**ROZDZIAŁ 1**

**PODSTAWY FUNCKJONOWANIA PRZEDSIĘBIORSTW ZAJMUJĄCYCH SIĘ SPRZEDAŻĄ ARTYKUŁÓW BIUROWYCH**

* 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)

1. *Bazy danych NoSQL*, Wyższa Szkoła Bankowa w Gdańsku 2021, s.4 [↑](#footnote-ref-1)
2. *Bazy danych NoSQL*, Wyższa Szkoła Bankowa w Gdańsku 2021, 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)