**Referat**

Przedmiot: Pracowania dyplomowa II

Student: Artur Ziemba

Grupa: C

1. Krótkie przedstawienie

Node.js jest cross-platformowy, działającym niezależnie od środowiska językiem programowania, napisanym w językach c/c++ oraz javascript, wydanym 27 marca 2009 roku, zaprojektowanym przez Ryana Dahla. Głównym jego zastosowaniem jest oprogramowanie serwerów i narzędzi sieciowych, działających po stronie serwera. Język ten cieszy się dużą popularnością, dzięki czemu, mimo względnie krótkiego okresu życia środowiska, zaowocowało ogromną ilością projektów open-source, tysiącami członków należących do społeczności okołojęzykowej oraz powstaniem wydarzeń takimi jak NodeConf, Node Interactive. Obecnie wiele najwiekszych firm korzysta z Node.js. Groupon, IBM, Linkedln, Microsoft, Netflix, PayPal, Yahoo.

1. Przeznaczenie

Node.js zalecany jest do tworzenia aplikacji z dużą liczbą operacji wejścia/wyjścia, strumieniowania danych np. video, Single Page Applications, aplikacji udostępniających API w formacie JSON, z intensywną wymianą danych w czasie rzeczywistym na wielu urządzeniach, np. portalach społecznościowych. Jest on szybki i lekki, może być stosowany do pisania bramek API - opisuje, jak poszczególne elementy lub warstwy oprogramowania komunikują się ze sobą. Node.js pozwala na zoptymalizowanie pracy oraz uzyskanie skalowalności dzięki asynchronicznemu przetwarzaniu danych dostarczanych do aplikacji, w związku z czym idealnie nadaje się do obsługi komunikacji wymagającej pracy w czasie rzeczywistym. Funkcje napisane w Node.js wykonują się równolegle, korzystając z tak zwanych wywołań zwrotnych (ang. callback). Dzięki temu nie powstaje problem blokowania określonych funkcjonalności programu w czasie pracy innych niezależnych jego części. Przy pomocy wywołań zwrotnych możemy zapewnić zasygnalizowanie lub zwrócenie uzyskanych wyników, bądź obsługę błędu powstałego w czasie działania bloku kodu.

1. Modułowość

Praca z Node.js opiera się głównie o korzystanie ze zbioru modułów - biliotek wspierających określone funkcjonalności. Zapewniają one pracę między innymi z plikami systemowymi, z urządzeniami wejścia/wyjścia, protokołami internetowymi (dns, http, tcp, spdy), plikami binarnymi, źródłami danych oraz funkcjami kryptograficznymi. Zmniejszają one złożoność czyli nakład pracy przy tworzeniu własnej funkcjonalności. Dzięki wsparciu package managera (od roku 2010) nazywanego npm, programiści mogą bez przeszkód udostępniać napisane przez siebie moduły i biblioteki lub w prosty sposób zaimportowywać ogólnie dostępne moduły i używać ich w swoich projektach. Najpopularniejszymi modułami są Express.js, Socket.IO, czy Meteor. Npm jest automatycznie włączony w środowisko Node.js. Jest obsługiwany za pomocą linii komend systemu operacyjnego. Ilość ogólnodostępnych modułów to ok 0.5 mln. Npm korzysta, tak jak i inne globalnie działające narzędzia JavaScriptowe, z plików zależności w formacie json. Opisują one wersję wykorzystywanych modułów i pozwalają za pomocą jednoliniowej komendy na szybką i łatwą instalację wszystkich używanych pakietów w lokalnym środowisku.

1. Paradygmat

Architektura Node.js pozwala na tworzenie oprogramowania sterowanego zdarzeniami. Jest to paradygmat programowania, w którym kolejność wykonywania kodu zależy od zdarzeń mających miejsce w czasie życia aplikacji (run time), na przykład interakcji użytkownika, czy otrzymania określonych sygnałów. Kiedy aplikacja pełni rolę serwera to przetwarzania zapytań otrzymywanych ze strony klienta oraz uruchomionych przez nie zdarzeń. Przy tym paradygmacie wyróżniamy pętlę główną, która jest odpowiedzialna za wywoływanie triggerów. W Node.js jest to na przykład funkcja nasłuchująca określonego adresu, pod który klienci kierują określone zapytania. Paradygmat ten daje możliwość jednoczesnego wykonywania paru funkcjonalności – np. jednoczesnego zapisywania danych do bazy, przetwarzania innej części zapytania i przygotowywanie odpowiedzi w jednym okresie czasu. Node.js zapewnia w ten sposób działanie asynchroniczne, bez bezpośredniego użycia technologii wielowątkowej.

1. Asynchroniczność

Node.js pracuje wykorzystując tylko jeden wątek, używając technologii asynchronicznego wejścia/wyjścia. Zapewnia to synchronizację wykonywania wielu operacji bez konieczności czekania na zakończenie operacji i zwolnienie dostępu do zasobu. Wywołanie asynchroniczej funkcji nie czeka na wyniki, dzięki czemu nie blokujemy wątku wykonawczego. Po zakończonym wywołaniu uruchomiona zostaje funkcja zwrotna. Pomimo, że Node.js działa na jednym wątku, dzięki asynchroniczności potrafi obsłużyć więcej zapytań niż np. serwer HTTP Apache. Node korzysta ze wzorca obserwatora który dokonuje operacji na zbiorze obiektów obserwujących według swojego stanu. W celu obsługi asynchronicznego wejścia/wyj scia korzysta z biblioteki multiplatformowej języka c - libuv. Wątki te działają na wielu rdzeniach systemu, na którym działa serwer. Wątek główny przydziela zadania dla wątków kolejno ze współdzielonej kolejki funkcji. Kiedy wątek zakończy wykonywanie przydzielonego mu zadania, informuje o tym wątek główny. Biblioteka libuv zajmuje się odpowiednim podziałem zadań oraz przydzieleniem zasobów tak, aby w jak najlepszy sposób wyważyć nakład pracy między wieloma wątkami.

1. Architektura komunikacji

W tradycyjnym podejściu odpowiedzialne za obsługę przychodzących połączeń są określone wątki lub procesy sytemu operacyjnego, które w porównaniu z technologią Node.js wymagają względnie więcej zaalokowanych zasobów. W celu obsługi zapytania przychodzącego do aplikacji Node.js, program rejestruje się w systemie operacyjnym, a każde otrzymane zapytanie alokuje zasoby na stercie programu i wywołuje określone wywołanie zwrotne. Najbardziej popularna metoda na zarządzanie komunikacją między instancjami korzystającymi z serwera używającego technologii Node.js jest wykorzystanie frameworku Restful Api - Rest. Jest to wzorzec architektury oprogramowania, który opisuje sposób operowania zapytaniami pomiędzy Api, w prosty sposób poprzez obsługę zadań oraz odpowiedzi. Daje możliwość na stworzenie komunikacji bez potrzeby wiedzy na temat instancji korzystających z przesyłanych zasobów - może integrować środowiska napisane w różnych językach, działające na różnych platformach. Wykorzystuje on protokuł http - post, get, put, delete, options, head, trace oraz connect. Opis typu zasobów, jakie wymieniamy, jest określony w nagłówku http. Rest wykorzystuje schemat wymiany json lub xml.

1. Specyfikacja problemu

Stworzony miał zostać system do zarządzania tablicami zadań, który musi posiadać:

autoryzowany dostęp dla użytkowników systemu,

intuicyjnej aplikacji,

współpracy z innymi użytkownikami,

przejrzysty sposób prezentować proces realizacji poszczególnych zadań zorganizowanych w tablicach.

Właściciel określonej tablicy powinien mieć możliwość zarządzania dostępem,

Wymiana komunikatów w modelu klient-serwer.

System współpracuje z bazą danych.

1. Mean Stack

Do opracowania rozwiązania zdecydowałem się skorzystać z Mean Stack. Skrót odnosi się do frameworków oraz technolgii Mongodb, Express, AngularJS oraz Node.js. Do użycia wszystkich wymagany jest tylko jeden język programowania - javaScript, zarówno do obsługi warstwy frontendowej aplikacji jak i backendowej. Jako iż Technologia Node.js została już opisana, przejdę do opisu pozostałych technologii.

MongoDB

system zarządzania bazą danych zorientowany na dokumenty. Operuje na nierelacyjnych bazach danych. czyli No sql Używa struktur json jako schematów budowy bazy danych.

ExpressJS

Framework służący do szybkiego wymagającego jak najmniejszych nakładów pracy wytwarzania zarówno backendu aplikacji internetowych, jak i aplikacji mobilnych. Jest to najpopularniejszy framework do tworzenia serwerów w technologii Node.js.

AngularJS

Wykorzystujący wzorzec projektowy MVC/MVP/MVP polegający na oddzieleniu od siebie poszczególnych warstw aplikacji - logiki, widoku oraz kontrolera, framework wykorzystujący dodatkowe tagi w języku html w celu prostego w obsłudze i niewymagającego dodatkowej logiki tworzenia dynamicznych stron internetowych.

1. Komunikacja

Do komunikacji między warstwą frontendową i backendowową po stronie klienta wykorzystałem wysokopoziomową bibliotekę Ajax oraz technologię Restful api. Ajax polega na zarządzaniu asynchroniczną komunikacją. Dzięki temu aplikacja może wykonywać inne funkcje, mimo oczekiwania na odpowiedź ze strony serwera, za sprawą odseparowania warstwy wymiany danych od pozostałych warstw aplikacji. Do opisu wymienianych struktur użyłem standardu json. Angular prezentuje użytkownikowi dynamiczną aplikację internetową, odpowiada za przyjmowanie danych. z pomocą ajaxa wysyła oraz odbiera dane wysyłane do serwera. Serwer Node.js z użyciem technologii express, wykorzystując metodę routingu, dopasowuje zapytanie do odpowiednich funkcji serwisu, przetwarza otrzymane dane. Baza danych zarzadzana przez Mongodb przechowuje informacje użytkowników serwisu oraz w odpowiedzi zwraca odpowiednie dane z powrotem do expresa i on spowrotem do frontendu prezentując dane użytkownikowi.

1. Testy aplikacji – manualne

Do testów została użyta przeglądarka Google Chrome oraz skrypty node.js korzystające z modułu request. Każdy test został wykonany ok 5 razy, a podane wartości przedstawiają średnią czasu pracy.

Testy wydajnościowe - zachowanie serwisu przy różnym obciążeniu bazy danych.

Mimo dużej ilości posiadanych informacji korzystanie z serwisu jest komfortowe, ponieważ czas obsługi zapytania nie zostaje znacząco wydłużony.

Testy obciążeniowe serwera - zachowanie aplikacji w przypadku równoczesnej obsługi wielu przychodzących zapytań. Przy dużej liczbie przychodzących zapytań użytkowanie serwisu jest wciąż komfortowe. Aplikacja wykazuje się skalowalnością.

Testy sprawnościowe serwera - ilość kolejno wysłanych po sobie zapytań, które zostały obsłużone przez serwer w określonym czasie.

Osiągnięte wyniki są zadowalające, jednak łatwo zauważyć, że technologia ta wykazuje się największą sprawnością przy obsłudze równoległych zapytań. Średni uzyskany czas dla obsługi równoległych zapytań jest prawie 2 razy krótszy niż wysłanych kolejno.

1. Podsumowanie

Zostały osiągnięte wszystkie założenia oraz wymagania aplikacji. Stworzony projekt doskonale nadaje się do użytkowania przez zorganizowane zespoły do wykonywania określonych zadań. Dostarcza przejrzysty interfejs oraz nieskomplikowaną prezentację danych dla końcowych użytkowników. Niewątpliwie największymi zaletami tej technologii jest łatwość budowania wymagających serwisów internetowych poprzez użycie asynchronicznej obsługi wejścia/wyjścia, które pozwala na przetwarzanie wielu funkcji w jednym czasie, tworząc wyjątkowo szybkie w działaniu aplikacje, dzieki czemu jest idealnym rozwiązaniem w kwestii pracy nad wieloma rozwiązaniami webowymi.