UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Tenisový IS implementovaný pomocí Oracle ADF Web Fusion

Bc. David Pohl

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. David Pohl**
Osobní číslo:	**I10408**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Tenisový IS implementovaný pomocí Oracle ADF Web Fusion**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V první části diplomové práce bude popsáno vývojové prostředí JDeveloper a technologie, které se používají pro vytváření robustních webových aplikací (architektura JEE, Oracle RDBMS, JPA, JSF, Oracle ADF Web Fusion, Ajax atd.) Práce bude dále obsahovat popis jejich vzájemné spolupráce založené na modelu MVC. V závěru úvodní části bude provedeno kvalitativní srovnání Oracle ADF Web Fusion frameworku s jinými vybranými frameworky. Cílem praktické části bude analýza a návrh tenisového informačního systému za pomoci jazyka UML a následná implementace systému s využitím Oracle ADF Web Fusion frameworku. Aplikace bude obsahovat profily hráčů a turnajů, statistické údaje o výkonnosti hráčů, toku financí atd.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

Kyte, Thomas. Oracle návrh a tvorba aplikací. Brno : Computer Press, a.s., 2007. 80-251-0569-5.
Oracle. Oracle Application Development Framework - Oracle ADF. [Online] [2012] http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html.
Oracle. Java EE Tutorials. [Online] [2012] http://www.oracle.com/technetwork/java/javaee/documentation/tutorials-137605.html.

Vedoucí diplomové práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice
Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 19. 8. 2013 David Pohl

Poděkování

Rád bych poděkoval panu Ing. Zdeňku Šilarovi za rady a připomínky, které mi poskytl v průběhu vypracovávání mé diplomové práce.

Anotace

Tato diplomová práce se zabývá vývojem aplikace pomocí technologie Oracle ADF Web Fusion. Teoretická část obsahuje popis použitých technologií a kvalitativní porovnání Oracle ADF Web Fusion frameworku s vybranými Java frameworky. Praktická část je zaměřena na analýzu a návrh tenisového informačního systému pomocí jazyka UML a jeho následnou implementaci s využitím Oracle ADF Web Fusion frameworku.

Klíčová slova

Oracle ADF, JDeveloper, informační systém, Java EE, framework, UML

Title

Tennis IS implemented by Oracle ADF Web Fusion

Annotation

This diploma thesis deals with the development of applications using Oracle Fusion Web ADF. The theoretical part contains a description of the technology and quality compared to Oracle Fusion Web ADF framework with selected Java frameworks. The practical part is focused on the analysis and design of tennis information system using UML and its following implementation using Oracle Fusion Web ADF framework.

Keywords

Oracle ADF, JDeveloper, information system, Java EE, framework, UML

Obsah

1 Úvod

Téma Tenisový IS implementovaný pomocí Oracle ADF Web Fusion bylo zvoleno s cílem představit základní vlastnosti a výhody práce s Oracle ADF frameworkem. Diplomová práce je rozdělena do dvou částí.

V první, teoretické části jsou představeny technologie, které se používají pro vytváření robustních webových aplikací. Hlavním bodem teoretické části je porovnání vybraných Java frameworků a stručné představení základních rysů jednotlivých frameworků. Následně je na základě vymezených kritérií provedeno jejich vzájemné porovnání reprezentované tabulkou či grafem. Cílem této části je vytvořit vhodné podmínky pro rozhodnutí vedoucí k volbě vhodného Java frameworku.

Druhá část diplomové práce se zaměřuje na praktickou ukázku tvorby informačního systému pomocí Oracle ADF Web Fusion frameworku. Jak název práce napovídá, jedná se o tenisový informační systém. Z tohoto důvodu je provedena analýza tenisového prostředí, z jejíchž výsledků vycházejí prvotní návrhy. Pro implementaci systému je zvoleno vývojové prostředí JDeveloper, jehož součást tvoří mimo jiné Oracle ADF. Výsledkem by měl být funkční tenisový informační systém splňující základní požadavky, jako jsou například ukládání a editace dat nebo prezentace obsahu podle typu oprávnění.

Služby každého informačního systému jsou využívány vždy několika uživatelskými skupinami, které se od sebe odlišují úkony, jež smějí se systémem vykonávat. Někteří mohou obsah pouze prohlížet, další např. editovat a ti nejprivilegovanější smějí obsah vytvářet. Hlavní cílovou skupinu uživatelů tvoří především tenisoví fanoušci, kteří chtějí získat co nejvíce možných informací o svých oblíbených hráčích. Proto je jedním ze základních pilířů systému nabídka velkého množství různorodých statistik v podobě tabulek a grafů. Tyto informace však vytvářejí uživatelé, kterým je umožněno přihlásit se do systému. Přihlášený uživatel pak spravuje příslušnou část systému na základě jeho uživatelské role. Mezi základními funkcemi aplikace by nemělo chybět vytváření nových uživatelů, turnajů, zaznamenávání výsledků či editace vlastního uživatelského profilu.

2 Použité technologie

Při tvorbě informačních systémů a aplikací obecně je možné vybírat z široké nabídky technologií. Tyto technologie se od sebe odlišují především svým účelem, pro který jsou využívány. V případě, že je již jasně definována povaha aplikace, je nutné účelově vybrat technologie, které budou schopny zprovoznit příslušnou část aplikace. Například pro webové aplikace bude ve většině případů použit nějaký databázový systém či vývojová platforma apod. O konkrétním databázovém systému nebo konkrétní vývojové platformě v praxi rozhoduje většinou zákazník, mající určitou představu o produktu, nebo firma zabývající se danou problematikou. Pro tvorbu Tenisového IS byly technologie vybrány na základě jejich osvědčené vzájemné spolupráci.

2.1 Oracle RDBMS

RDBMS (Relational Database Management Systém) je databázový systém, jehož základem je relační model [1]. Umožňuje data ukládat, i s nimi manipulovat. RDBMS rozlišuje následující typy operací:

- logické,
- fyzické.

V případě logických operací aplikace určuje jaký obsah je požadován. Aplikace například požaduje vyhledat název oddělení, ve kterém pracuje příslušný zaměstnanec, nebo uložit nový záznam do tabulky.

U fyzických operací RDBMS určuje, jakým způsobem budou operace prováděny. Například při dotazování tabulky může databáze použít index pro nalezení řádků, načíst data do paměti a provést celou řadu kroků, než uživateli zobrazí výsledek. RDBMS ukládá a načítá data tak, že fyzické operace jsou pro databázové aplikace transparentní.

RDBMS dále implementuje objektově orientované funkce, jako jsou:

- uživatelem definované typy,
- dědičnost,
- polymorfizmus.

Oracle databáze tak rozšířila relační model na objektově-relační model, z čehož vznikl objektově-relační databázový systém (ORDBMS), který umožňuje ukládání komplexnějších *business* modelů do relační databáze.

Jedním z charakteristických rysů RDBMS je nezávislost fyzického uložení dat na logické struktuře dat. Oracle databáze obsahuje databázové schéma (kolekce datových struktur nebo objektů schématu), které je vlastněno databázovým uživatelem a má stejné jméno jako uživatel. Objekty schématu jsou uživatelem vytvořené struktury, jež přímo odkazují na data v databázi. Mezi nejdůležitější objekty se řadí tabulky a indexy.

Tabulka popisuje entitu, například *zamestnanci*. Lze ji definovat názvem a sadou sloupců. Obecně platí, že každému sloupci je přiřazeno jméno a datový typ. Tabulka obsahuje sadu řádků a sloupce identifikované atributem, přičemž řádek představuje instanci entity. Pro každý sloupec lze definovat pravidla neboli integritní omezení. Například NOT NULL integritní omezení nedovolí v žádném řádku příslušného sloupce uložení prázdné hodnoty.

Nad sloupci lze vytvořit volitelnou datovou strukturu index, jenž může zvýšit výkon načítání dat. Indexy jsou užitečné v případech, kdy se aplikace často dotazuje na konkrétní řádek nebo rozsah řádků.

Veškeré operace s daty jsou v databázi Oracle prováděny pomocí příkazů SQL (Structured Query Language). Jedná se o deklarativní jazyk popisující, co je třeba udělat. Na rozdíl od procedurálních jazyků, jako je C, které popisují, jak by se měly věci udělat. SQL se používá pro vytváření tabulek, dotazů a pro úpravu dat v tabulkách. Dále umožňuje provádět následující činnosti:

- dotazování dat,
- vložit, aktualizovat a odstraňovat řádky v tabulce,
- vytvořit, nahradit, měnit a rušit objekty,
- řídit přístup k databázi a jejím objektům,
- záruka konzistence a integrity databáze.

SQL sjednocuje předcházející úkoly v jednom uceleném jazyce. Oracle SQL je implementace standardu ANSI (American National Standards Institute). Podporuje řadu funkcí, které přesahují standardní SQL.

Oracle RDBMS dále nabízí:

- PL/SQL - procedurální rozšíření Oracle SQL. Podporuje proměnné, podmínky, cykly a výjimky.
- Řízení transakcí (Transaction Management) - databáze musí zajistit, aby mohlo současně pracovat více uživatelů, aniž by si navzájem narušovali data.

2.2 MVC

Jedná se o softwarovou architekturu [2], která je složena ze tří nezávislých komponent:

- datový model,
- uživatelské rozhraní,
- řídící logika.

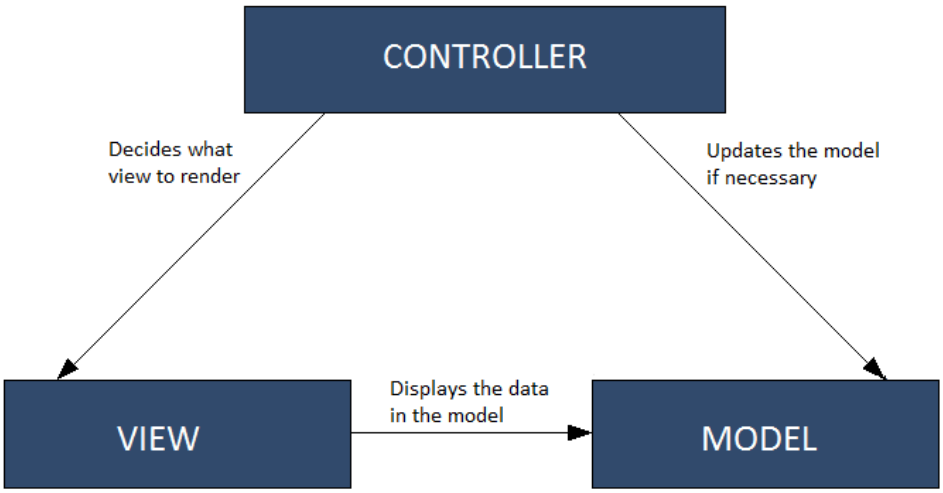
Změna některé z nich ovlivňuje ostatní komponenty pouze minimálně. Aplikace vytvořená podle architektury MVC vyžaduje vytvoření následujících komponent:

- Model (model) - reprezentuje data a *business* logiku aplikace.
- View (pohled) - zobrazuje uživatelské rozhraní.

- Controller (řadič) - má na starosti tok událostí a obecně aplikační logiku.

Pro pochopení vzájemné spolupráce jednotlivých vrstev je zde uveden jednoduchý příklad:

- V uživatelském rozhraní je vykonána nějaká akce (např. stisknutí tlačítka).
- Objekt uživatelského rozhraní informuje *Řadič* o provedené akci.
- Ten na základě dané akce přistoupí k *Modelu* a zaktualizuje ho (např. aktualizuje objednávku).
- *Model* zpracuje změněná data (vyhodnotí objednávku). Na základě aktualizace *Modelu* zobrazí komponenta *Pohled* příslušná data (např. vypíše obsah objednávky).
- Jak znázorňuje Obrázek 1, *Pohled* získává data přímo z *Modelu*, zatímco *Model* nepotřebuje žádné informace o komponentě *Pohled* - je na ní nezávislý.
- Uživatelské rozhraní čeká na další akci, která by celý cyklus znovu zahájila.



Zdroj: www.codeutil.files.wordpress.com

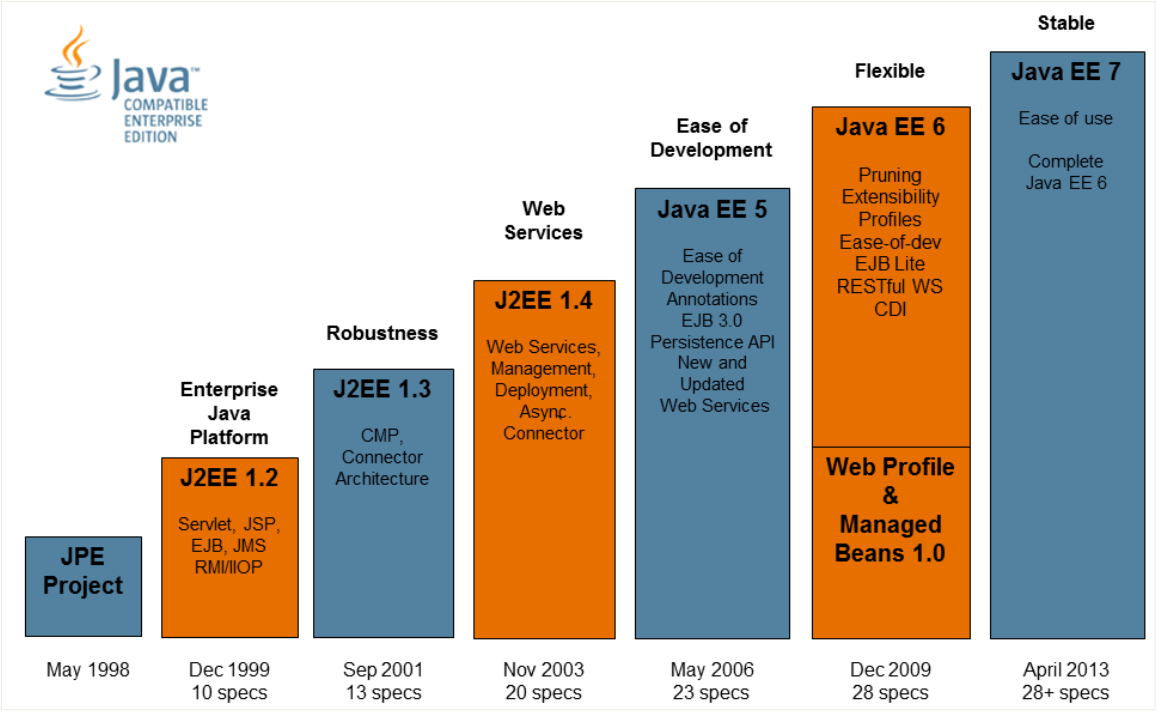
Obrázek 1 – MVC

2.3 Architektura JEE

Java Platform, Enterprise známá také jako Java 2 Enterprise Edition nebo J2EE je součást platformy Java. Využití nalezne především ve vývoji a provozu podnikových aplikací a informačních systémů [3].

Součástí platformy Java EE jsou specifikace pro:

- Java Servlets, Java Server Pages (JSP), Java Server Faces (JSF),
- Enterprise Java Beans (EJB), Spring,
- Java Connector Architecture (JCA), Hibernate,
- Java Messaging Services (JMS),
- portlety,
- webové služby.



Zdroj: www.blog.eisele.net

Obrázek 2 – Vývoj architektury JEE

Pro platformu Java EE jsou aplikace vyvíjeny na základě API (Application programming interface) a dalších částí definovaných v příslušných specifikacích. Prostředím pro běh těchto aplikací bývá tzv. Aplikační Server. Aplikace by měla být přenositelná. To znamená provozuschopná na jakémkoliv Aplikačním Serveru kteréhokoliv dodavatele.

Open Source aplikační servery:

- GlassFish,
- JBoss,
- JOnAS,
- Apache Geronimo,
- Apache Tomcat (částečná implementace).

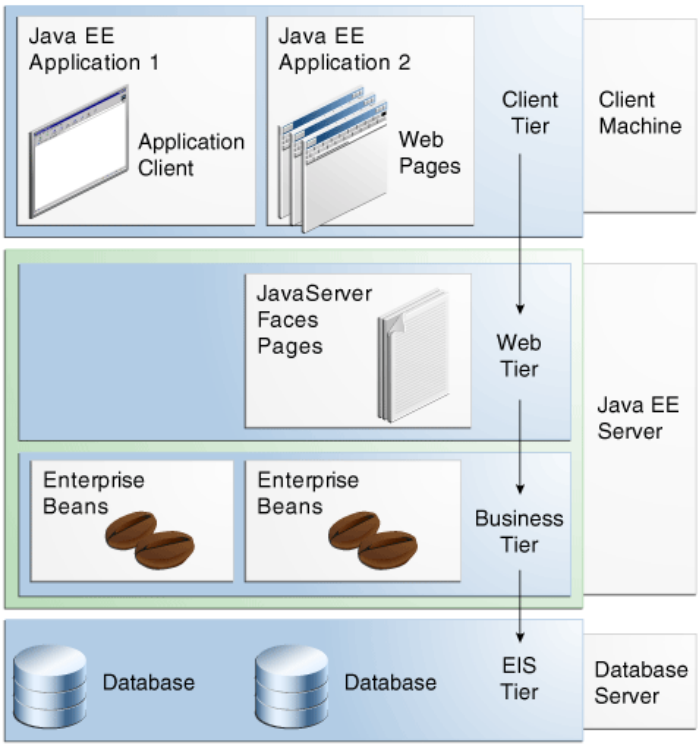
Komerční aplikační servery:

- IBM WebSphere,
- BEA WebLogic,
- Sun Java System Application Server
- Oracle AS.

Java EE aplikační servery:

- JBoss,
- IBM WebSphere,
- BEA WebLogic.

Funkce aplikace jsou rozděleny do izolovaných funkčních oblastí, tzv. vrstev (Obrázek 3). Vícevrstvé aplikace se obvykle skládají z klientské, střední a datové vrstvy. Vrstva klient obsahuje klientský program, který odesílá požadavky střední vrstvě. Ta obsahuje *business* funkce, které na základě požadavků zpracují příslušná data. Většinou se jedná o uložení dat do datového uložiště v datové vrstvě.



Zdroj: www.stackoverflow.com

Obrázek 3 – Architektura JEE

Klientská vrstva se skládá z aplikačních klientů přistupujících na Java EE server obvykle z jiných počítačů. Klienti vytvářejí požadavky na server, který je zpracovává a vrací zpět odpovědi. Klienty Java EE mohou být různé druhy aplikací, mnohdy se nejedná ani o Java aplikace. Může se jednat o webový prohlížeč, samostatnou aplikaci nebo jiné servery.

Vrstva Web se skládá z komponent, které se zabývají interakcí mezi klienty a *Business* vrstvou. Její hlavní úkoly jsou následující:

- Dynamické generování obsahu klientovi v různých podobách.
- Shromáždění informací od uživatelů klientské vrstvy a navrácení správných výsledků z komponent *business* vrstvy.
- Řízení navigace mezi stránkami na straně klienta.
- Udržení stavu dat v rámci session uživatele.

Business vrstva se skládá z komponent poskytující *business* logiku aplikace. *Business* logika je kód poskytující funkce pro konkrétní *business* oblasti, jako jsou např. finanční

průmysl nebo elektronické obchody. U správně navržené podnikové aplikace se funkční jádro nachází v komponentách *business* vrstvy.

Vrstva EIS (enterprise information systems) se skládá z databázových serverů, ERP systémů a dalších zdrojů dat. Tyto zdroje jsou obvykle umístěny na jiném stroji než JavaEE server a je k nim přistupováno pomocí komponent v *business* vrstvě.

2.3.1 JSP

Java Server Pages (JSP) technologie umožňuje webovým vývojářům a návrhářům rychle a snadno udržovat dynamické webové stránky. JSP technologie zrychluje vývoj webových aplikací nezávislých na platformě. Odděluje uživatelské rozhraní od generování obsahu, což umožňuje měnit vzhled stránky, aniž by docházelo ke změně obsahu [4].

Stránka JSP se skládá z:

- skriptovacích elementů,
- direktiv,
- instrukcí (akcí),
- elementů XML.

Tabulka 1 – Přehled elementů JSP

Přehled elementů			
	Deklarace	**Skriptlet**	**Expression**
Znaky uzavření	<%! %>	<% %>	<%= %>
Obsahuje	Jednu nebo více deklarací v jazyce Java.	Kód v jazyce Java.	Jeden výraz v jazyce Java.
Účel	Vytvoří název a může mu přiřadit hodnotu.	Informuje systém, aby vykonal určitou akci.	Vrací hodnotu.
Spuštění	Při první návštěvě stránky nebo v okamžiku, kdy kontejner JSP znovu inicializuje stránku.	Kdykoliv, kdy někdo navštíví stránku.	Kdykoliv, kdy někdo navštíví stránku.

Zdroj: cs.wikipedia.org

Pro webové vývojáře nebo návrháře se znalostí HTML přináší JSP následující výhody:

- Použití JSP technologie bez znalosti jazyka Java. JSP lze použít bez znalosti Java skiptletů. Ačkoliv už skriptlety nejsou zapotřebí pro generování dynamického obsahu, jsou stále podporovány kvůli zajištění zpětné kompatibility.
- Rozšíření JSP jazyka: vývojáři Java knihoven mohou rozšířit jazyk JSP o handlery využívající nové, mnohem jednodušší a čistší API. Tím vzroste počet připojitelných, znovupoužitelných knihoven tagů, což snižuje množství kódu potřebného pro napsání výkonné webové aplikace.
- Snadné psaní a údržba stránek: JavaServer Pages Standard Tag Library (JSTL) EL (Expression Language) byl aktualizován a integrován do JSP technologie. EL tak může být používán místo skriptletů.

2.3.2 Java Beans

Jedná se o opakovaně použitelné softwarové komponenty, s kterými lze vizuálně manipulovat. V podstatě se jedná o Java třídy dodržující určité konvence. Java Beans jsou perzistentní, mají schopnost ukládat, uchovávat a obnovovat svůj stav. Java Beans funkce, jako jsou vlastnosti, události a metody, jsou řízeny builder nástrojem. Builder nástroj je platforma umožňující vývojářům pracovat s Java Beans. Prostřednictvím režimu návrhu tohoto nástroje může vývojář přizpůsobit vzhled beany (změnu jejího chování), interakci s jinými beanami nebo je zakomponovat do apletů, servletů a aplikací [5].

Java Beans by měly dodržovat následující konvence:

- výchozí bezparametrický konstruktor,
- poskytnutí getter a setter metod,
- zavedení java.io.Serializable, neboť umožňuje aplikacím a frameworkům ukládat a obnovovat stav Java Beans.

2.3.3 EJB

Enterprise JavaBeans (EJB) je softwarová komponenta poskytující čisté Java prostředí pro vývoj a provoz distribuovaných aplikací. Jedná se o softwarové moduly, které obsahují *business* logiku aplikace. Nacházejí se v běhovém modulu nazývaným EJB kontejner, který poskytuje velkou řadu rozhraní a služeb EJB, včetně bezpečnosti a podpory transakcí. Vzhledem k tomu, že EJB disponuje vlastností přenositelnosti, může vývojář snadno vytvářet nové aplikace na již existujících beanách [6].

K dispozici jsou tři typy EJB:

- Session beans - provádí různé operace, výpočty nebo přístupy k databázi.
- Entity beans - reprezentují data, která mohou být řádkem v databázové tabulce.
- Message driven beans - jsou generovány pro zpracování JMS (Java Messaging Service) zpráv.

2.3.4 JSF

JSF (JavaServer Faces) technologie zahrnuje sadu API rozhraní pro reprezentaci komponent uživatelského rozhraní a řízení stavu. Dále zpracovává události, ověřuje vstupy, definuje stránkovou navigaci, umožňuje internacionalizaci a dostupnost. Hlavní myšlenkou této technologie, vyvinuté společností Sun Microsystems, je možnost čistějšího vývoje webových aplikací.

Hlavním cílem JSF je snadnost použití. JSF architektura jasně definuje oddělení aplikační logiky od samotné prezentace. Tato konstrukce umožňuje členům vývojového týmu zaměřit se pouze na příslušnou část vývojového procesu.

Uživatelské rozhraní je definováno speciálními XML tagy, kterým standardní Java Beans předávají data k zobrazení. XML tagy se importují z tzv. TLD (Tag Library Description) souborů. TLD soubory jsou součástí příslušných JavaServer Faces knihoven. Jednotlivým tagům jsou v rámci dané knihovny přiřazeny Java třídy.

Aplikační logika se zakládá na konfiguraci webové aplikace a implementaci backing bean. Konfigurace JSF není složitá. Stačí vytvořit soubor faces-config.xml v adresáři WEB-INF. Zde se nakonfiguruje mapování backing bean na jména, která se budou využívat pro odkazy na data v definicích GUI [7].

2.3.5 JPA

JPA neboli Java Persistence API je Framework programovacího jazyka Java, jenž usnadňuje práci s ukládáním objektů do databáze pomocí objektově relačního mapování (ORM). Data v databázi jsou reprezentovány objekty, kterým říkáme entity. Entitní třídy reprezentují tabulky v databázi, přičemž instance těchto tříd představují konkrétní řádky v příslušné tabulce [8].

Mezi implementace JPA patří následující produkty:

- Oracle Toplink,
- Hibernate,
- OpenJPA.

2.4 AJAX

AJAX (Asynchronous JavaScript and XML) představuje označení pro technologie vývoje interaktivních webových aplikací, které mění obsah stránek bez nutnosti jejich opětovného načítání. Oproti běžným webovým aplikacím vytvářejí uživateli příjemnější prostředí. Na druhou stranu ale vyžadují použití moderních webových prohlížečů.

Ajax aplikace jsou vyvíjeny ve spolupráci s technologiemi HTML (nebo XHTML) a CSS, které se starají o prezentaci informací. Dále pro zobrazení dynamických změn prezentovaných informací kooperuje s DOM (Document Object Model) a JavaScriptem. A o asynchronní výměnu dat s webovým serverem je využit XMLHttpRequest [9].

Výhody

- Odstranění nutnosti znovunačtení a překreslení celé stránky.
- Větší plynulost práce, blížící se běžným desktopovým aplikacím.
- Potenciál snížit zátěž na webové servery a síť obecně.

Nevýhody

- Může zvýšit počet vyměňovaných HTTP požadavků.
- Webové stránky se chovají jako plnohodnotné aplikace se složitou vnitřní logikou.
- Nelze předávat URL stránky, ve kterých už bylo pomocí technologie AJAX něco „naklikáno“.
- Síťová latence, potřeba komunikace přes Internet má negativní vliv na rychlost odezvy a interaktivitu UI.
- Předpoklad moderního prohlížeče podporujícího potřebné technologie.

2.5 XML

Extensible Markup Language je textový značkovací jazyk, který se na webu postupně stává standardem pro výměnu dat [10]. Podobně jako u HTML, jsou data identifikována pomocí tagů, které se uzavírají do ostrých závorek. Všeobecně jsou tagy známé jako značky (markup). XML dokument je Unicode text, v Česku obvykle kódovaný jako UTF-8. Správně strukturovaný xml dokument musí mít minimálně následující vlastnosti:

- Obsahuje právě jeden kořenový (root) element.
- Neprázdné elementy jsou ohraničeny startovací a ukončovací značkou. Prázdné elementy lze označit tagem „prázdný element“.
- Hodnoty atributů jsou uzavřené v uvozovkách.
- Elementy mohou být vnořeny, ale nesmí se překrývat.

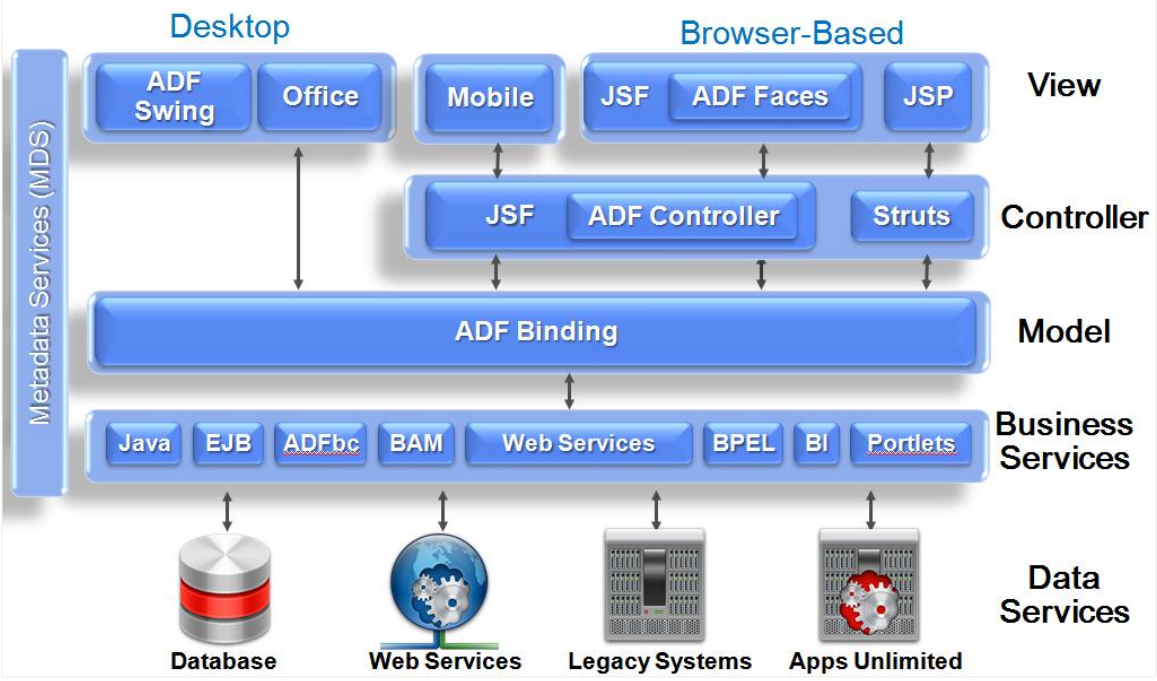
3 Oracle ADF Web Fusion Framework

Oracle Application Development Framework (ADF Oracle) je end-to-end aplikační framework, který je postaven na Java platformě, Enterprise Edition (Java EE) standardech a open-source technologiích. Oracle ADF lze použít pro implementování podnikových řešení, která budou schopna hledat, zobrazovat, vytvářet, upravovat a ověřovat data pomocí internetu, bezdrátové sítě nebo webové služby. Oracle ADF zjednodušuje a urychluje rozvoj tím, že umožňuje uživatelům soustředit se na logiku tvorby aplikací spíše, než na kódování. Používá se spolu s Oracle JDeveloper a Oracle ADF, což poskytuje prostředí pokrývající celý životní cyklus vývoje od návrhu až po nasazení [11].

3.1 Oracle ADF architektura

Aplikace vytvořené Fusion Web technologií dosahují čistého oddělení *business* logiky, stránkové navigace a uživatelského rozhraní díky důrazu na MVC architekturu, jež je znázorněna na Obrázek 4.

- Vrstva model představuje datové hodnoty vztahující se k aktuální stránce.
- View vrstva obsahuje UI stránek, což umožňuje zobrazení nebo modifikaci dat.
- Vrstva controller zpracovává vstupní data od uživatele a určuje navigaci stránek.
- Vrstva business service zpracovává přístup k datům a zahrnuje business logiku.



Zdroj: www.docs.oracle.com

Obrázek 4 – Oracle ADF architektura

Jádrem frameworku je vrstva Model, jež umožňuje jednotný způsob, jak navázat jakékoliv uživatelské rozhraní k libovolné *business* službě bez nutnosti psaní kódu. Ostatní moduly, které tvoří technologii Fusion web aplikace jsou:

- ADF Business Components - zjednodušují vytváření *business* služeb.
- ADF Faces rich client - nabízí bohatou knihovnu s podporou AJAX UI komponent pro webové aplikace založených na JavaServer Faces (JSF).
- ADF Controller - integruje JSF s ADF modelem. ADF Controller rozšiřuje standardní JSF Controller dalšími funkcemi, jako je například možnost opakovaného použití Task Flows.

3.1.1 ADF Business Components

Při vytváření servisně orientovaných Java EE aplikací se implementuje základní *business* logika jako jedna nebo více *business* služeb. Tyto *backend* služby poskytují klientům způsob k dotazování, vkládání, upravování a odstraňování *business* dat vyžadovaných při prosazování vhodných *business* pravidel. ADF *Business* komponenty jsou předem vytvořené objekty aplikace, které urychlují poskytnutí vysoce výkonných, databázově zaměřených služeb. Poskytují připravené implementace Java EE návrhových vzorů a ověřené postupy.

Entity object

Entity object představuje řádek v databázové tabulce. Zjednodušuje modifikaci údajů pomocí operací jazyka DML (data manipulation language). Zapouzdřuje *business* logiku, čímž zajistí dodržování *business* pravidel. Umožňuje vytvořit vazby mezi jednotlivými *entity objects*, tak jak jsou v podkladovém databázovém schématu. Tím vytváří vrstvu *business* doménových objektů, které lze použít ve více aplikacích.

View object

View object představuje SQL dotaz a zjednodušuje práci s jeho výsledky. Jazyk SQL umožňuje spojit, filtrovat, řadit a agregovat data do požadovaného tvaru podle dotazu koncového uživatele, zastoupeným v uživatelském rozhraní. To zahrnuje schopnost odkazovat na *view object* jinými *view objects*, čímž se vytvoří *master-detail* hierarchie. Když koncoví uživatelé mění data v uživatelském rozhraní, *view objects* spolupracují s *entity objects* kvůli ověření a uložení změn.

Application module

Aplikační modul je transakční komponenta, jež se používá pro práci s aplikačními daty. Definuje datový model spolu s procedurami a funkcemi (tzv. servisními metodami) týkajících se logické jednotky práce, vztahující se k požadavku koncového uživatele [12].

3.1.2 ADF Model Layer

Vrstva model odděluje implementaci *business* služeb a poskytuje jednotné programovací rozhraní pro různé typy služeb. Datové ovládací prvky poskytují toto rozhraní použitím standardních metadatových rozhraní. Ta popisují operace služeb a shromažďuje údaje o vlastnostech, metodách a zúčastněných typech. V JDeveloperu jsou funkce a vlastnosti *business* služeb označeny ikonami na panelu *Data Controls panel*. Přetáhnutím na stránku se vytvoří UI komponenty. JDeveloper automaticky vytvoří vazby mezi UI komponentami

a službami. Za běhu přečte vrstva model informace popisující datové ovládací prvky a datové vazby od příslušných XML souborů a implementuje obousměrné spojení mezi uživatelským rozhraním a *business* službou.

Oracle ADF poskytuje *Data Control* implementace pro většinu běžných *business service* technologií. Použitím JDeveloper a Oracle ADF společně s *drag and drop* vázáním dat umožňuje vybudovat uživatelské rozhraní. Kromě podpory ADF aplikačních modulů, poskytuje ADF Model podporu pro následující technologie:

- Enterprise JavaBeans (EJB) session beans and JPA entities,
- JavaBeans,
- Web services,
- XML,
- CSV.

3.1.3 ADF Controller

Ve vrstvě Controller, kde je řízení toku webové aplikace klíčovým prvkem, nabízí ADF Controller vylepšenou navigaci a stavovou správu modelu. JDeveloper umožňuje deklarativně vytvořit *Page Flows*, kde lze definovat řízení aplikace mezi stránkami, metodami *Managed beans* nebo nastavit volání do jiných *Task Flows*.

Tyto *Task Flows* mohou být opětovně používány, vnořeny ve stránkách nebo v jiném *Task Flow*. Pokud jsou vnořené ve stránkách, stávají se z nich regiony, které obsahují vlastní sadu navigací, což umožňuje uživatelům zobrazit několik různých stránek a funkcí, aniž by opustili hlavní stránku.

3.1.4 ADF Faces Rich Client

ADF Faces rich client (dále ADF Faces), je sada standardních komponent JSF, které obsahují vestavěnou funkci AJAX. AJAX je kombinací technologií JavaScript, dynamického HTML (DHTML), XML, a XMLHttpRequest komunikačních kanálů. Tato kombinace umožňuje zobrazení požadavků vykonaných na straně serveru bez nutnosti opětovného načtení celé stránky.

Zatímco AJAX umožňuje aplikacím používat standardní internetové technologie, JSF poskytuje řízení na straně serveru, což snižuje závislost na množství JavaScriptu typickou pro běžné AJAX aplikace. K dosažení těchto *front-end* schopností, používají ADF Faces komponenty vykreslovací *kit*, který zpracovává zobrazení komponent. Dále obohacuje JavaScript o objekty potřebné pro speciální funkce. Tato vestavěná podpora umožňuje vytvářet moderní aplikace bez nutnosti kódování JavaScriptu nebo potřeby rozsáhlé znalosti jednotlivých technologií na *front-endu* či *back-endu*.

ADF Faces poskytuje více než 150 *rich* komponent, včetně hierarchických datových tabulek, stromových menu apod. Dále poskytují datové vizualizační komponenty, jako jsou například dynamické grafy. Každá komponenta podporuje úpravy a stylování spolu s internacionalizací a dostupností [11].

Oracle ADF podporuje kromě ADF Faces také následující zobrazovací technologie:

- Apache MyFaces Trinidad - jedná se o otevřený zdrojový kód od Oracle darovaný firmě Apache Software Foundation. Tvoří základ pro komponenty ADF Faces.
- Java Swing a Swing ADF - ADF Swing je vývojové prostředí pro vytváření aplikací Java Swing, které používají ADF Model vrstvu.
- ADF Mobil - framework pro tvorbu mobilních aplikací postavených na komponentním modelu JSF.
- Microsoft Excel - umožňuje vytvářet tabulky, které jsou vázány na data pomocí stejných principů jako ostatní zobrazovací technologie.

4 Porovnání vybraných frameworků

Zvolit vhodný framework pro vývoj aplikace nemusí být vždy snadné, obzvlášť při absenci předchozích zkušeností. V této kapitole budou představeny specifické rysy několika vybraných Java frameworků. Podle zvolených kritérií bude provedeno porovnání jednotlivých frameworků. Na závěr budou detailněji vyzdviženy rozdíly mezi těmi nejpoužívanějšími a Oracle ADF. K porovnání byly vybrány:

- JSF (popsán v kapitole 2.3.4),
- Spring Framework,
- Apache Struts,
- Stripes,
- Apache Tapestry,
- Apache Wicket.

4.1 Oracle ADF

Oracle Application Development Framework je *end-to-end* Java EE framework, který pokrývá pracovní postupy struktury MVC architektury. Nabízí jak vizuální tak i deklarativní přístupy při budování J2EE požadovaného řešení a poskytuje mnoho předpřipravených návrhových vzorů a rozmanitých vizuálních nástrojů, které na oplátku pomáhají podporovat rychlý vývoj podnikových aplikací. Infrastruktura, kterou poskytuje Oracle ADF, pomáhá vývojáři vybudovat kvalitní řešení za pomoci komponent uživatelského rozhraní a *business service* komponent, jež komunikují s databází. Oracle ADF poskytuje zdrojový kód pro podnikové infrastruktury a vyžaduje pouze deklarativní konfigurace, čímž minimalizuje nutnost jakéhokoliv kódován [13].

Vlastnosti Oracle ADF:

- výkonné komponenty pro bohaté podnikové aplikace,
- vylepšené řízení stránkové navigace a operací,
- kontextové akce (události),
- desktop integrace,
- podpora vlastních komponent,
- *drag-and-drop* funkce,
- těsná integrace s webovými službami,
- deklarativní vázání dat,
- deklarativní *business* služby,
- deklarativní *end-to-end* zabezpečení,
- deklarativní přizpůsobení aplikace,
- opakovatelná použitelnost,
- nezávislá platforma,
- výborné vývojové prostředí (JDeveloper),
- produktivita vývojářů.

4.2 Spring Framework

Jedná se o open source framework pro vývoj J2EE aplikací. První verze byla vydána Rodem Johnsonem v roce 2002. Později byl za pomoci Juergena Hoellera rozšířen a uvolněn pod názvem Spring Framework [14].

Spring Framework může být používán jakoukoliv Java aplikací. V Java komunitě se stal populární jako alternativa k EJB nebo jako jeho nadstavba. Hlavním cílem jeho vzniku bylo usnadnění vývoje podnikových aplikací:

- Odstranění těsných programových vazeb využitím návrhového vzoru Inversion of Control.
- Umožňuje zvolit si vhodnou implementaci *business* vrstvy pro aplikační architekturu.
- Podporuje implementace komponent pro přístup k datům (JDBC, ORM, Hibernate apod.).
- Díky abstrakci zjednodušuje používání dalších částí J2EE.
- Umožňuje správu a konfiguraci řídících *business* komponent.

Inversion Control

Návrhový vzor Inversion of Control označován jako IoC kontejner, funguje na principu přesunutí zodpovědnosti za objekty z aplikace na framework. Na jeho využití je postaveno jádro Springu.

Dependency Injection

Objekty lze získat prostřednictvím speciálního případu Inversion of Control, který se nazývá Dependency Injection (vsazování závislostí). Nabízí tři základní způsoby vložení objektů:

- Setter Injection,
- Constructor Injection,
- Interface Injection.

Framework vytváří objekty na základě načtení konfiguračního souboru ve formátu XML, který obsahuje jejich definice. Spring jakožto modulární framework umožňuje využití jen té části, která je momentálně vhodná k vyřešení daného problému. Jeho úkolem je zjednodušit návrh J2EE aplikací se zaměřením na architekturu aplikace místo na technologii. Dále se zaměřuje na snadnou testovatelnost, neinvazivnost a modulárnost.

4.3 Apache Struts

Apache Struts představuje open source framework pro tvorbu webových aplikací. Je převážně založený na technologiích JSP, servletech a JavaBeans. Proto je velmi spjatý s webovým kontejnerem. Důvod proč Struts přispívá k tvorbě kvalitní webové aplikace, je

ten, že nás nutí dodržovat stanovená pravidla. Ty mohou práci zpočátku omezovat, ale nakonec je právě díky nim velice efektivní.

Aplikace jsou založeny na modelu MVC, přičemž vrstva View je implementována pomocí JSP a zodpovídá za zobrazení výsledků uživateli. Vrstvu Controller reprezentuje servlet ActionServlet, který zachytává požadavky od uživatele a vybírá vhodné View pro navrácení dat. Vrstva Model, jež je implementována pomocí JavaBeans, obsahuje aplikační logiku a komunikuje s perzistentním uložištěm dat.

Struts dále nutí dodržovat standardy pro lokalizaci textů. Podle pravidla nepatří texty do programů, ale do speciálního konfiguračního souboru. Můžeme např. vytvořit soubor pro českou a anglickou verzi. O ošetřování formulářů se stará třída ActionForm, která validuje hodnoty ve vytvořeném formuláři. Na nesprávně vytvořený formulář upozorňuje chybovými hláškami [15].

4.4 Stripes

Framework Stripes je open source webový aplikační framework založený na vzoru MVC. Jeho podstata spočívá v odlehčenosti oproti Struts od některých Java technologií, jako jsou anotace a generika. To zdůrazňuje myšlenku, že použití jednoduchých konvencí snižuje konfigurační zatížení frameworku. V praxi to znamená, že aplikace nepotřebuje téměř žádné konfigurační soubory, čímž se redukuje doba vývoje a údržby kódu [16].

Hlavní cíle:

- Ulehčit vývoj webových aplikací.
- Poskytnout jednoduché, avšak výkonné řešení běžných problémů.
- Rychle pochopitelný pro nové vývojáře.
- Snadná rozšiřitelnost frameworku bez nutnosti konfigurace posledních změn.

4.5 Apache Tapestry

Apache Tapestry je komponentově orientovaný webový framework. Jedná se o open source framework, který se koncepčně podobá již zmíněnému JSF. Klade důraz na jednoduchost, snadnost použití a produktivitu při vývoji. Využívá modulární přístup k vývoji webových aplikací tím, že má silné vazby mezi komponentami uživatelského rozhraní nacházejících se na webové stránce a jejich odpovídajícími Java třídami [17].

4.6 Apache Wicket

Apache Wicket je odlehčený komponentově založený webový aplikační framework. Svou koncepcí se podobá frameworkům JSF a Tapestry. Wicket není tradičním MVC frameworkem, svým vzorem se více podobá GUI frameworkům jako je např. Swing. Wicket aplikace se skládají z komponent využívající *listener* delegáty, které reagují na HTTP požadavky předané odkazem nebo formulářem. Podobně jako Swing komponenty reagují na myš a klávesy.

Použitím XHTML pro šablony vynucuje oddělení prezentace od aplikační logiky a umožňuje upravovat šablony pomocí konvenčních WYSIWYG návrhových nástrojů. Každá komponenta je vázána na XHTML prvek, a tím se stává zodpovědnou za jeho zobrazení v konečném výstupu [18].

4.7 Vzájemné porovnání

Programátorovi, který nemá žádné zkušenosti s některým z uvedených frameworků, může činit obtíže vybrat si ten správný, jenž by právě jemu vyhovoval. Ke správnému výběru pomohou vyhodnocení na základě následujících kritérií:

- cena,
- popularita,
- podpora komunity,
- rychlost vytvoření prototypu,
- počet pracovních příležitostí,
- dostupná literatura.

Jelikož Oracle ADF jako jediný z uvedených frameworků není zcela zdarma, bude nejprve provedeno vzájemné porovnání bezplatných frameworků, ze kterých se na základě hodnocení vyberou zástupci, s kterými bude dále porovnáván Oracle ADF Framework [19].

Tabulka 2 – Obecné porovnání frameworků

	Struts	**Spring**	**Stripes**	**JSF**	**Wicket**	**Tapestry**	**Oracle ADF**
Popularita (počet Google vyhledání / měsíc)	110 000	18 100	1 600	**165 000**	4 400	9 900	-
Komunita (poslední vydání)	16.6.2013	**6.8.2013**	17.5.2012	13.6.2013	10.7.2013	29.4.2013	1.7.2013
Počet pracovních příležitostí (jobs.cz)	5	**81**	0	4	16	0	4
Počet pracovních příležitostí (monster.com)	**456**	185	17	317	10	11	47

Zdroj: www.openlogic.com

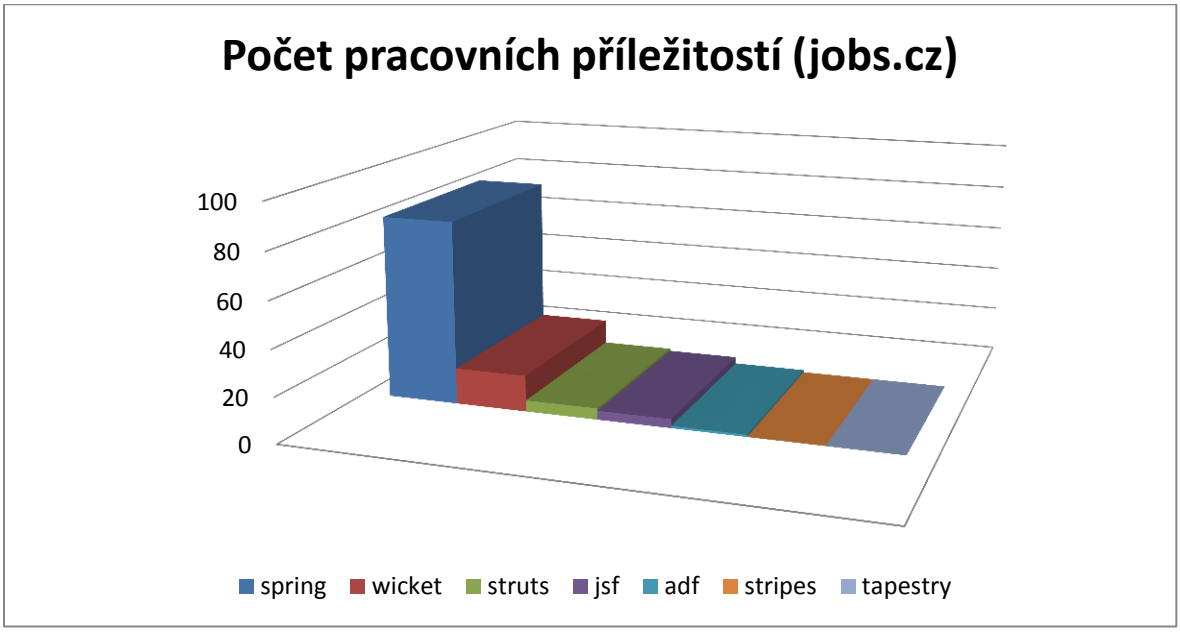
Pokud by se programátor rozhodoval striktně na základě **popularity** (měřeno jako počet vyhledání frameworku za měsíc na serveru google.com), zvolí JSF. Při této metodě výběru si je nutné uvědomit, že Sun, IBM a Oracle investovali spoustu prostředků do reklamy, aby

se JSF dostal do povědomí. Tudíž je popularita velmi ovlivněna marketingovými rozpočty. V případě rozhodování podle aktivity příslušné **komunity** založené na tom, jak často vycházejí nejnovější verze. Povedou kroky k frameworku Stripes.

Jedním z hlavních důvodů pro výběr frameworku je snížení množství kódu, který je nutné napsat, nebo zvýšení produktivity vývojového týmu. Bohužel, měření produktivity vývojového týmu je nereálné. Různí vývojáři píší kód různou rychlostí. Někteří vývojáři se učí velmi rychle, jiným to trvá déle. Proto se kritériem, založeném na rychlosti vytvoření prototypu, nelze příliš řídit.

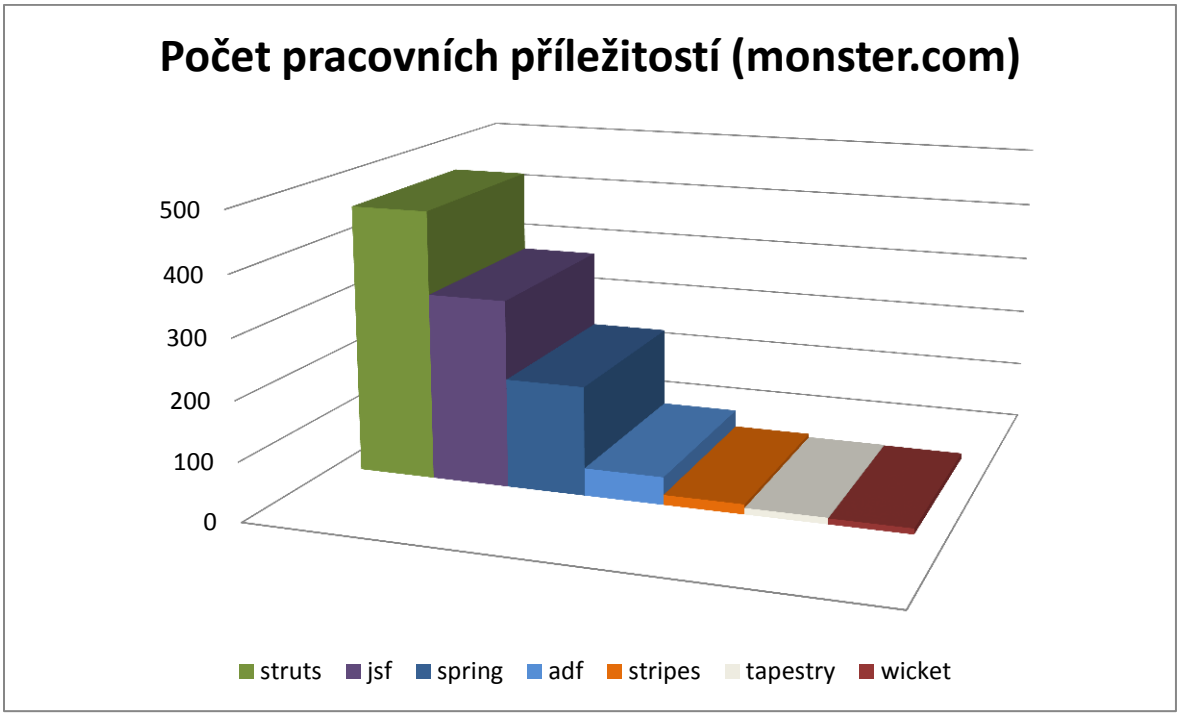
Obecně platí, že dostupnost dokumentace je běžná, a tak by bylo dobré kritérium **dostupné literatury** zúžit pouze na počet knih věnovaných příslušnému frameworku. Za těchto podmínek je podle serveru amazon.com vítězem JSF framework. Na druhou stranu vyšší počet knih může také znamenat, že je obtížné daný framework pochopit. S dostupností česky psané literatury už je to o poznání horší. Například o Oracle ADF není dostupná žádná.

Ani jedno z doposud uvedených kritérií pravděpodobně programátora nepřesvědčí do takové míry, aby se jednoznačně rozhodl pro konkrétní framework. Posledním kritériem **je počet nabídek pracovních pozic**. Měření probíhala na pracovních portálech jobs.cz a monster.com. Výsledky měření jsou přehledně znázorněny pomocí Graf 1 a Graf 2. Pro český trh práce má největší perspektivu znalost Spring frameworku. Pro zahraniční pak znalost Struts frameworku.



Zdroj: autor

Graf 1 - Počet pracovních příležitostí (jobs.cz)



zdroj: autor

Graf 2 - Počet pracovních příležitostí (monster.com)

Na základě vyhodnocení posledního kritéria, které se jeví jako nejpřínosnější, byly pro srovnání s Oracle ADF vybrány frameworky Struts a Spring. V následující tabulce (Tabulka 3) jsou uvedeny některé významné technické (z hlediska vývoje softwaru) rozdíly mezi vývojem pomocí Oracle ADF, Struts a Spring.

Tabulka 3 - Technické porovnání frameworků

Kategorie	Struts / Spring Framework	Oracle ADF Framework
Framework	Založených na akcích	Založený na komponentách
Licence	Open source frameworky	Open source pro vývoj a licencovaný pro produkční verzi
Zdrojový kód	Vývojář musí psát kód pro všechny UI komponenty	Vestavěné UI komponenty
Připojení k databázi	Vývojář musí explicitně napsat kód pro připojení databáze	Vestavěná konfigurace pro databázové připojení

Kategorie	Struts / Spring Framework	Oracle ADF Framework
SQL	Vývojář musí napsat a vykonat SQL příkazy	Deklarativní přístup vytváření požadovaných SQL příkazů
Transakce	Vývojář se musí starat o řízení transakcí	Transakce jsou řízeny frameworkem
Uzavření připojení	Vývojář musí zavřít všechna otevřená připojení, jinak by to mohlo vést k vyvolání výjimky	Připojení se uzavře, pokud je rozsah platnosti mimo kontext
Navigace stránek	Navigace stránek je založena na XML konfiguračních souborech, o které by se měl starat vývojář	O navigaci stránek se stará Controller, pokud je deklarován
Validace	Každá validace by měla být napsána ve specifických validačních metodách a každá validační chyba by měla být řešena zvlášť a zároveň směřovat na příslušnou chybovou stránku	Sofistikované ověřování pomocí adaptérů poskytovaných ADF na různých úrovních: na úrovni UI, AM úrovni a na úrovni entit, vývojář může rozhodnout podle daného případu, v jaké části se bude validace nacházet
Ochrana	Uživatel musí navrhnout a vyvinout přihlašovací stránku	Deklarativní přístup k zabezpečení umožňuje vývojářům zajistit bezpečnost na různých úrovních: stránky, regiony, profily, komponenty
Autentizace	Vývojář musí ověřit uživatelské jméno a heslo porovnáním s údaji ze zdroje	I když to není přímo funkce ADF, WebLogic poskytuje různé způsoby ověřování, které lze snadno spojit s ADF aplikací: SQLAuthentication, SAMLAuthentication, LDAPAuthenntication, atd.
Autorizace	Vývojář musí poskytnout kontrolu přístupu a měl by ručně omezit uživatelům přístup pouze k příslušným business operacím	Bezpečnostní implementace umožňuje přístup k uživatelům registrovaným v aplikaci
Průvodce	Vývojáři nenajdou téměř žádné průvodce na pomoc při budování aplikace	ADF je o průvodcích, pokud uživatel správně využívá těchto průvodců, ADF se postará o veškerý kód

Kategorie	Struts / Spring Framework	Oracle ADF Framework
Rychlost vývoje	Pro vytvoření normálního CRUD operačního modulu, potřebují vývojáři v průměru 3 dny	Pro vytvoření normálního CRUD operačního modulu, potřebují vývojáři 10min za předpokladu řádné konfigurace AM, EO, VO
Servery	Frameworky mohou využít webové aplikační servery, jako jsou Apache Tomcat, JBoss, WebSphere atd.,	Vestavěné konfigurace Weblogic serveru velmi usnadňují nasazení

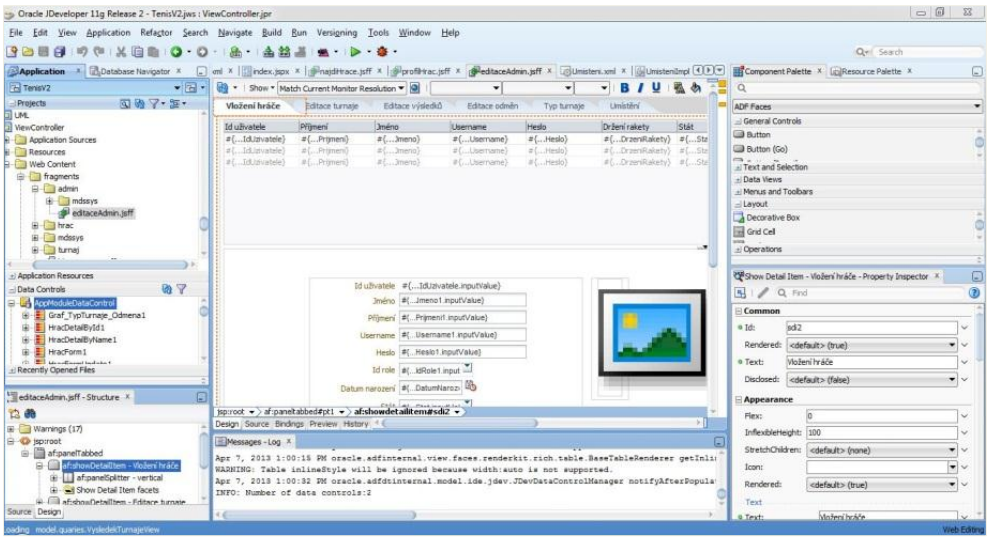
Zdroj: www.aptude.com

5 Použité nástroje

V kapitole Použité nástroje budou představeny nástroje, které nejenom umožnily kompletní realizaci informačního systému, ale také přinášely výhody v podobě snadnosti použití, přehledného vývojového prostředí apod. Mimo jiné se mezi jednotlivými druhy nástrojů objevují grafický editor, nástroj pro datové modelování nebo integrované vývojové prostředí pro tvorbu programů.

5.1 Oracle Fusion Middleware 11.1.2.3.0

Je přední platforma pro podnikové inovace určená pro podniky a *cloud*. Umožňuje podnikům vytvářet a používat výkonné obchodní aplikace s maximálním využitím informačních technologií - potenciál moderní hardwarové i softwarové architektury. Jeho součástí je vývojové prostředí JDeveloper 11.1.2.3.0 a aplikační server WebLogic Server 11gR1.



Zdroj: autor

Obrázek 5 - Prostředí JDeveloper

5.2 JDeveloper 11.1.2.3.0

Jedná se o integrované vývojové prostředí (IDE - Integrated Development Environment) od firmy Oracle Corporation. Slouží z velké části pro tvorbu programů v jazyce Java a pro vývoj v SOA (Service Oriented Applications) standardu. Dále poskytuje nástroje zajišťující kompletní životní cyklus aplikací. Tzn.:

- modelování,
- kódování,
- ladění,
- testování,
- nasazení.

Většina programátorů využívá prostředí JDeveloperu především pro vývoj aplikací v jazyce Java. Dále je zde ale k dispozici i podpora pro jiné jazyky jako např. Java Script, PHP, HTML, XML apod. Vývojové prostředí je založeno na standardech JDK (Java Development Kit) a J2EE (Java to Enterprise Edition).

JDeveloper poskytuje kromě standardní podpory pro programování i další možnosti především v oblasti integrace komponent (Java Bean, EJB, Web Services) do svých produktů (Oracle Database, Oracle Application Server) [20].

JDeveloper 11g je dostupný ve třech edicích: Java Edition, J2EE Edition a Studio Edition. Pro tvorbu aplikace byla využita edice Studio Edition (Obrázek 5).

Studio Edition

- ADF Databinding,
- ADF Faces,
- ADF Faces Skin Editor,
- ADF Mobile,
- ADF Business Components,
- ADF Swing,
- ADF Deployment,
- BPEL Designer,
- ESB Designer,
- Portlet Development,
- Portlet/JSF Bridge,
- Oracle BI Ee.

5.3 WebLogic Server 11gR1

Oracle WebLogic Server je Java EE aplikační prostředí, které podporuje nasazení kritických aplikací pomocí robustní, bezpečné a škálovatelné infrastruktury. Systém správy WebLogic Serveru zahrnuje širokou škálu činností, od vytváření a konfigurace serverových domén až k nasazení a zabezpečení aplikací, monitorování a řešení problémů serveru, aplikace a problémů s výkonem.

5.4 Oracle Database XE 11.2

Oracle Database 11g Express Edition (Oracle Database XE) je bezplatná verze relační databáze. Pro správu databáze lze použít intuitivní administrativní rozhraní založené na webovém prohlížeči. Rozhraní umožňuje vytvářet:

- tabulky, pohledy a další databázové objekty,
- import, export,
- zobrazení dat v tabulce,
- spouštění dotazů a skriptů.

5.5 Oracle SQL Developer Data Modeler 2.0

Oracle SQL Developer Data Modeler je nástroj pro datové modelování, který usnadňuje a zlepšuje komunikaci mezi datovými architekty, databázovými administrátory, vývojáři a uživateli, což zjednodušuje modelování a samotný vývojový proces. Jeho použitím mohou uživatelé vytvářet, prohlížet a upravovat, logické, relační, fyzické, multidimenzionální a datové typy modelů. Umožňuje také generování DDL skriptů, což zvyšuje produktivitu.

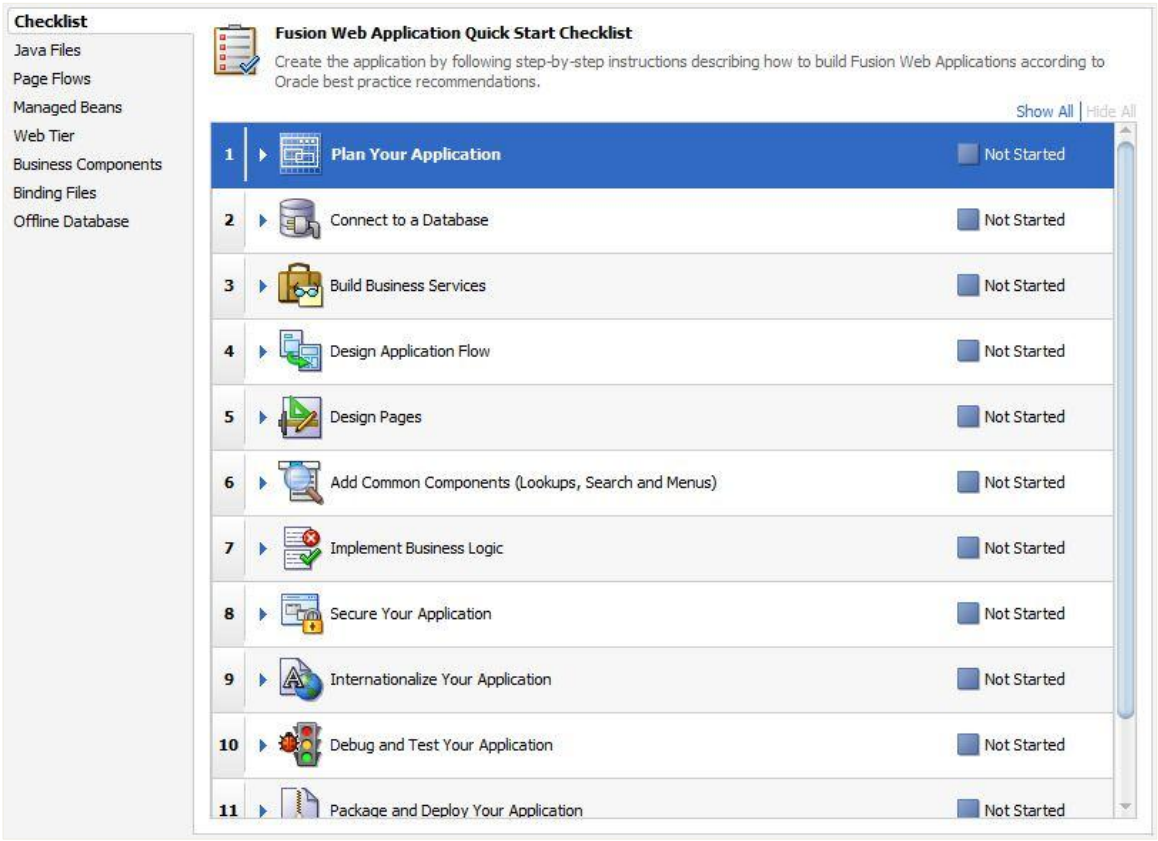
5.6 Adobe Photoshop CS5

Kvalitní a profesionální bitmapový grafický editor, který je světovým standardem pro editaci grafiky v nejvyšší kvalitě. Adobe Photoshop CS5 je ideálním nástrojem pro tvorbu fotografií a obrázků nebo jejich úpravu. Velice snadno si poradí také s tvorbou webové grafiky nebo exportem snímků do HTML.

6 Analýza a návrh IS

Před samotnou implementací je vhodné řádně zanalyzovat zadaný úkol. Shromáždit důležité požadavky kladené na výslednou aplikaci a na základě jejich vyhodnocení pomocí příslušných diagramů navrhnout odpovídající modely.

Analýzu i návrh lze provést v různých nástrojích nezávislých na vývojovém prostředí zvoleném pro samotnou implementaci. Jelikož ale JDeveloper spolu s Oracle ADF Web Fusion frameworkem vytvářejí dobré podmínky pro pokrytí životního cyklu aplikace od modelování až po nasazení, bude v této kapitole naznačeno i jeho využití (Obrázek 6).

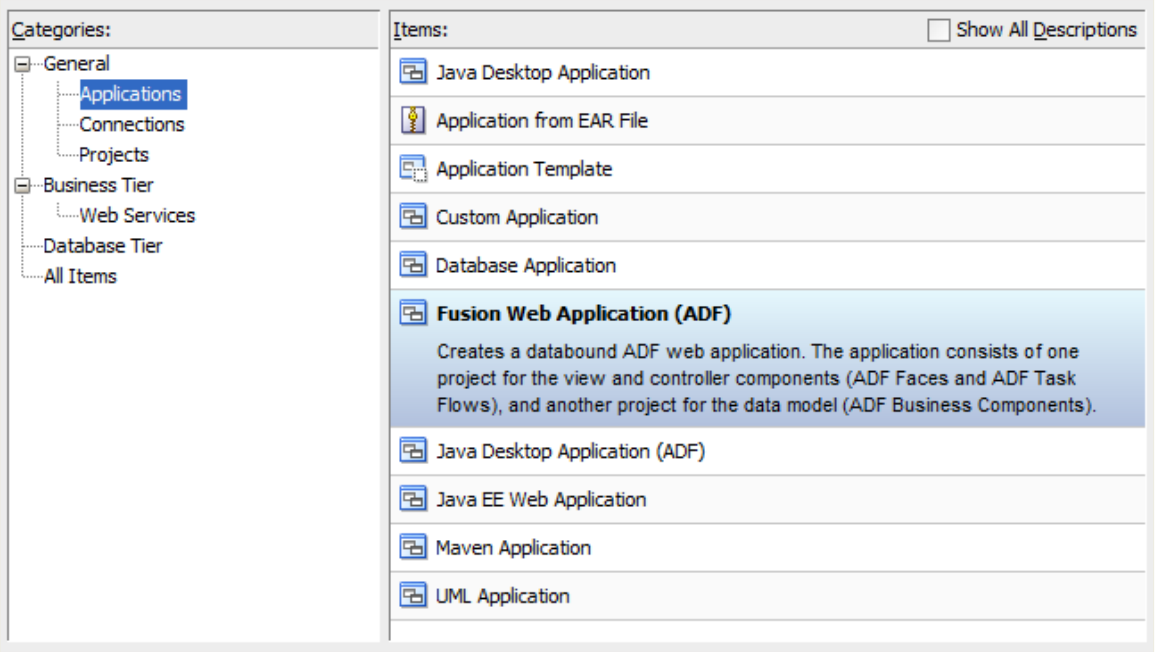


Zdroj: autor

Obrázek 6 - Checklist

6.1 Plánování aplikace

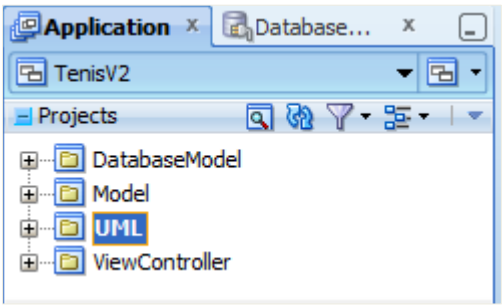
První krok vychází z předpokladu, že Web Fusion aplikace je již založena (Obrázek 7). Vytvořená aplikace obsahuje projekt model pro *business* komponenty a projekt *view/controller* pro stránky a *Task Flows*.



Zdroj: autor

Obrázek 7 - Vytvoření nové ADF aplikace

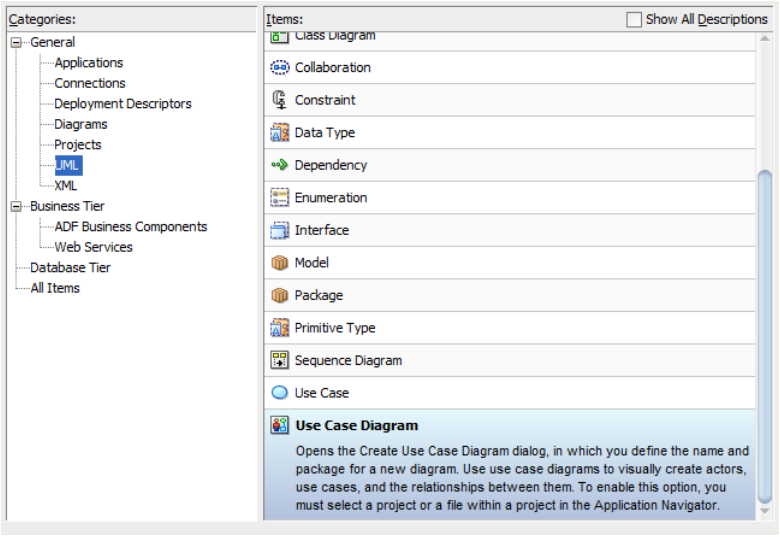
V rámci plánování aplikace lze využít některou z následujících akcí: návrh datového modelu, vytvoření případů užití, navrhnout uživatelská rozhraní, definování *business* pravidel apod. Tyto možnosti jsou dostupné ihned po založení příslušného projektu. Pro potřeby aplikace byl mimo jiné založen projekt UML (Obrázek 8). Kliknutím pravým tlačítkem myši nad názvem aplikace v okně Application – New Project – v záložce General se zvolí Project a v okně Items položka UML Project.



Zdroj: autor

Obrázek 8 – Vytvoření UML projektu

Nyní je možné podobným postupem jako při založení UML projektu vytvořit např. Use Case Diagram nebo Sequence Diagram. Pro potřeby vyvíjené aplikace byl vytvořen diagram analytických tříd, diagram případů užití (Obrázek 9) a datový model.

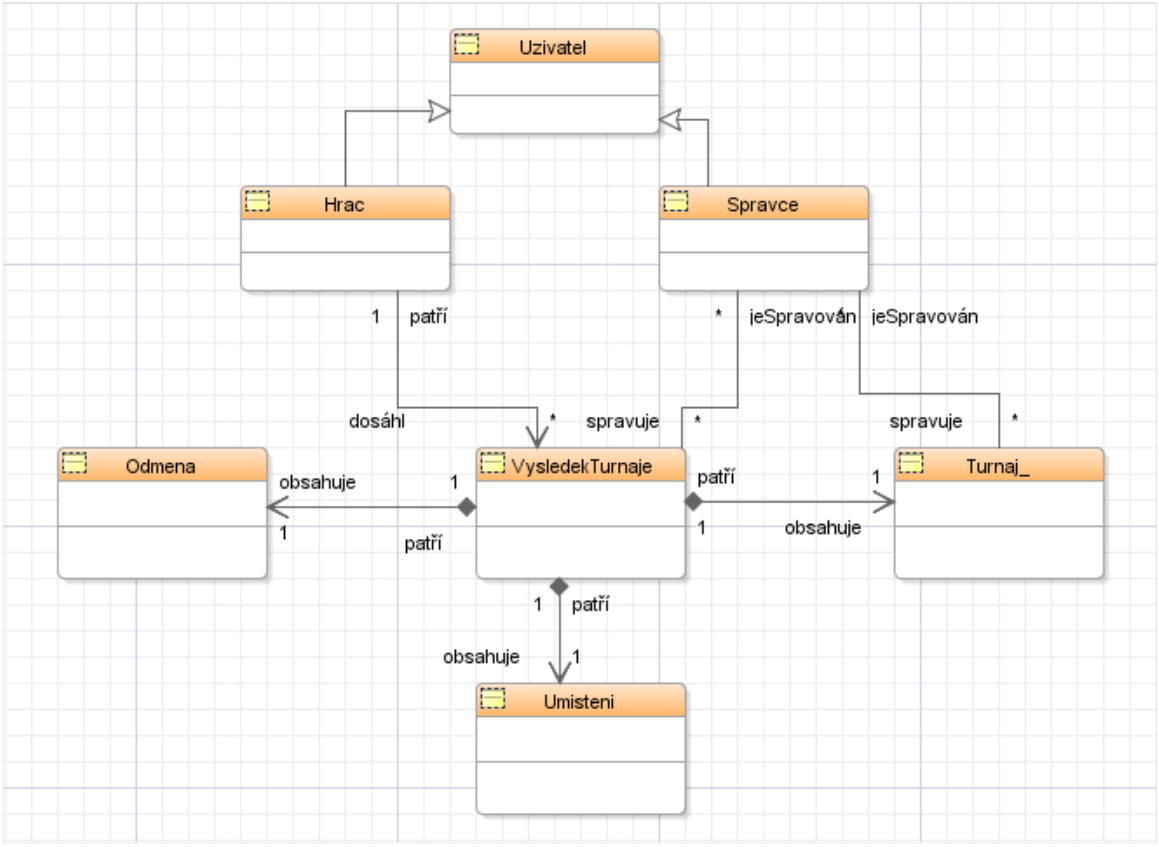


Zdroj: autor

Obrázek 9 - Vytvoření Use Case Diagramu

6.1.1 Diagram analytických tříd

Na základě vyhodnocení jednotlivých požadavků byl vytvořen diagram analytických tříd, znázorněný na Obrázek 10. Základními prvky tohoto diagramu jsou třídy Uzivatel a VysledekTurnaje. Ze třídy Uzivatel vycházejí další dvě třídy Hrac a Spravce. VysledekTurnaje je složen ze tříd Odmena, Turnaj a Umisteni.



Zdroj: autor

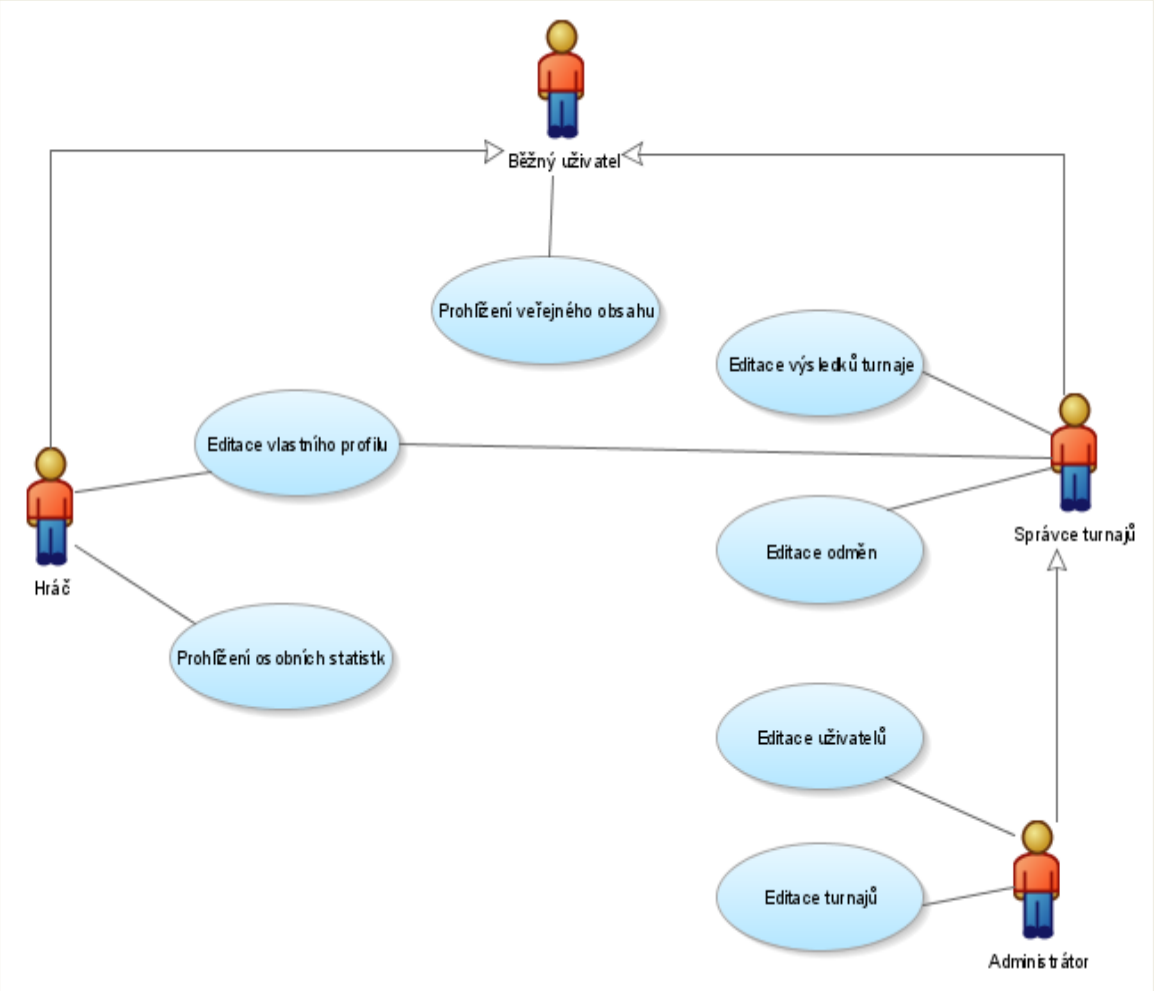
Obrázek 10 - Diagram analytických tříd

6.1.2 Případy užití

Diagram případů užití (Obrázek 11) znázorňuje, jaké uživatelské role v aplikaci figurují. Jedná se celkem o čtyři role:

- běžný uživatel,
- hráč,
- správce turnaje,
- administrátor (správce systému).

Pod rolí „Běžný uživatel“ si lze představit návštěvníka webových stránek, který pouze vyhledává pro něj zajímavé informace, jako jsou například herní statistiky oblíbeného hráče nebo finanční odměny na jednotlivých turnajích. Tento druh uživatele nemá možnost přihlášení do systému a nemůže tak žádným způsobem ovlivňovat jeho obsah.



Zdroj: autor

Obrázek 11 – Use Case Diagram

Další rolí umožňující přihlášení se do systému, je role „Hráč“. Tato role je vytvořena pro tenisového hráče, jenž díky přístupu do systému může spravovat své osobní údaje, které se

pak prezentují na webových stránkách. Dále může sledovat osobní statistiky týkající se jeho herních výsledků a finančních odměn.

„Správce turnajů“ je první role, která větší měrou zasahuje do obsahu systému. Uživatel s touto rolí získá oprávnění k editaci výsledků jednotlivých turnajů. Dále je jeho povinností evidovat finanční odměny pro aktuální sezonu.

Role, která má ze všech nejvíce pravomocí, nese název „Správce systému“. Tato role má možnost pracovat nad všemi databázovými tabulkami. Řeší především vytváření nových uživatelů a turnajů. Dále může například:

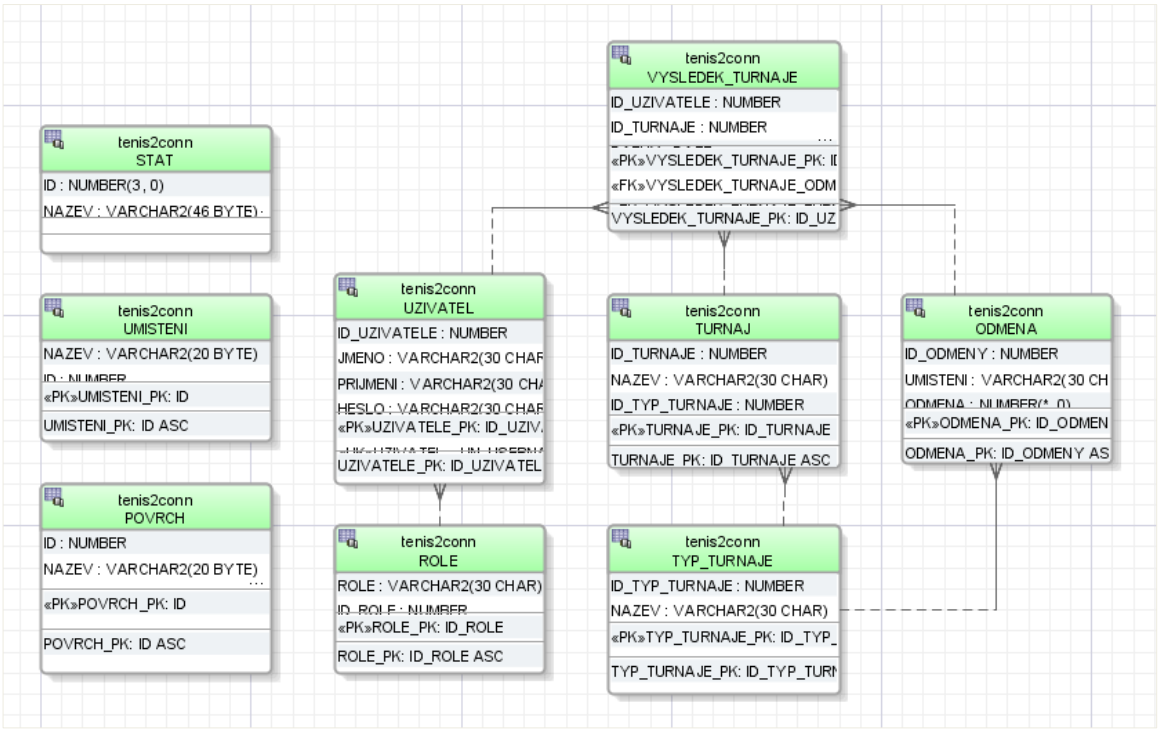
- Přidělovat práva ostatním uživatelům.
- Vytvářet nové typy turnajů.
- Editovat druhy povrchů apod.

6.1.3 Datový model

Nejprve je třeba založit nový projekt typu Database Project a v něm vytvořit Database Diagram. Poté je možné navrhnout tabulky, pohledy, vazby mezi tabulkami apod. Z výsledného diagramu lze vytvořit objekty v databázi (nutnost funkčního připojení k databázi).

Další možností vytvoření diagramu je jeho vygenerování na základě existujících objektů v databázi (nutnost funkčního připojení k databázi). Tuto možnost je vhodné využít, pokud již existuje funkční databáze a je požadována synchronizace vlastností databáze s datovým modelem aplikace. Z diagramu (Obrázek 12) si lze všimnout, že schéma obsahuje celkem devět tabulek.

Každá z tabulek má svůj význam, ale těmi nejpodstatnějšími tabulkami jsou UZIVATEL a VYSLEDEK_TURNAJE. UZIVATEL obsahuje údaje jako jméno, příjmení, uživatelské jméno, heslo apod. Díky vazbě na tabulku ROLE jsou dále uživatelé rozděleni do příslušných podskupin. Tabulka VYSLEDEK_TURNAJE pomocí vazeb na tabulky UZIVATEL, TURNAJ a ODMENA shromažďuje důležitá data, podle kterých se v aplikaci vytváří většina statistik.



Zdroj: autor

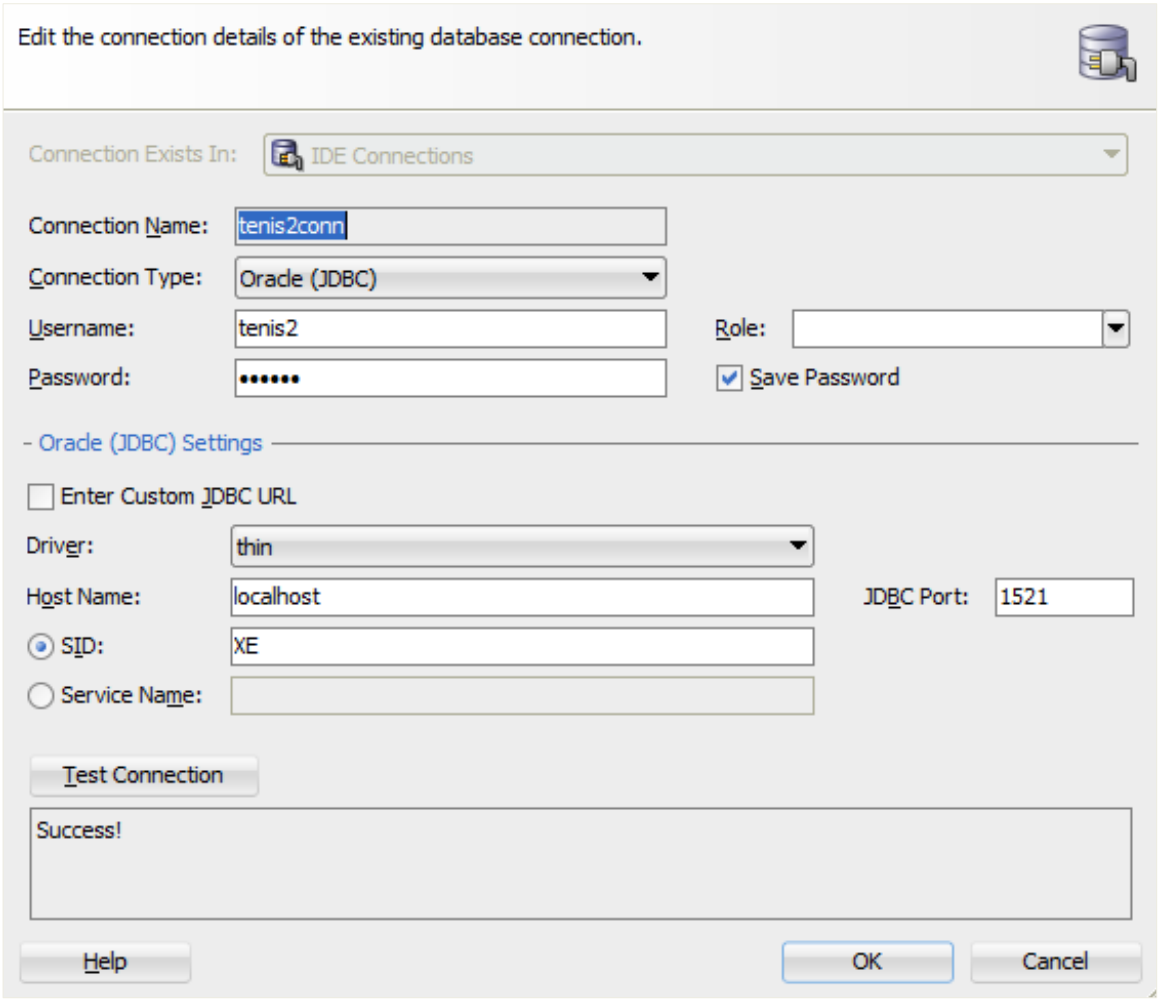
Obrázek 12 – ER diagram

7 Implementace IS pomocí technologie Oracle ADF Web Fusion

Samotná implementace navazuje na provedenou analýzu a návrh. Jak už bylo zmíněno v kapitole zabývající se porovnáním jednotlivých frameworků, Oracle ADF Web Fusion poskytuje tipy a rady prostřednictvím průvodce. Implementace je rozdělena do několika fází, z nichž většina nabízí několik variant k jejich vyřešení (Obrázek 6).

7.1 Připojení k databázi

Pokud je k dispozici databáze obsahující schéma, na kterém lze postavit aplikaci, a jsou známy následující informace: název hostitele, port, SID nebo název služby, uživatelské jméno a heslo, nic nebrání připojení databáze (Obrázek 13). Pro přístup k objektům v databázi, používá Oracle ADF v aplikaci definované databázové připojení. Připojení je možné vytvořit přímo v aplikaci, nebo ho vytvořit jako IDE zdroj a potom jej zkopírovat do aplikace. Pro připojení databáze byla zvolena druhá varianta.



Zdroj: autor

Obrázek 13 - Připojení databáze

7.2 Vytvoření aplikační logiky

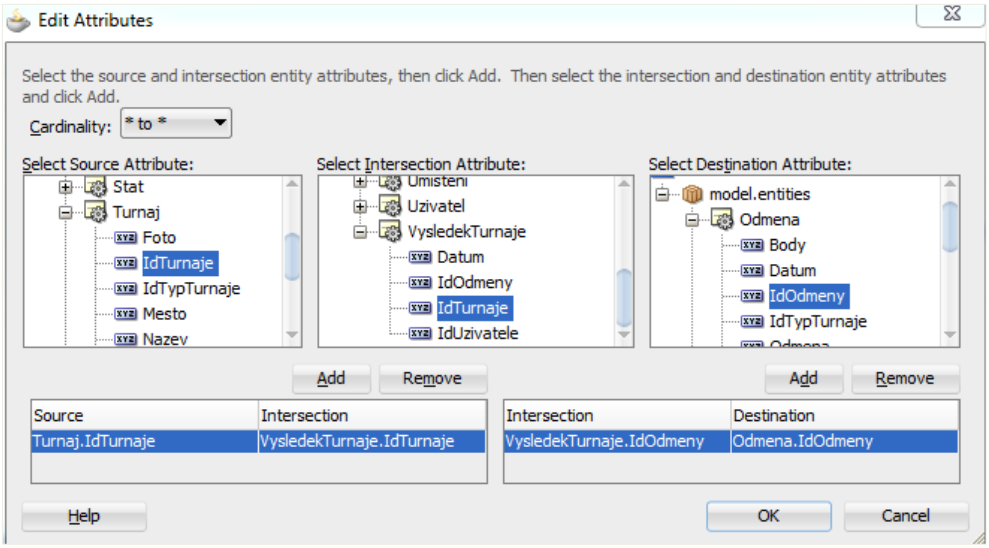
Aplikace již disponuje připojenou databází obsahující tabulky. Aby bylo možné v aplikaci přistupovat k databázi a funkcím aplikační logiky, musí se vytvořit ADF *business* komponenty. Typický projekt „model“ se skládá z několika různých typů *business* komponent:

- entity objects - reprezentují řádky dat v tabulkách a zpracovávají *business* logiku.
- view objects - představují SQL dotazy.
- associations - definují vztahy mezi entity objects.
- view links - definují vztahy mezi view objects.
- application module - definuje datový model.

Zdroj: autor

Obrázek 14 - Možnosti tvorby business komponent

Všechny tyto komponenty lze vytvořit jednoduše pomocí průvodce (Obrázek 14). Mezi jednotlivými *Entity objects* se automaticky vytvoří příslušné vztahy (Associations). V databázi se však mohou nacházet tabulky ve vztahu typu M:N, u kterých se vztahy nevytvářejí automaticky. Automaticky se vytvoří pouze vztahy 1:N, což pro bezchybný chod aplikace nestačí. V případě této konkrétní aplikace bylo nutné zadefinovat vztah M:N mezi entitami Turnaj a Odmena přes entitu VysledekTurnaje (Obrázek 15).

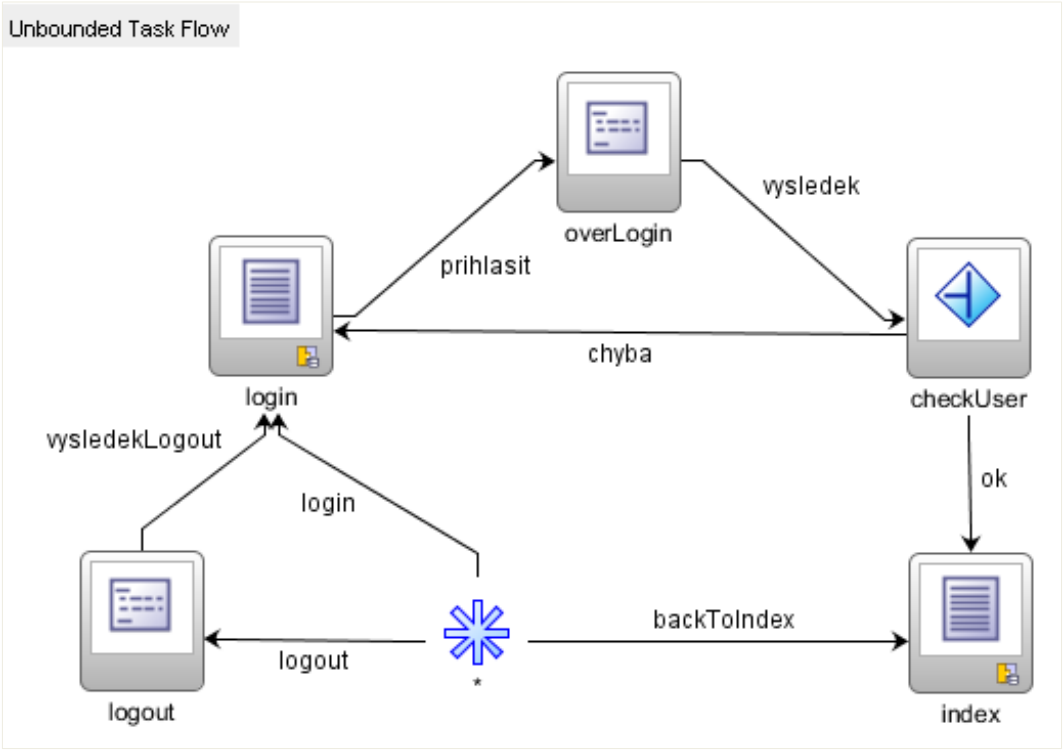


Zdroj: autor

Obrázek 15 - Nastaveni M:N

7.3 Návrh navigace aplikace

Dalším krokem je definování propojení mezi jednotlivými stránkami aplikace. Tato činnost se provádí v modelu ViewController. Aplikace obvykle obsahuje jeden *Unbounded Task Flow*, který definuje hlavní tok aplikace a několik *Bounded Task Flows*, které se vyskytují v příslušných částech aplikace. A *Bounded Task Flows* může být složen ze stránek nebo fragmentů stránek. Jak je zřejmé z Obrázek 16, aplikace využívá jeden *Unbounded Task Flow*, který na základě výsledků příslušných operací řídí navigaci mezi stránkami index.jspx a login.jspx. Samotná stránka index.jspx pak ve své hlavní části obsahuje dynamický region, který umožňuje měnit svůj obsah na základě aktuální činnosti aplikace. Dynamický region zobrazuje příslušné fragmenty stránek, přičemž každý fragment má svůj *Bounded Task Flow*.

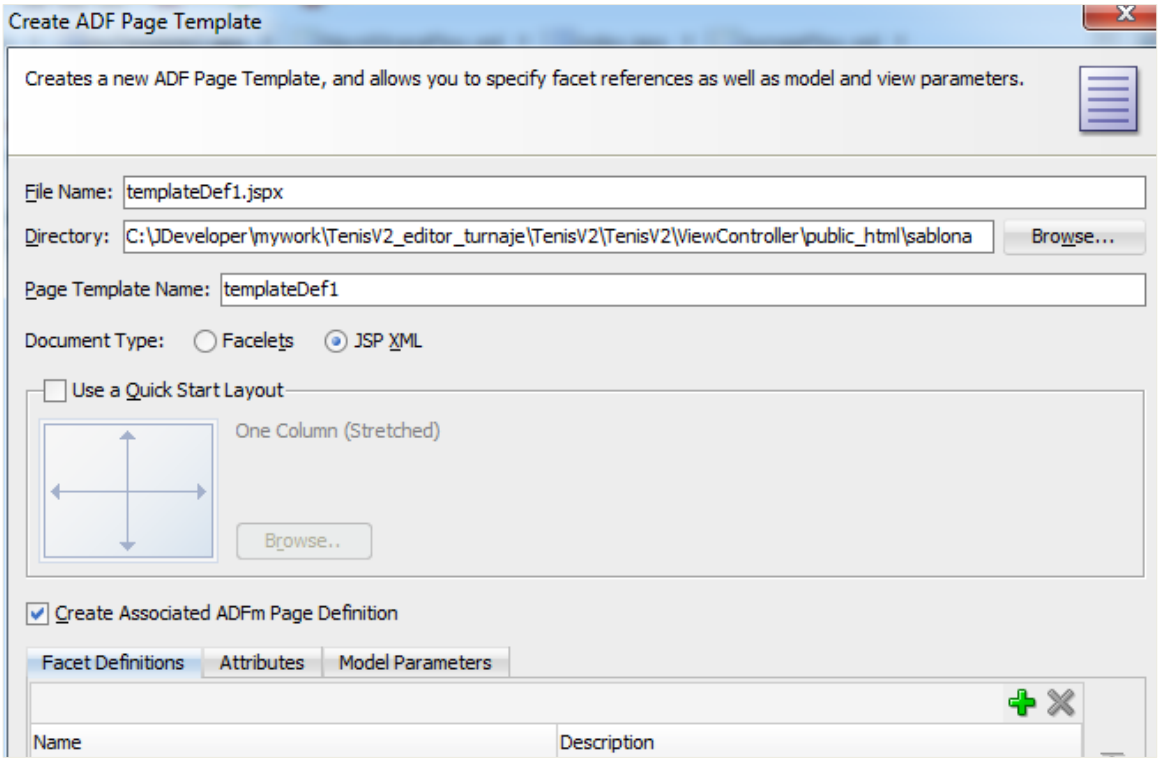


zdroj: autor

Obrázek 16 - Unbounded Task Flow

7.4 Návrh vzhledu stránek

Následující část je zaměřena na návrh stránek z hlediska vzhledu a rozložení komponent. Před vytvořením nové stránky, lze nejprve navrhnout šablonu, která zaručí jednotný vzhled pro celou aplikaci. Vytvoření šablony se provádí v projektu ViewController pomocí průvodce a jeho položky ADF Page Template (Obrázek 17).



zdroj: autor

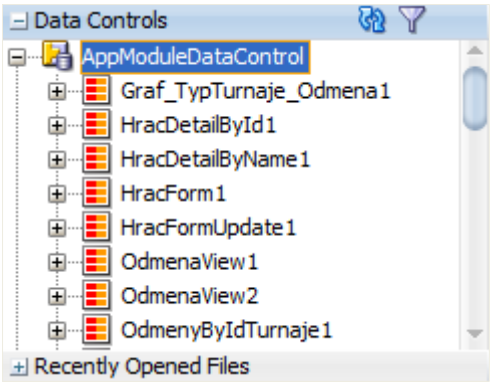
Obrázek 17 - Vytvoření šablony

Průvodce umožňuje vytvořit šablonu na základě předem definovaných layoutů stránek. Pokud není vyžadováno nějaké specifické rozložení a postačují ta běžně používaná, stačí zaškrtnout volbu Use a Quick Start Layout, čímž se z nabídky vybere odpovídající typ layoutu.

V aplikaci představuje šablonu soubor myTemplate2.jspx. Šablona obsahuje specifický vzhled odpovídající povaze aplikace. Dále také klasické rozložení *header*, *menu*, *content*, *footer*. Oblast *content* pak obsahuje *facet main* přidaný pomocí průvodce v části Facet Definition. Tento *facet* umožňuje vložení dynamického regionu, který jak již bylo zmíněno dříve, zobrazuje na základě kontextu různé fragmenty stránek.

7.5 Přidání společných komponent

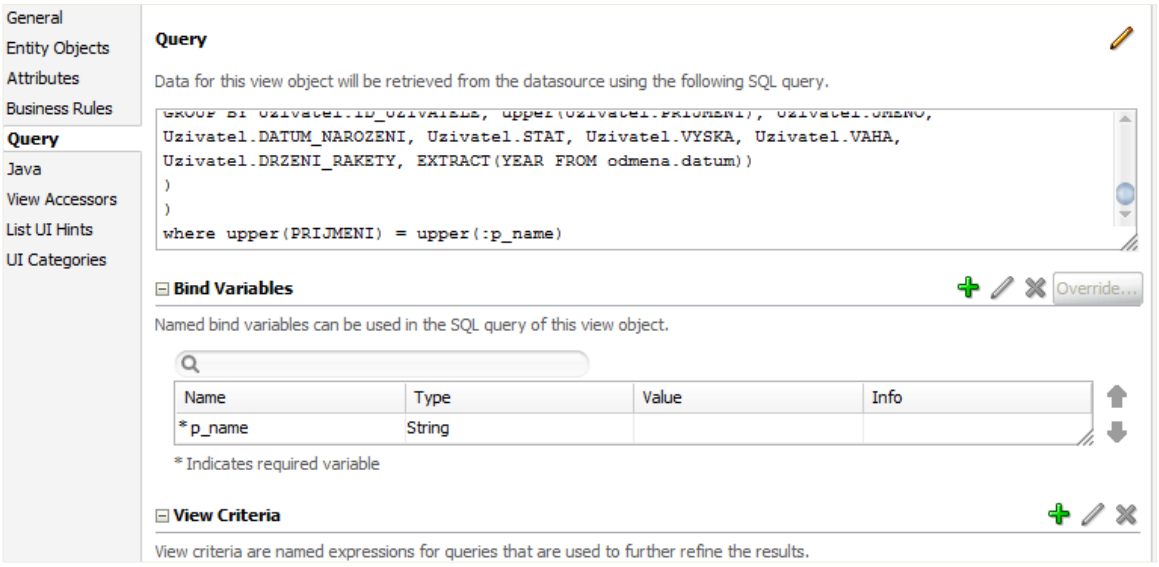
Nyní už jsou vytvořeny *business* komponenty a stránky. Dále je možné obohatit stránky přidáním vyhledávacích formulářů založených na dotazech a vztazích definovaných ve *view objects* nebo menu nabídek založených na hierarchii stránek definovaných v *Task Flows*. V Navigatoru v části Data Controls se nachází *business* logika jednotlivých komponent (Obrázek 18).



zdroj: autor

Obrázek 18 - Data Controls

V aplikaci je s oblibou používán vyhledávací formulář, který při úspěšném vyhledání zobrazí příslušné informace v podobě tabulky, grafu apod. Takový formulář je založený na *view* objektu, jehož datovým zdrojem je SQL dotaz. Tomuto dotazu je předávána formulářem proměnná, zpravidla ID, podle které se pak odlišuje zobrazovaný výsledek.

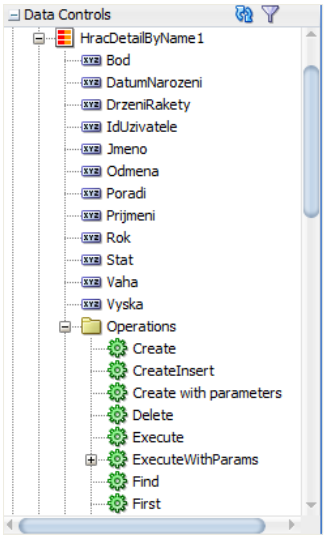


zdroj: autor

Obrázek 19 - View Object

Obrázek 19 uvádí příklad již vytvořeného *view* objektu, který je založen na SQL dotazu. Tomuto dotazu je předávána proměnná p_name, která ve formuláři představuje příjmení hledaného hráče. Při úspěšném nalezení zobrazí informace o hráči.

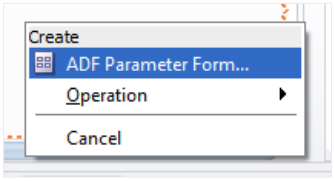
Tento druh formuláře je vytvořen pomocí operace ExecuteWithParams, která se nachází u příslušného DataConrtol v části Operations (Obrázek 20). Poté stačí tuto ikonu stylem *drag and drop* umístit na nějaké místo ve stránce.



zdroj: autor

Obrázek 20 – ExecuteWithParameters

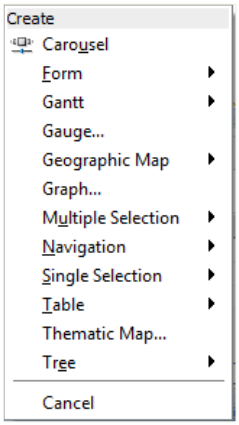
Po vložení zmíněné ikony na stránku se zobrazí dialogové okno, kde se zvolí možnost ADF Parameter Form (Obrázek 21). Po dokončení průvodce vznikne v místě vložení formulář. Samotný formulář však nic nezobrazí.



zdroj: autor

Obrázek 21 - ADF parameter form

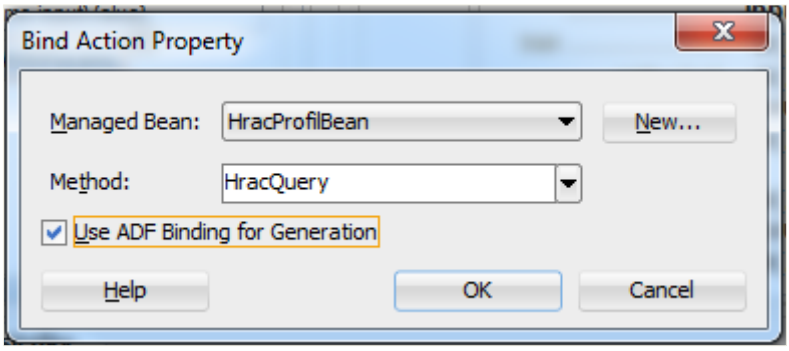
Aby se zobrazila například tabulka, graf nebo formulář je nutné na stránku přetáhnout příslušný *Data Control*, v případě popisované aplikace HracDetailByName1. Po vložení na stránku se objeví dialogové okno s nabídkou několika možností, jak příslušná data zobrazit (Obrázek 22).



zdroj: autor

Obrázek 22 - Data Control HracDetailByName1

Uvedený příklad řeší situace, kde se jedním formulářem vyhledávají informace z příslušného *Data Control*. Kdyby však bylo potřeba pomocí jednoho vyhledávacího formuláře zobrazit informace ze dvou a více různých *Data Controls*, je třeba provést úpravy, které nejsou dostupné v žádném oficiálním tutoriálu. V aplikaci, konkrétně ve fragmentu najdiHrace.jsff, jsou použity tři operace ExecuteWithParams pro zobrazení informací ze třech různých *Data Controls*. Pokud by byl použit postup, který byl uveden výše, obsahovala by stránka tři vyhledávací formuláře, které by uživatel postupně odeslal. Takováto funkčnost není příliš uživatelsky přívětivá. Mnohem pohodlnější by bylo vyplnit a odeslat jeden formulář a zároveň zobrazit najednou údaje ze všech třech *Data Controls*. V tomto případě je nutné, stejným způsobem jako v předešlé ukázce, vložit do stránky tři formuláře spolu s příslušnými *Data Controls*.



zdroj: autor

Obrázek 23 – Managed bean

U prvního formuláře se pomocí dvojkliku nad tlačítkem zobrazí okno Bind Action Property (Obrázek 23), kde je nutné vytvořit *Managed Beanu*, ve které bude definováno, jaké formuláře se zpracují po kliknutí na tlačítko prvního formuláře. Zdrojový kód by měl vypadat následovně.

```
public String HracQuery()
{
      BindingContainer bindings = getBindings();
      OperationBinding operationBinding
      =bindings.getOperationBinding("ExecuteWithParams");
      Object result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
            return null;
      }
      operationBinding =
      bindings.getOperationBinding("ExecuteWithParams1");
      result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
            return null;
      }
      operationBinding =
      bindings.getOperationBinding("ExecuteWithParams2");
      result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
```

```
            return null;
      }
      return null;
}
```

Tato část kódu je vygenerována automaticky.

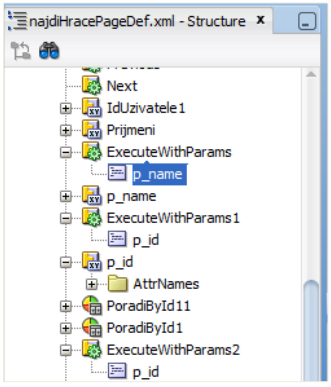
```
      BindingContainer bindings = getBindings();
      OperationBinding operationBinding =
      bindings.getOperationBinding("ExecuteWithParams");
      Object result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
            return null;
      }
```

Následující řádky kódu je nutné dopsat ručně. Výsledkem bude, že po kliknutí na tlačítko prvního formuláře, proběhne zpracování všech tří formulářů.

```
      operationBinding =
      bindings.getOperationBinding("ExecuteWithParams1");
      result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
            return null;
      }

      operationBinding =
      bindings.getOperationBinding("ExecuteWithParams2");
      result = operationBinding.execute();
      if (!operationBinding.getErrors().isEmpty())
      {
            return null;
      }
```

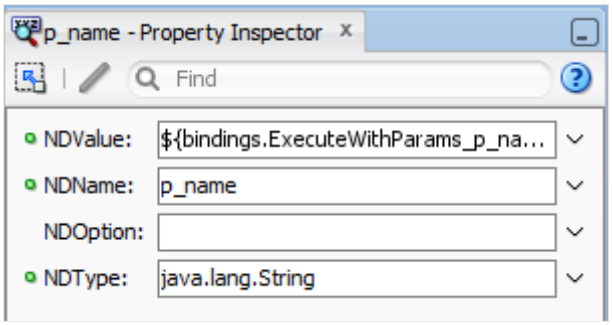
Nyní jsou formuláře provázány z hlediska jejich zpracování. Každý z formulářů však může přijímat jiný parametr, např. ID nebo jméno. V tomto konkrétním příkladu očekává první formulář příjmení hráče, zatímco zbylé dva formuláře očekávají ID hráče. Po přepnutí stránky z režimu Design do režimu Bindings se v navigatoru zobrazí okno Structure, které umožňuje přistoupit k proměnným jednotlivých formulářů a definovat, odkud se bude načítat hodnota proměnné.



zdroj: autor

Obrázek 24 – Proměnné formuláře

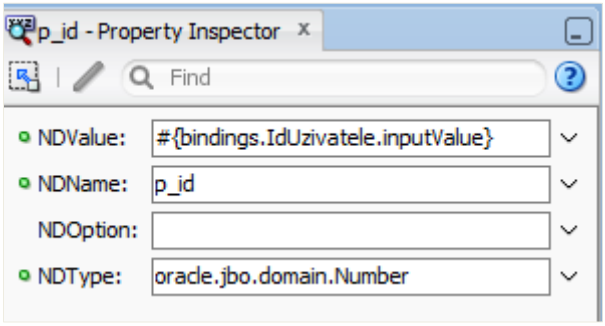
Vybráním proměnné v okně Structure (Obrázek 24) se v okně Property Inspector (Obrázek 25) zobrazí příslušné vlastnosti. Důležitou vlastností je NDValue, která určuje, odkud si bere formulář hodnotu proměnné před samotným zpracováním. U prvního formuláře se zadává hodnota ručně, takže není nutné nic měnit.



zdroj: autor

Obrázek 25 – Defaultní nastavení proměnné

U zbylých dvou je třeba přepsat původní defaultní kód (Obrázek 26). Nový kód představuje zdroj, odkud se bude hodnota proměnné načítat. V tomto případě se hodnota načítá z *read only* formuláře, který je výsledkem zpracování prvního formuláře.



zdroj: autor

Obrázek 26 - Změněné nastavení proměnné

V této fázi se na stránce nacházejí celkem tři formuláře, přičemž všechny reagují pouze na zpracování prvního formuláře. To znamená., že ty formuláře, které není třeba kvůli zpracování spouštět, nemusí být zobrazeny. Aby se zachovaly vazby potřebné pro funkčnost formulářů, není možné je jen tak smazat. Aby nebyly formuláře na stránce viditelné, ale zároveň si zachovaly všechny funkční vazby, je nutné se přepnout z režimu Design do režimu Source. Vyhledat kód patřící formuláři a ten umístit mezi komentáře.

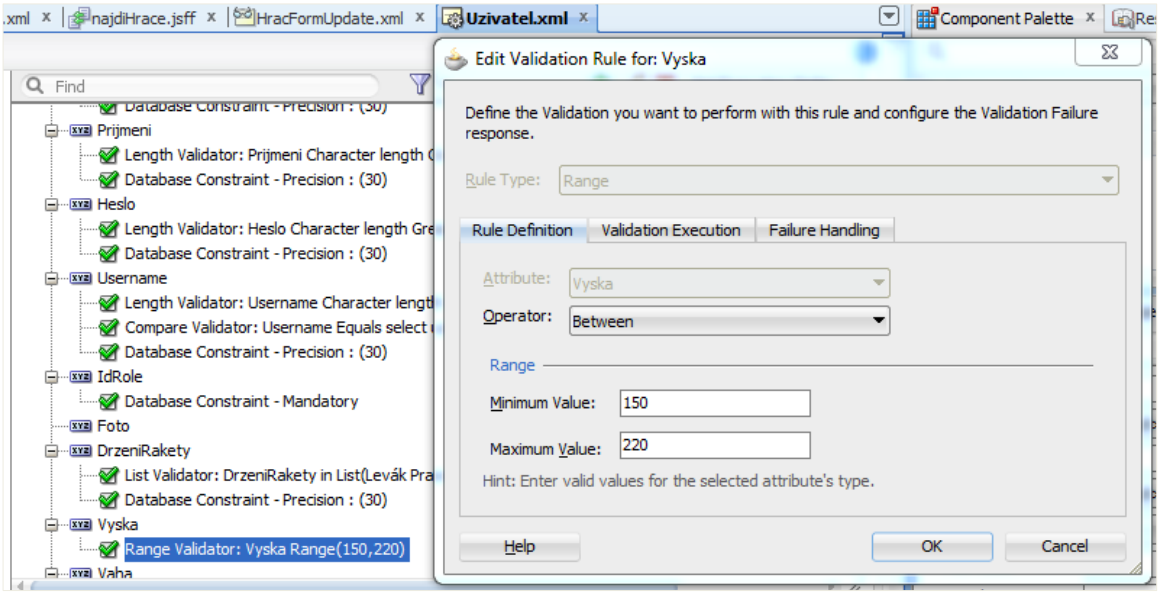
7.6 Implementace aplikační logiky

Další krokem, je za předpokladu existence *business* komponent, vytvoření validačních pravidel. ADF validační pravidla se obvykle aplikují na *entity objects*. Editací těchto

pravidel se mimo jiné docílí ověření zpracovávaného formuláře. Například v momentě, kdy je v poli formuláře očekávané číslo, není přípustné, aby do něj uživatel zadal textový řetězec. Pokud se tak stane, aplikace zobrazí uživateli hlášku upozorňující na chybu a popř. nabídne vzor pro správné vyplnění.

Existuje celá řada vestavěných ověřovacích pravidel, která lze deklarativně vytvořit pomocí dialogů a editorů. Taková pravidla poskytují vlastní chybové zprávy a určují, kdy se pravidlo aplikuje. Jelikož jsou *business* komponenty vytvářeny z databázových tabulek, jsou automaticky jejich součástí i některá omezení. Například maximální délka či velikost datového typu nebo také NOT NULL omezení apod.

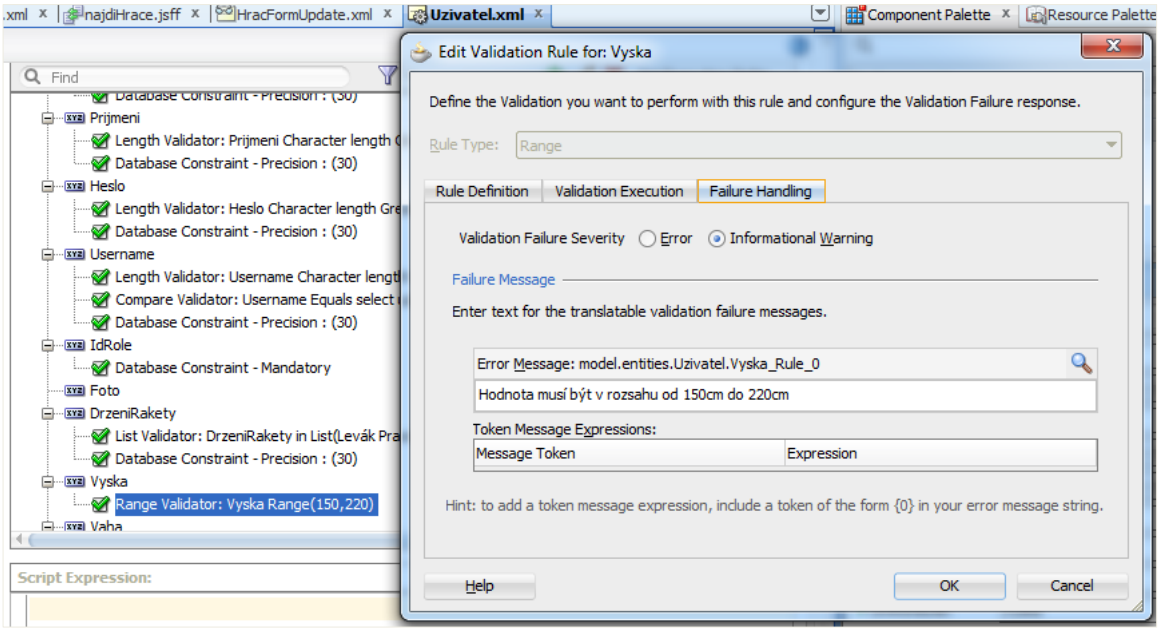
Tato základní pravidla a omezení lze doplnit o další, specifičtější pravidla. V aplikaci je mimo jiné využito validačního pravidla Range. Tento typ pravidla vymezuje hranice přípustných hodnot. Jak znázorňuje Obrázek 27, *entity object* Uzivatel nastavuje u atributu Vyska rozsah přípustných hodnot (150,220).



zdroj: autor

Obrázek 27 – Nastavení validačního pravidla

Pokud uživatel zadá hodnoty mimo stanovený rozsah, zobrazí se mu varování, které ho upozorní na chybu a zároveň předloží vzor, v jakém rozsahu by měla být hodnota zadána. Text hlášky se definuje, jak napovídá Obrázek 28, v záložce Failure Handling.



zdroj: autor

Obrázek 28 – Nastavení chybové hlášky

7.7 Zabezpečení aplikace

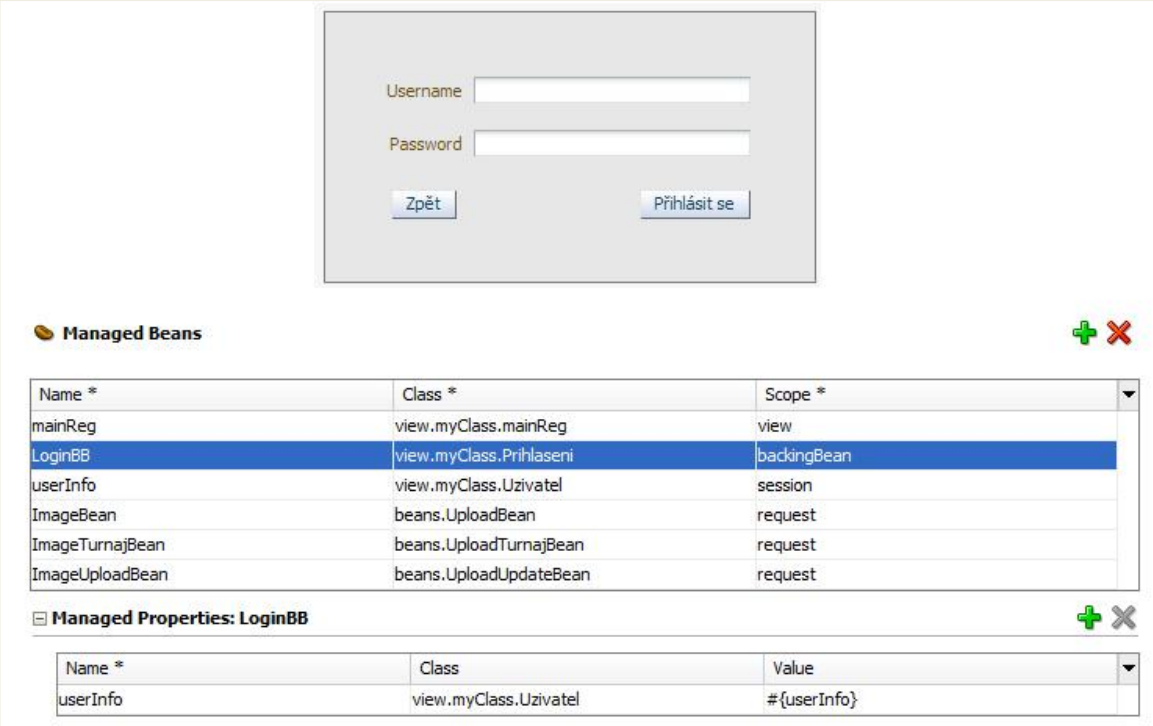
K zabezpečení aplikace poskytuje Oracle ADF funkcionalitu ADF Security. Přes Application – Secure – Configure ADF Security lze spustit průvodce, který nabídne postup k vytvoření zabezpečení založeném na omezení přístupu na základě uživatelských rolí. Jak napovídá Obrázek 29, vytvořeným uživatelům se přiřadí odpovídající role a privilegia pro dané části aplikace (*Task Flows*).

zdroj: autor

Obrázek 29 – Vytvoření uživatelských rolí

ADF Security je sice užitečná a zajímavá varianta jak zabezpečit aplikaci, ale není jediná. Pro potřeby přihlašování do tenisového informačního systému byla využita varianta, kdy je

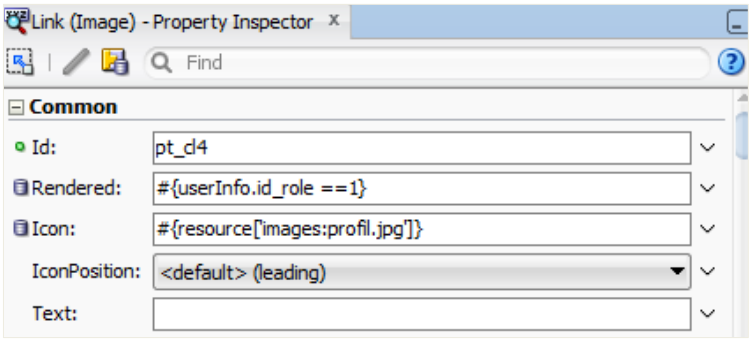
uživatel ověřován na základě hodnot v databázových tabulkách. Ověřuje se přihlašovací jméno a heslo. Podle uživatelské role se pak uživateli zobrazuje příslušný obsah stránek. Tato varianta vyžaduje odpovídající *Task Flow* (Obrázek 16), přihlašovací stránku a *Managed Beans* (Obrázek 30).



zdroj: autor

Obrázek 30 – Přihlašovací formulář

Po úspěšném přihlášení do systému, je uživateli v menu zobrazena nová položka odpovídající jeho privilegiím. Obrázek 31 představuje způsob, jakým lze uživatelům s různými rolemi zobrazovat odlišný obsah. V tomto konkrétním případě se jedná o položku menu, která se zobrazí pouze uživateli s id role rovno 1.



zdroj: autor

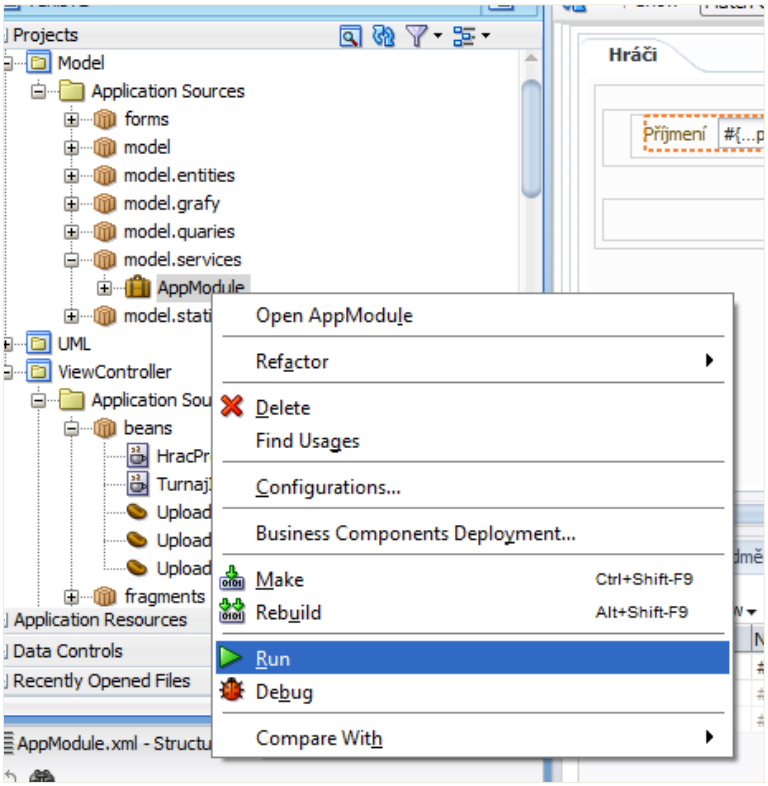
Obrázek 31 – Využití vlastnosti Rendered

7.8 Internacionalizace aplikace

Dalším krokem je internacionalizace aplikace. Tenisový informační systém sice tuto funkcionalitu nevyužívá, ale pokud by bylo potřeba do budoucna aplikaci rozšířit tímto směrem, Oracle ADF poskytuje potřebné prostředky. Pro zobrazení textů podle příslušného jazyka, je nejprve nutné nadefinovat zdroje těchto textů tzv. *Resource Bundle*. Pro každou lokalizaci jeden. Za běhu aplikace se pak vyzvedne správná sada na základě aktuální lokalizace uživatele.

7.9 Krokování a testování aplikace

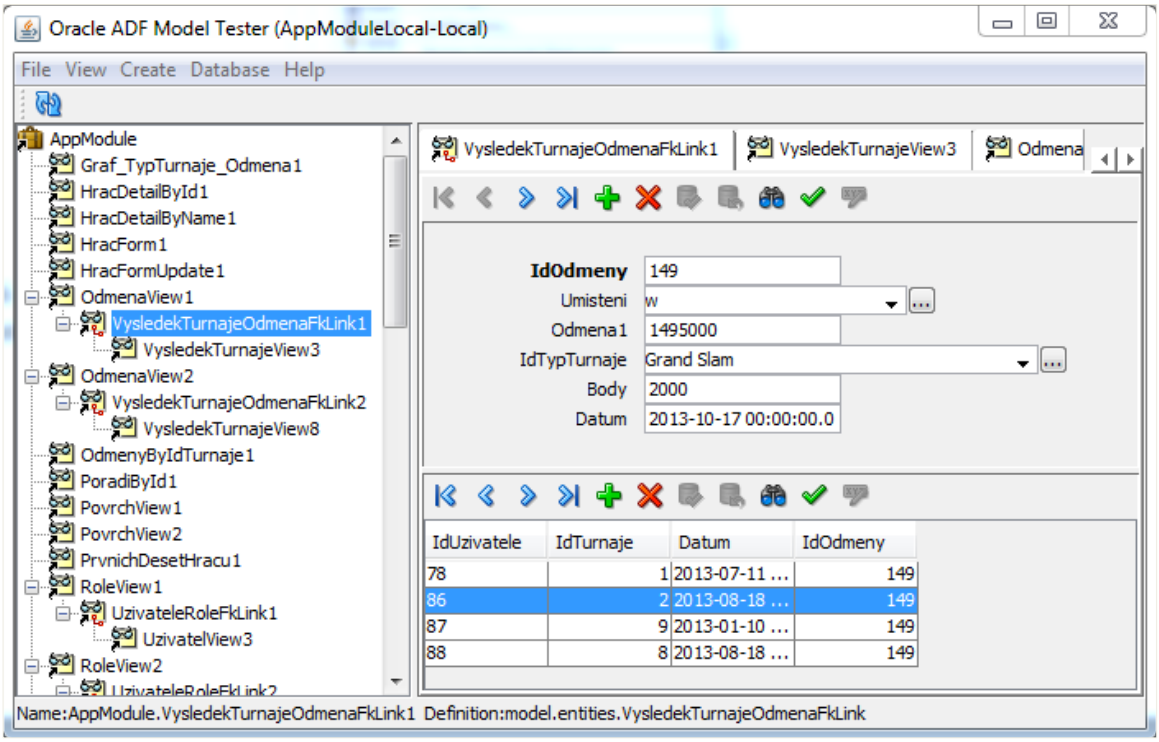
Ačkoliv je testování aplikace uvedené jako desátý krok v pořadí, lze aplikaci testovat už v průběhu tvorby. Například pokud jsou již vytvořeny *business* komponenty, ale zatím nejsou použity na žádné stránce a nelze tak ověřit jejich funkčnost pomocí spuštění ve webovém prohlížeči, nabízí Oracle ADF variantu v podobě aplikačního modulu. Ten se vytvořil spolu s *business* komponentami. Po jeho spuštění (Obrázek 32) dojde k zobrazení okna Oracle ADF Model Tester.



zdroj: autor

Obrázek 32 – Spuštění aplikačního modulu

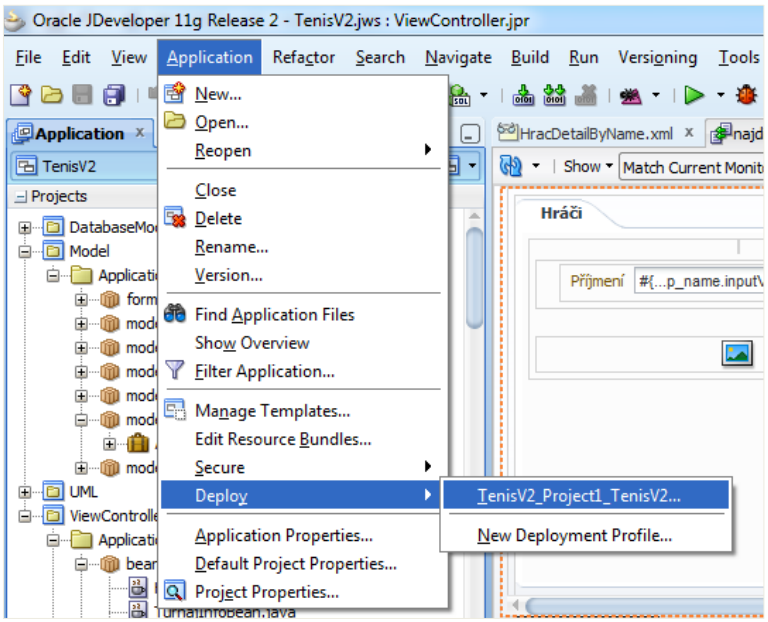
Pomocí Model Testeru je možné ověřit funkčnost jednotlivých komponent (Obrázek 33). Dovoluje vkládat nová data, editovat stávající, vyhledávat pomocí parametrů, mazat apod. Dále lze prostřednictvím Model Testeru zkontrolovat, zda se správně vytvořily vazby mezi příslušnými komponentami.



zdroj: autor

Obrázek 33 – Model tester

Pokud už aplikace obsahuje stránky, je možné v okně Navigator nad některou z nich kliknout pravým tlačítkem myši a v kontextovém menu zvolit položku Run. Příslušná stránka se následně spustí ve webovém prohlížeči. Samozřejmě za předpokladu, že se úspěšně zkompiluje. Prostřednictvím prohlížeče lze klasickým způsobem zkontrolovat, zda se aplikace chová podle očekávání.

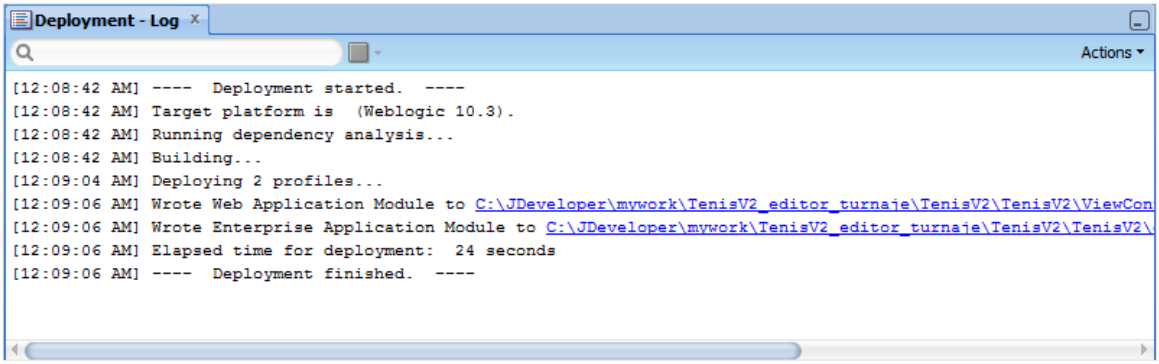


zdroj: autor

Obrázek 34 – Deployment aplikace

7.10 Zabalení a nasazení aplikace

Další kapitolou, která sahá nad rámec popisovaného příkladu, je nasazení aplikace do produkčního prostředí. Aplikaci lze zabalit jako EAR archiv a přenést na aplikační server. Obrázek 34 znázorňuje, jak spustit příslušný dialog, jehož úkolem je provést postupem vytvoření EAR archivu. Pokud je akce úspěšná zobrazí se v okně Deployment kromě logu samotného průběhu, také umístění vytvořeného EAR archivu (Obrázek 35).



zdroj: autor

Obrázek 35 – Deployment log

7.11 Využití aplikace v rámci SOA (service-oriented architecture)

Závěrečný krok popisuje způsoby, jak lze aplikaci využít v rámci SOA. Prvním krokem je vytvoření *business* událostí. *Business* události jsou vyvolávány příslušnými *entity objects* a informují externí *business* procesy. Druhým krokem je vytvoření aplikačního modulu jako externí služby, čímž se vytvoří servisní rozhraní, které umožňuje klientům pracovat s aplikačním modulem pomocí webových služeb.

8 Interakce s aplikací

Úkolem praktické části bylo vytvořit tenisový informační systém. Jedná se o webovou aplikaci, která slouží pro sběr a zpracování dat a k následné prezentaci informací s ohledem na užitečnost pro koncového uživatele.

Data představují údaje o tenisových hráčích, výsledcích, turnajích, finančních odměnách apod. Koncovému uživateli, kterým je zpravidla tenisový fanoušek, pak informační systém prezentuje informace o tom, jakého hráč dosáhl na turnaji výsledku, kolik si připsal bodů a finančních odměn. Informace jsou prezentovány především pomocí přehledných grafů a tabulek. K plnění dat je třeba dalších uživatelů, kteří mají možnost přihlásit se do systému. Podle příslušných práv, specifikovaných pro každou uživatelskou roli, pak se systémem pracují.

8.1 Nepřihlášený uživatel

Aplikace bude nejprve představena z pohledu uživatele, který je pouhým návštěvníkem a nemá tedy možnost přihlásit se do systému. Na Obrázek 36 je zachycena úvodní stránka, jejímž účelem je seznámit uživatele s obsahem aplikace a také v jaké formě mu bude obsah prezentován. Stránka je navržena s ohledem na snadnou orientaci. V menu nepřihlášeného uživatele se nacházejí následující čtyři položky:

- Hlavní - úvodní strana.
- Hráči - sekce věnovaná profilům hráčů.
- Turnaje - sekce věnovaná profilům turnajů.
- Statistiky - obsahuje statistické údaje v podobě tabulek a grafů.

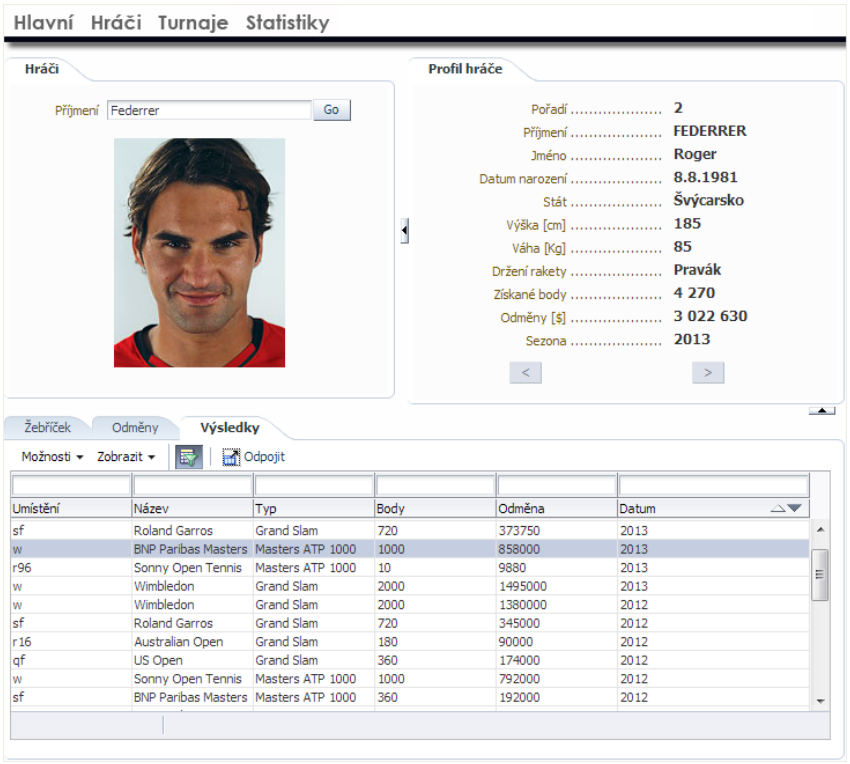
V sekci *Hráči* lze pomocí formuláře vyhledat příslušného hráče a zobrazit tak jeho profilové údaje a další zajímavé informace (Obrázek 37). Kromě fotografie a jeho základních údajů se ve spodní části stránky nacházejí tři záložky:

- žebříček,
- odměny,
- výsledky.

V záložce *Žebříček* je k dispozici graf znázorňující umístění v tenisovém žebříčku v uplynulých sezonách. Další zajímavý graf, který znázorňuje získané odměny za umístění v jednotlivých sezonách, nalezneme pod záložkou *Odměny*. V poslední záložce se, jak je patrné z obrázku, nachází tabulka představující informace o tom, kdy a jakých turnajů se hráč zúčastnil. Zároveň jakého umístění dosáhl a kolik bodů a peněz za něj obdržel. Tabulku je možné filtrovat a řadit dle potřeb. V případě dalšího zpracování údajů, umožňuje aplikace tabulku vyexportovat do dokumentu .xlsx (Microsoft Excel).

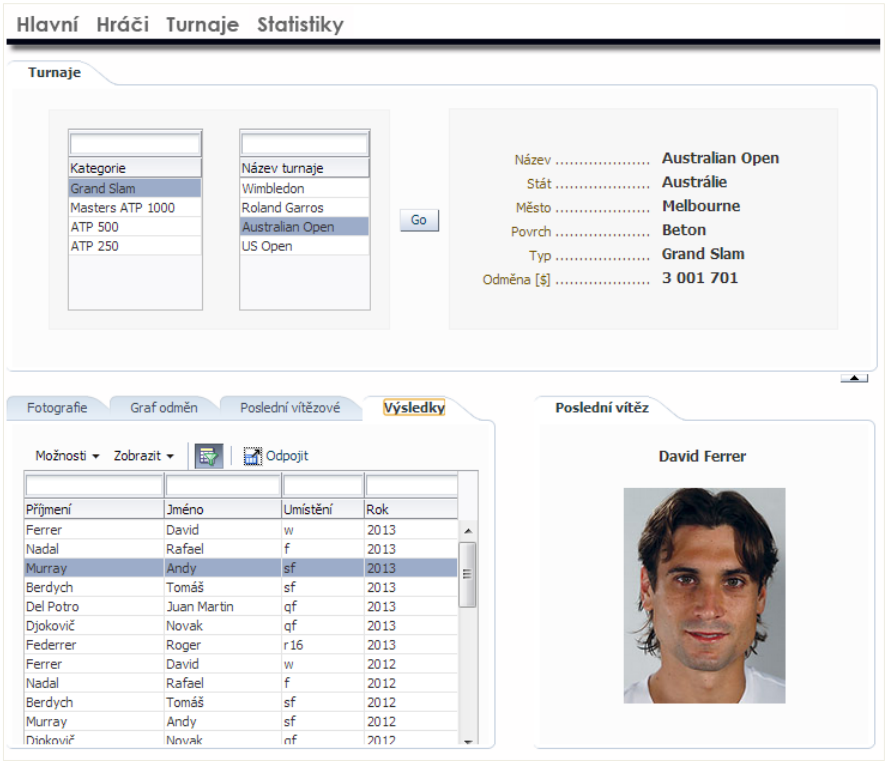
zdroj: autor

Obrázek 36 – Hlavní strana



zdroj: autor

Obrázek 37 – Strana hráči



zdroj: autor

Obrázek 38 – Strana turnaje

V části věnované turnajům si lze podle příslušné kategorie turnaje vybrat konkrétní turnaj (Obrázek 38). Po kliknutí na tlačítko formuláře nám aplikace zobrazí všechny dostupné informace:

- základní údaje,
- fotografie,
- graf odměn,
- poslední vítězové,
- výsledky.

V záložce *Fotografie* nalezne uživatel ilustrační fotografii příslušného turnaje. Záložka *Graf* odměn obsahuje graf prezentující vývoj odměn v jednotlivých sezonách. V následující záložce si může uživatel prohlédnout fotogalerii obsahující vítěze turnaje v odehraných sezonách. V poslední záložce, jejíž náhled je znázorněn na obrázku, je k dispozici tabulka obsahující výsledky hráčů na zvoleném turnaji. Uživatel získá povědomí o tom, jak se jednotliví hráči umístili v uplynulých sezonách.

Hlavním úkolem systému je informovat uživatele o výsledcích a tenisových statistikách. Proto byl na sekci *Statistiky* kladen velký důraz. Bylo vytvořeno velké množství tabulek a grafů, které uživatele informují o podstatných údajích týkajících se hráčů, turnajů a finančních odměn. Tabulka 4 znázorňuje počet výskytů zmíněných statistik.

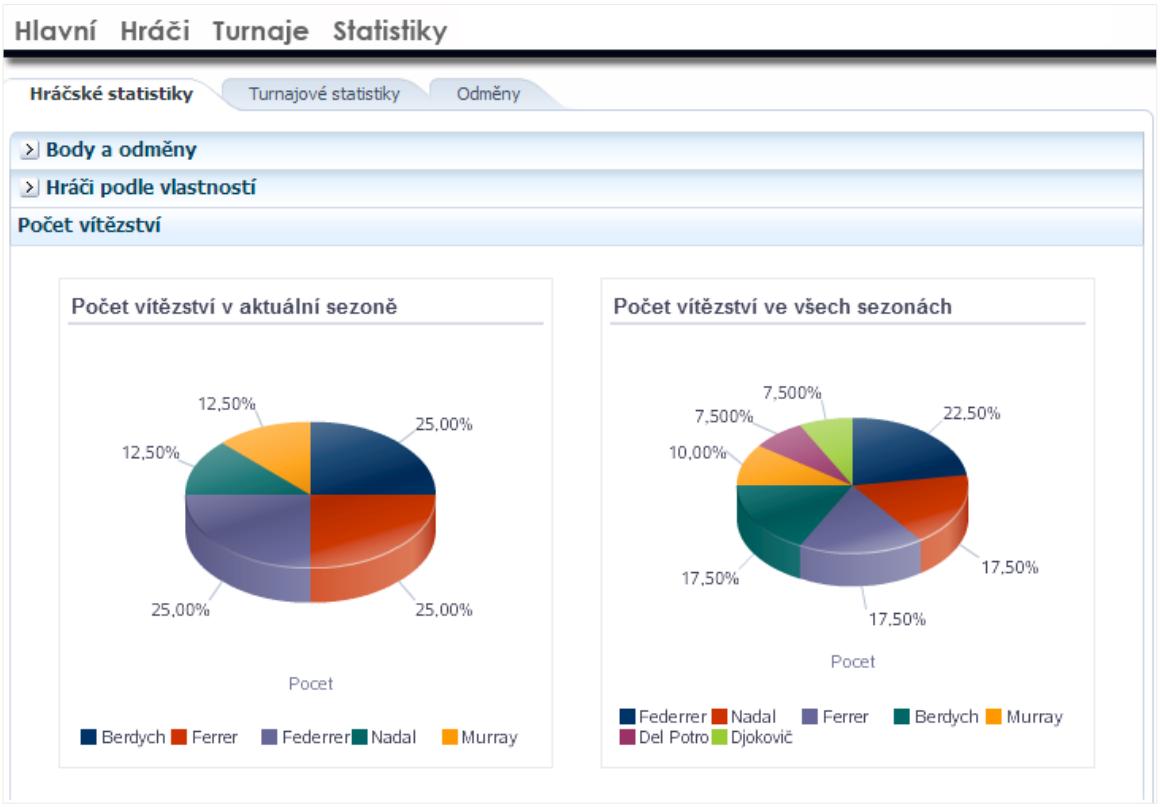
Tabulka 4 - Počet statistik

	Hráči	**Turnaje**	**Odměny**	**Celkem**
Tabulky	6	2	-	8
Grafy	11	12	4	27

zdroj: autor

V další ukázce (Obrázek 39) se pozornost obrací na sekci *Statistiky*. Sekci rozdělují tři záložky, ve kterých se nacházejí statistiky následujícího zaměření:

- hráčské statistiky,
- turnajové statistky,
- odměny.



zdroj: autor

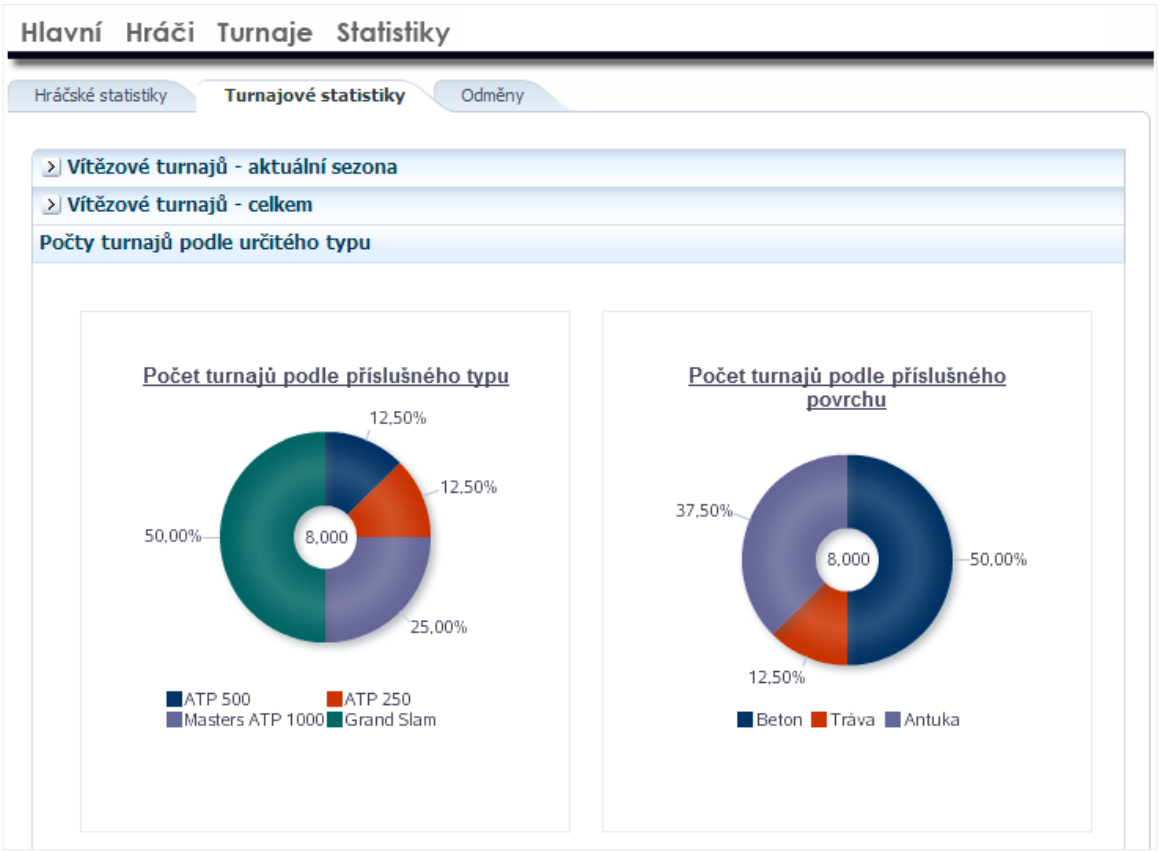
Obrázek 39 – Hráčské statistiky

V záložce *Hráčské statistiky* je uživateli k dispozici panel obsahující tři části:

- *Body a odměny* - obsahuje tabulky s informacemi o umístění hráčů v žebříčku a to jak bodovém tak finančním.
- *Hráči podle vlastností* - pomocí sloupcových grafů informuje o počtu hráčů ve věkových, výškových a váhových skupinách apod.
- *Počet vítězství* - graficky znázorňuje hráče podle počtu vyhraných turnajů.

Obrázek 40 prezentuje obsah záložky *Turnajové statistiky*. Ta obsahuje panel se třemi položkami:

- vítězové turnajů - aktuální sezona,
- vítězové turnajů - celkem,
- počty turnajů podle určitého typu.



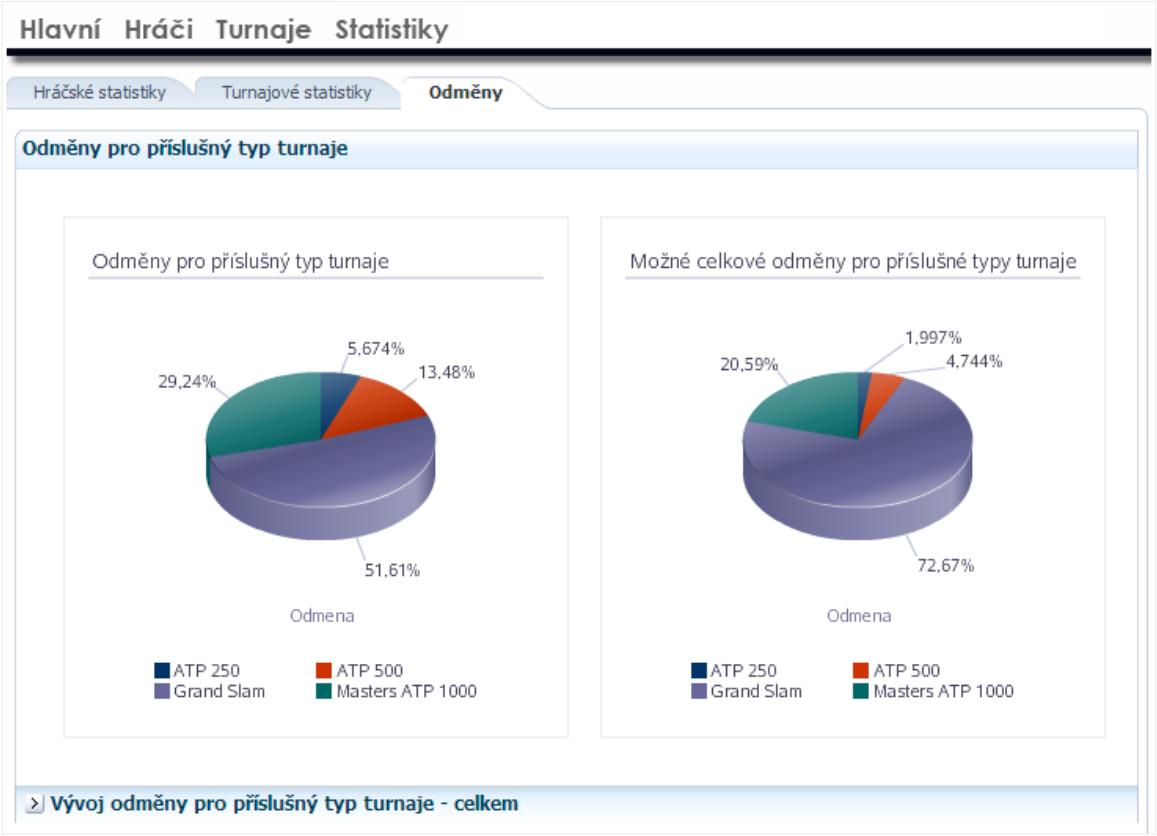
zdroj: autor

Obrázek 40 - Turnajové statistiky

Pod první položkou nalezne uživatel celkem čtyři koláčové grafy znázorňující vítěze určitého typu turnaje v aktuální sezoně. Podobné informace nalezne i v položce druhé s tím rozdílem, že grafy zahrnují vítěze ze všech uplynulých sezon. Obsah poslední položky, jehož náhled je k dispozici na obrázku, graficky znázorňuje počet turnajů na základě typu turnaje a povrchu, na kterém se odehrává.

Poslední záložka Odměny (Obrázek 41) obsahuje panel se dvěma položkami:

- *Odměny pro příslušný typ turnaje* - jak napovídá obrázek, grafy znázorňují, jak velké odměny jsou k dispozici u příslušného typu turnaje a poměr těchto odměn v závislosti na počtu turnajů daného typu.
- *Vývoj odměny pro příslušný typ turnaje - celkem* - rozšiřuje původní grafy o časovou osu a znázorňuje tak vývoj odměn za uplynulé roky.

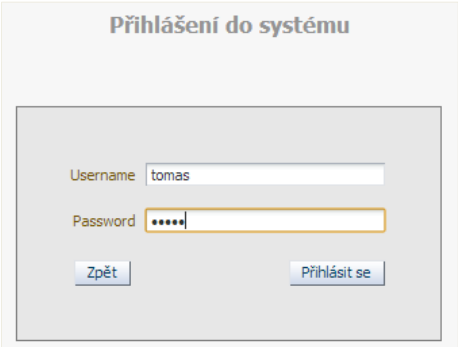


zdroj: autor

Obrázek 41 - Odměny

8.2 Hráč

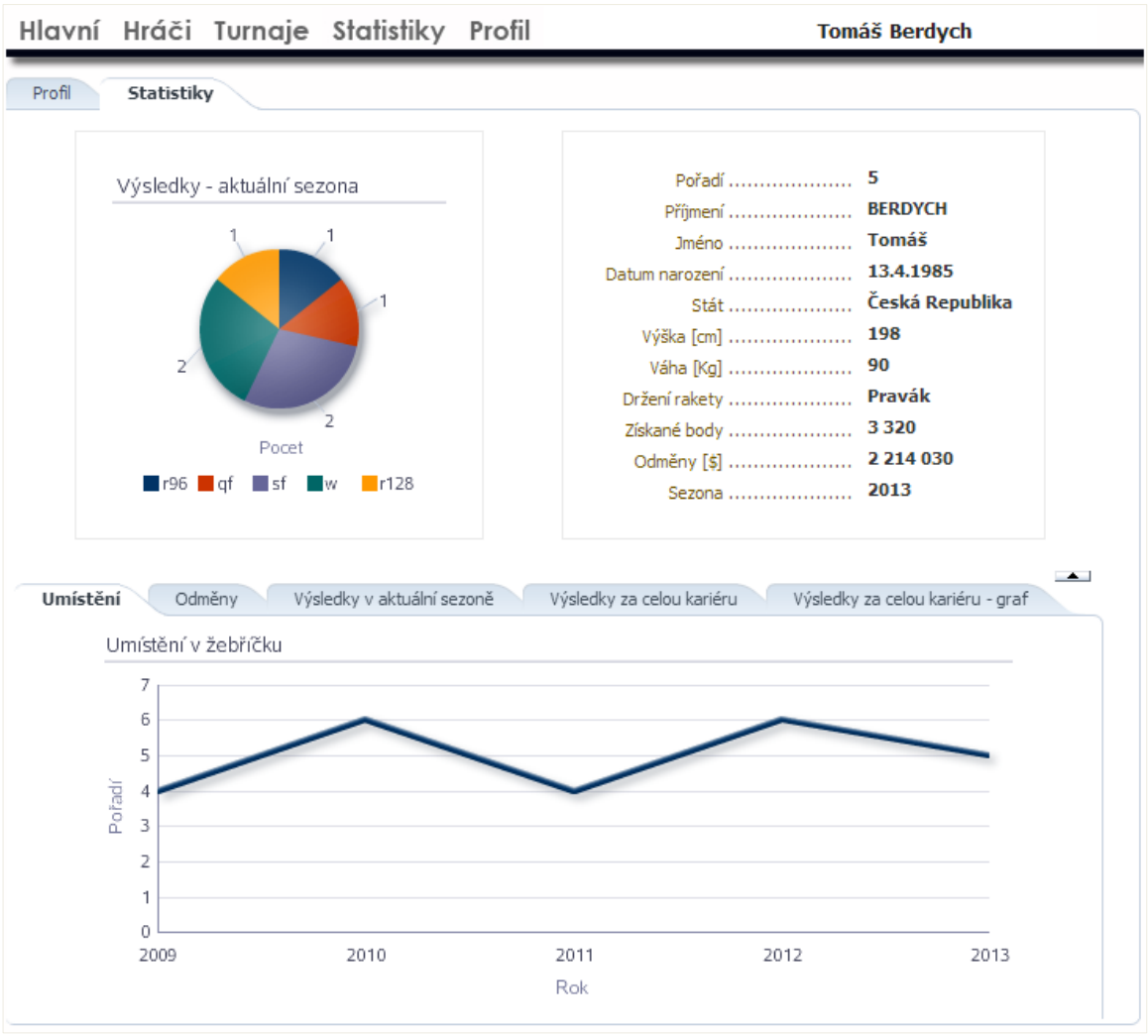
Následující část je zaměřena na práci se systémem z pohledu přihlášených uživatelů. Na Obrázek 42 se nachází přihlašovací formulář, ke kterému se uživatel dostane kliknutím na odkaz *Login*, umístěným v pravém horním rohu hlavní stránky. Pokud přihlášení proběhne úspěšně, zobrazí se uživateli hlavní stránka rozšířená o jednu položku v menu.



zdroj: autor

Obrázek 42 – Přihlášení do systému

Obrázek 43 představuje prostředí přihlášeného uživatele s rolí Hráč. Menu je rozšířené o položku *Profil* a vpravo od menu se dále nachází jméno přihlášeného uživatele.



zdroj: autor

Obrázek 43 – Profil hráč

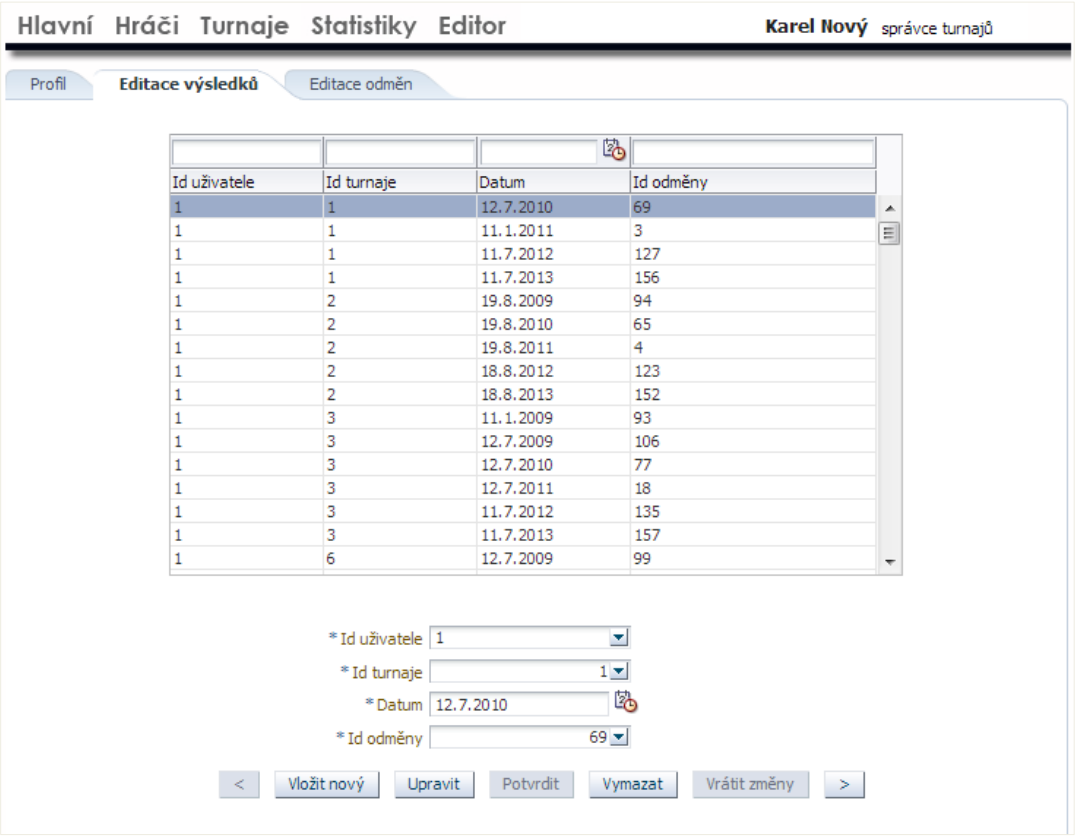
Po kliknutí na položku *Profil* se uživateli nabídne prostředí se dvěma záložkami:

- profil,
- statistiky.

V záložce *Profil*, se nachází základní informace o uživateli s možností editace v případě, že některá z informací již není aktuální. Ve druhé záložce se nachází statistiky patřící pouze přihlášenému hráči. Jak napovídá obrázek, je hráči k dispozici grafické znázornění dosažených výsledků v aktuální sezoně. Dále jeho osobní údaje a ve spodní části se nachází panel s pěti záložkami:

- *Umístění* - grafické znázornění umístění v žebříčku v jednotlivých sezonách.
- *Odměny* - grafické znázornění získaných odměn za umístění na turnajích v jednotlivých sezonách.

- *Výsledky v aktuální sezoně* - tabulka obsahující informace jakých výsledků hráč dosáhl v aktuální sezoně.
- *Výsledky za celou kariéru* - tabulka obsahující informace jakých výsledků hráč dosáhl ve všech odehraných sezonách.
- *Výsledky za celou kariéru - graf* - grafické znázornění počtu jednotlivých umístění za celou kariéru pomocí koláčového grafu.



zdroj: autor

Obrázek 44 – Profil správce turnajů

8.3 Správce turnajů

Další uživatelskou rolí s přístupem do systému je správce turnajů, jehož menu je rozšířeno o položku *Editor* (Obrázek 44). Po kliknutí na položku *Editor* se uživateli zobrazí panel se třemi záložkami:

- profil,
- editace výsledků,
- editace odměn.

Záložka *Profil* umožňuje uživateli upravovat osobní údaje skrze příslušný formulář. V dalších dvou záložkách uživatel vkládá, upravuje nebo maže výsledky turnajů respektive odměny za dosažené umístění na turnaji určitého typu.

zdroj: autor

Obrázek 45 – Profil správce systému

8.4 Správce systému

Na Obrázek 45 je zachyceno prostředí pro správu systému. V menu přibyla položka s názvem *Editor*. Vpravo od menu je zobrazeno jméno přihlášeného uživatele a jeho role. Správce systému má k dispozici panel se záložkami, ve kterých se nachází formuláře pro editaci příslušných tabulek. Jak napovídá obrázek, jedná se o následující záložky:

- vložení uživatele,
- editace turnaje,
- editace výsledků,
- editace odměn,
- typ turnaje,
- umístění,
- povrch,
- stát.

9 Závěr

Cílem teoretické části diplomové práce bylo představit vybrané Java frameworky a provést jejich vzájemné porovnání. V praktické části pak vytvoření tenisového informačního systému pomocí Oracle ADF Web Fusion. V úvodní části byly představeny jednotlivé frameworky spolu s jejich charakteristickými vlastnostmi. Následně byla vymezena kritéria, podle kterých probíhalo srovnání. Výsledek představují přehledně zpracované tabulky a grafy. Samotný Oracle ADF v této konkurenci obstál a právem se může řadit mezi vhodné kandidáty pro vytváření robustních databázových aplikací.

Výsledkem praktické části je funkční informační systém, jenž splňuje veškeré kladené požadavky. Aplikace umožňuje přihlašování uživatelů do systému a na základě jejich dané role jim zobrazuje příslušné administrativní prostředí umožňující správu dat všech databázových tabulek. Především se jedná o editaci uživatelů a turnajů, zaznamenávání tenisových výsledků a vytváření finančních odměn. Koncový uživatel, zpravidla tenisový fanoušek navštěvující informační systém prostřednictvím webových stránek, získává přehledné informace o hráčích a turnajích. Informace může vyhledávat, filtrovat i exportovat. Aplikace dále nabízí velké množství statistik prostřednictvím přehledných tabulek a grafů. Tenisoví hráči se mohou přihlásit do systému a aktualizovat svůj profil prezentovaný běžnému uživateli. Po přihlášení dále naleznou své osobní statistiky zpracovávané od začátku jejich kariéry. Na základě grafů a tabulkově zpracovaných výsledků je tenista schopen vyvodit důsledky a zaměřit se tak na zlepšení své hry. V budoucnu by mohl být informační systém dále rozšířen o možnost sledování online výsledků právě probíhajících zápasů nebo například o videogalerii obsahující sestřihy zápasů.

Vývojové prostředí JDeveloper spolu s Oralce ADF vytvářejí mocný nástroj pro tvorbu robustních aplikací. Na to jak velký v sobě skrývá potenciál, však není Oracle ADF příliš využívanou variantou. Pro české firmy to platí dvojnásob. K Oracle ADF není dostupná žádná česky psaná literatura. Většina dostupných tutoriálů vychází pouze ze základního modelového příkladu společnosti Oracle, který zdaleka nenaplňuje potenciál ADF. Tato skutečnost také působila největší překážky při samotné implementaci. Čas strávený hledáním možného řešení specifické úlohy několikanásobně převyšoval její samotnou implementaci.

Cílem diplomové práce nebylo vytvoření návodu pro budování Fusion aplikací, i tak by práce mohla být dobrým příkladem, čeho lze touto technologií dosáhnout.

Literatura

1. ORACLE. *Introduction to Oracle Database* [online]. 2011 [cit. 2013-08-21]. Dostupné z: docs.oracle.com/cd/E11882_01/server.112/e25789/intro.htm

2. *Model-View-Controller* [online]. 1998 [cit. 2013-08-21]. Dostupné z: http://ootips.org/mvc-pattern.html

3. ORACLE. *An Introduction to the Java EE Platform* [online]. 2012 [cit. 2013-08-21]. Dostupné z: http://docs.oracle.com/javaee/6/firstcup/doc/gcrky.html

4. ORACLE. *JavaServer Pages Overview* [online]. 2012 [cit. 2013-08-21]. Dostupné z: http://www.oracle.com/technetwork/java/overview-138580.html

5. JANSSEN, Cory. *What are JavaBeans* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.techopedia.com/definition/7865/javabeans

6. JANSSEN, Cory. *What is Enterprise JavaBeans (EJB)* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.techopedia.com/definition/443/enterprise-javabeans-ejb

7. ORACLE. *JavaServer Faces Technology Overview* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/overview-140548.html

8. Java Persistence API. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-08-21]. Dostupné z: http://cs.wikipedia.org/wiki/Java_Persistence_API

9. AJAX. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-08-21]. Dostupné z: http://cs.wikipedia.org/wiki/AJAX

10. ORACLE. *Introduction to XML* [online]. 2006 [cit. 2013-08-21]. Dostupné z: http://docs.oracle.com/javaee/1.4/tutorial/doc/IntroXML2.html

11. ORACLE. *Introduction to Building Fusion Web Applications with Oracle ADF* [online]. 2011 [cit. 2013-08-21]. Dostupné z: http://docs.oracle.com/cd/E16162_01/web.1112/e16182/intro.htm#BHCFFJHC

12. MILLS, Duncan, Peter KOLETZKE a Avrom ROY-FADERMAN. *Oracle JDeveloper 11g Handbook*. USA: McGraw-Hill, 2010. ISBN 978-0-07-160238-9.

13. KASARLA, Srivastav. *Introduction to Building Fusion Web Applications with Oracle ADF* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.aptude.com/it/easyblog/entry/developing-enterprise-applications-oracle-adf-vs-other-j2ee-frameworks

14. JOHNSON, Rod. *Introduction to the Spring Framework* [online]. 2005 [cit. 2013-08-21]. Dostupné z: http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

15. FOUNDATION, The Apache Software. *The Apache Struts web framework* [online]. 2005 [cit. 2013-08-21]. Dostupné z: http://struts.apache.org/

16. GUNTER, Ben. *Stripes Framework* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.stripesframework.org/display/stripes/Home

17. KOLESNIKOV, Alexander. *Tapestry 5: Building Web Applications: A step-by-step guide to Java Web development with the developer-friendly Apache Tapestry framework*. Birmingham: Packt Publishing, 2008. ISBN 1-84719-307-2.

18. VAYNBERG, Igor. *Apache Wicket Cookbook*. Birmingham: Packt Publishing, 2011. ISBN 1-84951-160-8.

19. ZORGDRAGER, Kelby. *Choosing the Right Java Web Development Framework* [online]. 2010 [cit. 2013-08-21]. Dostupné z: http://www.openlogic.com/wazi/bid/188143/Choosing-the-Right-Java-Web-Development-Framework

20. ORACLE. *Oracle JDeveloper* [online]. 2013 [cit. 2013-08-21]. Dostupné z: http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html

Seznam obrázků

Seznam tabulek