



POLITECHNIKA ŁÓDZKA

INSTYTUT INFORMATYKI

PRACA DYPLOMOWA INŻYNIERSKA

Interaktywne nauczanie bazodanowych języków zapytań

Autor:

Łukasz Ochmański

Nr albumu: 183566

Promotor:

dr. Inż Krzysztof Myszkorowski

Łódź, 8 lutego 2015

Streszczenie

Informatyka

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Praca dyplomowa inżynierska

Interaktywne nauczanie bazodanowych języków zapytań

Łukasz Ochmański

Nr albumu: 183566

Celem pracy jest stworzenie interaktywnego samouczka wspierającego polskich studentów w nauce bazodanowych języków zapytań. Główną zaletą systemu jest łatwość dostępu i brak konieczności instalacji silników bazodanowych typu Oracle database client czy też konfiguracji narzędzi typu SQL Server Management Studio. Niekiedy konfiguracja przerasta umiejętności studentów, a w najlepszych wypadku pochłania godziny lub dni. Tego typu działania nie są głównym celem dydaktycznym przedmiotu o nazwie „Podstawy baz danych”.

Aby skupić uwagę studentów na nauce języka, postanowiłem stworzyć ten oto serwis.

Spis treści

Rozdział 1	5
1.1. Wstęp	5
1.2. Cel projektu	7
1.3. Opis struktury pracy	7
1.3.1. Rodział 1	7
1.3.2. Rozdział 2	7
1.3.3. Rozdział 3	7
1.3.4. Rozdział 4	8
1.3.5. Rozdział 5	8
1.3.6. Rozdział 6	8
1.3.7. Rozdział 7	9
1.3.8. Rozdział 8	9
Rozdział 2	10
2.1. Założenia dotyczące projektu	10
2.2. Dostęp	11
2.3. Wymagania sprzętowe	12
Rozdział 3	13
3.1. Architektura systemu	13
3.1.1. Sposoby komunikacji	13
3.1.2. Model trójwarstwowy	13
3.1.3. Data Access Object (DAO)	15
3.2. Warstwa prezentacji (front-end)	16
3.2.1. HTML 5	17
3.2.2. JavaScript	17
3.2.3. jQuery	18
3.2.4. WebSocket	19
3.2.5. STOMP	21
3.2.6. JSON	21
3.2.7. Bootstrap	22
3.2.8. CSS 3	22
3.3. Warstwa dostępu do danych (back-end)	23
3.3.1. JVM	23
3.3.2. Java 8	23
3.3.3. JDBC	24
3.3.4. Apache Tomcat 8	24
3.3.5. Spring framework	25
3.3.6. Spring MVC	Error! Bookmark not defined.
3.3.7. Spring boot	25
3.3.8. Spring session	25
3.3.9. H2 embedded	25
3.3.10. Gradle	26
3.4. Warstwa danych	27
3.4.1. Instancja STUDENT	28
3.4.2. Instancja ADMINISTRATOR	28
3.5. Techniki, wzorce architektoniczne oraz programistyczne	29
3.5.1. DAO	29
3.5.2. MVC	29

3.5.3.	Subscribe.....	29
3.5.4.	Listener	29
3.5.5.	Singleton	29
3.5.6.	Wyrażenia Lambda.....	29
Rozdział 4	29
4.1.	Schemat działania	29
4.1.1.	Przykładowy workflow.....	29
4.1.2.	Weryfikacja odpowiedzi.....	29
4.1.3.	Przyznanie punktów.....	29
4.1.4.	Błędna odpowiedź.....	29
4.2.	Opis wybranych funkcjonalności	29
4.2.1.	Praca grupowa.....	29
4.2.2.	Praca indywidualna.....	29
4.2.3.	Forum DISQUS	29
4.2.4.	Komunikaty.....	29
4.2.5.	Materiały naukowe	29
4.2.6.	HELP	29
4.3.	Szczegółowy opis implementacji.....	30
4.4.	Dokumentacja techniczna	30
Rozdział 5	31
5.1.	Przykłady działania aplikacji oraz dokumentacja użytkownika.....	31
5.2.	Praca zespołowa.....	31
Rozdział 6	32
6.1.	Bezpieczeństwo.....	32
6.1.1.	WebSocket Authentication	32
6.1.2.	Uprawnienia do bazy danych.....	32
6.1.3.	Izolacja	32
6.1.4.	Atomowość	32
6.1.5.	Transakcyjność.....	32
6.2.	Dostępność i niezawodność.....	32
6.2.1.	Load Balancing	32
6.2.2.	Hot backup.....	32
6.2.3.	Incremental backup.....	32
6.3.	Analiza wydajności.....	32
6.3.1.	Porównanie WebSocket vs HTTP.....	32
6.3.2.	H2 vs Oracle	32
6.3.3.	H2 vs MS SQL Server	32
Rozdział 7	33
7.1.	Perspektywy rozwoju	33
7.2.	Wnioski i podsumowanie.....	34
Spis rysunków.	34
Spis tabel.	34
Spis listingów	34
Bibliografia	34
Załączniki	35

Rozdział 1

1.1. Wstęp

Historia SQL

Rozwój relacyjnych baz danych, który miał miejsce w latach 70-tych ubiegłego wieku uwarunkował konieczność opracowania języka do manipulacji, wyciągania i obsługi danych w bazach.

Pierwszym oficjalnym językiem relacyjnych baz danych, był SEQUEL (Structured English Query Language), opracowany przez pracowników firmy IBM (Raymond F. Boyce oraz Donald Chamberline). Zaimplementowany w 1973 roku w SYSTEM R – pierwszym silniku bazodanowym opartym o model relacyjny (jednak pierwszy komercyjny system RDBMS to wdrożenie firmy ORACLE w 1979 r.).

Jak sama nazwa wskazuje, SEQUEL to język w domyśle przyjazny dla użytkownika, służący odpytywaniu baz. Jednym z założeń była łatwość tworzenia zapytań, operacji na zbiorach za pomocą słów kluczowych w języku angielskim. Język miał być intuicyjny i prosty. Te cechy to także założenia samego modelu relacyjnego i chyba właśnie dlatego, systemy baz danych oparte o model relacyjny podbiły świat i są do dziś dominującymi środowiskami bazodanowymi.

Nazwa ewoluowała – SEQUEL, okazała się być nazwą zastrzeżoną przez brytyjską firmę przemysłu lotniczego. Stąd została skrócona do znanej obecnie formy czyli SQL (Structured Query Language).

Najważniejszymi systemami RDBMS (Relational DataBase Management System), w których podstawowym językiem jest SQL, to oczywiście : MS SQL Server, Oracle, DB2, MySQL, PostgreSQL, Sybase.

Standardy, dialekty

Konkurencyjność rynku spowodowała konieczność ustandaryzowania języka SQL i na szczęście stało się to już w roku 1986, kiedy został opracowany przez ANSI pierwszy standard określany jako SQL:86. Podkreślam, że na szczęście tak szybko, bo choć istnieją istotne różnice np. w nazwach implementowanych funkcji, to ogólne zasady dla relacyjnych baz danych, różnych

producentów, są spójne. Ma to znaczenie szczególnie podczas integracji platform i dla nas, pracujących w różnych środowiskach.

Powstały, więc dialekty językowe. Transact-SQL (T-SQL) – historycznie wprowadzony przez Sybase, rozwijany do dziś przez Microsoft w SQL Server. Inne dialekty, mające duże znaczenie na rynku to oczywiście PL/SQL (firmy ORACLE) oraz SQL/PSM (najpopularniejszy silnik relacyjny w serwisach WWW – MySQL).

Pomimo różnic w dialektach, nazwach funkcji, typach danych – istnieje szeroki wspólny mianownik – relacyjny model oparty o teorię zbiorów. Dlatego, jeśli poznasz T-SQL, odnajdziesz się szybko np. w bazie MySQL czy Oracle.

Standardy ANSI są regularnie aktualizowane. Od 1986 roku zostało opublikowanych szereg wersji (aktualnie obowiązująca to ANSI SQL:2011/2011), wprowadzających porządek w nowych funkcjonalnościach. Np. w SQL:2003 zostały wprowadzone standardy związane z obsługą XML.

Kategorie SQL

Istnieją grupy istotnych akronimów, które powinny być znane każdemu użytkownikowi bazy danych. W zależności od zastosowań są to :

DDL – Data Definition Language – czyli komendy dot. tworzenia, modyfikacji obiektów w bazie np. CREATE TABLE, ALTER VIEW, DROP,

DML – Data Modification Language (UPDATE, INSERT, DELETE)

DCL – Data Control Language – kontrola uprawnień (GRANT, DENY, REVOKE)

TCL – Transaction Control Language – obsługa transakcji np. BEGIN TRANSACTION, COMMIT, ROLLBACK.

I w końcu DQL – Data Querying Language – czyli polecenie SELECT

Dotyczy on tylko podzbioru języka SQL – związanego z pisanem zapytań (kwerend).

1.2. Cel projektu

Celem głównym pracy jest zaprojektowanie i zbudowanie aplikacji internetowej przy użyciu dostępnych technologii typu open source w celu nauczania bazodanowych języków zapytań w standardzie SQL-92. Natomiast celem stworzenia aplikacji jest wsparcie studentów w opanowaniu biegłego posługiwania się językiem SQL, którego znajomość jest kluczowa w poznawaniu niezbędnej wiedzy teoretycznej z zakresu systemów baz danych.

Użytkownicy, zdobywają kompetencje swobodnego tworzenia dowolnych kwerend SQL i umiejętności korzystania z wiedzy zawartej w bazach. Poznają metodykę i narzędzia do sprawnego przetwarzania dużych ilości danych, wyciągania tylko istotnych informacji

1.3. Opis struktury pracy

1.3.1. Rozdział 1

Wstęp

W tym rozdziale jest opisana geneza powstania relacyjnych baz danych i języków zapytań.

Cel projektu

W tym rozdziale znajduje się charakterystyka ogólny opis prezentowanego projektu.

Opis struktury pracy

Skrócone zestawienie wszystkich rozdziałów.

1.3.2. Rozdział 2

Założenia dotyczące projektu

Założenia typu, wymagana składnia, niedozwolone operacje, możliwości i ograniczenia techniczne, czego można oczekiwać od systemu, a czego nie.

Potencjalne trudności

Sformułowanie najistotniejszych zagadnień, problemów którym trzeba się przyjrzeć i rozwiązać.

Na tym będzie się skupiać praca.

Dostęp

Ogólne informacje dotyczące lokalizacji aplikacji, sposobów dostępu, kod źródłowego, wymaganych uprawnień.

Wymagania sprzętowe

Specyfikacja wymagań technicznych, niezbędnych do uruchomienia aplikacji.

1.3.3. Rozdział 3

Architektura systemu

Ogólny zarys budowy aplikacji. Model graficzny.

Warstwa prezentacji (front-end)

Ogólny zarys, zasada działania i funkcje pełniące przez warstwę prezentacji.

Warstwa dostępu do danych (back-end)

Ogólny zarys, zasada działania i funkcje pełniące przez warstwę dostępu do danych.

Warstwa danych

Opis silnika bazodanowego, możliwości, trybów pracy, protokołów komunikacji i sposobów integracji z klientami. Utworzone instancje, schematy i użytkownicy.

Techniki, wzorce architektoniczne, wzorce programistyczne

Opis zastosowanych technik i wzorców projektowych (design patterns).

1.3.4. Rozdział 4**Schemat działania**

Zasada funkcjonowania i przepływu informacji pomiędzy poszczególnymi modułami.

Opis funkcjonalności

Szczegółowe wyjaśnienie zasady działania najważniejszych funkcjonalności systemu.

Szczegółowy opis implementacji

Analiza kodu źródłowego.

Dokumentacja techniczna

Dokumentacja dla programistów chcących rozwijać istniejącą aplikację i dla osób chcących korzystać z dostępnych metod API (Application Programming Interface).

1.3.5. Rozdział 5**Przykłady działania aplikacji**

Graficzna prezentacja, pokaz możliwości aplikacji.

Praca zespołowa

Prezentacja modułu do pracy grupowej.

1.3.6. Rozdział 6**Bezpieczeństwo**

Poruszenie tematów związanych z separacją danych, uprawnień użytkowników, dostępu do ocen przez niepowołane osoby.

Dostępność i niezawodność

Tutaj zostaną poruszone kwestie dostępności do systemu w czasie i miejscu. Odporność na ataki, awarie, wirusy, zawieszenia, wyjątki, błędy. Akcja przedsięwzięte w celu uniknięcia problemów.

Analiza wydajności

Porównanie dostępnych na rynku baz oraz porównanie dostępnych protokołów, które mogą zastąpić istniejące.

1.3.7. Rozdział 7**Perspektywy rozwoju**

Lista planowanych funkcjonalności

Wnioski i podsumowanie

Wnioski i podsumowanie.

1.3.8. Rozdział 8

Spis rysunków.

Spis tabel.

Spis listingów

Bibliografia

Załączniki

Rozdział 2

2.1. Założenia dotyczące projektu

- Aplikacja została stworzona do użytku wszystkich zainteresowanych i może być używana nieodpłatnie przez wszystkich studentów w celach dydaktycznych,
- W poniższej pracy będę używał akronimu INBJZ jako skrót od „Interaktywne nauczanie bazodanowych języków zapytań”.
- INBJZ jest aplikacją typu open source i można korzystać z kodu źródłowego w celach niekomercyjnych.
- Użyte przeze mnie biblioteki i frameworki H2, Apache Tomcat, Spring są typu Open Source i korzystanie z kodu źródłowego mojej aplikacji zobowiązuje wszystkich jego użytkowników do udostępnienia całego kodu i podania autora na zasadach omówionych w licencji.
- Podczas korzystania z aplikacji należy być ostrożnym, gdyż na pewno istnieje wiele możliwości „złamania” systemu. Staralem się jednak jak najbardziej taką możliwość ograniczyć i wszystkie zmiany wykonane na głównym schemacie są odwracane przy każdym zapytaniu.
- Aby dokonać trwałych zmian należy stworzyć swój schemat i to powinno zwiększyć ochronę danych.
- Nie ma gwarancji, ani zabezpieczeń, że inny użytkownik nie usunie informacji wprowadzonych przez innego użytkownika.
- Każda skuteczna zmiana w bazie jest logowana (zakończona powodzeniem) i widoczna przez administratora systemu.
- Wszystkie próby (skuteczne i nieskuteczne) udzielenia odpowiedzi są logowane i widoczne przez administratora systemu.
- Dialekt SQL bazy danych H2 powinien być zgodny z dialektem MySQL oraz MS SQL Server.
- Baza wspiera rozróżnianie wielkości znaków, jeśli używa się cudzysłówów.
- Zarówno producent silnika bazodanowego H2 jak i autor aplikacji (Łukasz Ochmański) nie ponoszą odpowiedzialności za problemy spowodowane użytkowaniem aplikacji.
- Wspierane kodowanie to UTF-8. Możliwa jest również obsługa Unicode. Można zatem śmiało wprowadzać polskie znaki.
- Dla poprawnego działania aplikacji zalecane jest używanie najnowszej przeglądarki Firefox.
- Stabilność systemu nie jest gwarantowana. Projekt został stworzony jedynie do domowego użytku.
- W razie jakichkolwiek problemów, należy się kontaktować ze mną poprzez forum na stronie lub e-mail.

2.2. Dostęp

Adres

Wersja beta aplikacji jest dostępna pod adresem: <http://5.135.146.42:8080/>.

Godziny dostępu

W planach system będzie dostępny całą dobę. Jednak w przypadku wystąpienia błędów wymagany będzie restart, co spowoduje unieruchomienie aplikacji na nie więcej niż 5 minut.

Docelowa grupa odbiorców

W początkowej fazie wdrożenia nie będzie ograniczeń co do osób korzystających z serwisu. Każdy może anonimowo korzystać z INBJZ. Docelowo będą to studenci przedmiotu „Podstawy baz danych” na wydziale Fizyki Technicznej Informatyki i Matematyki Stosowanej Politechniki Łódzkiej kierunku informatyka. W obecnej wersji można się przedstawić poprzez wpisanie swojego identyfikatora w prawym górnym rogu ekranu. Nie ma przeprowadzanej żadnej autentykacji, więc można się podać za dowolną osobę i nie ma z tego tytułu żadnych negatywnych konsekwencji.

VCS (Version Control System)

Repozytorium kontroli wersji jest prowadzone na bieżąco w systemie GIT. Można sklonować sobie całe repozytorium poprzez wydanie polecenia:

```
git clone https://ochman2000@code.google.com/p/interaktywne-nauczanie-jezykow-bazodanowych/
```

Aby jednak brać czynny udział w rozwoju projektu należy mnie o to poprosić. Wtedy udostępnię zasoby.

Kod źródłowy i użyte narzędzia

Projekt został napisany w języku Java. Aby go uruchomić należy posiadać Java Development Kit 8. Następnie gotowy projekt można zaimportować do dowolnego środowiska programistycznego za pomocą systemu budowania (Build Automation Software) Gradle. Gotowy projekt jest skonfigurowany do działania w IntelliJ 14. Kod źródłowy można również podejrzeć za pomocą dowolnej przeglądarki internetowej pod adresem:

<https://code.google.com/p/interaktywne-nauczanie-jezykow-bazodanowych/>

2.3. Wymagania sprzętowe

Przeglądarka

Wymagana przeglądarka zdolna do obsługi JavaScript, HTML5, CSS3 oraz WebSocket. Strona działa na wszystkich najnowszych przeglądarkach, ale zalecana jest Mozilla Firefox.

Strona kodowania

UTF-8

System operacyjny

Dowolny system operacyjny z graficznym interfejsem użytkownika i działającą przeglądarką.

Rozdzielczość ekranu

Zalecana rozdzielczość: 1366x768 pikseli.

Pamięć operacyjna

Zdolna uruchomić przeglądarkę Firefox czyli około 512MB

Rozdział 3

3.1. Architektura systemu

3.1.1. Sposoby komunikacji

W aplikacji są używane trzy główne protokoły komunikacji:

HTTP

Poprzez ten protokół serwowane są statyczne strony HTML, pliki CSS oraz pliki z kodem JavaScript do klienta i od klienta. Za pomocą prostych metod GET i POST można uzyskać prawie wszystkie funkcjonalności omawianej aplikacji.

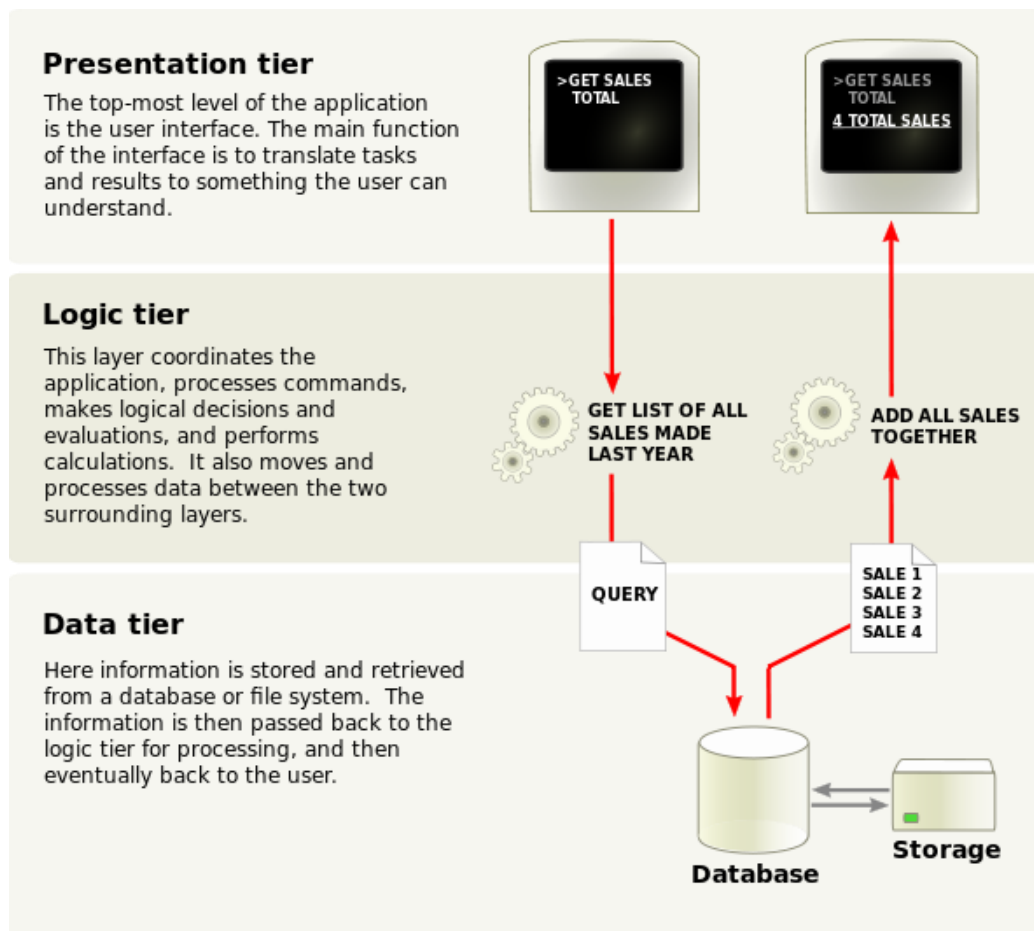
WebSocket

Poprzez ten protokół następuje wymiana danych pomiędzy serwerem, a klientem. Wszystkie zapytania i komendy wpisywane w polu tekstowym na stronie HTML są wysyłane w postaci JSON (JavaScript Object Notation) poprzez tekst lub strumień bajtów, następnie wysyłane do serwera na jeden z dostępnych portów bezpośrednio w niższej warstwie modelu TCP/IP. Zapewnia to najwyższą efektywność przy jednoczesnym zapewnieniu pewności transmisji.

JDBC (Java Database Connectivity) komunikuje się ze sterownikiem bazy danych i tłumaczy wszystkie zapytania w postaci tekstowej z poziomu kodu napisanego w języku Java na natywne komendy udostępnione przez producenta bazy danych. Następnie dane są zwracane poprzez sterownik do programu wysokiego poziomu i interpretowane przez programistę, czyli mnie.

3.1.2. Model trójwarstwowy

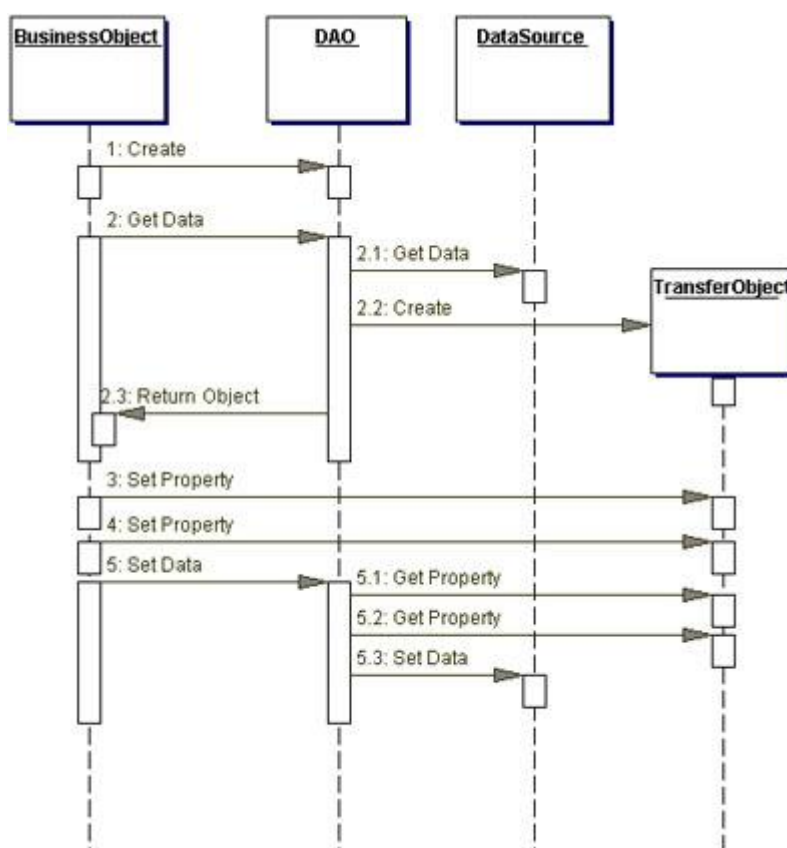
Cała architektura projektu opiera się o klasyczny model trójwarstwowy (Three-Tier-Architecture), gdzie moduły systemu zostały oddzielone i uniezależnione od siebie. Każdy z modułów jest samodzielny i może w każdej chwili zostać wymieniony. Poniższy rysunek prezentuje zastosowany model:



źródło: http://en.wikipedia.org/wiki/Multitier_architecture

3.1.3. Data Access Object (DAO)

Aby umożliwić modularność projektu należało użyć techniki hermetyzacji obiektów, dzięki której jest możliwa łatwa zamiana dostępnych modułów przedstawionych w punkcie powyżej. Technika polega na wydzieleniu fragmentów kodu, które łączą się z poszczególnymi warstwami i stworzeniu standardowego interfejsu, który jest zawsze używany do komunikacji pomiędzy modułami. Interfejs w języku Java jest to zbiór definicji metod poprzez, które programista zawsze odwołuje się w swoim programie. Implementacja metod jest dostarczana w zależności od potrzeby. W ten sposób można jedną linią kodu przełączyć źródło pobierania danych. Może zdarzyć się, że zaistnieje potrzeba stworzenia wersji systemu, która będzie pobierała dane jedynie z plików tekstowych. Innym razem zaistnieje potrzeba pobierania danych z odległego serwera poprzez gniazda TCP/IP, a jeszcze innym razem z zewnętrznego WebService'u. Ani użytkownik, ani programista nie zauważą różnicy w funkcjonowaniu programu. Wszystkie metody będą działać i będą dostarczać dane w taki sam sposób jak wcześniej. Jedyna różnica będzie polegać na efektywności i skutkach działania programu. Ważne jest, aby zaplanować za czasu metody i klasy, które mogą ulec zmianie. Wówczas zmiany w kodzie będą minimalne. Diagram interakcji przedstawiony na rysunku poniżej:



źródło: <http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

W przypadku mojej aplikacji istnieje bardzo duże prawdopodobieństwo, że baza danych zostanie wymieniona, gdyż H2 jest niewystarczająca do wielu rozwiązań. Z tego powodu w moim kodzie zostały wydzielone interfejsy DatabaseDAO.java, który zawiera metody takie jak:

```
public interface DatabaseDao {  
  
    List<String[]> executeQuery(String sql) throws SQLException;  
    String executeStmt(String sql) throws SQLException;  
    String update(String sql);  
    String[] getLabels(String sql);  
}
```

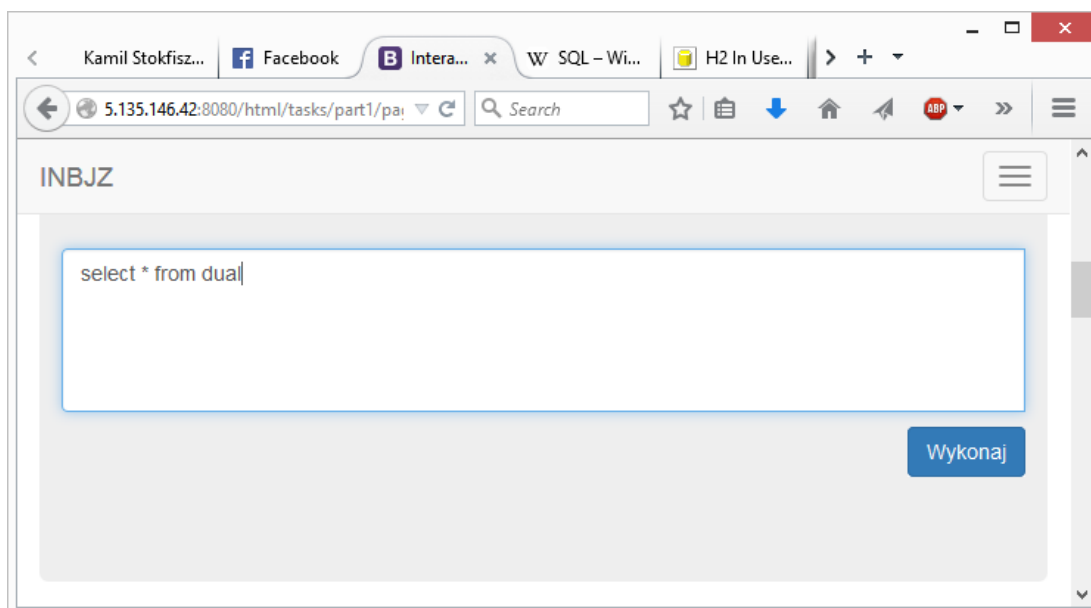
Kolejnym przykładem użycia techniki DAO jest wydzielenie interfejsu dostępu do repozytorium zadań. Nie wiadomo gdzie zadania i odpowiedzi będą się znajdowały w przyszłości. Może to być plik XML, baza danych, dokument tekstowy lub statyczna klasa Java'owa. Jak na razie wybrałem składowanie odpowiedzi w bazie danych.

```
public interface TaskRepository {  
  
    Task getTaskById();  
    Task getTask(int chapter, int number);  
    List<Task> getTasks(int chapter);  
    List<Task> getTasks();  
}
```

3.2. Warstwa prezentacji (front-end)

W tym rozdziale zajmę się opisem warstwy prezentacji (presentation-tier) wymienioną w punkcie 7.2. Warstwa prezentacji jest w bezpośrednim kontakcie z użytkownikiem i zazwyczaj ma formę graficzną. Dawniej, zanim powstały przeglądarki internetowe i kolorowe monitory używano terminali TTY do wydawania koment. W dzisiejszych czasach odpowiednikiem takiego terminala jest przeglądarka, która wchodzi w interakcję z użytkownikiem i rozpoznaje jego zamiary. Poprzez kliknięcia na odpowiednie przyciski interpretuje chęci internauty i oddelegowuje komendy do serwera, który przetwarza żądanie, a następnie odpowiada przeglądarce. Ta z kolei wyświetla żądany rezultat.

3.2.1. HTML 5



HTML to język znaczników, który pozwala opisać strukturę strony internetowej. HTML przeszedł przez wiele wersji. Początkowo w pierwszej publicznie dostępnej specyfikacji w internecie (nazwanej uprzednio „HTML Tags”) zawierała 22 znaczniki, z których do dzisiaj nadal używanych jest 13. Pierwsza wersja była oparta o język SGML (ang. Standard Generalized Markup Language). Kolejne wersje na przestrzeni 5 lat były ciągle zmieniane aż do 1997 roku i wersji HTML 4.0. Jedną z wersji, która najdłużej wyznaczała standard w sieci była specyfikacja języka HTML 4.0. Od tamtej pory pojawiło się także kilka odmian HTML 4.0 takich jak np. XHTML 1.0. Wprowadzenie HTML5 zajęło dużo czasu, ze względu na dopracowanie HTML5 do poziomu, który mógłby być akceptowalnym standardem używanym przez wiele rodzajów urządzeń. Wersja HTML5 został zatwierdzony 3 miesiące temu w październiku 2014 roku. HTML5 zawiera w obecnej specyfikacji [w3c ref] 89 znaczników. Jedną z głównych zalet języka, jest niezależność od systemu operacyjnego oraz sprzętu komputerowego. Pozwala to uruchamiać aplikacje HTML na szerokiej gamie urządzeń specjalistycznych i konsumenckich.

3.2.2. JavaScript

Język JavaScript jest prawdopodobnie najczęściej używanym językiem programowania na świecie. Do zalet języka należy prostota składni, łatwość dostępu i wizualny efekt działania. Najczęściej JavaScript używa się do wykonywania animacji na stronie HTML. Niektórzy twierdzą, że kolejna wersja języka JavaScript odegra ważną rolę także w innych dziedzinach, gdyż przyrost mocy

obliczeniowej urządzeń elektronicznych jest tak duży, że prawie każdy procesor jest w stanie poradzić sobie z interpretowaniem kodu JavaScript w czasie rzeczywistym.

W JavaScript istnieje też realny problem: JavaScript jest absurdalnie liberalnym językiem. Zamierzeniem twórcy tego języka było sprawienie, aby był on jak najłatwiejszy w użyciu dla początkujących programistów. Jednak w rzeczywistości to tylko utrudnia znajdowanie problemów w programach, ponieważ system ich nie pokazuje.

Drugim problemem jest to, że JavaScript jest kompilowany dynamicznie, co oznacza, że nie można dowiedzieć się o istniejących problemach przed uruchomieniem. Brak silnego typowania utrudnia rozpoznanie oczekiwanych wartości funkcji. Obiekty są rzutowane na różne typy w zależności od sytuacji.

Mimo swojej nazwy, język JavaScript ma niewiele wspólnego z językiem Java. Zbieżność ta jest wynikiem marketingowych zabiegów i nie powstała w wyniku racjonalnego wyboru. W 1995 r., gdy firma Netscape opublikowała JavaScript, język Java był intensywnie promowany i zdobywał ogromną popularność. W ten sposób powstała ta nazwa.

Gdy obsługę JavaScriptu wprowadzono także w innych przeglądarkach niż Netscape, sporządzono dokument zawierający opis, jak dokładnie ten język powinien działać. Język, który w nim opisano został nazwany ECMAScript, po nazwie organizacji standaryzacyjnej, która zajęła się napisaniem tego standardu.

Standard ECMAScript opisuje język programowania ogólnego przeznaczenia i nie ma w nim ani słowa o jego integracji z jakąkolwiek przeglądarką internetową. W związku z tym JavaScript to ECMAScript plus dodatkowe narzędzia do manipulowania stronami internetowymi i oknami przeglądarki.

JavaScript wciąż ewoluuje. W najnowszych przeglądarkach obsługiwana jest wersja ECMA Script 5. W czerwcu 2015 planowane jest wprowadzenie nowej wersji ECMA Script 6.

3.2.3. jQuery

Do 2005 roku JavaScript, choć był dostępny w przeglądarkach internetowych, służył najczęściej jedynie do tworzenia bardzo prostych animacji, walidacji formularzy lub projektowania rozwijanego menu. Te nieskomplikowane zadania nie wymagały użycia specjalnych bibliotek, bo najczęściej konieczna ilość kodu mieściła się w kilkunastu wierszach.

Wraz z popularyzacją technologii AJAX, a także bogatych, interaktywnych interfejsów aplikacji internetowych, objętość kodu drastycznie wzrosła. Posługiwanie się nim dodatkowo utrudnił fakt, że przeglądarki internetowe (w szczególności Internet Explorer) w różny sposób implementowały szczegóły dostępu do dokumentu HTML (model DOM), deklaracje zdarzeń lub tworzenie obiektu XMLHttpRequest.

Mimo to apetyt deweloperów rósł – a JavaScript nie zawsze potrafił za nim nadążyć. Zapaleńcom dawał się we znaki brak kilku podstawowych funkcji znanych z innych języków, choćby wyszukiwania, czy dany obiekt znajduje się w tablicy oraz usuwania z tekstu spacji.

Wymienione utrudnienia stały się powodem powstania kilkunastu bibliotek przeznaczonych dla popularnego języka skryptowego. Jedne starały się rozwiązać palące problemy, rozszerzając wbudowane obiekty, inne udostępniały coś na kształt dodatkowych klas i modułów. Większość jako swój podstawowy cel stawiała ułatwienie obsługi drzewa DOM i ukrycie różnic w interpretacji kodu przez przeglądarki.

Obecnie biblioteki JavaScript można podzielić na dwa rodzaje: ułatwiające dynamizację istniejącego kodu HTML poprzez uproszczenie dostępu do DOM, CSS i AJAX oraz definiujące widgety i budujące cały interfejs użytkownika bezpośrednio w JavaScript. jQuery należy do pierwszej z wymienionych kategorii – udostępnia interfejs doskonale współpracujący z „klasycznymi” dokumentami HTML. Jeśli chcemy tworzyć aplikacje internetowe generujące interfejs użytkownika w JavaScript (jak np. Gmail), prawdopodobnie wygodniej będzie skorzystać z innych bibliotek, np. Dijit lub Ext-JS.

3.2.4. WebSocket

WebSockets służy do nawiązywania połączenia dwukierunkowego przez TCP. Protokół ten został ustandaryzowany przez organizację IETF (ang. Internet Engineering Task Force) w 2011 roku. Web Socket może być zastosowany wewnątrz każdego rodzaju klientów i serwerów internetowych. Całe połączenie przez protokół jest odseparowane od protokołu HTTP. Co pozwala na tworzenie wszelkiego rodzaju aplikacji czasu rzeczywistego. W konsekwencji komunikacja między przeglądarką użytkownika a serwerem może być realizowana w czasie rzeczywistym bez narzutu nagłówków protokołu HTTP. Wraz z wprowadzeniem WebSockets zaprezentowano dwa nowe schematy URI - ws: i wss: do połączeń szyfrowanych.

Do tej pory aplikacje bazujące na HTML komunikowały się z serwerem synchronicznie: zapytanie i odpowiedź. Jeśli serwer potrzebował więcej czasu na wykonanie zadania to można było czekać albo dokonywać co pewien czas zapytań o stan procesu (ang. polling; metoda

przeglądania). Dzięki dwukierunkowej komunikacji powstała możliwość powiadamiania przeglądarki o chęci zmiany za pośrednictwem jednego gniazda TCP. Za pomocą standardowego protokołu HTTP było to dotychczas niemożliwe. W odróżnieniu od polling nowa metoda nosi nazwę push, ponieważ chęć wysłania wiadomości nie musi być poprzedzona żądaniem. Serwer wypycha wiadomość do klienta bez jego woli, a nie serwuje tak jak dawniej.

Aby używać technologii WebSocket wymagana jest przeglądarka, która wspiera ten protokół oraz serwer. W moim przypadku jest to Apache Tomcat 8, który już od wersji 7 posiadał wbudowane wsparcie.

Aby ustawić połączenie z poziomu przeglądarki internetowej w aplikacji web'owej musimy wykonać jedną czynność przez HTTP, którą jest 'handshake'. Handshake wykonywany jest na porcie 80, aby serwer HTTP mógł go przetworzyć. Przykładowa próba połączenia widnieje poniżej.

Zapytanie klienta przez HTTP:

```
GET / chat HTTP /1.1
Host : server . example . com
Upgrade : websocket
Connection : Upgrade
Sec - WebSocket - Key : x3JJHmDL1EzLkh9GBhXDw ==
Sec - WebSocket - Protocol : chat , superchat
Sec - WebSocket - Version : 13
Origin : http :// example . com
```

Odpowiedz serwera przez HTTP, zawierająca pole „Sec-WebSocket-Accept”:

```
HTTP /1.1 101 Switching Protocols
Upgrade : websocket
Connection : Upgrade
Sec - WebSocket - Accept : HSmrc0sMlYUkAGmm5OPpG2HaGWk =
Sec - WebSocket - Protocol : chat
```

W zapytaniu klienta pole klucza 'Sec-WebSocket-Key' to losowa liczba enkodowana skrótem base64. W odpowiedzi na zapytanie serwer dokonuje konkatencji klucza z unikalnym polem GUID. Następnie wartość jest zakodowana algorytmem skrótu SHA-1, ponownie enkodowana

przez base64 i zwracana w odpowiedzi w polu 'Sec-WebSocket-Accept'. Połączenie na tym etapie jest ustanowione.

Dane mogą być wymieniane w ramach z niewielkim narzutem na nagłówek i payload. Długość wysyłanych danych może być dowolna, aż do odczytania końcowego bitu FIN. Protokół używa prefixów ws:// oraz wss:// - oznaczają one połączenie Web Socket oraz Web Socket Secure, które są odpowiednikami HTTP oraz HTTPS, czyli połączenia nieszyfrowanego oraz szyfrowanego.

3.2.5. STOMP

STOMP is the Simple (or Streaming) Text Orientated Messaging Protocol.

STOMP provides an interoperable wire format so that STOMP clients can communicate with any STOMP message broker to provide easy and widespread messaging interoperability among many languages, platforms and brokers.

STOMP is a very simple and easy to implement protocol, coming from the HTTP school of design; the server side may be hard to implement well, but it is very easy to write a client to get yourself connected. For example you can use Telnet to login to any STOMP broker and interact with it!

Many developers have told us that they have managed to write a STOMP client in a couple of hours to in their particular language, runtime or platform into the STOMP network. So if your favored language/runtime of choice does not offer a good enough STOMP client don't be afraid to write one.

3.2.6. JSON

JSON, JavaScript Object Notation jest formatem tekstowym, bazującym na podzbiorze języka JavaScript. Typ MIME dla formatu JSON to application/json. Format został opisany w dokumencie RFC 4627.

JSON jest jednym z nieformalnych sposobów przekazywania danych do aplikacji opartych o AJAX. W typowych przypadkach dane w formacie JSON są pobierane z serwera jako tekst przy wykorzystaniu obiektu XMLHttpRequest języka JavaScript, a następnie przekształcane w obiekt. Tekst powinien być kodowany za pomocą UTF-8, który jest w JSON domyślny.

Komunikat JSON jest literałem obiektu języka Javascript, który w tym języku jest tablicą asocjacyjną. Wszystkie dane są zmiennymi (nie stanowią kodu wykonywalnego) a nazwy składników (właściwości) obiektów są otoczone cudzysłowami. Wartości mogą być typu string

(napis otoczony cudzysłowem), number (liczba typu double), stanowić jedną ze stałych: false, null, true, być tablicą złożoną z takich elementów lub obiektem. Obiekty i tablice mogą być dowolnie zagnieżdżane. Cały komunikat jest kodowany w unikodzie i domyślnie jest to UTF-8.

3.2.7. Bootstrap

Bootstrap powstał w 2010 roku i swoje istnienie zawdzięcza ówczesnym deweloperom pracującym przy Twitterze. Idea tego frameworka jest całkiem prosta i sprowadza się do dostarczenia elastycznych reguł CSS do tworzenia szybkich layoutów. Pewnym uzupełnieniem jest kod JavaScript, dzięki któremu możemy dodać naszym wystylizowanym elementom trochę dynamiki.

CSS przygotowany przez autorów frameworka obejmuje większość znaczników HTML. Część z nich, tych bardziej podstawowych, odpowiedzialnych za formatowanie tekstu posiada style przypisane bezpośrednio do konkretnych tagów. W automatyczny sposób sformatowane mamy więc np. nagłówki, paragrafy, czy też cytaty. W pozostałych przypadkach musimy zastosować konkretne klasy.

Twórcy Bootstrapa pomyśleli o naprawę wielu różnych problemach które spadają na webdeveloperów i dostarczyli nam klasy dzięki którym stworzymy praktycznie wszystko, począwszy od ładnych przycisków, poprzez piękne tabele i formularze, aż do całych responsywnych layoutów.

3.2.8. CSS 3

Kaskadowe arkusze stylów (CSS - Cascading Style Sheets) są nieodłącznym elementem współczesnej wersji języka HTML. Język CSS odpowiada za wizualną prezentację stron internetowych w przeglądarkach.

Można wprawdzie stworzyć witrynę, używając jedynie czystego HTML, jednak będzie to witryna prosta i mało atrakcyjna. Style pomogą znacznie ją wzbogacić, dlatego ich znajomość jest dzisiaj niezbędna. Ponadto stosowanie stylów oszczędza wiele pracy koniecznej dawniej do sformatowania dokumentów za pomocą tradycyjnych technik - nierzadko zmiana jednego drobnego parametru w arkuszu stylów pozwala zmienić wygląd całej witryny.

Zaledwie kilka lat temu twórcy stron WWW korzystali z języka HTML i wyłącznie za jego pomocą „rzeźbili” ostateczny kształt projektowanej witryny. Pomimo ogromnych ograniczeń, jakie oferowała ta metoda, powstawały bardzo ciekawe projekty łamiące wszelakie ograniczenia.

Wraz z wprowadzeniem pierwszej specyfikacji kaskadowych arkuszy stylów, a później jej drugiej odsłony twórcy stron złapali wiatr w żagle. Od teraz języki HTML oraz jego bezpośredni następca XHTML stały się jedynie zbiorem elementów odpowiedzialnych za określanie właściwości poszczególnych składników strony, np. nagłówków, akapitów czy tabel. Natomiast cały proces formatowania ich wyglądu lub umiejscowienia na stronie został zlecony kaskadowym arkuszom stylów. Rozwiązanie takie pozwoliło na znaczną poprawę komfortu tworzenia stron oraz odchudzenie kodu.

Zacięta konkurencja na rynku przeglądarek doprowadziła do sytuacji, w której ich autorzy zostali zmuszeni do ścisłego trzymania się różnych specyfikacji i natychmiastowego wprowadzania nowości i poprawek. Świetnym przykładem działania konkurencji na rynku przeglądarek jest fakt, że wiele z nich aktualnie obsługuje elementy wstępnej wersji specyfikacji CSS 3, która nie została jeszcze skończona i oficjalnie zatwierdzona.

3.3. Warstwa dostępu do danych (back-end)

3.3.1. JVM

Java Virtual Machine (JVM) to rodzaj wirtualnego komputera, który ma swój zestaw rejestrów, zestaw instrukcji, stos i pamięć dla programów.

Dzięki standaryzacji maszyny wirtualnej, programy napisane w Javie są uniwersalne, tzn. wykonują się identycznie w każdym systemie operacyjnym.

Programy napisane w Javie są kompilowane do poziomu kodu pośredniego, nazywanego kodem bajtowym Javy (bytecode). Kod bajtowy jest interpretowany przez wirtualną maszynę JVM do postaci programu wykonywalnego dla danego systemu operacyjnego.

3.3.2. Java 8

Java 8 jest przełomowym wydaniem SDK z dwóch powodów:

Wprowadzono Stream API, które pozwala wykonywać operacje na zbiorach podobnie jak w językach deklaratywnych typu SQL. Drugim powodem jest wprowadzenie wyrażeń lambda, które zrewolucjonizowały sposób pisania programów w języku Java. Dodatkowo zamieszczam listę najważniejszych zmian i wprowadzonych udogodnień w wersji ósmej:

- 101 Generalized Target-Type Inference
- 103 Parallel Array Sorting
- 104 Annotations on Java Types
- 107 Bulk Data Operations for Collections
- 109 Enhance Core Libraries with Lambda
- 117 Remove the Annotation-Processing Tool (apt)
- 120 Repeating Annotations
- 122 Remove the Permanent Generation
- 126 Lambda Expressions & Virtual Extension Methods
- 135 Base64 Encoding & Decoding
- 150 Date & Time API
- 153 Launch JavaFX Applications
- 155 Concurrency Updates
- 160 Lambda-Form Representation for Method Handles
- 161 Compact Profiles
- 162 Prepare for Modularization
- 174 Nashorn JavaScript Engine
- 184 HTTP URL Permissions
- 185 JAXP 1.5: Restrict Fetching of External Resources

3.3.3. JDBC

Java, jako uniwersalny język programowania, daje możliwość dostępu do baz danych.

Przenośność aplikacji bazodanowych tworzonych w Javie zapewnia interfejs JDBC – opracowany przez firmę Sun Microsystems w 1996 roku. Umożliwia on konstruowanie i wykonywanie poleceń SQL’owych z poziomu kodu Javy. Dzięki JDBC aplikacje bazodanowe napisane w Javie są niezależne od sprzętu oraz stosowanej bazy danych (niezależność od systemu operacyjnego zapewnia sama Java). Należy jednak zaznaczyć, iż nadal możliwe jest stworzenie specyficznych wywołań lub użycie typów danych, występujących tylko w przypadku konkretnej bazy danych.

3.3.4. Apache Tomcat 8

Apache Tomcat jest serwerem aplikacji na licencji Apache Software License pozwalającym uruchomić nam aplikacje webowe napisane w JAVA. Spełnia specyfikację firmy SUN jeśli chodzi o Java Servlets oraz Java Server Pages. Zaletą tego serwera jest to, że jest w całości napisany w JAVA stąd jest dostępny na wszystkie platformy. Tomcat jest kontenerem webowym przechowującym aplikacje napisane w Javie EE. Jest to Javowski odpowiednik serwera Apache HTTP (Apache HTTP jest wykorzystywany w udostępnianiu stron napisanych w języku HTML). Dostęp do aplikacji jest identyczny jak w przypadku zwykłych stron internetowych. Z punktu

widzenia użytkownika/klienta, korzystanie z aplikacji nie różni się od tych które zostały napisane w języku PHP, Java czy Python, ponieważ Tomcat bazuje na protokole HTTP.

3.3.5. Spring framework

Spring Framework jest biblioteką tworzenia aplikacji dostępną na zasadach open source, która umożliwia uproszczenie procesu tworzenia oprogramowania biznesowego za pomocą języka Java. Biblioteka ta pozwala osiągnąć ten cel przez udostępnienie programistom modelu komponentów oraz zbioru postych i spójnych API, które w efektywny sposób izolują ich od złożonego kodu podstawowego, wymaganego w skomplikowanych aplikacjach.

W ostatnich dziewięci latach zakres tej biblioteki znacznie się zwiększył, ale pomimo tego pozostała prosta i łatwa w użyciu. Obecnie składa się ona z około dwudziestu modułów, które mogą być podzielone na 6 podstawowych obszarów funkcjonalnych:

- Dostęp do danych (integracja),
- Sieć WWW,
- Programowanie aspektowe (AOP),
- Instrumentacja,
- Podstawowy kontener,
- Testy

3.3.6. Spring boot

Spring Boot jest zbiorem ponad 40 mikroprojektów dostępnych na oficjalnej stronie Spring. Są one bardzo wygodne dla osób rozpoczynających pracę ze Springiem. Gotowe pakiety pozwalają na uruchomienie aplikacji w mniej niż 15 minut. Mikroprojekty Spring Boot nadają się świetnie do integracji z istniejącymi projektami. Można łatwo dołączyć je do swojej aplikacji używając narzędzi do automatycznego budowania, takich jak Maven, Gradle. Nie trzeba już szukać w internecie przykładów zastosowania i modyfikować ich do własnych potrzeb. Spring Boot dostarcza zbiór działających aplikacji, których nie trzeba modyfikować. Można je jedynie rozwijać według własnych potrzeb. Nie ma również konieczności zawierania wymaganych bibliotek, konfigurowania zewnętrznych serwerów, czy nawet środowisk programistycznych. Wystarczy dołączyć zależność do pliku POM w przypadku Maven lub .gradle w przypadku Gradle i aplikacja powinna zadziałać natychmiast.

3.3.7. H2 embedded

H2 jest jednym z najlżejszych systemów zarządzania relacyjną bazą danych. H2 została całkowicie napisana w języku Java i jej głównymi zaletami są: wieloplatformowość i szybkość działania. Baza H2 jako jedyna może działać całkowicie wewnątrz wirtualnej maszyny Javy. Może zostać

wbudowana w aplikację i przechowywana w pamięci operacyjnej. Rozmiar samej biblioteki bazy danych wynosi jedynie 1.5MB i jest dostępna w postaci pliku z rozszerzeniem *.jar. H2 jest również dostępna w pakiecie SpringBoot, co gwarantuje bardzo niską barierę wejścia.

Doświadczonemu programiście wystarczy zaledwie 10 minut, aby znaleźć bibliotekę, zainstalować i uruchomić aplikację komunikującą się z bazą H2 poprzez JDBC.

3.3.8. Gradle

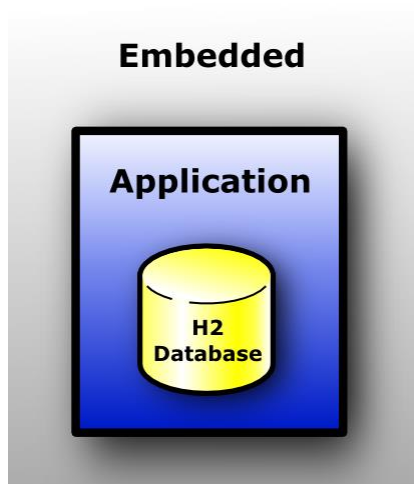
Gradle jest potężnym narzędziem do budowy projektów. Gradle łączy w sobie cechy innych narzędzi takich jak Ant oraz Maven, eliminując przy tym nadmiarowość kodu jaki należało napisać podczas konfiguracji projektu. Do tej pory trudność sprawiało zaimplementowanie jakiegokolwiek algorytmu w skrypcie budującym; trudno było wyrazić nawet tak podstawowe konstrukcje jak if czy for, ciężko było wykonać nawet pozornie proste taski, których wykonywanie nie zostało przewidziane przez developerów Anta/Mavena, które nie są dostępne w postaci gotowych tasków lub pluginów. Potrzebna jest dogłębna znajomość tych narzędzi, by zrozumieć złożony skrypt budujący. Poza tym podejście "build by convention" nie jest wspierane (Ant), albo skutecznie utrudnia konfigurację (Maven). Ponadto wsparcie dla budowy projektów wielomodułowych jest niewystarczające, skrypty budujące są niepotrzebnie duże i nieczytelne z uwagi na narzut samego języka (XML).

Składnia zaczerpnięta z języka Groovy pozwala na większą elastyczność i daje większe możliwości. Można dosłownie napisać program, który będzie czyścił, instalował, kompilował i budował warunkowo, w zależności od okoliczności, czasu i miejsca nasze projekty dokładnie tak jak sobie tego życzymy. Ponadto w Gradle zostały podjęte zdecydowane kroki, aby wyeliminować problemy pojawiające się w poprzednich wersjach narzędzi do budowy projektów. Do tej pory należało uważnie umieszczać zależności, które mogły powodować występowanie cykli. W Gradle użyto skierowanego acyklicznego grafu do rozwiązania problemu.

3.4. Warstwa danych

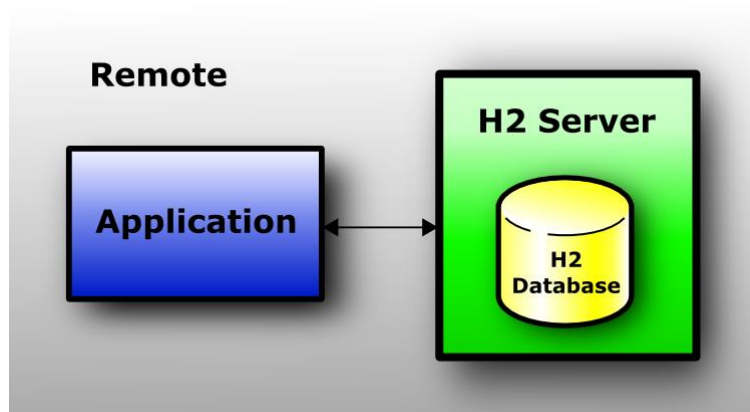
Jak już wcześniej wspomniałem w projekcie użyto bazy danych H2. Technologia została wybrana ponieważ zapewnia najszybszy dostęp do aplikacji, które działają na tej samej instancji wirtualnej maszyny Java JVM. Zgodnie ze specyfikacją producenta H2 może działać w trzech trybach:

- Wbudowany (embedded)



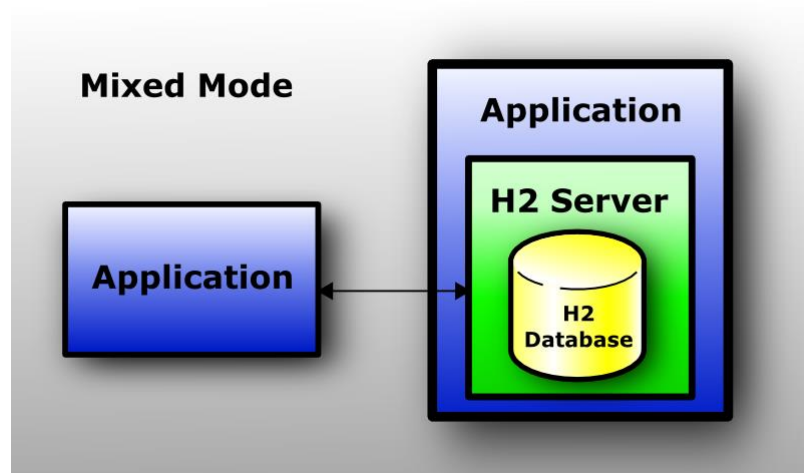
Ten tryb pozwala na najszybszy dostęp do danych, gdyż cała baza może być przechowywana w pamięci operacyjnej.

- Serwer



Standardowy tryb typu klien-serwer, gdzie jest dostępna liczba połączeń i należy się łączyć poprzez sterownik JDBC lub ODBC. Komunikacja odbywa się poprzez protokół TCP/IP. W razie nadmiaru żądań klienci są kolejkowani i muszą czekać na odpowiedź.

- Mixed



Ten tryb pozwala na korzystanie z zasobów bazy danych poprzez kilku użytkowników jednocześnie. Mogą to być osobni klienci, czy osobne procesy w systemie łączące się przez standardowy sterownik JDBC i jednocześnie sama aplikacja działająca wewnątrz tego samego procesu w tej samej wirtualnej maszynie Java. Oczywiście proces korzystający z zasobów z wewnątrz JVM będzie działał kilkadziesiąt, może nawet kilkaset razy szybciej.

3.4.1. Instancja STUDENT

Do potrzeb projektu INBJZ należało użyć dwie osobne instancje bazy danych. Jedna z nich będzie służyć do manipulacji i nauki i będzie w całości dostępna dla studentów. Wewnątrz bazy utworzono dwóch użytkowników: SA (Super Administrator) oraz Student. W późniejszym etapie implementacji zdecydowałem się zaniechać korzystania z użytkownika Student, gdyż nie przynosi on żadnych korzyści. Aby studenci mogli używać wszystkich funkcji systemu, w tym tworzyć tabele, schematy, indeksy zdecydowałem się odblokować wszystkie zasoby tej instancji bazy na ich użytek. Natomiast ochrona głównego schematu została oddelegowana do samej warstwy dostępu do danych. To aplikacja dba o spójność danych testowych. Do tego celu wykorzystałem prostą metodę odwracania wszystkich zmian po każdej transakcji w głównym schemacie. W ten sposób żadnej użytkownik systemu nie ma szans usunięcia choćby krotki danych. Wszystkie dane są zachowane w skrypcie, który jest przechowywany na samym serwerze i może być modyfikowany tylko przez programistę lub administratora serwera.

3.4.2. Instancja ADMINISTRATOR

Druga instancja zakłada, że przywileje będą podzielone na dwie, a w przyszłości może na trzy grupy.

3.5. Techniki, wzorce architektoniczne oraz programistyczne

3.5.1. DAO

3.5.2. MVC

3.5.3. Subscribe

3.5.4. Listener

3.5.5. Singleton

3.5.6. Wyrażenia Lambda

Pozi

Rozdział 4

4.1. Schemat działania

4.1.1. Przykładowy workflow

Zaci

4.1.2. Weryfikacja odpowiedzi

Zaci

4.1.3. Przyznanie punktów

Zaci

4.1.4. Błędna odpowiedź

4.2. Opis wybranych funkcjonalności

4.2.1. Praca grupowa

4.2.2. Praca indywidualna

4.2.3. Forum DISQUS

4.2.4. Komunikaty

4.2.5. Materiały naukowe

4.2.6. HELP

Pozi

4.3. Szczegółowy opis implementacji

Okablowanie Poziome

Pozi

4.4. Dokumentacja techniczna

API

Pozi

Rodział 5

5.1. Przykłady działania aplikacji oraz dokumentacja użytkownika

Okablowanie Poziome

5.2. Praca zespołowa

Pozi

Rodział 6

6.1. Bezpieczeństwo

6.1.1. WebSocket Authentication

6.1.2. Uprawnienia do bazy danych

6.1.3. Izolacja

6.1.4. Atomowość

6.1.5. Transakcyjność

6.2. Dostępność i niezawodność

6.2.1. Load Balancing

6.2.2. Hot backup

6.2.3. Incremental backup

Pozi

6.3. Analiza wydajności

6.3.1. Porównanie WebSocket vs HTTP

6.3.2. H2 vs Oracle

6.3.3. H2 vs MS SQL Server

Pozi

Rozdział 7

7.1. Perspektywy rozwoju

Moduł logowania

Resetowanie haseł

Przypominanie zapomnianych haseł

Single Sign-On, Kerberos, LDAP, Active Directory

Blokowanie dostępu

Czarna lista

Moduł oceniania

Wykresy ocen, SVG

Rozkład normalny, SVG

Statystyka grupy, roku

Archiwum ocen

Wykrywanie plagiatu

Rozbudowa panelu administracyjnego

Automatyczne dodawanie pytań

Modyfikacja kont użytkowników

Analiza logów

Personalizacja wyglądu

Integracja z platformą Moodle

Interaktywność

Zapamiętywanie wpisywanych zapytań

Podpowiedzi składni w czasie rzeczywistym

Podpowiadanie nazw obiektów

Analiza stylu i składni

Angielska wersja językowa.

7.2. Wnioski i podsumowanie

Okablowanie Poziome

Pozi

Spis rysunków.

Spis tabel.

Spis listingów

Bibliografia

[1] Jakub Kasprzak

Język SQL – historia, standardy

Styczeń 2013

URL <http://www.sqlpedia.pl/jezyk-sql-historia-standardy/>

[2] Wikipedia

<http://pl.wikipedia.org/wiki/SQL>

[3] <http://www.h2database.com/html/links.html#projects>

[4] Core J2EE Patterns - Data Access Object

<http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

[5] Mutli-their-architecture

http://en.wikipedia.org/wiki/Multitier_architecture

[6] Marijn Haverbeke

JavaScript I wszystko jasne

<http://www.bt4.pl/kursy/javascript/wszystko-jasne>

[7] Kurs jQuery. Część 1: Wprowadzenie. Zalety, podstawowe zasady, pierwszy skrypt i rozszerzenia

<http://webhosting.pl/Kurs.jQuery.Czesc.1.Wprowadzenie.Zalety.podstawowe.zasady.pierwszy.skrypt.i.rozszerzenia>

[8] STOMP The Simple Text Oriented Messaging Protocol

<https://stomp.github.io/>

[9] JSON, JavaScript Object Notation

<http://pl.wikipedia.org/wiki/JSON>

[10] Bootstrap

<http://www.altcontroldelete.pl/artykuly/jak-szybko-stworzyc-ladny-css-z-frameworkiem-bootstrap/>

[11] Bartosz Danowski Wstęp do CSS

<http://webmaster.helion.pl/index.php/pcss-wstep>

[12] Cezary Bronny Technologia JDBC w praktyce

<http://students.mimuw.edu.pl/~zbyszek/bazy-danych/JDBC-CB227567.pdf>

[13] W. Wheeler, J. White, Spring w praktyce, Manning Publication 2014

Załączniki