Fundusze Europejskie
Wiedza Edukacja Rozwój

Politechnika Wrocławska

Unia Europejska
Europejski Fundusz Społeczny

„ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej”

Politechnika Wrocławska

Wydział Informatyki i Zarządzania

Zaawansowane Technologie Webowe

Laboratorium

Temat:	Projekt grupowy – wybór backend
Opracował:	mgr inż. Piotr Jóźwiak
Data:	lipiec 2020
Liczba godzin:	2 godziny

Table of Contents

Wstęp 3
Cel laboratorium 3
Przegląd wybranych technologii backend'owych 3
Ruby on Rails 4
Laravel 5
Express.js 6
Django 7
Pyramid 8
Flask 9
Podsumowanie 10

Wstęp

O roli backendu w technologiach webowych zdążyliśmy już coś powiedzieć na poprzednich laboratoriach. Dla przypomnienia należy przytoczyć jedynie istniejący podział developerów (czy też programistów) pracujących nad aplikacjami internetowymi na kategorie:

- Front-end
- Back-end
- Full stack

Na dzisiejszym laboratorium skupimy się na wybraniu i rozpoczęciu pracy nad Back-end'em projektu grupowego. Jak już mówiliśmy back-end to część aplikacji znajdująca się na zewnętrznym serwerze, do którego użytkownik nie ma bezpośredniego dostępu (ang. Server-side). Ponieważ ta część systemu dobrze jest odizolowana od części publicznej, dlatego to tutaj odbywają się procesy zarządzania całym systemem. Wynika to oczywiście z zapewnienia bezpieczeństwa. System ten pełni rolę, można by rzec, szarej eminencji. Nie widać go na zewnątrz, ale jest on niezbędny do działania kluczowych funkcjonalności.

Oczywiście istnieją klasy serwisów internetowych, które nie posiadają back-endu. Takim przykładem mogą być proste statyczne serwisy, które pełnią jedynie rolę wizytówek czy portfolio. Nie jest wymagana w takich sytuacjach żadna skomplikowana dynamika, przez co nie ma potrzeby wprowadzania backendu.

Cel laboratorium

Celem tego laboratorium jest przyjrzenie się dostępnym technologiom back-end'owym oraz na podstawie zdobytej wiedzy podjęcie decyzji nad wybraniem najlepiej dopasowanej technologii do implementacji projektu grupowego. Aby wspomóc decyzję pokrótce scharakteryzujemy najpopularniejsze (choć jedynie wybrane spośród wielu) technologie wykorzystywane do implementacji logiki biznesowej.

Oczywiście żadna z grup nie jest zmuszona do wybrania jednej z omówionych technologii. Nic nie stoi na przeszkodzie, aby wybrać dowolną inną technologię niż przedstawione w niniejszym opracowaniu. Każdy wybór należy oczywiście umotywować.

Przegląd wybranych technologii backend'owych

Wybór odpowiedniej technologii pracującej na tyłach aplikacji webowej jest kluczową decyzją umożliwiającą osiągnięcie sukcesu danego rozwiązania. Niezależnie od tego czy jest to prosty startup, czy implementacja wielkiego systemu, bardzo ważne jest ustalenie jakiego języka użyć w projekcie. Dobrze dobrana technologia zapewni szybkość działania, skalowalność oraz łatwość modernizacji i zarządzania wdrożonym produktem.

Pytanie o wybór najlepszej technologii back-end może zostać sprowadzona do odpowiedzi na poniższe pytania:

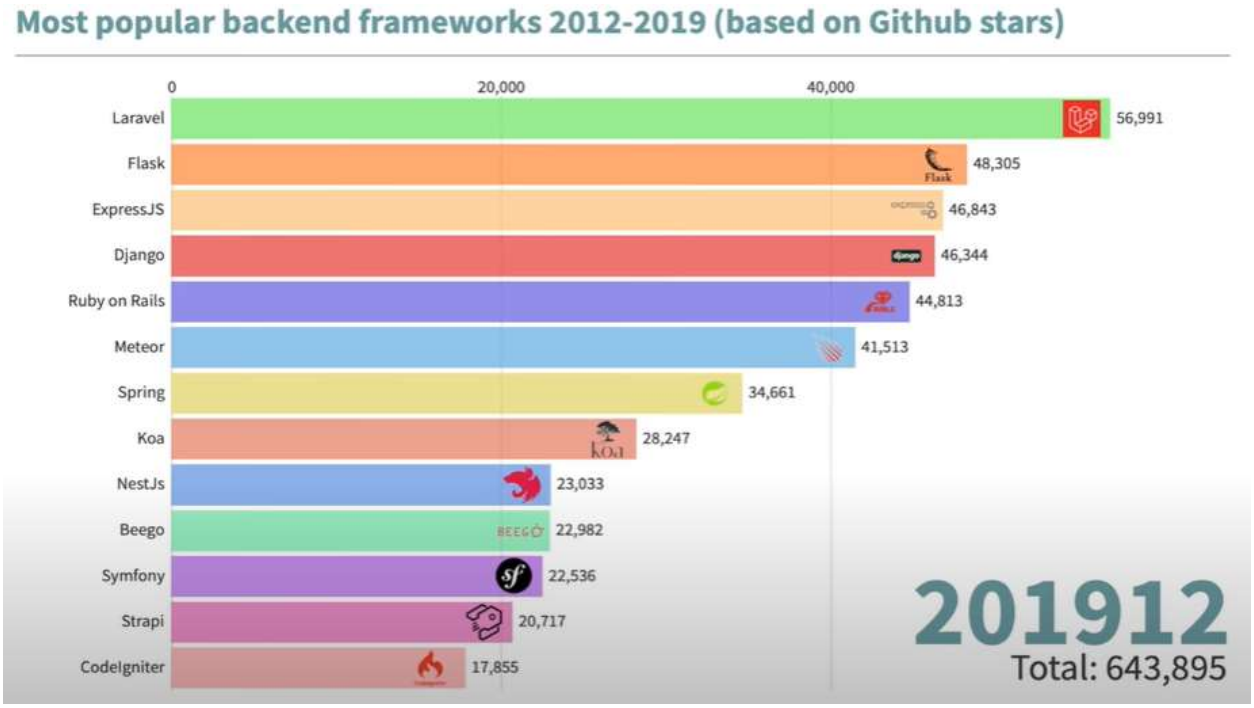
1. Jaka jest dziedzina problemu – każda technologia ma swoje własne zastosowania biznesowe, a tym samym zestaw zalet i wad.
2. Charakterystyka języka programowania – zależy od tego co chce się osiągnąć.
3. Jaka jest dostępność do ekspertów w danej technologii w zespole – jeśli zespół głównie składa się z programistów np. Java, to trudno oczekiwać, że w szybki i bezbolesny sposób będą oni pisali np. Pythonie.

Niezwykle trudno jest określić, jaka jest różnica między dwoma językami programowania. Dzieje się tak, ponieważ wszystkie języki programowania mają własną rzeszę zwolenników, którzy odmawiają wszelkiego rodzaju krytyki. Dlatego niezwykle ważne jest, aby nie tylko zrozumieć różnicę między nimi, ale także zdawać sobie sprawę z wpływu, jaki może wywrzeć zły język.

Rzućmy okiem na najpopularniejsze technologie według badania *Stack Overflow* z udziałem 100 000 programistów. Według nich w roku 2019 najpopularniejsze technologie stosowane w back-end to:

1. Ruby on Rails
2. Laravel
3. Express.js
4. Django
5. Pyramid
6. Flask

Podobny skład wyłania się z badania na podstawie Github:



Dla wielu może wydawać się ta lista dość zaskakująca. Nie ma tu np. .Net, a Java Spring wcale nie jest na samym szczycie. Dlatego przyjrzyjmy się krótkiej charakterystyce tych technologii. Przypominam, że wybór grupy projektowej nie jest ograniczony do w/w listy. Nic nie stoi na przeszkodzie by wybrać inną, nieopisaną technologię w swoim projekcie.

Ruby on Rails

Ruby on Rails jest bardzo dobrze znany od dawna z tego, że jest łatwy do czytania i pisania. W Railsach popularna jest także platforma dodatkowa dla Rubiego, która bardzo ułatwia tworzenie aplikacji internetowych.

Ruby on Rails obecnie znajduje się na szczycie listy najlepszych technologii backendowych do wykorzystania w 2019 roku.

Ponad 10% wszystkich zapytanych programistów wskazało Ruby, jako ich język pierwszego wyboru. Jego popularność szybko rośnie - ponad 46% respondentów postrzega Ruby jako swoje ulubione środowisko do pracy.

Rozwój Ruby on Rails jest bardzo szybki, co czyni go wyjątkowo dobrym dla startupów. Został stworzony w oparciu o podejście MVC (Model-View-Controller), co sprawia, że nawet najbardziej skomplikowane aplikacje można łatwo rozszerzyć o nowe funkcje lub logikę biznesową. Podobnie jak Django, istnieje kilka naprawdę zadziwiających przykładów działania Ruby on Rails, czyniąc go godnym zaufania frameworkiem.

Głównym celem Ruby on Rails jest szybkość i łatwość pisania kodu, co oszczędza czas i podnosi jakość implementacji. Bardzo poważną wadą tego rozwiązania jest powolne działanie w porównaniu z konkurencją. Także wymagania wobec serwera i problemy ze skalowaniem są często wskazywaną bolączką przez programistów.

Społeczność Ruby on Rails jest ogromna - a mówiąc o niej, mamy na myśli to, że w zasadzie w każdym dużym mieście społeczność spotyka się regularnie. Na wiele pytań, które może zadać zespół programistów, gdzieś już udzielono odpowiedzi. Istnieją różne biblioteki open source (znane również jako gems), więc ktoś wykonał już wiele pracy. Znalezienie dobrej dokumentacji dla wspomnianych gem'ów może być trudne, szczególnie dla mniej popularnych bibliotek.

Przykłady wdrożeń: Airbnb, Couchsurfing, Kickstarter, Shopify, GitHub, Bloomberg.

Przykład kodu:

```ruby
# fibonacci sequence

# input n
n = 7
def fibonacci(n)
# if n = 1, we want to output 1
  if n == 1
    1
# if n = 2, we want to output 1
  elsif n == 2
    1
# if n > 2, we want to output the sum of the previous two values
  else
    fibonacci(n-1) + fibonacci(n-2)
  end
end

# print the output of n
puts "#{n}'s fibonacci value is #{fibonacci(n)}"
```

Laravel

Język PHP znajduje się w pierwszej dziesiątce w sieci, głównie ze względu na niezwykłą popularność WordPressa i często używane frameworki. Ogromna popularność zawsze wpływa na dużą społeczność skupioną wokół PHP, dzięki czemu w sieci odnajdziemy wiele opublikowanych już rozwiązań, które pomogą nam rozwijać się szybciej przy niższych kosztach.

PHP jest dobrze znane z możliwości integracji wielu baz danych, jest bardzo dynamicznym językiem. Zapewnia wszystkie niezbędne funkcje, których programiści potrzebują do tworzenia aplikacji internetowych, pracy z bazą danych i problemów z uwierzytelnianiem.

Wśród frameworków PHP warto zajrzeć do Laravel. Laravel jest nieprzypadkowo jedną z najlepszych technologii backendowych do wykorzystania w 2019 roku.

Laravel to framework PHP MVC, który przyspiesza proces programowania dzięki eleganckiemu i prostemu wzorcowi składni, migracji bazy danych i narzędzi do tworzenia schematów. Korzystne jest również proste i bardzo bezpieczne uwierzytelnianie.

Dokumentacja Laravel jest bardzo przejrzysta, uporządkowana i zrozumiała. Posiada wbudowane wsparcie do MVC, jednak architektura systemu nie ogranicza nas do tego jednego modelu. Laravel umożliwia programistom tworzenie zarówno małych, jak i dużych aplikacji. Zapewnia także bezpieczną strukturę budowania API, która może wspierać budowanie aplikacji hybrydowych.

PHP ma swoich zwolenników i krytyków. Nie każdy zespół programistów jest chętny do pracy z PHP, ale mimo wielu krytyków, deweloperzy często z niego korzystają.

Przykłady realizacji: laracasts.com, events.arsenal.com

```
<p class="message-controls no-margin">
    <!-- <i class="sprite-project sprite-project-pin-gray pull-right float-none-xs"></i> -->

    <button class="pull-right">
        <i class="fa fa-ban pull-right"></i>
    </button>

    @if ($data->owner_id == app('user')->id)
        <button type="button" class="js-button-message-edit pull-right" data-id="{{ $data->id }}">
            <i class="fa fa-pencil"></i>
        </button>
    @endif

    <button class="pull-right">
        <i class="fa fa-thumb-tack pull-right"></i>
    </button>
</p>
```

Express.js

Patrząc na listę najpopularniejszych technologii, JavaScript nadal pozostaje jednym z najczęściej używanych języków programowania. Pozostaje niezmiennym liderem od dłuższego czasu. Prawie 70% programistów wskazało JavaScript jako framework pierwszego wyboru.

Express.js zasłużenie znajduje się na liście najlepszych technologii backendowych w 2019 roku. Prawdopodobnie dlatego także większość programistów przyjęła framework Node.js jako preferowaną technologię. Express.js zapewnia niezwykłą skalowalność i niesamowitą obsługę równoległości. Obecnie technologia ta jest wykorzystywana w usługach lub aplikacjach, które wymagają szybkiego wykonania wielu zadań z ogromną liczbą użytkowników w tym samym czasie.

Express.js jest platformą server-side dla Node.js. Jest to również jeden z najlepiej wspieranych frameworków Node.js. Posiada bardzo rozbudowaną społeczność miłośników tej technologii. Express.js

wraz z Node.js mogą być używane do tworzenia interfejsów API dla jednostronicowych, wielostronicowych, hybrydowych aplikacji mobilnych i internetowych.

Framework Express.js ma jeszcze jedną ważną zaletę. Pozwala zespołowi programistów używać tego samego języka, którym jest JavaScript, zarówno w back-end, jak i w front-end. W rezultacie proces tworzenia jest znacznie szybszy i łatwiejszy, ponieważ jedna osoba może zarządzać zarówno warstwami prezentacji, jak i dostępu do danych.

Przykłady wdrożeń: Uber, PayPal, Netflix, Linkedin

```
// Load required packages
var express = require('express');

// Create our Express application
var app = express();

// Create our Express router
var router = express.Router();

// Initial dummy route for testing
// http://localhost:3000/api
router.get('/', function(req, res) {
  res.json({ message: 'You are running dangerously low on beer!' });
});

// Register all our routes with /api
app.use('/api', router);

// Start the server
app.listen(3000);
```

Django

Z wyników ankiety Stack Overflow wynika, że Python staje się najszybciej rozwijającym się głównym językiem programowania. Okazuje się, że Python jest także jednym z najbardziej poszukiwanych języków wśród programistów. Dlatego też programiści coraz częściej wskazują na Python'a jako na język, który chcieliby się nauczyć – 25,1% deweloperów.

Django jest obecnie najpopularniejszym frameworkiem sieciowym Pythona, który zapewnia szybki rozwój i bardzo czysty, pragmatyczny projekt. Obecnie Python działa na dowolnej platformie oraz jest open source – dostarczając łatwo skalowalne rozwiązania. Ponieważ wykorzystuje zestaw komponentów, zapewnia standardowy sposób tworzenia witryn internetowych w bardzo łatwy i szybki sposób. Głównym celem Django jest ułatwienie tworzenia złożonych witryn internetowych opartych na bazie danych.

Django jest obecnie jednym z najlepszych rozwiązań do tworzenia minimalnie opłacalnego produktu, na którym można dalej budować, ponieważ staje się w pełni funkcjonalny. Oznacza to, że zawiera już wszystkie niezbędne narzędzia do tworzenia praktycznie każdej dodatkowej funkcji dla tego produktu. Django realizuje wzorzec MTV (model-template-view).

Ponadto Django oferuje jeden z najwyższych poziomów bezpieczeństwa. Jednocześnie może łączyć się z wieloma istniejącymi aplikacjami innych firm, takimi jak Twitter, Facebook czy aplikacje płatnicze.

Bardzo istotną zaletą tego frameworka jest funkcjonalność automatycznego generowania panelu administratora, z możliwością dalszego dostosowywania. Pozwala to zaoszczędzić wiele czasu związanego z implementacją tej części systemu.

Właściwa dokumentacja jest bardzo ważna dla zaoszczędzenia czasu programisty, co przekłada się na oszczędność pieniędzy klientów. Skalowalność Django jest również jedną z najważniejszych części rozwoju i wzrostu projektu. Posiada także jedną z największych społeczności wraz z biblioteką open source, która zapewnia programistom wspaniałe narzędzie pomagające im w dostarczaniu funkcjonalności w krótszym czasie.

Przykłady realizacji: Instagram, Disqus, NASA.

```
8   {% if message_list %}
9   <table class="message_list">
10      <thead>
11          <tr>
12              <th>Date</th>
13              <th>Time</th>
14              <th>Message</th>
15          </tr>
16      </thead>
17      <tbody>
18          {% for message in message_list %}
19          <tr>
20              <td>{{ message.log_date | date:'d M Y' }}</td>
21              <td>{{ message.log_date | date:'H:i:s' }}</td>
22              <td>
```

Pyramid

Pyramid bardzo ułatwia pisanie aplikacji internetowych. Wraz z rozwojem aplikacji Pyramid będzie oferować wiele funkcji, dzięki którym pisanie złożonego oprogramowania będzie wymagało mniej wysiłku. Ponadto Pyramid działa we wszystkich obsługiwanych wersjach Pythona.

Pyramid jest najczęstszą alternatywą dla Django. Niektórzy programiści twierdzą, że jest to framework do pisania frameworków, natomiast inni nazywają go Flask'iem na sterydach.

Pyramid zawiera wbudowane podstawowe mechanizmy routingu adresów URL wraz z widokami, przydatne funkcjonalności autoryzacyjne oraz całkiem spory zestaw wtyczek i aplikacji firm trzecich, które mogą zapewnić np. szkielet będący odpowiednikiem panelu administracyjnego, jaki otrzymujemy w Django. Jego zestaw narzędzi jest nieco większy niż ten dostarczony w Flask. Jednak w tym środowisku nie ma wbudowanego wsparcia dla ORM, szablonów czy też usług formularzy.

Pyramid nadaje się zarówno do małych aplikacji, jak i dużych serwisów. Jedną z najczęstszych skarg na Pyramid jest to, że oferuje tak wiele opcji, że rozpoczęcie nowego projektu może być trudne ze względu na problem z podjęciem decyzji z czego będzie się korzystać.

Przykład wdrożenia: Dropbox, SurveyMonkey, Reddit.

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

Flask

Flask jest prawie bezpośrednim przeciwieństwem Django. Jest to microframework, który zawiera tylko minimalne narzędzia i wymaga ręcznej konfiguracji wszystkiego innego. Oznacza to, że programiści mają pełną elastyczność w doborze narzędzi.

Bez problemu można użyć innego ORM niż popularny SQLAlchemy, (przy okazji, polecam przetestowanie PonyORM, chociaż nie jest jeszcze gotowy do produkcji, ponieważ nie obsługuje migracji). „Chcesz innego języka szablonów niż Jinja2, bądź moim gościem!” Dzięki takiemu podejściu Flask świetnie sprawdza się wszędzie tam, gdzie programiści chcą mieć pełną kontrolę nad każdym elementem swojej aplikacji.

Flask najlepiej sprawdza się w projektach, w których architektura jest oparta na mikro serwisach. Z drugiej strony, uruchomienie projektu za pomocą Flaska może być trudniejsze ze względu na czas i wysiłek wymagany do napisania i skonfigurowania wszystkiego ręcznie - podczas gdy wszystkie te elementy są po prostu dostarczane w Django. Wybierając Flask, w zasadzie zamieniamy wygodę na większą elastyczność.

Przykład wdrożenia: Pinterest, LinkedIn.

```python
from flask import Flask
app = Flask(__name__)


@app.route('/')
def hello():
    return "Hello World!"


@app.route('/<name>')
def hello_name(name):
    return "Hello {}!".format(name)

if __name__ == '__main__':
    app.run()
```

Podsumowanie

Przedstawiona powyżej charakterystyka opisuje jedynie wybrane dostępne technologie. Zanim podejmiemy decyzję o dobraniu odpowiedniego framworku należy przyjrzeć się jeszcze innym ofertom, chociażby takim jak ASP.Net Core czy Java Spring.

Na koniec tego laboratorium należy przygotować krótkie sprawozdanie przedstawiające wybraną technologię do pisania Back-end'u. Jednocześnie należy rozpocząć prace implementacyjne nad projektem. Należy także zapewnić odpowiedni podział zadań pomiędzy członków zespołu. Mile widziane będzie wykorzystanie wsparcia z np. tablic kanbanowych dostępnych w środowiskach DevOps opisanych na poprzednim laboratorium. Postęp prac implementacyjnych będzie prezentowany prowadzącemu na kolejnych laboratoriach.