SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Evidenčné číslo: *FCHPT-5414-49748*

INTERNETOVÝ MODUL PRE MODELOVANIE A SIMULÁCIE PROCESOV

DIPLOMOVÁ PRÁCA

2013 **Bc. Juraj Malinič**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

INTERNETOVÝ MODUL PRE MODELOVANIE A SIMULÁCIE PROCESOV

DIPLOMOVÁ PRÁCA

FCHPT-5414-49748

Študijný program: automatizácia a informatizácia v chémii a potravinárstve

Číslo študijného odboru[1]: 2621

Názov študijného odboru: 5.2.14 AUTOMATIZÁCIA

Školiace pracovisko: Ústav informatizácie, automatizácie a matematiky

Vedúci záverečnej práce/školiteľ: Ing. Ľuboš Čirka, PhD.

Bratislava 2013 **Bc. Juraj Malinič**

1 Prvé štvorčíslie kódu podľa vyhlášky Štatistického úradu Slovenskej republiky č. 114/2011 Z.z., ktorou sa vydáva Štatistická klasifikácia odborov vzdelania

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2012/2013
Evidenčné číslo: FCHPT-5414-49748

STU FCHPT

ZADANIE DIPLOMOVEJ PRÁCE

Študent:	**Bc. Juraj Malinič**
ID študenta:	49748
Študijný program:	automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14 automatizácia
Vedúci práce:	Ing. Ľuboš Čirka, PhD.

Názov práce: **Internetový modul pre modelovanie a simuláciu procesov**

Špecifikácia zadania:

Cieľom práce je upraviť a doplniť knižnicu modelov technologických procesov.

Úlohy:
1. Rešerš dostupných riešení virtuálnych laboratórií.
2. Naštudovať jazyky PHP (OOP) a JavaScript (AJAX, jQuery).
3. Naštudovať architektúru MCV.
4. Programovo realizovať moduly v PHP a MySQL.
5. Overiť funkčnosť a vypracovať postup inštalácie.
6. Vypracovať dokumentáciu.

Rozsah práce: 50

Riešenie zadania práce od: 18. 02. 2013
Dátum odovzdania práce: 25. 05. 2013

L. S.

Bc. Juraj Malinič
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

POĎAKOVANIE

Týmto sa chcem úprimne poďakovať školiteľovi Ing. Ľubošovi Čirkovi PhD., za cenné rady, ktoré mi pomohli pri programovaní modulu a taktiež pri písaní tejto práce. Zároveň chcem poďakovať svojej sestre Veronike Maliničovej, za pomoc pri kontrole pravopisu. Rovnako aj svojim rodičom za podporu, ktorú mi poskytli v ťažkých chvíľach počas štúdia.

Abstrakt

Juraj Malinič: Internetový modul pre modelovanie a simulácie procesov. [Diplomová práca].
Slovenská technická univerzita v Bratislave. Vedúci: Ing. Ľuboš Čirka, PhD.

Diplomová práca obsahuje stručný opis modulu pre simulovanie zásobníka kvapaliny, ktorý je naprogramovaný v jazyku JavaScript.

Postup simulácie a návod na používanie modulu pre simulácie. Taktiež obsahuje stručný popis frameworku, na ktorom je modul postavený a technológie, ktoré boli použité pri programovaní aplikácie. V posledných kapitolách je popísané administrátorské rozhranie a postup inštalácie aplikácie na server.

Abstract

Juraj Malinic: Internet module for modeling and simulation of processes. [Master thesis].
Slovak Technical University in Bratislava. Supervisor: Ing. Lubos Cirka, PhD.

The thesis presents a description of the module for simulating fluid tank which is programmed in JavaScript.

Process simulation and instructions for the simulation module. It also contains a description of the framework on which the module is built and technologies that have been used to code aplication. The last chapter describes administration interface and how to install applications on the server.

Obsah

Zoznam obrázkov

Zoznam skratiek

PHP – Profesional Home Page/Form interpreter

HTML – HyperText Markup Language

CSS – Cascade Style Sheet

MVC – Model View Controller

pub. – public, verejná premenná vytvorená v JavaScripte v module pattern funkcii.

_. - privátna premenná vytvorená v JavaScripte v module pattern funkcii.

Úvod

Zmyslom virtuálneho laboratória je sprístupniť technologické procesy prostredníctvom internetu. Existuje niekoľko spôsobov ako si študent môže vyskúšať prácu s nejakým technologickým procesom.

V najlepšom prípade príde do laboratória, kde spraví potrebné merania a vyskúša si prácu so zariadením. Výhodou tohto spôsobu je, že si študent vyskúša reálne situácie, ktoré pri simulovaní nemusia nastať. Nie každý však má prístup do laboratória, taktiež nie je možné vstupovať do laboratória v ľubovoľnom čase.

Druhou možnosťou je pristupovať k procesom prostredníctvom vzdialeného prístupu. V tomto prípade už je reč o vzdialenom laboratóriu. Výhodou vzdialeného prístupu je, že študent nemusí byť reálne prítomný v laboratóriu. Teda prístup k procesu je oveľa flexibilnejší. To znamená, že študenti už môžu pristupovať k procesom aj mimo vyučovacieho času. Avšak aj v tomto prípade je nutnosť existencie procesu. Ďalšou nevýhodou je, že v tom istom čase môže na jednom procese pracovať len jeden študent.

Existuje však možnosť ako si študenti môžu vyskúšať prácu s procesom aj bez prístupu do laboratória, či čakania na prístup cez vzdialený prístup. Tou možnosťou je využitie virtuálneho laboratória – internetových modulov pre simulácie. V tomto prípade môže k jednému procesu pristupovať viacero užívateľov v ľubovoľnom čase.

Internetový modul vytvorený v tejto diplomovej práci bude vytvorený prostredníctvom HTML 5, jQuery a CSS 3. Vďaka týmto technológiám budú jednotlivé výpočty bežať na strane prehliadača, teda aj po stiahnutí stránky bude možné meniť parametre a simulovať. Teda užívateľ nebude viazaný internetovým pripojením.

Jednotlivé simulačné moduly bude možno vkladať prostredníctvom administrátorského rozhrania. V tomto rozhraní bude taktiež možné vytvárať položky hlavnej ponuky a upravovať titulnú stránku ako aj ostatné statické stránky.

1 Technologické riešenia

Existujúce moduly pre simulácie sú riešené prostredníctvom rôznych technológii. Každé riešenie má svoje pre a proti. Nie je možne jednoznačne povedať, ktoré je najlepšie. Moduly riešené prostredníctvom jazyka PHP vyžadujú existenciu webového servera, na ktorom prebiehajú výpočty, keďže PHP je jazyk bežiaci na strane servera. Z tohto dôvodu sa vo väčšine prípadov jedna o offline výpočty. Užívateľ zadá parametre. Na serveri sa vypočítajú parametre. Potom sa výsledky vykreslia užívateľovi. Ak sa užívateľ rozhodne zmeniť niektorý parameter, modul začne počítať s novými parametrami odznova.

Aj v prípade PHP jazyka je možne realizovať výpočty v reálnom čase, a to prostredníctvom AJAX-u. JavaScriptom sa na server posielajú požiadavky a server vracia aktuálne hodnoty. V takom prípade, ak študent zmení parameter výpočtu, zmenia sa parametre v reálnom čase a server pokračuje vo výpočtoch. Nevýhodou tohto spôsobu je riziko prerušenia spojenia prehliadača a servera. Ďalšou nevýhodou je samotne PHP, ktoré nie je synonymom rýchlosti a úspornosti. Veľmi ľahko môže vzniknúť situácia, kedy skript vyčerpá pamäť, ktorú mu server pridelil a dôjde k prerušeniu výpočtu.

Aby odpadli tieto riziká, je možne výpočet prenechať prehliadaču. Jedným z možných riešení ako preniesť ťarchu výpočtu zo servera na prehliadač, je jazyk JavaScript. Aj pri použití jazyka JavaScript existuje riziko prerušenia simulácie. Javascript je dynamický jazyk s dynamikou na strane klienta, a na výpočet využíva práve prostriedky klienta. V prípade, že prehliadač beží na pomalom počítači s nedostatočným výkonom, nemusí zvládnuť náročnejšie výpočty. Simuláciu prostredníctvom Javascriptu je možné rozšíriť o animácie a tým zatraktívniť celý modul pre študenta.

2 Použité technológie

2.1 PHP

2.1.1 Objekty, metódy, premenné

[1] [2]

PHP alebo Profesional Home Page/Form interpreter je jeden z najrozšírenejších scriptovacích jazykov. Prvopočiatky PHP siahajú do roku 1944, kedy Rasmus Lendorf napísal počítadlo prístupov na webové stránky v Perl-e. PHP prebralo štruktúru z Perl-u a C-éčka.

Aj keď je PHP jednoduchý jazyk, má v sebe množstvo integrovaných funkcii a taktiež možnosť pracovať s objektmi.

Je vecou názoru či objekty v PHP majú alebo nemajú zmysel. V prípade, že sa jedná o krátky program, využitie objektov je náročne a v skutočnosti je jednoduchšie naprogramovať ho procedurálne. Ako náhle sa pracuje na rozsiahlom projekte, sú objekty na nezaplatenie.

Objekty sú štruktúry obsahujúce rôzne premenné a funkcie. Aj napriek tomu, že sa pracuje s objektmi, v kóde sa sa slovo *object* nevyskytuje. Na zadefinovanie objektu sa používa slovo *class* (trieda). Trieda je akási šablóna, podľa ktorej sa objekty vytvárajú.

```
class pes
{
        private $meno;
        construct($m)
        {
                $this -> meno = $m;
```

```
        }
        public function stekaj()
        {
                echo 'haf haf';
        }
        public static function stekniRaz()
        {
                echo 'haf';
        }
}
```

Je definovaná trieda *pes*. V tejto triede je definovaná privátna premenná *meno*. Samotný objekt je vlastne inštancia triedy, teda:

```
$dunco = new pes('dunčo');
$dunco -> stekaj();
```

Výstup z takto napísaného programu bude: „haf haf“.

Premenné rovnako ako funkcie (metódy) môžu byť statické, verejne, privátne. Rozdiel medzi nimi je v spôsobe volania. Zatiaľ čo verejné môžu byť volané z vnútra, ale aj z vonku triedy, privátne môžu byť volané len z vnútra triedy. Existujú aj chránené metódy. Tie sa na rozdiel od privátnych metód, môžu volať v zdedených triedach. Nemôžu sa však volať z vonku.

V prípade statickej premennej alebo statickej funkcie, nie je potrebné vytvárať inštanciu.

```
pes::stekniRaz();
```

Výstupom takéhoto programu bude „haf“.

2.1.2 Dedičnosť

[2]

Umožňuje odvodiť triedu od inej, pričom odvodená trieda môže obsahovať nové metódy alebo upravovať existujúce metódy. Existujúce metódy sa môžu, ale aj nemusia v zdedenej triede používať. V triedach sa pri dedení môže prepisovať názov metódy. V originále sa to nazýva 'method overriding'. Netreba si to pliesť s 'overloadingom', čo znamená preťažovanie.

2.1.3 Abstrakcia

[2]

Abstrakcia je zameranie sa na kľúčové vlastnosti prvku. V OOP sa to využíva pri abstraktných triedach. Takáto trieda sa sama o sebe nedá vytvoriť. Vytvoriť sa dajú len zdedené triedy.

2.1.4 Polymorfizmus

[2]

Polymorfizmus je mechanizmus, ktorý umožňuje objektom rôznych typov odpovedať na volanie rovnakej metódy rôznym spôsobom.

Sú definované automobily, ktoré sú v podstate objekty. Tieto objekty sú umiestnené na nejakom parkovisku. Dispečer potrebuje aby sa Automobil A presunul na miesto Z, automobil B na miesto Y a automobil C na miesto X. Najjednoduchšie by bolo, zavolať metódu autoChod(), ktorá by odoslala správne auto na správne miesto.

Polymorfizmus zabezpečí, že sa pri volaní zistí do akej triedy automobil patrí, a každé auto sa bude správať tak, ako má.

2.1.5 Objekty v ľudskej reči

[2]

Automobil je objekt reálneho sveta, vodič nastúpi do automobilu a naštartovaním ho inicializuje. Pri riadení má k dispozícii volant, pedále, páky. Po stlačení akceleračného pedálu automobil začne zrýchľovať, po stlačení brzdového pedálu začne automobil brzdiť. Rovnakým spôsobom pracujú aj objekty v PHP. Pri vytvorení inštancie sa objekt inicializuje. Po zavolaní príslušnej metódy, bude objekt vykonávať naprogramované úkony. Ako je to v príklade s automobilom, začne zrýchľovať alebo spomaľovať.

2.1.6 MVC

[3]

MVC alebo Model-view-Controller, je softvérová architektúra, ktorá oddeľuje prezentačnú vrstvu, dátový model a riadiacu logiku do troch samostatných vrstiev, tak že zmena jednej nemá vplyv alebo má len minimálny vplyv na ostatné.

Aplikácie využívajúce MVC vyžadujú vytvorenie troch komponentov:

- modelu, ktorý reprezentuje informačnú vrstvu, s ktorou aplikácia pracuje.
- view-u, ktorý prevádza informácie reprezentované modelom, do podoby vhodnej pre prezeranie užívateľom.
- kontrolór, reaguje na udalosti a požaduje informácie od modelu, ktoré potom predáva view-u.

2.2 JavaScript a JQUERY

2.2.1 JavaScript

[4]

JavaScript je skriptovací jazyk, ktorý sa používa hlavne na tvorbu webových stránok. Tvorcom jazyka JavaScript je Brendan Eich. Vyvinutý bol pod názvom Mocha, neskôr premenovaný na LiveScript. Pred uverejnením jazyka bol jeho názov zmenený na JavaScript, najmä kvôli vtedajšej popularite jazyka Java. Napriek tomu, že jazyk nesie v názve slovo Java, má s jazykom Java len málo spoločné. Autor pôvodne vychádzal z jazyka Self.

Microsoft vyvinul vlastnú verziu s podobnou syntaxou pod názvom Jscript, no táto verzia nedodržuje špecifikáciu ECMAScript. Preto dochádza k nepresnostiam pri prekladaní kódu internetovými prehliadačmi IE nižších verzii a ostatnými modernými prehliadačmi.

2.2.2 jQuery

[5]

jQuery je JavaScriptová knižnica optimalizovaná takmer na všetky najpoužívanejšie prehliadače. jQuery kladie dôraz na interakciu JavaScriptu s HTML. jQuery bolo vydané v januári 2006 a jeho autorom je John Resig. Je najpoužívanejšou JavaScriptovou knižnicou. jQuery je šírene pod MIT licenciou. Jeho syntax je navrhnutá pre čo najjednoduchšiu navigáciu dokumentu, výber DOM elementov, spravovanie udalostí, vývoj AJAX aplikácii a tvorbu animácii.

2.2.3 Module Pattern

[6]

Module Pattern je návrhový vzor slúžiaci na implementáciu konceptu softvérových modulov do jazyka, ktorý ich nepodporuje alebo podporuje len čiastočne. Je to akési zaobalenie funkcii, kde sú funkcie a premenné obalené v jednej funkcii. V takejto funkcii môžu pristupovať vnorené funkcie do premenných z materskej funkcie. Na začiatku sa definuje objekt v JavaScripte napríklad:

```
if(typeof DOPRAVA == 'undefined')
  var DOPRAVA={};
```

Do takto zadefinovaného objektu sa môžu vkladať ďalšie objekty ľubovoľnej hĺbky. V objekte doprava existujú objekty nákladná, osobná a v nich sú objekty auta. Konkrétna funkcia alebo objekt auto sa potom zapisuje ako anonymná funkcia vložená do premennej resp. do objektu.

```
DOPRAVA.Osobna.Auto.Jaguar = function(){
//obsahfunkcie
};
```

V takto zadefinovanej funkcii sa vytvoria premenné *pub* a _, do ktorých sa budú definovať verejne a privátne metódy alebo premenné. Na konci funkcie je return, ktorý vráti premennú pub. V HTML sa potom pripojí súbor s funkciou, v ktorom sa táto funkcia rovno aj volá. Vďaka takémuto volaniu má potom programátor prístup k pub funkciám aj z vonku. Funkcie môžu byť volané z HTML prostredníctvom značky <script>, prípadne z nejakej udalosti.

Ak v objekte DOPRAVA.Osobna.Auto.Jaguar existuje funkcia pub.zapniSvetla() a programátor chce, aby po kliknutí na tlačítko sa na aute rozsvietili svetlá, tak ju zavolá nasledovne:

<a href="javascript:void();"
onClick="DOPRAVA.Osobna.Auto.Jaguar.zapniSvetla();">...

Na stránke môžu byť naraz inicializované desiatky funkcii a pritom pri jednotlivých udalostiach sa bude pristupovať presne k tej, ktorá je potrebná.

2.3 HTML 5

2.3.1 Popis

[7]

Web sa neustále zlepšuje a vyvíja. Nové a iniciatívne weby vznikajú každý deň a tým posúvajú hranice HTML stále ďalej a ďalej. Developeri a poskytovatelia obsahu stále hľadajú nové technológie, aby mohli zabezpečiť pokročilé funkcie a jednoduchšie zaobchádzanie so stránkami pre užívateľa. Sú zrážaní a obmedzovaní jazykom a prehliadačmi. Práve preto od roku 2004 spoločnosti W3C HTML WG a WHATWG začali vyvíjať HTML 5.

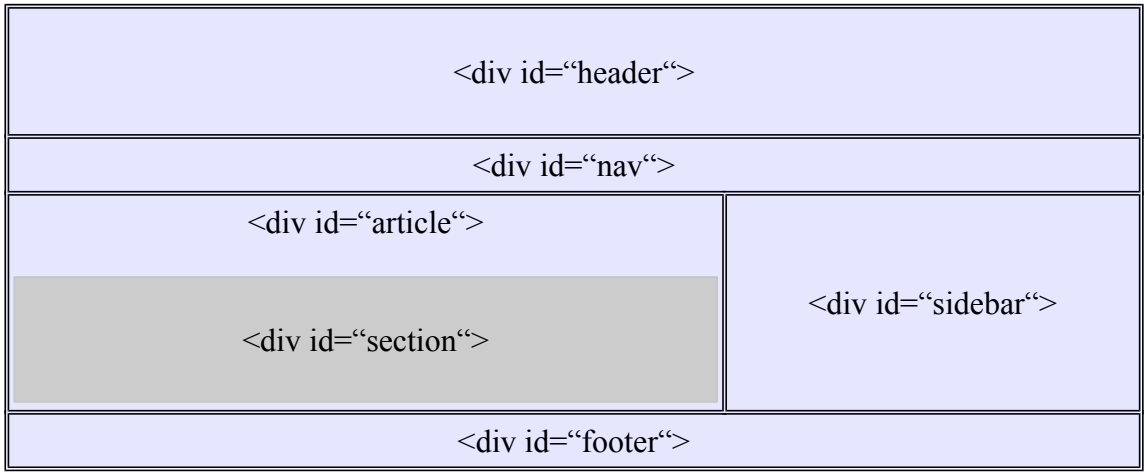
HTML 5 zavádza a vylepšuje širokú škálu funkcii. Ide o nástroje na pokročilú prácu s formulármi, rôzne API a multimédia. Na snahách W3C sa podieľali aj tvorcovia najpoužívanejších prehliadačov Safari, Mozilla, Opera, Chrome, IE.

2.3.2 Porovnanie s xHTML

[8]

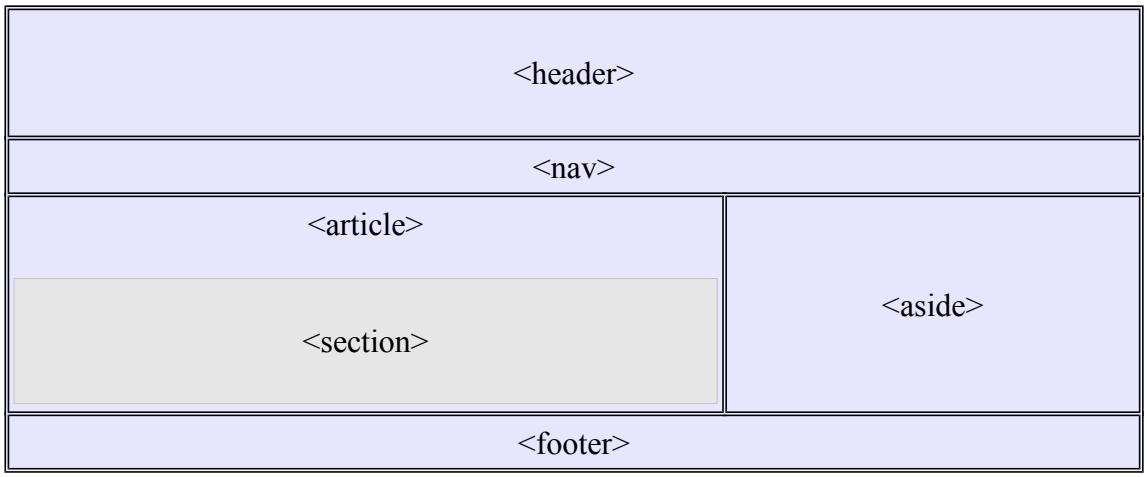
HTML 5 prichádza s celou sadou nových elementov, ktoré výrazne uľahčujú štruktúrovanie stránok. Drvivá väčšina stránok sa skladá z časti ako hlavička, pätička, nejaké stĺpce, navigačná lišta a podobne. Takéto rozdelenie pri starších verziách HTML realizovala prostredníctvom značky * ,* v ktorej bol atribút *id* prípadne *class* slúžiaci

na identifikáciu, že daný odsek je vlastne hlavička stránky (obr. 1). HTML 5 zaviedlo nové elementy, ktoré lepšie opisujú časti stránky, a teda na označenie hlavičky stačí použiť nový element *header* (obr. 2).



Obr. 1: Rozloženie stránky pomocou HTML 4.x

Takto rozložená stránka, zložená z divov a identifikátorov, môže zaberať zhruba 7 KB bez textu a formátovania. Prepísaním tejto stránky do HTML 5 sa jej veľkosť zníži na 5 KB.



Obr. 2: Rozloženie stránky pomocou HTML 5

Na obr. 2 je rovnaká stránka ako na obr. 1, len prepísaná do HTML 5. Na prvý pohľad vidno, že kód je jednoduchší a prehľadnejší. Má jednoduchšiu štruktúru a použité značky sú samo opisujúce. Použitím HTML 5 sa nielen uľahčuje práca, ale v konečnom dôsledku sa zmenšuje aj veľkosť dokumentu.

Okrem týchto formátovacích značiek prinieslo HTML 5 aj rôzne multimediálne značky, napríklad značka <*video*>. Vkladanie videa na stránku nebolo nikdy jednoduchšie, stačí správne nastaviť atribúty značky a nechať prehliadač nech si zvoli potrebné užívateľské rozhranie.

2.4 CSS 3

[9]

Jazyk HTML nastaví obsah stránky. Samotný obsah nie je veľmi príťažlivý. Jednotlivým častiam stránky treba dať určitú formu, teda to, ako bude stránka v konečnom dôsledku vyzerať. Formu stránky definujú kaskádové štýly, alebo v skratke CSS. Rovnako ako HTML 5 aj CSS 3 prináša veľa nových možností. Medzi najdôležitejšie patria 3D efekty, animácie, CSS Sprite, textové efekty a veľa ďalších.

V starších verziách CSS sa mohlo definovať, aby pri prechode myšou element zmenil napríklad farbu. Bola to však okamžitá zmena, teda ak bol definovaný červený obdĺžnik a užívateľ ukázal na tento obdĺžnik myšou farba sa zmenila na modrú. V CSS 3 nastavením prechodu sa tento istý úkon animuje a farba sa nezmení okamžite, ale plynule prejde od červenej až po modrú, podľa nastavenej dĺžky prechodu.

3 Framework

3.1 Základné časti

Framework, ktorý tvorí jadro modulu pre simulácie, bol vytvorený špeciálne pre potreby tohto modulu. Pri tvorbe modulu bola striktne dodržaná architektúra „Model 2“, alebo MCV. V základnom priečinku sú vytvorené priečinky z jednotlivými časťami frameworku. Názvy priečinkov sú samo-opisujúce. V priečinku model sú umiestnené triedy modelu, v priečinku controller sú umiestnené metódy kontrolóra a v priečinku view sú umiestnené šablóny, štýly, obrázky a JavaScripty. Okrem týchto priečinkov sa v základnom priečinku nachádza aj priečinok config. V tomto sú uložené konfiguračné súbory pre auto-load, pripojenie k databáze a iné.

3.2 Auto-load

Pri načítaní aplikácie sa ako prvý pripojí skript *auto.php*. Úlohou tohto skriptu je pripojenie všetkých tried k aplikácii. V priečinku *config* sa nachádza konfiguračný skript, v ktorom je jedno jedine pole. Do tohto poľa sa vkladajú názvy súborov, ktoré auto-load nemá pripájať.

O načítavanie súborov sa stará funkcia *getDirectoryList()*. Vstupným parametrom tejto funkcie je názov priečinku. Výstupom tejto funkcie je pole, v ktorom sú uložené názvy jednotlivých súborov. Potom stačí výstup z funkcie roztočiť v cykle foreach a pripojiť každý súbor k aplikácii. Takýmto spôsobom sa pripoja triedy kontrolóra aj modelu, rovnako aj konfiguračné triedy.

3.3 Databázová vrstva

3.3.1 Základné informácie

Databázová vrstva slúži na vyberanie a ukladanie informácii z databázy. Tento framework vie pracovať len s MySQL databázou. Databázovú vrstvu tvorí jedná trieda, v ktorej sú metódy na pripojenie k databáze, ako aj na jednotlivé úkony.

Pripojenie k databáze vyžaduje informácie o databázovom serveri, prihlasovacie meno, heslo a názov databázy, ku ktorej sa pripája. Tieto informácie sú uložené v konfiguračnom súbore. Ak nie sú tieto informácie nastavené správne, aplikáciu nie je možné spustiť. Na obrazovke sa v takomto prípade objaví chybové hlásenie: „Nie je možné prihlásiť sa k databáze, prihlasovacie údaje nie sú správne!“. Ak sa aplikácii podarí prihlásiť k databáze, ale databáza neexistuje na obrazovke sa objaví chybové hlásenie: „Databáza {názov databázy} neexistuje!“

3.3.2 Metóda get_list()

Táto metóda slúži na výber z databázy. Vstupným parametrom tejto metódy je pole. Ak nie je na vstupe pole, metóda vráti chybu. Vo vstupnom poli sú povolené kľúče *cols, filter, filtermask, table, expand, sort, group.*

V kľúči *cols* sa vkladá pole, v ktorom sú vymenované požadované stĺpce z databázy. Toto pole nie je požadované, ak sa pod týmto kľúčom uloží prázdny reťazec, alebo prázdne pole, vyberú sa z databázy všetky stĺpce. Rovnako aj keď *cols* nie je vôbec zadané.

Pod kľúčom filter sa zadáva pole, v ktorom sa nachádzajú polia s podmienkami. Zápis podmienok je nasledovný:

```
'filter'=>array(
        'mask1'=>array('{stlpec}','{=,>,<,...}',{hodnota}),
 ),
 'filtermask'=>'mask1 OR mask2 OR maskn',
```

Ak nie je pod kľúčom *filtermask* zadaný žiaden reťazec s náhradníkmi, tak predvolená podmienka bude spájaná logickým operátorom AND. V opačnom prípade sa spája podľa zadefinovaného reťazca. V prípade, že sa *filtermask* nezadáva, nie je potrebné v poli *filter* pomenúvať kľúče.

Pod kľúčom *table* je uložený azda najdôležitejší parameter a to názov tabuľky. V prípade, že nie je tento parameter zadaný, nie je možne zostaviť SQL príkaz, a preto metóda vráti chybu.

Group a sort slúžia na usporadúvanie a spájanie výsledku. Pod kľúčom *group* sa skrýva pole v ktorom sú uložené názvy stĺpcov, ktoré spájajú výsledky. *Sort* je o niečo zložitejší, pod ním sa skrýva pole, v ktorom sú ďalšie polia s dvomi hodnotami. Prvá udáva názov stĺpca, podľa ktorého sa má výsledok usporiadať. V druhej je smer, v ktorom sa má výsledok usporiadať.

V metóde *get_list()* existuje možnosť spájať tabuľky. A to na základe kľúčového slova *JOIN*. Spájanie tabuliek sa definuje pod kľúčom *expand.* Zápis je nasledovný:

```
'expand'=>array(
 '{nazov tabuľky}'=>array('{typ spojenia
 ,,left,right,natural...}',array('stlpec1','stlpec2')),
 ),
```

V prípade, že sú správne nastavené vstupné parametre, metóda *get_list()* vygeneruje SQL príkaz a zavolá metódu *run_query()*. Táto metóda vráti pole, ktoré sa ďalej spracuje v kontroleri.

3.3.3 Metóda get_item()

Rovnako ako metóda *get_list()*, aj táto metóda slúži na výber z databázy. Rozdielom je to, že nevracia zoznam hodnôt, ale len jeden konkrétny záznam.

Princíp fungovania metódy *get_item()* je takmer rovnaký ako fungovanie metódy *get_list()*. Rozdielom je, že na vstupe môže byť okrem poľa *filter* aj parameter *pk*, v ktorom je uložený primárny kľúč.

Táto metóda vyberá len korektný záznam z jednej tabuľky, preto nie je možné, aby bola rozšírená. Ak by sa na vstupe nachádzal parameter *expand*, metóda by vrátila chybu.

3.3.4 Metóda update_item()

Narozdiel od predchádzajúcich dvoch metód, táto slúži na aktualizácie záznamov v databáze. Na vstupe do metódy je pole v ktorom sú povolené dva parametre, a to *table*, *data* a *filter*. Zápis filtra je rovnaký ako v predchádzajúcich metódach. Pod kľúčom *data* sú uložené polia, v ktorých sú uložené údaje, ktoré sa budú aktualizovať. Pole ma nasledovný zápis:

array({názov stlpca v databáze} => {nová hodnota}),

Na začiatku sa roztočí v cykle *foreach* pole s dátami, ktoré sa budú aktualizovať. V tomto cykle sa vytvoria dve polia, v jednom budú dáta a v druhom názvy stĺpcov, do ktorých sa budú dáta zapisovať. SQL príkaz sa buduje tak, že sa do premennej zapíše reťazec: *'UPDATE {názov tabuľky}'*. V druhom kroku sa otvorí zátvorka a k tomuto reťazcu sa doplnia cez funkciu *implode(),* názvy stĺpcov. Uzavrie sa zátvorka, za ňou sa dopíše kľúčové slovíčko *VALUES* a znovu sa otvorí zátvorka, do ktorej sa zapíšu hodnoty, ktoré sa budú aktualizovať v rovnakom poradí ako názvy stĺpcov. Taktiež sa spoja rovnakou funkciou.

Do databázy sa môžu ukladať aj reťazce, ktoré môžu obsahovať špeciálne znaky. Niektoré znaky môžu prerušiť SQL príkaz, dokonca poškodiť databázu. Musí sa

zabezpečiť ošetrenie týchto dát. Dáta sa ošetrujú v prvom cykle, keď sa zapisujú do poľa dát. Sú v nich zakódované znaky pomocou funkcie *htmlentities()*. Taktiež sa ošetria biele znaky a zároveň aj úvodzovky a apostrofy.

Návratovou hodnotou tejto metódy je *true* alebo *false*.

3.3.5 Metóda add_item()

Ako aj z názvu metódy vyplýva, jedná sa o pridávanie dát do databázy. Vstupom do tejto metódy je rovnako pole, ktoré obsahuje názov tabuľky a taktiež dáta, ktoré sa budú do databázy ukladať. Rovnako na vstupe môže byť aj typ príkazu. V tomto prípade sa môžu dáta ukladať príkazom *INSER INTO* alebo *REPLACE*. Táto možnosť je kvôli tomu, že v niektorých prípadoch je treba vkladať aj primárny kľúč, ktorý nie je možné prepísať. V takom prípade sa použije príkaz *REPLACE*. Tento príkaz zabezpečí to, že ak hodnota ukladaného primárneho kľúča neexistuje, spraví *INSERT* do databázy, v opačnom prípade sa spraví *UPDATE* riadku.

Rovnako ako aj pri predchádzajúcej metóde, aj táto metóda vracia pri úspechu 1 alebo *true.* A pri neúspechu 0 alebo *false*.

3.3.6 Metóda run_query()

Každá z predchádzajúcich metód buduje SQL príkaz, jedná sa iba o nejaký reťazec, takpovediac po anglicky napísaná veta. Na to, aby bola príkazom nejako ovplyvnená databáza, je potrebné tento príkaz spracovať. Metóda *run_query()* vytvorí spojenie s databázou a spracuje SQL príkaz. Táto metóda sa volá v každej z predchádzajúcich metód. Práve táto metóda predáva metódam výsledky, ktoré tieto metódy vracajú.

Na výstupe z tejto metódy je buď pole s výsledkami, alebo informácia o úspechu či neúspechu.

3.4 Spustenie aplikácie

Aplikácia sa nespúšťa štandardne ako bežná php či html stránka. Teda nespúšťa sa štandardne cez *index.php*, ale cez súbor *start.php*. Zavolaním tohto súboru sa spustí *autoload* a inicializujú sa metódy kontrolóra. Podľa parametrov v url adrese sa zavolá požadovaná časť aplikácie. Ak sa v url adrese nenachádzajú žiadne parametre, zavolajú sa predvolené.

Kontrolór spracuje požiadavky a zavolá metódy modelu. Model spracuje požiadavku kontrolóra a vyžiada si dáta z databázy. Tieto dáta následne model odovzdá kontrolóru, ktorý ich prerozdelí do jednotlivých časti stránky, ktoré následne zavolá.

3.5 Užívateľské prostredie

V základnom zobrazení je aplikácia zložená z troch časti:

- hlavičky
- hlavnej časti
- pätičky

Hlavička aj pätička sú rozmiestnené fixne. To znamená, že budú umiestnené stále na tom istom mieste, aj keď sa užívateľ bude po obrazovke pohybovať, hlavičku bude vždy vidieť v hornej časti obrazovky a pätičku v dolnej časti.

Pätička je zobrazená staticky a v podstate sa nemení. Sú v nej zapísané informácie o autorovi a licencii, pod ktorou je aplikácia šírená.

Hlavička je z časti dynamická, aj keď sa mení iba sporadicky. V hlavičke sa nachádza statické logo a hlavné menu aplikácie. Jednotlivé položky hlavného menu sú uložené v databáze. Pri budovaní hlavného menu sa zavolá metóda *get_menu()*. Z databázy sa vyberú položky hlavného menu. Jedná sa o názov, SEO názov, poradie a umiestnenie. Podľa umiestnenia sa rozdelia položky na hlavné kategórie a pod kategórie. Menu je tvorené neočíslovaným zoznamom, z ktorého sú pomocou CSS odstránené odrážky. Celý zoznam je potom naformátovaný do riadku a jednotlivé

položky sú vyobrazené ako tlačítka. Kvôli jednoduchosti a rýchlosti načítania stránky sa podkategórie nevysúvajú pomocou jQuery, ale pomocou CSS.

Hlavná časť aplikácie je úplne dynamická. Zobrazuje sa v nej statický aj dynamický obsah. Statický obsah je celý uložený v databáze, okrem nastavenia sa teda vyberá aj konkrétny obsah. Tento obsah je už v databáze naformátovaný a po výbere sa jednoducho vypíše na obrazovku. Nie je nutné ho ďalej spracovávať. Dynamický obsah má časť informácii uložených v databáze: Jedná sa o informácie o stránke. Zavolaním metódy *get_html_content()* sa na základe vstupného parametra, ktorým je identifikačné číslo dokumentu, vyberú informácie o stránke. Pri dynamickom obsahu sa v databáze nachádza informácia, ktorý model sa používa na spracovanie obsahu. Kontrolór tento model zavolá a inicializuje. V samotnom view-e sa volajú na príslušných miestach príslušné metódy a celý obsah sa vygeneruje a vypíše užívateľovi.

Existuje ešte jeden typ dokumentov. Je to akási kombinácia dynamického a statického obsahu. V priečinku *upload* sú uložené html dokumenty. Jednotlivé cesty k dokumentom sú uložené v databáze. Po zavolaní takejto stránky sa takýto dokument vyhľadá na serveri a následne vloží na stránku. V tomto prípade môže byť dokument formátovaný aj prostredníctvom externého štýlu. Informácia o tom, a teda aj cesta k samotnému štýlu, musí byť rovnako uložená v databáze. Pri takto uložených dokumentoch je dôležité, aby neobsahovali html značky *html, head* alebo *body*. Tieto značky sa generujú automaticky a samotný dokument sa ukladá do značky *article*. Cesty k CSS a JS sa ukladajú do databázy do stĺpcov na to určených, oddeľujú sa bodkočiarkou. Aplikácia tieto dáta rozdelí. Do hlavičky stránky ukladá už jednotlivé cesty. V prípade CSS sa môže použiť aj iný spôsob a to taký, že do databázy sa uloží cesta k CSS-sku, v ktorom sú importované všetky potrebné štýly.

3.6 SEO linky

V dnešnej dobe sa čím ďalej tým viac skloňuje slovo SEO alebo Search Engine Optimisation, čo po slovenský znamená optimalizácia pre vyhľadávače. Dôležitou časťou SEO je práve url adresa. Jednak SEO priateľské odkazy majú radšej vyhľadávače, no zároveň aj pre užívateľa sú ľahšie zapamätateľné. Adresu

simulator.sk/zasobnik si užívateľ zapamätá ľahšie ako adresu *simulator.sk/index.php?stranka=1&proces_id=123546&sto_dalsich_parametrov*.

3.7 Htaccess

Htaccess je dodatočný konfiguračný súbor. Využíva sa na zmeny v nastaveniach webového servera. Nastavenia platia pre priečinok, v ktorom je htaccess umiestnený. Táto aplikácia využíva *mod_rewrite.* Jedná sa o modul, ktorý je súčasťou apache servera. Slúži na presmerovanie stránky alebo na zmenu url adresy. Práve v súbore htaccess sa vytvárajú pekné SEO odkazy, ktoré sú opísané v kapitole 3.6. Na to, aby aplikácia vedela čítať SEO linky, je potrebné aktivovať *mod_rewrite.* To sa robí v htaccesse príkazom:

```
RewriteEngine  on
RewriteBase    /
```

Prvý príkaz aktivuje mód prepisovania url, druhým príkazom sa serveru povie, že tento priečinok má brať ako koreňový adresár. Spustenie módu prepisovania je však len začiatok. V druhom kroku treba zabezpečiť, aby server v prípade nesprávnej adresy, korektne vrátil chybu. Jedná sa o chybu *404*, čo znamená, že stránka nebola nájdená.

```
RewriteCond %{REQUEST_URI} !^/_check
RewriteRule ^_.*\.php 404.php [L]
```

Po ošetrení chybového hlásenia je možné uľahčiť si prácu pri ďalšom programovaní tým, že sa vytvoria statické presmerovania na priečinky s obrázkami, JavaScriptami alebo s CSS. *RewriteRule ^img/(.*)$ view/img/$1 [L]*, takto zapísané pravidlo zabezpečí, že ak sa v html, v ľubovoľnom mieste, bude volať akýkoľvek obrázok z priečinka */img/*, server bude vedieť, že tento sa v skutočnosti nachádza v priečinku */view/img/*.

Keďže sa táto aplikácia nespúšťa štandardne súborom *index.php*, treba aby server vedel čo má robiť, ak sa otvorí priečinok s touto aplikáciou. Túto situáciu vyrieši jednoduché pravidlo v súbore htacces: *RewriteRule ^$ /start.php [L,QSA]*. Preložené do

slovenčiny, ak za lomítkom nenasleduje nič iné, zavolá sa skript *start.php*.

Podobným pravidlom sa prepisujú aj SEO odkazy.

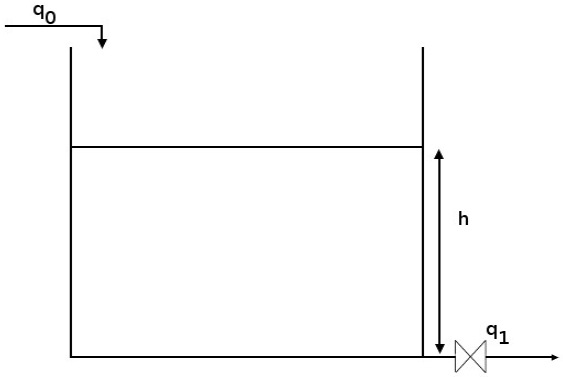
RewriteRule ^procesy/zasobnik/?$ /start.php?page_id=1 [L,QSA]
RewriteRule ^procesy/2_zasobniky/?$ /start.php?page_id=2 [L,QSA]
RewriteRule ^procesy/miesac/?$ /start.php?page_id=3 [L,QSA]
RewriteRule ^uvod/?$ /start.php [L,QSA]

4 Zásobník kvapaliny

4.1 Základné výpočty

[10]

V reálnom svete je zásobník kvapaliny reprezentovaný ako nejaká nádrž, do ktorej priteká a vyteká kvapalina o nejakom prietoku (obr. 3).



Obr. 3: Zásobník kvapaliny

V zásobníku je potrebné vypočítať výšku hladiny *h*. Na to, aby bolo možné vypočítať výšku hladiny, je potrebné vedieť hodnotu vstupného prietoku q_0, plochu prierezu *F* a konštantu ventilu *k11*. Výstupný prietok sa potom počíta podľa vzťahu.

$$q1(t) = k11\sqrt{h(t)}$$

Dosadením do materiálovej bilancie a za predpokladu, že derivácia výšky hladiny podľa času je rovná nule, je možné vyrátať výšku hladiny v ustálenom stave.

$$\dot{m}_0 = \dot{m}_1 + \frac{dm(t)}{dt}$$

$$q_0 = q_1 + F\frac{dh(t)}{dt}$$

$$q_0 = k11\sqrt{h(t)} + F\frac{dh(t)}{dt}$$

$$h^s = \left(\frac{q_0^s}{k_{11}}\right)^2$$

Pred začatím simulácii sa vypočíta výška hladiny v ustálenom stave s maximálnym možným vstupným prietokom, tým sa stanový maximálna výška hladiny, ktorú je možné v zásobníku dosiahnuť. Počíta sa to kvôli následujúcej animácii výšky hladiny. Taktiež je potrebné vyrátať percentuálne naplnenie zásobníka. Percentuálne naplnenie je potrebné kvôli správnemu vykresleniu analógového ukazovateľa.

Na výpočet aktuálnej hladiny sa používa metóda Runge-Kutta. Jedna sa o iteračnú metódu, ktorá slúži na riešenie obyčajných diferenčných rovníc. Táto metóda bola vyvinutá okolo roku 1900 v Nemecku matematikmi C. Runge and M. W. Kutta. Detailný opis tejto metódy sa nachádza na stránke:

http://en.wikipedia.org/wiki/Runge-Kutta_methods

Model pre výpočet výšky v jednotlivých bodoch vychádza zo vzťahu:

$$q_0 = k11\sqrt{h(t)} + F\frac{dh(t)}{dt}$$
$$F\frac{dh(t)}{dt} = q_0 - k11\sqrt{h(t)}$$
$$\frac{dh(t)}{dt} = \frac{q_0}{F} - \frac{k11\sqrt{h(t)}}{F}$$

Výpočty prebiehajú na strane klienta, teda v prehliadači. Preto je potrebné prepísať vzorce do jazyka, ktorému prehliadač rozumie a teda do JavaScriptu. Výpočet maximálnej výšky hladiny:

```
_.xmaximal = Math.pow((_.maxq1/_.k11),2);
```

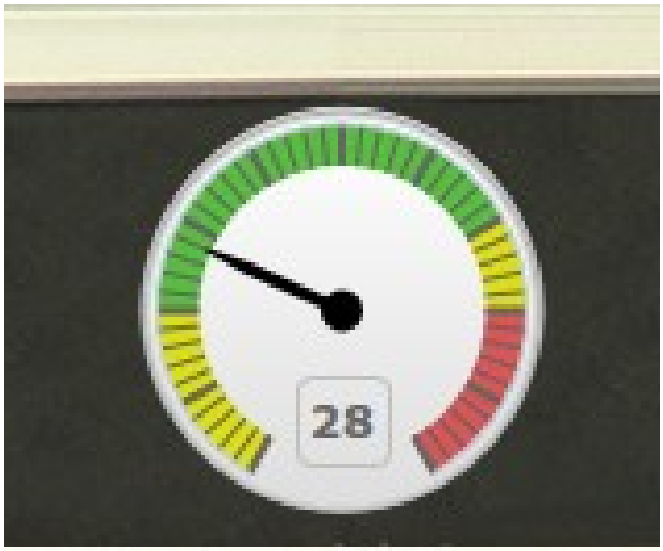
Výška hladiny je zapísaná do privátnej premennej, v ktorej ostáva až do znovu načítania stránky. Práve táto premenná slúži na výpočet percentuálneho naplnenia zásobníka.

Simulácia sa spúšťa verejnou funkciou *simuluj()*. V prvom kroku sa overí, či sú parametre korektne nastavené a či nič neblokuje spustenie výpočtu. Ak je všetko v poriadku, zapíše sa informácia o spustení simulácie a taktiež sa do privátnej premennej

uloží časovač *setInterval*. Daná premenná sa neskôr používa na zastavenie simulácie. V intervale sa volá verejná funkcia výpočet. V tejto metóde sa práve využíva metóda Runge-Kutta. Vypočítaná hodnota hladiny sa uloží do privátnej premennej a zavolá sa metóda *animuj()*. Metódou *animuj()* sa vykresľujú hodnoty na obrazovku.

4.2 Užívateľské prostredie

Užívateľské prostredie je navrhnuté tak, aby čo najvernejšie pripomínala laboratórne zariadenie. V spodnej časti sa nachádza ovládací panel, na ktorom sú kontrolky indikujúce stav zariadenia, tlačidlo na zapnutie alebo vypnutie zariadenia. Vedľa tlačidla na zapínanie sa nachádza aj spúšťanie simulácie. V pravom dolnom rohu sa nachádza analógový ukazovateľ (obr. 4), ktorý zobrazuje aktuálne naplnenie zásobníka v percentách.

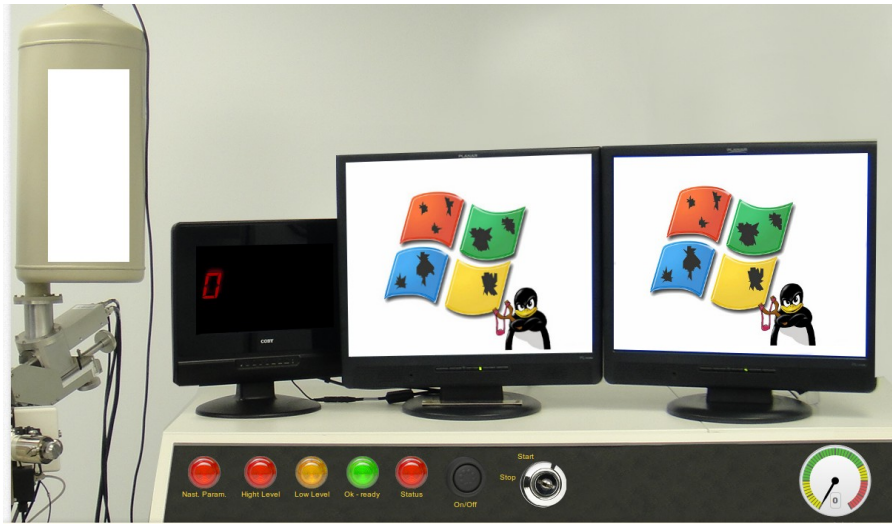


Obr. 4: Percentuálny ukazovateľ výšky hladiny

Farebne sú zvýraznené kritické hodnoty. Zelenou farbou je označený normálny stav zásobníka kvapaliny. Žltou farbou sú označené varovné situácie, na začiatku je to nízka hladina v zásobníku, na konci varovanie o prepĺňaní zásobníka. Ak sa ručička nachádza na začiatku v žltej zóne, indikuje to nízku hladinu. V takom prípade zároveň svieti žlté varovné svetlo. Ak sa ručička nachádza v žltej zóne na konci ukazovateľa, nesvieti žiadne varovanie. Toto označenie signalizuje, že sa zásobník blíži k

nebezpečnému stavu. Varovné svetlo sa rozsvieti, až keď sa ručička dostane do červenej zóny. Analógový ukazovateľ je jeden z mnohých modulov z balíka highcharts. Highcharts je možné využívať v prípade neziskových aplikácii bezplatne. Po inicializovaní zobrazuje ukazovateľ číslo nula. Po inicializovaní sa nastaví časovač, ktorý každých 250 milisekúnd skontroluje aktuálnu výšku hladiny a vypočíta percentá na základe známej maximálnej výšky. V tomto cykle sa taktiež overuje, či je simulácia zapnutá a či sa hodnota zmenila. Táto kontrola nie je nutná, ale zabraňuje zbytočnému vykresľovaniu. V niektorých prehliadačoch dochádzalo k blikaniu hodnoty a nevyzeralo to najlepšie.

Na ľavej strane (obr. 5) sa nachádza zásobník kvapaliny, v ktorom sa zobrazuje aktuálna výška hladiny. Hladina v tomto zásobníku je animovaná. Počas výpočtu sa plynule napĺňa a vyprázdňuje.



Obr. 5: Celková schéma zapojenia

V strede obrazovky (obr. 6) sú umiestnené tri monitory. Na menšom, ktorý je umiestnený vľavo, sa zobrazuje nastavený čas simulácie, ktorý sa odpočítava. Po spustení simulácie sa od nastaveného času odpočíta zmena času. Tento čas sa aktualizuje v každom cykle.

Na strednom monitore (obr. 6) je zobrazený graf rovnako z balíka highcharts. Tento graf sa inicializuje pri načítaní stránky. Rovnako ako aj analógový ukazovateľ, aj

tento graf spúšťa časovač, ktorý sťahuje aktuálne hodnoty času a výšky hladiny. Tento interval je nastavený na 1500 milisekúnd, teda na 1,5 sekundy. Zobrazovanie sa aktualizuje v tomto intervale z jednoduchého dôvodu, ak by totiž vykresľovalo v rovnakom intervale ako prebieha výpočet, bol by graf neprehľadný. Obsahoval by totiž veľmi veľa bodov. Aj v tomto prípade sa sleduje, či je spustená simulácia. V prípade, že simulácia bola zastavená alebo výpočet by neprebiehal, vykresľoval by graf nezmyselné údaje.

Na poslednom monitore (obr. 6) sa nachádza formulár, prostredníctvom ktorého sa nastavujú parametre zásobníka a čas simulácie. Po uložení hodnôt sa na tomto monitore zobrazí ukazovateľ prietoku spolu s posuvníkom, ktorým sa mení hodnota prietoku. Hodnota prietoku sa zobrazuje percentuálne. Posuvník pracuje tiež v percentách. Hodnota sa po inicializácii nastaví na hodnotu prietoku zadanú vo formulári. Ukazovateľ pracuje rovnako ako ukazovateľ výšky hladiny. Aktualizuje sa každých 250 milisekúnd. Pri aktualizácii sa sleduje či je zariadenie zapnuté a či sa hodnota prietoku zmenila. Nesleduje sa či je simulácia spustená, pretože prietok je možné meniť aj pred spustením simulácie. Posunutím posuvníka sa udalosťou *onChange* zavolá verejná funkcia *zmenPrietok()*. Okrem toho, že sa ukladá aktuálna hodnota v percentách, prepočíta sa aj na reálnu hodnotu prietoku, ktorá sa používa na výpočet hladiny.

Obr. 6: Zobrazenie po načítaní stránky

Po načítaní stránky sú monitory neaktívne (obr. 6), až po kliknutí na tlačítko

spustenia zariadenia sa zobrazia nastavené ukazovatele. Na prvom monitore, ktorý zobrazuje odpočítavanie času simulácie, sa po zapnutí monitora zobrazuje číslo nula. Po uložení parametrov nastavených na treťom monitore sa toto číslo zmení na čas, ktorý je nastavený vo formulári (obr. 7).

Obr. 7: Zobrazenie času simulácie pred a po nastavení parametrov

4.3 Spracovanie a validácia vstupných parametrov

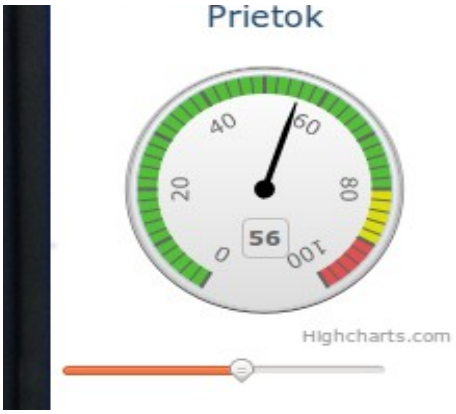
Po odoslaní formuláru sa kontroluje či sú hodnoty uvedené v jednotlivých poliach čísla. Kontroluje sa či zadané hodnoty nie sú biele znaky. Ak prejde cez validáciu hodnôt, testuje sa či nie sú zadané hodnoty fyzikálne správne. Potom čo parametre prejdu cez validáciu, posunú sa hodnoty na ďalšie spracovanie. Ak sa zistí chyba, zabliká kontrolka parametrov (obr. 8).

Vypočíta sa maximálna výška hladiny, ktorá sa uloží do privátnej premennej pre ďalšie výpočty. Vypočíta sa percentuálny prietok a inicializuje sa highcharts analógový ukazovateľ. Po tom ako prejdu hodnoty spracovaním a nenájde sa žiaden problém, rozsvieti sa zelená kontrolka OK. Zmení sa status zariadenia na inicializované a odblokuje sa simulácia.

Obr. 8: Kontrolky po nastavení parametrov

4.4 Ovládanie vstupného prietoku

Na ovládanie vstupného prietoku sa používa posuvník, ktorý je vlastne formulárový prvok z HTML 5. Jedná sa o *input* typu *range*. HTML 5 je v dnešnej dobe už dobre podporovaná, aj napriek tomu je *range* problematický. Aj keby sa mohlo zdať, že vinníkom bude ako obvykle *Internet Explorer,* nie je tomu tak, nekompatibilný prehliadač je tentokrát *Firefox*. Je to zarážajúce, keďže Firefox dlhú dobu patril medzi najlepšie prehliadače ako aj zo strany používania, tak aj programovania webových aplikácii. Pozitívom ale je, že aj tento problém sa dá jednoducho vyriešiť. Riešením je jQuery plugin, ktorý vygeneruje tento prvok aj v prehliadačoch ktoré ho nepodporujú (obr. 9).



Obr. 9: Ukazovateľ prietoku spolu s posuvníkom na nastavovanie prietoku

Krok posuvníka je nastavený na jedna, keďže sa prietok mení v percentách. Minimálna hodnota je nastavená na nulu a maximálna na 100. Percenta sú tiež farebne označené. Zelenou je zvýraznený rozsah 0 až 80 percent, žltou farbou od 80 do 90 percent a 90 až 100 percent je zvýraznené červenou farbou.

```
pub.zmenPrietok = function()
{
        _.percq1 = $('#flowSetter').val();
        _.q1 = (_.percq1/100)*_.maxq1;
        var point = _.chartThree.series[0].points[0]
        point.update(_.percq1);
}
```

Hodnota aktuálneho prietoku v percentách sa načíta z formulárového prvku *range*. Táto hodnota sa použije na výpočet prietoku. Prietok sa ukladá do premennej *_.q1*. Keďže sa táto premenná používa pri výpočtoch, aplikácia reaguje na zmenu okamžite.

4.5 Animovaná výška hladiny

Vykresľovanie hladiny bolo trochu komplikovanejšie, než by sa na prvý pohľad zdalo. Jednotlivé časti zariadenia sú na absolútnych pozíciach. Podľa pravidiel absolútneho pozícovania, sa umiestnenie prvku s absolútnou pozíciou vzťahuje k okrajom prvku s relatívnou pozíciou. Na vykreslenie hladiny sa používajú dva elementy. Prvý je umiestnený absolútne na vyhradené miesto, druhý element je vnorený vnútri nadriadeného elementu. Zväčšovaním vnoreného elementu sa jeho výška zväčšovala smerom dole. Ak by bolo v tomto elemente použité modré pozadie vyzeralo by to akoby gravitácia nepôsobila na vodu. Preto je modré pozadie použité v absolútne umiestnenom elemente. Po načítaní stránky je teda zobrazené modré pozadie a cez neho je zobrazený element bielej farby, ktorý ho úplne vypĺňa.

Animácia výšky hladiny sa začína výpočtom percent naplnenia zásobníka. Ako referenčná hodnota sa používa maximálny prietok. Vypočítaná hodnota sa odpočíta od čísla 100. To znamená že ak je zásobník naplnený na 5 percent, výška bieleho elementu sa nastaví na 95 percent.

```
Perc = Math.round((_.x/_.xmaximal)*100);
perc2 = 100-perc;
$('#tank1_level_indicator').animate({'height':perc2+'%'},100);
```

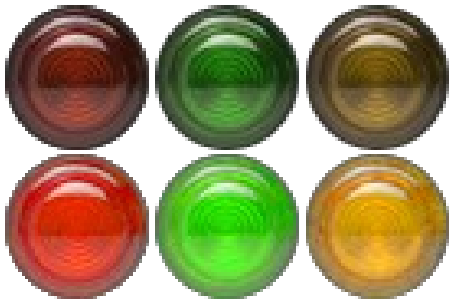
Aby vykresľovanie hladiny vyzeralo plynulejšie, používa sa na to metóda *animate(),* ktorá je súčasťou knižnice jQuery. Rýchlosť vykresľovania je nastavená na 100 milisekúnd. Táto hodnota bola určená metódou pokus – omyl. Pri menšej hodnote už nebol prechod badateľný. Pri vyššej už bolo vykresľovanie spomalené a nevyzeralo to najlepšie. Pred každým vykreslením hladiny sa kontroluje stav hladiny a na základe nastavených hraničných hodnôt sa spúšťajú alarmy.

```
if((_.x/_.xmaximal)*100 >= 80)
        _.alarm('hight');
 else
        _.alarmStop('hight');
```

Privátna funkcia *_.alarm()* na základe vstupného parametra rozsvieti indikátor stavu hladiny. Taktiež zhasne indikátor opačného stavu. Toto zhasínanie nie je za bežných okolností potrebné, pretože o zhasnutie alarmu sa postará funkcia *_.alarmStop()*. V niektorých situáciach však dochádzalo k tomu, že sa zásobník rýchlo preplnil. Pretože vysoká hladina sa overuje ešte pred nízkou, dochádzalo k tomu, že pol sekundy svietili obe kontrolky, a keď hladina zodpovedala len jednej.

Okrem rozsvietenia kontrolky sa pri vysokej hladine spusti aj varovný zvukový signál. Zvukový signál sa spúšťa prostredníctvom JavaScriptu, ale na prehrávanie sa používa prvok HTML 5 *audio*. Vytvorí sa globálna premenná *audioElement,* do ktorej sa zapíše element audio. Pridá sa mu atribút *src*, do ktorého sa zapíše cesta k audio súboru. V prípade alarmu sa spustí prehrávanie príkazom *audioElement.play()*. Pri ukončení alarmu sa zvukový signál vypne príkazom *audioElement.pause().*

Zapínanie a vypínanie indikátorov a teda aj vlastné rozsvecovanie je kombináciou jQuery a CSS. jQuery vymení triedy kontrolky, a teda rozsvieti alebo zhasne svetlo. Zhasnuté aj rozsvietené kontrolky sú súčasťou jedného obrázka. Pomocou CSS sa vyberá konkrétna časť obrázka pre jednotlivé triedy.



Obr. 10.: CSS sprite kontroliek

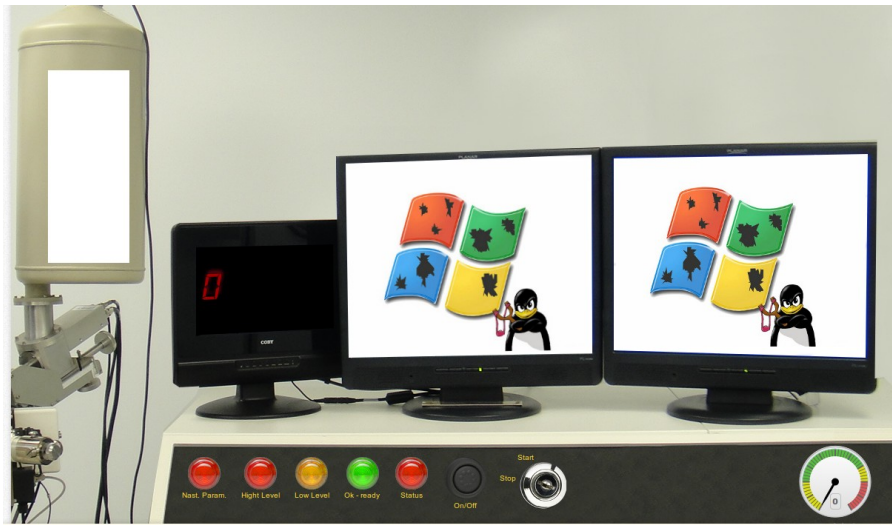
Kontrolky sú vytvorené v programe GIMP, v ktorom boli orezané a zmenená farba. Základná kontrolka bola odfotografovaná v električke, originálna je iba červená zhasnutá kontrolka. Ostatné kontrolky sú jej kópie, v ktorých boli zmenené kanály

farieb. Takto jednoducho sa dosiahol efekt svietiacich kontroliek.

```
.led{width:46px;height:46px;background:url("../img/kontrolky.png") no-repeat;}
.cervena_nesvieti{background-position: 0 0;}
.cervena_svieti{background-position: 0 -45px;}
.zlta_nesvieti{background-position: -92px 0;}
```

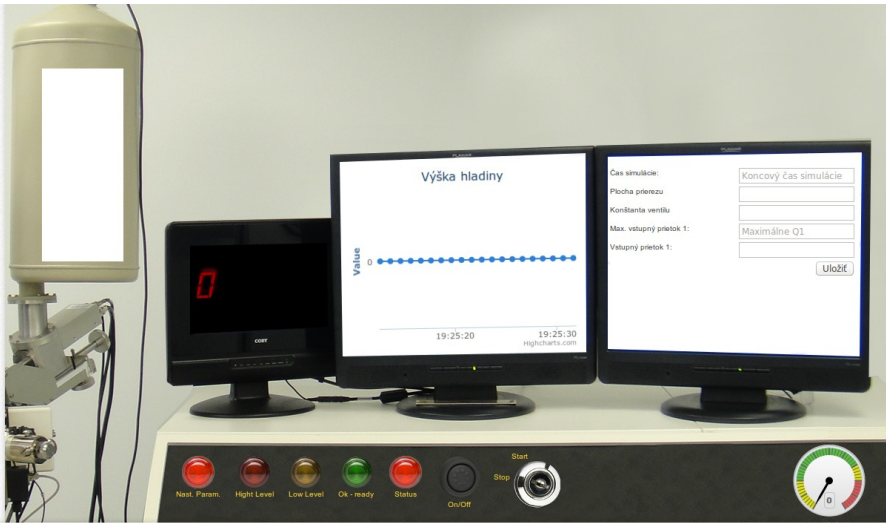
4.6 Aktivácia zariadenia a spustenie simulácie

Aby bolo používanie čo najreálnejšie, je zariadenie po načítaní stránky vypnuté. Rovnako ako aj reálne zariadenie po príchode do laboratória, aj toto zariadenie sa musí zapnúť. Zariadenie sa zapína stlačením tlačítka ON/OFF. Po stlačení sa rozsvietia všetky kontrolky. Zároveň sa na monitoroch zobrazí obrázok, ktorý signalizuje spúšťanie počítača.



Obr. 11.: Spúšťanie zariadenia

Kontrolky zhasnú asi po troch sekundách. Po zhasnutí kontroliek sa prepnú aj obrazovky z módu spúšťania do pracovného módu. Po spustení nezhasnú všetky kontrolky, ostane svietiť červené varovné svetlo, ktoré signalizuje nenastavené parametre. Rovnako svieti aj červené svetlo status, ktoré informuje že zariadenie nie je pripravené.



Obr. 12.: Zariadenie po spustení

Kliknutím na štart sa overí či sú nastavené parametre. Keďže parametre nie sú nastavené, začnú striedavo blikať kontrolky nastavenie parametrov a status OK. Simulácia sa nespustí a štartovací kľúč sa vráti do polohy stop.

Nastavením parametrov sa monitor prepne na ovládanie prietoku, zhasne kontrolka parametrov. Status zmení farbu na žltú a rozsvieti sa zelená kontrolka OK. V prípade, že sa teraz klikne na tlačítko štart, prepne sa štartovací kľuč do polohy štart. Kontrolka statusu zmení farbu na zelenú a spustí sa simulácia. Začne sa vykresľovať graf, meniť ukazovateľ hladiny a napĺňať zásobník.

Kliknutím na stop sa simulácia pozastaví. Simulácia sa dá obnoviť kliknutím na štart. Parametre ostanú nastavené a pokračuje sa tam, kde simulácia skončila. Pre reštart simulácie je potrebné kliknúť na stop, čím sa zastaví simulácia a potom na tlačítko ON/OFF. Zariadenie sa vypne. Po opätovnom zapnutí je zariadenie pripravené na novú simuláciu.

5 Administrátorské rozhranie

5.1 Prihlásenie do administrácie

Prihlásenie do administrácie je prístupné na adrese */administracia*. Ide o jednoduchú stránku s formulárom pre prihlásenie. Po zadaní mena a hesla sa v databáze skontroluje či zadané meno existuje a či sa heslo zhoduje s heslom zadaným do databázy. Po overení zhody sa skontroluje či prihlásený administrátor nie je blokovaný. Ak nie, sprístupní sa mu administrátorské rozhranie.

Heslo je v databáze uložené ako *md5 hash*. Užívateľom zadané heslo sa upraví funkciou *md5()* na *hash*. Následne sa porovná z heslom uloženým v databáze.

5.2 Pridávanie administrátorov

Keďže aplikácia môže mať viacero administrátorov, je potrebné, aby existovala možnosť čo najjednoduchšie pridávať administrátorov. Na prihlásenie do administrácie sa používa prihlasovacie meno a heslo. Do databázy sa okrem mena a hesla ukladá aj informácia o poslednom prihlásení a IP adresy posledného prihlásenia. Informácia o čase posledného prihlásenia sa ukladá vo formáte *timestamp,* čiže počtu sekúnd od *1.1.1970*. Po kliknutí na záložku administrátori, sa zobrazí zoznam aktuálnych administrátorov. Pod zoznamom je odkaz, ktorý zobrazuje formulár pre pridávanie administrátorov.

Kliknutím na tento odkaz sa zobrazí box, v ktorom sú jednotlivé položky formulára. Pre pridanie administrátora je potrebné vyplniť meno, email a heslo. Infomácia o registrovaní, spolu s menom a heslom, bude odoslaná na zadanú emailovú adresu. V prípade, že nie je vyplnené žiadne heslo, systém vygeneruje náhodné heslo.

Po prihlásení môže každý administrátor zmeniť svoje prihlasovacie heslo. Hesla všetkých administrátorov môže meniť len hlavný administrátor. Hlavný administrátor môže tiež povoliť alebo zablokovať prístup ostatným administrátorom. Môže ho aj

definitívne zmazať, taktiež upraviť práva administrátora.

5.3 Vytváranie položiek v menu

Položky v menu sa vytvárajú v karte menu. Po rozbalení karty sa zobrazí stromová štruktúra hlavného menu. Existuje verejné menu a administrátorské menu. Vo verejnom menu sa vytvárajú a editujú položky, ktoré sa zobrazujú všetkým užívateľom.

V administrátorskom menu sú položky, ktoré sú zobrazené len prihláseným administrátorom. Aj keď je modul celý naprogramovaný v php, je nutné ho v databáze zaregistrovať. Pri registrovaní sa zadá názov kontrolóra a pri načítaní tejto stránky sa zavolá metóda *get_html_content()*, práve z registrovaného kontrolóra.

V tejto záložke sa nastavujú SEO informácie: názov v url, popis, meta popis a kľúčové slova.

5.4 Vytváranie dokumentov

V záložke dokumenty sa vytvárajú statické dokumenty, prípadne sa nahrávajú statické HTML dokumenty. V tejto záložke sa nastaví názov, SEO názov a meta popis dokumentu. Tento dokument sa uloží do niektorej položky z menu.

V prípade, že sa zvolí statický dokument, zobrazí sa CK editor. V tomto editore je možné vytvárať dokumenty rovnako ako v prostredí kancelárskeho balíka Office. Takýmto spôsobom sú vytvárané dokumenty s informáciami o procesoch, návody na používanie aplikácie, ale aj úvodná stránka aplikácie. Takto vytvorený dokument sa celý ukladá do databázy. Pri načítaní stránky sa z databázy vyberie celý obsah a vloží sa na stránku.

Pre nahrávanie dokumentov je potrebné zvoliť HTML dokument. V tomto prípade sa zobrazí formulárový prvok, v ktorom sa zvolí cesta k súboru. Súbor sa uloží na server a do databázy sa zapíše cesta k tomuto súboru.

Existuje tiež možnosť meniť popis dokumentov ktoré sú naprogramované ako

samostatné moduly v php. Tento popis môže byť formátovaný pomocou CK editora alebo vložený ako neformátovaný text.

5.5 Regenerovanie .HTACCESS

HTACCESS slúži v prvom rade na nastavovanie pekných url. Moduly a dokumenty sa štandardne zobrazujú cez odkaz s parametrami. Aby bolo možné použiť SEO odkaz, musí sa správne zapísať do súboru HTACCESS.

Súbor HTACCESS môže byť upravovaný dvoma spôsobmi. Prvým spôsobom je ručné dopísanie cez textové pole, v ktorom sa zobrazí aktuálny súbor. V prípade vytvorenia viacerých dokumentov to môže byť zdĺhavé. Taktiež nie každý pozná správnu syntax zápisu pravidla pre prepis url adresy. Preto je naprogramovaný aj automat, ktorý po spustení aktualizuje súbor HTACCESS. Na aktualizáciu vychádza z položiek v hlavnom menu, ako aj z jednotlivých dokumentov.

Vlastníkom tohto súboru musí byť užívateľ, pod ktorým beží aj webový server. Ak niektorý užívateľ upraví tento súbor, prípadne nakopíruje vlastný súbor, nebude možné upravovať ho prostredníctvom webového rozhrania. Aby bolo možne upravovať aj takto nakopírovaný súbor HTACCESS, je nutné, aby užívateľ zmenil práva súboru na 777, prípadne zmenil vlastníka súboru.

6 Inštalácia Aplikácie

6.1 Nastavenie servera

Aby aplikácia pracovala správne musí sa aj správne nainštalovať. Pred inštaláciou aplikácie si treba overiť či sú na serveri nainštalované a sprístupnené potrebné knižnice.

Pre správne fungovanie je potrebné, aby na serveri bolo nainštalované PHP verzie 5.2 alebo vyššej. Okrem PHP je potrebné mať nainštalovaný aj databázový server MySQL. Tieto dve podmienky musia byť splnené aby bolo možné apliká̆ciu nainštalovať a spustiť. Avšak aby bolo možné plne využívať možnosti aplikácie je nutné aby na serveri bol nainštalovaný a povolený modul mod_rewrite a CURL, taktiež musí byť povolené použitie súboru HTACCESS.

6.2 Inštalácia databázy

Aplikáciu nie je možné spustiť bez nainštalovaných tabuliek v databáze MySQL. Minimálne je potrebné importovať tabuľky *menu* a *document*. So základnými tabuľkami je možné používať aplikáciu so základnými funkciami. Aby fungovala aj administrácia je potrebne naimportovať tabuľku *user*.

Po importovaní tabuliek je potrebné pripojiť databázu k aplikácii. V konfiguračnom súbore sa nastaví server databázy, prihlasovacie meno, heslo a názov databázy, ku ktorej sa bude aplikácia pripájať.

6.3 Prvé spustenie aplikácie

Aplikáciu je nutné umiestniť na webový server, po otvorení v prehliadači sa automaticky načíta úvodná stránka. V prípade, že výstup bude len biela strana, znamená to že aplikácia nie je korektne nainštalovaná.

Dôvod, prečo sa aplikácia nespustila, je možné zistiť jednoducho spustením

režimu hlásenia chýb, takzvaným debug režimom. Debug sa spúšťa pridaním parametra *debug=1* do url adresy. V tomto režime sa zobrazia php chyby a varovania. Ak ani po zapnutí režimu *debug* sa nezobrazí žiadna chýba, znamená to, že sa súbor HTACCESS nespracoval správne. Nespracovanie môže vyť spôsobené nesprávnym umiestnením súboru HTACCESS alebo blokovanie súboru nastaveniami servera. Aby bolo odhaľovanie chyby jednoduchšie, je dobré pozrieť sa na server do chybového logu. Na posledných riadkoch bude práve chýba, ktorá spôsobila nespustenie aplikácie.

Po odstránení všetkých chýb, ak nejaké nastali, je potrebné prihlásiť sa do administrácie a zmeniť heslo administrátora. Základné prihlasovacie meno je *admin* a heslo rovnako ako meno *admin*.

Záver

Cieľom práce bolo vytvoriť modul pre simulácie chemicko technologických procesov. Vytvorený bol model zásobníka kvapaliny. Výpočty zabezpečuje JavaScript, čo je jazyk bežiaci na strane klienta. Vďaka tomu je možné robiť skokové zmeny a sledovať reakcie systému, počas simulácie. Zásobník plynule reaguje na zmeny prietoku. Výstup je možne sledovať na grafe, ale aj ako hladinu kvapaliny, ktorá sa dynamicky mení. Prietok ale aj naplnenie hladiny je možné sledovať na analógových ukazovateľoch.

Na naprogramovanie stránky bolo použité objektovo orientované programovanie v jazyku PHP. Model zásobníka bol vytvorený pomocou Module pattern. Aplikácia je optimalizovaná na moderné prehliadače. Aby bola kompatibilná aj so staršími verziami, sú v nej použité rôzne JavaScriptové pluginy, ktoré sprístupňujú HTML 5 aj pre prehliadače, ktoré ho nepodporujú.

Zoznam použitej literatúry

[1] *PHP (programovací_jazyk)*. [online].[Cit. 24.5.2013]. Dostupné na: http://sk.wikipedia.org/wiki/PHP_(programovací_jazyk)

[2] *Objektovo orientovane programovanie v normálnej ľudskej reči*. [online]. [Cit. 24.5.2013]. Dostupné na: http://www.zajtra.sk/programovanie/165/objektovo-orientovane-programovanie-v-normalnej-ludskej-reci

[3] *Model view controller*. [online]. [Cit. 24.5.2013] Dostupné na: http://en.wikipedia.org/wiki/Model–view–controller

[4] *JavaScript*. [online].[Cit. 24.5.2013] Dostupné na: http://sk.wikipedia.org/wiki/JavaScript

[5] *JQuery*. [online].[Cit. 24.5.2013] Dostupné na: http://sk.wikipedia.org/wiki/JQuery

[6] *Module pattern*. [online]. [Cit. 24.5.2013] Dostupné na: *http*://en.wikipedia.org/wiki/Module_pattern

[7] *HTML 5 nové vlastnosti*. [online].[Cit. 24.5.2013] Dostupné na: http://programujte.com/clanek/2010082200-html5-nove-vlastnosti/

[8] *Seznámeni s HMTL 5*. [online].[Cit. 24.5.2013] Dostupné na: http://interval.cz/clanky/seznameni-s-html-5/

[9] Makzan: *Programueme hry v HTML5*, Computer Press, 2012

[10] Bakošová M., Fikar M.: *Riadenie procesov*, Vydavateľstvo STU, Bratislava, 2008