Politechnika Łódzka
Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja webowa do organizacji wydarzeń kulturalnych i dystrybucji biletów

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
Promotor: dr inż. Marcin Kacprowicz
Dyplomant: Maciej Pracucik
Nr albumu: 216869
Kierunek: Informatyka Stosowana
Specjalność: Technologie gier i symulacji komputerowych

Łódź, 2020/2021

Instytut Informatyki
90-924 Łódź, ul. Wólczańska 215, budynek B9
tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Spis treści

1 Wprowadzenie

W dobie trwającego rozwoju technologicznego, każdy aspekt życia codziennego jest usprawniany i przenoszony do internetu. Nieinaczej jest z promowaniem i sprzedażą biletów na wydarzenie kulturowe pokroju: spektakli w teatrze, koncertów, festiwali. Aplikacja ConcertsApp ma służyć dokładnie temu, ma zminimalizować czas, jaki należałoby kiedyś przeznaczyć na zdobycie biletów. Czy też na uzyskanie informacji o wydarzeniach z interesującej konsumenta dziedziny, lub odbywającę się w pobliżu.

1.1 Problematyka

??

1.2 Cel i założenia pracy

Celem niniejszej pracy dyplomowej jest stworzenie aplikacji webowej, która umożliwi w łatwy i szybki sposób na promowanie wydarzeń kulturowych i zakup biletów na nie. Zakup odbywać się będzie za pomocą płatności online, jedynie podając dane karty kredytowej lub debetowej.

Zakres pracy obejmuję zaprojektowanie i implementację aplikacji klienckiej i serwerowej, przy wykorzystaniu, najnowszych i bardzo popularnych na rynku pracy, technologii. Do strony klienckiej został wykorzystany React, z koleji do stworzenia serwera użyto Node.JS.

1.3 Struktura pracy inżynierskiej

Pierwszy rozdział ma posłużyć jako wprowadzenie do problemu podjętego w niniejszej pracy dyplomowej. Następny przedstawia aspekty takie jak: konkurencyjne rozwiązania do tworzonej aplikacji, analiza najpopularniejszych metod płatności online, opowiada o bazach danych oraz przybliża metody tworzenie aplikacji webowych. Trzeci rozdział przedstawia wykorzystane technologie do implementacji aplikacji oraz uzasadnia dlaczego akurat ona została wybrana a nie inna. Czwarty rozdział pokazuje jak przebiegał proces twórczy, czyli projektowanie i implementacja.

2 Teoria niezbędna do realizacji projektu

2.1 Analiza istniejących portali do promowania i dystrybucji biletów na wydarzenia kulturowe

Najpopularniejszymi portalami zbliżonymi do tworzonej aplikacji są zdecydowanie:

- Eventim
- GoingApp

Eventim jest znacznie starszym portalem, co można stwierdzić chociażby, po jego szacie graficznej, łatwo to zauważyć porównując do wyżej wymienionego GoingApp. W swojej ofercie ma wydarzenia pokroju: koncertów, przedstawień teatralnych, oper czy baletów.
Same bilety sprzedawane są w formie papierowej, przychodzą na maila, lub istnieje możliwość zakupu biletu mobilnego. Kupno może odbyć się zarówno poprzez strone internetową, jak również poprzez aplikację mobilną. Płatności można dokonać za pomocą przelewu tradycyjnego, karty kredytowej, szybkiego przelewu Dotpay
GoingApp charakteryzuje się bardziej nowoczesną i przejrzystą szatą graficzną w porównaniu do Eventim. Oferty obu portali są bardzo do siebie zbliżone, lecz tutaj można znaleźć takie wydarzenia, jak chociażby imprezy związane z filmem czy jedzeniem. Jednakże nie posiada on biletów na balet lub opery.
Wejściówki można, podobnie jak w Eventim nabyć poprzez ich stronę internetową lub aplikację mobilną. Sam bilet jest dostępny w formie dokumentu PDF, lub kodu QR, dostęp do niego mamy zarówno poprzez maila,którego otrzymujemy zaraz po zakupie, oraz aplikację. Użytkownik może dokonać płatności za pomocą systemów płatniczych takich jak: payU, eCard, MasterPass oraz Paymento®
Aplikacja webowa tworzona na potrzeby niniejszej pracy inżynierskiej czerpie z obu portali najlepsze cechy. Z Eventim czerpie różnorodność wydarzeń, z koleji z GoingApp przejrzystość interfejsu i bilety w formie, bardzo popularnego kodu QR. Metody płatności zostają ograniczone do kart płatniczych: debetowej i kredytowej. Podawane są numer karty, data ważności oraz kod CVV.

2.2 Zakupy internetowe

Sklepy internetowe z roku na rok rosną w siłę, przybywa ich liczba w zatrważającym tempie.

Aż 73% ankietowanych w raporcie przygotowanym przez Gemius Polska[12] deklaruje kupowanie online i ta forma zakupów cieszy się niezmiennie dobrym wizerunkiem wśród kupujących. Brak sklepu internetowego stanowi ogromną niekorzyść dla obecnych przedsiębiorców. Zakupy online charakteryzują się dużo większym wyborem produktów, łatowścią w porównywaniu ofert czy też łatowścią w znalezieniu interesujących produktów. Lecz, to nie z wyżej wymienionych powodów, sklepy w sieci cieszą sie taką popularnością. Wśród najczęściej wymienianych powodów są: dostępność przez całą dobę (aż 82% wybrało ten powód), brak konieczności wyjazdu do sklepu - 78%, nieograniczony czas wyboru - 72% oraz atrakcyjniejsze ceny niż w sklepach tradycyjnych - 71%.

Niestety taka forma zakupów ma też swoje słabe strony, najpopularniejszymi wymienianymi, napotkanymi problemami są: wysokie koszta dostawy, długi czas oczekiwania na dostawę, irytujące reklamy produktów, wcześniej poszukiwanych. Również bardzo często wymienianą przeciwnością jest uszkodzona przesyłka w transporcie, wynikać to może ze źle zapakowanej i zabezpieczonej paczkim lub z winy firmy kurierskiej i jej pracowników.

Co jednak sprawia, że klienci decydują się na wybór danego portalu na zakupy?

Najczęściej wybór sklepu pada dzięki kodom rabatowym, dopiero w dalszej kolejności padają hasła takie jak dokładne informacje o warunkach zamówienia, dostępne na stronie dane firmy czy przejrzysta i funkcjonalna strona internetowa. Wydawać by się mogło, że to na te kolejne cechy portali, powinno się w pierwszej kolejności zwracać uwagę, bo to dzięki nim w pierwszej chwili można stwierdzić czy strona jest fałszywa, czy też nie.

2.3 Analiza najpopularniejszych internetowych metod płatności online

Najpopularniejszymi metodami płatności w internecie, według raportu "E-commerce w Polsce 2020"[12] są kolejno:

- szybki przelew przez serwis płatności np. payU, przelewy24
- przelew tradycyjny
- płatność kartą płatniczą przy składaniu zamówienia
- płatności mobilne np. BLIK

Nie zostały wymienione płatności pokroju wysyłki za pobraniem, płatności w sklepie przy odbiorze, ponieważ nie są to płatności realizowane online, a tych dotyczy analiza przedstawiona

w niniejszym rozdziale.

Zdecydowanie nie powinna dziwić obecność szybkich przelewów na pierwszym miejscu tego rankingu. Aż 70% ankietowanych odpowiedziało, że choć raz korzystało z tej metody, przy robieniu zakupów przez internet. Cechują się błyskawiczynym czasem realizacji, w przeciwieństwie do tradycyjnych przelewów. Zaledwie 46% ankietowanych zdecydowało się choć raz na przelew tradycyjny. Różnica jest znaczna, jednakże nie powinna ona dziwić, ponieważ to oszczędność czasu, poniekąd sprawia, że klienci decydują się na zakupy online w pierwszej kolejności. Przelewy tradycyjne dodatkowo wydłużają czas realizacji zamówienia, ponieważ ich przetwarzanie odbywa się jedynie w dni robocze, o wyznaczonych godzinach, różnych, w zależności od banku. Na płatność kartą decyduje się 40% zapytanych, jest to dość zaskakujące, zważywszy na fakt, że jest to zdecydowanie jedna z najszybszych metod płatności. Do jej realizacji niezbędne jest jedynie numer karty, data wygaśnięcia oraz kod CVV. Powodem na dość niską popularność tej metody, mogą być dwie rzeczy, strach przed podawaniem danych karty, żeby nie zostały skradzione. Inną alternatywą czemu, tak mało klientów sklepów internetowych decyduję się na inne metody, jest brak karty podczas składania zamówienia. Pierwsza z nich wydaję się być bardziej prawdopodobna, mało kto kupując produkty przez internet ma akurat przy sobie kartę płatniczą, by odczytać z niej niezbędne liczby. Zdecydowanie łatwiej jest zalogować się do banku i wykonać przelew czy to tradycyjny czy szybki.

Dziwić może obecność płatności mobilnych na ostatnim miejscu z zaledwie 35% wykorzystania. Jest to sposób bardzo wygodny, zważywszy na fakt iż mało kto nie posiada przy sobie telefonu. Potrzebna jest jeszcze tylko aplikacja banku i można dokonywać dowolnych płatności. W przypadku BLIK-u generowany jest sześcio cyfrowy ciąg liczb, który wystarczy wpisać w odpowienie pole, zatwierdzić płatność w aplikacji i gotowe.

2.4 Bazy danych jako środek przechowywania danych

"Baza danych to zorganizowany zbiór ustrukturyzowanych informacji, czyli danych, zwykle przechowywany w systemie komputerowym w formie elektronicznej. Bazą danych steruje zwykle system zarządzania bazami danych (DBMS). Dane i system DBMS oraz powiązane z nimi aplikacje razem tworzą system bazodanowy, często nazywany w skrócie bazą danych.". [13] Innymi słowy jest to kontener na dane, w dowolnej postaci, mogą to być liczby, ciągi znaków, a nawet zdjęcia czy filmy. Dane te nie są, najczęściej, przetrzymywane lokalnie na komputerach, tylko na serwerach czy w chmurze.

Dlaczego więc korzysta się z baz danych a nie np. z arkuszy kalkulacyjnych?

Odpowiedź jest bardzo prosta, arkusze kalkulacyjne nie zostały stworzone do pracy z ogromną ilością danych, przy jednoczesnym dostępie, nawet kilkuset lub więcej użytkowników. Są wręcz idealne do pracy z mniejszą ilościa danych dla jednego lub małej grupy użytkowników, którzy nie potrzebują wielu skomplikowanych funkcji do manipulacji danymi. Bazy danych z koleji, przeznaczone są do pracy z ogromnymi ilościami informacji, umożliwiając dodatkowo jednoczesny dostęp do nich, wielu użytkownikom na raz. Co więcej praca z bazami danych, charakteryzuje się wysokim bezpieczeństwem i szybkością wykonywanych operacji. Operacje na danych, tworzenie zapytań odbywa się za pomocą logiki i języka o wysokim stopniu złożoności. Jako przykład może posłużyć, zdecydowanie najpopularniejszy z nich, czyli język zapytań SQL. Bazy danych dzielimy, między innymi na:

- relacyjne
- hurtownie danych
- NoSQL
- chmurowe

Relacyjne bazy danych zyskały ogromną popularność w latach 80[13]. Dane zorganizowane są w tabelach, składające się z wierszy i kolumn. Po dziś dzień stanowią jedne z najpopularniejszych baz danych, jak nie najpopularniejsze, dostępne na rynku. Przykładami relacyjnych baz danych są: MySQL i Microsoft SQL Server.
Hurtownie danych, inaczej centralne repozytorium danych, to typ bazy danych wykorzystywany do wykonywania zapytań i analizy. Ma on umożliwić i wspierać działania z zakresu analizy biznesowej, w szczególności analityki. Często operuje na danych historycznych, pochadzących z wielu źródeł. Jej umiejętności analityczne pozwalają przedsiębiorstwom cenne dane biznesowe, które ułatwiają podejmowanie decyzji.[14]
Baza danych NoSQL, inaczej nierelacyjna, cechuje się przechowywaniem danych w nieuporządkowany i częściowo uporządkowanych oraz manipulowanie nimi. Od relacyjnych baz różni je przede wszystkim to, że w relacyjnych mają jasną strukturę organizacji danych. W nierelacyjnych dane, pochodzące z tej samej kolekcji, mogą posiadać kompletnie różne atrybuty. Szerzej na temat baz NoSQL zostanie omówione w następnym podrodziale.
Ostatnim typem bardzo popularnym na rynku baz danych są bazy chmurowe. Bazy te charakteryzują się przede wszystkim tym, że dane przetrzymywane są na prywatnej, publicznej lub hybrydowej platformie przetwarzania danych w chmurze. Najpopularniejszymi bazami w

chmurze są Microsoft Azure SQL Database, Oracle Database, Google Cloud SQL oraz Amazon Relational Database Service.

2.5 Nierelacyjne bazy danych

"Bazy danych NoSQL są zamiennie nazywane „nierelacyjnymi" lub „nie SQL", aby podkreślić fakt, że potrafią obsłużyć ogromne ilości szybko zmieniających się, nieustrukturyzowanych danych innymi sposobami niż relacyjna (SQL) baza danych z wierszami i tabelami." [6]
Co jednak wyróżnia bazy typu NoSQL na tle relacyjnych baz danych?
Ich głównym wyróżnikiem zupełnie inny schemat przechowywania danych. W bazach nierelacyjnych dane nie są przechowywane w tabelach, mogą być dowolnie skalowane, każdy wiersz może zawierać różne kolumny (atrybuty opisujące dany obiekt), nie jest wymuszana relacja między obiektami.
Bazy typu SQL są znacząco lepsze jeśli skalowalność zachodzi wertykalnie, a horyzontalnie atrybuty są jasno określone. Nie jest wskazane aby każdy obiekt, z jednej tabeli, opisany był innymi atrybutami.
W przypadku prostych obiektów, opisanych małą ilością atrubutów, lepszym wyborem będą bazy typu NoSQL. Z koleji dla bardziej skomplikowanych encji, lepiej wykorzystać bazy relacyjne, ze względu na ich schludność i uporządkowanie, zdecydowanie łatwiej uniknąć niechcianego bałaganu i błędów.
Bazy NoSQL dzielimy, ze względu na typ na[4]:

- Klucz - wartość
- Dokumentowe
- Grafowe
- Kolumnowe

Pierwsze z nich opierają się na kolekcji słowników, w których z kluczem powiązane są wartości różnych atrybutów encji. Dodatkowo wykorzystuje się funckje haszujące do przyspieszenia odczytu. Jako przykłady mogą posłużyć: Windows Azure Table Storage oraz Amazon SimpleDB.
Bazy dokumentowe stosuje się do przechowywania dokumentów posiadających różne atrybuty oraz mają możliwość zagnieżdżania jednych dokumentów w drugie. Przykładem wymienionego typu baz jest wykorzystane przy implementacji projektu baza MongoDB.
Bazy grafowe oparte są na grafach i algorytmach grafowych. Każdy obiekt jest innym węzłem

w grafie, a relacje między nimi to krawędzie. Przykłady to: Titan, Giraph. Ostatnim typem baz nierelacyjnych są bazy kolumnowe. Oparte są na architekturze hybrydowej, wykorzystują techniki i podejści relacyjnej bazy oraz bazy klucz-wartość do przechowywania schematów danych. Przykładem takiej bazy jest Cassandra.
Podsumowując bazy nierelacyjne lepiej sprawdzają się przy małej ilości atrybutów, łatwiej uniknąć bałaganu, dlatego też to właśnie nierelacyjna baza danych została wybrana do realizacji projektu, a dokładniej MongoDB. O tym dlaczego akurat MongoDB, zostanie przedstawione w następnym rodziale 3.5.

2.6 Techniki tworzenia aplikacji

Przy tworzeniu aplikacji webowych istnieją dwa podejścia, mówiące o tym jak należy tworzyć aplikację. Są to Single-Page Application (SPA) oraz Multiple-Page Application (MPA, czyli tradycyjne strony internetowe).
SPA jest aplikacją webową lub stroną internetową, która nie przeładowuje swoich stron za pomocą serwera oraz interakcja z użytkownikiem zachodzi, za pomocą dynamicznego zmieniania aktualnie wyświetlanej strony. Kod źródłowy strony jest zapisywany tylko przy pierwszym załadowaniu, oraz dodatkowe zasoby są ładowane, tylko wtedy kiedy jest to wymagane, w zależności od zachowania użytkownika. SPA są interaktywne oraz przyjazne użytkownikowi, są bardziej responsywne niż tradycyjne strony, ponieważ ładują się tylko raz i ich komunikacja z serwerem, jest ograniczona do minimum. [2]
MPA jest klasycznym podejściem do tworzenia stron internetowych. Praktycznie każde kliknięcie, w dowolną rzecz na stronie, wysyła zapytanie do serwera o wyrenderowanie nowej strony w przeglądarce. Każda strona jest innym plikiem, nie ma możliwości jak w SPA, że renderowany jest tylko wymagany komponent. Cała strona musi być ponownie rerenderowana.
Kiedy należy używać jakiego podejścia?
Microsoft w poradniku do tworzenia aplikacji w .NET, proponuje poniższą tabelkę decyzyjną[7]:

Do stworzenia aplikacji na potrzeby pracy inżynierskiej zostało wybrane podejście SPA, gwarantuje ono szybsze, przyjemniejsze oraz bardziej responsywne zachowanie aplikacji dla użytkownika.

Factor	Traditional Web App	Single-Page Application
Required Team Familiarity with JavaScript/TypeScript	Minimal	Required
Support Browsers without Scripting	Supported	Not Supported
Minimal Client-Side Application Behavior	Well-Suited	Overkill
Rich, Complex User Interface Requirements	Limited	Well-Suited

Tabela 1: Tabela decyzyjna wyboru pomiędzy SPA, a MPA.

2.7 REST API

REST oznacza w skrócie Representational State Transfer, jest to architektura zaproponowana przez Roya Fieldinga, jako nowe podejście do projektowania usług internetowych. Architektura ta jest niezależna od wszelkich podstawowych protokołów, w tym HTTP. Jednak w najbardziej typowych implementacjach REST protokół HTTP pełni funkcję protokołu aplikacji.[8] Architektura REST oparta jest na kilku głównych zasadach, oto 6 najważniejszych[8]:

- Interfejsy API są oparte na zasobach - dowolnym obiekcie, danych lub usłudze, które są dostępne dla klienta
- Zasób ma identyfikator URI służący do unikatowej identyfikacji tego zasobu
- Interakcja z usługą odbywa się poprzez wymianę reprezentacji zasobów. Najpopularniejszym formatem wymiany danych jest JSON.
- Do wykonywania operacji na zasobach używa się standardowych zapytań HTTP, najczęściej używane operacje to GET, POST, PUT, PATCH i DELETE.
- Interfejsy API REST korzystają z bezstanowego modelu żądań. Każde zapytanie do serwera musi posiadać niezbędne informacje do zrozumienia zapytania. Stan sesji jest przetrzymywany tylko i wyłącznie po stronie klienta.
- Interfejsy API REST są sterowane za pomocą hipermedialnych linków, zawartych w reprezentacji.

Najpopularniejszymi metodami do operacji na zasobach, jak zostało wyżej wymienione są:

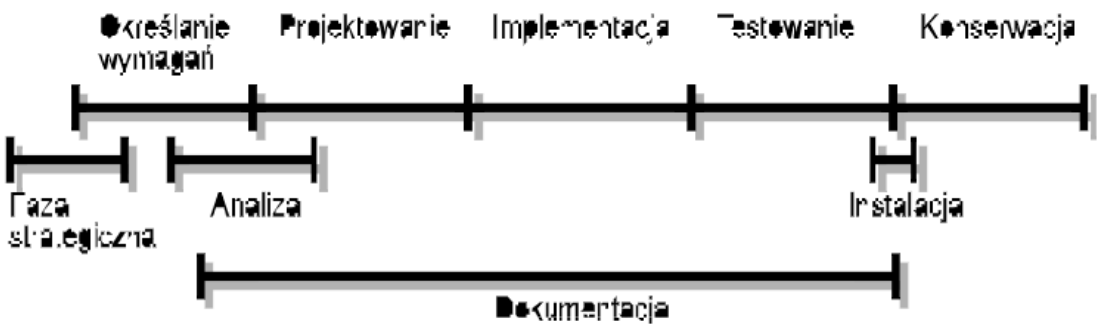
- GET - pobiera reprezentację danego zasobu np. strona potrzebuje listę wszystkich produktów dostępnych do zakupu na stronie

- POST - tworzy nowy zasób, np. użytkownik rejestruje się na danej stronie internetowej i przy kliknięciu "ZAREJESTRUJ" wysyłane jest zapytanie POST, o dodaniu nowego użytkownika do bazy danych.
- PUT - tworzy zasób lub aktualizuje istniejący
- PATCH - wykonuje częściową aktualizację zasobu, np. użytkownik zmienia adres zamieszkania, podany wcześniej
- DELETE - usuwa zasób

W zależności od podanego URI, może zostać zwrócony zasób w formie kolekcji lub pojedynczego elementu. Przykładowo podając URI `https://adventure-works.com/orders`, dla metody GET uzyskamy listę wszystkich zamówień. Z koleji podając URI `https://adventure-works.com/orders/1` uzyskamy pojedyncze zamówienie, np. o id = 1. Każda metoda będzie zachowywać się podobnie dla podanych wyżej URI, z tą różnicą, że przykładowo metoda POST zamiast pobrać listę wszystkich zamówień, utworzy nowe zamówienie, metoda PUT zaktualizuje wszystkie zamówienia, w zadany sposób.
Architektura REST API cieszy się ogromną popularnością, między innymi z tego powodu została wybrana do stworzenia aplikacji.

2.8 Projektowanie i tworzenie systemu informatycznego

W inżynierii oprogramowania, czyli inaczej wiedzy technicznej, opisującej wszystkie fazy cyklu życia oprogramowania, istnieje wiele różnych modeli cyklu życia oprogramowania. Do najpopularnieszych należą: kaskadowy, piramidy oraz spiralny. Każdy z nich w podobny sposób przedstawia każdy krok, jaki powinien przejść program, aby mógł zostać wypuszczony na rynek. Na poniższym rysunku przedstawiono jak wygląda model kaskadowy, w dalszej części zostaną omówione, po krótce, jego poszczególne etapy.



Rysunek 1: Model kaskadowy

Każda z faz cechuje się innym zestawem czynności jakie należy wykonać. W fazie strategicznej głównie zachodzi rozmowa z klientem, opracowywany jest cel przedsięwzięcia, zakres, ogólne wymagania i analiza rozwiązań. Określony również zostaje wstępny harmonogram działań nad projektem.

Faza określenia wymagań cechuje się, jak sama nazwa wskazuje, określeniem wymagań. Cele podane w fazie poprzedniej są zamieniane na faktyczne wymagania jakie musi posiadać oprogramowanie. Wymagania dzielimy odpowiednio na funkcjonalne i niefunkcjonalne. Wymagania funkcjonalne to takie, które opisują funkcję lub czynności wykonywane przez system. Z koleji niefunkcjonalne opisują ograniczenia, przy zachowaniu których system powinien realizować swoje funkcję.

W fazie analizy odpowiada się na pytanie: jak system ma działać? W odpowiedzi na to pytanie otrzymujemy model systemu, opisujący jak postawione wymagania zostaną zrealizowane, nie wchodząc w szczegóły implementacyjne.

W fazie projektowania ponownie odpowiada się na pytanie, tym razem jak system ma być zaimplementowany. W wyniku czego powstaje projekt sposobu implementacji.

Kolejno przchodzi się do fazy implementacji, jest to moment, w którym wszystkie czynności związane z projektowaniem zostają zakończone i wcielone w faktyczny program.

Po fazie implementacji następuje faza testowania. W tej fazie następuje sprawdzenie systemu, czy jest zgodny z postawionymi wymaganiami, oraz czy nie posiada jakichś błędów mogących doprowadzić do nieporządanego działania lub błędów z samym wykonywaniu się oprogramowania.

W kolejnej fazie, czyli fazie instalacji, następuje przekazanie systemu użytkownikowi, pod koniec czego staje się on właścicielem systemu. W skład tej fazy wchodzi dodatkowo: szkolenie użytkowników i administratorów, instalacja sprzętu i przeniesienie oprogramowania czy wype-

łenienie baz danych.
Na samym końcu cyklu życia oprogramowania, znajduje się część przeznaczona na konserwację programu. W tej fazie poprawiana jest jakość produktu, dostosowanie oprogramowania do zmian zachodzących w środowisku pracy oraz usuwanie wcześniej niewykrytych błędów.
Warto również wspomnieć o fazie dokumentacji, jak widać na obrazku 1, jest on wykonywany równolegle z praktycznie wszystkimi pozostałymi czynnościami. Tworzona jest dokumentacja, w której znajdują się takie elementy jak: podręcznik użytkownika, opis instalacji czy podręcznik administratora. Dokumentacja jest integralną częścią projektu i nie powinna być pomijana lub traktowana bez należytej uwagi.

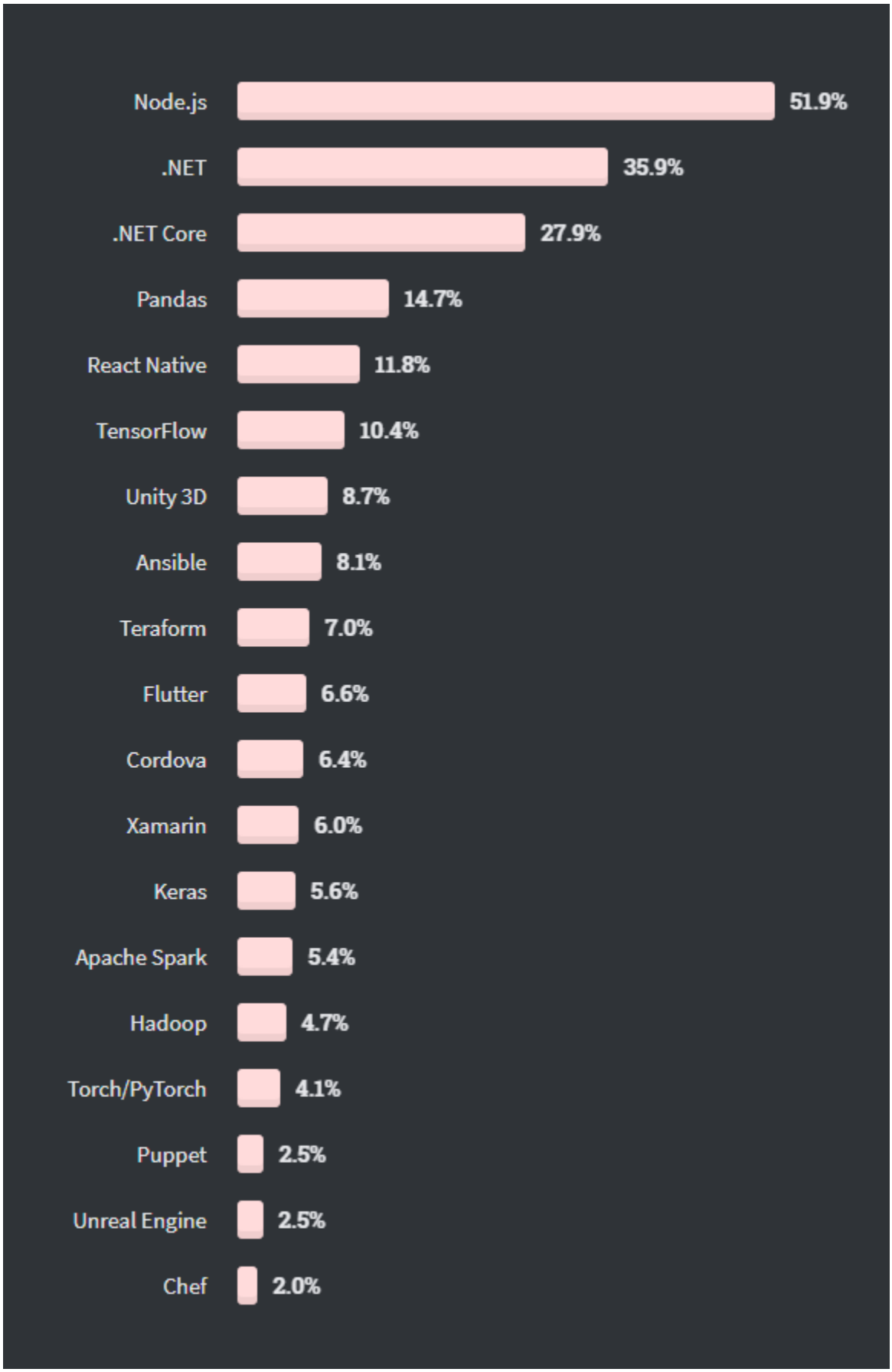
3 Narzędzia i technologie wybrane do realizacji projektu

3.1 Node.js

"Node.JS jest jest wieloplatformowym oprogramowaniem o otwartym kodzie, które pozwala deweloperom na tworzenie wszelkiego rodzaju oprogramowania w języku JavaScript pracującym po stronie serwera. Jest to środowisko uruchomieniowe, które działa poza przeglądarką, współpracujące bezpośrednio z systemem operacyjnym. W ten sposób środowisko Node udostępnia swoim aplikacjom API systemu operacyjnego, w tym dostęp do systemu plików, bibliotek systemowych czy uruchomionych procesów, w tym serwerów HTTP."[10] Wartym wspomnienia jest fakt, że jest oparty na silniku JavaScript Chrome V8.
Zalety jakie niesie ze sobą Node:

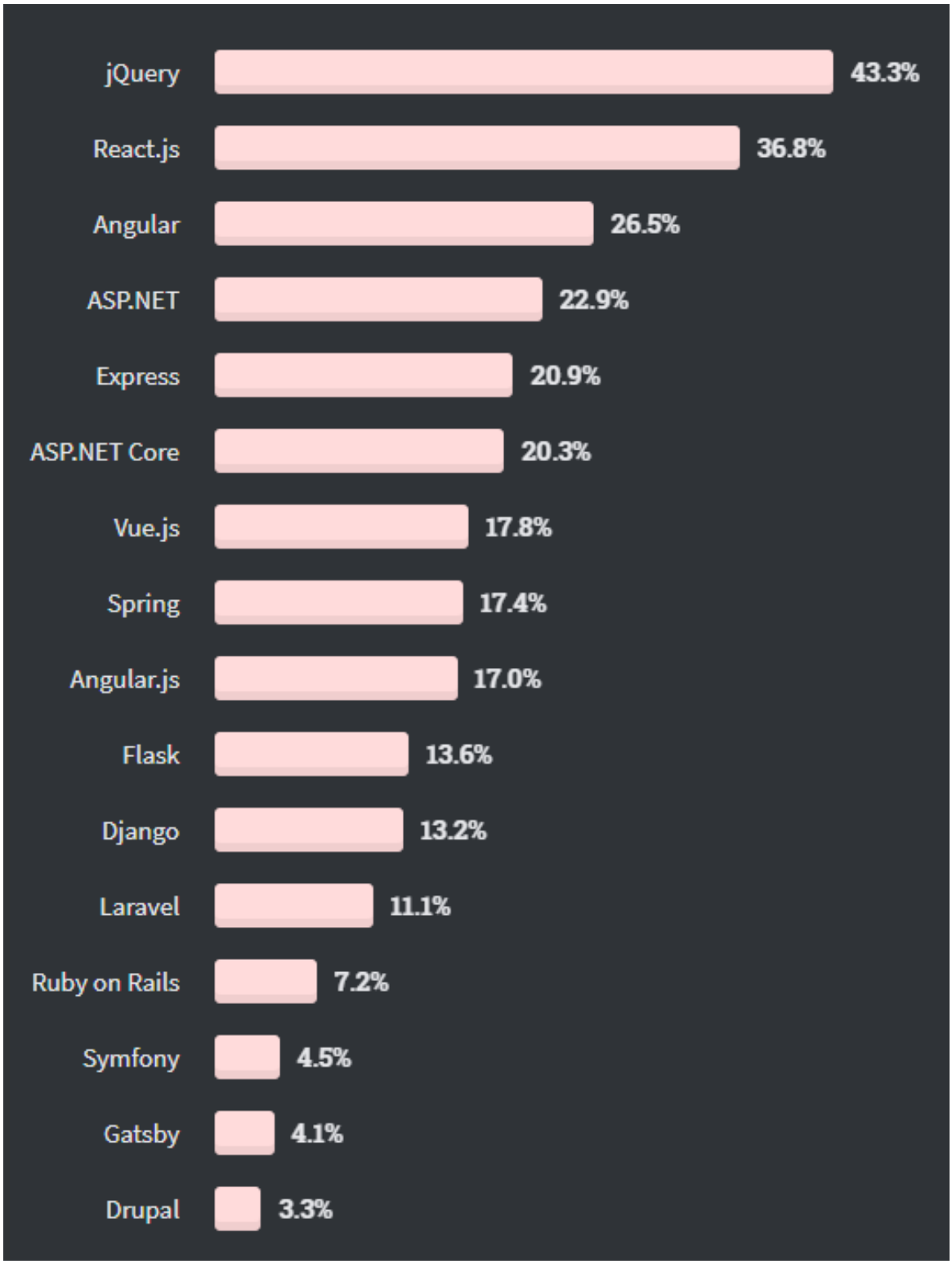
- wysoka wydajność, został zaprojektowany tak aby optymalizować wydajność i skalowalność aplikacji webowych
- pracując nad kodem po stronie klienta jak i po stronie serwera, poruszamy się w tym samym języku programowania
- dostęp do manadżera pakietów np. Yarn czy NPM, dzięki niemu uzyskuje się dostęp do setek tysięcy przeróżnych pakietów.
- jest przenośny. Można korzystać z niego zarówno na systemie macOS, Linux, jak również Windows.
- Dostęp do ogromnego community, gotowego służyć pomocą.

Node.JS jest bardzo uniwersalnym narzędziem, nie służy jedynie do tworzenia serwerów back endowych dla stron internetowych. Można za jego pomocą tworzyć zwykłe aplikacje, aplikacje z dziedziny IoT(Internet of Things), posiada nawet pakiety do uczenia maszynowego. Za pomocą Node jesteśmy w stanie wykonać przeróżne rzeczy ogranicza jedynie znajomość JavaScript oraz własna wyobraźnia.



Rysunek 2: Najpopularniejsze frameworki i biblioteki według ankiety Stack Overflow[11]

Jak zostało zaprezentowane na obrazku 2, Node.JS cieszy się ogromną popularnością. Uzyskał 51.9% na 33 913 ankietowanych, tę grupę stanowili jedynie profesjonalni developerzy.



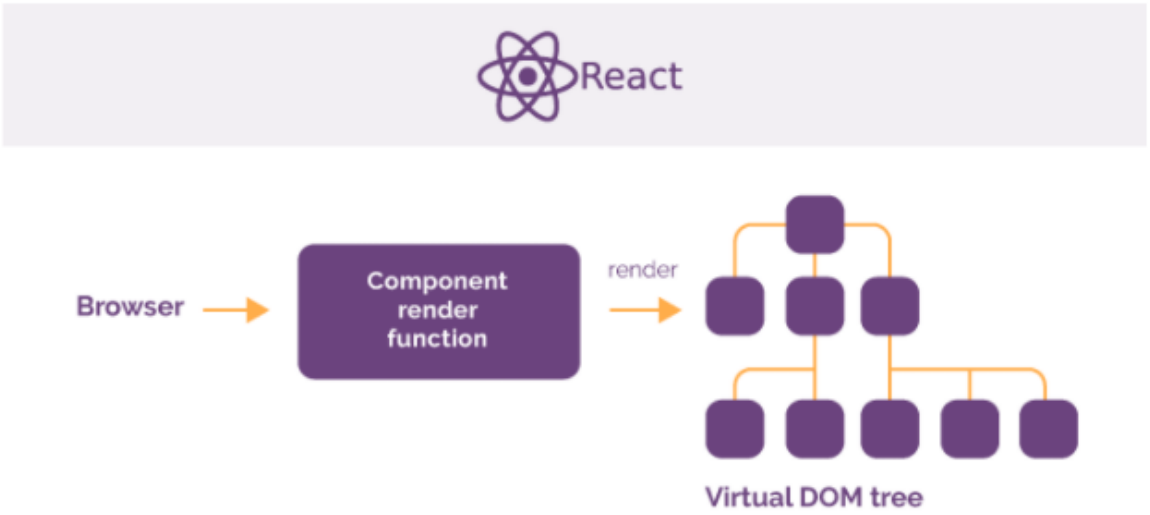
Rysunek 3: Najpopularniejsze webowe frameworki według ankiety Stack Overflow[11]

Dodatkowo Express, czyli framework do Node.JS, za pomocą, którego został stworzony cały back end projektu, znajduje się na drugim miejscu w rankingu popularności wśród frameworków

back endowych, osiągnął popularność na poziomie 20.9%. Wyprzedza takie frameworki jak: Java Spring, Flask, Django, Laravel czy Ruby on Rails, jedynie ASP.NET plasuje się wyżej w rankingu.

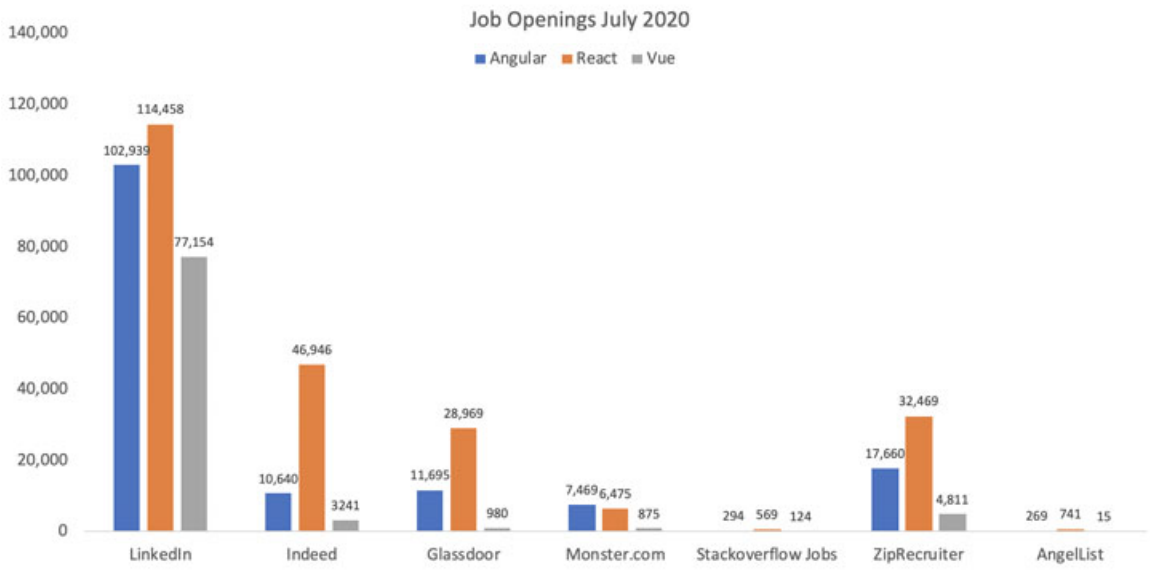
3.2 React.js

"React A JavaScript library for building user interfaces."[3] Tłumacząc na polski, React jest biblioteką do tworzenia interfejsów użytkownika. React jest oparty na komponentach, każdy komponent może posiadać własny stan i nim zarządzać, korzystając z komponentów można tworzyć bardzo złożone UI. Sporym plusem Reacta jest fakt, że raz poznany umożliwia również tworzenie aplikacji mobilnych przy wykorzystaniu React Native. Co dodatkowo wyróżnia React jest własna składnia, zbliżona do XML-a, czyli JSX. Jest to wymieszanie HTML-a i JavaScriptu, możemy dzięki niemu bez najmniejszych przeszkód umieszczać tagi znane z HTML-a, w kodzie JavaScriptowym, przy tworzeniu komponentów. Dodatkowo własne komponenty, również podczas tworzenia struktury dokumenty, mają składnię taką jak HTML.
Kolejnym wyjątkowym aspektem Reacta jest Virtual DOM. Jest to koncept programistyczny gdzie wirtualna reprezentacja UI jest przetrzymywana w pamięci i synchronizowana z prawdziwym DOM-em. Umożliwia to podawanie UI w jakim stanie ma się znajdować i jakie komponenty mają być wyrenderowane, również upewnia się, że faktyczny DOM odowiada temu stanowi. Ściagą to z programisty konieczność stworzenia samemu takich elementów jak: obsługę wydarzeń np. kliknięcie przycisku, manualne aktualizowanie DOM-u oraz manipulację atrybutami.



Rysunek 4: Skrótowa prezentacja działania Reacta

Konkurencje dla Reacta stanowi Angular i Vue, jednakże to React króluje zarówno jeśli chodzi o popularność, pobrania, gwiazdki na GitHubie, liczbe pakietów, które są zależne od Reacta. Niekwestionowym mistrzem jest również w kategoriach GitHub "Used by", tematach tworzonych na GitHubie.[1] Dodatkowym atutem przemawiającym za Reactem jest ilość pracy na rynku. Jak zostało zaprezentowane na obrazku poniżej, ponownie React króluje i w tej kategorii.



Rysunek 5: Zapotrzebowanie na rynku, na specjalistów z danego frameworku.[5]

3.3 TypeScript

Czym jest TypeScript? W skrócie jest to JavaScript rozszerzający go o statyczne definicje typów, takie jak np. string, number czy boolean. Został stworzony przez firmę Microsoft, przy wsparciu Google. Pisząc w TypeScripcie nie trzeba używać typów, inteferfejsów, typów generycznych czy też innych elementów udostępnianych. Można pisać zwykły JavaScript i pomimo, że zostaną wyświetlone błędy informujące o nieokreślonym typie, cały kod wykona się mimo wszystko. Każdy kod napisany w JavaScripcie będzie w pełni funkcjonalnym kodem TypeScriptowym. Jednakże pisząc w TypeScripcie przeglądarka nie rozumie tego języka, zna tylko i wyłącznie JavaScript, dlatego kod musi być kompilowany przy pomocy TypeScriptowego kompilatora lub Babela.[9] Dodatkowo sprawia to, że napisany kod będzie działał na dowolnym urządzeniu i przeglądarce, o ile te wspierają JavaScript.
Dlaczego programiści decydują się na TypeScript?
Wynika to przede wszystkim z obecności typów, pisząc w JavaScripcie należałoby pisać dodatkowe linijki kodu tylko i wyłącznie sprawdzające np. czy dane otrzymane przez serwer są odpowiednich typów, TypeScript robi wszystko za programiste. Błędy związane z typami należą do najczęściej popełnianych wśród programistów JavaScript, szczególnie w projektach o większej skali bardzo łatwo o takie błędy. Typować można nie tylko zmienne, ale też funkcje, zarówne argumenty jakie przyjmuje oraz typ zwracany.
TypeScript to nie tylko typy, posiada liczne dodatkowe funkcje, których JavaScript nie posiada. Są to chociażby wcześniej wspomnieniane interfejsy, można deklarować własne typy, typy generyczne oraz typy wyliczeniowe - enumy. Inną niedostępna funkcją jest chociażby przeciążanie funkcji. Wyglądem nie przypomina to kodu znanego z języków programowania typu Java czy C++, lecz spełnia dokładnie takie samo zadanie jak w tamtych językach.

3.4 Najistotniejsze wykorzystane biblioteki

W tej sekcji mogłoby znaleźć się ogromna ilość przeróżnych bibliotek, które są używane w projekcie. Sam Create-React-App(służy do generowania projektów Reactowych), sam w sobie pobiera kilkadziesiąt bibliotek. Tutaj jednak znajdą się tylko biblioteki, który zostały dodane na potrzeby projektu, a są to:

- Express
- Mongoose

- Stripe
- Nodemailer
- QRCode
- Material UI
- Redux

3.5 MongoDB

4 Proces tworzenia aplikacji

5 Podsumowanie

6 Możliwości dalszego rozwoju aplikacji

7 Bibliografia

References

[1] URL: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.

[2] Smita Deshmukh, Deepak Mane, and Abhijeet Retawade. "Building a Single Page Application Web Front-end for E-Learning site". In: 2019.

[3] Facebook. *React Docs*. URL: https://reactjs.org/.

[4] Adrian Horzyk. *NIERELACYJNE BAZY DANYCH – NoSQL I ASOCJACYJNE STRUKTURY DANYCH*. URL: http://home.agh.edu.pl/~horzyk/lectures/db/BazyDanychAccess-NoSQL.pdf.

[5] Daniel Hvidding and Ned Visolyaputra. *What's the most popular front-end framework?* 2020. URL: https://www.accenture.com/us-en/blogs/software-engineering-blog/hvidding-visolyaputra-front-end-framework.

[6] Microsoft. *Baza danych NoSQL — co to jest NoSQL?* URL: https://azure.microsoft.com/pl-pl/overview/nosql-database/.

[7] Microsoft. *Choose Between Traditional Web Apps and Single Page Apps (SPAs)*. URL: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.

[8] Microsoft. "Projekt internetowego interfejsu API". In: (). URL: https://docs.microsoft.com/pl-pl/azure/architecture/best-practices/api-design.

[9] Mirosoft. *TypeScript Docs*. URL: https://www.typescriptlang.org.

[10] Mozilla. *Wprowadzenie do Express/Node*. URL: https://developer.mozilla.org/pl/docs/Learn/Server-side/Express_Nodejs/Introduction.

[11] Stack Overflow. *2020 Developer Survey*. URL: https://insights.stackoverflow.com/survey/2020#developer-profile-coding-as-a-hobby.

[12] Gemius Polska. *E-commerce w Polsce 2020*. 2020. URL: https://www.gemius.pl/wszystkie-artykuly-aktualnosci/e-commerce-w-polsce-2020.html.

[13] Oracle Polska. *Co to jest baza danych?* URL: https://www.oracle.com/pl/database/what-is-database/.

[14] Oracle Polska. *Co to jest hurtownia danych?* URL: https://www.oracle.com/pl/database/what-is-a-data-warehouse/.

8 Spis rysunków

Spis rysunków

9 Spis tabel

Spis tabel