LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

PHP programmēšanas valodas un MySQL datubāzes iespēju pētījums projektu realizācijai

BAKALAURA DARBS

Darba autors: **Andrejs Vorobjovs**
Stud. apl. num.: av07032

Darba vadītājs: **Uldis Straujums**
Grāds: Dr.dat.
Amats: LU Datorikas fakultātes lektors

Rīga 2011

ANOTĀCIJA

Bakalaura darba mērķis ir izpētīt PHP programmēšanas valodu, pastāstot par tās iespējām, kāpēc jālieto šī valoda un kur tā tiek pielietota, kādas ir valodas priekšrocības un trūkumi, kā arī izpētīt MySQL datubāzi, tās iespējas, plusus un mīnusus, pielietošanu, efektivitāti un drošību.

Problēmu izraisa tas, ka lietotājiem bieži rodas jautājums: kādu programmēšanas valodu un datubāzi labāk izmantot kāda projekta realizācijai.

Darba teorētiskajā daļā aprakstīta vispārēja informācija par PHP un MySQL: tiek stāstīts par to vēsturi, pielietošanu, darbību un versijām, kā arī salīdzinājums ar citām programmēšanas valodām un datubāzēm. Praktiskajā daļā tiek pētītas PHP un MySQL iespējas, lai atrastu to lietderību, un aprakstīti daži piemēri, kas palīdzēs rezultātā noskaidrot, kāpēc PHP un MySQL ir ērtas projektu realizācijai.

Atslēgvārdi: PHP, MySQL, programmēšanas valoda, datubāze, tīmekļa vietne.

ABSTRACT

The aim of this bachelor work is to do research on PHP programming language, telling about its possibilities, why this language should be used and how it is used, what are advantages and disadvantages of this language, as well as to investigate MySQL database, its possibilities, pros and cons, usage, effectiveness and safety.

The problem often is that users do not know which programming language and database should be used for better realization of a project.

In the theoretical part of this work general information of PHP and MySQL, their history, use, activities and versions are described, also they are compared to other languages and databases. In the practical part PHP and MySQL possibilities have been investigated to find out that they are useful, and some examples, which could help to come to a conclusion why PHP and MySQL are convenient for realization of projects, have been described.

Keywords: PHP, MySQL, programming language, database, website.

SATURS

APZĪMĒJUMU SARAKSTS

PHP – Hypertext Preprocessor (programmēšanas valoda).

MySQL – My Structured Query Language (datubāze).

Perl – Practical Extraction and Report Language (programmēšanas valoda).

WAMP – Windows, Apache, MySQL, PHP (serveris).

Apache – atvērtā pirmkoda tīmekļa serveris.

HTML – Hyper Text Markup Language (iezīmēšanas valoda).

XML – Extensible Markup Language (iezīmēšanas valoda).

XHTML – Extensible HyperText Markup Language (iezīmēšanas valoda).

URL – Uniform Resource Locator (standartizēta resursa adrese Internetā).

CGI – Common Gateway Interface (saskarnes standarts).

GUI – Graphical User Interface (lietotāja saskarnes paveids).

C – programmēšanas valoda.

GET – HTTP metode.

POST – HTTP metode.

ASP – Active Server Pages (Microsoft tehnoloģija).

IDE – Integrated Development Environment (programmatūra).

Cookie – tīmekļa servera nosūtītā neliela teksta virkne klientam.

Task Scheduler – uzdevumu plānotājs Windows operētājsistēmās.

Cron – uzdevumu plānotājs UNIX operētājsistēmās.

API – application programming interface (gatavu klašu, procedūru, funkciju, struktūru un konstantu kopums).

mSQL – mini SQL (datubāze).

InnoDB – viena no MySQL apakšsistēmām.

SSL – Secure Sockets Layer (kriptogrāfiskais protokols).

phpMyAdmin – tīmekļa saskarne MySQL administrēšanai.

Skriptu valoda – programmēšanas valoda operāciju secības veidošanai, kurus izpilda lietotājs.

Atvērtais programmas pirmkods (Open Source) – kods, kurš ir pieejams caurskatīšanai un izmaiņām.

IEVADS

Mūsdienās daudzi cilvēki lieto interneta pakalpojumus un ar katru gadu lietotāju skaits pieaug. Eksistē sekojoši būtiski interneta pakalpojumi: informācijas noskaidrošana, sazināšanās ar citām personām, interneta veikalu izmantošana, maksājumu veikšana, kā arī datu ielādēšana un uzglabāšana. Neskatoties uz šīm iespējām, vairumam lietotāju dažreiz ir interesanti noskaidrot, kā katra tīmekļa vietne tiek realizēta un piedāvāta publiski, un kā to var panākt, lai rezultātā, kad vietne būs gatava, to saturu varētu izmainīt, nepielietojot programmēšanu.

Ļoti bieži internetā var atrast kādas jaunas vietnes, un tas ir iespējams ne tikai tāpēc, ka kādas organizācijas vai firmas reklamē sevi un savu produktu, bet arī tāpēc, ka lietotāji grib pārbaudīt un paplašināt savas prasmes, iemācīties izveidot savas vietnes un izvietot tās internetā.

Jebkuram lietotājam bez programmēšanas prasmēm būs grūti uzsākt veidot vietni, bet ar interneta palīdzību viņš varēs atrast visu nepieciešamo informāciju kā to darīt. Biežāk lietotā tīmekļa vietne ir tāda, kurā, pēc tās izveidošanas, ir iespējams izmainīt lapas saturu, nepielietojot programmēšanu. Šādu tīmekļa lapas veidu sauc par dinamisku. Rodas jautājums – kāda programmēšanas valoda jālieto un kur glabāt informāciju veidojot vietni?

Eksistē daudz dažādas programmēšanas valodas interneta lapu veidošanai. Populārākās ir PHP, JavaScript, Ruby, Perl un citas. Tā kā mana tēma ir saistīta ar PHP valodu, kuras pielietošana ir aktuāla šodien un es izmantoju to savā darba vietā, esmu izvēlējies pētīt un stāstīt tieši par PHP iespējām.

Runājot par informācijas glabāšanu, eksistē daudz datubāzes, kuras var izmantot – MySQL, Oracle, Apache Derby un citas. Tā kā mana tēma ir saistīta ar MySQL, kuru biežāk izmanto informācijas glabāšanai, kad vietnes kods ir rakstīts uz PHP, es stāstīšu par MySQL datubāzes iespējam.

Tātad kopumā mans darba mērķis ir izpētīt PHP un MySQL, lai rezultātā atvieglotu lietotājiem izvēli, kādu programmēšanas valodu un datubāzi izmantot projektu realizācijai. Lai sasniegtu šo mērķi, ir jāveic sekojoši uzdevumi:

1) Apskatīt izstrādes aspektu – datu glabāšana. Ir jānoskaidro, kāda iespēja ir vairāk uzticama, kā arī veiktspējas novērtēšana, pie kādiem datiem varam nokļūt ātrāk, vai

varam saņemt datus pēc kāda kritērija, vai vairākiem lietotājiem ir iespēja vienlaicīgi sūtīt savus datus un kādu datu apjomu var glabāt;

2) Apskatīt izstrādes aspektu – datu šifrēšana. Ir jānoskaidro, kādu no iespējām labāk izmantot datu šifrēšanai, tas ir kāda no iespējām labāk aizsargā datus;

3) Apskatīt izstrādes aspektu – lietotāja autentifikācija. Ir jānoskaidro, kādas funkcijas pastāv, lai varētu realizēt šīs iespējas ar PHP un MySQL, un vai vispār tās var realizēt, kā arī jānoskaidro, kura no iespējam ir vairāk droša, uzticama, uzturama, saprotama, lietojama, verificējama, pareiza, kā arī novērtēt veiktspēju;

4) Apskatīt izstrādes aspektu – informācijas apmaiņa starp klientu un serveri. Ir jānoskaidro, kādi ir plusi un mīnusi, strādājot ar interneta veikalu, kas bija izstrādāts ar PHP un MySQL, kā arī noskaidrot, kā informācijas apmaiņu padarīt par efektīvu, saprotamu, lietojamu un uzticamu;

Darbs tika sadalīts vairākās nodaļās:

Pirmajā nodaļā tiek aprakstīta teorētiskā daļa – par PHP, MySQL, par to vēsturi, pielietošanu, darbību un versijām, kā arī salīdzinājums ar citām programmēšanas valodām un datubāzēm;

Otrajā nodaļā tiek aprakstīta praktiskā daļa – pētītas PHP un MySQL iespējas un aprakstīti daži piemēri, kas palīdzēs rezultātā noskaidrot, kāpēc PHP un MySQL ir ērtas projektu realizācijai. Stāstot par iespējām un piemēriem, tika aprakstītas to priekšrocības un trūkumi.

Trešā, ceturtā un piektā nodaļas satur rezultātus, secinājumus un izmantotās informācijas avotus attiecīgi.

Darbā ir pieejamas diagrammas, tabulas, attēli un koda fragmenti, kuri ir saistīti ar pētījumu un tīmekļa vietnes izveidošanas piemēriem.

Tiek lietots WAMP serveris vietnes attēlošanai un datubāzes pieslēgšanai.

1. TEORĒTISKĀ DAĻA

Šajā nodaļā [25] tiek aprakstīta vispārēja informācija par PHP programmēšanas valodu un MySQL datubāzi, par to, kas tās vispār ir, kāda ir viņu vēsture, par to pielietošanu, darbību un versijām. Tiek arī aprakstīts PHP salīdzinājums ar dažām citām programmēšanas valodām un MySQL salīdzinājums ar mSQL un PostgreSQL datubāzēm.

1.1. Kas ir PHP?

PHP [5] ir skriptu valoda ar atvērto programmas pirmkodu (Open Source). Tā praktiski tiek lietota, lai veidotu dinamiskas tīmekļa vietnes, ir pieejama vairumam operētājsistēmu, un tajā ir iekļauts daudz mūsdienīgu serveru un vairāku datubāzu atbalsts. PHP kodu var ieviest HTML lappusē. PHP kods tiek interpretēts ar tīmekļa serveri un ģenerē HTML kodu. Tai laikā, kad HTML kodu nolasa lietotāja interneta lapu pārlūkprogramma, tāda kā Mozilla Firefox, PHP programmas kodu nolasa un izpilda tīmekļa servera programmatūra. PHP spēj veikt XML un XHTML failu automātisko ģenerāciju, saglabājot tos servera failu sistēmā. Tā kā PHP kods ir „Open source", to var izmantot, izmainīt un izplatīt citiem lietotājiem.
Mūsdienās PHP ir viena no populārākajām tīmekļa lapu veidošanas valodām, pateicoties savai vienkāršībai, izpildīšanas ātrumam, funkcionalitātei un izejošo kodu izplatīšanai uz PHP licences bāzes.
Izvēloties PHP, lietotājs saņem tīmekļa servera un operētājsistēmas izvēles brīvību.

.

1.1.1. Vēsture

PHP autors ir dāņu programmētājs Rasmuss Lerdorfs [11]. 1994.gadā viņš izveidoja nelielu „Perl" skriptu kopumu savai tīmekļa vietnei un nosauca to par „Personal Home Page Tools". Skriptam bija atvērtais programmas pirmkods un tā uzdevums bija attēlot Lerdorfa rezumējumu, izsekot, kas to lasa, un vākt tīmekļa lapas apmeklētāju statistiku. To arī mēdz saukt par PHP programmēšanas valodas radīšanas sākumu.

Lai potenciāliem lietotājiem būtu iespēja apskatīt Lerdorfa pilno rezumējumu versiju un vācamo statistiku, viņš sūtīja viņiem URL norādi. Apmeklētāju sekošanai viņš izveidoja uz „Perl" valodas CGI skriptu, kuru ielika tīmekļa lappuses HTML tagā.
1995.gadā skripti tika pārrakstīti un papildināti ar jaunām iespējam „C" valodā.
Lerdorfs paplašināja un uzlaboja savu personīgo izstrādi saņemot atsauksmes no lietotājiem.

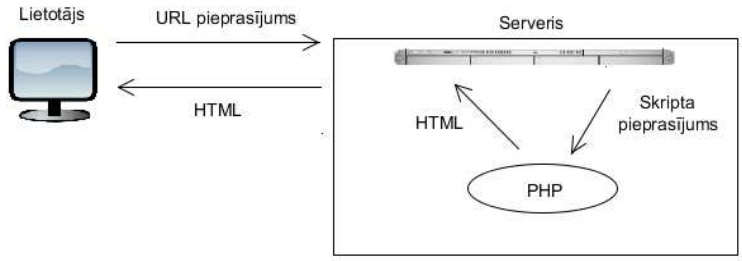
1.1.2. Pielietošana

PHP valodas popularitāte ir saistīta ar dinamisku tīmekļa vietņu izveidošanu. PHP ir labāk, ātrāk un vieglāk iemācīties, nekā citas valodas. Tā iekļauj sevī iebūvētos līdzekļus lielā salikumā, lai izstrādātu tīmekļa pielikumus. Daži pamat līdzekļi [13] ir:
1) mijiedarbība ar lielu datubāzu daudzumu;
2) GET un POST parametru automātiskā izdabūšana;
3) Ielādētu uz servera failu apstrāde;
4) Darbs ar „cookies" un sesijām;
5) Darbs ar HTTP autorizāciju.
Šodien PHP pielieto vairāk nekā simts tūkstoši izstrādātāji. PHP tiek pielietots trīs pamat apgabalos [12]:
1) Tīmekļa veidošana. Nozare, kurā PHP tiek pielietots visplašāk. Ir nepieciešams PHP parsētājs (programma, kas sadala lielus datu blokus mazākās, vieglāk interpretējamās daļās), tīmekļa serveris un pārlūkprogramma. Jābūt uzinstalētam PHP un strādājošam tīmekļa serverim, lai būtu iespēja apskatīt PHP skriptu darbības rezultātus pārlūkprogrammā;
2) Skriptu veidošana komandu rindai. Tas nozīme, ka var izveidot PHP skriptu, kurš var darboties neatkarīgi no tīmekļa servera un pārlūkprogrammas. Salīdzinot ar tīmekļa veidošanu, ir vajadzīgs tikai PHP parsētājs. Uz Windows platformām ar „Task Scheduler" palīdzību regulāri pildāmiem skriptiem ļoti noder šis pielietošanas veids;
3) GUI pielikumu veidošana. Šī pielikuma veidošana tiek lietota reti un vairāk attiecās uz lietotājiem, kuri grib pārzināt PHP.

1.1.3. Darbība

PHP izpildās uz servera. To atvērto pirmkodu nav iespējams ieraudzīt pārlūkprogrammā, kura sūta serverim pieprasījumu uz lappusi ar PHP kodu, serveris padod lappusi PHP interpretatoram, lai uzģenerētu HTML kodu, kurš tiek padots atpakaļ serverim, un tad serveris sūta lappusi lietotājam. Rezultātā, viss, ko lietotājs redz, ir HTML kodā izvadīta lappuse. PHP pēc skripta izpildīšanas gaida uz servera jauno pieprasījumu [1].

Zemāk ir piedāvāts shematisks izskats, kā šie procesi izpildās:



1.1.3.1. att. **Lietotāja-servera mijiedarbība tīmekļa lappuses pieprasīšanas laikā, kad PHP kods tiek pārveidots HTML kodā**

Ja lietotājs pieprasītu HTML lapu, tad serveris vienkārši saņemtu HTML kodu un sūtītu lappusi lietotājam.

1.1.4. Versijas

Tabulā 1.1.4.1. ir īsi pastāstīts par PHP versijām, kuras bija izstrādātas 10 gadu laikā [14].

1.1.4.1. tabula

PHP programmēšanas valodas versijas

Nosaukums	*Apraksts*
PHP	Personal Home Page Tools. Saturēja skriptu kopumu.
PHP/FI	Personal Home Page / Forms Interpreter. Pievienots datubāzu atbalsts. PHP/FI iekļauj sevī automātisko formu apstrādi, iekļaušanu HTML tekstā, mainīgo noformēšanu „Perl" stilā un citas funkcijas.
PHP 3	Versija, kura ir līdzīga mūsdienu PHP valodai. Varēja strādāt ar lielo datubāzu skaitu, protokolu un atbalstīt lielo API skaitu.

Nosaukums	*Apraksts*
PHP 4	Iekļauj sevī sesiju atbalstu, drošu apstrādes iespēju ievadītai informācijai un citas iespējas. Ieguva palielināto ātrdarbību.
PHP 5	XML atbalsts. Ieviestas atvērtas, aizvērtas un aizsargātas metodes, saskarne un objektu klonēšana.

1.2. Ievads par PHP

Šajā apakšnodaļā tiek pastāstīts par PHP valodas pamat sintaksi, komentāru rakstīšanu, mainīgo tipiem, konstantēm, operācijām, operatoriem un cikliem.

1.2.1. Pamat sintakse

Kā agrāk bija minēts, PHP kodu var ieviest HTML lappusē, un tas ļauj rakstīt abus kodus vienā failā. PHP skriptu bloks sākas ar „<?php" tagu un beidzas ar „?>" tagu (šo PHP deskriptoru sauc par XML stilu). Tekstu, kas atrodas starp šiem diviem tagiem, tīmekļa serveris uztver kā PHP, bet teksts, kas atrodas ārpus PHP tagiem, tiek sūtīts uz tīmekļu pārlūkprogrammu kā regulārais HTML [1].

Piemērā zemāk tiek uzrakstīts vienkāršais skripts, kurš sūta „This is HTML" pārlūkprogrammai kā HTML tekstu un „This is PHP" kā PHP tekstu:

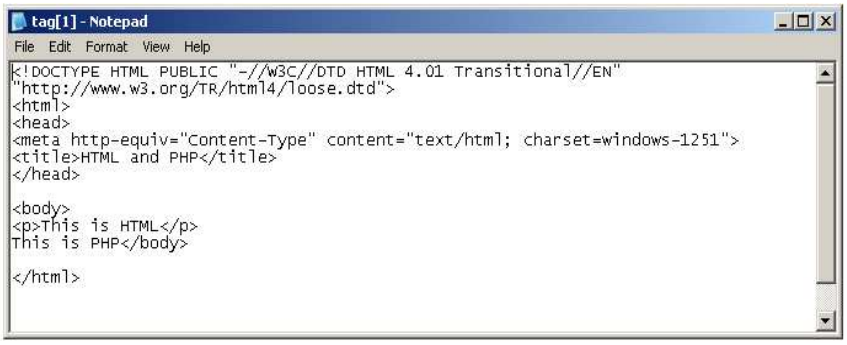
```
<html>
<body>
<p>This is HTML</p>
<?php echo „This is PHP"; ?>
</body>
</html>
```

Rezultātā pārlūkprogramma mums izvadīs uz ekrāna:

1.2.1.1. att. **HTML un PHP teksta izvads pārlūkprogrammā**

Teksta izvadei PHP kodā tiek izmantots apgalvojums „echo”.

Ja lietotājs apskata lappuses pirmkodu, viņš redzēs HTML izveidoto lapu bez jebkādiem PHP tagiem:



1.2.1.2. att. **Lappuses pirmkods**

1.2.2. Komentāru rakstīšana

Komentāri ir nepieciešami kodu kopumu aprakstīšanai, lai autoram vai citiem lietotājiem būtu skaidrs, kā kods strādā. Atšķirībā no HTML komentāriem, kurus jebkurš lietotājs var ieraudzīt lappuses pirmkodā, PHP komentāri netiek sūtīti tīmekļa pārlūkprogrammā un tos nav iespējams ieraudzīt pirmkodā.

PHP atbalsta trīs sekojošus stila komentārus [1]:

1) Pirmais izmanto simbolu „#”, piemēram – # Tas ir pirmais komentāru stils;

2) Otrais izmanto „//” simbolus, piemēram – // Tas ir otrais komentāru stils;

3) Trešais izmanto „/* ... */” simbolus, lai būtu iespēja uzrakstīt komentārus vairākās rindās, piemēram:

/* Tas ir trešais komentāru stils.

Tas tiek rakstīts vairākās rindās. */.

Pirmais un otrais stils ļauj rakstīt komentārus tikai vienā rindā. Tālāk tiek piedāvāts skripts ar HTML un trīs PHP stilu komentāriem:

<html>

<body>

<!-- HTML comment. -->

* This is HTML *

<?php

 # First PHP comment.

// Second PHP comment.

/ Third PHP comment*

*in two lines. */*

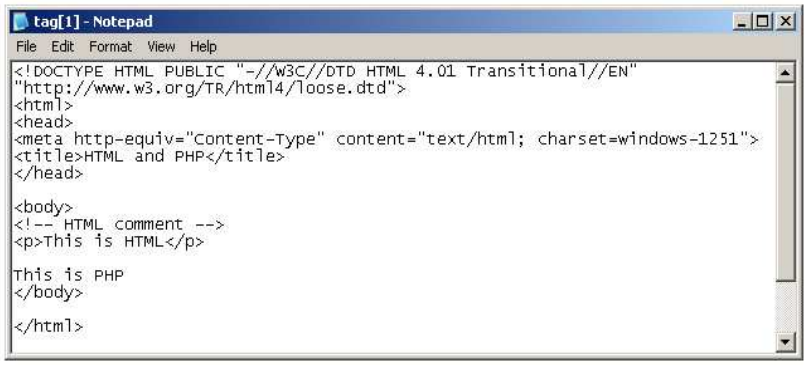
echo "This is PHP";

?>

</body>

</html>

Kā redzams attēlā 1.2.2.1., apskatot lappuses pirmkodu, HTML komentārs tiek izvadīts, bet PHP komentāri nav:



1.2.2.1. att. **Lappuses pirmkods pielietojot HTML un trīs PHP stilu komentārus**

1.2.3. Mainīgo tipi

PHP satur 8 mainīgo tipus – četri no tiem ir skalāri tipi, divi sajauktie tipi, kā arī „resource" (resursu) un „NULL" tipi [2].

Skalārie tipi ir:

1) „Boolean" – glabā „true" (patieso) un „false" (aplamo) nozīmes;

2) „Integer" – veseli skaitļi;

3) „Float" – decimāli skaitļi;

4) „String" – simbolu rindas.

Sajauktie tipi ir:

1) „Array" – masīvi;

2) „Object" – objekti.

Katram mainīgā vārdam jāsakās ar simbolu „$", piemēram, „$name". Mainīgā vārds var sastāvēt no simbolu, ciparu un „_" zīmes kombinācijas, piemēram, „$name_number1". Pēc „$" zīmes pirmais simbols nedrīkst būt cipars. Ja burtu reģistrs atšķiras, tad mainīgie ir dažādi, piemēram, „$name" un „$Name" ir divi dažādi mainīgie.

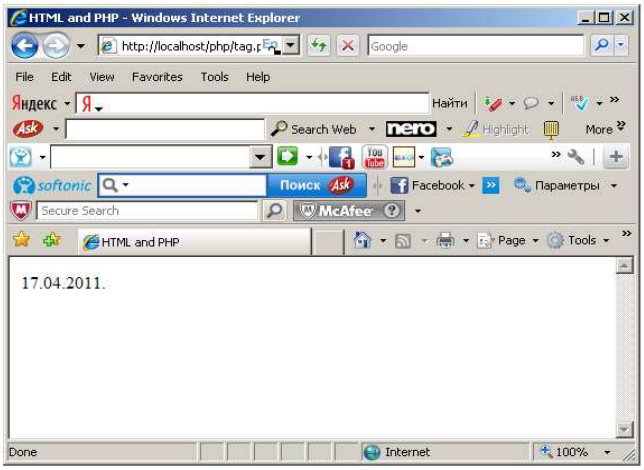
1.2.4. Konstantes

Konstante ir vērtību identifikators, kuru var definēt ar funkcijas „define()" palīdzību. Konstantei nav „$" simbolu priekšā, un kad tā ir definēta, to vērtību jau nevar nomainīt. Vērtības konstantei var būt tikai simboli un cipari [1].

Zemāk tiek uzrakstīts skripts konstantes izmantošanai:

```
<html>
<body>
<?php
define ('Diena', '17.04.2011.');
echo constant('Diena');
?>
</body>
</html>
```

Rezultāta uz ekrāna tiek izdrukāts datums:



1.2.4.1. att. **Konstantes vērtības izvads**

1.2.5. Operācijas

PHP valodā ir iespējams izmantot aritmētiskās, piešķiršanas, salīdzināšanas, loģiskās un citas operācijas. Nākošajā lappusē ir izveidota tabula, kurā ir pieejama informācija par operācijas izskatu, aprakstu un uzrakstīts piemērs katrai operācijai. Tabulā nav iekļautas visas iespējamās PHP operācijas.

1.2.5.1. tabula

PHP programmēšanas valodas operācijas

Operācija	*Apraksts*	*Piemērs*
Aritmētiskās operācijas		
+	Saskaitīšana	<?php $a=1; $a=$a+5; echo $a; ?> (rezultāts ir 6)
-	Atņemšana	<?php $a=5; $a=$a-3; echo $a; ?> (rezultāts ir 2)
*	Reizināšana	<?php $a=2; $a=$a*4; echo $a; ?> (rezultāts ir 8)
/	Dalīšana	<?php $a=6; $a=$a/4; echo $a; ?> (rezultāts ir 1.5)
%	Modulis (atlikuma atrašana)	<?php $a=7; $a=$a%4; echo $a; ?> (rezultāts ir 3)
++	Palielināšana	<?php $a=4; $a++; echo $a; ?> (rezultāts ir 5)
--	Samazināšana	<?php $a=4; $a--; echo $a; ?> (rezultāts ir 3)
Salīdzināšanas operācijas		
= =	Ir vienāds	<?php $a=4; $b=5; if ($a==$b) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘false’)
!=	Nav vienāds	<?php $a=4; $b=5; if ($a!=$b) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘true’)
>	Ir lielāks	<?php $a=5; $b=3; if ($a>$b) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘true’)
<	Ir mazāks	<?php $a=5; $b=3; if ($a<$b) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘false’)

Operācija	*Apraksts*	*Piemērs*
Loģiskās operācijas		
&&	And	<?php $a=5; $b=3; if ($a<6 && $b>1) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘true’)
||	Or	<?php $a=5; $b=3; if ($a<5 || $b>5) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘false’)
!	Not	<?php $a=5; $b=3; if (!($a= =$b)) {echo 'true'; } else {echo 'false'; } ?> (rezultāts ir ‘true’)

1.2.6. Operatori un cikli

Lai lietotājs varētu pievienot darbībai vairākus iespējamos risinājumus, PHP valodā ir piedāvāti sekojoši operatori un cikli [6].

Operatori:

1) „if” – ja izteiksme ir patiesa, tad koda bloks, kas tai seko, izpildās;

2) „if...else” – ja izteiksme ir patiesa, tad izpildās pirmā koda daļa, ja izteiksme nav patiesa, tad izpildās otrā koda daļa;

3) „if...elseif...else” – pielieto, lai būtu iespēja izmantot vienu no vairākiem koda blokiem;

4) „switch” – līdzīgs „if” operatoram, tikai „switch” operatorā nosacījums var pieņemt jebkādu daudzumu atšķirīgu nozīmju tajos gadījumos, kad viņa aprēķināšanas rezultāts pieņem vienkāršu tipu („integer”, „string”, vai „float”).

Cikli:

1) „while” – cikls izpildās, kamēr izteiksme ir patiesa;

2) „for” – „while” cikla kompakta forma;

3) „foreach” – cikls izpildās katram elementam no masīva;

4) „do...while” – atšķirībā no „while” cikla ir tāda, ka nosacījums tiek pārbaudīts beigās.

Ar šo es beidzu teorijas stāstījumu par PHP programmēšanas valodu un tiecos klāt pie MySQL datubāzes aprakstīšanas.

1.3. Kas ir MySQL?

MySQL [7] ir ātra un viena no populārākām relāciju datubāzu vadības sistēmām. Tā ir lietotne, kas sarakstīta C++ valodā, ar atvērto pirmkodu, kuru var izmantot un modificēt (pirmkodu ir iespējams lejupielādēt). Datubāze ļauj glābāt, meklēt, kārtot un izņemt informāciju. MySQL rūpējas par informācijas drošību un ļauj to izdabūt un izvietot ar vienas rindas palīdzību. Dati, kuri glabājās datubāzē var būt jebkādi – pasūtījumu saraksts, attēlu kopums vai liels informācijas daudzums. Informācijas dabūšana no datubāzes notiek ātrāk, nekā no faila. MySQL programmatūra sastāv no dažiem komponentiem, tādiem, kā MySQL serveris (kas atbild par datubāzu darbināšanu un vadību), MySQL klients (kas atbild par saskarni serverim) un daudz utilītu uzturēšanai. Kopā ar PHP programmēšanas valodu, kura satur lielu funkciju kopumu strādājot ar datubāzēm, MySQL veido labu mijiedarbību lielu projektu realizēšanas skaitam.

1.3.1. Vēsture

1979. gadā Zviedrijas izgudrotājs Montijs Videniuss [10] izstrādāja datubāzes vadības līdzekli, kuru nosauca UNIREG. Tā tika modificēta katru gadu, un 1994. gadā „TcX" kompānija to izmantoja, lai izstrādātu pielikumu pasaules tīmeklim. Pielietojot UNIREG un mSQL datubāzes utilītus, izstrādātājiem izdevās izveidot API savai sistēmai. Veidojot datubāzes pirmkoda izmaiņas gadu laikā, jau 1995.gadā parādījās MySQL datubāze versija 1.0, kura bija pieejama tikai „TcX" kompānijai. Tirgū MySQL parādījās 1996. gadā.

1.3.2. Pielietošana

Šodien MySQL [4] ir uzstādīts uz vairākiem miljoniem datoru visā pasaulē. Viens no tās popularitātes iemesliem ir integrēšana ar PHP programmēšanas valodu, kura kopā ar Apache un MySQL veido labāku sasiešanas ražotspēju. MySQL ir ātra un ērta izmantošanā, to biežāk lieto tīmekļa vietnes informācijas glabāšanai, kuru var dabūt, padodot datubāzes serverim nepieciešamo vaicājumu. Datubāze ir vajadzīga, lai varētu strādāt ar tabulām – veidot, papildināt, dzēst tabulas, ievadīt tās datus (kurus arī var papildināt un dzēst), apstrādāt vaicājumus un citas funkcijas. Pirms lietotājam

parādās iespēja strādāt ar informāciju, kas atrodas MySQL datubāzē, viņam vajag savienoties ar datubāzes serveri.

1.3.3. Darbība

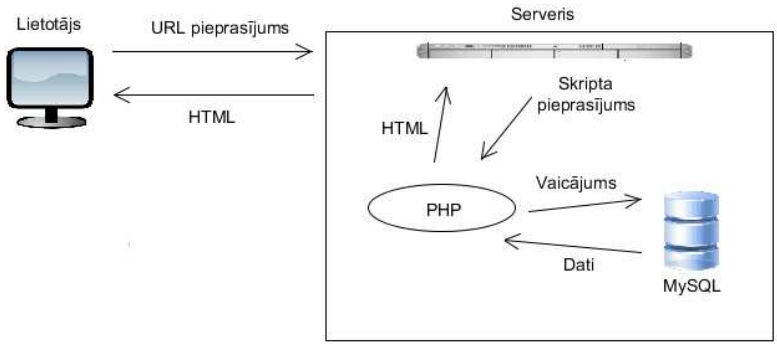
MySQL darbojas uz vairākām dažādām sistēmas platformām, ieskaitot Linux, Windows, Mac OS X, OpenBSD un citām [1].

Lai varētu izvilkt informāciju no datubāzes ir nepieciešams pieslēgties serverim. MySQL servera pieslēgšanai tiek izmantota funkcija „mysql_connect", bet datubāzes pieslēgšanai – „mysql_select_db", kuras tiek rakstītas PHP failā. Kods ir sekojošs:

```
<?php
$db = mysql_connect("Hostname","User","Password");
mysql_select_db("DatabaseName",$db);
?>
```

„Hostname" ir domēna vārds, kas identificē konkrētu datoru kāda tīkla domēna apakšdomēnā. „User" ir lietotāja vārds, kas ir servera procesa vadītājs. „Password" ir lietotāja parole. „DatabaseName" ir datubāzes nosaukums, kurai lietotājs grib pieslēgties.

Zemāk ir piedāvāts shematisks izskats, kā tiek veikts vaicājums un datu gūšana no MySQL datubāzes:



1.3.3.1. att. **Dinamisku tīmekļu vietnes darbība, lietojot PHP un MySQL**

1.3.4. Versijas

Nākošajā lappusē tabulā 1.3.4.1. ir īsi pastāstīts par dažām MySQL versijām [8], kuras bija izstrādātas 15 gadu laikā.

1.3.4.1. tabula

MySQL datubāzes versijas

Nosaukums	*Apraksts*
MySQL	1995.gads. Izstrādāta pirmā versija.
MySQL 3.23	2001.gads. Tika iekļauts InnoDB atbalsts.
MySQL 4.0	2003.gads. Parādījās iespējas – transakcija, InnoDB tabulas, unikoda atbalsts, SSL atbalsts, datubāzes bibliotēka.
MySQL 4.1	2004.gads. Parādījās ātrākais klientu-servera protokols ar gatavo vaicājumu atbalstu.
MySQL 5.0	2005.gads. Kursoru, procedūru, funkciju, trigeru un apskatu pielietošana.
MySQL 5.1	2008.gads. Segmentēšana, darba plānotājs, papildināšana ar funkcijām XML apstrādei.
MySQL 5.5	2010.gads. Pārstrādāta iekšējo bloķēšanu sistēma [9], jauns mehānisms vaicājumu optimizācijai, RANGE un LIST operācijas.

1.4. Programmatūras kvalitāte

Programmatūras kvalitāte [26] ir galvenais aspekts, veidojot produktu. Pastāv daudz programmatūras vēlamās īpašības. Dažas no tām piemērojamas kā pašam produktam, tā arī procesam produkta radīšanai. Lietotājs vēlas, lai programmatūra būtu uzticama, efektīva un viegli pielietojama. Izstrādātājs vēlas, lai tā būtu verificējama, uzturama, pārnesama un paplašināma. Projekta menedžeris vēlas, lai programmatūras attīstības process būtu produktīvs un viegli kontrolējams.

Programmatūras izstrādāšanai pielieto noteiktu procesu. Vajag, lai produkts būtu lietotājdraudzīgs un lai process būtu efektīvs. Ja process prasa rūpīgu testu datu plānošanu pirms sistēmas dizaina un izstrādes sākuma, programmatūra kļūst uzticamāka. Bez tam, ir īpašības, tādas kā efektivitāte, kas attiecas kā uz procesu, tā arī uz produktu. Šeit būtu lietderīgi izanalizēt vārdu „produkts”. Parasti tā sauc to, ko piegādā pircējam. Lai produkts būtu kvalitatīvs un atbilstu parastām pircēja prasībām, labāk izmantot PHP programmēšanas valodu un MySQL datubāzi.

Realizējot produktus, jānodrošina dažas programmatūras kvalitātes, tādas kā pareizība, uzticamība, veiktspēja, lietojamība, verificēšana, drošība, uzturamība, atkārtota lietojamība, pārnesamība, saprotamība, produktivitāte un citas.

Ar definīciju „pareizība" tiek pieņemts, ka specifikācijas sistēmai ir pielietojamas un ka ir iespējams bez kļūdām noteikt, vai programma atbilst specifikācijām. Šādas specifikācijas mūsdienu programmatūrām reti sastopamas. Pareizība ir programmatūras un tās specifikācijas atbilstība.

Produktam ir jābūt uzticamam. Produkti, kuriem nevar uzticēties, ātri pazūd no tirgus. Diemžēl, kad runa ir par programmatūrām, šie produkti vēl nav ieguvuši uzticības statusu. Tos bieži vien izplata ar esošām kļūdām. Programmatūras lietotāji zina, ka programmai var būt kļūdas.

Katrs produkts strādā noteiktajā veiktspējas līmenī. Programmatūras veiktspēja ir samērā augsta, bet ir atkarīga no datora resursiem. Veiktspēja ir kvalitāte, kurai būtu jāatbilst lietotāja prasībām. Resursu efektīva izmantošana ietekmē sistēmas veiktspēju. Veiktspēja ietekmē sistēmas pielietošanas iespējas. Ja programmatūra darbojas pārāk lēni, tā ietekmē lietotāja darbu un neatbilst viņa prasībām. Veiktspēja ietekmē programmatūras mērogojamību.

Programmatūra ir lietojama, ja cilvēks, kas to lieto, uzskata, ka tā ir vienkārša pielietošanā. Šīs apzīmējums atspoguļo lietojamības subjektīvo būtību. Saskarne ir svarīgs komponents lietotājam. Īpašības, kas iesācējam ir pieņemamas, atšķiras no tām, ko vēlas pieredzējis lietotājs.

Programmatūra ir verificējama, ja tās īpašības var viegli verificēt. Piemēram, ir svarīgi, lai būtu iespējams verificēt programmatūras veiktspēju vai pareizību. Verifikāciju var veikt, pielietojot formālu vai neformālu analīzes metodiku vai arī testējot to.

Sistēma ir droša, ja tā apkalpo tikai autorizētus lietotājus, un aizsargā šo lietotāju tiesības un informāciju. Drošība ir svarīga jebkuras sistēmas lietošanā, tā ir svarīga arī publiskajos tīklos. Drošība ir informācijas sistēmu svarīga īpašība, kurai lietotāji uztic savu datu aizsardzību.

Terminu „programmatūras uzturēšana" parasti lieto sakarā ar modifikācijām, ko veic programmatūrā pēc tam, kad tā sāk darboties. Uzturēšanu agrāk uzskatīja par „kļūdu labotāju", un bija rūgti apzināties, ka tik daudz pūļu tika veltīts vienkārši defektu labošanai.

Produkta evolūcija nosaka, ka mēs liekam produktam veidot šī paša produkta jaunu versiju, produkta atkārtotā lietošana nozīmē, ka produkts – ar nelielām izmaiņām – veido citu produktu. Atkārtota lietošana var būt pielietota dažādos detalizēšanas pakāpes līmeņos, taču izrādās, ka to labāk pielietot programmatūras komponentiem nekā visam produktam kopumā.

Programmatūra ir pārnesama, ja to var lietot dažādās vidēs. Termins „vide" attiecās uz aparatūru vai programmatūru, tādu kā noteiktu operētājsistēmu. Pārnesamība ir ekonomiski svarīga, jo tā palīdz amortizēt ieguldījumus programmatūras sistēmā dažādās vidēs un vienas vides dažādās paaudzēs.

Saprotamība ir produkta iekšējā īpašība, un tā palīdz panākt citas īpašības, tādas kā verifikācijas darbību. Lietotājs uzskata sistēmu par saprotamu, ja tās uzvedība ir paredzama. Ārējā saprotamība ir faktors produkta lietojamībā.

Produktivitāte ir programmatūras izstrādāšanas procesa īpašība, ko var attiecināt uz tās efektivitāti un veiktspēju. Efektīva procesa rezultāts ir produkta ātrāka piegāde.

Dotos kritērijus es pielietoju apakšnodaļā 1.5., kurā salīdzināju PHP ar citām programmēšanas valodām un MySQL ar citām datubāzēm, kā arī praktiskajā daļā, kurā tika pētīti ērtākas iespējas projektu realizācijai.

1.5. Salīdzinājumi

Šajā apakšnodaļā tiek aprakstīts PHP programmēšanas valodas salīdzinājums ar Perl, ColdFusion un ASP, kā arī MySQL salīdzinājums ar mSQL un PostgreSQL datubāzēm.

1.5.1. PHP salīdzinājums ar citām programmēšanas valodām

1. Ja salīdzina PHP un Perl programmēšanas valodas, tad PHP [16] priekšrocība ir tāda, ka šī valoda tika izstrādāta skriptu rakstīšanai tīmekļa serveriem, bet Perl valoda tika izstrādāta, lai darītu daudz vairāk, tāpēc arī Perl valodu uzskata par sarežģītu. Tas attiecās uz Perl kodu, kuru ir sarežģīti saprast, ja to ir uzrakstījis cits programmētājs. PHP valodas struktūra ir vairāk saprotama, un to ir vieglāk ieviest HTML lappusē. PHP satur Perl funkcionalitātes – konstrukcijas, sintaksi un citas funkcijas, tai laikā tā nav sarežģītāka par Perl. Atšķirībā no Perl valodas, PHP attīstās ātrāk un turpina progresēt.

Teorētiski Perl (programmēšanas valoda) ir universāls rīks ļoti paplašinātu uzdevumu risināšanai un tā netika izstrādāta tīmekļa programmēšanai. PHP jau pašā sākumā tika izstrādāts tīmekļa veidošanai.

Perl satur sekojošu kodolu – funkciju un noteikumu kopumu, kuri nav atkarīgi no platformas, operētājsistēmas un citām lietām, bet PHP valodai šāda kodola nav. No tā izriet PHP īpašību skaits.

Pirmā īpašība ir pārnesamība. PHP funkciju kopums praktiski ir atkarīgs no sniedzēja. „Pareizie" sniedzēji vienmēr raksta, ar kādām opcijām tika vākts viņu PHP. Ja PHP vietne ir izstrādāta ar funkciju palīdzību, kuras mitinātais nepiedāvā, tad paplašināt PHP funkciju kopumu nav iespējams. Perl šajā nozīmē ir vairāk stabils. Tā ir vienāda visur un to kods ir vairāk pārnesams.

Nākamā īpašība ir funkciju skaits. PHP piedāvā lielu funkciju skaitu. To šobrīd ir vairāk par 3000. Reālajā mitinātajā var ieraudzīt aptuveni 1000 no viņām. Tik liels kopums neiet PHP valodai par labu. Daži programmētāji zina dažādu operatoru kopumu. Tas sarežģī svešu kodu lasīšanu, kodu apmaiņu un savstarpējo izstrādāšanu. Perl valoda piedāvā programmētājam ierobežotu standartu rīku kopumu. Tas atvieglo kodu apmaiņu. Programmētājs vienmēr var saņemt nepieciešamo iespēju, pieslēdzot vajadzīgo moduli.

Perl piedzīvo mazāk izmaiņu pārejot no versijas uz versiju un satur vairāk attīstītu izstrādāšanas līdzekļu.

Runājot par tīmekļu izstrādāšanu PHP valodai paradās sekojošas priekšrocības:

1) PHP interpretators integrējas tīmekļa serverī, kas palielina PHP produktivitāti. Perl jāizmanto „mod_perl" moduli, lai panāktu PHP, bet vairāki sniedzēji nepiešķir šādu iespēju;

2) PHP kļūdas tiek paziņotas klientam, bet nerakstās logā „error_log";

3) PHP satur lielo iebūvētu funkciju kopumu, lai strādātu ar HTTP protokolu. Perl programmētāji var saņemt visas šis funkcijas, pieslēdzot vajadzīgo moduli, taču PHP valodā šīs funkcijas ir jau iebūvētas.

Taču visām šīm priekšrocībām ir arī trūkumi, kas saistās ar resursu drošību:

1) PHP ir iebūvēts serverī, kas sarežģī uzbrukumu avotu diagnostiku. PHP skripts var būt izvietots jebkurā servera direktorijā, kas izveido papildinātu risku. Servera apmeklētājs saņem iespēju augšupielādēt serverī ne tikai bildes vai citus failus, bet arī PHP skriptus, kas ir bīstami. PHP paziņo par kļūdām atbildot uz HTTP vaicājumu, kas ir ērti izstrādātājam un bīstami drošībai, jo jebkurš apmeklētājs var uzzināt par

kļūdām mūsu programmās. Ļaundaris var uzzināt jūsu PHP versiju un uzlauzt jūsu resursu.

Tika konstatētas sekojošas priekšrocības un trūkumi salīdzinot PHP un Perl programmēšanas valodas [26]:
– PHP interpretators integrējas tīmekļa serverī, kas palielina PHP produktivitāti. Perl jāizmanto „mod_perl" moduli, lai panāktu PHP, bet vairāki sniedzēji nepiešķir šādu iespēju;
– PHP kļūdas tiek paziņotas klientam, bet nerakstās logā „error_log". Tas ir ērti izstrādātājam, bet bīstami no drošības viedokļa, jo jebkurš lietotājs var uzzināt par kļūdām mūsu programmās;
– PHP satur lielu funkciju kopumu, lai strādātu ar HTTP protokolu, bet Perl valodā vajag ieslēgt moduli, lai šīs funkcijas būtu pieejamas;
– Perl kods ir vairāk pārnesams.

Perl ir labāka par PHP kopumā, jo, neskatoties uz kopīgiem plusiem, tā ir drošāka. Taču ja runa iet par tīmekļa lapas veidošanu, kura ietvers sevī lielu apmeklēšanu un tās budžets būs ierobežots, tad labāk izmantot PHP valodu.

2. ASP [16] nav īsti programmēšanas valoda – tā ir Microsoft tehnoloģija, kas ļauj dinamiski veidot tīmekļa lappuses uz servera puses. Valodas, kuras lieto ASP programmēšanai, ir VBScript nu JScript. Lielākais ASP trūkums ir tas, ka viņš strādā tikai uz sistēmām, kuras lieto Microsoft Internet Information Server. VBScript valodas sintakse ir nepatīkamāka salīdzinot ar PHP, bet tā interpretatora ražotspēja ir augstāka nekā PHP valodai. JScipt ir cita problēma – nepatīkamo OLE Automation datu tipu apstrāde, kas ved pie grūtībām kļūdu atrašanā. Ir viedokļi, ka ASP ir daudz lēnāks un masīvāks salīdzinājumā ar PHP. ASP priekšrocība ir tāda, ka to var viegli lietot cilvēki, kuri spēj strādāt ar Visual Basic.

Gan ASP, gan PHP tiek pielietotas, lai veidotu tīmekļu vietnes. Atšķirībā no statiskām HTML tīmekļa lappusēm, ASP un PHP tīmekļa vietnes ir dinamiskas un ļauj lietotājiem veikt informācijas apmaiņu pielietojot vietņu datubāzes. ASP ir nepieciešams Microsoft serveris vietnes darbībai, bet PHP strādā, izmantojot Linux vai Unix serveri. PHP programmas strādā arī uz Windows un Solaris, bet ASP var strādāt tikai uz Windows dibinātām platformām. Ja programmētājs agrāk lietoja C++

valodu, tad viņam būtu ērtāk lietot PHP, jo PHP izmanto C/C++ kā bāzes valodu, un vairākas sintakses ir vienādas. Tā kā vairāki joprojām izmanto C++, PHP ir vairāk populārs nekā ASP. ASP sintakse un saskarne ir līdzīga Visual Basic programmēšanai. Tas ir tāpēc, ka Visual Basic ir pamatā saistīts ar Microsoft produktiem un programmām. PHP lietošana ir bezmaksas, tai laikā, ka par ASP programmām jāmaksā. PHP ir ļoti elastīga, ja runā par pieslēgšanu datubāzēm. Tā var pieslēgties vairākām datubāzēm, kuras biežāk ir MySQL, un tā tāpat kā PHP nemaksā neko. ASP valodai nepieciešams iegādāties MS SQL, kas ir Microsoft produkts. Runājot par ātrumu, ja lietotājam tas ir nepieciešams, tad labāk izmantot PHP. PHP kods izpildās ātrāk nekā ASP kods, tāpēc, ka PHP kods izpildās savā atmiņā, tai laikā kad ASP izmanto serveri un uz COM pamatoto arhitektūru. Strādājot ar PHP, vairāki rīki, kas asociējas ar programmu, ir „Open Source" programmatūra, un par to nevajag maksāt. ASP valodai ir nepieciešams iegādāties papildus rīkus, lai strādātu ar to programmām.

Tika konstatētas sekojošas priekšrocības salīdzinot PHP un ASP [26]:

– Gan PHP, gan ASP tiek lietoti, lai izstrādātu dinamisku tīmekļu vietnes;
– Atšķirība no ASP, PHP lietošana ir bezmaksas;
– PHP ir ātrāks par ASP, kas palielina PHP produktivitāti;
– PHP satur daudz līdzekļu, lai vadītu un atbalstītu MySQL datubāzi;
– PHP atrastās kļūdas aptur sistēmas darbību, kas padara to par vairāk uzticamu valodu.

PHP ir labāka par ASP, jo tā ir vairāk uzticama un produktīvāka.

3. ColdFusion [16] valodai ir vairāk attīstīta kļūdu apstrādes sistēma un darbība ar datubāzes abstrakciju. ColdFusion stiprā puse ir tā meklētājprogramma, kaut arī tiek minēts, ka meklētājprogramma nav tas objekts, kam jābūt iekļautam tīmekļa servera skriptu valodā. PHP strādā gandrīz uz jebkuras platformas, bet ColdFusion tikai uz Win32, Solaris, Linux un HP/UX. ColdFusion ir labāks IDE un ar to ir vieglāk strādāt, bet PHP prasa sākotnējas zināšanas par programmēšanu. ColdFusion ir izstrādāta neprogrammētajiem, bet PHP – programmētājiem.

Atšķirībā no ColdFusion, PHP ir valoda ar atvērto pirmkodu un tai ir pieejams lielu skriptu skaits. ColdFusion ir ļoti viegli iemācīties un salīdzinot ar PHP to ir vieglāk lasīt. PHP projekta izstrādāšana var aizņemt ļoti daudz laika, jo viss ir saistīts

ar sintaksi un lielu kodu apjomu. ColdFusion projektiem ir nepieciešams mazāks koda apjoms un mazāk laika. ColdFusion labi integrējas ar Flex un satur „Custom Tags", kas atvieglo tīmekļu vietņu veidņu vadīšanu un pieejamību.

Tika konstatētas sekojošas priekšrocības un trūkumi salīdzinot PHP un ColdFusion [26]:
– ColdFusion ir ātrāks par PHP, kas palielina ColdFusion produktivitāti;
– ColdFusion ir vieglāks par PHP, kas palielina ColdFusion saprotamību, uzticamību un lietojamību;
– PHP nodrošina labāku veiktspēju un drošību, salīdzinot ar ColdFusion.

ColdFusion kopumā varētu būt labāka par PHP, jo tā ir vairāk saprotama, uzticama un to var ātri iemācīties. Taču ja runa iet par tīmekļa lapas veidošanu, tad labāk izmantot PHP valodu, jo tā ir drošāka.

1.5.2. MySQL salīdzinājums ar citām datubāzēm

1. mSQL [18] datubāze pateicoties mazu funkciju skaitam un vienkāršotai drošības sistēmai ātrāk par MySQL izpilda sekojošas operācijas:
1) Izveido un dzēš tabulas;
2) Atlasa informāciju;
3) Iesprauž informāciju tabulā, ja tā satur nelielu aiļu un atslēgu skaitu.
MySQL pārspēj mSQL datubāzi sekojoši:
1) Lielu apjomu rezultātu augšupielādes operācija;
2) Tabulu apstrādes operācija, kas satur lielu aiļu skaitu un garus ierakstus;
3) Sakārtošanas un grupēšanas operācijas.
Strādājot ar MySQL, katram pieslēgumam tiek veidota atsevišķa straume, kas ļauj katram pieslēgumam nesagaidīt cita pieslēguma pabeigšanu. mSQL datubāzē kad tiek uzstādīts viens pieslēgums, citiem jāgaida tās nobeigums. Kad pirmais pieslēgums beidzas, otrais sākas un pārējie gaida savu kārtu.

Par disku veida telpas izmantošanas efektivitāti. MySQL satur konkrētus datu tipus, ar kuru palīdzību var veidot tabulas, kas aizņem minimālu telpu. Kā piemērs labvēlīgais MySQL datu tips ir MEDIUMINT, kuru vērtības lielums ir 3 baiti. Ja datubāzē glabājas aptuveni 100 miljoni ieraksti, tad pat 1 baita ekonomijas vērtību

katram ierakstam ir grūti pārvērtēt. mSQL datubāzē ailu tipu izvēle ir nabadzīgāka, tāpēc arī samazināt tabulu izmēru ir grūti.
Par Perl saskarni. MySQL Perl saskarnes ir praktiski identiskas savam analogam no mSQL, kaut arī ir apveltīts ar dažām papildinātām iespējām.
Par izstrādāšanas ātrumu. MySQL izstrādātāju pamata komanda ir neliela, bet esam pieraduši C un C++ valodās kodus rakstīt ļoti ātri. Tā kā funkcijas, operatori un citas lietas nav realizētas mSQL datubāzē, šai sistēmai būs grūti panākt MySQL.
Par instrumentālām programmām. Gan mSQL, gan MySQL datubāzēm bija izveidoti vairāki instrumentāli rīki. Tā kā programmu pārnese no mSQL uz MySQL nesastāda grūtības, gandrīz visas interesantas vietnes, kuras bija izstrādātas sākumā mSQL datubāzei, ir pieejamas arī MySQL datubāzē.

Tika konstatētas sekojošas priekšrocības un trūkumi salīdzinot MySQL un mSQL [26]:
– MySQL ir drošāka par mSQL, kas padara MySQL arī vairāk lietojamu un uzticamu;
– MySQL nodrošina labāku veiktspēju nekā mSQL;
– MySQL ir ātrāks par mSQL, kas palielina MySQL produktivitāti;
– gan MySQL, gan mSQL neatbalsta SQL valodu pilnībā, un šis trūkums var ierobežot lietotājus, ja viņi vēlas izstrādāt sarežģītus projektus.

MySQL ir labāka nekā mSQL, jo tā ir ātrāka, drošāka un vairāk uzticama, kā arī tā tiek pilnveidota mūsdienās.

2. MySQL datubāzei ir dažas sekojošas priekšrocības un trūkumi, salīdzinot ar PostgreSQL [17]:
1) MySQL pārspēj PostgreSQL darba ātrumā;
2) MySQL labāk strādā Windows vidē;
3) MySQL satur lielu API skaitu citām valodām un tiek atbalstīts ar lielu programmēšanas valodu skaitu;
4) MySQL datubāzē paredzēta tabulu izveidošanas iespēja bez transakcijas;
5) Transakciju atbalsts MySQL datubāzē nav tik labi pārbaudīts, kā PostgreSQL;
6) PostgreSQL darbu var paātrināt, izpildot kodu glabātu procedūru vidē;
7) Iespraušanas, dzēšanas un atjaunošanas komandu darbs PostgreSQL datubāzē izpildās lēnāk nekā MySQL.

Kāpēc labāk izmantot MySQL salīdzinot ar PostgreSQL? Teorētiski ir rakstīts, ja izstrādātājs vēlas izveidot tīmekļa vietni un veiktspēja ir svarīga, tad MySQL būs labāka izvēle, jo MySQL ir ātrāka un ir veidota, lai labi strādātu ar tīmekļa bāzētu serveri. Ja rodas vēlme izveidot vēl vienu lietojumu, kurā tiks izmantotas transakcijas un ārējās atslēgas, tad labāk izmantot PostgreSQL. MySQL ir relatīvi ātrāka par PostgreSQL un datubāzes veidošana ir vienkāršāka. MySQL replicēšana, atšķirībā no PostgreSQL, bija pilnīgi notestēta.
Labāk izmantot PostgreSQL nevis MySQL, ja izstrādātājs vēlas izveidot sarežģītu datubāzi. PostgreSQL nodrošina procedūras valodas izmantošanu uz servera, transakcijas izmantošanu, atmiņas procedūras izmantošanu, ģeogrāfisko datu izmantošanu, kā arī R-Trees izmantošanu.

Kopumā [26] ir grūti pateikt, kura no šīm divām datubāzēm ir labāka, jo viss ir atkarīgs no tā, ko lietotājs grib panākt. Gan MySQL, gan PostgreSQL nodrošina labu veiktspēju un produktivitāti, abas datubāzes ir lietojamas, saprotamas, uzticamas, drošas un abas var mijiedarboties ar PHP. Taču mana tēma ir saistīta ar MySQL, un turpmāk praksē pielietosim tiešu viņu.

1.6. Noslēgums

Pirmajā nodaļā bija pastāstīts kopumā par PHP programmēšanas valodu un MySQL datubāzi, par to vēsturi, pielietošanu, darbību un versijām, kā arī pastāstīts par PHP sintaksi, operatoriem, konstantēm, operācijām, cikliem, mainīgo tipiem un komentāriem. Bija arī īsuma pastāstīts par PHP salīdzinājumu ar citām trim programmēšanas valodām un MySQL salīdzinājumu ar divām citām datubāzēm.
Ar šo es beidzu stāstīt par darba teorētisko daļu un tiecos klāt pie praktiskās daļas.

2. PRAKTISKĀ DAĻA

Šajā nodaļā tiek pētītas iespējas un piedāvāti daži piemēri, kas palīdzēs rezultātā noskaidrot, kāpēc PHP un MySQL ir ērtas projektu realizācijai. Stāstot par iespējām, tika aprakstītas to priekšrocības un trūkumi. Kritēriji tika paņemti no 1.4. apakšnodaļas.

Tika apskatītas sekojošas iespējas:

1) Datu glabāšana – kāda iespēja ir vairāk uzticama, kā arī veiktspējas novērtēšana, pie kādiem datiem varam nokļūt ātrāk, vai varam saņemt datus pēc kāda kritērija, vai vairākiem lietotājiem ir iespēja vienlaicīgi sūtīt savus datus un kādu datu apjomu var glabāt;

2) Datu šifrēšana – kāda no iespējām nodrošina labāku drošību;

3) Autentifikācija – kāda no iespējām ir droša, uzticama, uzturama, saprotama, lietojama, verificējama, pareiza, kā arī veiktspējas novērtēšana;

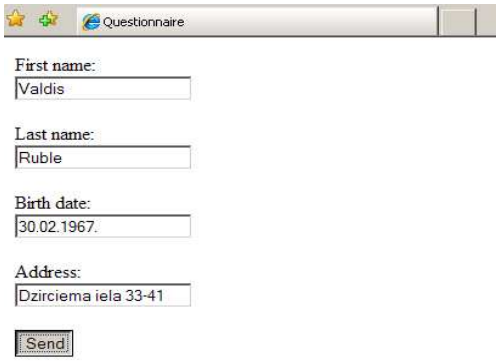
4) Informācijas apmaiņa starp klientu un serveri – kā šo iespēju padarīt par efektīvu, saprotamu, lietojamu un uzticamu.

2.1. Datu glabāšana

Tīmekļa vietnes izstrādāšanas laikā rodas jautājums par informācijas saglabāšanas iespēju, lai turpmāk varētu to izmantot. Eksistē divas pamat iespējas, kur var glabāt datus – datnēs un datubāzēs. Dotā pētījuma mērķis ir noskaidrot, kādu no iespējām labāk izmantot – pie kādiem datiem varam nokļūt ātrāk, vai varam saņemt datus pēc kāda kritērija, vai vairākiem lietotājiem ir iespēja vienlaicīgi sūtīt savus datus un kādu datu apjomu var glabāt, kāda iespēja ir vairāk uzticama un novērtēt veiktspēju. Sāksim ar datnēm.

2.1.1. Datne

Datne ir datu kopa, ko glabāšanas, pārsūtīšanas vai apstrādes procesā uzskata un identificē kā vienotu veselumu. Datnei ir daudz formātu veidu, bet manā pētīšanas laikā tika izmantots vienkāršais tekstu fails. Pieņemsim, ka lietotājam ir nepieciešams aizpildīt un nosūtīt anketu, kuras forma ir piedāvāta nākošajā lappusē:



2.1.1.1. att. **Anketas forma**

Dotajā piemērā, kad klienta vārda, uzvārda, dzimšanas datuma un adreses lauki tiek aizpildīti un ir uzspiesta poga „Send", dati tiek ierakstīti tekstu failā vienā rindā. Visas aizpildītās anketas ierakstās vienā failā. Lai datus varētu ierakstīt failā vajag izpildīt trīs darbības – atvērt failu (ja tas vēl neeksistē, tad tas ir jāizveido), ierakstīt datus failā, aizvērt failu. Tādas pašas darbības ir vajadzīgas, lai datus varētu izvilkt no faila (atvērt failu, atlasīt datus no faila, aizvērt failu). PHP valodā funkcija „fopen()" kalpo failu atvēršanai. Failu var atvērt tikai lasīšanai, tikai ierakstīšanai, vai lasīšanai un ierakstīšanai kopā. Izsaucot funkciju „fopen()" mēs padodam tai divus parametrus – faila atrašanās vietu, kuru jāatver, un faila režīmu, kas norāda, ko mēs darīsim ar failu. Tipiska kļūda, kas var rasties faila atvēršanas laikā, ir atļaujas neesamība faila lasīšanai vai ierakstīšanai tajā. Piemērām, ja bija saņemts paziņojums – „*Warning: fopen(php/questionnaire.txt) [function.fopen]: failed to open stream: Permission denied in php/anketa.php on line 25"*, tas nozīme, ka vajag pārliecināties vai mums ir tiesības izmantot failu, kuram gribam pieslēgties. Funkcija „fwrite()" kalpo datu ierakstīšanai failā. Ierakstīšanas laikā padodam funkcijai divus parametrus – failu, kuru atveram un datus, kurus ierakstīsim failā. Pēc šīs operācijas izpildīšanas aizveram failu ar „fclose()" palīdzību.

Anketas datus, kuri bija redzami attēlā 2.1.1.1. mums vajag saglabāt teksta failā, un to var izdarīt sekojoši:

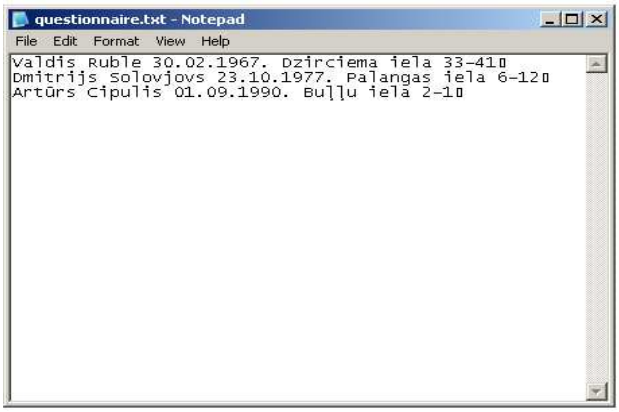
$file = "questionnaire.txt"; -> fails, kuru izmantosim

$fp = fopen($file, "a+"); -> atveram failu un padodam parametru "a+", kas atver failu ierakstu papildināšanai un lasīšanai, sakot no faila ieraksta beigām, ja tāds bija.

*fwrite($fp, $data);* -> ierakstam datus, kas atrodas mainīgā „$data", failā.

fclose($fp); -> aizveram failu.

Pieņemsim, ka jau bija nosūtītas trīs anketas. Attēlā 2.1.1.2. ir piedāvāts teksta faila ieraksts:



2.1.1.2. att. **„Questionnaire" faila dati**

Eksistē vēl daudz citas funkcijas, kuras pielieto strādājot ar failiem, bet tās pētījumā nav nepieciešamas. Pārejam pie otrās datu glabāšanas iespējas.

2.1.2. Datubāze

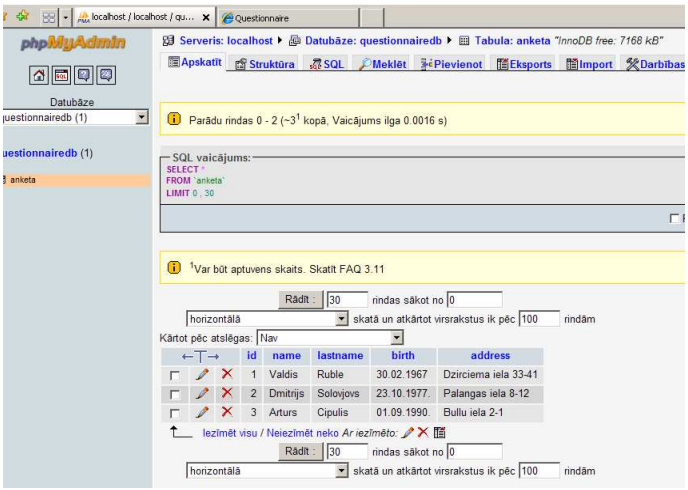
Datubāze ir datu kopums, kas ar speciālas pārvaldības sistēmas starpniecību organizēts tā, lai nodrošinātu ērtu informācijas izvilkšanu. Tagad mēs atkal strādāsim ar anketu, tikai šoreiz dati būs ierakstīti MySQL datubāzē. Izveidosim phpMyAdmin saskarnē datubāzi „questionnairedb", kurā izveidosim tabulu „anketa" ar piecām ailēm – id, name, lastname, birth un address. „Id" aile kalpos par katra ieraksta unikālo identifikatoru un četras pārējās ailes saturēs lietotāju aizpildītu anketu datus. Lai būtu iespēja saglabāt datus izveidotā datubāzē, ir nepieciešams pieslēgties tai, pielietojot PHP kodu. Par to bija stāstīts apakšnodaļā 1.3.3. Pieslēdzamies datubāzei:

```
<?php
$db = mysql_connect("localhost","user","12345");
mysql_select_db("questionnairedb ",$db);
?>
```

Pārbaudot visus aizpildītos laukus, ierakstam datus tabulā, pielietojot kodu:

```
<? php
$result = mysql_query ("INSERT INTO anketa (name, lastname, birth, address)
VALUES ('$name', '$lastname', '$date', '$address)");
?>
```

Rezultātā, pieņemot, ka bija sūtītas trīs anketas, varam apskatīt datus datubāzē:



2.1.2.1. att. **Anketas dati datubāzē**

Apskatot abus variantus, es ķeros klāt pie pētījuma vērtējuma, kuru aprakstīšu nākamajā punktā.

2.1.3. Vērtējums

Pētīšanas gaitā tika konstatēti sekojoši trūkumi strādājot ar datnēm:

– kad datnes apjoms kļūs lielāks, darbs ar to kļūs lēnāks;

– datnē ir grūti atrast konkrētu ierakstu vai ierakstu grupu. Lai sameklētu nepieciešamo informāciju, vajag izlasīt un pārbaudīt katru ierakstu atsevišķi;

– daži lietotāji nevar nosūtīt savus anketas datus vienlaicīgi, jo datne tiek bloķēta, kad viens lietotājs izmanto to anketas aizpildīšanas laikā, lai ieraksti nesajauktos. Nav zināms, cik drīz konkrētam lietotājam būs iespēja aizpildīt anketu;

– var rasties nepieciešamība pievienot vai dzēst ierakstus no datnes vidus, ko ir grūti izdarīt, jo vajadzēs izlasīt visu datni, veikt izmaiņas un tad atkārtoti ierakstīt visu datni.

Strādājot ar MySQL datubāzi tika konstatētas sekojošas priekšrocības:

– MySQL nodrošina ātrāku piekļūšanu datiem nekā datnes;

– MySQL ļauj viegli atrast konkrētu ierakstu. Var realizēt pieprasījumu datu izvilkšanai, norādot kritērijus;

– daži lietotāji var nosūtīt savus anketas datus vienlaicīgi, jo MySQL satur iebūvētos mehānismus, kuri apstrādā paralēlas pieteikšanās datubāzei;

– MySQL nodrošina patvaļīgu piekļūšanu datiem;

– MySQL var glabāt lielu datu apjomu.

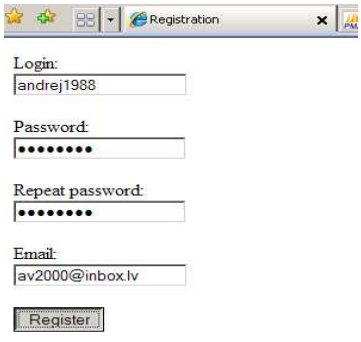
MySQL tika galā ar problēmām, kuras bija konstatētas, strādājot ar datnēm. Pēc agrāk aprakstītajiem kritērijiem, var izdarīt secinājumu, ka labāk izmantot MySQL datubāzi datu glabāšanai. Datubāzes pielietošana piedāvā labāku veiktspēju, nekā datnes. Izmantojot datubāzi, sistēma būs vairāk uzticama.

2.2. Datu šifrēšana

Veidojot tīmekļa vietnes var rasties jautājums par vērtīgu datu aizsargāšanu [15], lai tos nevarētu izlasīt citi lietotāji, nezinot atslēgu vai paroli. To var paveikt pielietojot šifrēšanu, pārveidojot datus tāda formātā, lai tos varētu izlasīt paredzamais izziņas saņēmējs. Datu šifrēšanu pielieto vairākās vietnēs, īpaši tajās, kurās notiek elektroniskie pasūtījumi. Atkarība no vēlamā drošības līmeņa var šifrēt arī lietotāju vārdus, telefona numurus, adreses un citus datus. PHP valoda atbalsta dažas šifrēšanas funkcijas, bet mēs izpētīsim trīs no tām – „md5()", „crypt()" un „sha1()" funkcijas. Dotā pētījuma mērķis ir noskaidrot kādu no iespējām labāk izmantot datu šifrēšanai, tas ir kāda no iespējām labāk aizsargā datus. Sāksim ar „md5()" funkciju.

2.2.1. md5() funkcija

Funkcija izrēķina MD5 rindas jaukšanu, izmantojot MD5 RSA DATA Security, Inc algoritmu, un atgriež 32 zīmju heksadecimālo skaitli. Funkciju biežāk pielieto [19] paroles glabāšanai kādas vietnes datubāzē. Pieņemsim, ka mēs gribam piereģistrēties sociālā tīklā, izveidojot lietotājvārdu, paroli un ievadot savu ē-pastu, uz kuru saņemsim reģistrācijas apstiprinājumu. Aizpildīta reģistrācijas forma ir piedāvāta zemāk:

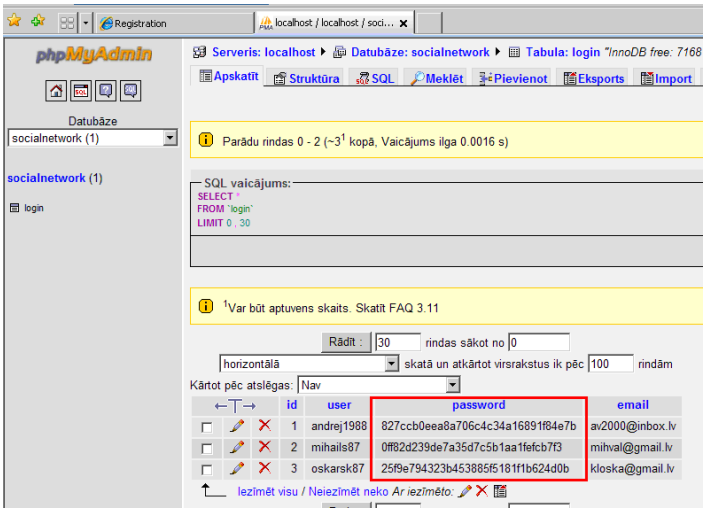


2.2.1.1. att. **Reģistrācijas forma**

Kad mēs aizpildām reģistrācijas formu un uzspiežam uz ”Register” pogu, sistēma pārbauda vispirms, vai bija aizpildīti visi lauki, vai ievadītais lietotājs jau nebija agrāk reģistrēts, vai sakrīt abas paroles un vai e-pasta adrese bija ievadīta pareizi. Ja bija konstatēta veiksmīga pārbaude, tad pārējam pie paroles šifrēšanas, pielietojot „md5()” funkciju. Rindas fragments ir:

$_POST['password'] = md5($_POST['password']);

Paroli, kuru ievadījām, funkcija pārveido 32 zīmju heksadecimālā skaitlī un tad rezultāts tiek saglabāts datubāzē. Pieņemsim, ka bija reģistrēti trīs lietotāji, un tagad MySQL datubāzē varam apskatīt viņu datus, ieskaitot šifrētas paroles. Zemāk attēlā 2.2.1.2. varam apskatīt datubāzes datus, kur paroles bija šifrētas ar „md5()” funkcijas palīdzību:



2.2.1.2. att. **Reģistrētu lietotāju dati datubāzē**

Pieteikšanās laikā ar paroli notiek tāda pati operācija. Ievadot lietotājvārdu un paroli, sistēma pārbauda, vai bija aizpildīti visi lauki, vai lietotājs ir reģistrēts un vai parole ir pareiza. Paroli pārlaiž caur „md5” un salīdzina ar paroli no datubāzes – ja rindas sakrīt, tad ievadītā parole ir pareiza un lietotājs tiek pieteikts sistēmā. Pārējam pie otrās datu šifrēšanas iespējas.

2.2.2. crypt() funkcija

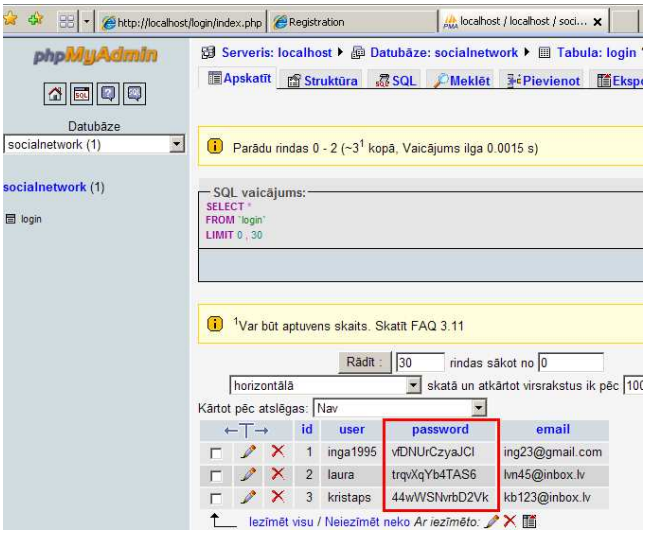
Funkcija [20] atgriež šifrētu pēc standarta UNIX algoritma rindu. Pielietojot to datu šifrēšanai, mēs padodam tai divus parametrus – pirmais parametrs ir rinda, kuru šifrēsim un otrais (neobligāts) parametrs ir „salt” secība, kura atbild par datu šifrēšanu. Ja nebija ievadīts otrais parametrs, tas tiek izvēlēts nejauši. Šo funkciju mēs

arī pielietosim paroles glabāšanai datubāzē. Atkārtoti aizpildām reģistrēšanas formu, kas bija pieejama attēlā 2.2.1.1. un uzspiežam uz „Register" pogu. Ja bija konstatēta veiksmīga pārbaude, pārējam pie paroles šifrēšanas, pielietojot „crypt()" funkciju. Rindu fragments ir:

$salt = substr($_POST['password'], 0, 2);

$_POST['password'] = crypt($_POST['password'], $salt);

Redzam otro ievadīto parametru, kas nozīme, ka tiek izmantots standarts DES-algoritms. Paroli, kuru ievadījām, funkcija šifrē ar 2-simbolu „salt", izmantojot paroles pirmos divus simbolus kā determinantu, un tad rezultāts tiek saglabāts datubāzē. Pieņemsim, ka bija reģistrēti trīs lietotāji, un tagad MySQL datubāzē varam apskatīt viņu datus, ieskaitot šifrētas paroles. Attēlā 2.2.2.1. varam apskatīt datubāzes datus, kurās paroles bija šifrētas ar „crypt()" funkcijas palīdzību:



2.2.2.1. att. **Reģistrētu lietotāju dati datubāzē**

Pirmie divi simboli šifrētā parolē ir pirmie divi simboli no lietotāja ievadītās paroles. Pieteikšanās laikā ar paroli notiek tāda pati operācija. Paroli pārlaiž caur „crypt" un salīdzina ar paroli no datubāzes – ja rindas sakrīt, tad ievadītā parole bija pareiza un lietotājs tiek pieteikts sistēmā. Pārējam pie trešās datu šifrēšanas iespējas.

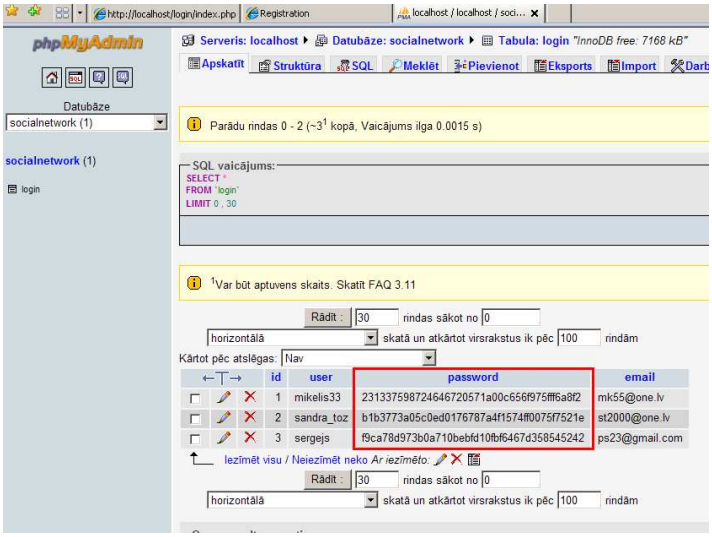
2.2.3. sha1() funkcija

Funkcija izrēķina rindas jaukšanu, izmantojot US Secure Hash Algorithm 1 algoritmu. Saņemot ieejā rindu funkcija atgriež 40 zīmju heksadecimālo skaitli. Šo funkciju, tāpat, ka „md5()" un „crypt()" funkcijas, mēs pielietosim paroles glabāšanai

datubāzē. Atkārtoti aizpildām reģistrēšanas formu, kas bija pieejama attēlā 2.2.1.1. un uzspiežam uz „Register” pogu. Ja bija konstatēta veiksmīga pārbaude, pārējam pie paroles šifrēšanas, pielietojot „sha1()” funkciju. Rindu fragments ir:

$_POST['password'] = sha1($_POST['password']);

Paroli, kuru ievadījām, funkcija pārveido 40 zīmju heksadecimāla skaitlī un tad rezultāts tiek saglabāts datubāzē. Pieņemsim, ka bija reģistrēti trīs lietotāji, un tagad MySQL datubāzē varam apskatīt viņu datus, ieskaitot šifrētās paroles. Zemāk attēlā 2.2.3.1. varam apskatīt datubāzes datus, kurās paroles bija šifrētas ar „sha1()” funkcijas palīdzību:



2.2.3.1. att. **Reģistrētu lietotāju dati datubāzē**

Pieteikšanās laikā ar paroli notiek tāda pati operācija. Ievadot lietotājvārdu un paroli, sistēma pārbauda, vai bija aizpildīti visi lauki, vai lietotājs ir reģistrēts un vai parole ir pareiza. Paroli pārlaiž caur „sha1” un salīdzina ar paroli no datubāzes – ja rindas sakrīt, tad ievadīta parole bija pareiza un lietotājs tiek pieteikts sistēmā. Apskatot trīs variantus, es ķeros klāt pie pētījuma vērtējuma, kuru aprakstīšu nākamajā punktā.

2.2.4. Vērtējums

Pētīšanas gaitā tika konstatēti sekojoši trūkumi un priekšrocības, strādājot ar trijām funkcijām:

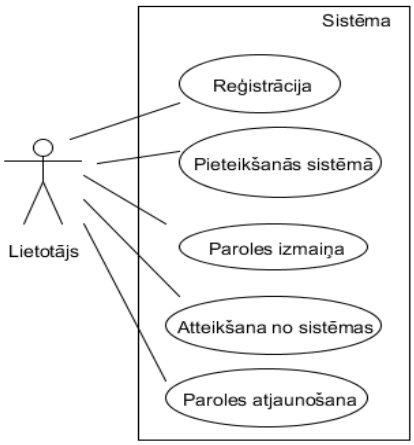
– „md5()” un „sha1()” funkcijām jāpadod rinda, kuru gribam šifrēt, tai pašā laikā „crypt()” funkcijai jāpadod divi parametri (rindu un „salt” secību), kas nozīme, ka izstrādātājam jāizdara lieku operāciju;

– „sha1()” funkcijas rezultāts ir strikti determinēts. Vienai un tai pašai rindai funkcija atgriezīs vienu un to pašu rezultātu katrā tās izsaukšanas reizē;
– pirms lietot „sha1()” funkciju nav nepieciešams zināt, kā izskatījās paroles rinda. Pietiek zināt, vai ievadītā parole sakrīt ar paroli, kurai tiek pielietota funkcija;
– praksē „sha1()” un „md5()” funkcijas pielieto biežāk, nekā „crypt()”;
– „crypt()” funkcija ir mazāk aizsargāta nekā „md5()” un „sha1()” funkcijas, jo „crypt()” funkciju ir iespējams atšifrēt (kas nav labi no drošības puses), pārējās divas nevar.

Labākā no izpētītām funkcijām ir „sha1()”, kura nodrošina drošāku informācijas aizsardzību, nekā abas pārējās funkcijas.

2.3. Lietotāju autentifikācija

Eksistē ļoti daudz tīmekļu vietņu, kuras lieto pieteiktie un nepieteiktie lietotāji. Šīs vietnes var būt e-pasti, interneta veikali, ziņu vietnes, sociālais tīkls un citas. Atšķirība ir tāda, ka pieteiktiem lietotājiem ir vairāk privilēģijas, nekā nepieteiktiem – galvenā no tām ir konfidenciālās informācijas piekļuve. Lai varētu nodrošināt informācijas konfidencialitāti, vajag ierobežot piekļuvi tām. Lai būtu drošam, ka tieši tā persona ieguva piekļuvi informācijai, pielieto autentifikācijas procedūru. Lietotājam vajag uzrādīt sistēmai unikālo informācijas veidu, tā saucamo autentifikācijas faktoru, piemēram, lietotājvārdu un paroli, atslēgu, pirkstu nospiedumus un citu.
Lietotāju reģistrācija, ieeja un izeja no sistēmas, paroles izmaiņa un paroles atjaunošana ir četri galvenie elementi, kuri ir iekļauti autentifikācijas modulī. Nākošajā lappusē ir piedāvāta lietojumu diagramma, kurā ir redzamas lietotāja darbības, saistītas ar autentifikāciju:



2.3.1. att. **Lietojumu diagramma autentifikācijas modulim**

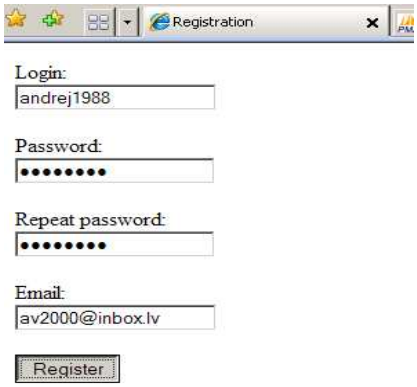
Šajā punktā tiek izpētītas lietotāja autentifikācijas iespējas – kādas funkcijas pastāv, lai varētu realizēt šīs iespējas ar PHP un MySQL, un vai vispār tās var realizēt, kā arī noskaidrot, kura no iespējam ir vairāk droša, uzticama, uzturama, saprotama, lietojama, verificējama, pareiza, kā arī novērtēt veiktspēju. Tiek apskatītas autentifikācija ar lietotājvārdu un paroli, biometrijas autentifikācija un autentifikācija ar USB-atslēgu.

2.3.1. Lietotājvārds un parole

Pirmo veidu, kuru pētīsim, ir lietotāja autentifikācija caur lietotājvārdu un paroli. Šīs autentifikācijas veids ir visizplatītākais mūsdienās. Nepieciešami sekojoši komponenti, lai lietotājam būtu iespēja pieteikties sistēmā, ievadot lietotājvārdu un paroli:

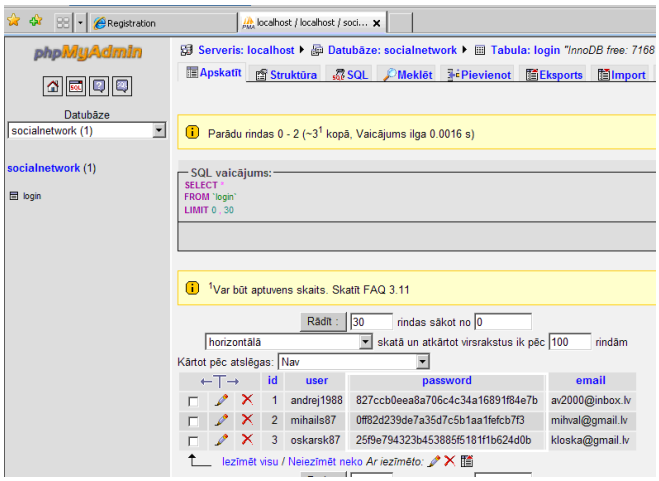
1) Lietotājiem jābūt iespējai piereģistrēt savu izvelēto lietotājvārdu un paroli. Dati tiek glabāti datubāzē. Parole tiek šifrēta;
2) Lietotājiem jāļauj pieteikties sistēmā, ievadot savus reģistrētus datus;
3) Lietotājiem jābūt iespējai atteikties no sistēmas;
4) Lietotājiem jābūt iespējai izmainīt savu paroli;
5) Lietotājiem jābūt iespējai atjaunot savu paroli, ja viņi to ir aizmirsuši.

Apskatīsim katru komponentu, sākot no reģistrācijas. Pieņemsim, ka lietotājam jāpiereģistrējas sociālā tīkla vietnē, ievadot lietotājvārdu, paroli un savu e-pastu. Nākošajā lappusē ir piedāvāta aizpildīta reģistrācijas forma no apakšnodaļas 2.2.1:



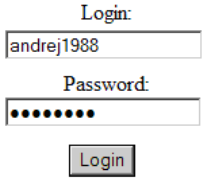
2.3.1.1. att. **Reģistrācijas forma**

Pēc „Register" pogas uzspiešanas dati tiek saglabāti MySQL datubāzē. Paroles ir šifrētas ar „md5()" funkciju. Izmantoju agrāk aprakstītu datubāzi „socialnetwork" un tabulu „login" no apakšnodaļas 2.2.1. Tagad, kad dati ir saglabāti datubāzē (kā redzams attēlā 2.3.1.2.), ķeramies klāt pie pieteikšanas sistēmā punkta.



2.3.1.2. att. **Reģistrētu lietotāju dati datubāzē**

Piereģistrētie lietotāji tagad var pieteikties sistēmā. Attēlā 2.3.1.3. lietotājs ievada savu lietotājvārdu ar paroli un uzspiež pogu „Login":

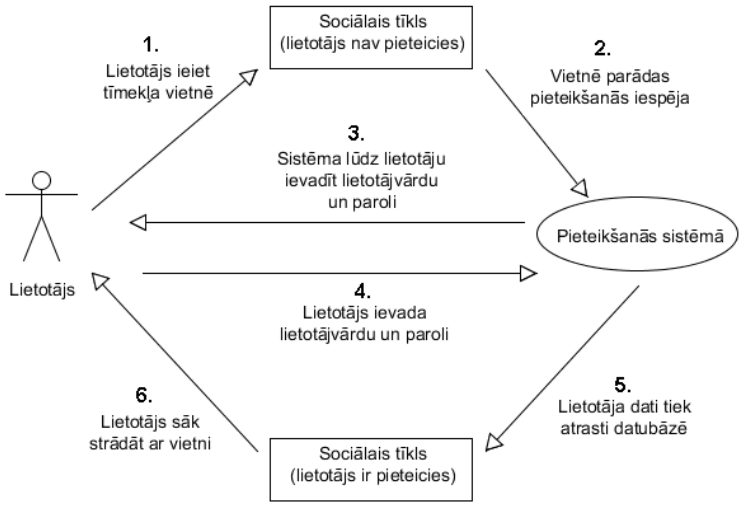


2.3.1.3. att. **Pieteikšanās sistēmā forma**

Tālāk notiek datu pārbaude MySQL datubāzē – vai eksistē lietotājs ar ievadīto lietotājvārdu un vai pareizi bija ievadīta parole. Ja lietotājs ir ierakstīts datubāzē un ievadīta parole sakrīt ar paroli no datubāzes, lietotāja identifikators tiek reģistrēts ar seansa ($_SESSION) palīdzību. Zemāk ir pieejams PHP koda fragments, kurš atrod saņemtus datus datubāzē, un ja tādi bija atrasti, tad novieto lietotājvārdu sesijā:

```
$sql = "SELECT * FROM login WHERE user='$myusername' and password='$mypassword' ";
$result=mysql_query($sql);
$count=mysql_num_rows($result);
if($count==1){ $_SESSION['user'] = $myusername; }
```

Tagad, kad lietotājs veiksmīgi pieteicies sistēmā, viņam ir parādījusies piekļuve konfidenciālai informācijai. Neveiksmīgas pieteikšanās gadījumā lietotājs saņems paziņojumu par kļūdu – nav aizpildīti visi lauki, lietotājs neeksistē vai nepareiza parole. Zemāk ir shematiski parādīts, kā darbojas pieteikšanās sistēmā:



2.3.1.4. att. **Pieteikšanās sistēmā shēma**

Kad gribam atteikties no sistēmas, mēs uzspiežam uz pogu „Logout", sesija tiek iznīcināta un piekļuve slepenai informācijai tiek aizslēgta.

Kā agrāk bija rakstīts, lietotājam jābūt iespējai izmainīt savu paroli. Vispirms mēs ievadam veco paroli, kura tiek pārbaudīta datubāzē. Ja paroles sakrīt, sistēma piedāvās mums ievadīt jauno paroli. Rezultātā sistēma saglabā jauno paroli datubāzē virs vecās. Var gadīties, ka lietotājs aizmirsis savu paroli. Izveidosim scenāriju, kurš izveidos nejaušo vērtību parolei un sūtīs lietotājam jauno versiju uz e-pastu. Scenārijs sastāvēs

no divām funkcijām. Pirmā ģenerēs nejaušu paroli un ievietos to datubāzē. Koda fragmentā tiek paņemts vārds no vārdnīcas un tam tiek pievienots skaitlis:

$new_pass = get_word (5, 10);

$add_number = rand (0, 99);

$new_pass .= $add_number;

$result = mysql_query ("UPDATE login set password = md5('".$new_pass."') where user = '".$username."'");

Funkcija „get_word()" saņem nejaušu vārdu no vārdnīcas, kuru pielieto paroles ģenerācijai. To mēs precīzāk neapskatīsim, jo galvenais ir saprast, kā paroles atjaunošana strādā vispār. Tagad, kad pirmā funkcija uzģenerēja un saglabāja jauno paroli datubāzē, šī parole tiek padota otrai funkcijai, kura atrod e-pastu datubāzē pēc lietotājvārda un jaunas paroles, un ar „mail()" PHP funkcijas palīdzības sūta paroli lietotājam pa viņa e-pastu. Koda fragmentā parādīts, kā e-pasts tiek paņemts no datubāzes un kā uz tā tiek sūtīts ziņojums:

$result = mysql_query ("SELECT email from login where user = '".$username."'");

$row = $result->fetch_object();

$email = $row->email;

$message = "Your new password is ".$password."";

mail($email, 'Information', $message);

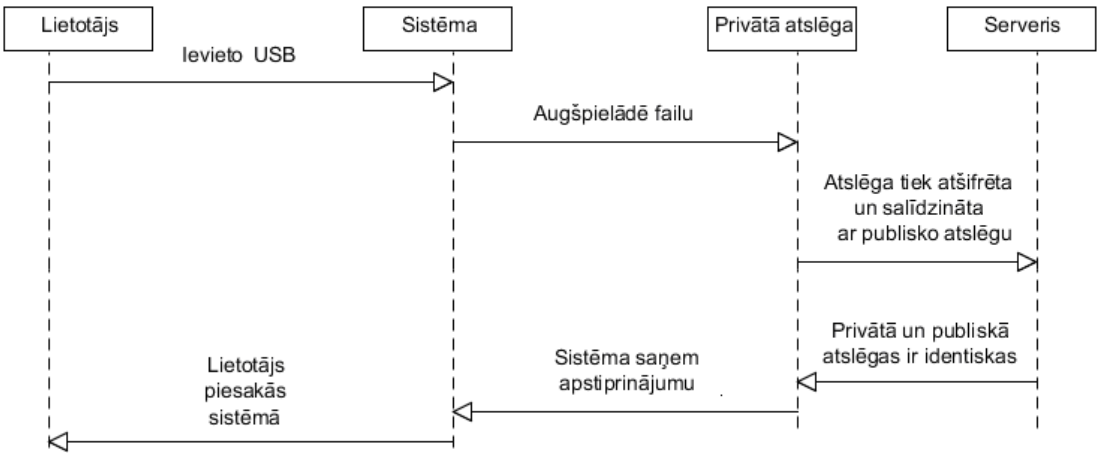
Saņemot jauno paroli lietotājam uzreiz būs iespēja to nomainīt, kad viņš būs pieteicies sistēmā.

Kopumā varu pateikt, ka pirmā izpētīta autentifikācijas iespēja ir skaidri un bez grūtībām realizējama. Apskatīsim nākošo iespēju.

2.3.2. USB-atslēga

Otrais veids [22] ir lietotāja autentifikācija pielietojot USB-atslēgu. Šo veidu ir grūti apiet, jo lietotājs izmanto unikālo fizisko objektu, lai pieteiktos sistēmā. Tāpat, ka ar lietotājvārdu un paroles autentifikāciju, lietotājam jābūt iespējai piereģistrēt savus datus. Tiek ģenerēts atslēgu pāris – privāts un publisks. Publisko atslēgu novieto serverī, bet privāto novieto USB-atmiņā. Privātā atslēga tiek pakļauta šifrēšanai un saglabājas identificētā failā. Kad lietotājs grib pieteikties sistēmā, viņš ievieto USB sistēmas ieejā, sistēma augšupielādē failu no USB atmiņas un salīdzina

datus serverī. Attēlā 2.3.2.1. ir shematiski parādīts, kā notiek autentifikācija caur USB-atslēgu:



2.3.2.1. att. **Pieteikšanās sistēmā caur atslēgu**

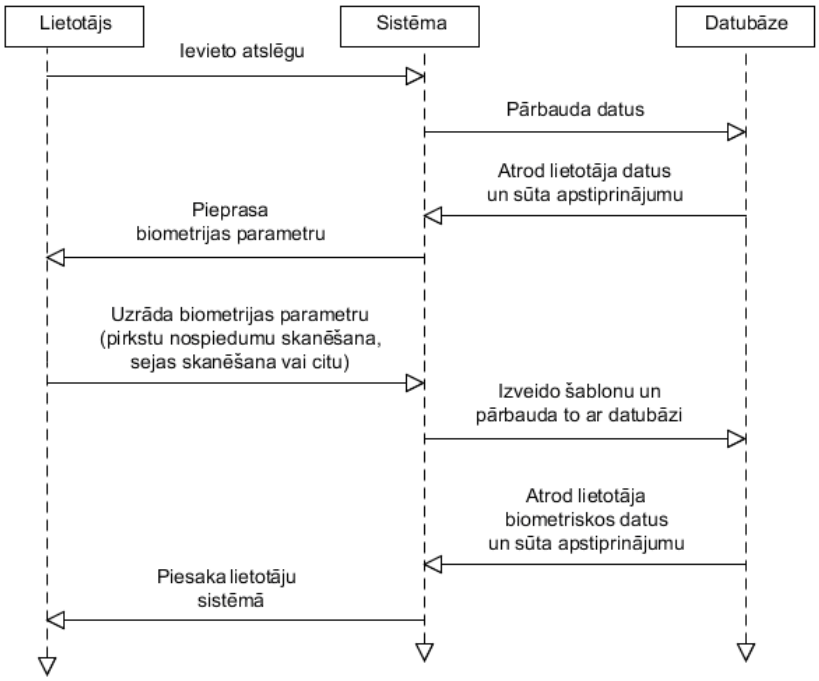
Pateicoties šai iespējai, lietotājam nevajadzēs atcerēties savu paroli. Kad lietotājs grib atteikties no sistēmas, viņš izņem USB-atslēgu no ieejas. PHP valodā tiek izmantota „ssh2_auth_pubkey_file()" funkcija, kura nolasa privāto atslēgu no faila un salīdzina to ar publisko atslēgu, kas atrodas serverī. Tātad doto autentifikācijas metodi ir iespējams realizēt ar PHP. Apskatīsim nākošo iespēju.

2.3.3. Biometrija

Trešais veids ir biometrijas autentifikācija [21], kas ļauj identificēt lietotāju pēc viņa fiziskām pazīmēm – sejas, pirkstu nospiedumiem, balss un citām. Ikviena biometrijas tehnoloģija tiek pielietota pa etapiem:

1) objekta skanēšana;
2) individuālās informācijas izdabūšana;
3) šablona formēšana;
4) šablona salīdzinājums ar datubāzi.

Kad lietotājs vēlas pieteikties sistēmā, viņš vispirms identificē sevi ar karti vai personas numuru. Sistēma saņem šos datus, atrod datubāzē lietotāja failu, kurā glabājas viņa biometrijas dati. Tad lietotājs uzrāda kādu no saviem biometrijas parametriem (atkarīgi no sistēmas prasībām), sistēma izveido digitālo attēla reprezentāciju, izveido šablonu un pārbauda attēlu datubāze. Pēc veiksmīgas datu pārbaudes lietotāju pieteic sistēmā. Shematiski tas var izskatīties sekojoši:



2.3.3.1. att. **Pieteikšanās sistēmā caur biometrijas parametriem**

Lietotājs var piereģistrēt savus datus tikai firmas (kurai pieder kāda sistēma) klātienē, lai firma ievietotu datus datubāzē un iesniegtu karti, ar kuru lietotājs uzsāks pieteikšanu. Dotajā piemērā lietotājiem nevajag raizēties par paroles atcerēšanos, jo tādas nebūs, līdz ar to varam izslēgt paroles izmainīšanas un atjaunošanas funkcijas. Biometrijas autentifikācija ir labākā metode no drošības viedokļa, taču to nevar realizēt PHP valodā. Apskatot trīs variantus, es ķeros klāt pie pētījuma vērtējuma, kuru aprakstīšu nākamajā punktā.

2.3.4. Vērtējums

Pētīšanas gaitā tika konstatēti sekojoši trūkumi un priekšrocības autentifikācijas realizācijā:

– autentifikācija ar lietotājvārdu un paroli ir visizplatītākā mūsdienās un to izmanto vairākās tīmekļa vietnēs;

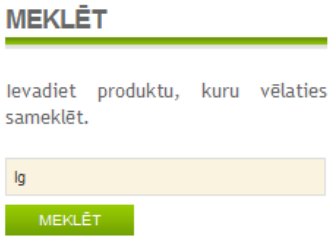
– autentifikāciju ar lietotājvārdu un paroli ir vieglāk uzlauzt nekā divas pārējās metodes, jo ļaundarim nav vajadzīgi cita lietotāja USB-atslēgu vai biometrijas dati. Pietiek tikai ar datu ievadīšanu;

– autentifikācija ar USB-atslēgu ir realizējama PHP valodā un dod labāku drošību nekā autentifikācija ar lietotājvārdu un paroli, jo ļaundarim nebūs iespējas veikt uzlaušanu bez faila no cita lietotāja USB-atslēgas;
– autentifikācija ar USB-atslēgu ir ērtāka lietošanā, jo lietotājam nevajag atcerēties paroli;
– autentifikācija ar USB-atslēgu un biometrijas autentifikācija ir vairāk uzticamas, jo tās ir grūtāk uzlauzt, nekā autentifikāciju ar lietotājvārdu un paroli;
– biometrijas autentifikācija ir labāka un efektīvāka no drošības viedokļa, taču to nevar realizēt PHP valodā, jo PHP nevar skanēt cilvēka fiziskās pazīmes.

Labākā no izpētītām iespējām, kuras var realizēt, pielietojot PHP un MySQL, ir autentifikācija ar USB-atslēgu, jo tā ir drošāka, uzturama, saprotama, lietojama un verificējama.

2.4. Informācijas apmaiņa starp klientu un serveri

Šajā nodaļā apskatīsim, kā notiek informācijas apmaiņa starp klientu un serveri [23], pamatojoties uz interneta veikalu pakalpojumiem. Pētīšanas laikā tika noskaidrots, kādi ir plusi un mīnusi, strādājot ar interneta veikalu, kas bija izstrādāts ar PHP un MySQL. Kā informācijas apmaiņu padarīt par efektīvu, saprotamu, lietojamu un uzticamu.
Pieņemsim, ka mums vajag atrast noteiktu produktu, ievadot tā nosaukumu meklēšanas pieprasījumā, kā ir redzams attēlā 2.4.1.:



2.4.1. att. **Meklēšanas pieprasījuma forma**

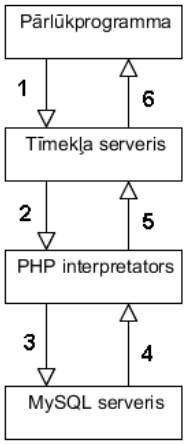
Lietotājs aizpilda meklēšanas lauku un uzspiež uz pogu „Meklēt”. Tad tīmekļa pārlūkprogramma sūta HTTP-vaicājumu noteiktai tīmekļa lappusei. PHP fails, kas atbild par meklēšanas funkciju, iekļauj sevī attiecīgo rindu, kura ir redzama koda fragmentā:

$result = mysql_query("SELECT * FROM data WHERE name like '%$search%' ORDER BY name",$db);

Tīmekļa serveris pieņem vaicājumu uz failu, izvelk to un padod uz apstrādāšanu PHP interpretatoram, kas uzsāk scenārija sintakses analīzi. Scenārijs satur pieslēgšanas MySQL datubāzei (*$db = mysql_connect("localhost","user","12345");*
*mysql_select_db("*inet_shop*",$db);*) un vaicājuma izpildes komandas (mūsu gadījuma – „LG" firmas ierīču meklēšana). Tad PHP pieslēdzas MySQL serverim un sūta tam attiecīgo vaicājumu. Augstāk koda fragmentā ir redzams, ka mēs meklējam informāciju par visiem produktiem, kuri iekļauj sevī „lg" vārdu, un tie būs sakārtoti pēc nosaukuma. MySQL serveris pieņem vaicājumu datubāzei, apstrādā to (pārbauda ievadīto nosaukumu ar produkta nosaukumiem no datubāzes) un tad sūta rezultātus, tas ir visus „LG" produktus, atpakaļ PHP interpretatoram. Tālāk PHP interpretators beidz scenāriju izpildi, formatējot vaicājuma rezultātus HTML vidē un atgriež rezultātu tīmekļa serverim HTML formātā. Tīmekļa serveris pārsūta pārlūkprogrammai HTML lappusi, kurā lietotājs var apskatīt meklētu produktu sarakstu.

Dotais process ir atkarīgs no lietotā scenārija un datubāzes servera. Itin bieži tīmekļa servera programmatūra, PHP interpretators un datubāzes serveris funkcionē uz viena datora. Tai laikā biežāk datubāzes serveris strādā uz cita datora. Tas tiek darīts drošības iemesla vai efektīvas slodzes izkārtojuma dēļ.

Šim piemēram atbilst sekojoša bāzes arhitektūra, kura ir pieejama attēlā 2.4.2.:



2.4.2. att. **Datubāzes bāzes arhitektūra tīmekļa lietojumam**

Rezultāti mūsu pieprasījumam par „lg" produktiem var būt sekojoši:

1) Uz ekrāna parādās visi produkti, kuri iekļauj ievadīto nosaukumu;

2) Uz ekrāna neparādās neviens produkts (produktu datubāzē nav);

Tagad, kad produkts ir atrasts, un lietotājs vēlas to iegādāties, viņš iemet to pirkšanas „grozā", kurā būs pieejama informācija par produkta nosaukumu, cenu, skaitu un pilno summu. Par interneta veikalu drusku plašāk.

Pavisam, jebkuram interneta veikalam jānodrošina sekojošas iespējas:

1) Lietotājam jābūt iespējai apskatīt produktu kategorijas;

2) Lietotājam jābūt iespējai apskatīt jebkuru produktu un iemest to „grozā";

3) Lietotājam jāsaņem informācija par kopīgo pirkšanas summu, par piegādi un par maksājumu apstrādi;

4) Administratoram jābūt iespējai strādāt ar informāciju (pievienot, rediģēt, dzēst, apskatīt).

Eksistē divas metodes produkta, kuru izvēlas pasūtītājs, izsekošanai – viena atbild par izvēlētā produkta novietošanu datubāzē, otra – seansa mainīgās izmantošana. Seansa mainīgās izvēlēto produktu izsekošanai izmantošana lappušu pāriešanas procesā ir vienkāršāka realizācijā, jo tas neprasa pastāvīgu vaicājumu datubāzei. Šī metode ļauj izvairīties no datubāzes aizpildīšanas ar nevajadzīgiem datiem, kuri pienāk no apmeklētājiem, kuri vienkārši vēlas apskatīt produktus un pastāvīgi maina savas domas.

Pirmajai metodei es izveidoju tabulu „basket", kurā tiek glabāta informācija par produktiem, kurus piereģistrēts lietotājs pagaidām iemet „grozā". Procesā ir iespējams papildināt „grozu", iztukšot to, un katru reizi lietotājs pieslēgsies datubāzes tabulai, kurā informācija mainīsies. Attēlā 2.4.3. ir piedāvāta tabula, kuru lietotājs papildina ar produktiem, bet vēl nav pasūtījis tos:

←T→	id	name	price	username	fullprice	count
	1	BenQ X725	69.836	user	88.4	2
	2	ALcom TS 415	9.401	user	11.9	4
	3	Acer P5270 DLP	308.89	user	391	2
	4	Acer EX5230	134.3	user	170	1

2.4.3. att. **„Basket" tabulas aizpildīšana**

Ja mēs tomēr izvēlēsimies nopirkt šos produktus, tad pāriesim pie apmaksāšanas punkta, aizpildīsim to un kad tiek uzspiegta poga „Pasūtīt" – tabula „basket" tiek iztukšota (pasūtītāja dati un produkti būs ierakstīti jau citā tabulā).

Otrajā metodē varam izmantot seansa palīdzību, tas ir, izveidot seansa mainīgo vai mainīgo salikumu lietotāja izvelēto produktu glabāšanai. Tad, kad lietotājs uzsāks veikt apmaksu, informācija tiks ievietota datubāzē.

Apskatīsim scenāriju, kas atbild par produktu pievienošanu „grozam”:

”$_SESSION[‘basket’] = array();” ir masīvs, kurā glabāsies informācija par produktu numuru un produktu skaitu;

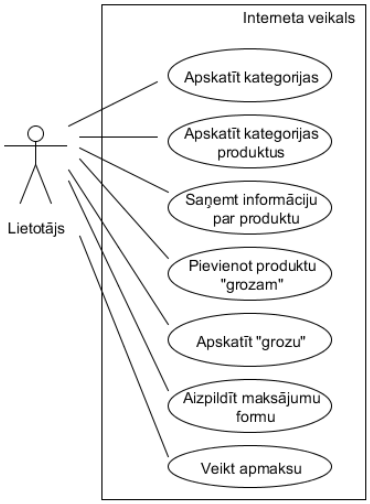
”$_SESSION[‘item’] = calc_item($_SESSION[‘basket’]);” funkcija summē visu produktu skaitu no „groza”;

”$_SESSION[‘total_price’] = calc_price($_SESSION[‘basket’]);” funkcija summē kopīgu produktu maksājumu, kas atrodas „grozā”.

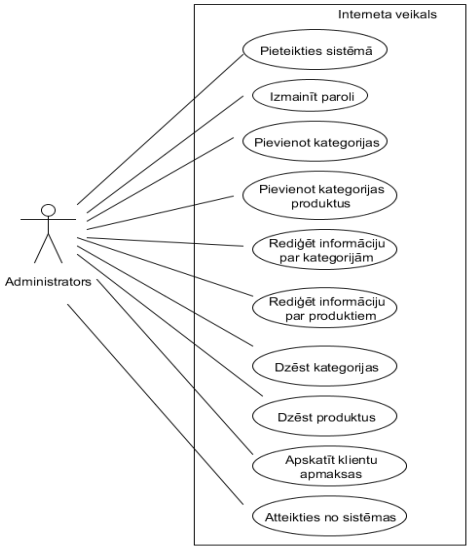
Kad visi nepieciešamie produkti būs pievienoti „grozam”, un klientam būs pieejama informācija par apmaksu, viņš aizpildīs maksājumu formu un izdarīs pasūtījumu. Liela problēma, kas var rasties pirkšanas laikā, ir informācijas atjaunošana uz galvenā servera. Eksistē ļoti maza varbūtība, ka administrators mainīs produkta cenas, kad klients veiks pasūtījumus. Nepamanot izmaiņas, lietotājs veiks maksājumu pēc jaunām cenām. Lai izvairītos no šīs problēmas administratoram labāk atjaunot informāciju naktī vai arī izslēgt serveri, veikt izmaiņas un tad ieslēgt serveri atpakaļ.

Veidojot interneta veikalu, jāveido arī tīmekļa saskarni administratoram, kurā būs nepieciešams vispirms pieteikties, lai varētu strādāt ar informācijas pievienošanas, rediģēšanas un dzēšanas opcijām.

Slēdzienā zemāk ir shematiski parādīti attēlos 2.4.4. un 2.4.5. lietotāja un administratora mijiedarbība ar interneta veikala serveri:



2.4.4. att. **Lietotāja-servera mijiedarbība**



2.4.5. att. **Administratora-servera mijiedarbība**

Pētīšanas gaitā tika konstatēti sekojoši trūkumi un priekšrocības, strādājot ar informācijas apmaiņu starp klientu un serveri [24]:

– pastāv iespēja, ka informācija var būt izmainīta pasūtīšanas laika, kas ietekmēs klienta apmaksas procesu, tas ir klients gribēs iegadāties produktu par vienu cenu, bet pēc pasūtīšanas no viņa kartes tiks noņemts vairāk naudas;

– seansa mainīgā izmantošana produkta pievienošanas „grozam" laikā ir labāka metode, nekā pastāvīga datubāzes izmantošanas metode, jo tā ļauj izvairīties no datubāzes aizpildīšanas ar nevajadzīgiem datiem, kuri nonāk no klientiem, kuri apskata katalogu un katru reizi maina savas domas;

– ērtāk, uzticamāk un efektīvāk izdarīt tā, lai pieteiktie lietotāji veiktu pirkumus, lai viņiem būtu iespēja apskatīt agrākos pasūtījumus, un lai administratoram būtu iespēja apvienot visus pirkumus konkrētai personai.

2.5. Noslēgums

Otrajā nodaļā bija izpētītas PHP programmēšanas valodas un MySQL datubāzes iespējas un piedāvāti daži piemēri, kuru uzdevums bija palīdzēt noskaidrot, kāpēc PHP un MySQL ir ērtas projektu realizācijai. Stāstot par iespējam, bija arī aprakstītas to priekšrocības un trūkumi.

Ar šo es beidzu stāstīt par darba praktisko daļu un tiecos klāt pie rezultātiem un secinājumiem, kurus aprakstīšu trešā un ceturtā nodaļās.

3. REZULTĀTI

Lai sasniegtu izvirzītos mērķus, es veicu sekojošo uzdevumus:

1) Pirmais uzdevums bija izpētīt datu glabāšanas iespējas. Bija jānoskaidro, kāda iespēja ir vairāk uzticama, pie kādiem datiem varam nokļūt ātrāk, vai varam saņemt datus pēc kāda kritērija, vai vairākiem lietotājiem ir iespēja vienlaicīgi sūtīt savus datus un kādu datu apjomu var glabāt. Darba gaitā nācu pie slēdziena, ka MySQL datubāze ir labāka datu glabāšanai. Šīs datubāzes pielietošana piedāvā labāku veiktspēju, nekā datnes. Izmantojot šo datubāzi, sistēma būs vairāk uzticama;

2) Otrais uzdevums bija izpētīt datu šifrēšanas iespējas. Bija jānoskaidro, kādu no iespējām labāk izmantot datu šifrēšanai, tas ir kāda no iespējām labāk aizsargā datus. Darba gaitā nācu pie slēdziena, ka labāka no PHP izpētītām funkcijām ir „sha1()", kura nodrošina drošāku informācijas aizsardzību;

3) Trešais uzdevums bija izpētīt lietotāju autentifikācijas iespējas. Bija jānoskaidro, kādas funkcijas pastāv, lai varētu realizēt šīs iespējas ar PHP un MySQL, un vai vispār tās var realizēt, kā arī jānoskaidro, kura no iespējam ir drošāka, uzticama, uzturama un lietojama. Darba gaitā nācu pie slēdziena, ka labāka no izpētītām iespējām, kuras var realizēt, pielietojot PHP un MySQL, ir autentifikācija ar USB-atslēgu, jo tā ir drošāka, uzturama, lietojama un uzticama;

4) Ceturtais uzdevums bija izpētīt informācijas apmaiņas starp klientu un serveri iespēju. Bija jānoskaidro, kādi ir plusi un mīnusi, strādājot ar interneta veikalu, kas bija izstrādāts ar PHP un MySQL, kā arī noskaidrot, kā informācijas apmaiņu padarīt par efektīvu un uzticamu. Darba gaitā nācu pie slēdziena, ka informācija var būt izmainīta pasūtīšanas laika, kas ietekmēs klienta apmaksas procesu, un tas ir trūkums; seansa mainīgā izmantošana produkta pievienošanas „grozam" laikā ir labāka metode, nekā pastāvīga datubāzes izmantošanas metode; ērtāk, uzticamāk un efektīvāk izdarīt tā, lai pieteiktie lietotāji veiktu pirkumus, lai viņiem būtu iespēja apskatīt agrākos pasūtījumus, un lai administratoram būtu iespēja apvienot visus pirkumus konkrētai personai.

4. SECINĀJUMI

Bakalaura darba mērķis bija izpētīt PHP programmēšanas valodas un MySQL datubāzes iespējas, kas rezultāta palīdzēja noskaidrot, kāpēc PHP un MySQL ir ērtas projektu realizācijai. PHP ir viena no populārākām tīmekļa programmēšanas valodām, kā arī MySQL ir viena no populārākām datubāzēm. PHP ir serveru scenāriju valoda, kas ir izstrādāta dinamiskas tīmekļa vietnes veidošanai. MySQL ir ātra un uzticama relāciju datubāzu vadīšanas sistēma, kas ļauj efektīvi glabāt, meklēt un izņemt informāciju. PHP un MySQL labi mijiedarbojas un tās var izmantot vairākās operētājsistēmās. Datus var glabāt datnēs un datubāzēs, taču otrajam variantam ir vairāk priekšrocības – MySQL nodrošina ātrāku piekļūšanu datiem nekā datnes; MySQL ļauj viegli atrast konkrētu ierakstu, norādot kritērijus; MySQL var glabāt lielāku datu apjomu, nekā datnes, kā arī citas priekšrocības. PHP valodai ir dažas funkcijas, kuras atbild par datu šifrēšanu – „md5()" ir biežāk lietota, bet „sha1()" nodrošina drošāku informācijas aizsardzību. Lietotāju autentifikācijas pētīšanas laikā bija konstatēts, ka lietotājvārda un paroles izmantošana ir visizplatītākā pasaulē, kas ļauj lietotājiem izveidot pašiem savus datus. Tomēr pastāv iespēja, ka viņu dati var būt uzlauzti. Autentifikācija ar USB-atslēgu dod labāku drošību nekā autentifikācija ar lietotājvārdu un paroli un ir ērtākā lietošanā, jo lietotājam nevajag atcerēties paroli. Biometrijas autentifikāciju nevar realizēt PHP valodā. Strādājot ar informācijas apmaiņu starp klientu un serveri, pielietojot interneta veikalu, bija konstatēts, ka informācija var būt izmainīta pasūtīšanas laikā, kas ietekmēs klienta apmaksas procesu, un seansa mainīgās izmantošana produkta pievienošanas „grozam" laikā ir labāka metode, nekā pastāvīga datubāzes izmantošanas metode.

Pētīšanas laikā tika konstatēts, kādas PHP un MySQL iespējas padara tās par uzticamām, lietojamām, verificējamām, drošām, uzturamām, saprotamām, un kādas iespējas nodrošina labu veiktspēju un produktivitāti.

Apskatot šīs iespējas varam secināt, ka PHP programmēšanas valoda un MySQL datubāze ir ērtas projektu realizācijai.

5. IZMANTOTĀS INFORMĀCIJAS AVOTI

Darba veikšanas laikā tika izmantoti sekojošas grāmatas un interneta resursi:

1. **Ullman, Larry.** *PHP 6 and MySQL 5 for dynamic web sites.* 2008;

2. **Burch, Carl.** *Programming via PHP.* 2007;

3. E-studijas Latvijas Universitātē [tiešsaiste]. Rīga: Latvijas Universitāte – [atsauce 09.04.2011.]. Pieejams: http://www.estudijas.lu.lv/;

4. MySQL rokasgrāmata (krievu val.). – [atsauce 11.04.2011.]. Pieejams: http://www.mysql.ru/docs/man/Introduction.html;

5. PHP: Hypertext Preprocessor – [atsauce 11.04.2011.]. Pieejams: http://www.php.net/;

6. PHP Tutorial – [atsauce 13.04.2011.]. Pieejams: http://www.w3schools.com/PHP/default.asp;

7. MySQL – [atsauce 13.04.2011.]. Pieejams: http://www.iho.ru/faq/manual/mysql.html;

8. MySQL realizēšanas vēsture (krievu val.). – [atsauce 14.04.2011.]. Pieejams: http://ru.wikipedia.org/wiki/MySQL;

9. MySQL version numbers – [atsauce 14.04.2011.]. Pieejams: http://searchsystemschannel.techtarget.com/feature/MySQL-version-numbers;

10. MySQL dibināšanas vēsture (krievu val.). – [atsauce 14.04.2011.]. Pieejams: http://www.getinfo.ru/article438.html;

11. Rasmus Lerdorf – [atsauce 12.04.2011.]. Pieejams: http://en.wikipedia.org/wiki/Rasmus_Lerdorf;

12. Usage of PHP – [atsauce 12.04.2011.]. Pieejams: http://en.wikipedia.org/wiki/PHP;

13. PHP pielietošanas sfēra (krievu val.). – [atsauce 12.04.2011.]. Pieejams: http://ru.wikipedia.org/wiki/PHP;

14. PHP vēsture (krievu val.). – [atsauce 12.04.2011.]. Pieejams: http://ru.wikipedia.org/wiki/История_PHP;

15. PHP un drošība (krievu val.). – [atsauce 04.05.2011.]. Pieejams: http://i-vd.org.ru/books/php/security.shtml;

16. PHP and other languages – [atsauce 04.05.2011.]. Pieejams: http://php.net/manual/en/faq.languages.php;

17. MySQL un PostgreSQL (krievu val.). – [atsauce 04.05.2011.]. Pieejams: http://mysql.ru/docs/man/MySQL-PostgreSQL_features.html;

18. MySQL un mSQL (krievu val.). – [atsauce 04.05.2011.]. Pieejams: http://mysql.ru/docs/man/Compare_mSQL.html;

19. Internet-Programmēšana (krievu val.). – [atsauce 04.05.2011.]. Pieejams: http://www.mirsite.ru/;

20. Crypt function (krievu val.). – [atsauce 05.05.2011.]. Pieejams: http://www.php.net/manual/ru/function.crypt.php;

21. Pieejas drošība izmantojot biometrijas metodes (krievu val.). – [atsauce 08.05.2011.]. Pieejams: http://www.rusdoc.ru/articles/zaschita_dostupa_biometricheskimi_metodami/15886/;

22. SSH-autentifikācija ar USB-atslēgas palīdzības (krievu val.). – [atsauce 08.05.2011.]. Pieejams: http://habrahabr.ru/blogs/personal/88540/;

23. What is client-server architecture – [atsauce 10.05.2011.]. Pieejams: http://www.wisegeek.com/what-is-client-server-architecture.htm;

24. Klienta servera arhitektūra (krievu val.). – [atsauce 10.05.2011.]. Pieejams: http://belani.narod.ru/1/Lklser2.htm;

25. Product Usability – [atsauce 19.05.2011.]. Pieejams: http://www2.uiah.fi/projects/metodi/168.htm;

26. **Tucker, Allen B.** *Computer science handbook.* 2004.

DOKUMENTĀRĀ LAPA

Bakalaura darbs „PHP programmēšanas valodas un MySQL datubāzes iespēju pētījums projektu realizācijai" izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai. Piekrītu sava darba publicēšanai internetā.

Autors: Andrejs Vorobjovs _____

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: Uldis Straujums _____

Recenzents: _____

Darbs iesniegts Datorikas fakultātes sekretariātā 2011.gadā _____
Metodiķe: _____

Darbs aizstāvēts bakalaura darbu gala pārbaudījuma komisijas sēdē 2011.gadā _____ , prot. Nr. _____, vērtējums _____

Bakalaura darba pārbaudījumu komisija: _____