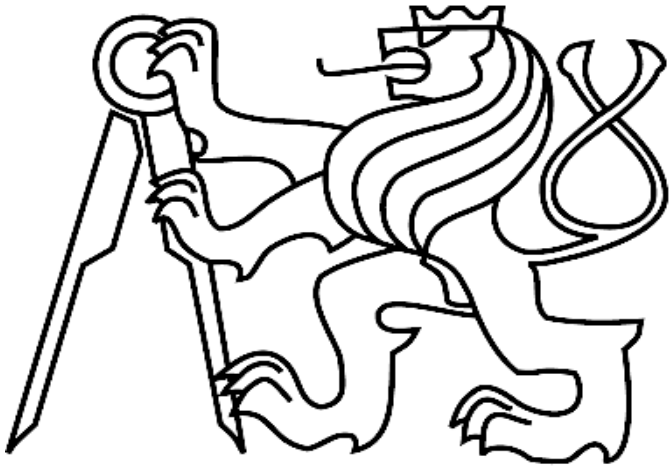
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Rozšíření databáze zkušebních obvodů

Martin Hübner

Vedoucí práce: Ing. Petr Fišer PhD.

Studijní program: Elektronika a informatika, strukturovaný, bakalářský
Obor: Výpočetní technika

červen 2009

Poděkování

Děkuji mému vedoucímu práce, panu Ing. Petru Fišerovi PhD., za cenné rady, připomínky a podněty.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Paze dne 3. 6. 2009 ….....

Abstract

This bachelor thesis resumes on diploma thesis by Jaroslav Pachola from january 2007. It describes widening web based database of benchmarks for testing of logic synthesis algorithms. Database of benchmarks utilizes HTML, CSS and AJAX on client side and PHP, PostgreSQL and Python on server side. It allows to catalogue benchmarks, search them using various criteria and download benchmark files that match those criteria – individually or all at once.

The objective of this thesis is to design and implement capability of adding results of benchmarks, capability of organizing the benchmarks into categories and some small functions. The thesis is concerned with describing design and implementation, safety problems and testing. The thesis is complemented by source code, installation guide and user guide.

Abstrakt

Práce navazuje na diplomovou práci Jaroslava Pacholy z roku 2007. Zabývá se především rozšiřováním webové databáze zkušebních obvodů pro testování algoritmů pro logickou syntézu. Databáze zkušebních obvodů využívá technologií HTML, CSS a AJAX pro klientskou část a PHP, PostgreSQL a Python pro serverovou část. Slouží ke katalogizaci zkušebních obvodů. Umožňuje vkládat zkušební obvody do databáze, vyhledávat je podle zadaných kritérií, stahovat ty které daným kritériím vyhovují a to buďto jednotlivě, nebo hromadně.

Cílem práce bylo především navrhnout a implementovat možnost do katalogu přidávat výsledky zpracování zkušebních obvodů, možnost kategorizovat instance zkušebních obvodů a další drobností. Práce se zabývá stručným popisem návrhu a implementace, bezpečností i testováním aplikace. K práci jsou přiloženy zdrojové texty a uživatelská příručka i instalační příručka.

Obsah

Seznam obrázků

1 Úvod

Tato práce navazuje na práci „Databáze zkušebních obvodů“ od Jaroslava Pacholy z ledna 2007, která se kromě jiného zabývala vytvořením aplikace s webovým uživatelským rozhraním pro katalogizaci zkušebních obvodů pro testování algoritmů pro logickou syntézu. Tato práce se zabývá především úpravou a rozšířením původní webové aplikace. Okrajově se také zabývá úpravami a rozšířeními řádkových skriptů v jazyce Python, které s webovou aplikací spolupracují.

Údaje o zkušebních souborech jsou ukládány do relačního databázového serveru PostgreSQL, ale díky použití univerzální knihovny AdoDB je možné využít i jiné databázové systémy. Soubory popisující zkušební obvody jsou ukládány do k tomu určeného adresáře na serveru. Serverová část webové aplikace je napsána především v jazyce PHP a z části také v jazyce Python. V tomto ohledu se nová aplikace od původní příliš neliší. V klientské části aplikace využívám kromě HTML a CSS i technologii AJAX. Díky tomu je ovládání aplikace pro uživatele více pohodlné.

1.1 Stručný popis jednotlivých kapitol práce

Kapitola 2 se zabývá nejen popisem původní práce na kterou tato navazuje. Popisuji zde jak základní funkci aplikace tak i některé zajímavé detaily. Je zde také uveden přehled drobných nedostatků, které jsem v původní aplikaci objevil.

Kapitola 3 stručně a obecně popisuje cíle mé práce. Především se zaměřuje na to jak by měla vypadat budoucí webová aplikace, ale nastiňuje i úpravy a vylepšení, které přímo s webovou aplikací nesouvisí.

Kapitola 4 navazuje na kapitolu 3. Rozvíjí její jednotlivé myšlenky více do detailu. Jsou zde konkrétně specifikované a okomentované požadavky kladené na novou verzi webové aplikace a tuto práci jako takovou.

Kapitola 5 popisuje konkrétní postupy při návrhu implementaci aplikace. Jsou popisovány především obecně, ale ve vhodných případech je zde drobná ukázka konkrétních řešení. Zabývá se také analýzou různých alternativních řešení.

Kapitola 6 se zabývá problematikou spojenou s bezpečností webových aplikací. První část je pojata obecně a druhá konkrétně popisuje bezpečnostní opatření použité v původní i nové verzi webové aplikace. Dále zkoumá možná bezpečnostní rizika nové verze aplikace a obhajuje důvody proč nebyla tato rizika eliminována.

Kapitola 7 nejprve popisuje testování softwaru obecně a poté se zabývá konkrétně testováním nové verze databáze benchmarků. Především se zabývá popisem scénářů pro systémové black box testy.

V závěru se ohlížím nad provedenou prací a nastiňuji různé možnosti pokračování.

1.2 Stručný popis aplikace

K čemu vlastně tato aplikace bude sloužit? Jejím úkolem je především katalogizovat a uchovávat zkušební obvody. Uživatelům umožňuje zkušební obvody přidávat, upravovat jejich popis i mazat. Jelikož se očekává, že v databázi bude za běžného provozu uloženo alespoň několik stovek zkušebních obvodů, je kvůli pohodlí uživatelů třeba je efektivně organizovat. Jedna z možností je je organizovat do sad, neboli setů.

Sady vznikly jako výsledek práce institucí z oblasti výzkumu a vzdělávání a obsahují zkušební obvody v rozličných formátech. Tyto formáty popisují zkušební obvody na různé úrovni, od popisu chování v behaviorálních modelech, až po detailní popis struktury obvodu. Některé formáty jsou lépe čitelné člověkem, jiné hůře, některé jsou snáze počítačově analyzovatelné a jiné velmí obtížně. Každý formát obsahuje množinu atributů, které jsou pro popis obvodů v něm relevantní. Původní práce používala pětici formátů - konkrétně formáty PLA, KISS2, BENCH, BLIF a SLIF. Podrobnosti o jednotlivých formátech lze nalézt v [1], kapitola 4. Já podporu pro tyto formáty zachoval, ale žádné nové nepřidal. Aby měli uživatelé více možností jak organizovat zkušební obvody, tak jsem do nové verze aplikace implementoval možnost zařadit instance zkušebních obvodů do kategorií. Každá instance může náležet do více kategorií současně.

Dále aplikace bude umožňovat ukládat k jednotlivým instancím zkušebních obvodů i tzv. výsledky. Výsledek je uskupení informací, které byly získáný z instance zkušebního obvodu. K výsledku může být přiložen soubor popisující např. obvod, ale není to podmínkou.

1.3 Dodatek

Některé části této kapitoly jsou inspirovány, nebo přímo přebrány z [1], kapitoly 1.

K aplikaci jsem vytvořil instalační a uživatelskou příručku. Obě jsou umístěný na CD, jeho obsah monitoruje příloha B.

2 Analýza původního stavu aplikace

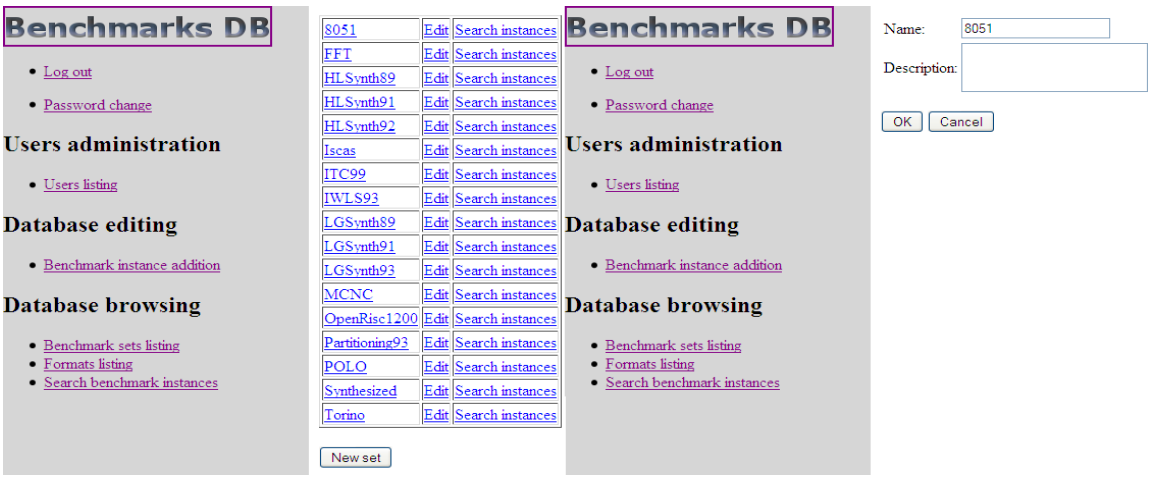
2.1 Popis původní aplikace

Databáze zkušebních obvodů je webová aplikace, která umožňuje jejím uživatelům uchovávat zkušební obvody a informace o nich. Umožňuje katalogizovat zkušební obvody. vyhledávat je podle různých kritérií a následně dané soubory hromadně či jednotlivě stahovat. Serverová část aplikace využívá databázový systém PostgreSQL a jazyk pro vytváření dynamických webových stránek PHP. Pro analýzu zkušebních obvodů a některé další pomocné obvody je využít jazyk Python. Klientská část aplikace využívá HTML a CSS. Aplikace obsahovala tyto konkrétní funkcionality:

2.1.1 Správa sad zkušebních obvodů

Očekává se, že v databázi zkušebních obvodů bude za běžného provozu uloženo velmi mnoho zkušebních obvodů. Proto původní aplikace obsahovala možnost tyto zkušební obvody organizovat. Původní aplikace umožňovala organizovat zkušební obvody do sad.

Každý uživatel (včetně nepřihlášených) měl možnost zobrazit všechny sady v databázi s přímým linkem do vyhledávače instancí. Pokud je přihlášen uživatel s příslušnými právy, byla mu přístupná i možnost editovat existující a přidávat nové sady.



Obr. 1 : Původní aplikace - správa sad zkušebních obvodů

2.1.2 Správa formátů

Aplikace umožňovala uchovávat instance zkušebních obvodů popsané soubory s různým formátem zápisu. Uchovávala u každé instance informaci v jakém formátu je soubor popisující danou instanci.

Nepřihlášený uživatel měl možnost zobrazit všechny formáty podporované aplikací. Přihlášený uživatel s příslušným oprávněním měl možnost navíc formáty editovat.

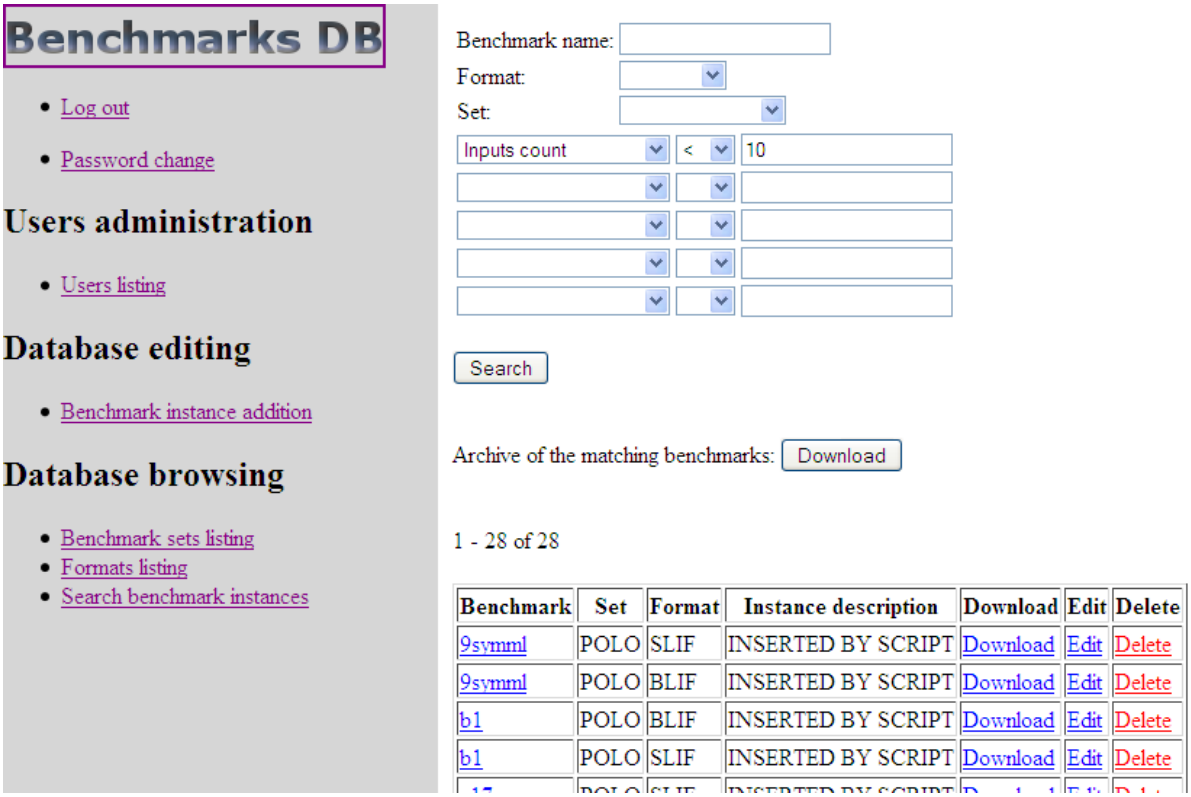
2.1.3 Správa uživatelů

Pokud měl aktuálně přihlášený uživatel příslušné oprávnění, bylo mu pomocí webového rozhraní umožněno měnit vlastní heslo. Pokud měl k tomu určené speciální oprávnění, mohl ediovat záznamy o uživatelích, mazat uživatelské účty a přidávat nové uživatelské účty. Pouze účet, pomocí něhož byl uživatel přihlášen byl nesmazatelný.

2.1.4 Hledáni instancí zkušebních obvodů

I nepřihlášený uživatel měl přístupné všechny možnosti vyhledávání instancí. Konkrétně se jednalo o hledání podle jména zkušebních obvodů, podle sad, podle formátů a podle hodnot až pěti atributů. Ke specifikování hodnot atributů bylo k dispozici šest operátorů (= < > <= >= <>). Zde byl jeden nedostatek plynoucí z technologií, které byly použity k tvorbě této aplikace. Aplikace totiž nepoužívala skripty na straně webového prohlížeče a řešení tohoto problému pomocí skriptů na straně serveru by nebylo dostatečně uživatelsky přívětivé. Konkrétně formulář umožňoval hledat podle atributů, které neobsahuje vybraný formát.

Po zobrazení výsledků měl každý uživatel, kromě základních informací o instancích a linku na zkušební obvod každé instance, k dispozici možnost stáhnout zip archív se všemi nalezenými instancemi, nebo stahovat jednotlivé instance. V případě že byl přihlášen uživatel s příslušnými právy, měl možnost instance navíc editovat a mazat. V případě, že počet nalezených instancí přesáhoval zadaný počet, bylo k dispozici stránkování.



Obr. 2 : Původní aplikace - hledání instancí

2.1.5 Řádkové konfigurační scripty

Pro snadnější instalaci byly k dispozici skripty v jazyce Python, které z uživatelem snadno pochopitelného metakonfiguračního souboru vygenerovaly všechny potřebné soubory s konfiguracemi. Toto nebylo zcela dořešeno, neboť bylo třeba doplnit uživatelské jméno a heslo k přístupu do databáze do skriptu *Datebase.py*.

2.1.6 Konfigurační PHP scripty

Po vygenerování konfiguračních souborů bylo třeba připravit databázové úložiště. Nejprve bylo nutné spustit skript create*Tables.php*, který vygeneroval potřebné tabulky a závislosti a omezení mezi nima. Dále bylo třeba naplnit tabulky základními daty (jména sad, formátů, atributů a vztahy mezi nimi). K tomu sloužil skript *initTables.php*. Pro úspěšné spuštění skriptu bylo nutné se nejprve přihlásit do aplikace, jinak se skript nespustil a ohlásil : „Access denied“.

2.1.7 Řádkové parsovací scripty

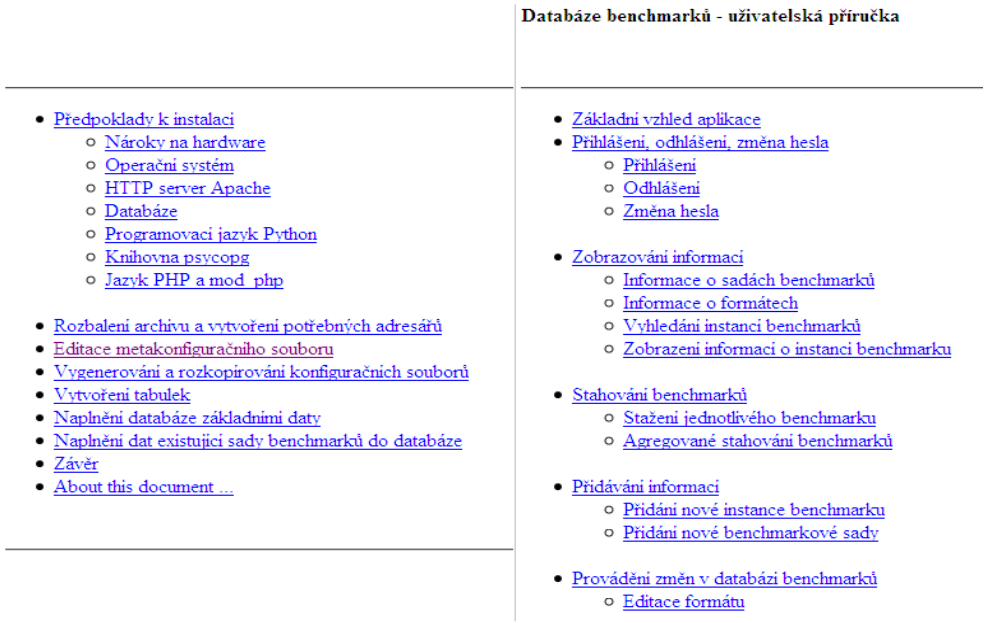
Pro usnadnění práce při přidávání jednotlivých instancí byly k dispozici skripty v jazyce Python, které z aktuálně nahrávaného souboru zjistily typické atributy pro formát nahrávaného souboru. Ty je pak možné automaticky vyplnit do databáze. Tyto skripty byly integrovány s webovým rozhraním.

2.1.8 Řádkové plnící scripty

Pokud uživatel chtěl nahrát do databáze velkou skupinu (např. sadu) instancí zkušebních obvodů, měl k dispozici řádkové skripty v jazyce Python, které byly schopny nahrávat data a vyplňovat atributy do databáze automaticky.

2.1.9 Dokumentace a návody

K aplikaci byla vytvořena instalační a uživatelská příručka.



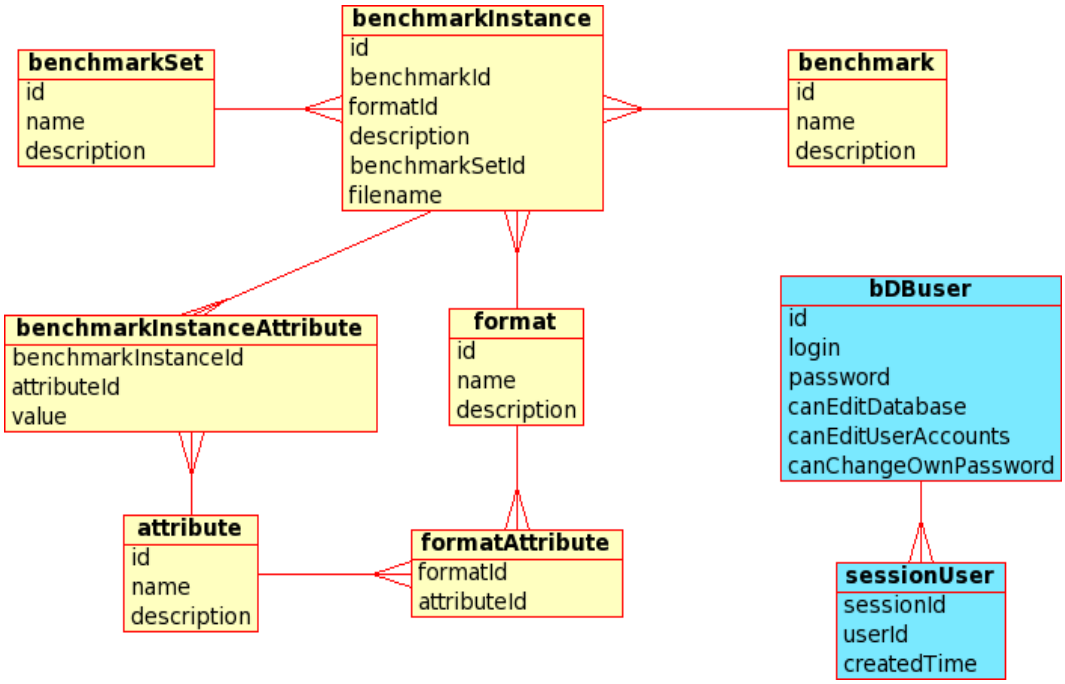
Obr. 3 : Ukázka dokumentace původní aplikace

2.1.10 Úložiště souborů

Aplikece ukládala soubory instancí do adresáře *./benchmarkFiles*. V adresáři *./benchmarkFiles* byly navíc ještě vytvořeny podadresáře nesoucí názvy sad. Do těchto podadresářů byly dané soubory ukládány.

2.2 Struktura databáze původní aplikace

Původní struktura databáze obsahovala 9 tabulek. 7 z nich sloužilo pro samotné ukládání bechnmarků, dvě sloužily k zabezpečení přístupu uživatelů. Jejich vzájemné vztahy jsou patrné z následujícího schématu:



Obr. 4 : Struktura databáze původní aplikace

2.2.1 Tabulky pro zabezpečení přístupu

Zabezpečení přístupu zajišťovaly dvě tabulky. Tabulka bdbuser uchovávala inforace o registrovaných uživatelích včetně jejich přístupových práv. Tabulka sessionUser sloužila k identifikaci přihlášeného uživatele na základě relace.

2.2.2 Tabulky pro ukládání dat o zkušebních obvodech

Pro ukládání samotných užitečných dat do databáze sloužilo 7 tabulek. Konkrétně se jednalo o tabulky benchmarkSet, benchmarkInstance, benchmark, benchmarkInstanceAttribute, attribute, formatAttribute a format.

- benchmarkSet - informace o sadách zkušebních obvodů
- benchmarkInstance - informace o instancích zkušebních obvodů
- benchmark - informace o zkušebních obvodech
- benchmarkInstanceAttribute - hodnoty jednotlivých atributů u instancí zkušbních obvodů
- attribute - informace o atributech
- formatAttribute - výčet atributů, které podporoval každý formát
- format - informace o formátech

2.2.3 Používané datové typy

V aplikaci byly použity dva datové typy:

- INTEGER - pro identifikátory jednotlivých záznamů (id),
- CHARACTER - pro datové sloupce.

Datové sloupce byly dvojího typu. Běžné s maximální délkou 100 znaků, a sloupce pro dlouhé popisy s maximální délkou 2000 znaků.

2.2.4 Používaná integritní omezení

Pro hlídání integrity dat byla použita následující integritní omezení:

- UNIQUE - zaručuje jedinečnost daného sloupce v každém řádku (např. id, nebo název formátu)
- FOREIGN KEY - svazuje hodnotu sloupce s hodnotou sloupce v jiné tabulce, např. sloupec formatId v tabulce benchmarkInstance musí obsahovat id některého řádku z tabulky format, jinak takový řádek v tabulce benchmarkInstance postrádá významu.

2.3 Drobné nedostatky původní aplikace

Přestože byla aplikace napsána i otestována pečlivě a představovala ucelenou práci, objevil jsem při jejím zkoumání následující drobnosti, které by stály za to vylepšit. Některé již byly zmíněny výše, ale přes to je v tomto souhrnu pro přehlednost uvádím:

- Při pokusu mazání různých položek aplikace dané položky bez varování mazala - to není ideální reakce - uživatel běžně čeká, že aplikace zobrazí varování a až po potvrzení ze strany uživatele dannou položku smaže. Konkrétně se tento problém týkal mazání uživatelů a instancí (jiné položky původní aplikace neumožňovala mazat pomocí webového rozhraní).

- V případě mazání instancí zkušebních obvodů se smazal pouze záznam o instanci v databázi. Soubor obsahující samotnou instanci na disku zůstával.

- Aplikace umožňovala přidat dva uživatele se stejným uživatelským jménem. Následkem čehož mohlo dojít při přihlašování ke kolizi uživatelských jmen a vzniku bezpečnostních rizik.

- Aplikace umožňovala přidat dve sady se stejným jménem. Navíc byla v aplikaci chyba, která za určitých okolností znemožnila přidat novou sadu. Tato chyba byla způsobena překlemem ve scriptu *initTables.php*.

- Pokud uživatel zadal špatné uživatelské jméno nebo heslo nebyl na to žádným způsobem upozorněn. To není pro uživatele příjemné.

- Pro konfiguraci aplikace bylo třeba ručně upravit soubor *Database.py* .

- Formulář hledání instancí umožňoval vyplnit atributy, které neobsahoval zvolený formát. Toto bylo kvůli použití pouze HTML na klientské straně.

- Při zadání nesmyslného vstupu do pole „hodnota atributu“ ve formuláři pro hledání instancí (např. alfnumerických znaků) aplikace havarovala.

- Některá vstupní políčka ve formuláři hledání instancí nebyla popsána. Zde je diskutabilní jestli je toto nedostatek, neboť z logiky formuláře lze snadno význam těchto políček odhadnout.

- Aplikace uživatele směrovala vždy na úvodní stránku. Uživatelsky příjemnější je návrat na místo vzniku problému, či vyvolání akce.

- Webová aplikace negenerovala HTML kód podle norem.

2.4 Dodatek

Detailní informace k původní práci lze nalézt v [1]. Nemalá část této kapitoly také z této práce čerpala, především z kapitoly 7.

3 Přehled cílů práce

Hlavním cílem této práce bylo navrhnutí a implementace některých dalších možností do původní webové aplikace se současným zachováním všech možností původní aplikace. Jednalo se především o:

- Zavedení možnosti evidovat výsledky získané po zpracování instancí zkušebních obvodů.
- Zavedení možnosti organizovat instance zkušebních obvodů do kategorií. Každá instance zkušebního obvodu bude moci být přiřazena do libovolného množství kategorií. Uživatel bude mít možnost upravovat stávající, přidávat nové a mazat stávající kategorie.
- Zavedení možnosti hromadné manipulace s instancemi zkušebních obvodů.
- Umožnění práce odděleně se zkušebními obvody a jejich instancemi.
- Umožnění mazání celých sad zkušebních obvodů.
- Umožnění mazání formátů.
- Umožnění upravování, přidávání nových a mazání neblokovaných atributů.
- Zvýšení komfortu práce s celou aplikací - konkrétně zavedení filtrování nabízených atributů podle formátu a „našeptávání“ jmen při vyplňování dotazů hledání.
- Další drobné úpravy odstraňující drobné nedostatky původní práce.

4 Požadovaná funkcionalita aplikace

Základním požadavkem na budoucí aplikaci bylo zachovat původní možnosti aplikace, opravit drobné nedostatky a dále aplikaci rozšířit o několik rozsálejších funkcionalit a vylepšit několik drobností:

4.1 Hlavní okruhy rozšíření

4.1.1 Evidence výsledků

Jeden z hlavních požadavků na budoucí vylepšenou aplikaci bylo umožnit ke každé instanci vložit jeden nebo i více výsledků. Instance výsledku se bude do aplikace vkládat jak ve formě záznamu do databáze, tak ve formě souboru, který je nahráván do souborového úložiště. U každé instance výsledku bude, podobně jako u instancí zkušebních obvodů, veden záznam v databázi o formátu instance, hodnotách atributů a instanci zkušebního obvodu, ke které instance výsledku náleží.

Množiny formátů instancí zkušebních obvodů a instancí výsledků nebudou shodné. Bude moci ale nastat situace, že formát instance zkušebního obvodu se bude shodně jmenovat jako formát výsledku. Shodně pojmenované formáty se ale budou moci lišit v množině atributů, které budou připouštět.

Množina atributů bude shodná jak pro formáty instancí zkušebních obvodů, tak pro formáty instancí výsledků.

Výsledky se budou přidávat pomocí vyhledávače instancí zkušebních obvodů. V tabulce s nalezenými instancemi bude vidět kolik výsledků ta která instance obsahuje a bude zde také tlačítko pro přidání nového výsledku do databáze.

4.1.2 Hromadná správa instancí zkušebních obvodů

Původní aplikace obsahovala pouze hromadné stahování instancí zkušebních obvodů podle zadaných vyhledávacích kritérií. K této možnosti přibude možnost instance zkušebních obvodů podle zadaných vyhledávacích kritérií mazat.

4.1.3 Kategorizace instancí zkušebních obvodů

Původní aplikace byla schopna organizovat instance zkušebních obvodů pouze do sad zkušebních obvodů. Toto řešení nebylo úplně vhodné, protože sady zkušebních obvodů by logicky měli organizovat především zkušební obvody a ne jejich instance. Proto bylo rozhodnuto zavést možnost organizovat instance zkušebních obvodů do kategorií. Každá instance zkušebního obvodu bude nyní moci být přítomna v libovolném množství kategorií, nebo nebude muset být přítomna v žádné.

Zavedení možnosti kategorizovat instance zkušebních obvodů se umožní změnit význam sad zkušebních obvodů. Nyní budou sady zkušebních obvodů obsahovat zkušební obvody a ne jejich instance jako v původní aplikaci.

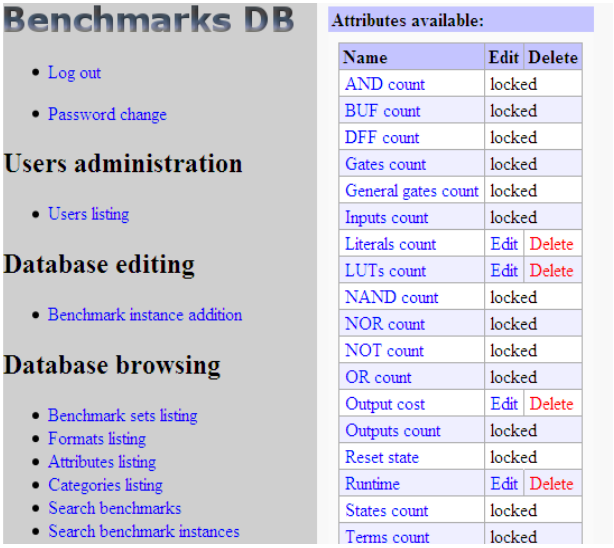
Je požadováno také rozšířit automatické plnění databáze o podporu kategorií. K automatickému plnění databáze bude sloužit (stejně jako u původní aplikace) řádkový Pythonovský skript *BenchmarkFiller.py*.

Uživatel s příslušnými oprávněními bude mít možnost přes webové rozhraní kategorie spravovat. To znamená, že je bude moci přidávat nové a editovat i mazat stávající.

4.1.4 Umožnění evidovat atribute webovým rozhraním

Aby byla nová verze aplikace ucelená, rozhodl jsem se zpřístupnit uživatelům evidenci atributů. Uživatel bude moci pomoci webového rozhraní přidávat nové a editovat i mazat některé stávající atributy. Nelze očekávat, že toho bude běžně uživatel využívat, ale ve specifických případech to najde uplatnění. Např. v případě, že uživatel bude potřebovat nestandardně popsat buďto instanci zkušebního obvodu, nebo výsledku a nebude na to stačit kolonka „description“. Výhodou těchto popisků bude možnost podle nich vyhledávat. Nevýhodou bude fakt, že mohou nabývat pouze celočíselných hodnot.

Jak jsem naznačil výše, editace a mazání se bude týkat pouze některých atributů. Konkrétně se bude jednat pouze o atributy, které nejsou využívané řádkovými parsovacími scripty. Tyto scripty totiž nejsou schopny bez těchto atributů pracovat správně. Takové atributy jsou v přehledu atributů označeny jako zamčené.



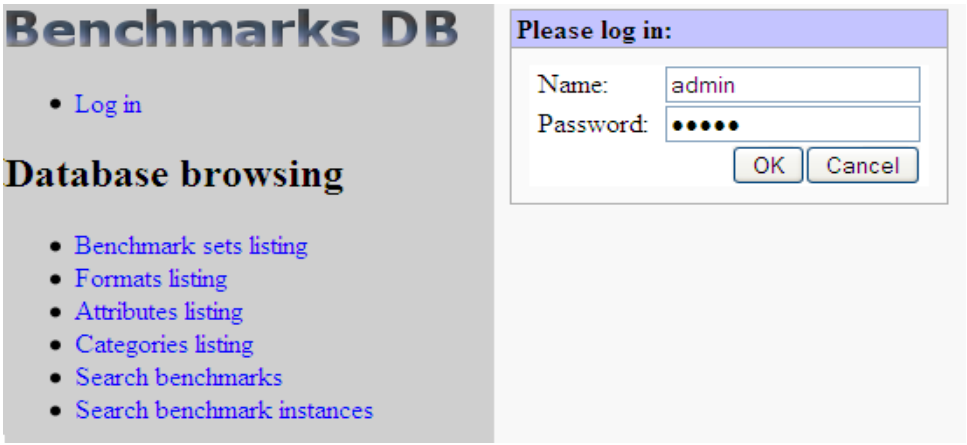
Obr. 5 : Přehled atributů

4.2 Drobnosti hodné vylepšení

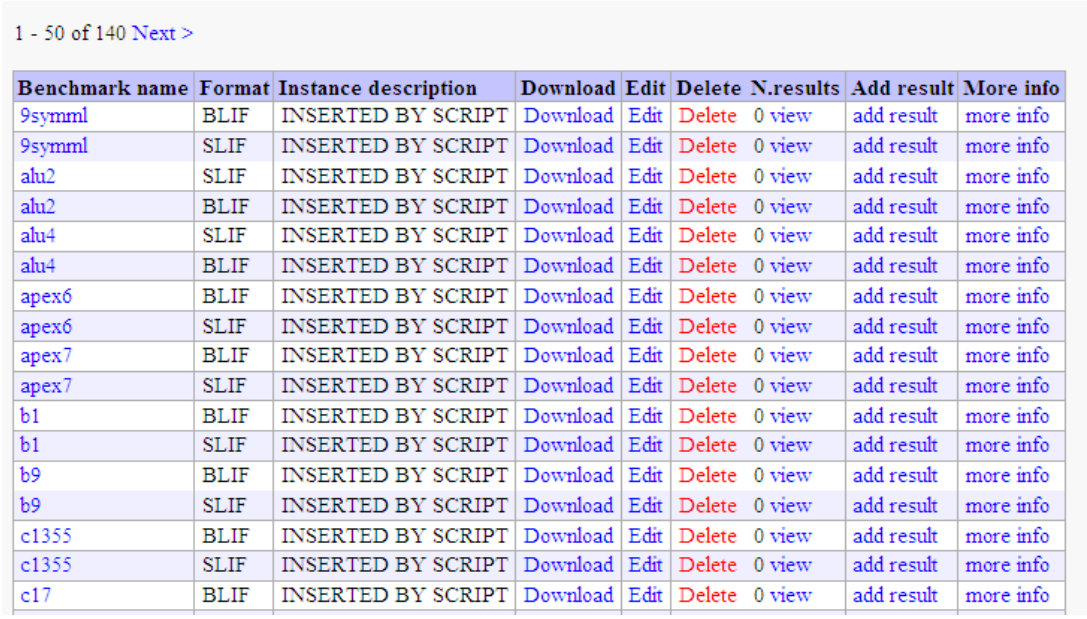
Původní aplikace vykazovala drobné nedostatky a omezení. Některé z nich jsem se rozhodl odstranit. Jednalo se především o následující:

4.2.1 Drobné designové úpravy

Pomocí několika jednoduchých opatření zpříjemnit uživateli práci s webovým rozhraním. Např. se jedná o zvýraznění odkazu po najetí na něj myší, nebo barevné odlišení sudých a lichých řádků v tabulkách. Další změny ukážu pomocí náhledů do nové aplikace.



Obr. 6 : použití nových barev, orámování logických celků, úprava přihlašovacího formuláře tak, aby lépe zapadl do celé aplikace



1 - 50 of 140 Next >

Benchmark name	Format	Instance description	Download	Edit	Delete	N.results	Add result	More info
9symml	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
9symml	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
alu2	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
alu2	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
alu4	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
alu4	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
apex6	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
apex6	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
apex7	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
apex7	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
b1	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
b1	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
b9	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
b9	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
c1355	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
c1355	SLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info
c17	BLIF	INSERTED BY SCRIPT	Download	Edit	Delete	0 view	add result	more info

Obr. 7 : barevné rozlišení sudých a lichých řádků v tabulkách

4.2.2 Drobná funční vylepšení

- Vytvořit tzv. našeptávač názvů pomocí technologie AJAX a tím zpříjemnit uživateli přidávání nových instancí a vyhledávání zkušebních obvodů i jejich instancí.

- Pomocí technologie AJAX zamezit tomu, aby nebylo možné ve vyhledávači instancí zkušebních obvodů hledat podle hodnot atributů, které nejsou relevantní pro vybraný formát.

- Přidání navigačních odkazů i pod zobrazené výsledky ve vyhledávači instancí.

- Umožnit vytvářet nové a mazat existující formáty.

- Upravit aplikaci tak, aby uživatele nesměrovala stále na úvodní stranu.

- Při pokusu o smazání jakékoliv položky zobrazit varování, zda si je uživatel svojí akcí jistý.

- Umožnit mazat celé sady zkušebních obvodů. Při smazání sady se samozřejmě smažou i všechny zkušební obvody a jejich instance.

4.2.3 Opravy drobných nedostatků

Aplikace obsahovala i několik drobných funkčních nedostatků. Mým úkolem je bylo opravit. Jednalo se konkrétně o následující:

- Oprava chyby v aplikaci, která umožňuje aby byli v databázi dva uživatelé se stejným uživatelským jménem.

- Oprava chyby v aplikaci, která umožňuje aby byly v databázi dvě sady se stejným jménem, dále také opravit chybu, která za určitých okolností znemožní přidat novou sadu.

- Opravit mazání instancí tak, aby byly mazány i soubory instancí.

5 Popis návrhu a implementace

Při implementaci serverové části aplikace byly využity velmi podobné technologie jako v původní práci - tj. PHP ve verzi 5, PostgreSQL ve verzi 8 a Python 2.6 (scripty jsou, stejně jako u původní aplikace, v Pythonu 3 nefunkční a nejsou nijak připraveny na strojový překlad kóku pro nový Python 3). Teoreticky by měla aplikace funovat na libovolném systému splňujícím výše uvedené požadavky, ale byla testována jen na systému s operačních systémech Linux a FreeBSD s web serverem Apache.

Při implementaci klientské části bylo využito HTML a CSS. Navíc oproti původní aplikaci byl použit také AJAX. Díky tomu bylo možné uživatelské rozhraní vytvořit více uživatelsky přívětivé.

Při tvorbě nové verze aplikace jsem plně vycházel ze struktury aplikace původní. Ta je popsána v [1], kapitole 7.4.

Stejně jako v původní aplikaci je serverová strana naprogramována ve dvouch programovacích jazycích. Řešení jejich vzájemné spolupráce jsem zachoval - tj. komunikují mezi sebou pomocí příkazové řádky. Stejně tak jsem neměnil způsob konfigurace aplikace pomocí metakonfiguračního souboru. Více se o této problematice lze dozvědět v [1], v kapitole 7.1.

Postup implementace se dá pomyslně rozdělit na tři etapy.

5.1 Návrh nové struktury databáze

Při pohledu na požadavky na novou verzi databáze zkušebních obvodů je zřejmé, že původní struktura databáze nemohla stačit na jejich rozumné uspokojení. Bylo ho potřeba minimálně rozšířit
o nové možnosti. Konkrétně o možnost kategorizovat zkušební obvody a možnost ukládat výsledky.

Bylo také rozhodnuto, že není zcela vhodné, aby tabulka benchmarkSet byla ve vztahu s tabulkou benchmarkInstance. Již název tabulky napovídá, že její místo je spíše v návaznosti na tabulku benchmark. Je přirozenější aby podmnožinou „sad zkušebních obvodů“ byly „zkušební obvody“ a ne „instance zkušebních obvodů“.

Změny provedené v původní struktuře databáze:

- sady zkušebních obvodů jsou vázané ke zkušebním obvodům a né jejich instancim.
- Vztah mezi sadou zkušebních obvodů a zkušebním obvodem byl zvolen jako M:N (tzn. zkušební obvod může náležet do více (nebo žádné) sady a může obsahovat více (nebo žádný) zkušebních obvodů.

Seznam rozšíření původní struktury databáze:

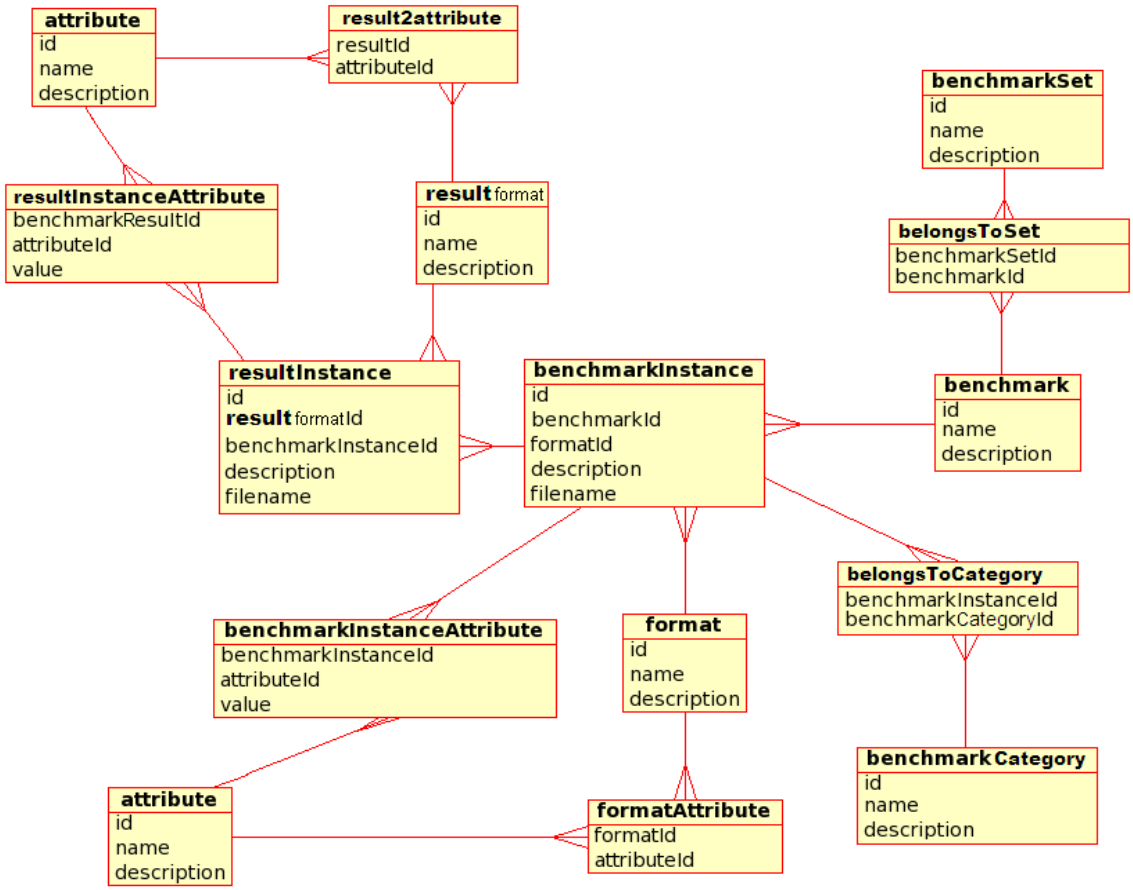
- Přidání tabulek belongsToCategory a benchmarkCategory - k evidenci kategorií instancí zkušebních obvodů a jejich vztahu mezi zkušebními obvody (M:N).

- Přidání tabulek resultInstance, resultFormat, format2attribute a resultInstanceAttribute - k evidenci výsledků, formátů výsledků a vztahů mezi nimi.

Část struktury databáze zajišťující evidenci a autorizaci uživatelů nebylo třeba rozšiřovat.

5.1.1 Nové řešení struktury databáze

Po důkladném zvážení všech pro a proti jsem se dopracoval k následující podobě struktury databáze: (pro přehlednost je tabulka attribute v obrázku zobrazena 2x a není zobrazena část zajišťující přistup uživatelů)



Obr. 8 : Nová struktura databáze

5.1.2 Analýza alternativních struktur databáze

Při navrhování tohoto finálnístruktury databáze jsem zvažoval několik alternativních řešení. V následujících odstavcích je trochu přiblížím.

5.1.2.1 Tabulka benchmarkSet

U tabulky benchmarkSet byl první zádrhel. V původním struktuře databáze byla ve vztahu s tabulkou benchmarkInstance. Zvažoval jsem dvě varianty. První varianta spočívala v ponechání původního stavu. Pro tuto variantu hovořilo především to, že by nebylo potřeba pro zachování stávajících funkcionalit aplikace zasahovat do kódu aplikace při přechodu k nové struktuře databáze. Proti této variantě hovořilo zase to, že ne zcela přesně odrážela reálné potřeby. Z logiky věci plyne, že sada zkušebních obvodů má obsahovat zkušební

obvody a ne přímo jejich instance. Proto jsem se rozhodl tuto variantu zavrhnout a tabulku benchmarkSet dát do vztahu s tabulkou benchmark i přestože to obnášelo více úsilí. Navíc jsem touto změnou umožnil aby jeden zkušební obvod mohl patřit do více sad, což dříve nebylo možné.

5.1.2.2 Tabulka attribute

Při navrhování části struktury databáze týkající se uchovávání výsledků bylo jasné, že bude potřeba oddělit formáty instancí zkušebních obvodů od formátů instancí výsledků. Tak bylo také učiněno. Naskytla se zde ale otázka. Je nutné aby formáty výsledů mohly obsahovat atributy z odlišné množiny, než formáty instancí zkušebních obvodů? Pro variantu s oddělenými množinami formátů hovořilo především to, že takto navržená struktura databáze byla přehlednější. Nevýhodou tohoto řešení byl fakt, že je třeba se starat o dvě množiny s velmi podobným významem. Bylo by vhodné nějakým způsobem řešit jejich synchronizaci. Proto jsem se rozhodl uchovávat atributy pro formáty instancí zkušebních obvodů i pro formáty instancí výsledků v jedné množině, resp. tabulce.

5.1.2.3 Kategorie instancí zkušebních obvodů

Při návrhu části struktury databáze týkající se kategorií instancí zkušebních obvodů jsem řešil také jednu otázku. Mají mít možnost být kategorizovány instance zkušebních obvodů, nebo přímo zkušební obvody? Rozhodl jsem, že je vhodnější, aby bylo možno kategorizovat instance zkušebních obvodů. Pro tuto variantu jsem se rozhodl proto, že instance zkušebních obvodů do této chvíle neměly možnost být organizovány. Zatímco zkušební obvody mohly být organizovány do sad.

5.1.3 Změna úložiště souborů

Kvůli změně struktury databáze, konkrétně umožnění aby jeden zkušební obvod byl součástí více sad zkušebních obvodů, jsem přistoupil ke změně stromové adresářové struktury na ukládání souborů instancí. Původně byly soubory ukládany do podadresářů nesoucích název sady do které dané zkušební obvody patřily. Aby v případě, že zkušební obvod bude obsažen ve více sadách, nebyly soubory instancí uloženy na disku vícenásobně, jsem se rozhodl ukládat všechny soubory instancí přímo do adresáře *benchmarFiles*. Aby byla zajištěna jedinečnost jmen souborů, jsem do všech názvů souborů zakomponoval id instance.

Toto úložiště v nové verzi aplikace slouží i pro ukládání souborů s výsledky. Také do jména souborů s výsledky je pro zamezení duplicit vkládáno jejich id.

Toto řešení mělo ale jednu nevýhodu. Pokud uživatel stahoval soubor z aplikace, tak bylo jméno stahovaného souboru narušeno právě tím id. Toto bylo třeba napravit. PHP naštěstí obsahuje řadu funkcí pro práci s řetězci a tudíž bylo řešení snadné. Pro zjištění na jaké pozici v řetězci je daný první výskyt znaku posloužila funkce *strpos()*, pro samotné oddělení řetězců funkce *substr()*. Více o těchto funkcích lze nalézt na [5].

5.2 Přizpůsobení webové aplikace nové struktuře databáze

V této etapě jsem neimplementoval žádné nové funkce, ani jsem neopravoval žádné nedostatky v aplikaci. Kvůli změně vazeb tabulky benchmarkSet původní aplikace nemohla bez patřičných zásahů fungovat v nové struktuře databáze. K vyřešení tohoto problému stačilo, až na několik vyjímek, pouze upravit SQL dotazy v PHP a Python scriptech. Jako příklad uvedu sql dotaz ze scriptu *benchmark_SHOW.php*, který spočte počet instancí zkušebních obvodů, které obsahuje daná sada:

SELECT COUNT(*) FROM benchmarkInstance WHERE benchmarkId = $benchmarkId

byl nahrazen tímto dotazem:

SELECT COUNT(*) FROM sti WHERE benchmarkId = $benchmarkId

kde sti je pohled vytvořený sql dotazem:

CREATE VIEW sti AS
SELECT benchmarkinstance.id, benchmarkinstance.benchmarkid, benchmark.id AS benchmarkid, benchmarkinstancecategory.id AS benchmarkcategoryid, benchmarkinstance.formatid, benchmarkinstance.description, benchmarkinstance.filename
FROM ((((benchmarkinstance
LEFT JOIN benchmark ON benchmarkinstance.benchmarkid =benchmark.id)
LEFT JOIN belongsto ON benchmark.id=belongsto.benchmarkid)
LEFT JOIN benchmark ON belongsto.benchmarkid=benchmark.id)
LEFT JOIN belongstocategory ON benchmark.id=belongstocategory.benchmarkinstanceid)
LEFT JOIN benchmarkinstancecategory ON
belongstocategory.benchmarkinstancecategoryid = benchmarkinstancecategory.id

5.3 Přizpůsobení řádkových scriptů nové struktuře databáze

Většinu řádkových skriptů v Pythonu nebylo třeba upravovat. Byly určeny k parsování zkušebních obvodů. To se ale neplatí o skriptu *BenchmarkFiller.py*. Ten slouží k uložení instancí zkušebních obvodů do databáze a tudíž ho bylo nutné upravit, aby fungoval s novou strukturou databáze. Úprava se především týkala, podobně jako u úpravy webové aplikace, úpravou sql dotazů.

5.4 Implementace nových funkcí

V druhé etapě jsem se zabýval implementací dalších funkcionalit podle požadavků ze zadání. Zde již bylo nutné provádět rozsáhlejší zásahy do stávajícího kódu a také kód rozšířit o nové části. Bylo také např. potřeba, jak již bylo zmíněno, změnit strukturu úložiště souborů.

5.4.1 Implementace kategorizování instancí

Po vymyšlení struktury databáze nové aplikace, která již počítala s kategoriemi byla samotná implementace rutiní. Při tvorbě kódu jsem vycházel z původní aplikace.

5.4.2 Implementace možnosti přidávat výsledky

Podobně jako u implementace možnosti kategorizovat instance zkušebních obvodů jsem postupoval i u implementace možnosti přidávat k instancím výsledky. Jednalo se o rutiní činnost a při tvorbě kódu jsem vycházel z původní aplikace.

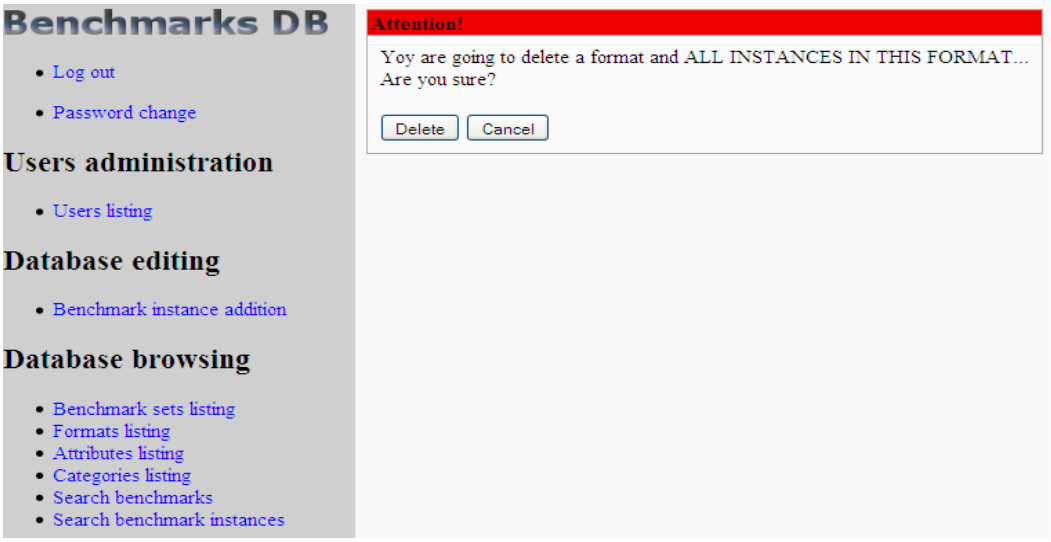
5.4.3 Implementace hromadného mazání instancí zkušebních obvodů

Implementaci hromadného mazání instancí zkušebních obvodů jsem vyřešil podobně jako bylo v původní aplikaci vyřešeno agregované stahování instancí. Předal jsem scriptu, který mazání vykonával část sql dotazu specifikující instance zkušebních obvodů. Poté již nebyl problém tento sql příkaz upravit tak, aby instance nezobrazoval, ale mazal.

5.4.4 Implementace varovných hlášení

Původní aplikace se při mazání jakékoliv položky k uživateli nechovala přívětivě. Položku rovnou bez varování smazala. Proto bylo rozhodnuto implementovat varování před definitivním smazáním položky.

Problém jsem vyřešil velice jednoduše, ale účelně. Pro varování jsem mezi stránku s odkazem na smazání položky a skript mazající položku vložil stránku, která zobrazuje varování a dává možnost uživateli ještě akci odvolat. Takto vypadá např. varování před smazáním formátu:



Obr. 9 : Varování před mazáním formátu

5.4.5 Implementace filtru atributů ve vyhledávači instancí

Jelikož původní aplikace využívala na klientské straně pouze technologii HTML a CSS, nebylo možné efektivně zařídit, aby uživatel při hledání instancí zkušebních obvodů nemohl zadat parametry atributů, které nejsou relevantní pro vybraný formát.

Toto jsem se rozhodl vyřešit. Využil jsem k tomu technologie AJAX. Informace o technologii AJAX i části zdrojových kódů jsem čerpal na [4].

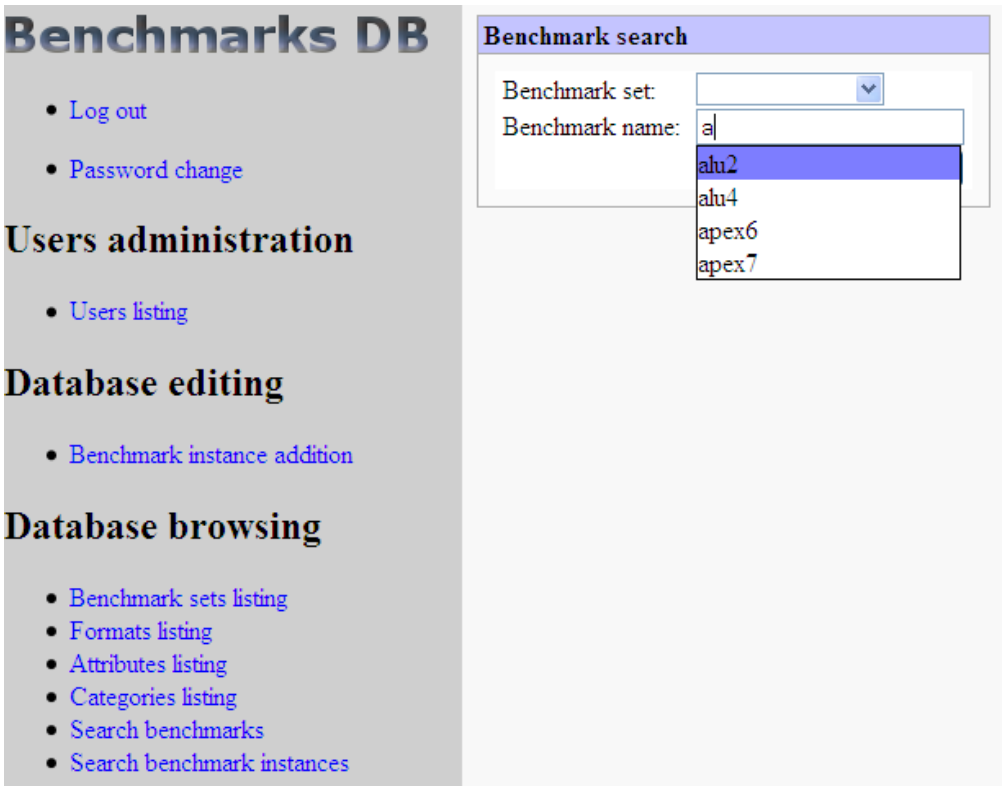
Pro přehlednost kódu jsem aktivní scripty na straně klienta nevložil do souboru *benchmarkInstances_SEARCH.php*, ale do souboru *benchmarkInstances_SEARCH.js*, který jsem do stránky připojil. Dále bylo nutné vytvořit serverový skript, který podle formátu vrátí atributy, které daný formát obsahuje. K tomuto účelu jsem se rozhodl využít PHP (je možné využít libovolnou technologii serverových skriptů). Tento skript se nachází v souboru *benchmarkInstances_SEARCH_AJAX.php*.

5.4.6 Implementace našeptávačů

Našeptávač je funkce některých aplikací, která umožňuje zobrazit uživateli, zatím co píše, tipy na dokončení slova, nebo slovního spojení. Na internetu poprvé toto řešení zavedla firma Google v jejich vyhledávači. Více informací v [3].

V databázi zkušebních obvodů jsem použil našeptávače na třech místech. Ve formuláři pro přidávání nových instancí zkušebních obvodů, ve formuláři pro hledání zkušebních obvodů a ve formuláři pro hledání instancí zkušebních obvodů.

Implementace našeptávačů je velice podobná jako implementace filtru atrbutů, kterou jsem popsal v předchozím odstavci.



Obr. 10 : Ukázka našeptávání

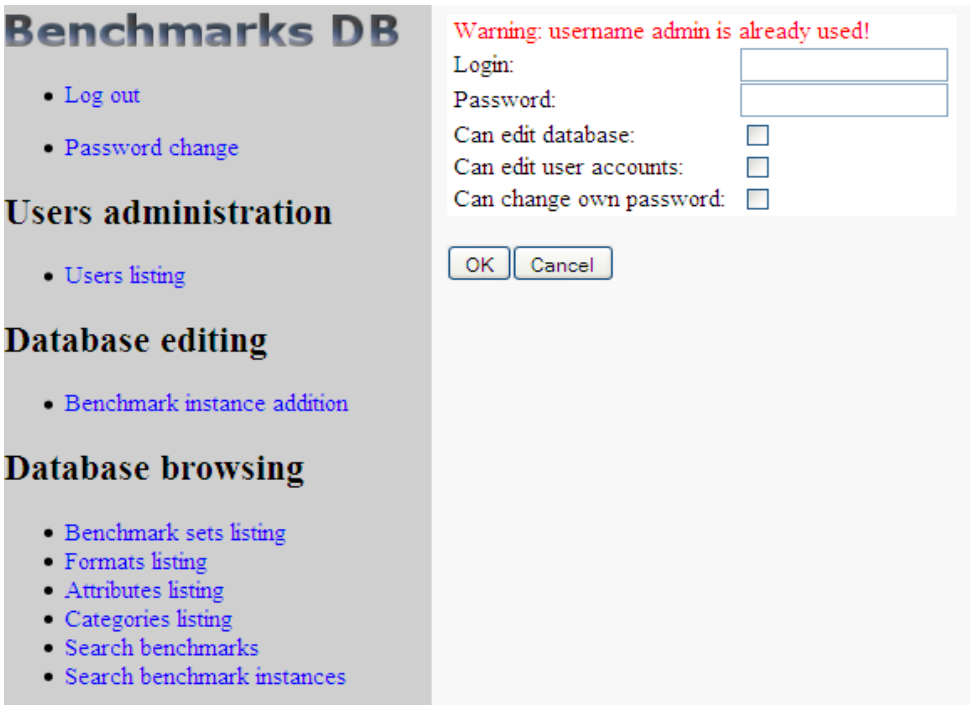
5.5 Implementace drobných oprav

Zde popíšu jak jsem se vypořádal s některými drobnými opravami.

5.5.1 Zamezení vložení dvou položek se stejným jménem

Jeden z nejvážnějších nedostatků původní aplikace spočíval v tom, že pomocí webového rozhraní mohl uživatel vložit dvě položky (např. uživatele, zkušební obvod atp.) nesoucí stejné jméno. Tento stav jsem považoval za nežádoucí a rozhodl se ho napravit.

Implementace této opravy nebyla složitá. Ve skriptu starající se o uložení nové položky do databáze jsem vložil kód, který kontroluje zda již neexistuje položka nesoucí shodné jméno se zamýšlenou novou položkou. V případě, že jméno ještě použito nebylo, skript samozřejmě položku vloží do databáze. V opačném případě script vrátí uživatele na formulář vytvářející novou položku se zobrazeným upozorněním o použití daného jména. Takové varování vypadá např. takto:



Obr. 11 : Varování o existenci uživatele se jménem admin

5.5.2 Úprava implementace mazání instancí zkušebních obvodů

Další drobný nedostatek, který jsem v aplikaci našel spočíval v tom, že při mazání instancí zkušebních obvodů aplikace nesmazala soubor s instancí zkušebního obvodu ze souborového úložiště. Tento nedostatek jsem vyřešil pomocí PHP funkce *exec()* a unixového příkazu *rm*. Konkrétně:

```
exec("rm $fileName"); // promenna $fileName obsahuje samozrejme nazev souboru
```

5.5.3 Implementace upozornění na špatné přihlášení

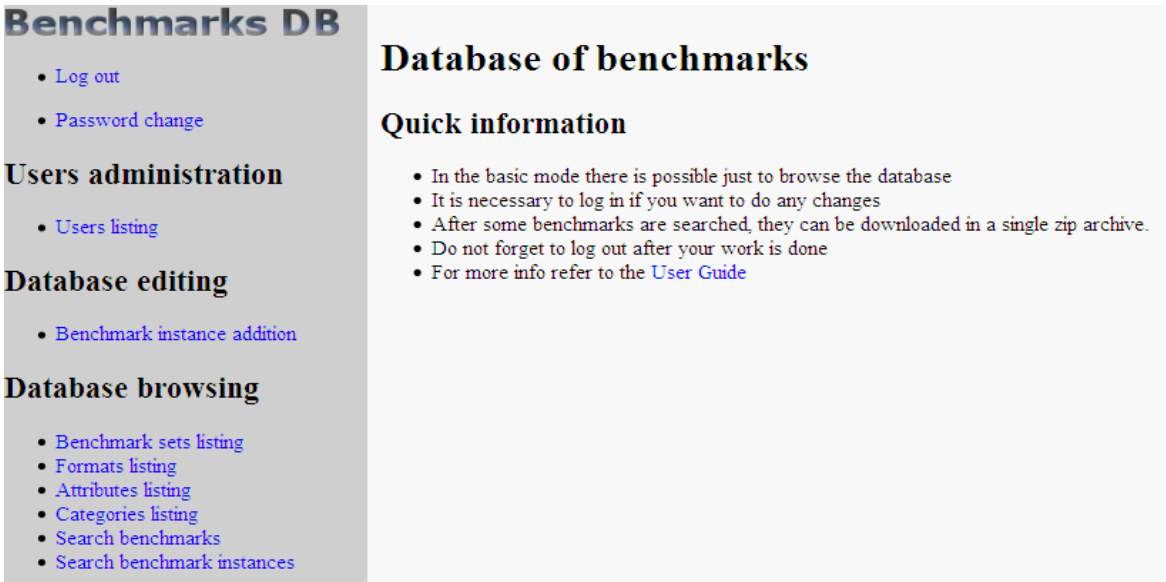
V původní aplikaci, pokud člověk při přihlašování do systému zadal špatné heslo nebo uživatelské jméno, nebyl na tuto skutečnost nijak upozorněn. Proto jsem tuto drobnost doplnil. Konkrétně jsem to vyřešil tak, že script kontrolující přihlašovací údaje v případě neúspěchu odešle na úvodní stránku příznak přes url o této skutečnosti. Úvodní stránka následně upozornění zobrazí.

5.5.4 Inplementace ověření vstupů ve vyhledávači instancí

Původní aplikace při zadání nesmyslných (např. alfnumerických) znaků do pole pro hodnotu atributu havarovala. Toto bylo potřeba ošetřit. Vyřešil jsem to velmi jednoduše. Konkrétně jsem do skriptu vložil otestování, zda sql příkaz, který byl sestaven pomocí uživatelem zadaných vstupů byl v pořádku. Pokud byl v nepořádku, nechal jsem zobrazit chybové hlášení. V opačném případě jsem nechal zobrazit relevantní výsledky.

5.6 Výsledná aplikace

Pro ilustraci jak bylo vyřešeno uživatelské rozhraní s novými funkcemi zde uvádím náhled hotové aplikace. Je vidět, že oproti původní verzi aplikace jsem drobně vylepšil design. V menu je částečně vidět, že jsou k dispozici nové funkce podle požadavků.



Obr. 12 : Úvodní stránka nové aplikace

6 Analýza bezpečnosti aplikace

V dnešní době Internetu jsou aplikace i data vystaveny mnoha bezpečnostním rizikům. Podle důležitosti aplikace a dat se kterými pracujeme je třeba tato rizika zhodnotit a zavést opatření, která sníží riziko poškození, nebo zneužití aplikace i dat.

Tato aplikace pravděpodobně nebude pro případné útočníky příliš lákavá, ale i přesto nebylo na bezpečnost aplikace úplně zapomenuto. Hlavní motivací je snaha zamezit aby případný útočník nevyužil aplikaci k ovládnutí počítače na kterém bude provozována. Jak již bylo dříve uvedeno, webové rozhraní je napsáno pomocí skriptovacího jazyka PHP. Při nešikovně psaném kódu můžeme učinit aplikaci velice snadno zranitelnou. Mezi nejběžnější útoky na webové aplikace patří:

- SQL injection
- Directory traversal
- Cross site scripting
- Denial of service attack

6.1 SQL injection

SQL injection je technika útoku, která využívá špatně ošetřené vstupy webové aplikace k ovlivňování SQL dotazu ve prospěch útočníka. Pro ochranu vstupů proti této technice je vhodné escapovat potenciálně nebezpečné znaky. Pro použití s databází PostgreSQL je pro bezpečné escapování v PHP funkce *pg_escape_string()*. Pro MySQL je to *mysql_real_escape_string()*. Více informací o těchto funkcích je v [5].

Na straně databáze je vhodné uživateli, pomocí kterého aplikace přistupuje k databázi, omezit práva (např. znemožnit mu mazání tabulek pokud to neovlivní fungování aplikace). Tímto opatřením omezíme možné škody, které by mohl útočník pomocí SQL injection napáchat.,

Podrobnější informace lze nalézt v [3].

6.2 Directory traversal

Pomocí útoku technikou directory traversal se útočník může, v případě nechráněného serveru nebo chybně napsaného scriptu, dostat k libovolnému souboru na serveru, nebo dokonce spustit škodlivý kód na serveru.

Typický chybně napsaný script je takový, který zobrazuje stránku z parametru URL. Např.:

http://www.server.com/view.php?page=shop.html

Pokud není script i server patřičně ošetřen, tak pomocí dosazení da parametru page může útočník způsobit škody.

Podrobnější informace lze nalézt v [3].

6.3 Cross site scripting

Metoda útoku cross site scripting využívá podobně jako SQL injection špatně ošetřených vstupů aplikace. Útočník může tohoto nedostatku využít k vložení vlastního kódu do stránek. Toho může využít např. k změně vzhledu, nebo rozbití webové aplikace. V nejhorším případě může útočník zfalšovat přihlašovací formulář a následně ukradnout citlivé přihlašovací údaje nic netušícího uživatele aplikace.

Pro obranu proti tomuto druhu útoku je vhodné filtrovat speciální HTML znaky ze vstupů. V php k tomuto dobře poslouží funkce *htmlspecialchars()*. Více informací o této funkci je možno najít na [5].

Podrobnější informace lze nalézt v [3].

6.4 Denial of service attack

Jedná se o metodu útoku především na webové aplikace, kdy útočník pomocí značného množství náhodných požadavků na server způsobí nefunkčnost, nebo přinejmenším značné zpomalení, webové aplikace pro řádné uživatele.

Obrana proti takovému typu útoků je poměrně obtížná. Nelze nikdy úplně zaručit, že je server a aplikace na něm chráněna na 100%. Prevencí proti DoS útoku je správně nakonfigurovaný server a efektivně napsaný kód aplikace.

Podrobnější informace lze nalézt na [3].

6.5 Ukládání uživatelských hesel

Původní aplikace ukládala hesla uživatelů do databáze. Toto řešení není úplně vhodné, neboť dává správci aplikace zbytečně přístup k heslům uživatelů. Ale i kdyby byl správce aplikace člověk na svém místě a hesla nezneužil, je toto neření nevhodné. Správce totiž těžko může garantovat bezpečnost databázového úložiště na 100%. Pokud by nějaký útočník získal přístup k databázi, logicky by získal i hesla uživatelů. A to zcela zbytečně.

Eliminace tohoto rizika je velice jednoduchá. Do databáze lze ukládat otisk hesla uživatele a při přihlašování kontrolovat otisk zadaného hesla s otiskem uloženým v databázi. Pro generování otisků byl použit algoritmus SHA1. Pro vytváření otisků je k dispozici v PHP funkce *sha1()* (více o ni v [5], která ze vstupního řetězce vygeneruje řetězec s otiskem v pseudokódu. Jeho implementace je tudíž velice jednoduchá.

6.5.1 SHA1

SHA1 je hashovací funkce, která nahrazuje dnes již z bespečnostních důvodů nevyhovující hashovací funkci MD5. Ze vstupních dat vytváří otisk (někdy též označovaný jako hash, fingerprint, kontrolní součet atp.) fixní délky 160ti bitů. Jeho hlavní vlastností je, podoně jako u algorimu MD5, že i jen málo odlišná vstupní data vytváří zcela jiný otisk. Např.:

pro řetězec „heslo“ je otisk „6e017b5464f820a6c1bb5e9f6d711a667a80d8ea“ (v pseudokódu)
pro řetězec „Heslo“ je otisk „894f36e5fe639267301de83d341819acc0a14d4b“ (v pseudokódu)

Sice bezpečnost algoritmu SHA1 byla odborníky zpochybněna, neboť obsahuje vlastnosti díky kterým existuje riziko, že útočník z otisku spočítá s běžnými výpočetními prostředky heslo uživatele v čase kratším než je čas potřebný k útoku hrubou silou, jedná se o rozumný kompromis mezi složitostí implementace a bezpečností. V současné době se k ukládání hesel běžně používá. Je ve vývoji několik bezpečnějších hashovacích funkcí, ale ty nejsou zatím běžně k dispozici. Další možností jak zvýšit bezpečnost uložených hesel je použití kombinaci několika hashovacích funkcí. Toto řešení jsem se ale rozhodl nepoužít.

Detailní informace lze nalézt v [3].

6.6 Ochrana přenášených dat

V prostředí Internetu si nikdy nemůžeme být jisti, zda někdo mezi naším počítačem a serverem se kterým komunikujeme neoprávněně neposlouchá. V případě, že aplikace pracuje s citlivými daty je vhodné komunikaci mezi serverem a uživateli šifrovat. Původní práce šifrování přenášených dat neřešila a ani já jsem se tím nezabýval, protože databáze zkušebních obvodů neobsahuje tak citlivá data. Vhodnou konfigurací webového serveru je možné šifrované spojení realizovat.

6.7 Oprávnění uživatelů

V nové verzi aplikace jsou uživatelská práva řešena shodně jako v původní aplikaci. Uživatelské účty mohou nabývat libovolnou kombinaci třech různých oprávnění. Jsou to:

- právo editovat databázi
- právo editovat uživatelské účty (kromě vlastního)
- právo měnit vlastní heslo

Je na správci aplikace, aby každý uživatel databáze zkušebních obvodů měl práva, která potřebuje, ale neměl zbytečně ta práva, která nepotřebuje.

Podrobnější informace lze nalézt v [1], kapitolách 3.11 a 6.

7 Testování aplikace

Každá jen trochu rozsáhlejší aplikace musí být systematicky testována. Testování patří mezi důležité fáze vývoje jakéhokoliv software. Rozlišují se tyto metodiky:

- black box testování - testující nemá informace o tom jak uvnitř pracuje testovaný systém, zaměřuje se pouze na vstupy a výstupy
- white box testování - testující při testování kontroluje krom vstupů a výstupů údaje uvnitř systému, nebo tyto údaje účelově ovlivňuje

V aplikaci jsem nenašel vhodné jednotky pro jednotkové testování a v požadavcích na aplikaci není schopnost pracovat pod vysokou zátěží. Proto jsem se zaměřil především na systémové black box testování.

7.1 Skupiny scénářů systémových testů

7.1.1 Instalace aplikace

Předpoklady: Aplikace není naistalována (databáze aplikaci určená neobsahuje žádné záznamy, složka aplikaci určená neobsahuje žádné soubory)

Akce: Zakopírujeme všechny potřebné soubory na místo určení, pomocí metakonfiguračního souboru vygenerujeme konfigurační soubory a spustíme scripty pro vytvoření tabulek a naplnění tabulek základními daty.

Úspěch: Aplikaci se podařilo nainstalovat včetně všech jejich součástí.

Neúspěch: Aplikaci nebo její části se nepodařilo nainstalovat.

7.1.2 Testování automatického plnění databáze

Předpoklady: aplikace je správně nainstalována, nakonfigurována a neobsahuje data, která chceme do aplikace pomocí automatického plnění přidat

Akce: spustíme script pro automatické plnění databáze se všemi potřebnými parametry

Úspěch: všechna data, která jsme pomocí parametrů scriptu předložili, byla úspěšně vložena do databáze a všechny soubory týkající se vkládaných dat byly nakopírovány do příslušné složky (defaultně benchmarkFiles)

Neúspěch: do databáze nebyla vložena všechna očekávaná data, nobo všechny očekávané soubory nebyly nakopírovány do příslušné složky

7.1.3 Testování funkce webového rozhraní - prohlížení dat

7.1.3.1 Vypsání seznamu sad zkušebních obvodů a podrobností k nim

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vypsat seznam sad zkušebních obvodů, podrobnosti k jednotlivým sadám zkušebních obvodů, vyzkoušíme přímý link do vyhledávací sekce.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.2 Vypsání seznamu formátů a podrobností k nim

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vypsat seznam formátů, podrobnosti k jednotlivým formátům, vyzkoušíme přímý link do vyhledávací sekce.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.3 Vypsání seznamu atributů a podrobností k nim

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vypsat seznam atributů, podrobnosti k jednotlivým neuzamčených atributům.

Úspěch: Aplikace zobrazila očekávaná data. Uzamčené atributy není možné editovat ani mazat.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala. Uzamčené atributy bylo možné smazat nebo editovat.

7.1.3.4 Vypsání seznamu kategorií a podrobností k nim

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vypsat seznam kategorií a podrobností k nim.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.5 Vyhledávání zkušebních obvodů podle zadaných kritérií

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vyhledat zkušební obvody na základě různorodých kritérií.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.6 Vyhledávání instancí zkušebních obvodů podle zadaných kritérií

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vyhledat instance na základě různorodých kritérií.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.7 Zobrazení výsledků dané instance

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Zobrazíme výsledky dané instance.

Úspěch: Aplikace zobrazila očekávaná data.

Neúspěch: Aplikace vypsala nesprávná data, nebo havarovala.

7.1.3.8 Stažení souboru instance a jehího výsledku

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Stáhneme soubor instance a soubory jejích výsledků.

Úspěch: Podařilo se stáhnout všechny stahované soubory.

Neúspěch: Aplikace neposkytla správné, nebo všechny soubory, nebo havarovala.

7.1.3.9 Agregované stažení vyhledaných instancí zkušebních obvodů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vyhledat instance na základě různorodých kritérií. Poté stáhneme archív se soubory instancí.

Úspěch: Podařilo se stáhnout archív obsahující všechny relevantní soubory instancí.

Neúspěch: Archív se nepodařilo stáhnout, nebo neobsahoval všechny relevantní soubory instancí.

7.1.3.10 Agregované smazání vyhledanych instancí zkušebních obvodů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Necháme vyhledat instance na základě různorodých kritérií. Poté jedním kliknutím smažeme všechny relevantní instance a jejich soubory.

Úspěch: Podařilo se smazat všechny relevantní instance a jejich soubory.

Neúspěch: Všechny relevantní instance nebo jejích soubory se nepodařilo smazat.

7.1.4 Testování funkce webového rozhraní - vkládání dat

7.1.4.1 Vložení nové instance zkušebního obvodu s manuálním vypněním atributů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Pomocí formuláře vložíme novou instanci do aplikace. Checkbox s možností automaticky vyplnit atributy nezaškrtneme.

Úspěch: Instanci se podařilo úspěšně vložit. Je jak v databázi, tak v souborovém úložišti.

Neúspěch: Nová instance není správně uložena buďto v souborovém úložišti, nebo v databázi.

7.1.4.2 Vložení nové instance zkušebního obvodu s automatickým vypněním atributů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Pomocí formuláře vložíme novou instanci do aplikace. Zvolíme možnost automaticky vyplnit atributy.

Úspěch: Instanci se podařilo úspěšně vložit. Je jak v databázi, tak v souborovém úložišti. Atributy byly správně vyplněny.

Neúspěch: Nová instance není správně uložena buďto v souborovém úložišti, nebo v databázi, popř. byly špatně vyplněny atributy.

7.1.4.3 Vložení nového výsledku instance

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: K dané instanci vložíme instanci výsledku.

Úspěch: Instanci výsledku se úspěšně podařilo vložit.

Neúspěch: Instanci výsledku se nepodařilo vložit.

7.1.4.4 Editace a smazání u zkušebních obvodů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme zkušebních obvodů a poté ho smažeme.

Úspěch: zkušebních obvodů se podařilo úspěšně jak editovat tak smazat.

Neúspěch: zkušebních obvodů se nepodařilo buďto editovat, nebo smazat.

7.1.4.5 Editace a smazání formátu instancí zkušebních obvodů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme formát instancí a poté ho smažeme.

Úspěch: Formát instancí zkušebních obvodů se podařilo jak editovat tak smazat.

Neúspěch: Formát instancí zkušebních obvodů se nepodařilo buďto editovat, nebo smazat.

7.1.4.6 Editace a smazání formátu výsledků

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme formát instancí a poté ho smažeme.

Úspěch: Formát výsledků se podařilo jak editovat tak smazat.

Neúspěch: Formát výsledků se nepodařilo buďto editovat, nebo smazat.

7.1.4.7 Editace a smazání kategorií instancí

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme kategorii instancí a poté ji smažeme.

Úspěch: Kategorii instancí se podařilo jak editovat tak smazat.

Neúspěch: Kategorii instancí se nepodařilo buďto editovat, nebo smazat.

7.1.4.8 Editace a smazání atributu

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme atribut a poté ho, pokud není zamčený, smažeme.

Úspěch: Atribut se podařilo jak editovat tak (pokud nebyl zamčený) smazat.

Neúspěch: Atribut se nepodařilo buďto editovat, nebo smazat.

7.1.4.9 Editace a smazání zkušebního obvodu

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme zkušební obvod a poté ho smažeme.

Úspěch: Zkušební obvod se podařilo jak editovat tak smazat.

Neúspěch: Zkušební obvod se nepodařilo buďto editovat, nebo smazat.

7.1.4.10 Editace a smazání instance zkušebního obvodu

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme instanci zkušebního obvodu a poté ji smažeme.

Úspěch: Instanci zkušebního obvodu se podařilo jak editovat tak smazat.

Neúspěch: Instancí zkušebního obvodu se nepodařilo buďto editovat, nebo smazat.

7.1.4.11 Editace a smazání výsledku instance zkušebního obvodu

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Nejprve editujeme instanci výsledku zkušebního obvodu a poté ji smažeme.

Úspěch: Instanci výsledku se podařilo jak editovat tak smazat.

Neúspěch: Instanci výsledku se nepodařilo buďto editovat, nebo smazat.

7.1.5 Testování funkce webového rozhraní - správa uživatelských účtů

7.1.5.1 Editace vlastního hesla

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Přihlásíme se jako daný uživatel a změníme mu pomocí formuláře heslo.

Úspěch: Heslo bylo úspěšně změněno.

Neúuspěch: Heslo se nepodařilo změnit, nebo aplikace havarovala.

7.1.5.2 Vypsání uživatelů

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Zobrazíme evidenci uživatelů.

Úspěch: Evidence uživatelů se správně a kompletně zobrazila.

Neúspěch: Evidence uživatelů se nezobrazila správně, nebo aplikace havarovala.

7.1.5.3 Editace uživatele

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Pomocí formuláře změníme některému z uživatelů heslo, nebo přístupová práva.

Úspěch: Změny se podařilo uložit.

Neúspěch: Změny se nepodařilo uložit.

7.1.5.4 Smazání uživatele

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Uživatel má dostatečná přístupová práva.

Akce: Smažeme některého z evidovaných uživatelů.

Úspěch: Uživatel byl úspěšně smazán.

Neúspěch: Uživatele se smazat nepodařilo.

7.1.6 Testování funkce webového rozhraní - uživatelská práva

V této fázi testování jsem se zaměřil nejen na to, zda každý přihlášený uživatel může provádět všechny akce které mu přísluší, ale zejména zda uživatel nemůže provádět akce, které mu nepřísluší.

Předpoklady: Aplikace je správně nainstalována a nakonfigurována. Jsou vytvořeny různé uživatelské účty s různými přístupovými právy.

Akce: Přihlášení se pod různými uživatelskými účty a provádění povolených i nepovolených akcí.

Úspěch: Aplikace všech uživatelům povolí vše na co mají práva a zakáže vše na co práva namají.

Neúspěch: Aplikace některému z uživatelů buďto dovolí něco ne co nemá práva, nebo nedovolí něco na co práva má.

7.2 Ohlédnutí za testováním

Testování pomohlo k zajištění správné funkce a spolehlivosti aplikace. Nyní aplikace pracuje v rámci těchto scénářů správně. I když jsem se snažil, aby scénáře bezezbytku pokryly všechny možné situace, nelze samozřejmě stoprocentně zaručit, že je aplikace zcela bezchybná.

Aplikaci jsem testoval pouze na systému PC s nainstalovaným OS Linux Ubuntu „Hardy Heron“. Pilotní nasazení bylo na systému s OS FreeBSD. Je málo pravděpodobné, že by aplikace na jiných systémech pracovala nesprávně, zaručit to ale nelze.

8 Závěr

Cílem této práce bylo navázat na diplomovou práci Jaroslava Pacholy z roku 2007. Především bylo jejím cílem rozšířit webovou aplikaci o zadané možnosti a opravit drobné nedostatky původní webové aplikace.

Serverová část nové verze aplikace je postavena na stejných technologiích jako aplikace původní. To znamená že využívá databázového stroje PostgreSQL a scriptovacího jazyka PHP. Klientská část nové aplikace využívá navíc oproti původní verzi i technologii AJAX. Uživateli to přináší větší komfort při práci s aplikací. Komfort ovládání byl dále vylepšen drobnými designovými úpravami.

Myslím že práce splnila požadavky, které na ni byly kladeny. Navíc řeší i několik drobností, které jsem odhalil v průběhu vypracovávání práce.

Přestože jsem databázi zkušebních obvodů v mnoha ohledech zdokonalil, zbývá samozřejmě ještě hodně prostoru k dalším vylepšením. Například by mohlo být užitečné dát uživatelům ještě více nástrojů na hromadnou správu instancí zkušebních obvodů i jejich výsledků. Užitečná by mohla také být možnost řazení výsledků hledání instancí podle zvolených kritérií. Dále by také aplikace mohla evidovat který uživatel provedl kterou akci. Správci aplikace by jistě usnadnilo práci zjednodušení procesu instalace aplikace, nebo uzpůsobení aplikace tak, aby ji bylo možno provozovat na běžných komerčních webových serverech. Praktické používání aplikace jistě přinese celou řadu dalších podnětů k vylepšení.

Seznam použitých zdrojů a literatury

Při tvorbě této bakalářské práce a programování webové aplikace jsem vycházel z následujících zdrojů:

Neinternetové zdroje

[1] Databáze benchmarkových obvodů
Diplomová práce, ČVUT FEL katedra počítačů, Jaroslav Pachola, 2007

[2] PHP Programujeme profesionálně
Computer Press 2003, Kolektiv autorů

Zdroje na internetu

[3] Wikipedie – Otevřená encyklopedie
www.wikipedia.org

[4] W3Schools – online návody pro vytváření webových stránek
www.w3schools.com

[5] oficiální stránky projektu PHP
www.php.net

A Obsah přiloženého CD

- benchdb – adresář obsahující veškeré zdrojové texty databáze zkušebních obvodů
- benchmarks – adresář obsahující některé sady zkušebních obvodů
- installationguide – adresář obsahující instalační příručku ve formátu HTML
- userguide – adresář obsahující uživatelskou příručku ve formátu HTML
- benchdb.odt – text bakalářské práce ve formátu odt
- benchdb.pdf – text bakalářské práce v platformově nezávislém formátu pdf

B Seznam použitých zkratek

- HTML – HyperText Markup Language
- CSS – Cascading Style Sheet
- AJAX – Asynchronous Java script & Xml
- PHP – Personal Home Pages
- OS – Operační Systém
- PC – Personal Computer
- SHA1 – Secure Hash Algorithm
- MD5 – Message Digest algorithm
- SQL – Structured Query Language
- DoS – Denial of Service
- URL – Uniform Resource Locator