Wyższa Szkoła Ekonomi i Innowacji
w Lublinie

Wydział Transportu i Informatyki

Kierunek: Informatyka

Specjalność: Inżynieria oprogramowania

Daniel Mariusz Dereziński

22443

Projekt oraz wdrożenie reaktywnego intranetu dla firmy – wykorzystanie technologi Meteor.js, Bootstrap i MongoDB

Praca inżynierska napisana na Wydziale Transportu i Informatyki pod kierunkiem Prof. Grzegorza Marcina Wójcika

Lublin 2015

Spis treści

Rozdział 1

Wstęp

1.1 Intranet i jego historia

Intranet jest to sieć komputerowa ograniczająca się do komputerów np. w danym przedsiębiorstwie lub innej organizacji, dostępna wyłącznie dla pracowników danej organizacji. Intranet dostarcza szeroki zakres informacji oraz usług z wewnętrznych systemów IT organizacji, które nie są dostępne z publicznego — zewnętrznego — Internetu. Firmowy Intranet dostarcza między innymi centralny punkt wewnętrznej komunikacji, współpracy. Intranet stanowi także pojedynczy punkt dostępu do wewnętrznych jakich zewnętrznych zasobów organizacji. W najprostszej formie intranet budowany jest z wykorzystaniem sieci typu *LAN* (sieć lokalna) oraz *WAN* (rozległa sieć komputerowa) [5].

Coś o historii intranetu....

1.2 Intranet obecnie oraz jego zastosowania

1.3 Cel pracy

W dzisiejszych czasach wiele organizacji / firm wykorzystuje w swojej działalności z jakieś formy intranetu — komunikacja, praca zespołowa. Są to rozwiązania oparte o darmowe systemy CMS, komunikatory, kalendarze, systemy do zarządzania zadaniami. Celem niniejszej pracy było opracowanie oraz wdrożenie systemu intranetowego dla firmy zajmującej się produkcją oprogramowania. Firmy z tej branży pracują w oparciu o projekty. Jednym z podstawowych celów intranetu jest wymiana wiedzy oraz komunikacja. Zaprojektowana oraz zaprogramowana aplikacja umożliwia dodawanie artykułów, dodawania kategorii, dodawanie projektów, komunikację w obrębie projektów wraz z możliwością dodawania artykułów. Aplikacja pozwala utworzyć profil dla organizacji, zaprosić użytkowników do tak utworzonego profilu w celu podjęcia wspólnej pracy. Możliwe jest

także tworzenie kont dla użytkowników nie powiązanych z żadną organizacją.

W rodziale 2 zostanie przedstawione ....

Rozdział 2

Technologia

Rozdział ten przedstawia wykorzystane technologie oraz języki programowania użyte podczas projektowania oraz programowania aplikacji Intranet. Aplikacja powstała z wykorzystaniem *JavaScript*, framework aplikacji sieciowych *Meteor.js*, nierelacyjnej bazy danych *MongoDB* oraz framework'a CSS *Bootstrap*, *HTML5*, *CSS3*, *less* — dynamiczny jeżyk arkuszy stylów oraz gotowy szablon dla panelu administracyjnego *AdminLTE* wykorzystujący Bootstrap.

2.1 JavaScript

Język programowania JavaScript został użyty do zaprogramowania zarówno części serwerowej (*back-end*) jaki i części odpowiedzialnej za interakcje z użytkownikiem (*front-end*) — interfejs użytkownika. Obecne strony WWW a w szczególności aplikacje dostępne przez przeglądarkę (Gmail, Google Docs, Google Maps, Facebook) szeroko korzystają z JavaScript w celu dostarczenie wielofunkcyjnego oraz interaktywnego interfejsu użytkownika. Jednym z powodów wykorzystanie JavaScript była możliwości wykorzystania go po stornie serwera oraz klienta. Najpopularniejszy obecnie sposób tworzenia stron/aplikacji internetowych wyróżnia trzy warstwy — warstwę struktury (HTML), warstwę prezentacji (CSS) oraz warstwę zachowania (JavaScript) [2].

Internet powstał jako zbiór statycznych dokumentów HTML, które były powiązane hiperłączami. Po wzroście popularności oraz rozmiaru sieci, autorom stron przestały wystarczać dostępne narzędzia. Widoczna stała się potrzeba poprawienia interakcji z użytkownikiem. U jej podstaw leżała chęć zmniejszenie ilości połączeń z serwerem w celu realizowania prostych zadań takich jak walidacja formularzy. W tym czasie pojawiły się dwie możliwości - aplety Javy oraz język *LiveScript*, który został zaproponowany przez firmę Netscape w roku 1995. Został on dołączony do przeglądarki Netscape 2.0 pod nazwą JavaScript [2].

Możliwość zmieniania statycznych elementów stron internetowych została bardzo szybko przyjęta przez rynek. Producenci przeglądarek internetowych szybko dostosowani swoje produktu do obsługi JavaScript'u. Microsoft wyposażył w taką obsługę swoją przeglądarkę Internet Explorer (IE) od wersji 3.0. Jednak była to kopia języka JavaScript — *JScript*, wzbogacona o kilka funkcjonalności przeznaczonych tylko dla IE. W wyniku coraz większych różnić pomiędzy przeglądarkami podjęto próbę standaryzacji różnych implementacji języka. Próbę tą podjęło Europejskie Stowarzyszenie na rzecz Systemów Informatycznych i Komunikacyjnych (ECMA). Została stworzona specyfikacja ECMAScript. Obecnie obowiązuje standard ECMA-262 [3] — JavaScript jest jego najpopularniejszą implementacja.

Wzrost popularności JavaScriptu miał miejsce w czasie Pierwszej Wojny Przeglądarek (1996-2001) [2]. Było to także okres tak zwanej bańki internetowej. W tym czasie o udział w rynku walczyli dwaj główni producenci przeglądarek Netscape oraz Microsoft. Firmy te kusiły klientów za pomocą coraz to nowych dodatków i ozdóbek wprowadzanych do przeglądarek oraz do stosowanych w nich wersji JavaScriptu. W tym czasie wiele osób wyrobiło sobie negatywną opinie na temat tego języka, który w wyniku wspomnianych działań oraz braku standaryzacji bez przerwy ulegał modyfikacją. Pisanie programów było koszmarem. Skrypty napisane w oparciu o jedną przeglądarkę nie chciały nie chciały działać w drugiej. Producenci przeglądarek, skupieni na rozszerzaniu o nowe funkcjonalności, nie dostarczali odpowiednich narzędzi do rozwijania aplikacji [2].

Niespójności pomiędzy przeglądarkami była tylko częścią problemu. Drugą częścią byli sami autorzy stron, którzy upychali w witrynach zbyt wiele zbędnych funkcjonalności. Bardzo często korzystali z wszystkich nowych możliwości dostarczanych przez przeglądarkę, przez co strony były „upiększane" o kwiatki takie jak animacje na pasku stanu, jaskrawe kolory, migające napisy, trzęsące się okna przeglądarek, płatki śniegu, obiekty podążające za kursorem itp., bardzo często utrudniało korzystanie ze stron. Tego typu nadużycia są także powodem złej reputacji JavaScriptu. Problemy te doprowadziły do traktowania języka JavaScript za niewiele więcej niż zabawkę przeznaczoną dla projektantów interfejsów.

Po zakończeniu Pierwszej Wojny Przeglądarek sposób wytwarzania aplikacji sieciowych uległ zmianie. Zmiany — na lepsze — zostały zapoczątkowane przez kilka procesów [2]:

- Microsoft wygrał wojnę i na okres około pięciu lata wstrzymał się od dodawania nowych funkcjonalności do przeglądarki Internet Explorer oraz do samego JavaScriptu. Dzięki temu producenci innych przeglądarek zyskali czas na dogonienie a czasem nawet przewyższenie możliwości IE.

- Ruch na rzecz standardów sieciowych zyskał przychylność programistów jaki i producentów przeglądarek. Standardy chronią programistów od konieczności progra-

mowania funkcjonalności dwa (lub więcej) razy na wypadek, gdyby coś nie działało, w którejś z przeglądarek. Co prawda nadal nie istnieje środowisko, które spełniało by wszystkie możliwe standardy.

- technologie i sposoby programowania osiągnęły bardzo dojrzały poziom, na którym można już zajmować się zagadnieniami takim jak użyteczność, dostępność czy też progresywne ulepszanie.

Dzięki nowym, zdrowszym metodologiom programiści zaczęli uczyć się lepszych sposób korzystania z już dostępnych narzędzi. Po wydaniu aplikacji takich jak *Gmail* czy *Google Maps*, które w bardzo szerokim stopniu wykorzystują programowanie po stronie klienta, oczywiste stało się, że JavaScript to dojrzały, jedyny w swoim rodzaju i potężny prototypowy język obiektowy [2]. Dobrym przykładem jego ponownego odkrycia jest szeroka akceptacja funkcjonalności dostarczanej przez obiekt `XMLHttpRequest`, który dawniej był obsługiwany tylko przez przeglądarkę Internet Explorer. Obiekt ten jednak został zaimplementowany przez wiele przeglądarek. `XMLHttpRequest` umożliwia wykonywanie żądań HTTP i pobieranie zawartości serwera w celu aktualizacji pewnych części strony bez konieczności przeładowanie jej całej. Dzięki temu narodził się nowy gatunek aplikacji sieciowych, które przypominają samodzielne aplikacje desktopowe. Takie aplikacje określamy mianem *AJAX*.

JavaScript po rewolucji spowodowanej rozwojem technologii AJAX zaczął być używany przez programistów do tworzenia rzeczywistych i ważnych aplikacji. Obecnie mamy wiele aplikacji sieciowych JavaScript począwszy od Twittera, przez Facebook, aż po GitHub [1]. Jedną z ciekawszych cech JavaScriptu jest to, że musi on działać wewnątrz *środowiska*. Najpopularniejszym środowiskiem jest przeglądarka internetowa. Istnieją także inne możliwości — JavaScript może działać po stornie serwera, na pulpicie lub wewnątrz tzw. *rich media*[1].

Od chwili wydania przeglądarki Google Chrome w roku 2008 ciągle oraz w bardzo szybkim tempie poprawia się wydajność działania JavaScript, co jest wynikiem silnej konkurencji pomiędzy producentami poszczególnych przeglądarek internetowych. Wydajność nowoczesnych maszyn wirtualnych JavaScript zmienia rodzaje aplikacji tworzonych dla sieci. Fantastycznym przykładem jest jslinux[2] — utworzony w JavaScript emulator pozwalający na wczytanie jądra systemu Linux, pracę w powłoce oraz kompilację programów w C.

[1] Aplikacje *rich media* — Flash, Flex — tworzy się przy użyciu ActionScriptu, który jest oparty o ECMAScript

[2] `http://bellard.org/jslinux/`

2.1.1 Przegląd cech JavaScript

JavaScript jest językiem programowania imperatywnym oraz strukturalnym. Wspiera większość składni programowania strukturalnego z języka C, np. pętle `while`, instrukcję wyboru `switch`, pętle `do while` oraz wiele innych. Wyjątkiem jest zasięg zmiennych. W JavaScript zasięg zmiennych z wykorzystaniem słowa kluczowego `var` to zasięg do całego ciała funkcji. Nowy standard ECMAScript 2015 (ES6) wprowadza zakres zmiennych co do bloku instrukcji z wykorzystaniem słowa kluczowego `let` — co oznacza że język ten ma obecnie zakres zmiennych co do ciała funkcji jak i do bloku instrukcji. Podobnie jak C JavaScript rozróżnia wyrażenia oraz instrukcji [6].

Jak większość języków skryptowych także JavaScript jest językiem dynamicznie typowanym. Typy powiązane są z wartościami a nie ze zmiennymi. Na przykład do zmiennej x może być przypisana wartość typu liczbowego a następnie możemy do takiej zmiennej przypisać na przykład łańcuch znaków [6].

JavaScript jest prawie w całości obiektowy. Obiekty w JavaScript są to tablice asocjacyjne rozszerzone o prototypy. Nazwy właściwości obiektu to ciągi znaków. Obiekty wspierają dwie różnoznaczne składnie — z kropką `obj.x = 10` oraz z nawiasami kwadratowymi `obj["x"] = 10`. Właściwości oraz ich wartości mogą być dodawane, zmienianie oraz usuwane w każdej chwili działania programu. Większość właściwości obiektu (oraz te dziedziczone w łańcuch prototypów) mogą być wymienione z wykorzystaniem pętli `for ... in`. JavaScript ma dość skromny wachlarz obiektów wbudowanych, między innymi funkcje `Function` oraz obiekty daty — `Date`. JavaScript oferuje również wbudowaną funkcje `eval`, która może wykonywać instrukcje dostarczone w postaci ciągów znaków podczas działania programu [6].

JavaScript jest także językiem funkcyjnym. Funkcję są typu pierwszej klasy. Oznacza to, że JavaScript wspiera przekazywanie funkcji jako argumentów do innych funkcji, zwracania ich jako wartości innych funkcji, przypisywania ich do zmiennych oraz zapisywanie ich w strukturach danych. JavaScript wspiera również funkcje anonimowe [4]. Funkcje w JavaScript jako takie posiadają właściwości oraz metody takie jak `.call()` i `bind`. Funkcje zagnieżdżone to funkcje zdefiniowane wewnątrz innych funkcji. Funkcja taka jest każdorazowo tworzona podczas wywołania funkcji zewnętrznej, w której została ona zdefiniowana. Dodatkowo każda tworzona funkcja tworzy *domkniecie*: zasięg zmiennych funkcji zewnętrznej, włączając w to zmienne lokalne oraz wartości argumentów, stają się częścią wewnętrznego stanu każdego wewnętrznego obiektu funkcji, nawet po zakończeniu wykonywania zewnętrznej funkcji [6].

JavaScript wykorzystuje prototypy, w odróżnieniu od innych języków zorientowanych obiektowo wykorzystujących klasy, w procesie dziedziczenia. W JavaScript można za symulować wiele cech dziedziczenia opartego na klasach wykorzystując prototypy [6]. Nowe

wydanie ES6 wprowadza do JavaScript składnie umożliwiającą definiowanie klas.

Funkcje obok swojej typowej roli w JavaScript mogą także tworzyć obiekty. Poprzedzając wywołanie funkcji instrukcją wbudowaną `new` zostanie utworzona instancja prototypu dziedzicząca właściwości oraz metody z konstruktora (włączając w to właściwości z prototypu obiektu wbudowanego `Object`). ECMAScript 5 oferuje metodę `Object.create` pozwalającą na jawne tworzenie instancji bez automatycznego dziedziczenia z prototypu `Object`. Właściwość `prototype` konstruktora określa jaki obiekt zostanie użyty jako wewnętrzny prototyp nowo utworzonego obiektu. Nowe metody mogą zostać dodane poprzez modyfikowanie prototypu funkcji użytej jako konstruktor. Wbudowane konstruktory takie jak `Array` lub `Object`, także posiadają prototyp, który może być modyfikowany. Modyfikowanie prototypy `Object` jest uznawane za złą praktykę ponieważ prawie wszystkie obiekty w JavaScript dziedziczą metody oraz właściwości z prototypu obiektu `Object` [6].

W odróżnieniu od wielu języków zorientowanych obiektowo w JavaScript nie ma różnicy pomiędzy definicją funkcji oraz definicją metody. Różnica pojawia się podczas wywołania funkcji oraz metody. Kiedy funkcja wywoływana jest jako metoda obiektu, słowo kluczowe `this` wewnątrz ciała funkcji jest powiązane z obiektem na rzecz, którego dana metoda została wywołana [6].

JavaScript posiada wbudowane implementacje, opartą na funkcjach, wzorców takich jak *cecha* oraz *domieszka*. Jawna, oparta na funkcjach, delegacja nie wspiera kompozycji, to natomiast nie jawna delegacja ujawnia się za każdym razem gdy przechodzony jest łańcuch prototyp np. w celu znalezienia metody, która nie należy bezpośrednio do obiektu. Gdy metoda zostanie odnaleziona wywoływana jest w kontekście danego obiektu. Dlatego dziedzinie w JavaScript jest realizowane przez delegacje, która to przypisana jest to właściwości prototyp konstruktora funkcji [6].

Typo JavaScript uruchamiany jest w jakimś środowisku np. przeglądarka internetowa, które dostarcza obiekty oraz metody, przez które uruchamiany skrypt może oddziaływać na środowisko np. obiekt DOM strony internetowej.

JavaScript przetwarza wiadomości z kolejka po jednej na raz. Podczas ładowania nowej wiadomości, JavaScript wywołuje funkcję powiązaną z daną wiadomości tworząc ramkę stosu wywołania (są to argumenty funkcji oraz zmienne lokalne). Stos wywołania zmniejsza się oraz powiększa zgodnie z potrzebami wywołanej funkcji. Nazwane jest to pętlą zdarzeń, opisywaną także jako „działaj do ukończenia", ponieważ każda wiadomość jest całkowicie przetwarzana zanim przejdziemy do kolejnej wiadomości. Jednakże wbudowany model konkurencji nadaje pętli zdarzeń charakter nie blokujący. Operacje wejścia/wyjścia realizowane są z użyciem zdarzeń oraz wywołań zwrotnych (funkcji zwrotnych). Oznacza to, że JavaScript może przetworzyć kliknięcie myszy podczas czekania na dane z zapytania

do bazy danych [6]

Do funkcji może zostać przekazana nie określona ilość argumentów. Funkcja ma do nich dostęp poprzez parametry oraz także przez lokalny obiekt `arguments` [6].

Jak wiele języków skryptowych, tablice jak i obiekty mogą zostać utworzone przez zwięzłą składnie. Te literały stanowią także podstawę formatu *JSON*[3] [6].

JavaScript wspiera również wyrażenia regularne w sposób podobny do *Perl*, z zwięzła oraz bogatą składnią do manipulowania tekstem, znacznie bardziej zaawansowaną niż wbudowane funkcje do obróbki łańcuchów znaków.

2.2 Meteor.js oraz MongoDB

[3]JavaScript Object Notation - lekki format wymiany danych komputerowych, jest to format tekstowy. Opisany w RFC 4627

Bibliografia

[1] Mike Cantelon i Marc Harter i TJ Holowaychuk i Nathan Rajlich. *Node.js w akcji*. Wydawnictwo Helion, Gliwice, 2014.

[2] Stoyan Stefanov. *JavaScript programowanie obiektowe*. Wydawnictwo Helion, Gliwice, 2010.

[3] Wikipedia. Ecmascript. `https://en.wikipedia.org/wiki/ECMAScript`, Listopad 2015.

[4] Wikipedia. First-class function. `https://en.wikipedia.org/wiki/First-class_function`, Listopad 2015.

[5] Wikipedia. Intranet. `https://en.wikipedia.org/wiki/Intranet`, Listopad 2015.

[6] Wikipedia. Javascript. `https://en.wikipedia.org/wiki/JavaScript`, Listopad 2015.