**ROZDZIAŁ 1**

**Podstawy aplikacji internetowych**

* 1. **Podstawowe pojęcia**
     1. Bazy danych

Bazy danych używane są praktycznie we wszystkich aplikacjach czy też stronach internetowych z których korzystamy na co dzień. Dzięki nim możemy zapisywać, analizować, przetwarzać rozmaite zbiory danych. Spójrzmy na to w ten sposób, że po wejściu na daną stronę może okazać się, że będziemy musieli utworzyć konto, żeby w pełni móc skorzystać z usług. Wejdziemy więc na podstronę z formularzem rejestracyjnym, gdzie po wypełnieniu formularza i zaakceptowaniu polityki prywatności danej strony klikniemy by nas zarejestrowało. Gdy wszystko się zgadza nasze dane wpisane w formularz „powędrują” do takiej bazy danych gdzie zostaną zapisane. Bazy te dzielą się na dwa podstawowe typy relacyjne (z ang. Relational Database Management System, RDBMS) oraz nierelacyjne (z ang, Not Only SQL, NoSQL). Wybór typu bazy danych zależy od indywidualnych potrzeb projektu. W aplikacji użyta zostanie baza danych nierelacyjna mongodb, dlatego też praca będzie oparta również o ten typ. Będzie korzystać z bezpłatnego hostingu poprzez Amazon AWS z serwerem europejskim umieszczonym we Frankfurcie, w Niemczech. Dostępne w darmowej wersji będzie 512mb co będzie wystarczające do tego projektu. W fazie rozwoju będzie możliwość rozszerzyć dostępną przestrzeń za opłatą.

Systemy zarządzania bazami danych opartych na relacjach (z ang. Relational Database Management System, RDBMS), takie jak Oracle lub PostgreSQL, cieszą się w obecnych czasach dużym uznaniem wśród programistów, którzy doceniają istotę wydajności baz danych oraz jej normalizacji. Jednakże, koncepcja NoSQL staje się coraz bardziej popularna, zwłaszcza jeśli chodzi o rozwiązania sterowania danymi (z ang. modern data-driven solution). Chociaż niektórzy programiści są sceptyczni jeśli chodzi o nierelacyjne bazy danych, lecz wraz z rozwojem oraz rozpowszechnieniem się rozwiązań typu BigData, NoSQL stało się niezbędną formą dla wielu systemów. Warto pamiętać również, że jedno nie wyklucza drugiego to znaczy, że bazy relacyjne i nierelacyjne mogą współistnieć w jednym systemie. Ponadto, ogólnie uważa się, że wyszukiwanie danych jest prostsze w relacyjnych systemach RDBMS, dlatego też często niezbędne jest stosowanie obu rodzajów rozwiązań równocześnie.[[1]](#footnote-1)

Termin NoSQL, wywodzący się z języka angielskiego „Not Only SQL”, który odnosi się do systemów zarządzania danymi, nieopierającymi się na modelu relacyjnym. Odrzucenie tego modelu skutkuje możliwością przechowywania danych bez konieczności stosowania ściśle określonych schematów. Dodatkiem stosowania systemów NoSQL w projektach jest brak potrzeby stosowania złączeń, co w znacznym stopniu ułatwia skalowanie w poziomie i skutkuje bardziej wydatnymi zapytaniami. Termin NoSQL częstokrotnie mylony jest z daną technologią, lecz w praktyce opisywany jest jako zbiór rozwiązań, które służą do przechowywania danych, różniących się od modelu relacyjnego. Termin „Not Only SQL” podkreśla możliwość używania elementów języka SQL, zwłaszcza składniowo podobnych języków zapytań.[[2]](#footnote-2)

Chcąc porównać relacyjne bazy danych oraz bazy NoSQL, który nie posiada relacji oraz często nie ma zdefiniowanego języka zapytań SQL jak i nie mają z góry określonego schematu może posłużyć się przykładem. Możemy założyć, że będziemy implementować system do przechowywania danych o pracownikach pewnej organizacji. Każdy z pracowników może posiadać po kilka numerów telefonów.

Tabela 1 Telefony\_komórkowe

|  |  |  |  |
| --- | --- | --- | --- |
| Id\_Telefon | Pracownik\_Id | Numer tel. | Typ\_Id |
| 1 | 1 | 516245333 | 1 |
| 2 | 2 | 508202332 | 2 |
| 2 | 3 | 668512563 | 1 |

Tabela 2 Pracownicy

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pracownik\_Id | Imię | Nazwisko | Wiek | Wynagrodzenie | Wydział\_Id |
| 1 | Paweł | Kowalski | 28 | 6000 | 1 |
| 2 | Gaweł | Nowak | 32 | 5000 | 2 |

Tabela pierwsza (Telefony\_komórkowe), będzie zawierać informacje o numerach telefonów, którymi posługują się pracownicy. Druga Tabela (Pracownicy), będzie zawierać podstawowe dane o pracownikach takie jak: imię, nazwisko, wiek, wynagrodzenie oraz klucz po którym będzie można zlokalizować dział w którym dany pracownik pracuje. Bez wątpienia, można zauważyć co najmniej kilka relacji. Pierwsza występuje pomiędzy tabelami Telefony oraz Pracownicy, gdzie wykorzystano klucz obcy Pracownik\_Id do dokonania realizacji, który wskazuje na konkretny wiersz w tabeli drugiej (Pracownicy). Analogicznie wygląda to z Wydział\_Id oraz Typ\_Id. Takie podejście jest typowe dla relacyjnych baz danych to znaczy, że wszystkie dane muszą być odpowiednio (atomowo) podzielone pomiędzy kolumnami. Ważne żeby nie trzymać w tej samej tabeli np. imienia i nazwiska (złączenie danych), z powodu niezgodności z tzw. pierwszą postacią normalną. Teraz należy rozważyć jak dany przykład mógłby zostać zrealizowany w podejściu NoSQL?

Tabela 3 Wszystkie\_dane

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pracownik\_Id:  1 | Imię:  Paweł | Nazwisko:  Kowalski | Wiek:  28 | Zarobki:  6000 | Tel\_prywatny:  516245333 | Tel\_służbowy:  605223456 |
| Pracownik\_Id:  2 | Imię:  Gaweł | Nazwisko:  Nowak | Wiek:  32 | Zarobki:  5000 | Tel\_domowy:  425234555 | Tel\_służbowy: |

Tabela trzecia różni się od przykładów z użyciem bazy SQL w dwóch miejscach. Pierwszą różnicą jest różnorodność wierszach, każdy może zawierać różne „kolumny”. Zauważyć można również, że nie ma tutaj z góry przyjętego formatu. Widać wyraźnie, że jeden wiersz może zawierać informacje o telefonie prywatnym, podczas gdy drugi dotyczy telefonu służbowego. W podejściu NoSQL nie istnieją typowe relacje, takie jak możemy znaleźć w tradycyjnych systemach baz danych, gdzie jest narzucana struktura przez silnik zarządzania bazą danych, dlatego też można stworzyć pola z „kluczami” podstawowymi i obcymi ale to programista jest odpowiedzialny za utrzymywanie oraz egzekwowanie ich. Konsekwencją tego jest to że bazy NoSQL są bazami z nierelacyjnymi modelami danych. W kontekście przykładu z telefonami, NoSQL może być preferowany nad RDBMS w sytuacjach, gdzie dane różnią się pomiędzy poszczególnymi rekordami. W Bazach nierelacyjnych łatwiej jest osiągnąć skalowanie horyzontalne, co pozwoli na bardziej elastyczne zażądanie zmiennymi informacjami. W przeciwieństwie do modelu relacyjnego, w którym dodanie nowej kolumny wyłącznie dla jednego rekordu jest niezgodne z normami. W modelu nierelacyjnym struktura danych może przyjmować formę par kluczy i wartości. W przypadku adresów, dokumentów o różnych strukturach lub niestandardowych encji, utrzymanie tych informacji jest zwykle prostsze w niż w bazach RDBMS.[[3]](#footnote-3)

* + 1. ORM

Interfejs dostępu do baz danych jest narzędziem programistycznym, ułatwiającym dostęp oraz manipulację danymi w bazach danych. Jednym z popularniejszych podejść jest mapowanie obiektowo-relacyjne (ORM), które pozwala na integrowanie struktury obiektowej systemu z bazą danych o charakterze relacyjnym co może być mylące z powodu tego, że bazą danych użytą w pracy jest baza NoSQL (Mongodb), która również pozwala na korzystanie z ORM. Dzięki rozwiązaniom ORM, programiści programujących w takich językach jak C#, Java czy javascript są w stanie manipulować danymi bez konieczności korzystania z języka SQL. Zamiast operować na rekordach, aplikacje mogą używać gotowych obiektów, które są wynikiem zapytań do bazy danych.[[4]](#footnote-4)

W projekcie zostanie użyte Prisma ORM, która jest otwartym narzędziem do mapowania obiektowo-relacyjnego. Składa się z następujących części: Prisma Client to bezpieczny w typach, automatycznie generowany kreator zapytań dla Node.js oraz TypeScript, Prisma Migrate jest to system migracyjny, Prisma Studio która jest graficznym interfejsem użytkownika służącym do przeglądania oraz edycji danych w bazie danych i można uruchamiać tylko lokalnie. Prisma Client może być użyty w dowolnej aplikacji backendowej Node.js opartej na architekturze REST API, GraphQL, gRPC lub w innym rozwiązania wymagającym dostępu do bazy danych.

Każdy projekt wykorzystujący narzędzie z Prisma ORM zaczyna się od pliku schematu Prisma. Schemat ten pozwala programistą na zdefiniowanie modeli aplikacji w bardzo intuicyjnym języku modelowania danych zawierających również połączenie z bazą danych oraz definiuje generator. W schemacie trzeba skonfigurować trzy rzeczy: źródło danych określające połączenie z bazą danych za pomocą zmiennej środowiskowej, generator, który wskazuje potrzebę wygenerowania klienta Prismy oraz model danych dla przykładu model o nazwie Użytkownik będzie zawierać pola o nazwie ID które za typ przyjmie string (tekstowy typ danych) oraz można ustawić, że będzie wygenerowany automatycznie następnie imię które również będzie stringiem i będzie to pole obowiązkowe oraz numer telefonu, który ustawimy typ danych jako number (w języku typescript jest to typ określający wszystkie cyfry całkowite oraz zmiennoprzecinkowe). Warto zaznaczyć, że w innych językach programowania np. Java, liczby całkowite mają osobne typy INT, BYTE, SHORT, LONG oraz liczby zmiennoprzecinkowe DOUBLE czy FLOAT.

Patrząc na relacyjne bazy danych i pracę z nimi, które stanowi wąskie gardło w  rozwoju aplikacji debugowanie zapytań SQL lub złożonych obiektów ORM często

pochłania godziny czasu programistycznego. Prisma ORM ułatwia programistom zrozumienie zapytań do bazy danych, dostarczając bezpiecznie typowo API do przesyłana zapytań do bazy danych, które zwracają obiekty JavaScript, więc wnioskując głównym powodem używania Prisma ORM jest zwiększenie produktywności programistów. Poważnym problemem narzędzi bazodanowych, istniejących w ekosystemie Node.js i  TypeScript jest konieczność dokonania poważnego wyboru pomiędzy produktywnością, a  kontrolą. W czystym języku zapytań SQL istnieje pełna kontrola, lecz niska produktywność z powodu czystych ciągów SQL do bazy danych, które jest uciążliwe oraz wiąże się z dużym nakładem pracy. Kolejnym problemem jest brak bezpieczeństwa typów dla wyników zapytania, które można ręcznie wprowadzić, ale znów wymaga to ogromnego wkładu pracy oraz dużych przeorganizowań za każdym razem, gdy zmienia się schemat bazy danych lub zapytania, aby zachować synchronizację typów. Istnieją również kreatory zapytań SQL w których jest wysoka kontrola i średnia produktywność. Największym ich minusem jest ciągłe myślenie ze strony programisty w kategoriach SQL co wiąże się z  kosztem poznawczym oraz praktycznym przekładaniem danych relacyjnych na obiekty. OMR-y służą głównie do przejścia przez różnice między obiektami, a podstawowymi elementami bazy danych czyli relacją.[[5]](#footnote-5)

* + 1. Rest API

REST (z ang. Representational State Transfer) jest to architektura używana do projektowania usług używanych na wielu różnych platformach oraz środowiskach. Ma na celu wsparcie interoperacyjności oraz sieci WWW (z ang. World Wide Web). Nieskończoność dostępu i uniwersalność konsumpcji na różnych platformach to dwie kluczowe cechy tej architektury. Jest powszechnie stosowana sposobem publikowania usług w internecie. API (z ang. Application Programming Interface) czyli interfejsy programowania aplikacji oraz REST są standardową częścią projektowania mikro usług. REST API jest znane jako API internetowe, które składa się z punktów końcowych. Każdy punkt jest konkretnie zaimplementowaną funkcjonalnością procesu biznesowego. Interfejsy te są dostępne za pomocą protokołu HTTP (z ang. Hypertext Transfer Protocol), zawierającego standardowe zapytania takie jak: GET, POST,DELETE oraz PUT. RESTful API jest wywoływane za pomocą adresu URI (z ang. Uniform Resource Identifier).

Wyzwaniem jakim stało przed REST API było zdefiniowanie standardowego formatu komunikacji czyli zapytanie (z ang. Request) i odpowiedzi (z ang. Response). Na początku stosowano nieformalny tekst opisu interfejsów. Następnie pojawiły się dokumenty JSON (z ang. JavaScript Object Notation), który był standardem jako zwykły tekstm łatwy do identyfikacji oraz przetwarzania dla maszyn w różnych sieciach i platformach. Jednym z wyzwań pozostaje stworzenie standaryzowanego sposobu opisu usług REST. Specyfikacja OpenAPI jest jednym z rozwiązań tego dylematu. Luźne powiązania i elastyczny dostęp do usług REST mogą otwierać szerokie możliwości przesłania niepoprawnych danych wejściowych w żądaniu, co może prowadzić do błędów w usłudze, które mogą zostać niezauważone podczas standardowych procesów testowych i statycznej analizy kodu. Różne dodatkowe biblioteki dostarczanych przez różnych dostawców stosowane w procesie tworzenia interfejsów REST, wprowadza dodatkową złożoność w testowaniu. W takim razie wykrywanie i rozwiązywanie ewentualnych błędów w usłudze staje się kluczowe dla zapewnienia jej stabilności, zwłaszcza gdy jest ona kluczowa dla danego przedsiębiorstwa obsługującego klienta końcowego. Niezależnie jakie dane zostaną przesłane jako żądanie, odpowiedź powinna być dostarczona w sposób elegancki, minimalizując potencjalne zakłócenia i błędy czasu wykonania. Różne wysiłki badawcze zostały podjęte w celu złagodzenia wyzwań związanych z testowaniem interfejsów REST API poprzez wprowadzenia różnych platform i automatycznego generowania podejść do testów jednostkowych. Jednak nadal są braki w przeglądu stanu wiedzy.[[6]](#footnote-6)

Na rynku są też inne podejścia do tworzenia interfejsów programistycznych, są nimi na przykład GraphQL czy gRPC. Zestawiając je ze sobą można dopatrzeć się różnic w pomiarach czasu odpowiedzi wydajności czy też ilości przesłanych danych. Jeśli rozpatrywanym przypadkiem była by ilość pobieranych danych to możemy zauważyć, że interfejs GraphQL najlepiej sobie z tym poradził, jednakże jeśli dane byłyby średnie oraz małej wartości pobranych rekordów, to wydajność REST API i GraphQL były by bardzo zbliżone z niewielką przewagą na korzyść REST API. Przewaga REST API również dotyczy małej ilości danych w przypadku prostych struktur zapytań. Technologia gRPC osiągnęła słabsze wyniki wydajnościowe dwu, a czasem nawet czterokrotnie. GRPC wyróżnia się od REST API i GraphQL pobieraniem o połowę mniej danych podczas zapytań do serwera. Gdyby rozważać przykład, którym jest dodanie użytkownika do bazy danych, zauważalna jest przewaga REST API nad innymi technologiami, który osiąga w kryterium wydajnościowym i czasu odpowiedzi lepsze wyniki, dzięki czemu osiąga dwukrotnie lepsze wyniki. Możemy zauważyć, że dokonanie ostatecznego wyboru interfejsu jest nieco bardziej skomplikowane niż mogłoby się wydawać na pierwszy rzut oka. Przy podejmowaniu decyzji co do sposobu implementacji aplikacji, trzeba wziąć pod uwagę wiele czynników, takich jak wielkość danych przesyłanych pomiędzy serwerem, a użytkownikiem, ilość użytkowników oraz wydajność urządzeń. Rozważyć również należy czy zależy wygodę w korzystaniu z interfejsu. Wielu programistów najczęściej wybiera REST API, ze względu na swoją popularność oraz wygodę lecz nie zawsze jest to najlepszy wybór. REST API może być najlepszym wyborem w przypadku dużej ilości danych, ale również dobrze sprawdził by się GraphQL, jednakże przy małej ilości danych warto również rozważyć korzystanie z gRPC.[[7]](#footnote-7)

* + 1. Języki programowania

Systemy informatyczne nie mogły by mieć miejsca bez języków programowania. Wraz z rozwojem informatyki powstaje wiele ich rodzajów, służących do różnych rzeczy. Tak na przykład język JavaScript służy do tworzenia stron internetowych ze strony wizualnej (front-end) oraz ze strony obsługi serwera itp. (back-end). Jest to narzędzie które powstało już w latach dziewięćdziesiątych ubiegłego wieku i przeszło długą drogę rozwoju. Jednym z najważniejszych wydarzeń jest integracja go z Wasm (WebAssembly), który jest formatem instrukcji binarnych umożliwiających wysokowydajne wykonanie kodu w przeglądarkach internetowych. Pozwala programistom na uruchomienie kodu napisanego w językach takich jak C, C++ czy Rust obok JavaScriptu, otwierając nowe możliwości dla aplikacji internetowych co zapewnia ścieżkę do poprawy wydajności aplikacji internetowych. Programiści mogą wykorzystać Wasm do przeniesienia zadań wymagających dużych zasobów takich jak renderowanie czy obliczanie w celu osiągnięcia praktycznie natywnej szybkości w przeglądarce. JavaScript odgrywa kluczową rolę w budowaniu progresywnych aplikacji webowych (PWA), umożliwiając dynamiczne i interaktywne funkcję, które sprawiają, że aplikacje są atrakcyjne dla użytkownika. W miarę wzrostu popularności tych aplikacji, programiści muszą zmierzyć się z dostosowaniem do nowego paradygmatu budowania aplikacji internetowych. Przyszłość JavaScriptu jest ściśle wiąże się z nieustannie zmieniającym się krajobrazem ram i bibliotek. Wiodące biblioteki takie jak React, Angular czy Vue.js mają znaczący wpływ w jaki developerzy projektują interfejsy użytkownika oraz aplikacje jednostronicowe. Dodatkowo na rynku pojawiają się nowe biblioteki, które muszą sprostać konkretnym potrzebą takim jak zarządzanie stanem, routowanie, renderowanie po stronie serwera. Dlatego dla programistów jest tak ważne śledzenie najnowszych trendów w celu pozostaniem konkurencyjnym w dynamicznie rozwijającej się branży tworzenia aplikacji internetowych. Dodatkowo język JavaScript jest językiem bardzo elastycznym wykorzystywanym w rozwoju międzyplatformowym. Biblioteka jak na przykład React Native oparta na tym języku pozwala na budowanie aplikacji mobilnym na urządzenia z androidem jak i oparte na ios, a Electron umożliwia programistom budowanie aplikacji desktopowych. To wszystko sprawia, że język JavaScript jest atrakcyjny dla programistów, którzy za pomocą jednego kodu mogą dotrzeć do wielu platform.[[8]](#footnote-8)

Istnieje również język oparty na JavaScript i jest nim TypeScript. Różnica pomiędzy tymi językami jest to, że TypeScript jest rozszerzony o system typów. By został odczytany przez przeglądarkę musi zostać skompilowany do JavaScriptu. Wynalazcą tego języka była firma Microsoft z wkładem Googla. Jest on najbardziej udanym spośród języków stopniowo typowanych oraz najpopularniejszym językiem kompilowanym do JavaScriptu. Warto wspomnieć, że każdy program napisany w JavaScript jest również programem TypeScript. Dodanie do TypeScriptu typów pomaga zaoszczędzić wiele godzin debugowania. Gdy firmy tworzące duże aplikacje mające miliony linijek kodu, muszą coś w nim zmienić typy pozwalają dokładniej poznać co spowodują zmiany dla pewnych modułów na poziomie kompilacji.[[9]](#footnote-9)

Najpopularniejszym językiem wykorzystywanym do tworzenia aplikacji internetowych jest PHP. Wiele aplikacji już istniejących oraz systemów wykorzystuje właśnie ten język co jest najlepszą motywacją do ciągłego rozwoju. Ten język również posiada wiele szkieletów jak na przykład Laravel, który jest wykorzystywany do niewielkich projektów, niewymagających wielkiego wkładu finansowego. Kolejnym popularnym szkieletem jest Symfony, przeznaczony głównie do projektów złożonych i wymaga większych wkładów finansowych.[[10]](#footnote-10)

Python to również bardzo popularny język, szczególnie w dzisiejszych czasach ze względu na sztuczną inteligencję oraz uczenie maszynowe. Jest to język wysokiego poziomu, który powinien być dla ludzi prosty do pisania oraz czytania oraz dla komputerów do przetwarzania i odczytywania. Python używa interpretera napisanego w jezyku C do przekształcenia kodu na język maszynowy. [[11]](#footnote-11) Sam język ma wiele zalet takich jak działanie na wielu platformach i systemach operacyjnych czy istniejące wiele bibliotek posiadające Pythonowe API, a nawet możliwość korzystania z bibliotek języka C (.so, .dll). Dodatkowo Python jest wygodniejszy w nauce niż wiele innych języków, ponieważ kod realizujący tą samą funkcjonalność przy podobnym poziomie obsługi błędów i czytelności, zazwyczaj jest krótszy niż kod w C, co ułatwia jego prezentację i omówienie.[[12]](#footnote-12)

Java jest również jednym z najpopularniejszych i najlepszych języków używanym na całym świecie, który pozwala tworzyć programy na poziomie przedsiębiorstwa. Język ten oparty jest na języku Oak, wynalezionym przez Sun Microsystems na początku lat dziewięćdziesiątych. Celem tego języka było stworzenie nowej generacji inteligentnych urządzeń, które działają niezależnie od systemu operacyjnego. By to osiągnąć programiści z Sun Microsystems planowali użyć języka C++, ale nie doszło to do skutku. Oak nie odniósł jednak sukcesu i już w 1995 roku zmienili nazwę na Java oraz wprowadzili zmiany w celu rozwijania WWW. Java jest językiem zorientowanym obiektowo (OOP – Object-Oriented Programming), bardzo podobnym do C++. Został jednak uproszczony przez usunięcie części, które są najbardziej podatne na błędy. Kod napisany w języku Java jest konwertowany na kod bajtowy po kompilacji, a kod bajtowy jest odczytywany przez interpreter.[[13]](#footnote-13)

C++ jest wielozadaniowym językiem programowania. Stworzony w Bella Labs w 1979 roku przez Biyarne'a Stroustrupa w celu rozszerzenia możliwości języka programowania C oraz wprowadzić programowanie obiektowe. Początkowo został nazwany „C with Classes” co znaczy C z klasami, które są używane do programowania obiektowego, lecz w 1983 roku został zmieniony na nazwę obecną C++. Język ten ma zdolność kompilowania programów napisanych w języku C, lecz bez wzajemności, gdyż język C nie posiada takiej funkcji. Najczęściej język ten jest używany do w produkcji komponentów związanych z systemami operacyjnymi. Jest też używany do produkcji gier, tworzenia programów kilent-serwer.[[14]](#footnote-14)

* + 1. Interfejs Użytkownika frameworki

Angular, React czy też Vue.js są najpopularniejszymi szkieletami dla JavaScriptu. Są wykorzystywane do do tworzenia interfejsów graficznych za pomocą SPA (z ang. Single Page Application). Podając przykład Reacta komponenty wielokrotnego użytku wykorzystane w interfejsie opisywane za pomocą JSX, który jest formatem zapisu kodu HTML wewnątrz języka JavaScript, jeżeli chodzi o TypeScript jest to TSX. W interfejsie Vue używane są komponenty, które wykorzystują szablony HTML do swojej implementacji. Angular, który jest następcą Google Angular JS, jest rozwijany przez ten sam zespół. Budowa Angulara narzuca programiście podział komponentów na trzy elementy, którymi są logika, interfejs użytkownika oraz style, podczas gdy Vue trzyma w jednym pliku wszystkie komponenty. W większości nowoczesnych stron internetowych wykorzystuje się JS (JavaScript) do zwiększenia responsywności i interaktywności. W tradycyjnych witrynach, nowy dokument HTML pobierany jest za każdym razem, kiedy użytkownik przechodzi na nową stronę, co jest powolne. Natomiast w nowoczesnych szkieletach elementy są pobierane tylko wtedy gdy elementy zostaną zmienione co znacznie przyspiesza działanie strony.[[15]](#footnote-15)

Angular jest frameworkiem, który występuje głównie w dwóch wersjach jakimi są Angular JS oraz Angular 2+. Angular JS jest to starsza wersja oparta na języku JavaScript, niekontynuowaną już przez twórców. Natomiast Angular 2+ jest nowszym rozwiązaniem, opratym na TypeScript i HTML. Angular 2+ jest wykorzystywany do tworzenia aplikacji klienckich w HTML, CSS, JavaScript lub TypeScript. Zapewnia aplikacjom czytelną i elastyczną strukturę co ułatwia jej zrozumienie i utrzymanie co sprawia, że ten sam kod można wykorzystywać w różnych aplikacjach. Aplikacje te składają się z modułów NgModule, które są kontenerami dal kodu związanego z daną domeną aplikacji. Zapewniają kontekst kompilacji swoim elementom i zawierają składniki dostawców usług i inne pliki. Aplikacje napisane przy użyciu Angulara posiadają moduł główny nazywany AppModule, który inicjuje uruchomienie aplikacji oraz może zawierać wiele modułów podrzędnych. Angular częściowo opiera się na wzorcu MVC (z ang. Model View Controller), w którym komponent pełni rolę kontrolera za to szablon – widoku. Szkielet ten zapewnia kontrolę nad skalowaniem aplikacji, umożliwiając budowę modeli danych w RxJSm Imutable.js lub innym modelu push w celu zapewnienia szybkości i wydajności aplikacji internetowych.[[16]](#footnote-16)

React który jest frameworkiem pierwotnie rozwijanym przez Facebook jest odpowiednikiem w języku JavaScript dla biblioteki XHP w PHP, umożliwiającej tworzenie niestandardowych komponentów. Zadebiutował w maju 2013 roku, skupiając się głównie na renderowaniu interfejsu użytkownika. Jego największymi zaletami jest niewielki nakład nauki i używania i lżejsza konstrukcja zestawiając go z innymi technologiami. Rozwój jest istotny w ramach zmiany podejścia do tworzenia aplikacji internetowych. Jego największym osiągnięciem jest optymalizacja funkcji DOM, który jest sposobem reprezentacji XML i HTML, co ma znaczenie ponieważ manipulacja funkcjami DOM jest kosztowna pod względem zasobów obliczeniowych. Został zaprojektowany tak żeby manipulacje DOM były jak najbardziej zminimalizowane poprzez wykorzystanie wirtualnego DOM co przyśpiesza działanie aplikacji.[[17]](#footnote-17)

Vue jest to nowoczesny framework JavaScript, często porównywanym do Angulara czy Reacta. Stworzony została przez firmę Google i jest bardziej skoncentrowany na warstwie widoku oraz jest bardziej uproszczony od Angulara. W przypadku Vue również wykorzystywany jest wirtualny DOM tak jak w przypadku Reacta, ale bez konieczności korzystania z oddzielnej modułowej biblioteki. Opisywany jest jako framework postępowy, który jest elastyczny i skalowalny, nadając się do małych projektów jak i do pełnowymiarowych aplikacji jednostronicowych. Jego zaletą jest to, że nie jest ściśle związany z wzorcem MVVM. Aktywni programiści cały czas rozwijają Vue i dostosowują go do potrzeb społeczności. Framework posiada automatyczne śledzenie zależności komponentów, optymalizując tym samym czas i pracę programisty. Z powodu swojej szybkości i efektywności umożliwia skupienie się na istotnych funkcjonalnościach aplikacji.[[18]](#footnote-18)

Oceniając wydajność technologii używając testów oraz analizę wsparcia technicznego oferujących przez developerów danego frameworka można zauważyć, że najszybciej rozwijającym oraz uruchamiającym się szkieletem jest Angular. Zawdzięcza to efektywnemu przetwarzaniu procesów związanych z uruchamianiem aplikacji, radzi sobie również z rożnymi przeglądarkami internetowymi. Jeśli weźmiemy pod uwagę framework wykorzystywany do tworzenia małych i średnich aplikacji to będzie nim Vue. Proces budowy i uruchamiania aplikacji w Vue jest podobny do tego w React, wiec ciężko stwierdzić, który szkielet działa szybciej. Angular otrzymujący wsparcie od strony Google jest często aktualizowany, co gwarantuje wydawanie nowych wersji co 6 miesięcy. Developerzy przywiązują dużą wagę do udoskonaleń tego szkieletu. W przypadku Reacta jego rozwój następuje poprzez interakcję ze społecznością programistów. Jest aktualizowany poprzez zmiany wysyłane przez twórców oraz zewnętrznych twórców, które muszą przejść przez proces recenzji na platformie GitHub. Każdy programista może wprowadzić zmiany, które muszą być zaakceptowane przez twórców, którzy reagując krytycznie na błędy znalezione w starszych wersjach. Vue z drugiej strony jest najmłodszym z wymienionych frameworków i chociaż jest stworzony przez byłego developera Anulara, nie cieszy się tak dużą społecznością profesjonalistów, którzy wspierają tą technologie. Licencja open source pomaga ciągłemu wzrostu społeczności korzystającej z Vue.[[19]](#footnote-19)

Jeśli poddalibyśmy testom wszystkie wymienione szkielety, podzielonym na siedem kategorii opartych na działaniach CRUD (Create, Read, Update, Delete), można zauważyć, że czas tworzenia się wierszy podczas ładowania strony najlepiej prezentuje się w frameworku React, ze względu na wydajność mikro serwisów użytych w tej technologii. Jeśli rozpatrywać odświeżanie strony to wszystkie szkielety wypadają podobnie, co sugeruje niemal identyczną wydajność procesów przetwarzania podglądu załadowanych informacji. Angular wypada najszybciej, kiedy przychodzi do wyboru wiersza poprzez jego zaznaczenie. Szybszy jest o średnio 30%. Biorąc pod lupę edycję losowych wierszy, React okazuje się najmniej wydajnym szkieletem. Spowodowane jest to czasem wykonania zewnętrznych funkcji edycji danych. Patrząc na usuwanie wierszy od pięćdziesięciu do tysiąca rekordów jest się w stanie zauważyć, że Angular radzi sobie bardzo szybko z usuwaniem małych ilości danych, sugerując, że częste oczyszczanie danych z małych pakietów jest najbardziej optymalne w tej technologii. Patrząc na szybsze usuwanie większych pakietów danych, React i Vue radzą sobie najlepiej przy czym czas miał rozkład logarytmiczny. Testując utworzenie dziesięciu tysięcy rekordów, Vue wykazuje się największą wydajnością co prawdopodobnie wynikało z efektywności procesów używanych w tej technologii. Z drugiej strony React jest najwolniejszym frameworkiem w tworzeniu danych, głównie z powodu korzystania z zewnętrznych funkcji. Framework dzięki dobrej komunikacji między komponentami zbliżył się do Vue. Rozpatrując ilość kodu który jest potrzebny do utworzenia testowych aplikacji, szkielet Vue wymaga go najmniej, choć różnice między konkurencyjnymi technologiami są niewielkie.[[20]](#footnote-20)

Vue jako szkielet interfejsów użytkownika wydaje się być najszybszy, osiągając najlepszą wydajność dzięki wykorzystaniu mikro procesów, sprawnie komunikujących się oraz efektywnie operujących na dużych zbiorach danych. Na drugim miejscu plasuje się na drugim miejscu pod względem szybkości zwłaszcza jeśli chodzi o małe zbiory danych. React, który uplasował się na ostatnim miejscu z wymienionych szkieletów, wypada najwolniej, zwłaszcza na operacjach związanych z edycją danych, chociaż jego wewnętrzne funkcje działają szybko. Warte zauważenia jest to, że zewnętrzne funkcjonalności mogą spowolnić całą aplikację. Podsumowując, wszystkie frameworki mogą być wydajne, należy jednak być na bieżąco z nowymi technologiami i ich dokumentacją.[[21]](#footnote-21)

* + 1. Systemy kontroli wersji

VCS (z ang. Version Control Systems), czyli systemy kontroli wersji jest oprogramowaniem zezwalającym na przeglądanie oraz gromadzenie kolejnych wersji kodu źródłowego. Patrząc wstecz systemem kontroli wersji cieszącym się największą popularnością przez wiele lat był CVS, a później zgodny z nim SVN (z ang. Subversion). Lecz w dzisiejszych czasach największą popularnością cieszy się stworzony przez Linusa Torvaldsa system o nazwie Git. Różni się on od poprzedników tym, że Git jest rozproszonym systemem kontroli wersji co oznacza, że projekty magazynowane są na głównym serwerze, ale również na indywidualnych stanowiskach pracy. Narzędzie Git umożliwia właśnie ich scalanie i synchronizację[[22]](#footnote-22).

Programiści używający Gita na co dzień muszą posługiwać się specyficzną terminologią, która to określa poszczególne czynności, powodując, że na początku stosowania systemu, może utrudniać zrozumienie, co tak naprawdę się dzieje. Miejscami gdzie profesjonaliści przetrzymują kolejne wersje kodu źródłowego, noszą imię repozytoriów. Istnieją ich dwa rodzaje lokalne oraz zdalne. Częstym skrótem używanym do skrócenia długiego słowa repozytorium używa się słowa „repo”. Pliki zapisywane są w trzech różnych lokalizacjach: w folderze projektu nazywanym również jako katalog roboczy (z ang. working directory), w lokalnym „repo” znajdującym się w podkatalogu .git w folderze projektu oraz w zdalnym repozytorium przechowywanym na serwerze, na przykład na platformach takich jak: GitLab, GitHub, Azure DevOps, Bitbucket czy inne. Wszelkie modyfikacje plików w projekcie są monitorowane, co znaczy, że każda zmiana pliku jest śledzona. Cyklicznie należy zaakceptować te zmiany poprzez dokonanie jednego zatwierdzenia co w terminologii Gita nazywane jest commitem. Czasownik angielskiego pochodzenia commit oznacza przesładnie zmienionej części projektu do lokalnego repozytorium. W języku polskim proces ten nazywa się zatwierdzeniem zmian. Następnie możemy wypchnąć zmiany do zdalnego repozytorium co w terminologii znaczy commit push. Przed wypchnięciem zmian powinno się zobaczyć czy repozytorium nie ma na serwerze nowszej wersji projektu, działanie te nazwane jest fetch, na polski pobranie. Zestawienie push i fetch używane jest do synchronizacji repozytorium. Każdy programista będący członkiem zespołu może pobrać najnowszą wersję projektu ze zdalnego repozytorium co w terminologii nazywa się pull. Klonowanie repozytorium oznacza pobraniem całego projektu ze zdalnego „repo” na komputer, na którym nie było żadnej wersji projektu. Należy zwrócić uwagę na trzy główne korzyści korzystania z systemu kontroli wersji. Najważniejszą zaletą jest regularne tworzenie kopii kodu na komputerze programisty, jak zarówno na zdalnym serwerze. Kolejną zaletą jest możliwość tworzenia wersji, co znaczy, że specjaliści mają dostęp do nie tylko najnowszej wersji produktu jak również do w całości oraz wybranych plikach. Wersjonowanie powiązane jest z taką zaletą, że każda zmiana w zdalnym repozytorium jest podpisana członkiem zespołu dokonującym zmiany. To wszystko pozwala na skuteczne zarządzaniem projektu, śledzenie jego rozwoju bez konieczności osobistego monitorowania go na każdym stanowisku pracy oraz efektywne zarządzanie zespołem pracującym nad projektem dzięki posiadaniu wiarygodnych danych o jego postępie.[[23]](#footnote-23)

Git jest narzędziem wiersza poleceń służącym do tworzenia własnego serwera kontroli wersji. Istnieje wiele publicznie dostępnych platform umożliwiających łatwe utworzenie zdalnego repozytorium bez potrzeby konfiguracji. Platformy te oferują interfejs internetowy do przeglądania repozytoriów, które posiadają wiele dodatkowych funkcji na przykład zarządzanie uprawnieniami dostępu do gałęzi. Najpopularniejsze serwisy, które oferują trzymanie repozytoriów to na przykład GitHub, GitLab, Azure DevOps Services (kiedyś Visual Studio Team Foundation Services) czy Bitbucket. Atlassian, który jest właścicielem Bitbucket, dostarcza popularny klient Git nazwany SourceTree, posiadający graficzny interfejs użytkownika, który dziła nie tylko z Bitbucket, ale również z innymi usługami Git. SourceTree jest często wykorzystywany obok wbudowanego klienta w nowszych wersjach Visual Studio. Wiele operacji dokonywanych prze Gita takich jak akceptowanie czy wysyłanie commitów, można wykonywać bezpośrednio z Visual Studio, kiedy to SourceTree jest używany do bardziej zaawansowanych działań lub rozwiązywania problemów.[[24]](#footnote-24)

* + 1. Modelowanie systemów

Unified Modeling Language czyli w skrócie UML jest Zunifikowanym językiem Modelowania, który jest graficzny i służy do specyfikowania, wizualizowania, konstrukcji oraz dokumentacji fragmentów w systemach informatycznych. To umożliwia standaryzację procesu tworzenia wycinków systemu, które obejmują różnorodne elementy, takie jak funkcję systemowe lub schematy baz danych. Język ten integruje zalety z różnych obszarów, takich jak modelowanie danych poprzez diagram związków encji, modelowanie przepływów danych za pomocą diagramów przepływu, analiza obiektowa związana z modelowaniem obiektowym oraz zarządzanie złożonością przy użyciu komponentów. Początkowo, Zjednoczony Język Modelowania (UML) był używany gównie do tworzenia systemów informatycznych. Jednakże, ze względu na jego efektywność, jako narzędzia komunikacji, zaczął być stosowy równocześnie w innych dziedzinach takich jak telekomunikacja, transport, przemysł obronny i lotniczy czy usługi bankowe. Zazwyczaj UML wykorzystywany jest wraz z jego reprezentacją graficzną gdzie jego elementy są reprezentowane przez symbole, które są używane do tworzenia diagramów. Dzięki temu stworzone schematy umożliwiają precyzyjne śledzenie procesów. Od wersji UML 2.5.1 wyróżnia się trzynaście głównych typów diagramów oraz trzy rodzaje interakcji: diagramy struktury, takie jak diagramy klas, obiektów, struktur złożonych, komponentów, pakietów oraz wdrożeń. Diagramy zachowania takie jak diagramy przypadków użycia, maszyn stanowych oraz czynności oraz diagramy interakcji, takie jak diagramy przeglądu interakcji, sekwencji, komunikacji i czasowe. UML nie jest narzędziem programowania wizualnego, lecz modele stworzone w tym języku, mogą być powiązane z różnymi językami programowania. Istotnym faktem jest to, że model stworzony w UML może być przekształcony, na przykład, w tabele relacyjnej bazy danych, a także na podstawie istniejącej implementacji można stworzyć model graficzny. Warto zaznaczyć, że poziom szczegółowości diagramów zależy od fazy projektowania oprogramowania. Ogólnego diagramu można użyć na etapie analizy problemu, jednakże w trakcie dokumentacji technicznej konieczne jest użycie bardziej szczegółowych diagramów.[[25]](#footnote-25)

W standardzie UML każdy obiekt reprezentowany jest przez prostokąt, który zawiera jego elementy składowe. Prostokąt służy do umieszczania nazw elementów, nazw z polami i metodami lub samych nazw z ich polami. Można oraz zalecane jest dołączenie informacji o typie, typie zwracanym lub argumentach do składników elementów. W dodatku, składniki klasy mogą być oznaczone różnymi modyfikacjami dostępu takimi jak: „+” oznaczający składnik publiczny, „#” oznaczający składnik chroniony, „-” oznaczający składnik prywatny oraz „~” oznaczający składnik dostępny w obrębie projektu (package).[[26]](#footnote-26)

Oznaczenia metod abstrakcyjnych w klasie abstrakcyjnej powinny być wyróżnione kursywą oraz podkreśleniem. Interfejsy są przedstawione podobnie jak klasy, lecz nazwy interfejsów muszą być poprzedzone słowem interface. Implementacja interfejsu w danej klasie jest oznaczona pustym białym grotem strzałki na końcu przerywanej linii, a klasa, które implementuje interfejs musi zaimplementować jego metody. Rozważając klasy abstrakcyjne, klasa, która dziedziczy musi implementować abstrakcyjne metody. Diagramy klas UML mają zaletę opisywania związków między klasami. Relację pomiędzy nimi mogą być określone z uwzględnieniem cech krotności, takich jak jeden obiekt, od zera do trzech obiektów, „\*” - dowolna ilość obiektów, „3 - \*” - od trzech do dowolnej ilości obiektów. Ważnym jest umieszczenie krotności po obu stronach zależności. Jeśli krotność nie została podana zakłada się, że wynosi jeden. Związki między klasami mogą być zależnościami, agregacjami częściowymi, asocjacjami, agregacjami całkowitymi i dziedziczeniem. Zależność informuje, że klasa musi korzystać z informacji o drugiej i zmiana w jednej klasie może wymagać zmiany w drugiej. Oznaczenia używane w trzech typach „<<call>> dla operacji wywołanych w jednej klasie przez drugą, „<<create>>” oznacza, że klasa A jest odpowiedzialna za tworzenie instancji klasy B, „<<instantiate>>” oznacza że, obiekt A jest reprezentacją klasy B, „<<use>>” oznacza, że klasa A wykorzystuje klasę B do realizacji swoich funkcjonalności. Zależność jest najbardziej podstawową relacją, w której jedna klasa czasowo wykorzystuje inną i wie o jej istnieniu. Zmiana w jednej klasie, może powodować wymóg zmiany w drugiej, lecz nie zawsze. Projektowanie diagramu z jak najmniejszą liczbą zależności ułatwia rozbudowę projektu.[[27]](#footnote-27)

Diagram klas obejmuje zbiór interfejsów, klas i kooperacji oraz powiązania między nimi. Diagram ten służy do opisywania struktury systemu, skupiając się głownie na relacjach między klasami z pominięciem innych szczegółów. Rozważając bardziej skomplikowane systemy, nie wszystkie elementy muszą być przedstawione na jednym diagramie. Kompletny model systemu jest efektem połączenia wszystkich diagramów, wraz z ich elementami oraz relacjami. Wybranie konkretnych klas do uwzględnienia na diagramie jest świadomym procesem, zależnym od analizy i projektowania systemu.[[28]](#footnote-28) Diagram przypadków użycia (z ang. use case diagram) jest narzędziem służącym do przedstawienia funkcjonalności systemu oraz jego relacji z otoczeniem, zawierając interakcje między użytkownikami korzystającymi z systemu oraz usług widocznych z zewnątrz. Pełni kluczową rolę w projektowaniu systemu składając się z różnych elementów. Opisuje funkcjonalności oraz otoczenie systemu. Przypadki użycia stanowią scenariusze związane z celami użytkowników, definiując oczekiwane zachowania systemu. Istotne są dla analizy systemu i służy jako podstawa do komunikacji między uczestnikami projektu. Poszczególny przypadek użycia posiada określone cechy: nazwa, opis, zależności i relacje, przepływ zdarzeń, wymagania specjalne oraz początkowe i końcowe, diagramy aktywności. Kluczowym elementem jest przepływ zdarzeń to znaczy sekwencja czynności potrzebnych do osiągnięcia funkcjonalności opisanej w danym przypadku użycia.[[29]](#footnote-29)

Podsumowując tworzenie modeli UML służy temu, by lepiej zrozumieć potrzeby klienta i przedstawić działanie programu w sposób zrozumiały dla wszystkich zaangażowanych. Diagramy umożliwiając opisanie zasada funkcjonowania aplikacji oraz prezentując aplikacje graficznie co pomaga w analizie systemu z różnych perspektyw. Modelowanie w języku UML pozwala projektowanie aplikacji już we wczesnych fazach tworzenia projektu, co umożliwia łatwe rozszerzenie jej o nowe funkcjonalności w przyszłości. Model UML jest przydatny przy większych projektach, pozwalając programistom na zrozumienie sposobu implementacji oraz działania systemu. Mimo iż że nie jest to język programowania graficznego, modele UML mogą być powiązane z różnymi językami programowania, co ułatwia przeniesienie projektu do konkretnej technologii, na przykład w formie tabel w bazie danych.[[30]](#footnote-30)

1. *Bazy danych NoSQL*, Wyższa Szkoła Bankowa w Gdańsku 2021, s.4 [↑](#footnote-ref-1)
2. Ibidem, s.3 [↑](#footnote-ref-2)
3. *Bazy danych NoSQL*, Wyższa Szkoła Bankowa w Gdańsku 2021, s.4-5 [↑](#footnote-ref-3)
4. *Interfejsy komunikacyjne baz danych,* Wyższa Szkoła Bankowa w Gdańsku 2021, s.5 [↑](#footnote-ref-4)
5. Dane za stroną internetową: <https://www.prisma.io/> (dostępne 07.04.2024r.). [↑](#footnote-ref-5)
6. Adeel Ehsan, Mohammed Ahmad M. E. Abuhaliqa, Cagatay Catal, Deepti Mishra, *RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions,*

   *Department of Computer Science & Engineering, Qatar University, Doha 2713, Qatar*

   *Software, Data and Digital Ecosystem Group, Educational Technology Laboratory, Department of Computer Science, Norwegian University of Science and Technology, 2815 Gjøvik, Norway, 2022r.* [↑](#footnote-ref-6)
7. Mariusz Śliwa, Beata Pańczyk, *Porównanie wydajności interfejsów programistycznych na przykładzie REST API, GraphQL i gRPC,* Department of Computer Science, Lublin, 2021, s. 360-361 [↑](#footnote-ref-7)
8. Urinboev Abdushukur Abdurakhimovich, *THE FUTURE OF JAVASCRIPT: EMERGING TRENDS AND TECHNOLOGIES*,Włochy, 2023, s. 12-14 [↑](#footnote-ref-8)
9. Nate Black, *Boris Cherny on TypeScript,* IEEE SOFTWARE, 2020r., s. 98-99 [↑](#footnote-ref-9)
10. Pawelec Kamil , Kopniak Piotr, *Porównanie szkieletów do wytwarzania aplikacji internetowych dla języka PHP,* Wydawnictwo Politechniki Lubelskiej, 2022r., s. 26-27 [↑](#footnote-ref-10)
11. Dr Charles R. Severance, *Python for Everybody. Exploring Data in Python* 3,2023r.,s. 9 [↑](#footnote-ref-11)
12. Robert Ryszard Paciorek, *Matematyka dla Ciekawych Świata,* 2023r., s. 1 [↑](#footnote-ref-12)
13. Tursunbek Sadriddinovich Jalolov, *PROGRAMMING LANGUAGES, THEIR TYPES AND BASICS, Uzbekistan, 2023r., s. 147* [↑](#footnote-ref-13)
14. Ibidem *s. 147* [↑](#footnote-ref-14)
15. Konrad Bielak, Bartłomiej Borek, Małgorzata Plechawska-Wójcik, *Analiza obciążeniowa aplikacji internetowych z użyciem szkieletów Angular, React i Vue,* Department of Computer Science, Lublin University of Technology, Lublin, 2022r. s. 77 [↑](#footnote-ref-15)
16. Ibidem, s. 79 [↑](#footnote-ref-16)
17. Konrad Bielak, Bartłomiej Borek, Małgorzata Plechawska-Wójcik, *Analiza obciążeniowa aplikacji internetowych z użyciem szkieletów Angular, React i Vue,* Department of Computer Science, Lublin University of Technology, Lublin, 2022r. s. 79 [↑](#footnote-ref-17)
18. Ibidem, s. 80 [↑](#footnote-ref-18)
19. Konrad Bielak, Bartłomiej Borek, Małgorzata Plechawska-Wójcik, *Analiza obciążeniowa aplikacji internetowych z użyciem szkieletów Angular, React i Vue,* Department of Computer Science, Lublin University of Technology, Lublin, 2022r. s. 80 [↑](#footnote-ref-19)
20. Ibidem, s. 80-82 [↑](#footnote-ref-20)
21. Ibidem, s. 82 [↑](#footnote-ref-21)
22. Jacek Matulewski, *Wersjonowanie i backup kodu*, Toruń, 2020r., s.1 [↑](#footnote-ref-22)
23. Ibidem, s. 1-2 [↑](#footnote-ref-23)
24. Ibidem, s. 2 [↑](#footnote-ref-24)
25. Konrad Szynalski, Dawid Różański, *Wprowadzenie do modelowania w języku UML,* Wrocławska Wyższa Szkoła Informatyki Stosowanej "Horyzont", Wrocław, 2022r., s. 31 [↑](#footnote-ref-25)
26. Ibidem, s. 32 [↑](#footnote-ref-26)
27. Ibidem, s. 32 [↑](#footnote-ref-27)
28. Ibidem, s.34 [↑](#footnote-ref-28)
29. Ibidem, s. 33-34 [↑](#footnote-ref-29)
30. Ibidem, s. 36 [↑](#footnote-ref-30)