pracy.

Aplikacja wykorzystuje statyczne obrazy będące zdjęciami satelitarnymi w przypadku obrazów z dużej wysokości lub zdjęciami wykonanymi z pokładów samolotów. Pomimo dużej atrakcyjności ma ono bardzo ograniczoną możliwość zmiany oglądanych danych, ograniczone do ilości wykonanych zdjęć, dodatkowo większość danych pochodzi z okresu ostatnich 60 lat.

Przykład takiej sytuacji został przedstawiony na rysunku 3.1, obraz terenu na którym powstanie miasto Las Vegas w roku 1950. Jak teren ten wyglądał w roku 1977 widzimy na rysunku 3.2, pomimo widocznych zmian teren ten nadal w dużym stopniu jest pustynny, dopiero na rysunku 3.3 widzimy aktualny stan miasta.

Dzięki funkcji zmiany punktu i kąta patrzenia, pokazywania ciekawych miejsc czy chociażby włączania trybu w którym budynki nabierają formy przestrzennej, 3D, możemy poprzez zabawę i wirtualne wycieczki poszerzać naszą wiedzę o otaczającym nas świecie.



Rysunek 3.1: Las Vegas w 1950 roku.



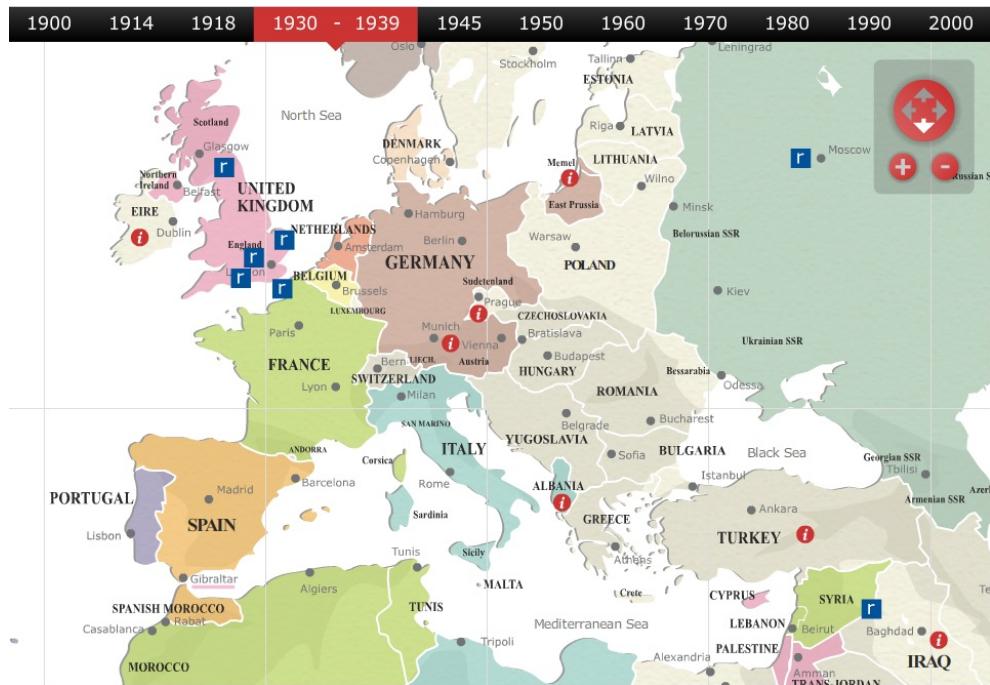
Rysunek 3.2: Las Vegas w 1977 roku.



Rysunek 3.3: Las Vegas w 2012 roku.

3.2. Aplikacje wykonane w technologii Flash

Podczas prac badawczych natrafiono na kilka aplikacji których główna część aplikacji została wykonana w technologii flash. Przykładami są interaktywna mapa historii europy [wor13] czy mapa prezentująca zmianę granic państw w XX wieku [fla13b] przykład na rysunku 3.4. Oba projekty zawierają bogatą szatę graficzną, ich wykonanie jest bardzo estetyczne. Zaletą tworzenia programów w tej technologii jest wykorzystanie mocy obliczeniowej karty graficznej do wyświetlania obrazów(dostępne od wersji 10.1 [fla13a]), dzięki temu procesor jest może wykonywać inne zadania.



Rysunek 3.4: Europa w latach 30.

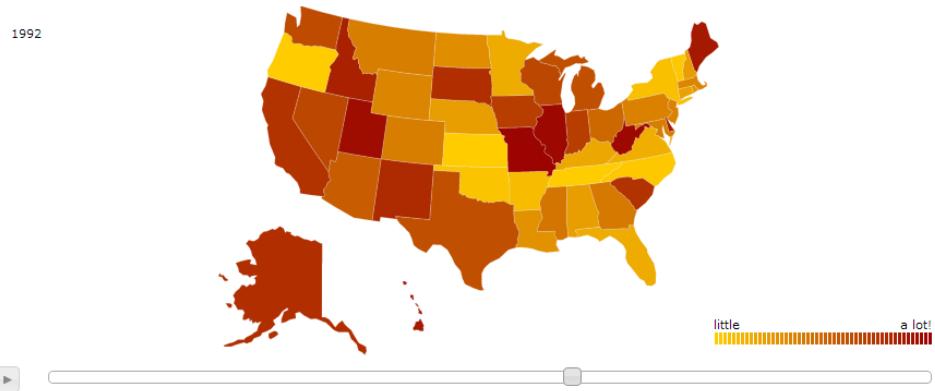
Wykorzystanie języka ActionScript wydaje się dobrym wyborem, niestety posiada kilka wad które go eliminują. Pierwszą z nich jest brak wsparcia technicznego dla wszystkich urządzeń mobilnych, przez co jej wykorzystanie ogranicza rynek docelowy. Wymagana jest instalacja detykowanego odtwarzacza który pozwala na korzystanie z programów stworzonych w tym języku. Ponieważ tworzona aplikacja jest jednym komponentem wymagane jest całkowite ściągnięcie z serwera na lokalny komputer aby można było z niej korzystać. Ostatnim aspektem jest brak kompatybilności elementów flash-owych z głównymi wyszukiwarek internetowymi, ich indeksacja jest utrudniona(możliwe jest jedynie wykorzystanie tagów) a ich zawartość całkowicie niewidoczna w procesie indeksowania. Z uwagi na powyższe wady technologia nie została wykorzystana w omawianej pracy.

3.3. Wykorzystanie JavaScript

Część aplikacji pracujących na danych geograficznych jest tworzona przy użyciu skryptowego języka jakim jest JS. Przykład zastosowania jest widoczny w bibliotece “maMap” [jsm13]. Pozwala ona na

proste pisanie rozwiązań w których zapewniona interakcja z użytkownikiem podnosi wartość programu. Przykład mapy z osią czasu przedstawiony na rysunku 3.5 wskazuje rozkład średniej rocznej temperatury w stanach ameryki północnej na w przekątne ostatnich 50 lat.

► Heat map timeline with jQuery UI Slider



Rysunek 3.5: Mapa ciepla w USA.

3.4. Wybrane technologie

Po analizie dostępnych rozwiązań zdecydowano się na wykorzystanie w pracy następujących technologii.

- HTML 5 - język HTML jest budulcem każdej strony internetowej, w wersji piątej zostały dodane rozwiązania które zwiększają jego możliwości
- JavaScript - umożliwia interakcję użytkownika ze stroną internetową i zmiany konkentu bez konieczności odświeżania całej zawartości
- Less - agrreguje i porządkuje style CSS

4. Ogólny opis rozwiązania

4.1. Cel projektu

Celem projektu jest stworzenie aplikacji internetowej dodającej do tradycyjnych map dodatkowego wymiaru, czasu. Powinna dostarczać możliwości obserwowania zmian zachodzących na określonym terenie w danym okresie. Dodana oś czasu będzie miała za zadanie kontrolowanie okresu dla którego widoczne będą dane. Dostarczone informacje przez użytkownika będą przedstawiane w czytelny i umożliwiający szybką analizę sposób. Przykładowe dane wejściowe to zmiany terytorialne w przeszłości, ukażanie rozwoju miast. Końcowa wizualizacja danych oprócz prostych kształtów geometrycznych musi zapewniać obsługę zewnętrznych plików graficznych jak i najnowszych rozwiązań służących do pracy z grafiką.

4.2. Problemy do rozwiązania

– Przechowywanie i transmisja danych

Nie jest określony jeden format danych który będzie używany w aplikacji. Oprócz informacji opisujących położenie punktów zawarte również będą szczegółowe dane dotyczące ich wyglądu i zachowania, również obrazy graficzne mogą się znajdować w opisie mapy.

– Wizualizacja

Szeroki zakres informacji który może opisywać mapę musi być odpowiednio wyświetlany, w sposób czytelny i ułatwiający analizę dużych zbiorów danych. W jednym miejscu zostaną zebrane różnego rodzaju dane które muszą współpracować ze sobą tworząc spójną kompozycję.

– Bezpieczeństwo

Dane dostarczane przez zewnętrzny serwer powinny być wiarygodne. Użytkownik posiadający kopię pliku znajdującą się na dysku lokalnym z informacjami o mapie musi mieć możliwość sprawdzenia jej autentyczności z aktualną wersją na serwerze.

– Optymalizacja rozwiązania

Aplikacja musi w każdym momencie zapewniać płynną pracę. Praca ze szczegółową mapą posiadającą wiele informacji musi przebiegać płynnie, bez dużych opóźnień.

4.3. Użytkownicy

- Administrator

Osoba która posiada dostęp do serwera na którym działa aplikacja korzystająca z omawianego frameworku. Ponieważ tworzony program jest jedynie narzędziem służącym do pracy z mapami, a nie kompletną witryną internetową wymagana jest jego integracja z zewnętrznym rozwiązaniem.

- Moderator

Pzygotowanie i edycja map powinna być wykonana przez uprawnione osoby. Może to być na przykład osoba posiadającą dużą wiedzę na dany temat, posiada ona możliwość edycji mapy.

- Uczeń

Użytkownik z najmniejszymi uprawnieniami, jedynie do przeglądania mapy.

4.4. Granice systemu

System umożliwia moderatorowi:

- Tworzenie i edycję zestawów danych przechowywanych na serwerze
- Tworzenie własnych filtrów edytujących mapę w zależności od zmiany położenia lub czasu
- Zapis informacji o mapie do pliku na dysku lokalnym
- Zmiana typu mapy

System umożliwia uczniowi:

- Przeglądanie gotowych map
- Pracę bez konieczności stałego dostępu do internetu.
- Sprawdzanie autentyczności pliku

System nie umożliwia:

- Wykorzystywania plików video

4.5. Lista możliwości

Praca z mapami w trybie offline

- Wczytywanie pliku - dane używane w aplikacji mogą zostać wczytane z dwóch źródeł, z lokalnego dysku lub zimportowane z serwera.
- Przechowywanie danych po stronie klienta

Szeroki wachlarz dostępnych sposobów wizualizacji danych

- Korzystanie z grafiki wektorowej
- Współpraca z plikami graficznymi
- Obsługa animacji poprzez svg

Współpraca z zewnętrzny aplikacjami

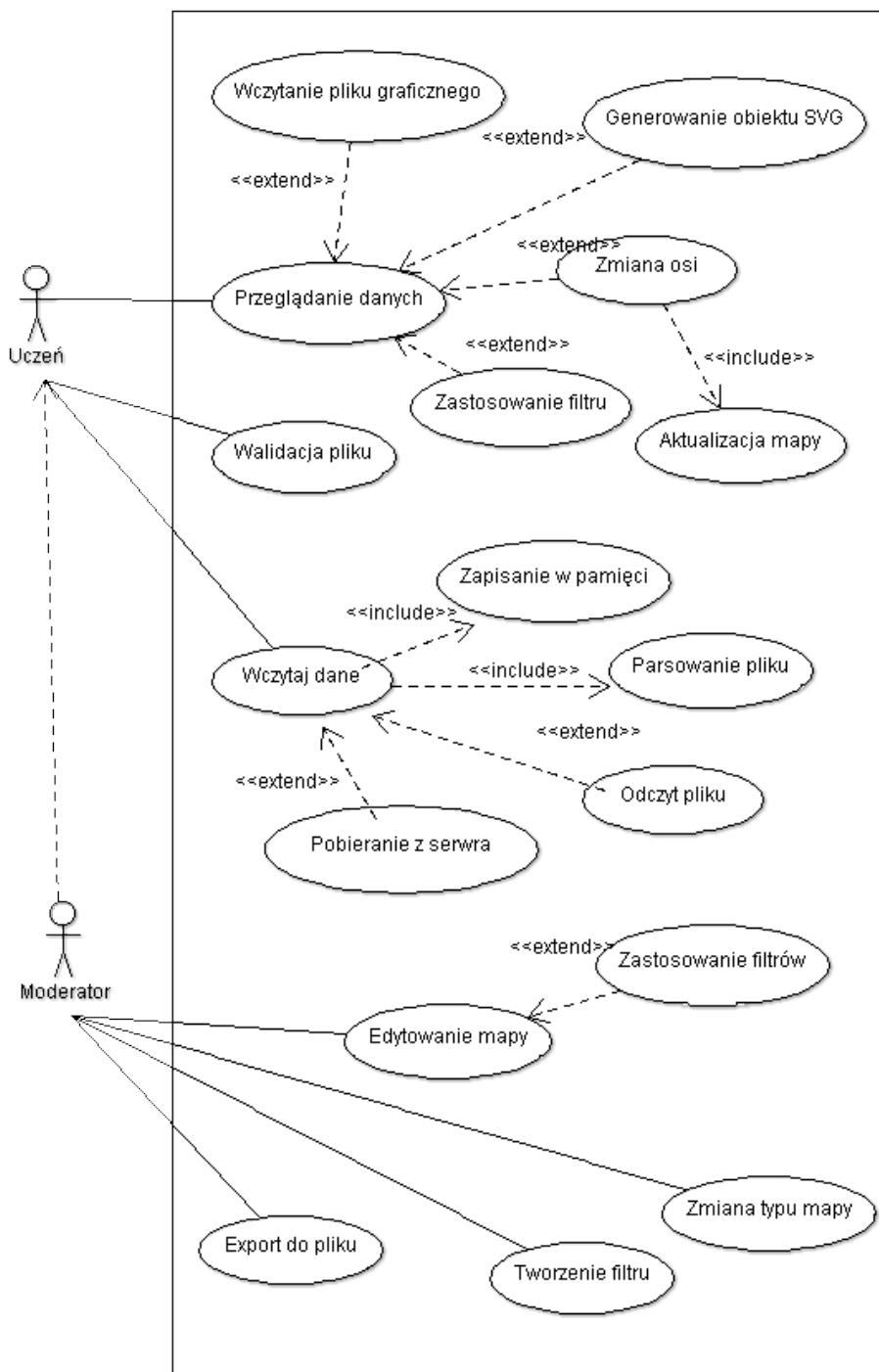
- Obsługa plików GML
- Parsowanie danych z formatu xml do pamięci storage

Tworzenie interaktywnych filtrów

- Filtry obsługujące zmianę czasu
- Filtry obsługujące zmianę wysokości nad ziemią

4.6. Specyfikacja wymagań

4.6.1. Ogólny diagram przypadków użycia



Rysunek 4.1: Diagram przypadków użycia.

4.6.2. Definicje przypadków użycia

Poniżej znajduje się opis trzech podstawowych przypadków użycia występujących w aplikacji. Opisane zostały warunki rozpoczęcia każdego z nich i zakończenia, zarówno pozytywnego jak i negatywnego.

Nazwa przypadku użycia	Wczytanie danych z pliku
Typ przypadku użycia	Ogólny
Aktorzy	Uczeń
Warunki wstępne	Brak
Warunki końcowe dla sukcesu	Dane zostają zapisane w pamięci lokalnej przeglądarki
Warunki końcowe dla niepowodzenia	Dane nie są poprawnie odczytane.
Scenariusz główny	<ol style="list-style-type: none">1. Użytkownik wybiera plik.2. Dane podlegają parsowaniu.3. Poprawnie odczytane dane zostają zapisane w Storage.
Scenariusz alternatywny	<ol style="list-style-type: none">1. Użytkownik wybiera plik.2. Dane podlegają parsowaniu.3. Dane nie zostają odczytane poprawnie.4. Użytkownik zostaje poinformowany o zauważonym błędzie.

Tablica 4.1: Przypadek wczytania danych

Nazwa przypadku użycia	Wyświetlenie danych
Typ przypadku użycia	Ogólny
Aktorzy	Uczeń
Warunki wstępne	Dane znajdują się w pamięci lokalnej
Warunki końcowe dla sukcesu	Dane zostają wyświetcone
Warunki końcowe dla niepowodzenia	Mapa nie jest aktualizowana
Scenariusz główny	<ol style="list-style-type: none"> 1. Użytkownik wybiera punkt w czasie. 2. Dane dla wybranego okresu zostają wyrenderowane i pokazane. 3. Informacje dla najbliższego otoczenia podlegają generacji i ukryciu.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1. Użytkownik wybiera punkt w czasie. 2. Dane dla wybranego okresu nie zostają odnalezione 3. Nie aktualne dane zostają schowane.

Tablica 4.2: Przypadek wyświetlania danych

Nazwa przypadku użycia	Walidacja autentyczności pliku
Typ przypadku użycia	Ogólny
Aktorzy	Uczeń
Warunki wstępne	Uczeń posiada plik na dysku i jego sumę kontrolną z serwera
Warunki końcowe dla sukcesu	Pojawienie się komunikatu o autentyczności pliku
Warunki końcowe dla niepowodzenia	Ostrzeżenie o dokonaniu zmian w pliku
Scenariusz główny	<ol style="list-style-type: none"> 1. Użytkownik wybiera zakładkę do walidacji pliku. 2. Podanie ścieżki do pliku i sumy kontrolnej. 3. Obliczenie aktualnej sumy dla pliku i porównanie go z podanym. 4. Użytkownik zostaje poinformowany o edycji pliku.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1. Użytkownik wybiera zakładkę do walidacji pliku. 2. Podanie ścieżki do pliku i sumy kontrolnej. 3. Obliczenie aktualnej sumy dla pliku i porównanie go z podanym. 4. Użytkownik zostaje poinformowany o zmianach w pliku.

Tablica 4.3: Przypadek kontroli pliku

4.6.3. Wymagania niefunkcjonalne

Oprócz wymagań dotyczących prawidłowego funkcjonowania aplikacji, wymagane jest aby zapewnione były poniższe punkty.

- System powinien być skalowalny, powinien umożliwiać dostęp wielu użytkownikom równocześnie przy zachowaniu wymagań wydajnościowych.
- Aplikacja powinna działać identycznie na różnych środowiskach i przeglądarkach internetowych.

5. Opis rozwiązania

Poniższy rozdział zawiera omówienie technologii które zostały wykorzystane w procesie tworzenia aplikacji, ze szczególnym uwzględnieniem funkcjonalności które okazały się najbardziej przydatne. Następnie zawarto szczegółowy opis sposobów rozwiązania problemów które należało rozwiązać w trakcie tworzenia interaktywnej mapy.

5.1. Wykorzystane technologie

5.1.1. JavaScript

Wybór języka JavaScript do tworzenia aplikacji został podkutowany jego dużymi możliwościami na polu tworzenia projektów do użytku na przeglądarkach internetowych. Pozwala na łatwe aktualizowanie jedynie wybranych danych na ekranie monitrona bez konieczności przeładowania całej strony. Jest to szczególnie istotne w omawianym przypadku, oznacza to brak potrzeby wczytywania i renderowania całego tła mapy za każdym razem gdy nastąpi nawet mała zmiana.

Bardzo ważną cechą języka jest fakt jego wykonywania po stronie klienta. Oznacza to wykonywanie obliczeń i korzystanie z pamięci znajdującej się na komputerze osoby która korzysta z aplikacji. Pozwala to na zmniejszenie wymagań w stosunku do serwera (wykonuje on mniej operacji i zapisuje mniej danych w swojej pamięci)

Obecnie powstają frameworki których zadaniem jest stworzenie małej, lekkiej aplikacji webowej wykorzystującej w maksymalny sposób omawiany język.[BG13] Przykładem takiego rozwiązania jest AngularJs, korzysta on z architektury MVC w której większość pracy została przeniesiona na stronę klienta. Kontroler wykonuje obliczenia w przeglądarce użytkownika, odciąża to główny serwer, pozwala na szybszą i płynniejszą pracę większej ilości osób.

Nie wątpliwą zaletą takiego podejścia jest wzrost odporności na ataki typu Dos(en. Denial of Service), polega on na przeciążeniu aplikacji dostarczającej określone dane lub usługi do momentu gdy przestaje ona odpowiadać na jakiekolwiek zapytania. Aplikacje które wykonują większość operacji po stronie serwera(różnego typu obliczenia matematyczne, analizę danych) aby sprostać dużej ilości klientów która występuje w tego typuatakach nakłada duże wymagania w stosunku do wykorzystywanej sprzętu elektronicznego. Przeniesienie ciężaru z serwera na końcowego klienta wykonywania większości operacji pozwala nie tylko na obsłużenie większej ilości użytkowników ale także może wpływać na zmniejszenie kosztów utrzymania serwera.

Prosty przykład na stronie <http://angularjs.org/> prezentuje w jak prosty sposób można stworzyć prostą listę rzeczy do zrobienia. Poniżej zaprezentowano fragment odpowiedzialny za wyświetlenie zadań z dodatkowym polem określającym czy zostało już wykonane.

```
1      //fragment kodu html
2      <div ng-controller="TodoCtrl">
3          ...
4          <li ng-repeat="todo in todos">
5              <input type="checkbox" ng-model="todo.done">
6              <span>{{todo.text}}</span>
7          </li>
8          ...
9      </div>
10
11
12     //kontroler
13     function TodoCtrl($scope) {
14         $scope.todos = [
15             {text:'learn angular', done:true},
16             ...
17         }
```

Listing 5.1: AngularJs

5.1.2. Możliwości HTML5

HyperText Markup Language, hipertekstowy język znaczników, pozwala na opisanie struktury informacji zawartych na stronie internetowej, to dzięki niemu przeglądarka może rozróżnić takie elementy jak hiperłącze, akapit czy chociażby nagłówek.

Podobnie jak w przypadku XML, tak i tutaj wymagane jest aby wykorzystywane znaczniki umieszczone były w nawiasach ostrokańnych a każdy z nich miał swoje domknięcie.

Poprawnym zapisem jest `<p>Wiadomość<\p>` który oznacza pojedyńczy akapit. Zapis `<p>Wiadomość<p>`, który różni się od poprzedniego brakiem znaku "\" w drugim znaczniku czyni ten zapis niepoprawnym. Istnieje możliwość aby wykorzystać pojedyńczy znacznik, przykładem jest `
` określający wstawienie nowej linii w miejscu wystąpienia tagu.

Obecnie powszechnie używany standart HTML w wersji czwartej ma wiele ograniczeń, z tego powodu pracowano nad jego następcą. 22 stycznia 2008 W3C opublikował HTML5, następca posiada wiele dodatkowych rozwiązań pomocnych w pracy. Posiada dodatkowe znaczniki ułatwiające pracę takie jak audio, video przechowywujące odpowiednio pliki muzyczne i wideo. Może dokonywać wstępnej walidacji formularzy, nie tylko sprawdzać obecność danych wejściowych ale także ich zgodność z wyrażeniami regularnymi. Umożliwia przechowywanie danych nie tylko w postaci ciasteczek ale większych struktur nazywanych Storage, dokładny opis tej funkcjonalności został zawarty w sekcji 5.2.6.

5.1.3. Less

Do stworzenia bardziej zaawansowanych arkuszy stylów CSS (en. Cascading Style Sheets) wykorzystano narzędzie które pozwala na łatwe tworzenie i utrzymywanie reguł którym jest Less, jego głównymi zaletami są:

- Deklarowanie zmiennych.

Jeśli chcemy aby jedna wartość(np. kolor) była wykorzystana w kilku miejscach możemy zadeklarować ją w głównym pliku a następnie używać zmiennej zamiast wartości, pozwala to na szybką i bezproblemową edycję.

Chcąc wykorzystać kolor czerwony zawarty w kolorystyce uczelni AGH deklarujemy zmienną @aghRed, następnie w miejscu gdzie chcemy go wykorzystać używamy istniejącą zmienną

```
1 @aghRed: #a71930;
2
3 bgcolor: @aghRed;
```

Listing 5.2: Deklaracja zmiennych

- Mixiny

Możliwe jest tworzenie klas posiadających właściwości które będą określone w innej, wcześniej zadeklarowanej. Pozwala to na tworzenie kodu bardziej czytelnego, unikanie powtórzeń.

```
1 .RoundBorders {
2   border-radius: 5px;
3   -moz-border-radius: 5px;
4   -webkit-border-radius: 5px;
5 }
6 #menu {
7   color: gray;
8   .RoundBorders;
9 }
```

Listing 5.3: Mixin

- Osadzanie elementów według dziedziczenia

Tworząc zagnieżdzoną strukturę strony często elementy wewnętrzne zależą od elementów nadrzędnych. Wygląd komórki może się różnić w zależności od miejsca w tabeli. W tworzeniu tak zagnieżdzonych struktur pomocna okazuje się omawiana cecha.

5.2. Przechowywanie i transmisja danych

5.2.1. GIS

GIS(Geographical Information Systems) jest to System Informacji Geograficznych, stworzony w celu dokonywania pomiarów, analizy i przedstawiania danych związanych z informacjami geograficznymi. Dane są pobieranie w trakcie naziemnych pomiarów jak i zbierane przy użyciu sztucznych satelitów. Informacje te są następnie zapisywane do określonych formatów dzięki czemu możliwe jest ich dalsze przetwarzanie.

Wykorzystywany układ współrzędnych

Istnieje wiele układów współrzędnych które służą do opisu pojedyńczego punktu na powierzchni ziemi. Na stronie <http://www.spatialreference.org/ref/epsg/> (dostęp 20.11.2013) znajduje się lista zawierająca przykłady zapisu punktu na kuli ziemskiej przy użyciu różnych formataw, zawierają one niezbędne infomacje do poprawnego zapisu danych w określonym formacie.

Tradycyjny zapis oparty jest na wykorzystaniu stopni, minut i sekund, jest on obecnie wypierany przez nowszy, łatwiejszy do obliczeń komputerowych. Wersją która ma za zadanie być ogólnoświatowym formatem jest WGS(World Geodetic System). Jego ostatnia wersja WSG84 jest powszechnie używana w urządzeniach do nawigacji. Poniżej zaprezentowano jak wygląda zapis w starszym i nowszym punktu określającego położenie budynku A-0 uczelni AGH.

– Pierwotny zapis

50°03'52.2803", 19°55'23.7968"

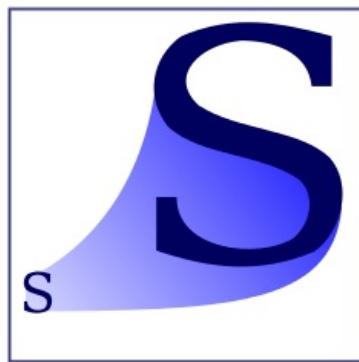
– Zapis unowocześniony

50.06452231874906, 19.923276901245117

Drugi zapis został wykorzystany do przechowywania danych w plikach, tak aby wymiana i wspólna praca była jak najprostsza. Wybór ten sprawił że nie występuje problem konwersji punktu do formatu czytelnego dla komputera, można go bez problemu odczytać jako liczbę.

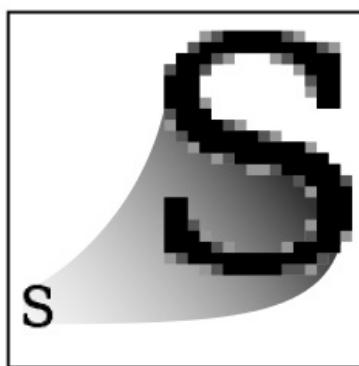
5.2.2. Format zapisu

W celu przechowywania obiektów graficznych zdecydowano się na wykorzystanie grafiki wektorowej. Oprócz bezproblemowej skalowalności, pozwala ona również na szeroką i łatwą edycję. Obraz jest w tym przypadku przechowywany jako zbiór formuł matematycznych m.in. proste i łuki. Dzięki takiemu podejściu obraz w dowolnej skali na identyczną jakość, widać to na rysunku 5.1. Przykładowymi formatami są GML(Geography Markup Language) otwarty standart XML dla wymiany danych GIS, Shapefile czy tradycyjny kartezjański układ współrzędnych.



Rysunek 5.1: Zapis wektorowy

Przeciwnym podejściem jest grafika rastrowa, jest to zapis punktów o określonym kolorze i położeniu na płaszczyźnie. Efektem takiego podejścia jest pogorszenie obrazu w momencie dużego powiększenia. Przykładem są formaty takie jak ADRG,ASC czy RGB. Porównanie litery Ś"i jej 7-krotnego powiększenia w tym zapisie zaprezentowano na rysunku 5.2



Rysunek 5.2: Zapis rastrowy

5.2.3. Format pliku

W celu wypełnienia założeń projektu jakim jest umożliwienie przechowywania danych w zewnętrznym pliku przeprowadzono analizę dostępnych formatów zapisu danych kartograficznych.

- SWDE(Standard Wymiany Danych Ewidencyjnych) - polski format służący głównie do wymiany danych ewidencyjnych gruntów i budynków
- SWING(Standard Wymiany INformacji Geodezyjnych) - format danych geodezyjnych służący do wymiany danych pomiędzy bazami danych systemów informatycznych SIT(wikipedia.pl)
- KML(Keyhole Markup Language) - otwarty standart oparty na XML-u, stworzony głównie dla potrzeb GoogleEarth
- GML(Geography Markup Language) - stworzony przez OGC(Open Geospatial Consortium) międzynarodową organizację założoną w celu tworzenia i rozpowszechniania standarów związanych z informacjami dotyczącymi geograficznych i geoprzestrzennych danych

Dwa polskie formaty są stosunkowo mało popularne, są one głównie wykorzystywane w celu komunikacji pomiędzy rządowymi bazami danych. Zawierają one bardzo określone typy danych co sprawia że są mało przydatne dla niniejszego projektu. KML jest formatem który posiada szeroki wachlarz dostępnych znaczników a jego duża popularność dostarcza dużą bazę plików gotowych do wykorzystania. Niestety został on stworzony by współpracować z określonym programem, zaimplementowane zostały jedynie elementy które mogą zostać wykorzystane w nim. Ostatni przykład został stworzony w uniwersalnym celu do ogólnego użytku, posiada on bardzo dużo możliwości, adaptacji do konkretnych rozwiązań [SY08] z uwagi na ten fakt zdecydowano się na wybór tego rozwiązania. Nie zostaną wykorzystane jego pełne możliwości, jedynie te które są przydatne w projektowanej aplikacji. Jedną z ciekawych i przydatnych możliwości podczas jego wykorzystania jest zgodność z formatem KML.

5.2.4. Pliki graficzne

Obrazy graficzne które mogą wystąpić w zbiorze danych wejściowych zdecydowano się przechowywać w postaci osobnych plików, nie są one załączane w głównym. Są to pliki które mogą stanowić alternatywne tło mapy, lub fragment wzbogaczający wizualnie dane.

Chcąc wykorzystać zasób zapisany w formacie jpg lub png należy w odpowiadającym znaczniku podać jego ścieżkę względową lub bezwzględną w przypadku gdy znajduje się na komputerze użytkownika lub adres internetowy pod którym się znajduje.

Zaletami takiego podejścia jest:

- Łatwa reużywalność zasobów graficznych, jeden plik może być wykorzystany przez kilka map bez potrzeby jego powielania.
- Dzięki separacji fragmentów alfanumerycznych od binarnych zachowana jest maksymalna czytelność danych.

5.2.5. Parser plików

Założenia projektu zakładają dostarczanie wstępnego zestawu danych opisujących mapę poprzez plik tekstowy zawierający dane w określonym formacie. Może on znajdować się na zewnętrznym serwerze lub dysku lokalnym. W pierwszym przypadku zostanie on najpierw ściagnięty do pamięci lokalnej aby następnie mógł być wykorzystany, w drugim po podaniu ścieżki dostępu do niego od razu nastąpi jego analiza. Aby było to możliwe stworzony został parser którego zadaniem jest odczytanie danych i przygotowanie ich aby możliwe było ich wykorzystanie w dalszej pracy aplikacji.

Wybrany format danych wybrany podczas prac badawczych, GML, zawiera bardzo obszerny zbiór elementów pozwalających opisywać bardzo szeroki wachlarz obiektów, takich jak mosty, budynki pojazdy. Taka dokładność nie jest wymagana w tym przypadku z tego powodu parser nie obsługuje całego języka.

Ponieważ można przygotować plik w formacie GML w taki sposób aby przypominał zapis KML zdecydowano się zwrócić szczególną uwagę aby parser poprawnie rozpoznawał i dekodował dane również z tej postaci, pozwoli to na dostęp do dużej bazy gotowych informacji. Wczytane informacje podlegają

przetworzniu ze składni XML, na którym bazuje GML do notacji JSON która jest używana podczas pracy map.

5.2.6. Storage

Aby stworzyć framework który będzie w stanie działać przy minimalnej konieczności konfiguracji dodatkowych środowisk postanowiono aby dane w pierwszej kolejności były przechowywane po stronie klienta. Do tego celu nadaje się funkcjonalność stworzona w ostatniej wersji HTML którą jest Storage. Pozwala on na przechowywanie danych w przeglądarce użytkownika [Pil10]. Różnicą w stosunku do ciasteczek które również potrafią przechowywać dane są:

- Większy rozmiar dostępnej pamięci m.in. Chrome 5MB , IE 10MB
- Informacje przechowywane są po stronie użytkownika, nie są przesyłane za każdym razem do serwera.
- Informacja może być przechowywana przez długi okres czasu.

Wadą tego typu pamięci jest jej interfejs. Obecnie przechowywany sposób danych to mapowanie w postaci napis->napis. Wymusza to aby dane które chcemy przechować były zapisane jako ciągi znaków. Przykład 5.2.6 przedstawia w jaki sposób możemy obiekt zawierający imię i nazwisko zapisać w pamięci. Linia 4 przedstawia obiekt w postaci której chcielibyśmy go przechować. Niestety zwykłe przypisanie do zmiennej w pamięci powoduje że jedynie typ instancji zostaje zapisany. Aby móc zapisać w poprawnej formie dane musimy dokonać serializacji danych. Czynność tą możemy wykonać przy pomocy metody “stringify” z obiektu JSON, wynikiem jest ciąg znaków który możemy bez problemu zapisać w pamięci sesyjnej. Do odzyskania pierwotnego obiektu, odtworzenia go z zapisanego napisu wykorzystujemy metodę “parse” również z obiektu JSON.

```

1 uzytkownik={};
2 uzytkonwik.imie='Jan'
3 uzytkownik.nazwisko='Kowalski'
4 //uzytkownik : Object {imie: "Jan", nazwisko: "Kowalski"}
5
6 sessionStorage.u1 = uzytkownik
7 //sessionStorage.u1 : "[object Object]"
8
9 sessionStorage.u2 = JSON.stringify(uzytkownik)
10 //sessionStorage.u2 : "{\"imie\":\"Jan\", \"nazwisko\":\"Kowalski\"}"
11
12 uzytkonwik2 = JSON.parse(sessionStorage.u2)
13 //uzytkonwik2 : Object {imie: "Jan", nazwisko: "Kowalski"} 
```

Listing 5.4: Wykorzystanie Storage

Dodatkowo nie można pominąć faktu istnienia dwóch typów.

- Session Storage

Dane przechowywane są w kontekście sesji użytkownika, są one tracone w momencie zamknięcia okna przeglądarki.

- Local Storage

Teoretycznie dane są przechowywane w nieskończoność, do momentu kiedy użytkownik nie usunie ich. Zamknięcie sesji nie powoduje usunięcia danych.

Zapisywanie informacji po stronie klienta ma na celu zachowanie aktualnego stanu mapy, dokonanych zmian i naniesionych poprawek, informacje o poprzednich są zbędne. Sytuacja ta jednoznacznie wskazuje że lepszym wyborem jest pamięć sesyjna(Session Storage).

Podsumowanie dostępne na stronie <http://www.html5rocks.com/en/features/storage> (dostęp 31.10.2013) pozwala na sprawdzenie minimalnej wersji przeglądarki od której wspierane jest to rozwiązanie. Wskazuje ono na największą dostępność spośród dostępnych sposobów przechowywania informacji po stronie klienta.

5.2.7. Transmisja danych

Ważnym aspektem który należy rozwiązać jest sposób przesyłania danych. Problem ten jest szczerze istotny w omawianej pracy z uwagi na możliwość przesyłania dużej ilości informacji o granicach lub innych liniach przedstawionych na mapie. Do opisu kwadratowego obszaru wymagane jest przesłanie informacji o minimum 4 punktach. Jeżeli będziemy chcieli przekazać dokładniejszy zarys obszaru, zaprezentować granicę państwa lub linię frontu wojennego linia prosta w większości przypadków będzie zbyt ogólnym przybliżeniem, nie oddającym prawdziwej sytuacji.

Projekt zakłada korzystanie z lokalnej pamięci komputera użytkownika podczas pracy, jednak dostarczenie informacji, danych wejściowych nie powinno być ograniczone do wczytania ich z pliku tekstowego, należy pamiętać o możliwości przesłania ich z serwera aplikacji do użytkownika. W sytuacji takiej, gdzie w jednym momencie przesłane zostaną wszystkie informacje dotyczące mapy, ilość danych może być znaczna.

Z raportu Akamai wynika że średnia przepustowość łączy internetowych dla użytkowników korzystających z puli adresów IP przeznaczonych dla Polski w I kwartale 2012 r. wynosiła 5Mb/s <http://www.rp.pl/artykul/924483.html> (dostęp 13.10.2014). Jest to bardzo dobry wynik który plasuje Polskę w czołówce rankingu. Pomimo tego wyniku nie można pominąć faktu optymalizacji przesyłanych danych, wymieniane dane pomiędzy użytkownikiem a serwerem powinny być jak najmniejsze. Duża popularność urządzeń mobilnych w których dostęp do internetu jest zapewniany często poprzez sieć bezprzewodową a dostęp do interentu nie jest jeszcze tak dogodny jak jest to w przypadku użytkowników stacjonarnych wymusza minimalizowanie przesyłanych informacji.

Kolejnym powodem dla którego odpowiedzi serwera powinny być jak najlepsze jest koszt pracy samego serwera. Jest to szczególnie widoczne w dużych aplikacjach mających wielu użytkowników, czas jaki jest przeznaczany dla pojedynczego jest mnożony przez ich ilość. Z tego powodu zawsze podczas zwiększenia ilości użytkowników korzystających z aplikacji następuje zwiększenie stawianych wymagań wobec serwera, w ostatczności wymagane jest wykorzystanie kolejnej fizycznej maszyny. Celem pro-

gramisty tworzącego kod który będzie wykorzystywał zasoby serwera(zarówno jego czas procesora jak i pamięć) jest dbanie aby moment w którym niezbędne będzie korzystanie z większej ilości maszyn nastąpił przy jak największej ilości użytkowników.

Z uwagi na omówione aspekty zdecydowano się aby w przypadku korzystania z danych nie znajdujących się na lokalnym dysku komputera wszystkie informacje dotyczące mapy zostały przesłane jako plik w określonym formacie. Zostanie on zapisany w pamięci komputera użytkownika i podlegał dalszej pracy. Takie podejście minimalizuje wymagania nakładane na serwer i czas zajęcia procesowa.

5.3. Wizualizacja

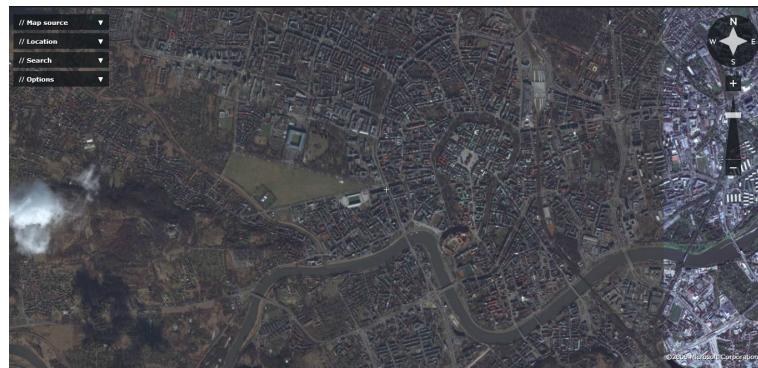
Poniższa sekcja zawiera próbę analizy dostępnych metod prezentacji danych zarówno zbiorów punktów, zdjęć jak i animacji. Główną uwagę zwrócono na 3 główne sposoby wizualizacji grafik. Ostatnia część zawiera zbiór dostępnych funkcjonalności, stanowi przewodnik po dostępnych możliwościach i ukazuje w jaki sposób można personalizować dane.

5.3.1. Rodzaje map

Tworząc aplikację która ma dostarczać informacji korzystających z map należy zapoznać się dostępymi źródłami. Z powodu szerokiego wyboru poniżej omówione zostały jedynie aplikacje które dostarczają informacji ogólnoswiatowych. Na polskim rynku dostępnych jest kilka rozwiązań, ich główną wadą jest ograniczenie terytorialne, dodatkowo często nie dostarczają one obrazów satelitarnych, przykładem jest zumi (<http://zumi.pl>).

Windows Maps

Rysunek 5.3 przedstawia obraz otrzymany wykorzystując Bing Maps. W tym konkretnym przykładzie widzimy znaczną różnicę kolorów, obecność chmur pomniejsza wartość tych zdjęć. Należy wspomnieć o ubogim interfejsie dostarczanym użytkownikowi.



Rysunek 5.3: Bing Maps.

Yahoo Maps

Kolejnym dostarczycielem danych kartograficznym jest Yahoo, przykład znajduje się na rysunku 5.4. Interfejs jest zbliżony do Bing Maps, jednak obszar na którym możemy przedładać zdjęcia jest mniejszy, tereny oddalone od większych miast nie są w pełni uwzględnione. Z tego powodu nie stanowi w pełni akceptowalnego rozwiązania.



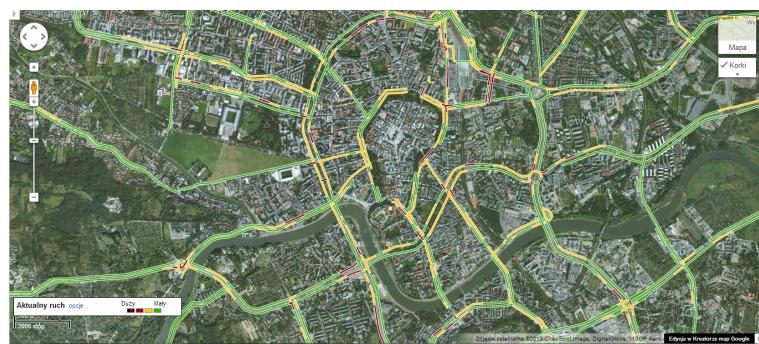
Rysunek 5.4: Yahoo Maps.

Apple Maps

Kolejną dużą marką która dostarcza informację jest Apple. Niestety nie ma wersji która pozwalałaby na dostęp do tej usługi z powszechnie używanych komputerów stacjonarnych. Dodatkowo jakość dostarczanych informacji jest zlej jakości.

Google Maps

Rysunek 5.5 przedstawia obraz otrzymany w aplikacji Google Maps. Dodatkowo włączona opcja prezentacji natężenia ruchu jedynie potwierdza duże możliwości i łatwość obsługi. Przyjazny interfejs sprawia że praca jest prosta i pozwala na osiągnięcie bardzo dobrych wyników. Z uwagi na wielkie korzyści i duże możliwości konfiguracji rozwiązanie to zostało wybrane jako bazowe w tworzonej pracy.



Rysunek 5.5: Google Maps.

5.3.2. Metody prezentacji

Poniższa sekcja zawiera opis dostępnych metod służących do prezentacji danych w przeglądarce internetowej. Przedstawia zalety jak i wady każdego z rozwiązań.

Flash

Technologia pozwalająca na tworzenie interaktywnych rozwiązań posiadających bogatą szatę graficzną.

Wadą tego rozwiązania wynikającą ze specyfikacji całego języka jest bardzo słaba integracja z dokumentami HTML, nie możliwa jest pełna interakcja z pozostałymi elementami na stronie. Problem ten nie dotyczy kolejnego podejścia.

Canvas

Bardzo ciekawym i wartym zainteresowania dodanym elementem w nowej wersji HTML jest obecność znacznika canvas. Pozwala on na dynamiczne, skryptowe renderowanie kształtów i obrazów. Dzięki temu możliwe stało się tworzenie animacji czy nawet gier działających w przeglądarce bez konieczności używania dodatkowych wtyczek czy programów.

Przykładem wielkich możliwości jakie dostarcza udoskonalony język jest fakt iż już w roku 2011 powstała pierwsza trójwymiarowa gra stworzona w całości przy użyciu HTML5 [Tec11]. Przykład grafiki widoczny jest na rysunku 5.6.



Rysunek 5.6: Pierwsza gra 3D w HTML5.

Funkcjonalnością która wydaje się być szczególnie przydatna w stosunku do omawianego projektu jest możliwość rysowania kształtów geometrycznych. Przykładem wykorzystania zaprezentowano na rysunku 5.7 na którym przy pomocy okręgu zaznaczono rynek główny w Krakowie i jego okolice, możliwość zmiany transparentości narysowanego kształtu pozwala aby obraz pod nim był nadal widoczny.

Niestety rozwiązanie to nie może zostać wykorzystane wraz z wybranym modelem prezentacji danych geograficznych. Rysunek 5.7 prezentuje potencjalne możliwości jednak nawet analiza kodu 5.5 wskazuje podstawowe wymaganie wykorzystanie tego rozwiązania. Bazowy element w obrębie którego możliwa jest akcja musi być elementem typu canvas. Otrzymany efekt jest wynikiem wykorzystania statycznego obrazu który został zimportowany z pamięci komputera i wykorzystany jako tło. Zabieg ten pozwala na zaprezentowanie możliwości rozwiązań ale uniemożliwia wykorzystanie funkcjonalności które dostarczane są przez Google Maps(zmiana punktu patrzenia, stopnia przybliżenia i inne). Z tego powodu zdecydowano się na zaprzestanie prac z tym konkretnym rozwiązaniem i nie wykorzystanie go w docelowej aplikacji.



Rysunek 5.7: Zaznaczenie przez canvas

```
1
2  var canvas = document.getElementById('myCanvas');
3  var context = canvas.getContext('2d');
4  var imageObj = new Image();
5
6  imageObj.onload = function() {
7    context.drawImage(imageObj, 69, 50);
8    context.beginPath();
9    context.arc(canvas.width / 2, canvas.height / 2, 90, 0, 2 * Math.PI, false);
10   context.fillStyle = "rgba(255, 0, 0, 0.5)";
11   context.fill();
12   context.stroke();
13 };
14 imageObj.src = './gm_1.jpg';
15 .
16 .
17 .
18 <canvas id="myCanvas" width="200" height="100">
19 </canvas>
```

Listing 5.5: Canvas

SVG

SVG(en. Scalable Vector Graphics) służy do prezentacji danych, tworzenia animacji i płynnych zmian kształtów. Jest to powszechnie dostępny format zapisu grafiki wektorowej który oprócz prostych kształtów geometrycznych pozwala na korzystanie z zaawansowanych filtrów i efektów.

Wykorzystywany jest wektorowy format zapisu danych, został on dokładniej omówiony w sekcji 5.2.2. W wyniku takiego zabiegu plik svg opisujący obraz graficzny może mieć mniejszy rozmiar od zapisu rastrowego, jednocześnie niezależnie od wielkości jego jakość jest taka sama.

Autorzy artykułu “Zastosowanie języka SVG do wizualizacji danych geoprzestrzennych” [JB] zwracają również uwagę na tekstowy format zapisu, jest on mniej oszczędny niż binarny ale pozwala na edycję dowolnym edytorem tekstu co pozwala na przeszukiwanie zawartości np. wyszukiwarkom internetowym.

W celach porównawczych wykorzystano logo AGH (rys. 5.8) o rozmiarach 305x591 pikseli o gęstości 150 dpi (en. Dots per inch, określa ilość indywidualnych punktów na odcinku jednego cala, 2.54cm), po zapisie na dysku twardym zajmował on 75 KB pamięci. Następnie wykonano zapis grafiki w testowanym formacie zajmował on 3KB , efekt został przedstawiony w listingu 5.6.



Rysunek 5.8: Logo AGH

```

1 <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
2   <g transform="translate(0.000000,591.000000) scale(0.100000,-0.100000)" fill="#1
3     e1e1e" stroke="none">
4     <path d="M984 5863 c70 -97 112 -224 143 -429 15 -97 17 -289 20 -1761 14 -1653 99 0
5       100 0 0 1543 c0 933 -4 1594 -10 1673 -12 158 -37 262 -89 369 -61 127 -136 208
6       -265 285 1-38 23 36 -50z"/>
7     <path d="M1115 5878 c164 -131 256 -283 306 -503 123 -100 3 -1627 4 -1628 94 0 95 0 0
8       1618 c-1 1317 -3 1631 -14 1692 -50 274 -208 455 -485 556 1-66 24 40 -32z"/>
9     <path d="M1317 5867 c208 -110 326 -253 395 -477 123 -75 3 -1647 2 -1648 100 0 100 0
10      0 1609 c0 1744 2 1676 -54 1826 -62 163 -173 282 -343 368 -57 29 -193 71 -263 81
11      -44 6 -44 5 37 -37z"/>
12     <path fill="#00693c" d="M52 3578 13 -1433 23 -79 c76 -271 252 -444 522 -515 146 -38
13       153 -36 65 15 -170 97 -261 190 -331 339 -84 180 -78 49 -81 1668 1-3 1437 -100 0
14       -101 0 3 -1432z"/>
15     <path fill="#00693c" d="M350 3622 c0 -909 4 -1417 11 -1472 12 -101 54 -228 100 -307
16       73 -124 220 -238 384 -298 170 -26 -35 27 c-61 47 -164 160 -199 218 -61 102 -100
17       219 -121 361 -6 44 -10 594 -10 1478 10 1407 -100 0 -100 0 0 -1388z"/>
18     <path fill="#00693c" d="M650 3607 c0 -906 4 -1433 10 -1487 35 -282 148 -465 368 -594
19       8 -5 -3 14 -22 42 -74 103 -115 233 -141 452 -13 106 -15 346 -15 1558 10 1432
20       -100 0 -100 0 0 -1403z"/>
21     <path fill="#a71930" d="M2237 3518 c-3 -1478 -3 -1494 -25 -1604 -27 -139 -69 -258
22       -121 -334 1-39 -60 40 23 c68 37 186 155 228 226 21 36 50 101 65 143 55 168 55
23       152 55 1686 10 1412 -100 0 -99 0 -4 -1492z"/>
24     <path fill="#a71930" d="M2537 3563 c-3 -1354 -4 -1453 -21 -1523 -50 -211 -144 -362
25       -301 -488 1-40 -32 65 24 c255 93 402 249 472 501 123 80 3 1443 3 1442 -101 0
26       -100 0 -3 -1447z"/>

```

```

12 <path fill="#a71930" d="M2838 3573 1-3 -1438 -23 -80 c-63 -228 -190 -384 -403 -498
   -83 -44 -75 -45 56 -12 292 74 468 241 547 521 123 79 3 1433 3 1432 -100 0 -100 0
   -3 -1437z"/>
13 <path d="M1480 925 c-307 -69 -464 -387 -331 -670 101 -216 388 -308 704 -224 138 10
   -3 225 -3 226 -33 29 c-32 28 -35 29 -142 29 1-111 0 23 -31 c21 -29 23 -42 26
   -199 13 -169 -25 -7 c-58 -14 -155 23 -199 75 -48 58 -70 135 -71 251 -1 96 1 110
   27 162 31 64 71 102 136 131 60 26 195 21 260 -11 25 -12 46 -22 48 -22 2 0 3 40 3
   88 10 89 -37 12 c-60 20 -238 23 -313 6z"/>
14 <path d="M244 889 c14 -17 26 -39 26 -49 0 -10 -61 -197 -135 -416 -74 -219 -135 -400
   -135 -401 0 -2 38 -3 85 -3 184 0 34 108 35 107 150 3 150 3 37 -111 37 -110 128 0
   129 0 -29 83 c-97 284 -244 690 -265 732 -37 77 -58 85 -220 85 1-137 0 26 -31z
   m201 -369 c24 -71 41 -131 37 -134 -8 -9 -192 -7 -192 1 0 4 21 71 47 150 34 103
   50 138 56 127 5 -9 28 -74 52 -144z"/>
15 <path d="M2263 883 c22 -37 22 -45 25 -450 13 -413 120 0 119 0 0 190 0 190 135 0 135
   0 0 -190 0 -190 120 0 120 0 0 450 0 450 -120 0 -120 0 0 -185 0 -185 -134 0 -134
   0 -4 148 c-3 158 -9 176 -59 206 -20 12 -53 16 -128 16 1-100 0 22 -37z"/>
16 </g>
17 </svg>
```

Listing 5.6: Logo AGH w zapisie SVG.

Google Maps Overlay

Wraz z wykorzystaniem Google Maps jako bazowym źródłem danych otrzymujemy API(en.Application programming interface) umożliwiające tworzenie różnego rodzaju elementów których wygląd można w dowolny sposób edytować i zmieniać według potrzeb użytkownika. W prosty sposób możliwe jest otrzymanie efektu który został stworzony przy pomocy canvas na rysunku 5.7 zachowując pełną swobodę w dalszej pracy.

Pakiet google.maps zawiera kilka klas których zadaniem jest generowanie adekwatnych obiektów. Klasy te zostały omówione poniżej.

Marker

Pojedynczy punkt na mapie o określonych właściwościach które można dowolnie edytować, tak aby jego wygląd, zachowanie i położenie było zgodne z oczekiwaniami.

Polyline

Linia stworzona z dwóch lub więcej punktów. Tworzone odcinki mogą być liniami prostymi lub przedstawiać zakrzywienie ziemi, nie jest możliwe tworzenie bardziej skomplikowanych łuków.

Polygon

Tworzy ciąg punktów które w określonej kolejności tworzą zamknięty obszar.

Circle

Na podstawie środka i jego promienia tworzy okrąg z posiadający obszar wewnętrzny o określonych cechach.

Rectangle

Tworzy prostokąt posiadający przeciwnieległe boki w określonych punktach.

OverlayView

Specjalna klasa służąca do wyświetlania własnych wartości graficznych, takich jak zewnętrzny plik graficzny. Pozwala na łatwe łączenie ogólnodostępnych danych jakimi są mapy z własnymi zdjęciami, rysunkami itp.

Intuicyjna i szybka edycja danych jest zapewniona poprzez udostępnianie klasy "DrawingManager". Tworzy ona dodatkowy element na ekranie użytkownika służący do pracy z wartościami znajdującymi się na ekranie. Komponent ten wymaga jedynie wybrania odpowiedniego elementu a następnie wyznaczenie wymaganych informacji, zazwyczaj są to jeden lub dwa punkty niezbędne do stworzenia elementu. Pozostałe właściwości mogą być edytowane w dalszym etapie prac według potrzeb.

5.3.3. Przybliżanie zbiorów danych

Do wyznaczenia zamkniętego obszaru należy podać wszystkie punkty określające jego granice. Używając komputerów możliwy jest jedynie dyskretny zapis danych, oznacza to że tylko niektóre punkty znajdują się określone a odcinki pomiędzy nimi musi zostać zastąpiony danymi wygenerowanymi. Najprostszą metodą jest połączenie poszczególnych punktów liniami prostymi otrzymując proste kształty geometryczne, przy jej pomocy posiadając 3 punkty na płaszczyźnie możliwe jest otrzymanie trójkąta. Rozwiążanie takie jest wystarczające przy tworzeniu prostych figur, posiadających małą liczbę wierzchołków. Przy tworzeniu bardziej skomplikowanych staje się ono niepraktyczne, określenie setek punktów dla nieregularnego obszaru byłoby czasochłonne przez co jest nieefektywne. Rozwiązaniem tego problemu jest oszczędzanie danych, określenie rozwiązania przybliżonego na podstawie znanych danych.

Można rozróżnić dwa rodzaje wyznaczanie wyniku quasi-otpymalnego:

Aproxymacja

Wygenerowany wynik jest przybliżeniem punktów wyjściowych, stworzona linia nie musi przechodzić przez nie.

Interpolacja

Wygenerowana funkcja interpolacyjna przechodzi przez wszystkie punkty określone w zbiorze wejściowym, jest jego dokładnym otwrotem. Z tego powodu nadaje się lepiej do omawianeego celu.

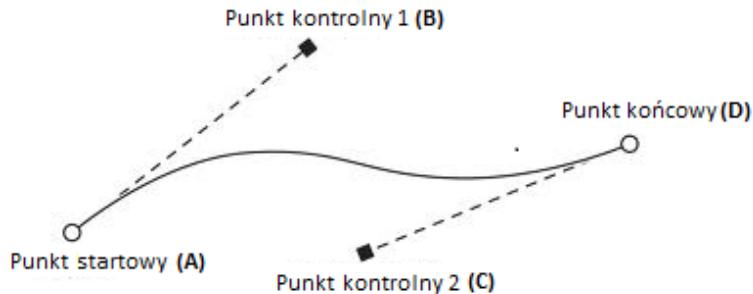
Najprostrzym rodzajem jest interpolacja funkcji sklejonych stopnia pierwszego. Określa ona zbiór prostych odcinków łączących pary poszczególnych punktów. To dzięki niej przy użyciu 3 punktów możliwe jest stworzenie trójkąta.

Interpolacja wielomianowa polega na określeniu jednej funkcji przechodzącej przez pożądane punkty posiadającej określone parametry. Złożoność tego zadania wzrasta drastycznie wraz z rozmiarem zbioru wejściowego z tego powodu nie została ona użyta w omawianym przypadku.

Kolejnym z rozwiązań jest skorzystanie z krzywej Beziera. Rysunek 5.9 przedstawia prosty przykład pokazujący niezbędne informacje do jej stworzenia dla zbioru cztero-elementowego. Dwa punkty kontrolne, mają za zadanie określić zachowanie ściszk. Prosta implementacja czyni to rozwiązanie dobrym wyborem omawianego problemu.

Jego wzór dla podanego przykładu to:

$$P(t) = A(1-t)^3 + 3Bt(1-t)^2 + 3Ct^2(1-t) + Dt^3 \text{ dla } 0 \leq t \leq 1 \quad (5.1)$$



Rysunek 5.9: Krzywa Beziera

5.3.4. Oś czasu

Tematem pracy oprócz wykorzystania i prezentacji danych na bazie informacji geograficznych jest ich połączenie z osią czasu, która pozwoliłaby na zmianę prezentowanych danych ze względu na interesujący nas okres czasu.

Ponieważ własna implementacja osi czasu która nieograniczałaby użytkownika, pozwalałaby mu w pełni korzystać możliwości gotowej aplikacji jest zbędna w sytuacji obecności wielu tego typu rozwiązań, postanowiono wykorzystać istniejące rozwiązanie.

Przeprowadzając badania rozwiązań głównymi aspektami na które zwrócono uwagę było:

- Typ licencji

Niezwykle ważne jest aby końcowy projekt był całkowicie otwarty dla dalszego rozwoju, pozwalał użytkownikom na adaptację rozwiązań dla swoich potrzeb. Z uwagi na to licencja musi pozwalać na modyfikację kodu źródłowego.

- Wykorzystywane technologie

Pamiętając że efektem końcowym powinien być framework odpowiadający szerokiemu gronowi odbiorców, będący jednocześnie darmowym ważne aby języki programowania wykorzystane przy jego tworzeniu też taki były. Oznacza to nie korzystanie z płatnych, wymagających licencji oprogramowań. Używane standardy powinny być rozpropagowane i używane przez szerokie grono obiorców.

Po analizie dostępnych rozwiązań zdecydowano skorzystać z widget-u o nazwie Timeline który został stworzony przez projekt SIMILE działający na uczelni MIT. Pozwala on na bardzo dużą konfigurowalność dzięki czemu jego modyfikacja, nawet bez ingerencji w kod źródłowy jest bardzo prosta. Projekt jest w stanie stworzyć kilka pasm które będą określały interwały czasu. Pozwala to aby wynik końcowy był przejrzysty bez względu czy prezentujemy wydarzenia które miały miejsce w okresie kilku minut czy setek lat.

Rysunek 5.10 prezentuje przykładową oś. Pasma odpowiadają zakresom czasu, nie muszą być jednakowe w każdym miejscu. W zaprezentowanym przykładzie dzień 23 listopada uznany został za warty dokładniejszemu przyjżeniu, łatwo możemy z dokładnością co do godziny zmieniać zakres czasu. Natomiast kolejny miesiąc może być o wiele mniej ciekawy, a obszar przeznaczony dla niego być mniejszy niż ten posiadany przez wymieniony powyżej dzień.

Dla porównania 5.11 prezentuje o wiele prostszą konfigurację w której dolne pasmo podzielone jest przez miesiące, natomiast góra przez tygodnie.



Rysunek 5.10: Timeline



Rysunek 5.11: Timeline - prosty przykład

5.4. Bezpieczeństwo danych i aplikacji

Tworząc aplikację która będzie używana przez dużą grupę obiorców, oprócz osób z pełnymi uprawnieniami do dowolnej akcji również osoby które nie powinny mieć możliwości zmiany danych należy przewidzieć i przygotować odpowiednie zabezpieczenia. Dostęp do dostarczanych funkcjonalności należy zapewnić poprzez zbiór ról i uprawnień, zostały one opisane w sekcji 4.3.

Niezwyczajne jest aby pamiętać o strukturze aplikacji i ograniczeniach jakie niosą za sobą wykorzystane technologie. Każdy użytkownik “uczeń” według nomenklatury opisanej powyżej jest w posiadaniu pliku którego edycja powinna być zachowana jedynie dla moderatora. Niestety nie jest możliwe zapewnienie nieedytowalności pliku tekstowego, z tego powodu nawet bez dostępu do uprawnień edycyjnych dostarczanych przez framework każdy może edytować plik w edytorze tekstowym, zmienić granice obszarów czy chociażby przedziały czasowe. Aby użytkownicy mieli pewność że plik z którego korzystają i mapa stworzona na jego podstawie jest wiarygodna i posiada poprawne dane proponowanym rozwiązaniem jest wykorzystanie jednego z algorytmu tworzących skrót z dowolnego ciągu danych, jednym z najczęściej wykorzystanym do takich celów jest MD5. Otrzymany wynik można zamieścić na serwerze w widocznym miejscu obok pliku z danymi w miejscu gdzie jedynie moderator ma dostęp. Dzięki takiej operacji użytkownik będzie mógł porównać wygenerowany skrót pliku którego chce użyć z wartością na stronie, identyczne wartości potwierdzą autentyczność danych.

5.4.1. Funkcje haszujące

W pewnych sytuacjach nie potrzebujemy lub nie chcemy przechowywać oryginalnego pliku lub ciągu znaków. Częstym wykorzystaniem jest utworzenie skrótu hasła w module logowania do różnych aplikacji. Czynność ta utrudnia poznanie rzeczywistych wartości w momencie dostępu do nich przez nieuprawnione osoby. Jedną z cech dobrej funkcji haszującej jest spełnienie wymagań dla funkcji jednokierunkowej. Wymagane jest aby funkcja była łatwa do obliczenia ale trudna, lub wręcz niemożliwa do odwrócenia.

Innym częstym wykorzystaniem, zaproponowanym w poprzedniej sekcji, jest weryfikacja integralności pliku lub wiadomości. Umieszczenie w jednym miejscu pliku i jego skrótu umożliwia sprawdzenie czy plik który posiadamy jest identyczny z tym znajdującym się na stronie, w przypadku braku zgodności możliwe jest ściągnięci poprawnej wersji, w pozostałych sytuacjach posiadamy pewność autentyczności danych bez potrzeby każdorazowego ściągania pliku z serwera, oszczędzamy przez to czas poświęcony na ściągnięcie pliku, który może zajmować wiele megabajtów.

Istnieje wiele funkcji haszujących, wybór konkretnej z nich zależy od celu użycia najpopularniejszymi są MD5 i SHA-1.

- MD5

Opracowany w 1991 roku przez Rona Rivesta, z dowolnego ciągu danych generuje 128-bitowy skrót, pozwala to na utworzenie 32 znakowego ciągu w systemie szesnastkowym. Ważną cechą jest zmiana skróty w przypadku najmniejszej edycji wejściowych danych. Tabela 5.1, stworzona przy pomocy strony <http://www.sha1-online.com/>, prezentuje przykład zmiany jednej

litery, zmiany “ó” na “o” w słowie Kraków. Wygenerowany skrót dla tej wartości jest zupełnie inny. Z uwagi na szybkość działania jest powszechnie używana.

- SHA-1

Przedstawiciel funkcji z rodziny SHA, opublikowany w 1995 roku. W przeciwieństwie do MD5 przy zapisie skrótu korzysta z 160-bitów, oznacza do 40 znaków w systemie heksadecymalnym.

- SHA-2

Zbiór czterech wariantów zastępujących SHA-1. Wersją zapewniającą najwyższą obecnie niezawodność i bezpieczeństwo jest SHA-512, do zapisu wykorzystuje 512 bitów co pozwala na otrzymanie 128 znakowego skrótu.

Algorytm	Wejściowy ciąg	Wygenerowany skrót
MD5	AGH Kraków	c9c16a90d4938da799b5aa7d6f37edce
MD5	AGH Krakow	fc6a83206673002588490eb05b89313f
SHA-1	AGH Kraków	6205880f638b3e584578125b14231c30c3ab9fea
SHA-1	AGH Krakow	a536aa59ed829801cd0f099c4ae3ba34ca57ca9f
SHA-512	AGH Kraków	49d9e2deb053a58dc0a778758e5(...)61499145e36353
SHA-512	AGH Krakow	6b65c56ca2e5567a5f18dd0d7465(...)f46166c29d7105

Tablica 5.1: Rezultaty funkcji haszujących

5.5. Optymalizacja rozwiązania

5.5.1. Wydajność parsera

W celach sprawdzenia wydajności zaimplementowanego parsera przeprowadzono testy porównawcze. W pierwszej próbie wykorzystano dwa pliki zawierające dane w formacie który pozwolił na jego analizę, pierwszy “plik1.kml” składał się z jednego obszaru i jednego stylu określającego preferencje graficzne, jego dokładana zawartość została zawarta w załączniku A.1. Drugi plik “plik2.kml” zawierał informacje o granicy wszystkich stanów USA, na potrzeby badań on również zawierał informacje o jednym stylu. Wykonano proces wczytania ich zawartości, mierząc za każdym razem czas potrzebny na zakończenie procesu. Wyniki zamieszczono w zbiorczej tabeli 5.3.

Dokładne informacje dotyczące zawartości plików zamieszczone w tabeli 5.2

Nazwa pliku	Ilość punktów	Ilość poligonów
plik1.kml	13	1
plik2.kml	13697	133

Tablica 5.2: Pliki testowe

Test	13 punktów	12697 punktów
I	3	343
II	13	449

Tablica 5.3: Czas dostępu [ms]

5.5.2. Reprezentacja graficzna

W celu optymalizacji działania aplikacji zastosowano kilka rozwiązań mających na celu przyśpieszenie działania. Dotyczą one sposobu w jaki aplikacja renderuje graficzną część projektu.

Stopniowa generacja obiektów

Nie wszystkie obiekty dotyczące mapy są renderowane podczas jej inicjalizacji. Wstępna konfiguracja pozwala na stworzenie mapy z osią czasu nie pokazującą całego przedziału czasowego do którego odwołują się informacje wejściowe, możliwe jest ustawnienie aby na ekranie pojawiły się informacje dotyczące wydarzeń z jednego dnia podczas gdy dane mogą zawierać informacje z okresu kilku lat. Stworzony algorytm można przedstawić w następujących krokach.

1. Renderowanie jedynie elementów widocznych na ekranie (mających miejsce w widocznym okresie czasu).
2. Renderowanie dodatkowych elementów zawierających się w otoczeniu wynoszącym 10% aktualnego okresu a następnie ukrycie ich. Takie rozwiązanie pozwala, w większości przypadków, na redrowanie nawet dużych i skomplikowanych obiektów bez wpływu na działanie użytkownika (posiada on dane wygenerowane w pkt. 1 i nie oczekuje na wygenerowanie kolejnych).

3. Podczas przesunięcia lini czasu w pierwszym momencie elementom w najbliższym sąsiedztwie ustawiana jest flaga widoczności, jeśli zmiana okresu była niewielka akcja ta jest wystarczająca aby pokazać wszystkie wymagane elementy, w przeciwnym przypadku następuje generacja brakujących. W obu przypadkach po zakończniu powtarzaj jest pkt 2. Obiekty które nie powinny być widoczne nie są niszczone, jedynie chowane.

Różne poziomy renderowania

Typy map z których będzie pracował framework pozwalają na wyświetlanie danych dla różnych wysokości nad poziomem morza. Funkcjonalność ta jest wykorzystana w celu optymalizacji dzielenia podczas pracy na różnych poziomach złożenia. Możliwe jest stworzenie bardzo szczegółowego opisu terenu na dużego zbliżenia i kolejnego wyświetlanego przy oddalonym widoku. Przykładowo granice miasta Krakowa w momencie oglądania ich z wysokości kilkuset metrów mogą zawierać zarysy budynków i ulic, na wysokości kilkuset kilometrów widok może się zmienić w prosty zarys granic miasta.

Jednocześnie takie rozwiązanie pozwala na wstawianie do mapy zewnętrznych plików graficznych widocznych jedynie na określonej wysokości, może to być aktualne zdjęcie wykonane z pokładu samolotu, albo rysunek dostarczający dodatkowych informacji.

6. Implementacja

W działaniu programu można wyodrębnić 2 główne etapy, jest to dostarczenie danych zgonych z formatem aplikacji a następnie ich wyświetlenie. Zapewniono obsługę zewnętrznych plików, części formatu GML który jest bardzo obszerny i może symulować inne. Interpretacja graficzna danych została zoptymalizowana na potrzeby pracy z dużymi zbiorami danych, tak aby wyświetlenie nawet dużej ilości informacji umożliwiało płynną pracę.

6.1. Struktura projektu

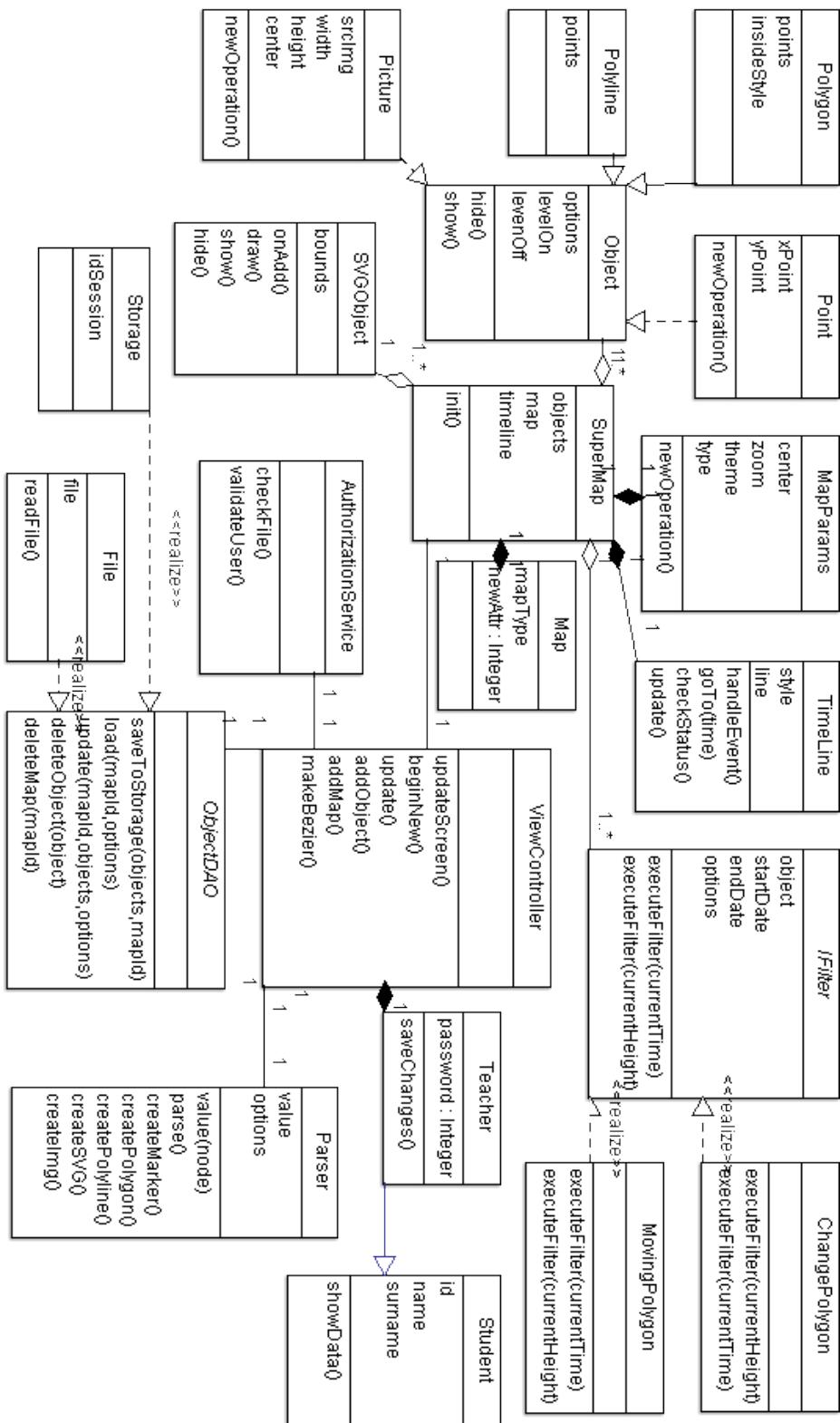
Struktura klas w aplikacji została stworzona w oparciu o wzorzec projektowy MVC(en. Model–view–controller). Zakłada on podział na 3 główne warstwy, są nimi:

- Controller - jego zadaniem jest przyjmowanie danych wejściowych i na ich podstawie odpowiednia reakcja, która może być aktualizują modelu lub warstwy prezentacyjnej. W aplikacji funkcję tą spełniają serwisy i kontrolery.
- View - warstwa prezentacyjna, zapewnia wizualną prezentację informacji. Rolę tą w omawianym projekcie spełnia HTML i CSS.
- Model - przechowuje dane, stanowi bazę danych. Jego rola została przeniesiona na natywne rozwiązańe jakim jest Storage.

Założenia wzorca nie zostały w pełni spełnione, jego adaptacja została wykonana na potrzeby projektu. Stworzono podstawy do rozwoju i prostego przystosowania rozwiązania do wymagań klienta, w bardzo prosty sposób można zmienić między innymi sposób przechowywania danych aby spełnić.

Zastosowano również paradygmat odwróconego sterowania(en. Inversion of control, IoC) polegający na przeniesieniu poza obiekt metod służących do kontroli jego poszczególnych elementów. Główną zaletą tego podejścia jest uproszczona konfiguracja programu do potrzeb użytkownika. Przykładem zastosowania jest metoda korzystania z filtrów w aplikacji, które mogą być tworzone przez użytkownika. Ich potencjalnie duża ilość wczytywana jednocześnie podczas startu generowałaby zbędne obciążenie, obecnie są one pomijane podczas inicjalizacji i wczytywane pojedyńczo jedynie wtedy gdy ich obecność jest niezbędna.

Struktura klas obecnych w aplikacji została zaperzentowana na rysunku 6.1.



Rysunek 6.1: Diagram klas.

6.2. Minimalna konfiguracja

Do rozpoczęcia korzystania z gotowej aplikacji należy zapewnić obecność konfiguracji niezbędnej do jej pracy na docelowej stronie. Przykład minimalnej wersji która spełnia wymagane został pokazany w listingu 6.5.

Z uwagi na naturę projektu, aplikacja wykorzystywana w przeglądarce, jak każdy dokument stworzony w HTML musi składać się z 3 głównych części:

-lini zawierającej informację o używanej wersji HTML, w poniższym przykładzie dotyczy ona HTML5

-deklaracji sekcji header, zawiera odwołania do zewnętrznych plików i delkaracje funkcji, stylów -część właściwa strony która zawiera informacje właściwe, widoczne dla użytkownika strony.

Linie 5-7 mają za zadanie wczytanie zewnętrznych plików zawierające delkaracje metod w JS. 8 wczytuje plik zawierający główną klasę frameworku, poniżej znajduje się wczytanie niezbędnych stylów CSS dla działania osi czasu. 13-23 jest to opis metody która inicjuje działanie mapy, jest ona wykonywana po załadowaniu statycznych fragmentów strony.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?
      sensor=false&key={KEY_ID}">
5     <script type="text/javascript" src=".js/specialMap/lib/jquery-1.6.2.min.js"></
6       script>
7     <script type="text/javascript" src=".js/specialMap/lib/mxn/mxn.js?(googlev3)"></
8       script>
9     <script type="text/javascript" src=".js/specialMap/lib/timeline-1.2.js"></script
10      >
11     <script type="text/javascript" src=".js/specialMap/src/timemap.js"></script>
12     <script type="text/javascript" src=".js/specialMap/src/utils.js"></script>
13
14   <link type="text/css" rel="stylesheet" href=".css/examples.css"/>
15
16   <script>
17     var sm;
18     $(function() {
19       tm = SpecialMap.init({
20         mapDivId: "mapId",
21         timeDivId: "timeId",
22         dataType: "sessionStorage",
23         bandIntervals: [
24           Timeline.DateTime.YEAR
25         ],
26       });
27     });
28   </head>
29   <body>
```

```

28     <div id="specialMap">
29         <div id="timeId"></div>
30         <div id="mapId"></div>
31     </div>
32 </body>
33 </html>
```

Listing 6.1: Minimalna konfiguracja.

6.3. Interfers do map

W procesie badań literaturowych podjęto decyzję aby jako głównego źródła map wykorzystać Google Maps 5.3.1. Postanowiono jednocześnie umożliwić użytkownikowi możliwość zmiany i wyboru innego dostarczyciela danych. Niestety nawet krótka analiza interfejsów dostarczających do pracy nad różnymi typami map wskazuje na brak kompatybilności pomiędzy nimi, oznacza to brak możliwości stworzenia jednej implementacji działającej na różnych typach. Problem ten rozwiązano poprzez wykorzystanie Mapstraction, biblioteki której zadaniem jest płynne przechodzenie pomiędzy różnymi dostarczycielami i ich interfejsami.

Listening 6.2 pokazuje przykład wykorzystania biblioteki. Stworzenie instancji Mapstraction polega na wywołaniu konstruktora z podaniem id elementu który znajduje się na stronie i będzie używany jako nasza mapa a także typ mapy. Rezultatem jest obiekt który posiada wspólny interfejs dla różnych dostarczycieli danych.

```

1
2 var map = new Mapstraction(mapId, 'google');
3
4 var point = new mxn.LatLonPoint(0,0);
5 var marker = new mxn.Marker(point);
6 map.addMarker(marker);
```

Listing 6.2: Wykorzystanie Mapstraction.

6.4. Odczyt danych

Zgodnie z wnioskami omówionymi w sekcji 5.2.3 do przechowywania danych wykorzystano format GML. Wykorzystanie pliku tekstowego pozwala na swobodne tworzenie i edycję plików wejściowych. W celu odczytu danych z pliku i zapisu ich w pamięci sesyjnych przeglądarki(Storage) stworzona została klasa FileParser. Posiada ona jednoargumentowy konstruktor przyjmujący ciąg znakowy będący zawartością pliku.

Ze względów bezpieczeństwa JavaScript nie zezwala na bezpośredni dostęp do plików lokalnych użytkownika. Jest to pozytywne zachowanie zapewniające że żaden kod nie jest w stanie bezwiednie zmieniać zawartość dysku. Wymusza to korzystanie z eventów które będą aktywowane w momencie kiedy plik zostanie wybrany. W momencie zakończenia procesu odczytu zawartość pliku przekazywana jest do parsera który zapisuje dane w pamięci przeglądarki.

W przeciwnieństwie do języków które posiadają klasyczną koncepcję klasy JavaScript posiada jedynie obiekt, z uwagi na ten fakt tworzenie metod i procedur następuje odmiennie. Istnieją 3 podstawowe sposoby:

- Wykorzystanie globalnej funkcji.

Tworzony obiekt zawiera odowłanie do ciała metody, które jest globalne. Wadą takiego rozwiązania jest tworzenie wielu globalnym funkcji których nazwa może kolidować z innymi elementami, np przykład wykorzystywany w bibliotekach.

- Stworzenie funkcji wewnętrznej.

Podobnie jak w poprzednim przypadku tworzona jest funkcja, tym razem jej ciało znajduje się w obiekcie. Minusem jest konieczność odtwarzania metody przy każdej sytuacji tworzenia obiektu. Operacja taka może być kosztowna i negatywnie wpływać na pracę aplikacji.

- Prototypowanie.

Posiadając definicję obiektu możemy dodać do niego metodę. Sposób ten nie posiada wad opisanych powyżej, z tego powodu podejście zostało wykorzystane w stworzonej aplikacji. Przykład można zaobserwować w listeninu 6.5.

```
1
2 var gml;
3 function readFile(file,start,stop) {
4     var reader = new FileReader();
5     reader.onloadend = function(evt) {
6         if (evt.target.readyState == FileReader.DONE) {
7             gml = new FileParser( evt.target.result );
8             gml.parse()
9         }
10    };
11
12    var blob = file.slice(start, stop);
13    reader.readAsBinaryString(blob);
14 }
```

Listing 6.3: Odczyt pliku

```
1
2 GmlParser.value = function(e) {
3     a = GXml.value(e);
4     a = a.replace(/^\s*/, "");
5     a = a.replace(/\s*$/, "");
6     return a;
7 }
8
9 GmlParser.prototype.createMarker = function(point,begin, end, name, desc, style,
iconUrl) {
```

```
10  var markeroptions = this.opts.markeroptions;
11  var icontype = this.opts.icontype;
12  var icon = this.styles[style];
13
14  if (!markeroptions.icon) {
15    markeroptions.icon = icon;
16  }
17  var m = new mxn.Marker(point, markeroptions);
18
19  this.gmarkers.push(m);
20  item = {};
21  item.start = begin;
22  item.end = begin;
23  item.title = name;
24  item.desc = desc;
25  item.iconUrl = iconUrl;
26  p = {};
27  p.lat = point.lat();
28  p.lon = point.lng();
29  item.point = p
30  return item;
31 }
32
33 GmlParser.prototype.createPolyline = function(points, color, width, opacity,
34   name, desc) {
35  var polylineoptions = this.opts.polylineoptions;
36  var p = new GPolyline(points, color, width, opacity, polylineoptions);
37  this.gpolylines.push(p);
38
39 GmlParser.prototype.createPolygon = function(points, id, pId, begin, end, color,
40   width, opacity, fillcolor, fillopacity, name, desc) {
41
42  pointsArray = []
43  item = {};
44  for (var i = 0; i < points.length; i++) {
45    point = points[i];
46    p = {};
47    p.lat = point.lat();
48    p.lon = point.lng();
49    if (p.lon == null || p.lat == null || isNaN(p.lat) || isNaN(p.lon)) {
50      console.log("Null points");
51    } else {
52      pointsArray.push(p)
53    }
54  }
55  item.start = begin;
56  item.end = end;
57  item.polygon = pointsArray;
58  item.title = name;
```

```
59     item.mId = id;
60     item.pId = pId;
61     item.color = color;
62     item.width = width;
63     item.opacity = opacity;
64     item.fillcolor = fillcolor;
65     item.name = name;
66     item.desc = desc;
67
68     return item;
69 }
70
71 GmlParser.prototype.processing = function(doc) {
72     var that = this;
73     var xmlDoc = GXml.parse(doc)
74
75     var styles = xmlDoc.documentElement.getElementsByTagName("Style");
76     for (var i = 0; i < styles.length; i++) {
77         readStyle(styles[i]);
78     }
79
80     var placemarks = xmlDoc.documentElement.getElementsByTagName("Placemark");
81     items = [];
82     itemsMarkers = [];
83     for (var i = 0; i < placemarks.length; i++) {
84         var id = placemarks[i].getAttribute('id');
85         var timeSpan = placemarks[i].getElementsByTagName("TimeSpan")[0];
86         var begin = GmlParser.value(timeSpan.getElementsByTagName('begin')[0])
87         var end = GmlParser.value(timeSpan.getElementsByTagName('end')[0])
88         var name = GmlParser.value(placemarks[i].getElementsByTagName("name")[0]);
89         var desc = GmlParser.value(placemarks[i]
90             .getElementsByTagName("description")[0]);
91         if (desc == "") {
92             var desc = GmlParser
93                 .value(placemarks[i].getElementsByTagName("text")[0]);
94             desc = desc.replace(/\$\[name\]/, name);
95             desc = desc.replace(/\$\[geDirections\]/, "");
96         }
97         var style = GmlParser.value(placemarks[i]
98             .getElementsByTagName("styleUrl")[0]);
99         var coords = GXml.value(placemarks[i]
100            .getElementsByTagName("coordinates")[0]);
101
102         var path = coords.split(" ");
103
104         multiGeometry = getChildrenTag(placemarks[i], 'MultiGeometry')[0];
105         if(multiGeometry) {
106             allPolygons = multiGeometry.getElementsByTagName("Polygon")
107         } else {
```

```
108     allPolygons = {};
109 }
110
111 onePoint = getChildrenTag(placemarks[i], 'Point')[0];
112 var width = that.styles[style].width;
113 var color = that.styles[style].color;
114 var opacity = that.styles[style].opacity;
115 var fillopacity = that.styles[style].fillopacity;
116 var fillcolor = that.styles[style].fillcolor;
117 var iconUrl = that.styles[style].iconUrl;
118
119
120 if (onePoint) {
121     coords = GXml.value(onePoint.getElementsByTagName("coordinates")[0]);
122     var bits = coords.split(",");
123     var point = new mxn.LatLng(parseFloat(bits[1]),
124         parseFloat(bits[0]));
125     that.bounds.extend(point);
126     itemsMarkers.push(that.createMarker(point, begin, end, name, desc, style,
127         iconUrl));
128 }
129
130 for (var j = 0; j < allPolygons.length; j++) {
131     polygon = allPolygons[j];
132     coords = GXml.value(polygon.getElementsByTagName("coordinates")[0]);
133     path = coords.split(" ");
134
135     if (path.length > 1) {
136         var points = [];
137         var pId = polygon.getAttribute('id');
138         for (var p = 0; p < path.length; p++) {
139             var bits = path[p].split(",");
140             if (parseFloat(bits[1]) == null
141                 || parseFloat(bits[0]) == null
142                 || isNaN(parseFloat(bits[1]))
143                 || isNaN(parseFloat(bits[0]))) {
144                 } else {
145                     var point = new mxn.LatLng(parseFloat(bits[1]),
146                         parseFloat(bits[0]));
147                     points.push(point);
148                 }
149             this.pointsCount = this.pointsCount + points.length;
150             var linestring = placemarks[i]
151                 .getElementsByTagName("LineString");
152             if (linestring.length) {
153                 that.createPolyline(points, color, width, opacity,
154                     name, desc);
155             }
156 }
```

```

156
157     var polygons = placemarks[i].getElementsByTagName("Polygon");
158     items.push(that.createPolygon(points, id, pId, begin, end,
159         color, width, opacity, fillcolor, fillopacity,
160         name, desc));
161 } else {
162     var bits = path[0].split(",");
163     var point = new mxn.LatLng(parseFloat(bits[1]),
164         parseFloat(bits[0]));
165     that.bounds.extend(point);
166     that.createMarker(point, name, desc, style);
167 }
168 }
169 }
170 sessionStorage.polyline = JSON.stringify(items);
171 sessionStorage.marker = JSON.stringify(itemsMarkers);
172 }
```

Listing 6.4: Parser plików.

6.4.1. Dodawanie filtrów

Rozwinięciem podstawowej możliwości tworzenia prostych kształtów geometrycznych jest opcja tworzenia filtrów które mogą wpływać na prezentację danych.

Przykładowym użyciem jest możliwość stworzenia animacji płynnego przejścia z jednego kształtu w inny. Proces ten jest wykonany w następujących krokach:

- Przekazanie do aplikacji kształtu zawierającego informacje o stanie początkowym i końcowym.
- Dla każdej zmiany daty obliczenie aktualnego procentowego zakończenia procesu transformacji.
- Obliczenie położenia nowego punktu, tak aby leżał pomiędzy dwoma w odpowiednim stosunku obliczonym w poprzednim kroku.

```

1
2 var percent;
3 if (now < start) percent = 0;
4 else if (now > end) percent = 1;
5 else percent = 1 - ((end - now) / (end - start));
6
7 var pm = item.placemark;
8 var points=[], pt1, pt2;
9
10 for (var x=0; x<pm.points.length; x++) {
11     pt1 = pm.points[x];
12     pt2 = ep[x];
13     points.push(new mxn.LatLngPoint(
14         (pt1.lat + ((parseFloat(pt2.lat) - pt1.lat) * percent)),
```

```

15     (pt1.lng + ((parseFloat(pt2.lon) - pt1.lng) * percent)))
16   );
17
18   if (item.tween) item.map.removePolyline(item.tween);
19   item.placemark.hide();
20   item.tween = new mxn.Polyline(points),
21   item.tween.addData({
22     color: theme.lineColor,
23     width: theme.lineWeight,
24     opacity: theme.lineOpacity,
25     closed: true,
26     fillColor: theme.fillColor,
27     fillOpacity: theme.fillOpacity
28   });
29 item.map.addPolyline(item.tween);

```

Listing 6.5: Filt animujący.

Istnieje możliwość tworzenia nowych filtrów dla potrzeb konkretnych map lub danych.

6.4.2. SVG

Zapewniono aby użytkownik miał możliwość dodawania własnych elementów stworzonych w SVG do aplikacji, klasa SGSObject ma za zadanie przechowywanie informacji o nim i zapewnienie niezbędnych metod. Stworzony konstruktor przyjmuje dwa punkty które określają skrajne punkty prostokątnego obszaru w którym będzie wyświetlany obraz, zdecydowano aby były to północno-wschodni i południowo-zachodni.

```

1
2 function SVGObject(swBound, neBound, elements, map) {
3
4   this.swBound_ = swBound;
5   this.neBound_ = neBound;
6   this.elements_ = elements;
7   this.map_ = map;
8   this.div_ = null;
9   this.setMap(map);
10 }
11
12 SVGObject.prototype.onAdd = function() {
13
14   var div = document.createElement('div');
15   div.style.border = 'none';
16   div.style.borderWidth = '0px';
17   div.style.position = 'absolute';
18
19   var svgHolder = document.createElementNS("http://www.w3.org/2000/svg", "svg");
20   svgHolder.setAttribute("version", "1.2");
21

```

```
22  for ( el in elements ) {
23    createSVGEElement(el.type,el.properties,svgHolder);
24  }
25
26  div.appendChild(svgHolder);
27  this.div_ = div;
28  var panes = this.getPanes();
29  panes.mapPane.appendChild(this.div_);
30 };
31
32 SVGObject.prototype.draw = function() {
33
34   var overlayProjection = this.getProjection();
35
36   var sw = overlayProjection.fromLatLngToDivPixel(this.swBound);
37   var ne = overlayProjection.fromLatLngToDivPixel(this.neBound);
38
39   var div = this.div_;
40   div.style.left = sw.x + 'px';
41   div.style.top = ne.y + 'px';
42   div.style.width = (ne.x - sw.x) + 'px';
43   div.style.height = (sw.y - ne.y) + 'px';
44 };
45
46 SVGObject.prototype.onRemove = function() {
47   this.div_.parentNode.removeChild(this.div_);
48 };
49
50 SVGObject.prototype.hide = function() {
51   if (this.div_) {
52     this.div_.style.visibility = 'hidden';
53   }
54 };
55
56 SVGObject.prototype.show = function() {
57   if (this.div_) {
58     this.div_.style.visibility = 'visible';
59   }
60 };
61
62 function createSVGEElement( eltType, properties,svgElement) {
63   var elt = document.createElementNS('http://www.w3.org/2000/svg', eltType);
64   for ( prop in properties ) {
65     elt.setAttribute(prop, properties[prop]);
66   }
67   svgElement.appendChild(elt);
68 }
```

Listing 6.6: Klasa do obsługi SVG

6.4.3. Zdjęcia

Aby zapewnić jak największą funkcjonalność zapewniono możliwość dodawania zewnętrznych plików graficznych. Podobnie jak w przypadku SVG stworzono klasę która zapewnia metody to jej prawidłowej pracy. Główną różnicą jest zmiana w klasie odpowiedzialnej za stworzenie podstawowego elementu przechowywującego obraz. Fragment odpowiedzialny za tworzenie obiektu SVG został zastąpiony kodem który ma za zadanie dla podanej ścieżki do pliku graficznego dodaniem tagu img, który powszechnie przechowuje tego typu informacje.

```

1 PictureObject.prototype.onAdd = function() {
2
3
4     var div = document.createElement('div');
5     div.style.borderStyle = 'none';
6     div.style.borderWidth = '0px';
7     div.style.position = 'absolute';
8
9     var img = document.createElement('img');
10    img.src = this.image_;
11    img.style.width = '100%';
12    img.style.height = '100%';
13    img.style.position = 'absolute';
14    div.appendChild(img);
15
16    this.div_ = div;
17
18    var panes = this.getPanes();
19    panes.mapPane.appendChild(div);
20}

```

Listing 6.7: Klasa do obsługi SVG

6.5. Krzywa Bezier-a

Aby umożliwić moderatorowi sprawną pracę podczas tworzenia i edycji zbioru danych, możliwość dodawania płynnych krzywych bez konieczności wyznaczania wielu punktów stworzono implementację algorytmu wyznaczania punktów leżących na krzywej Bezier-a.

Przedstawione wzory matematyczne które zostały użyte do obliczenia położnia w miejscu t . Główna metoda *bezier* posiada 2 argumenty wejściowe, pierwszym jest wartość z przedziału od 0 do 1 określający procentowe położenie pomiędzy położeniem startowym a końcowym, drugim jest tablica punktów.

$$B(t) = \sum_{i=1}^n B_{n,k}(t)p_k \quad (6.1)$$

$$B_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k} \quad (6.2)$$

```
1 bezier = function(t, points){  
2     var _x = 0;  
3     var _y = 0;  
4     var length = points.length;  
5     for (var i = 0; i < length; i++) {  
6         _x += points[i].x * bernstein(length-1,i,t);  
7         _y += points[i].y * bernstein(length-1,i,t);  
8     }  
9     return {x: _x, y: _y};  
10 }  
11  
12 bernstein = function(n,m,t){  
13     return combination(n,m)*Math.pow(t,m)*(Math.pow((1-t),(n-m)));  
14 }  
15  
16 factorial=function(n){  
17     wynik = 1;  
18     while (n > 0)  
19     {  
20         wynik = wynik * n ;  
21         n -- ;  
22     }  
23     return wynik;  
24 }  
25  
26 combination=function(n,m){  
27     return factorial(n)/(factorial(m)*(factorial(n-m)));  
28 }
```

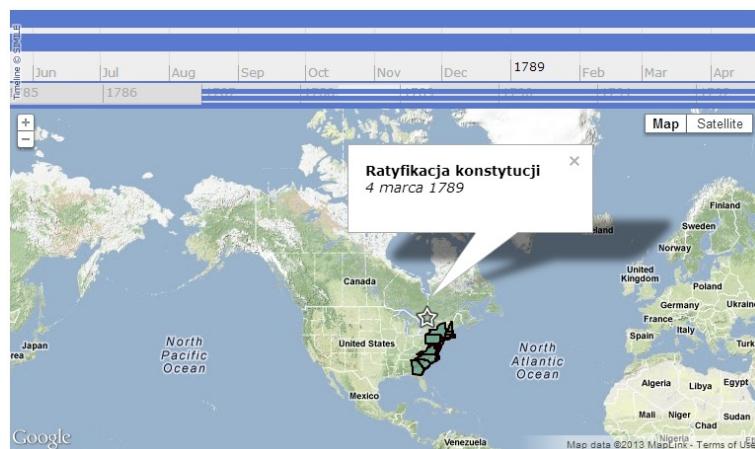
Listing 6.8: Aproksymacja danych

7. Przykład użycia

Poniższy rozdział zawiera przykład użycia stworzonej aplikacji. Prezentuje jedynie niewielką część możliwości.

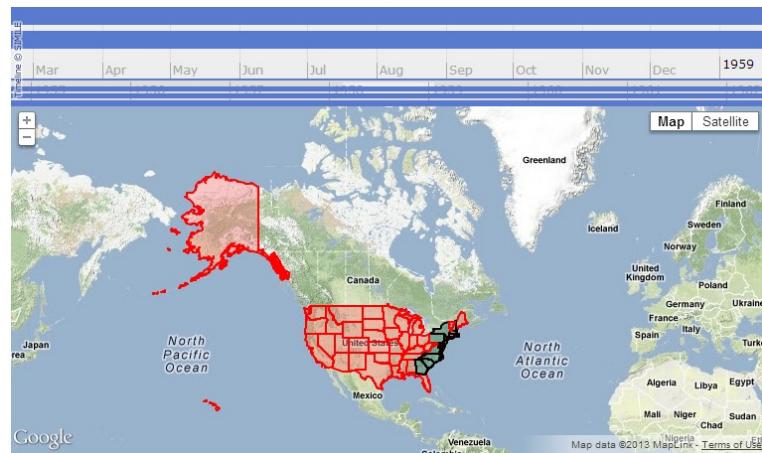
7.1. Historia USA

W celu zobrazowania powstawania USA poniżej zaprezentowano 3 stany jego rozwoju. Na rysunku 7.1 znajduje się obszar państwa w momencie ratyfikacji konstytucji. Kolorem czarnym zaznaczone są 13 pierwszych stanów które uznawane są za założycielskie. Pokazano również możliwość dodawania opisu do wydarzeń dodanych do sceny, umożliwia to dodanie dodatkowych informacji które mogą wspomóc proces dydaktyczny.



Rysunek 7.1: Rok 1789.

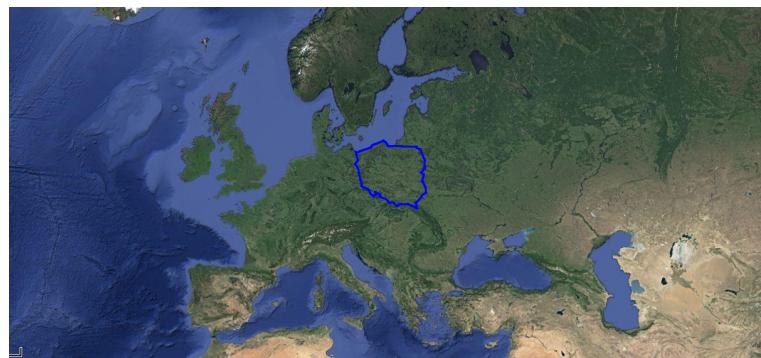
Kolejny rysunek ukazuje aktualny stan granic. Widać że utrzymuje się on od 1959 roku.



Rysunek 7.2: Rok 1959.

7.2. Granice Polski i Rzeczypospolitej Obojga Narodów

Kolejny przykład przedstawia możliwość zmiany rodzaju danych w zależności od stopnia przybliżenia. Rysunek 7.3 przedstawia granice wytyczone przy pomocy funkcji tworzącej linie bezier-a. Przy tak dużym oddaleniu szczegółów rysunku nie są widoczne, nie można przekazać wielu szczegółów.



Rysunek 7.3: Polska obecnie.

W prosty sposób można określić ogólne granice w dowolnym punkcie w czasie. Poniższy rysunek przedstawia wynik obszaru który został określony przez 105 punktów.



Rysunek 7.4: Polska w roku 1691.

W celu dostarczenia większej ilości informacji, możliwe jest załączenie pliku graficznego, w tym przypadku jest to zdjęcie mapy która przedstawia dokładny obszar Rzeczypospolitej Obojga Narodów. Jeszcze bliższe przybliżenie pozwali na odczytanie nazw miejscowości i obszarów geograficznych.



Rysunek 7.5: Rzeczypospolita Obojga Narodów, mapa.

8. Podsumowanie

Wysoki rozwój technologii komputerowej pozwala na redefiniowanie pojęć znanych człowiekowi od długiego czasu. Niniejsza praca udowadnia tą tezę, mapa która od zawsze była tylko niezmienną, graficzną reprezentacją terenu przy wykorzystaniu nowoczesnych technologii może stać się interaktywnym obiektem pozwalającym na ukazanie niedostępnych wcześniej danych w zupełnie nowy sposób.

Główna technologia wybrana do tworzenia aplikacji wykorzystywanej w przeglądarkach internetowych posiadającą możliwość interakcji użytkownika z danymi powinna dawać możliwość aktualizacji jedynie wybranych fragmentów danych, tak aby wyeliminować zbędne renderowanie stałych fragmentów. W takim przypadku dobrym wyborem jest JavaScript który dzięki założeniu postawionemu podczas jego tworzenia idealnie sprawdza się w tego typu zadaniach.

Format zapisu danych powinien zapewniać obsługę dowolnych punktów na kuli ziemskiej jak i dodatkowych elementów które mogą być dołączone do wejściowego zbioru danych. Ogólnodostępny format WGS pozwala na proste przechowywanie informacji o położeniu obiektów, jego zapis jest zarówno czytelny dla człowieka jak i dla komputera. Wybrany format zapisu plików, GML, pozwala na łatwe przechowywanie niemal wszystkich informacji w jednym miejscu, jedynie obrazy graficzne wymagają podania ścieżki do ich fizycznej lokalizacji.

Wybór Google Maps jako podstawowe źródło informacji geograficznych, dostarczających statyczny obraz map dało dostęp do ogromnej bazy danych. Niestety najnowsze rozwiązania nie zawsze zapewniają kompatybilność ze starszymi, przykładem jest jeden z elementów HTML w wersji 5, canvas. Jego wykorzystanie okazało się niemówzliwe w połączeniu wybranymi mapami których interfejs został stworzony bez możliwości ich wykorzystania. Zaawansowaną grafikę udało się stworzyć przy pomocy SVG, formatu który nawet skomplikowany obraz potrafi zapisać w zwięzlej i krókiej formie oszczędzając miejsce na dysku przy zachowaniu szczegółowości i ilości detali. Stworzone algorytmy do stopniowego generowania obiektów i ich zmiany w zależności od atrybutów mapy pozwalają na minimalizowanie wymagań w odniesiu do procesora i szybsze dostarczanie wartościowych danych.

Zwrócono również uwagę na bezpieczeństwo tworzonego projektu. Powstała aplikacja jest niezależna od jakichkolwiek zewnętrznych baz danych, dlatego ewentualna kontrola dostępu jest wykonywana przez nadzирующą wartość którą jest serwis używający map. Kontrola poprawności danych jest możliwa dzięki wykorzystaniu funkcji tworzących sumę kontrolną, użytkownik może dokonać jej porównania z wartością do której autentyczność jest niezaprzecjalna. Taka czynność daje pewność że dane które posiada użytkownik są poprawne i stworzone przez autoryzowane osoby.

Zastosowane metody optymalizacji framework-u pozwoliły na zwiększenie płynności działania, jedynie niezbędne akcje są wykonywane podczas jego pracy. Operacje zbędne, lub takie które mogą zostać wykonane w późniejszym czasie są zatrzymywane i uruchamiane jedy gdy jest to niezbędne.

Podsumowując stworzona aplikacja spełnia wszelkie warunki aby można było ją uznać za użyteczną podczas procesu nauki. Posiada intuicyjny interfejs pozwalający na łatwą obsługę, dostarcza cennych informacji a szeroki zakres prezentacji danych, możliwość tworzenia płynnych przejść kształtów, kolorów sprawia że jest interesująca i przyciąga uwagę użytkownika. Wykorzystanie jej m.in. na lekcji historii pozwoli zmienić statyczne reprezentacje ruchów wojennych w ciekawe i interaktywne, taka forma może zwiększyć przyswajalność widzy i zachęcić do częstych powtórek poza salą lekcyjną.

A. Dodatek A

A.1. Oryginalny plik kml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <kml>
3   <Document>
4     <name><! [CDATA[US States]]></name>
5     <open>1</open>
6     <Style id="Style_5">
7       <LabelStyle>
8         <color>9900ffff</color>
9         <scale>1</scale>
10      </LabelStyle>
11      <LineStyle>
12        <color>990000ff</color>
13        <width>2</width>
14      </LineStyle>
15      <PolyStyle>
16        <color>997f7fff</color>
17        <fill>1</fill>
18        <outline>1</outline>
19      </PolyStyle>
20    </Style>
21    <Placemark id="pm251">
22      <name><! [CDATA[Hawaii (1959)]]></name>
23      <Snippet maxLines="0">empty</Snippet>
24      <description></description>
25      <TimeSpan>
26        <begin>1989</begin>
27      </TimeSpan>
28      <styleUrl>#Style_5</styleUrl>
29      <MultiGeometry>
30        <MultiGeometry>
31          <Polygon id="g867">
32            <altitudeMode>clampToGround</altitudeMode>
33            <outerBoundaryIs>
34              <LinearRing>
35                <coordinates>
36                  -159.335174733889, 21.9483433404175
```

```

37 -159.327130348878,22.0446395507162
38 -159.295025589769,22.1248124949548
39 -159.343195828355,22.1970166285359
40 -159.391366885913,22.2291198667724
41 -159.576012589057,22.2131796383001
42 -159.712505933171,22.1490592515515
43 -159.800814224332,22.0366665967853
44 -159.736592652746,21.9644203111023
45 -159.640246973766,21.9483657695954
46 -159.576021285803,21.8841361312636
47 -159.439545188912,21.8680716835921
48 -159.335174733889,21.9483433404175
49         </coordinates>
50     </LinearRing>
51   </outerBoundaryIs>
52 </Polygon>
53 </MultiGeometry>
54 </MultiGeometry>
55 </Placemark>
56 </Document>
57 </kml>
```

Listing A.1: Plik KML

A.2. Zapis w pamięci sesyjnej

```

1 [{"start":"1989","end":"2010","polygon": [{"lat":21.9483433404175,"lon"
    :-159.335174733889}, {"lat":22.0446395507162,"lon":-159.32713034887797}, {"lat"
    :22.1248124949548,"lon":-159.295025589769}, {"lat":22.1970166285359,"lon"
    :-159.34319582835496}, {"lat":22.2291198667724,"lon":-159.391366885913}, {"lat"
    :22.2131796383001,"lon":-159.57601258905697}, {"lat":22.1490592515515,"lon"
    :-159.71250593317097}, {"lat":22.0366665967853,"lon":-159.800814224332}, {"lat"
    :21.9644203111023,"lon":-159.73659265274603}, {"lat":21.9483657695954,"lon"
    :-159.640246973766}, {"lat":21.8841361312636,"lon":-159.576021285803}, {"lat"
    :21.8680716835921,"lon":-159.439545188912}, {"lat":21.9483433404175,"lon"
    :-159.335174733889}], "title": "Hawaii (1959)", "mId": "pm251", "pId": "g867", "color":
    "#ff0000", "width": 2, "opacity": 0.59765625, "fillcolor": "#ff7f7f", "name": "Hawaii
    (1959)", "desc": ""}]
```

Listing A.2: Zapis w Storage

A.3. Wynik końcowy



Rysunek A.1: Hawaje

Spis rysunków

2.1	Mapa świata z 1489 roku.	9
3.1	Las Vegas w 1950 roku.	11
3.2	Las Vegas w 1977 roku.	11
3.3	Las Vegas w 2012 roku.	11
3.4	Europa w latach 30.	12
3.5	Mapa ciepła w USA.	13
4.1	Diagram przypadków użycia.	17
5.1	Zapis wektorowy	25
5.2	Zapis rasterowy	25
5.3	Bing Maps.	30
5.4	Yahoo Maps.	31
5.5	Google Maps.	31
5.6	Pierwsza gra 3D w HTML5.	32
5.7	Zaznaczenie przez canvas	33
5.8	Logo AGH	34
5.9	Krzywa Beziera	37
5.10	Timeline	38
5.11	Timeline - prosty przykład	38
6.1	Diagram klas.	44
7.1	Rok 1789.	56
7.2	Rok 1959.	57
7.3	Polska obecnie.	57
7.4	Polska w roku 1691.	57
7.5	Rzeczpospolita Obojga Narodów, mapa.	58
A.1	Hawaje	63

Spis tablic

4.1	Przypadek wczytania danych	18
4.2	Przypadek wyświetlenia danych	19
4.3	Przypadek kontroli pliku	20
5.1	Rezultaty funkcji haszujących	40
5.2	Pliki testowe	41
5.3	Czas dostępu [ms]	41

Bibliografia

- [BE07] Thiru Thangarathinam Michael Kay Alessandro Vernet Sam Ferguson Bill Evjen, Kent Sharkey. *Professional XML*. Wrox, 2007.
- [BG13] Shyam Seshadri Brad Green. *AngularJS*. OReilly Media, 2013.
- [Cho09] C. Choi. *A palaeolithic map from 13,660 calBP: engraved stone blocks from the Late Magdalenian in Abauntz Cave (Navarra, Spain)*. Journal of Human Evolution, 2009.
- [Fau13] Christian Faulhammer. What is a minimal working example. Website, 2013.
- [fla13a] Flash rendering. http://help.adobe.com/en_US/as3/mobile/WS08cf58784027fd117735d34612d0a8759d1-8000.html, 2013.
- [fla13b] Map of world. <http://www.nationalarchives.gov.uk/cabinetpapers/themes/maps-interactive/maps-in-time.htm>, 2013.
- [GS10] Cameron Turner Gabriel Svensson. *Beginning Google Maps API 3*. Apress, 2010.
- [Har03] Elliotte Rusty Harold. *Effective XML: 50 Specific Ways to Improve Your XML*. Addison-Wesley Professional, 2003.
- [Irv28] Washington Irving. *A History of the Life and Voyages of Christopher Columbus*. G. & C. Carvill, 1828.
- [JB] Stanisława Porzycka Justyna Biała. Zastosowanie języka svg do wizualizacji danych geoprze- strzennych.
- [jsm13] ammap. <http://www.ammap.com/>, 2013.
- [Lam09] L. Lamport. *LaTeX system przygotowywania dokumentów*. Wydawnictwo Ariel, Krakow, 2009.
- [Mae12] Kazuaki Maeda. Performance evaluation of object serialization libraries in xml, json and binary formats. *IEEE Press*, 2012.
- [MK08] Andrzej Chybicki Marcin Kulawiak, Mariusz Łuba. Web-based gis technologies dedicated for presenting semi-dynamic geospatial data. *IEEE Press*, 2008.
- [Pil10] M. Pilgrim. *Dive into HTML5*. 2010.

- [PL10] Frank Salim Peter Lubbers, Brian Albers. *Pro HTML5 Programming*. Apress, 2010.
- [SM08] Weichang Du Sai Ma, Minruo Li. Service composition for gis. *IEEE Press*, 2008.
- [SY08] Shaotao YuancRuan RenZongd Sheng Yea, Feng XueZhia. Gml – an open-standard geospatial data format. *IEEE Press*, 2008.
- [Tec11] Techtrendy. 2011.
<http://techtrendy.pl/title,Pierwsza-gra-3D-napisana-w-HTML5,wid,14102779,wiadomosc.html?ticaid=6107dc>.
- [Wan11] Guanhua Wang. Improving data transmission in web applications via the translation between xml and json. *IEEE Press*, 2011.
- [wor13] Worldology. <http://www.worldology.com/>, 2013.