Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Systém pro vyhodnocování sportovní přípravy v judu

Plzeň 2012 Ondřej Havlíček

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 1. července 2012

Ondřej Havlíček

Abstract

The thesis, System for Evaluating sports training in Judo, aims at developing a Java application with multithread graphical user interface which manage Sport tests and Training dairy data. The application helps with creating outputs in pdf formats. The thesis describe evolution of the application from specification to distribution and general methods of evaluating training. Using another frameworks like JasperReports or JFreeChart and MySQL database in data layer.

Obsah

1 Úvod

Software pro vyhodnocování sportovní přípravy je ucelenou aplikací vyvíjenou pro potřeby sportovních center mládeže (dále SCM) i jednotlivých klubů juda v České republice. Jeho vytvoření bylo podníceno jednak potřebou uchovávat množství testů pro potřeby jak samotných oddílů, kvůli vyhodnocování efektivity přípravy, tak i SCM, kde se k výsledkům motorických testů přihlíží při hodnocení celkové přípravy. Dále se systém využívá pro vyhodnocování testovacích cyklů v rámci jednotlivých reprezentačních družstev.

Systém pro motorické testy společně s tréninkovým deníkem a závodními výsledky přináší zpětnou vazbu pro trenéry, kteří následně upravují tréninkový plán podle aktuálních potřeb v jednotlivých fázích přípravy.

Cílem klubů, SCM i reprezentačních družstev je, aby systém byl naprosto intuitivní a uživatelsky přívětivý, ale zároveň splňoval náročná kritéria v podobě plně automatizovaného generování výsledků přípravy a dalších statistik popsaných dále v dokumentu. Dále pak, aby splňoval požadavky pro vzdálený přístup k datům přes internet.

V práci je vysvětlen způsob vyhodnocování sportovní přípravy v judu, vytváření a hodnocení tréninkového plánu s použitím tréninkového deníku a již existující systémy. Dále je popsaný návrh a implementace nového systému v jazyce Java za použití databáze MySQL, frameworku pro výstup do PDF/HTML formátů JasperReports, nástroje pro tvorbu a export grafů JFreeChart a okrajově knihoven SwingX.

Dále se v práci zabývám testováním výsledné aplikace, jejím nasazením do běžného provozu v Judoclubu Plzeň a ve Středisku volného času v Plzni včetně distribuce spolu s některými jejími problémy.

2 Sportovní příprava v judu

2.1 Judo jako sport

Soutěžní judo se rychle rozšiřuje po celém světě a stává se stále oblíbenějším sportem pro cvičence ve všech věkových a dá se říci i váhových kategoriích. V judu se využívá rychlých pohybů, dobré rovnováhy, koordinace pohybu a sebedisciplíny. V zápasech se používají zejména hody, držení, páky a škrcení. Všechna tato odvětví musí dobrý judista dokonale ovládat.

Judo vzniklo v Japonsku v klášteře Eišo-ji v Tokiu[But09] a převzalo do sebe japonskou kulturu a filozofii. Založil ho Jigoro Kano jako úpravu klasického systému sebeobrany Jiu-Jitsu v roce 1882. Do současné podoby prošlo ještě mnoha dalšími úpravami, rozšiřováním a vývojem.

Od roku 1964 je judo olympijským sportem. V tomto roce však mohli soutěžit pouze muži. Ženské judo se na olympiádu dostalo až o dlouhých 28 let později v Barceloně.

2.2 Motorické testy

Pohybovou činnost ovlivňuje velmi mnoho pohybových schopností a dovedností současně. To platí v judu především. Zde se zapojují velmi významnou měrou síla, rychlost, vytrvalost i obratnost.

Motorické schopnosti jsou skryté vlastnosti jedince. Můžeme je však testovat při použití specifických disciplín. Schopnosti se u každého jedince s věkem rozvíjejí. Je však nutné trénovat a tyto schopnosti dále aktivně rozvíjet, aby nedošlo k jejich nečinnosti.[Měk05]

Motorické testy jsou právě prostředkem pro testování některých specifických dovedností. Test je vlastně sestavou jeho disciplín. Jejich prováděním jsme schopni určit aktuální schopnosti jedince. Při znalosti určitého souboru výsledků testů jsme schopni určit i vývoj konkrétních dovedností v čase.

Je nutné poznamenat, že ačkoli test jako takový obsahuje konkrétní počet disciplín, nemusí se vždy provádět jako celek. Často je vybrána podmnožina disciplín, které se v konkrétním testovacím případě otestují.

V judu se používají především tři typy testů. Ty jsou rozděleny podle věkové kategorie. Jedná se o test Junior+, který slouží k testování především juniorů a mužů. Často se tento test používá i pro výkonostně lepší dorostence. Dále je to test přímo pro dorostence a test pro žactvo.

Platí, že se vždy celá skupina testuje podle jednoho konkrétního testu, i když mohou mít mezi sebou cvičenci menší věkové rozdíly.

2.2.1 Hodnocení motorických testů

Hodnocení je uzpůsobeno povaze soutěžního juda, kde jsou sportovci rozdělováni podle věku a váhové kategorie. Například pro juniory a muže existují váhové kategorie -60, -66, -73, -81, -90, -100 a +100 (kg) [Srj95]. Testy se samozřejmě vyhodnocují podle váhové i věkové skupiny. Pro názornost přidávám tabulku vyhodnocení disciplíny bench-press pro muže. Viz tabulka 1.

	1	2	2	4
60 kg	85	80	75	70
66 kg	90	85	80	75
73 kg	95	90	85	80
81 kg	100	95	90	85
90 kg	110	105	100	90
100 kg	120	115	110	100
+100 kg	125	120	115	110

Tabulka 1: Tabulka pro hodnocení bench-pressu mužů.

2.2.2 Výběr talentované mládeže

Jedním z důvodů pro testování je výběr talentované mládeže. Na judo se hlásí mnoho dětí školního(a částečně i předškolního) věku, kteří ještě velmi významně rozvíjí své motorické schopnosti a je potřeba z nich vybrat talentované jedince pro SCM[1]. SCM jsou rozmístěna po celé republice a nejlepší judisté si v nich vylepšují své schopnosti a dovednosti pod vedením profesionálních trenéru s mnohaletými závodními a trenérskými zkušenostmi.

[1]SCM – Sportovné centra mládeže jsou organizace podporující rozvoj talentované mládeže.

2.2.3 Testy v rámci oddílů

V rámci jednotlivých sportovních oddílů se pravidelně provádí motorické testy. To přináší několik výhod. Jednou z nich je samozřejmě vyhodnocení schopností konkrétního sportovce jako jedince. Další a důležitější výhodou je vzájemné porovnání výsledků mezi členy skupiny.

Relativní hodnocení je ve skupině mnohem prospěšnější než hodnocení absolutní. Sportovci se navzájem motivují k lepším výsledkům a v rámci celé skupiny se vyselektují výrazně slabší nebo naopak silnější jedinci. To zásadně pomáhá při rozdělování cvičenců do skupin.

2.2.4 Testy v kontextu reprezentačních družstev

Ve většině sportů, a v judu především, je potřeba, aby všechny schopnosti byly vyvážené a na dobré úrovni, testují se i ti nejlepší judisté, tedy ti, co jsou v jednotlivých reprezentačních družstvech. Pro ně přináší testy hlavně informace o tom, v jakých oblastech (síla, rychlost, vytrvalost, obratnost) jsou nejslabší.

I když jsou reprezentanti na velmi dobré úrovni, často mají v jedné z těchto oblastí mírně horší výsledky. Pak je na nich, aby se v tréninku specializovali na rozvoj té které konkrétní oblasti. V případě vrcholových sportovců se samozřejmě nehledí pouze na tyto sestavy testů, ale jsou spolu s dalšími sportovními, ale i lékařskými testy (funkční kapacita plic, hormonální vyšetření atd.) jedním z hodnocených aspektů.

2.3 Příprava tréninkového plánu

Během roku se střídají přípravná a závodní období. Tréninkový plán je připravován v závislosti na těchto obdobích. Mezi základní sledované parametry přípravy v judu patří technicko-taktická příprava, randori[2], silová příprava, funkční rozvoj a regenerace.

Během jednotlivých fází přípravy se mění poměr těchto parametrů. Ve fázi přípravné převládá funkční rozvoj, silová příprava a technicko-taktická příprava. Ve fázi závodní pak výrazně převládá randori a regenerace.

Příprava tréninkového plánu je komplexní záležitost. Je třeba brát ohledy na předchozí přípravu i na budoucí činnosti (soustředění, turnaje…). Důležitý je i psychický stav týmu.

2.4 Tréninkový deník

Přípravu tréninkového plánu je nutné provádět dopředu. Zároveň je potřeba jí zpětně vyhodnocovat. K tomu slouží tréninkový deník. Do deníku se zapisuje pouze čas strávený v konkrétním dni danou činností (parametrem přípravy) a počet tréninkových baterií.

2.4.1 Hodnocení týdne

Jednotlivé kluby jsou často povinné evidovat přípravu a vhodným způsobem ji prezentovat. Například ve Středisku volného času v Plzni se každý jednotlivý týden

[2]Randori je forma cvičného boje v judu, kdy oba judisté bojují naplno.

eviduje a výsledky se předkládají ke kontrole vedení. Proto je potřeba aby deník obsahoval funkci pro vygenerování výstupu v některém z obecně známých formátů (například PDF nebo HTML).

2.4.2 Hodnocení bloků přípravy

V některých speciálních případech je potřeba vyhodnotit několik týdnů společně. Často se vyhodnocují bloky týdnů v případě, že se zpětně hodnotí příprava, kdy se porovnají výsledky z turnajů nebo motorických testů s časy strávenými v jednotlivých přípravných činnostech. Výsledky slouží pouze jako informace trenérům a není tedy potřeba generovat výstupy s hodnocením bloků.

2.5 Původní systém

Jak jsem již popsal, program sjednotí dva již existující systémy napsané v Excelu.

Původní systém tvořily 2 nezávislé sešity progamu Microsoft Excel, které nebyly nikterak provázané a doprovázela je poměrně velká množina chyb a nepřesností. O uživatelské přívětivosti se ani nedá mluvit. Pro vytváření nových položek tabulky bylo nutné kopírovat některé řádky s vyhodnocovacími pravidly z jednoho místa na druhé, často i mezi listy sešitu. To s sebou přinášelo samozřejmě další problémy a dávalo možnost vzniku dalších chyb.

2.5.1 Motorické testy

Systém pro vyhodnocování motorických testů tvoří jeden excelový sešit se dvěma listy (junioři/juniorky, dorostenci/dorostenky). Na každém listu byl řádek osbahující kolonky jméno, příjmení, váha, a výsledky v jednotlivých disciplínách. Vedle každé disciplíny bylo pole s vyhodnocovací funkcí pro danou disciplínu.

Při vytvoření nového testu, bylo nutné vytvořit nový list a zkopírovat do něj tolik řádek, kolik se mělo testu zůčastnit osob. Pokud uživatel neuměl vložit data včetně formátování, musel pak ještě ručně rozšiřovat sloupce.

Ani tisk nebyl tak jednoduchý, jak by se přepokládalo. Často se tabulka vložila tak, že přesahovala přes hranice tisku a vytiskly se 2 nebo i 4 stránky.

2.5.2 Tréninkový deník

Podoba tréninkového deníku byla snad ještě horší, než podoba motorických testů. Představoval ho pouze jeden list. Vždy při vytvoření nového týdne si uživatel musel vytvořit novou tabulku a přidat k ní novou řádku se součty jejich buňek.

V aplikaci sice byla nějaká tlačítka, kterým byla přiřazena makra pro jistou automatizaci. Naneštěstí tato makra nebyla vždy plně funkční a někdy dokonce rozhodila celou strukturu stránky. Tudíž se nepoužívala.

Složité bylo i vyhodnocení bloku týdnů, jelikož se musely kopírovat řádky, kde byly vzorce pro součet. Bylo tedy nutné provést kopírování hodnot přes pravé tlačítko myši, což nebylo zrovna uživatelsky přívětivé.

3 Specifikace programu

3.1 Funkce produktu

Hlavní funkce se dělí do dvou kategorií. První kategorií jsou funkce související s částí pro motorické testy.

- Správa osob v databázi (přidávání/odebírání),
- správa tréninkových skupin (vytváření, mazání, přidávání/ubírání osob)
- vytvoření, vyhodnocení a uložení jednotlivých testů,
- výpis výsledků jednotlivých testů (dle data testu),
- výpis výsledků všech testů konkrétního jedince,
- vyhodnocení výsledků jednotlivce a grafická vizualizace,
- vytvoření reportu (formát PDF) o vývoji výsledků jednotlivce.

Druhou kategorií jsou funkce související s částí tréninkového deníku.

- Vytvoření nového týdne,
- uložení hodnot pro jednotlivé dny i celé týdny sportovní přípravy,
- výpis hodnot jednotlivých týdnů včetně součtů pro jednotlivé ukazatele,
- výpis hodnot pro jednotlivé bloky přípravy (obvykle 2-4 týdny dohromady) včetně součtů pro jednotlivé ukazatele,
- grafická vizualizace poměru časů strávených sledovanými ukazateli v jednotlivých blocích),
- vytvoření reportu s vizualizacemi jednotlivých bloků (formát PDF).

3.2 Třídy uživatelů

Typický uživatel systému je trenér sportovního družstva. Předpokládá se, že uživatel je seznámen se základním ovládáním PC. Vše důležité pro obsluhu produktu bude popsáno v uživatelské dokumentaci.

3.3 Provozní prostředí

3.3.1 Softwarové požadavky

Systém bude naprogramován v jazyce Java. Poběží tedy všude tam, kde je Java nainstalovaná.

Data budou ukládána v multiplatformním databázovém systému MySQL, který bude nainstalován na vzdáleném serveru. Hosting na serveru, nainsatlování databáze a nahrání schématu databáze je součástí projektu.

3.3.2 Hardwarové požadavky

Hardwarové požadavky jsou totožné s požadavky Javy. Doporučené minimální požadavky jsou alespoň 1Gb RAM a 1GHz procesor. Mezi další požadavky patří možnost připojení k síti internet(kvůli databázi), tedy LAN, Wi-Fi, Bluetooth či jiné rozhraní pro připojení k internetu.

3.3.3 Vnější systémy a softwarové komponenty

Produkt bude spolupracovat s databází na vzdáleném serveru. Bude spolupracovat s nástroji JasperReport a JFreeChart, které budou přímo součástí distribuce produktu.

3.4 Uživatelská dokumentace

K aplikaci bude dodána uživatelská dokumentace ve formátu pdf, která bude určena pro třídu uživatelů popsanou výše, tedy bude velice podrobně řešit jednotlivé případy užití.

4 Analýza a návrh

Analýza problému vychází z analýzy již hotových řešení popsaných v kapitole 2.6 a z dokumentu specifikace zadání[Spe12]. Výsledný program přebírá funkčnosti původních řešení, vylepšuje je, odstraňuje chyby a přináší příjemné grafické uživatelské prostředí, v němž nedává uživateli prostor, aby celý program, či dokonce data znehodnotil.

Velkou výhodou bylo, že oba produkty i doménu aplikace jsem velmi dobře znal. Nečinilo mi tedy problémy porozumění specifikovaným požadavkům a bylo pro mě jednodušší i samotné naprogramování aplikace, včetně návrhu funkcí, které nebyly přímo specifikovány. Zároveň byla mnohem jednodušší i analýza problému.

4.1 Grafické uživatelské rozhraní

Aby bylo uživatelské prostředí přívětivé, bylo nutné vymyslet rozvržení komponent tak, aby bylo pro co největší skupinu operací stejné či velmi podobné.

Jelikož se jedná o poměrně rozsáhlý program, není jednoduché vytvořit základní stukturu okna a tu dodržovat v celé aplikaci. Nakonec jsem zvolil pár kompromisů zobrazení a pro všechny (až na správu osob) činnosti jsem navrhl, až na pár detailů, stejné rozložení komponent.

4.1.1 Základní rozvržení komponent

Pro operace (zobrazování, změna, vyhodnocení a exportování výsledků) v části motorických testů a tréninkového deníku jsem navrhl rozhraní se třemi základními panely.

Levý panel slouží pro výběr testu či týdne, který je reprezentovaný datumem a v případě testu i jménem trenéra, který ho provedl. Po výběru položky se okamžitě zobrazí potřebná data v druhé části okna.

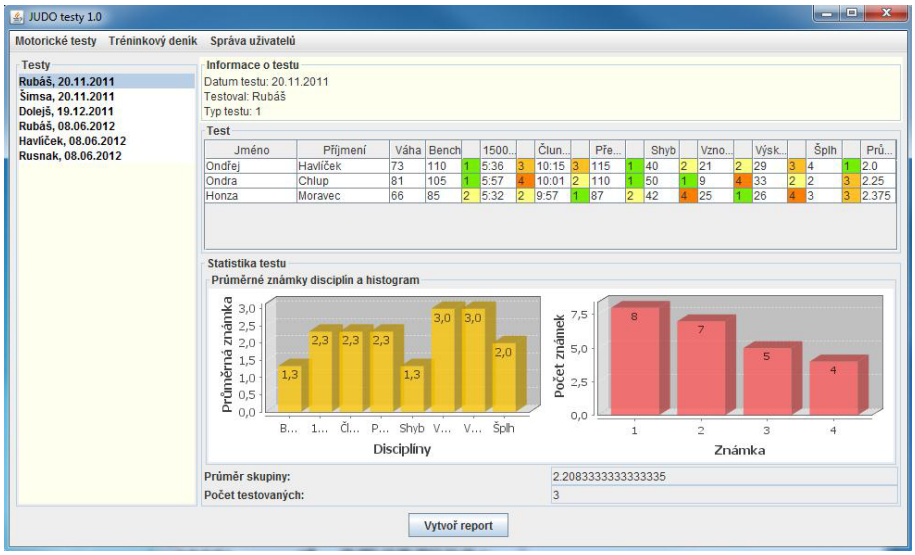
Zbytek okna je rozdělen na dva panely, které zobrazují informace o aktuálně vybrané položce. První (horní) panel, obsahuje tabulku a základní informace o položce. Dolní panel pak zobrazuje statistiku zobrazených dat v podobě grafů. Obsah těchto částí se samozřejmě liší podle aktivity, kterou uživatel provádí. Bližší popis tedy musí být uveden pro každou činnost zvlášť.

Vespodu celého okna je lišta pro tlačítka, obsahující tlačítko pro vytvoření výstupu do formátu PDF.

Motorické testy V případě, že je zobrazeno okno s hodnocením motorických testů, pak tabulka obsahuje v každém řádku jméno, příjmení, váhu a výsledky v testových disciplínách spolu s jejich známkami. Na konci řádky je průměrná známka ze všech disciplín.

Statitistika představuje 2 grafy. Prvním grafem je sloupcový graf, kde každý sloupec vyjadřuje průměrnou známku dosaženou v jedné konkrétní disciplíně testovanými jedinci. Druhý graf je histogram známek. Dále se zde zobrazí průměr celé skupiny (průměr průměrů jednotlivců) a počet osob v testu.

V případě vyhodnocení jedince se graf redukuje na jeden s významem vývoje výsledků v jednotlivých skupinách v čase. Je pak možné sledovat sportovcův vývoj v grafické podobě sloupcového grafu.

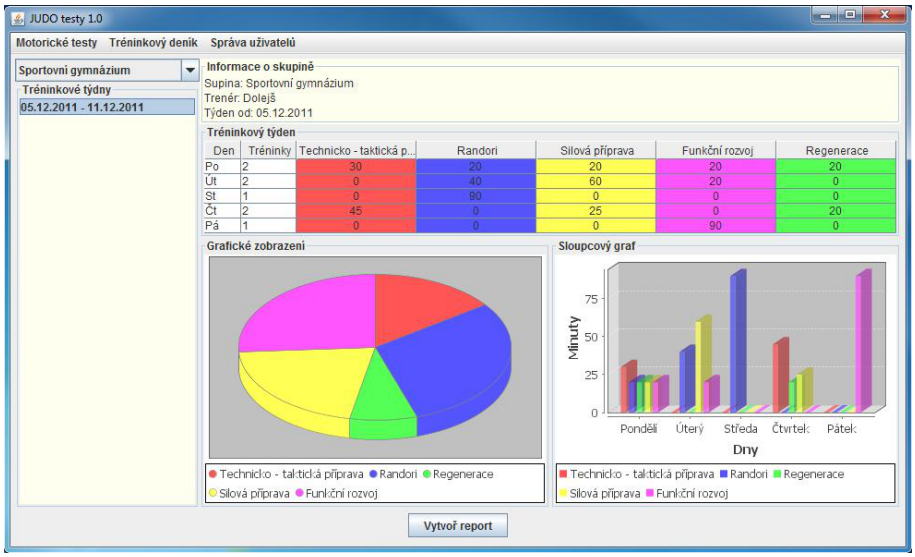


Obrázek 1: Okno motorických testů.

Tréninkový deník Pro práci s tréninkovým deníkem obsahuje tabulka data o sledovaných parametrech přípravy, viz 2.3. Je zde oproti motorickým testům také neměnný počet řádek. Tedy 5 řádků reprezentujících pět dní přípravy.

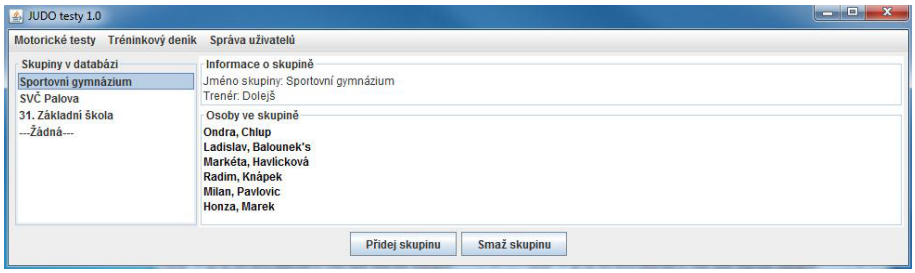
Grafy představují v prvním případě poměr časů strávených jednotlivými ukazateli přípravy v koláčovém grafu. V druhém případě jde o zobrazení poměru v konkrétních dnech ve sloupcovém skupinovém grafu.

V tomto případě jsou sloupce v tabulce obarvené podle barvy, jakou se zobrazují data v grafech. To přináší uživatelům lepší orientaci mezi grafy a tabulkou.



Obrázek 2: Okno tréninkového deníku.

Správa skupin Rozložení panelů pro správu skupin jsem navrhl částečně podle základního rozvržení pouze s jednou malou změnou. Hlavní okno neobsahuje kromě výpisu osob, žádné další údaje, chybí tedy tabulka a část s grafy. Dále obsahuje navíc tlačítka přidat/odebrat skupinu. Nicméně graficky se zásadním způsobem neliší.



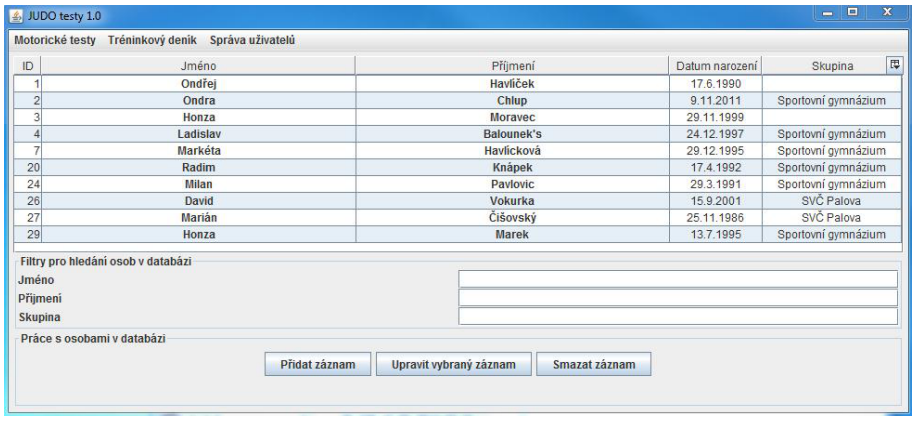
Obrázek 3: Okno správy skupin.

Správa osob Správa osob je jediný případ, kdy se liší rozvržení komponent zásadnějším způsobem. Jelikož mají informace o osobách v databázi ryze tabulkový charakter, obsahuje okno jednu tabulku, kde jsou zobrazeny všechny osoby v databázi.

Podle požadavků bylo potřeba osoby filtrovat podle jména, příjmení a skupiny, přidal jsem zde tedy panel pro zadání filtrovacích parametrů. Vespodu okna se nachází panel s tlačítky pro přidání/úpravu/smazání záznamu o osobě z databáze.

4.1.2 Zadání parametrů nového testu a týdne

Pro zadání parametrů nových testů a týdnů vkládaných do programu jsem zvolil vyskakovací okna. Zde už by bylo dodržení společných rysů rozvržení panelů asi proti požadavku na přívětivé a intuitivní ovládání. Oba případy jsou tedy navrženy zásadně rozdílně.

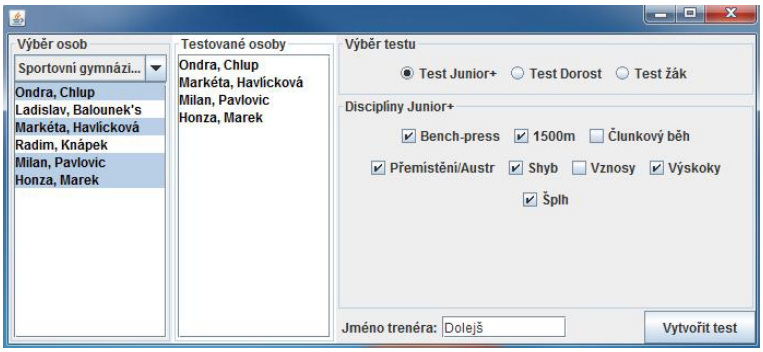


Obrázek 4: Okno správy osob.

Nový test V případě zadání nového testu jsem řešil zásadní problém v tom, že jeden konkrétní test může absolvovat buďto celá skupina nebo její podmnožina, ale zároveň i kombinace osob z více skupin. To přináší otázku, jakým způsobem vybrat osoby k testu.

Vymyslel jsem způsob, který slučuje tyto požadavky s nepříliš složitou obsluhou. V části pro výběr uživatelů jsou 2 seznamy osob. Nad levým panelem je navíc výběr skupiny. Levý panel pak obsahuje osoby, které jsou ve vybrané skupině. Pomocí drag&drop akce uživatel přenese osoby, které chce testovat do seznamu v pravo.

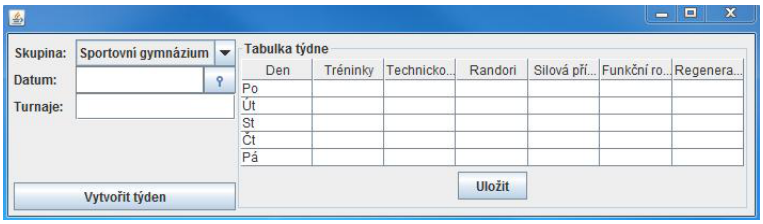
Dále je potřeba vybrat typ testu a požadované disciplíny. K tomu jsou již navrženy standartní zaškrtávací pole s volbou „jedna z více“ v případě typu testu či výběr libovolného nenulového počtu polí v případě disciplín.



Obrázek 5: Okno přidání nového testu.

Přidání nového týdne V případě nového týdne jsem vytvořil jednoduché okno pro zadání skupiny, datumu, počtu turnajů a tlačítka pro vytvoření nového týdne. Ten pak bylo nutné editovat.

Po další rozvaze a komunikaci s uživateli jsem k tomuto přidal ještě tabulku, aby bylo možné rovnou zadat potřebaná data. K tabulce bylo nutné přidat ještě tlačítko pro uložení týdne.



Obrázek 6: Okno přidání nového týdne.

4.1.3 Další prvky grafického rozhraní

Součástí uživatelského rozhraní jsou i další vyskakovací okna a dialogy. Jejich návrh probíhal v podstatě během programovací fáze. Jedná se vždy o jednoduchá okna buďto informačního charakteru (pouze tlačítko „OK“), charakteru dialogu s volbami ANO či NE popřípadě formulářový typ okna, kde je několik polí k vyplnění a potvrzovací tlačítko.

4.2 Databáze

Při návrhu jsem se zabýval i přípravou datové vrstvy. Uchovávat data v jednotlivých počítačích nebylo možné, protože je potřeba, aby k nim mělo přístup více trenérů v klubu najednou. Tím pádem bylo nutné použít databázi. Napadla mě dvě možná řešení.

4.2.1 XML databáze a XSLT

Jednou z možností bylo vytvoření XML[3] databáze. V případě uložení do XML databáze by bylo možné výsledné reporty generovat pomocí XSLT[4] a poměrně robustně se dotazovat pomocí XPath. Navíc je XML „technologie dneška“, tudíž je o ní všude spousta informací a existuje rozsáhlá množina technologií, které jej zpracovávají.

Nevýhodou je, že ačkoli XML je technologie poměrně etablovaná, s XML databází jsem se ještě nesetkal. Tudíž by její návrh přinesl velkou spoustu problémů a nepříjemností. Tedy především z důvodu neznalosti XML databází jsem se rozhodl, že se touto cestou nevydám.

4.2.2 Relační databáze a JasperReports

Druhou možností je použití klasické relační databáze. Pro vytváření reportů se v tomto případě musí použít nějaký nástroj na generování výstupů přímo z objektů Javy. Bylo by samozřejmě možné data načíst z databáze, převést nějakou dostupnou technologií do XML a pak udělat XSL resp. XSL-FO[5] transformaci do PDF nebo HTML. Nicméně toto řešení je poměrně komplikované.

Rozhodl jsem se, že data načtená z databáze přetransformuji do výstupního formátu nástrojem JasperReports a grafy pro tento výstup vygeneruji pomocí knihovny JFreeChart.

[3] eXtensible Markup Language - jazyk popisující data nezávisle na platformě.

[4] Technologie zabývající se zpracováním XML. Hlavním účelem je transformace dokumentů do různých formátů například XML, HTML, XHTML, TXT nebo i grafické SVG.[Her10]

[5] Dvoukroková transformace - v prvním kroku převod XML na FO (formátovací objekty) a ve druhém FO do výstupního formátu (například PDF, nebo RTF)

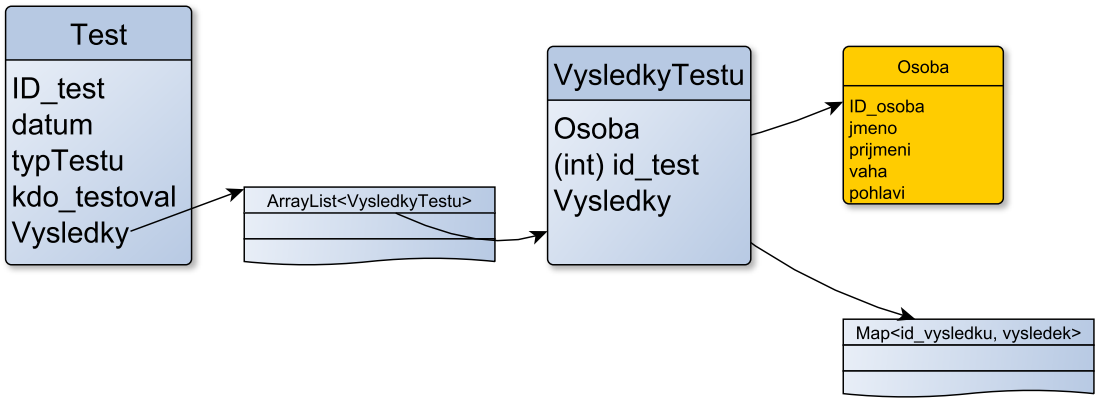
4.2.3 Model databáze

Použitelnost databáze závisí do jisté míry na jejím návrhu. Proto jsem jejímu návrhu věnoval maximální pozornost. Cenné rady jsem našel především v knize Databáze bez předchozích znalostí [Opp06]. Celý E-R-A model databáze je v příloze A.

4.3 Datové struktury

Po návrhu databáze je nutné navrhnout objekty, které budou data z databáze uchovávat. Některé jsou jednouduché sestávající se z jedné třídy. Další mohou být složitou strukturou objektů a jejich kolekcí. V práci jsem navrhl dvě složitější struktury, které popíšu pomocí diagramu.

Nejdříve bylo nutné rozmyslet, jak bude program uchovávat informace o testech. V databázi k tomu slouží 3 tabulky. Navrhl jsem tedy řešení o třech třídách. Jeden konkrétní testový případ představuje třída `MotorickyTest`. Ta obsahuje základní informace o testovém případě a list ojektů `VysledkyTestu`. Každý tento objekt představuje jednu osobu a její výsledky. Informace o osobě jsou v jeho proměnné `osoba` a jednotlivé výsledky v konkrétních disciplínách jsou uloženy v mapě `vysledky`. Situaci popisuje obrázek 7.



Obrázek 7: Struktura pro uložení testu.

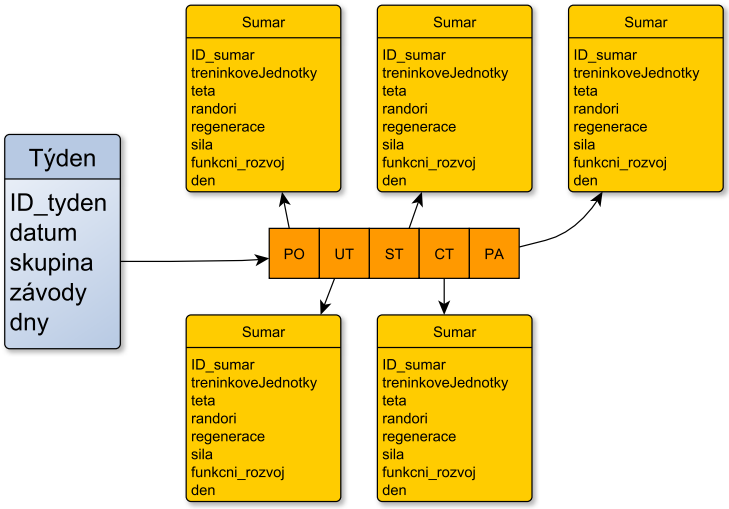
Dále bylo potřeba navrhnout strukturu pro motorické testy. Zde jsem se rozhodl pro poněkud statickou formu struktury. Je vytvořena třída `Tyden,` která obsahuje základní informace o týdnu deníku a pole pěti objektů `Sumar`.V každém z nich jsou uloženy informace o jednom konkrétním dni. Struktura je vyobrazena v obrázku 8.

4.4 UML modely pro analýzu chování

UML[6] je univerzální jazyk pro vizuální modelování systémů[Arl03]. Je hojně používán v objektově orientovaném programování. Jeho součástí je poměrně velká skupina digramů - diagram tříd, diagram nasazení, digram užití atd. Při návrhu jsem použil diagram užití.

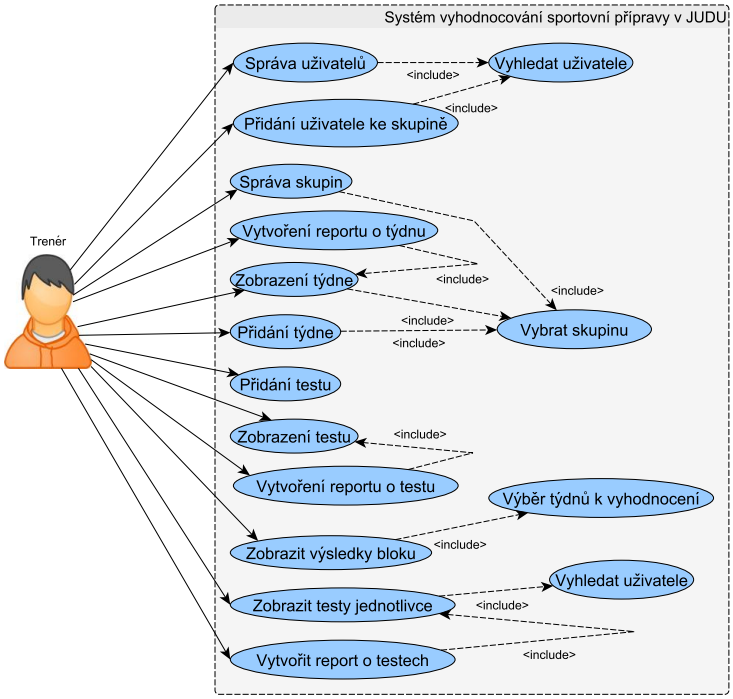
Tento diagram patří do skupiny diagramů chování. Popisuje jakým způsobem budou uživatelé ovlivňovat a používat systém. Vychází z analýzy specifikace požadavků a slouží následně jako vodítko při vytváření programu. Dále je využit v jednom

[6]Unified Modeling Language, unifikovaný modelovací jazyk



Obrázek 8: Struktura pro uložení týdne.

z testů viz kapitola 6.4. Jak je vidět na obrázku 9, v tomto systému je pouze jeden aktér. Tedy pouze jedna role s jakou může uživatel k softwaru přistupovat.



Obrázek 9: Diagram případů užití.

V příloze B je navíc zjednodušený diagram tříd. Obsahuje všechny důležité třídy včetně jejich vzájemných vazeb, metod a atributů.

5 Implementace

V této kapitole bych rád popsal implemetační detaily práce. Programovací část samozřejmě obsahovala velké množství „implementačně nudných“, ale zároveň několik

(pro mě) nových věcí. Právě tyto pro mě nové, nebo i zajímavé, bych rád detailněji rozebral.

5.1 Přístupnost aneb vlánka v uživatelském rozhraní

Při programování grafického rozhraní, je nutné dbát na dodržení několika zásad. Rozhraní je první co může na našem programu uživatele zaujmout, či odradit. Správně napsaná aplikace by neměla „zamrzat“ a měla by být stále schopná reagovat na zadání uživatele [Tut06]. Pokud je přece jen nutno, aby byl program nedostupný, je vhodné aby zobrazil, že pracuje.

V práci používám například `SwingWorker`. Pokud je v programu potřeba provést dlouhotrvající operaci, jako načtení velkých dat z databáze nebo generování složitého výstupu, je vhodné tyto akce provádět ve vlastním vlákně. To velmi dobře umožňuje právě `SwingWorker`.

V ukázce 1 je vidět vlákno které je vytvořeno jako anonymní vnitřní třída a jeho funkcí je generování výstupního PDF souboru. V případě, že by se generování neprovádělo ve vlastním vlákně, došlo by k tomu, že po stisku tlačítka by do dokončení akce zůstalo stisknuté a ostatní funkce rozhraní byly nedostupné - bylo by takříkajíc „zamrzlé“. To je způsobené tím, že všechny akce související s rohraním se provádějí v jednom konkrétním vlákně – EDT[7]. A tam spadá i kód který se provádí v obsluze tlačítek.

V ukázce 2 je pak vidět celá třída, dědící od `SwingWorkeru`. Vlákno, které tato třída reprezentuje v metodě `doInBackground()` načte z databáze osoby. Po jejím dokončení tato data vloží do modelu tabulky a zobrazí.

Ukázka kódu 1: SwingWorker - použití.

```
1  public void actionPerformed(ActionEvent e) {
2
3    SwingWorker<Void,Void> vlakno = new SwingWorker<Void,Void>(){
4      @Override
5      protected Void doInBackground() throws Exception {
6        new Report_TestJednotlivec();
7        return null;
8      }
9    };
10
11   vlakno.execute();
12 }
```

Ukázka kódu 2: SwingWorker - použití.

```
1  public class Worker_SpravaUzivatelu
2      extends SwingWorker<List<Osoba>, List<Osoba>> {
3
4    JTable tabulka;
5
6    public Worker_SpravaUzivatelu(JTable table) {
7      tabulka = table;
8    }
```

[7] Event Dispatch Thread

```
9
10    @Override
11    protected List<Osoba> doInBackground() throws Exception {
12      return Datova.Databaze.nactiOsobyList();
13    }
14
15    @Override
16    protected void done() {
17      try {
18        tabulka.getModel().setNewData(this.get());
19        tabulka.getModel().fireTableDataChanged();
20      }
21      catch (Exception e1) {
22        e1.printStackTrace();
23      }
24    }
```

5.2 Práce s databází MySQL

Pro obsluhu databáze slouží třída `Databaze` z balíku `Datova`. Pro připojení do databáze je použita technologie JDBC neboli Java DataBase Connectivity. JDBC API poskytuje jednotný přístup k databázím, ke kterým existuje JDBC ovladač, což je dnes většina používaných databází. Tím odstiňuje uživatele od konkrétního typu databáze.

V ukázce 3 je vidět spůsob jakým je možné se připojit do databáze a získat objekt `Connection,` přes který je pak možné provádět dotazy. Url pro připojení obsahuje adresu, port a název databáze. Navíc může obsahovat i další paramery. V tomto případě kódování a nastavení automatického znovupřipojení v případě výpadku spojení.

Ukázka kódu 3: Vytvoření připojení do databáze.

```
1   String url =
2   "jdbc:mysql://10.77.32.124:3306/db_judoclubplzen?" +
3   "autoReconnect=true&useUnicode=true&characterEncoding=utf8";
4
5   Connection conn;
6   Class.forName("com.mysql.jdbc.Driver").newInstance();
7   conn = DriverManager.getConnection(url, "name", "pass");
```

Samotné dotazy do databáze jsem ukládal do objektu `PreparedStatement` způsobem z ukázky 4. Tedy vytvořením jeho instance z instance objektu připojení k databázi. Výhodou tohoto řešení je především přehlednost a jeho univerzálnost. Místo otazníků do dotazu můžeme dosadit jakékoli hodnoty. Vélké ulehčení to přinese v případě řetězců, kdy není nutné řešit použití speciálních znaků jako například zpětné lomítko nebo při ukládání čísel, kdy program může provést typovou kontrolu.

Ukázka kódu 4: Vytvoření PreparedStatement objektu s dotazem.

```
1   PreparedStatement pst = pripojeni.prepareStatement("UPDATE
      vysledek SET hodnota = ?, znamka = ? WHERE id_vysledek = ?"
      );
```

```
2       pst.setString(1, hodnota);
3       pst.setInt(2, znamka);
4       pst.setInt(3, id_vysledku);
5
6       pst.execute();
```

5.2.1 Transakce v databázích

V některých případech vkládání hodnot do databáze, při perzistování složitých objektů, jsem potřeboval data vkládat v transakci[8]. Jednoduchý způsob popisuje kód 5.

Ukázka kódu 5: Vytvoření transakce.

```
1 pripojeni.setAutoCommit(false);
2   try {
3     //Zde je seznam dotazů, které mají být součástí transakce.
4     pripojeni.commit();
5   } catch (SQLException e) {
6       pripojeni.rollback();
7   }
8 pripojeni.setAutoCommit(true);
```

5.3 JasperReports

JasperReports[9] je knihovna pro generování kvalitních dokumentů, jejich prohlížení, tisk a export do většiny známých formátů jako jsou PDF, HTML, Excel, ale třeba i OpenOffice. Její použití je zdarma a je šířena pod GNU General Public Licence[10].

Značnou výhodou je, že je celá napsaná v Javě, a je tedy snadno přenositelná mezi platformami. Navíc popis grafického rozložení dokumentu ukládá do XML[11] souborů, tudíž zde java poskytuje nezávislý kód a XML nezávislá data.

5.3.1 Proces vytvoření výstupu

Vygenerování výstupu je proces sestávající ze čtyř kroků, který popisuje obrázek 10. Soubory `jrxml` jsou klasické XML, které popisují vzhled šablony, formát a umístění vkládaných dat a obsahují statické texty. Tyto soubory je možné editovat v programu iReport[12]. Soubory `jasper` představují serializované objekty vytvořené překladem z `jrxml`.

S programem je možno distribuovat oba typy souborů. V případě `jrxml` je výhodou, že je možné šablonu dynamicky měnit. Nevýhodou je, že uživatel má možnost do šablony zasahovat, což nemusí být vždy žádoucí. Navíc překlad trvá podle testů[13] v průměru 208 ms, takže v případě distribuce již přeložených souborů tento čas ušetříme.

[8] Množina dotazů do databáze, která se buď celá provede, nebo celá neprovede.

[9] http://jasperforge.org/projects/jasperreports

[10] Licence pro svobodný software, vyžadující, aby aplikace odvozené z licencovaného softwaru byly šířeny pod toutéž licencí.

[11] eXtensible Markup Language - jazyk popisující data nezávisle na platformě.

[12] http://jasperforge.org/projects/ireport

[13] Na sestavě Intel Core 2 Duo 2.53 GHz, 3GB DDR3 při deseti testech.

5.3.2 Vkládání dat reportu

Při návrhu reportu umísťujeme do šablony tři typy dat. Jedná se o `parametr`, `field` a `variable`. `Parametr` je jedna konkrétní hodnota, `field` je opakující se hodnota (například pro tabulky, seznamy atd.) a `variable` je proměnná jejíž hodnota se vypočte při samotném generování.

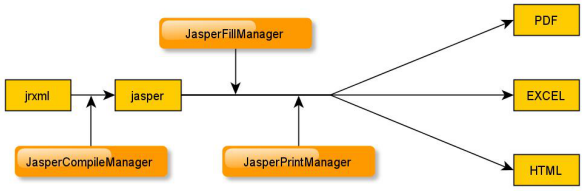
Parametry se do reportu vkládají přes kolekci `Map<String, Object>`. Pole se do reportu vloží přes objekt implementující rozhraní `JRDataSource`. Jedním z takových objektů je `JRBeanCollectionDataSource`, který obsahuje kolekci JavaBean objektů viz kód 6 nebo `JRMapCollectionDataSource`, který obsahuje kolekci map, viz kód 7.

Ukázka kódu 6: Vytvoření datového zdroje reportu – JRBeanCollectionDataSource.

```
Collection<Map<String, ?>> col = new Vector<Map<String, ?>>();
for (int i= 0; i< 5; i++) {
  Map<String, String> map = new HashMap<String, String>();

  map.put("technika", "60");
  map.put("regenerace", "40"));

  col.add(map);
}
JRDataSource jrds = new JRMapCollectionDataSource(col);
```

Ukázka kódu 7: Vytvoření datového zdroje reportu – JRMapCollectionDataSource.

```
List<Vysledek> vysledky = new ArrayList<Vysledky>();
vysledky.add(new Vysledek("technika", 60));
vysledky.add(new Vysledek("regenerace", 40));

JRDataSource jrds = new JRBeanCollectionDataSource(vysledky));
```



Obrázek 10: Schéma vytvoření dokumentu v JasperReports.

5.3.3 Vygenerování sestavy

Pomocí statických metod z třídy `JasperExportManager`je možné jednoduše vygenerovat sestavu. Pro vygenerování výstupu ve formátu PDF se použije metoda `exportReportToPdfFile()`.

5.4 JFreeChart

JFreeChart[14] je knihovna pro zobrazování grafů v Java aplikacích a jejich export. Její použití je zdarma a je šířena pod GNU Lesser General Public Licence[15] a celá je napsaná v Javě.

5.4.1 Vytvoření grafu

Proces vytvoření grafu má své obecné zákonitosti, které platí pro jakýkoli typ grafu. Základem každého grafu je „Dataset“, tedy objekt, který uchovává data pro vykreslení grafu. Existuje několik základních datasetů. Jako příklad bych uvedl `DefaultPieDataset` pro koláčové grafy nebo `DefaultCategoryDataset` například pro sloupcové skupinové grafy. V současné verzi (1.0.14) existuje 53 různých datasetů.

Grafy kreslené na komponenty v programu jsou objekty typu `JFreeChart`. O jejich vykreslení se starají statické metody třídy `ChartFactory`. Vykreslení jednoduchého koláčového grafu by provedla metoda `ChartFactory.createPieChart3D`.

5.5 Výběr osob k testování – D&D

Jak je popsáno v kapitole 4.1.2, při vytváření nového testu se výběr osob provádí přesunutím vybraných osob mezi dvěma panely tedy metodou Drag and Drop (D&D)[16]. Jelikož se nejedná o jednoduchý přenos řetězců, ale celých objektů (konkrétně kolekce objektů), musel jsem tento přenos doprogramovat.

Nejprve bylo nutné vytvořit třídu, která umožní samotný přenos dat, tedy jakýsi kontejner. Implementoval jsem tedy rozhraní `Transferable`. Nejdůležitější metodou je zde `getTranferData()`, která vrací přenášená data poté co je provedena akce drag. Zároveň jsou zde metody, které rozhodují jestli jsou přenášená data podporována, nebo vrací typ přenášených dat.

Dále jsem implementoval rozhraní `TransferHandler`, které již zajišťuje přenos dat mezi komponentami. Zde jsou nejdůležitějšími metodami `createTransferable()` a `importData()`.

První jmenovaná metoda zajistí, aby při začátku akce byla vybraná data zabalena do objektu pro přenos. Druhá metoda zajistí, aby po přesunu dat nad správnou komponentu došlo ke správnému vložení. Třída obsahuje i metodu pro zjištění kompatibility cílové komponenty s přenášenými daty `canImport(TransferSupport)`.

Ukázka kódu 8: Handler pro přenos dat mezi komponentami.

```
1 @Override
2 @SuppressWarnings({ "unchecked", "rawtypes" })
3 public boolean importData(TransferSupport support) {
4   boolean result = false;
5
6   if (!canImport(support)) {
7     return false;
```

[14] http://www.jfree.org/jfreechart/

[15] GNU LGPL je často používaná licence pro softwarové knihovny. Jednoduše říká, že dílo, které pouze linkuje program pod LGPL nemusí mít nutně (L)GPL licenci.

[16] Je technika přenášení textu, objektů, souborů z jedné komponenty do druhé, popřípadě i mezi programy pouhým přetažením myší.

```
8     }
9     //Získání cílové komponenty a její přetypování na JList
10    JList destination = (JList) support.getComponent();
11    //Získání přenášených dat
12    try{
13      Transferable listOsob = (TransferableListOsob) support
14        .getTransferable().getTransferData();
15      /*Vložení osob zabalených v objektu TransferableListOsob
16      do modelu JListu - cílové komponenty*/
17      for (Osoba osoba : ((TransferableListOsob) listOsob).osoby){
18       ((DefaultListModel)destination.getModel()).addElement(osoba
             );
19      }
20      //Přenos proběhl v pořádku
21      result = true;
22    }
23    return result;
24  }
```

5.6 Uložení nastavení databáze

Pro uložení nastavení připojení k databázi jsem musel využít externího souboru. Mezi uvažované možnosti patřilo uložení do klasického textového souboru nebo serializace objektu[17] s potřebnými řetězci.

Z důvodu snadného získání dat ze serializovaného objektu a horší možnosti editace class souboru než textového souboru „zvenčí“ jsem zvolil serializaci objektu. Velkou výhodou tohoto přístupu je to, že Java všechny potenciálně problémové operace, jako konverze čísel a podobně, řeší za programátora.

Vytvořil jsem jednoduchou třídu implementující rozhraní Serializable viz kód 9. Implementace tohoto rozhraní je základní podmínkou pro serializaci objektů. Navíc všechny atributy třídy musejí být serializovatelné (základní typy jsou všechny serializovatelné). Je možné atribut „vyřadit“ ze serializace, existuje proto modifikátor `transient`.

Ukázka kódu 9: Třída pro serializaci

```
1  @SuppressWarnings("serial")
2  public class DatabazeSerial implements Serializable{
3    String databaze;
4    String jmeno;
5    String heslo;
6    String adresa;
7  }
```

Při spuštění programu se otestuje existence souboru se serializovanými přístupovými daty k databázi. Pokud soubor neexistuje, zobrazí se okno pro zadání přístupových dat. Pokud již existuje, načtou se data způsobem, který ukazuje kód 10.

Ukázka kódu 10: Načtení

[17] Serializace objektů je postup, kdy se z instance objektu vytvoří proud bajtů, který se uloží na disk. Pak jej lze zpět deserializovat do javovského objektu.

```
1 ObjectInputStream inputStream = new ObjectInputStream(new
    FileInputStream("databaze"));
2 DatabazeSerial des = (DatabazeSerial)inputStream.readObject();
3
4 String userName = des.getJmeno();
5 String password = des.getHeslo();
6 String url = "jdbc:mysql://"+des.getAdresa()+"/"+des.getDatabaze
    ();
```

6 Testování aplikace

Testování aplikace pro mě byla naprostá novinka. Kromě základního „program dělá co má, pokud mu dodám správná data“ jsem žádné velké testování nikdy nedělal. Velmi cenné informace mi přinesla diplomová práce Anny Borovcové[Bor08], která se sice zabývá testováním webových aplikací, ale obsahuje informace, které jsou platné i pro testování klasického softwaru. Při výběru testů jsem se řídil obecnými postupy a radami právě z této knihy. Nakonec jsem vybral následující testovací mix.

6.1 Analýza a testování okrajových hodnot

Provedl jsem analýzu okrajových hodnot při zadávání testů a týdnů v deníku. V některých případech neexistovala přirozená hranice a musel jsem jí navrhnout tak, aby zároveň plnila restriktivní funkci a zároveň, aby nedošlo k situaci, že validní hodnota bude mimo meze intervalu. Příkladem může být počet tréninkových jednotek v tréninkovém deníku. Minimum je jistě nula, ale maximum se určuje těžko. Ze zkušenosti vím, že v praxi jsou nejběžnější dvoufázové tréninky, ale znám i případy čtyřfázových tréninků v určitém krátkém časovém období. Nastavil jsem proto maximum na 5.

Test jsem prováděl zadáváním číselných hodnot na hranicích intervalů platnosti a zároveň nečíselných hodnot. Zkoumal jsem, jak program reaguje na tyto vyjímečné stavy.

6.2 Test povinných polí

Tento test do jisté míry souvisí již s analýzou dat pro datový model databáze. Bylo nutné vyhodnotit, které hodnoty jsou ve formulářich povinné a které volitelné.

Pak jsem testoval, jestli program správně reaguje na nevyplněné povinné pole. Bral jsem v úvahu jak jedno nevyplněné pole, tak i více nebo všechna pole.

6.3 Testování ekvivalence

Další test zařazený do mixu byl test ekvivalence. Ten spočívá v tom, že se interval povolených hodnot vstupu rozdělí na podintervaly, pro které by měl být stejný výstup. Poté se testuje, zda opravdu pro všechny hodnoty v podintervalech systém generuje stejné výsledky.

U tohoto testu se projevilo to, že jsem zároveň autorem programu. V odborné terminologii bych pak řekl, že jsem testoval „white box“ [Bor08] neboli bílou skříňku[18]. Jelikož jsem při testování znal detailně implementaci jednotlivých funkcí a věděl jsem, že pokud by ekvivalence selhala, musela by být chyba v přepisu jednotlivých tabulek do polí Javy. Tento test jsem obešel a provedl vizuální kontrolu přepisu hodnot z vyhodnocovacích tabulek do kódu, místo zadávání hodnot do GUI a kontrolování výstupu.

6.4 Testování případů užití

Posledním z testů v mixu, který jsem prováděl, byl test případů užití. Použil jsem předem definované případy užití, popsané v kapitole 4.4. Jednoduše řečeno jsem zmapoval uživatelské akce, které je potřeba provést k vykonání jedné konkrétní akce a připravil tak jakési scénáře užití. Testováním je pak myšleno procházet těmito scénáři a zkoumat, zda se program chová podle předpokladů.

6.5 Akceptační test

Tento test je asi přirozenou částí vývoje každého softwarového produktu. Při převzetí zákazník kontroluje, zda produkt odpovídá všem, ve specifikaci [Spe12], domluveným bodům. Testovali jsme především generování výstupů a funkce grafického rozhraní. Test nepřinesl žádné nové chyby.

7 Nasazení a servis

Jedním z bodů zadání bylo i nasazení aplikace do reálného provozu. Rozhodl jsem se, že nasazení proběhne v Judoclubu Plzeň (dále JCP) a Středisku volného času v Plzni (dále SVČ). Z časového důvodu jsem se rozhodl pro nasazení pouze v Plzni, ale plzeňský judoclub patří mezi kluby s nejdelší historií a největší členskou základnou v republice, tudíž se řadí mezi nejvhodnější objekty pro nasazení.

Samotné nasazení proběhlo velmi hladce. V obou klubech jsem předvedl základní funkčnost a předal uživatelskou příručku. Uživatelé se snadno zorientovali a začali program používat. V případě JCP si zkusili dokonce 2 testy. Ve SVČ na to bohužel nebyl čas a tak jsme zkusili pouze tréninkový deník. V obou klubech byli se systémem spokojeni a v nové sezóně 2012/2013 program nasadí.

7.1 Problémy s databází

Ještě před nasazením jsem však narazil na dva, poměrně zásadní, problémy. Prvním z nich bylo umístění databáze. Původně jsem předpokládal, že databázi umístím na server, na kterém hostuje JCP své webové stránky. Na sereru služba MySQL bez prolémů běží a stránky jí využívají, tak jsem předpokládal, že to nebude problém. Databázi jsem na server importoval bez problémů, ale nemohl jsem se k ní připojit. Po dlouhém zkoumání jsem zjistil, že do databáze se lze přihlašovat pouze jako k místní databázi, tedy na adrese 127.0.0.1. Což znamená, že program by musel běžet na tom samém stroji, jako databáze.

[18] Někdy nazývaná také jako clear/transparent/glass box. Znamená, že tester zná implementační detaily.

Bohužel se nejedná pouze o tohoto jednoho konkrétního poskytovale hostingu, ale funguje to takto na všech mnou navštívených serverech pro hosting webu. Pravda, někde nabízejí povolení jedné vzdálené adresy. To by ale můj problém nevyřešilo. Jediná možnost by byla zaplatit za virtuální nebo dedikovaný server.

Problém jsem vyřešil tak, že jsem databázi nahrál na svůj vlastní domácí server s veřejnou IP adresou. V budoucnu však toto řešení nebude možné. Proto v současné době řeším možnost nasadit databázi na jiných strojích. V době odevzdání práce však databáze stále běží na mém domácím serveru.

Dalším problémem, avšak ne tak zásadním, bylo odlišné chování vzdálené a místní databáze. Problém se projevoval velmi zvláštním způsobem – nebylo možné provádět část dotazů. Nakonec jsem zjistil, že databáze má jiné implicitní nastavení ignorování velikosti písmen na platformě Windows a UNIX. Musel jsem tedy ještě přepsat dotazy do vhodné formy.

7.2 Způsob distribuce

Program je distribuován na datovém nosiči CD/DVD/Flash (dále jen nosič) jako instalační exe soubor pro systém windows nebo jako jar soubor a přídavné složky s knihovnami a dodatečnými soubory pro ostatní operační systémy. Celý proces instalace je detailně popsán v uživatelské příručce v příloze D.

Zároveň jsou na datovém nosiči údaje pro připojení do databáze, které musí uživatel zadat do programu v případě, že používá databázi od výrobce aplikace. Je jasné, že v tom případě je nutné před předáním programu vytvořit novou databázi v databázovém systému a informace o ní přidat na nosič.

Pokud chce uživatel využívat, z nějakého důvodu, vlastní databázi, má na nosiči k dispozici SQL kód pro vytvoření struktury databáze.

8 Závěr

V práci jsem se zabýval vývojem aplikace na vyhodnocování sportovní přípravy v judu. Program jsem vytvářel od specifikace zadnání přes analýzu, návrh datové vrstvy (databáze), implementaci, testování a nakonec jsem ho nasadil do reálného provozu, čímž jsem splnil všechny body zadání.

Důležitým výsledkem je kladné přijetí mezi uživateli a jeho plánované použití v sezóně 2012/2013. Nové řešení oproti starému výrazně zjednodušuje práci a přidává typové kontorly, které, narozdíl od původního systému, neumožňují vložit nekonzistentní data, což v minulosti vedlo k problémům při vytváření výstupů. Velmi úspěšné se zdá být automatické generování výsledných reportů a aplikace se jeví v reálných podmínkách rychlá a spolehlivá.

V budoucnu by šlo přidat možnost provázat program s databází Českého svazu juda, kde jsou indormace o všech jeho členech a tyto informace navázat na osoby v databázi. Zjednodušila by se tak například kontrola placení členských příspěvků klubu, zápis o získání rozhodcovských či trenérských licencí, získání výkonostního stupně či kontrola zaplacení licenční známky pro starty v soutěžích.

Práce mi přinesla mnohé poznatky o tvorbě grafických uživatelských rozhraní v Javě. Naučil jsem se používat frameworky JasperReports a JFreeChart, které jsou velmi zdařilé a v budoucnu použitelné. Byla to má první aplikace, která jako datovou

vrstvu používá databázi. V tomto ohledu jsem získal velmé množství cenných informací. Celkově si myslím, že mi práce rozšířila mnohé obzory v oblasti programování a celkově vývoje softwaru od zadání až po distribuci.

Použitá literatura

[But09] Alex. Butcher: *JUDO - průvodce bojovým uměním*. Ottovo nakladatelství s.r.o., Praha, 2009. ISBN 978-80-7360-579-7.

[Měk05] K. Měkota, J. Novosad: *Motorické schopnosti*. Univerzita Palackého v Olomouci, Olomouc, 2005. ISBN 80-244-0981-X.

[Her10] Pavel Herout: *XSLT 2.0 a SVG prakticky*. KOOP nakladatelství, České Budějovice, 2010. ISBN 978-80-7232-406-4

[Arl03] Jim Arlow, Ila Neustadt: *UML a unifikovaný vývoj aplikací*. Computer Press, Praha, 2003. ISBN 80-7226-947-X

[Opp06] Andrew J. Oppel: *Databáze bez předchozích znalostí*. Computer Press, Brno, 2006. ISBN 80-251-1199-7

[Tut06] Sharon Zakhour, Scott Hommel, Jacob Royal, Isaac Rabinovitch, Tom Risser, Mark Hoeber: *The Java Tutorial: a Short Course on the Basics, 4th Edition*. Prentice Hall, 2006. ISBN 978-0321334206

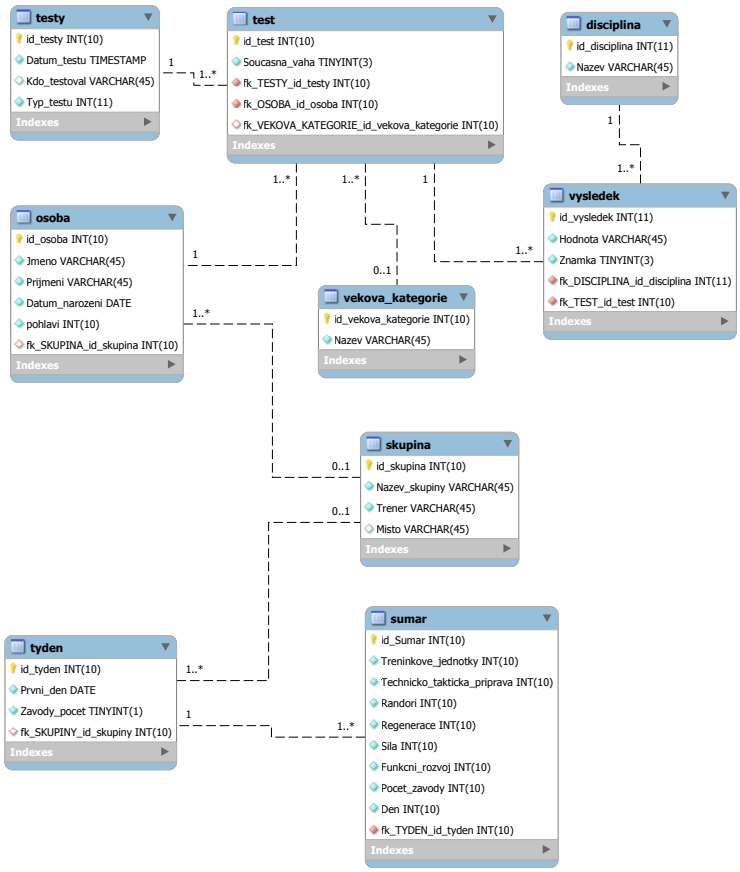
[Bor08] Anna Borovcová: *Testování webových aplikací* Online dostupné z: http://testovanisoftwaru.blogspot.cz/p/dptestovanisoftwarupdf.html

[Spe12] Ondřej Havlíček: *Dokument specifikace zadání*. Jako příloha na CD.

[Srj95] Josef Letošník: *Soutěžní řád juda*. Jako příloha na CD.

[Sim] Ed Simons: *iReport Ultimate Guide*. Online dostupné z: http://www.opus-college.net/devcorner/iReport-Ultimate-Guide-3.pdf

A ERA model


testy
id_testy INT(10)
Datum_testu TIMESTAMP
Kdo_testoval VARCHAR(45)
Typ_testu INT(11)
Indexes
test
id_test INT(10)
Soucasna_vaha TINYINT(3)
fk_TESTY_id_testy INT(10)
fk_OSOBA_id_osoba INT(10)
fk_VEKOVA_KATEGORIE_id_vekova_kategorie INT(10)
Indexes
disciplina
id_disciplina INT(11)
Nazev VARCHAR(45)
Indexes
vysledek
id_vysledek INT(11)
Hodnota VARCHAR(45)
Znamka TINYINT(3)
fk_DISCIPLINA_id_disciplina INT(11)
fk_TEST_id_test INT(10)
Indexes
osoba
id_osoba INT(10)
Jmeno VARCHAR(45)
Prijmeni VARCHAR(45)
Datum_narozeni DATE
pohlavi INT(10)
fk_SKUPINA_id_skupina INT(10)
Indexes
vekova_kategorie
id_vekova_kategorie INT(10)
Nazev VARCHAR(45)
Indexes
skupina
id_skupina INT(10)
Nazev_skupiny VARCHAR(45)
Trener VARCHAR(45)
Misto VARCHAR(45)
Indexes
tyden
id_tyden INT(10)
Prvni_den DATE
Zavody_pocet TINYINT(1)
fk_SKUPINY_id_skupiny INT(10)
Indexes
sumar
id_Sumar INT(10)
Treninkove_jednotky INT(10)
Technicko_takticka_priprava INT(10)
Randori INT(10)
Regenerace INT(10)
Sila INT(10)
Funkcni_rozvoj INT(10)
Pocet_zavody INT(10)
Den INT(10)
fk_TYDEN_id_tyden INT(10)
Indexes

B Class Diagram

pkg src

HlavniOkno
(from Prezentacni)
- serialVersionUID:long=1L
- contentPane:JPanel
- hlavniOkno:JFrame
- main(in args:String[*]):void
- «constructor» HlavniOkno()

VyhodnoceniTestu
(from Datova::Testy)
- vahovaKategorie(in typTestu:int, in osoba:Osoba):int
- vyhodnotTest(in test:MotorickyTest):int

«annotations»
ZobrazitCekaniPruhledne
(from Prezentacni)
- «annotations» paint(in g:Graphics):void
- run():void

MotorickyTest
(from Datova::Testy)
- datum:String
- id_testy:int
- kdo_testoval:String
- typTestu:int
- vysledky:ArrayList<T1->VysledkyTestu>=new
- getVysledky():ArrayList<T1->VysledkyTestu>
- «constructor» MotorickyTest(in id_testy:int, ir
- «annotations» toString():String
- «constructor» MotorickyTest()
- nactiDataZDB():void
- pocetVysledku():int
- getDatum():String
- setDatum(in datum:String):void
- getID_test():int
- setID_test(in ID_test:int):void
- getKdo_testoval():String
- setKdo_testoval(in kdo_testoval:String):void
- getTypTestu():int
- setTypTestu(in typTestu:int):void

«annotations»
NovyMotorickyTest
(from Prezentacni::Testy)
- test:MotorickyTest
- discipliny:List<T1->Integer>
- tabulka:TabulkaTestu
- panelTlacitek:JPanel
- panelTabulky:JPanel
- panelHlavni:JPanel
- «constructor» NovyMotorickyTes
- vytvoritPanelTlacitek():JPanel
- vytvoritPanelTabulky():JPanel
- vytvoritPanelHlavni():JPanel
- TabulkaTestu
- EditorClunk
- EditorDalka
- EditorDesetinne
- EditorCele
- TabulkaTestu_MODEL

VysledkyTestu
(from Datova::Testy)
- id_testu:int
- osoba:Osoba=new Osoba()
- mapaVysledku:Map<T1->Integer,T2->Vysledek>=new HashMap<Integer, Vysledek>()
- «annotations» toString():String
- getId_testu():int
- setId_testu(in id_testu:int):void
- getOsoba():Osoba
- setOsoba(in osoba:Osoba):void
- getMapaVysledku():Map<T1->Integer,T2->Vysledek>
- setMapaVysledku(in mapaVysledku:Map<T1->Integer,T2->Vysledek>):void

PridaniTydne
(from Prezentacni::Denik)
- serialVersionUID:long=1L
- contentPane:JPanel
- datePicker:JXDatePicker
- comboBox:JComboBox<T1->Skupina>=null
- boxModel:DefaultComboBoxModel<T1->Skup
- txtTurnaje:JTextField
- tyden:DenikTyden=null
- tabulkaTyden:TabulkaTydne
- vytvorenyTyden:boolean=false
- main(in args:String[*]):void
- «constructor» PridaniTydne()
- vytvoritPanelHlavni():JPanel
- vytvoritLevyPanel():JPanel
- «annotations» run():void
- TabulkaTydne
- «annotations» Editor
- TabulkaTydne_MODEL

DenikTyden
(from Datova)
- «constructor» DenikTyden()
- «constructor» DenikTyden(in
- getSkupinaID():int
- setSkupinaID(in skupinaID:in
- getTrener():String
- setTrener(in trener:String):voi
- toString():String
- getDatum():String
- setDatum(in datum:String):vo
- getSkupina():String
- setSkupina(in skupina:String
- getId_tyden():int
- setId_tyden(in id_tyden:int):v
- getZavody():int
- setZavody(in zavody:int):void
- getDny():Suma[*]
- setDny(in dny:Suma[*]):void

«annotations»
TreninkovyDenik
(from Prezentacni::Denik)
- infoSkupina:JTextArea
- vyberSkupiny:JComboBox<T1->Skupina>
- listTydnu:JList<T1->DenikTydenZjednoduseny>
- tabulka:TabulkaTydne
- panelGrafy:JPanel
- panelGrafKolac:ChartPanel=new ChartPanel(null)
- panelGrafSloupce:ChartPanel=new ChartPanel(null)
- tyden:DenikTyden=null
- panelTlacitek:JPanel
- panelHlavni:JPanel
- panelInformacni:JPanel
- listTydnu1:List<T1->DenikTydenZjednoduseny>
- «annotations» run():void
- «constructor» TreninkovyDenik()
- vytvoritPanelLevy():JPanel
- vytvoritPanelTabuly():JPanel
- vytvoritPanelGrafy():JPanel
- vytvoritPanelInformacni():JPanel
- vytvoritPanelHlavni():JPanel
- vytvoritPanelTlacitek():JPanel

DenikTydenZjednoduseny
(from Datova)
- datum:String
- skupina:int
- id_tyden:int
- «constructor» DenikTydenZjednoduseny(in id_tyden:int, in datum:String, in s
- «annotations» toString():String
- getDatum():String
- setDatum(in datum:String):void
- getSkupina():int
- setSkupina(in skupina:int):void
- getId_tyden():int
- setId_tyden(in id_tyden:int):void

«annotations»
MotorickeTesty
(from Prezentacni::Testy)
- infoSkupina:JTextArea
- listTydnu:JList<T1->MotorickyTest>=new JList<MotorickyTest>()
- tabulka:TabulkaTestu
- tyden:DenikTyden=null
- txtPrumerSkupiny:JTextField
- txtPocetTestovanych:JTextField
- panelGrafPrumer:ChartPanel=new ChartPanel(null)
- panelGrafHistogram:ChartPanel=new ChartPanel(null)
- «constructor» MotorickeTesty()
- jmenoSkupiny(in cisloSkupiny:int):String
- MotorickyTestRenderer
- TabulkaTestu_MODEL
- TabulkaTestu
- EditorClunk
- EditorDalka
- EditorDesetinne
- EditorCele

«annotations»
PridaniTestu
(from Prezentacni::Testy)
- listVybranychOsob:JList<T1->Osoba>
- listOsob:JList<T1->Osoba>
- listVybranychOsobModel:DefaultListModel<T1->Osoba>
- comboBox:JComboBox<T1->Skupina>
- contentPane:JPanel
- panelOsob:JPanel
- panelHlavni:JPanel
- panelTypTestu:JPanel
- panelDiscipliny:JPanel
- btnGroup:ButtonGroup
- rdbtnTestJunior:JRadioButton
- rdbtnTestDorost:JRadioButton
- rdbtnTestIk:JRadioButton
- chckbxBenchpress:JCheckBox
- chckbx1500m:JCheckBox
- chckbxClunkovyBeh:JCheckBox
- chckbxPremisteni:JCheckBox
- chckbxShyb:JCheckBox
- chckbxVznosy:JCheckBox
- chckbxVyskoky:JCheckBox
- chckbxSplh:JCheckBox
- panelTlacitka:JPanel=new JPanel()
- panelUmenoTrenera:JPanel
- lblTrener:JLabel
- txtTrener:JTextField
- typTestu:int
- main(in main:String[*]):void
- «annotations» run():void
- vytvoritLevyPanel():JPanel
- vytvoritListVybranychOsob():JComponent
- vytvoritListOsob():JComponent
- vytvoritHlavniPanel():JPanel
- vytvratVsechnyDiscipliny():void
- vytvoritPanelTypTestu():JPanel
- vytvorPanelDiscipliny():JPanel
- disciplinyList():List<T1->Integer>
- testovatelne():boolean
- vytvoritPanelTlacitek():JPanel
- OsobaTransferHandler
- OsobaRenderer

«annotations»
Jednotlivci
(from Prezentacni::Testy)
- vyberSkupiny:JComboBox<T1->Skupina>
- list:JList<T1->Osoba>
- tabulka:TabulkaTestu
- panelGrafVyvoj:ChartPanel=new ChartPanel(null)
- panelGraf:JPanel
- lblPopis:JLabel
- btnReport:JButton
- «constructor» Jednotlivci()
- vytvoritPanelVyberOsoby():JPanel
- vytvoritLabelPopisu():JLabel
- vytvoritPanelTlacitek():JPanel
- vytvoritPanelTabulky():JPanel
- OsobaRenderer
- TabulkaTestu
- TabulkaTestu_MODEL

Report_TestSkupina
(from Prezentacni::Reporty)
- «constructor» Report_TestSkupina(in test:MotorickyTest, in grafZnamky:Image, in grafHistogram:Image)

Report_TestJednotlivec
(from Prezentacni::Reporty)
- «constructor» Report_TestJednotlivec(in listTestu

MotorickeTestyGrafy
(from Prezentacni::Testy)
- grafPrumerDisciplin_bar(in test:MotorickyTest):JFreeChart
- grafHistogram_bar(in test:MotorickyTest):JFreeChart
- grafVyvojJednotliveVCase(in listTestu:List<T1->MotorickyTest>):JFreeChart

«annotations»
SpravaUzivatelu
(from Prezentacni::SpravaOsob)
- VELIKOST_TXT_POLE:int=20
- table:TabulkaOsob=new TabulkaOsob()
- txtJmeno:JTextField
- txtPrijmeni:JTextField
- txtSkupina:JTextField
- panelHorni:JPanel
- panelVyhledavani:JPanel
- panelTlacitek:JPanel
- panelSpodni:JPanel
- «constructor» SpravaUzivatelu()
- vytvorPanelHorni():JPanel
- vytvorVyhledavaniTextu():void
- vytvoriPanelVyhledavani():JPanel
- vytvorPanelSpodni():JPanel
- vytvorPanelTlacitek():JPanel
- vytvorPanelTabulky():Component
- PredavaneParametryUpravaZaznamu

«annotations»
TabulkaOsob
(from Prezentacni::SpravaOsob)
- sorter:TableRowSorter<T1->TabulkaO
- «constructor» TabulkaOsob()
- newFilter(in regexVzor:String, in slou
- VypisZahlavi
- TabulkaOsob_datumRenderer
- TabulkaOsob_JmenoPrijmeniRender
- TabulkaOsob_SkupinaRenderer

Vysledek
(from Datova::Testy)
- doszamyVysledek:String
- znamka:int
- disciplina:int
- idVysledek:int
- «constructor» Vysledek(in idVysledek:int, in disciplina:int
- «constructor» Vysledek()
- «annotations» toString():String
- getIdVysledek():int
- setIdVysledek(in idVysledek:int):void
- getDoszazenyVysledek():String
- setDoszazenyVysledek(in doszazenyVysledek:String):void
- getZnamka():int
- setZnamka(in znamka:int):void
- vyhodnot():void
- getDisciplina():int
- setDisciplina(in disciplina:int):void

Osoba
(from Datova)
- idOsoba:int
- jmeno:String
- prijmeni:String
- vaha:int
- pohlavi:int
- datumNarozeni:String
- skupina:String
- skupinaInt:int
- «constructor» Osoba()
- «constructor» Osoba(in idOsoba:int, in jmeno:S
- «constructor» Osoba(in idOsoba:int, in jmeno:S
- getIdOsoba():int
- getSkupinaInt():int
- setSkupinaInt(in skupinaInt:int):void
- setIdOsoba(in idOsoba:int):void
- getJmeno():String
- setJmeno(in jmeno:String):void
- getPrijmeni():String
- setPrijmeni(in prijmeni:String):void
- getVaha():int
- setVaha(in vaha:int):void
- getPohlavi():int
- setPohlavi(in pohlavi:int):void
- getDatumNarozeni():String
- setDatumNarozeni(in datumNarozeni:String):voi
- getSkupina():String
- setSkupina(in skupina:String):void
- «annotations» toString():String
- «annotations» hashCode():int
- «annotations» equals(in obj:Object):boolean

Sumar
(from Datova)
- id_sumar:int
- treninkoveJednotky:int
- teta:int
- rand:int
- rege:int
- sila:int
- funk:int
- den:int
- PONDELI:int=0
- UTERY:int=1
- STREDA:int=2
- CTVRTEK:int=3
- PATEK:int=4
- «constructor» Sumar()
- «constructor» Sumar(in id_sumar:int, in trenin
- getId_sumar():int
- setId_sumar(in id_sumar:int):void
- getTreninkoveJednotky():int
- setTreninkoveJednotky(in treninkoveJednotky:
- getTeta():int
- setTeta(in teta:int):void
- getRand():int
- setRand(in rand:int):void
- getRege():int
- setRege(in rege:int):void
- getSila():int
- setSila(in sila:int):void
- getFunk():int
- setFunk(in funk:int):void
- getDen():int
- setDen(in den:int):void

Report_DenikTyden
(from Prezentacni::Reporty)
- «constructor» Report_DenikTyden(in tyden:DenikTy

TreninkovyDenikGrafy
(from Prezentacni::Denik)
- grafSkupinyTydnu_kolac(in vybraneTydny:List<T1->DenikTyden>):JFreeCh
- grafSkupinyTydnu_bar(in vybraneTydny:List<T1->DenikTyden>):JFreeChart
- grafTydne_bar(in tyden:DenikTyden):JFreeChart
- grafTydne_kolac(in tyden:DenikTyden):JFreeChart

«annotations»
TreninkovyDenikSumar
(from Prezentacni::Denik)
- vyberSkupiny:JComboBox<T1->Skupina>
- listTydnu:JList<T1->DenikTydenZjednoduseny>
- tabulka:TabulkaTydne
- panelGrafy:JPanel
- panelGrafKolac:ChartPanel=new ChartPanel(null)
- panelGrafSloupce:ChartPanel=new ChartPanel(null)
- tyden:DenikTyden=null
- panelTlacitek:JPanel
- panelHlavni:JPanel
- panelInformacni:JPanel
- listVybraneTydnyUpIne:List<T1->DenikTyden>=new LinkedList<DenikTyden>()
- «annotations» run():void
- «constructor» TreninkovyDenikSumar()
- vytvoritPanelLevy():JPanel
- vytvoritPanelTabuly():JPanel
- vytvoritPanelGrafy():JPanel
- vytvoritPanelInformacni():JPanel
- vytvoritPanelHlavni():JPanel
- vytvoritPanelTlacitek():JPanel
- TabulkaTydne
- TabulkaTydne_MODEL

Worker_SpravaUzivatelu
(from Prezentacni)
- tabulka:JTable
- tabulky:TabulkaOsob
- «constructor» Worker_SpravaUzivatelu(in table:JTable)
- «annotations» doInBackground():List<T1->Osoba>
- «annotations» done():void

«annotations»
TransferableOsoba
(from Prezentacni::Testy::PridaniTestu::OsobaTransferHandler)
- dtf:DataFlavor=new DataFlavor(TransferableOsoba.class, "Ob
- «constructor» TransferableOsoba(in o:Osoba)
- «annotations» getTransferDataFlavors():DataFlavor[*]
- «annotations» isDataFlavorSupported(in flavor:DataFlavor):boc
- «annotations» getTransferData(in flavor:DataFlavor):Object

TabulkaOsob_MODEL
(from Prezentacni::SpravaOsob)
- setNewData(in data:List<T1->Osoba>):void
- «annotations» getRowCount():int
- «annotations» getColumnCount():int
- «annotations» getValueAt(in rowIndex:int, in column
- vratVybranouOsobu(in row:int):Osoba
- getColumnName(in column:int):String
- isCellEditable(in row:int, in column:int):boolean
- getColumnClass(in column:int):Class<T1->?>

«annotations»
modifiers = Deprecated
- **getTransferDataFlavors() (Operation)**
- «annotations» modifiers = Override
- **isDataFlavorSupported() (Operation)**
- «annotations» modifiers = Override
- **getTransferData() (Operation)**
- «annotations» modifiers = Override

«annotations»
SpravaUzivatelu_pridejZaznam
(from Prezentacni::SpravaOsob)
- txtJmeno:JTextField
- contentPane:JPanel
- txtPrijmeni:JTextField
- buttonGroup:ButtonGroup=new ButtonGroup()
- datePicker:JXDatePicker
- rdbtnMuz:JRadioButton
- rdbtZena:JRadioButton
- format:SimpleDateFormat=new SimpleDateFormat("YYYY-MM-dd")
- comboBox:JComboBox<T1->Skupina>
- Property1:SpravaUzivatelu
- «constructor» SpravaUzivatelu_pridejZaznam(in table:JTable)

«annotations»
SpravaUzivatelu_zmenZaznam
(from Prezentacni::SpravaOsob)
- contentPane:JPanel
- txtJmeno:JTextField
- txtPrijmeni:JTextField
- datePicker:JXDatePicker
- comboBox:JComboBox<T1->Skupina>
- format:SimpleDateFormat=new SimpleDateFormat("YYYY-MM-dd")
- «constructor» SpravaUzivatelu_zmenZaznam(in parametry:PredavaneParametryUpravaZaznamu, in tabulka:JXTable)
- vratIndexSkupiny(in jmenoSkupiny:String, in skupiny:Vector<T1->Skupina>):int

«annotations»
SpravaSkupin
(from Prezentacni::SpravaSkupin)
- listOsob:JList<T1->Osoba>
- infoSkupina:JTextArea
- ListOsob:List<T1->Osoba>
- «constructor» SpravaSkupin()
- OsobaRenderer
- MiKeyEventPostProcessor

«annotations»
SpravaSkupin_pridejZaznam
(from Prezentacni::SpravaSkupin)
- contentPane:JPanel
- txtNazev:JTextField
- txtTrener:JTextField
- txtMisto:JTextField
- «constructor» SpravaSkupin_pridejZaznam(in list:JList<T1->Skupina>)

Skupina
(from Datova)
- jmeno:String
- ID:int
- trener:String
- misto:String
- getMisto():String
- setMisto(in misto:String):void
- «constructor» Skupina(in jmeno:String, in iD:int, in trener:String)
- «constructor» Skupina(in jmeno:String, in trener:String, in misto:String)
- «annotations» toString():String
- getJmeno():String
- setJmeno(in jmeno:String):void
- getID():int
- setID(in iD:int):void
- getTrener():String
- setTrener(in trener:String):void

Databaze
(from Datova)
- pripojDatabazi():Connection
- updateTestyHodnota(in id_vysledku:int, in hodnota:String, in znamka:int):void
- ulozTestyHodnota(in hodnota:String, in znamka:int, in disciplna:int, in id_test:int):void
- updateSumarHodnota(in id_tydne:int, in ukazatel:int, in den:int, in value:String):void
- nactiTestyJednotlivce(in id_osoba:int):List<T1->MotorickyTest>
- kontrolaDeniku_jeVolneDatum(in prvniDen:String, in id_skupiny:int):boolean
- ulozTest(in test:MotorickyTest):void
- ulozTyden(in tyden:DenikTyden):void
- ulozOsobu(in osoba:Osoba):void
- ulozSkupinu(in skupina:Skupina):void
- nactiOsobyList():List<T1->Osoba>
- nactiOsobu(in idOsoba:int):Osoba
- nactiOsobySkupinyVector(in IDSkupiny:int):Vector<T1->Osoba>
- nactiSkupinyVector():Vector<T1->Skupina>
- zmenZaznamOsoby(in osoba:Osoba):void
- nactyTydnyVector(in IDSkupina:int):Vector<T1->DenikTydenZjednoduseny>
- nactiTyden(in ID_tyden:int):DenikTyden
- nactiTestyVector():Vector<T1->MotorickyTest>
- nactiTest(in id_testu:int):MotorickyTest
- smazatOsobu(in id_osoba:int):void
- smazatSkupinu(in id_skupiny:int):void

«use»

Generated by UModel www.altova.com

C Příklady reportů

Vyhodnocení motorického testu

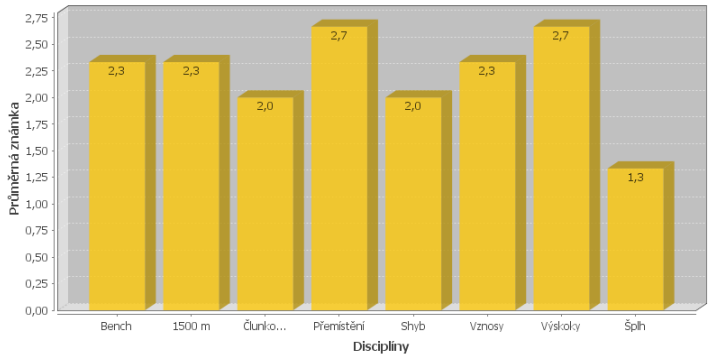
Vygenerováno v programu: JudoTesty 1.0
Pátek 22 červen 2012 03-22-27
Autor: Ondřej Havlíček

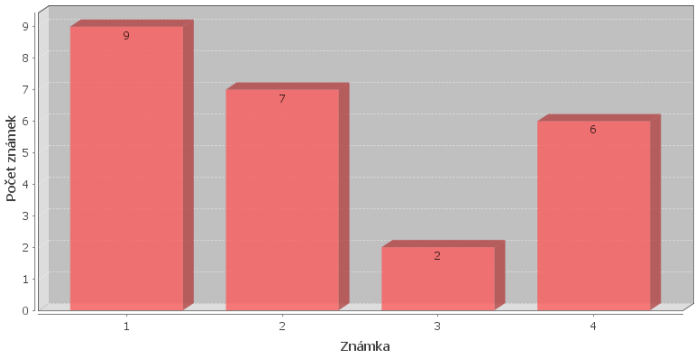
Trenér: Rubáš

Datum: 01.06.2011

Typ testu: Junior+

Jméno	Příjmení	Váha	Bench		Přemístění		1500m		Člunkový běh		Shyb		Vznosy		Výskoky		Šplh		Průměr
Ondřej	Havlíček	115	112.5	4	115	3	6:18	2	10:15	1	40	1	21	1	29	1	3.2	1	1.75
Ondra	Chlup	81	100	1	110	1	5:57	4	10:01	2	50	2	9	4	33	2	2	4	2.5
Honza	Moravec	66	85	2	87	2	5:32	2	9:57	1	42	4	25	1	26	4	3	3	2.375





Vyhodnocení tréninkového týdne

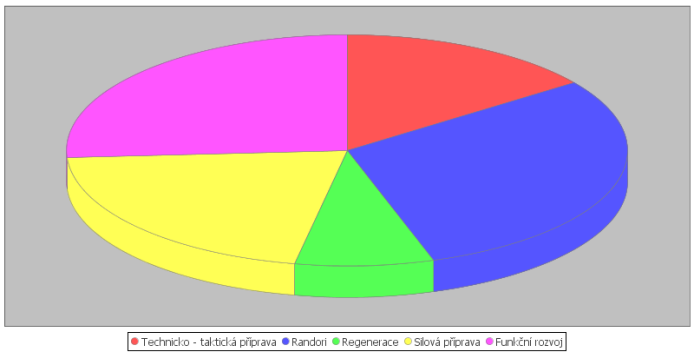
Vygenerováno v programu: JudoTesty 1.0
Pátek 22 červen 2012 02-58-51
Autor: Ondřej Havlíček

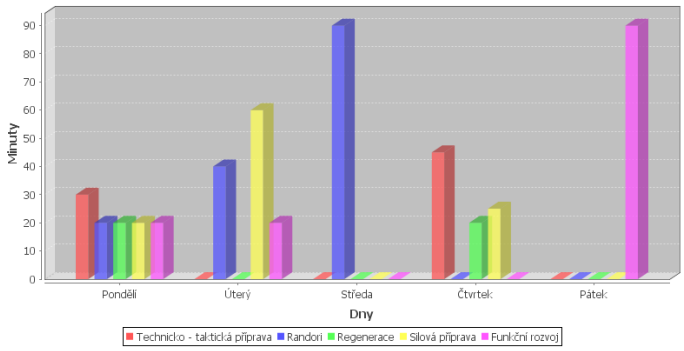
Skupina: Sportovní gymnázium

Trenér: Dolejš

Týden: 2012-05-28

Den	Technicko taktická příprava	Randori	Regenerace	Síla	Funkční rozvoj
Pondělí	30	20	20	20	20
Úterý	0	40	0	60	20
Středa	0	90	0	0	0
Čtvrtek	45	0	20	25	0
Pátek	0	0	0	0	90





D Uživatelská příručka

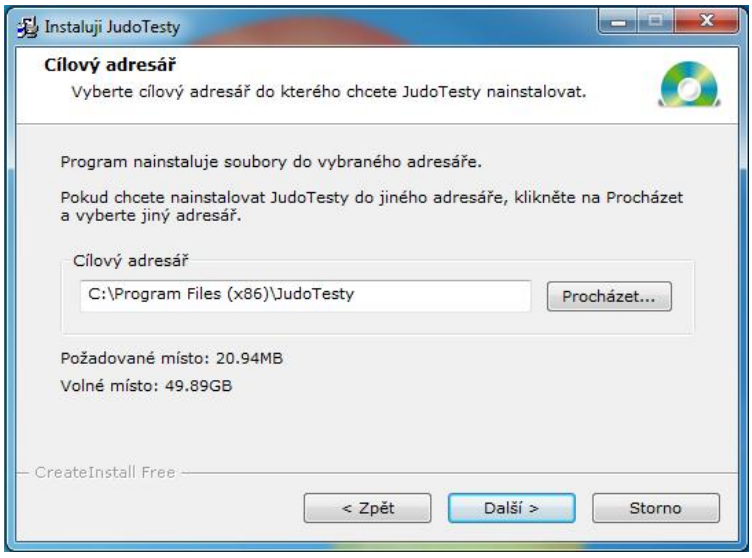
1 Instalace

Před instalací je potřeba zajistit, aby byla na Vašem počítači nainstalována Java. Ta je k dispozici na webu http://java.com/en/download/.

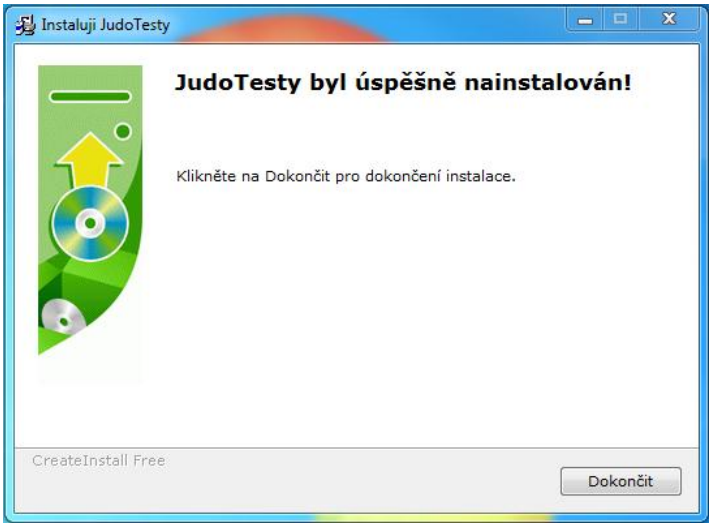
1.1 Windows

Pro systém Windows je k dispozici automatický instalátor. Ten Vás provede celou instalací. Instalace se provádí poklepáním na soubor *setup.exe*. Po jeho spuštění se otevře následující okno.

Zde stiskněte tlačítko ***Další***. V další fázi vyberete umístění, kam bude program nainstalován. Pokud nemáte důvod, neměňte přednastavené umístění a pouze stiskněte ***Další.***



Instalační program pak provede všechny potřebné kroky pro úspěšné nainstalování programu na Váš počítač. Nakonec stiskněte tlačítko ***Dokončit***.



Instalační program zárovneň s instalací vytvoří na Vaší ploše odkaz, kterým jednoduše aplikaci spustíte.

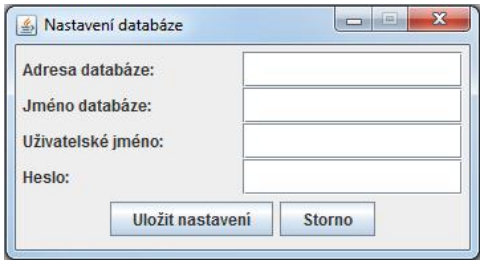
1.2 Ostatní operační systémy

Instalace na ostatních operačních je mírně komplikovanější. Je potřeba ze složky OthersOS zkopírovat **všechny** soubory a složky a uložit je do složky, kde se standartně nacházejí Vaše spustitelné soubory. Pak se program bude spouštět poklepáním na soubor Aplikace.jar. Je vhodné vytvořit k tomuto souboru zástupce/link a program spouštět přes něj.

Zároveň jsou s programem šířeny i zdrojové kódy a sestavovací skript pro Ant[1]. Pomocí něho je možné znovu vygenerovat spustitelný jar soubor.

2 První kroky

Ještě před prvním spuštěním programu si připravte Vaše údaje k připojení do databáze. Tyto údaje jste dostali spolu s programem na CD nebo flash disku. Při prvním spuštění programu se objeví formulář pro jejich zadání.

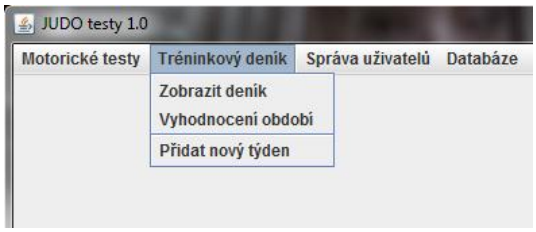


Vyplňte potřebné údaje a stiskněte tlačítko ***Uložit nastavení***. Pokud byste data zadali nepřesně nebude možné se připojit k databázi a získat potřebná data. Tudíž nebude většina funkcí dostupných. V takovém případě se pokuste údaje zadat znovu viz kapitola 7 .

1 Ant je automatizovaný sestavovací program pro vytváření a distribuci softwarových balíků a je dostupný z adresy http://ant.apache.org/.

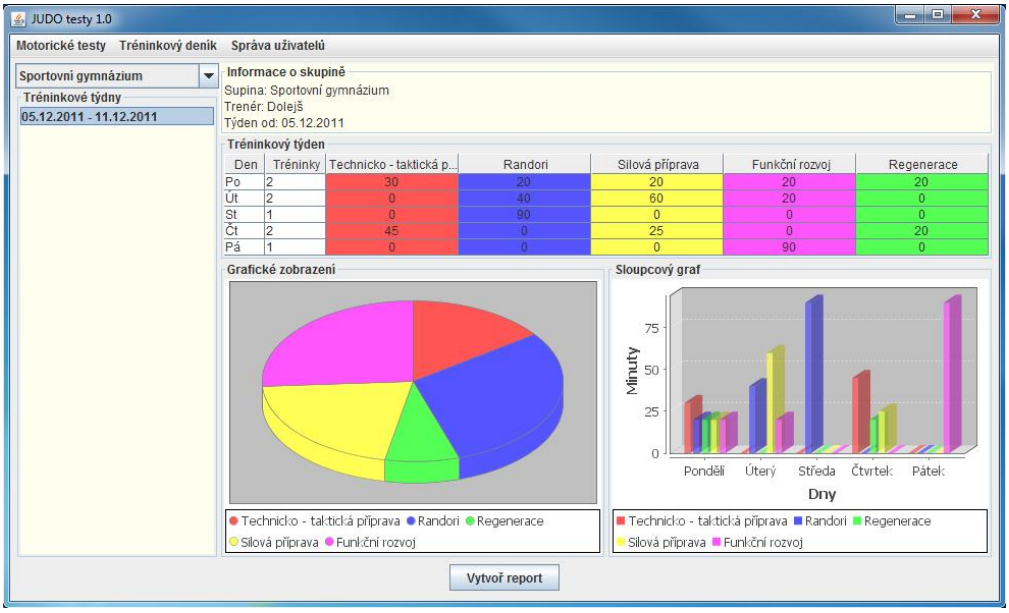
3 Tréninkový deník

Funkce tréninkového deníku jsou přístupné z položky ***Tréninkový deník*** v horním menu. Můžete zde vybrat možnost pouhého zobrazení deníku, vyhodnocení bloku týdnů nebo zadání nového týdne.



3.1 Zobrazení, úprava a generování informací z deníku

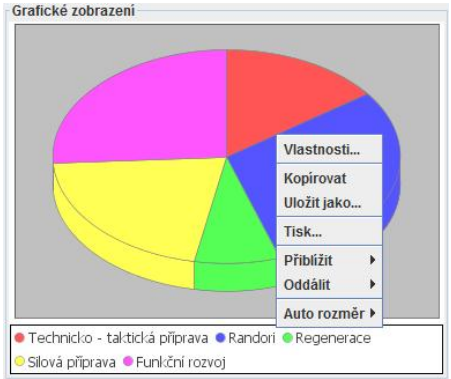
Po vybrání možnosti **Zobrazit deník** se zobrazí základní okno pro práci s deníkem.



V levé části okna je panel pro výběr skupiny a týdne. Po výběru skupiny v horní části panelu a vybrání libovolného týdne z tohoto panelu se automaticky zobrazí údaje o tomto týdnu včetně grafů a základních informací.

V tomto okně je zároveň možné měnit dříve vytvořená a uložená data. Jednoduše poklepejte na buňku v tabulce, kterou je nutno změnit. Tím se buňka zpřístupní pro editaci. V případě, že zadáte správnou hodnotu (tedy čas v minutách v rozmezí 0-120 minut) a stisknete Enter, dojde ke změně hodnoty. Zároveň se patřičně změní i grafy ve spodní části okna.

Z tohoto okna je možno uložit obrázek z libovolného grafu do vlastního souboru. Stačí pouze kliknout pravým tlačítkem myši na obrázek grafu a zvolit ***Uložit jako...*** Po vybrání umístění výsledného souboru se obrázek uloží. Obrázek se ukládá v proporcích a velikosti v jaké je aktuálně zobrazen v okně.



Posledním využitím je generování výstupu do formátu pdf. Generování se provádí stisknutím tlačítka ***Vytvořit report*** a výběrem umístění generovaného souboru. Při výběru souboru zároveň zvolíte jméno vygenerovaného souboru. Jméno může i nemusí obsahovat koncovku pdf. V případě, že jí nezadáte, bude automaticky doplněná. V případě, že koncovku správně zadáte, další se nepřidá.

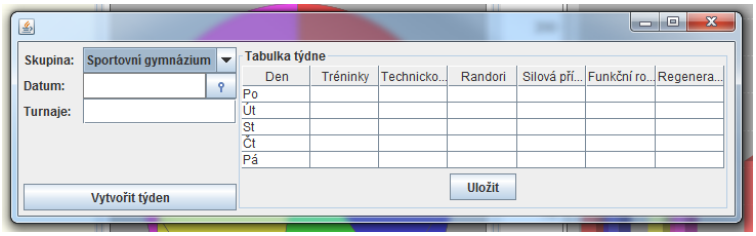
3.2 Vyhodnocení bloku týdnů

V případě, že potřebujete vyhodnotit blok (skupinu) týdnů, je potřeba vybrat položku ***Vyhodnocení období***. Rozložení okna je velice podobné tomu předchozímu, avšak přibylo zde tlačítko ***Vyhodnotit výběr týdnů***.

Nejprve je potřeba provést výběr týdnů k vyhodnocení. Stiskněte tedy na klávesnici tlačítko *CTRL* a postupně klikejte myší na požadované týdny. Lze vybrat i všechny klávesovou zkratkou *CTRL+A*. Po vybrání týdnů stiskněte tlačítko ***Vyhodnotit výběr týdnů***. Opět se v hlavním okně zobrazí tabulka s vyhodnocením vybraných týdnů a grafy. Obrázky grafů lze taktéž uložit do souboru postupem jako v kapitole 3.1.

3.3 Přidání nového týdne

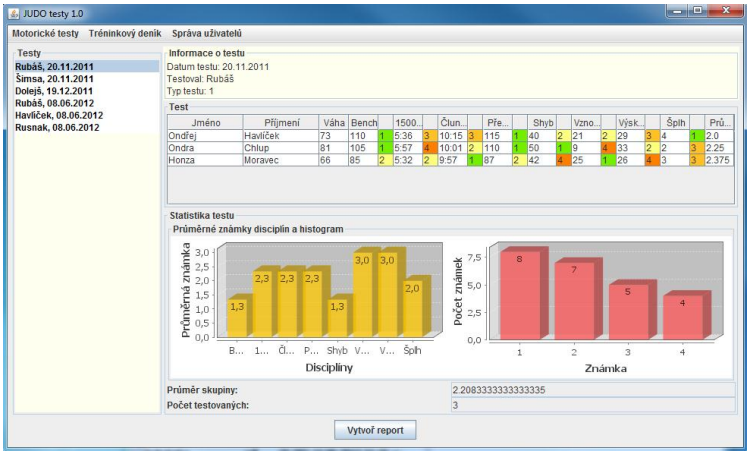
Nový týden se přidá v dialogu vyvolaném stisknutím ***Přidat nový týden*** v menu Tréninkový deník. Otevře se následující okno:



Je potřeba vybrat skupinu, které se týká týden, dále datum týdne – což je datum které připadá na pondělí tohoto týdne a počet turnajů, které se daný týden konali. Poté je třeba stisknout tlačítko ***Vytvořit týden***. Tím se zpřístupní tabulka pro zadání údajů týdne. Není potřeba zadat najednou všechny údaje. Tabulku je možné doplnit i později – viz kapitola 3.1. Po vyplnění dat je, pro jejich uložení, ještě potřeba stisknout tlačítko ***Uložit***.

4 Motorické testy

Po vybrání možnosti ***Zobrazit testy*** z menu Motorické testy dojde ke zobrazení následujícího okna:



4.1 Zobrazení, úprava a generování výstupů z testů

V levé části okna je opět panel pro výběr zobrazovaného testu. Po kliknutí na libovolný z týdnů se okamžitě výběr projeví zobrazením informací o testu, tabulky s dosaženými hodnotami a grafy.

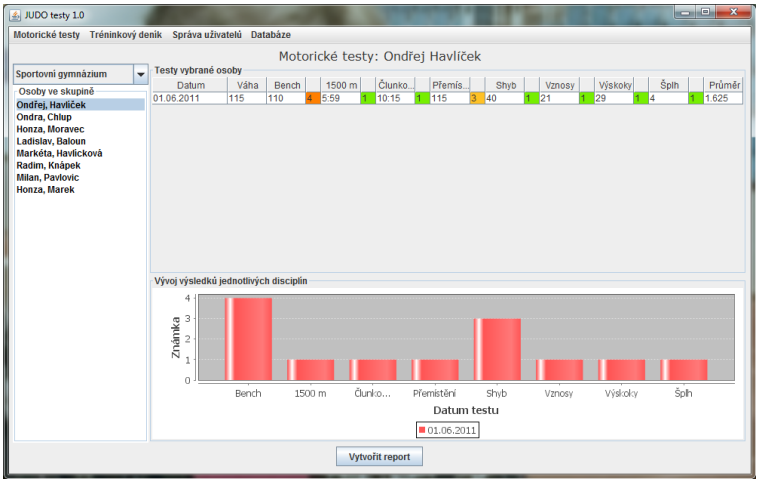
I zde je možné dříve vyvořené testy upravit jednoduchým poklikáním do buňky s dosaženým výsledkem a tento změnit. Změna se okamžitě projeví jak jejím vyhodnocením a určením známky, tak i překreslením grafů. Známka je zde počítána podle tabulek a není ji možné ručně změnit.

Grafy lze opět uložit jako obrázek kliknutím do jednoho z grafů pravým tlačítkem myši a výběrem tlačítka ***Uložit jako***...

Stejně jako v případě deníku lze generovat výstupy do formátu pdf stisknutím tlačítka ***Vytvořit report***. Všechno probíhá stejně jako v případě tréninkového deníku.

4.2 Výsledky v testech jednotlivců a generování výstupů

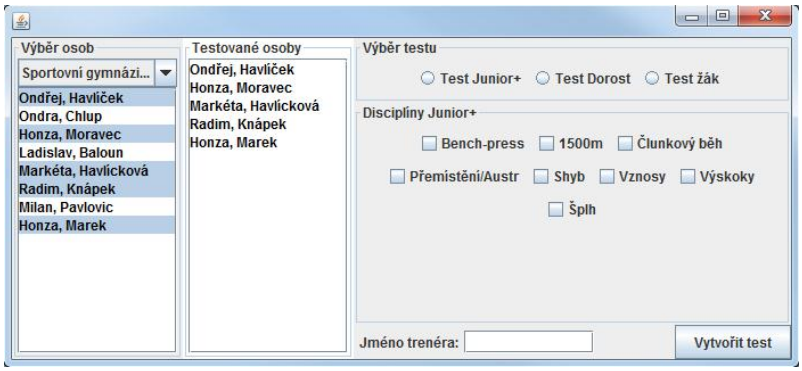
Kromě výsledků v jednotlivých dnech pro všechny osoby v testu je možné zobrazit výsledky ve všech testech pro konkrétního jedince.



Pro zobrazení těchto testů je potřeba vybrat skupinu, do které jedinec patří a pak v listu kliknout na něho samotného. Opět se okamžitě zobrazí tabulka všech testů, které daný jedinec absolvoval a graf jeho vývoje v čase. Zde není možné upravovat jednotlivé výsledky. Ale lze samozřejmě vygenerovat report o výsledcích jednotlivce do pdf.

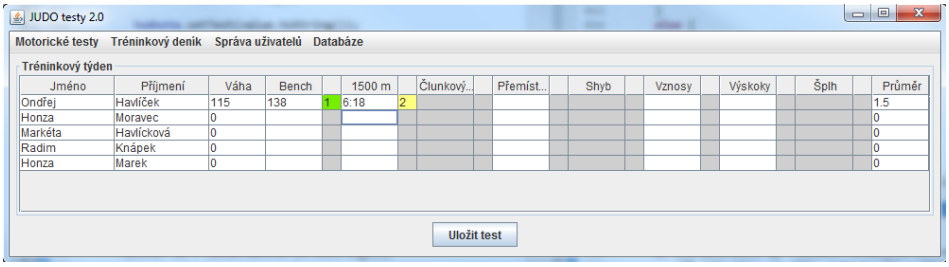
4.3 Vytvoření nového testu

Nový test se vytváří ve dvou krocích. Nejprve se vyberou osoby pro test, typ testu a testované disciplíny. Výběr osob se provádí výběrem skupiny a přetažením osob z levého panelu do pravého, jak je vidět na obrázku dále.



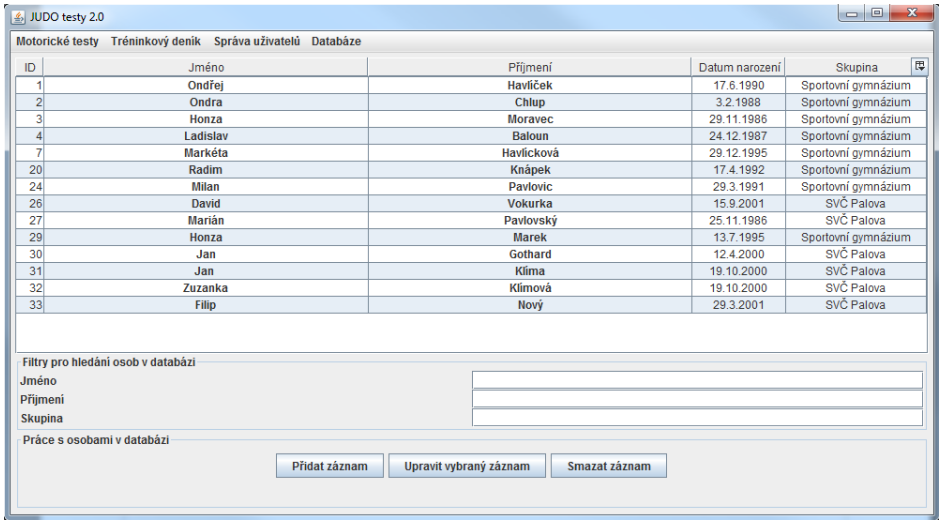
Dále je potřeba vybrat jeden z typů testu a zaškrtnout požadované disciplíny. Pro každý test je důležité zadat i jméno trenéra. Bez jeho jména nelze test vytvořit. Nakonec stiskněte tlačítko ***Vytvořit test***.

Poté se zobrazí okno s tabulkou všech osob, které byly vybrány a jejich disciplín. Po vyplnění této tabulky stiskněte tlačítko ***Uložit test***. Tím se všechny údaje uloží do databáze.



5 Správa uživatelů

Všechny uživatele, které máte v systému zadány si můžete prohlédnout po stisknutí tlačítka ***Správa osob*** z menu Správa uživatelů. Zobrazí se následující okno:

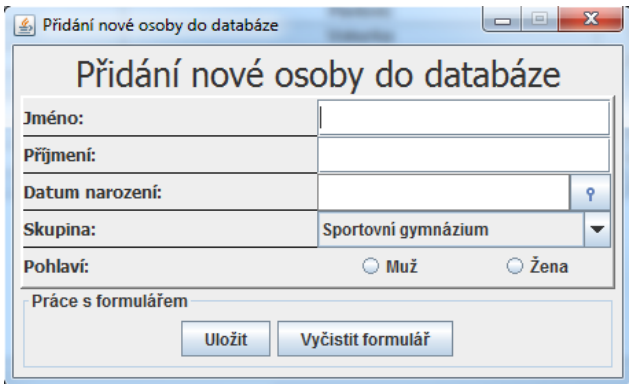


Vidíte základní údaje o všech dříve zadaných osobách. Data lze řadit podle jednotlivých sloupců jednoduše klinutím na název sloupce.

Dostupný je i filtr pro vyhledávání. Data lze filtrovat podle jména, příjmení nebo skupiny. Do příslušné kolonky pouze zadáte začátek výrazu který vyhledáváte a z tabulky se automaticky vyřadí nevyhovující položky.

5.1 Přidání záznamu

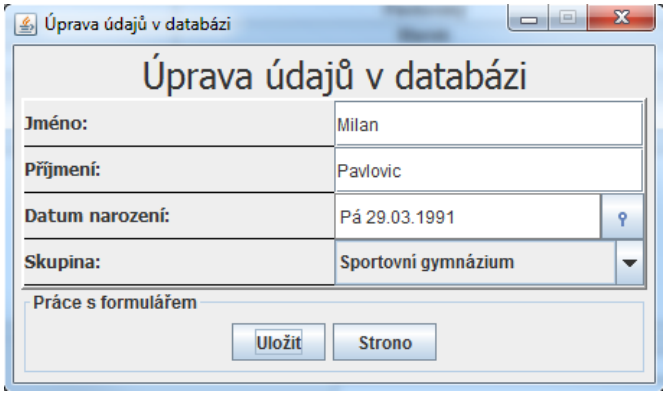
Novou položku do databáze přidáte stisknutím tlačítka ***Přidat záznam***. Pak se zobrazí dialogové okno s položkami pro zadání potřebných informací.



Všechny položky jsou povinné. Pokud některou položku nevyplníte, zobrazí se hláška a údaje se nezapíší.

5.2 Úprava osoby v systému

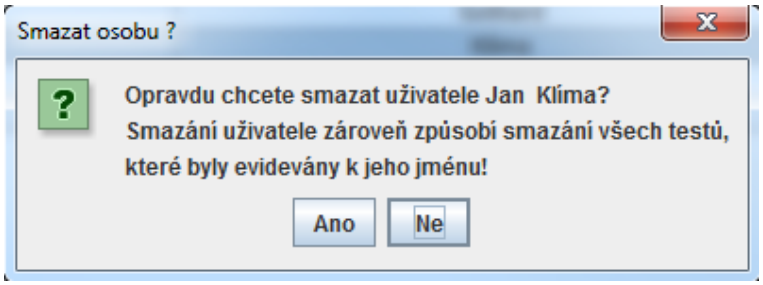
Pokud chcete upravit údaje o osobě, která již v databázi existuje vyvoláte stejné okno jako v předchozí kapitole najetím na konkrétní osobu v tabulce a stisknutím tlačítka ***Upravit vybraný záznam***. Do dialogu se předvyplní údaje, které jsou u vybrané osoby aktuální. Po úpravě potřebných polí, stiskněte tlačítko ***Uložit*** a data se zapíší do databáze. V případě, že změnu údajů provést nechcete, stiskněte tlačítko ***Storno***.



5.3 Smazání osoby ze systému

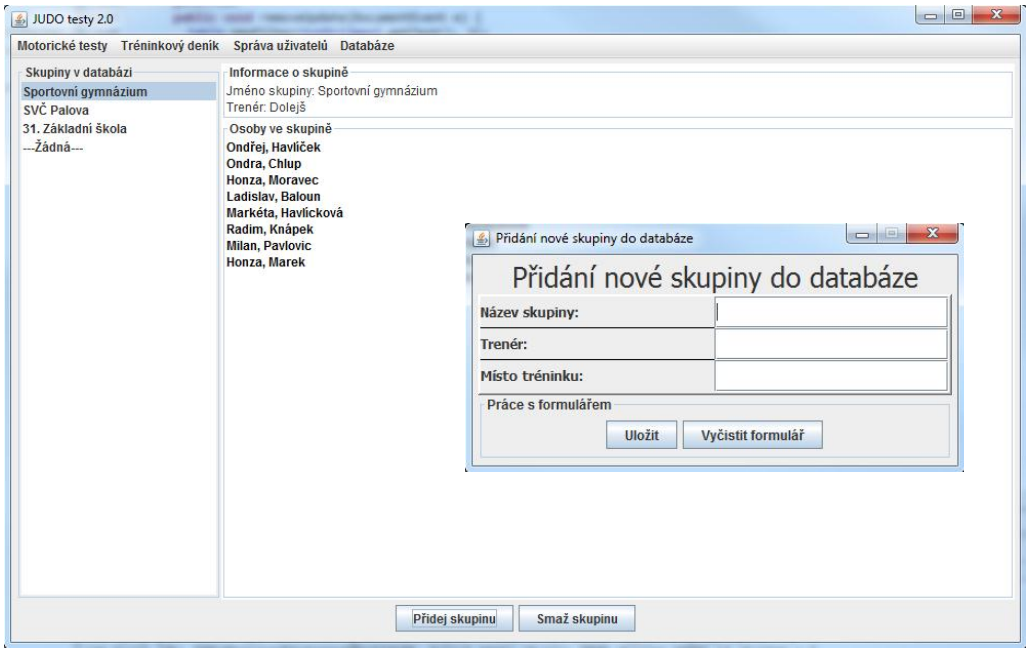
Je samozřejmě možné osobu ze systému vymazat. Nicméně to s sebou přináší určité problémy. Pokud se totiž jedná o osobu, která se již zůčastnila některých z testů, budou zároveň s ní ze systému vymázána data o všech jeho testech. Dejte si tedy pozor abyste se smazáním osoby nepřipravili o potřebná data.

Smazání provedete výběrem jedné osoby v tabulce a stiskem tlačítka ***Smazat záznam***. Poté se zobrazí varování před smazáním důležitých dat. Zde pro smazání stiskněte ***Ano***.



6 Správa skupin

Přidávání a mazání skupin lze provádět po stisknutí tlačítka ***Správa skupin*** v menu Správa uživatelů. Mazání i přidání skupiny probíhá stejně jako mazání a přidávání Osoby. Po stisku tlačítka ***Přidat skupinu*** je nutné zadat všechny potřebné údaje a odeslat je tlačítkem ***Uložit***.

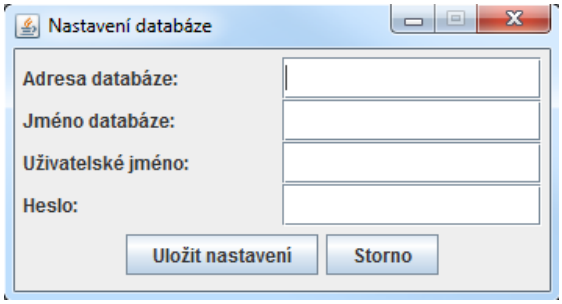


V okně správy skupin lze procházet všechny skupiny a zobrazovat jejich členy.

7 Databáze

V případě, že nevyužíváte databázi od výrobce softwaru, můžete přístup k ní nastavovat ručně. Nedoporučujeme do tohoto nastavení zasahovat, pokud přesně nevíte co děláte.

V opačném případě je k dispozici dialog pro nastavení připojení k databázi. Je potřeba nastavit IP adresu stroje, na kterém běží databáze, jméno databáze a přístupové údaje. Po změně údajů restartujte pro jistotu celou aplikaci.



8 Servis a řešení chyb

Pokud v systému naleznete jakoukoli chybu, můžete kontaktovat autora na adrese ondrej.hav@gmail.com. V případě problémů s databází nejprve zkontrolujte, zda jste připojeni k internetu. Pokud problém přetrvává, opět kontaktujte autora.