LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

APDROŠINĀŠANAS BROKERA AUTOMATIZĒTAS RĒĶINU APMAKSAS SISTĒMAS IZSTRĀDE UN INTEGRĀCIJA

Maģistra darbs

Autors: Aleksejs Osovitnijs

Stud. apl. nr.: ao05015

Vadītāja: Alina Vasiļjeva

RĪGA 2011

ANOTĀCIJA

Mūsdienīgā pasaulē liela daļa uzņēmumu, kuri saistīti ar pārdošanu, agri vai vēlu nonāk pie biznesa procesu optimizācijas ar informācijas sistēmas ieviešanas palīdzību.

Maģistra darba mērķis ir optimizēt un saīsināt biznesa procesu izpildīšanās laikus ar automatizētās rēķinu apmaksas sistēmas ieviešanu apdrošināšanas brokera uzņēmumā.

Pateicoties izvēlētajai arhitektūrai izstrādātu sistēmu var izmantot vienlaicīgi vairāki lietotāji. Izstrādes rīki ir bezmaksas, kas samazināja sistēmas realizācijas izmaksas. Pateicoties skriptu valodām *PHP* un *JavaScript* sistēma nav atkarīga no operētājsistēmas. Sistēma ir realizēta uz jauna, bet ļoti daudzsološa *PHP* ietvara *Yii*, un ir pilnīgi uzrakstīta *OOP* stilā ar *MVC* modeļa atbalstu. Autors apraksta projekta izstrādi ar *Yii* ietvaru un izdara secinājumus par ietvara efektivitāti.

ABSTRACT

Nowadays almost all companies that deal with selling goods or providing services sooner or later come to business optimization processes by information systems integration within the company.

The goal of the Master's Thesis is to optimize and reduce the time of business processes by automated billing system developing and integration within the company.

By virtue of the chosen architecture the system can be used by several users at once. Development tools are free, which noticeably reduce distribution costs. Due to the *PHP* and Java Script languages, the system will not be dependable on the operational system. The system will be developed using new but very promising *PHP* framework *Yii* and written completely in *OOP*, supported by *MVC* model, which makes it possible to smoothly develop the system without serious consequences. The author describes development of the project based on Yii framework and draw the conclusions about the efficiency of the framework.

AUTOREFERĀTS

Maģistra darbā, balstoties uz uzņēmuma tekošajiem biznesa procesiem un to analīzi, ir izskatīta rēķinu sistēmas izstrāde konkrētam apdrošināšanas brokerim. Darba autors vadīja projektu no un līdz, ieskaitot:

- Pārrunas ar klientu
- Tekošo biznesa procesu aprakstu un analīzi
- Plānojamā sistēmas funkcionāla sastādīšanu
- Personāla atlasi projekta realizācijai
- Realizācijas līdzekļu izvēli

Rezultātā tika izveidota apdrošināšanas brokera rēķinu sistēma, kas tiek galā ar tai noteiktajiem uzdevumiem.

Par sistēmas pamatu tika ņemts *Yii* ietvars. Maģistra darbā autors veic populāru *PHP* ietvaru analīzi, kā arī izskaidro, kāpēc izvēlējās tieši šo ietvaru. Darbā autors arī stāsta par šī ietvara iespējām un priekšrocībām.

APZĪMEJUMU SARAKSTS

MVC – *Model View Controller* – Modelis Skats Kontrolieris

HTML - *HyperText Markup Language* - Hiperteksta iezīmēšanas valoda, valoda HTML

YII – *Yes! It is*! – Jā! Tas ir! (PHP valodas ietvars)

WEB – *World Wide Web* - globālais tīmeklis

OOP – *Object Oriented Programming* – objekt-orientēta programmēšana

XSS - *Cross Site Scripting* – starplapu kodēšana

SQL - *Structured Query Language* - strukturēta vaicājumvalo

AP – ActiveRecord – speciālais modelis darbām ar datu bāzi

CRUD – *Create Read Update Delete*- modeļas bāzes funkcijas

SQL - *Structured Query Language* - strukturēta vaicājumvaloda, ko izmanto, lai piekļūtu datu bāzes resursiem

IT – *Information Technology* – informācijas tehnoloģijas

URL - *Uniform Resource Locator* - vienotais resursu vietrādis

PDF – *Portable Document Format* – portatīvs dokumenta formāts

API - *Application Program Interface* - lietojumprogrammas saskarne

PDO - *PHP Data Objects* **–** PHP paplašinājums, kas nodrošina izstrādātājiem vienkāršu un universālu interfeisu, lai piekļūtu dažādām datu bāzēm

HTTP - *Hyper Text Transport Protocol* - hiperteksta transporta protokols

SATURS

IEVADS

Mazajā un vidējā biznesā konkurence ir ļoti liela, tāpēc šo klašu pārstāvji cenšas kaut kādā veidā optimizēt savus biznesa procesus. Ko dod procesu uzlabošana ar elektroniskās sistēmas ieviešanu? Tas ļauj ievērojami palielināt datu apstrādes ātrumu, neuzturot kompānijā milzīgu darbinieku štatu, uzlabo konkurētspēju un elastību kompānijā, padara kompāniju adekvātu tirgus izmaiņām, principiāli uzlabo produktu un pakalpojumu kvalitāti un kontroli. Bieži vien uzņēmuma biznesa procesu optimizācija notiek, ieviešot dažāda veida informācijas sistēmas. Mūsdienās ļoti daudzas informācijas sistēmas tiek izstrādātas kā tīmekļa bāzētas lietojumprogrammas ar programmnodrošinājumu un programmēšanas valodās ar atvērtu sākotnējo kodu. Tas savukārt nozīmē, ka uzņēmumiem jāmaksā tikai par tādas informācijas sistēmas izstrādi arī turpmāk, pie nepieciešamības, par papildus funkcionalitāti (ideālā gadījumā).

YII ietvars ir viens no jaunākajiem, tomēr visātrāk attīstītošiem un perspektīviem ietvariem *PHP* programmēšanas valodā. Tāpat *Yii* ir *MVC* ietvars, kurā biznesa loģika, *HTML* izvads un datu bāze veido trīs atsevišķus komponentus, kuri savstarpēji iedarbojas (par to tiks stāstīts zemāk).

Yii ietvara dibinātājs ir Qiang Xue. Ietvars parādījās 2008.gada 1.janvārī. Qiang iepriekš izstrādāja un uzturēja *Prado* ietvaru. 2008.gada 3.decembrī, pēc aptuveni viena izstrādes gada, *Yii 1.0* tika oficiāli palaists masveida lietošanā.

Maģistra darbā autors iepazīstina lasītāju ar dažiem *PHP* ietvariem un to īpatnībām, kā arī paskaidro, kāpēc ir izvēlējies tieši *YII* ietvaru, lai optimizētu un saīsinātu biznesa procesu izpildīšanas laikus, ar automatizētās rēķinu apmaksas sistēmas ieviešanu apdrošināšanas brokera uzņēmumā.

1. SISTĒMAS PRASĪBU DEFINĒŠANA

Šajā nodaļā tiek aprakstīti uzņēmuma tekošie biznesa procesi, kā arī veikta šo biznesa procesu analīze ar mērķi noformulēt prasības pret nākamo rēķinu sistēmu.

1.1. Klienta uzņēmuma apraksts

Klients ir firma "MARKO BROKERI", kura atrodas pēc adreses Rīga, Lubānas iela 129A, Latvija, reģistrācijas numurs 50003789611. Firma tika dibināta 2004.gadā, uz doto brīdi firmā strādā 10 darbinieki.

Uzņēmuma galvenā darbības sfēra – tā ir apdrošināšanas aģentu un brokeru darbība (tā ierakstīts reģistrā). Uz doto brīdi firma nodarbojas ar:

- vietu iznomāšanu autostāvvietā
- automobiļu garāžu iznomāšanu
- automobiļu mazgāšanu
- apdrošināšanas un brokeru pakalpojumiem. Uzņēmums sadarbojas ar visām populārākajām Latvijas apdrošināšanas aģentūrām. Uzņēmums sniedz savus pakalpojumus visu diennakti, un darbinieku darba laiks ir organizēts maiņās.

Uzņēmumā jau ir viena informācijas sistēma, kura veic klientu uzskaiti, kuri iznomā garāžas telpas. Šo sistēmu izstrādāja ne mūsu firma, tomēr, pēc direktora teiktā, ir uzprogrammēta kvalitatīvi, pilnībā apmierina viņu vajadzības un tai nav nepieciešama modifikācija.

Uz doto brīdi uzņēmumā tiek izrakstītas aptuveni no piecdesmit līdz sešdesmit *OCTA* apdrošināšanas katru dienu.

1.2. Uzņēmuma biznesa procesu apraksts

Šajā nodaļā autors apraksta uzņēmuma biznesa procesus, kuru izpilde tiks mainīta, ieviešot un integrējot jaunu informācijas sistēmu. Visus nākamos biznesa procesus darbinieki veica manuāli, izmantojot kaut kādus parocīgus līdzekļus, tādus kā *MS Excel*, *MS Office*, *Acrobat Reader* utt.

1.2.1. Apdrošināšanas polises pirkšana

Apdrošināšanas polises pirkšana – tas ir galvenais biznesa process, kurš norisinās uzņēmumā. Uz doto brīdi uzņēmums pārdod tādus apdrošināšanas veidus kā:

- **OCTA -** Obligātā Civiltiesiskās Atbildības (OCTA) apdrošināšana.
- **KASKO** – automobiļu apdrošināšana.
- **Īpašuma apdrošināšana**
- **KOCTA** – Krievijas Obligātā Civiltiesiskās Atbildības apdrošināšana.
- **Zaļā karte** – dod iespēju braukāt kā apdrošinātai personai pa valstīm, kuras neietilpst ES (Krievija, Baltkrievija, Černogorje utt.). Darbības termiņš no 15 dienām līdz gadam.
- **Robežlīgums** – apdrošināšana Latvijā automobiļiem ar krievu, ukraiņu, baltkrievu numuriem. Darbības termiņš no 1 mēneša līdz 1 gadam.
- **CTA** – Civiltiesiskā apdrošināšana. Visi civiltiesiskās atbildības veidi.
- **KASKO BEDRES** –KASKO polise, kas darbojas uz automobiļa riepām.
- **LR iebraucošo apdrošināšana** – apdrošināšana iedzīvotājiem no citām valstīm, kuri vēlas iegūt pastāvīgās uzturēšanās atļauju Latvijā.
- **Ceļojumu apdrošināšana**

Apdrošināšanas pirkšanas laikā uzņēmuma darbiniekam ir obligāti jāaizpilda personīgus datus par apdrošināšanas turētāju, datus par apdrošināšanas īpašnieku, kā arī polises datus. Tamdēļ, ka uzņēmums sadarbojas ar vairākām vadošām Latvijas apdrošināšanas kompānijām, uzņēmuma darbiniekam sākumā jāatrod piemērotu apdrošināšanas kompāniju un tikai pēc tam jāaizpilda apdrošināšanas polisi, norādot tur izvēlēto apdrošinātāju. Tālāk darbiniekam jāaizpilda datus par klientu un jāiereģistrē viņu izvēlētās apdrošināšanas kompānijas informācijas sistēmā.

1.2.2. Apdrošināšanas polises pagarināšana

Beidzoties jebkuras apdrošināšanas polises termiņam, to var pagarināt. Faktiski apdrošināšanas polises pagarināšana ir jaunās apdrošināšanas polises reģistrācija, bet jau ar esošajiem klienta datiem. Līdz informācijas sistēmas ieviešanai uzņēmuma darbiniekam nācās manuāli meklēt datus par klientu un iepriekšējo apdrošināšanas polisi un uz šo datu pamata izrakstīt jaunu apdrošināšanas polisi.

1.2.3. Klienta meklēšana

Daudzas operācijas uzņēmumā saistās ar klientu informāciju, vai tas būtu kopsavilkums par klientu, vai parādnieku saraksts. Šajā nolūkā darbiniekiem nācās manuāli meklēt klientu informāciju visās apdrošinātāju informācijas sistēmās, tālāk apkopot tās *Ms Excel* failā vai vēl kaut kur.

1.2.4. Atgādinājumi par rēķina apmaksu

Pērkot apdrošināšanas polisi, to var apmaksāt uz vietas, pie nepieciešamības apmaksāt bankā. Ja klients vēlas apmaksāt apdrošināšanas polisi bankā, tad viņam tiek izrakstīts rēķins ar apmaksas termiņu. Tāpat kā visās biznesa sfērās, pastāv tādi klienti, kuri aizmirsa vai nevēlas apmaksāt rēķinu, tāpēc uzņēmuma darbinieki apzvana parādniekus un risina jautājumu par apmaksu individuālajā kārtībā.

1.2.5. Parādnieku saraksta sastādīšana

Šis biznesa process cieši saistās ar *„Atgādinājumiem par rēķina apmaksu”* (1.2.4.punkts). Eksistē tādi klienti, kuri pat pēc atgādinājuma neapmaksā rēķinus. Tādā gadījumā uzņēmuma darbinieki sastāda parādnieku sarakstus, pēc kuru sastādīšanas kļūst skaidrs, cik naudas viņi ir parādā uzņēmumam.

1.2.6. Uzņēmuma peļņas pārskata sastādīšana no apdrošināšanas polišu pārdošanas

Katra mēneša beigās notiek apdrošināšanas kompāniju naudas norēķini ar uzņēmumu. Šajā nolūkā uzņēmums sagatavo speciālus izrakstīto apdrošināšanas polišu sarakstus, kuri satur:

- Klienta datus
- Izrakstītās polises datus
- Polises summu
- Brokera procenta summu

Tālāk šos sarakstus uzņēmums nosūta apdrošināšanas kompānijām, kuras savukārt pārbauda šos sarakstus un maksā uzņēmumam tā nopelnīto brokera procentu.

1.2.7. Brokera rēķina sastādīšana apdrošināšanas kompānijām

Uz *„Uzņēmuma peļņas pārskata sastādīšana no apdrošināšanas polišu pārdošanas”* (1.2.6.p.) pamata uzņēmums izraksta rēķinu apdrošināšanas kompānijai, kurā tiek norādīta brokera peļņa.

1.2.8. Apmaksāto rēķinu saraksta sastādīšana

Lai uzzinātu uzņēmuma apgrozījumu pēc apmaksātajām apdrošināšanas polisēm, kompānijas darbiniekiem ir jāveido apmaksāto rēķinu saraksts. Tas ir pietiekami darbietilpīgs process, kurš aizņem daudz laika.

1.2.9. Atgādinājums par apdrošināšanas perioda derīguma termiņa beigām

Savu klientu ērtībai uzņēmumam ir jāatgādina saviem klientiem par to, ka apdrošināšanas polises derīguma termiņš drīz beigsies. Līdz informācijas sistēmas ieviešanai šis process aizņēma ļoti daudz laika un sastāvēja no vairākiem posmiem:

- Katras nedēļas sākumā uzņēmuma darbiniekiem bija jāsastāda apdrošināšanas polišu sarakstu, kurām drīz jābeidzas derīguma termiņam.
- Katru dienu, izejot no šī saraksta, darbinieki manuāli izsūtīja īsziņu ar brīdinājumu.

1.2.10.Unikāla rēķina numura ievērošana

Firmā pastāv vairāki izejošo rēķinu veidi, tādi kā:

- Rēķini klientiem par apdrošināšanas polises apmaksu
- Rēķins apdrošināšanas kompānijām par brokera peļņas apmaksu

Katram no šiem rēķiniem jābūt savam unikālam numuram, vienlaicīgi vēlams, lai visi rēķinu numuri ietu pēc kārtas. Rēķinu numuri izskatās sekojošā veidā:

2010-50 , 2010-51, 2010-52, 2010-53 …

Pirmie 4 cipari līdz domuzīmei – tas ir gads, nākamie cipari – rēķina kārtas numurs šajā gadā. Ja rēķins ir anulēts, tad rēķina numurs vienalga paliek ierezervēts un uzņēmumam jānorāda pārskatos priekš VID, ka rēķins tika anulēts.

1.2.11.Darbinieku darba alga

Uzņēmuma darbinieku, kuri nodarbojas ar apdrošināšanas polišu pārdošanu, darba alga ir tieši atkarīga no viņu maiņā pārdoto polišu skaita, jo daļu no viņu algas veido procents no šī daudzuma. Līdz informācijas sistēmas ieviešanai darbinieku darba algu rēķināja šādā veidā:

- Pēc maiņas grafika sastādīja darbinieka darba intervālus (datums un stundas)
- Pēc šiem darba intervāliem rēķināja pārdotās apdrošināšanas polises, uzņēmuma peļņu un paša darbinieka nopelnīto

1.2.12.Ceturkšņa pārskata sastādīšana pēc parakstītajām polisēm priekš Latvijas brokeru asociācijas

Katru ceturksni gadā uzņēmuma direktors manuāli izgāja cauri visām parakstītajām apdrošināšanas polisēm un sastādīja pārskatu priekš Latvijas brokeru asociācijas.

1.3. Biznesa procesu analīze

Ievadāmā informācijas sistēma pamatā saīsinās izpildes laiku un iesaistīto darbinieku skaitu noteiktam biznesa procesam. Protams, katru procesu var pildīt tikai viens darbinieks, bet tādā gadījumā izpildes laiks palielināsies. 2.1.tabulā ir uzskaitīti visi biznesa procesi, kuri būs skarti saistībā ar informācijas sistēmas ieviešanu uzņēmumā. Zemāk parādītās tabulas dati ņemti no uzņēmuma direktora vārdiem.

1.1. tabula

Biznesa procesu apraksts

Punkts	Biznesa process	Patērētais laiks	Iedarbināto darbinieku skaits
1.2.1	Apdrošināšanas polises pirkšana	10 - 30 min.	1
1.2.2	Apdrošināšanas pagarināšana	10 - 30 min.	1
1.2.3	Klienta meklēšana	30-40 min.	1
1.2.4	Atgādinājums par rēķina apmaksu	2 stundas	2
1.2.5	Parādnieku saraksta sastādīšana	2 stundas	2
1.2.6	Uzņēmuma peļņas pārskata sastādīšana par apdrošināšanas polišu pārdošanu	3 stundas	1
1.2.7	Brokera rēķina sastādīšana apdrošināšanas kompānijām	1.5 stundas	1
1.2.8	Apmaksāto rēķinu saraksta sastādīšana	2 stundas	2
1.2.9	Atgādinājums par apdrošināšanas derīguma termiņa beigām	3 stundas	2
1.2.10	Unikāla rēķina numura ievērošana	5 min.	1
1.2.11	Darbinieku darba alga	40 min.	1
1.2.12	Ceturkšņa pārskata sastādīšana pēc parakstītajām polisēm Latvijas brokeru asociācijai	2 stundas	1

1.3.1. Darba procesa tekošā modeļa problēmas

Uz doto brīdi uzņēmumā nav nekāda kopīga klientu un izrakstīto apdrošināšanas polišu reģistra, tomēr, reģistrējot jaunu apdrošināšanas polisi, dati par klientu tiek ievadīti vienā no izvēlētās apdrošināšanas kompānijas sistēmām. Tādējādi, klienta informācija glabājas apdrošināšanas kompānijas informācijas sistēmā. Šajā informācijas sistēmā, protams, ir realizēta klienta informācijas meklēšana/apskate/rediģēšana/pievienošana, tomēr problēma ir tāda, ka klients var nopirkt apdrošināšanas polisi no sākuma vienā apdrošināšanas kompānijā, pēc tam citā. Tādā gadījumā klienta izrakstīto apdrošināšanas polišu izsekošana kļūst apgrūtinoša. Ļoti bieži klients zvana pa telefonu un saka, ka pirka polisi klienta uzņēmumā, var nosaukt savu vārdu, uzvārdu, personas kodu un vēlas uzdot kaut kādu jautājumus par apdrošināšanas polisi. Šādā gadījumā ir operatīvi jāatrod informāciju par klientu, kas dotajā gadījumā gandrīz nav iespējami.

Pērkot *KASKO* polisi, maksu par to var sadalīt vairākās daļās. Pirmo maksājumu klients veic tieši uz vietas, bet nākamos ik pēc kāda laika. Kārtējo maksājumu izsekošana pēc *KASKO* polisēm bija praktiski neiespējama.

Apgrūtināti bija arī *„Atgādinājumi par rēķina apmaksu”* (1.2.4.punkts) un *„Atgādinājums par apdrošināšanas polises derīguma termiņa beigām”* (1.2.9.punkts). Šie abi procesi prasa lielu rūpību un izslēdz jebkādu cilvēka kļūdīšanās faktoru, jo klients paliek ļoti neapmierināts, ja viņam kļūdas pēc zvana vai nosūta paziņojumu par neesošo parādu vai atgādina par apdrošināšanas derīguma termiņa beigām. Šos abus procesus var automatizēt, izsūtot ziņojumus ar atgādinājumu.

Procesu *„Atgādinājums par apdrošināšanas polises derīguma termiņa beigām”* (1.2.9.punkts) līdz informācijas sistēmai pildīja apsargs, kurš katru nakti dežūrēja un izsūtīja ziņojumus. Par šo darbu apsargam piemaksāja noteiktu summu. Protams, ne visi klienti bija apmierināti ar to, ka atgādinājums atnāk naktī.

Tādi biznesa procesi kā:

- Parādnieku saraksta sastādīšana (1.2.5.punkts),
- Uzņēmuma peļņas pārskata sastādīšana par apdrošināšanas polišu pārdošanu (1.2.6.punkts),
- Brokera rēķina sastādīšana apdrošināšanas kompānijām (1.2.7.punkts),
- Apmaksāto rēķinu saraksta sastādīšana (1.2.8.punkts),

ir ne kas cits kā uzņēmuma pārskati, kurus var viegli ģenerēt automātiski, pareizi saplānojot datu bāzi. Laiks šo procesu izpildei būtu ne ilgāks par 5 minūtēm uz katru pārskatu.

Process *„Unikāla rēķina numura ievērošana”* (1.2.10.punkts) ir aktuāls tāpēc, ka ļoti bieži firmas darbinieki kļūdās rēķinu numuros un dažiem rēķiniem uzliek jau esošo numuru vai pat pārlec par vairākiem numuriem uz priekšu. Tas nav ļoti labi no grāmatvedības redzes viedokļa, jo:

- Nosakot esošā rēķina numuru kaut kādam citam rēķinam, nākas labot rēķina numuru un pārtaisīt maksājuma uzdevumu
- Pārlecot par vairākiem numuriem uz priekšu, nākas vai nu aizpildīt tukšu numura nišu ar nākamajiem rēķiniem, vai skaidrot *VID*(Valsts Ieņēmumu Dienests), kāpēc tā notika, jo šādi lēcieni izraisa aizdomas *VID*.

Rēķinu sistēmā šo procesu var automatizēt, un pati sistēma var kontrolēt unikālu numuru piešķiršanu pēc noteikta algoritma.

Visu biznesa procesu izpildes laiki ir stipri atkarīgi no tādas informācijas meklēšanas kā: klienta dati, klientu apdrošinājuma polišu dati, maksājumu dati. No tā seko, ka nākamās sistēmas pamatā būs 3 galvenie moduļi:

- **Klientu reģistrs** – šajā modulī glabāsies visa personīgā informācija par klientu, tāda kā: vārds, uzvārds, dzīves adrese, telefona numurs, e-pasts, personas kods utt.
- **Apdrošināšanas polišu reģistrs** – šajā daļā glabāsies informācija par apdrošināšanas polisi: kas par polisi, kas izdevis, kad izrakstīta, caur kādu apdrošināšanas firmu nopirkta, kas ir polises īpašnieks, kas ir polises turētājs.
- **Apdrošināšanas polišu apmaksas reģistrs** – dažus apdrošināšanas polišu veidus var apmaksāt pa daļām (piemēram, *KASKO*). Šādā gadījumā apmaksu par apdrošināšanas polisi var sadalīt vairākās daļās (1, 2, 3, 4, 6, 12). Katrai apmaksai būs savus numurs, apmaksas termiņš un apmaksas summa.
- **Atskaišu modulis** – šis modulis atbildēs par nepieciešamajām atskaitēm, tas ģenerēs atskaites, izejot no uzdotajiem parametriem.

1.4. Plānojamās sistēmas prasības

Analizējot dotos biznesa procesus un apkopojot direktora un darbinieku vēlmes, autors darba turpinājumā aprakstīs prasības rēķinu apmaksas sistēmai. Šajā nodaļa autors apskata jaunas rēķinu apmaksas sistēmas funkcionalitāti biznesa līmenī. Biznesa līmenis nozīmē, ka autors neapraksta katru funkciju detalizēti, t.i., neapraksta katru ievadīto lauku atsevišķi un funkcijas izpildīšanas kartību. Šajā nodaļa tiek aprakstīti unikālie procesi, kuriem izpildīšanās gaita notiek neparasti, neviendabīgi. Visi procesi tiek apkopotas moduļos. Katras funkcijas aprakstā tiek iekļauts:

- Mērķis – priekš kam process ir paredzēts;
- Apraksts – procesa apraksts.

1.4.1. Klientu reģistrs

Klients – tā ir persona, kura apdrošina kaut ko, dotajā gadījumā klients ir turētājs. Tāpat kartiņā tiks saglabāti īpašnieka dati, tas ir, personas, kurai pieder objekts. Piemēram, gadījumā, ja klients apdrošina automobili, bet automobilis ir nopirkts līzingā, tad īpašnieks būs banka, bet turētājs persona, kura izmanto automobili. Īpašnieka dati būs piesaistīti

noteiktai polisei, jo vienam klientam (turētājam) var būt apdrošināti vairāki objekti. No tā seko, ka katram klientam var būt vairāki apdrošināti objekti un „īpašnieka" dati katrā gadījumā var būt dažādi. Nepieciešama informācija gan par turētāju, gan par īpašnieku ir sekojoša:

- Vārds, Uzvārds / Nosaukums
- Pers. kods / Reģ. Numurs
- Jur. adrese
- E-pasts
- Tālrunis
- Komentāri
- Banka
- Banka kods
- Bankas konts

1.4.2. Apdrošināšanas polises reģistrs

Kā jau tika rakstīts iepriekš, polise var būt 10 veidu: OCTA, KASKO, Īpašums, KOCTA, Zaļā karte, Robežlīgums, CTA, KASKO BEDRES, LR iebraucošo apdrošināšana, Ceļojumi. Uzņēmumā apdrošināšanas polisi OCTA noformē aptuveni no piecdesmit līdz sešdesmit cilvēku dienā, tāpēc klienta personīgos datus bieži vien saglabā tikai apdrošināšanas polišu informācijas sistēmās, jo visu klientu saglabāšana aizņemtu ļoti daudz laika, un nav zināms, vai klients vērsīsies arī nākamreiz. Tāpēc jābūt iespējai pievienot „anonīmu" klientu (ievadot tikai datus par automobili un tālruņa numuru) gadījumā, ja tiek pirkta apdrošināšanas polise OCTA. Polises pirkšanu jāuzrāda atskaitēs. Pēc pogas „Izveidot jaunu polisi" nospiešanas parādīsies tukši lauki (to daudzums var būt dažāds, atkarībā no polises veida), kurus ir nepieciešams aizpildīt. Tiks arī piedāvāts ievadīt/mainīt īpašnieka datus, kuri pēc noklusējuma būs vienādi ar turētāja datiem.

Pēc polises derīguma termiņa beigām būs iespēja pagarināt polisi, kas pēc būtības ir jaunas, tādas pašas polises pievienošana, tam pašam objektam.

Ja polise jau ir apmaksāta vai izdarīts pirmais maksājums, tad polises datus vairs mainīt nevar.

Būs iespēja anulēt polisi. Šādā gadījumā polise un tās neapmaksātie maksājumi netiek ņemti vērā, bet tiek uzrādīti pārskatos. Anulējot polisi, obligāti jāieraksta komentāru, kādu iemeslu dēļ polise tika anulēta.

Pirmā pielikumā ir parādīta lauku sarakstu tabula, kuriem jābūt noteikta veida apdrošināšanas polisē.

1.4.2.1. Pielikums KASKO

Apdrošināšanas praksē mēdz būt situācijas, kad cilvēks nopirka *KASKO* pa Latviju, bet pēc nedēļas sagribēja aizbraukt, teiksim, uz Krievijas Federāciju. Šajā gadījumā klients var noformēt sev vienreizēju izbraukumu – pielikumu noteiktai *KASKO* polisei. *KASKO* pielikums satur sekojošo informāciju:

- Cena
- Pielikuma numurs
- Izrakstīšanas datums
- No kāda līdz kādam datumam
- Valsts, kur klients izbrauks

Pielikums tiek piesaistīts polisei un apmaksāts vienā reizē.

1.4.3. Apdrošināšanās polises apmaksas reģistrs

KASKO un Īpašuma apdrošināšanas polišu apmaksu var sadalīt vairākos maksājumos. Ievadot maksājumu skaitu, parādās vairākas tukšas rindas – samaksu sagataves. Apmaksas lauki:

- **Numurs:**
    - ja maksājuma veids ir bezskaidras naudas **pārskaitījums,** tad maksājumam jānumurējas automātiski, pretējā gadījumā numuru nav jāliek. Maksājuma numerācijas metode ir aprakstīta 1.2.10.punktā.
    - ja maksājuma veids **Tiešs pārskaitījums, Pārskaitījums vai Terminmāls,** tad jābūt iespējai ievadīt maksājuma kvīts numuru (kvīts numurs ir ierakstīts maksājuma kvīts grāmatiņā katrā lapā)
- **Datums līdz** – līdz kuram jāsamaksā
- **Summa** – cik jāsamaksā
- **Apmaksas datums** – kad veikts maksājums
- **Maksājuma veids** – izkrītošais saraksts (pārskaitījums, tiešs pārskaitījums, terminmāls vai skaidrā naudā). Ja samaksa tiek veikta skaidrā naudā, tad automātiski tiek uzlikts ķeksītis laukā „Apmaksa”.

- **Komisijas nauda** – cik naudas uzņēmums nopelnīja no šī maksājuma (nopelnītais procents tiek ievadīts apdrošināšanas kompānijas datos)
- **Pārskata datums** – datums (formātā: mēn./gads), kurš pēc noklusējuma ir vienāds ar „datumu līdz", tomēr to var mainīt manuāli. Pēc šī datuma sistēma noteikts, kādam pārskatam (pēc mēneša un datuma) attiecināt vienu vai citu maksājumu.
- **Apmaksa** – ķeksītis, maksājums apmaksāts vai nē.

Maksājumu varēs nosūtīt pa e-pastu tieši no sistēmas, kā arī izdrukāt to. Abos gadījumos maksājums tiek ģenerēts *PDF* formātā.

1.4.4. Apdrošināšanas kompānijas

Sistēmā būs iespēja pievienot/dzēst/rediģēt apdrošināšanas kompānijas. Informācija par apdrošināšanas kompānijām būs sekojoša:

- Nosaukums
- Reģ. Numurs
- Juridiskā adrese
- Faktiskā adrese
- Telefons
- E-pasts
- Banka
- Bankas kods
- Bankas konts
- KASKO uzcenojums
- OCTA uzcenojums
- Īpašums uzcenojums
- KOCTA uzcenojums
- Zaļā karte uzcenojums
- Robežlīgums uzcenojums
- CTA uzcenojums
- KASKO BEDRES uzcenojums

Visa informācija, kura saistās ar uzcenojumiem, glabāsies un tiks reģistrēta sistēmā procentu veidā. No šiem laukiem sistēma rēķinās kompānijas peļņu pēc noteiktiem apdrošināšanas polišu veidiem.

1.4.5. Atskaites modulis

Sistēmā būs paredzēti vairāki atskaišu veidi:

- **Apmaksātie rēķini** - kur, apmaksas datums iekrīt uzdotajā periodā.
- **Debitori (neapmaksātie rēķini)** – kur, apmaksas datums ir tukšs, bet „apmaksāt līdz" iekrīt ievadīta perioda.
- **Pārskaitījumi** – tie maksājumi, kurus veica ar pārskaitījumu un kuru noslēgšanas datumi iekrīt noteiktā intervālā
- **Atskaite par brokeru veikto piesaisti** – atskaite, kurā parādīts, cik naudas nopelnīja uzņēmums, sadalīta pa apdrošināšanas kompānijām. Šo atskaiti uzņēmuma darbinieks katru mēnesi nodod apdrošināšanas kompānijās, lai saņemtu nopelnīto brokera procentu.

Katrai atskaitei varēs uzdot vairākus parametrus. Ja parametrs ir tukšs, sistēmai jāņem visus iespējamos šī parametra lielumus:

- **Periods (no, līdz)**
- **Apdrošināšanas kompānija** – var izvēlēties vairākas apdrošināšanas kompānijas.
- **Apdrošināšanas veids** – var izvēlēties vairākus parametrus.
- **Lietotājs** – izvēlēties maksājumus pa apdrošināšanas polisēm, kuras tika izrakstītas ar noteiktu uzņēmuma darbinieku palīdzību.

Visām atskaitēm, izņemot "*Atskaiti par par brokeru veikto piesaisti* ", jāizvada sekojošo informāciju:

- **Polises Nr.**
- **Klients** – vārds, uzvārds (ja fiziska persona, tad ņemt īpašnieka datums, ja juridiska – tad turētāja datums).
- **Izsniegšanas datums**
- **Apdrošināšanas veids**
- **Valūta**
- **Maksājums** (kāds maksājums pēc skaita: 1/1, ¼, 2/4 utt.). *Šī informācija tiek izvadīta atskaitē „apmaksātie maksājumi"* .
- **Saņemtā prēmija** – cik apmaksai no šī maksājuma
- **Kopējā prēmija** – cik maksāja polise
- **Klienta tālrunis**

- **Komisija –** tiks izvadīta procentos. *Šī informācija tiek izvadīta atskaitē „apmaksātie maksājumi”*.
- **Komisijas naudā** - *Šī informācija tiek izvadīta atskaitē „apmaksātie maksājumi”*.
- **Apmaksas veids** – (pārskaitījums vai skaidra nauda). *Šī informācija tiek izvadīta atskaitē „apmaksātie maksājumi”*.

Rindu grupēšanai jābūt pēc apdrošināšanas veidiem. Katra grupējuma beigās jānorāda kopsummu, skaidrās naudas kopsummu. Atskaites beigās arī jānorāda kopsummu, kopsummu ar pārskaitījumu, kopsummu skaidrā naudā pēc visām apdrošināšanas firmām.

“Atskaitei par brokeru veikto piesaisti” jāģenerē rēķinu *PDF* formātā.

1.4.6. Informācijas meklēšana

Operatīva nepieciešamās informācijas meklēšana sistēmā būtiski samazinās jebkura procesa izpildes laiku, jo visu procesu pamatā ir nepieciešamās informācijas meklēšana. Kompānijas darbinieku intervijas gaitā tika atklāts, ka visbiežāk meklē informāciju pēc:

- Mašīnas numura
- Apdrošināšanas polises numura
- Uzņēmuma reģistrācijas numura vai personas koda
- Vārda, uzvārda vai firmas nosaukuma

1.4.7. Sistēmas lietotāji

Sistēmā būs paredzēti divu veidu lietotāji:

Administrators – ir piekļuve visām datu bāzes opcijām. Šim lietotājam būs iespēja pievienot/dzēst lietotājus no sistēmas, vienlaicīgi norādot, pie kādas grupas pieder lietotājs.

Vienkāršs lietotājs – ir ierobežota piekļuve datu bāzei. Neredz dažus laukus un atskaites. Nevar pievienot vai mainīt jebkādus citus lietotājus.

1.4.8. Vispārēji

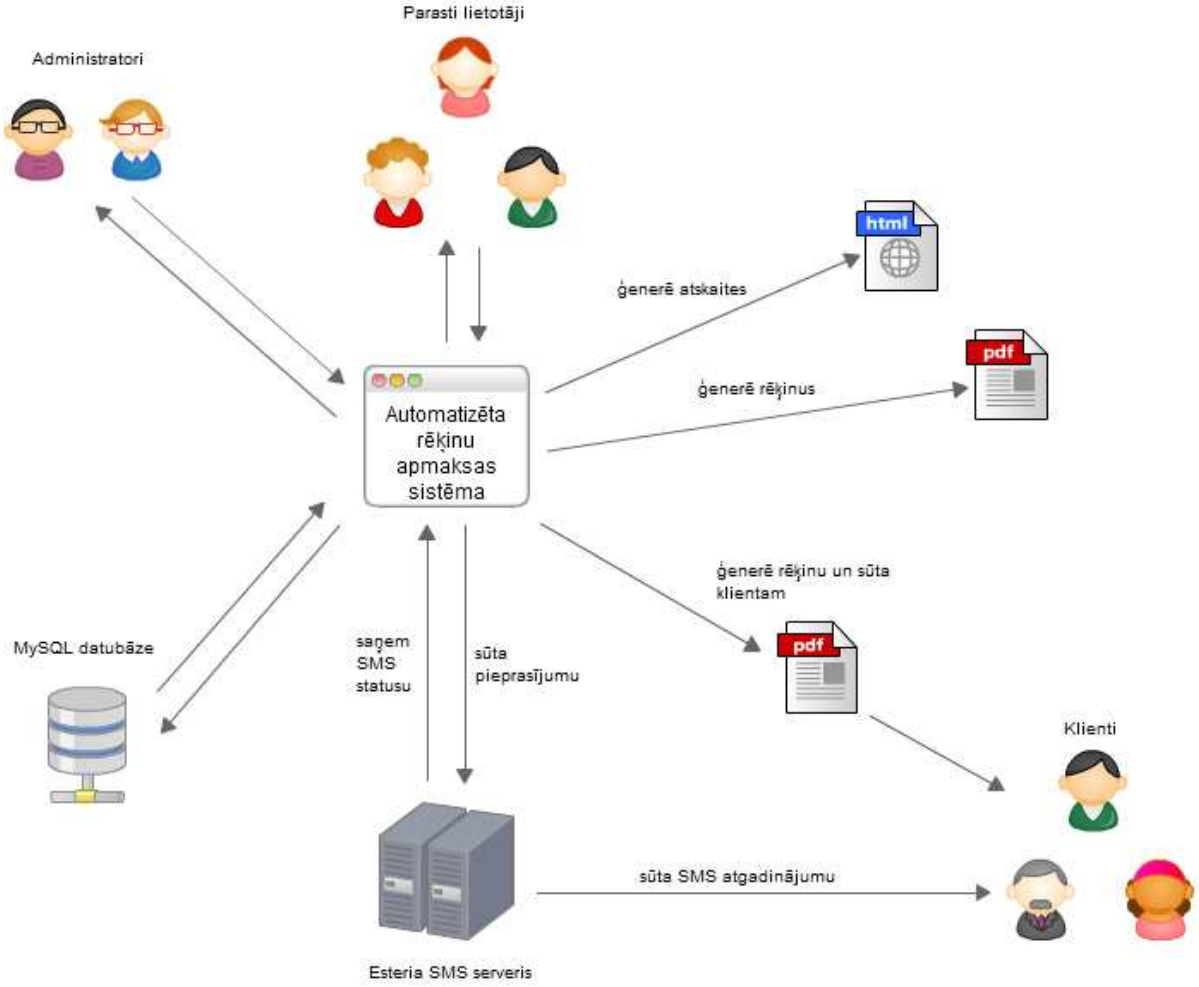
- Sistēmas valoda – latviešu.
- Visus laukus sistēmā (vārds, uzvārds utt.) jāsaglabā augšējā reģistrā, neatkarīgi no tā, kā tie tika ievadīti.
- Katram klientam var būt vairākas polises.
- Iespēja apskatīt iepriekšējas klienta polises.

2. SISTĒMAS ARHITEKTŪRAS UN IZSTRĀDES RĪKU IZVĒLE

Viens no svarīgākajiem veiksmīga projekta realizācijas punktiem ir arhitektūras un izstrādes līdzekļu izvēle. Dotajā nodaļā tiek aprakstīts, kādu arhitektūru un kādus līdzekļus autors izvēlējās sava projekta realizācijai.

2.1. Automatizētas rēķinu apmaksas sistēmas darbības princips

2.1.attēlā shematiski ir parādīts informācijas sistēmas darbības princips.



2.1 att. **Darbības principa shēma**

Shēmā attēlots:

- **Lietotāju grupa** – tie ir vienkārši sistēmas lietotāji, kuri ievadīs informāciju un saņems to atkarībā no ievadītajiem kritērijiem.
- **Administratoru grupa** – šai grupai ir tiesības darīt tās pašas operācijas kā lietotāju grupai, bet, izņemot to, šai grupai būs piekļuve daža informācija un opcijas, kuras nav pieejamas lietotājiem.

- Informācijas sistēma ģenerēs atskaites *HTML* formātā ar minimālu stilu kopumu, bez jebkādiem grafiskiem elementiem. Stili būs izstrādāti, ņemot vērā to, ka, izdrukājot atskaiti, tika patērēts vismazākais printera tonera daudzums. Atskaitēm izvēlēts *HTML* formāts, jo tā ģenerēšanai tiek patērēts vismazākais laika daudzums (tas pats *PDF* formāts strādās ilgāk).
- Informācijas sistēma ģenerēs klientu rēķinus *PDF* formālā, jo bieži nākas sūtīt klientam rēķinus uz el. pastu, bet ne visiem klientiem rēķins var atspoguļoties korekti (rēķinā tiek izmantoti attēli).
- Atgādinājums par apdrošināšanas polises derīguma termiņa beigām īstenots caur *SMS* ziņojumiem. Šajā nolūkā tiks izmantots *SMS* izsūtījuma blakus serviss. Lai kaut kā kontrolētu *SMS* apstrādi, sistēma saņems no *SMS* servera parametrus ar statusu.

2.2. Kāpēc sistēma tika izstrādāta no nulles, izmantojot ietvaru

IT risinājumu tirgū pastāv noteikts programmas pakešu daudzums, ar kuru palīdzību principā var realizēt doto sistēmu, bet autors tomēr nolēma realizēt sistēmu no nulles ar PHP ietvara un datu bāzes *MySQL* palīdzību:

- Dotā informācijas sistēma nav pārāk liela no izstrādes redzes viedokļa.
- Sistēma ir pietiekami specifiska savā veidā. Ja ņem par pamatu kaut kādu rēķinu apmaksas sistēmu, tad to ir stipri jāiestata un jāmaina zem klienta prasībām. Pagaidām nav arī tik viegli pielāgot paņemto rēķinu apmaksas sistēmu.
- Autoram ir liela darba pieredze ar PHP valodu.
- Projekts ir ierobežots finansiāli. Varētu mēģināt realizēt projektu, teiksim, ar *1C*, bet:
  - *1C* cenas kritērijs neapmierināja klientu.
  - projekta izstrādātājam nav darba pieredzes ar *1C*. Ja būtu nosacījums, ka sistēmu jārealizē tieši ar *1C*, tad izstrādātājs neuzņemtos to realizēt.
- Viss instrumentārijs ir bez maksas un klientam nevajag maksāt par gatavām programmatūras izmantošanas licencēm.

2.3. Projekta realizācijai izmantojamie resursi

2.3.1. Projekta izstrādātāju komanda

Dotajā projektā piedalījās:

- **Dizaineris** – dizainerim bija dots uzdevums uzzīmēt vienkāršu un ne spilgtu dizainu gaišos toņos, lai tas nekairinātu acis.

- **Metieris** – cilvēks, kurš dizainera uzzīmēto dizainu sametināja HTML formātā.
- **Programmētāji** – dotajā projektā piedalījās divi programmētāji.
- **Projekta vadītājs** – cilvēks, kurš komunicēja ar klientu, sagatavoja prasības sistēmai, izvēlējās izstrādes līdzekļus, novērtēja projektu naudas un laika ekvivalentā.

Darba autors tieši piedalījās šajā projektā kā projekta vadītājs, metieris un viens no sistēmas programmētājiem.

2.3.2. Projektā izmantojamās tehnoloģijas

Projektā pēc iespējas tika izmantotas bezmaksas tehnoloģijas un risinājumi tā realizācijai.

- **Mysql 5.0.51a** – visas informācijas uzglabāšanai, tiek izmantota datu bāze *MySQL*, jo tā ir pietiekami ātra un stabila.
- **PHP 5.3.3-7**
- **Apache 2.2.16**
- **Wkhtmltopdf 0.99** – ļoti lietderīga *UNIX* utilīta, kura konvertē jebkuru *HTML* dokumentu *PDF* formātā.
- **SMS serveris** – *SMS* serveri sniedz kompānija Esteria. Šo kompāniju izvēlējās klients, autoram arī bijusi darba pieredze ar šo kompāniju.
- **Yii framework 1.1.7** – tas ir nosacīti jauns, bet ļoti perspektīvs *PHP* ietvars. Visa loģika un viss kods ir uzbūvēts tajā. Novembra sākumā autors apmeklēja konferenci *WEB* tehnoloģiju jomā zem nosaukuma WebConf 2010, kura notika Latvijā. Viens no šīs konferences ziņotājiem bija Aleksandrs Makarovs (Voroņeža, Krievija), kurš ir viens no *Yii* ietvara izstrādātājiem, kurš sīki pastāstīja galvenos ietvara momentus.
- **SVN (subversion) –** brīva centralizēta versiju vadības sistēma, kuru 2004.gadā oficiāli izlaida kompānija *CollabNet Inc.* Sistēmas izstrādē piedalījās vairāki cilvēki, tāpēc ir jānodrošina versiju kontroli, lai katram projekta dalībniekam būtu svaiga projekta versija un viņš nevarētu nejauši aiztriept nepieciešamos failus vai informāciju tajos.
- **Mantis -** *Kļūdu sekošanas sistēma (angļu bug tracker system)* – programma, kuras mērķis ir palīdzēt programmatūras izstrādātājiem (programmētājiem, testētājiem uc), ieverot un kontrolēt kļūdas (*bugs*) atrastos programmā, ieverot un kontrolēt klientu priekšlikumus, ka arī sekot kļūdu novēršanas procesam un priekšlikumu izpildīšanas vai neizpildīšanas.
- **MySQL WorkBench 5.2.33 CE** – bezmaksas lietojumprogramma, kurā grafiskā veidā var noprojektēt datu bāzi. Pēc projektēšanas izveidoto bāzi varēs eksportēt uz *MySQL* datu bāzi. *MySQL WorkBench* ir neaizvietojama apjomīgu datu bāzi

projektēšanas programma, kur no pirmā acu uzmetiena viss šķiet saprotams, kādas tabulas, pa kādām atslēgām un kā saistītas. Attaisot bāzi šajā programmā, visi sakari kļūst acīmredzami. 2.1.punktā ir parādīts izstrādātās sistēmas ER modelis, kas ir sastādīta šajā programmā.

Ļoti populāra un ļoti izplatīta saikne tīmekļa lietojumprogrammas izstrādei ir *PHP, MySQL, Apache*. Pie tam visi komponenti ir *Open Source* projekti, kas pievieno treknu plusu šīs saites izvēlei.

3. YII IETVARS KĀ PROJEKTA REALIZĀCIJAS LĪDZEKLIS

3.1. PHP valoda

PHP (Hypertext Preprocessor – Hiperteksta Procesors) – tā ir plaši izmantojama scenāriju valoda ar atvērtu izejas kodu.

PHP – programmēšanas valoda, speciāli izstrādāta tīmekļa bāzētas lietojumprogrammas (skripti, scenāriji) izstrādei. PHP valodas scenāriji izpildās serverī, kurš atbalsta *PHP* programmēšanās valodu. Valodas sintakse ir ļoti līdzīga tādu valodu sintaksei, ka *C* un *Java*.

Scenāriji PHP valodā var tikt veidoti serverī atsevišķu failu veidā, vai arī var būt integrēti html lapās (3.1. att.).

```
<html>
<head> <title>Пример</title> </head> <body>
<?php echo "Hi, I'm a PHP script!"; ?> </body> </html>
```

3.1. att. **PHP valodas piemērs**

PHP scenārijs ievērojami atšķiras no scenārijiem *Perl* valodās tāpēc, ka tas ir izstrādāts speciāli tīmekļa lietojumprogrammas veidošanai, līdz ar to izstrādātājs „iebūvē" *PHP* kodu *HTML* lapās, nevis otrādi, kad raksta programmu ar vairākām komandām HTML izvadei. *PHP* kods ir iekļauts speciālajos sākotnējā un beigu tegos, kas ļauj tiem ieiet un iziet no „PHP režīmā".

PHP var ģenerēt un pārveidot ne tikai *HTML* dokumentus, bet arī dažāda formāta attēlus - *JPEG, GIF, PNG; PDF* un *FLASH* failus. *PHP* var veidot datus jebkurā teksta formātā, tai skaitā *XHTML* и *XML*.

PHP – kross-platformas tehnoloģija. *PHP* distributīvs der vairākām operētājsistēmām, tai skaitā *Linux, Unix* dažādas modifikācijas, *Microsoft Windows*, *Mac OS* un daudziem citiem. *PHP* atbalsta vairāki tīmekļa serveri, tādi ka, *Apache*, *IIS*(Microsoft Internet Information Server), *Microsoft Personal Web Server* un citi.

Vairākiem serveriem *PHP* valoda tiks piegādāta divos variantos – moduļa veidā un *CGI* preprocesora veidā.

PHP atbalsta darbu ar *ODBC* un vairākas datu bāzes: *MySQL, MSQL, Oracle, PostgreSQL, SQLite* un citus.

PHP programmēšanas valoda, it īpaši kopā ar populārāko datu bāzi *MySQL* – ir optimāls variants jebkuras sarežģītības tīmekļa mājas lapas veidošanai.

PHP valoda pastāvīgi pilnveidojas, un tā noteikti ilgi vēl dominēs tīmekļa programmēšanās valodu jomā [1].

3.1.1. Valodas evolūcija

3.1.1.1. PHP/FI

1994.gadā Rasmus Lerdorfs (Rasmus Lerdorf) dāņu programmētājs (pašlaik dzīvojošs Kanadā) ir izveidojis skriptu kopu uz *Perl/CGI*, kura apstrādā *HTML* dokumentu šablonus, viņa tiešsaiste rezumējumu apmeklētāju izvadei un uzskaitei. Lerdorfs nosauca kopu *Personal Home Page* (Privātā Mājas Lapa). Drīzumā *Perl* funkcionalitātes un ātruma — skriptu interpretētājs — kļuva par maz, un Lerdorfs, izmantojot *C* valodu, izstrādājis jaunu šablonu interpretētāju *PHP/FI*, (angļu val. Personal Home Page / Forms Interpreter — «Privātā Mājas Lapa / Formu interpretētājs»). 1997. gadā pēc ilgstošās beta-testēšanas ir iznākusi uzrakstītā *C-PHP/FI 2.0*. valodā apstrādātāja otrā versija. To ir lietojuši apmērām 1% (aptuveni 50 tūkstoši) no visiem pasaules interneta domēniem. [2]

3.1.1.2. PHP 3

PHP 3.0 versija tika ievērojami pārstrādāta, kā rezultātā tika noteikts programmēšanas valodas mūsdienīgs izskats un stils. 1997.gadā divi izraēliešu programmētāji Endijs Gutmans (Andi Gutmans) un Zēvs Suraskis (Zeev Suraski) pilnīgi pārrakstījuši interpretētāja kodu. Pēc 9 mēnešu publiskās testēšanās *PHP 3.0* oficiāli tika izlaists 1998.gada jūnijā. [2].

Viena no *PHP 3.0* stiprākām pusēm bija iespēja paplašināt kodolu ar papildus moduļiem. Vēlāk paplašinājumu uzrakstīšanas interfeisa piesaistījusi *PHP* valodai daudzus izstrādātājus, strādājošus pie saviem moduļiem, kas deva *PHP* valodai iespēju strādāt ar ļoti daudzām datu bāzēm, protokoliem, atbalstīt daudzus *API*. Liels izstrādātāju skaits sekmēja ātru valodas attīstību un tās popularitātes strauju pieaugumu.

Pilnīgi jauna programmēšanas valoda ir guvusi jaunu vārdu. Izstrādātāji atteicās no personālas pielietošanas papildinājuma, kura bija iekļautā *PHP/FI* abreviatūrā un valoda tika pārdēvēta par *PHP*.

1998. gada beigās *PHP* jau lietoja desmit tūkstošiem lietotāju. Simts tūkstošiem mājas lapu ziņojuši par *PHP* programmēšanas valodas lietošanu. Tajā laikā PHP 3.0 valoda tika uzstādītā aptuveni 10 % Interneta serveriem.

3.1.1.3. PHP 4

1998.gada ziemā, praktiski uzreiz pēc oficiālās *PHP 3.0* izlaides Endijs Gutmans un Zēv Suraskis uzsākuši *PHP* kodola pārveidošanu. Galvenais uzdevums bija sarežģīto aplikāciju veiktspējas palielināšana un bāzes koda modulitātes uzlabošana. Jauns dzinējs, dēvēts par *Zend Engine*, veiksmīgi ticis galā ar izvirzītiem uzdevumiem un pirmo reizi bija prezentēts 1999.gada vidū. Bāzēts uz šī dzinēja un ievadāms daudz jaunas funkcijas, *PHP 4.0*, oficiāli iznāca 2000. gadā maijā. Papildus uzlabotai veiktspējai *PHP 4.0* valodai bija vēl daži svarīgi jauninājumi, tādi, ka sesiju atbalsts, izvada buferizācija, drošāka lietotāja ievadāmas informācijas apstrādāšanās metode un vairākas jaunas valodu konstrukcijas.[2]

PHP 4 atjauninājumu izlaišanas pabeigšana tika ieplānota 2007. gadā beigās. Taču līdz 8. augusta 2008. gada tika izlaisti drošības kritiskie atjauninājumi. No 9 augusta 2008. gada *PHP 4* atbalsts bija pabeigts.

3.1.1.4. PHP 5

PHP piektā versija tika izlaista 13. Jūlija 2004 gadā. Izmaiņās iekļauj sevī *Zend* kodola atjaunošanu(Zend Engine 2), kas būtiski palielināja interpretatora efektivitāti. Tika ieviests *XML* valodas atbalsts. Pilnīgi pārstrādātas *OOP* funkcijas, kuri kļuvusi ļoti līdzīgi *Java* valodas modelei. Tika ievests destruktors, atvērtas, aizvērtas un aizsargātas locekļi un metodes, gala locekļi un metodēs, saskarnes un objektu klonēšana. Tomēr inovācijas tiek veiktas ar mērķi saglabāt lielāko saderību ar iepriekšējo versiju kodu. Šobrīd jaunāka stabila ir *PHP 5.3.6.* versija.

3.1.1.5. PHP 6

PHP sestā versijas joprojām atrodas izstrādāšanās stadijā kopš 2006 gadā oktobra. Tajā tiek veikti daudzi jauninājumi, tādi, ka, regulāro izteiksmju *POSIX* un „garus” superglobalus masīvus izslēgšana no kodola, direktīvu *safe_mode*, *magic_quotes_gpc* un *register_globals* izslēgšana no *php.ini* konfigurācijas faila. Ka arī daudz uzmanības tika pievērstas *Unicode* atbalstam[5].

3.1.2. Priekšrocības

PHP ir ļoti līdzīga *C* valodai, tāpēc profesionālajam programmētajam nesagādās grūtības to apgūt.

PHP valoda ir diezgan viegla apgūšanai un to var apgūt tīmekļa programmētājs, kurš pagaidām nepārzina citas programmēšanas valodas. *PHP* ir ļoti populāra programmēšanas valoda visā pasaulē, tāpēc sameklēt *PHP* izstrādātāju nebūs grūti.

PHP atbalsts ir kļuvis par standartu virtuālās mitināšanas kompānijās, līdz ar to virtuālā mitināšana priekš *PHP* ir salīdzinoši lēta.

PHP valoda ir bezmaksas valoda. Saite *PHP*, *Apache*, *MySQL*, kļuvusi par ļoti populāru bezmaksas tīmekļa lietojumprogrammas izstrādāšanas vīdi.

3.1.3. Trūkumi

Nesaskaņota funkciju sintakse un neortogonalitāte – valodas sintakse nav saskaņota, piemēram, daļa no funkcijām priekš darba ar masīviem sākas ar prefiksu *array_*, citai daļai šī prefiksa nav. Daļas no rindas funkciju nosaukumiem sākas ar prefiksu *str*, citām funkcijām šāda prefiksa nav. Tajās pašās rindas funkcijās apstrādājamo rindu var nodot gan kā pirmo, gan arī kā pēdējo argumentu, kas sajauc programmētāju prātus un tādējādi prasa pastāvīgu dokumentācijas izpēti.

Atgriezeniskās savienojamības trūkums starp valodas versijām – kods, kas izveidots priekš agrākām valodas versijām, nestrādā vai strādā nekorekti ar vēlākām valodas versijām

Daudzu baitu kodējumu trūkums valodas kodolā – rindu uzturēšana ar daudzu baitu kodējumiem, tādiem kā *UTF-8*, tiek realizēta caur *mbstring* paplašinājumu.

„Rīcības brīvība” – *PHP* valoda nav pārāk izvēlīga pret funkciju nosaukumiem, pasludināšanu un mainīgo inicializāciju, klašu mainīgo pasludināšanu, klašu izveidi utt. No vienas puses, tā ir priekšrocība, laika un koda rindu ekonomija, kā arī iesācējam ir viegli sākt programmēt *PHP* valodā. Diemžēl rīcības brīvības dēļ tīmekļa lietojumprogramma ir sarežģītāka, iesācēja rokās var pārvērsties tādā putrā, kurā pašam programmētājam būs grūti tikt skaidrībā.

3.2. PHP ietvari

Ietvars (angl. *framework* — karkass, struktūra, darbības joma) – informāciju tehnoloģijās tā ir programmas sistēmas struktūra; programmatūra, kas atvieglo liela programmas projekta izstrādi un dažādu komponentu apvienošanu. Atšķirībā no bibliotēkām, kuras apvieno sevī pēc funkcionalitātes līdzīgas apakšprogrammas, ietvars satur sevī lielu skaitu pēc nozīmes dažādu bibliotēku. Lieto arī vārdu **ietvars,** bet daži autori vispār izmanto to kā galveno [6].

3.2.1. 3 pieejas WEB pielikuma izstrādei

Sākot izstrādāt tīmekļa lietojumprogrammu tīrā PHP, izstrādātājs saņem standartus *PHP* instrumentus (standartas PHP funkcijas), kā arī papildus moduļus, kurus var uzstādīt *PHP* instalācijas laikā. Tas nozīmē, ka:

- visu projekta failu struktūru jums nāksies izdomāt pašiem
- datus, kuri ienākt datu skriptā, jums nāksies apstrādāt un pārbaudīt pašam vai meklēt kaut kādas blakus palīgklases. Piemēram, apskatīsim no formas iegūto datu apstrādi. No formas iegūto datu apstrāde var, piemēram, izskatīties sekojoši:
  - pārbaudīt iegūto lielumu tipu;
  - pārliecināties, ka nozīmēm ir pieļaujamie lielumi (vai formāts);
  - dzēst no teksta aizliegtus tagus (aizsardzība no *XSS*);
  - *SQL* pieprasījumu parametros dienesta simbolus nomainīt pret to *escape* secīgumu (aizsardzība no *SQL Injection*);
  - ja parametri saistās savā starpā, pārbaudīt šīs saiknes;
  - utt.
- Tāpat formu pārbaudes, informācijas bāzē pievienošanas kodam, informācijas izvēles un izvades mehānismam jābūt ne tikai rakstītam**,** bet arī notestētam**.** Pie kam vairumā gadījumu tās ir vienveidīgas operācijas, kuras atkārtojas praktiski visos pielikumos.
- Paša projekta arhitektūru nāksies izdomāt pastāvīgi

Protams, šim kodam jābūt ne tikai rakstītam**,** bet arī notestētam**.** Kā redzat, tas ir darbietilpīgs process. Pie kam vairumā gadījumu tās ir vienveidīgas operācijas, kuras atkārtojas praktiski visos pielikumos.

Uz doto brīdi tīmekļa lietojumprogrammas izveidei ir trīs pamatpieejas (vai to kombinācijas)[7]:

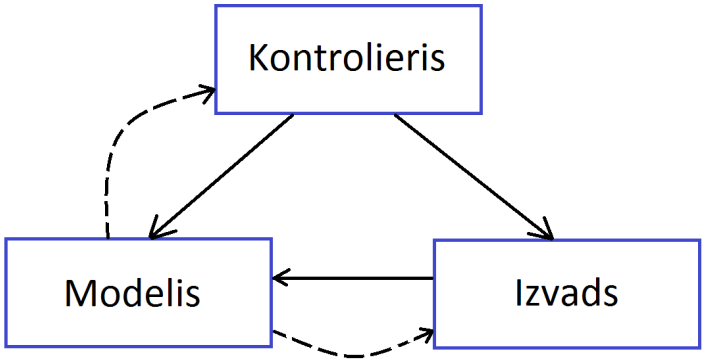
1. Izmantot „tīro” *PHP* un standartveida un papildus bibliotēkas. Šis ir pats darbietilpīgākais variants, tomēr vienlaicīgi pats elastīgākais. Jūs varat realizēt praktiski jebkādas funkcijas un vienlaicīgi nodrošināt maksimālu ražīgumu. Tiesa, ir viena nianse. Labu produktu jūs iegūsiet tikai pēc pielikuma testēšanas un optimizācijas, bet tas nav tik viegls process kā šķiet no pirmā acu uzmetiena**.**
2. Izmantot gatavu risinājumu. Uz doto brīdi priekš visiem plaši izplatītiem mājas lapu tipiem pastāv gatavi risinājumi. Piemēram, *WordPress*, *Joomla*, *PHPbb* un daudzi citi. Te var vispār iztikt bez programmēšanas, jo šie risinājumi ir veidoti kā gatavas mājas lapas

(blogi, portāli, forumi u.tml.). Jums jāveido tikai kontents. No šejienes arī nosaukums *CMS* (*angl., Content Management System*) – satura vadības sistēmas. Ja ar standarta funkcijām nepietiek, tad var uzrakstīt spraudņi (vai atrast gatavu). Principā daudzas šādas sistēmas nodrošina labu ražīgumu, bet tikai tajos uzdevumos, priekš kuriem tie sākotnēji tika projektēti. Tas nozīme, kā jūs varat pievienot jums nepieciešamās funkcijas, bet vienlaicīgi ražīgums (resursu patēriņš) var būt daudz sliktāks nekā pirmajā variantā.

3. Izmantot ietvaru (framework) - principā ietvaru var uzskatīt par papildus bibliotēku. Tomēr pastāv būtiska atšķirība. Jūs izmantojat bibliotēku pielikuma funkciju paplašināšanai. Savukārt ietvars papildus nosaka lietojumprogrammasarhitektūru (savstarpējās saiknes starp komponentiem). Ja izmanto analoģijas ar mājas būvniecību, tad par bibliotēkām var uzskatīt ķieģeļus, logu un durvju blokus, bet par ietvaru – pamatus un nesošās sienas (zinu, salīdzinājums ir pietiekami dziļš, tomēr parāda būtību). Principā ietvara izmantošana – tas ir kaut kas vidējs starp pirmo un otro variantu. No vienas puses, rīcības brīvība tiks ierobežota salīdzinājumā ar pirmo variantu, tomēr šie ierobežojumi ir nebūtiski salīdzinājumā ar gataviem risinājumiem.

Principā programmētājs izmanto visas šīs trīs pieejas. Galvenais ir noteikt, kāda pieeja visvairāk piemērota konkrētā uzdevuma risināšanai. Piemēram, gatavais risinājums būtiski ekonomē laiku un izdevumus, izstrādājot vienkāršu mājas lapu ar standartu funkciju kopumu. Izstrādājot kaut kādu nopietnu datu bāzes vadības sistēmu (piemēram, klientu uzskaites sistēma, rēķinu sistēma), ir vērts padomāt par sistēmas izstrādi ar kaut kāda ietvara palīdzību.

Šajā kursā mēs apskatīsim ietvaru, kas atbilst *MVC* (*angļ., Model – View - Controller*) arhitektūrai. Zemāk (sk. att. 3.2.) ir parādītas saites starp šiem komponentiem.



3.2. att. **Saites starp komponentiem**

- Kontrolieris ir vadības centrs. Tas saņem pieprasījumus no pārlūkprogrammas un apstrādā tos**.**
- Modelis tiek izmantots darbam ar datiem (piemēram, datu lasīšanai/ierakstam/atjaunošana DB u.tml.**).**
- Izvads ģenerē *HTML* lapu, kura tiek nosūtīta pārlūkprogrammai.

No diagrammas (sk. att. 2.1.) redzams, ka modelis neko nezina par pārējiem komponentiem, bet izvads nezina par kontrolieri. Tādējādi arhitektūra ļauj sadalīt mājas lapas arhitektūru trijos galvenos posmus.

3.2.2. Populārāko ietvaru apskats

3.2.2.1. Zend Framework

Šis ir brīvais ietvars tīmekļa lietojumprogrammas vai tīmekļa servisu izstrādei. *Zend* cenšas sekot *PHP* garam, piedāvā vienkāršus interfeisus un spēcīgas funkcijas pielikumu izstrādei. Tas piedāvā paplašinājumus mūsdienīgām, ātrām un drošām mājas lapām.

Ietvars balstās uz *MVC* idejām. Tiek izstrādāts ar kompāniju *Zend*, kas ir pašas *PHP* izstrādātājs.

Bez *MVC* komponentiem *Zend Framework* satur daudzas bibliotēkas, kuras ir noderīgas lietojumprogrammas uzbūvei. Eksistē arī komponenti integrācijai ar blakus servisiem, piemēram, *YouTube*, *del.icio.us* un daudziem citiem. Sākot ar 1.6 versiju, tiek piegādāts ar *JavaScript* ietvaru *Dojo*, kā arī ietver sevī komponentus darbam ar to . [8]

Zend Framework ir pats nevainojamākais kods. Ietvars ir ļoti elastīgs. Prasa labas *PHP* un *OOP* zināšanas. Nāksies nedaudz pieskaņot pirms izmantošanas. Sausa, bet pietiekami pilnīga tehniskā dokumentācija.

3.2.2.2. CakePHP

Programmas ietvars tīmekļa lietojumprogrammas izveidei, kas uzrakstītas *PHP* valodā un balstās uz atvērtas programmatūras principiem. *CakePHP* arī balstās *MVC* projektēšanās šablonu.

Sākotnēji ietvars veidots kā populārā *Ruby on Rails* klons, un daudzas idejas tika paņemtas tieši no turienes [9].

1. Sava failu struktūra
2. Spraudņu kopas uzturēšana
3. Datu abstrakcija (*PEAR::DB*, *ADOdb*, un pašu *Cake* izstrāde)
4. *DBVS* kopas uzturēšana (*PostgreSQL*, *MySQL*, *SQLite*, *Oracle*)

3.2.2.3. Code Igniter

CodeIgniter — ietvars, kas domāts pilnvērtīgu tīmekļa sistēmu un aplikāciju izstrādei *PHP* valodā. Izstrādāts ar kompāniju *EllisLab*, kā arī Riku Elisu (Rick Ellis) un Polu Burdiku (Paul Burdick).

Daudzi *PHP* programmētāji *CodeIgniter* uzskata par piemērotu izvēli tiem, kas tikko sāka izmantot tīmekļa ietvarus savā darbā. *CodeIgniter* izmanto kā kodolu priekš jaunās *CMS ExpressionEngine* komercversijas.

Kā šablona sistēmu *CodeIgniter* izmanto gan lielāko daļu mūsdienu *CMF(angļ., Content Management Framework)*, kas uzrakstīti *PHP* valodā, gan arī pašu *PHP*. Tomēr pazīstami arī risinājumi ar alternatīvu šablonu pieslēgšanu, tādiem kā Smarty vai TinyButStrong [10].

Code Igniter ir arī lieliska dokumentācija, tas ir ļoti elastīgs. Pie nepieciešamības ir viegli izmantot blakus kodu.

3.2.2.4. Kohana

Kohana, agrāk saucās par *Blue Flame*, un tas ir *PHP5* tīmekļa ietvars ar atvērtu kodu, kurš izmanto arhitektūras modeli *MVC*. Tā mērķi – būt drošam, vieglam un vienkāršam izmantošanā.

Kohana projekts tika veidots kā *PHP* ietvara *CodeIgniter* atzarojums zem nosaukuma *Blue Flame*. Galvenais iemesls bija pāreja uz sabiedrībai atvērtāku izstrādes modeli, jo daudzus lietotājus neapmierināja izstrādes ātrums un kļūdu labojums *CodeIgniter*. Riks Elliss — *CodeIgniter* izveidotājs un īpašnieks — bija laimīgs redzēt sava projekta atzarojumu, bet palīdzēt atteicās; viņš pamudināja jaunu projektu uz savas dokumentācijas izveide (tamdēļ *Kohana* ir trūcīgāka dokumentācija, nekā *CodeIgniter*) un ieteica pārdēvēt projektu. 2007.gada jūlijā *Blue Flame* tika pārdēvēts par *Kohana*, lai nākotnē izvairītos no problēmām ar autortiesībām.

Nosaukums *Kohana* tika izvēlēts, kad izstrādātāji sāka pārskatīt dzimto amerikāņu vārdus, lai izvēlētos vārdu, kas nepārkāptu autortiesības. *Kohana Sui* valodā nozīmē „ātrs”. Japāņu

valodā tam ir arī nozīme „mazais zieds", ukraiņu – „mīļā" un pašas pazīstamākās lauku bezdelīgas vārds (Kohana) — visām šīm nozīmēm nav nekāda sakara ar nosaukumu [11].

3.2.2.5. Symfony

Symfony — brīvs ietvars, kas uzrakstīts *PHP5* valodā un izmanto projektēšanās šablonu MVC.

Symfony piedāvā ātru tīmekļa lietojumprogrammas izstrādi un vadību, ļauj viegli risināt tīmekļa programmētāja viendabīgus uzdevumus. Strādā tikai ar *PHP 5* (>=5.2.4 un vēlams ne 5.2.9). Uztur daudzas datu bāzes (*MySQL*, *PostgreSQL*, *SQLite* vai jebkura cita *PDO*-savienojama ar *DBVS*). Informācijai par relācijas datu bāzi projektā jāsaistās ar objektu modeli. To var izdarīt ar *ORM* instrumenta palīdzību. *Symfony* tiek piegādāts ar diviem no tiem: *Propel* un *Doctrine*.

Symfony nemaksā neko un tiek publicēts zem MIT licences. Projektu sponsorē franču kompānija *Sensio* [12].

3.2.3. Salīdzinājums

Zemāk (3.1. tabula) uzskatāmā veidā ir parādīta augstāk aprakstīto ietvaru salīdzinošā tabula [13].

3.1. tabula

Ietvaru salīdzinājums

	Zend Framework	**CakePHP**	**Code Igniter**	**Kohana**	**Symfony**
Nepieciešamais zināšanu līmenis	PHP5, OOP, projektēšanas šabloni.	PHP, OOP, prasme tikt galā ar ietvara sākotnējo kodu.	PHP, OOP pamati	PHP5, OOP	PHP5, OOP, ORM, konsole.
Paredzamie projekti	Vidēji – lieli	Mazi – vidēji	Mazi – lieli	Mazi – lieli	Lieli
PHP4	Nē	Jā	Jā	Nē	Nē
PHP5	Jā	Jā	Jā	Jā	Jā
Stingra katalogu struktūra	Nē (rekomendācijas)	Jā	Jā	Jā	Jā
Oficiāls internacionalizācijas	Jā	1.2 versijā	Jā	Jā	Jā

atbalsts					
Uzstādīšanas un iestatīšanas sarežģītība	Augsta	Zema	Zema	Zema	Augsta
Prasa iestatīšanu	Daudz	Nedaudz	Nedaudz	Nedaudz	Jā
Pilns ORM atbalsts	Nē	Jā (ne pārāk ērts)	Nē (var izmantot Doctrine)		Jā (Propel, Doctrine)
Dokumentācija un piemēri	Laba	Ir	Lieliska	Laba	Lieliska
Unit-testi priekš ietvara sākotnējā koda	Jā	Jā	Nē		Jā
Angliska biedrība	Jā	Jā	Forums, Wiki, tutoriāli, blogi	Forums, blogi	Jā
Krievu biedrība	Jā	Jā	Dokumentācija, forums, blogi	Nē	Jā
Licence	New BSD	MIT	Sava	BSD-style	MIT

3.3. Yii ietvars

3.3.1. Kas ir Yii

Yii — tas ir ļoti efektīvs, uz komponentu struktūru balstīts PHP ietvars tīmekļa lietojumprogrammas izstrādei. Tas ļauj maksimāli pielietot koda atkārtotas izmantošanas koncepciju un var būtiski paātrināt tīmekļa izstrādes procesu. Nosaukums Yii *(izrunā kā Yee vai [ji:])* nozīmē vienkāršs (*easy*), efektīvs (*efficient*) un paplašināms (*extensible*).

Jūs, iespējams, pazīstat autorus (*Qiang Xue, Xiang Wei Zhuo*) pēc pietiekami elastīga, bet pietiekami lēna ietvara *Prado*.

Yii ietvara dibinātājs ir *Qiang Xue*. Ietvars parādījās 2008.gada 1.janvārī. *Qiang* iepriekš izstrādāja un uzturēja *Prado* ietvaru. 2008.gada 3.decembrī, pēc aptuveni viena izstrādes gada, *Yii 1.0* tika oficiāli palaists masveida lietošanā. Tā ārkārtēji iespaidīgi ražīguma rādītāji salīdzinājumā ar citiem *PHP* uzreiz piesaistīja ļoti pozitīvu uzmanību, un tā popularitāte un ieviešana turpina attīstīties arvien ātrākos tempos.

3.3.2. Ideju aizgūšana

Yii ietver sevī daudzas idejas un citu, pazīstamu tīmekļa ietvaru un lietojumprogrammas darbus. Zemāk ir parādīts īss tā saraksts, ar ko līdzinās *Yii*:

- **Prado** - galvenais ideju avots priekš *Yii* ietvara. *Yii* mantoja komponentu struktūru, notikumu programmēšanu, *DBVS* abstrakcijas kārtu, modulitāti, internacionalizāciju, kā arī daudzas citas savas funkcijas un šablonus.
- **Ruby On Rails** - *Yii* aizguva Convention *over configuration* (var sākt strādāt, bet konfigurēt – pie nepieciešamības). *Yii* arī aizguva projektēšanas šablona *active record* implementāciju tā *ORM* kārtai.
- **JQuery** - integrēta ar *Yii* kā pamata *JavaScript* bibliotēka.
- **Symfony** - *Yii* aizguva filtru dizainu un spraudņu arhitektūru.
- **Joomla** - *Yii* aizguva moduļu projektejumu un ziņojumu tulkošanas shēmu.

Protams, dokumentācija nav tik liela vienkārša un liela kā *CodeIgniter*, jo pats ietvars mazliet sarežģītāks, taču lasot to, vienalga viss ir saprotams. Internetā ir pieejams arī pilnīgs tulkojums krievu valodā.

3.4. Prasības

Uz *Yii* pamata uzbūvētu tīmekļa lietojumprogrammas palaišanai jums būs nepieciešams tīmekļa serveris ar *PHP* versijas 5.1.0 un augstāk.

Izstrādātājiem, kuri vēlas izmantot *Yii*, ārkārtēji lietderīgi būs izprast objekt-orientētās programmēšanas (OOP) koncepciju, jo *Yii* – tas ir stingri objekt-orientēts ietvars.

3.5. Kam Yii būs labākā izvēle?

Yii — tas ir ietvars vispārējās nozīmes programmēšanai, ko var izmantot praktiski jebkuru web pielikumu izstrādei. Pateicoties savam vieglumam un moderniem kešēšanas līdzekļiem, Yii ir īpaši piemērots pielikumu izstrādei ar lielu trafika plūsmu, tādiem kā portāli, forumi, kontenta vadības sistēmas (CMS), elektroniskās komercijas sistēmas u.c.

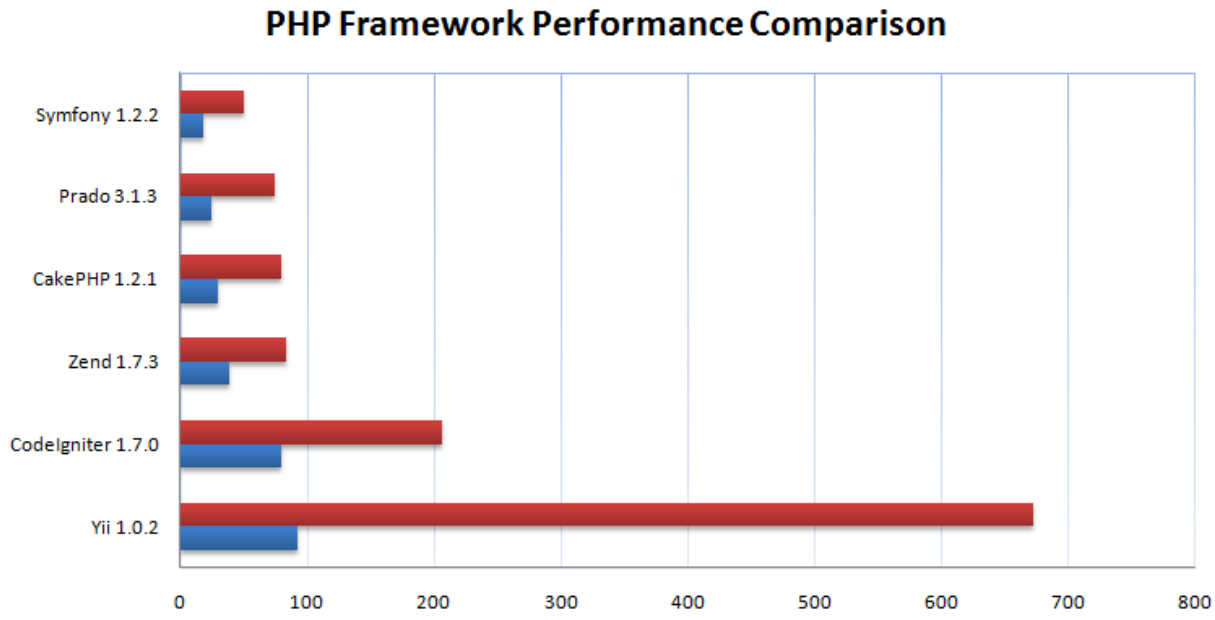
3.6. Yii salīdzinājumā ar citiem ietvariem

Līdzīgi lielākai daļai citu *PHP* ietvaru, *Yii* ir *MVC* ietvars. *Yii* priekšorcības salīdzinājumā ar citiem ietvariem slēpjas tā efektivitātē, plašās iespējās un kvalitatīvā dokumentācijā. Sākotnēji *Yii* noprojektēts ļoti rūpīgi, lai atbilstu visām prasībām nopietnas tīmekļa lietojumprogrammas izstrādes laikā. *Yii* nav ne kaut kāda projekta blakus produkts, ne

blakus risinājums kopa. Tas ir lielas autoru pieredzes tīmekļa programmu izstrādē rezultāts, kā arī populārāko tīmekļa ietvaru un lietojumprogrammu pētījumu rezultāts [14].

3.7. Ražīgums

Yii ir ļoti ražīgs ietvars. Zemāk attēlotais grafiks rāda, cik efektīvs ir *Yii* salīdzinājumā ar citiem populāriem *PHP* ietvariem (skat. att. 3.3). Grafikā *RPS(angļ., requests per second)* atšifrē kā „pieprasījumi sekundē", apraksta, cik pieprasījumu sekundē var apstrādāt pielikums, kas uzrakstīts uz ietvara pamata. Jo lielāks ir šis skaitlis, jo efektīvāks ir ietvars. Kā redzams, Yii pēc ražīguma pārspēj visus citus ietvarus. Ražīgums – tā ir Yii priekšrocība, īpaši tad, kad ieslēgts APC paplašinājums [15].



	Yii 1.0.2	CodeIgniter 1.7.0	Zend 1.7.3	CakePHP 1.2.1	Prado 3.1.3	Symfony 1.2.2
RPS with APC	673	206	83	79	75	50
RPS without APC	93	79	38	29	24	18

3.3.att. **Ietvaru ražīguma salīdzinājums**

3.7.1. Kāpēc Yii ir tik ātrs

Yii ātrums ir liels, jo tas plaši izmanto „slinkas" ielādes tehniku. Piemēram, tas neietver klases failu, kamēr klase netiks izmanto pirmo reizi, un neveido objektu, kamēr objektu neizsauca. Citu ietvaru ražīgums daļēji ir mazs tamdēļ, ka tie ietver inicializāciju (tas ir, lietotāja sesijas, savienojums ar datu bāzi), neatkarīgi no tā, vai to izmantos pieprasījuma laikā, vai nē.

3.7.2. Izvēlētā lietotne ražīguma testam

Tā kā mērķis ir salīdzināt minimālo slodzi uz katru ietvaru, testa lietojumprogrammai jābūt vienkāršai. Šajā nolūkā tika izvēlēts teksta „*Hello world*” izvads uz ekrāna ar komandas *echo* palīdzību. Visas ietvara papildus iespējas (piemēram, sesijas) tika ietvertas salīdzinājuma taisnīguma nodrošināšanai.

3.7.3. Kāpēc "Hello World"

Tika nolemts testēt uz lietotne "*Hello World*", galvenokārt, lai panāktu noteikto mērķi, tas ir, lai noskaidrotu minimālās slodzes uz ietvariem. Daudzi iebildīs, ka "*Hello World*" pielikumam nav jēgas, jo reāliem pielikumiem ir saskarsme ar sarežģītākiem uzdevumiem, tādiem kā pieprasījumi uz datu bāzi. Tas nav pilnībā patiesi. Patiesībā, īpaši liela mēroga Web 2.0 lietojumprogrammās, mēs bieži sastopamies ar scenārijiem, kas ir ļoti tuvi "*Hello World*". Piemēram, lietojumprogrammai ir jāatbild uz AJAX pieprasījumu, kam jāatgriež tekošo datumu; lielākā lapas daļa tiks kešēta un pielikumam vienkārši vajadzēs izņemt saglabātos datus un parādīt tos.

Cits iemesls, kāpēc tika izvēlēts tests uz „*Hello world*”, ir tāds, ka gadījumā, ja tiek salīdzinātas sarežģītākas iespējas (teiksim, pieprasījumi uz datu bāzi), kļūst sarežģīti pārliecināties par salīdzinājuma taisnīgumu. Piemēram, viens ietvars var ļoti efektīvi pildīt pieprasījumus uz datu bāzi, bet tam trūkst kešēšanas shēmas; savukārt cits ietvars daudz lēnāk strādā ar datu bāzi, bet tam ir ļoti stipra kešēšanas shēma ražīguma atvieglošanai. Ir ļoti daudz ietekmējošo faktoru taisnīgu ražīguma testu veikšanai.

3.7.4. Testēšanas instruments un vide

RPS rādītāji tika iegūti ar instrumenta *ApacheBench* palīdzību ar komandu "*ab –t30 –c10 URL*" (tas ir, paralēlisma līmenis 10, testa izpilde 30 sekunžu laikā). Lai iegūtu precīzu rādītāju priekš katra ietvara, pirms katras testa palaišanas tika apturēts *apache web* serveris, bet tad palaists no jauna. Testa piemērs pie katra ietvara tika palaists vairākas reizes, lai „uzsildītu” testēšanas vidi. Testēšanas vide bija sekojoša[15]:

- **Operētājsistēma:** Red Hat Enterprise Linux Server release 5.2
- **Web serveris:** Apache httpd 2.0.40
- PHP: 5.2.6, any non-essential extensions are disabled
- **CPU**: Dual Intel Xeon 3.2GHz
- **OII**: 2GB
- **Cietais disks**: 73GB 15K RPM SCSI/SAS HDD

APC.ini iestatījumi sekojoši:

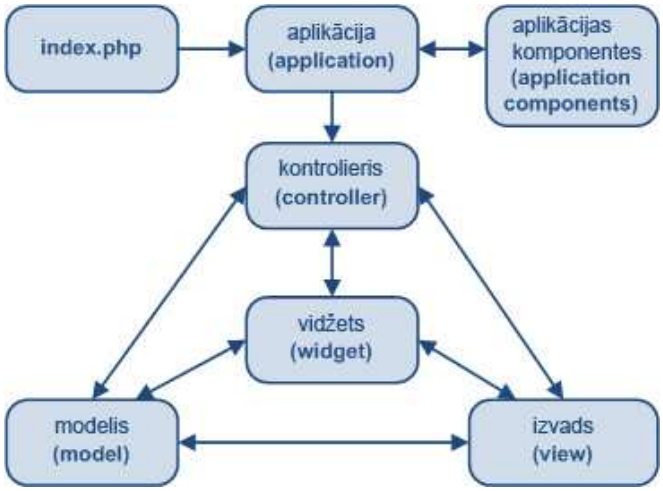
```
apc.enabled=1
apc.shm_segments=1
apc.optimization=0
apc.shm_size=32
apc.ttl=7200
apc.user_ttl=7200
apc.num_files_hint=1024
apc.mmap_file_mask=/tmp/apc.XXXXXX
apc.enable_cli=1
apc.cache_by_default=1
apc.stat=0
```

3.8. Modelis-Izvads-Kontrolieris (MVC)

Yii izmanto projektēšanas šablonu Modelis-Izvads-Kontrolieris (*MVC*, *Model-View-Controller*), kas tiek plaši pielietots programmēšanā.

MVC ir virzīts uz biznesa loģikas atdalīšanu no lietotāju interfeisa, lai izstrādātāji varētu viegli mainīt atsevišķas lietojumprogrammas daļas, neskarot citas. *MVC* arhitektūrā modelis sastāv no datiem un biznesa loģikas noteikumiem, izvads, jeb skats, atbild par lietotāja interfeisu (piemēram, teksts, ievades lauki), bet kontrolieris nodrošina mijiedarbību starp modeli un izvadu.

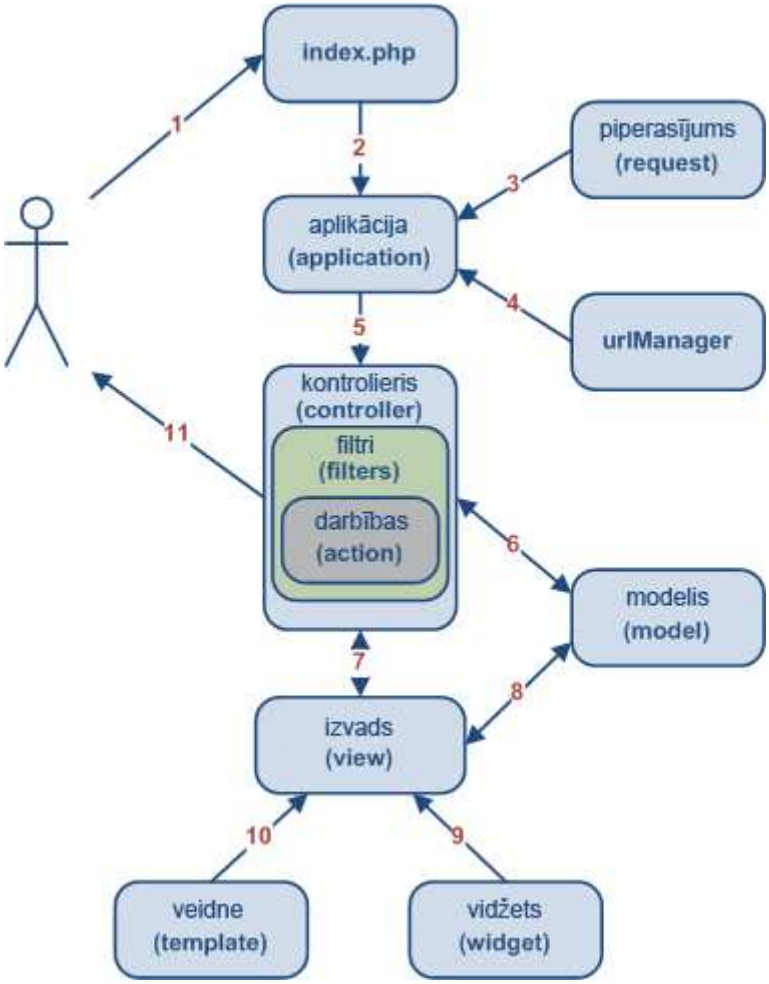
Yii arī izmanto priekšējo kontrolieri, ko sauc par aplikāciju (*angļ.*, *application*), kurš kalpo par pieprasījuma izpildes kontekstu. Aplikācija veic lietotāja pieprasījuma apstrādi un nodot to turpmākai apstrādei atbilstošam kontrolierim (3.4.att.).



3.4.att. **Yii pielikuma struktūra**

3.8.1. Yii lietojumprogrammas tipisks darba secīgums

Nākamā diagramma apraksta lietotāja pieprasījuma apstrādes procesa secīgumu pielikumā (3.5.att.).



3.5.att. **Lietotāja pieprasījuma apstrādes process**

1. Lietotājs ievada pārlūka, piem.: http://www.example.com/index.php?r=post/show&id=1, bet tīmekļa serveris apstrādā to, palaižot inicializācijas skripta *index.php* izpildi;
2. Inicializācijas skripts veido lietojumprogrammas eksemplāru un palaiž to izpildei;
3. Lietojumprogramma saņem sīku informāciju par lietotāja pieprasījumu no pielikuma komponenta *request;*
4. Lietojumprogrammu nosaka pieprasītais kontrolieris un darbība ar komponenta *urlManager* palīdzību. Dotajā piemērā kontrolieris saucās *post*, kas attiecas pie *PostController* klases, bet darbība — *show*, kuras būtību nosaka kontrolieris;

5. Lietojumprogrammas veido pieprasītā kontroliera eksemplāru lietotāja pieprasījuma turpmākai apstrādei. Kontrolieris nosaka darbības *show* atbilstību *actionShow* metodei kontroliera klasē. Tālāk tiek veidoti un pielietoti filtri (piemēram, *access control*, *benchmarking*), kuri saistās ar doto darbību, un, ja filtri atļauj, tad darbība tiek izpildīta;
6. No datu bāzes, darbība nolasa modeli *Post* ar *ID*, kas vienāds ar 1;
7. Darbību veido izvads *show* ar modeļa *Post* datiem;
8. Izvads saņem un atspoguļo modeļa *Post* atribūtus;
9. Izvads izpilda dažus vidžetus(*angļ., widgets*);
10. Izveidotais skats tiek pievienots lapas maketā;
11. Darbību pabeidz izvada formēšana un izvada rezultātu lietotājam.

3.8.2. Izstrādes process

Zemāk ir aprakstīts vispārējs tīmekļa lietojumprogrammas izveides process, izmantojot ietvaru. Process paredz, ka prasību analīze jau ir veikta, tāpat kā nepieciešamā pielikuma uzbūves analīze.

1. Direktoriju struktūras izveide. Utilītprogrammu *yiic* var izmantot, lai paātrinātu šo procesu;
2. Lietojumprogrammas konfigurēšana ar konfigurācijas faila palidzību. Šis posms var prasīt arī uzrakstīt dažus lietojumprogrammas komponentus (piemēram, lietotāju vadības komponents);
3. Modeļa klases izveide katram izmantojamam datu tipam. Visu jūs interesējošo *active record* modeļu automātiskai ģenerācijai var izmantot *Gii* instrumentu;
4. Kontroliera klases izveide katram lietotāju pieprasījuma tipam. Lietotāju pieprasījumu klasifikācija ir atkarīga no tekošajām prasībām. Vispārējā ganījumā, ja modeļa klasi lietotājs jau izmanto, jāpastāv atbilstošai kontroliera klasei, *Gii* utilītprogramma var automatizēt šo procesu;
5. Darbību un to skatu izveide. Tieši šeit tiek pildīts pamatdarbs;
6. Nepieciešamo filtru konfigurācija darbībām ar kontrolieru klasēm;
7. Noformēšanas tēmas izveide pēc nepieciešamības;
8. Ziņojumu tulkošana gadījumā, ja nepieciešama lokalizācija;
9. Datu un skatu identifikācija ar mērķi kešēt tos, pielietojot nepieciešamu kešēšanas tehniku.
10. Ražīguma iestatīšana un attīšana.

Katrā no attēlotajiem posmiem var būt nepieciešams veidot un pielietot testus.

3.9. Ieviešanas

- http://www.stay.com/ - tūrisma pakalpojumi internetā. Ļoti liels projekts. Ietver sevī viesnīcas izvēlētajā pilsētā, kā arī to rezervāciju un apmaksu. Mājas lapā ir iespēja bezmaksas apskatīties tūrisma maršrutu noteiktā pilsētā.
- http://www.realself.com/ - mājas lapa, kura saistās ar kosmētiskām operācijām uz cilvēka ķermeņa. Šeit var apskatīties fotogrāfijas pirms un pēc, palasīt atsauksmes, uzzināt operāciju cenas, apspriest vai uzdot jautājumu par kaut kādu interesējošo tēmu
- http://www.znno.com/ - biznesa katalogs.
- http://www.chive-project.com/ - *Chive* ir *MySQL* datu bāzes vadības sistēma caur *http* protokolu. Sistēma uztur lielāko daļu vajadzīgo operāciju un komandu, kas nepieciešami izstrādātājiem un sistēmas administratoriem (datu bāzes, tabulas, indeksi, atslēgas, trigeri, izvades, procedūras, vadības privilēģijas, imports/eksports).
- http://www.awx.co.za – šī kompānija atrodas Dienvidamerikā un piedāvā videospēles internetā. Īss mājas lapas funkcionālo iespēju apraksts: forums, čata logs, aptauju sistēma, privātie ziņojumi
- http://www.costaricaemfoco.com.br/ - Ziņu portāls (Kosta Rika, Brazīlija).
- http://tvoja4zida.net – Nekustamā īpašuma pārdošana/pirkšana Polijā.

4. PROJEKTA IZSTRĀDES PROCESS

Šajā nodaļā autors apraksta informācijas sistēmas izstrādes posmus un procesus.

4.1. Sistēmas izstrādes posmi

Sistēmas izstrādes posmi sastāvēja no vairākiem soļiem:

1) Projekta vadītājs vairāku interviju gaitā ar uzņēmuma vadītāju sastādīja nelielu tehnisko uzdevumu, kurš sastāvēja no sākotnējām un pamata prasībām pret sistēmu. Tehniskais uzdevums tika uzrakstīts brīvā veidā, neizmantojot specifisku terminoloģiju, bet rēķinoties ar to, lai klients saprastu, par ko ir runa un kāds funkcionāls tiks īstenots sistēmā. Faktiski tā bija biznesa prasību specifikācija, tomēr detalizētākā līmenī, kur tika ņemtas vērā tādas lietas kā:
    - kādiem laukiem jābūt
    - kādiem laukiem jābūt obligātiem
    - kādam atskaitēm jābūt sistēmā

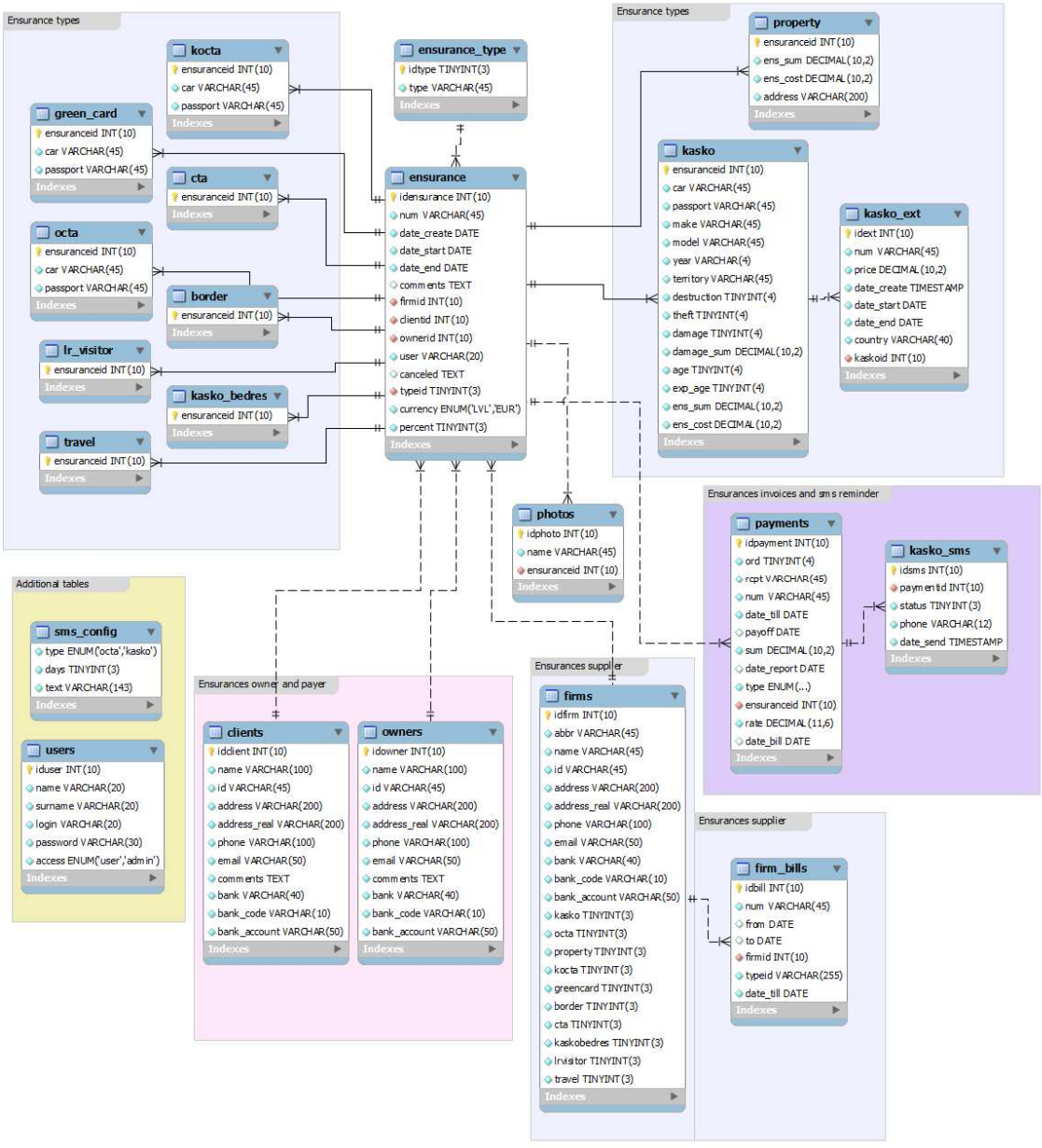
      utt.
2) Pēc tam tehniskais uzdevums tika koriģēts ar uzņēmuma vadītāju.
3) Uz tehniskā uzdevuma pamata dizaineris izgatavoja sistēmas galveno lappušu attēlus, pēc kurām varēja saprast, kādos toņos tiks izpildīta sistēma, kāda izmēra būs ievadāmie lauki, kā tiks nokomponēta pamatinformācija.
4) Tālāk notika mājas lapas dizaina saskaņošanas posms.
5) Pēc saskaņošanas visas skices tika nodotas metierim, kurš sametināja no attēla *HTML* veidnes.
6) *HTML* veidnes tika nodoti sistēmas programmētājiem, kuri savukārt noprojektēja ER sistēmas modeli un uztaisīja datu bāzi. Programmētāji sāka darbu ar sistēmu.
7) Tikko kādi sistēmas moduļi bija gatavi, tie uzreiz tika atrādīti pasūtītājam, kurš savukārt testēja tos.
8) Tajā laikā, kad pasūtītājs testēja moduļus, programmētāji realizēja sekojošu funkcionālu.
9) Ja testējamos moduļos atradās kļūdas, tad viens no programmētājiem strādāja pie to novēršanas, bet otrs turpināja strādāt ar nākamajiem uzdevumiem.

Projekta izstrādes gaitā klients, protams, atrada, izdomāja, atcerējās kaut kādu papildus funkcionālu. Tādā gadījumā, papildus funkcionālas prasības tika dokumentētas tehniskajā uzdevumā un ietverts projekta izstrādes ķēdē. Principā darbs ar automatizētu rēķinu sistēmas realizāciju noritēja pēc metodoloģijas *Agile.* Šīs metodoloģijas galvenās idejas ir sekojošas:

- Personas un to mijiedarbība ir svarīgākā, nekā procesi un instrumenti.
- Strādājošs programmatūra ir svarīgāks, nekā plaša dokumentācija.
- Sadarbība ar klientu ir svarīgāka, nekā pārrunas par līgumiem.
- Spēja reaģēt uz izmaiņām svarīgāka nekā sekošana plānam.

4.2. Automatizētas sistēmas ER modelis

Datu bāzes projektēšanas gaitā atsevišķs ER modelis, uz kura pamata tika veidota datu bāze, netika veidots. Projektā autors uzreiz sāka ar datu bāzes izveidi grafiskā veidā, kas savukārt deva uzskatāmu priekšstatu par to, kā izveidota bāze, kādi pastāv sakari (4.1.att.). Faktiski tas arī ir ER modelis, tikai sīkākā veidā.



4.1.att. **Projekta ER modelis**

Dotajā diagrammā (4.1.att.) ir redzamas tabulu grupas:

4.2.1. Grupa „Ensurance types”

Sarunas gaitā ar klientu tika noskaidrots, ka visiem apdrošināšanas polišu veidiem ir kopīgie lauki (tādi kā apdrošināšanas polises numurs, izveides datums, derīguma termiņš utt.), kā arī dažādi lauki, kuri ir viena tipa apdrošināšanas polisēs, bet kuru nav cita tipa apdrošināšanas polisēs. Tāpēc galvenā tabula „*ensurance*” atbild par pašu apdrošināšanas polises objektu un katras apdrošināšanas polises kopīgiem laukiem. Tālāk katram apdrošinājuma veidam ir papildus tabula ar unikāliem laukiem šim apdrošināšanas polises veidam. Saikne starp tabulu „*ensurance*” un katru no palīgtabulām ir viens pret vienu. Pagaidām ne visiem apdrošināšanas polišu veidiem ir papildus lauki. Pie katra apdrošināšanas polises veida var ielādēt apdrošināšanas objekta fotogrāfijas.

KASKO apdrošināšanas polisei ir tāds jēdziens kā „Pielikums”. Gadījumā, kad klients vēlas, lai *KASKO* polise darbotos ārzemēs uz noteiktu laiku, viņš var pievienot pielikumu savai polisei par atsevišķu samaksu.

4.2.2. Grupa “Ensurance invoices and sms reminder”

Dažus apdrošināšanas veidus var sadalīt vairākos maksājumos, tāpēc apdrošināšanas maksājumu apmaksa ir izdalīta atsevišķā tabulā ar attiecību viens (polise) pret daudzajiem (maksājumiem). Sistēmā tiek kārtota *SMS* piegādes uzskaite tikai *KASKO* apdrošināšanām, un šī uzskaite saglabājas tabulā „*kasko_sms*”.

4.2.3. Grupa „Ensurances owner and payer”

Katrai apdrošināšanas polisei objekta īpašnieks un objekta maksātājs var atšķirties. Tas gadās, ja, piemēram, apdrošināšanas objekts atrodas līzingā vai kredītā, tad objekta īpašnieks ir banka, bet maksātājs – persona, kura izmanto šo objektu. Tāpēc ir izveidotas 2 tabulas „*clients*” un „*owner*”, kurām ir saite viena pret daudzajiem tabulā „*ensurance*”.

4.2.4. Grupa „Ensurances supplier”

Apdrošināšanas brokeris sadarbojas ar vairākām apdrošināšanas firmām. Sistēmā ir paredzēta apdrošināšanas firmu pievienošana, rediģēšana, dzēšana. Ar katru apdrošināšanas firmu brokerim ir vienošanām par procentuālo attiecību, kuru brokeris saņem, pārdodot apdrošināšanu. Visi šie dati glabājas tabulā „*firms*”. Pievienojot jaunu apdrošināšanas polisi, obligātā kārtība tiek norādīta firma, caur kuru ir noformēt apdrošināšanas polise. Tabulā „*firm_bills*” glabājas visi rēķini, kurus apdrošināšanas brokeris izstāda apdrošināšanas kompānijām, lai saņemtu savu komisijas procentu.

4.2.5. Grupa „Additional tables”

Šajā grupā ir savāktas sistēmas palīgtabulas. Tabulā „*sms_config*” glabājas *SMS* ziņojumu šabloni, kuri tiek nosūtīti klientam automātiskajā režīmā un pirms cik dienu nosūtams atgādinājums. Tabula „*users*” satur sevī visus sistēmas lietotājus, kā arī piekļuves līmeni informācijas sistēmai.

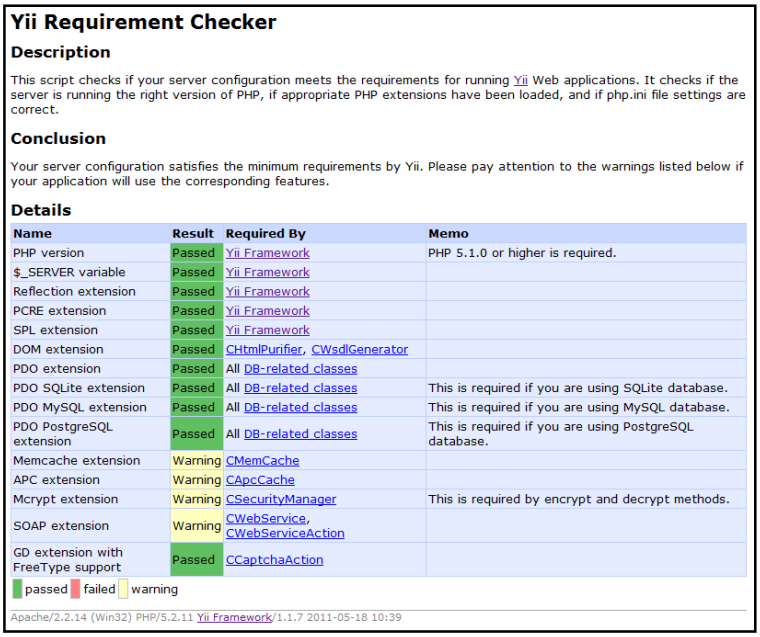
4.3. Izstrāde ar Yii ietvara palīdzību

Dotajā nodaļā tiek aprakstīta tīmekļa lietojumprogrammas realizācijas praktiskā daļa ar *Yii* ietvaru palīdzību. Protams, netiek aprakstīts viss informācijas sistēmas realizācijas process, bet autora skatījumā galvenie un interesantākie momenti. Autors mēģināja piemeklēt vienkāršus, bet tanī pat laikā uzskatāmus piemērus, kuri demonstrē ietvara funkcionālu.

4.3.1. Izstrādes vides sagatavošana

Iesākumam ir neieciešams ielādēt pēdējo *Yii* ietvara versiju uz sava datora. Pēc tam ir nepieciešams atarhivēt atsevišķajā mapē *Document Root*. Tālāk, izpakotajā arhīvā saturēsies sekojošas mapes (uz doto brīdi versijā Yii 1.1.7):

- **Demos** – mape ar gataviem projektiem – piemēriem.
- **Framework** – pats ietvara kodols. Kodols tiek pieslēgts tieši katram projektam. Pie nepieciešamības var pieslēgt vienu un to pašu kodolu vairākiem projektiem.
- **Requirements** – tests uz sistēmas prasībām *Yii* ietvaram. Palaišanu īsteno ar kopumu pārlūkā: http://127.0.0.1/yii/requirements/ (ceļš var atšķirties)
  Darba rezultāts redzams 4.2.attēlā.



Yii Requirement Checker

Description

This script checks if your server configuration meets the requirements for running Yii Web applications. It checks if the server is running the right version of PHP, if appropriate PHP extensions have been loaded, and if php.ini file settings are correct.

Conclusion

Your server configuration satisfies the minimum requirements by Yii. Please pay attention to the warnings listed below if your application will use the corresponding features.

Details

Name	Result	Required By	Memo
PHP version	Passed	Yii Framework	PHP 5.1.0 or higher is required.
$_SERVER variable	Passed	Yii Framework	
Reflection extension	Passed	Yii Framework	
PCRE extension	Passed	Yii Framework	
SPL extension	Passed	Yii Framework	
DOM extension	Passed	CHtmlPurifier, CWsdlGenerator	
PDO extension	Passed	All DB-related classes	
PDO SQLite extension	Passed	All DB-related classes	This is required if you are using SQLite database.
PDO MySQL extension	Passed	All DB-related classes	This is required if you are using MySQL database.
PDO PostgreSQL extension	Passed	All DB-related classes	This is required if you are using PostgreSQL database.
Memcache extension	Warning	CMemCache	
APC extension	Warning	CApcCache	
Mcrypt extension	Warning	CSecurityManager	This is required by encrypt and decrypt methods.
SOAP extension	Warning	CWebService, CWebServiceAction	
GD extension with FreeType support	Passed	CCaptchaAction	

passed failed warning

Apache/2.2.14 (Win32) PHP/5.2.11 Yii Framework/1.1.7 2011-05-18 10:39

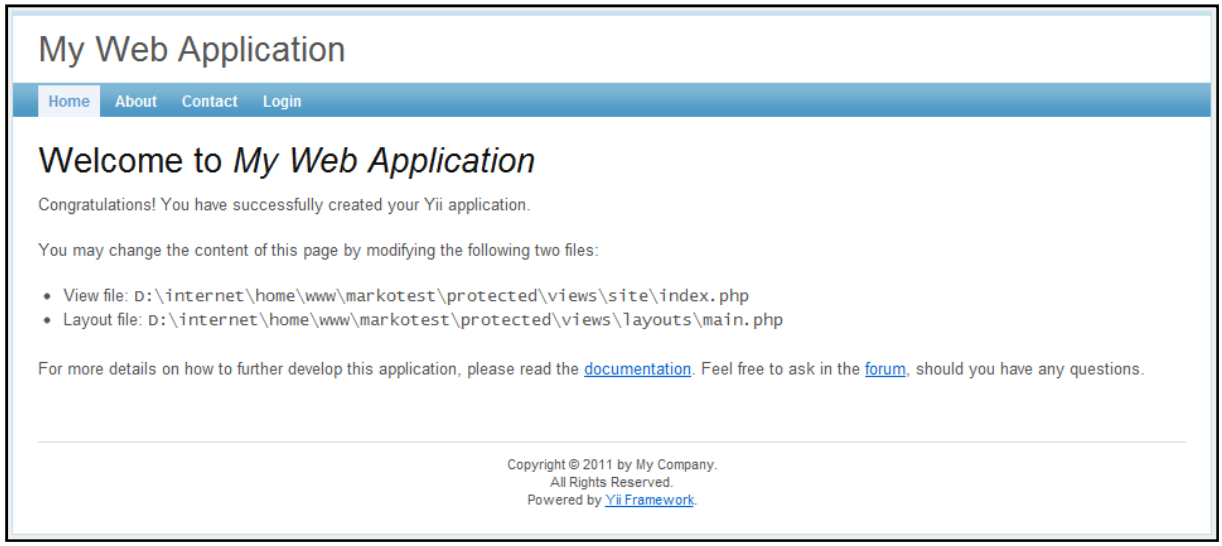
4.2. att. **Sistēmas prasību pārbaude**

Tālāk ir nepieciešams izveidot jaunu projektu. Tas tiek darīts ar komandas rindas palīdzību (4.3.att.).

```
D:\internet\home\www>php yii/framework/yiic.php webapp markotest
Create a Web application under 'D:\internet\home\www\markotest'? [Yes|No] _
```

4.3.att. **Web pielikuma karkasa ģenerācija**

Pēc šīs komandas izpildes mapē *www* tiks izveidots projekts (mape *markotest*), ar stingru mapju un failu struktūru. Tagad var pārbaudīt tikko izveidoto lietojumprogrammas karkasu, ierakstot pārlūkā: http://127.0.0.1/markotest/ (4.3.att.).



4.3.att. **Veiksmīgi izveidots lietojumprogrammas karkass**

Ieejas punkts tīmekļa pielikumā ir fails *index.php*. Šajā failā tiek pieslēgts ietvara kodols un galvenais mūsu lietojumprogrammas konfigurācijas fails (4.4.att.).

```
$yii=dirname(__FILE__).'/../yii/framework/yii.php';
$config=dirname(__FILE__).'/protected/config/main.php';
```

4.4.att. **Ietvara kodola pieslēgšana**

4.3.2. Datu bāzes pieslēgšana

Mūsdienu IT pasaulē neviens projekts nevar apieties bez kaut kādas datu bāzes, tāpēc tagad sāksim pieslēgšanos mūsu *MySQL* datu bāzei, kas izveidota no ER modeļa, programmā *MySQL Workbench* (4.2.paragrāfs). Šajā nolūkā ir nepieciešams atvērt galveno konfigurācijas failu *main.php* (4.5.att.) un aizvietot sekojošas rindas (gadījumā, ja tiek izmantots *MySQL*):

```
'db'=>array(
    'connectionString' => 'mysql:host=localhost;dbname=markobrokers',
    'emulatePrepare' => true,
    'username' => 'markobrokers',
    'password' => 'LT2pDxqz8mVWDJuy',
    'charset' => 'utf8',
),
```

4.5.att. **Datu bāzes konfigurācija**

4.3.3. CRUD (Scaffolding) operāciju realizācija

CRUD — (angļ., *create read update delete — „pievienošana, lasīšana, atjaunošana, dzēšana”*) saisināts nosaukums četram datu parvaldīšanās bāzes funkcijam[16].

Dotajā daļā autors apraksta, cik ātri var izveidot pilnvērtīgu *CRUD(angļ., create-read-update-delete)* noteikta datu bāzes tabulai. Koda ģenerācija tiks veikta ar *Gii* grafisko koda ģeneratoru, kurš pietiekami stipri atvieglo uzdevumu. Protams, paliek iespēja uztaisīt *CRUD* ģenerāciju arī no komandas rindas, tomēr dotajā dāļā tas netiks aprakstīts. Kā piemēru,autors paņem vienu no izstrādājamā projekta tabulām, tabulu „*Clients*” (4.1.att.). Protams, ģenerācija ir parādīta tikai sākotnējā stadijā un pēc tās notiek pirmkoda manuālā koriģēšana saskaņā ar tehniskā uzdevuma nosacījumiem, tomēr šeit ir parādīts tikai *Gii* ģeneratora darba princips.

Gii koda ģenerators ļauj ģenerēt kontrolierus, kuri realizē *CRUD* operācijas, kā arī šī operācija tiek saukta par *scaffolding*[17]. Pēc noklusējuma *Gii* ģenerators ģenerē standartu šablonu komplektu modeļu un kontrolieru, bet pēc nepieciešamības ir iespējams pievienot savus šablonus, lai ģenerētu specifiskāku kodu. Piemēram, var nomainīt formu veida ģenerācijas šablonu, lai forma atbilstu projekta gatavam dizainam, to var būtiski samazināt laiku, realizējot lielu projektu, jo turpmākais ģenerējamais kods mainīsies mazākos apjomos.

4.3.3.1. Gii ieslēgšana

Tagad vajadzīgajām tabulām datu bāzē var realizēt *CRUD* operācija. Pirmkoda uzrakstīšanas vietā tagad būs ērti izmantot pirmkoda ģeneratoru *Gii*. *Gii* ir pieejams, sākot ar versiju 1.1.2. Agrāk šiem pašiem mērķiem izmantoja jau pieminēto *yiic* utilītprogrammu. Lai aktivizētu *Gii* darbu, ir nepieciešams rediģēt galveno konfigurācijas failu *main.php* (4.6.att.):

```
'gii'=>array(
    'class'=>'system.gii.GiiModule',
    'password'=>'Enter Your Password Here',
    // If removed, Gii defaults to localhost only. Edit carefully to taste.
    'ipFilters'=>array('127.0.0.1','::1'),
),
```

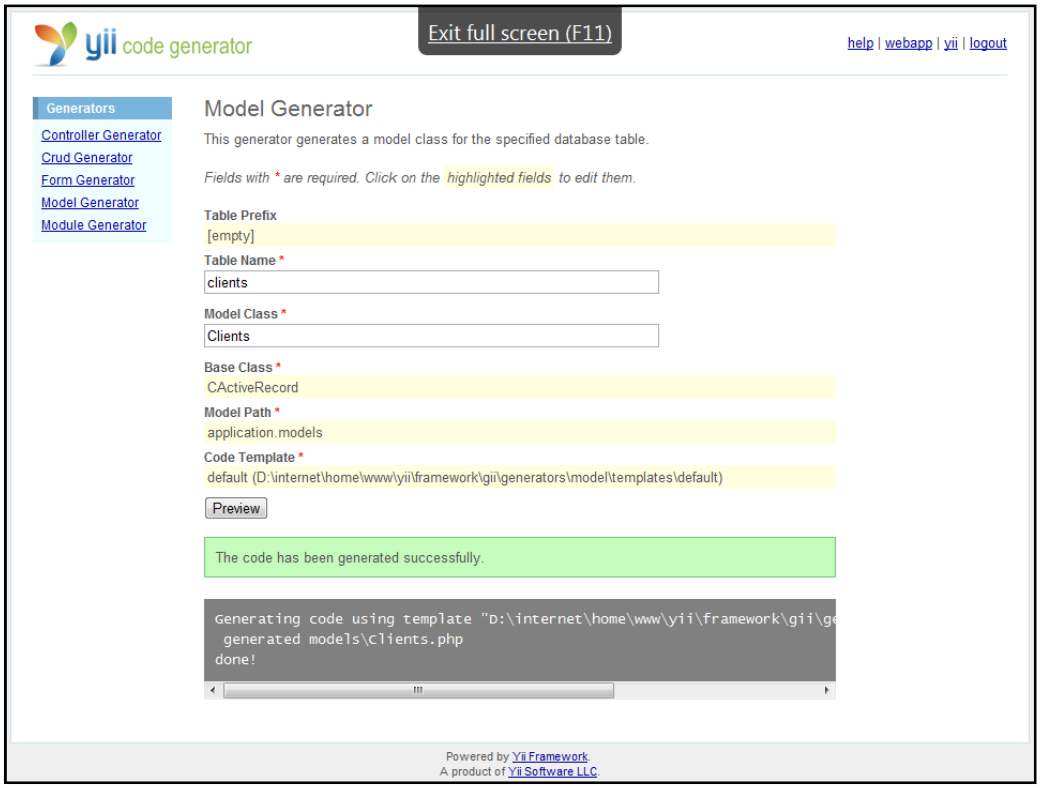
4.6.att. **Gii Web koda ģeneratora konfigurācija**

Pēc ieviestajām izmaiņām pirmkoda ģenerators ir pieejams pēc adreses http://127.0.0.1/markotest/index.php?r=gii. *Gii* ir pieejamas 4 opcijas:

- Controller Generator
- Crud Generator
- Form Generator
- Model Generator
- Module Generator

4.3.3.2. Modeļa ģenerācija

Ieejot *Gii* ģeneratorā un izvēloties opciju „*Model Generator*", kuru nospiežot, mēs redzam sekojošu logu (4.7.att.):

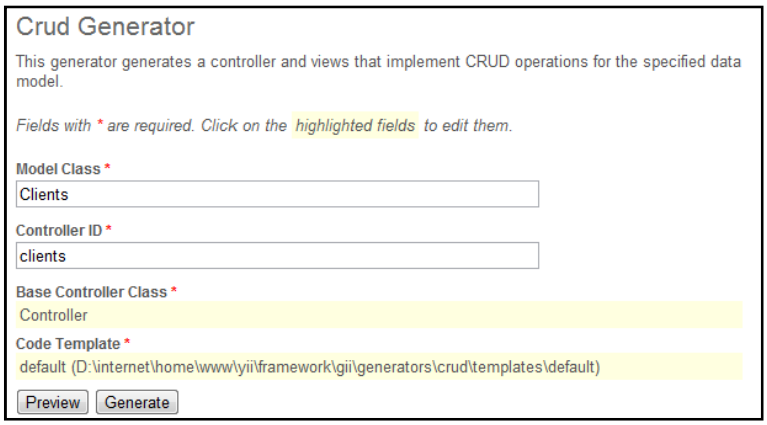


4.7.att. **Modeļa izveide „Clients" tabulai**

„*Table name*" laukā ir nepieciešams ierakstīt tabulas nosaukumu, kurai mēs vēlamies izveidot modeli. Modeļa klases nosaukumu *Gii* piedāvā automātiski, tomēr lietotājam paliek iespēja mainīt to. Pēc tabulas nosaukuma ierakstīšanās var nospiest pogu *Preview* un, ja viss ir veiksmīgi, pogu *Generate*. Šādā veidā tika izveidots jauns modelis *Clients*.

4.3.3.3. CRUD ģenerācija

Pēc modeļa klases ģenerācijas ir nepieciešams ģenerēt kodu, kurš realizē tai *CRUD* operācijas. Šajā nolūkā *Gii* ģeneratorā izvēlamies opciju *CRUD Generator* (4.8.att.). *CRUD generator* ierakstām mums vajadzīgo modeli, *Controller ID* ģenerators ievadīs automātiski, tomēr pie vēlēšanās var ierakstīt savu *Controller ID* (4.8.att.).



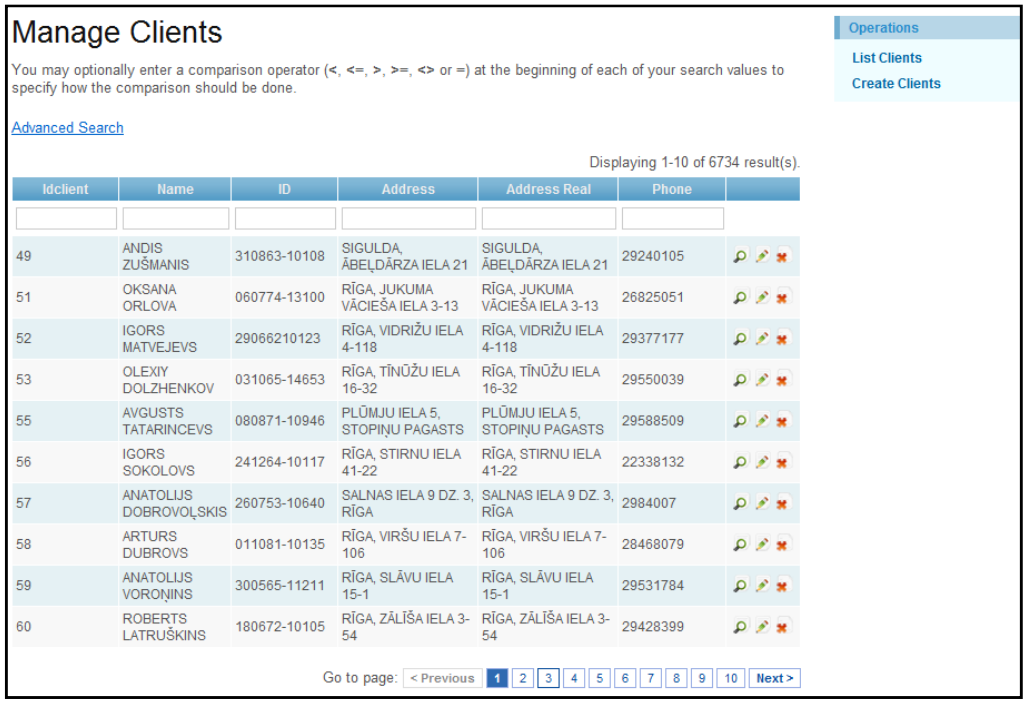
4.8.att. **CRUD ģenerācija**

Pēc pogas *Preview* nospiešanas ģenerators parādīs, kādi faili un kādās vietās tiks noģenerēti.

Piekļuve klientu vadības lapai būs sekojoša: http://127.0.0.1/markotest/index.php?r=clients

4.3.3.4. Rezultāts

Pēc veiktajām operācijām parādīties skaista lietotāju tabula. Var noklikšķināt uz tabulas nosaukuma, lai sakārtotu ierakstus pēc atbilstošās kolonnas. Atbilstošās rindas apskatei, rediģēšanai vai dzēšanai var izmantot pogas. Var arī pāriet uz dažādām lapām, filtrēt rezultātus un veikt meklēšanu pēc tām (4.9.att.). Viss tas neprasa koda uzrakstīšanu.



4.9.att. **Gii koda ģeneratora noģenerētais izvads**

4.3.4. Darbs ar kontrolieriem

Izmantojot *Gii* pirmkoda ģeneratoru, tas veido modeļu klases, skatus, kā arī kontrolierus, faktiski darbs ar kontrolieriem paredz izmaiņas, jau gatavā kontroliera piestrādi. Izstrādājamā rēķinu sistēmā lietotājam nav tiesību neko darīt un redzēt, kamēr viņš nav autorizējies sistēmā. Apskatīsim uz piemēra, kā tas realizēts ar ietvara palīdzību. Konfigurācijas failā *main.php* mēs izveidojām masīva atslēgu *homeUrl* (4.10.att):

```
// uncomment the following to define a path alias
// Yii::setPathOfAlias('local','path/to/local-folder');

// This is the main Web application configuration. Any writable
// CWebApplication properties can be configured here.
return array(
    'basePath'=>dirname(__FILE__).DIRECTORY_SEPARATOR.'..',
    'name'=>'MARKO',
    'homeUrl' => array('site/login'),
    ...
);
```

4.10.att. **Mainīgā homeUrl lielums**

Yii ietvarā kontrolieri un darbības tiek atpazītas pēc to indikatoriem. Lietotājs vēršas pie kontroliera un darbības ar maršruta (*route*) palīdzību. Maršruts tiek veidots, apvienojot kontroliera un darbības identifikatorus, kas atdalīti ar slīpu svītru. Mūsu gadījumā maršruts

site/login norāda uz *login* kontroliera darbību *SiteController*, un pēc noklusējuma saite „*http://127.0.0.1/index.php?r=site/login*" novedīs tieši pie šī kontroliera un darbības izsaukšanas.

Kontrolierī *SiteController* mums ir metode *actionLogin*(4.11.att), kuru izsauc, vēršoties pie mūsu lietojumprogrammas pēc noklusējuma.

```
public function actionLogin(){
    $model=new LoginForm;

    // collect user input data
    if(isset($_POST['LoginForm']))
    {
        $model->attributes=$_POST['LoginForm'];
        // validate user input and redirect to the previous page if valid
        if($model->validate() && $model->login())
            $this->redirect(Yii::app()->user->returnUrl);
    }
    // display the login form
    $this->render('login',array('model'=>$model));
}
```

4.11.att. **Metodes *actionLogin* sākotnējais kods**

Sākumā veidojam jaunu modeļa instanci *LoginForm*. Modelī tiek nodoti *POST* parametri, kurus mēs ieguvām autorizācijas formas aizpildīšanas rezultātā, tajā gadījumā, ja forma patiešām tika nosūtīta. Tālāk notiek iegūto lauku validācija un pēc tam ielogošanas mēģinājums. Sākumā notiek iegūto lauku validācija uz modeļa *LoginForm* metodes *rules()* pamatā (4.12.att.). Pilnu *LoginForm* pirmkodu var skatīt 5. pielikumā.

```
public function rules()
{
    return array(
        // username and password are required
        array('username, password', 'required'),
        // rememberMe needs to be a boolean
        array('rememberMe', 'boolean'),
        // password needs to be authenticated
        array('password', 'authenticate'),
    );
}
```

4.12.att. **Modeļa *LoginForm* metode *rules()***

Metode pārbauda obligāto lauku esamību. Ja pārbaudes *required*, *boolean* bija standartas *Yii* ietvara pārbaudes, tad pēdējā pārbaude *authenticate* ir lietotāju metode un veic pārbaudi uz

paroles pareizību (4.13.att.). Tāpat metode *rules()* satur pietiekami daudz iebūvētu validācijas noteikumu, tādu kā:

- **required** – lauks obligāts
- **filter** – var uzdot savu funkciju filtrācijai
- **match** –sakritība pēc regulāras izteiksmes šablona
- **email** – vai lauks ir e-pasta adrese
- **url** – vai lauks ir saite
- **unique** – vai ir unikāls lielums datu bāzē
- **compare** – salīdzinājums ar citu lauku
- **length** – ierobežojums pēc rindas izmēra. Tiek arī nodoti papildus ierobežojumi *min* un *max*
- **in** – vai lauks atbilst kādam no uzskaitītajiem lielumiem
- **numerical** – vai lauks ir ciparu lauks
- **type** – kādam tipam jābūt lielumam
- **file** – vai lielums ir ielādēts fails
- **default** – ja lielums nav ielikts, tiek ņets lielums pēc noklusējuma
- **boolean** – vai lielums ir *bool* tips
- **date** – vai lielums ir datums

Pats pārbaudes kods ir uzrakstīts tajā pašā modelī.

```
public function authenticate($attribute,$params)
{
    if(!$this->hasErrors())  // we only want to authenticate when no input errors
    {
        $this->_identity=new UserIdentity($this->username,$this->password);
        $this->_identity->authenticate();
        switch($this->_identity->errorCode)
        {
            case UserIdentity::ERROR_USERNAME_INVALID:
                $this->addError('username','Lietotāja vārds/Parole ir nepareiza!');
                break;
            case UserIdentity::ERROR_PASSWORD_INVALID:
                $this->addError('password','Lietotāja vārds/Parole ir nepareiza!');
                break;
            default:
                break;
        }
    }
}
```

4.13.att. **Login un paroles pārbaude**

Tamdēļ, ka *Yii* ietvaram ir standarta pārbaude uz paroles pareizību, mēs kā reiz to izmantojām, nododot lietotāju un paroli klasei *UserIdentity*. Klase *UserIdentity*, tā ir standarta autorizācijas klase, kuru *Yii* ietvars rada, ģenerējot projektu, tāpēc tā atrodas „*/protectes/components/UserIdentity.php*".

4.3.4.1. Autorizācijas iestatījumi kontrolieriem

Dotā sistēma ir aizvērta sistēma, tāpēc mums ir nepieciešams aizvērt pieeju visiem kontrolieriem, neautorizētiem lietotājiem. Šajā nolūkā katram kontrolierim ir metode *accessRules()*, kas atbild par pieejas tiesībām kontrolierim un, pēc nepieciešamības, katrai darbībai. Šīs metodes realizācija ir apskatama 4.14.attēlā (@ - nozīmē autentificēts lietotājs).

```
public function accessRules(){
    return array(
        array('allow',
            'users'=>array('@'),
        ),
        array('deny',
            'users'=>array('*'),
        ),
    );
}
```

zīm. 0.1

4.14.att. ***accessRules*** **realizācija**

4.3.5. Skati

Kā jau rakstīts iepriekš, *Yii* ietvars ir bazēts uz *MVC* projektēšanās šablona. Tāpēc viss *HTML* koda izvads ir atdalīts no biznesa loģikas. Metode *actionLogin*, par kuru tika runāts iepriekš, mēģinot neveiksmīgi autorizēties vai ja autorizācija vispār nav notikusi, izvada autorizācijas formas skatu (4.11.att., pēdējā rinda). Pats skats atrodas mapē „*/protected/views/site/login.php*". un izskatās, kā parādīts 4.15.attēlā.

```
<h2 id="auth">Autorizācija:</h2>
<div id="login">
<?php $form=$this->beginWidget('CActiveForm', array(
    'id'=>'login-form',
    'enableAjaxValidation'=>false,
)); ?>
    <div>
        <?php echo $form->label($model,'username'); ?>
        <?php echo $form->textField($model,'username'); ?>
    </div>
    <div>
        <?php echo $form->label($model,'password'); ?>
        <?php echo $form->passwordField($model,'password'); ?>
    </div>
    <div class="row rememberMe">
        <?php echo $form->label($model,'Atcerēties mani'); ?>
        <?php echo $form->checkBox($model,'rememberMe'); ?>
    </div>

    <div id="submit">
        <?php echo CHtml::submitButton('Ielogoties', array('class' => 'button')); ?>
    </div>

<?php $this->endWidget(); ?>
</div>
```

4.15. att. **skata login.php pirmkods**

Šajā skata viss ir ļoti pārskatāmi, *beginWidget un endWidget* atbilstoši, tas ir formas sākums un beigas. Protams, vidžeti (angļ., *widgets*) *Yii* ietvarā nebeidzas ar formas pievienošanu, pastāv daudzi citi lietderīgi vidžeti. Tālāk notiek teksta birkas un ievades lauku izvade. Katram objektam mēs nododam mainīgo *$model*, kas satur visu nepieciešamo, kas jāzina, atspoguļojot objektu (nosaukums, lielums pēc noklusējuma utt.).

4.3.6. Active Records un DAO

Yii ietvars ļauj strādāt projektā gan ar datu piekļuves objektu (*angļ., DAO)*, gan ar *ActiveRecords. Active Record* realizē populāru objektrelācijas projicēšanas pieeju (*angļ., ORM*). Katra *ActiveRecord* klase atspoguļo datu bāzes tabulu (vai priekšstatu), *ActiveRecord* eksemplārs – rindu šajā tabulā, bet kopējās *CRUD* operācijas ir realizētas kā *AcitveRecords* metodes. Rezultātā mēs varam strādāt ar lielāku objektorientētību. Piemēram, jā mēs izmantojam *AcitveRecords*, nepieciešamā lietotāja meklēšana notiek sekojošā veidā: $user=User::model()->find('LOWER(login)=?',array($username));

Pie nepieciešamības izmantot sarežģītāku *SQL* loģiku, *Yii* ietvars arī ļauj strādāt ar *DAO* objektiem, kuri atļauj rakstīt pieprasījumus tīrā *SQL*.

4.3.7. Migrācijas

Migrācijas *Yii* ietvarā parādījās tikai 1.1.6.versijā, ņemot vērā, ka tekošā stabilā versija ir 1.1.7, tas ir nesen. Kā parasti noteiktai migrācijai ir sava klase, kurā ir noteiktas metodes

Up() un *Down()*, bet darbs notiek no komandas rindas (*yiic*). Projekta izstrādes sākumā migrācija netika realizēta un izstrādātāji strādāja ar vienu kopīgu datu bāzi, kura atradās serverī un bija vienmēr pieejama.

4.3.8. Rēķinu ģenerācijas PDF formātā

Viena no svarīgākajām prasībām pret sistēmu bija rēķinu ģenerācija un saglabāšana/nosūtīšana *PDF* formātā. Līdz tam autoram ir bijusi *PDF* failu ģenerācijas pieredze ar *PHP* bibliotēkas *fpdf* palīdzību. Šī metode ir pietiekami sarežģīta, jo bija problēma ar latviešu burtiem, tie nekorekti atspoguļojās, tas tika risināts, konvertējot visus simbolus dokumentā no *UTF8* uz CP1257. Tomēr pati svarīgākā problēma bija *PDF* dokumenta izstrāde, jo vajadzēja ģenerēt *PDF* failā ne tikai vienkāršo tekstu(tas nebija sarežģīti), bet ģenerēt *PDF* failā rēķinu ar grafiskiem elementiem un tabulas datiem, tas aizņēma daudz laika, jo elementu zīmēšanas princips tur balstās uz koordinātu asu punktiem.

Šajā projektā tika atrasts jauns šīs problēmas risinājums (varbūt kādam tas arī nav gluži jauns). Visi rēķini tiek ģenerēti HTML formātā, kas ļoti atvieglo uzdevumu, bet tad servera programma *wkhtmltopdf* konvertē HTML failu PDF formātā uz tīmekļa pārlūka WEBKit bāzes. Projekts *wkhtmltopodf* sākās 2009.gada jūlijā un uz doto brīdi pēdējā programmas versija ir 0.10.0 rc2. Programmas darbības princips ir ļoti vienkāršs. 4.16. attēla tiek parādīts pirmkoda piemērs, kā tā tiek izmantota realizējamā projektā.

```
file_put_contents($htmlPath, $html);
exec('xvfb-run -a -s "-screen 0 1024x768x24" wkhtmltopdf -s A4 --dpi 200 "'.$htmlPath.'" '.$pdfPath);
```

4.16.att. **PDF ģenerācija**

xvfb-run – tas ir virtuāls X11 servera koda buferis, kurš taisa visas grafiskās operācijas atmiņā, nerādot izvadu uz ekrāna [18]. Tamdēļ, ka dotais *Linux* serveris nestrādā grafiskā režīmā, bet *HTML* faila konvertēšana norisinās ar tīmekļa pārlūka *WEBKit* palīdzību, nākas emulēt grafisko režīmu šādā veidā.

4.3.9. SMS atgādinājumu izsūtīšanas realizācija

SMS ziņojumu izsūtīšana notiek katru dienu plkst. 13:00. Tamdēļ, ka ziņojumu nosūtīšana ir autonoma, izsūtīšanas izpildes skripts ir ierakstīts *CRON*. *CRON* – tas ir uz laika balstīts darba plānotājs *UNIX* veidā operētājas sistēmās. Citiem vārdiem sākot, ar *CRON* palīdzību var uzkonfigurēt jebkāda skripta palaišanu noteiktā laikā brīdī *Unix* tipa serveros.

Palaižot skriptu, *Yii* lietojumprogrammas eksemplārs netiek veidots, scenārijs ir uzrakstīts uz tīra PHP. Skripts skar tādus apdrošināšanas veidus kā *KASKO* un *OCTA*. Iesākumā skripts nolasa konfigurācijas datus katram apdrošinājuma veidam (4.17.att.).

```
/* OCTA {{{ */

// get octa config
$sql = 'SELECT * FROM sms_config WHERE type="octa"';
$db_result = $db->query($sql);
$octa_config = $db_result->fetch_array();
$db_result->close();
```

4.17.att. **Konfigurācijas nolasīšana priekš OCTA**

Pēc tam tiek izpildīts *SQL* pieprasījums, kurš izvēlas visas apdrošināšanas polises, kurām drīz beigsies derīguma termiņš. Atlases algoritms ir ļoti vienkāršs, tiek atlasītas visas apdrošināšanas, kurām derīguma termiņš beidzas pēc noteikta dienu skaita. Konfigurācijas mainīgais *$octa_config['days']* – atbild par dienu skaitu, kurās nosūtīt brīdinājumu (4.18.att.).

```
// select octas to send sms
$date_end = date('Y-m-d', time() + ($octa_config['days'] * 3600 * 24));
$sql = <<<SQL
SELECT o.car, c.phone
FROM ensurance e, octa o, clients c
WHERE e.typeid=1
    AND e.date_end="$date_end"
    AND CHAR_LENGTH(c.phone)=8
    AND e.clientid=c.idclient
    AND o.ensuranceid=e.idensurance
SQL;
$db_result = $db->query($sql);
if(!$db_result){
    printf("Error: %s\n", $db->error);
}
```

4.18.att. **Apdrošināšanas polišu OCTA atlase**

Tālāk tiek palaists cikls, kurš ar bibliotēkas *CURL* palīdzību nosūta ar *GET* pieprasījumu datus uz kompānijas serveri, kas nodarbojas ar SMS izsūtīšanu (4.19.att.).

CURL – bibliotēka, kura ļauj savienoties un sazināties ar dažāda veida serveru tipiem un ar dažādiem protokolu tipiem[19].

```php
$ch = curl_init();
while($octa = $db_result->fetch_array()){
    $url = SMS_URL.
            '?username='.LOGIN.
            '&password='.PASSWORD.
            '&from='.SENDER.
            '&text='.urlencode(str_replace('#NUMURS#', $octa['car'], $octa_config['text'])).
            '&to='.$octa['phone'];
    curl_setopt($ch, CURLOPT_URL, $url);

    curl_exec($ch);
    sleep(1);
}
curl_close($ch);
$db_result->close();
```

4.19.att. **Nosūtīšana ar SMS atgādinājumu**

4.4. Secinājumi par Yii ietvara izmantošanu

Jebkurš ietvars atvieglo un paātrina produkta izstrādi, jo daudzi uzdevumi jau ir atrisināti. Izstrādātājs var vairāk koncentrēties uz pielikuma biznesa loģikas realizācijas, bet tādas lietas kā datu validācija, lietotāju autorizācija, mijiedarbība ar datu bāzi vairumā gadījumu jau ir realizētas ietvarā.

Active Records – realizē mijiedarbību ar datu bāzi. Visi pieprasījumi iziet caur klases metožu izsaukšanu. izstrādātājam nav jāraksta *SQL* kods. Tāpat risinājums ir universāls (*PDO*), un neierobežo izstrādātāju datu bāzes vadības sistēmas tipa izvēlē.

Yii ietvarā ir iebūvēta lietotāju autorizācijas sistēma, kura balstās uz lietotāju konfigurācijas failiem, ko var nofigurēt ar vienkāršu noteikumu masīvu.

Ietvarā ir lielisks grafiskais koda ģenerators *Gii* (*CRUD, scaffolding* ģenerēšana) ar grafisko interfeisu. Ģenerējot klases, tas radā, kādi faili un kādās vietās tika pievienoti

Ietvaram ir daudz lietderīgu vidžetu (*angļ., widgets*), kuri ļauj viegli veidot bieži izmantojamos lietotāja interfeisa elementus, piemēram, saraksts ar filtrāciju, lapošanu, šķirošanu (CGridView).

Modeļos ir iebūvēta datu validācijas sistēma, kura viegli konfigurējas. Ir daudz iebūvēto validatoru (4.3.4.nodaļa).

Yii ietvars ir viens no pašiem ātrākajiem *PHP* ietvariem. Salīdzinājumā tiek izmantota aplikācija *hello world*, kas pārbauda ietvara kodola klašu ielādes ātrumu, kā arī *Yii* inicializāciju (3.7.nodaļa).

Yii izmanto populāru projektēšanas šablonu *MVC*, kas ļauj labāk strukturēt pielikumu un nodalīt loģiku no priekšstata.

Ietvarā ir realizēta ļoti detalizēta kļūdu novēršanas sistēmas. Pie ieslēgtas novēršanas var iestatīt kļūdu līmeni. Rodoties kļūdai, tiek ievadīts detalizēts apraksts ar kļūdu un tās izsaukuma vietu.

Pēc noklusējuma ietvars izmanto tīro *PHP* kā šablonizatoru, tāpēc *HTML* skatu ģenerēšanās ātrums ir ļoti augsts. Pie vēlēšanās ir iespēja pieslēgt jebkuru citu šablonizatoru.

Autors uzskata, ka jāpievērš uzmanība *Yii* ietvaram, jo tas ir viens no perspektīvākajiem ietvariem, kas uzrakstīts *PHP* valodā. Pie tam ietvars aktīvi attīstās, mājas lapā ir ļoti laba, detalizēta dokumentācija par *API* (ir arī pilns tulkojums uz krievu valodu), ievadkurss bloga izveidei un pietiekami liela biedrība.

Yii ir pietiekami zems ieiešanas līmenis, tomēr, izejot bloga izveides kursu, kļūst saprotama ietvara filozofija. Protams, reālā projektā izstrādātājs sastopas ar grūtībām kā jau jebkurā ietvarā, bet, izlasot dokumentāciju, tās ir viegli atrisināmas.

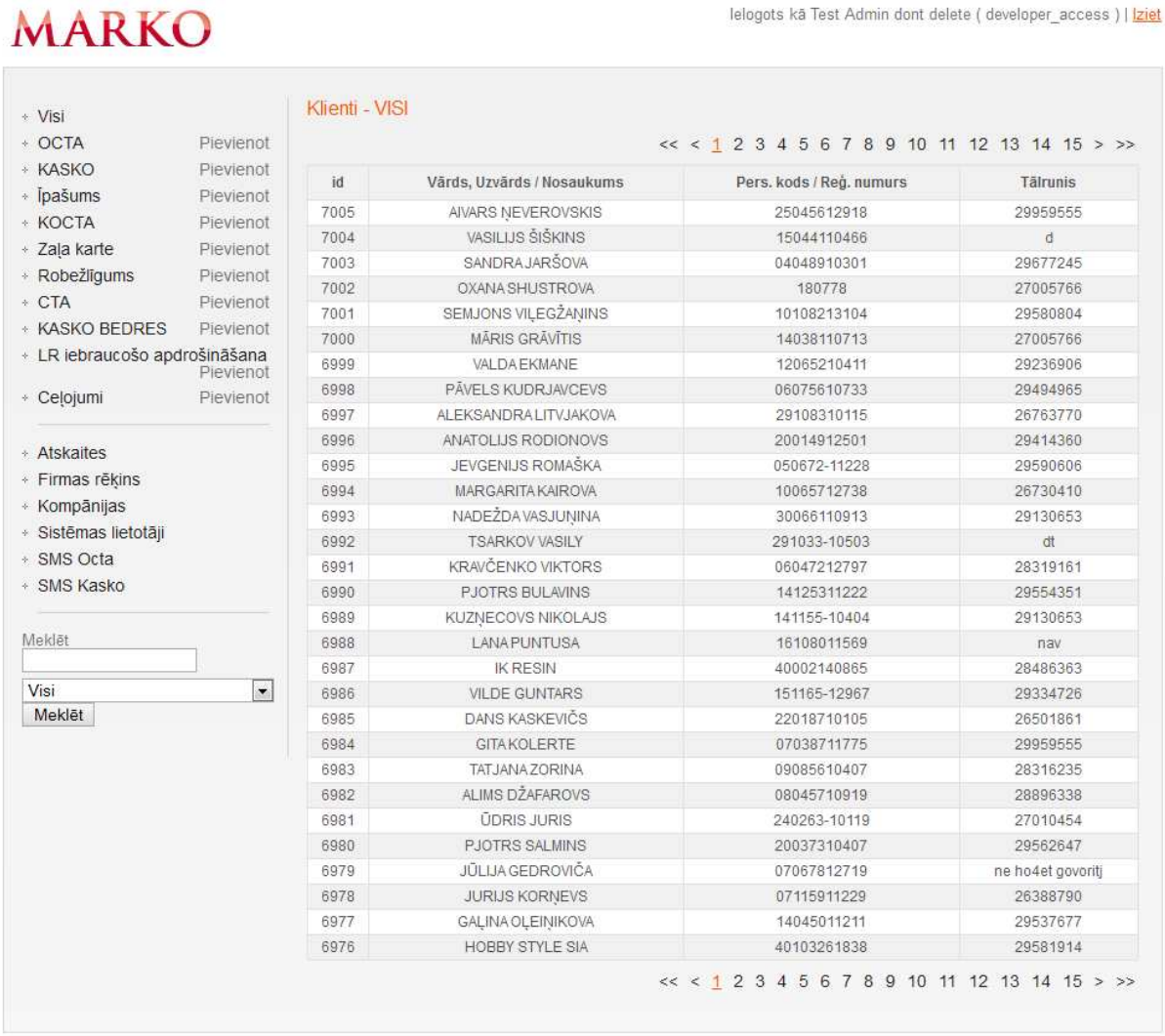
5. SISTĒMAS SASKARNE UN LIETOJAMĪBA

Projektā ir ļoti svarīgs darba ātrums un vienkāršums, jo:

- Uzņēmumā ir diezgan daudz klientu, un gadās, kad jaunas apdrošināšanas polises reģistrācijai stāv vesela rinda, tāpēc jaunas polises reģistrācijai un vajadzīgās informācijas meklēšanai sistēmā jābūt maksimāli ātrai izpildījumā.
- Uzņēmuma personāls ir cilvēki, kuriem IT joma ir sveša, viņi var pieļaut kļūdas neuzmanības dēļ, tāpēc cilvēcisko faktoru jānoved līdz minimumam.

5.1. Sistēmas galvenais logs

Galvenais sistēmas logs ar administratora tiesībām izskatās kā parādīts 5.1.att.



5.1.att. **Sistēmas galvenais logs**

Sistēmas saskarne sadalīta divās daļās:

- Pa kreisi atrodas saita galvenā izvēlne, kura visu laiku paliek nemainīga, izņemot gadījumus, kad sistēmā autorizējas parasts lietotājs, tad daļa opciju nav pieejamas.
- Pa labi atrodas klientu saraksts pēc apdrošināšanas veida (nospiežot uz apdrošināšanas veida pa kreisi), vai arī tiek attēloti visi uzņēmuma klienti.

Nospiežot pogu ”Pievienot” pie jebkura apdrošināšanas veida, atveras jaunas apdrošināšanas pievienojuma forma (aprakstīts 5.2 punktā).

Tāpat kreisajā pusē atrodas klientu saraksts. Meklējot klientu, iespējams saīsināt meklēšanas apgabalu, izvēloties noteiktu apdrošināšanas veida grupu, pēc kura notiks meklēšana. Klientu meklēšana iespējama pēc sekojošiem lauciņiem:

- Klienta vārds, uzvārds;
- Tālrunis;
- Aprošināšanas polises numurs;
- Personas kods;
- Automašīnas valsts reģistrācijas numurs;
- Transportlīdzekļa reģistrācijas apliecības numurs.

Informācijas sistēmas lietotāju ērtības labā meklēšana strādā šādā veidā:

- No meklējamās frāzes tiek dzēsta domuzīme;
- Garumzīmes meklēšanas laikā netiek ņemtas vērā;
- Mazie un lielie burti netiek ņemti vērā meklēšanas laikā;
- Meklētājs meklē ne tikai pilnīgu frāzes sakritību, bet arī tikai daļēju, tas ir, ievadot „289”, meklētājs atlasīs visus klientus, kuru meklēšanai paļautajos lauciņos būs cipari „289”, piemēram, tas var būt tālruņa numurs „28934345”.

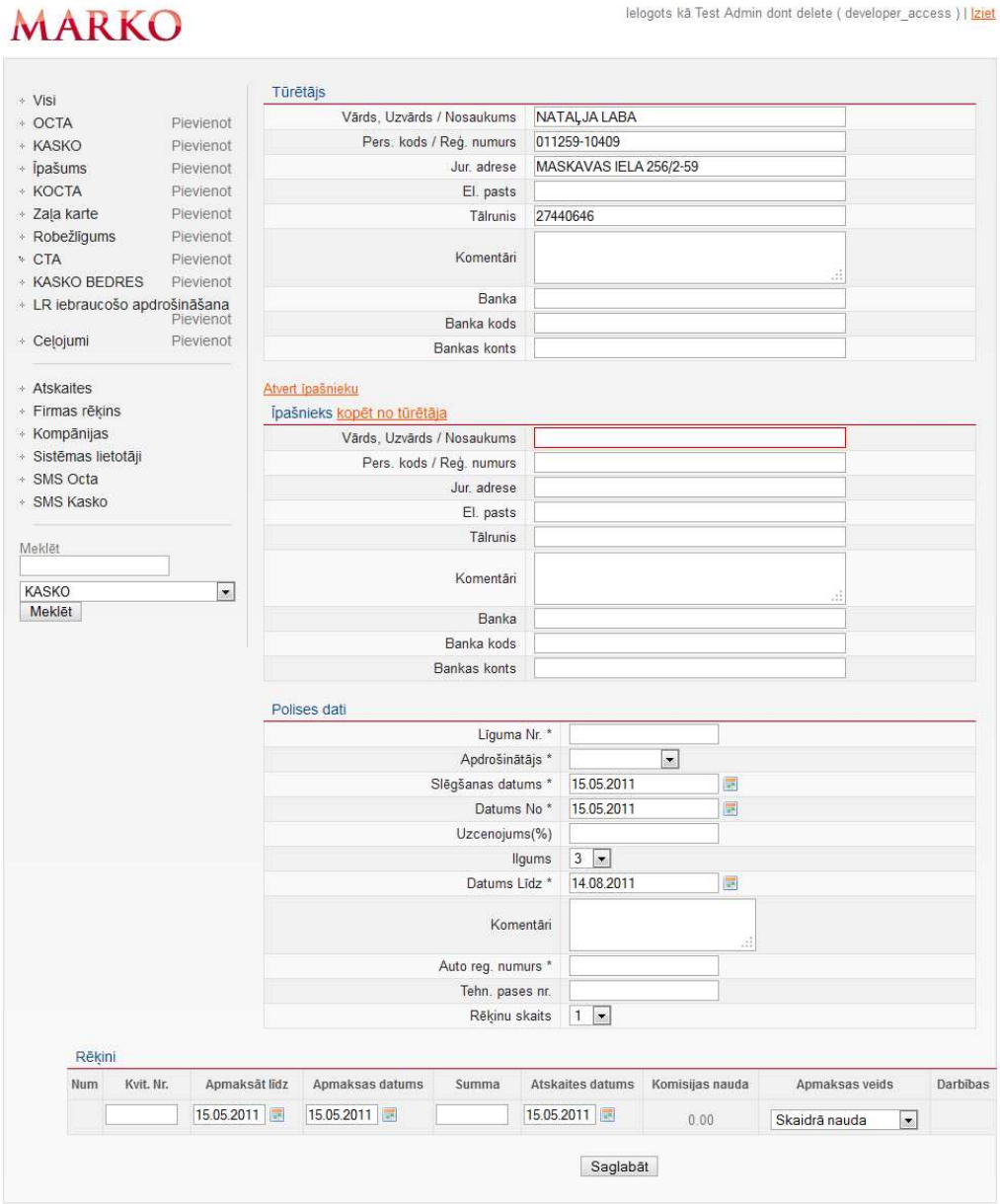
5.2. Jaunas apdrošināšanas polises pievienošana

Jaunas apdrošināšanas polises ievades uzsākšanai ir svarīgi noskaidrot, vai sistēmā jau ir konkrētais klients, lai novērstu klientu dublēšanos. Klientu var noteikt caur meklēšanas formu, bet tādā gadījumā lietotājam nākas veikt papildus darbības: atrast lietotāju, izmantojot meklētāju, ja meklēšana klientu atrod, tad izvēlēties to un viņa kartītē izvēlēties vajadzīgās polises pielikumu; ja tāda klienta nav, tad nospiest taustiņu „pievienot” pie vajadzīgā apdrošināšanas veida (zīm. 5.1). Visas šīs darbības atņem uzņēmuma darbiniekiem laiku, tādēļ mēs piedāvājam sekojošu apdrošināšanas polises pievienošanas shēmu (tiek aplūkota OCTA polises pievienošana):

1) Nospiežot taustiņu „pievienot” pie vajadzīgāapdrošināšanas veida, tiek atvērts logs ar uzņēmuma reģistrācijas numura vai personas koda ievadni (zīm. 5.2.)

Рис. 5.2. первая форма добавления страхового полиса

2) Ja klients ar šādu personas kodu vai reģistrācijas numuru jau eksistē, tad tiek atvērta šī klienta polises pievienošanas forma, pretējā gadījumā visi dati jāievada ar roku (zīm.5.3).

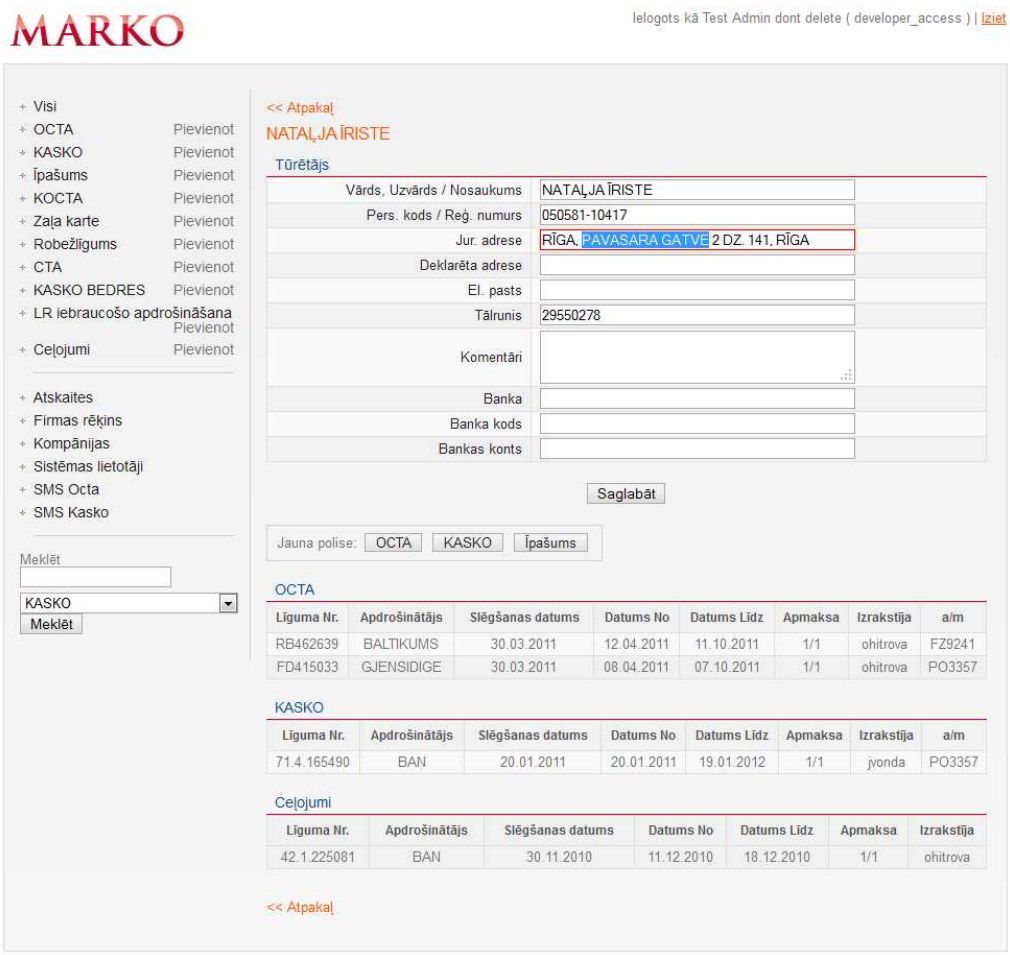


5.3.att. **Apdrošināšanas polises pievienošanas otrais solis**

Polises pievienošanas laikā īpašnieka dati ne vienmēr tiek aizpildīti un tie ne vienmēr ir vajadzīgi, tādēļ tie ir slēpti un tiek atvērti nospiežot saiti „Atvērt īpašnieku". Tādi lauki kā: „Slēgšanas datums, Datums no, Datums līdz, Apmaksāt līdz, Apmaksas datums" tiek aizpildīti automātiski, par pamatu ņemot sistēmas laiku; mainot lauku „Ilgums", automātiski mainās lauks „Datums līdz". Visi šie uzlabojumi veikti, lai polises aizpildīšanas laiks samazinātos.

5.3. Klienta kartīte

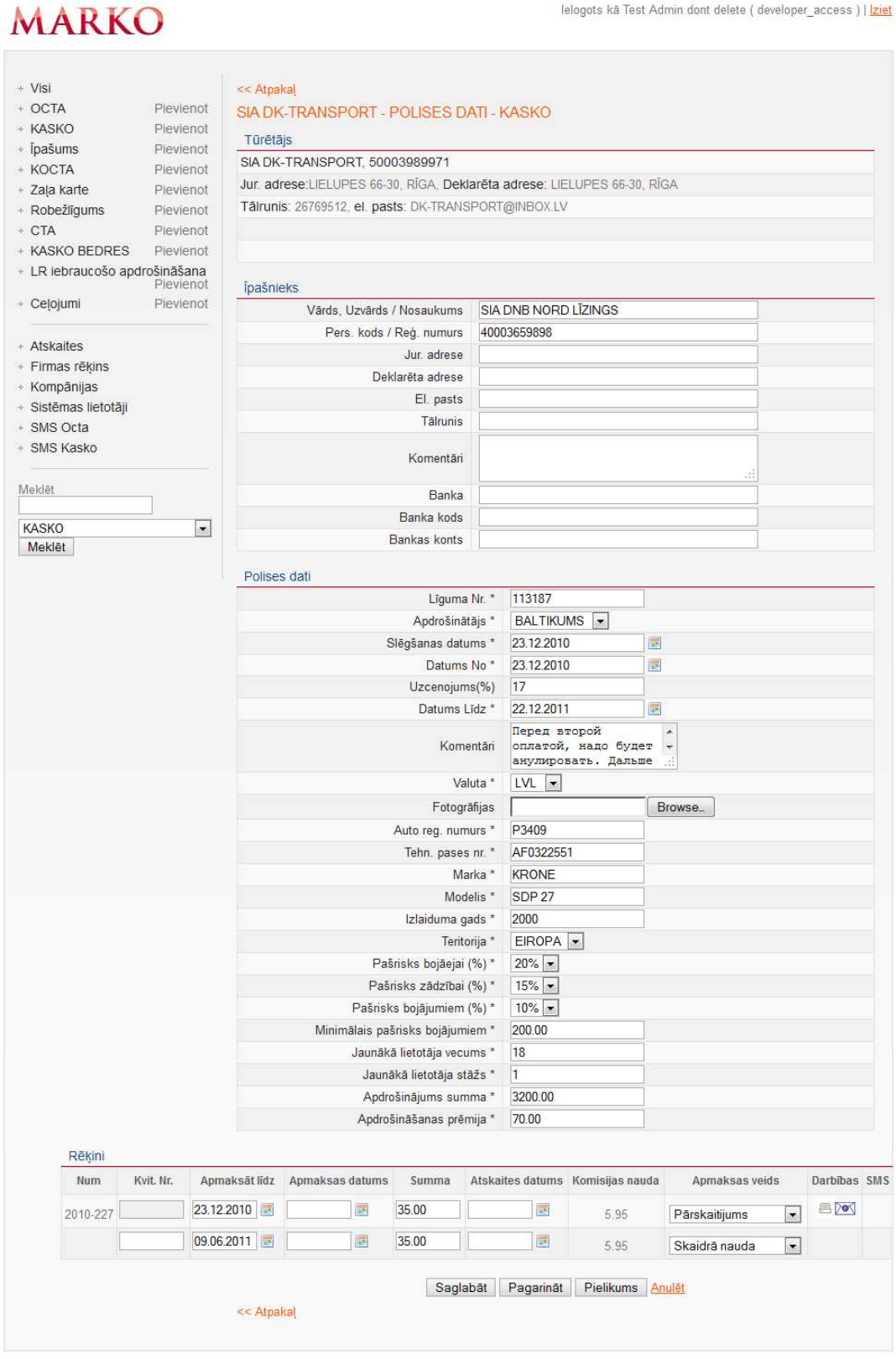
Kā jau minēts iepriekš, katrai apdrošināšanas polisei ir turētājs un īpašnieks. Klients parasti ir turētājs, kuram var būt vairākas apdrošināšanas polises, bet katrai apdrošināšanas polisei var būt savs īpašnieks (zīm. 5.4.). Klienta kartītē var mainīt datus par klientu, pievienot vienu no trim apdrošināšanas veidiem (visi apdrošināšanas veidi netiek piedāvāti, lai nesarežģītu izvēlni), kā arī aplūkot datus par visām jau piešķirtajām apdrošināšanas polisēm un vajadzības gadījumā aplūkot konkrēto polisi.



5.4.att. **Klienta kartīte**

5.4. Apdrošināšanas polises aplūkošana

Pēc nospiešanas uz konkrēto apdrošināšanas polisi (zīm. 5.4), atveras polises aplūkošanas/mainīšanas forma. Šeit var mainīt datus par īpašnieku un apdrošināšanas polisi.



5.5.att. **Apdrošināšanas polises aplūkošana**

Šajā piemērā atvērta *KASKO* apdrošināšana un lietotājam ir administratora tiesības (zīm.5.5). *KASKO* polise ir sadalīta divos maksājumos, kas redzami nodaļā „rēķini" , pie kam pirmā maksājuma veids ir „pārskaitījums", tādēļ tam ir pielikts automātisks numurs un šo maksājumu iespējams izprintēt *PDF* formātā (skat. 2. pielikumu), tāpat nosūtīt uz e-pastu. Visas šīs funkcijas pieejamas kolonā „Darbības". Tāpat KASKO apdrošināšanas polisi var pagarināt un veikt tai „pielikumu", ja gadījumā klients brauc uz ārzemēm un vēlas paplašināt polises darbības apgabalu.

Tikko būs saņemts pirmais maksājums, tā polises datu mainīšana parastam lietotājam tiks liegta, lai samazinātu kļūdas iespēju. Ja polise ir kļūdaina, tad to varēs atcelt, nospiežot pogu „anulēt" un aizpildot anulēšanas iemeslu, kura ievadīšana ir obligāta, lai pēc laika uzņēmuma vadītājs varētu uzzināt anulēšanas iemeslus.

Katram maksājuma ir lauks „Atskaites datums". Pēc šī lauka informācijas sistēma nosaka laika periodu, kurā iekrīt maksājums atskaišu ģenerācijas laikā. Parasti šis lauks ir vienāds ar lauku „Apmaksas datums", bet dažos gadījumos var atšķirties.

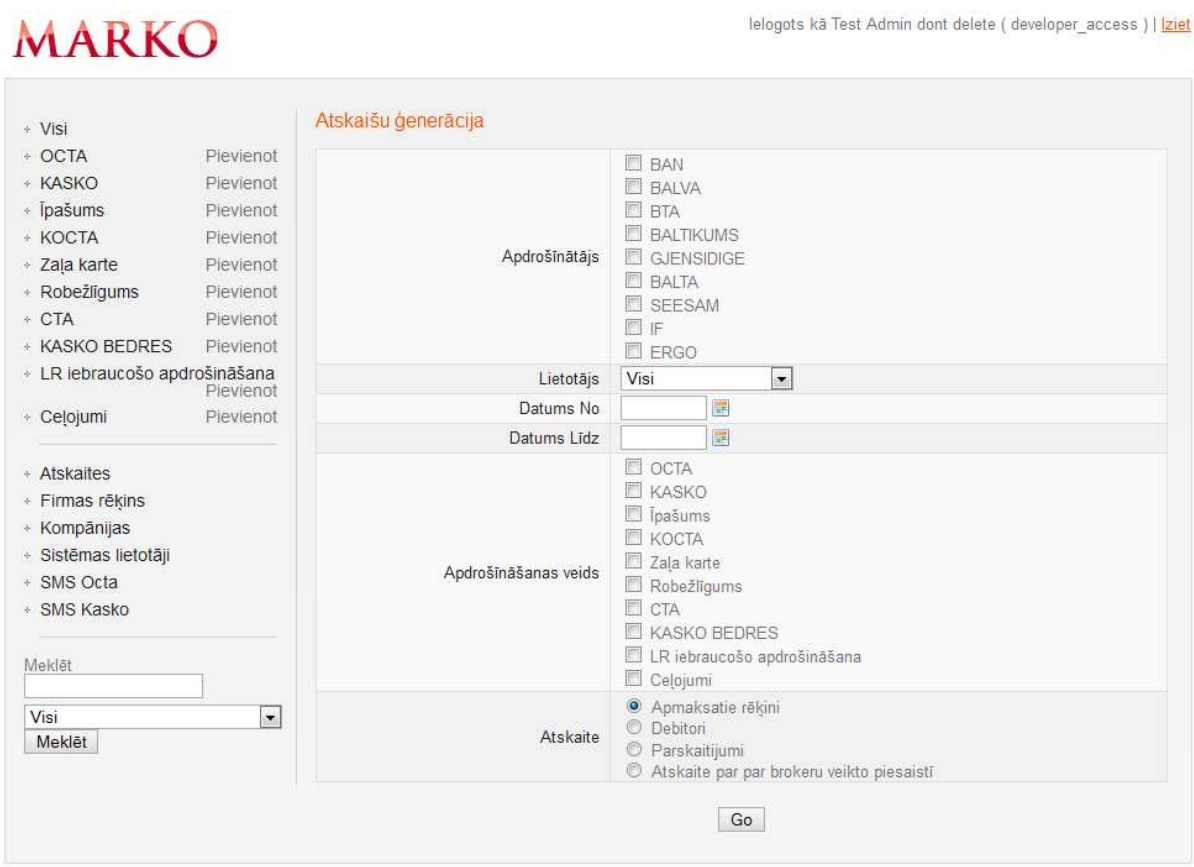
5.5. Atskaišu modulis

Viens no svarīgākajiem uzņēmuma kontroles moduļiem ir atskaišu sistēma. Informācijas sistēmā ir četri atskaišu veidi (5.6.att):

- **Apmaksātie rēķini** – atskaite izvada apmaksāto rēķinu sarakstu (skat. 3. pielikumu);
- **Debitori** – atskaite izvada datus par maksājumiem, kuriem beidzies apmaksas termiņš;
- **Pārskaitījumi** – atskaite izvada maksājumu sarakstu, kuri tikuši apmaksāti ar bankas pārskaitījumu;
- **Atskaite par brokeru veikto piesaisti** – atskaite izvada apmaksāto maksājumu skaitu un brokera peļņu no katra maksājuma. Šī atskaite tiek nodota apdrošināšanas firmai ar mērķi iegūt savu peļņas daļu (skat. 4. pielikumu).

Atskaišu ģenerācijas ērtībām un elastīgumam, katrā atskaitē var:

- Atzīmēt apdrošināšanas kompānijas, starp kurām notiks izvēle;
- Ģenerēt atskaiti pēc noteikta sistēmas lietotāja;
- Izvēlēties brīvu atskaites laika intervālu;
- Atzīmēt atskaitē iekļaujamos apdrošināšanas veidus;



5.6.att. **Atskaišu ģenerācijas forma**

6. AUTOMATIZĒTAS RĒĶINU APMAKSAS SISTĒMAS IEVIEŠANAS PROBLĒMAS

6.1. Cilvēciskais faktors

Kļūdas, kurām raksturīgs cilvēciskais faktors, raksturīgas gandrīz visam, tāpat tas ir arī realizētajā sistēmā. Problēma bija apstāklī, ka, pievienojot jaunu apdrošināšanas polisi un pēc tās apmaksas polises datus neviens vairs nespēja mainīt. Tas tika darīts ar uzņēmuma vadītāja ziņu, jo, pēc loģikas, pēc rēķina apmaksas polise tiek uzskatīta par aktīvu un to var tikai anulēt (no grāmatvedības viedokļa tas ir vienīgais pareizais variants). Tomēr daži sistēmas lietotāji ievadīja pareizus datus apdrošināšanas kompānijas sistēmā, bet realizējamā bilingvālajā sistēmā datus aizpildīja kļūdaini vai neaizpildīja vispār. Šādā gadījumā sistēmā parādījās nekorekta informācija, kuru nebija iespējams labot. Visbiežāk tas attiecās uz polišu numuriem, kurus ģenerē apdrošināšanas kompāniju sistēmas, kurus nepieciešams vienkārši dublēt. Pēc šiem gadījumiem bija tālruņa zvani mums (izstrādātājiem) ar lūgumu labot dažus datus, kas neietekmē grāmatvedības datus. Šī problēma tika atrisināta sekojoši:

- Dota iespēja administratoram mainīt nepieciešamos laukus;
- Īpaši apmācīt administratoru, izskaidrojot riska pakāpi.

6.2. Klienta informācijas dublēšana

Produkta ieviešanas laikā tika atklāta sekojoša problēma:

Daži uzņēmuma darbinieki, pievienojot jaunu polisi, nepārbaudīja klienta datu eksistenci sistēmā un ievadīja klienta datus no jauna, kā rezultātā parādījās viena klienta dublējoši ieraksti. Šī problēma tika atrasta tikai pēc kāda laika, kad šādu klientu skaits jau bija ievērojami pieaudzis. Problēmas risināšanai autors piedāvā:

- Pievienot papildus soli jaunas apdrošināšanas polises pievienošanai (zīm. 5.2.) – obligātu personas koda vai uzņēmuma reģistrācijas numura ievadīšanu. Pēc šī soļa notiek automātiska meklēšana pēc šiem datiem.
- Apvienot visas klientu – dublikātu grupas automātiskā režīmā.

6.3. Atgādinājuma īsziņu kontrole

Projekta izstrādes laikā netika veikta īsziņu izsūtīšanas kontrole, ņemot vērā to, ka kompānijai, kura piedāvāja īsziņu izsūtīšanas pakalpojumus, bija pašai sava izejošo īsziņu kontroles saskarne. Gadījumos, kad klients nav informēts par savas apdrošināšanas polises termiņa beigām, šī īsziņa ir viens no galvenajiem atgādinājumiem. Pēc šī pakalpojuma

ieviešanas bija gadījumi, ka klients iegādājies polisi un nav saņēmis brīdinājuma īsziņu par tās termiņa beigām, un ticis sodīts. Izrādījās, ka īsziņu pakalpojumu nodrošinošai kompānijai ir ļoti grūti sekot tam, kādas īsziņas kuram tikušas aizsūtītas. Risinājums bija atgādinājuma īsziņu kontrole, kas izskatījās šādi:

1. Informācijas sistēmā katrā polisē ir lauks „SMS statuss", kas uzrāda, vai īsziņa ir izsūtīta vai nē;
2. Tāpat uz uzņēmuma korporatīvo e-pastu katru dienu tiek izsūtīta atskaite ar detalizētu klientu sarakstu(zīm. 6.1).:

108 JURIS OSIPOVS 37127125122 OCTA
131 SERGEJS KOVIĻINS 37126441034 OCTA
979 EVIJA PUTNIŅA 37126090667 OCTA
2772 OLGA JEGOROVA 37129636666 OCTA
3335 JĀNIS GULBIS 37127156326 OCTA
3362 EDGARS LINGE 37129829239 OCTA
4978 MARIJA ANOŠČENKO 37129931300 OCTA
5018 SIA ASTERE 37129631022 OCTA
5028 VITĀLIJS JESINS 37129650100 OCTA
5970 DENISS OKUĻEVIČS 37122351835 OCTA
5971 MAKSIMS SMOĻAŅINOVS 37127639211 OCTA
6674 VLADIMIRS BIČKOVS 37129889962 OCTA
2336 JERJOMINA SVETLANA 37129840058 KASKO

6.1.att. **atskaite par nosūtītam atgadinājumiem**

7. RĒĶINU APMAKSAS SISTĒMAS ATTĪSTĪBAS PLĀNI

Rēķinu apmaksas sistēma jau ir ieviesta un uzņēmums to aktīvi īsteno. Protams, ka pēc sistēmas palaišanas vajadzēja ieviest dažus uzlabojumus, kas bija salīdzinoši viegli veicami, tomēr bija izmaiņas, kas nebija vienkāršas un to risinājumi bija globāli.

7.1. Kases aparāta pieslēgšana

Šai brīdī uzņēmumā apdrošināšanas polišu apmaksā notiek galvenokārt caur kases aparātu, uzņēmuma darbinieki ievada apmaksas datus ar roku. Par cik uzņēmums dienā pārdod diezgan lielu skaitu apdrošināšanas polišu (dienā tās var būt 50 – 60 polises, pārsvarā *OCTA*), uzņēmuma vadītājs vēlas šo procesu vienkāršot. Nepieciešams savienot kases aparātu ar bilingvālo sistēmu, tomēr pašlaik autoram nav nepieciešamās prakses kases aparāta pieslēgšanā.

7.2. Polises automātiskās iegādes internetā radīšana

Mūsdienās ļoti daudz apdrošināšanas brokeru piedāvā iegādāties polisi internetā, bez firmas darbinieku tiešas iejaukšanās. Uzņēmuma plānos ir šādas sistēmas ieviešana. Bez izstrādātās bilingvālās sistēmas iekļaušanas polišu iegāde internetā būtu samērā vienkārša. Problēma ir apstāklī, ka polišu iegādi internetā vajadzēs savienot ar izstrādāto sistēmu, lai dati par plisēs pirkšanu tiktu automātiski ievadīti klienta kartītē.

NOBEIGUMS UN SECINĀJUMI

Pēc automatizētas rēķinu sistēmas implementēšanas tiek būtiski saīsināts darījumprocesu izpildīšanas laiks. Uz šo brīdi visus darījumprocesus var viegli izpildīt tikai viens uzņēmuma darbinieks un viena darījumprocesa izpildīšanas laiks nepārsniedz desmit minūtes (tabula 1.1 atspoguļo darījumprocesu izpildīšanās laikus līdz sistēmas implementēšanai). Piekļuve pie izstrādātās automatizētās rēķinu sistēmas ir tikai uzņēmuma darbiniekiem t.i., sistēma paredzēta iekšējai lietošanai, tāpēc nopirkto apdrošināšanas polišu skaits nepalielinājās, bet būtiski saīsinājās viena klienta apstrādes laiks. Nākotnē, pēc automātiskās polises iegādes internetā realizācijas, šo rādītāju plānots paaugstināt. Klients ir ļoti apmierināts ar izstrādāto sistēmu, jo tagad viņš var pārskatīt notikušos pirkumus uzņēmumā, var analizēt pārdošanas datus, izdarīt pārdošanas efektivitātes secinājumus un pieņemt lēmumus. Tāpat tika optimizēti tādi izdevumi, kā īsziņas atgādinājumi (tagad nevajag papildus maksāt darbiniekam, lai viņš manuāli sūtītu atgādinājumus), un šīs process tiek pilnīgi automatizēts un pilnīgi izslēdz cilvēkfaktora kļūdas.

Apdrošināšanas brokera darbības princips ir relatīvi vienveidīgs un standartizēts, tāpēc arī ir plāni piedāvāt šo sistēmu citiem apdrošināšanas brokeriem.

Automatizētas rēķinu sistēmas kopija apskatama šeit:

- Adrese: http://markotest.trialine.lv
- Lietotājs: developer_access
- Parole: developer_access
- Visi dati ir reāli, bet to var mainīt, jo tā ir sistēmas kopija

Pēc rēķinu apmaksas sistēmas funkcionālās prasības sastādīšanas, nākamais solis bija izstrādes rīku izvēle. Sakarā ar realizējamās rēķinu apmaksas sistēmas specifiku, neviens no gataviem pieejamiem *PHP* valodas risinājumiem pilnīgi neatbilst sistēmas prasībām. Sistēmas izstrādāšana ar PHP valodu „tīrā" veidā tika izslēgta, jo šīs risinājums ir ļoti darbietilpīgs process. Autors nolēma izstrādāt sistēmu ar vienu no daudzajiem PHP ietvariem. Maģistra darba autoram ir desmit gadu pieredze tīmekļa lietojumprogrammas izstrādāšanā *PHP* valodā un pēdējais ietvars, kurš pievērsa autora uzmanību, ir diezgan jauns, bet ļoti perspektīvs ietvars *Yii*, kurš, pēc viena no tā autoru vārdiem, tika radīts ar mērķi ātri risināt reālus uzdevumus. Iepazīšanās ar ietvaru sākas no *WebConf 2010* konferences, kura norisinājās Rīgā, novembra sākumā. Tā arsenālā ir viss nepieciešamais, lai varētu ātri radīt tīmekļa lietojumprogrammu (4.4. punkts). Šīm ietvaram diezgan zems ieiešanas slieksnis, precīza

dokumentācija un dominējoša pozīcija ātruma rādītājos (3.3.att). Šai brīdī sistēmā glabājas apmēram septiņi tūkstoši lietotāji, vienlaicīgi lieto trīs darbinieki un nekādas ātrdarbības problēmas nav pamanītas. Izstrādājot ar *Yii* ietvara palīdzību, tiek radīts ļoti elegants un viegli pārskatāms lietojumprogrammas pirmkods. Pēc viena no autoritāriem Krievijas federācijas IT interneta resursa datiem(http://habrahabr.ru), Yii ietvars ieņem otro pozīciju (8.1. att) pēc ietvaru popularitātes, neskatoties uz savu vecumu(*Yii* pirmā versija tiek radīta 2008. Gadā sākumā).[20]

8.1.att **Ietvaru popularitāte**

Ar *Yii* ietvaru palīdzību, izstrādes komanda sasniedza visus uzstādītos mērķus, un turpmākos līdzīgos projektos autors izmantos tieši šo ietvaru kā pamatu sistēmas izstrādāšanā.

Lietojumprogrammas realizācijas gaitā tika atrasts elegants risinājums *PDF* failu ģenerācijā *UNIX* konsoles programmas *wkhtmltopdf* veidā, kurā viegli un ātri konvertē jebkādu *HTML* dokumentu *PDF* formātā.

Pamatojoties uz savu desmit gadu pieredzi tēmekļa lietojumprogrammas izstrādē, es ar pārliecību varu teikt, ka Yii ietvars ir labākais ko esmu biju redzējis savējā profesionālajā mūžā *PHP* ietvaru klāstā.

IZMANTOTA LITERATŪRA UN AVOTI

Tīmekļa vietnes un elektroniskie izdevumi:

1) *„Учебник PHP”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://256bit.ru/education/php4/introduction.html
2) *„History of PHP”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://php.net/manual/en/history.php.php
3) *„PHP 4 end of life announcement”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://www.php.net/archive/2007.php#2007-07-13-1
4) *„PHP 6 - Unicode Completion Stats”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://www.php.net/~scoates/unicode/render_func_data.php
5) *„PHP Wiki”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://wiki.php.net/todo/php60
6) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „Фреймворк”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/Фреймворк
7) *„Как создать свой сайт на PHP? Или зачем нужны фреймворки?”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://www.simplecoding.org/kak-sozdat-svoy-site.html
8) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „Zend Framework”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/Zend_Framework
9) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „CakePHP”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/CakePHP
10) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „CodeIgniter”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/Code_Igniter
11) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „Kohana”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/Kohana
12) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts „Symfony”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/Symfony
13) *Aleksandrs Makarovs, „Ietvaru salidzināšanās tabūla”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://rmcreative.ru/playground/php-frameworks/
14) *„Yii ievāds”* [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://yiiframework.ru/doc/guide/ru/quickstart.what-is-yii
15) „Performance of Yii” [tiešsaiste] – [atsauce 17.03.2011]. Pieejams internetā: http://www.yiiframework.com/performance/

16) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts* „CRUD” [tiešsaiste] – [atsauce 02.05.2011]. Pieejams internetā: http://ru.wikipedia.org/wiki/CRUD
17) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts* „Scaffolding”[tiešsaiste] – [atsauce 02.05.2011]. Pieejams internetā: http://en.wikipedia.org/wiki/Scaffold_(programming)
18) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts* „Xvfb”[tiešsaiste] – [atsauce 02.05.2011]. Pieejams internetā: http://en.wikipedia.org/wiki/Xvfb
19) *Materiāls no Vikipēdijas – brīvas enciklopēdijas, raksts* „CURL” bibliotēka[tiešsaiste] – [atsauce 02.05.2011]. Pieejams internetā: http://en.wikipedia.org/wiki/CURL
20) „Какой PHP фреймворк вы используете? И почему?”[tiešsaiste] – [atsauce 02.05.2011]. Pieejams internetā: http://habrahabr.ru/blogs/php/116030/

Maģistra programmas kursa darbs

21) Osovitnijs, A. Tīmekļa lietojumprogrammu izstrādes ietvars YII: kursa darbs. Datorikas fakultāte. Rīga: Latvijas Universitāte, 20010. 30 lpp.

PIELIKUMI

Maģistra darbam ir šādi pielikumi:

1) Noteikta apdrošināšanas veida lauku saraksts
2) Automātiski ģenerējamais rēķins *PDF* formātā
3) „Apmaksātie rēķini” atskaites izskats
4) Atskaite par brokeru veikto piesaisti
5) Modeļa *LoginForm* pilns pirmkods
6) Klases *UserIdentity* pilns pirmkods

1. pielikums

Noteikta apdrošināšanas veida lauku saraksts

	OCTA	**KASKO**	**Īpašums**	**KOCTA**	**Zaļā karte**	**Robežlīgums**	**CTA**	**KASKO bedres**	**LR iebraucošo apdrošināšana**	**Ceļojumi**
Klienta personas dati		x	x							
Līguma numurs(nav ID)	x	x	x	X	X	X	X	x	x	x
Apdrošināšanas uzņēmums (izkritošs lauks)	x	x	x	X	X	X	X	x	x	x
Slēgšanās datums	x	x	x	X	X	X	X	x	x	x
Sākuma datums	x	x	x	X	X	X	X	x	x	x
Beigas datums	x	x	x	X	X	X	X	x	x	x
Automobiļa reģistrācijas numurs	x	x		X	X					
Tehniskas pases numurs	x	x		X	x					
Marka		x								
Modelis		x								
Izlaiduma gads		x								
Tālrunis	x	x								
Teritorija (Latvija, Baltija, Eur, NVS)		x								
Fotogrāfijas (līdz 10)		x								
Pašrisks, pilns sabrukums, %		x								
Pašrisks, zādzība, %		x								
Pašrisks, bojājumi, %		x								
Pašrisks, bojājumi, summai jābūt ne mazākai par		x								
Visjaunākais lietotājs		x								
Vismazākais stāžs		x								
Apdrošināšanas summa		x	x							

Apdrošināšanas premija – cik maksā polise		x	x							
Komentāri	x	x		x	X	X	x	x	x	x
Maksājumu skaits	1	x	x	1	1	1	1	1	1	1
Apdrošināšanas objekta adrese			x							
Polises apdrošināšanas laiks (lauks tiek rēķināts automātiski vai ierakstās manuāli)	x	x	x	x	X					
Apmaksas veids (izkrītošs lauks, skaidrā nauda vai pārskaitījums)	x	x	x	x	X	x	x	x	x	x
Operators	x	x	x	x	x	x	x	x	x	x

2. pielikums

Automātiski ģenerējamais rēķins PDF formātā

MARKO

LUBĀNAS IELA 129A, RĪGA, LV1021
25987070 | e-pasts: info@markopolise.lv

DATUMS: **23.12.2010**
RĒĶINS Nr.: **2010-227**

MAKSĀTĀJS:	SIA DK-TRANSPORT	**REĶ. NR:** 2010-227
ADRESE:	LIELUPES 66-30, RĪGA	
PERS.KODS/REG.NUM.:	50003989971	
BANKA:		
KONTS:		
KODS:		

SAŅĒMĒJS:	SIA MARKO BROKERI	**REĶ. NR:** 2010-227
ADRESE:	LUBĀNAS IELA 129A, RĪGA, LV1021	
PERS.KODS/REG.NUM.:	50003789611	
BANKA:	Swedbank AS	
KONTS:	LV46HABA0551026940648	
KODS:	HABALV22	

Gadījumā, ja Jums rodas sarežģījumi ar kārtējā rēķina apmaksu, lūdzam laicīgi ziņot uzskaitvedības nodaļai pa telefonu 67136366

APMAKSAS TERMIŅŠ:	**23.12.2010**
RĒĶINA VALŪTA:	**LVL**

PAKALPOJUMA NOSAUKUMS	SUMMA
Apdrošināšanas prēmija par KASKO polisi a/m P3409	**35.00**
Kopā samaksai LVL:	**35.00**

MARKO BROKERI SIA **Natalja Iriste**

Paraksts _____ **Z.V.**

Atskaite par apmaksatiem rēķinem (Rēkinu skaits: 5)

04.03.2011 - 06.03.2011

BALTIJAS APDROŠINĀŠANAS NAMS AAS

N.p.k	Polises Nr.	Kvit. Nr.\Rek. Nr	Nosaukums	Slegšanas datums	Veids	Valūta	Kopēja premija	Tālrunis	Apmaksas datums	Komisija, %	Komisija, Ls	Apm. veids	Komentāri
1	NB195289	AB86195	MIHAILS SAPUNOVS	04.03.2011	OCTA	LVL	42.92	29868678	2011-03-04	8%	3.43	Skaidrā nauda	281
2	NB195430	AB86247	BORISS RIŽOVS	05.03.2011	OCTA	LVL	8.79	26543220	2011-03-05	8%	0.70	Skaidrā nauda	283
						Kopā:	**51.71**			**Kopā:**	**4.14**		
						Kopā - Skaidrā nauda:	**51.71**						

IF P&C INSURANCE AS

N.p.k	Polises Nr.	Kvit. Nr.\Rek. Nr	Nosaukums	Slegšanas datums	Veids	Valūta	Kopēja premija	Tālrunis	Apmaksas datums	Komisija, %	Komisija, Ls	Apm. veids	Komentāri
1	SD161206	AB86242	KANUŅNIKOVS VALĒRIJS	05.03.2011	OCTA	LVL	21.91	26565160	2011-03-05	8%	1.75	Skaidrā nauda	283
2	SD161318	ZU86217	STEPANS DZJAMULIČS	06.03.2011	OCTA	LVL	21.18	65210402	2011-03-06	8%	1.69	Skaidrā nauda	282
						Kopā:	**43.09**			**Kopā:**	**3.45**		
						Kopā - Skaidrā nauda:	**43.09**						

BTA AAS

N.p.k	Polises Nr.	Kvit. Nr.\Rek. Nr	Nosaukums	Slegšanas datums	Veids	Valūta	Kopēja premija	Tālrunis	Apmaksas datums	Komisija, %	Komisija, Ls	Apm. veids	Komentāri
1	DB600991	AB86220	SERGEJS VASIĻJEVS	06.03.2011	OCTA	LVL	32.03	29541378	2011-03-06	10%	3.20	Skaidrā nauda	282
						Kopā:	**32.03**			**Kopā:**	**3.20**		
						Kopā - Skaidrā nauda:	**32.03**						

Pavisam kopā:	**126.83**
Pavisam kopā - Skaidrā nauda:	**126.83**
Pavisam kopā komisija:	**10.79**

Atskaite par par brokeru veikto piesaisti Nr.

SIA MARKO BROKERI, reg. Nr. 50003789611

(Brokers, tā reģ. Nr.)

par periodu: 04.03.2011 - 06.03.2011

BALVA AAS

N.p.k	Polises Nr.	Klients	Izdošanas datums	Derīgs no	Apdr. veids	Valūta	Kopēja premija	Brokera %	Brokera nauda	Kompanijas nauda
1	CD254498	PODOSINOVSKA JEKATERINA	04.03.2011	04.03.2011	OCTA	LVL	25.10	7%	1.76	23.34
2	CD254572	JEVGEŅIJS KOMARŅICKIS	04.03.2011	04.03.2011	OCTA	LVL	26.47	7%	1.85	24.62
3	CD254657	VJAČESLAVS DOMBROVSKIS	04.03.2011	05.03.2011	OCTA	LVL	9.65	7%	0.68	8.97
4	CD254798	OĻEGS BURAKOVS	05.03.2011	07.03.2011	OCTA	LVL	23.68	7%	1.66	22.02
5	CD254846	ARKADIJS MAČEHINS	05.03.2011	05.03.2011	OCTA	LVL	7.92	7%	0.55	7.37
6	CD254849	ĒRIKS PUMPA	05.03.2011	05.03.2011	OCTA	LVL	5.68	7%	0.40	5.28
7	CD254867	SIA TRIALTER	05.03.2011	05.03.2011	OCTA	LVL	23.50	7%	1.65	21.86
8	CD254900	DAMIRS AĻUŠEVS	05.03.2011	05.03.2011	OCTA	LVL	30.30	7%	2.12	28.18
9	CD254910	NATĀLIJA SOLOMEIČUKA	05.03.2011	05.03.2011	OCTA	LVL	8.96	7%	0.63	8.33
10	CD254987	ULDIS MUIŽNIEKS	06.03.2011	08.03.2011	OCTA	LVL	2.50	7%	0.17	2.33
							163.76	**Kopā:**	**11.46**	**152.30**

Modeļa LoginForm pilns pirmkods

```
<?php

/**
 * LoginForm class.
 * LoginForm is the data structure for keeping
 * user login form data. It is used by the 'login' action of 'SiteController'.
 */
class LoginForm extends CFormModel
{
	public $username;
	public $password;
	public $rememberMe;

	private $_identity;

	/**
	 * Declares the validation rules.
	 * The rules state that username and password are required,
	 * and password needs to be authenticated.
	 */
	public function rules()
	{
		return array(
			// username and password are required
			array('username, password', 'required'),
			// rememberMe needs to be a boolean
			array('rememberMe', 'boolean'),
			// password needs to be authenticated
			array('password', 'authenticate'),
		);
	}

	/**
	 * Declares attribute labels.
	 */
	public function attributeLabels()
	{
		return array(
			'rememberMe'=>'Atcēreties mani',
		);
	}

	/**
	 * Authenticates the password.
	 * This is the 'authenticate' validator as declared in rules().
	 */
	public function authenticate($attribute,$params)
	{
		if(!$this->hasErrors())  // we only want to authenticate when no input errors
		{
			$this->_identity=new UserIdentity($this->username,$this->password);
			$this->_identity->authenticate();
			switch($this->_identity->errorCode)
			{
				case UserIdentity::ERROR_USERNAME_INVALID:
					$this->addError('username','Lietotāja vārds/Parole ir nepareiza!');
					break;
				case UserIdentity::ERROR_PASSWORD_INVALID:
					$this->addError('password','Lietotāja vārds/Parole ir nepareiza!');
```

```
                                        break;
                                default:
                                        break;
                                }
                        }
                }

                /**
                 * Logs in the user using the given username and password in the model.
                 * @return boolean whether login is successful
                 */
                public function login()
                {
                        if($this->_identity===null)
                        {
                                $this->_identity=new UserIdentity($this->username,$this->password);
                                $this->_identity->authenticate();
                        }
                        if($this->_identity->errorCode===UserIdentity::ERROR_NONE)
                        {
                                $duration=$this->rememberMe ? 3600*24*30 : 0; // 30 days
                                Yii::app()->user->login($this->_identity,$duration);
                                return true;
                        }
                        else
                                return false;
                }
}
```

6. pielikums

Klases UserIdentity pilns pirmkods

```
<?php

/**
 * UserIdentity represents the data needed to identity a user.
 * It contains the authentication method that checks if the provided
 * data can identity the user.
 */
class UserIdentity extends CUserIdentity{
        // private $_id;

   public function authenticate(){
      $username=strtolower($this->username);
      $user=User::model()->find('LOWER(login)=?',array($username));
      if($user===null)
         $this->errorCode=self::ERROR_USERNAME_INVALID;
      else if($user->password != $this->password)
         $this->errorCode=self::ERROR_PASSWORD_INVALID;
      else{
         // $this->_id=$user->iduser;
         // $this->username=$user->login;

                                $this->setState('access', $user->access);
                                $this->setState('creds', $user->name.' '.$user->surname. ' ( '.$user->login.' )');
         $this->errorCode=self::ERROR_NONE;
      }
      return $this->errorCode==self::ERROR_NONE;
   }

}
```