

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH

PRACA DYPLOMOWA
INŻYNIERSKA

Szablon pracy dyplomowej
inżynierskiej/magisterskiej, wersja 0.6

Engineering/master thesis template, version 0.6

AUTOR:

Imię Nazwisko

PROWADZĄCY PRACĘ:

tytuł, Imię Nazwisko, Jednostka

Opracował: Tomasz Kubik <tomasz.kubik@pwr.edu.pl>
Data: maj 2021



Tekst zawarty w niniejszym szablonie jest udostępniany na licencji Creative Commons: *Uznanie autorstwa – Użycie niekomercyjne – Na tych samych warunkach, 3.0 Polska*, Wrocław 2021. Oznacza to, że wszystkie przekazane treści można kopiować i wykorzystywać do celów niekomercyjnych, a także tworzyć na ich podstawie utwory zależne pod warunkiem podania autora i nazwy licencjodawcy oraz udzielania na utwory zależne takiej samej licencji. Tekst licencji jest dostępny pod adresem: <http://creativecommons.org/licenses/by-nc-sa/3.0/pl/>. Podczas redakcji pracy dyplomowej stronę tę można usunąć. Licencja dotyczy bowiem zredagowanego opisu, a nie samego latexowego szablonu. Latexowy szablon można wykorzystywać bez wzmiankowania o jego autorze.

Streszczenie

Streszczenie w języku polskim powinno zmieścić się na połowie strony (drugą połowę powinien zająć abstract w języku angielskim).

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Słowa kluczowe: raz, dwa, trzy, cztery

Abstract

Streszczenie in Polish should fit on the half of the page (the other half should be covered by the abstract in English).

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Keywords: one, two, three, four

Spis treści

Spis rysunków

Spis tabel

Spis listingów

Skrty

-

OGC (ang. *Open Geospatial Consortium*)
XML (ang. *eXtensible Markup Language*)
SOAP (ang. *Simple Object Access Protocol*)
WSDL (ang. *Web Services Description Language*)
UDDI (ang. *Universal Description Discovery and Integration*)
GIS (ang. *Geographical Information System*)
SDI (ang. *Spatial Data Infrastructure*)
ISO (ang. *International Standards Organization*)
WMS (ang. *Web Map Service*)
WFS (ang. *Web Feature Service*)
WPS (ang. *Web Processing Service*)
GML (ang. *Geography Markup Language*)
SRG (ang. *Seeded Region Growing*)
SOA (ang. *Service Oriented Architecture*)
IT (ang. *Information Technology*)

Rozdział 1

Wstęp

1.1. Geneza pracy

Usługi sieciowe, zarówno te dostępne publicznie jak i te realizowane dla celów prywatnych, pełnią kluczową rolę w kontekście funkcjonowania współczesnej sieci internetowej. Zapewne nikt z nas, nie jest w stanie wyobrazić sobie kształtu obecnego Internetu bez takich rozwiązań, sieciowych jak obsługa poczty elektronicznej, realizacja transferu plików, czy też przede wszystkim dostęp do aplikacji oraz witryn internetowych. Szczególnie w obrębie ostatniej spośród wymienionych usług, na przestrzeni ostatnich lat zauważyć można bardzo duży... liczbą zmian dotyczących sposobu ich definiowania oraz realizacji. Powodem pojawiania się tych zmian, jest niewątpliwie konieczność zachowania bezpieczeństwa oferowanych rozwiązań, uwzględniając coraz to większy ruch sieciowy, generowany przez nieustannie zwiększającą się... siłą liczbą użytkowników Internetu. Ponadto, od nowoczesnego systemu internetowego, wymaga się coraz to większego poziomu skalowalności, a także pełniejszego działania.

Poparciem niniejszych słów, może być treść wydawanego w kilkuletnich odstęgach czasu raportu firmy Cisco, dotyczącego przewidywań, oraz trendów sieciowych (tj. Cisco Annual Internet Report). Zgodnie z przedstawionymi w przytoczonym raporcie informacjami, a także porównując informacje te, z faktycznymi wartościami wskaźników dotyczących ruchu w internecie, zaobserwować możemy niemalże trzykrotny wzrost globalnego ruchu sieciowego na przestrzeni ostatnich pięciu lat. Ponadto, liczba klienckich urządzeń, sieciowych, wykorzystywanych w celu uzyskania dostępu do usług udostępnianych w Internecie, na przestrzeni analogicznego przedziału czasowego, zwiększyła się z wartości 2,4 urządzenia na osobę, do poziomu niemalże czterech hostów sieciowych przypadających na pojedynczego reprezentanta globalnej populacji.

Należy także zwrócić uwagę, jakiego typu ruch sieciowy pełni dominującą... rolę w kontekście dzisiejszego Internetu. Ponad 80% globalnego konsumenckiego ruchu internetowego stanowi... dane dotyczące usług wideo, około dziesięć procent światowego ruchu obejmują... pozostałe treści udostępniane w ramach aplikacji oraz witryn internetowych, a pozostałe 10% to ruch generowany m.in. przez usługi transferu plików, poczty elektronicznej, czy też gier online. Na podstawie tych informacji, zauważyć możemy, że ponad 90% całości danych, przesyłanych w ramach globalnej sieci, musi być przetwarzanych przez aplikacje internetowe, bądź usługi sieciowe z nimi powiązane. Dlatego też, zaawansowane witryny internetowe komunikujące się z usługami sieciowymi, zwane dziś systemami internetowymi, tworzone są z wykorzysta-

niem coraz to bardziej udoskonalonych modeli architektonicznych, pozwalających na coraz to łatwiejszą budowę i rozwój rozwiązań, przystosowanych do potrzeb aktualnego ruchu sieciowego [?].

Jednym z pierwszych, a także najbardziej podstawowych podejść do projektowania i implementacji systemów internetowych było wprowadzenie modelu architektury definiującego aplikacje monolityczne. W modelu tym, użytkownik aplikacji, wykorzystując oprogramowanie klienckie, którym w tym przypadku jest przeglądarka internetowa, wysyła żądanie uzyskania zasobu definiującego odpowiedni adres url ((ang. *Uniform Resource Locator*)). Łączenie to, odwołując się bezpośrednio do fizycznego zasobu zlokalizowanego na serwerze, który przed dostarczeniem do klienta było przetwarzane przez serwer w celu uzupełnienia go danymi uzyskanymi z zewnętrznych źródeł, m.in. z systemu bazodanowego. Odpowiednio przygotowana statyczna zawartość odpowiedzi serwera, przybierająca postać pliku HTML (ang. *HyperText Markup Language*) było następnie przesyłane bezpośrednio do przeglądarki internetowej. Podejście to, wywołując się całościowym brakiem dynamiki działania systemu internetowego, ponieważ każde zdarzenie wywoływane przez oprogramowanie klienta, wymagało zaadresowania i wygenerowania nowego łączenia w kierunku serwera, którego odpowiedzią było, a nowa zawartość warstwy prezentacyjnej systemu.

W związku z zauważeniem pewnej regularności dotyczącej funkcjonowania większości systemów internetowych, związanej z faktem niejednokrotnego generowania nieznacznie różniących się odpowiedzi serwera, a także w związku z rozwojem języka skryptowego JavaScript oraz technologii Flash, aplikacje w ramach architektury monolitycznej zaczęły uwzględniać obsługę łańcuchów, zawierających przetworzone fragmenty warstwy prezentacyjnej. Ponadto, możliwa stała się dynamiczna podmiana określonych fragmentów treści, bez konieczności ponownego pozyskiwania pozostałej zawartości widoku. Usprawnienie to, opierające się na technice realizacji łańcuchów, asynchronicznych w ramach JavaScript (ang. *AJAX - Asynchronous JavaScript and XML*) pozwoliło na poprawę wydajności działania aplikacji internetowych przyczyniając się do zmniejszenia kosztów generowania zapytań, a także redukcji rozmiaru pojedynczej odpowiedzi serwera. Rozwiązanie to, nie było jednak bezpośrednio na strukturę systemu, którego głównymi mankamentami były: pojedynczy centralny punkt przetwarzania łańcuchów, a także brak separacji logiki działania systemu od warstwy prezentacyjnej.

Niedoskonałości omówionego powyżej modelu zostały zniwelowane poprzez wprowadzenie architektury zorientowanej na serwisy (ang. *SOA - Service Oriented Architecture*). W podejściu tym, dokonano separacji warstwy prezentacyjnej systemu, a także wszystkich pozostałych funkcjonalności dotyczących logiki biznesowej oraz przetwarzania danych. Reużywalne oraz autonomiczne usługi sieciowe pozwalały na realizację określonych funkcji systemu, a sposób komunikacji klienta z usługą, jak i komunikacji pomiędzy poszczególnymi serwisami definiowany było, przez standaryzowane kontrakty. Zdefiniowanie architektury zorientowanej na serwisy umożliwiło budowę skalowalnych systemów internetowych, których poszczególne części mogły być realizowane w dowolnej technologii, a implementacja nowej funkcjonalności nie wymagała przebudowy pozostałych komponentów. Rozwiązanie to, wprowadzało jednak dodatkowy narzut dla każdej z przesyłanych wiadomości, wynikający ze zwiększonej skomplikowanej struktury łańcuchów, tworzonej z wykorzystaniem języka XML (ang. *Extensible Markup Language*). Ponadto, wraz ze wzrostem poziomu zaawansowania systemu internetowego, autonomizacja oraz reużywalność poszczególnych komponentów malała, ze względu na powstawanie specyficznych dla określonego rozwiązania zależności [?].

W związku z coraz to większymi wymaganiami dotyczącymi aplikacji internetowych, dominująca obecnie architektura rozproszonych usług sieciowych zastąpiona została, a poprzez model uwzględniający warstwy klienckie oraz interfejs programowania aplikacji (*ang. Application Programming Interface*). W przypadku nowoczesnych systemów internetowych, oba z tych komponentów budowane są w oparciu o architekturę n-warstwową (*ang. N-Tier Architecture Application*). W ramach niniejszego modelu, klient wysyła dane do interfejsu API, który na początku przetwarza jego treść, a następnie wywołuje usługę utworzoną w celu realizacji określonego zadania. Celem serwisu jest przetworzenie logiki biznesowej dla danej funkcjonalności, a także odwołanie się do usług dostępu do danych w celu ich uzyskania z zewnętrznego źródła informacji. Odpowiednio przygotowana odpowiedź jest następnie przekazywana do warstwy obsługi danych, która zwraca ją określönemu klientowi. W przeciwieństwie do pierwszego z przytoczonych modeli, odpowiedź API nie jest dokument HTML, a jedynie dane dotyczące zasobu, które chce uzyskać klient. Sam zasób natomiast, nie jest elementem warstwy prezentacji systemu a zbiorem danych lub typem operacji, które można na tym zbiorze wykonać. Upraszczając, stwierdzić można, że API pełni rolę pośrednika pomiędzy warstwą prezentacji a zbiorem danych oraz operacji ich przetwarzania, a także dostarczania. Poszczególne usługi realizujące logikę biznesową aplikacji zawarte są bezpośrednio wewnątrz API, co nie oznacza jednak, że nie mogą odwoływać się do serwisów zewnętrznych. Takie podejście do budowania systemów internetowych zapewnia zarówno skalowalność poszczególnych aplikacji wchodzących w skład systemu, jak i rozwiązuje problemy architektury SOA związane z zależnościami występującymi pomiędzy usługami. Dlatego też, architektura ta jest powszechnie wykorzystywana w celu budowy i zarządzania nowoczesnymi oraz zaawansowanymi systemami internetowymi [?].

Zarówno zdecentralizowana architektura zorientowana na serwisy, jak i centralna architektura oparta o interfejs programowania aplikacji, w przeciwieństwie do architektury monolitycznej, dostarcza zdecydowanie większą możliwość związanych z ewaluacją działania poszczególnych komponentów systemu. Dzięki powstaniu ostatnich dwóch, spośród trzech przedstawionych modeli architektonicznych, możliwe jest nie tylko zbudowanie efektywnie działającej aplikacji internetowej, ale także ciężej, a ocena poprawności implementacji jej komponentów, w celu ustawicznego doskonalenia całego systemu.

Niniejsza praca, traktująca bardziej o ewaluacji efektywności działania interfejsów programowania aplikacji, w kontekście jednych z dwóch najpopularniejszych środowisk rozwoju oraz uruchamiania api. Ponadto, porównane zostaną parametry wydajnościowe w kontekście określonych przypadków użycia interfejsu API, bardziej tego niezbadanego powszechnie wykorzystywanej architektury systemów internetowych.

1.2. Cel i zakres pracy

Celem pracy jest porównanie wydajności działania interfejsów programowania aplikacji, tworzonych z wykorzystaniem języków programowania C# oraz JavaScript. Interfejsy, wykonywane są w dwóch różnych środowiskach uruchomieniowych. Dla języka C#, środowiskiem tym jest platforma .NET, natomiast dla języka JavaScript – platforma NodeJS. Analiza porównawcza, obejmuje zarówno aspekty dotyczące efektywności działania samego interfejsu programowania aplikacji, jak i elementów wchodzących w skład tworzonego systemu. Wśród omawianych rozwiązań, wyróżniamy następujące:

mappery obiektowo-relacyjne, systemy bazodanowe, czy teŹ mechanizmy zarzÄdzania pamieciÄ... podrzÄcznÄ... Niektóre spoŹródŹ wymienionych moduŹów stanowiÄ... integralnÄ... czÄŁÄ API, natomiast pozostaŹe sŹuŹÄ... do rozszerzenia jego funkcjonalnoŹci.

Zakres pracy obejmuje: przeglÄd literaturowy, implementacjÄ Źrodowiska badawczego, realizacjÄ badaŹ, oraz opracowanie wynikÄw. PrzeglÄd literatury tyczy siÄ aspektÄw zwiÄzanych ze strukturÄ... i zasadÄ... dziaŹania interfejsÄw programowania aplikacji, a takŹe kwestii dotyczÄcych wykonywania pomiarÄw wydajnoŹci dla poszczegŹlnych operacji sieciowych. Operacje sieciowe, realizowane sÄ... w ramach obsŹugi ŹÄ... dania przez API. Etap implementacji Źrodowisk badawczych skŹada siÄ z budowy interfejsÄw w oparciu o porÄwnywane Źrodowiska rozwoju i uruchamiania aplikacji, a takŹe konfiguracji platformy lokalnej oraz platform chmurowych, pozwalajÄcych na przeprowadzanie analizy dziaŹania systemÄw. Realizacja badaŹ, przeprowadzona zostaŹa, a pod kÄtem pomiaru czasu odpowiedzi na ŹÄ... dania uŹytkownika koŹcowego biorÄc pod uwagÄ aspekty: wywoŹania serii ŹÄ... daŹ, obsŹugi wspŹŹliwÄce procesÄw, dostÄpnoŹci zasobÄw platformy hostingowej, a takŹe moŹliwoŹci oferowanych przez mappery obiektowo-relacyjne oraz systemy bazodanowe. Celem etapu opracowania wynikÄw jest przedstawienie, wizualizacja oraz analiza rŹŹnic wartoŹci czasÄw odpowiedzi interfejsÄw API na poszczegŹlne ŹÄ... dania, w odniesieniu do przeprowadzonych badaŹ,. Zastosowanymi kryteriami oceny podczas przeprowadzanej analizy jest czas odpowiedzi interfejsu programowania aplikacji dla wygenerowanego ŹÄ... dania, a takŹe maksymalna liczba ŹÄ... daŹ, jakie jest w stanie obsŹuŹyÄ okreŹlone API. Przedstawione kryteria, uwzglÄdniane zostaŹy w odniesieniu do wykorzystywanego Źrodowiska uruchomieniowego oraz technologii implementacyjnej. Przeprowadzone badania, majÄ... sŹuŹyÄ wskazaniu zarŹwno pozytywnych aspektÄw, jak i problemÄw dotyczÄcych wydajnoŹci dziaŹania aplikacji tworzonych z wykorzystaniem porÄwnywanych technologii. Ponadto, celem jest takŹe przedstawienie moŹliwoŹci zwiÄkszenia efektywnoŹci implementowanych interfejsÄw programowania aplikacji.

1.3. Struktura pracy

Niniejsza praca, podzielona zostaŹa, na szeŹÄ rozdziaŹów.

Pierwszy z nich, napisany zostaŹ, w celu zobrazowania dziedziny rozwaŹanego problemu, a takŹe podkreŹlenia jego wagi w kontekŹcie zagadnienia usŹug sieciowych. Ponadto, w rozdziale tym zdefiniowano cel popeŹnionej pracy oraz przedstawiono zakres czynnoŹci realizowanych w ramach przeprowadzonych badaŹ,.

W rozdziale drugim dokonano wprowadzenia teoretycznego do tematyki interfejsÄw programowania aplikacji oraz testowania usŹug sieciowych. Wprowadzenie to, w odniesieniu do interfejsÄw API dotyczy zarŹwno struktury i zasady dziaŹania omawianej usŹugi sieciowej, jak i sposobu realizacji poŹÄ... czÄ, tej usŹugi z zewnÄtrznymi ŹródŹami danych. W ramach wprowadzenia do tematyki testowania usŹug sieciowych wyŹŹniono fundamentalne pojÄcia teorii testowania oraz omŹwiono dostÄpne modele realizacji testÄw. Co wiÄcej, nakreŹlono strategiÄ wykonywania pomiarÄw wydajnoŹci w kontekŹcie usŹug pracujÄcych w sieciach komputerowych. W niniejszym rozdziale, zawarto rŹwnieŹ przeglÄd pozycji literaturowych, pomocnych w trakcie realizacji badaŹ, a takŹe przeglÄd technologii informatycznych, zastosowanych w procesie implementacji Źrodowiska badawczego oraz wykonania pomiarÄw.

W ramach trzeciego z rozdziałów, zdefiniowano i omówiono kałŁdy z aspektów rozwałŁanego problemu badawczego. DziÄ™ki temu, mołŁliwe stałŁo siÄ™ sformułŁowanie zbioru rozwałŁanych scenariuszy badawczych.

W celu realizacji badałŁ, opartych o zdefiniowane w rozdziale trzecim scenariusze badawcze, nalełŁy zaprojektowaÄ™ oraz zaimplementowaÄ™ odpowiednio dostosowane łŁrodowisko badałŁ. Poszczególne kroki realizacji tego łŁrodowiska, zarŁwno te, dotyczÄ™ce jego fizycznej struktury, jak i te, ktŁre tyczÄ™ siÄ™ implementacji interfejsów programowania aplikacji, opisane zostałŁy w rozdziale czwartym niniejszej pracy.

PiÄ™ty z rozdziałów, ma na celu przedstawienie rezultatów wynikajÄ™cych z przeprowadzonych prac naukowych. Rezultaty te, w obrÄ™bie niniejszego rozdziału, zostałŁy zgrupowane wzglÄ™dem zdefiniowanych uprzednio scenariuszy badawczych, realizowanych w odpowiednio przystosowanym łŁrodowisku. Ponadto, dla uzyskanych wartołŁci pomiarowych, dotyczÄ™cych kryteriów poszczegłŁnych badałŁ, wykonano testy parametryczne, dziÄ™ki ktŁrym mołŁliwa jest ocena istotnołŁci statystycznej zaobserwowanych rŁłŁnic wynikowych. Co wiÄ™cej, wyniki kałŁdego z realizowanych scenariuszy badawczych podane zostałŁy krytycznej analizie.

Ostatni z rozdziałów pełŁni rolÄ™ podsumowania. Autor przedstawia w nim uzyskane efekty wykonanej pracy, a takłŁe nakrełŁa mołŁliwołŁci zwiÄ™zane z dalszym rozwojem badałŁ.

Rozdział 2

Wprowadzenie teoretyczne

2.1. Wykorzystywane terminy

W niniejszej pracy, posłużyono się terminologią... dystynktywną... z punktu widzenia realizacji, rozwoju oraz ewaluacji usług sieciowych. Najbardziej istotne spośród wykorzystywanych terminów wymieniono poniżej. Dla każdego z pojęć, przedstawiono obcojęzyczne tłumaczenie, a także zdefiniowano spójny oraz zwięzły opis.

Usługa sieciowa

Web Service

Rodzaj systemu informatycznego cechujący się permanentnym wykonywaniem zdefiniowanych funkcji, tuż po uzyskaniu zlecenia. Zlecenie to, przybiera postać danych, przekazanych w ramach systematycznej struktury. Sposób dostarczenia zlecenia, jego format, a także metoda odpowiedzi na zlecenie, definiowane są... poprzez protokół, sieciowy z którego korzysta dana usługa.

Interfejs Programowania Aplikacji

Application Programming Interface

Zbiór reguł, oraz struktur programistycznych określający metod oraz cel interakcji pomiędzy komponentami oprogramowania. Pojęcie interfejsu programowania aplikacji jest niezależne od warstwy implementacji systemu i dotyczy się dowolnego rodzaju programu komputerowego. Interfejs API definiowany jest na poziomie kodu źródłowego poszczególnych fragmentów oprogramowania, a jego zadaniem jest dostarczenie wymaganych specyfikacji struktur programistycznych, a także protokołu, pozwalającego na ich wykorzystanie przez zewnętrzny komponent programowy.

API wykonane w technologii REST

RESTful API

Interfejs programowania aplikacji, bazujący zarówno w swojej strukturze, jak i funkcjonalności na zbiorze ściśle określonych reguł, dostarczanych w ramach metodologii REST. Reguły metodologii tej, implementowane są... najczęściej w stosunku do interfejsów programowania aplikacji, które wykorzystują... w kontekście zewnętrznej komunikacji protokołów, hipertekstowy. Podstawowa charakterystyka interfejsu programowania aplikacji

opartego o zbiór reguł, REST definiowana jest poprzez bezstanowość transmisji danych, pojęcia zasobów i reprezentacji, a także cech jednolitości interfejsu komunikacyjnego.

Kontroler

Controller

Struktura programistyczna (w przypadku języków programowania opartych o paradygmat obiektowy - klasa), której celem jest obsługa... dania dostarczonego od aplikacji klienckiej, zweryfikowania zgodności jego treści, a następnie przekierowanie wykonania programu do odpowiedniej metody warstwy logiki biznesowej. Po zakończeniu wszystkich operacji dotyczących omawianego zapytania, metoda dostępna w ramach klasy kontrolera formułuje odpowiedź zwracaną bezpośrednio do konsumenta... dania.

Serwis

Service

Struktura programistyczna (w przypadku języków programowania opartych o paradygmat obiektowy - klasa), której zadaniem jest realizacja obliczeń, związanych z określonym obiektem, lub domeną..., w ramach której określony obiekt się znajduje. Klasy serwisów (nazywane również klasami logiki biznesowej), stanowią... centralny punkt przetwarzania w ramach interfejsów programowania aplikacji, a także pełni rolę pośrednika pomiędzy komponentami odpowiedzialnymi za zarządzanie... danie (tj. metodami kontrolerów) oraz pozyskiwanie danych z określonych źródeł, (tj. metodami repozytoriów).

Repozytorium

Repository

Struktura programistyczna (w przypadku języków programowania opartych o paradygmat obiektowy - klasa), której rolą... jest wykonywanie operacji na obiektach modelu danych komunikujących się bezpośrednio z wykorzystywanym systemem bazodanowym. Operacje zawarte wewnątrz metod klas repozytoriów mogą... mieć postać kwerend lub komend definiowanych w języku zapytań, dostarczanych przez serwer bazodanowy, mogą... operować na dostępnej w pamięci aplikacji strukturze danych, bądź... teść odwoływać się do zewnętrznego zbioru bazodanowego, manipulując instancjami klas zdefiniowanego wewnątrz aplikacji modelu. W ostatnim z omawianych przypadków, aktualizacje wartości w ramach encji modelu danych są... identyfikowane przez maper obiektowo-relacyjny, który generuje polecenia języka zapytań, systemu bazodanowego, mające na celu synchronizację stanu danych aplikacji oraz zewnętrznego źródła informacji. Metody klas repozytoriów są... wywoływane wewnątrz metod logiki biznesowej.

Mapper obiektowo-relacyjny

Object-relational mapper

Oprogramowanie, którego głównym zadaniem jest konwersja struktury klas modelu danych do fizycznej organizacji tabel w ramach systemu bazodanowego. Ponadto, mapper obiektowo-

relacyjny dostarcza zbiór włączonych oraz metod stanowiących fasadę™ dla niskopoziomowych procedur dostępu do bazy danych, a także modyfikacji danych w niej zawartych.

Pamięć podręczna

Cache

Wydzielony fragment pamięci cechujący się szybkim czasem dostępu, wysoką przepustowością transmisji, a także ograniczonym okresem trwania przechowywania danych. Pamięć ta, w kontekście webowego interfejsu programowania aplikacji, wykorzystywana jest w celu przechowywania wyników czysto realizowanych operacji, a także magazynowania uprzednio dostarczonych do klienta fragmentów odpowiedzi na zapytania.

Przetwarzanie współbieżne

Concurrent Computing

Technika programistyczna oparta o wykorzystanie wielu współistniejących procesów oraz wątków, dostępnych w obrębie jednej aplikacji, a także odwołujących się do współdzielonych struktur danych. Poszczególne wątki stanowią elementy składowe pojedynczego procesu i są uruchamiane na tej samej centralnej jednostce przetwarzania. Procesor wykonuje operacje przełączania pomiędzy kontekstami poszczególnych wątków, dzięki czemu uzyska możliwość wykonania wielu spośród tych elementów w sposób równoległy. Z punktu widzenia interfejsu programowania aplikacji, który implementuje technikę przetwarzania współbieżnego, wyróżniamy punkty końcowe, które definiowane są jako osobne wątki aplikacji. Dlatego też, interfejs programowania aplikacji cechuje się dostępnością mimo jednoczesnego przetwarzania wielu, niekiedy długotrwałych zapytań, klientów.

Algorytm metaheurystyczny

Metaheuristic algorithm

Technika projektowania algorytmów nie zapewniających gwarancji uzyskania optimum dla rozważanego problemu, jednakże pozwalająca na zbudowanie systemu, dostarczającego rozwiązanie złozonego zagadnienia w akceptowalnym czasie, a także uzyskiwanego przy wykorzystaniu akceptowalnej ilości zasobów sprzętowych. Algorytm metaheurystyczny, poza konwencjonalnymi regułami stosowanymi w ramach standardowych wzorców programowania, implementuje reguły rozwiązywania problemów oparte na losowości, będące teoretycznie wywnioskowane na podstawie zjawisk fizycznych.

Punkt końcowy interfejsu programowania aplikacji

API Endpoint

Punkt końcowy usługi sieci web definiuje jeden z końców kanału komunikacyjnego pomiędzy aplikacją klienta a serwerową. W momencie interakcji interfejsu programowania aplikacji z odrębnym systemem, punkt styku dwóch usług sieciowych w ramach omawianej interakcji nazywany jest punktem końcowym. W odniesieniu do wewnętrznej struktury interfejsu programowania aplikacji, punkt końcowy wywołuje zwołanie z nim metod klasy kontrolera, a samo powołanie identyfikowane jest (w przypadku

usŁ,ugi sieciowej bazujÄ...cej na protokole hipertekstowym i metodologii REST) poprzez nazwÄ™ zasobu, rodzaj metody, a takŁe parametry ŁÄ...dania.

Ł»Ä...danie realizowane w ramach usŁ,ugi protokoŁ,u hipertekstowego

HTTP Request

Struktura danych, wysyŁ,ana od aplikacji klienckiej (tj. aplikacji internetowej, przeglÄ...darki, czy teŁ programu klienta HTTP) w kierunku usŁ,ugi sieciowej. Ł»Ä...danie protokoŁ,u hipertekstowego charakteryzuje siÄ™ jednoznacznie zdefiniowanÄ... strukturÄ..., uwzglÄ™dniajÄ...cÄ... m.in. unikalny identyfikator zasobu, listÄ™ zdefiniowanych nagŁ,ÄłwkÄłw, ciaŁ,o ŁÄ...dania oraz jednÄ... z dziewiÄ™ciu dopuszczalnych metod HTTP.

OdpowiedŁs usŁ,ugi protokoŁ,u hipertekstowego

HTTP Response

Struktura danych, wysyŁ,ana przez usŁ,ugÄ™ sieciowÄ... w kierunku aplikacji klienckiej. OdpowiedŁs HTTP, ma na celu poinformowanie klienta serwisu webowego o statusie realizacji, wysŁ,anego przez niego uprzednio ŁÄ...dania. Podstawowymi elementami odpowiedzi usŁ,ugi protokoŁ,u hipertekstowego sÄ...: ciaŁ,o odpowiedzi (zdefiniowane najczÄ™Łciej z wykorzystaniem notacji JSON lub jÄ™zyka XML), kod odpowiedzi (liczba determinujÄ...ca stan wykonania ŁÄ...dania), a takŁe zbiÄłr informacji nagŁ,Äłwkowych dotyczÄ...cych typu danych zawartych w odpowiedzi, czy teŁ fizycznych informacji o serwerze usŁ,ugi sieciowej.

Kod odpowiedzi usŁ,ugi protokoŁ,u hipertekstowego

HTTP Response Code

Liczba determinujÄ...ca status realizacji ŁÄ...dania wysŁ,anego przez aplikacjÄ™ klienckÄ... Kod odpowiedzi stanowi jednÄ... z wymaganych skŁ,adowych dotyczÄ...cych standardowego rezultatu zwracanego w ramach usŁ,ugi opartej o protokół, hipertekstowy. WyróżniÄłmoŁemy piÄ™ kategorii kodów odpowiedzi, niosÄ...cych ze sobÄ... odmiennÄ... informacje. Kategoriami tymi sÄ...: kody informacyjnej odpowiedzi (100-199), kody poprawnej odpowiedzi (200-299), kody wiadomoŁci o przekierowaniu (300-399), kody bł,Ä™du aplikacji klienckiej (400-499), oraz kody bł,Ä™du aplikacji serwerowej (500-599).

Czas odpowiedzi usŁ,ugi protokoŁ,u hipertekstowego

HTTP Response Time

WyraŁony w milisekundach, przedziaŁ, czasu od momentu otrzymania ŁÄ...dania wygenerowanego przez aplikacjÄ™ klienckÄ..., do chwili zwrócenia rezultatu wykonywanych przez usŁ,ugÄ™ sieciowÄ... obliczeŁ,. Liczba ta, stanowi jednÄ... z wartoŁci pomiarowych, w kontekście efektywnoŁci dział,ania interfejsu programowania aplikacji.

Obiektowa notacja JavaScript

JavaScript Object Notation - JSON

Format definicji, reprezentacji, a także wymiany danych w postaci obiektów niezależny od określonego języka programowania. Obiektowa notacja JavaScript jest powszechnie wykorzystywana jako format komunikatów przekazywanych pomiędzy interfejsami programowania aplikacji a systemami klienckimi. W odróżnieniu od języka reprezentacji danych opartego o znaczniki *Extensible Markup Language - XML*, obiektowa notacja JavaScript cechuje się mniejszym rozmiarem przesyłanych obiektów (poprzez redukcję liczby metadanych), jednolitym standardem niezależnym od technologii, a także brakiem przechowywania informacji o typie poszczególnej wartości zadanego obiektu.

Testy wzorcowe

Benchmark

Rodzaj ewaluacji oprogramowania, której zadaniem jest określenie referencyjnego poziomu wydajności dla testowanego systemu. Metryki, uzyskane w ramach testów wzorcowych, mogą zostać wykorzystane jako wartości ograniczeń, względem testów obciążeniowych oraz przeciwnych.

Testy dymne

Smoke testing

Metoda testowania oprogramowania, której celem jest sprawdzenie poprawności funkcjonowania poszczególnych elementów systemu. Testy dymne, wykonywane są przed testami wydajnościowymi, po to aby upewnić się co do braku błędów implementacyjnych w ramach analizowanego oprogramowania.

Testy wydajności podstawowej

Baseline performance testing

Metoda ewaluacji oprogramowania, pozwalająca na weryfikację działania systemu w warunkach analogicznych do realiów standardowego działania. Na podstawie testów wydajności podstawowej, określonych wartości metryk, które będą miały zastosowanie jako punkt odniesienia dla kolejnych rodzajów testów. Ponadto, wykorzystując standard pomiaru wydajności aplikacji internetowych (taki jak np. APDEX), wartości uzyskane w ramach ewaluacji podstawowych, mogą posłużyć w celu określania punktów satysfakcji, tolerancji oraz frustracji.

Testy obciążeniowe

Load testing

Rodzaj testów, które mają na celu określenie maksymalnego poziomu natężenia operacji, jakie mogą być generowane w kierunku oprogramowania. W kontekście niniejszej pracy, operacjami tymi są dane wysyłane do interfejsu programowania aplikacji. Kluczowym aspektem testu obciążeniowego jest zdefiniowanie progu obciążenia aplikacji, powyżej którego system jest nie w stanie generować poprawnych odpowiedzi w akceptowalnym czasie.

Testy przeciŁŁeniowe

Stress testing

Metoda ewaluacji oprogramowania, w ramach ktŁrej natŁŁenie operacji generowanych w kierunku testowanego oprogramowania zwiŁkszone jest ponad ustalony prŁŁg tolerancji. Celem testu przeciŁŁeniowego jest obserwacja sposobu dziaŁ,ania systemu, w momencie, w ktŁrym nie jest on w stanie przetwarzaŁ otrzymywanych ŁŁŁ...daŁ,, w sposŁb poprawny.

Asercja

Assertion

WyraŁŁenie typu prawda/faŁ,sz, zdefiniowane w dowolnym miejscu programu, ktŁre przyjmuje wartoŁŁ prawdziwŁ... w momencie speŁ,nienia hipotezy zawartej w ramach okreŁłonego przypadku testowego. Praktyczne podejŁcie do procesu testowania funkcjonalnoŁci oprogramowania, sprowadza siŁ do definiowania hipotez oraz ciŁŁgŁw operacji w kontekŁcie przypadkŁw testowych, a nastŁpnie weryfikacji tych hipotez z wykorzystaniem asercji.

2.2. Interfejsy programowania aplikacji

Webowy interfejs programowania aplikacji to usŁ,uga sieciowa, ktŁrej celem jest realizacja zadaŁ,, zleonych przez oprogramowanie klienta. Zadania te, dotyczŁ... operacji wykonywanych w kontekŁcie okreŁłonych zasobŁw. WyrŁŁniŁ moŁŁemy operacje zwane zapytaniami (tj. dotyczŁ...ce pozyskiwania danych z ich ŹsrŁdeŁ), a takŁŁe komendami (tj. zwiŁ...zane z wykonywaniem operacji na danych).

Interfejsy API, budowane sŁ... z wykorzystaniem protokoŁ, u HTTP, dlatego teŁŁ w ich kontekŁcie moŁŁemy mŁwiŁ o komunikacji bezstanowej definiujŁ...cej pojŁcia ŁŁŁ...dania oraz odpowiedzi. W zwiŁ...zku z charakterystykŁ... protokoŁ, u hipertekstowego, zarŁwno ŁŁŁ...danie jak i odpowiedŁs cechuje siŁ regularnŁ... strukturŁ... zawierajŁ...cŁ... predefiniowane elementy.

ŁŁŁ...danie protokoŁ, u http wysyŁ,ane jest od aplikacji klienta do interfejsu API. PodstawowŁ... skŁ,adowŁ... tego polecenia stanowi unikalny identyfikator zasobu URI (*ang. Uniform Resource Identifier*), na podstawie ktŁrego moŁŁliwe jest okreŁłenie fragmentu dziedziny obsŁ,ugiwane go modelu danych. Informacja ta jednak, nie jest wystarczajŁ...ca w kontekŁcie realizacji jednej z funkcjonalnoŁci, zdefiniowanych w ramach API. ŁŁŁ...danie klienta, musi zostaŁ uzupeŁ,nione o jednŁ... z dziewiŁci u ustalonych metod http, obsŁ,ugiwane go wersjŁ protokoŁ, u, a takŁŁe zbiŁr linii nagŁ,Łwkowych. Opcjonalnie, informacja wysyŁ,ana w kierunku interfejsu, moŁŁe zostaŁ wzbogana o zawartoŁŁ tekstowŁ... okreŁłanŁ... ciaŁ,em ŁŁŁ...dania (*ang. Request body*). Taki zbiŁr informacji, pozwala na jednoznacznŁ... identyfikacje fragmentu kodu programu, ktŁry ma zostaŁ wykonany wewnŁ...trz interfejsu programowania aplikacji. W tabelach ?? oraz ?? przedstawiono kolejno listŁ zdefiniowanych metod protokoŁ, u hipertekstowego wraz z wyjaŁnieniem ich przeznaczenia, a takŁŁe zbiŁr najczŁŁiej wykorzystywanych linii nagŁ,Łwkowych, w kontekŁcie realizacji ŁŁŁ...daŁ,,.

Po wykonaniu kodu programu przypisanego do okreŁłonego rodzaju polecenia generowanego przez aplikacje klienckŁ... , z interfejsu programowania aplikacji zwracana jest odpowiedŁs na ŁŁŁ...danie (*ang. HTTP response*). Analogicznie do instrukcji realizacji danej czynnoŁci, takŁŁe odpowiedŁs dotyczŁ...ca statusu jej wykonania jest ustrukturyzowana zgodnie

Tab. 2.1: Zbiór dozwolonych metod protokołu hipertekstowego

Nazwa metody	Opis
GET	Pozyskanie danych dotyczących pojedynczej instancji określonego zasobu lub grupy instancji z opcjonalnym uwzględnieniem warunków kwalifikacji poszczególnej instancji do grupy.
POST	Definiowanie nowej instancji dotyczącej określonego typu zasobu. Przy zastosowaniu metody POST, wymagane jest zdefiniowanie ciała, a następnie, jako części składowej generowanej instrukcji.
PUT	Aktualizacja treści zawartości instancji występującej w ramach odwrotnej ścieżki do określonego zasobu. Przy zastosowaniu metody PUT, wymagane jest zdefiniowanie ciała, a następnie, jako części składowej generowanej instrukcji.
DELETE	Usunięcie istniejącej instancji dotyczącej określonego typu zasobu.
PATCH	Aktualizacja fragmentu zawartości instancji występującej w ramach odwrotnej ścieżki do określonego zasobu. Przy zastosowaniu metody PATCH, wymagane jest zdefiniowanie ciała, a następnie, jako części składowej generowanej instrukcji.
HEAD	Pozyskanie zbioru linii nagłówkowych, które byłyby dostarczone wraz z ciałem odpowiedzi w ramach odpowiedzi wykorzystującej metodę GET. Wygenerowanie odpowiedzi HEAD umożliwia określenie charakteru danych, przed ich ewentualnym pozyskaniem.
OPTIONS	Pozyskanie informacji dotyczących charakterystyki oraz struktury serwera. Definiując dane typu OPTIONS, klient może dowiedzieć się o dopuszczalnych metodach HTTP obsługiwanych przez serwer, czy też uzyskać informacje o nazwie serwera oraz wykorzystywanym systemie operacyjnym.
CONNECT	Ustanowienie dwukierunkowej komunikacji pomiędzy klientem a serwerem. W przypadku realizacji komunikacji szyfrowanej, dane typu CONNECT pozwala na zestawienie zabezpieczonego tunelu pomiędzy hostami.
TRACE	Wygenerowanie komunikatu diagnostycznego w ramach prośby zwrotnej, którego celem jest osiągnięcie kłopotliwego z hostów, biorących udział, w komunikacji.

Tab. 2.2: Zbiór najczęściej wykorzystywanych linii nagłówkowych w kontekście ŁŁ...dania protokołu, u hipertekstowego

Linia nagłówkowa	Znaczenie	Dopuszczalna zawartość
accept	Typ zawartości, którą jest w stanie przetwarzać aplikacja kliencka	Identyfikator typu MIME (<i>ang. Multipurpose Internet Mail Extensions</i>) lub zapis <i>*/*</i> oznaczający dowolną zawartość
accept-encoding	Sposób kodowania znaków, rozumiany przez stronę klienta	Zbiór formatów kodowania zdefiniowany w ramach rejestru formatów IANA
accept-language	Język naturalny, preferowany przez stronę kliencką...	Pojedyncza wartość reprezentująca określony kraj lub region, bądź lista niniejszych wartości wraz z parametrem istotności poszczególnego kodu lokalizacji
content-length	Długość ciała, a ŁŁ...dania wyrażona w bajtach	Liczba naturalna
content-type	Format zawartości ciała, a ŁŁ...dania	Identyfikator typu MIME wraz ze sposobem kodowania wiadomości
cookie	Zbiór informacji pozwalających na wprowadzenie oraz utrzymanie stanowego charakteru transmisji	Zestaw par klucz-wartość, gdzie klucz jest wartością tekstową, a wartość przyjmuje postać dowolną...
origin	Informacja determinująca pochodzenie ŁŁ...dania	Ciąg tekstowy składający się z nazwy protokołu, nazwy hosta oraz numeru portu
user-agent	Specyfikacja techniczna oprogramowania klienta	Ciąg znaków zawierający informacje o nazwie produktu, jego wersji, platformie sprzętowej, czy też systemie operacyjnym

z wytycznymi zawartymi w definicji protokołu, u hipertekstowego. W ramach rezultatu zwróconego przez API wyróżnionych należy: adres docelowy klienta, kod statusu, ciało odpowiedzi, a także zbiór linii nagłówkowych. Informacja zawarta w ramach kodu statusu, determinuje powodzenie realizowanej operacji, a treść dostarczanych linii nagłówkowych, może zostać wykorzystana w celu wnioskowania o charakterystyce odbywającej się komunikacji. Ciało odpowiedzi powinno zawierać dane dotyczące definiowanego w ramach identyfikatora dane zasobu, w przypadku dane, wykorzystujących metodę GET. W kontekście pozostałych danych, zgodnie z wytycznymi dokumentu RFC (*ang. Request For Comments*) o numerze 7230, powinno ono posiadać charakter informacji pomocniczej, bądź też pozostać puste [?]. W ramach tabel ?? oraz ??, wymienione zostały, kolejno: zbiór najczęściej zwracanych linii nagłówkowych w kontekście odpowiedzi na dane, a także przedziały liczbowe dla kodów statusu odpowiedzi, wraz z ich semantyką.

Tab. 2.3: Zbiór najczęściej zwracanych linii nagłówkowych w kontekście odpowiedzi protokołu, u hipertekstowego

Linia nagłówkowa	Znaczenie	Dopuszczalna zawartość
access-control-allow-credentials	Określenie, czy odpowiedź serwera ma być obsługiwana z kodu JavaScript aplikacji klienckiej, w momencie gdy nagłówek dane dotyczy poświadczającego, zezwala na ich dołączenie	Wartość prawda/fałsz
access-control-allow-origin	Informacja dotycząca pochodzenia klienta, który może ubiegać się o otrzymanie odpowiedzi od serwera	adres hosta klienckiego lub symbol gwiazdki oznaczający zezwolenie dla wszystkich hostów
cache-control	Dane konfiguracyjne dotyczące obsługi pamięci podręcznej	Zbiór par klucz-wartość określających zachowanie pamięci cache w kontekście określonej komunikacji
content-length	Długość ciała odpowiedzi wyrażona w bajtach	Liczba naturalna
content-type	Format zawartości ciała odpowiedzi	Identyfikator typu MIME wraz ze sposobem kodowania wiadomości
cross-origin-resource-policy	Polecenie ignorowania dane, realizowanych pomiędzy źródłami bądź witrynami w kontekście określonego zasobu	Wartość prawda/fałsz
expires	Data wygaśnięcia ważności niniejszej odpowiedzi	Określona data
server	Nazwa hosta dostarczającego odpowiedź klientowi	Ciąg znaków

Przedstawiony w niniejszy sposób interfejs programowania aplikacji scharakteryzować należy jako deterministyczny system wejściowo-wyjściowy. Ponadto, należy zauwa-

Tab. 2.4: Zbiór kodów statusu odpowiedzi protokołu, u hipertekstowego

Przedział, liczbowy	Semantyka w kontekście odpowiedzi
100 - 199	Zbiór kodów informacyjnych - ŁŁ...danie jest aktualnie przetwarzane
200 - 299	Zbiór kodów poprawnej odpowiedzi - wystosowane ŁŁ...danie zostało zrealizowane poprawnie
300 - 399	Zbiór kodów przekierowań, - istnieje możliwość wiele akceptowalnych odpowiedzi dla ŁŁ...dania b...dŁ realizacja określonej operacji wymusza odwołanie się pod adres identyfikujący...cy odmienny zasób
400 - 499	Zbiór kodów błędów po stronie klienta - wygenerowane ŁŁ...danie zawiera błąd, oczekiwany zasób nie istnieje, klient nie jest uwierzytelniony lub nie posiada określonego poziomu uprawnień,
500 - 599	Zbiór kodów błędów po stronie serwera - pomimo poprawnej struktury wygenerowanego ŁŁ...dania, serwer nie jest w stanie zrealizować przydzielonej mu operacji

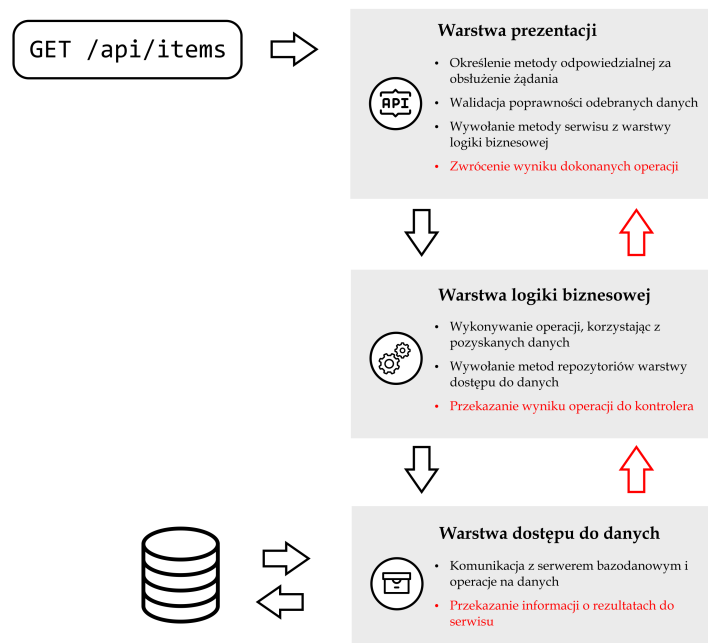
ŁŁy, ŁŁe w ramach systemu tego, występuje zjawisko inercji, powodowane koniecznością realizacji zdefiniowanego w ramach API kodu programu. Na podstawie tego założeń, ewaluacji działania oraz wydajności interfejsu programowania aplikacji przeprowadzić można poprzez wprowadzanie określonego wejścia (tj. generowanie ŁŁ...dania) oraz obserwacji wartości zwróconej na wyjściu (tj. uzyskana odpowiedź).

Proces przetwarzania ŁŁ...dania wewnątrz interfejsu API

Po uzyskaniu ŁŁ...dania otrzymanego od strony klienta, zadaniem interfejsu programowania aplikacji jest wybór określonej klasy kontrolera, a także zawartej w niej metody. Każda z klas kontrolerów stworzona jest w celu obsługi operacji związanych z konkretnym zasobem, a poszczególne metody tej klasy implementuje zachowanie które ma zostać wywołane w kontekście dostarczonego typu oraz identyfikatora polecenia.

Wewnątrz metody klasy warstwy kontrolerów, wywoływane zostają operacje zdefiniowane w usługach warstwy biznesowej. Usługi te, realizowane mogą być zarówno wewnątrz api jak i stanowi odrębny system internetowy. Klasy warstwy logiki biznesowej, zwane serwisami, złozone z metod, których głównym celem jest weryfikacja poprawności otrzymanych informacji w kontekście obsługiwanych zasobów, a także pozyskiwanie danych oraz wykonywanie operacji na nich, poprzez odwołanie się do metod warstwy dostępu do danych.

Zbiór klas warstwy dostępu do danych, stanowi ostatni z logicznych poziomów, definiowanych w ramach architektury API. Fragmenty kodu zdefiniowane w tej warstwie, zwane repozytoriami, mają za zadanie obsługi komunikacji pomiędzy interfejsem programowania aplikacji, a określonym źródłem danych. Ponadto, metody klas repozytoriów, dostarczają warstwie logiki biznesowej interfejs operacji na danych. Dzięki temu, ŁŁ...danie może być przetwarzane od warstw najwyższych (tj. warstwy kontrolerów) do warstwy najniższej (tj. warstwy dostępu do danych), natomiast odpowiedź na ŁŁ...danie jest konsolidowana w kierunku odwrotnym [?]. Na ilustracji ?? przedstawiono przepływ informacji wewnątrz interfejsu API, od momentu wygenerowania ŁŁ...dania do chwili uzyskania odpowiedzi.



Rys. 2.1: Proces przetwarzania danych wewnątrz interfejsu API

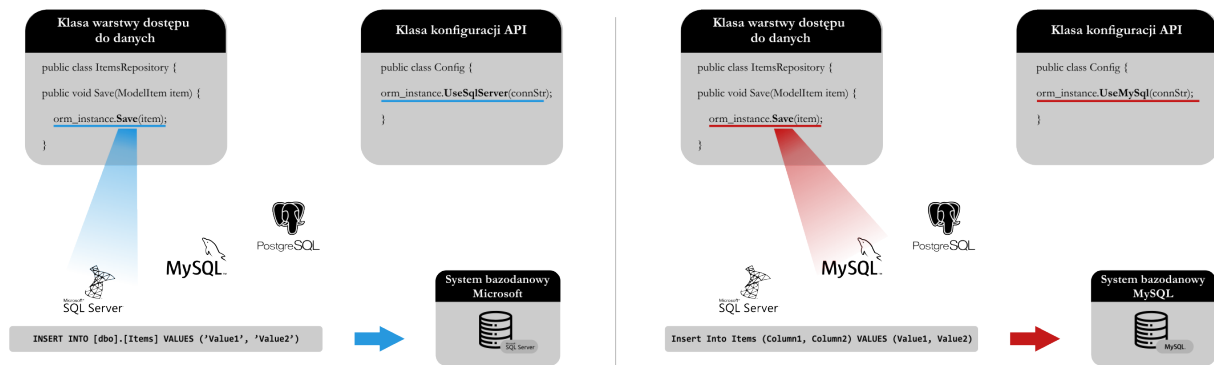
Konwersja obiektowo-relacyjna

W celu uproszczenia procesu pozyskiwania oraz modyfikacji danych z zewnętrznych źródeł, a także unifikacji sposobu interakcji z nimi, w ramach interfejsów programowania aplikacji, powszechnie wykorzystywane jest oprogramowanie zwane mapperem obiektowo-relacyjnym (ang. *Object-Relational Mapper*). Założeniem oprogramowania tego, jest zdefiniowanie warstwy abstrakcji pomiędzy interfejsem programowania aplikacji a językiem programowania będącym zbiorem poleceń, wykorzystywanym w ramach obsługi źródła danych.

Podstawowe składowe oprogramowania typu ORM to jednolity interfejs operacji na zbiorze danych, klasy kontekstu bazodanowego, a także metody obsługi komunikacji z bazą danych.

Dzięki wprowadzeniu jednolitego interfejsu operacji na danych, niezależnie od źródła informacji z jakim komunikuje się API, wydanie konkretnego polecenia do dowolnego systemu bazodanowego równoznaczne jest z całokształtem wywołaniem funkcji o takiej samej sygnaturze. Stosując takie podejście, konstruktor interfejsu programowania aplikacji nie staje się uzależniony od źródła danych z którym pracuje. Ponadto, istnieje możliwość zamiany lub polecenia dodatkowego systemu bazodanowego, a operacja ta, nie wpływa w jakikolwiek sposób na działanie interfejsu API. Niniejsza zależność została zilustrowana na rysunku ??

Dystynktywnym elementem oprogramowania mappera obiektowo-relacyjnego jest klasa kontekstu bazodanowego. Klasa ta, jest kontenerem struktur w ramach których wyróżniamy zbiory elementów modelu danych, a także konfigurację poszczególnych ich właściwości. Podstawową ideą omawianej konwersji dziedziny obiektowej do domeny relacyjnej jest zdefiniowanie zbioru klas, opisujących wykorzystywane zasoby, a następnie odwzorowanie ich w relacyjnym modelu danych, obsługiwanych przez wybrany system bazodanowy. Klasa kontekstu pozwala na określenie, które spośród struktur danych zdefiniowanych w ramach API powinny zostać rzutowane na obiekty tabel generowanych w obrębie bazy danych. Ponadto, dla właściwości każdego z klas modelu danych, zdefinio-



Rys. 2.2: Zasada działania oprogramowania mappera obiektowo-relacyjnego w kontekście jednolitego interfejsu operacji na zbiorze danych

Ważną rolę w konfiguracji gry, która zostanie przetransformowana do modelu relacyjnego. W zakresie klasy kontekstu bazy danych, opisywane są... także relacje, jakie mają... zostaną wygenerowane pomiędzy poszczególnymi elementami modelu.

W celu nawiazania, utrzymania, a także zakończenia komunikacji z zewnętrznym źródłem danych, oprogramowanie ORM wykorzystuje klasy zwane konektorami. Klasy te, dostarczają... przejrzysty interfejs obsługi połączenia, który następnie jest opakowywany w zuniifikowany interfejs, dostępny bezpośrednio dla twórcy API [?].

Uwierzytelnienie oraz autoryzacja

Proces uwierzytelnienia oraz autoryzacji użytkownika odwołuje się do interfejsu programowania aplikacji, przedstawiając rolę w trzech następujących krokach.

Pierwszym z nich, jest wygenerowanie danych odwołującego się do punktu końcowego odpowiedzialnego za obsługę uwierzytelnienia wewnątrz API. Następnie, musi posiadać ciągło, zawierające informacje powiadczające o konkretnym użytkowniku. Najczęściej, informacją... tą... jest nazwa użytkownika oraz hasło.

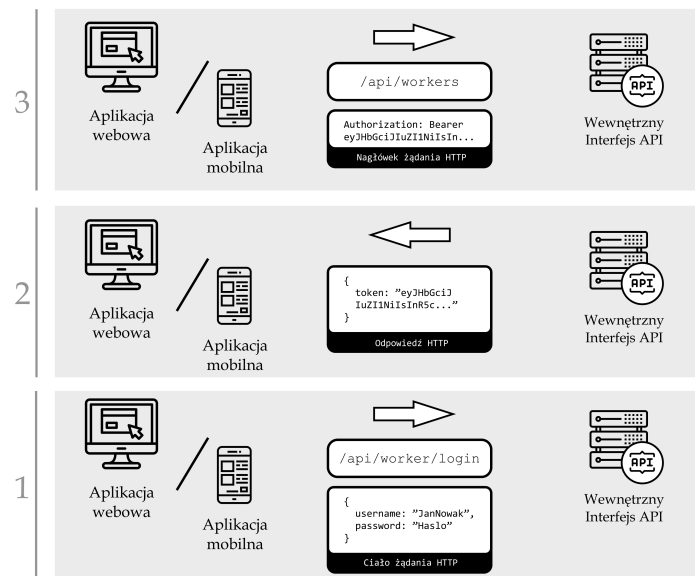
Następnie, dostarczone referencje są analizowane przez mechanizmy uwierzytelnienia implementowane w ramach API. W rezultacie tych operacji, zwrócona zostaje pozytywna odpowiedź zawierająca token autoryzujący bądź... jeśli negatywna, posiadająca... w sobie informację o błędzie uwierzytelnienia klienta.

Strona kliencka może dysponować operacjami przed interfejsem programowania aplikacji, uwzględniając w ramach linii nagłówkowej... danych token uwierzytelniający. Dostarczona w ten sposób informacja, pozwala na identyfikację użytkownika w ramach interfejsu API, a także na określenie przypisanego użytkownikowi poziomu uprawnień. W ramach struktury tokenu, zawarta jest także informacja o jego czasie ważności, dlatego też, procedura uwierzytelniania musi być regularnie ponawiana [?].

Na rysunku ??, zilustrowany został, proces uwierzytelnienia i autoryzacji aplikacji klienta przez interfejsem programowania aplikacji.

Separacja zapytań, oraz komend w kontekście odwołania, do źródła, danych

Wraz z rosnącą... liczbą... danych, obsługiwanych w ramach zaawansowanych interfejsów programowania aplikacji, zauważalne zostało zjawisko asymetrii w kontekście typów wiadomości generowanych przez klientów. Zapytania dotyczące pozyskiwania



Rys. 2.3: Proces uwierzytelnienia oraz autoryzacji uŁytkownika przed interfejsem API

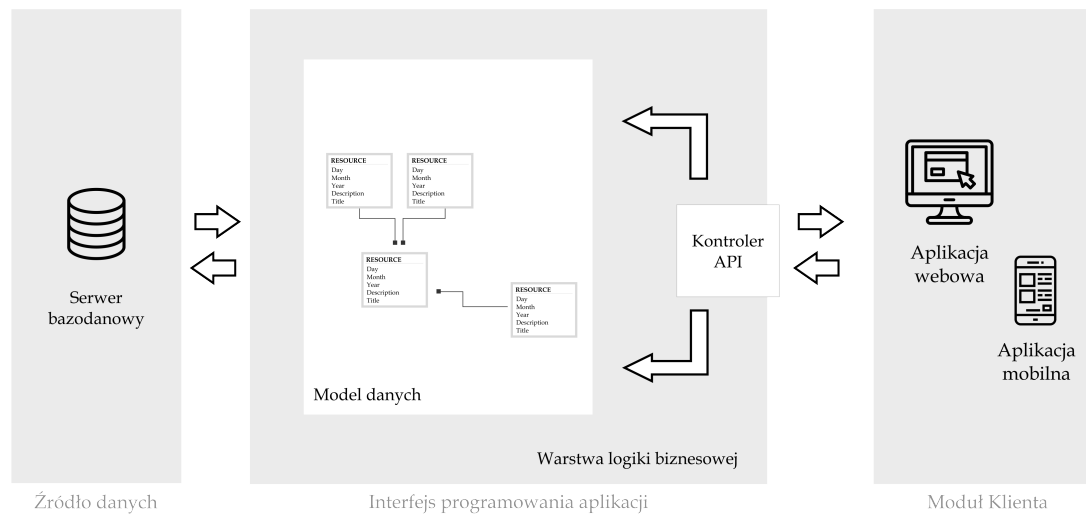
danych z API realizowane jest z nieporównywalnie większą... cząstotliwością... niż operacje ich modyfikacji. Dlatego też, zdefiniowany został, wzorec projektowy dotyczący separacji zapytań, oraz komend generowanych względem usługi sieciowej (*ang. Command Query Responsibility Segregation*).

Zastosowanie przedstawionego powyżej wzorca projektowego wiąże się z koniecznością budowy dwóch osobnych modeli danych. Pierwszy z nich, wykorzystywany jest w kontekście odczytu informacji. Na modelu tym, dokonywana jest najczęściej operacja optymalizacji, której celem jest redukcja rozmiaru składowych modelu, a także szybkości przetwarzania bardziej złożonych struktur danych jego części. Drugi z modeli danych, znajduje zastosowanie w aspekcie modyfikacji określonych zasobów. Biorąc pod uwagę standardowy sposób eksploatacji interfejsu programowania aplikacji, model ten cechować się może niższą wydajnością. W zależności od specyfiki określonej usługi sieciowej, optymalizowany może być albo model odczytu, albo też model zapisu. Ponadto, implementując wzorec projektowy separacji zapytań, oraz komend, wykorzystano dwa osobne zewnętrzne źródła danych, przystosowane do wydajniejszego wykonywania określonego typu operacji, będące też dostosowane do obsługi większego ruchu sieciowego.

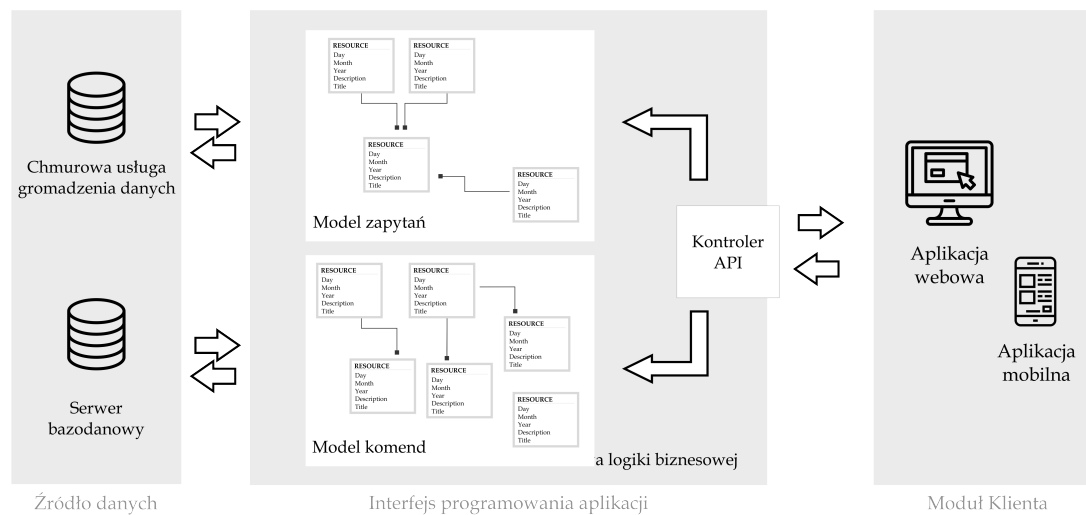
Niewątpliwymi zaletami, wynikającymi z zastosowania opisywanego wzorca projektowego są: zwiększenie efektywności operacji realizowanych z dużą częstotliwością, możliwość korzystania z osobnych źródeł, danych dla operacji odczytu oraz zapisu, zachowanie zasady pojedynczej odpowiedzialności (*ang. Single Responsibility Principle*) względem klasy logiki biznesowej API, a także redukcja liczby wstrzykiwanych zależności (*ang. Dependency Injection*) w ramach klas kontrolerów interfejsu [?].

Na ilustracjach ?? oraz ?? przedstawiono kolejno schemat przetwarzania... dań, wewnętrzny API z uwzględnieniem wzorca CQRS, a także przy wykorzystaniu pojedynczego modelu danych.

Odwólc się do ilustracji ?? należy zaznaczyć, że jest to jedna z możliwości implementacji wzorca CQRS, uwzględniająca wykorzystanie odseparowanych źródeł, danych. Architektura separacji zapytań, oraz komend w kontekście... dań,



Rys. 2.4: Schemat przetworzenia ŁŁÄ...dania przez interfejs API dla architektury z jednym modelem danych



Rys. 2.5: Schemat przetworzenia ŁŁÄ...dania przez interfejs API dla architektury wykorzystują...cej wzorzec projektowy CQRS

wysyłanych do interfejsu API, mógłby być takŁŁe wprowadzona przy wykorzystaniu pojedynczego systemu bazodanowego.

Konwencja REST

Niezależnie od struktury wewnętrznej omawianych usług sieciowych, współŁŁczesne interfejsy programowania aplikacji projektowane sąŁŁ tak, aby ich zewnętrzna warstwa (tj. widziana z perspektywy aplikacji klienckiej) cechowałaŁŁa sięŁŁ jednolitąŁŁ kompozycjąŁŁ.

Jednym z najpopularniejszych sposobów zapewnienia jednolitego interfejsu komunikacyjnego pomiędzy klientami i usługami sieciowymi, przetwarzającymi informacje z wykorzystaniem protokoŁŁu HTTP, jest konwencja oraz styl architektoniczny REST (*ang. Representational State Transfer*).

Konwencja ta, definiuje zbiór zasad dotycząŁŁcych m.in. zachowania usługi sieciowej w kontekście przetwarzania ŁŁÄ...dania konkretnego typu, struktury i elementów odpowiedzi na określone ŁŁÄ...danie, semantyki wykorzystywanych statusów rezultatu przetwarzania,

bezzastanowienia charakteru komunikacji, czy teŹ syntaktyki odwołaŹ, do poszczegŹlnych punktŹw koŹ,cowych.

W kontekŹcie stopnia implementacji stylu architektonicznego REST w ramach interfejsu programowania aplikacji, wprowadziŹ naleŹy pojŹcie modelu dojrzalŹ,oci Richardsona (ang. *Richardson Maturity Model*). PojŹcie to, definiuje cztery poziomy przystosowania interfejsu API do omawianej w niniejszej sekcji konwencji.

W odniesieniu do poziomu zerowego, powinnoŹciŹ... interfejsu programowania aplikacji jest udostŹpnienie usŹ,ug w ramach pojedynczego adresu sieciowego, niezaleŹnie od wykorzystywanych metod HTTP. Struktura Źadania klienckiego, w sposŹb jednoznaczny dostarczaŹ ma informacjŹ na temat wykonywanego wewnŹ... trz usŹ,ugi sieciowej dziaŹ,ania.

Zasada poziomu pierwszego, odnosi siŹ do charakterystyki interfejsu API jako usŹ,ugi zorientowanej na zasoby. NiezaleŹnie od czynnoŹci, jaka ma zostaŹ wykonana przez omawianŹ... usŹ,ugŹ sieciowŹ..., opis tej czynnoŹci wskazywaŹ ma na zasŹb ktŹrego ona dotyczy.

ReguŹ, a stanowiŹca definicjŹ poziomu trzeciego, zwiŹzana jest z semantykŹ... poszczegŹlnych typŹw ŹeŹ... daŹ, protokoŹ, u hipertekstowego. ŹeŹ... dane o takim samym adresie sieciowym, peŹ,niŹ powinno odmiennŹ... rolŹ, w zaleŹnoŹci od rodzaju ŹeŹ... dania HTTP.

Ostatnim z poziomŹw dojrzalŹ,oci interfejsu programowania aplikacji opartego o konwencjŹ REST jest reguŹ, a HATEOAS (ang. *Hypertext As The Engine Of Application State*). ReguŹ, a ta, definiuje interfejs API jako ŹsrŹdŹ, o informacji dotyczŹ... cej obsŹ,ugi stanu caŹ,ego systemu internetowego (tj. usŹ,ugi sieciowej wraz z aplikacjami klienckimi). Klient, po uzyskaniu odpowiedzi serwera na ŹeŹ... dane, powinien na podstawie zawartoŹci tej odpowiedzi mŹc zdefiniowaŹ przysŹ, e czynnoŹci, ktŹre wolno mu wykonaŹ [?].

2.3. Testowanie oprogramowania

Aspekt badawczy niniejszej pracy, zwiŹzany jest z realizacjŹ... procesu ewaluacji wydajnoŹci interfejsŹw programowania aplikacji, pod kŹtem wykorzystania odmiennych Źrodowisk implementacyjnych oraz uruchomieniowych. Proces ten, jest tylko jednym z wielu elementŹw domeny testowania oprogramowania, ktŹrej charakterystyka uwzglŹdnia zbierŹ sztywno zdefiniowanych reguŹ, cechujŹ... cych siŹ wysokim poziomem sformalizowania. W nastŹpnych sekcjach niniejszego akapitu dokonane zostaŹ, o wprowadzenie dotyczŹ... ce zagadnienia ewaluacji oprogramowania, nakreŹlone zostaŹ, y zasady testowania systemŹw informatycznych, zdefiniowano taksonomiŹ technik testowania, a takŹe omŹwiono proces przeprowadzania ewaluacji wydajnoŹci usŹ,ugi sieciowej jakŹ... jest interfejs programowania aplikacji.

Wprowadzenie do zagadnienia ewaluacji oprogramowania

Ewaluacja poszczegŹlnych skŹ,adowych tworzonego oprogramowania jest niezbŹdnŹ... czŹ... procesu budowy systemu informatycznego, niezaleŹnie od jego charakterystyki, czy teŹ wykorzystywanej do jego budowy technologii. W rozumieniu ogŹlnym, proces testowania czŹ...sto sprowadzany jest do zbioru dwŹch czynnoŹci. CzynnoŹciami tymi sŹ... uruchamianie oprogramowania, a takŹe eksploracja jego funkcjonalnoŹci w celu dostrzegania tych, w ramach ktŹrych zauwaŹyŹ moŹna niezgodnoŹ... ich dziaŹ,ania w stosunku do specyfikacji. Takie wnioskowanie jednak jest niepeŹ,ne, i uwzglŹdnia ono tylko jeden z etapŹw skŹ,adajŹ... cych siŹ na caŹ, y proces testowania. Dziedzina ewaluacji cech programŹw komputerowych, poszerzyŹ naleŹy ponadto o takie elementy jak: plano-

wanie testów, wybór kryteriów oceny oprogramowania, nadzór oraz kontrola realizacji badań, projektowanie przypadków testowych, czy też analiza spełnienia ustalonych kryteriów zakończenia.

Wybraliśmy mołemy znacząco liczbę definicji testowania oprogramowania, a każda z nich wprowadza inny poziom szczegółowości. Ponadto, wiele spośród formułowanych pojęć nawiązuje do różnych aspektów omawianego procesu. Zgodnie z jedną z najbardziej generycznych definicji, wprowadzoną przez Hetzla w publikacji [?], proces testowania oprogramowania określiliśmy jako zbiór wszystkich czynności, które nakierowane są na weryfikację atrybutów i właściwości programu, a także sprawdzenie tego, czy określony system spełnia założone wymagania. Definicja ta, względem wielu innych popularnych sformułowań, dotyczących ewaluacji oprogramowania, uwzględnia możliwość zastosowania statycznych technik testowania. Ponadto, jej autor bierze pod uwagę fakt, że w ramach procesu ewaluacji, oceniany powinien być łącznie z artefaktami tworzonych w ramach systemu, a nie tylko i wyłącznie kod źródłowy programu.

Taksonomia technik testowania

Jako jedno z podstawowych kryteriów podziału technik testowania oprogramowania, wskazać należy rodzaj czynności wykonywanej przez stronę testującą, której realizacja prowadzi do uzyskania charakterystyki programu poddanego ewaluacji. Według kryterium tego, wyróżniamy statyczne oraz dynamiczne techniki testowania.

Pierwsze, spośród przytoczonych metod, opierają się na analizie artefaktów oprogramowania (takich jak m.in.: kod źródłowy, specyfikacja, dokumentacja, czy też lista wymagań) bez ich wykonywania. Jako praktyczne przykłady przedstawionej techniki, zdefiniować należy: generowanie metryk kodu źródłowego programu, analizę przepływu sterowania, formalne dowodzenie poprawności działania, a także interpretację grafów wywołania.

Metody dynamiczne natomiast, związane są z weryfikacją właściwości poszczególnych elementów systemu informatycznego w trakcie jego wykonywania. Ten rodzaj testowania, nie uwzględnia formalnych struktur liczbowych, jakimi są grafy przepływu sterowania, czy też metryki kodu źródłowego programu. Zorientowany jest on, na odbiór systemu z perspektywy korzystającego z niego klienta.

Innym z rozważanych kryteriów podziału technik testowania jest ich umiejscowienie metody względem określonego fragmentu procesu wytwórczego. W nawiązaniu do tego aspektu, zdefiniować należy pojęcie poziomu testów, które jest powiązaniem sposobu ewaluacji oprogramowania z etapem jego realizacji. Istotnym rozróżnianiem danych poziomów testów jest założenie różnorodności celów testowania, a także testowanych obiektów, określanych w kontekście każdej z warstw. W odniesieniu do najbardziej popularnych systematyk poziomów testowania wyróżniamy następujące elementy:

- testy jednostkowe (zwane także modułowymi lub testami komponentów)
- testy integracyjne
- testy systemowe
- testy akceptacyjne

Pierwszy z poziomów, dotyczy znajdowania niezgodności specyfikacyjnych w obrębie logicznie oddzielonych jednostek oprogramowania (*ang. Software Units*). Każda z jednostek, powinna być testowana w izolacji od pozostałych elementów systemu. Ze względu na założenie różnorodności, informatycznych, a także statystycznie wy-

soki współczynnik wzajemnych zależności modułów, warunek ten często nie może zostać spełniony. W takich sytuacjach, aby dostarczyć zależności do testowanej jednostki, budowane są... moduły zastępcze, imitujące poprawne zachowanie określonego fragmentu programu (*ang. Mocks*). Omawiany poziom testowania, często postrzegany jest jako jeden z etapów procesu wytwarzczego, szczególnie w ramach takich technik jak rozwój oprogramowania napędzany testowaniem (*ang. Test Driven Development*).

Celem kolejnego z poziomów testowania jest weryfikacja poprawności współoddziaływania indywidualnych komponentów, a także prawidłowości funkcjonowania interfejsów definiowanych pomiędzy nimi. Przykładem współdziałania jednostek oprogramowania może być współpraca interfejsu programowania aplikacji, będącego systemem poddawany analizie w ramach niniejszej pracy, a także określonego silnika bazodanowego. W zależności od liczby weryfikowanych powińzań, pomiędzy poszczególnymi jednostkami oprogramowania, a także w odniesieniu do liczby samych modułów będących częścią testowanego fragmentu systemu, wyróżniamy małe, średnie oraz dużej skali integracji.

Na temat testów systemowych, należy pamiętać, wtedy, gdy wszystkie z elementów rozwijanego informatycznego zostały ze sobą powiązane w sposób spójny. Celem testów, realizowanych w ramach tego poziomu, jest weryfikacja wysokopoziomowej funkcjonalności oprogramowania, a także wykonywanie scenariuszy ewaluacji systemu z poziomu regularnego użytkownika (*ang. End-to-End testing*).

Ostatnim z wymienionych poziomów ewaluacji są... testy akceptacyjne. Przedmiotem oceny w ramach tego rodzaju testów jest gotowe rozwiązanie informatyczne w postaci komercyjnego produktu. Podmiot odpowiedzialny za realizację omawianych testów przygotowuje listę kryteriów akceptacji (*ang. acceptance criteria*), a następnie na podstawie obrotów testowanego rozwiązania, potwierdza lub odrzuca spełnienie każdego z nich. Celem omawianych ewaluacji nie jest znajdowanie błędów działania systemu, a nabranie zaufania co do jakości jego funkcjonalnych oraz niefunkcjonalnych atrybutów.

Wykonując testy definiowane w ramach kolejnych poziomów ewaluacji, weryfikowane zostają... na początek funkcjonalne, a następnie niefunkcjonalne elementy testowanego systemu. Jako weryfikację elementów funkcjonalnych, rozumieć należy wszystkie te czynności, które podejmowane są... w ramach wszystkich wymienionych powyżej poziomów testów, z wyjątkiem testów akceptacyjnych. Ewaluacja niefunkcjonalna natomiast, odnosi się tylko do ostatniego spośród wyróżnionych poziomów testowania.

Podział, charakteryzujący przedmiot ewaluacji względem omawianych aspektów definiuje pojęcia testów funkcjonalnych oraz niefunkcjonalnych i w kontekście niniejszej pracy jest on podziałem kluczowym.

Badania przeprowadzone w ramach tej pracy posiadają... charakter testów niefunkcjonalnych, a ich wykonanie poprzedzone jest weryfikacją funkcjonalną..., której poprawność traktowana jest jako wymóg.

Ewaluacja wydajności interfejsów programowania aplikacji

Zgodnie z teorią... przedstawioną w sekcji ?? niniejszej pracy interfejs programowania aplikacji postrzegamy jako deterministyczny system wejściowo-wyjściowy o charakterze dyskretnym. Takie podejście, w znaczący sposób ułatwia proces ewaluacji wydajności interfejsów API.

Definiowanie interfejsu API jako systemu pobudzanego pojedynczym wejściem, a także generującego pojedynczą wartość wyjściową..., pozwala na wykorzystanie sposobu oceny wydajności zwanego testem czarnoskrzynkowym (*ang. Black-box testing*). W ramach tego rodzaju testu, określone kryterium ewaluacji wyliczane jest jako różnica war-

toŁ>ci pomiaru na wyjŁ>ciu systemu, wzglÄ™dem tej, ktÅłrej kalkulacja nastÄ... piŁ, a na jego wejŁ>ciu. Taki rodzaj testu, umoŁliwia wyliczenie metryki wydajnoŁ>ci, bez koniecznoŁ>ci przygotowywania systemu do przeprowadzenia procesu testowania.

Podstawowym kryterium oceny wydajnoŁ>ci interfejsu programowania aplikacji jest czas odpowiedzi na ŁÄ...danie. MetrykÄ™ tÄ™, okreŁliÄŹ naleŁy jako czas od momentu wygenerowania ŁÄ...dania przez stronÄ™ klienta, do chwili uzyskania przez niego odpowiedzi. ZauwaŁyÄŹ naleŁy rÅłwnieŁŁ zaleŁnoŁ>ÄŹ czasu odpowiedzi na ŁÄ...danie, zarÅłwno od rozmiaru ŁÄ...dania jak i wielkoŁ>ci jego odpowiedzi. Ponadto, czynnikiem wpŁywajÄ...cym na uzyskany rezultat pomiaru jest niewÄ...tpliwie przepustowoŁ>ÄŹ Ł, Ä...cza sieciowego pomiÄ™dzy klientem a serwerem.

Aby rezultaty uzyskane w ramach oceny wydajnoŁ>ci API mogŁy zostaÄŹ postrzegane jako rzetelne, omÅłwione powyŁlej czynniki muszÄ... cechowaÄŹ siÄ™ statycznÄ... charakterystykÄ..., bÄ...dŁŹ teŁŁ zostaÄŹ caŁkowicie wyeliminowane z procesu testowego. W ramach niniejszej pracy, rozmiar odpowiedzi generowanej przez interfejsy programowania aplikacji jest staŁy, niezaleŁnie od zastosowanej technologii, a takŁŁe Ł>rodowiska uruchomieniowego. Wynika to z zastosowania pojedynczego i ustrukturyzowanego modelu danych, ktÅłry jest identyczny, niezaleŁnie od API. W odniesieniu do zmiennnoŁ>ci prÄ™dkoŁ>ci Ł, Ä...cza internetowego, aspekt ten zostaŁ, wyeliminowany poprzez przeprowadzanie testÅłw w obrÄ™bie lokalnej sieci komputerowej, a takŁŁe umiejscowienie interfejsÅłw oraz systemÅłw bazodanowych w ramach tej wŁaŁ>nie sieci.

Kolejnym z kryteriÅłw oceny wydajnoŁ>ci, uwzglÄ™dnianym w ramach procesu testowania usŁ,ug sieciowych jest poprawnoŁ>ÄŹ uzyskanej odpowiedzi w odniesieniu do liczby klientÅłw, rÅłwnolegle generujÄ...cych ŁÄ...dania. Kryterium to, charakteryzuje siÄ™ silnÄ... korelacjÄ... z czasem odpowiedzi na pojedyncze zapytanie.

UwzglÄ™dniajÄ...c oba opisane powyŁlej parametry, podstawowy schemat scenariusza badawczego dotyczÄ...cego ewaluacji wydajnoŁ>ci API skŁ,ada siÄ™ z nastÄ™pujÄ...cych testÅłw:

- testy linii bazowej - pojedynczy klient generuje ŁŁadania w kierunku API w celu zdefiniowania Ł>redniego czasu odpowiedzi usŁ,ugi w standardowych warunkach jej dziaŁ,ania. W ramach niniejszej pracy, podczas wykonywania omawianego testu, zdefiniowane zostanÄ...ponadto wartoŁ>ci wspÅłczynnikiÅłw satysfakcji, toleracji oraz frustracji, bÄ™dÄ...ce skŁ,adowymi wskaŁŹnika jakoŁ>ci APDEX. WartoŁ>ci te, stanowiÄ...uogÅłnionÄ...ocenÄ™ wydajnoŁ>ci i poŁ,uŁÄ...jako punkt odniesienia dla kolejnych testÅłw.
- testy obciÄ...Łleniowe - uwzglÄ™dniana zostaje zmienna liczba klientÅłw generujÄ...cych ŁÄ...dania, w kontekŁ>cie ktÅłrej ustalane sÄ...Ł>rednie czasu odpowiedzi na ŁÄ...danie, a takŁŁe dokonywane zostaje odniesienie uzyskanego rezultatu do wspÅłczynnikiÅłw zdefiniowanych uprzednio w ramach miary APDEX.
- testy przeciÄ...ŁŁajÄ...ce - liczba klientÅłw generujÄ...cych ŁÄ...danie zostaje dobrana w taki sposÅłb, aby doprowadziÄŹ do obciÄ...Łlenia testowanej usŁ,ugi, niepozwalajÄ...cego na poprawne funkcjonowanie interfejsu programowania aplikacji. Ewaluacja ta, ma na celu znalezienie punktu krytycznego w kontekŁ>cie dziaŁ,ania testowanego oprogramowania.

Poza czarnoskrzynkowymi testami wydajnoŁ>ci, cechujÄ...cymi siÄ™ omÅłwionÄ...powyŁlej strukturÄ..., przeprowadzone mogÄ...zostaÄŹ ewaluacje efektywnoŁ>ci dziaŁ,ania poszczegÅłlnych fragmentÅłw usŁ,ugi sieciowej, w ramach ktÅłrych interfejs programowania aplikacji postrzegaÄŹ naleŁy w odmienny sposÅłb, aniŁŁeli jako system wejŁ>ciowo-wyjŁ>ciowy.

PrzykŁ,adem oceny wydajnoŁ>ci okreŁ>lonego fragmentu interfejsu programowania aplikacji moŁŁe byÄŹ ewaluacja moduŁ,u realizacji zaawansowanych operacji obliczeniowych, do-

stającego z poziomu punktu kołowego API. W takim przypadku, oprogramowanie musi zostać dostosowane do przeprowadzenia procedury testowej, a metryka wydajności - powiązana z postępowaniem realizacji obliczeń.

2.4. Wykorzystywane narzędzia i technologie

Zarówno w trakcie procesu implementacji badanych interfejsów programowania aplikacji, jak i procedurze przeprowadzenia badań, pod kontem ich wydajności, wykorzystano obszerny zbiór sprawdzonych i powszechnie stosowanych rozwiązań, technologicznych. W ramach niniejszej sekcji, opisane zostanie każde z nich.

C#

C# jest wieloparadygmatowym, a także nowoczesnym językiem programowania ogólnego przeznaczenia, charakteryzującym się bezpieczeństwem i niezawodnością w aspekcie typowania struktur danych. Pierwsza z wersji tego języka, stworzona została przez Andersa Hejlsberga w roku 1998. Od tamtej chwili, do momentu napisania niniejszej pracy, opublikowanych zostało 9 kolejnych, stabilnych wydań, projektu C#. Każda z następnych wersji omawianego języka programowania wprowadzała zarówno usprawnienia w kontekście ekosystemu budowy i kompilacji programów źródłowych, jak i wzbogacała interfejs bibliotek funkcyjnych o kluczowe z punktu widzenia doświadczonego programisty rozwiązania. Do rozwiązań, tych, zaliczyć należy między innymi: mechanizmy programowania współbieżnego, typy anonimowe, operatory zmiennych typów niezdefiniowanych, obsługa referencji, typy generyczne, czy też wyrażenia lambda.

W ramach niniejszej pracy, język C# wykorzystany został, do implementacji jednego z dwóch zbiorów interfejsów programowania aplikacji. Ze względu na zastosowanie rozwiązań, z zakresu przetwarzania współbieżnego (tj. operacji asynchronicznych oraz wielowątkowych) udostępnianych przez omawiany język programowania, interfejsy API realizowane w tej technologii mogą obsługiwać w sposób równoległy, dane pochodzące od wielu klientów, a także utrzymać sekwencyjny charakter przetwarzanych procedur niezależnie od czasu ich wykonywania. Należy także zwrócić uwagę na mechanizm wewnętrznych usprawnień, wydajnościowych implementowanych w ramach kompilatora i uruchamiany w momencie tłumaczenia kodu języka do tzw. języka pośredniego (*ang. Intermediate Language*). Dzięki zastosowaniu przedstawionego mechanizmu, operacje zdefiniowane przez programistę mogą być modyfikowane w procesie kompilacji, tak aby nie wpłynęły na zaimplementowaną funkcjonalność, zwikszał c jednocześnie wydajność generowanego programu [?].

.NET Core

.NET Core postrzegany należy jako środowisko budowy, kompilacji oraz wykonywania rozwiązań, implementowanych w języku C#. Przedstawiana technologia stanowi podzbior bibliotek, dzięki którym programista jest w stanie budować systemy różnorodnego przeznaczenia, a także uruchamiać je w wielu wspieranych środowiskach programowych. W przeciwieństwie do technologii .NET Framework będącej poprzednikiem .NET Core, aplikacje tworzone na bazie omawianej biblioteki mogą być wydawane nie tylko na system operacyjny Windows, ale także na systemy Linux oraz MacOS.

W ramach omawianego środowiska wykorzystywany zostaje komponent języka C# zwany biblioteką standardową (*ang. .NET Standard Library*). Biblioteka ta jest

współna dla wielu środowisk uruchomieniowych, a zawarte w niej funkcjonalności, traktowane są jako metody ogólnego przeznaczenia.

Ponadto, środowisko .NET Core, w ramach procesu budowy i kompilacji rozwiązania nawiązuje komunikację z komponentem wspólnej infrastruktury (*ang. Common Infrastructure*). Komponent ten, podobnie jak biblioteka standardowa, współdzielony jest przez wiele środowisk wykonawczych. W kontekście wspólnej infrastruktury, wspomnieć należy o wspólnej specyfikacji języka (*ang. CLS - Common Language Specification*), wspólnym systemie typów (*ang. CTS - Common Type System*), a także środowisku uruchomieniowym wspólnego języka (*ang. CLR - Common Language Runtime*). Wykorzystanie między innymi tych trzech elementów, pozwala na budowę systemu dostępnego na wielu platformach [?].

W kontekście realizowanej pracy, technologia .NET Core ulegała zmianom, a jako środowisko uruchomieniowe dla interfejsów programowania aplikacji tworzonych w języku C#. W obrębie technologii tej, poza przedstawionymi powyżej komponentami, wyróżnić można modyfikacje natywnej biblioteki ASP.NET Core, stanowiące zbiór metod przydatnych w procesie definiowania internetowych usług sieciowych oraz aplikacji webowych. Dzięki zastosowaniu ASP.NET Core operacje takie jak, między innymi: obsługa definicji kontrolerów API, zarządzanie stanem ciała, a także dane oraz jego rzutowaniem na określony typ danych, czy też implementacja mechanizmów uwierzytelniania i autoryzacji klienta, wykonane mogą zostać na wysokim poziomie abstrakcji z jednoczesnym zapewnieniem należytego poziomu ich wydajności.

Entity Framework Core

Entity Framework Core stanowi narzędzie stworzone przez firmę Microsoft, którego zastosowaniem jest mapowanie obiektowo-relacyjne realizowane w kontekście usług internetowych tworzonych z wykorzystaniem języka C# oraz uruchamianych na platformie .NET Core. Przedstawiana biblioteka zapewnia programiście zorientowany obiektowo interfejs, za pomocą którego można uzyskać dostęp do danych, a także je definiować oraz przetwarzać. Zbiory obiektów mogą być składowane zarówno w relacyjnych jak i niereleacyjnych bazach danych. Niniejsza biblioteka, podobnie do środowiska uruchomieniowego .NET Core, jest rozwiązaniem wieloplatformowym i może być wykorzystywana przy budowie systemów internetowych wdrażanych na systemach Windows, Linux oraz macOS.

Zastosowanie biblioteki mapera obiektowo-relacyjnego jak jest Entity Framework Core umożliwia zastosowanie podejścia zorientowanego na kod źródłowy w kontekście aplikacji komunikujących się i wykorzystujących zewnętrzne zbiory danych (*ang. Code-First Approach*). Podejście to, polega na definiowaniu w ramach kodu źródłowego zbioru klas modelu danych, które następnie przekształcane do postaci tabel określonego systemu bazodanowego. Przedstawiona operacja przekształcania wykonywana jest bezpośrednio za pomocą mechanizmów mapera obiektowo-relacyjnego.

Niezaprzeczalną zaletą wykorzystania biblioteki ORM jak jest Entity Framework Core stanowi możliwość operowania na jednolitym interfejsie realizacji operacji na danych, niezależnie od obsługiwanego systemu bazodanowego. Oznacza to, że w momencie zmiany dostawcy zewnętrznego źródła danych, zawartość kodu źródłowego programu nie musi podlegać modyfikacji [?].

MediatR

MediatR to otwartoŹródło,owa biblioteka jÄ™zyka C#, z wykorzystaniem ktÄ™rej zaimplementowany moŹle zostaŹ wzorzec projektowy, dotyczÄ...cy separacji odpowiedzialnoŹci za obsŹugÄ™ zapytaŹ, oraz komend przetwarzanych przez usŹugÄ™ sieciowÄ... Kluczowym elementem biblioteki MediatR jest para generycznych interfejsów, za pomocÄ... ktÄ™rych implementowana jest obsŹuga zarÄ™wno ŹŁÄ...daŹ, jak i zapytaŹ, dotyczÄ...cych danych. Interfejsami tymi sÄ... kolejno: *IRequest* - struktura programistyczna implementowana przez klasy definiujÄ...ce zawartoŹciaŹ, a ŹŁÄ...dania lub komendy, a takŹle powiÄ...zany z niÄ... *IRequestHandler*, ktÄ™ry jest konkretyzowany przez klasÄ™ definicji metody obsŹugi ŹŁÄ...dania bÄ...dŹs operacji na danych.

Ponadto, naleŹy rÄ™wnieŹ podkreŹliŹ znaczenie metody *Send* dostÄ™pnej w ramach gŹŹwnego API pakietu MediatR. DziÄ™ki niej, wywoŹana moŹle zostaŹ procedura obsŹugi okreŹlonego ŹŁÄ...dania lub operacji, z dowolnego miejsca kodu Źródłowego interfejsu programowania aplikacji [?].

JavaScript

JavaScript to wielofunkcyjny oraz wieloplatformowy skryptowy jÄ™zyk programowania cechujÄ...cy siÄ™ wysokim poziomem abstrakcji. Najbardziej popularnym przeznaczeniem omawianego jÄ™zyka jest budowa systemów internetowych, a takŹle mobilnych. HistorycznÄ... rolÄ... technologii JavaScript byŹo udostÄ™pnianie programiŹcie funkcjonalnoŹci umoŹliwiajÄ...cych okreŹlanie rÄ™Źnorodnych sposobów interakcji pomiÄ™dzy uŹytkownikiem serwisu internetowego, a jego statycznymi elementami. Podstawowym Źródowiskiem wykonania oraz interpretacji omawianego jÄ™zyka byŹa uwczeŹnie przeglÄ...darka internetowa. Wraz z pojawieniem siÄ™ serwerowego Źródowiska uruchomieniowego NodeJS, przeznaczonego dla jÄ™zyka JavaScript, popularnoŹÄ... omawianej technologii wzrosŹa w gwaŹtownym tempie. Zmianie ulegŹo rÄ™wnieŹ gŹŹwne przeznaczenie technologii, ktÄ™ra od tej pory staŹa siÄ™ peŹnoprawnym jÄ™zykiem programowania, stosowanym w kontekŹcie budowy zarÄ™wno systemów internetowych, rozwiÄ...zaŹ, mobilnych, jak i programów desktopowych.

JÄ™zyk JavaScript uznÄ... naleŹy za technologiiÄ™ charakteryzujÄ...cÄ... siÄ™ typowaniem sŹabym oraz dynamicznym. W zwiÄ...zku z zastosowaniem przez twórców rozwiÄ...zania takiego wŹaŹnie podejŹcia, tworzone kody programów naraŹone sÄ... na wystÄ™powanie zjawisk niezgodnoŹci typów, a takŹle niejawnej koercji. Ponadto, w kontekŹcie mechanizmów omawianego jÄ™zyka, realizacja operacji przetwarzania współŹbieŹnego oraz wykonania metod asynchronicznych, zaleŹna jest w caŹkowitym stopniu od rozwiÄ...zaŹ, implementacyjnych poczynionych w ramach Źródowiska uruchomieniowego. Oznacza to, Źle przetwarzanie i wykonywanie operacji wielowÄ...tkowych moŹle cechowaŹ siÄ™ zrÄ™ŹnicowanÄ... wydajnoŹciÄ..., w zaleŹnoŹci od konkretnego interpretera jÄ™zyka.

NiewÄ...tpliwymi zaletami technologii JavaScript sÄ...: skŹadnia cechujÄ...ca siÄ™ niskim poziomem ŹŹonoŹci poleceŹ, moŹliwoŹÄ... dowolnego wykorzystywania wielu współŹród wspieranych paradygmatów programowania, modułowoŹÄ... i skalowalnoŹÄ... implementowanych rozwiÄ...zaŹ, a takŹle elastycznoŹÄ... w kontekŹcie operowania na wykorzystywanych strukturach danych [?].

W ramach niniejszej pracy, jÄ™zyk JavaScript zastosowany zostaŹ, w celu implementacji jednego z dwóŹch zbiorów badanych interfejsów programowania aplikacji. Tworzone w omawianym jÄ™zyku API, wykonywane bÄ™dÄ... w Źródowisku uruchomieniowym NodeJS.

TypeScript

TypeScript stanowi statycznie typowany nadzbiór języka JavaScript. Określenie to, oznacza że omawiana technologia nie jest strictly językiem programowania, a tylko określoną... grupą... instrukcji oraz procedur, które w... czy może do języka JavaScript, po to, aby zapewnić w jego kontekście statyczny sposób typowania danych. Technologia TypeScript nie może być wykorzystywana samodzielnie, a środowisko wykonawcze JavaScript jest wymagane w celu uruchomienia skompilowanego modułu, definiowanego zgodnie ze składnią... omawianego języka.

Kluczowym elementem przedstawianej technologii jest transpiler języka TypeScript o nazwie tsc (*ang. TypeScript Compiler*). Program ten, uruchamiany jest tuż przed rozpoczęciem procedury interpretacji kodu JavaScript i przekształca on metody odpowiedzialne za obsługę typów danych, do struktur dostępnych w ramach standardowej implementacji języka. Dlatego też, z punktu widzenia środowiska uruchomieniowego, dostępne programistom mechanizmy definicji typów czy interfejsów, nie są... znane.

Celem zastosowania omawianego nadzbioru językowego jest możliwość kontroli zgodności definiowanych obiektów programistycznych pod kątem ich wewnętrznej struktury. Ponadto, wykorzystanie TypeScript umożliwia weryfikację faktu nieumyślnego odwołania się do struktury typu nieokreślonego, jeszcze przed rozpoczęciem procesu interpretacji kodu [?].

NodeJS

NodeJS jest środowiskiem uruchomieniowym języka JavaScript zbudowanym w oparciu o otwartość silnik interpretacji kodu Chrome V8. Dzięki zastosowaniu omawianego środowiska uruchomieniowego, kod języka JavaScript może być wykonywany poza ekosystemem przeglądarki internetowej. Rozwój niniejszej technologii, doprowadził, do diametralnej zmiany w obszarze zastosowania języka JavaScript, a także gwałtownego wzrostu jego popularności w kontekście budowy systemów internetowych.

Podobnie do rozwijania Microsoft .NET Core, platforma NodeJS składa się nie tylko ze środowiska uruchomieniowego, ale także ze zbioru bibliotek oraz narzędzi linii komend. Aplikacje budowane na bazie omawianej technologii cechują się zastosowaniem architektury sterowanej zdarzeniami (*ang. Event-driven architecture*), która ponadto wzbogacona jest (dzięki wykorzystaniu mechanizmu promieni) o obsługę operacji asynchronicznych. Co więcej, rozwijanie definiowane na podstawie środowiska NodeJS posiadają... budowę modułów... co przyczynia się do zwiększenia ich zdolności w kontekście skalowania systemów.

Należy również uwypuklić generyczność charakterystyki środowiska NodeJS. Nie jest ono przeznaczone strictly do definiowania i wdrażania usług sieciowych, a wykorzystywane jest do uruchamiania dowolnego kodu języka JavaScript, jaki może zostać stworzony za pomocą... jego składni. Dlatego też, aby dostarczać mechanizmy dotyczące specyficznych funkcjonalności, ekosystem NodeJS może być rozbudowywany poprzez otwartość moduły. Liczną... modułów tych, a także popularność ich wykorzystania stanowi niewątpliwie o sile omawianej technologii [?].

ExpressJS

ExpressJS stanowi bibliotekę środowiska NodeJS, dostarczającą... zbiór metod pozwalających na budowę webowych interfejsów programowania aplikacji w tym w... środowisku. Pakiet ExpressJS cechuje się minimalizmem w kon-

tekście złoŹono ci udostępnianych programów operacji, elastyczności... dotyczą... współpracy z zewnętrznymi pakietami, a także wysoka wydajność... działania tworzonych aplikacji, poprzez eliminację złoŹonych funkcjonalności przetwarzania zasobów w obrębie... dania.

Koncepcja przetwarzania zasobu uzyskanego od klienta w ramach ExpressJS sprowadza się do potokowej obsługi dostarczonego wejścia, przez kolejne funkcje pośredniczące (ang. *Middleware functions*). Ostatnia z funkcji ma za zadanie zwrócić odpowiedź wygenerowaną na podstawie operacji wykonywanych przez wszystkie poprzednie metody. Ciężem funkcji pośredniczącej może być zarówno kod źródłowy zdefiniowany przez programistę, jak i ten dostarczony poprzez referencję do zewnętrznego pakietu. Do każdej z metod przekazywane są... parametry... dania, odpowiedzi, a także referencje do następnego middleware. Dzięki uruchomieniu napisanej w taki sposób aplikacji w środowisku NodeJS, poszczególne funkcje pośredniczące mogą być realizowane asynchronicznie [?].

Prisma

Prisma ORM to narzędzie pełniące rolę mapera obiektowo-relacyjnego wykorzystywanego w kontekście implementowania interfejsów programowania aplikacji w języku JavaScript. Dystynktywnymi cechami omawianego narzędzia jest prostota definiowania rzutowanego modelu danych. W przypadku pakietu Prisma, cała struktura modelu danych opisywana jest w ramach jednego pliku zwanego plikiem schematu. W pliku tym, określone zostają zarówno właściwości poszczególnych encji modelu, jak i relacje między nimi występujące...

Analogicznie do narzędzia Entity Framework Core, Prisma ORM wspiera zarówno relacyjne jak i nierelacyjne systemy bazodanowe. Ponadto, zauważyć należy kompatybilność omawianego narzędzia z omawianymi wcześniej technologiami... TypeScript [?].

Mongoose

Biblioteka Mongoose stanowi narzędzie rzutowania obiektowego modelu danych definiowanego w ramach kodu programu, do postaci obiektowej nierelacyjnej bazy danych MongoDB. Technologii tej nie należy nazywać maperem obiektowo-relacyjnym, gdyż wykonywane przez nią... operacje synchronizują... struktury danych, które zarówno po stronie kodu, jak i po stronie źródła, a danych cechują... się obiektami... natury...

Poleganie interfejsu programowania aplikacji z nierelacyjnym źródłem danych obsługiwanym przez bibliotekę Mongoose, prowadzi do zwiększenia efektywności oraz zmniejszenia czasu wykonania operacji na danych. Jednakże, w związku z naturą przechowywanych informacji, a także brakiem uwzględnienia w ich strukturze metadefinicji, zastosowanie omawianego mechanizmu może również prowadzić do braku spójności źródła danych, a także niemożliwości zastosowania usprawnień, wydajnościowych w kontekście bazy danych [?].

Apache JMeter

Narzędzie Apache JMeter to otwartoźródłowe oprogramowanie stworzone w języku Java. Program ten, wykorzystywany jest do przeprowadzania ewaluacji wydajności oprogramowania sieciowego opierającego swoje działania o protokoły HTTP oraz FTP. Testy efektywności działania usług mogą zostać przeprowadzane w trybie lokalnym (tj. z wykorzystaniem jednego hosta wysyłającego... dania do określonej usługi sieciowej),

jak i w trybie rozproszonym (tj. budowana zostaje hierarchia hostów bTMd...cych generatorami ŁŁ...daŁ,,). W ramach niniejszej pracy, zastosowany został, drugi z przedstawionych trybów ewaluacji.

Działanie oprogramowania Apache JMeter sprowadza się do wykonywania pętli testowej, w ramach określonych grup wątków. Pętla testowa symuluje sekwencyjne generowanie żądań, w kierunku serwera, z uwzględnieniem stałej wartości opóźnienia pomiędzy wysłanymi pakietami. Grupa wątków natomiast, określa współrzędne charakterystyki obciążenia usługi sieciowej i może być utożsamiana zarówno z konkretnymi wątkami procesora lokalnego hosta, jak i z oddzielnymi pracującymi generatorami żądań, w trybie rozproszonym.

W celu zdefiniowania testu wydajności w ramach Apache JMeter, zbudowany powinien zostać plan testowy. Jest to podstawowa jednostka wyrażana w ramach niniejszego oprogramowania i skupia ona w sobie między innymi komponenty grup wątków, próbników (ang. *Samplers*), a także elementów nasłuchujących na odpowiedzi usługi (ang. *Listeners*). Zarówno próbniki, jak i elementy nasłuchujące mogą być powielane w ramach planu testowego, a także indywidualnie konfigurowane, w zależności od specyfiki usługi sieciowej.

Oprogramowanie Apache JMeter obsŁ, ugiwaÄŁ moŁŁna zarÄŁwno z poziomu narzÄ™dzia linii komend, jak i udostÄ™pnionego graficznego interfejsu uŁŁytkownika [?].

2.5. Przegląd literatury

W niniejszym rozdziale przedstawione zostaną... pozycje literaturowe, do których odnosi się siTM bTM gdzie opisywana praca dyplomowa. Pozycje te, podzielone zostały na oddzielne grupy, związane z określonymi... tematami...

Na początku, przedstawiona zostanie literatura poświęcona z aspektem budowy interfejsów programowania aplikacji oraz wprowadzeniem do wykorzystywanych technologii. Następnie, opisane zostaną pozycje traktujące o wydajności interfejsów API, a także o analizie działania powszechnie dostępnych serwisów internetowych opartych o metodologię REST. Kolejne prace, skupiają się na tematyce testowania usług sieciowych, teorii testowania, a także konfiguracji narzędzi dla testów rozproszonych. W następnej kolejności, wspomniane zostaną prace naukowe oraz dokumenty standaryzacyjne dotyczące sposobu działania protokołu przesyłania danych hipertekstowych. Ostatnią grupą pozycji literaturowych będą prace referencyjne dotyczące badań, wydajności systemów internetowych.

Pozycja [?] stanowi wprowadzenie do zaawansowanych konceptów języka C#, a także dostarcza informacji związanych z wykorzystaniem tego języka w środowiskach uruchomieniowych .NET oraz .NET Core. W początkowych rozdziałach przedstawiono sposób budowy, kompilacji oraz wykonywania programu w środowisku .NET. Kolejno opisana została struktura bazowych aplikacji uruchamianych w tym właśnie środowisku i tworzonych za pomocą języka C#. Ostatnim elementem wprowadzenia do opisywanej technologii było przedstawienie struktur języka w kontekście obiektowego paradygmatu programowania. W następnych sekcjach literatury, w sposób wyczerpujący poruszono tematykę bardziej zaawansowanych aspektów programowania w języku C#, którymi są między innymi: kolekcje i typy generyczne, delegaty i wyrażenia lambda, czy też cykl życia obiektu w pamięci programu. Ważnym tematem, poruszonym w ramach tej książki jest struktura oraz zasada działania środowiska .net core, którego podstawowym elementem interfejsów programowania aplikacji tworzonych w języku C#.

Analogicznie... do przedstawionej powyżej pozycji literaturowej, dotyczącej... jednak technologii NodeJS oraz języka JavaScript jest [?]. W ramach tej pracy zawarto obszernie wprowadzenie do platformy NodeJS uwzględniające ponadto kwestie obsługi operacji wejścia/wyjścia, wykonywania natywnego kodu JS, czy też przetwarzania operacji przez silnik NodeJS oraz bibliotek libuv. Znaczna część pracy, obejmuje przedstawienie zaawansowanych wzorców projektowych, których głównym przeznaczeniem jest obsługa zdarzeń, oraz operacji asynchronicznych. Wspomniane zostały także rozwiązania dotyczące skalowalności aplikacji z wykorzystaniem mechanizmów kolejkowania wiadomości.

Niezależnie od wykorzystywanej technologii, interfejsy programowania aplikacji, które zostały zbudowane na potrzeby tej pracy dyplomowej, oparte są o styl architektoniczny RESTful. Styl ten, jest pewnym zbiorem zasad projektowania usług sieciowych, określających również aspekty sposobu komunikacji klienta z usługą sieciową, jak i techniczne wymagania dotyczące przetwarzania danych. Dobre praktyki, które uwzględnia metodologia REST, zawarte zostały w pozycji literaturowej [?]. Autorzy tego dokumentu, na wstępie dokonują porównania architektury zorientowanej na zasoby, będącej podstawą konwencji REST, z popularną uprzednio architekturą zorientowaną na usługi. Następnie, przedstawiane są najlepsze praktyki, cele oraz reguły REST dotyczące projektowania interfejsu programowania aplikacji. Co więcej, w omawianej książce zawarte zostały także podstawowe oraz zaawansowane wzorce projektowania API, uwzględniające aspekty bezstanowości, paginacji, ogólności, a także identyfikacji zasobów interfejsu. Końcowe rozdziały książki, wprowadzają w kwestie testowania oraz bezpieczeństwa REST API, omawiają technikę kompozycji usług RESTful, a także przedstawiają rozwiązania (biblioteki oraz języki programowania) pozwalające na tworzenie interfejsów API zgodnych z metodologią REST.

Podstawowym celem działania interfejsu programowania aplikacji jest dostarczenie danych do konsumenta, bądź ich manipulacja zgodnie z jego życzeniem. Aby operować na danych, interfejs API musi komunikować się ze źródłem danych, którym najczęściej jest serwer bazodanowy. W celu dostarczenia metod komunikacji pomiędzy API a źródłem danych, które jednocześnie są niezależne od wykorzystywanego źródła, a także pozwalają na zarządzanie danymi z poziomu struktury języka, stworzone zostały biblioteki zwane maperami obiektowo-relacyjnymi (ang. Object-Relational Mappers). Dla API napisanego w języku C# podstawowym rozwiązaniem ORM jest biblioteka Entity Framework Core, która przedstawiona została w pozycji [?]. Pozycja ta, uwzględnia również opis działania najczęściej wykorzystywanych metod służących do manipulacji danymi, jak i rolę klasy kontekstu bazodanowego w procesie tłumaczenia operacji programistycznych na polecenia bazodanowe. Ponadto, dowiedziemy się jak przetwarzać zaawansowane typy danych (takie jak np. DateTime), czy też w jaki sposób wykorzystywać zapytania LINQ do budowania kwerend.

Dla interfejsu programowania aplikacji napisanego w języku JavaScript i uruchamianego w środowisku NodeJS, w przeciwieństwie do platformy .NET, zastosujemy zdecydowanie większą liczbę bibliotek pełniących rolę mapek obiektowo-relacyjnych. Biblioteki te, zostały opisane w pozycjach [?] i [?]. Pozycja [?] pełni rolę całościowego wprowadzenia do tematyki tworzenia interfejsów API, korzystając z platformy NodeJS, frameworka ExpressJS oraz nierelacyjnej bazy danych MongoDB. Rodział piąty tej pracy, traktujący o wykorzystaniu baz danych NoSQL, przybliży tematykę jednego z najczęściej wykorzystywanych mapek obiektowo-relacyjnych dla Node czyli mongoose. Przedstawiono tutaj sposób zestawienia połączenia z serwerem bazodanowym, tworzenia encji modelu, przekształcanego następnie na struktury bazy danych, a także wykonywania operacji dostępu do danych i ich modyfikacji. W pracy [?] natomiast, po-

rozważano nierelacyjne podejście do składowania danych typu geograficznego z podejściem relacyjnym, wykorzystując w tym przypadku biblioteki mongoose i sequelize. Oba mapery obiektowo relacyjne zostały użyte w ramach interfejsu API wykorzystującego technologię NodeJS/ExpressJS. Celem opisywanej pracy było przedstawienie różnic w czasach odpowiedzi API na uzyskane dane, dla różnej liczby danych geolokalizacyjnych, uwzględniając zastosowanie relacyjnych i nierelacyjnych baz danych.

Następne pozycje literaturowe, związane z analizą usług REST oraz wydajności webowych interfejsów programowania aplikacji.

Pozycja [?] stanowi analizę 500 serwisów internetowych z listy alexa.com 4000 najpopularniejszych publicznie usług sieciowych. Twórcy każdego z 500 serwisów deklarują zgodność swoich produktów z konwencją REST. Przeprowadzona analiza dotyczyła kluczowych aspektów technicznych związanych z funkcjonowaniem API, stopnia zgodności API z regułami dotyczącymi metodologii REST, a także przestrzegania najlepszych praktyk projektowania interfejsów programowania aplikacji, takich jak m.in. zastosowanie mechanizmu wersjonowania. W trakcie analizy, zaobserwowano określone trendy dla aplikacji REST API, takie jak m.in. rozpowszechnione wsparcie notacji JSON, czy wykorzystywanie narzędzi do dokumentacji generowanej programowo. Ponadto, zauważono, że tylko ok. 0.8% analizowanych serwisów webowych przestrzega w sposób ścisły reguł, zawartych w ramach konwencji REST.

Wydajność interfejsów programowania aplikacji, jako jeden z elementów miary jakości API została przedstawiona w pozycji [?]. Na początku pracy, jej autorzy wskazują na interakcję interfejsu programowania aplikacji z systemami klienckimi. Opisany został tutaj zestaw protokołów sieciowych wykorzystywanych podczas formułowania i transmisji danych, system zunifikowanych lokacji zasobów, a także semantyka interakcji w zależności od wykorzystywanych typów danych, protokołu hipertekstowego. Ponadto, wskazano najczęstsze przyczyny błędów przepływu danych dla http, uwzględniając działanie usług DNS, błędów połączenia, błędów leżących po stronie klienta, a także błędów wynikających z działania serwera. Kolejną częścią pracy, związanej ze składowymi metrykami jakości, do których według autorów, poza wydajnością, zaliczyć możemy: dostępność, procent danych, dla których uzyskano pozytywną odpowiedź, osiągalność, a także możliwość sprawdzenia stanu usługi w dowolnym momencie jej działania. Dodatkowo, w niniejszej pracy zaproponowano podejście oraz zestaw narzędzi pozwalających na dokonanie ewaluacji jakości interfejsu programowania aplikacji, zgodnie z przyjętą normą jakością.

Kolejnym etapem następującym po zdefiniowaniu metryki wydajności, jest ustalenie wartości tej metryki w kontekście testowanych usług sieciowych. Przytoczone poniżej pozycje literaturowe, związane z wykonywaniem pomiarów wydajności API, czyli testowaniem.

Pozycja [?] stanowi obszernie wprowadzenie do teorii testowania oprogramowania. W pierwszych rozdziałach tego dokumentu, wyjaśniono czym jest testowanie, dlaczego jest ono niezbędne podczas tworzenia oprogramowania, a także jak wygląda podstawowy proces wykonywania testów. Następnie przedstawiono proces testowania w kontekście tworzenia oprogramowania. Uwzględniono tu zarówno modele cyklu życia rozwoju systemów w powiązaniu z testowaniem, poziomy realizowanych testów, ich typy, jak i sposoby zarządzania testami. Kolejne rozdziały dotyczą siatek testowania statycznego (tj. testowania funkcjonalności lub modułu na poziomie jego specyfikacji lub implementacji bez wykonywania kodu testowanego oprogramowania), dostarczają teorii związanej z poszczególnymi technikami testowania rozwiązaniem, oraz przedstawiają aspekt organizacji, planowania, monitorowania oraz uwzględniania ryzyka w czasie dokonywania ewaluacji systemów. W

ostatnim z rozdziałów dokumentu, autorzy przedstawiają... narzędzia przydatne w procesie testowania, a także sposoby ich efektywnego wykorzystania w codziennej pracy.

Pozycja [?] zawiera wiele analogicznych treści do pracy opisanej powyżej, jednakże rozwija ona w sposób wyczerpujący, wspomniane tylko w poprzedniej pracy aspekty. W cząstce drugiej dokumentu zawarto dogłębnie analizę zagadnienia testowania statycznego, uwzględniając m.in. testowanie zgodności ze standardami oprogramowania, symboliczne wykonywanie kodu, a nawet wprowadzając aparat matematyczny do formalnego dowodzenia poprawności fragmentów oprogramowania. W ramach tej książki, przedstawiono także dynamiczną analizę systemu (tj. testowanie funkcjonalności lub modułu, u na poziomie wykonywanego kodu) uwzględniając cząstko występujące błędy związane m.in. z nieumiejętnym zarządzaniem strukturami pamięci programu. Ponadto, uwzględniono zagadnienie priorytetyzacji przypadków testowych, wprowadzając pojęcie miary średniego procenta wykrytych usterek. Autor dokumentu przedstawia także testowanie charakterystyk jakościowych zgodnie z normą ISO 9126 oraz ISO 25010, tworzenie dokumentacji w ramach zarządzania testowaniem, czy chociażby zarządzanie incydentami występującymi w ramach procesu ewaluacji oprogramowania.

W ramach pozycji [?], dowiedziemy się ponadto o testowaniu usług internetowych. Przedstawiono tutaj podstawową strukturę standardowej usługi sieciowej (w tym przypadku – usługi e-commerce) cechującą się siłą architektury, trwałością i wydajnością. Ponadto, wyjaśniono rolę każdej z warstw systemu, a także przedstawiono aspekty testowania oprogramowania w kontekście każdej z nich. Dodatkowo, zawarte zostały przykładowe przypadki testowe, dotyczące zarówno prezentacji danych w systemie, jak i dostępu do danych poprzez serwer webowy. Dla zaprezentowanych przypadków testowych, przedstawione zostały także scenariusze realizacji testów w postaci listy czynności i jakie należy podjąć, aby dokonać ewaluacji systemu.

Aspekty technologii testowania oprogramowania ujęte zostały także w pozycji [?]. Artykuł, ten, stanowi sekcję wprowadzającą do książki pt. Tutorial: Software Testing and Validation Techniques, tego samego autora. Pozycja ta, przedstawia przekrój technik oraz technologii testowania oprogramowania wykorzystywanych na przestrzeni ostatnich ok. 30 lat. Opisane zostały tutaj zarówno teoretyczne podstawy testowania, narzędzia i techniki analizy statycznej i dynamicznej, oceny efektywności przeprowadzanych testów, a także badania przeprowadzane w dziedzinie testowania i walidacji oprogramowania. Omawiany artykuł, w szczególności, pozytywnie oraz negatywnie aspekty poszczególnych technik oraz wskazuje przydatność określonych rozwiązań, do testowania oprogramowania różnego typu.

Ostatnią przytoczoną w ramach tego przeglądu literaturowego pozycją, dotyczącą teorii ewaluacji oprogramowania jest [?]. Pozycja ta, stanowi normę międzynarodowej organizacji normalizacyjnej (ang. International Organization for Standardization) dotyczącą weryfikacji jakości oprogramowania. Uwzględniono tu przede wszystkim znaczenie pojęć stosowanych w dziedzinie testowania oprogramowania, wprowadzono definicje dla określonych terminów oraz zjawisk występujących w ramach ewaluacji systemów, a także określono zgodność wprowadzanych przez standard konceptów, z konceptami zawartymi w standardach pochodnych. Główną częścią dokumentu, stanowi wprowadzenie szkieletu modelu jakości, uwzględniającego określone modele jakościowe, modele jakości w ujęciu, a także modele jakości produktu. Dodatkowo, przedstawiono cel oraz sposób wykorzystania modeli jakościowych, wyjaśniono również w postrzeganiu modeli jakościowych z punktu widzenia różnych interesariuszy, a także zdefiniowano relacje pomiędzy określonymi modelami. Dokument

ten, wraz z normą... ISO 9126, stanowi definicję pojęcia jakości w kontekście testowania oprogramowania.

Pozycja [?] stanowi przegląd narzędzi wykorzystywanych do testowania działania systemów komputerowych. Na początku księgi, wprowadzany jest termin zapewnienia jakości (ang. Quality Assurance), który w dzisiejszych czasach definiuje zakres odpowiedzialności osoby testującej oprogramowanie. Kolejno, przedstawiane są... kryteria sukcesu dotyczącego tworzonego systemu, a także fazy poszczególnych modeli rozwoju oprogramowania zorientowanych na procesy. Analogicznie do pozycji literaturowych przedstawionych uprzednio, w ramach tej pozycji określone zostały metryki i definicje jakości oprogramowania oraz omówiony został, proces realizacji testów. Główna część omawianego dokumentu skupiona jest wokół, narzędzi stosowanych do realizacji ewaluacji oprogramowania. Wyszczególniono tutaj narzędzie WinRunner, przedstawiając między innymi wykorzystywany w tym programie skryptowy język testów (ang. Test Script Language). Ponadto, przedstawiono architekturę oraz najważniejsze funkcjonalności narzędzi SilkTest, SQA Robot, LoadRunner, TestDirector, QuickTest Professional a także Apache JMeter. Ostatni z wymienionych programów, wykorzystywany zostanie w ramach niniejszej pracy dyplomowej, dlatego też dalszy przegląd tej pozycji literaturowej skupiony będzie na rozdziale dotyczącym właściwości tego narzędzia. Opis funkcjonalności aplikacji JMeter został, w niniejszej pozycji podzielony na sekcje związane z testowaniem rozwiązań, bazodanowych wykorzystujących interfejs JDBC (ang. Java Database Connectivity), a także sekcję dotyczącą testowania aplikacji bazujących w swoim działaniu na protokole hipertekstowym. Przedstawiono tutaj sposób tworzenia grup wątków reprezentujących użytkowników aplikacji, generowania ładania protokołu hipertekstowego, uruchomienia mechanizmu nasłuchiwanie na odpowiedź serwisu, dodawania licznika czasu, a także zapisywanie i przeglądanie rezultatów przeprowadzonego testu.

W ramach dokumentów [?] oraz [?] przedstawiono pełen zakres funkcjonalności dostępnych w ramach narzędzia Apache JMeter. Pierwsza z prac (tj. [?]), skupia się na wykorzystaniu narzędzia w celu wykonywania testów wydajności usług sieciowych, natomiast druga z pozycji (tj. [?]), przedstawia aplikację JMeter dla różnych kontekstów jej potencjalnego użycia. W obu pracach wyszczególnione zostają... podstawowe elementy, na które składa się środowisko testowe. Elementami tymi są: grupy wątków, komponenty obsługujące, kontrolery, komponenty nasłuchujące, liczniki czasu oraz asercje. Ponadto, omówiono elementy graficznego interfejsu użytkownika dla aplikacji, przedstawiono proces instalacji oraz uruchamiania narzędzia JMeter, a także zdefiniowano pojęcie planu testów. W kontekście pracy [?], poza wymienionymi uprzednio kwestiami, zobrazowany został, także proces wykonywania testu przeciwko... aplikacji dla usługi zorientowanej na serwisy (ang. Service-Oriented Application). Proces ten uwzględnia: tworzenie grupy wątków, konfigurację struktury ładania wysyłanego do usługi, uruchomienie testu, a także pozyskanie wyniku. W pracy [?] natomiast, analogiczny proces, możemy zaobserwować dla monolitycznej aplikacji internetowej oraz interfejsu programowania aplikacji. Ponadto, przedstawione zostały, zaawansowane opcje konfiguracji elementów nasłuchujących oraz liczników czasu, a także pokazany został, proces wykorzystania pośredniczącego serwera http, w celu dokumentowania realizowanych ładów.

Następne pozycje literaturowe omówione w ramach tej pracy, dotyczą... budowy oraz zasady działania internetowego protokołu hipertekstowego (ang. Hypertext Transfer Protocol), a także implementacji mechanizmu zarządzania stanem. Mechanizm ten, w związku z naturą protokołu http, nie jest w nim domyślnie realizowany.

Pozycja [?] stanowi techniczny dokument dotyczący semantyki oraz budowy internetowego protokołu hipertekstowego w wersji 1.1. Zdefiniowano w nim pojęcie zasobu

ŁEÄ...dania oraz omówiono cykl życia jego przetwarzania. Wskazano także moment, w którym zasób rekonstruowany jest przez serwer na podstawie jego efektywnego identyfikatora URI (ang. Uniform Resource Identifier). Ponadto, nakreślono pojęcie reprezentacji danych przesyłanych za pomocą... protokołu http, definiując określone pola nagłówkowe dotyczą...ce: typu danych, sposobu kodowania, języka danych, a także lokalizacji zasobu. Kolejne rozdziały dokumentu zawierają... informacje dotyczą...ce definicji dozwolonych metod protokołu http oraz znaczenia jakie te metody wprowadzają... w kontekście operacji na zasobie. W dokumencie przedstawiono także kody statusu odpowiedzi na ŁEÄ...dnie, grupując...c je w sposób semantyczny. Dla każdego z przedstawionych kodów statusu nakreślono kontekst, w jakim odpowiedź, oznaczona tym właśnie kodem, powinna być zwracana klientowi. Na końcu pracy, omówiono kwestie związane z bezpieczeństwem protokołu, takie jak: ataki bazujące...ce na wstrzykiwaniu kodu czy ochrona przed ujawnianiem informacji wrażliwych w identyfikatorach zasobów.

Pozycja [?] pozwala na poszerzenie wiedzy dotyczą...cej protokołu hipertekstowego w bardziej praktycznym kontekście. Podobnie jak w dokumencie [?], przedstawiono tutaj informacje teoretyczne dotyczą...ce architektury protokołu, definicji zasobów czy też ujednoliconego formatu ich adresowania. Ponadto, wskazano i scharakteryzowano określone typy połączeń, realizowanych z wykorzystaniem protokołu hipertekstowego. Co więcej, dla każdego z nich rozważono kwestie związane z wydajnością... połączenia pomiędzy klientem a serwerem. Kolejne rozdziały pracy [?] traktują... o identyfikacji klienta w ramach serwera, jego uwierzytelniania przed serwerem, a także szyfrowania danych przesyłanych pomiędzy tymi dwiema jednostkami. W niniejszej pracy wspomniano także o internacjonalizacji ŁEÄ...dał,, w kontekście zastosowania nagłówka „Accept-Language”. Ostatnie rozdziały dokumentu dotyczą... kwestii publikowania i dystrybucji zawartości. Wyszczególnione zostały tu takie elementy jak: web hosting, systemy publikacji treści, czy też mechanizm przekierowań, oraz równoważenie obciążenia.

Zgodnie z charakterystyką... protokołu http, realizuje on komunikację w sposób bezstanowy. Oznacza to, że domyślnie, pomiędzy klientem a serwerem nie jest utrzymywana sesja połączeniowa, a każde ŁEÄ...danie generowane przez klienta w kierunku serwera rozpatrywane jest indywidualnie. Rozwiązanie takie, pozwala na znacz...ce przyspieszenie działania protokołu hipertekstowego, a także uproszczenie jego konstrukcji. Jednakże, szczególnie w przypadku aplikacji internetowych komunikujących się z serwerem http, bezstanowy charakter protokołu bywa problematyczny w aspekcie kontekstu wysyłanych sekwencyjnie ŁEÄ...dał,,. Dlatego też, do protokołu http wprowadzono mechanizm zarządzania stanem opisany w dokumencie [?]. Dokument ten, definiuje pola nagłówkowe o nazwach „HTTP Cookie” oraz „Set-Cookie”. Pola te, mogą być używane przez serwery http w celu przechowywania stanu w ramach aplikacji klienckich, dając...c serwerom tym możliwość zarządzania, zawierając... stan sesji..., przy wykorzystaniu protokołu bezstanowego. W niniejszym dokumencie, dla obu przedstawionych pól wyszczególniono atrybuty składowe pola, a także określono znaczenie każdego z nich. Ponadto, dokument definiuje wymagania dla klienta http, dotyczą...ce możliwości wykorzystania mechanizmu zarządzania stanem. Pod uwagę wzięte zostały także kwestie bezpieczeństwa takie jak identyfikatory sesji, ślaba poufność danych, czy też zaufanie do usługi nazw domenowych w celu prawidłowego działania mechanizmu zarządzania stanem.

Ostatnia grupa pozycji literaturowych, zawartych w ramach niniejszego przeglądu literaturowego dotyczy badań, związanych z testowaniem wydajności aplikacji internetowych w środowisku rozproszonym. Pozycje przedstawione poniżej, będą stanowiły prace referencyjne względem niniejszej pracy dyplomowej.

Artykuł, [?] dotyczy porównania wydajności działania interfejsów programowania aplikacji tworzonych z wykorzystaniem platform .NET Core 3.1 oraz .NET 5. Celem powstania tego dokumentu była weryfikacja zjawiska wzrostu wydajności działania programów, tworzonych i uruchamianych z wykorzystaniem nowszej z platform firmy Microsoft. Praca ta, ma także na celu pomóc poznać odpowiedź na pytanie, czy kod źródłowy interfejsu programowania aplikacji o określonych funkcjonalnościach, a także korzystający z określonych narzędzi, powinien zostać zaktualizowany w taki sposób, aby wspierać najnowszy, stabilny wersję środowiska .NET. W ramach dokumentu, w celu realizowania pomiarów wydajności wykorzystano opisane w poprzednich rozdziałach narzędzie Apache JMeter, a także dedykowane środowisku .NET, bibliotekę BenchmarkDotNet. Kolejne rozdziały artykułu przedstawiają przygotowane środowisko testowe, plan wykonywanych testów, a także uzyskane rezultaty wraz z ich analizą. Autor pracy, zobowiązuje, wyniki sześciu testów wydajnościowych, biorąc pod uwagę proces serializacji oraz deserializacji obiektów typu JSON za pomocą bibliotek Newtonsoft.Json, a także System.Text.Json. Ponadto, przygotowany został test wyszukiwania wzorca z obszernym ciągiem tekstowym oraz test wykorzystania punktu końcowego jako klienta zewnętrznego API. Na podstawie otrzymanych rezultatów, wnioskujemy o około 24 procentowym średnim wzroście wydajności wykonywania operacji realizowanych w ramach testów. Ponadto, wykazano także dość znaczny (około 35 procentowy) średni spadek wydajności nowego rozwiązania względem poprzednika, w kontekście testów obciążeniowych.

Analogiczne badania przeprowadzono w ramach pracy [?]. W tym przypadku jednak, nie skupiały się one na aspekcie porównania technologii, a na sposobie wykonywania pomiarów, a także definiowaniu kryteriów oceny jakości. W pracy tej, interfejs programowania aplikacji zbudowany w oparciu o metodologię REST poddawany był, zmieniającym obciążeniom (tj. testy linii bazowej, testy obciążeniowe oraz testy przeciążeniowe). W czasie dokonywania ewaluacji monitorowano średni czas odpowiedzi serwera, zgodność kodów statusu zawartych w ramach uzyskiwanych odpowiedzi, informacje o zużyciu zasobów sprzętowych serwera, czy też wskaźnika satysfakcji klienta. Rezultaty przeprowadzonych badań, wykazały kluczowe znaczenie optymalizacji kodu źródłowego aplikacji, w kontekście realizacji rozbudowanych i skalowalnych usług sieciowych.

Rozdział 3

Opis problemu badawczego

W ramach niniejszego rozdziału omówiony został, podejmowany problem badawczy. W związku z jego złożonością..., autor pracy zdecydował, sięTM na podział, tego zagadnienia na określone aspekty, będące różnorodnymi względem funkcjonalności internetowych interfejsów programowania aplikacji. Na koniec tej części pracy, na podstawie sformułowanych kontekstów badawczych, zdefiniowano listęTM scenariuszy realizacji badań,,.

3.1. Przedstawienie aspektów problemu badawczego

Podjęmowany w ramach niniejszej pracy problem badawczy, dotyczy sięTM wydajności usług sieciowych, których charakterystyka wymaga zastosowanie wielu odrębnych komponentów programistycznych. Komponentami tymi, mogą być zarówno: biblioteki obsługujące proces mapowania obiektowo-relacyjnego, zastosowany wzorzec projektowy w kontekście wewnętrznej architektury API, rodzaj wykorzystywanego zewnętrznego źródła danych, czy też wybrana chmurowa platforma wdrożeniowa. W związku z mnogością... zagadnień, występujących w ramach niniejszego problemu badawczego, zdecydowano sięTM na dokonanie podziału, jego opisu w taki sposób, aby na podstawie każdego z aspektów, możliwe było zdefiniowanie wyspecyfikowanych scenariuszy realizacji badań,,.

Kiedy z aspektów problemu badawczego wymieniony poniżej, rozpatrywany będzie względem dwóch odrębnych zestawów technologii programistycznych (tj. C#/NET oraz JavaScript/NodeJS). Scenariusze badawcze, opracowane na podstawie, każdego z opisanych w tym podrozdziale aspektów, uwzględniają będące... porównanie uzyskanych wyników badań,, dla obu wymienionych rozwiązań,, informatycznych.

Wydajność i interfejsu API względem liczby żądań...dań,, generowanych przez klientów

Pierwszym z omawianych aspektów rozważanego problemu badawczego jest wpływ wydajności działania interfejsu programowania aplikacji, względem liczby klientów, którzy w sposób równoległy generują... żądań...dania w kierunku API.

Wydajność i, w kontekście tego właśnie aspektu, interpretowana jest poprzez metryki czasu odpowiedzi na żądanie, a także procentowe wartości wykorzystania zasobów sprzętowych interfejsu programowania aplikacji, takich jak centralna jednostka przetwarzania, czy też pamięć i operacyjna o dostępie swobodnym.

Zgodnie z obowiązującymi praktykami realizacji pomiarów wydajności usług sieciowych, charakterystyka ta, powinna być wyliczana w oparciu o technikę testowania rozproszonego (*ang. Distributed Testing*). Przedstawiana technika, zakłada wykorzystanie wielu odrębnych systemów informatycznych, wykonujących równolegle ewaluację obciążeniową. Wartości metryk, uzyskane dla każdej z maszyn przeprowadzających testy, powinny zostać zgrupowane, a także analizowane jako pochodzące z jednego źródła.

Zmiana wydajności, obserwowana powinna być wraz ze stałym zwiększeniem natężenia liczby klientów, a dla każdego z kolejnych przedziałów liczbowych, uwzględniających kolejne przyrosty wysyłanych pakietów, metryki wydajnościowe powinny być porównywane względem ustalonego wskaźnika referencyjnego. Kalkulacja tego wskaźnika natomiast, wykonywana jest w oparciu o ewaluację wydajności dla standardowych warunków pracy interfejsu programowania aplikacji. Przykładem takich warunków, może być realizacja testu obciążeniowego dla pojedynczej maszyny testującej.

Jednym z podstawowych, otwartych standardów, które może być wykorzystane do budowy wskaźnika referencyjnego jest APDEX (*ang. Application Performance Index*). Indeks ten, pozwala na zdefiniowanie trzech przedziałów liczbowych, określających odczucia klienta testującego oprogramowanie. Przedziały te, przedstawiane są jako progi satysfakcji, tolerancji oraz frustracji. Po wykonaniu testów odbywających się w standardowych warunkach pracy usługi sieciowej, ustalenie wartości metryk wydajnościowych dla każdego z trzech progów jest możliwe, a co za tym idzie, możliwe jest także porównanie wyników uzyskiwanych przy dowolnej liczbie klientów API, względem ustalonych progów.

Na podstawie weryfikacji wydajności interfejsu API względem liczby generowanych równolegle żądań, należy także ustalić graniczną wartość sumy maszyn klienckich, dla których interfejs programowania aplikacji jest w stanie obsługiwać zapytania, w czasie zawierającym się w przedziałach referencyjnych.

Korelacja charakterystyk wydajnościowych względem określonego zewnętrznego źródła danych

Kolejny z kontekstów problemu badawczego dotyczy wpływu zastosowania odmiennych zewnętrznych źródeł danych, na efektywność pracy usługi sieciowej jak... jest interfejs API.

Koncepcja wydajności w przypadku tego aspektu problemu, rozumiana jest w sposób analogiczny do pojęcia, wprowadzonego w ramach poprzedniej sekcji.

W związku z istotnością uwzględnienia zewnętrznych źródeł danych, jako elementów z którymi nieustannie komunikują się nowoczesne usługi sieciowe, w ramach niniejszego aspektu omawianego problemu badawczego, weryfikowany jest wpływ sposobu obsługi najpopularniejszych środków dostępnymi nieodpłatnie systemów bazodanowych, na efektywność operacji realizowanych przez poszczególne interfejsy programowania aplikacji. Problem badawczy, uwzględnia zastosowanie zarówno czterech relacyjnych systemów baz danych, jak i jednego nierelacyjnego.

Sposób obserwacji zmiany wydajności usługi sieciowej, również jest analogiczny, do tego, który został przedstawiony w ramach poprzedniego aspektu, jednakże należy zwrócić uwagę, na konieczność ponownego zdefiniowania przedziałów satysfakcji, tolerancji oraz frustracji dla wskaźnika referencyjnego. Referencja do wartości tego wskaźnika, obliczonego bez uwzględnienia połączenia z systemem bazodanowym, prowadziła do zaniżenia wartości ogólnej wydajności testowanego oprogramowania.

Kluczowym czynnikiem omawianego aspektu problemu badawczego jest zapewnienie deterministycznego charakteru stanu ŁÄ...cza sieciowego, wystÄ™pujÄ...cego pomiÄ™dzy serwerem bazodanowym a usŁ,ugÄ... interfejsu programowania aplikacji. Dlatego teŁ, w przypadku testów realizowanych w środowisku lokalnym, oba komponenty programowo-sprzÄ™towe powinny znajdowaÄ± siÄ™ w tej samej lokalizacji. Należy podkreślić jednak, że ważnym jest, aby obie usŁ,ugi informatyczne nie były uruchomione w ramach tego samego środowiska sprzÄ™towego. Dzięki temu, pozyskane wartości metryk wykorzystania zasobów sprzÄ™towych nie będą obciążone niedokładnościami. W kontekście realizacji badań, w środowisku chmurowym natomiast, ważnym jest wdrożenie zarówno API, jak i serwera bazy danych, w ramach tego samego centrum obliczeniowego, a także rozdzielenie obu usŁ,ug, pomiÄ™dzy odmienne fizyczne urządzenia. Ponadto, modyfikacji powinien ulec jeden z elementów kryterium wydajności, który definiowany jest jako czas odpowiedzi interfejsu na żądanie klienta. W związku z dyspersją geograficzną obu stron komunikacji, niemożliwym jest zachowanie przewidywalnego charakteru ŁÄ...cza sieciowego wykorzystywanego do transmisji danych. Twierdzenie to, implikuje konieczność realizacji pomiaru czasu działania usŁ,ugi sieciowej w sposób odmienny. Kryterium czasu odpowiedzi na żądanie, rozumiane w tym przypadku jest jako przedział, czasowy od momentu pozyskania żądania, do momentu zakończenia wszystkich operacji, realizowanych w kontekście tego żądania.

Efektywność realizacji złoŹonych obliczeŹ, oraz wsparcia dla programowania współbieżnego i metod asynchronicznych

Niniejszy aspekt problemu badawczego dotyczy wpływu wykorzystania, dostępnych w ramach określonego języka mechanizmów programowania współbieżnego, a także sposobu realizacji operacji asynchronicznych, na efektywność przeprowadzania kalkulacji w obrębie warstwy logiki biznesowej interfejsu programowania aplikacji.

Pojęcie efektywności dokonywanych kalkulacji postrzegane jest poprzez liczbę wykonanych iteracji głównej pętli zaimplementowanego algorytmu metaheurystycznego, rozwijającego określony problem z rodziny NP-trudnych.

W celu zachowania rzetelności badań, omawiany fragment problemu badawczego uwzględnia zastosowanie analogicznego algorytmu, realizującego te same operacje w ten sam sposób, a także rozwijającego ten sam problem obliczeniowy. W tym przypadku, zaobserwować będzie możliwość przystosowania technologii poddawanej ewaluacji, do dokonywania procesu zrównoleglania obliczeń, a także przeprowadzania wewnętrznej optymalizacji określonych linii zdefiniowanego kodu źródłowego.

Zaimplementowany algorytm metaheurystyczny, dostępny będzie bezpośrednio z poziomu punktów końcowych badanych interfejsów programowania aplikacji, a liczba iteracji głównej pętli algorytmu, mierzona będzie dla ustalonego, stałego czasu wykonania programu.

Ponadto, omawiany aspekt badawczy dotyczy także weryfikacji wydajności w kontekście zastosowania metod asynchronicznych. W związku z tymże znacząco odmienną strukturą badanych środowisk uruchomieniowych oraz języków programowania, mechanizmy obsługi operacji asynchronicznych zaimplementowane są w tych technologiach, na różnych poziomach obsługi programu. W jednym przypadku, obsługa operacji tych, jest wykonywana bezpośrednio w ramach języka programowania, natomiast w kontekście drugiej z technologii, metody których wynik nie jest dostarczany natychmiastowo, muszą zostać obsługiwane wewnętrznie środowiska uruchomieniowego.

Badanie weryfikacji wydajności dla funkcji asynchronicznych, oparte jest o odwołanie się interfejsu API, do współbieżnej pracy z nim hipertekstowej usŁ,ugi sieciowej,

pełniącą rolę pośrednika w dostępie do zdefiniowanych wewnętrznie informacji. Przedstawiona usługa sieciowa, zostanie zaimplementowana jako odrębne oprogramowanie i będzie niezależna od obu porównywanych technologii.

W kontekście drugiej z części aspektu problemu badawczego, metryką wydajności będzie czas odpowiedzi interfejsu na zapytanie.

Wpływ zastosowania wzorca projektowego podziału odpowiedzialności na efektywność realizacji operacji bazodanowych

Rozważany aspekt problemu badawczego dotyczy wpływu implementacji optymalizacji wydajnościowych w kontekście komunikacji interfejsu programowania aplikacji z zewnętrznym źródłem danych.

Wykorzystując konwencjonalną trójwarstwową architekturę interfejsu API, stosowany zostaje ten sam model danych, zarówno do operacji odczytu jak i zapisu. Powoduje to brak możliwości dostosowania modelu, względem specyfiki konkretnego rodzaju operacji. Wprowadzenie wzorca projektowego separacji zapytań, oraz komend ma na celu umożliwienie dokonania optymalizacji wydajnościowych wyizolowanych fragmentów modelu danych, a także wykorzystywanie ich tylko i wyłącznie w kontekście jednego typu operacji.

Co więcej, optymalizacja będzie wykonana nie tylko na poziomie modelu danych, ale także w ramach fizycznych struktur zawartych wewnętrznie obserwowanego systemu bazodanowego. Dlatego też, przedstawiany aspekt problemu badawczego dotyczy zastosowania zarówno odrębnych modeli danych wewnętrznie API, odrębnych struktur programistycznych obserwowanych dane modelu, jak i odseparowanych zewnętrznych źródeł danych.

Oba zastosowane źródła danych, powinny cechować się tak samo strukturą, jednakże każde z nich powinno wprowadzać charakterystyczne dla typu wykonywanych operacji, usprawnienia wydajnościowe. Ponadto, aby zachować spójność zawartości dostępnej dla klienta w ramach API, po odwołaniu się do systemu bazodanowego w celu zapisania rekordu, musi on zostać następnie zreplicowany do źródła danych obserwowanego operacji odczytu.

Wydajność, rozumiana poprzez czas odpowiedzi interfejsu API na zapytanie klienta, powinna zostać porównana z tą, wykazywaną przez usługę sieciową opierającą się na architekturze 3-warstwowej i wykorzystującą pojedyncze źródło danych.

Wpływ zastosowania mechanizmów pamięci podręcznej na wydajność interfejsów API

Problem badawczy w kontekście wykorzystania mechanizmów pamięci podręcznej, dotyczy porównania efektywności działania interfejsów programowania aplikacji implementujących standardowy oraz autorski mechanizm przechowywania rezultatów wykonanych uprzednio zapytań.

Standardowy mechanizm przechowywania danych, w ramach pamięci podręcznej uwzględnia powinien stać, czas wartości pojedynczego wpisu, a także jego uniemożliwienie w przypadku wykonania operacji modyfikujących dane. W takim przypadku, czas odpowiedzi na zapytanie powinien być zmniejszony w momencie konieczności odwołania się API do zewnętrznego źródła danych, a następnie zredukowany w przedziale czasowym, w ramach którego wpis pamięci podręcznej jest aktywny.

Zaimplementowany autorski mechanizm pamięci podręcznej wyróżnia bądzie siennym czasem wałnołci poszczególnych wpisów, który zależy bądzie od prawdopodobieństwa wywołania określonego punktu kołcowego, na podstawie informacji o liczbie historycznych wywołań. Czym większe istnieje prawdopodobieństwo ponownego wywołania punktu kołcowego, tym czas wałnołci rezultatu przechowywanego w pamięci podręcznej bądzie większy. Analogicznie do standardowego mechanizmu pamięci podręcznej, operacja modyfikacji danych unieważnia wszystkie społrząd wpisów, które odwołują... siennym do przekształconych informacji.

Celem niniejszego aspektu badawczego w ramach rozważanego problemu jest porównanie zmiany wydajności działania API, postrzeganej jako średni czas odpowiedzi na pytanie w ustalonym, stałym przedziale czasowym. Porównywane zostaną... mechanizmy standardowy oraz autorski, odrębnie dla każdej z technologii programistycznych.

Wpływ wdrożenia interfejsu API na dedykowanej platformie chmurowej na jego efektywność działania

Ostatni z przedstawianych aspektów problemu badawczego odnosi siennym do wpływu wydajności pracy interfejsu API, w zależności od rodzaju zastosowanej platformy chmurowej, na jakiej zostanie on wdrożony.

Interfejs programowania aplikacji jest usługą..., której funkcjonowanie jest nieodłącznie powiązane z serwerem internetowym. Serwer sieci Web pełni rolę warstwy opakującej, wewnątrz której działa mój interfejs API. Wdrożenie rozważanej usługi sieciowej w ramach sieci Internet, wiąże siennym w związku z tym z uruchomieniem serwera sieci web w ramach komputera eksponowanego w sieci rozległej. Należy również zauważyć konieczność zastosowania hipertekstowego serwera pocredniczego, po to, aby klient usługi, mógł, siennym z nim... komunikować z wykorzystaniem protokołu HTTP.

System informatyczny, składający siennym z przedstawionych powyżej komponentów mój zostanie uruchomiony w ramach wirtualnego serwera prywatnego, udostępnianego przez określonego dostawcę infrastruktury serwerowej. Model taki, definiowany jest jako infrastruktura w postaci usługi klienckiej (*ang. Infrastructure as a Service*). Ponadto, przygotowane oprogramowanie mój zostanie wdrożone na dedykowanej określonej technologii, platformie chmurowej. W ramach platformy tej, użytkownik, za pomocą... dostarczonego interfejsu komunikacji mój wdraża oraz konfiguruje działanie swojego oprogramowania. Taki model dostarczania zasobu z kolei, nazywany jest platformą... w postaci usługi klienckiej (*ang. Platform as a Service*).

Niniejszy aspekt problemu badawczego, dotyczy porównania wydajności API, w zależności od jego wdrożenia na generycznym wirtualnym serwerze prywatnym, a także dedykowanej platformie chmurowej, dostosowanej pod kątem określonego środowiska uruchomieniowego oraz języka programowania.

Wskaźnik ewaluacji efektywności działania interfejsu programowania aplikacji, obejmuje te same metryki, które przedstawione zostały w pierwszym społrząd omawianych aspektów problemu badawczego. Kryterium czasu odpowiedzi na pytanie... dane, musi zostać jednakże niezależne od niedeterministycznego charakteru pytania... dane internetowego, dlatego też, ten właśnie parametr, bądzie dotyczy, czasu od momentu otrzymania pytania... dane przez API, do chwili wygenerowania odpowiedzi na pytanie... dane.

3.2. Sformułowanie scenariuszy badawczych

Na podstawie przedstawionych w poprzednim podrozdziale aspektów problemu badawczego, sformułowane zostały konkretne scenariusze badawcze. W każdym ze scenariuszy, zdefiniowano zbiór czynności wykonywanych w ramach określonego badania, wyszczególniono kryteria porównawcze dla danej obserwacji, wymieniono dostosowywalne parametry badania, a także skonkretyzowano czynności, które muszą... zostać podjęte, jako warunki konieczne przed wykonaniem badania. Każdy ze scenariuszy badawczych odzwierciedlony został, w formie tabeli.

[c]||| Scenariusz badawczy - badanie przeprowadzone w kontekście systemów bazodanowych

Nazwa scenariusza badawczego:	
Wpływ zastosowanego systemu bazodanowego na efektywność działania interfejsu programowania aplikacji przy zmiennej liczbie klientów, klientów	
Topologia fizyczna środowiska badawczego:	
Konfiguracja pierwsza lokalnego środowiska badawczego ??	
Czynności implementacyjne:	
<ul style="list-style-type: none"> • Zaimplementowanie interfejsów programowania aplikacji w technologiach C#/.NET oraz NodeJS/ExpressJS. • Konfiguracja interfejsów programowania aplikacji w celu obsługi systemów bazodanowych: Microsoft SQL Server, MySQL, PostgreSQL, SQLite oraz MongoDB. • Konfiguracja topologii fizycznej środowiska badawczego. • Konfiguracja narzędzia do wykonywania testów wydajnościowych. 	
Czynności badawcze:	
<ul style="list-style-type: none"> • Realizacja testów wydajnościowych z uwzględnieniem zmiennej liczby klientów. • Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych. • Dostosowywanie wartości parametrów przeprowadzanego badania. • Analiza statystyczna otrzymanych wyników. 	
Warunki początkowe podjęcia czynności badawczych:	
Przed realizacją testów wydajnościowych zagwarantowana zostanie poprawność działania całego z interfejsów programowania aplikacji poprzez wykonanie ewaluacji funkcjonalnej.	

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej środowiska badawczego, urządzenia klienckie będą wysyłać dane protokołu hipertekstowego w kierunku interfejsu programowania aplikacji. Liczba danych, tych będzie sukcesywnie zwiększana, poprzez uruchamianie kolejnych równoległe pracujących wirtualnych oprogramowania testowego. Dla każdej wartości liczby urządzeń, klienckich, badanie zostanie wykonane dziesięciokrotnie, a uzyskane wyniki zostaną następnie uśrednione. Odpowiednie przedziały wartości omawianego parametru wyznaczać będą granice pomiędzy testami linii bazowej, obciążeniami, a także przeciwnymi. Badanie zostanie powtórzone dla każdego z rozważanych systemów bazodanowych w obrębie obu porównywanych technologii programistycznych. Po zgromadzeniu wyników badań, przeprowadzone zostaną parowe testy statystyczne wykazujące istotność różnic pomiarowych.

Kryteria porównawcze:

- Czas odpowiedzi interfejsu programowania aplikacji na dane klienta.
- Procent wykorzystania centralnej jednostki przetwarzania serwera, na którym uruchomiony został interfejs programowania aplikacji.
- Ilość wykorzystanej pamięci operacyjnej serwera, na którym uruchomiony został interfejs programowania aplikacji.

Parametry badania:

- Liczba klientów równoległe wysyłających dane.
- Technologia programistyczna zastosowana do implementacji interfejsu programowania aplikacji.
- Rodzaj systemu bazodanowego komunikującego się z interfejsem programowania aplikacji.

[c]||| Scenariusz badawczy - badanie przeprowadzone w kontekście realizacji operacji współdzielonych

Nazwa scenariusza badawczego:

Wpływ zastosowanej technologii programistycznej na wydajność realizacji operacji współdzielonych

Topologia fizyczna środowiska badawczego:

Konfiguracja druga lokalnego środowiska badawczego ??

Czynności implementacyjne:

- Zaimplementowanie genetycznego algorytmu metaheurystycznego dla symetrycznego problemu komiwojaźnika w językach programowania C# oraz JavaScript.
- Zaimplementowanie mechanizmów pomiaru czasu wykonania algorytmu
- Konfiguracja interfejsów programowania aplikacji w celu obsługi algorytmów metaheurystycznych z poziomu punktu końcowego API.
- Konfiguracja topologii fizycznej środowiska badawczego.
- Konfiguracja narzędzia do wykonywania testów wydajnościowych.

Czynności badawcze:

- Realizacja testów wydajnościowych dla porównywanych technologii programistycznych.
- Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych.
- Dostosowywanie wartości parametrów przeprowadzanego badania.
- Analiza statystyczna otrzymanych wyników.

Warunki początkowe podjęcia czynności badawczych:

Przed realizacją testów wydajnościowych zagwarantowana zostanie poprawność działania każdego z algorytmów metaheurystycznych. Ponadto, kod źródłowy programów implementujących algorytmy zostanie przekształcony w taki sposób, aby niezależnie od języka programowania, realizować operacje w sposób analogiczny.

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej środowiska badawczego, pojedyncze uruchomienie klienckie będzie wysyłać z ustaloną częstotliwością, dane wykonania algorytmu. Po odebraniu zapytania od klienta, algorytm będzie uruchamiany, a czas trwania obliczeń, będzie zawsze wartości stałą. W trakcie wykonywanych kalkulacji zliczana będzie liczba iteracji głównej pętli kodu algorytmu. Liczba ta, stanowi będzie kryterium porównawcze. Dla każdej z porównywanych technologii programistycznych wykonana zostanie seria dwudziestu cyklicznych uruchomień, klienta. Po zgromadzeniu wyników badań, przeprowadzone zostaną parowe testy statystyczne wykazujące istotność różnic pomiarowych.

Kryteria porównawcze:

- Liczba iteracji głównej pętli algorytmu metaheurystycznego.

Parametry badania:

- Człowiek: generowanie i...dał, protokołu hipertekstowego.
- Technologia programistyczna zastosowana do implementacji interfejsu programowania aplikacji.

[c]||||| Scenariusz badawczy - badanie przeprowadzone w kontekście obsługi operacji asynchronicznych

Nazwa scenariusza badawczego:

Wpływ zastosowanej technologii programistycznej na efektywność obsługi operacji asynchronicznych

Topologia fizyczna środowiska badawczego:

Konfiguracja trzecia lokalnego środowiska badawczego ??

Czynności implementacyjne:

- Zaimplementowanie interfejsów programowania aplikacji w technologiach C#/NET, NodeJS/ExpressJS oraz Python/Flask.
- Zdefiniowanie punktów końcowych odpowiedzialnych za komunikację badanego interfejsu z zewnętrznym API.
- Konfiguracja topologii fizycznej środowiska badawczego.
- Konfiguracja narzędzia do wykonywania testów wydajnościowych.

Czynności badawcze:

- Realizacja testów wydajnościowych dla porównywanych technologii programistycznych.
- Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych.
- Dostosowywanie wartości parametrów przeprowadzanego badania.
- Analiza statystyczna otrzymanych wyników.

Warunki początkowe podjęcia czynności badawczych:

Przed realizacją testów wydajnościowych zagwarantowane zostanie poprawne połączenie pomiędzy każdym z badanych interfejsów programowania aplikacji a zewnętrzną usługą sieciową. Co więcej, zweryfikowana zostanie poprawność implementacji funkcjonalności z zewnętrznej usługi sieciowej (tj. interfejsu API zaimplementowanego w języku Python)

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej środowiska badawczego, urządzenia klienckie będą równolegle wysyłać żądania http w kierunku interfejsu programowania aplikacji. Po odebraniu zapytania od klienta, interfejs API, zdecyduje się, gdzie ze znajdujących się w obrębie sieci lokalnej zewnętrznych usług sieciowych. Taki rodzaj operacji, kiedy strona wywołująca zleca wykonanie zadania, a odpowiedź na to zlecenie przyjdzie w dowolnym momencie, nazywamy operacją asynchroniczną. W ramach punktu końcowego ewaluowanego API, wykonane zostaną cztery operacje asynchroniczne dotyczące różnych sposobów operowania na danych. Po uzyskaniu odpowiedzi na wszystkie z czterech operacji, interfejs programowania aplikacji będzie zwracał, odpowiedź zawierającą informację o stopniu poprawności wykonania zleconych operacji, a także czas odpowiedzi na żądanie. Tak zdefiniowane badanie, będzie wykonywane dla określonej liczby klientów równolegle generujących żądania. Po zgromadzeniu wyników badań, przeprowadzone zostaną parowe testy statystyczne wykazujące istotność różnic pomiarowych.

Kryteria porównawcze:	
------------------------------	--

- | | |
|--|--|
| <ul style="list-style-type: none"> • Czas odpowiedzi interfejsu programowania aplikacji na żądanie klienta. • Procent poprawności wykonania zleconych operacji asynchronicznych. | |
|--|--|

Parametry badania:	
---------------------------	--

- | | |
|--|--|
| <ul style="list-style-type: none"> • Liczba klientów równolegle wysyłających żądań. • Technologia programistyczna zastosowana do implementacji interfejsu programowania aplikacji. | |
|--|--|

[c]||| Scenariusz badawczy - badanie przeprowadzone w kontekście zastosowania wzorca projektowego CQRS

Nazwa scenariusza badawczego:	
--------------------------------------	--

Wpływ implementacji wzorca projektowego podziału odpowiedzialności na wydajność obsługi żądań klienta	
---	--

Topologia fizyczna środowiska badawczego:	
--	--

Konfiguracja pierwsza lokalnego środowiska badawczego ??	
--	--

Czynności implementacyjne:	
-----------------------------------	--

- Zaimplementowanie interfejsów programowania aplikacji wykorzystujących wzorce projektowy CQRS w technologiach C#/ .NET oraz NodeJS/ExpressJS.
- Konfiguracja interfejsów programowania aplikacji w celu obsługi systemów bazodanowych: Microsoft SQL Server, MySQL, PostgreSQL, SQLite oraz MongoDB.
- Wprowadzenie usprawnień, wydajnościowych dotyczących odczytu i zapisu dla każdego z systemów bazodanowych
- Połączenie interfejsu API z dwoma systemami bazodanowymi tego samego typu i konfiguracja automatycznej replikacji danych w momencie ich zapisu
- Konfiguracja topologii fizycznej środowiska badawczego.
- Konfiguracja narzędzia do wykonywania testów wydajnościowych.

Czynności badawcze:

- Realizacja testów wydajnościowych z uwzględnieniem zmienności liczby klientów.
- Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych.
- Dostosowywanie wartości parametrów przeprowadzanego badania.
- Analiza statystyczna otrzymanych wyników.

Warunki początkowe podjęcia czynności badawczych:

Przed realizacją testów wydajnościowych zagwarantowana zostanie poprawność działania każdego z interfejsów programowania aplikacji poprzez wykonanie ewaluacji funkcjonalnej.

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej środowiska badawczego, uruchomienia klientów będących wysyłającą i odbierającą protokółu hipertekstowego w kierunku interfejsu programowania aplikacji. Liczba elementów, tych będzie sukcesywnie zwiększana, poprzez uruchamianie kolejnych równoległe pracujących wariantów oprogramowania testowego. Dla każdej wartości liczby uruchomionych, klientów, badanie zostanie wykonane dziesięciokrotnie, a uzyskane wyniki zostaną następnie uśrednione. Odpowiednie przedziały wartości omawianego parametru wyznaczą granice pomiędzy testami linii bazowej, obciążeniami, a także przeciwnymi. Badanie zostanie powtórzone dla każdego z rozważanych systemów bazodanowych w obrębie obu porównywanych technologii programistycznych. Po zgromadzeniu wyników badań, przeprowadzone zostaną parowe testy statystyczne wskazujące istotność różnic pomiarowych. Obserwacje uzyskane w ramach niniejszego badania, porównane zostaną z tymi, ustalonymi na podstawie pierwszego scenariusza badawczego ??.

Kryteria porównawcze:

- Czas odpowiedzi interfejsu programowania aplikacji na ŁŁÄ...danie klienta.
- Procent wykorzystania centralnej jednostki przetwarzania serwera, na którym uruchomiony został, interfejs programowania aplikacji.
- IloŁÄ± wykorzystanej pamiÄ™ci operacyjnej serwera, na którym uruchomiony został, interfejs programowania aplikacji.

Parametry badania:

- Liczba klientŲw równoległe wysył,ajÄ...cych ŁŁÄ...dania.
- Technologia programistyczna zastosowana do implementacji interfejsu programowania aplikacji.
- Rodzaj systemu bazodanowego komunikujÄ...cego siÄ™ z interfejsem programowania aplikacji.

[c]||||| Scenariusz badawczy - badanie przeprowadzone w kontekŁcie wykorzystania mechanizmŲw pamiÄ™ci podrÄ™cznej

Nazwa scenariusza badawczego:

PorŲwnanie efektywnoŁci obsŁugi ŁŁÄ...daŁ,, klienckich w staŁym czasie, uwzglÄ™niajÄ...c odmienne implementacje mechanizmŲw pamiÄ™ci podrÄ™cznej

Topologia fizyczna Łrodowiska badawczego:

Konfiguracja druga lokalnego Łrodowiska badawczego ??

CzynnoŁci implementacyjne:

- Zaimplementowanie interfejsŲw programowania aplikacji w technologiach C#/NET oraz NodeJS/ExpressJS.
- Zaimplementowanie mechanizmŲw pamiÄ™ci podrÄ™cznej (rozwiÄ...zanie autorskie oraz standardowe) w oparciu o bibliotekÄ™ Redis.
- Konfiguracja interfejsŲw programowania aplikacji w celu obsŁugi systemu bazodanowego MySQL.
- Konfiguracja topologii fizycznej Łrodowiska badawczego.
- Konfiguracja narzÄ™dzia do wykonywania testŲw wydajnoŁciowych.

CzynnoŁci badawcze:

- Realizacja testów wydajnościowych z uwzględnieniem zmienności czasu trwania testu.
- Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych.
- Dostosowywanie wartości parametrów przeprowadzanego badania.
- Analiza statystyczna otrzymanych wyników.

Warunki początkowe podjęcia czynności badawczych:

Przed realizacją testów wydajnościowych zagwarantowana zostanie poprawność działania całego z interfejsów programowania aplikacji, a także całego z zaimplementowanych mechanizmów pamięci podręcznej. Do autorskiego mechanizmu pamięci cache dostarczone zostaną przygotowane informacje historyczne, których zawartość dotyczy czystości wywołania określonych punktów kodowych interfejsu programowania aplikacji.

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej środowiska badawczego, pojedyncze uruchomienie klienckie będzie w sposób sekwencyjny wysyłać dane do zdefiniowanego zbioru punktów kodowych. Interfejsy programowania aplikacji, które implementowałyby określone mechanizmy pamięci podręcznej, zwracać będą odpowiedź na dane, której wartość różniłaby się od czasu odpowiedzi systemu bazodanowego, czy też pozyskać dane z cache. Pierwszy z systemów pamięci podręcznej uwzględniłby stały czas wartości wpisu, natomiast drugi system (tj. system autorski) wyliczałby czas wartości na podstawie czystości odwołania, do punktu kodowego. Zebrane rezultaty rozpatrywane będą dla różnych przedziałów czasowych. W każdym z przedziałów, wydajność systemu pamięci podręcznej determinowałaby średni czas odpowiedzi na dane oraz liczba odwołań, do systemu bazodanowego. Badanie zostanie powtórzone w obrębie obu porównywanych technologii programistycznych. Po zgromadzeniu wyników badań, przeprowadzone zostaną parowe testy statystyczne wykazujące istotność różnic pomiarowych.

Kryteria porównawcze:

- Średni czas odpowiedzi interfejsu programowania aplikacji na dane klienta w określonym przedziale czasowym.
- Liczba odwołań, interfejsu programowania aplikacji do systemu bazodanowego w określonym przedziale czasowym.

Parametry badania:

- Długość przedziału czasowego.
- Technologia programistyczna zastosowana do implementacji interfejsu programowania aplikacji.
- Rodzaj zaimplementowanego mechanizmu przechowywania danych w pamięci podręcznej.

[c]||||| Scenariusz badawczy - badanie przeprowadzone w kontekście wdrażania oprogramowania na platformach chmurowych

Nazwa scenariusza badawczego:

Zmiennej wydajności interfejsu API wdrożonego na generycznej oraz dedykowanej platformie chmurowej

Topologia fizyczna środowiska badawczego:

Konfiguracja pierwsza rozproszonego środowiska badawczego ??

Czynności implementacyjne:

- Zaimplementowanie interfejsów programowania aplikacji w technologiach C#/.NET oraz NodeJS/ExpressJS.
- Konfiguracja interfejsów programowania aplikacji w celu obsługi dedykowanych systemów bazodanowych (rodzaj systemu bazodanowego będzie zależny od wyników badań, uzyskanych poprzez realizację scenariusza badawczego ??).
- Wdrożenie interfejsów programowania aplikacji na wirtualnych serwerach prywatnych
- Wdrożenie interfejsów programowania aplikacji na dedykowanych względem określonej technologii platformach chmurowych
- Implementacja mechanizmów pomiaru czasu wykonywanych operacji wewnętrznych interfejsów API
- Konfiguracja topologii fizycznej rozproszonego środowiska badawczego.
- Konfiguracja narzędzia do wykonywania testów wydajnościowych.

Czynności badawcze:

- Realizacja testów wydajnościowych z uwzględnieniem zmiennej liczby klientów API.
- Obserwacja oraz gromadzenie wartości pomiarowych dotyczących kryteriów porównawczych.
- Dostosowywanie wartości parametrów przeprowadzanego badania.
- Analiza statystyczna otrzymanych wyników.

Warunki początkowe podjęcia czynności badawczych:

Przed realizacją... testów wydajnościowych zagwarantowana zostanie poprawność działania całego z interfejsów programowania aplikacji poprzez wykonanie ewaluacji funkcjonalnej. Ponadto, zweryfikowana zostanie dostępność całego z platform chmurowych w czasie wykonywania testów.

Opis scenariusza badawczego:

Po wykonaniu konfiguracji topologii fizycznej rozproszonego środowiska badawczego, uruchomienia klientki będącej równolegle generowaną... dania hipertekstowe w kierunku interfejsu programowania aplikacji. Poszczególne interfejsy API, analogicznie do obsługiwanego przez niego systemu bazodanowego, wdrożone zostaną w określonym środowisku chmurowym. Od momentu uzyskania... dania od aplikacji klientkiej, liczony będzie czas wykonywania operacji wewnątrz API. Punktem końcowym czasu realizacji obliczeń, będzie chwila wygenerowania odpowiedzi na... danie. Czas odpowiedzi liczony w ten sposób, będzie odpowiednio pomniejszony względem standardowego czasu odpowiedzi na... danie i nie będzie on uwzględniał, faktu dostarczenia oraz zwrócenia wiadomości http w ramach sieci rozległej. Badanie zostanie powtórzone w obrębie obu porównywanych technologii programistycznych, uwzględniając dwa systemy bazodanowe, dla których efektywność wynikać... ca ze scenariusza badawczego ?? okazała się największa. Uwzględniona zostanie ponadto zmienna liczba równolegle generowanych... dań. Po zgromadzeniu wyników badań, przeprowadzone zostaną... parowe testy statystyczne wskazujące istotność różnic pomiarowych.

Kryteria porównawcze:

- Czas obsługi... dania wewnątrz interfejsu programowania aplikacji.

Parametry badania:

- Liczba klientów równolegle generujących... dania.
- Wykorzystywana platforma chmurowa oraz wirtualny serwer prywatny.

Rozdział 4

Implementacja Środowiska badawczego

W niniejszym rozdziale przedstawiono proces implementacji Środowiska badawczego pod kątem realizacji zdefiniowanych uprzednio scenariuszy badawczych. Spośród składowych omawianego procesu wyróżniły należały: przygotowanie odmian topologii fizycznych Środowiska w zależności od scenariusza badawczego, budowa™ interfejsów programowania aplikacji z wykorzystaniem porównywanych technologii, czy też zastosowanie mechanizmów pozwalających na weryfikację™ działania API w aspekcie programowania współdzielonego. Ponadto, zaprezentowana została konfiguracja dedykowanych porównywanym technologiom platform chmurowych, a także struktura elementów składowych siatki™ na plan testowy, identyfikujący™ przeprowadzoną™ ewaluację™ w ramach narzędzia™ przeznaczonego do realizacji badań™.

4.1. Realizacja topologii fizycznych

W zależności od scenariusza badawczego, wykorzystywanego w kontekście przeprowadzonych ewaluacji, zbudowane zostały cztery odmienne warianty topologii fizycznych. Trzy spośród nich, tzn. siatki™ badań™, wykonywanych w Środowisku lokalnym, natomiast czwarta z topologii, związana jest z obserwacją™ funkcjonowania interfejsów programowania aplikacji uruchomionych w obrębie™ określonych platform chmurowych.

W odniesieniu do każdej z topologii zbudowanej w Środowisku lokalnym, zauważamy™ należały fakt zastosowania techniki rozproszonego testowania (*ang. Distributed Testing*), a także jasny podział, odpowiedzialności w kontekście wszystkich wykorzystywanych urządzeń™. Procedura zbierania obserwacji za każdym razem realizowana jest wewnątrz odpowiednio dostosowanej lokalnej sieci komputerowej cechującej siatkę™ brakiem dostępu™ do sieci Internet. Ponadto, w obszarze połączonych ze sobą™ urządzeń™, systemu komputerowego, dezaktywowane zostały, protokoły i usługi sieciowe generujące™ cykliczne komunikaty rozgłoszeniowe. Dzięki™ temu, wyeliminowano błędny™ pomiarowe o charakterze niedeterministycznym.

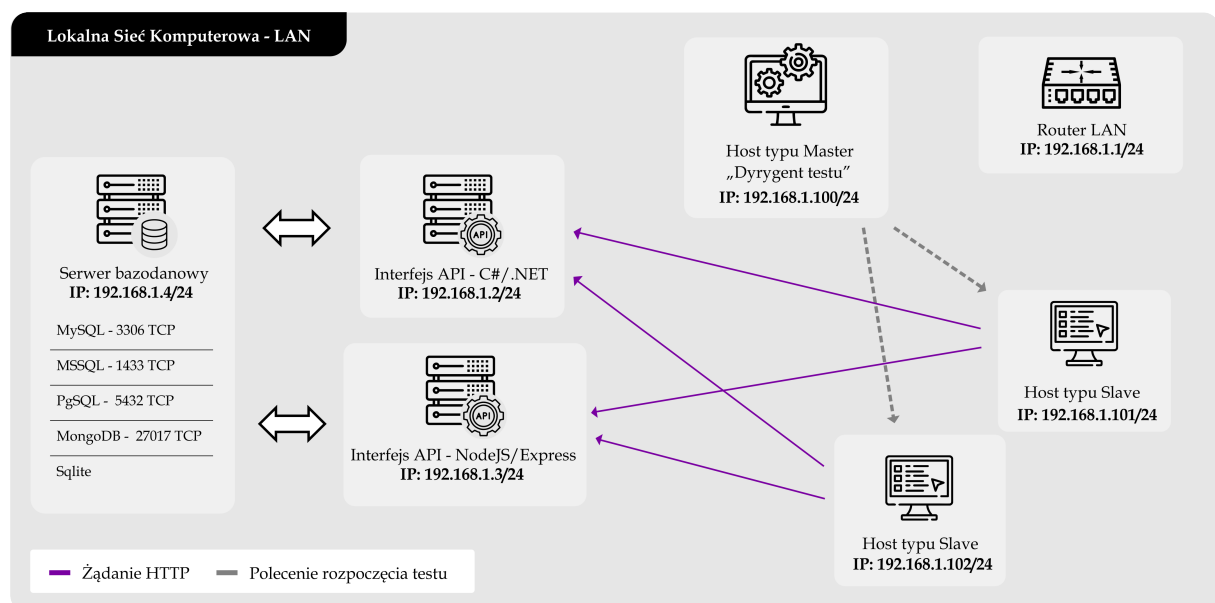
Odwólując™ do topologii sieciowej zbudowanej w Środowisku rozległym, zauważamy™ należały odmienny sposób pozyskiwania informacji o czasie przetwarzania pojedynczego Ładania. W tym przypadku, wykorzystywane narzędzie™ pomiarowe służy™ jako generator Ładania™, jednakże™ czasy odpowiedzi na Ładanie™, zwrócone przez to narzędzie™ nie mogą™ być™ brane pod uwagę™. Informacja o czasie przetwarzania Ładania™ zostaje dostarczana bezpośrednio z interfejsu programowania aplikacji.

Konfiguracja pierwsza lokalnej topologii fizycznej środowiska badawczego

Pierwsza spośród lokalnych topologii fizycznych środowiska badawczego zastosowana została, a w celu badania wpływu wykorzystania odmiennych systemów bazodanowych na wydajność interfejsów programowania aplikacji, a także w kontekście obserwacji istotności implementacji wzorca podziału odpowiedzialności.

W ramach niniejszej topologii, wyróżniono dwa interfejsy programowania aplikacji (tj. utworzone z wykorzystaniem technologii C#/.NET oraz NodeJS/Express), komunikujące się z jednym z pięciu serwerów bazodanowych (tj. MySQL Server, PostgreSQL Server, Microsoft SQL Server, SQLite oraz MongoDB). W określonym momencie czasu, utrzymywane jest tylko jedno aktywne połączenie pomiędzy jednym z API a jednym z serwerów bazodanowych. Ponadto, wewnątrz lokalnej sieci komputerowej, wyróżniono urządzenie komunikacyjne, którym jest router, a także trzy urządzenia końcowe. Pierwszy z hostów pełni rolę "dyrygenta testu", który dostarcza informacje o konfiguracji testowej bezpośrednio do dwóch pozostałych urządzeń końcowych. Te urządzenia z kolei, odpowiedzialne są za generowanie ładunków, zgodnie ze zdefiniowanym natężeniem, czystości i czasu trwania ewaluacji. Każde z oddzielnych urządzeń, fizycznie połączone jest z urządzeniem komunikacyjnym poprzez łącz przewodowe, o tej samej przepustowości (tj. 1Gb/s), a także identycznej kategorii przewodu (tj. kategoria 6).

Na ilustracji ?? przedstawiono schemat pierwszego wariantu lokalnej topologii fizycznej środowiska badawczego.

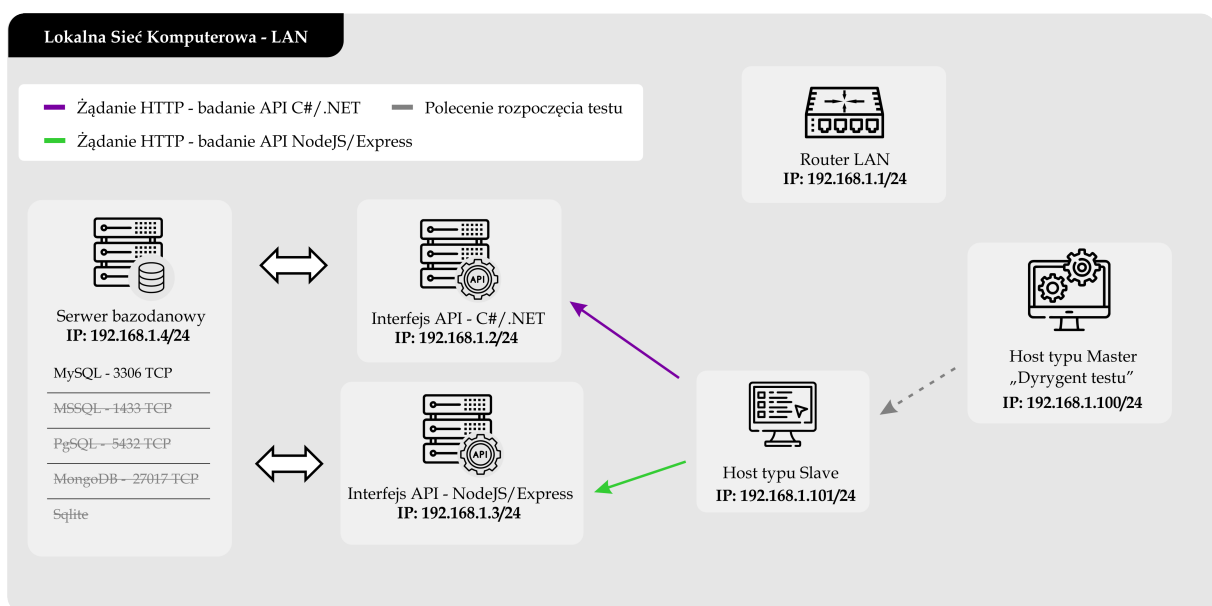


Rys. 4.1: Konfiguracja pierwsza lokalnej topologii fizycznej środowiska badawczego

Konfiguracja druga lokalnej topologii fizycznej środowiska badawczego

Druga topologia fizyczna środowiska badawczego, dotycząca ewaluacji w obrębie lokalnej sieci komputerowej, zbudowana została, a na potrzeby badań, zaimplementowanych mechanizmów programowania współbieżnego, a także przechowywania odpowiedzi na ładunek w ramach pamięci podręcznej.

W omawianej topologii wskazane zostały dwa interfejsy programowania aplikacji, których implementacja dokonana została, a w porównywanych dwóch technologiach programistycznych. Obie usługi sieciowe, komunikujące się z pojedynczą instancją serwera bazodanowego - w przypadku badania pamięci cache, bądź też nie odwołując się do niego wcale - w przypadku badania efektywności operacji współdzielonych. Spośród urządzeń, które, komunikacyjnych, wykorzystywanych do przeprowadzenia ewaluacji, wyróżniliśmy jedno urządzenie typu Master (tzw. Dyrygent testu), a także jedno hosta typu Slave (tzw. generator Ładów). Pierwszy z komputerów ma za zadanie przechowywać konfigurację wykonywanego badania, a także dostarczać komendy związane z rozpoczęciem i charakterystyką testu. Drugi host pełni rolę maszyny wytwarzającej i wysyłającej Ładania protokołu hipertekstowego, zgodnie z koncepcją nakreśloną przez uzyskany plan ewaluacji. Wszystkie urządzenia znajdują się w obszarze pojedynczej, przewodowej lokalnej sieci komputerowej. Analogicznie do konfiguracji pierwszej, każde przewodowe charakteryzuje się tym samym standardem oraz przepustowością.



Rys. 4.2: Konfiguracja druga lokalnej topologii fizycznej środowiska badawczego

Konfiguracja trzecia lokalnej topologii fizycznej środowiska badawczego

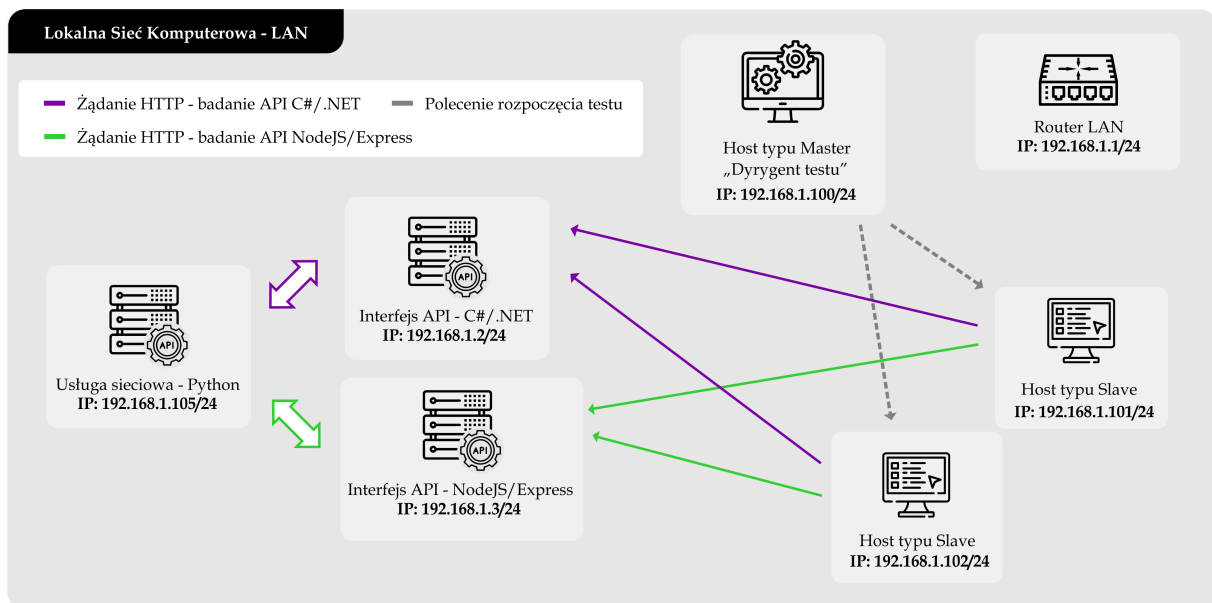
Ostatnia z lokalnych topologii fizycznych środowiska badawczego przystosowana została, a w celu umożliwienia realizacji badań, dotyczących wydajności obsługi operacji asynchronicznych.

W schemacie tym, zauważyć można występowanie trzech interfejsów programowania aplikacji. Analogicznie do topologii opisanych powyżej, dwa spośród trzech API zaimplementowane są w technologiach badawczych przedmiotem analizy tej pracy. Trzecia usługa sieciowa, służy do udostępniania zgromadzonych w niej danych, w związku z czym posiada ona tylko i wyłącznie punkty końcowe obsługiwane z wykorzystaniem metody GET. Punkty styku ostatniego z interfejsów dostarczają funkcjonalności pobierania danych o zróżnicowanym rozmiarze.

Ponadto, zbieżnie do konfiguracji pierwszej lokalnego środowiska badawczego, wyszczególnimy dwa urządzenia końcowe w roli generatorów Ładów, oraz jednego hosta działającego w trybie "Dyrygenta testu". Należy podkreślić, że ewa-

luacje dotyczÄ...ce kaÅŁdej z technologii, zarÅŁwno w scenariuszach badawczych wykorzystujÄ...cych tÄ™, jaki i pozostaÅŁe topologie, wykonywane sÄ... w odrÄ™bnych chwilach czasu. Implikuje to fakt, ÅŁe poÅŁÄ...czenie pomiÄ™dzy hostem badajÄ...cym, interfejsem badanym, a takÅŁe interfejsem pomocniczym jest aktywne tylko dla aktualnie badanego rozwiÄ...zania technologicznego.

Poza interfejsami programowania aplikacji oraz urzÄ...dzeniami koÅŁcowymi wskaÅŁaÄ... naleÅŁy urzÄ...dzenie sieciowe, jakim jest przewodowy router LAN, ktÅŁry ÅŁÄ...czy wszystkie elementy topologii w ramach pojedynczej sieci LAN. Standard oraz przepustowoÅŁÄ... wykorzystywanych ÅŁÄ...czy pozostaje niezmienna wzglÄ™dem pierwszej oraz drugiej z topologii fizycznych.



Rys. 4.3: Konfiguracja trzecia lokalnej topologii fizycznej Środowiska badawczego

Konfiguracja pierwsza rozległej topologii fizycznej środowiska badawczego

4.2. Budowa interfejsów programowania aplikacji

Interfejs API realizujący operacje CRUD stworzony z wykorzystaniem technologii C#/.NET

Interfejs API realizujący operacje CRUD stworzony z wykorzystaniem technologii NodeJS/Express

Algorytm metaheurystyczny dla symetrycznego problemu komiwojałera dostępnego z poziomu punktu końcowego

Mechanizm obsługi pamięci podręcznej z uwzględnieniem czystości wywołania punktów końcowych

Implementacja wzorca projektowego CQRS z uwzględnieniem replikacji danych pomiędzy dwoma źródłami

4.3. Konfiguracja generycznych oraz dedykowanych platform chmurowych

4.4. Konfiguracja narzędzia do realizacji badań,

Rozdział 5

Przeprowadzone badania

Rozdział 6

Podsumowanie

6.1. Wnioski z przeprowadzonych badań,,

6.2. Perspektywy rozwoju badań,,

Dodatek A

Instrukcja wdroeniowa

Jeli praca skoczya si wykonaniem jakiego oprogramowania, to w dodatku powinna pojawi si instrukcja wdroeniowa (o tym jak skompilowa/zainstalowa to oprogramowanie). Przydaoby si rwnie krtkie how to (jak uruchomi system i co w nim robi – zademonstrowane na jakim najproszym przypadku uycia). Mona z tego zrobi osobny dodatek,

Dodatek B

Opis zaczonej pyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartosci zaczonej pyty. Naley wymieni, co zawiera.