POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

1. KIERUNEK: Teleinformatyka

SPECJALNOŚĆ: Projektowanie Sieci Teleinformatycznych

PRACA DYPLOMOWA

INŻYNIERSKA

Tytuł pracy w języku polskim

Projekt i implementacja aplikacji webowej i mobilnej, katalogującej informacje o daniach w restauracjach

Tytuł pracy w języku angielskim  
*Design and implementation of a web and mobile aplication, storing data about dishes in restaurants*

AUTOR:

Jakub Cembrzyński

AUTOR:

Imię i nazwisko

PROWADZĄCY PRACĘ:

1. dr. inż. Wojciech Kmiecik, K2

OCENA PRACY:

WROCŁAW …….

# Wstęp

Rozwój informatyki jest napędzany chęcią konsumentów do ułatwiania życia, co leży w ludzkiej naturze. Każdy z nas nie lubi lub nie może sobie pozwolić na stratę czasu w życiu codziennym. Przez co coraz chętniej korzystamy z restauracji, aby zaoszczędzić czas na gotowaniu. Dobrze świadczy o tym wzrost popularności takich aplikacji jak pyszne.pl, czy pizzaportal. Aplikacje te pozwalają na szybsze oraz sprawniejsze zamawianie jedzenia na dowóz. Co jednak jeżeli chcielibyśmy zjeść coś w na miejscu, w restauracji? Google maps pozwala nam wyszukać restaurację w pobliżu wraz z ich oceną, na podstawie zadowolenia konsumentów. Jednakże aplikacja nie pozwala nam dokładnie sprawdzić menu danej restauracji. Można sobie wyobrazić sytuację, że włoska restauracja nie serwuje naszego ulubionego makaronu. Wtedy tracimy czas na dojście do tego miejsca lub w lepszej sytuacji tracimy czas na szukanie strony danego miejsca, tylko po to, żeby się dowiedzieć, że należy szukać dalej. Załóżmy jednak, że restauracja serwuje nasz ulubiony makaron. Sprawdzamy ocenę i jest ona dla nas na tyle wysoka, że decydujemy się odwiedzić dane miejsce. Jednak po zjedzeniu, okazuje się, że danie do najlepszych nie należało. Przyczyną takiej sytuacji może być fakt, że ocena restauracji nie zawsze może oznaczać, że każda pozycja w karcie jest takiej samej jakości. Można sobie wyobrazić sytuację, gdzie tylko 3 najpopularniejsze dania z karty są świetne, a reszta przeciętna. Większość klientów oceni bardzo dobrze dane miejsce, gdzie tak naprawdę ich ocena dotyczy jedynie części z dań jakie dana restauracja oferuję. To tak jakby idąc na film do kina, klient podejmowałby decyzję o wyborze jedynie na podstawie oceny miejsca, na którą składa się znacznie więcej rzeczy, niż pozycje w repertuarze.

Motywując się tym celem mojej pracy jest projekt i implementacja aplikacji internetowej i mobilnej, która katalogując dane o daniach w restauracjach, pozwoli rozwiązać te problemy. Będzie ona pozwalała użytkownikowi znaleźć restaurację, gdzie zje danie, które go interesuje. Dodatkowo będzie mógł zobaczyć jak inni klienci danej restauracji ocenili konkretne danie, dzięki czemu będzie mógł podjąć świadomą decyzję, co przełoży się na zaoszczędzenie czasu i zmniejszy frustrację spowodowaną rozczarowanie.

Moja praca będzie składać się z sześciu rozdziałów. W pierwszym rozdziale opisałem motywację do stworzenia pracy. W drugim rozdziale porównałem dostępne technologie pozwalające na implementacje tego typu aplikacji. Trzeci rozdział jest poświęcony analizie potrzeb użytkowników, co pozwoliło stworzyć listę wymagań funkcjonalnych oraz niefunkcjonalnych. Rozdział 4 zawiera projekt aplikacji. Budowę modułów, klas, komponentów, tabel w bazie danych oraz relacje między nimi, a także wykorzystane algorytmy. Rozdział 5 zawiera podsumowanie, które kończy pracę. Rozdział 6 zawiera bibliografię.

# Przegląd i analiza istniejących rozwiązań

Opierając się na podstawowych wymaganiach i trendach w projektowaniu aplikacji webowych powinna ona się składać z trzech modułów, każdy odpowiedzialny za inne funkcjonalności. Pierwszym z nich jest moduł odpowiedzialny za kontakt z użytkownikiem (ang. Front-end application). Jest to aplikacja, której zadaniem jest pobieranie oraz udostępnianie danych użytkownikowi. Drugim modułem jest baza danych, której zadaniem jest przechowywanie danych, z których korzysta cała aplikacja. Trzecim modułem jest aplikacja, której zadaniem jest pośredniczenie pomiędzy bazą danych, a interfejsem użytkownika (ang. Back-end application). Po analizie aktualnie wykorzystywanych rozwiązań na podstawie danych udostępnionych przez portal stackoverflow [1], który skupia programistów z całego świata, można wyróżnić następujące rozwiązania przeznaczone do tworzenia aplikacji webowych:

## Rozwiązania Front-endowe

Z listy wybrałem 3 frameworki do języka javascript, które pozwalają na budowanie interfejsów użytkownika. Są to rozwiązania wspierające model SPA (ang. Single Page Application). Jest to model, który opiera się na tym, że po wejściu na główny link pobiera się cała aplikacja z wszystkimi widokami. Skrypty umożliwiają renderowanie odpowiednich widoków oraz pobieranie danych dokładnie wtedy, kiedy one są potrzebne. Takie rozwiązanie zapewnia nieporównywalną płynność w korzystaniu z aplikacji do architektury MPA (ang. Multi Page Appliaction), gdzie każdy widok pobierany jest oddzielnie, powodując ładowanie pomiędzy nimi. Minusem SPA w porównaniu do MPA jest to, że w momencie, gdy aplikacja urośnie do dużych rozmiarów wejście na stronę po raz pierwszy będzie zabierać więcej czasu, niż gdyby zostało zastosowane podejście MPA.

Z oczywistych powodów do działania SPA wymagane jest wsparcie javascript w przeglądarce.

W modelu SPA jest widoczny podział pomiędzy front-endem, a back-endem, a w MPA oba moduły są mocno ze sobą związane. [2]

Analizując powyższe plusy i minusy obu modeli budowania aplikacji front-endowych, zdecydowałem się na SPA. Podejście zapewnia lepsze doświadczenie użytkownika, a wymóg działania Javascript w przeglądarce spełnia zdecydowana większość aktualnie wykorzystywanych przeglądarek. Rozdzielenie front-endu i back-endu pozwala na wykorzystywanie API przez inne aplikacji.

Wybrane rozwiązanie prezentują się następująco:

React.js

React jest aktualnie najczęściej i najchętniej wykorzystywanym rozwiązaniem aplikacji front-endowych. Jest utrzymywany przez Facebook’a i ma bardzo dużą społeczność, dzięki czemu można w łatwy sposób znaleźć rozwiązania problemów. Dzięki technologii Virtual-DOM komponenty są szybko renderowane, a możliwość importowania dodatkowych pakietów zapewnia skalowalność aplikacji. [3]

AngularJS

Jest również bardzo popularne rozwiązanie, które daje najwięcej możliwości z omawianej trójki, dzięki wsparciu dla modelu MVC (ang. Model-View-Controller). Framework jest wspierane prze Google oraz również ma dużą społeczność. Jednak AngularJS jest bardzo złożonym frameworkiem, który wymaga dużo wiedzy od programisty, aby osiągnąć zamierzone funkcjonalności. AngularJS bardzo dobrze sprawdza się w dużych i złożonych projektach. [3]

Vue.js

Jest to najnowsze z wymienionych rozwiązań, dzięki czemu można było rozwiązać, niektóre z problemów tworzenia aplikacji w dwóch pozostałych frameworkach. Efektem tego jest bardzo prosta budowa aplikacji, która nie sprawia problemów początkującym programistą. Dodatkowo wykorzystano koncepty, wykorzystywane w React’cie i Angularze, dzięki czemu można w łatwy sposób zmienić technologię na Vue.js. [3]

W mojej pracy zdecydowałem się na Vue.js. Jest to nowe rozwiązanie, które w szybkim tempie zyskuje uznanie, przez co można założyć, że jest rozwiązaniem przyszłościowym. Dodatkowym czynnikiem wpływającym na moją decyzję, że jest implementacja w aplikacji w Vue jest najprostsza z wymienionej trójki. Skutkuje to dużą szybkością w tworzeniu aplikacji.

## Rozwiązania Back-endowe

Najpopularniejszym typem aplikacji webowej pośredniczącej między interfesjem użytkownika, a bazą danych (ang. WebAPI) jest implementacja oparta o zasady architektoniczne REST (ang. Representational state transfer). REST jest zbiorem dobrych praktyk architektonicznych w budowaniu serwisów internetowych. Wyróżnia się ich 6:

1. Architektura klient – serwer
2. Bezstanowość (ang. Statelessness)
3. Cacheability
4. Odseparowanie warstw (ang. Layered system)
5. Kod na żądanie (ang. Code on demand)
6. Jednolite interfejsy (ang. Uniform interfaces)

Interfejs webowy oparty o REST jest prosty w implementacji, ale także w obsłudze. Dzieje się tak, ponieważ wszystkie operacje możliwe do wykonania na API oparte są o podstawowe metody dostępne w protokole HTTP:

1. GET – służąca do pobierania danych
2. POST – służąca do umieszczania danych określonego typu
3. PUT – służąca do nadpisywania danych
4. DELETE – służąca do usuwania danych

Dodatkową zaletą oparcia operacji na protokole HTTP jest obsługa błędów. Protokół dostarcza kody, z których można dowolnie korzystać. [4]

Decydując się na oparciu back-endu omawianego serwisu na architekturze REST, wybrane zostały frameworki, które pozwalają na stworzenie takiej aplikacji.

Express.js

Jest to framework webowy oparty o środowisko uruchomieniowe Node.js należące do MIT. Jak nazwa wskazuję kod tworzy się w tym przypadku w języku skryptowym JavaScript. Framework jest prostym rozwiązaniem do budowania interfejsów webowych. Rozwiązanie ma otwarte źródło oraz dużą społeczność, która zapewnia wsparcie w problemach, które można napotkać.[5] Minusem Express.js jest jego słaba wydajność. Aktualne trendy pokazują, że podczas tworzenia API dąży się do tego, aby było ono jak najmniejsze i jak najszybsze. Express jest oparty na wielu modułach Node.js, które zabierają zasoby na serwerze. [6]

Django

Django jest frameworkiem do tworzena interfejsów webowych w języku skryptowym python. Rozwiązanie ma otwarte źródło i jest w pełni darmowe. Ideą stojącą za Django jest zmniejszenie czasu implementacji aplikacji poprzez udostępnianie wielu gotowych funkcji. Dzięki temu programista może skupić się na wymaganiach funkcjonalnych i implementować je szybko i prosto, co z kolei poprawia czytelność kodu. Twórcy frameworka zadbali również o funkcję służące do zabezpieczania aplikacji, tak aby można było to osiągnąć w prosty sposób. Twórcy twierdzą, że aplikacje napisane za pomocą Django dobrze się skalują, a także sam framework daje możliwość implementacji aplikacji o szerokim zakresie funkcjonalności. [7]

Spring

Jest to framework, który zapewnia model programistyczny do tworzenia aplikacji webowych opartych na języku Java. Aplikacje napisane za pomocą Springa korzystają z JVM, przez co uruchamiają się na większości środowisk. Główne technologie zaimplementowane w framework (ang. Core technologies) pomagają w tworzeniu aplikacji. Jest to między innymi wsparcie dla wstrzykiwania zależności (ang. Dependecy injection), obsługi wydarzeń (ang. Event handling), walidacja (ang. Validation) oraz konwersja typów (ang. Type conversion). Udostępnione jest również wsparcie dla testów oraz zaimplementowane są takie wzorce projektowe jak model: model – widok – kontroler (ang. MVC – Spring MVC). [8]

ASP.NET Core

Jest to framework do tworzenia aplikacji webowych stworzony przez firmę Microsoft. Do działania wykorzystuję platformę .NET i uruchamia się na wielu platformach, w tym na Windows, Linux, macOS oraz Docker. Według „Tech Power Web Framework Benchmarks” ASP.NET Core wygrywa pod kątem szybkości obsługi zapytań z innymi wiodącymi frameworkami [9]. Rozwiązanie wspiera między innymi wstrzykiwanie zależności, obsługę wydarzeń, asynchroniczność, a także wzorce projektowe takie jak model – widok – kontroler (ang. MVC). Dostępna jest również dokumentacja, tworzona przez Microsoft, a także, dzięki dużej społeczności, rozwiązania popularnych problemów na forach internetowych. [10]

W mojej pracy zdecydowałem się na ASP.NET Core. Framework jest darmowy i ma otwarte źródło, co pozwoli zmniejszyć koszty aplikacji. Czynnikiem przesądzającym za tym wyborem jest duża wydajność i wsparcie dla platformy Docker, które znacznie ułatwia wdrożenie aplikacji na środowisko produkcyjne.

## Rozwiązania bazo-danowe

W tym przypadku należy zacząć od podjęcia decyzji, czy baza danych ma być relacyjne, czy nie. Bazy NoSQL’owe odróżnia przede wszystkim to, że nie korzystają one z języka SQL, a także, że nie trzeba definiować schematu danych, tak jak w bazach relacyjnych. Kolejną różnicą jest to, że bazy SQL skalują się wertykalnie poprzez zwiększanie wydajności, czy pojemności serwera, który udostępnia bazę. Bazy NoSQL’owe skalują się horyzontalnie poprzez dodawanie kolejnych serwerów (ang. Sharding), przez co są one rozproszone. Wśród najpopularniejszych rozwiązań znajdują się oba typy baz danych. [11]

MySQL

MySQL jest multiplatformowym, otwartym systemem opartym o język SQL. Posiada dużą społeczność, dzięki temu, że jest długo na rynku. Jest to darmowe rozwiązanie, które pozwoli ograniczyć koszty całej aplikacji. Dodatkowo MySQL, w przeciwieństwie do innych baz danych wspiera horyzontalne skalowanie, które pozwoli rozproszyć bazę danych. [11]

PostgreSQL

Jest to hybrydowe rozwiązanie pomiędzy typową relacyjną, a nie relacyjną bazą danych. Tak samo jak MySQL jest otwartym i darmowym systemem. Idealnie nadaje się do sytuacji, gdzie przechowywane dane nie zawsze będą tego samego formatu. Jest kompatybilne z wieloma platformami oraz z wieloma językami programowania. [11]

Microsoft SQL server

Dużym plusem rozwiązania jest fakt, że należy ono do Microsoftu. Korporacja zapewnia świetny support, dokumentację oraz naprawę błędów. MSSQL korzysta z T-SQL, który różni się nieco od „czystego” SQL. Rozwiązanie wspiera tylko platformy Windows oraz Linux. [11]

SQLLite

SQLLite jest bardzo ciekawym, multiplatformowym rozwiązaniem implementacji relacyjnej bazy danych opartej na języku SQL. Czynnikiem odróżniających SQLLite jest fakt, że cała baza danych jest przechowywana w jednym pliku, który znajduje się na tej samej maszynie, na której uruchomiona jest aplikacja. Plusem tego rozwiązania jest to, że wymaga ono znacznie mniej zasobów od innych rozwiązań. Jest to jednak okupione minusami. SQLLite dla dużych serwisów jest mniej wydajnym rozwiązaniem niż bardziej konwencjonalne aplikacje. Fakt umiejscowienia bazy danych przy aplikacji może być zarówno plusem, jak i minusem. Dla architektury składających się z jednego modułu jest to idealne rozwiązanie, które pozwoli zmniejszyć wymagane zasoby. Przykładem dobrego wykorzystania SQLLite są urządzenia Internetu rzeczy, które muszą przechowywać w swojej pamięci jakieś dane. [12]

MongoDB

MongoDB jest przykładem NoSQL’owej bazy danych opartej na dokumentach, o formie podobnej do plików formatu JSON. Udostępnione są biblioteki do najpopularniejszych języków programowania, które pozwalają w bardzo łatwy sposób mapować obiekty w kodzie na dokumenty MongoDB. Fakt, że jest to baza NoSQL’owa sprawia, że w sytuacji, gdy przechowywane są duże ilości tego samego typu danych rozwiązanie sprawdza się bardzo dobrze. Przykładem może być serwis internetowy z recenzjami filmów. Będzie on przechowywał głównie dokumenty zawierające recenzje wraz z komentarzami. Brak oparcia bazy o język SQL uniemożliwia tworzenie relacji między encjami. Możliwe jest łączenie dokumentów za pomocą odpowiednika polecenia SQL „join” jednak, nie zapewnia ono takiej wydajności jak w przypadku relacyjnych rozwiązań. [11]

Analizując wymienione rozwiązania zdecydowałem się na MySQL. Jest darmową bazą danych, która działa na zasadzie klient – serwer. Oparta jest na języku SQL i pozwala tworzyć relację, z których skorzystam w pracy. Dodatkowo zapewni wymaganą wydajność i możliwość dowolnego skalowania, jeżeli aplikacja rozrośnie się do dużych rozmiarów.

# Analiza potrzeb użytkowników

# Projekt aplikacji

# Podsumowanie

# Bibliografia

1. <https://insights.stackoverflow.com/survey/2019#technology> dostęp: 27.10.2019
2. https://medium.com/@jainshilpa1993/ultimate-death-match-spa-vs-mpa-82e0b79ae6b6
3. <https://medium.com/@afonsobarros/angular-react-vue-pros-and-cons-75e161311e86>
4. https://en.wikipedia.org/wiki/Representational\_state\_transfer#Architectural\_constraints
5. https://expressjs.com
6. <https://medium.com/sharenowtech/there-are-expressjs-alternatives-590d14c58c1c>
7. <https://www.djangoproject.com/start/overview/>
8. <https://spring.io/projects/spring-framework>
9. https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=plaintext
10. https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core
11. <https://www.xplenty.com/blog/the-sql-vs-nosql-difference/>
12. https://www.sqlite.org/whentouse.html