LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

TĪMEKĻA LIETOTNES IZSTRĀDE IMPERATĪVĀ UN FUNKCIONĀLĀ STILĀ

BAKALAURA DARBS

Autors: **Krista Puķe**
Studenta apliecības Nr. : kp09044
Darba vadītājs: profesors Dr.dat. Kārlis Čerāns

Rīga 2013

ANOTĀCIJA

Pētījuma centrā ir tīmekļa lietotņu izrāde funkcionālā un imperatīvā stilā. Darbā ir aprakstītas programmēšanas valodas, kas tiek iekļautas imperatīvajā vai funkcionālajā stilā un kuras lieto lai izveidotu tīmekļa vietni.

Pētījumā tiek izvirzīts un aprakstīts konkrēts uzdevums, kura realizācija aprakstīta imperatīvā stila valodā PHP un funkcionālā stila valodā Racket. PHP ir atvērtā koda servera puses valoda savukārt Racket ir vairāku-paradigmu loģiskā programmēšanas valoda. Abu valodu realizācija tiek salīdzināta atklājot to, ka izveidot tīmekļa lietotni ir iespējams kā izmantojot imperatīvā stila tā funkcionālā stila valodas.

Darbā tiek izmantota viena tabula un četrpadsmit attēli, darba apjoms ir 57 lapaspuses. Bakalaura darbam ir divi pielikumi un izmantoti 37 informācijas avoti.

ANOTATION

The scope of this research is a web application development in functional and imperative style. This paper outlines programming languages used to develop a web site in imperative, as well as functional style.

This research focuses on a specific task, the implementation of which is outlined in PHP imperative style language and Racket functional style language. PHP is open source server-side language, while Racket is multi-paradigm logical programming language. The implementation of both languages was compared with the conclusion that it is possible to develop a web application by using both imperative and functional style languages.

This bachelor thesis is written on 57 pages in Latvian, it contains 1 table, 14 images, 2 appendices and 37 sources of bibliography.

SATURS

APZĪMĒJUMU SARAKSTS

Nr.p.k	Apzīmējums	Nozīme
1.	Html	Hiperteksta iezīmēšanas valoda (no angļu: *HyperText Markup Language*) ir iezīmēšanas valoda, kas ir izstrādāta tīmekļa lappušu un citas pārlūkprogrammā attēlojamas informācijas glabāšanai
2.	MySql	Relāciju datubāzu vadības sistēma (DBVS)
3.	HTTP	Hiperteksta transporta protokols (no angļu: *HyperText Transfer Protocol*) ir lietojumslāņa protokols, kas paredzēts datu apmaiņai starp tīmekļa serveriem un pārlūkprogrammām
4.	Apache	Tīmekļa serveris
5.	Mainīgais	Programmēšanā - rakstzīme vai rakstzīmju secība, ko izmanto, lai apzīmētu kādu saglabājamu lielumu, kam ir savs vārds un vērtība un ko var mainīt programmas izpildes gaitā
6.	VT	VIA Technologies izstrādāta x86 virtualizācijas tehnoloģija
7.	CGI	Standarts, kas nosaka, kādā veidā parametri no HTML formas tiek nodoti programmai, kas dinamiski veido atbilstošo HTML lappusi
8.	WWW	Vispasaules tīmeklis (saīsinājums *WWW* vai *W3*, plašāk pazīstams kā *Tīkls*) ir sistēma jeb telpa ar savienotiem hiperteksta dokumentiem, kuriem piekļūst izmantojot Internetu
9.	JVM	Vide, kurā darbināt JAVA programmas
10.	CAE	Datoru atbalstīta, automatizēta inženierija
11.	CAD	Automatizēta izstrāde
12.	AutoCad	Lietojumprogramma, kas paredzēta datorizētajai projektēšanai(CAD), kur ir iespējams rasēt gan 2D, gan 3D objektus
13.	GIMP	GNU Image Manipulation Program ir rastra grafikas redaktors, ko lieto fotogrāfiju un citu attēlu apstrādei un izveidei
14.	GHC	Glazgova Haskell kompilators
15.	MySQL Workbench	Vizuāls datu bāzes dizaina rīks, kas pilnveido SQL attīstību, administrēšanu, datu bāzes projektēšanu, izveidošanu un uzturēšanu vienā integrētā izstrādes vidē uz MySQL datubāzes sistēmas

IEVADS

Tīmekļa vietne (angliski „*Website”*) ir tīmekļa lappušu kopums, ko veido caur HTTP protokolu Internetā pieejami HTML/XHTML dokumenti. Visas publiski pieejamās tīmekļa vietnes veido vispasaules tīmekli. [faa]

Interneta pārlūka izmantošana tiešā vai netiešā veidā ir saistīta ar tīmekļa vietņu aplūkošanu. Tīmekļa izstrāde ir tīmekļa vietņu vai lietotņu izveidošanas process, kas ietver sevī ietver tādus procesus kā tīmekļa dizaina veidošana, tīmekļa programmēšanu no klienta un servera puses, kā arī tīmekļa servera konfigurēšanu. [gaa]

Izstrādātajā darbā tiek aprakstītas tīmekļa lapu izstrādes valodas, kuras iekļaujas imperatīvā vai funkcionālā stilā. Stilu atšķirībai šajā darbā tiek likt liels uzsvars, jo izstrādājot tīmekļa vietni, bieži vien nespējam iedomāties kā norisinātos izstrādes gaita izmantojot funkcionālā stila valodas, piemēram, tādu kā Racket.

Darbā izvirzītais mērķis ir salīdzināt dažādas tehnoloģijas, tīmekļa vietņu izstrādei, un kā darba metode izmantota tehnoloģiju salīdzināšana, pielietojot konkrētu uzdevumu tīmekļa vietnes izstrādei.

Mērķa īstenošanai tika izvirzīti šādi uzdevumi:

- Aprakstīt tehnoloģijas tīmekļa vietņu izstrādei;
- Izstrādāt uzdevuma aprakstu un entītiju relāciju modeli;
- Izstrādāt un aprakstīt risinājumu valodā PHP;
- Izstrādāt un aprakstīt risinājumu valodā Racket.

Bakalaura darbs sastāv no četrām nodaļām, ievada, rezultātiem, secinājumiem un pielikumiem. Darba pirmajā nodaļā tiek aprakstītas tīmekļa lietotnes izstrādes valodas, kuras iekļaujas funkcionālā vai imperatīvā stilā. Otrajā nodāļā tiek aprakstīts uzdevums, entītiju relāciju diagramma. Trešā nodaļa tiek veltīta tīmekļa lietotnes aprakstam valodā php un ceturtā nodāļā tiek aprakstīta izstrāde funkcionālajā valodā Racket. Pielikumā tiek pievienots rakstītais kods un failu nosaukumi PHP un Racket valodās.

1. TEHNOLOĢIJU PĀRSKATS

Lai izveidotu tīmekļa lietotni pamatā ir jāizvēlas tehnoloģija, kurā to izstrādāt. Tīmekļa programmēšana ir Internetā bāzētu informācijas sistēmu izstrādes pamatā. Mūsdienīgas tīmeklī bāzētas sistēmas raksturo izmantoto tehnoloģiju dažādība, integrācija dažādu tīmekļa sistēmu starpā un resursu koplietošana. [haa]

1.1. Populārākās programmēšanas valodas

Katru gadu Janvārī tradicionāli tiek salīdzinātas programmēšanas valodas pēc *TIOBE* indeksa. *TIOBE* indeksu izmanto lai saprātīgi salīdzinātu programmēšanas valodu popularitātes augšanas vai dilšanas tendences. [aaa]

TIOBE indekss ir programmēšanas popularitātes rādītājs, kas tiek iegūts no vairāku meklētājprogrammu vaicājumiem, kas ietver programmēšanas valodas vārdu. Indekss tiek atjaunots katru mēnesi un tiek iegūts no šādām meklētājprogrammām: *Google, Blogger, Wikipedia, YouTube, Baidu, Bing, Amazon un Yahoo!*. Tomēr saskaņā ar TIOBE indeksu nevar noteikt kura ir labākā vai sliktākā programmēšanas valoda. [aab]

2013.gadā pēc TIOBE indeksa apskatītās valodas, izņemot ASP un RACKET, ierindojas pirmajās 50 pozīcijās, attiecīgi to reitings nav zemāks par 0.323%. Visaugstāk no visām apskatītajām valodām ierindojas JAVA, bet viszemāk Haskell (skat.1.1. tabulu).

1.1. tabula

Tīmekļa lietotņu izstrādes valodas un TIOBE indekss

Valoda	TIOBE indekss	Pozīcija
JAVA	16.914%	2
C#	6.119%	5
PHP	5.784%	6
Python	4.322%	8
Perl	2.276%	10
Ruby	1.670%	11
Lisp	0.894%	13

Scheme	0.396%	28
Haskell	0.323%	30
Racket	Nav informācija	Nav informācija
ASP	Nav informācija	Nav informācija

1.1.1. Imperatīvā stila programmēšanas valodas

Imperatīvā stila programmēšana reprezentē tradicionālo veidu, kā notiek

programmēšanas process. Imperatīvās programmēšanas process ir komandu virknes pierakstīšana noteiktu operāciju veikšanai. Imperatīvo stilu pārstāv: PHP, Ruby, Perl, Python, ASP, Java, C#. Imperatīvā programmēšana balstās uz programmas sadalīšanu vienā vai vairākos vienumos un to redzamību (angliski „visibility"), kur katrs vienums sastāv no vienas vai vairākām procedūrām (funkcijām), tādā veidā dažādiem vienumiem ir pieeja pie lokāliem un globāliem mainīgajiem. Šīs metodes ieguvums ir interpretācijas vienkāršība un izmantošanas elastība. [aca,ada]

Imperatīvā programmēšana nozīmē, ka programmētājs apzināti veido programmu kā priekšrakstu virkni, kuru noteiktā secībā izpildot, viņš tiešā veidā maina mašīnas atmiņas reģistru un ārējo iekārtu stāvokli, tādējādi no ieejas datiem iegūstot rezultātu. [aba]

1.1.1.1. PHP

PHP (angliski „ Hypertext Preprocessor") ir atvērtā koda servera puses skriptu valoda, kas pamatā domāta dinamisku Interneta lapu veidošanai. Tas var tikt uzskatīts par alternatīvu Microsoft'a ASP/VBScript/Jscript tehnoloģijām, Sun Microsystems' JSP/Java un CGI/Perl tehnoloģijai.

Tomēr PHP ir vairāk kā skriptu valoda. To var izmantot arī GUI aplikāciju izveidē kā arī palaist no komandrindas kā Perl vai Python. PHP ir platformneatkarīga valoda. To var darbināt gan Windows operētājsistēmā, gan uz daudzām Unix versijām un Mac OS X. Visbiežāk PHP tiek lietots kopā ar MySQL relāciju datu bāzi. Vēl tas ir savienojams ar Oracle, MS SQL, PostgreSQL, DB2 utt. [aap]

Tā kā PHP ir servera puses valoda, tad vajadzīgs attiecīgs tīmekļa serveris, uz kā skripti varētu izpildīties. Izplatītākais savienojums ir PHP un Apache tīmekļa serveris. Lai gan tas nebūt nav vienīgais serveris, ar ko PHP ir savietojams. PHP mijiedarbojas arī ar tādiem tīmekļa serveriem kā Microsoft IIS un Netscape Enterprise Server. [aap]

Sākotnējā PHP versija parādījās 1994.gadā, kad Dāni/Grenldndietis Rasmus Lerdorfs izveidoja CGI bināros failus, kuri varētu aizvietot Perl skriptus, kurus viņš lietoja tīmekļa lapās. Sākotnēji PHP veidoja lai tas apstrādātu formas, tādējādi PHP sākotnējais saīsinājums bija no *Personal Home Page*. 1997.gadā projektam pievienojās divi Izraēlieši Zeev Suraski un Andi Gutmans, kuri nolēma pārrakstīt parseri, jo tas nebija pietiekami labs lai veidotu e-komercijas tīmekļa lapas, kā rezultātā ieguva jaunu nosaukumu PHP: Hypertext Preprocessor. Līdz PHP 6 versijai tika izlabotas kritiskas kļūdas, tika pievienots spēcīgs OOP (Objektorientētās programmēšanas atbalsts) un Unicode atbalsts. [aar]

```
1  <html>
2   <head>
3   </head>
4   <body>
5   <?php echo '<p>Hello World</p>'; ?>
6   </body>
7  </html>
```

1.1. att. **Programma „Hello World" valodā PHP**

1.1.1.2. Ruby

Ruby ir interpretējama scenāriju valoda ātrai un vieglai objektu orientētai programmēšanai. Tam ir daudz iespēju teksta failu apstrādāšanai un sistēmas vadības uzdevumu veikšanai (kā Perl'ā). [afa] Ruby ir atvērtā koda programmēšanas valoda, kurā uzsvars tiek likts uz vienkāršību un produktivitāti. Valodas sintakse ir ļoti naturāla – līdzīga dabīgai valodai, un tāpēc ir salīdzinoši viegli apgūstama pat iesācējam. [aga]

Ruby ir salīdzinoši jauna programmēšanas valoda, tā parādījās 1995. gadā un tās autors ir Yukihiro Matsumoto. Ruby autoram bija pazīstams Perl un Python. Abās valodās viņš saredzēja trūkumus un kā objektorientētas programmēšanas fans viņš nolēma rādīt savu patiesi objektorientētu viegli pielietojamu scenāriju valodu. Pēc dažām mēnešiem paradījās pirmā interpretatora versija. Pēc kāda laika, autors reorganizēja Perl iespējas kā klašu bibliotēku un realizēja tos savā valodā. Kopš Ruby publiskošanas, valoda stauji guva popularitāti, tai parādījās jaunas iespējas un bibliotēkas, kas ļauj izmantot šo valodu gandrīz jebkurā sfērā un jebkuram nolūkām. Valoda turpina attīstīties arī šodien. [aea]

Ruby ir vienkārša sintakse, daļēji iespaidota ar Eiffel un Ada, valodai ir izņēmumu apstrādes iespējas, līdzīgas Java vai Python, vieglai kļūdu apstrādāšanai. Ruby operatori ir viegli pārdefinējami un tā ir pilnīgi objektorientēta valoda. Tas nozīmē, ka visi dati iekš Ruby ir objekti. Piemēram, skaitlis 1 ir klases Fixnum instance. Ruby uztur tikai vienkāršu mantojamību. Valodai ir pazīstama moduļu koncepcija(moduļi ir metožu kolekcijas), kur katra klase var importēt moduli un iegūt visas viņa metodes. Ruby nav nepieciešams deklarēt mainīgus, viņš izmanto mainīga vārdu, lai noskaidrotu mainīga redzesloku. Piemēram: 'var' –

lokālais mainīgais, ‘@var’ – instances mainīgais, ‘$var’ – globālais mainīgais utt. un var izmantot paplašinājumu bibliotēkas, ja OS to atļauj. Ruby uztur OS neatkarīgu vairākpavedienošanu (multithreading), neatkarīgi no tā vai OS to uztur vai nē. [afa]

```
1  #!/usr/bin/env ruby
2  print "HTTP/1.0 200 OK\r\n"
3  print "Content-type: text/html\r\n\r\n"
4  print "<html><body>Hello World!</body></html>\r\n"
```

1.2. att. **Programma „Hello World” valodā Ruby**

1.1.1.3. Perl

Perl *(Practical Extraction and Report Language)* ir augstā līmeņa, dinamiska programmēšanas valoda, kuru 1987. gadā radīja Larijs Valls. Tās izstrādes rīki vienmēr ir bijuši brīvi pieejami, tas nozīmē, ka ikviens bezmaksas var lejupielādēt Perl interpretatoru - programmu, kas nepieciešama, lai varētu darbināt Perl'ā rakstītas programmas. [aaz]

Fakts, ka tā ir dabūjama par velti, nav vienīgais, kas to ir padarījis ārkārtīgi populāru, it sevišķi **VT** pielietojumiem. Valoda ir ļoti piemērota dažāda veida manipulācijām ar tekstiem - *Perl* ir saīsinājums no *Practical Extraction and Report Language* (praktiska izvilkumu un atskaišu veidošanas valoda). Ir pavisam viegli manipulēt ar **CGI** standartā iekodētiem parametriem un veidot dinamiskas HTML lappuses. Valoda ir interpretēta, nevis kompilēta - tas nozīmē, ka *Perl* interpretators pakāpeniski lasa programmas tekstu un izpilda instrukcijas, nevis, kā tas būtu kompilētas valodas gadījumā, kad kompilators visu programmu pārveidotu datoram saprotamā mašīnvalodā, un tikai tad tā būtu izpildāma. Strādāt ar interpretētām valodām ir daudz vieglāk, jo ātri var pārbaudīt dažādas pieejas vienai problēmai. Vienīgais trūkums ir ātrdarbības samazināšanās, bet vienkāršām manipulācijām ar tekstiem tas nav tik būtiski. [ama]

Perl tiek strauji paplašināta, iekļaujot daudzas iebūvētas funkcijas, datu tipus, regulāro izteiksmju apstrādi, objektorientētās iespējas, funkcionālās programmēšanas līdzekļus un citas noderīgas lietas. Mūsdienās Perl ir populārs līdzeklis tekstuālas informācijas apstrādei, automātiskai WWW lapu veidošanai, sistēmu administrēšanai un citiem pielietojumiem. Veidojot Perl valodu, praktiskā noderība ir vērtēta augstāk par teorētisko eleganci. Ne velti pirmais vārds nosaukumā PERL tiek tulkots kā “practical”. Vēl viena būtiska iezīme ir līdzība uzbūvē ar dabiskajām valodām. Perl ar savām plašajām iespējām ir liela un sarežģīta valoda salīdzinot ar C vai Scheme. [ana]

```
1  #!/usr/bin/perl
2  # hello.pl -- my first perl script!
3  print "Content-type: text/html\n\n";
4  print "Hello, world!\n";
```

1.3. att. **Programma „Hello World” valodā Perl**

1.1.1.4. Python

Python (no angļu: 'pitons', izrunā kā 'paiton') ir augsta līmeņa programmēšanas valoda. Tā ir interpretējama objektorientētā skriptu valoda. Python valoda ir veidota, akcentu liekot uz programmētāja ērtībām pār datoru, programmas lasāmību pār ātrumu. [aau] Python ir objektorientēta valoda, taču šo programmēšanas principu neuzspiež, ļaujot programmēt gan objektorientētā, gan tradicionālā imperatīvā stilā, gan kā tas visbiežāk tiek praktizēts – abu šo stilu kombinācijā. [aav]

Jaunu programmēšanas valodu 1991. gadā radīja Gvido van Rossums (*Guido van Rossum*), kad viņš strādāja firmā *CWI* Nīderlandē. Tajā laikā viņš strādāja ar operētājsistēmu *Amoeba*, un bija nepieciešama valoda, kas būtu līdzīga ABC programmēšanas valodai, bet ar vieglu piekļuvi sistēmas funkcijām. 1989. gada beigās Gvido sāka projektēt valodu, kurai būtu paplašināšanas iespējas.

Pēc Gvido van Rossuma ieceres jaunās valodas nosaukumam bija jābūt īsam, unikālam un nedaudz noslēpumainam. Viņš to nosauca par "Python", kas bija aizgūts no BBC televīzijas komēdiju seriāla *Monty Python*. [aau]

Guido van Rossums mērķis bija radīt Python kā otro valodu tiem, kas programmē C vai C++. Tā būtu lietojama gadījumos, kad nepieciešams uzrakstīt kādu nelielu, vienkāršu programmu, kuru rakstīt C vai C++ būtu pārāk laikietilpīgi. Kaut arī radīta kā *otrā valoda*, tā pat reiz ir ieguvusi lielu popularitāti un daudziem kļuvusi arī kā *pirmā valoda*. Ja sākotnēji autors uzskatīja, ka Python tiks lietota pārdesmit rindiņu garu programmu rakstīšanai, tad pat reiz šajā valodā ir izveidotas daudzas sarežģītas aplikācijas, piemēram, *mailing-listu* menedžeris *mailman*, satura vadības sistēma *Zope*, un daudzas citas. Tas viss noticis pateicoties plašajam iespēju klāstam, kuru piedāvā Python un tā paplašinājumi. [aav]

```
1  #!/usr/bin/env python
2
3  print "Content-type: text/html\n\n"
4  print "<html>Hello world!</html>"
```

1.4. att. **Programma „Hello World" valodā Python**

1.1.1.5. ASP

Active Server Pages jeb ASP (no angļu: "Aktīvās servera lapas") ir Microsoft dinamiski ģenerējamu HTML lapu servera puses skripta dzinis un programmēšanas vide. Sākotnēji izlaists 1998. gadā kā Internet Information Services pievienojums operētājsistēmas Windows NT 4.0 papildpakā *Option Pack*, tas vēlāk kļuva par *Windows Server* brīvi iekļaujamu

komponentu. ASP pēdējā versija 3.0 iznāca 2000. gadā; tās pēctecis ir ASP.NET. No ASP.NET skatu punkta ASP sauc par klasisko ASP (*Classic ASP*). [aja]

ASP piedāvā septiņus iebūvētus objektus: *Application*, *ASPError*, *Request*, *ObjectContext*, *Response*, *Server* un *Session*. Tie ir ASP pamats un ir paredzēti datu nosūtīšanai klientam, informācijas iegūšanai no klientam pārlūkprogrammas, datu apmaiņā ar serveri, datu kešošanai un kļūdu apstrādei. ASP izmanto Active Scripting dzini, kas atbalsta Component Object Model (COM). Var pieslēgt ārējus COM komponentus. Var izmantot dažādas skriptu valodas, kuras atbalsta konkrētā servera dzinis. Populārākā (tā ir pēc noklusējuma) valoda ir VBScript. Iespējams vienā lapā vienlaicīgi izmantot vairākas valodas, attiecīgā bloka sākumā norādot, kura valoda tiek lietota. ASP lapu faila paplašinājums ir *.asp*, lai gan, attiecīgi nokonfigurējot serveri, var izmantot arī ierastos *.htm* vai *.html*. Programmējot, ASP kods tiek ielikt HLML lapā starp tagiem <% un %>. [aja]

ASP ir servera puses skriptu valoda, kuru var iekļaut HTML, lai radītu dinamiskas tīmekļa lapas. ASP kodu interpretē tīmekļa serveri, kas pēc tam izvada lapu pārlūkprogrammā. ASP ir ļoti līdzīgs PHP, to galvenokārt atbalsta Windows serveri, bet ir mēģināts izveidot arī uz LINUX servera, piemēram, *Sun's Chilisoft.* Valodu var izmantot lai saņemtu un nosūtītu e-pastus, veidot savienojumu ar citiem serveriem, iegūtu formas informāciju, saglabāt informāciju datubāzē, izveidot sīkdatnes vai sesijas ar interneta pārlūku, u.c. ASP var savienot ar vairāku veidu datubāzēm, bet visbiežāk to lieto ar *MS SQL*. Valodā tiek atbalstītas arī funkcijas (blokus ar kodu, kas var darīt konkrētas lietas vairākas reizes) un objektus (funkciju grupas, kas pārstāv reālās pasaules objektus, piemēram, kā grāmata vai lietotājs). [ala]

```
1  <html>
2  <body>
3  <%
4     Response.write "Hello World!"
5  %>
6  </body>
7  </html>
```

1.5. att. **Programma „Hello World" valodā ASP**

1.1.1.6. JAVA

Java ir objektorientēta programmēšanas valoda ko 1991. gadā izstrādāja *Sun Microsystems* darbinieki: Džeimss Goslings (*James Gosling*), Patriks Noutons (*Patrick Naughton*), Kriss Varts (*Chris Warth*), Eds Franks (*Ed Frank*) un Maiks Šeridans (*Mike Sheridan*), kā daļu no Zaļā projekta. Java valodas sākotnējais nosaukums bija Oak, tā tika veidota kā *C++* aizstājēja, lai gan tās funkcionalitāte ir tuvāka *Objective C*. [aai]

Programmēšanas valodu JAVA izstrādāja *Sun Microsystems*, lai izmantotu mobilā tīkla programmatūras nodrošinājumam. Taču drīz vien *Sun Microsystems* nācās šo valodu modificēt, lai tā būtu izmantojama tīmekļa pārlūkprogrammās. Tātad, lai būtu izmantojama arī *World Wide Web* (*WWW*) un valodā *HTML* (*Hypertext Markup Language*). Valoda JAVA ir viena no modernākajām un progresīvākajām programmēšanas valodām. Šī valoda ir objektorientēta, daudzplūsmu, augstražīga, kā arī tai piemīt daudzas citas moderno programmēšanas valodu īpašības. [aaj]

Java ir izstrādāta uz C++ pamata un ir ļoti līdzīga tai pēc stila un struktūras. Tā ir liela valoda ar vairākiem slāņiem. Lai apgūtu Java pilnībā, ir nepieciešams daudz laika un darba, taču Java pielietošana pēc tās apgūšanas ir ļoti efektīva, un ir iespēja pakāpeniski paaugstināt savu kvalifikāciju. [aha]

Java valoda ir vienkārša, jo ir veiksmīgs kompromiss starp koda īsumu un lasāmību un tā ir objektorientēta, viss kods strikti dalās pakotnēs un klasēs, izņemot 8 primitīvos tipus (int, boolen, u.c.) visi dati ir objektorientēti. Valoda ir arī tīklorientēta, jo resursus (datus, kodu) viegli savākt tīklā un ir viegli radīt klienta/servera vai daudzslāņu aplikācijas. Valodas interpretējamība veicina izstrādes ātrumu (pēc katras izmaiņas nav jāatkārto pilns kompilācijas/linkošanas/ielādes/testēšanas cikls), koda pārnesamību (programmu var darbināt uz jebkuras specifikai atbilstošas **JVM**), drošību (interpretators atšķirībā no datora dzelžiem, var labāk pārbadīt, vai kods nedara ko neatļautu). Programmētāja kļūdas un citas neparedzētas situācijas izraisa Javas izņēmumus (piemēram, mēģinājums rakstīt ārpus masīva robežām, ko C++ kods nekontrolē). Java platforma var izpildīt neuzticamus appletus un citu kodu, to darbinot Javas interpretatora smilšukastē. Javas programmu sakompilēto starpformātu (.CLASS failus) var darbināt uz visām platformām, uz kurām ir Javas interpretators. Klases ielādē izpildes laikā. Komponentes programmas izpildei var savākt arī, teiksim, Internetā. tīmekļa aplikācijās tātad viegli nomainīt izmantojamos datubāzu draiverus un citas palīgbibliotēkas. [aia]

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3          System.out.println("Hello, world!");
4      }
5  }
```

1.6. att. **Programma „Hello World" valodā JAVA**

1.1.1.7. C#

C# (izrunā: *sī šārp*) ir kompānijas Microsoft izstrādāta daudzparadigmu programmēšanas valoda. Sākotnēji tā bija paredzēta speciāli .NET izpildes videi, bet vēlāk tai tika apstiprināti ECMA un ISO standarti. C# ir viena no programmēšanas valodām, kas

paredzētas *Common Language Infrastructure (CLI)*. C# ir paredzēta kā vienkārša, moderna, plaša pielietojuma, objektorientēta programmēšanas valoda. Tās izstrādes komandu vada Anderss Heilsbergs. Jaunākā valodas versija ir C# 5.0, izlaista 2012. gada 15. augustā. [aan]

Kaut gan pati valoda ir samērā jauna (C# specifikācijas tika izlaistas 2000. gada jūnijā), pie tās pamatiem Microsoft speciālisti strādāja gadiem ilgi. C# tiek uzskatīts par mūsdienīgu objektorientētu valodu, kas daudz savu īpašību manto no C, C++ un Java valodām. Tas sintakse visvairāk ir līdzīga Java un C++ valodām un tā tiek uzskatīta par nākamo soli objektorientētu valodu attīstībā. [aao]

C# ir objektorientēta valoda, kas atbalsta visas ierastās koncepcijas un abstrakcijas, kas eksistē Java un C++ valodās. Kā tiek prasīts visām modernām valodām, C# atbalsta mantošanu, abstrakciju, polimorfismu un programmēšanu bāzētu uz interfeisiem. C# arī atbalsta ierastās C, C++ un Java konstrukcijas, kā klases, struktūras, interfeisi, pārskaitījuma tipi, lietotāja atribūtus u.t.t. Vēl C# valodai piemīt tādas C++ īpašības, ka operatoru pārlādēšana, lietotāja definētas konversijas un parametra nodošana pēc norādes. C# valodai atšķirībā no daudzām citām valodām nav savas palaišanas laika bibliotēkas (*runtime library*). Tā vietā C# paļaujas uz .NET Framework plašas klašu bibliotēkas visās savās vajadzībās – gan ievades/izvades operācijās, gan datu struktūrās, gan tīkla operācijās. [aao]

```
1.      using System;
2.      using System.Web;
3.      using System.Web.Services;
4.      using System.Web.Services.Protocols;
5.
6.      [WebService(Namespace = "http://tempuri.org/")]
7.      [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
8.      public class Service : System.Web.Services.WebService
9.      {
10.         public Service () {
11.         }
12.         [WebMethod]
13.         public string HelloWorld() {
14.             return "Hello World";
15.         }
16.     }
```

1.7. att. **Programma „Hello World”**

1.1.2. Funkcionālā stila programmēšanas valodas

Funkcionālā programmēšana nozīmē programmas veidošanu kā matemātisku funkciju, kas no ieejas datiem iegūst izejas datus. Šo funkciju savukārt definē izsaucot citas funkcijas, utt. līdz esam nonākuši līdz pamatfunkcijām, kas ir pašas valodas sastāvdaļa. No imperatīvās programmēšanas funkcijām tas atšķiras ar iespējami nelielu blakusefektu lietošanu, piemēram, funkcionālajā programmēšanā nemēdz lietot piešķiršanas operācijas, lai mainīgajos kopētu

jaunas vērtības. Vissenāko funkcionālo valodu LISP un tās dialektu Scheme joprojām lieto praksē, it īpaši tekstu apstrādē un mākslīgajā intelektā. [aoa]

1.1.2.1. Lisp

LISP ir jaudīga un arī ērta programmēšanas valoda. 1950.gadu beigās to izveidoja Masačūsetsas tehnoloģijas institūta darbinieks Džons Makkartijs (McCarthy). Šodien to pielieto augsta līmeņa programmētāji, ar datora palīdzību modelējot procesus fizikā, datorizētajā projektēšanā (Computer aided drafting (CAD)), datorizētajā inženiersistēmu vadībā (Computer aided engineering (CAE)), kā arī ģenētikā (computer-aided genetics(CAG)). LISP programmēšanas valodu lieto arī tūkstošiem cilvēku, kuri sevi galīgi neuzskata par programmētājiem, bet vēlas sev atvieglot dzīvi, automatizējot darbības kuras atkārtojas AutoCAD'ā vai GIMP'ā. Izprotot LISP valodu, tā palīdz izprast arī datora darbības principus. [apa]

Lisp sākotnēji tika veidota kā praktisku matemātisko apzīmējumu datorprogrammu. Valoda ātri kļuva par populāro mākslīgā intelekta programmēšanas valodu. Lisp ieviesa vairākas idejas datorzinātnēs, tai skaitā koku datu struktūras, automātiskas saglabāšanas vadība, dinamisku rakstīšanu un *self-hosting* kompilatoru. Valodas nosaukums cēlies no „*List processing* (sarakstu apstrādāšana)" . Saistītie saraksti ir viena no galvenajām Lips valodas datu struktūrām. Lisp pirmkods pats par sevi ir veidots no sarakstiem. Tā rezultātā Lisp programmas var manipulēt ar pirmkodu, kā datu struktūru, kas rada makro sistēmas ļaujot izveidot jaunu sintaksi vai pat domain-specific valodas integrētas Lisp. Pēc koda un datu savstarpējās aizvietojamības var atpazīt Lisp sintaksi. [ara]

Viss programmas kods tiek rakstīts S-izteiksmēs1, piemēram, funkcijas f izsaukums ar 3 argumentiem a, b un c tiek pierakstīts kā (f a b c). Tā kā pārsvarā darbs notiek ar strukturētiem sarakstos datiem, bieži vien ir nepieciešamība veikt kādas darbības ar katru saraksta elementu. [asa]

```
1  (defun app (env)
2    (declare (ignore env))
3    '(200
4      (:content-type "text/plain")
5      ("Hello, World!")))
6
7  (clack:clackup #'app)
```

1.8. att. **Programma „Hello World"**

1.1.2.2. Haskell

Haskell ir standartizēta dažāda pielietojuma pilnīgi funkcionāla programmēšanas valoda ar ne-striktu semantiku un stipru, statisku tipu sistēmu. Tā nosaukta par godu loģiķim

Haskelam Kurijam (*Haskell Curry*). Haskell ir balstīts uz lambda rēķiniem, tāpēc arī valodas logo ir lambda. [ata]

Lai arī Haskell implementē vecākās funkcionālās valodās realizētas iespējas, tajā tika realizētas arī jaunas iespējas. Galvenais Haskell radītāju jaunievedums funkcionālajās programmēšanas valodās ir tipu klases, kas ļauj veikt pārslogošanu. Pirmā Haskell versija (1.0) tika definēta 1990. gadā. Līdz 1997. gadam tika radīta jau 4. Haskell versija (1.4) un 1997. gadā tika pieņemts lēmums par stabilas Haskell versijas radīšanu. Šī versija tika nosaukta Haskell 98.[aua]

Populārākie valodas kompilatori ir GHC (Glasgow Haskell Compiler), kas ir pamatīgākais un nopietnākais no pieejamajiem kompilatoriem, un Hugs, kas ir ātrs un ērts un ļoti piemērots tiem, kas tikai sāk apgūt Haskell vai arī neprasa milzīgas iespējas un augstu programmu ātrdarbību. [aua]

Haskell ir liela, sarežģīta valoda, tai ir vairākas standartizētas versijas un daudzi nestandartizēti paplašinājumi. Tai ir daudzas kopīgas īpašības ar imperatīvām valodām un citām funkcionālām valodām. Tai ir arī vairākas unikālas īpašības. Galvenās valodas iezīmes ir: pēc noklusējuma norādošs caurspīdīgums, pēc noklusējuma slinka izpildīšana, statiska tipu sistēma, par pamatu ņemti F sistēmas lambda rēķini, algebriski datu tipi, šablonu pieskaņošana datiem, tipu klases, biežs *curry* lietojums. [ata]

```
1   import CPS
2   import UnsafeJS
3   import CDOM.Level2.DomUtils
4   import DOM.Level2.Dom
5   import DOM.Level2.Html2
6   import DOM.Level2.HTMLElement
7   import DOM.Level2.HTMLDivElement
8
9   main = getHTMLDocument $ \doc ->
10          documentBody doc $ \body ->
11          mkText doc "Hello World" $ \txt ->
12          mkDiv doc $ \dv ->
13          addChild txt dv $ \_ ->
14          addChild dv body $ id
```

1.9. att. **Programma „Hello World" valodā Haskell**

1.1.2.3. Racket

Racket (agrāk saukt par PLT Scheme) ir vairāku-paradigmu programmēšanas valoda, kas pieder Lisp/Scheme ģimenei. Racket ir cēls Lisp pēcnācējs un tas ir slavena ar savu eleganci un spēku. Viens no Racket izstrādēs mērķiem ir kalpot par platformu valodas radīšanai, dizainēšanai un īstenošanai. Valoda tiek izmantota dažādos kontekstos, piemēram, skriptu ģenerēšanai, vispārējas nozīmes programmēšanai, izglītībai un pētniecībai. Platforma nodrošina Racket valodas īstenošanu (ieskaitot sarežģītu izpildes laika sistēmu, dažādas bibliotēkas, JIT kompilatoru, u.c.), to kopā ar izstrādes vidi sauc par DrRacket (agrāk saukt

par DRScheme). Kodola valoda ir pazīstama ar savu plašo makrosistēmu, kas ļauj izveidot domēnspecifiskas valodas, valodas konstrukcijas, piemēram, klases, moduļus un atsevišķus dialektus Racket valodai ar atšķirīgu semantiku. [ava]

Iezīmes, kas atšķir Racket no citām valodām Lisp saimē ir integrēta iespēja paplašināt valodu. Racket paplašināmības funkcijas ir iebūvēta moduļu sistēma, kas ļauj kontrolēt kontekstjutīgo un moduļa līmeņa sintaksi. Piemēram, #%modle-begin forma ļauj patvaļīgu statisko analīzi par visu moduli. [ava]

Valodas galvenie rīki ir Racket kodola kompilators, interpretators un izpildes laika sistēma, DRRacket programmēšanas vide un RACO komandrindas rīks, lai izpildītu Racket komandas, kas ielādē paketes un bibliotēkas, u.c. [aza]

```
1  #lang web-server/insta
2  (define (start request)
3    (response/xexpr
4      (body (h1 "Hello World")))))
```

1.10. att. **Programma „Hello World" valodā Racket**

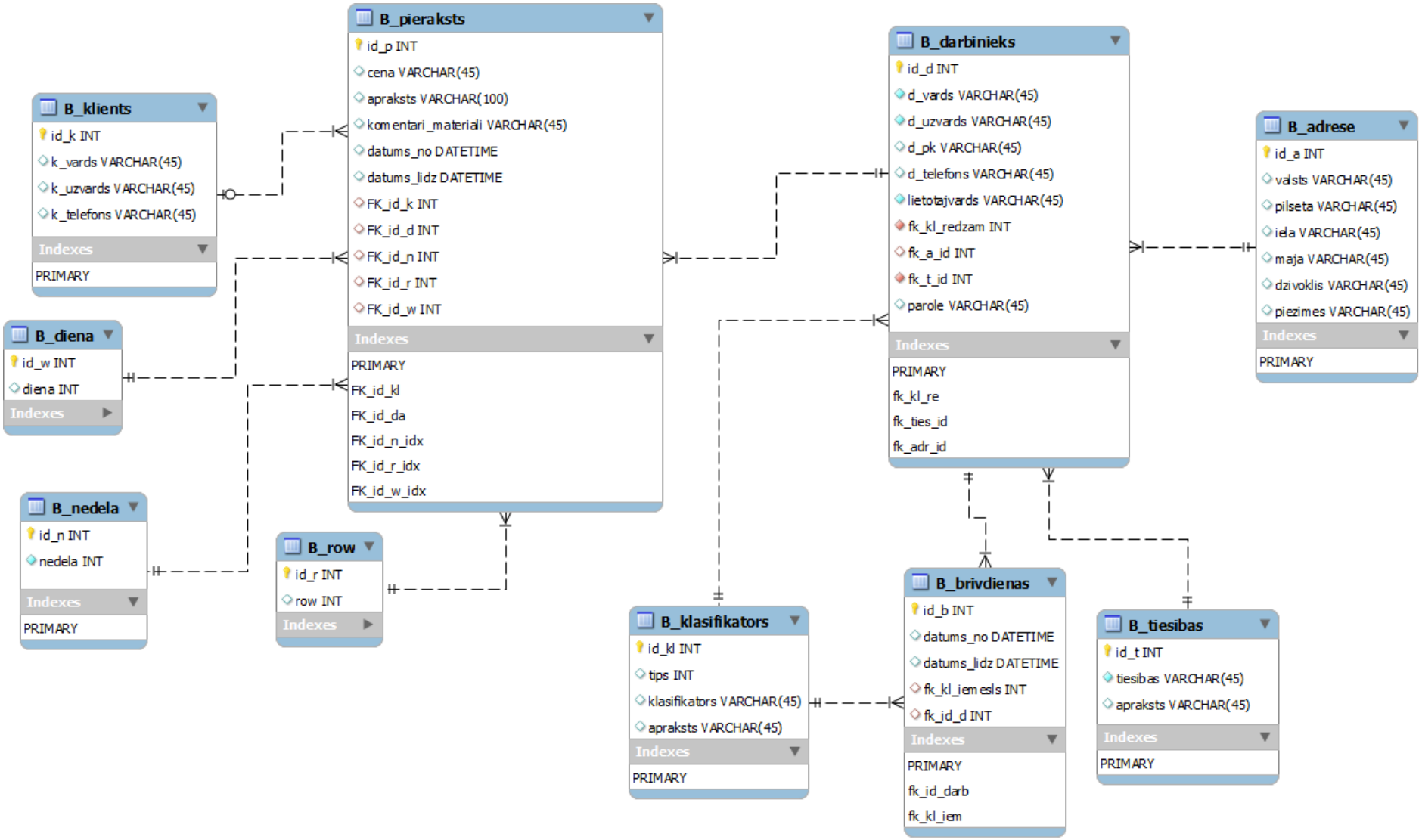
2. UZDEVUMA APRAKSTS

Ik uz soļa, Latvijas pilsētu centros, ir manāms, kāds neliels apkalpošanas uzņēmums, piemēram, skaistumkopšanas salons. Nereti salonos klientu pieraksts notiek rūtiņu blociņos, labākajā gadījumā *Microsoft Office* piedāvātajās izklājlapās. Šāds klientu pieraksts ir ne tikai nepārskatāms, bet arī lietotājam nedraudzīgs.

Izvirzīju uzdevumu izveidot „Klientu pierakstu sistēmu", kura būtu lietotājam draudzīga, vienkārša, pārskatāma un intuitīva. Uzdevuma pamatideja ir izstrādāt sistēmu izmantojot vienu datubāzi un maksimāli līdzīgas funkcionālās iespējas, kā sistēmai veidotai *php*, tā arī *Racket.* Sistēmu jāvar atvērt izmantojot interneta pārlūkprogrammu. Realizētajai sistēmai jābūt lietotājam draudzīgai, intuitīvai, ērti lietojamai un pārskatāmai jo tā paredzēta cilvēkiem bez speciālām iemaņām datu apstrādē un specializētās programmatūrās. Programmu jāveido pārskatāmi, nedēļas skatā, kur katram darbiniekam ir savu klientu pieraksta, pārskata iespēja un laika kontrole.

2.1. Datu modelis

ER-modelis ir grafisks līdzeklis datu bāzes struktūras attēlošanai. Tas sastāv no divu veidu objektiem – entītijām un relācijām. Entītijas tiek apzīmētas ar taisnstūriem ar noapaļotiem stūriem. Katra entītija reprezentē vienu tabulu. Entītijām ir atribūti (lauki tabulās). Relācijas ir tabulu saišu grafisks attēlojums. Relāciju attēlo, kā taisni ar specifiskiem galiem (skatīties modelī), kas apzīmē saites veidu. Saišu veidi ir viens pret viens, viens pret n (daudz). [caa] Kardinalitātes norāda vai nu relācijas, vai asociācijas veidu kā viens reālās dzīves objekts ir saistīts ar citu reālās dzīves objektu. Šīs saistības var būt: viens pret vienu, viens pret daudziem, daudzi pret daudziem. [daa]



2.1. att. **Entītiju relāciju modelis**

Lai realizētu uzdevumu, tika izveidots plašs entītiju relāciju (ER diagramma) diagramma. ER diagramma ir datu bāze modelēšanas metode, kura tiek izmantota lai izveidotu datu bāzes shēmu vai datu modeli, un izprastu veidojamās datu bāzes uzbūvi un pamatprincipus. Entītiju relāciju diagramma ļauj datus uztvert kā objektus un izprast to savstarpējo saistību. Savam uzdevumam ER diagrammu veidoju izmantojot MySQL Workbench datu modelēšanas rīku. MySQL Workbench ir bezmaksas rīks, kurš ir domāts tikai MySQL datubāzei. Ar šo rīku var izveidot datubāzes modeļus, izpildīt SQL pieprasījumus un veikt MySQL datubāzes administrēšanu. [baa]

Datu bāzes modeli veidoju pēc zvaigznes shēmas. Izveidotā diagramma sastāv no desmit savstarpēji savienotām entītijām, kuras savā starpā saista desmit kardinalitātes. Kadinalitātes attēlo kuras entītijas ir savā starpā saistītas, izveidotajā ER diagrammā pielietotās kardinalitātes ir daudz pret vienu.

Entītija B_pieraksts izveidotajā modelī ir vissvarīgākā, tā satur informāciju un ārējās atslēgas, kas raksturo vienu pierakstu. Entītija sastāv no vienpadsmit atribūtiem: *id_p, cena, apraksts, komentāri_materiali, datums_no, datums_lidz, fk_id_k, fk_id_d, fk_id_n, fk_id_r, fk_id_w*. Entītijā izmantotie datu tipi ir: number, varchar un datetime. *Id_p* ir primārā atslēga, tā identificē konkrētu pierakstu, atslēga ir izveidota tā, lai kautru reizi ievietojot datus tiek izveidots jauns unikāls identifikators, tādējādi nav nepieciešams ievietojot datus ģenerēt unikālu identifikatoru. Atribūts *cena* ir varchar tipa un tam ir atļauta vērtība *null*, atribūtā tiek glabāta vērtība no ievietotā pieraksta un apraksta pakalpojuma cenu. Atribūts *apraksts* ir identisks atribūtam *cena*, tad ir varchar tipa un atļauj saturēt *nulles* vērtību, šajā laukā tiek saglabāta vērtība no ievada un tā apraksta pakalpojumu. Atribūts *komentari_materiali* ir līdzīgs iepriekš aprakstītajam atribūtam, tad ir varchar tipa un atļauj saturēt *nulles* vērtības, lauka vērtība tiek paņemta no ievades un apraksta papildus nepieciešamos materiālus vai papildus komentārus. Atribūts *datums_no* un *datums_lidz* ir datetime tipa, šie ir *nenulles* datu lauki un tie tiek glabāti formātā yyyy-mm-dd hh24:mi:ss. *Datums_no* atspoguļo pieraksta sākuma datumu un laiku, turpretī *datums_lidz* satur pieraksta beigu datumu. Entītijai ir piecas ārējās atslēgas, visas atslēgas ir number tipa un tām jāsatur nenulles vērtība. *Fk_id_k* ir ārējā atslēga uz entītijas *b_klints* atribūtu *id_k,* tā norāda, kurš klients ir piesaistāms attiecīgajam ierakstam. *Fk_id_d* ir ārējā atslēga uz entītiju *b_darbinieks*, tā norāda, kurš darbinieks ir piesaistīts attiecīgajam pierakstam. *Fk_id_n* ir ārējā atslēga uz entītijas *b_nedēļa* atribūtu *id_n,* tā norāda nedēļu kurā ir veikts pieraksts. Atribūts *nedēļa* ir nepieciešams, jo kalendārajā skatā tiek attēlota nedēļa un pēc *fk_id_w* noteikta nedēļas diena, tādējādi identificējot un attēlojot, aizņemtību un konkrēto pierakstu kalendārā nedēļas skatā. *Fk_id_w* ir ārējā atslēga uz entītiju *b_diena*, tā identificē pieraksta dienu un tiek izmantota kalendārajā attēlojumā, papildus

izmantojot *fk_id_r*, ārējo atslēgu uz entītiju *b_row*, lai zinātu, kuras rūtiņas tabulā attiecīgajā dienā ir jāiekrāso, kā aizņemti laiki.

Entītija *B_darbinieks* glabā informāciju par darbiniekiem, tā satur desmit atribūtus: *Id_d, d_vards, d_uzvards, d_pk, d_telefons, lietotajvards, parole, fk_kl_redzamiba, fk_a_id, fk_t_id*. *Id_d* ir primārā atslēga, tā identificē konkrētu darbinieku, atslēga ir izveidota tā, lai kautru reizi ievietojot datus tiek izveidots jauns unikāls identifikators, tādējādi nav nepieciešams ievietojot datus ģenerēt unikālu identifikatoru. Atribūti *d_vards* un *d_uzvards* ir varchar tipa datu lauki, kuri nevar saturēt null vērtību. Atribūts *d_vards* ir darbinieka vārds, šis lauks tiek izmantots, veidojot pierakstu attēlojumu tīmekļa lietotnē. Atribūts *d_uzvards* satur darbinieka uzvārdu, vērtība laikā var aizņemt 45 simbolus ieskaitot atstarpes. Atribūtam *d_pk* ir varchar datu tips un šis lauks var saturēt null vērtības, atribūts satur darbinieka personas kodu. Atribūts *d_telefons* ir varchar lauks, kas var saturēt null vērību, lauks satur darbinieka telefona numuru. Laukam ir datu tips varchar, jo telefona numurs var saturēt simbolus, kā, piemēram, norādot valsts kodu '+371'. Atribūts *lietotajvards* ir varchar tipa lauks, kurš nevar saturēt null vērtību, šī lauka vērtība tiks izmantots lai konkrētais darbinieks varētu pieslēgties tīmekļa vietnei. Atribūtam *parole* ir varchar datu tips, kurš var saturēt null vērtības, šī lauka vērtība tik izmantota lai konkrētais darbinieks varētu pieslēgties tīmekļa vietnei. Entītija satur trīs ārējās atslēgas, visas atslēgas ir number tipa un tām jāsatur nenulles vērtība. Atribūts *fk_kl_redzamība* ir ārējā atslēga uz entītiju *b_klasifikators*, darbiniekam var norādīt redzamību, attiecīgi ja izvēlēta vērtība 1, tad darbinieks būs redzams, ja izvēlēta vērtība 0, tad darbinieks nebūs redzams. Atribūts *fk_a_id* ir ārējā atslēga uz entītiju *b_adrese*, vienai adresei var būt piesaistīti vairāki darbinieki, adrese norāda kurā vietā konkrētas darbinieks strādā. Atribūts *fk_t_id* ir ārējā atslēga uz entītiju *b_tiesības*, ir izdalītas divu veidu tiesības, ja izvēlēta vērtība A, tad konkrētais darbinieks var rediģēt, pievienot un dzēst visu informāciju par pierakstiem un darbiniekiem, ja izvēlēta vērtība B, tad darbinieks var rediģēt savu informāciju un savus pierakstus.

Entītija *B_klients* glabā informāciju par klientiem, tā satur četrus atribūtus: *Id_k, k_vards, k_uzvards, k_telefons*. *Id_k* ir primārā atslēga, tā identificē konkrētu klientu, atslēga ir izveidota tā, lai kautru reizi ievietojot datus tiek izveidots jauns unikāls identifikators, tādējādi nav nepieciešams ievietojot datus ģenerēt unikālu identifikatoru. Atribūts *k_uzvards* ir varchar tipa lauks, kas var saturēt nulles vērtību, šajā laukā tiek glabāta informācija, par klienta vārdu, informāciju paredzēts attēlot tīmekļa vietnē. Atribūtam *k_uzvards* datu tips ir varchar un tas var saturēt nulles vērtības, laukā tiek glabāts klienta uzvārds un informāciju paredzēts attēlot tīmekļa vietnē. Atribūtiem *k_vards* un *k_uzvards* ir atļautas null vērtības, jo ir paredzēts ka klients nevēlēsies sevi identificēt. Atribūts *k_telefons* ir varchar tipa lauks, kas

arī var saturēt nulles vērtības. Laukam ir tips varchar, jo telefona numurs var saturēt simbolus, kā, piemēram, norādot valsts kodu '+371'.

Entītija *B_nedela*, glabā visu nedēļu numurus vienā tabulā, un tā satur atribūtus *id_n, nedela*. Tabula satur nemainīgas piecdesmit piecas vērtības, attiecīgi gadā ir 52 nedēļas un viena vai divas dienas garajā gadā. Atribūts *id_n* ir number datu tipa un tā ir primārā atslēga, kura nevar saturēt null vērtības, tas identificē katru gada nedēļu un tā ir ārējā atslēga entītijai *b_pieraksts*. Atribūtam *nedela* ir number datu tips un tas nevar saturēt null vērtības.

Entītija *b_diena* satur nedēļas dienu numurus, dati šajā tabulā datu bāzē ir statiski. Entītijā ir divi atribūti: *Id_w, diena*. Atribūts *id_w* ir number datu tipa un tas nevar saturēt nulles vērtību. *Id_w* ir ārējā atslēga entītijai *b_pieraksts*. Atribūts *diena* ir number datu tipa un satur visu nedēļas dienu numurus: 1, 2, 3, 4, 5, 6, 7.

Entītija *B_row* satur rindu numurus, tabulā dati ir statiski, un šobrīd tiek glabātas 35 vērtības. Atribūts *id_r* ir number datu tipa un tas nevar saturēt nulles vērtības, atribūts ir ārējā atslēga tabulai *b_pieraksts*. Atribūtam *row* arī datu tips ir number, katrs ieraksts ir paredzēts kalendārajam skatam vienai pusstundai.

Entītija *B_adrese* glabā informāciju par darba atrašanās vietu, un tā satur sešus atribūtus: *id_a, valsts, pilsēta, iela, māja, dzivoklis, piezimes*. Atribūts *id_a* ir primārā atslēga entītijai *b_adrese*, tā datu tips ir number un tas nevar saturēt nulles vērtības. *Id_a* ir ārējā atslēga tabulai *b_darbinieks*, kur vairāki darbinieki var būt piesaistīti vienai adresei. Atribūtam *valsts* datu tips ir varchar un tam ir atļauts saturēt nulles vērtības, šajā atribūtā ir paredzēts glabāt valsts nosaukumu, piemēram, Latvija. Atribūtiem *pilsēta* un *iela* datu tips ir varchar un tie var saturēt informāciju līdz 45 simboliem, ieskaitot atstarpes. Atribūts *pilseta* glabā pilsētas nosaukumu, piemēram, Rīga, attiecīgi *iela* glabā ielas nosaukumu, piemēra, Raiņa bulvāris. Atribūtiem *maja* un *dzivoklis* arī datu tips ir varchar un tie var saturēt nulles vērtības. Atribūtā *maja* tiek glabāts mājas numurs vai nosaukums, attiecīgi *dzivoklis* saturēs dzīvokļa numuru. Datu tips ir norādīts varchar, jo ir paredzēts, ka dzīvokļa vai mājas numurs var būt nosaukums vai var saturēt simbolus.

Entītija *B_brivdienas* sastāv no pieciem atribūtiem un tā glabā informāciju par konkrēta darbinieka brīvdienām. Entītijā izmantotie datu tipi ir: number un datetime. *Id_b* atribūts ir ar datu tipu number, tā ir entītijas primārā atslēga un nevar saturēt nulles vērtības. Atribūts *id_b* ir ārējā atslēga entītijai *b_pieraksts*. Atribūtiem *datums_no* un *datums_lidz* datu tips ir datetime, šie ir *nenulles* datu lauki un tie tiek glabāti formātā yyyy-mm-dd hh24:mi:ss. Datums_no glabā informāciju par to no kura datuma un laika konkrētajam darbiniekam būs brīvdienas turpretī *datums_lidz* norāda brīvdienu beigu datumu un laiku. Atribūts *fk_kl_iemesls* ir ārējā atslēga uz entītiju *b_klasifikators*, datu tips ir number un lauks nevar

saturēt nulles vērtības. Šajā atribūtā tiek glabāts iemesls par brīvdienām (piemēram, atvaļinājums, brīvdienas, slimības lapa, u.c.). Atribūts *fk_id_d* ir number tipa un nevar saturēt nulles vērtības. *Fk_id_d* glabā darbinieka id no entītijas *b_darbinieks*, šādā veidā tiek norādītas darbinieks brīvās dienas.

Entītija *B_klasifikators* glabā klasifikatoru vērtības, tā sastāv no četriem atribūtiem: *Id_kl, tips, klasifikators, apraksts*. Entītijas *b_klasifikators* primārā atslēga *id_kl* ir ārējā atslēga tabulām *b_brivdienas, b_darbinieks.* Atribūta *id_kl* datu tips ir number un tas nevar saturēt nulles vērtības. Atribūtam *tips* ir number datu tips un tam ir atļauts saturēt nulles vērtības. *Tips* norāda kurai entītijai šis klasifikators ir paredzēts, piemēram, visi ieraksti ar tipa vērtību 1 ir attiecināmi uz entītiju *b_brīvdienas*. Atribūts *klasifikators* satur attēlojamās vērtības klasifikatorā un tā datu tips ir varchar ar pieļaujamo simbolu skaitu 45, ieskaitot atstarpes. Atribūts *apraksts* ir ar datu tipu varchar un ir paredzēts katra ieraksta aprakstam.

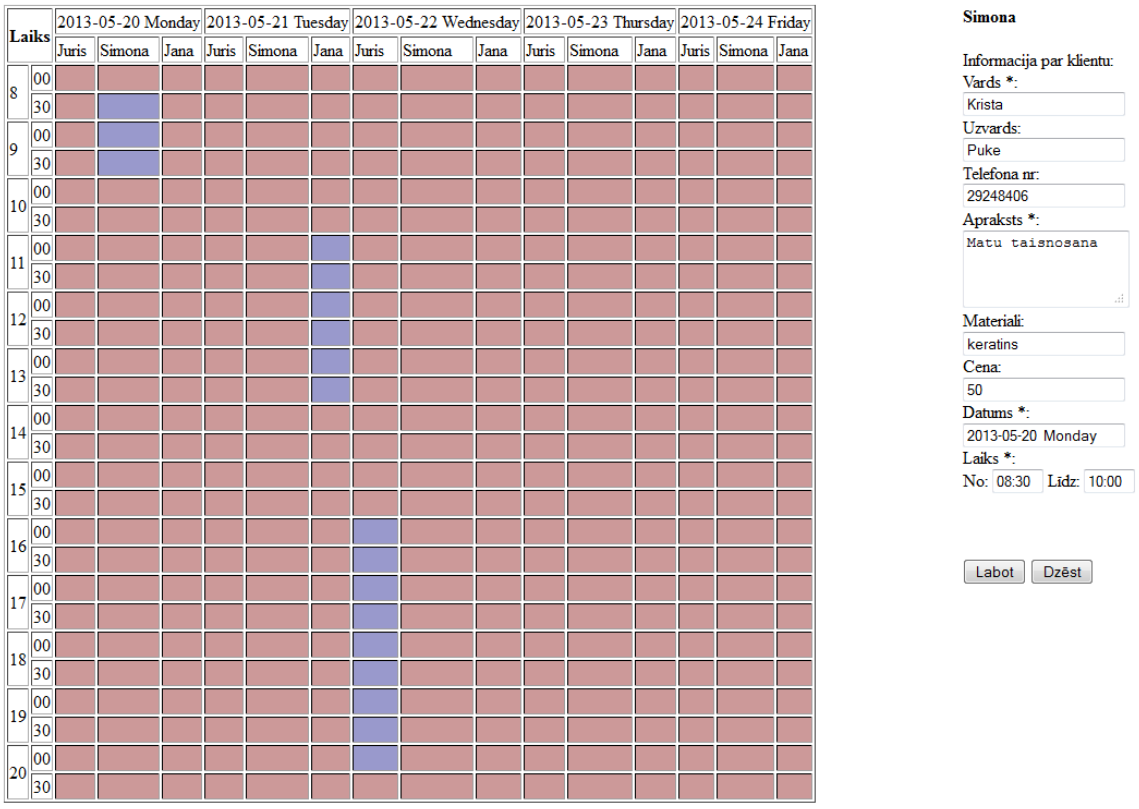
Entītija *B_tiesibas* glabā informāciju par darbinieku tiesībām, tā satur trīs atribūtus: *id_t, tiesibas, apraksts. Id_t* atribūts ir entītijas primārā atslēga, tā datu tips ir number un tas nevar saturēt nulles vērtības. Atribūtiem *tiesibas* un apraksts datu tips ir varchar. Atribūta *tiesibas* vērtības apzīmēs, konkrētas tiesības, kuras plašāk tiek aprakstītas atribūtā *apraksts*, piemēram, *tiesibas* vērtība ir „A" un *apraksta* vērtība ir „var visu rediģēt, dzēst un pievienot" .

3. REALIZĀCIJA PHP

PHP (Hypertext Preprocessor) ir atvērtā koda servera puses skriptu valoda, kas pamatā domāta dinamisku Interneta lapu veidošanai. PHP koda apstrāde notiek servera pusē. Tas ir, kad lietotājs atver Interneta lapu, tiek nosūtīts pieprasījums serverim, serveris apstrādā pieprasījumu- sameklē vajadzīgo HTML dokumentu, izpilda PHP kodu un nosūta rezultātu lietotāja pārlūkprogrammai. Tā kā PHP ir servera puses valoda, tad vajadzīgs attiecīgs tīmekļa serveris, uz kā skripti varētu izpildīties. Izplatītākais savienojums ir PHP un Apache tīmekļa serveris. [aap]

Lai darbinātu php es izvēlējos Apache tīmekļa serveri. Lai pilnvērtīgāk īstenotu sava uzdevuma ieceres papildus php tiek izmantots arī html, javacsript un ajax. Html galvenokārt uzdevumā tiek izmantots lai izveidotu kalendāram līdzīgu pierakstu attēlojumu, formu ar kuras palīdzību ir iespējams attēlot pieraksta informāciju, labot vai dzēst to. Savukārt lai labotu dzēstu datus papildus html tiek izmantots javascripts un ajax. Ajax izmanto lai attēlotu esošo pierakstu informāciju no datubāzes atbilstošos lauciņos html formā. Javascripts galvenokārt uzdevuma realizācijā tiek izmantots lai attēlotu paziņojumus un izsauktu ajax funkcijas, padarītu redzamu formu un aizpildītu ar vērtībām formas laukus.

Uzdevuma realizācijai tiek izmantotas vairākas funkcijas. Viena no galvenajām funkcijām ir divdimensionāla masīva izveidošana un attēlošana. Lai izveidotu divdimensionālo masīvu tiek izmantotas papildus funkcijas: konkrētas nedēļas katras dienas datuma iegūšana, darbinieku kopskaita, vārdu un identifikācijas vērtības iegūšana, stundu un minūšu piekārtošana attiecīgajām rindām un kolonām, katras šūnas id un klases norādīšana. Divdimensionālais masīvs sastāv no vērtībām: datums un laiks, darbinieka vārds, darbinieka id, stundas un minūtes, šūnas id un klases.



3.1. att. **Ekrānforma no realizācijas PHP**

Svarīga ir arī jauna pieraksta izveidošanas, jau esoša ieraksta labošanas un dzēšana funkcija. Jauna pieraksta ievadīšana, jau esoša ieraksta labošanas un dzēšana forma tiek izsaukta ar uzklikšķināšanu uz šūnas, kas attiecīgi izsauc javascript funkciju, kura saņem vairākas vērtības no šūnas un izsauc tālāk ajax funkciju, kura šīs vērtības nodod citam php failam. Šajā failā vērtības tiek saņemtas ar $_post un tālāk apstrādātas, attiecīgi ja šūna uz kuras tika uzklikšķināts ir brīvs laiks, tad tiek izvadīta pieraksta forma, kuru korekti aizpildot un klikšķinot uz „saglabāt" tiek izveidots jauns pieraksts. Ja tiek uzklikšķināts uz šūnas, kuras laiks ir atzīmēts kā aizņemts, tiek izvadīta forma ar jau eksistējošo informāciju par pierakstu. Šo informāciju var labot, mainot lauku vērtības un izvēloties pogu „labot" vai arī dzēst pieraksta informāciju izvēloties pogu „dzēst".

Ja tiek izvēlēts dzēst pierakstu, tad ar padoto pieraksta identifikatoru un vienu vaicājumu ieraksts no datu bāzes tiek izdzēsts. Izvēloties labot pieraksta informāciju tiek izsaukta funkcionalitāte līdzīga ievadīšanai, tikai labošanas gadījumā ir nepieciešamas papildus vērtības un pārbaudes.

Uzdevums php programmēšanas valodā ir realizēts izmantojot piecus failus, no kuriem četri ir php faili un viens javascript. Koda aptuvenais kopgarums, ieskaitot komentārus ir 710 rindiņas. Izstrādātais kods pārskatāmības nolūkos netika rakstīts vienā failā, bet izveidoti pieci atsevišķi.

4. REALIZĀCIJA RACKET

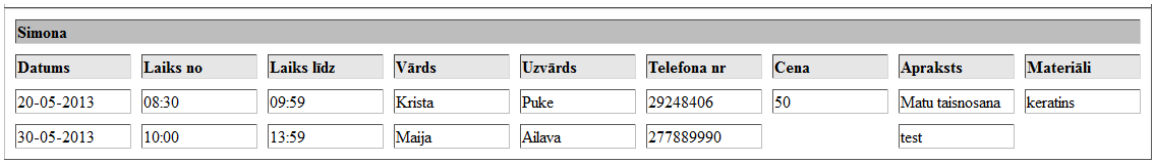
Par Racket programmēšanas valodu 7. Jūnijā 2010.gadā tika pārsaukta valoda PLT Scheme 5.1 versija. *Drracket* redaktors nodrošina avota izcelšana sintaksi, izpildes laika kļūdu attēlošanu, iekavu saskaņošanu. *DrRacket* ir pieejams Windows (95 un uz augšu), Mac OS X, Unix. [ava]

Realizācijai Racket valodā ir izveidots viens fails, kurā ir visa programma. Lai palaistu Racket failu tas jāatver *Drracket* programmēšanas vidē un jāklikšķina „Run".

Tā kā uzdevuma realizēšanā viena no galvenajām lietām ir piekļuve datubāzei, tad sākot darbu *DrRacket*, izveidoju, definēju savienojumu ar MySql datubāzi. Lai izveidotu tīmekļa vietni Racket nodefinēju kā #lang web-server/insta. Tīmekļa vietne tiek definēta ar šādu komandu (define (start request) (response/xexpr XX)), kur simbolu XX vietā tiek definēta html struktūra. Html struktūra uzdevumā tiek definēta vairākās daļās, tad beigās savienota vienā sarakstā. Izveidotais saraksts ar html struktūru tiek padots jau iepriekš minētajai tīmekļa vietnes definīcijai.

Katra darbinieka aizņemtības grafika attēlošanas izveidei tiek izveidotas un izmantotas piecas definīcijas. Viena no svarīgākajām definīcijām izveido MySql vaicājumu, kurā tiek paņemts darbinieks un tas tiek savienots ar visiem saviem pierakstiem, un katram pierakstam tiek piekārtots attiecīgais klients. MySql vaicājuma izpildes rezultātā tiek iegūts vektors, kuru ar saraksta definēšanu *vector->list* pārvērš par sarakstu no sarakstiem. Tālāk šis saraksts no sarakstiem tiek pārveidots par *html* tabulas struktūru, izveidojot definētās funkcijas : *render-cell* un *render-list.* Attiecīgā darbinieka aizņemtības saraksts no sarakstiem tiek padots pa vienam sarakstam funkcijai *render-list,* izmantojot *,@(map render-list XX),* kur XX ir saraksta no sarakstiem definējums. Tālāk *render-list* funkcijā tiek izsaukta funkcija *render-cell,* kurai tiek padots katrs saraksta elements izmantojot *,@(map render-cell XX),* kur XX ir *render-list* padotā saraksta katrs elements. Nodefinētā funkcija *render-cell* piekārto katram saraksta elementam html kolonas simbolus, piemēram, padots elements „Krista" , pēc funkcijas izpildes elements būs saraksts un izskatīsies šādi : `(td „Krista") , kas apzīmē vienu kolonu. Nodefinētā funkcija *render-list* piekārto katram saņemtajam sarakstam html rindas simbolus, piemēram, ja funkcijai *render-list* padod sarakstu („Krista" „Puķe" „Latvijas Universitāte" ), tad pēc funkcijas izpildes saraksts izskatās šādi: `(tr (td „Krista") (td „Puķe") (td „Latvijas Universitāte")), kas apzīmē vienu html tabulas rindu. Kad sarakts ar viena

darbinieka rindām un kolonām tiek izveidots, tam tiek piekārtots vēl tabulas definējums, darbinieka vārds un kolonu nosaukumi.



Simona								
Datums	**Laiks no**	**Laiks līdz**	**Vārds**	**Uzvārds**	**Telefona nr**	**Cena**	**Apraksts**	**Materiāli**
20-05-2013	08:30	09:59	Krista	Puke	29248406	50	Matu taisnosana	keratins
30-05-2013	10:00	13:59	Maija	Ailava	277889990		test	

4.1. att. **Ekrānforma no realizācijas Racket**

Kad darbinieku tabulas izveidotas, tās tālāk tiek apstrādātas funkcijās *list-start*, kas sakārto *html* struktūru no koda sākuma, darbinieku tabulām un līdz *html* beigām. *List-start* funkcija izveido sarakstu ar html kodu, kas sastāv no : *html, head, body* un darbinieku tabulu definēšanas, kas ievietotas *div* elementos. Lai DrRacket programmēšanas vidē klikšķinot uz „run" tīmekļa pārlūkprogrammā tiku attēlota izveidotā html struktūra ir jānodefinē *start request* funkcija, kurā tiek izsaukta *list-start* (skat. 4.2. att) .

```
(define (start request)
(response/xexpr
list-start
  ))
```

4.2. att. **Definētas funkcijas izsaukums**

REZULTĀTI

Izmantojot PHP programmēšanas valodu, kopā ar javascript un ajax ir izveidota dinamiska tīmekļa lietotne, kura tiek savienota ar MySql datubāzi. Tīmekļa vietni veidojot php valodā un izmantojot Xamp tīmekļa serveri, kas sastāv galvenokārt no Apache HTTP servera un MySQL datu bāzes, iespējams vietni atvērt ar pārlūkprogrammu. Tā pat kā valodā php , tā arī Racket ir iespēja izveidot html struktūru, kura abās programmēšanas valodās ir rakstāma bez liekām aizķeršanām, jo tai nav atšķirīgs pieraksts. Racket programmēšanas valodā ir iespēja izveidot tīmekļa vietni izmantojot DrRacket programmēšanas vidē iebūvētu definēto funkciju *(define (start request) (response/xexpr XX))* , kur XX vietā rakstot html struktūru. Lai Racket savienotu ar datu bāzi un izvadītu tīmekļa vietni pārlūkprogrammā, tā pat kā valodai PHP ir nepieciešams serveris, uzdevuma īstenošanai tika izmantots Xamp, kurš satur Apache serveri.

Salīdzinot PHP un Racket, vieglāk veidot ciklus ir php valodā, jo Racket nav iespējams definēt mainīgos. Koda pārskatāmība ir atkarīga no programmētāja, tomēr veidojot kodu pēc labās programmēšanas prakses Racket kods ir vairāk nepārskatāms, kā PHP. Lai izprastu un iedziļinātos programmēja rakstītajā kodā valodā Racket, ir nepieciešams daudz laika un pacietības, lai spētu izsekot visām definīcijām un atkarībām vienai definētajai funkcijai no citas. Racket ir arī tāda īpašība, kā definējot funkcijas tās obligāti jāraksta secīgi, nevar izsaukt funkciju no citas definētās funkcijas, ja izsauktā atrodas zem izsaucošās. Tas nozīmē, ka, lai izsauktu kādu definēto funkciju , tai ir jābūt iepriekš, failā augstāk, definētai. Turpretim PHP kodu var rakstīt mainot funkcijas vietām, jo funkciju atrašanās vieta nemaina koda izpildījumu. Tomēr ir jāuzmanās kā PHP tā arī Racket valodās, jo html struktūrai ir jābūt secīgai.

Lielākoties Racket programmēšanas valodā tiek strādāts ar sarakstiem, sarakstiem no sarakstiem, toties PHP ar mainīgajiem un dažādām to vērtībām. Lai definētu MySql vaicājumu php vai izmantot mainīgo un Racket saraksta definējumu, tomēr php var izpildīt praktiski jebkādu vaicājumu izmantojot *mysql_query* iebūvēto funkciju, tad nepieciešamās vērtības vai vienu vērtību paņemt un ievietot attiecīgi mainīgajā, turpretī Racket ir jāzina precīzi vai no vaicājuma vēlaties izgūt vienu vērtību, vai visu sarakstu, vai vienu kolonu, vai vienu rindu. Katrai Racket datu bāzes atlasīšanai ir sava nozīme. Ar *query-exec* var veidot ievadi vai dzēšanu no datubāzes, ar *query-rows* var atgriezt vektoru ar vairākām rindām, attiecīgi ar *query-row* var atgriezt tikai vienu rindu, ar *query-list* var atgriezt vienas kolonas visas vai vienu vērtību, ar *query-maybe-row* var atgriezt vienu rindu vai *#f(false),* ja nav

atgriežamu rindu, ar query-value var atgriezt tikai vienu vērtību, attiecīgi ar query-maybe-value atgriež vienu vērtību vai *#f(false),* ja nav atgriežama vērtība.

SECINĀJUMI

Pēc gūtās pieredzes izstrādājot tīmekļa vietni vieglāk rakstīt kodu ir PHP , kā Racket programmēšanas valodā, šim atzinumam ir vairāki aspekti. Lai iegūtu informāciju vai piemērus dažādu uzdevumu risināšanai, programmēšanas valodā PHP, viegli ir izmantot interneta vietnes vai arī, kā manā situācijā, lūgt palīdzību valodas pārzinātājiem un Bakalaura darba vadītājam, lai labāk izprastu php sintaksi. Turpretī programmēšanas valodas Racket rokasgrāmata un palīgmateriāli ir tikai angļu valodā un tie atrodas interneta vietnē http://racket-lang.org/.

Izvēlētās valodas un ietvari ir piemēroti tīmekļa vietņu izstrādei. Abās valodās ir iespēja izveidot savienojumu ar datu bāzi, kā arī ar vaicājumu palīdzību izgūt informāciju un to ievietot, dzēst vai labot. Abās valodās ir iespēja izveidot html struktūru, ar kuras palīdzību var konstruktīvi izvadīt informāciju pārlūkprogrammā un pievienot tai css.

Koda struktūra abās izvēlētajās valodās nav analoģiska, kaut gan uzdevums ir viens. Racket ir svarīga definīciju secība, turpretī PHP valodā funkcijas var rakstīt bez striktas secības. Programmēšanas valodai Racket ir sava programmēšanas vide DrRacket, kur katra kļūda vai neprecizitāte tiek iezīmēta un tiek piedāvāts sekot saitei uz procedūras vai struktūras aprakstu, turpretī rakstot PHP kodu var izvēlēties vienu no ļoti daudzajiem ietvariem, tomēr ietvaros netiek atzīmētas kļūdas, tās tiek atklātas atverot programmu ar pārlūkprogrammā. Par Racket struktūru var piebilst to, ka ne tikai definīciju secībai ir strikta nozīme, bet arī definīciju saturam, kā piemēru varu piebilst *map* funkcionalitāti, tā obligāti jāliek iekavās kopā ar sarakstu vai saraksta definīciju un pirms tā jābūt simboliem ‚@ un šo struktūru nevar rakstīt atsevišķā rindā, tai ir jāseko jau iepriekš rakstītam kodam.

Lai veidotu tīmekļa lietotni gan PHP programmēšanas valodā, gan Racket ir nepieciešama pieredze un iemaņas. Izsakot subjektīvo viedokli ērtāka izstrāde bija valodā PHP, jo bija pieejami pietiekami daudz interneta resursu gan latviešu, gan angļu valodā, kā arī bija viegli atrast zinošas personas pie kā konsultēties par valodas struktūru, savukārt Racket valodā bija grūti orientēties, jo valodai ir pieejams apraksts, dokumentācija tikai angļu valodā un man grūtības sagādāja tieši struktūras uztveršana. Pateicoties konsultācijai ar bakalaura vadītāju struktūru un funkciju definēšanu bija vieglāk saprast.

Tā kā uzdevuma definīcijā ir izveidots plašs entītiju relāciju modelis, tad realizēto uzdevumu var uzlabot ar papildus funkcionalitāti, kā piemēram, lietotāju konfigurēšana, autorizēšanās, atskaišu veidošana, brīvdienu atzīmēšana , u.c. Papildus funkcionalitāti ir

plānots pievienot, lai programmu varētu lietot un piedāvāt kā gatavu „Klientu pierakstu sistēmu”.

IZMANTOTĀ LITERATŪRA UN AVOTI

aao – A.Ulasevičs.(2006) Rīga. Referāts „Valodas C# 2.0 jauno īpašību apskats”

aav – Ģ.Folkmanis.(2006) Rīga. Referāts „Programmēšanas valoda „Python” ”

afa – S.Fedotovs.(2006) Rīga. Referāts „Valoda RUBY”

aka – A.Kirillovs.(2004) Rīga. Referāts „Programmēšanas valoda PHP”

ana – G.Jansona.(2002)Rīga. Referāts „Perl apskats ar piemēriem”

aua – Ervīns.(2005)Rīga. Referāts „Valoda Haskell”

aaa – M.James.(07.01.2013) „The Top Languages of 2012”.
Saite: http://www.i-programmer.info/news/98-languages/5298-the-top-languages-of-2012.html

aab – „TIOBE Programming Community Index Definition”
Saite: http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm

aai – JAVA(1997). „The Java Programming Language”
Saite: http://groups.engin.umd.umich.edu/CIS/course.des/cis400/java/java.html

aaj – WEBOPEDIA ™ „Java”
Saite: http://www.webopedia.com/TERM/J/Java.html

aan – Wikipwdia „C sharp”
Saite: http://lv.wikipedia.org/wiki/C_sharp

aap – G.Vārtiņa.(2009) „PHP vēsture un raksturojums.”
Saite: http://visiem.myartsonline.com/lasitava/lasit.php?mape=Progammeeshanas_valoda_PHP&raksts=PHP_veesture_un_raksturojums

aar – iinuu.lv „PHP: Kā radās PHP? Vēsture”
Saite: http://www.iinuu.lv/lv/it-guru/php-ka-radas-php-vesture

aau – J.Mendez.(2009) „Phyton”
Saite: https://lv.m.wikipedia.org/wiki/Python

aba – K.Apsītis. (2008) „Imperatīvās valodas un to atšķirības”
Saite: http://ante.lv/xwiki/bin/view/TrainingWebProgrammingWorkshop/App_ProgLanguages_ImperativeParadigm

aca – J.Zuters „Datorzinātnes, Algoritms. Programmēšanas valoda”
Saite: http://home.lu.lv/~janiszu/courses/eprg/01eprg.algorithm.pdf

ada – Estudijas.lu.lv kursa materiāls „Programmēšana”

Saite: http://estudijas.lu.lv/pluginfile.php/70470/mod_resource/content/0/EUCIP-3.pdf

aea – „About Ruby”

Saite: http://www.ruby-lang.org/en/about/

aga – @bombtrack (2008) „Ruby On Rails jeb web izstrāde vairs nekož!”

Saite: http://datuve.lv/raksts/1613/

aha – A.Ļevčenkovs, N. Kuņicina.(2006) „Ievads JAVA programmēšanā Industriālajā elektronikā”

Saite: http://www.ltn.lv/~kunicina/java.pdf

aia – ltn.lv.(2007) „Java tehnoloģijas”

Saite: http://www.ltn.lv/~apsitis/java-eim/de/intro_java_mod11_start.html

aja – wikipedia.org „Active Server Pages”

Saite: http://lv.wikipedia.org/wiki/Active_Server_Pages

ala – „Learn about ASP”

Saite: http://www.host-shopper.com/learn-about-asp.html

ama – „Programmēšanas valoda Perl”

Saite: http://www.ailab.lv/KF/1.4.11.3.7.3.htm

aoa – K.Apsītis.(2008) „Funkcionālā programmēšana”

Saite: http://ante.lv/xwiki/bin/view/TrainingWebProgrammingWorkshop/App_ProgLanguages_FunctionalParadigm

apa – A.Gulbis. (2011) „Bezmaksas AutoCad Lisp kodi”

Saite: http://www.celuprojekts.lv/bezmaksas-autocad-lisp-kodi/

ara – „Lisp (programming language)”

Saite: https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Lisp_%28programming_language%29.html

asa – A.Grocevs „Rekursijas grafiskais attēlojums LISP programmās”

Saite: http://smallserver.org/blog/projects/LISP_parser/LISP_article.pdf

ata – „Haskell”

Saite: http://lv.wikipedia.org/wiki/Haskell

ava – „Racket (programming language)”

Saite: http://en.wikipedia.org/wiki/Racket_%28programming_language%29

aza – „Welcome to Racket”

Saite: http://docs.racket-lang.org/guide/intro.html

baa – G.Plivna.(2010) „MySQL Workbench – datu modelēšanas rīks”

Saite: http://datubazes.wordpress.com/tag/datu-modelesana/

caa – @su. (2006) „Datu bāzes projektēšana”

Saite: http://datori.wordpress.com/2006/09/13/datu-bazes-projektesana/

daa – „ER diagrammas”

Saite: http://lv.wikipedia.org/wiki/ER_diagrammas

faa – „Tīmekļa vietne”

Saite: http://lv.wikipedia.org/wiki/T%C4%ABmek%C4%BCa_vietne

gaa – „Tīmekļa izstrāde”

Saite: https://lv.wikipedia.org/wiki/T%C4%ABmek%C4%BCa_izstr%C4%81de

haa – „Web programmēšana”

Saite: http://iti.rtu.lv/vitk/lv/studijas/kursi/dop514-web-programmesana

PIELIKUMI

Pielikums1. Izstrādātais kods PHP programmēšanas valodā

Cal11.php

```
<!DOCTYPE html>
< html>
< body onload="document.refresh();">
< title>Kalendars</title>
<script type='text/javascript' src = 'cal44.js' ></script>
<script type='text/javascript' src = 'myfunc.js' ></script>
<script type='text/javascript' src = 'prototype.js' ></script>
<style type="text/css">
td.NAV {
        background-color: #CC9999; color: black;
}
td.IR {
        background-color: #9999CC; color: black;
}
</style>
```

<?php

```
//$rows = 13;
//$hours = 8;
//$secon1 = 00;
//$secon2 = 30;
$today = date("Y-m-d");
$sysdate = date("Y-m-d H:i:s");

$db = mysql_connect('localhost', 'root', '') or die (mysql_error());
mysql_select_db("mydb", $db);
```

```
$result1 = mysql_query("SELECT * FROM b_darbinieks");
$get_vards = mysql_affected_rows(); //saskaita cik ir darbinieki
$izv_darbin = $get_vards; //izvada darbinieka vārdu tabulā
$datum_span = $get_vards ; //cik colspan datumu
$sk = $get_vards; // lai datumam un laikam
$week_days = 5; //lai darbiniekus reizinatu cik dienas nedelaa
$row = 0;
$cell = 1;
$divdiv= 'divdiv';
$darbn_id = array(); //glab'ajas darbinieku id
$galv = array();
$ska=0;

if(isset($_GET['week']))
{
        $week_number = date('W', strtotime($sysdate))+$_GET['week'];
        $previous_week = $_GET['week']-1;
        $next_week = $_GET['week']+1;
}
else
{
        $week_number = date('W', strtotime($sysdate));
        $previous_week = -1;
        $next_week = 1;
}


echo "<a href='cal11.php?week=$previous_week' ><<<<</a> <br>";
echo "<a href='cal11.php?week=$next_week' >>>>></a> <br>";

$result_week = mysql_query("SELECT id_n FROM B_nedela where nedela =
$week_number"); //paņem no db vērtību
$data_week = mysql_fetch_array($result_week);
$week = $data_week['id_n'];
```

```
$sk_cell = $get_vards; //lai insertotu cell skaitītajs
$fi =2; //for ciklam
$si =2;
$stundas = 14*2;
$stundas_start=8;
$ss=1; //lai stundaam saliktu minutes
$count1 = 2; //ir 2 nevis 0 tape ka pirmie 2 ir laiks //kolonas
$count2 = 3; //ir 3 nevis 0 jo pirmie tris ir diena/vards/darbinieka id //rindas
$sk_hours = 2; //skatītājs lai stundas izvadīti kaa divas vienu rowspan

$datums = 0; //datums insertam
$darbn = 1; //darbinieka vards insertam
$id_darbn = 2; //darbinieka id insertam

for($i=0;$i<=$stundas;$i++){
        if($i<=2){
                $galv[$i][0]='Laiks';
                $galv[$i][1]='Laiks';
        }
        else{
                $galv[$i][0]=$stundas_start;
                $count2 ++; // saskaita rindunu skaitu
        if ($ss%2 ==0){
                $galv[$i][1]='30';
        }
        else{
                $galv[$i][1]='00';
        }
        $ss++;
        if ($i%2==0){
                $stundas_start++;}
        }
}
```

```
for($va = 1;$va<=$get_vards;$va++){
        $day1 = 1; // nedēļas dienas pēc kārtas
        $result_day = mysql_query("SELECT id_w FROM B_diena where diena = $day1");
//paņem no db vērtību
        $data_day = mysql_fetch_array($result_day);
        $day = $data_day['id_w'];
        if($datum_span/$sk==1){//saliek vienu dienu datumu
        for($we = 1; $we<=$week_days; $we++){
                $year = date('Y'); // iegūst gadu
                $date = mktime(0,0,0,date("m"),date("d"),date("Y"));
                $dd = date('Y-m-d l', strtotime($year."W".$week.$day)); //."\n"
                for($fi;$fi<=$sk_cell+1;$fi++){
                $galv[0][$fi]=$dd;
                $count1++; //saskaita kolonas
        }
        $sk_cell +=3;
        $day++;
        }
$sk--;
}

for($we = 1; $we<=$week_days; $we++){
 $result1 = mysql_query("SELECT * FROM b_darbinieks where id_d = $izv_darbin");
$data = mysql_fetch_array($result1);
$vards = $data['d_vards'];
$dar_id = $data['id_d'];
// $darbn_id[$izv_darbin] = $dar_id;
$izv_darbin --;

if ($izv_darbin==0 && $we<=$week_days)($izv_darbin = $get_vards); // saliek visu
darbinieku vārdus reiz dienas
//echo "<td><b>$vards</b></td>";///////////////
        $galv[1][$si]=$vards;
        $galv[2][$si]=$dar_id;
```

```
            $si ++;
            }
}

for($z=3;$z<$count2;$z++){
            for($i=2;$i<$count1;$i++){
            $id = $i*0.01;
            $id +=$z;
            $galv[$z][$i] = $id;

            }
}
echo "Šodienas datums un laiks: $sysdate </br></br>";

?>
< div style="float:left; width:60%; margin-left:5px;">

< table border='1' cellpadding='1'>
<tr>
<!-- izvada vardu laiks-->
<td colspan='2' rowspan='2'><b><?php echo $galv[0][0] ?> </b></td>
<?php
//Izvada datumus
for($i=2;$i<$count1;$i+=$get_vards){
            echo "<td colspan='$get_vards'> ".$galv[0][$i]." </td>";
}

echo "</tr>";

echo"<tr> ";
for($i=2;$i<$count1;$i++) {
            echo "<td > ".$galv[1][$i]."</td>";
}
echo "</tr>";
for($z=3;$z<$count2;$z++){
```

```
echo "<tr>";
for($i=0;$i<1;$i+=2){
if($sk_hours%2==0){
        echo " <td rowspan='2'> ".$galv[$z][$i]." </td>";
}
$sk_hours++;
}
for($i=1;$i<2;$i++){
        echo " <td> ".$galv[$z][$i]." </td>";
}
for($i=2;$i<$count1;$i++) {
//Sameklē visādas vērtības lai varētu klasi norādīt
        $darbinieks = $galv[$id_darbn][$i];
        $datums_k = $galv[$datums][$i];
        $stunda = $galv[$z][0];
        $minute = $galv[$z][1];

        $year_from = substr($datums_k,0,4); //gads
        $month_from = substr($datums_k,5,2); //menesis
        $day_from = substr($datums_k,8,2); //datums

if ($minute == 00 || $minute == 0){
        $minute +=10;
//$stunda -=1;
}
elseif($minute==30){
        $minute+=10;
}
//Uzliek class lai var iekrāsot cellus
$datums_kad = date("Y-m-d H:i:s",
mktime($stunda,$minute,0,$month_from,$day_from,$year_from)); // pieraksta datumslīdz
//apskata vai šajā datumā sim lietotājam ir kaut kad ieplānots
$query = "SELECT * from mydb.b_pieraksts where fK_id_d = $darbinieks and
'$datums_kad' between datums_no and datums_lidz";
$resultt = mysql_query($query);
```

```
$row = mysql_affected_rows();
//ja ir tad cklass uzliek IR ja nav tad uz liek class NAV
if($row>=1){
        $class = 'IR';
}
elseif($row==0){
        $class = 'NAV';
}

//Iavada rowus un izsauc onclick funkciju
?>
<td class="<?php echo $class ?>" onclick="myfunc('<?php echo $galv[$darbn][$i] ?>',
'<?php echo $galv[$datums][$i] ?>', '<?php echo $galv[$z][$i] ?>',
'<?php echo $galv[$z][0] ?>', '<?php echo $galv[$z][1] ?>', '<?php echo
$galv[$id_darbn][$i] ?>', '<?php echo $i?>', '<?php echo $z?>')"
id = "id = <?php echo $galv[$z][$i] ?>"></td>
<?php
}
}
?>
< /table>
< /div>
< div id="left" style="float:right; width:30%">
< /div>
< /div>
< /br>
< /br>



< /body>
< /html>
```

Labot_dzest.php

```
<!DOCTYPE html>
< html>
```

```
< body>
< script type='text/javascript' src = 'myfunc.js' ></script>

<?php

$db = mysql_connect('localhost', 'root', '') or die (mysql_error());
mysql_select_db("mydb", $db);

$d_id = $_POST['ID']; //darbinieka id
$c = $_POST['col']; //kolonas id
$r = $_POST['row']; //rindas id
$pier_id = $_POST['pieraksts']; //pieraksta ID
$name = $_POST['name']; //klienta vards
$srname = $_POST['srname']; //klienta uzvards
$phonenr = $_POST['phonenr']; //telefina nr
$description = $_POST['description']; //apraksts
$materials = $_POST['materials']; //materiali
$price = $_POST['price']; //cena
$date = $_POST['date']; //datums
$date_from = $_POST['date_from'] ; //laiks no
$date_to = $_POST['date_to']; //laiks lidz
$dzest = $_POST['dzest'];

if($dzest=='dzest'){
//echo "dzest";
        $query_dzest = "DELETE FROM mydb.b_pieraksts where id_p = $pier_id";
        $result_dzest = mysql_query($query_dzest);
        echo "<script language=javascript>deleted()</script>";

}
else{
        echo "labot";
        $year_from = substr($date,0,4); //gads
        $month_from = substr($date,5,2); //menesis
        $day_from = substr($date,8,2); //datums

//pārbauda un salabo lietotāja ievadīto laiku TO
if(substr($date_to,1,1)==':'){
        $time_to=substr($date_to,0,1);
        $seconds_to = substr($date_to,2);
}
else{
        $time_to = substr($date_to,0,2);
        $seconds_to = substr($date_to,3);
}
//lai neparklajas 11:00 uz liek par 10:59 un 12:30 par 12:29
 if($seconds_to == 00 || $seconds_to == 0){
        $seconds_to = 59;
        $time_to -=1;
}
elseif($seconds_to == 30){
        $seconds_to = 29;
```

```
}

//Datums līdz
$to = date("Y-m-d H:i:s",
mktime($time_to,$seconds_to,0,$month_from,$day_from,$year_from)); // pieraksta
datumslīdz

$query = "SELECT * from mydb.B_klients where k_telefons = '".$phonenr."' and
lower(k_vards)= lower('".$name."')";
$result = mysql_query($query);
$row = mysql_affected_rows();

//Ja klients ar šādu vārdu un nelefona nr, neeksistē“ tad ieinserto datubāzē“
if ($row==0){
        $sql="INSERT INTO mydb.B_klients (k_vards, k_uzvards, k_telefons) VALUES
        ('$name','$srname','$phonenr')";
        $result2 = mysql_query($sql);
}
//paņem klianta id
$query1 = "SELECT id_k from mydb.B_klients where k_telefons = '".$phonenr."' and
lower(k_vards)= lower('".$name."')";
$result1 = mysql_query($query1);
$data_kli = mysql_fetch_array($result1);
$kl_id = $data_kli['id_k'];

$query_update = "UPDATE mydb.b_pieraksts
set
cena = '$price'
, apraksts = '$description'
,komentari_materiali = '$materials'
,datums_lidz = '$to'
,fk_id_k = $kl_id
where id_p = $pier_id
";
$result_update = mysql_query($query_update);
echo "<script language=javascript>ins()</script>";
echo "<script>window.history.back()</script>";
}
?>

< /body>
< /html>
```

Insert.php

```
<!DOCTYPE html>
< html>
< body>
< script type='text/javascript' src = 'myfunc.js' ></script>
```

```
<?php

$d_id = $_POST['ID']; //darbinieka id
$c = $_POST['col']; //kolonas id
$r = $_POST['row']; //rindas id
$name = $_POST['name']; //klienta vards
$srname = $_POST['srname']; //klienta uzvards
$phonenr = $_POST['phonenr']; //telefina nr
$description = $_POST['description']; //apraksts
$materials = $_POST['materials']; //materiali
$price = $_POST['price']; //cena
$date = $_POST['date']; //datums
$date_from = $_POST['date_from'] ; //laiks no
$date_to = $_POST['date_to']; //laiks lidz

//sakonektÄ“jÄ□s ar db
$db = mysql_connect('localhost', 'root', '') or die (mysql_error());
mysql_select_db("mydb", $db);

//parbauda vai visas obligatas vertibas ir aizpilditas
if($name == NULL || $phonenr ==NULL || $date == NULL || $date_from ==NULL ||
$date_to ==NULL){
        if($name == NULL) {$is1 = ' VÄ□rds ';
        }else $is1 ='';
        if($phonenr ==NULL) {$is2 = ' Telefona nr ';
        }else $is2 ='';
        if($date == NULL) {$is3 =' Datums ' ;
        }else $is3='';
        if($date_from ==NULL) {$is4= ' Laiks no ' ;
        }else $is4 ='';
        if($date_to ==NULL) {$is5= ' Laiks lÄ«dz';
        }else $is5 = '';
// izmet alertu ka nav aizpildīto obligatie lauki
echo "<script language=javascript>isnot('$is1','$is2','$is3','$is4','$is5')</script>";
```

```
//ja kāds lauks nav aizpildīts atgroež atpakaļ uz sākumu
echo "<script>window.history.back()</script>";
}

else{

$query = "SELECT * from mydb.B_klients where k_telefons = '".$phonenr."' and
lower(k_vards)= lower('".$name."')";
$result = mysql_query($query);
$row = mysql_affected_rows();

//Ja klients ar šā vārdu un nelefona nr, neeksistē“ tad ieinserto datu bāzē“
if ($row==0){
        $sql="INSERT INTO mydb.B_klients (k_vards, k_uzvards, k_telefons) VALUES
      ('$name','$srname','$phonenr')";
$result2 = mysql_query($sql);
}
//paņem klianta id
$query1 = "SELECT id_k from mydb.B_klients where k_telefons = '".$phonenr."' and
lower(k_vards)= lower('".$name."')";
$result1 = mysql_query($query1);
$data_kli = mysql_fetch_array($result1);
$kl_id = $data_kli['id_k']; //klienta id

$query3 = "SELECT id_r from mydb.b_row where row = $r";
$result3 = mysql_query($query3);
$rowa = mysql_fetch_array($result3);
$rw = $rowa['id_r']; //row id

$week_number = date('W', strtotime($date)); //nedeÄ¼as numurs
$query4 = "SELECT id_n from mydb.b_nedela where nedela = $week_number";
$result4 = mysql_query($query4);
$weeka = mysql_fetch_array($result4);
$wn = $weeka['id_n']; // week id
```

```
$wek_d = date("N", strtotime($date));
$query5 = "SELECT id_w from mydb.b_diena where diena = $wek_d";
$result5 = mysql_query($query5);
$dien = mysql_fetch_array($result5);
$week_d = $dien['id_w']; //week day id

$year_from = substr($date,0,4); //gads
$month_from = substr($date,5,2); //menesis
$day_from = substr($date,8,2); //datums


//pārbauda un salabo laiku FROM
if(substr($date_from,1,1)==':'){
        $time_from=substr($date_from,0,1);
//echo " stundas: $time_from </br>";
        $seconds_from = substr($date_from,2);
//echo "minutes: $seconds_from </br>";
}
else{
        $time_from = substr($date_from,0,2);
//echo " stundas: $time_from </br>";
        $seconds_from = substr($date_from,3);
//echo "minutes: $seconds_from </br>";
}


//pārbauda un salabo lietotāja ievadīto laiku To
if(substr($date_to,1,1)==':'){
        $time_to=substr($date_to,0,1);
        $seconds_to = substr($date_to,2);
}
else{
        $time_to = substr($date_to,0,2);
        $seconds_to = substr($date_to,3);
}
```

```
//lai neparklajas 11:00 uz liek par 10:59 un 12:30 par 12:29
if($seconds_to == 00 || $seconds_to == 0){
        $seconds_to = 59;
        $time_to -=1;
}
elseif($seconds_to == 30){
        $seconds_to = 29;
}

$from = date("Y-m-d H:i:s",
mktime($time_from,$seconds_from,0,$month_from,$day_from,$year_from)); //pieraksta
datums no
$to = date("Y-m-d H:i:s",
mktime($time_to,$seconds_to,0,$month_from,$day_from,$year_from)); // pieraksta datums
līdz

//pārbauda vai pareizi datums ievadits
if($from==$to || $from>$to){
        echo "<script language=javascript>from_to()</script>";
//aizmet atpakaļ uz sakumu
        echo "<script>window.history.back()</script>";
}
else{
//P'arbauda vai konkrets darbinieks, konkrētajā□ diena stnda un laika ir aizņ†emts
        $query6 = "SELECT * from mydb.b_pieraksts
        where fk_id_d=$d_id and fk_id_n=$wn and fk_id_w=$week_d
        and(
        ( ('$to' between datums_no and datums_lidz) or ('$from' between datums_no and
      datums_lidz) )
        or
        (datums_no > '$from' and datums_lidz < '$to' )
        )
        ";
$result6 = mysql_query($query6);
$resu = mysql_affected_rows();
```

```
if($resu==0){
$query_insert= "INSERT INTO mydb.b_pieraksts
(cena,apraksts,komentari_materiali,datums_no,datums_lidz,fk_id_k,fk_id_d,fk_id_n,fk_id_r,f
k_id_w)
VALUES('$price','$description','$materials','$from','$to',$kl_id,$d_id,$wn,$rw,$week_d)"
;
$result_insert = mysql_query($query_insert) or die(mysql_error());
echo "<script language=javascript>ins()</script>";
echo "<script>window.history.back()</script>";
}
else{
echo "NO NO INSERT" ;
}

}
}        ?>

< /body>
< /html>
```

Insertt.php

```
<!DOCTYPE html>
< html>
< body>
< script type='text/javascript' src = 'myfunc.js' ></script>

<?php

$db = mysql_connect('localhost', 'root', '') or die (mysql_error());
mysql_select_db("mydb", $db);

//$divdiv = $_POST['divdiv'];
$d_id = $_POST['ID'];
```

```
$date = $_POST['date']; //datums
$date_from = $_POST['date_from'] ; //laiks no
$darbn = $_POST['darbn'];
$col = $_POST['coll'];
$row = $_POST['roww'];


$year_from = substr($date,0,4); //gads
$month_from = substr($date,5,2); //menesis
$day_from = substr($date,8,2); //datums

//pārbauda un salabo laiku FROM
if(substr($date_from,1,1)==':'){
$time_from=substr($date_from,0,1);
$seconds_from = substr($date_from,2);

}
else{
        $time_from = substr($date_from,0,2);
        $seconds_from = substr($date_from,3);

}
$from_d=date("Y-m-d l", mktime(0,0,0,$month_from,$day_from,$year_from)); //pieraksta
datums
$from_tim = date("H:i",
mktime($time_from,$seconds_from,0,$month_from,$day_from,$year_from));//laiks no
//izveido datumu ko db parbauda
$from = date("Y-m-d H:i:s",
mktime($time_from,$seconds_from,0,$month_from,$day_from,$year_from)); //pieraksta
datums no

$query3 = "SELECT id_r from mydb.b_row where row = $row";
$result3 = mysql_query($query3);
$rowa = mysql_fetch_array($result3);
$rw = $rowa['id_r']; //row id
```

```
$week_number = date('W', strtotime($date)); //nedeÄ¼as numurs

$query4 = "SELECT id_n from mydb.b_nedela where nedela = $week_number";
$result4 = mysql_query($query4);
$weeka = mysql_fetch_array($result4);
$wn = $weeka['id_n']; // week id

$wek_d = date("N", strtotime($date));
$query5 = "SELECT id_w from mydb.b_diena where diena = $wek_d";
$result5 = mysql_query($query5);
$dien = mysql_fetch_array($result5);
$week_d = $dien['id_w'];


//skatās vai jau neeksistē ieraksts db
$query = "SELECT id_p from mydb.b_pieraksts
where fk_id_w = $week_d and fk_id_d = $d_id and fk_id_n = $wn and '$from' between
datums_no and datums_lidz"; //$from $rw
$result = mysql_query($query);
$row = mysql_affected_rows();


if ($row==1){
        $query6 = "SELECT id_p,   cena, apraksts, komentari_materiali, datums_no,
datums_lidz, fk_id_k, fk_id_d,fk_id_n,fk_id_r,fk_id_w from mydb.b_pieraksts
where fk_id_w = $week_d and fk_id_d = $d_id and fk_id_n = $wn and '$from' between
datums_no and datums_lidz";
$result6 = mysql_query($query6);
$selects = mysql_fetch_array($result6);

//Paņem visas vērtības no ieraksta
$p_id = $selects['id_p']; //pieraksta id
$price = $selects['cena']; //pieraksta cena
$description = $selects['apraksts']; //pieraksta apraksts
```

```
$materials = $selects['komentari_materiali'];
$date_from = $selects['datums_no']; // pieraksta datums no
$date_to = $selects['datums_lidz']; //pieraksta datums lidz
$kl_id = $selects['fk_id_k']; //pieraksta klienta id
$d_id = $selects['fk_id_d']; //pieraksta darbinieka id
$wn = $selects['fk_id_n']; //pieraksta nedÄ“Ä¼as id
$rw = $selects['fk_id_r']; //pieraksta row id
$week_d = $selects['fk_id_w']; // pieraksta nedÄ“Ä¼as diena


//saliek datumus korektus
$year_from = substr($date_from,0,4); //gads
$month_from = substr($date_from,5,2); //menesis
$day_from = substr($date_from,8,2); //datums

$time_from=substr($date_from,11,5); //laiks no
$time_to = substr($date_to,11,2); //laiks lidz

$seconds_to = substr($date_to,14,2); //minutes lidz
if ($seconds_to==59) {
        $seconds_to=00;
        $time_to+=1;
}
elseif($seconds_to==29){
        $seconds_to=30;
}

$time_to = date("H:i",
mktime($time_to,$seconds_to,0,$month_from,$day_from,$year_from)); // pieraksta
datumslÄ«dz izvadei

$from_d = date("Y-m-d l", mktime(0,0,0,$month_from,$day_from,$year_from)); //pieraksta
datums izvadei

//Atrod informaciju par klientu
```

```
$query7 = "SELECT k_vards,k_uzvards,k_telefons from mydb.b_klients where id_k =
$kl_id";
$result7 = mysql_query($query7);
$klients = mysql_fetch_array($result7);

$kl_vards = $klients['k_vards'];
$kl_uzvards = $klients['k_uzvards'];
$kl_numurs = $klients['k_telefons'];

echo "
<form action='labot_dzest.php' method='post' id='divdiv' >
<strong><a font size='6' id = 'darbn'>$darbn </a></strong></br>
<input type='text' name='ID' id = 'ID' value='$d_id' hidden>
<input type='text' name='col' id = 'col' value='$col' hidden></br><input type='text'
name='row' id = 'row' value='$rw' hidden >
<input type='text' name='pieraksts' id = 'pieraksts' value='$p_id' hidden>

Informacija par klientu: </br>
Vards *: </br><input type='text' name = 'name' id = 'name' value='$kl_vards' > </br>
Uzvards: </br><input type='text' name='srname' id = 'srname' value='$kl_uzvards' ></br>
Telefona nr: </br><input type='tel' name='phonenr' id = 'phonenr' value='$kl_numurs'
></br>
Apraksts *: </br><textarea cols='16' rows='3' maxlength='100' type='text' name='description'
id='description' >$description</textarea></br>
Materiali: </br><input type='text' name='materials' id='materials' value='$materials' ></br>
Cena: </br><input type='text' name='price' id='price' value='$price'></br>
Datums *: </br><input type='date' name='date' id='date' value='$from_d' readonly></br>
Laiks *: </br> No: <input type='datetime' name='date_from' id='date_from' size='3'
value='$time_from' readonly>
Līdz: <input type='datetime' name='date_to' id='date_to' size='3' value='$time_to'
></br></br>
<input type='text' name='dzest' id='dzest' hidden></br></br>
<input type='submit' id = 'labot' value = 'Labot' >
";
?>
```

```
<button type='button' onclick="deli()">Dzēst</button>
<?php
echo "</form>";

}
else{

echo "
<form action='insert.php' method='post' id='divdiv' >
<strong><a font size='6' id = 'darbn'>$darbn </a></strong></br>
<input type='text' name='ID' id = 'ID' value='$d_id' hidden></br>
<input type='text' name='col' id = 'col' value='$col' hidden></br> <input type='text'
name='row' id = 'row' value='$rw' hidden></br>
Informacija par klientu: </br>
Vards *: </br><input type='text' name = 'name' id = 'name'> </br>
Uzvards: </br><input type='text' name='srname' id = 'srname' ></br>
Telefona nr: </br><input type='tel' name='phonenr' id = 'phonenr' ></br>
Apraksts *: </br><textarea cols='16' rows='3' maxlength='100' type='text' name='description'
id='description'></textarea></br>
Materiali: </br><input type='text' name='materials' id='materials' ></br>
Cena: </br><input type='text' name='price' id='price' ></br>
Datums *: </br><input type='date' name='date' id='date' value='$from_d' readonly></br>
Laiks *: </br> No: <input type='datetime' name='date_from' id='date_from' size='3'
value='$from_tim' readonly>
Līdz: <input type='datetime' name='date_to' id='date_to' size='3' ></br></br>
<input type='submit' value = 'SaglabÄt' >
</form>";
}

?>
< /body>
< /html>
```

Myfunc.js

```
function myfunc(a,b,c,d,e,f,i,z)
{

        var darbn = a;
        var date = b ;

        var date_from = d+ ":"+e;
        var d_id = f ;
        var coll = i;
        var roww = z;
new Ajax.Request("insertt.php",
{
method: "post",
postBody:
"ID="+d_id+"&date="+date+'&date_from='+date_from+'&darbn='+darbn+'&coll='+coll+'&r
oww='+roww,
onComplete: showmsg
});
}

function showmsg(req){
//alert("abc");
        document.getElementById('left').innerHTML = req.responseText;
}

function isnot(a,b,c,d,e){
        alert('Obligātie lauki nav aizpildīti: '+a+b+c+d+e);
}

function from_to(){
        alert('Datums ievadīts nekorekti!');
}

//Ja dati insertoti veiksmigi, tad izvada pazinojumu
function ins(){
```

```
        alert('Dati saglabātif!');
}
function deli(){
//alert('Dati saglabÄ□tif!');
        document.getElementById('dzest').value ='dzest';
        document.forms["divdiv"].submit();
}
//Dati tiek izdzesti
function deleted(){
        alert('Ieraksts izdzēsts');
        history.go(-1);
}
```

Pielikums2. Izstrādātais kods Racket programmēšanas valodā

Racket_define.rtk

```
#lang web-server/insta
(require db)

(define c
        (mysql-connect #:server "localhost"
        #:database "mydb"
        #:user "root"
        )
)

;(define x (query-value pgc "select d_vards from b_darbinieks where id_d = 2"))

;(define z "three")
;(define a (map vector->list (query-rows c "select k_vards, k_uzvards, k_telefons from
b_klients where id_k>10")) )
(define query1 "select DATE_FORMAT(pie.datums_no,'%d-%m-%Y'),
DATE_FORMAT(pie.datums_no,'%H:%i'), DATE_FORMAT(pie.datums_lidz, '%H:%i')
,kli.k_vards, kli.k_uzvards, kli.k_telefons, pie.cena, pie.apraksts, pie.komentari_materiali
from b_pieraksts pie, b_darbinieks dar , b_klients kli
where dar.id_d = pie.fk_id_d
and kli.id_k = pie.FK_id_k and dar.id_d =1 order by pie.datums_no asc")
(define jan-q (map vector->list (query-rows c query1)) )
(define jan-v (query-value c "select d_vards from b_darbinieks where id_d = 1"))
(define (jan-vards)
`(tr ((style "background-color:#BDBDBD; font-weight:bold")) (td ((colspan "9")), jan-v)))

(define query2 "select DATE_FORMAT(pie.datums_no,'%d-%m-%Y'),
DATE_FORMAT(pie.datums_no,'%H:%i'), DATE_FORMAT(pie.datums_lidz, '%H:%i')
,kli.k_vards, kli.k_uzvards, kli.k_telefons, pie.cena, pie.apraksts, pie.komentari_materiali
from b_pieraksts pie, b_darbinieks dar , b_klients kli
where dar.id_d = pie.fk_id_d
```

```
and kli.id_k = pie.FK_id_k and dar.id_d =2 order by pie.datums_no asc")
(define sim-q (map vector->list (query-rows c query2)) )
(define sim-v (query-value c "select d_vards from b_darbinieks where id_d = 2"))
(define (sim-vards)
`(tr ((style "background-color:#BDBDBD; font-weight:bold"))(td ((colspan "9")), sim-v)))


(define (render-cell x)
        `(td ((width "110")) ,x)
)

(define (render-list y)
        `(tr ,@(map render-cell y))
)

(define jan
`(table ((border "1") (cellspacing "10") ) ,(jan-vards)
(tr ((style "background-color:#E6E6E6; font-weight:bold"))
(td "Datums") (td "Laiks no") (td "Laiks līdz")
(td "Vārds") (td "Uzvārds") (td "Telefona nr") (td "Cena") (td "Apraksts") (td "Materiāli")
) ,@(map render-list jan-q)
)
)
(define sim
`(table ((border "1") (cellspacing "10")) ,(sim-vards)
(tr ((style "background-color:#E6E6E6; font-weight:bold"))
(td "Datums") (td "Laiks no") (td "Laiks līdz")
(td "Vārds") (td "Uzvārds") (td "Telefona nr") (td "Cena") (td "Apraksts") (td "Materiāli")
) ,@(map render-list sim-q)
)
)

(define (start request)
        (response/xexpr
        list-start
```

```
))

(define list-start
        `(html
        (head (title "Kalendars"))
        (body (h1 "Kalendārs")
        (div , jan)
        (div , sim)
        ;(div , jur)
) ))
```

Bakalaura darbs „Tīmekļa lietotnes izstrāde imperatīvā un funkcionālā stilā” 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.

Autors: Krista Puķe

Rekomendēju darbu aizstāvēšanai

Vadītājs: profesors Dr.dat. Kārlis Čerāns 04.06.2013.

Recenzents: Dr.sc.comp. Alina Vasiļjeva

Darbs iesniegts Datorikas fakultātē 04.06.2013.

Dekāna pilnvarotā persona: metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījumu komisijas sēdē

13.06.2013. prot. Nr.

Komisijas sekretāre: