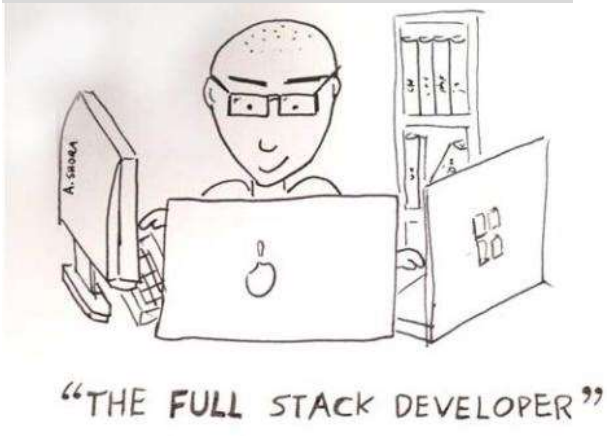
Wprowadzenie do Aplikacji Internetowych (Webowych)

dr inż. Grzegorz Rogus
rogus@agh.edu.pl

Tematyka wykładów

Od zera do webDevelopera



Plan wykładów

1. Architektura nowoczesnych aplikacji webowych – omówienie stosu technologicznego

2. Prezentacja standardów:

3. Nowoczesny JS
   ECMS7 (2016), TypeScript

Plan wykładów cd

4, 5. Frontend:

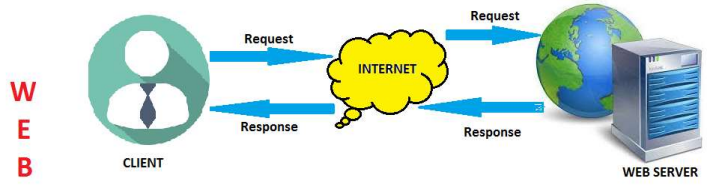
5, 6 . Backend:

7. Autentykacja
   Web Socket
   Biblioteka RxJs

Czym są Aplikacje Webowe

Perspektywa użytkownika końcowego

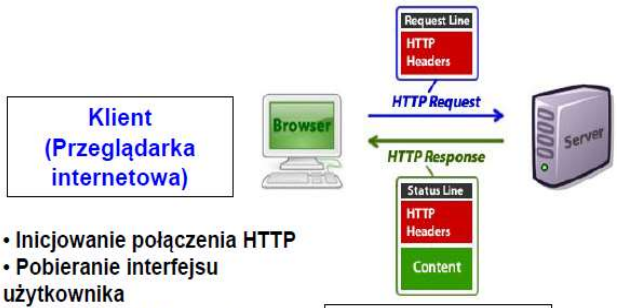
Rzeczywista architektura



W E B TECHNOLOGY

Client - Server Architecture

Podstawowa architektura: klient-serwer



Klient (Przeglądarka internetowa)

- Inicjowanie połączenia HTTP
- Pobieranie interfejsu użytkownika
- Prezentacja interfejsu użytkownika
- Interakcja z użytkownikiem
- Buforowanie odpowiedzi
- Kryptografia

Protokół HTTP

- Oparty na TCP
- Komendy tekstowe
- Transmisja 8-bitowa
- Bezstanowy, bezsesyjny

Serwer aplikacji oraz danych

- Obsługa żądań HTTP
- Rejestracja żądań
- Uwierzytelnianie i kontrola dostępu
- Kryptografia
- Wybór wersji językowej wysyłanych plików

Klient HTTP

Klient HTTP - przeglądarka - jest programem użytkowym, który odpowiada za:

- wysyłanie żądań pobrania dokumentów,
- wizualizację pobieranych dokumentów
- obsługę interakcji z użytkownikiem końcowym.

Serwer HTTP

Serwer HTTP - serwer WWW - program nieprzerwanie pracujący, obsługujący repozytorium dokumentów (np. HTML), które udostępnia sieciowym klientom HTTP.

Do zadań serwera HTTP należy:

- obsługa żądań HTTP i ich rejestracja w plikach dziennika (log files),
- uwierzytelnianie i kontrola dostępu użytkowników końcowych za pomocą nazwy i hasła,
- kryptograficzne szyfrowanie komunikacji sieciowej z klientem http,
- automatyczny wybór odpowiedniej wersji językowej dokumentu.

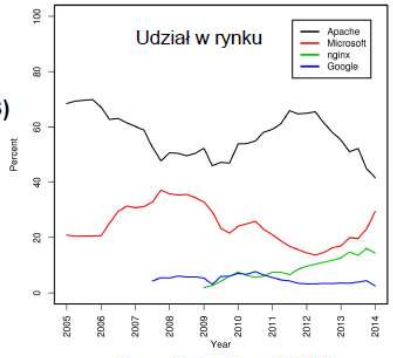
Serwery obsługujące protokół HTTP/HTTPS (Web-Serwery)

WebServer - komputer (serwer) - a najczęściej klaser komputerowy - obsługujący żądania HTTP/HTTPS, za pomocą odpowiedniego oprogramowania.

Główną a jednocześnie podstawową funkcjonalnością WebServerów jest przechowywanie, przetwarzanie i dostarczanie stron internetowych zapisanych za pomocą języka HTML, oraz obrazów, multimediów i skryptów, o które dodatkowo wzbogacone są strony internetowe.

Najpopularniejsze oprogramowanie WebServerowe:

- **APACHE**
- **Microsoft Internet Information Services (IIS)**
- **Nginx**
- **Google Web Server (GWS)**



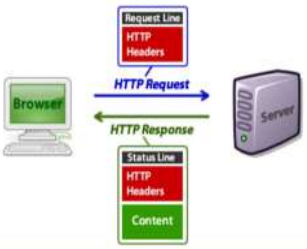
(dane Netcraft, maj 2014)

Protokół Http

Hyper Text Transfer Protocol (HTTP/1.1)
[Protokół przesyłania dokumentów Hiper-Tekstowych]

- **Oparty na TCP,**
- **Komendy tekstowe,**
- **Transmisja 8-bitowa,**
- **Bezstanowy i bezsesyjny.**

RFC 2616
(czerwiec 1999)
RFC (Request for Comments)



Bezstanowość" *(ang. stateless)*, co oznacza że nigdzie nie istnieje zapis stanu poprzednio wykonanych operacji, a kolejne transakcje są wykonywane niezależnie.

Najważniejsze metody protokołu HTTP:

- **HEAD** - wysyła żądanie przesłania nagłówka zawierającego meta-dane (informację), bez przesyłania samego zasobu.
- **GET** - wysyła żądanie pobrania konkretnego zasobu URI (np. strony internetowej napisanej w języku HTML).
- **POST** - wysyła żądanie do serwera akceptacji zasobu dołączonego do żądania.

pozostałe: **PUT, DELETE, TRACE, OPTIONS, CONNECT.**

Zakres kodów	Znaczenie
100 - 199	Informacyjne.
200 - 299	Żądanie (od klienta) powiodło się
300 - 399	Żądanie klienta zostało przekazane, wymagane są dalsze działania.
400 - 499	Żądanie klienta nie powiodło się.
500 - 599	Błąd serwera.

Nagłówek Http

Jak mówi klient (przeglądarka)?

http request

Linia startowa	Metoda sciezka http/wersja
Nagłówek	Nazwa1: wartość1 Nazwa2: wartość2
Ciało zapytania	dane, które wysyłamy do serwera (opcjonalnie)

Jak mówi serwer?

http response

Linia startowa	http/wersja kod statusu
Nagłówek	Nazwa1: wartość1 Nazwa 2: wartość2
Ciało zapytania	Zawartość pliku (opcjonalnie)

Nagłówek http - przykład

Jak mówi klient (przeglądarka)?

http request

```
GET /index.html HTTP/1.1

Host: www.example.com
UserAgent:
Mozilla (Ubuntu;Linux)
Firefox 69
Cookie: ala=makota
Accept: text/html
AcceptLanguage: en-us
```

Jak mówi serwer?

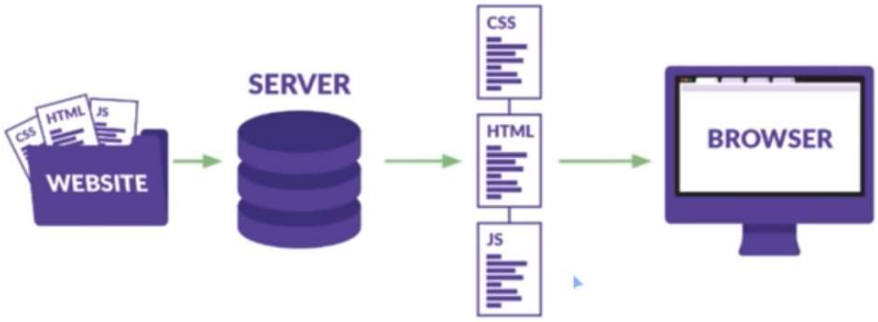
http response

```
HTTP/1.1 200 OK

Date: Mon,23 May 2018 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)
(RedHat/Linux)
ContentLength: 438
Connection: close
ContentType: text/html;
charset=UTF8

Czterysta trzydziesci osiem bajtów
odpowiedzi...
```

Siła drzemie... w nagłówkach

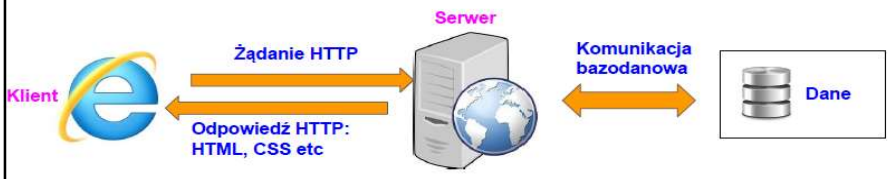
```
w standardzie HTTP 1.1 otrzymujemy:
31 nagłówków zapytania
34 nagłówków odpowiedzi
```

Model współpracy przeglądarka - serwer



Podstawowa architektura: klient-serwer

Klasyczny model komunikacji synchronicznej HTTP:



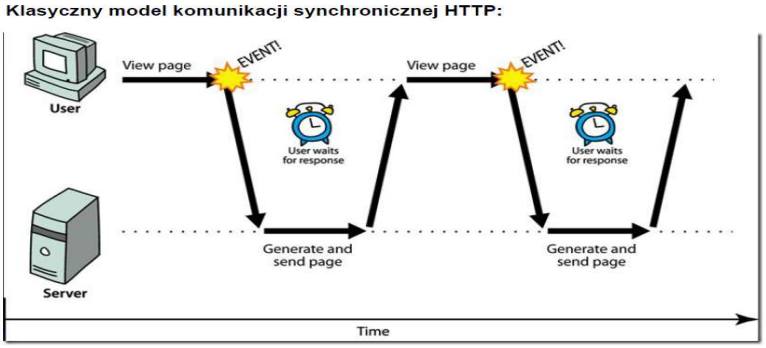
Jest to model synchronicznej komunikacji HTTP, gdzie klient wysyła żądanie, serwer je odbiera następnie przetwarza (np. komunikuje się z bazą danych) i na końcu generuje odpowiedź.

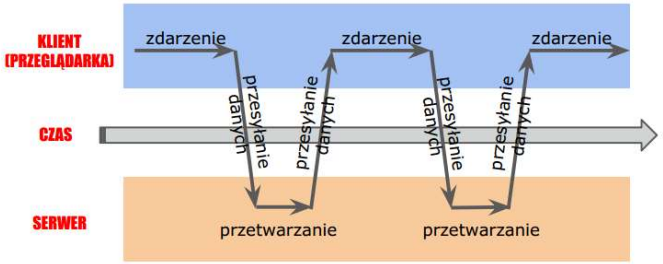
W sytuacji takiej klient musi czekać z kolejnym żądaniem do momentu kiedy nie dostanie odpowiedzi od serwera.

W modelu synchronicznym mamy bardzo mały poziom aktywności oraz interaktywności strony, strona musi być przeładowana (pobrana) po każdym żądaniu klienta, jeżeli strony są złożone to proces ten jest długi.

Podstawowa architektura: klient-serwer

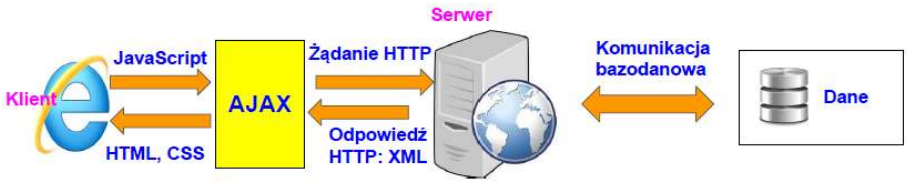
Klasyczny model komunikacji synchronicznej HTTP:





Podstawowa architektura: klient-serwer

Model komunikacji asynchronicznej HTTP:

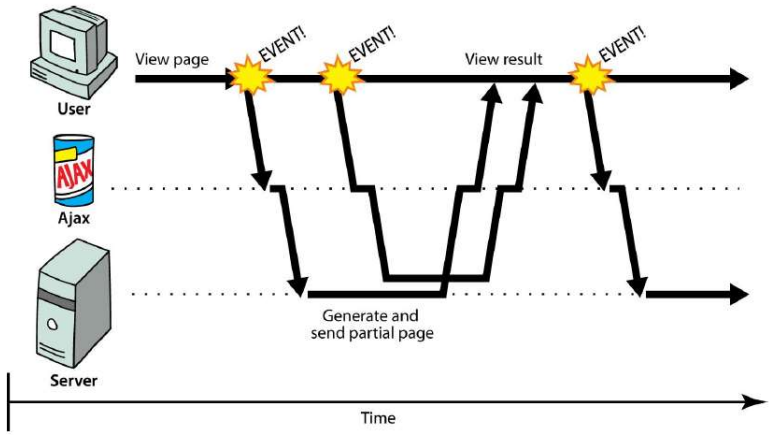


Jest to model asynchronicznej komunikacji HTTP, gdzie klient (przeglądarka) nie czeka na przyjście odpowiedzi na żądanie serwera, a wykonuje dalsze żądania. Pośredniczy w tym wbudowany w przeglądarkę mechanizm silnika AJAX.

W takim modelu nie ma konieczności przeładowania strony przy każdej operacji klienta, wystarczy, że zostaną odczytane brakujące dane, a dzięki odpowiednim narzędziom zmodyfikowana zostanie zawartość strony.

Podstawowa architektura: klient-serwer

Model komunikacji asynchronicznej HTTP:

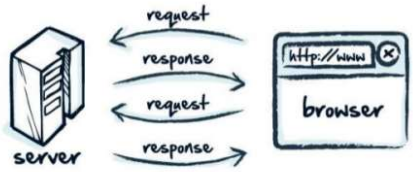


Od strony WWW do aplikacji internetowej

Na początku była strona statyczna

Tradycyjna strona internetowa

- Korzysta z HTML5, CSS3, JavaScript
- Bazuje na klasycznej architekturze klient-serwer
- Zawiera wiele stron



Strona internetowa WWW

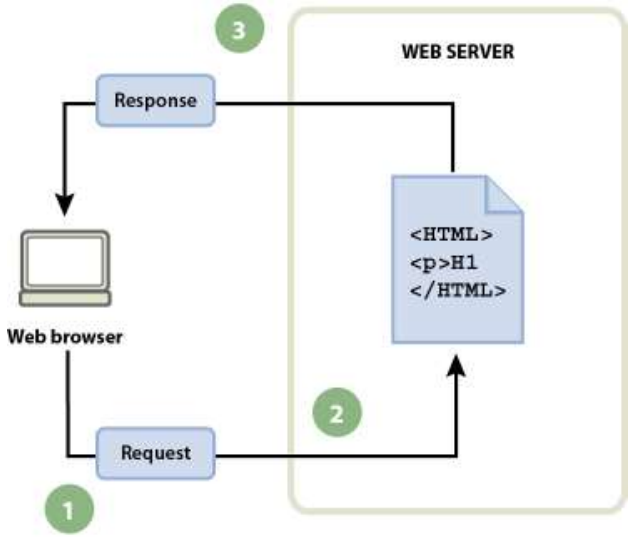
Strona Internetowa WWW - kolekcja logicznie połączonych ze zobą zasobów (najczęściej plików napisanych w języku **HTML** oraz multimediów), znajdujących się na jednym serwerze (w konkretnej kartotece) obsługującym żądania HTTP.

http://www.google.pl

http://www.onet.pl

- Zasoby w ramach jednej strony internetowej są ze sobą połączone za pomocą tzw. hiperłączy które wskazują na URL konkretnego zasobu.
- Pojęcie "*strony internetowej*" (inaczej witryny internetowej) łączy w sobie: dane (treść strony), prezentację (formatowanie) oraz logikę (strukturę).

Proces dostarczenia statycznej strony - podsumowanie

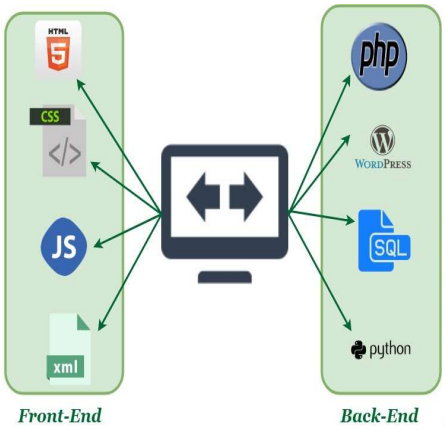


Jak rozróżniać stronę internetową od aplikacji dynamicznej

Strona internetowa (Statyczna strona WWW) - strona taka zawiera w kodzie dane, które są wyświetlane w przeglądarce internetowej. Zawartość strony nie zmienia się pod wpływem interakcji z użytkownikiem, zawsze wyświetlane są te same treści. Każda zmiana danych (treści strony) wymaga ingerencji programisty w kod strony. Użytkownik nie posiadający wiedzy na temat struktury strony oraz języka HTML nie będzie w stanie zmienić zawartości strony.

Aplikacja internetowa (Dynamiczna strona WWW) - strona dynamiczna jest generowana przez serwer HTTP pod wpływem żądań przychodzących od klienta na podstawie przesłanych parametrów i zmiennych. Strony takie dostosowują swoją zawartość w zależności od działań użytkownika w przeglądarce. W takim wypadku zmiany stanu strony mogą być wykonywane po stronie użytkownika dzięku wykorzystanie języków skryptowych np. JavaScript lub po stronie serwera wykorzystując do tego celu języki takie jak PHP, Perl, Python. W nowoczesnych aplikacjach internetowych wykorzystywane są obie metody. Dodatkowo najczęściej dane (czyli zawartość strony) jest przechowywana w bazach danych (np. SQL), z których są pobierane w zależności od wywoływanych przez użytkownika żądań.

Technologie Web

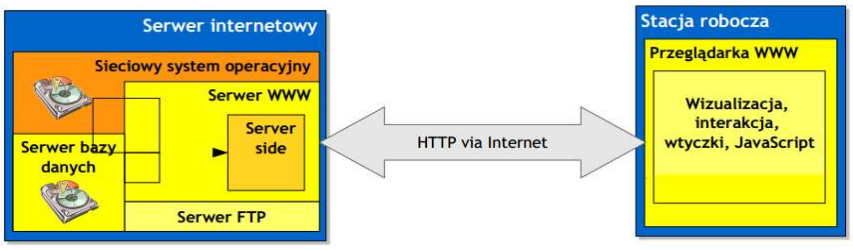


FRONT-END - pojęciowo odnosi się w technologiach internetowych do kodu wykonywanego po stronie użytkownika. W ogólności do tej kategorii można zaliczyć HTML, CSS oraz JavaScript.

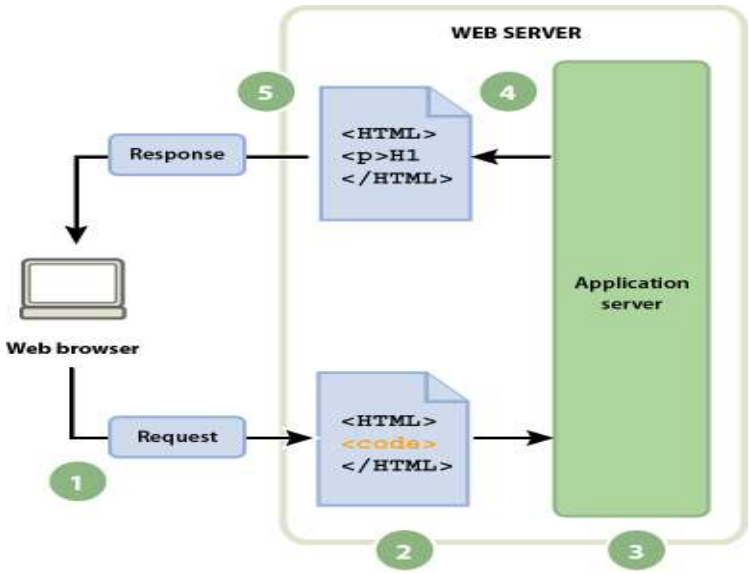
BACK-END - pojęciowo odnosi się w technologiach internetowych do kodu wykonywanego po stronie serwera. W ogólności do tej kategorii można zaliczyć PHP, Perl, CGI, Ruby, Java, C#, itp.

Rozwarstwienie warstwy serwera — model klient-serwer, strony dynamiczne

- Aby zawartość serwisu mogła się często zmieniać, treść musi być generowana po stronie serwera przez działające tam oprogramowanie (*server side*).
- Gdy klient zażąda dokumentu, oprogramowanie warstwy serwera pobiera aktualną zawartość z *zasobów dyskowych* serwera oraz *bazy danych*, buduje dynamicznie treść dokumentu HTML i przekazuje do warstwy klienta.
- Jeżeli dane na serwerze zostaną zmodyfikowane, następne odwołanie do dokumentu spowoduje ponowne zbudowanie aktualnej treści.
- Oprogramowanie *server side*, tworzy kolejną, połowiczną warstwę modelu.

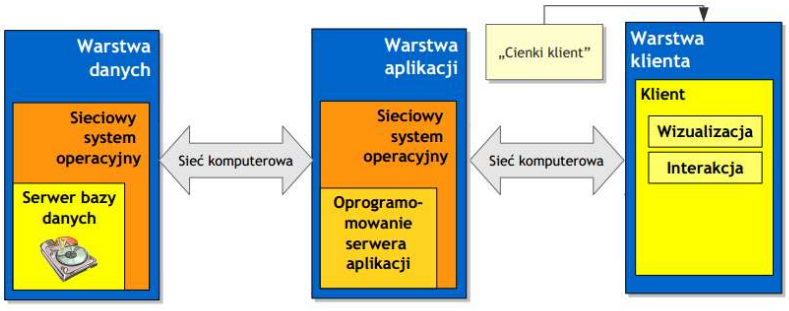


Proces dostarczenia strony dynamicznej – w praktyce

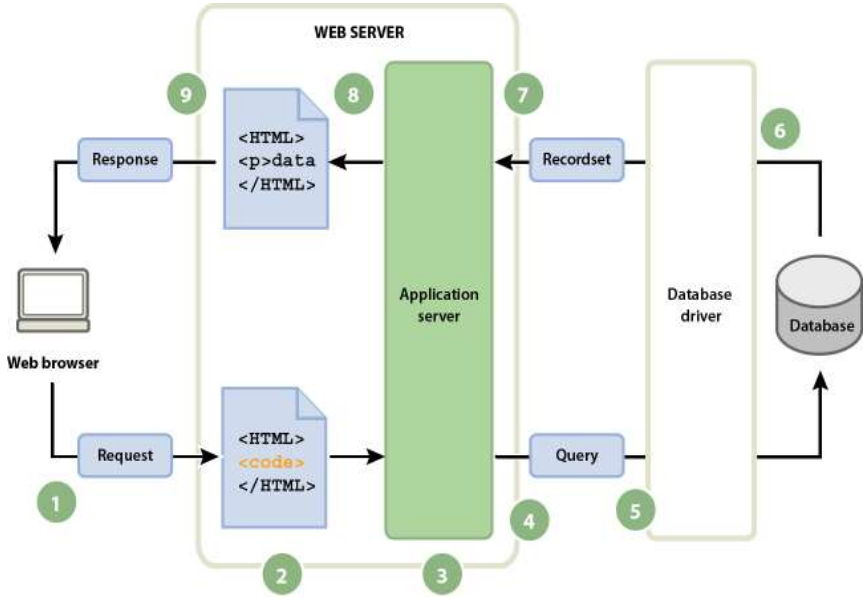


Rozwarstwienie warstwy serwera — model trójwarstwowy model klient-serwer

- Oprogramowanie działające po stronie serwera może być bardzo rozbudowane.
- Przejmuje ono na siebie wszystkie ważne funkcje przetwarzania danych zgodnie z tzw. *modelem biznesowym*.
- Oprogramowanie to realizuje funkcje *warstwy aplikacji* i staje się *trzecią warstwą* w *trójwarstwowym* modelu klient-serwer.



Generacja strony dynamicznej wersja szczegółowa



Dygresja: Wzorzec MVC

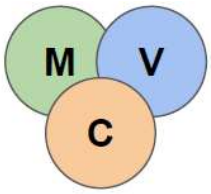
Model-View-Controller (MVC) ***[Model-Widok-Kontroler]*** **- jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:**

- **Model (dane / logika)**
- **Widok (prezentacja danych)**
- **Kontroler (interakcja z użytkownikiem + sterowanie aplikacją)**

Można go zaimplementować bez użycia bibliotek czy specjalistycznych platform programistycznych, stosując jasne reguły podziału na konkretne komponenty w kodzie źródłowym. W ten sposób każdy komponent aplikacji można niezależnie od siebie rozwijać, implementować i testować.

Wzorce projektowe - MVC

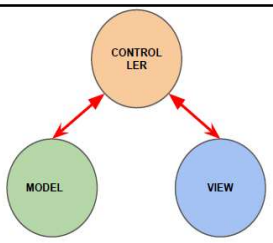
Jak działa MVC ?



1. Użytkownik wykonuje (określoną) czynność w aplikacji.
2. Wyzwalana jest procedura obsługi zdarzenia (wywołanego przez użytkownika) w kontrolerze.
3. Kontroler pozyskuje dane z modelu i następuje przekazuje je do widoku.
4. W widoku dane są prezentowane użytkownikowi.

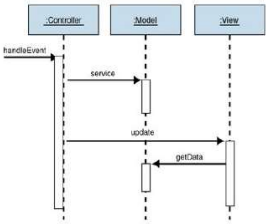
Przykład synchroniczny czat:

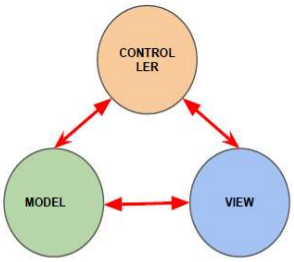
1. Użytkownik wpisuje wiadomość na czacie.
2. Wyzwalana jest procedura obsługi przekazania wiadomości.
3. Kontroler komunikuje się z modelem i zapisuje nową wiadomość.
4. Kontroler uaktualnia widok
5. Użytkownik widzi nową wiadomość w oknie czatu.



Model Pasywny

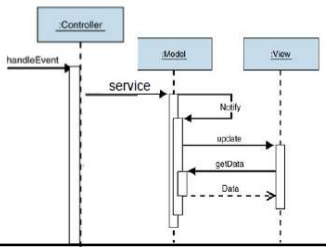
W modelu "pasywny" nie ma wymiany danych pomiędzy modelem a widokiem z pominięciem kontrolera, wszystkie akcje są wywoływane i sterowane przez kontroler.





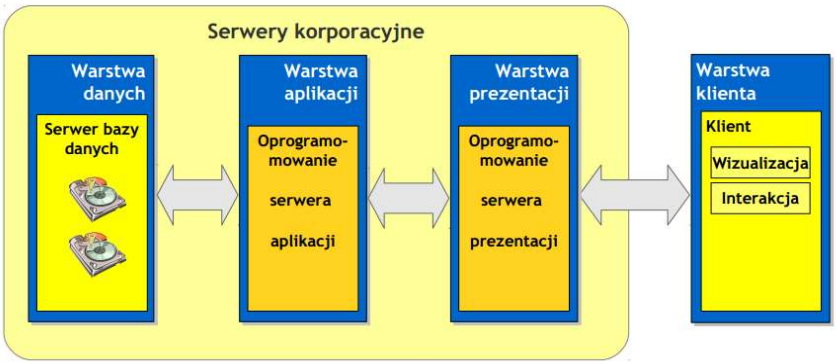
Model Aktywny

W model aktywnym model może bezpośrednio przekazywać dane do widoku z pominięciem kontrolera.

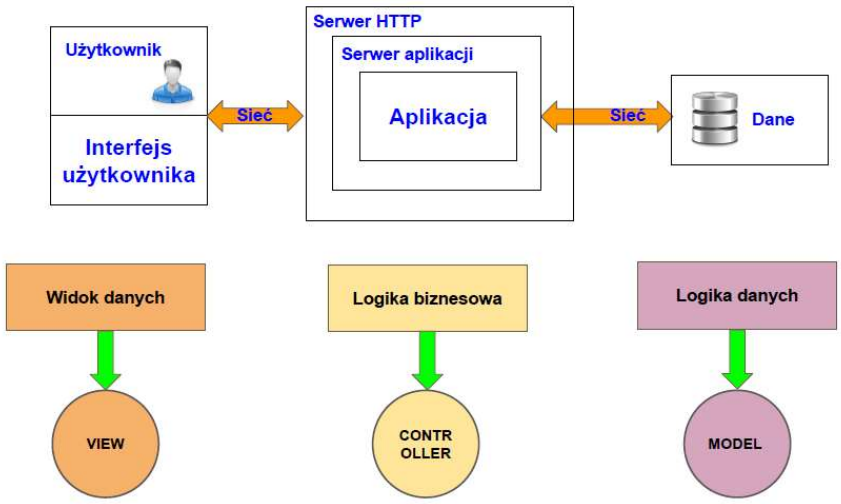


Rozwarstwienie warstwy serwera — model trójwarstwowy model klient-serwer

- Oprogramowanie warstwy aplikacji powinno być niezależne od tego, na jakim urządzeniu pracuje warstwa klienta. Tworzenie ostatecznej postaci dokumentu powinno być z niej wyłączone.
- Przygotowaniem *ostatecznej wersji* prezentowanych treści, odpowiednio do charakteru klienta zajmuje się *warstwa prezentacji*.



Architektura klient-serwer w modelu WWW



Zmiany w tworzeniu stron internetowych

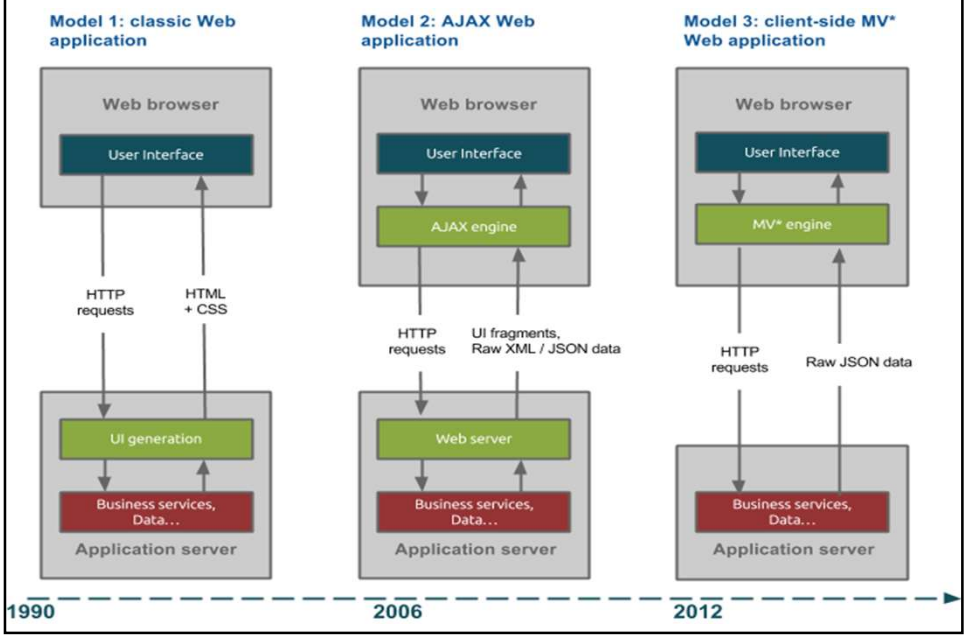
Warunki:

- nowoczesne przeglądarki,
- rozwój języka JavaScript,
- większy nacisk położony na wygodę użycia.

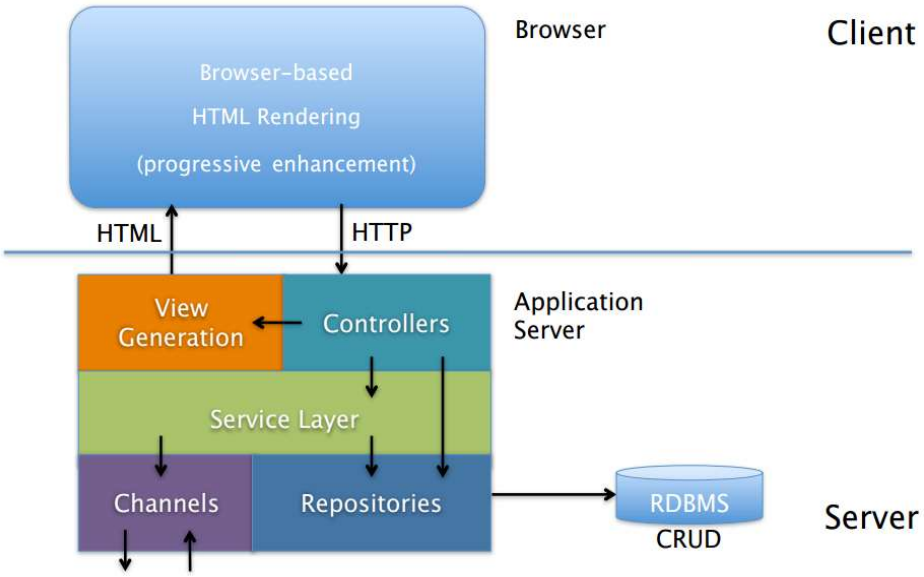
Zmiany w aplikacjach internetowych:

- z punktu widzenia użytkownika działają jak aplikacje desktopowe,
- szybki i dużo bardziej interaktywny interfejs,
- potrafią działać nawet offline,
- działają na wielu platformach (RWD).

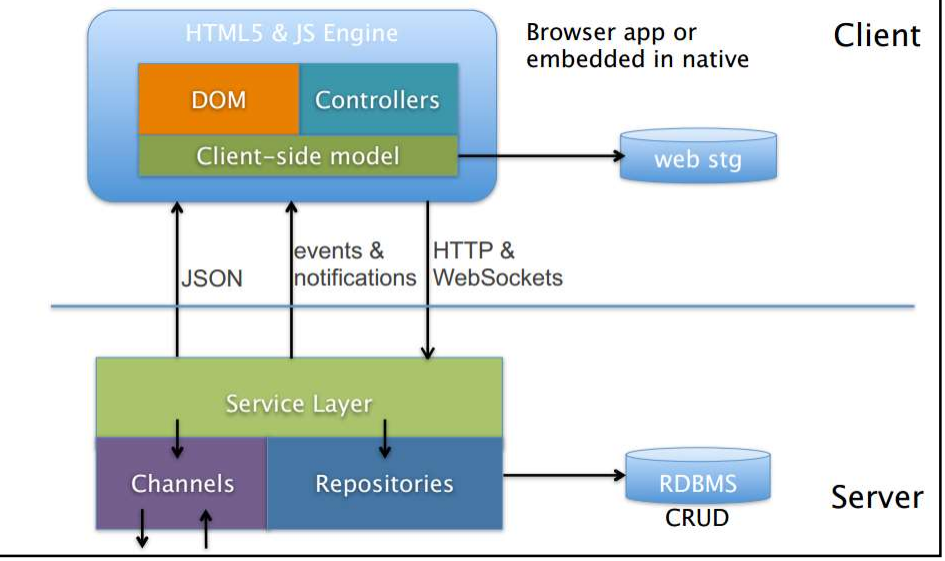
Typy modeli app webowych



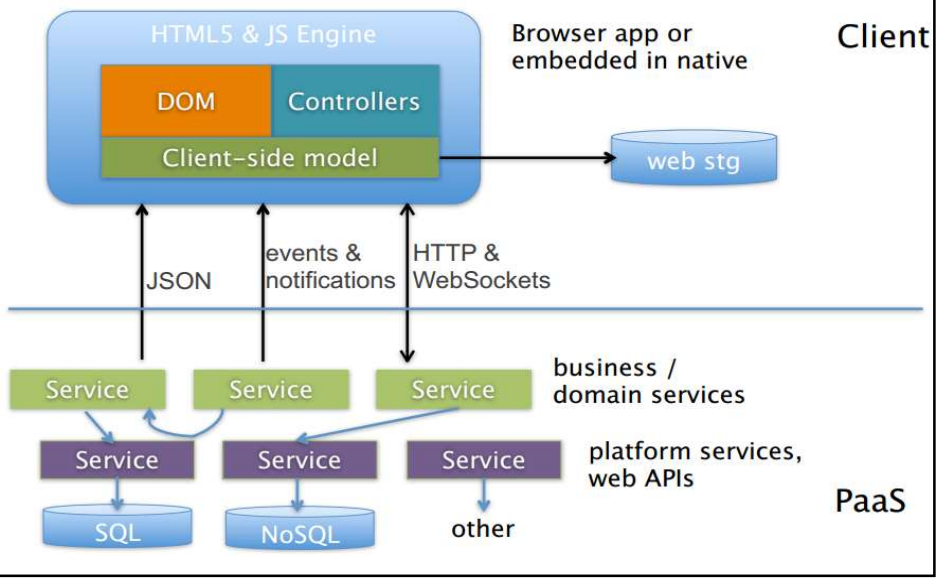
Od server-side app w kierunku inteligentnych klientów i serwisów

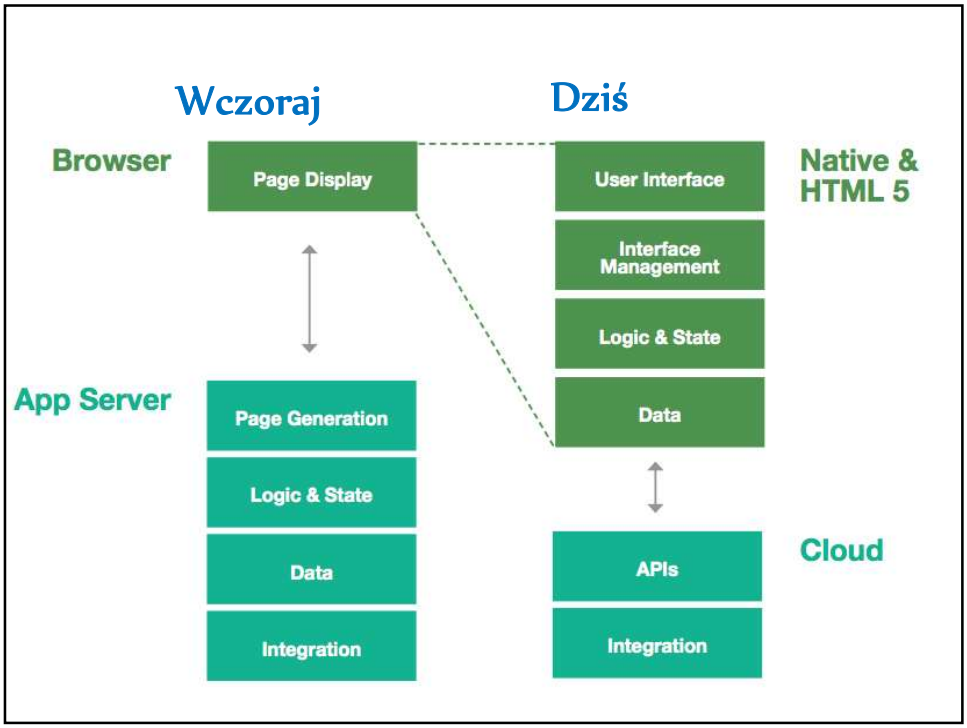


Od server-side app w kierunku inteligentnych klientów i serwisów



Od server-side app w kierunku inteligentnych klientów i serwisów

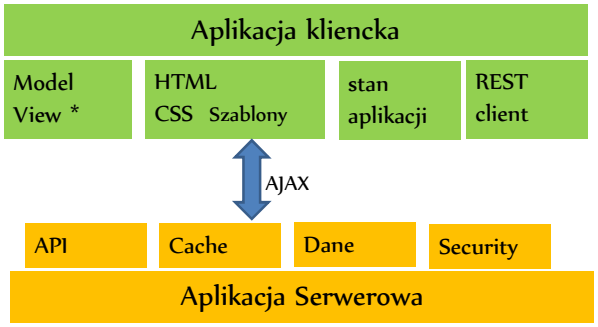




Aplikacja SPA (Single Page Aplication)

- wysoka wydajność
- user experience
- współpraca frontend – backend
- możliwość zmiany backendu niezależnie od frontendu (i vice versa)
- ten sam backend dla wielu aplikacji

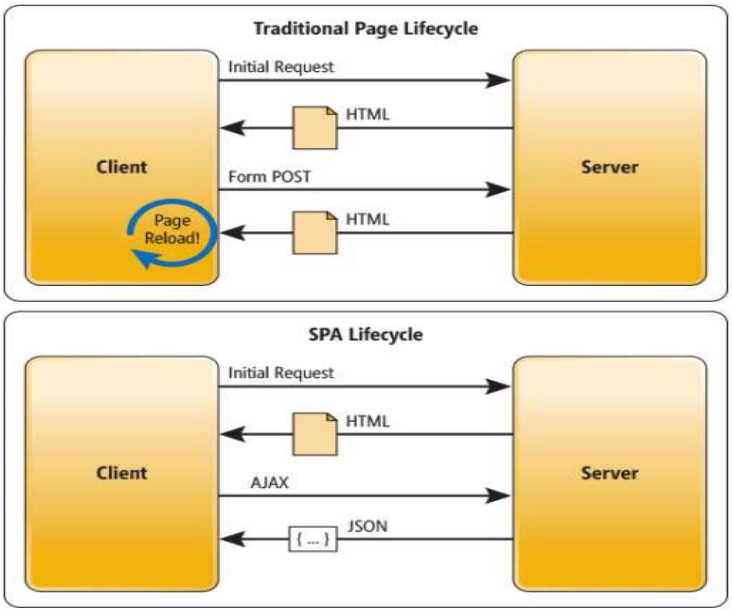
- SEO
- Kompatybilność miedzy przeglądarkami



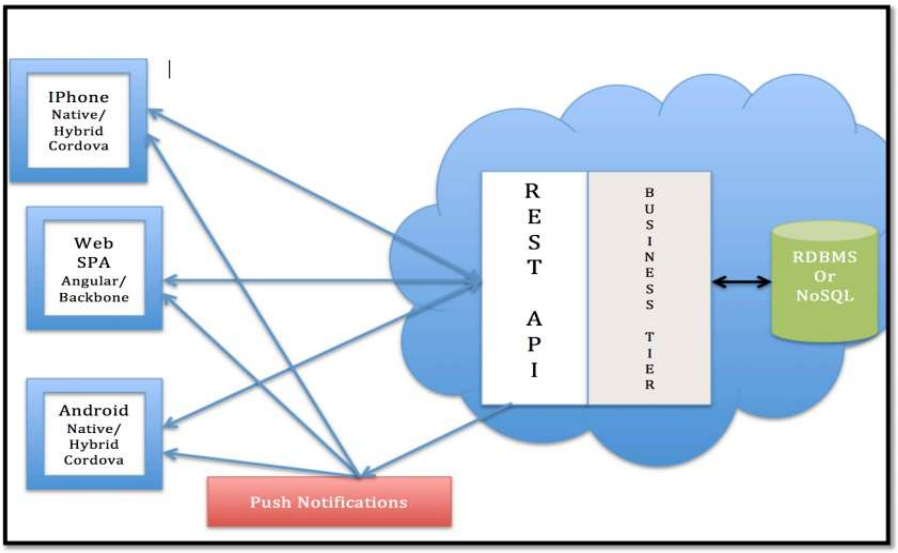
Single Page Applications (SPA)

- Strona startowa aplikacji jest jedyną stroną pobieraną w całości z serwera (potem interakcje Ajax, WebSocket)
- Strona aplikacji nie przeładowuje się w czasie pracy z nią
- Nie następuje nawigacja do innych stron
- Zmienia się stan (i wygląd) strony w przeglądarce
- User Experience (UX) podobny do aplikacji desktopowych
- Technologie: HTML5, CSS, JavaScript, Ajax, WebSocket
- Frameworki: Angular, ReactJS, Vue, …

SPA w akcji



Nowoczesna architektura Web



Nowoczesna architektura aplikacji Webowych - podsumowanie

- jeden backend wspólny dla wielu frontendów/klientów
  - Aplikacja HTML-owa
  - Natywna aplikacja mobilna
  - Inna aplikacja korzystająca z API
- backend nie generuje żadnego HTML-a (tylko przy inicjalizacji)
- odpowiada jedynie na żądania HTTP
- na wyjściu generuje XML lub JSON
- frontend HTML-owy napisany w JavaScript
- komunikacja z backendem z wykorzystaniem AJAX
- komunikacja fronten – backend z wykorzystaniem RestAPI (coraz częściej GraphQL)

Single Page Interface (SPI)

- Czy model SPA jest odpowiedni tylko dla aplikacji webowych (web applications) czy również dla witryn (web sites)?

– Pytanie otwarte, różne stanowiska

– Głos „na tak”: „The Single Page Interface Manifesto” (http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php)

- Problemy przy tworzeniu witryn mocno opartych o Ajax:

– Zakładki do „stanu strony” po jej zmianie Ajaksem, historia przeglądarki

– Indeksowanie przez wyszukiwarki (Search Engine Optimization (SEO))

– Model biznesowy oparty o liczbę odwiedzin stron

– Potrzeba wyskakujących okienek (pop-up)

– Wolne pierwsze wyświetlenie strony

Stan aplikacji SPA/SPI

- W tradycyjnej aplikacji webowej sekwencja stanów aplikacji to sekwencja stron
- W aplikacji SPA/SPI każda częściowa zmiana strony w przeglądarce (Ajax, DOM) skutkuje zmianą stanu
- Stany aplikacji SPA/SPI można podzielić na:
  - Stany fundamentalne (odpowiadałyby stronom w modelu tradycyjnym, warte tworzenia zakładek)
  - Stany drugorzędne
- Przykład kategoryzacji stanów: obsługa logowania
  - Ekran do wprowadzenia użytkownika i hasła (fundamentalny)
  - Informacje o błędach w formularzu logowania (drugorzędny)
  - Ekran powitalny po zalogowaniu (fundamentalny)

Zakładki i historia przeglądarki w SPA

- Opis problemu

– Dla aplikacji SPA samoistnie pojawi się tylko jeden wpis w historii przeglądarki

– Ewentualna zakładka będzie prowadzić do strony w wersji pobranej z serwera (stan początkowy)

– Programowa podmiana (JS) window.location (location.href) może spowodować odwołanie przeglądarki do serwera

- Rozwiązania

– Wykorzystanie adresów URL adresujących fragmenty stron (#)

Podmiana adresu URL (przez location.href lub location.hash) na różniący się od poprzedniego treścią po znaku # nie powoduje odwołania do serwera

– Wykorzystanie HTML 5 History API: history.pushState

[W obu przypadkach obsługa nawigacji Back/Forward jest programowa]

Rozwiązania pozostałych problemów modelu SPA/SPI

- Search Engine Optimization (SEO)

– Specjalny tryb nawigacji dla robotów (web crawlers)

• … (roboty obecnie ignorują JavaScript)

• Linki w formacie (Ajax-crawlable): „#!” (wyszukiwarka widząc taki URL zmienia go na zawierający „?_escaped_fragment_” a aplikacja odpowiada snaphotem strony HTML)

- Zwiększanie licznika odwiedzin stron

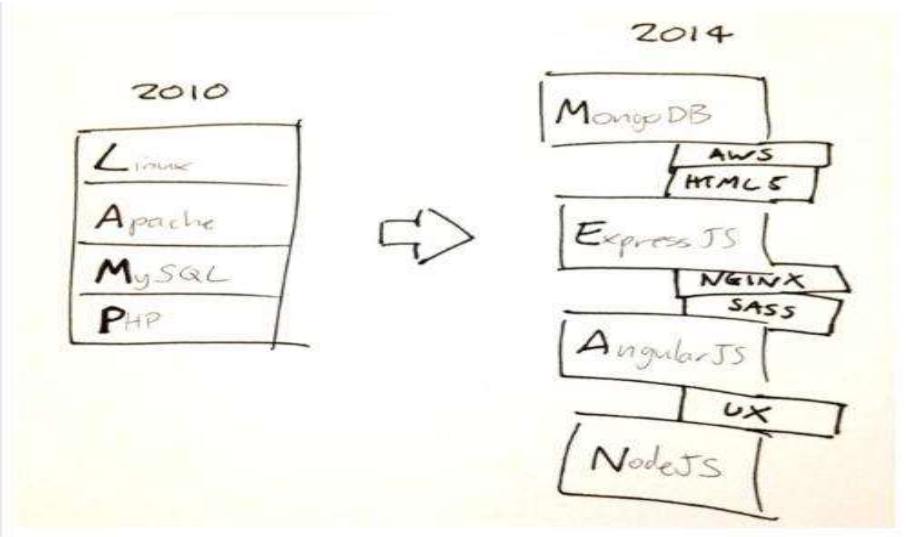
– Ramka IFRAME prowadząca do pustej strony ze skryptem

- Okienka pop-up

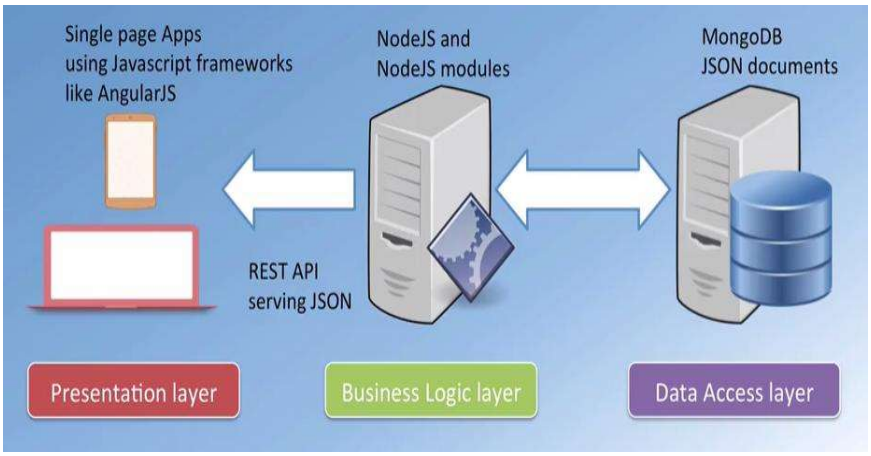
– Problemy: brak okien modalnych w przeglądarce, współdzielenie stanu między oknami

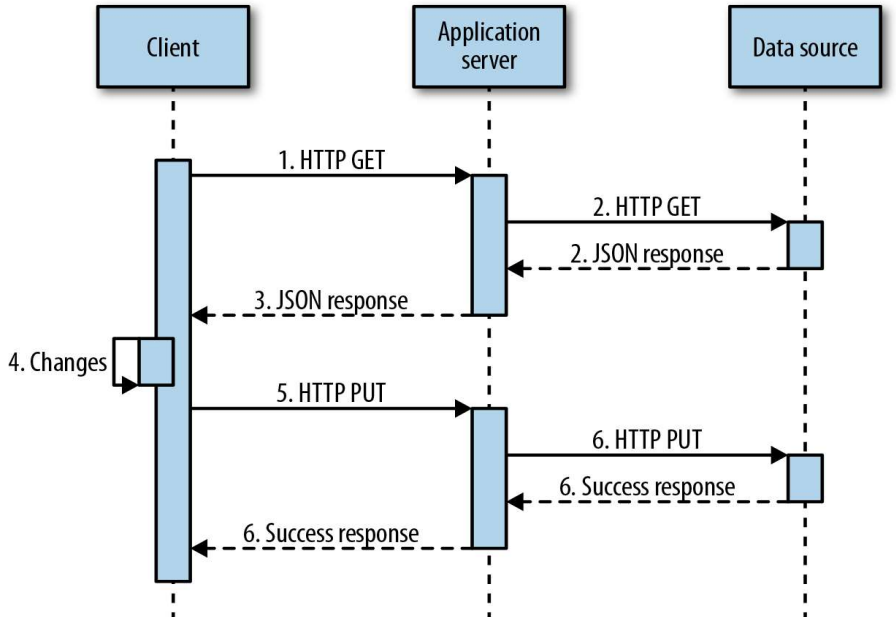
– Rozwiązanie: Symulowanie okienek modalnych/niemodalnych za pomocą elementów HTML z odpowiednim pozycjonowaniem

Stos technologiczny – kiedyś i dziś



Architektura aplikacji Webowych
nowoczesny stos technologiczny



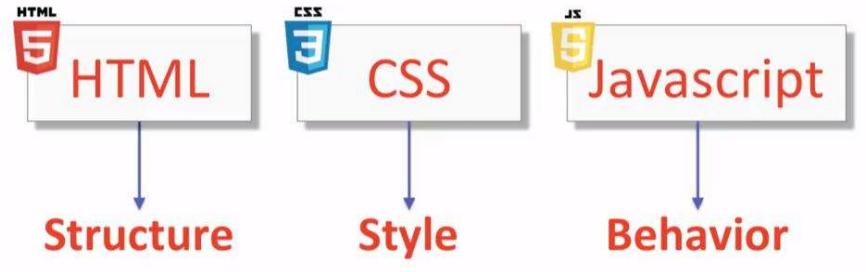


TECHNOLOGIE FRONT-END

BASIC FRONT-END –MUST HAVE

Podstawowe umiejętności:

- **HTML 5**;
- **Kaskadowe Arkusze Styli CSS**;
- **Język JavaScript**;



FRONT-END DEVELOPER

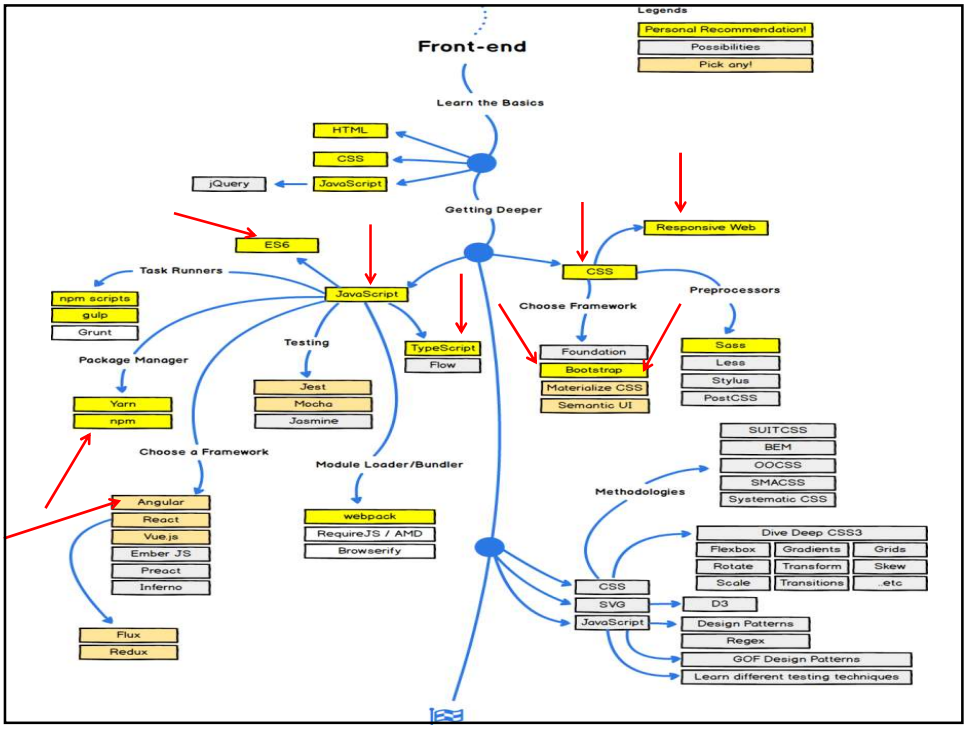
Umiejętności front-end developera:

- **Preprocesory i frameworki CSS**;
- **MV* frameworki JavaScript**;
- **Narzędzia do budowania front-end'u**;

TECHNOLOGIE FRONT-END'OWE

Obejmują:

- Język znaczników HTML
- Kaskadowe arkusze styli CSS
  - Preprocesowy CSS (Less, Saas, itp.)
  - Biblioteki (Bootstrap, Materialize, Material Design)
  - RWD (Responsive Web Design): media queries
- JavaScript
  - MV* frameworki: Angular, ReactJS, VueJS ....
  - Testy jednostkowe: Karma, Jasmine, Jest
  - Biblioteki: Vanilla JS, Babel JS
- Narzędzia
  - Npn, yard (managery pakietów)
  - Gulp, Grunt (automatyzacja)
  - WebPackage (budowa pakietów)


Front-end
Legends
Personal Recommendation!
Possibilities
Pick any!
Learn the Basics
HTML
CSS
JavaScript
jQuery
Getting Deeper
ES6
Task Runners
npm scripts
gulp
Grunt
JavaScript
Package Manager
Yarn
npm
Testing
Jest
Mocha
Jasmine
TypeScript
Flow
Choose a Framework
Angular
React
Vue.js
Ember JS
Preact
Inferno
Flux
Redux
Module Loader/Bundler
webpack
RequireJS / AMD
Browserify
CSS
Responsive Web
Choose Framework
Foundation
Bootstrap
Materialize CSS
Semantic UI
Preprocessors
Sass
Less
Stylus
PostCSS
Methodologies
SUITCSS
BEM
OOCSS
SMACSS
Systematic CSS
CSS
SVG
JavaScript
Dive Deep CSS3
Flexbox
Gradients
Grids
Rotate
Transform
Skew
Scale
Transitions
..etc
D3
Design Patterns
Regex
GOF Design Patterns
Learn different testing techniques