Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

Zaawansowane Techniki Bazodanowe

Krzysztof Koptyra, Dawid Tomaszewski

Raport końcowy

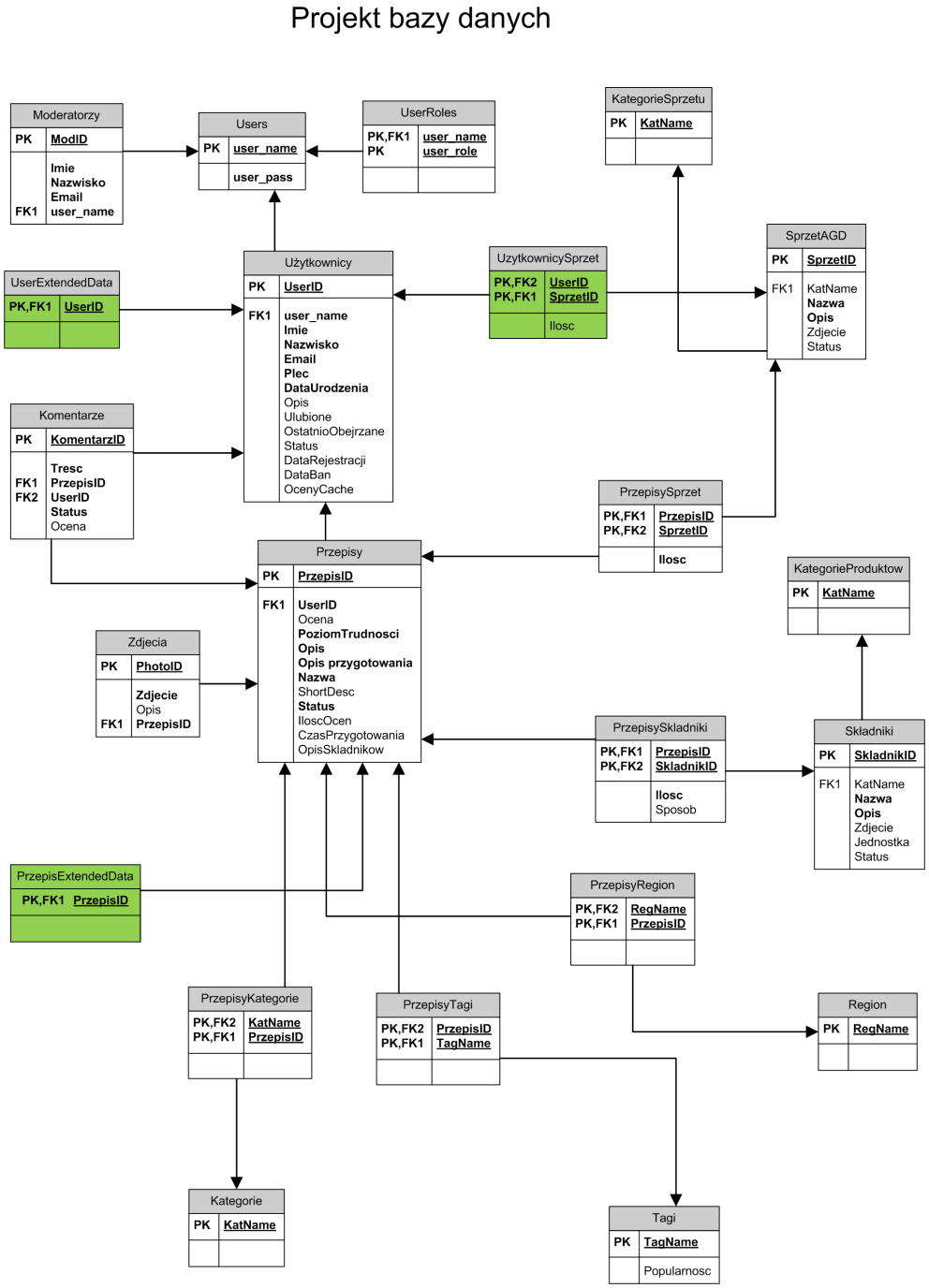
Prowadzący:
dr inż. Sebastian Ernst

Kraków 2011

Spis treści

1. Implementacja bazy danych.

Baza danych została utworzona zgodnie z poniższym diagramem:



Przykładowy fragment skryptu SQL tworzący tabelę użytkownika w bazie danych:

```
CREATE TABLE users (
        user_name VARCHAR(64) PRIMARY KEY NOT NULL,
        user_pass VARCHAR(64) NOT NULL
);
```

Przykładowy fragment pliku mapowania użytkownika:

```
package hibernate;

import java.util.HashSet;
import java.util.Set;

/**
 * Users generated by hbm2java
 */
public class Users  implements java.io.Serializable {


     private String userName;
     private String userPass;
     private Set serviceusers = new HashSet(0);

    public Users() {
    }


    public Users(String userName, String userPass) {
        this.userName = userName;
        this.userPass = userPass;
    }
    public Users(String userName, String userPass, Set
       serviceusers) {
       this.userName = userName;
       this.userPass = userPass;
       this.serviceusers = serviceusers;
    }

    public String getUserName() {
        return this.userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getUserPass() {
        return this.userPass;
    }
```

```
    public void setUserPass(String userPass) {
        this.userPass = userPass;
    }
    public Set getServiceusers() {
        return this.serviceusers;
    }

    public void setServiceusers(Set serviceusers) {
        this.serviceusers = serviceusers;
    }
}
```

Przykładowy fragment pliku XML zawierającego mapowanie użytkownika:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate
   Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/
   hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="hibernate.Users" schema="public" table="users">
    <id name="userName" type="string">
      <column length="64" name="user_name"/>
      <generator class="assigned"/>
    </id>
    <property name="userPass" type="string">
      <column length="64" name="user_pass" not-null="true"/>
    </property>
    <set inverse="true" name="serviceusers">
      <key>
        <column length="64" name="user_name"/>
      </key>
      <one-to-many class="hibernate.Serviceuser"/>
    </set>
  </class>
</hibernate-mapping>
```

2. Zdefiniowanie interfejsów do prezentacji, edycji i obsługi danych.

Aplikacja wykorzystuje standardowe formularze występujące w aplikacjach internetowych do pobierania od użytkownika niezbędnych danych (np. do wyszukiwania przepisów, podczas wprowadzania nowego przepisu czy przy wystawianiu komentarza).

3. Zdefiniowanie dokumentów do przetwarzania i prezentacji danych.

W aplikacji nie występują raporty ani dokumenty do przetwarzania danych. Ma ona charakter ogólnodostępnej dla użytkowników aplikacji internetowej. Można wskazać jedynie prezentację danych przy pomocy narzędzi dostarczonych przez JSF (np. tabele z danymi, etc.).

4. Zdefiniowanie panelu sterowania aplikacji.

Panel sterowania aplikacji został zaprojektowany w sposób ergonomiczny i estetyczny. Dzięki niemu obsługa aplikacji jest niekłopotliwa i intuicyjna dla użytkownika.

5. Zdefiniowanie makropoleceń dla realizacji typowych operacji.

W aplikacji Serwis Kucharski został użyty system mapowania relacyjno-obiektowego (ORM) Hibernate. Dzięki niemu typowe operację dostępu do danych, realizowane zazwyczaj poprzez zapytania SQL-owe, zostały uproszczone do odwoływania się bezpośrednio do obiektów. Znacząco ułatwia to pobieranie danych z bazy oraz ich modyfikację.

6. Uruchamianie i testowanie aplikacji.

W celu uruchomienia aplikacji niezbędny jest serwer, na którym zainstalowano Apache Tomcat w wersji 7.0 oraz bazę danych PostgreSQL 9.0. Problem ten nie dotyczy użytkowników aplikacji, gdyż jedynym wymogiem dla końcowego użytkownika jest komputer wyposażony w przeglądarkę internetową z dostępem do Internetu.

Aplikację testowano zarówno manualnie jak i poprzez zautomatyzowane narzędzia do testowania (m. in. Selenium, dzięki któremu wykonano wiele testów funkcjonalności serwisu).

7. Wprowadzanie danych.

Wprowadzanie danych w aplikacji Serwis Kucharski można podzielić na następujące etapy:

1. Automatyczne wprowadzanie danych przez skrypty SQL-owe, zapewniające początkowe dane w bazie danych. Jest ono w pełni zautomatyzowane i pozwala w krótkim czasie na stworznie początkowej struktury bazy danych serwisu. Ponieważ dane w skrypcie zostały sprawdzone i przetestowane, nie ma potrzeby weryfikacji ich poprawności.

2. Dane wprowadzane przez moderatorów i użytkowników serwisu, poprzez formularze na stronach serwisu. Jest ono dokonywane manualnie i jakkolwiek jest powolne z punktu widzenia pojedynczej osoby, to wielu użytkowników serwisu zapewni szybki przypływ danych. Dane są poddawane walidacji, dzieki czemu nie występuje ryzyko dodania do bazy danych mogących zakłócić prawidłowe działanie aplikacji.

8. Wdrażanie systemu do użytkowania.

Wdrożenie dotyczy jedynie administratora aplikacji, polega ono na skopiowaniu pliku WAR aplikacji do odpowiedniego folderu kontenera aplikacji Apache Tomcat oraz na uruchomieniu skryptów SQL-owych tworzących bazę danych. Po takich działaniach aplikacja jest gotowa do pracy.

9. Przeprowadzenie szkolenia użytkowników.

Aplikacja Serwis Kucharski ma charakter intuicyjnej aplikacji internetowej, nie jest więc wymagane szkolenie użytkowników. Każdy użytkownik potrafiący obsłużyć komputer PC powinien poradzić sobie z jej obsługą.

10. Zapewnienie dokumentacji technicznej i użytkowej.

Niniejsza dokumentacja powinna zaspokoić ciekawość użytkownika od strony technicznej (Projekt Konceptualny, Projekt Logiczny oraz Raport Końcowy), natomiast dokumentacja użytkowa nie jest niezbędna, ze względu na intuicyjność interfejsu użytkownika.

11. Zapewnienie obsługiwania systemu po wdrożeniu.

Obsługa aplikacji po wdrożeniu ogranicza się do zapewnienia stabilnego hostingu aplikacji i bazy danych. W przypadku wystąpienia problemów z aplikacją producent zapewnia pomoc techniczną.

12. Rozwijanie i modyfikowanie aplikacji.

Istnieje możliwość implementacji dodatkowych funkcjonalności do aplikacji Serwis Kucharski. Aplikacja została zaprojektowana w sposób umożliwiający niekłopotliwe dodawanie nowych możliwości oraz modyfikację obecnych.

Przykładowe funkcjonalności planowane do wdrożenia w następnych wersjach aplikacji:

- forum dla użytkowników,
- listy ulubionych przepisów,
- rozbudowane profile użytkowników,
- oceny komentarzy,
- informacje o kaloryczności posiłków,
- etc.

13. Opracowanie doświadczeń wynikających z realizacji projektu.

Realizowany projekt dostarczył autorom wielu doświadczeń związanych z projektowaniem aplikacji internetowych. Najważniejszą technologią użytą w projekcie była platforma Java Enterprise Edition połączona z frameworkiem JSF.

Java Server Faces okazał się być interesujący pod wieloma względami (zastosowanie wzorca MVC, wiele czynności zostało zautomatyzowanych w porównaniu do Java Server Pages – JSP, np. wyświetlanie tabel jest niezwykle proste i intuicyjne), jednak nie pozbawiony wad (wiele czynności bardzo prostych do wykonania w JSP okazało się być bardzo trudne lub wręcz niemożliwe do obejścia w JSF, przykładowo łączenie JSP z JSF lub system logowania z Apache Tomcat).

Kolejnym ciekawym doświadczeniem było użycie frameworka Hibernate do mapowania relacyjno-obiektowego. Dzięki niemu pozyskiwanie obiektów z bazy danych, ich modyfikacja czy dodawanie są znacząco uproszczone w porównaniu do ręcznego pisania zapytań SQL-owych. Kosztem, jaki trzeba było ponieść aby korzystać z tej technologii była dość pracochłonna konfiguracja i poznanie języka HQL (Hibernate Query Language).

Kolejnym doświadczeniem było użycie Apache Tomcata jako kontenera aplikacji. Jego wielkimi zaletami w porównaniu do większych serwerów aplikacji jak np. JBoss są niewielkie wymagania sprzętowe i mniej skomplikowana konfiguracja. Wadą okazały się trudności połączenia systemu autoryzacji oferowanej przez Tomcata z technologią JSF.

Niewątpliwie cennym doświadczeniem było używanie serwera SVN, co znacząco uławiało wieloosobową pracę nad aplikacją.

Nabyte doświadczenia z pewnością okażą się przydatne podczas realizowania kolejnych projektów informatycznych oraz będą owocować polepszaniem jakości wytwarzanego oprogramowania.

14. Opracowanie raportu końcowego.

Cały niniejszy dokument pełni rolę raportu końcowego.

15. Wykaz literatury, załączniki.

- *http://download.oracle.com/javaee/* – dokumentacja, tutoriale do Oracle Java EE,
- *http://java.sun.com/javaee/javaserverfaces/reference/api/index.html* – dokumentacja do JSF,
- *http://www.hibernate.org/docs* – dokumentacja do Hibernate,
- *http://www.postgresql.org/docs/* – dokumentacja do PostgreSQL,
- *http://tomcat.apache.org/tomcat-7.0-doc/index.html* – dokumentacja do Apache Tomcat 7.0,