UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE
Wydział Matematyki, Fizyki i Informatyki

Kierunek: Informatyka

Adam Wadowski

nr albumu: 303841

Porównanie wydajności relacyjnych i nierelacyjnych baz danych: Firebase i Java Derby, na przykładzie aplikacji wyszukującej gry według zadanych preferencji w technologii Spring Boot

Performance comparison of relative and non relative databases: Firebase and Java Derby, on example of application searching games according to given preferences in Spring Boot technology

Praca licencjacka

napisana w Katedrze Neuroinformatki i Inżynierii Biomedycznej pod kierunkiem dr Anny Gajos-Balińskiej

Lublin rok 2023

Spis treści

Wstęp

DO WYCIĘCIA/PRZEROBIENIA

Wielu programistów na początku budowy projektu zderza się z problemem wyboru najbardziej wydajnej bazy danych. Programiści oczekują często jak najmniejszego czasu oczekiwania na wołane przez nich zapytania, łatwej obsługi oraz dobrej integracji z ich aplikacjami, które tworzą do użytku masowego przez wielu użytkowników. Bazy danych zwiększają swoją objętość, a ich przeszukiwanie staje się coraz dłuższe. Osobiście borykałem się z tym wyborem już wiele razy, podczas budowy aplikacji w trakcie studiów, życia prywatnego, a także zawodowego. Najczęstszym problemem pojawiącym się na samym początku jest rozpoznanie, która baza danych będzie najbardziej efektywna dla naszej aplikacji, czy będzie to baza relacyjna, czy może nierelacyjna. Jednakże na początku drogi programisty nikt nie jest w stanie dokonać najlepszego wyboru bez odpowiedniego rozeznania i pomocy od osób trzecich.

Celem przedstawionej pracy jest pokazanie różnic w tworzeniu, implementacji oraz wydajności bazy relacyjnej i nierelacyjnej, na podstawie aplikacji webowej pisanej z użyciem techonologii Spring Boot.

Rozdział 1

Tworzenie aplikacji webowej

1.1. Problem wyboru technologii

Wielu programistów podczas tworzenia nowej aplikacji, zastanawia się na początku jaki język programowania wybrać, aby rozwiązać problem jak najlepiej. Obecnie do tworzenia aplikacji webowych używamy wiele technologii.

1.1.1. Popularne języki i narzędzia

Najpopularniejsze technologie to JavaScript, HTML/CSS, Python, SQL oraz Java. Aby przełożyć kod na coś widocznego, potrzebujemy kompilatora, który różni się w zależności od używanego języka. Dla przykładu używając języka Java naszym kompilatorem będzie *javac*. Korzystając z języka jakim jest Java, możemy używać wielu frameworków. Do stworzenia aplikacji internetowej najbardziej popularnym wyborem wśród programistów jest Spring Boot. Głównym powodem wyboru wyżej wymienionego narzędzia jest szybkość budowania aplikacji. W kilka minut możemy wygenerować działającą aplikację i opublikować ją na prywatnym serwerze. Wraz z szybkością idą: obszerna i ogólnodostępna dokumentacja, która potrafi poprowadzić krok po kroku nawet największego laika, możliwość prostego testowania oraz modularyzacja naszego projektu, dzięki której możemy w łatwy sposób zmieniać zależności dla konkretnego kontekstu.

1.1.2. Interakcja z użytkownikiem

Przy tworzeniu aplikacji webowej język Java służy nam jedynie do wystawienia tzw. kontrolerów, które pod określonym adresem kryją swoje funkcjonalności. Do odczytu oraz użycia takiego kontrolera potrzebny nam drugi język programowania, który stworzy widok dla danych pobranych z konkretnej ścieżki. Najpopularniejszym językiem stosowanym w tym momencie do współpracy z Javą jest Angular, który jest prężnie rozwijany przez firmę Google. Jest to framework napisany w języku TypeScript, którego zaletami są między innymi szybkość użycia, wydajność oraz właśnie wymieniona wyżej budowa z TypeScript. Pisanie w tym frameworku polega głównie na tworzeniu komponentów, które są modyfikowane zależnie od danych oraz logiki programu.

1.1.3. Bazy danych w aplikacjach webowych

W ten sposób jesteśmy w stanie stworzyć stronę, która nie ma możliwości zbierania danych w żaden sposób, do gromadzenia danych potrzebujemy więc bazy. Bazy danych dzielą się na wiele kategorii. Z uwagi na miejsce inicjalizacji możemy rozmawiać o lokalnych bazach danych lub typu klient-serwer. W pierwszym rodzaju są to najprostsze zbiory, które są gromadzone na jednym komputerze, a wszelkie zmiany będzie nanosił użytkownik. Drugi rodzaj, który jest przechowywany w zasobach serwera, jest traktowany jako osobny komputer, dostęp do niego możemy uzyskać poprzez połączenie sieciowe. Ze względu na architekturę, wyróżniamy dwa typy: jednowarstwowe oraz dwuwarstwowe. Bazy jednowarstwowe wykonują zmiany od razu, w przeciwieństwie do drugiego typu, w którym połączenie z serwerem odbywa się za pomocą specjalnego sterownika,

a kontrolowanie danych zależy od klienta. Ostatnim podziałem baz danych jest podział względem struktur danych, których używają. Jedne zwane prostymi lub kartotekowymi określają każdą tablicę jako osobny dokument, przez co nie ma między nimi żadnego połączenia. Drugie zwane relacyjnymi bazami danych określają wiele tablic, które mogą się łączyć. Na przykład dla książki możemy wyświetlić jej zawartość ale również z innej tabeli możemy wyświetlić bibliotekę, w której się znajduje. Do takiej operacji potrzebujemy unikalnego kodu oraz relacji między tabelami. Trzecim typem są bazy obiektowe, które są zdefiniowane tylko jednym standardem z 1993 roku. Ostatnim typem są strumieniowe bazy danych. Przedstawiają one dane w postaci zbioru strumieni. System zarządzania jest nazwany strumieniowym systemem zarządzania danymi (ang. Data Stream Management System). Jest to nowy typ, który znajduje się w fazie prototypowej i nie istnieją dla niego żadne rozwiązania komercyjne.

1.2. Projektowanie aplikacji

1.2.1. Etap planowania i analizy

W fazie planowania i analizy kluczową rolę odgrywają analitycy, którzy spotykają się z klientem w celu omówienia kluczowych aspektów projektu. W tym czasie omawiane są oczekiwane funkcjonalności, wymagania oraz oczekiwania względem wydajności aplikacji. Jest to etap, w którym zbierane są wszystkie niezbędne informacje, które będą potrzebne w kolejnych fazach projektu. Nawet jeśli tworzymy aplikację na własne potrzeby, warto poświęcić czas na dokładne zrozumienie i zdefiniowanie tych trzech kluczowych punktów. Zapisanie ich pomoże w późniejszym etapie projektowania i implementacji. Na koniec tej fazy powstaje ogólny zarys projektu, który będzie służył jako drogowskaz w kolejnych etapach.

1.2.2. Projektowanie UX/UI

Drugi etap skupia się na projektowaniu doświadczenia użytkownika (UX) oraz interfejsu użytkownika (UI). Bazując na informacjach zebranych w pierwszej fazie, projektanci tworzą wizualne reprezentacje aplikacji, które pomagają zrozumieć, jak będzie ona wyglądać i funkcjonować. Kluczowym celem tego etapu jest stworzenie interfejsu, który jest intuicyjny, interaktywny i przyjazny dla użytkownika. Dobre projektowanie UX/UI może znacząco wpłynąć na sukces aplikacji, ponieważ użytkownicy często oceniają aplikacje na podstawie ich wyglądu i użyteczności.

1.2.3. Tworzenie aplikacji

Faza trzecia to etap, w którym programiści przystępują do właściwej pracy nad kodem. Bazując na specyfikacjach z poprzednich faz, tworzą działające oprogramowanie. Jest to najbardziej pracochłonny etap całego procesu, wymagający nie tylko umiejętności technicznych, ale także zdolności do rozwiązywania problemów i dostosowywania się do ewentualnych zmian w specyfikacji.

1.2.4. Testowanie

Po zakończeniu fazy tworzenia, następuje etap testowania. Jego głównym celem jest sprawdzenie, czy aplikacja działa zgodnie z oczekiwaniami klienta oraz czy nie zawiera błędów. Testerzy przeprowadzają różnego rodzaju testy, od testów jednostkowych po testy integracyjne, aby upewnić się, że wszystko działa poprawnie.

1.2.5. Wdrożenie

Przedostatni etap polega na wdrożeniu aplikacji u klienta. W tym czasie programiści i inżynierowie ds. wdrożeń pracują razem, aby zapewnić płynne i bezproblemowe uruchomienie aplikacji w środowisku produkcyjnym.

1.2.6. Utrzymanie i rozwój

Ostatnia faza to utrzymanie i rozwój aplikacji. W tym etapie programiści i testerzy monitorują działanie aplikacji, analizując jej wydajność i rozwiązując ewentualne problemy. Jeśli klient zgłasza potrzebę wprowadzenia nowych funkcjonalności, proces projektowania i wytwarzania oprogramowania rozpoczyna się ponownie, zaczynając od fazy tworzenia.

Współczesne metody wytwarzania oprogramowania są stale rozwijane, aby sprostać rosnącym wymaganiom rynku. Nowe technologie, biblioteki i frameworki pojawiają się regularnie, oferując programistom narzędzia, które ułatwiają i przyspieszają proces tworzenia aplikacji. Jednak niezależnie od używanych narzędzi, kluczem do sukcesu jest zrozumienie potrzeb klienta i dostarczenie rozwiązania, które spełni te potrzeby.

1.3. Budowa aplikacji webowej

Tworzenie aplikacji webowej to proces skomplikowany i wieloetapowy, który wymaga połączenia różnych technologii, narzędzi i praktyk. Aby zrozumieć, jak jest zbudowana typowa aplikacja webowa, warto przyjrzeć się jej głównym składnikom.

1.3.1. Frontend

Frontend to część aplikacji, z którą bezpośrednio kontaktuje się użytkownik. Obejmuje ona interfejs użytkownika (UI) oraz wszystkie elementy wizualne, które są prezentowane w przeglądarce. Główne technologie używane w frontendzie to:

HTML - język znaczników, który opisuje strukturę strony. CSS - służy do stylizacji elementów HTML, decydując o wyglądzie strony. JavaScript (lub TypeScript w przypadku Angulara) - język programowania, który pozwala na tworzenie interaktywnych elementów strony.

1.3.2. Backend

Backend to serwerowa część aplikacji, która zajmuje się przetwarzaniem danych, komunikacją z bazą danych i realizacją logiki biznesowej. Główne składniki backendu to:

Serwer - maszyna lub oprogramowanie, które obsługuje żądania od klientów i zwraca odpowiednie dane. Baza danych - system przechowywania danych, który pozwala na ich szybkie wyszukiwanie, modyfikowanie i przechowywanie. API (Application Programming Interface) - zestaw reguł i mechanizmów, które pozwalają różnym częściom oprogramowania komunikować się ze sobą.

1.3.3. Proces tworzenia

Analiza wymagań - zrozumienie potrzeb użytkownika i określenie funkcjonalności aplikacji. Projektowanie - tworzenie makiet, wybór technologii i planowanie architektury aplikacji. Implementacja - właściwe programowanie, tworzenie kodu źródłowego aplikacji. Testowanie - sprawdzanie, czy aplikacja działa poprawnie i spełnia wszystkie wymagania. Wdrożenie - umieszczanie aplikacji na serwerze, aby była dostępna dla użytkowników. Utrzymanie - monitorowanie aplikacji, naprawianie błędów, aktualizacje i rozwijanie funkcjonalności. Podsumowując, budowa aplikacji webowej to nie tylko kwestia techniczna, ale także planowania, projektowania i testowania. Ważne jest, aby cały proces był dobrze zorganizowany i skoordynowany, co pozwoli na stworzenie funkcjonalnej, wydajnej i bezpiecznej aplikacji.

1.4. Optymalizacja i bezpieczeństwo aplikacji webowej

W dzisiejszych czasach, kiedy konkurencja w świecie aplikacji internetowych jest ogromna, nie wystarczy stworzyć funkcjonalną stronę. Aplikacja musi być również szybka, responsywna i przede wszystkim bezpieczna. W tym rozdziale omówimy kluczowe aspekty optymalizacji i zabezpieczania aplikacji webowych.

1.4.1. Optymalizacja

Minimalizacja i kompresja plików - zmniejszenie rozmiaru plików CSS, JavaScript i obrazów może znacząco przyspieszyć ładowanie strony. Wykorzystanie pamięci podręcznej - przechowywanie często używanych danych w pamięci podręcznej przeglądarki pozwala na szybsze ładowanie strony podczas kolejnych wizyt. Optymalizacja obrazów - stosowanie odpowiedniego formatu i rozmiaru obrazów, a także leniwe ładowanie (ang. lazy loading), które polega na ładowaniu obrazów dopiero wtedy, gdy są one widoczne dla użytkownika. Optymalizacja baz danych - regularne indeksowanie, czyszczenie i aktualizacja baz danych zapewniają ich wydajne działanie.

1.4.2. Bezpieczeństwo

Szyfrowanie - stosowanie protokołu HTTPS zapewnia, że dane przesyłane między serwerem a klientem są zaszyfrowane i trudne do przechwycenia. Autentykacja i autoryzacja - upewnienie się, że tylko uprawnieni użytkownicy mają dostęp do pewnych zasobów i funkcji aplikacji. Zabezpieczenie przed atakami - takimi jak SQL Injection, Cross-Site Scripting (XSS) czy Cross-Site Request Forgery (CSRF). Wymaga to stałego monitorowania, aktualizacji oprogramowania i stosowania najlepszych praktyk programistycznych. Regularne kopie zapasowe - w przypadku awarii lub ataku, ważne jest posiadanie aktualnych kopii zapasowych danych i kodu aplikacji, aby móc szybko przywrócić jej działanie.

1.4.3. Monitoring i aktualizacje

Nieustanny monitoring aplikacji pozwala na szybkie wykrywanie i reagowanie na potencjalne problemy. Regularne aktualizacje, zarówno samej aplikacji, jak i używanych technologii, zapewniają jej stabilność, wydajność i bezpieczeństwo.

Podsumowując, tworzenie aplikacji webowej to jedno, ale jej optymalizacja i zabezpieczanie to kluczowe kroki, które decydują o sukcesie projektu. Wysoka wydajność i bezpieczeństwo to czynniki, które przyciągają i zatrzymują użytkowników, dlatego nie można ich lekceważyć.

1.5. Integracja z innymi narzędziami i serwisami

Współczesne aplikacje webowe rzadko funkcjonują w izolacji. Często integrują się z różnymi zewnętrznymi narzędziami, platformami i serwisami, aby rozszerzyć swoje funkcjonalności, poprawić wydajność i dostarczyć użytkownikom bardziej kompleksowe doświadczenie. W tym rozdziale przyjrzymy się kluczowym aspektom integracji aplikacji webowych z innymi systemami.

1.5.1. API - klucz do integracji

API, czyli interfejs programistyczny aplikacji, to zestaw reguł i definicji, które pozwalają różnym aplikacjom komunikować się ze sobą. Dzięki API, aplikacja webowa może na przykład pobierać dane z mediów społecznościowych, korzystać z funkcji płatności czy integrować się z systemami zarządzania treścią.

1.5.2. Popularne integracje

Płatności - integracja z platformami takimi jak PayPal, Stripe czy PayU pozwala na szybkie i bezpieczne przeprowadzanie transakcji finansowych w aplikacji. Media społecznościowe - połączenie z Facebookiem, Twitterem czy Instagramem umożliwia na przykład udostępnianie treści, logowanie za pomocą konta w mediach społecznościowych czy analizę aktywności użytkowników. Analityka - narzędzia takie jak Google Analytics czy Hotjar dostarczają cennych informacji o zachowaniach użytkowników, co pozwala na optymalizację aplikacji i dostosowywanie jej do potrzeb odbiorców. Chmura - integracja z platformami chmurowymi, takimi jak AWS, Google Cloud czy Azure, pozwala na skalowanie aplikacji, przechowywanie danych czy korzystanie z zaawansowanych narzędzi do analizy i przetwarzania informacji.

1.5.3. Wyzwania związane z integracją

Integracja z zewnętrznymi narzędziami i serwisami niesie ze sobą pewne wyzwania. Należy między innymi uwzględnić:

Bezpieczeństwo - przesyłanie i odbieranie danych z zewnętrznych źródeł musi być zabezpieczone, aby chronić prywatność użytkowników i unikać potencjalnych ataków. Kompatybilność - różne systemy mogą korzystać z różnych technologii i standardów, co może powodować problemy z integracją. Aktualizacje - zarówno aplikacja, jak i zewnętrzne narzędzia, z którymi się integruje, są regularnie aktualizowane. Należy monitorować te zmiany i dostosowywać integrację, aby wszystko działało poprawnie. Podsumowując, integracja z innymi narzędziami i serwisami pozwala na wzbogacenie funkcjonalności aplikacji webowej i dostarczenie użytkownikom bardziej zaawansowanych i spersonalizowanych doświadczeń. Jednakże wymaga to również uwzględnienia pewnych wyzwań i stałego monitorowania połączeń z zewnętrznymi systemami.

Rozdział 2

Technologie wykorzystane w aplikacji webowej

Tutaj jakiś wstęp do rozdziału.

2.1. Sekcja

Tu sekcja.

2.1.1. Podsekcja

Tu podsekcja. Z tabelką.

1	3.196622222	2.581588889	4.150291803	8.154227869
2	3.34634188	2.45621453	4.479345902	7.572304918
3	2.862333333	2.087170085	3.834808197	6.379259016

Tabela 2.1: Podpis

Rozdział 3

Zbieranie danych badawczych

Tekst.

3.1. Sekcja

Zacytowane [1].

3.1.1. Podsekcja

Tekst.

Rozdział 4

Porównanie wydajności baz relacyjnych i nierelacyjnych

Tekst.

4.1. Sekcja

Tu obraz.

Podsumowanie

Tekst.

Bibliografia

[1] AUTOR, *TYTUŁ*, WYDAWNICTWO, ROK.

Spis tabel

Spis rysunków