LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

ASTERISK TEHNOLOĢIJU IZMANTOŠANA INFOLINE SISTĒMAS IZSTRĀDĒ

BAKALAURA DARBS

Autore: Jekaterina Ļimoņina

Studenta Apl.Nr: jl06013

Darba vadītājs: M. dat. Vadims Potehins

RĪGA 2010

Anotācija

Bakalaura darba pētījuma joma ir atklātā pirmkoda telefonijas platforma Asterisk.

Darba teorētiskais mērķis ir izpētīt Asterisk koncepcijas un piedāvātas iespējas un risinājumus, lai uz to bāzes varētu realizēt automatizētu telefonbanku sistēmu.

Darba praktiskais mērķis ir Infoline sistēmas projektēšana un izstrāde, pielietojot iegūtās zināšanas un prasmes. Infoline ir automātiskais telefona dienests, kas, izmantojot datus, kurus ievada abonents ar tonālu ievadi, maršrutē viņu pa izvēlnēm un ļauj viņam iegūt vajadzīgo informāciju, kura tiek nodota abonentam ar iepriekš ierakstīto balss ziņojumu palīdzību.

Rezultātā Infoline automatizēta uzziņu sistēma tika veiksmīgi realizēta un jau gadu darbojas Norvik bankās Krievijā.

Abstract

The Bachelor's Paper research area is an open source converged telephony platform Asterisk.

This work's theoretical purpose is to make a research on Asterisk concepts and provided possibilities and solutions, based on which an automated bank-by-phone system could be implemented.

This work's practical purpose is to design and develop an Infoline system using gained knowledge and skills. Infoline is a phone automated service that, using the data entered by caller by tonal input, routes the caller through menus and allows him to get necessary information, which is provided to caller by voice messages recorded in advance.

As a result, the automated Infoline information system was successfully implemented and has been working in Norvik bank in Russia for about a year.

Atslēgvārdi

Telefonija

Asterisk

AGI

IVR

dialplāns

Infoline

telefonbanka

automatizēta telefona uzziņu sistēma

Satura radītājs

APZĪMĒJUMU UN SAĪSINĀJUMU SARAKSTS

AGI – (Asterisk Gateway Interface) – Asterisk vārtejas saskarne

AMI – (Asterisk Manager Interface) – Asterisk pārvaldes saskarne

API – (Application Programming Interface) – lietojumu programmēšanas interfeiss

DTMF – (Dual-tone multi-frequency signaling) – divtoņu daudzfrekvenču signalizācija

IP – (Internet Protocol) – intertīkla protokols

IVR – (Interactive Voice Response) – interaktīva balss reaģēšana

ODBC – (Open Database Connectivity) – atvērtā datu bāzu savienojamība

PBX – (Private Branch Exchange) – privātā atzara centrāle

PSTN – (Public Switched Telephone Network) – publiskais komutējamais telefonu tīkls

SIP – (Session Initiation Protocol) – sesijas inicializācijas protokols

SMS – (Short Message Service) – īsziņu pakalpojums

VoIP – (Voice over IP) – IP balss pārraide

IEVADS

Darba mērķis ir iepazīsties ar atvērtas telefonijas platformas Asterisk iespējām, izveidot uz tās bāzes Infoline informācijas-uzziņu automatizētu telefonijas sistēmu un veikt to projektēšanas un realizācijas izpēti, kā arī sniegt secinājumus un priekšlikumus par jau izveidotu sistēmu.

Infoline ir automatizēta telefonbanka sistēma, kas ļauj bankas klientiem, zvanot pa tālruni, uzzināt informāciju par savu bankas kontu un veikt dažādas operācijas, tādas kā kartes bloķēšana. Informācija izsaucošam abonentam tiek sniegta ar iepriekš ierakstīto balsu ziņojumu palīdzību. Abonenta maršrutēšana pa izvēlnēm un datu ievadīšana notiek ar tonālu ievadi.

Infoline sistēma ir reāli izstrādāta ar darba autora dalību un jau gadu darbojas Krievijas Norvik bankās. Šajā brīdī tā tiek paplašināta ar jauno funkcionalitāti – iespēju veikt maksājumus pa telefonu.

Šāda tipa sistēmu izveide ir ļoti aktuāls uzdevums, jo tas ļauj, pirmkārt, automatizēt cilvēku darbu, noņemot slodzi no operatoriem, un, otrkārt, piedāvāt sistēmas klientiem diennakts ātru piekļuvi nepieciešamām funkcijām un informācijai, negaidot brīvu operatoru.

Darbs sastāv no 6 nodaļām.

Pirmajā tiek noformulēta problēma un aprakstītas prasības Infoline sistēmai, kā arī tiek dots plašāks priekšstats par pašu sistēmu.

Otrā nodaļa iepazīstina lasītājus ar IP telefonijas vēsturi, lai palīdzētu viņiem izprast, cik nesen tā parādījās, cik strauji attīstījās, kādas problēmas tā risina un kādas tai ir perspektīvas. Tiek arī īsi aplūkotas citas telefonijas platformas ar atvērtu pirmkodu.

Trešā nodaļa ir veltīta Asterisk platformai un tās plašām iespējām: tiek stāstīts par tādiem Asterisk jaudīgiem un elastīgiem instrumentiem kā dialplāns un AGI, par tādu funkcionalitāti kā balss pasts, konferenču organizēšana, tekstu pārvēršana runā, u.c.

Ceturtajā nodaļā tiek sniegts reāli strādājošs Infoline sistēmas risinājums, kurš tika izstrādāts un implementēts ar darba autora palīdzību, pielietojot iepriekšējās sadaļās aprakstītās zināšanas. Šeit tiek dotas arī projektēšanas rekomendācijas un metodikas, kas labi kalpoja šīs sistēmas izstrādē, un kuras, kā autors ļoti cer, būs lietderīgas visiem, kas apņemsies veidot līdzīgu sistēmu.

Piektajā nodaļā tiek aprakstīti nākotnes plāni Infoline sistēmai, kuri ietver sevī kā jaunas funkcionalitātes ieviešanu jau strādājošai sistēmai, tā arī Infoline ieviešanu citā bankā.

Sestajā nodaļā tiek prezentēti galīgie rezultāti un formulēti secinājumi par izstrādāto sistēmu

1. PROBLĒMAS FORMULĒJUMS

1.1. Iepriekšējā situācija

Mūsdienu informāciju un augstu tehnoloģiju pasaulē kompānijām, firmām, bankām un citiem iestādēm, kas piedāvā pakalpojumus, bieži rodas nepieciešamība piedāvāt klientiem iespēju uzzināt kādu informāciju pa tālruni vai caur SMS, jo telefons ir viens no populārākiem komunikācijas līdzekļiem, kas ir pieejams katram cilvēkam. Mūsu valstī, piemēram, šo servisu bieži piedāvā mobilie operatori: LMT, Tele2, Bite, – lai ļautu saviem klientiem saņemt informāciju par kartes atlikumu, mainīt tarifu plānu, pieslēgt jaunus pakalpojumus, utt.

Autors strādā bankā, un bankas klientiem ir iespēja saņemt informāciju par savu kontu ar SMS palīdzību. Informācijas iegūšana var notikt vai nu pēc iepriekš definēta notikuma – piemēram, SMS saņemšana pēc katras veiksmīgi notikušas banka transakcijas, - vai nu pēc klienta pieprasījuma: sūtot SMS ar pareizi noformētu vaicājumu, viņš saņem attiecīgu rezultātu – piemēram, bankas konta atlikumu. SMS sūtīšanas informatīvas sistēmas apraksts un realizācija ir apskatāmas autora kvalifikācijas darbā [3].

Bet informāciju saņemšana caur SMS ir vairāk piemērota jaunatnei, jo to rakstīšana prasa noteiktas iemaņas. Zināmas grūtības var rast arī pareizu iestatījumu veidošana internetbankā, lai saņemtu SMS pēc noteiktiem notikumiem, kā arī zināšanu iegūšana par SMS vaicājumu formātiem, lai saņemtu vajadzīgo informāciju. Tādejādi rodas nepieciešamība izveidot informatīvu-uzziņu telefonbanku sistēmu plašākai auditorijai, kas būtu vieglāk lietojama un saprotama. Līdz tam brīdim informāciju pa telefonu varēja uzzināt tikai, zvanot operatoriem, kas prasa daudz viņu darba laika, un rezultātā banka tērēja naudu lietai, kuru varēja automatizēt un izdarīt efektīvāk.

1.2. Problēmas pārskats

Izstrādājamās sistēmas mērķis bija aizvietot operatorus, sniedzot bankas klientiem informāciju automatizēti – noskaņojot viņiem iepriekš ierakstītas balsu ziņojumus. Komunicēšana ar klientu notiek caur tonālu ievadi – lai pārietu uz citu izvēlni, vai izvēlētos kādu funkciju, vai ievadītu kādu informāciju, klientam ir jānospiež attiecīgie cipari uz

telefona tastatūras. Klientam ir jānodod norādījumi izvēlnes veidā, lai viņam būtu skaidrs, kāda poga ir jānospiež, lai iegūtu vēlamu rezultātu.

„Infoline" automatizētas informāciju-uzziņas sistēmas pieprasītas funkcijas:

- Klienta savienošana ar operatoru – ja klientam rodas kādi jautājumi, vai ir nepieciešama personiska palīdzība;
- Klienta autorizācija sistēmā – notiek ar pēdējiem četriem klienta pases cipariem un četriem pēdējiem kartes cipariem;
- Kartes statusa noskaidrošana – vai ir bloķēta, vai nav;
- Kartes pieejama atlikuma noskaidrošana – atlikums kartes esošā valūtā;
- Kartes pēdējas operācijas noskaidrošana – pēdējas debeta un kredīta operācijas datums, summa un valūta;
- Kartes termiņa beigu noskaidrošana – kartes izbeigšanas termiņa mēnesis un gads;
- E-pasta saņemšana par pēdējām kartes operācijām – desmit pēdējas debeta un desmit pēdējas kredīta kartes operācijas;
- Kartes bloķēšana – ar apstiprinājuma pieprasījumu.

Realizācijai bija jāizmanto Asterisk platforma, jo tā jau bija plaši izmantota pie pasūtītajiem. Tā kā autoram iepriekš nekad nenācās sastapties ne ar Asterisk platformu, ne ar telefoniju vispār, tad pirmais uzdevums bija izpētīt tos, lai gūtu priekšstatu par visām tās iespējām un varētu izdomāt visefektīvāko realizācijas veidu, kuru pēc tām varētu izmantot atkārtoti līdzīgas sistēmas veidošanā, jo uzdevums bija iegūt nevis vienreizēju risinājumu, bet viegli uzturamu un modificējamu, kuru varētu pēc tām adaptēt vai viegli pārrakstīt citam līdzīgam uzdevumam. Tāpēc daudz uzmanības darbā tika veltīts tieši sistēmas projektēšanai un labas projektēšanas pamatprincipiem. Nedrīkstēja arī izmantot gatavus risinājumus vai gatavas bibliotēkas – bija uzdevums izveidot no nulles savu risinājumu, kas

nodrošinātu vienotu pieeju un atvieglotu projekta atkļūdošanu un modificēšanu, un pēc tām ļautu veidot jaunus risinājumus uz savu labi pazīstamu un saprotamu moduļu bāzes.

Svarīgi piebilst, ka Asterisk administrēšanas jautājumi šajā darbā nav apskatīti: telefonijas standarti, kanāli un protokoli ir ārpus šī darba robežām. Tā kā pasūtītājiem jau bija organizēts izsaukumu centrs, kas balstīts uz Asterisk platformu, nebija nepieciešamības iedziļināties to iestādīšanā un konfigurēšanā, jo risinājums bija jāintegrē jau gatavā un strādājošā sistēmā.

2. IESKATS IP TELEFONIJĀ

2.1. IP telefonijas vēsture

IP telefonija parādījās samērā nesen un attīstījās ļoti strauji [11]:

- 1993. gadā ASV tika izlaista pirmā programma balss pārraidei pa tīklu – Maven. Šis notikums tika uzskatīts par pirmo IP telefonijas koncepta parādīšanos vēsturē. Tajā pašā laikā kļuva populāra videokonferenču lietotne priekš Macintosh – CU-SeeMe.

- 1994. gadā Endeavor pavadoņa lidojuma laikā NASA pārraidīja Zemei savu attēlu ar CU-SeeMe palīdzību un vienlaikus ar Maven palīdzību – arī skaņu. Pēc šī notikuma abas programmas tika integrētas un radīja CU-SeeMe variantu ar audio un video iespējām vienlaikus.

- 1995. gadā notika ļoti svarīgs posms IP telefonijas vēsturē – kompānija VocalTec izlaida Internet Phone lietojumu, kas bija pirmais vēsturē risinājums, kurš ļāva lietotājiem veikt telefona zvanus caur Internetu. Pēc izlaistā produkta veiksmes citas kompānijas, novērtējot šīs tehnoloģijas perspektīvumu un labumu, sāka izlaist arī savus risinājumus.

- 1995. gada septembrī parādījās pirmais interaktīvais IP telefonijas risinājums – DigiPhone lietotne, kura piedāvāja dupleksa iespējas, ļaujot vienlaicīgi klausīties un runāt.

- 1996. gadā tika izsludināts VocalTec un Dialogic kompāniju kopējais projekts „Internet Telephone Gateway", kura mērķis bija izveidot speciālu vārteju starp Internet tīklu un PSTN, kurš ļautu parastajam telefonu aparātam strādāt caur Internetu. Vārteja veidā darbojās specializēta lietotne, kura izmantoja balss plati kā saskarni ar parastajām telefona līnijām un pārveidoja parastos telefona numurus IP adresēs un atpakaļ.

- 1997. gadā savienojums starp diviem attālinātiem telefona abonentiem caur Internetu kļuva par ikdienišķu lietu.

Bet, neskatoties uz acīmredzamu veiksmi, IP telefonija vēl neaizstāja tradicionālo telefoniju. Tomēr tā ieņēma savu pelnīto vietu, it īpaši korporatīvajā sektorā, kur ir svarīgi apvienot visas informācijas plūsmas vienā sakaru kanālā.

2.2. IP telefonijas principi

IP telefonija ir tehnoloģija, kas nodrošina balss sakarus caur datu pārraides tīkliem. Tās pamats ir balss sakaru konvertēšana datu paketēs caur ierīcēm, kas pieslēgtas pie IP-tīkla, tādām kā telefona aparāts vai dators.

IP-telefonijas tīkls sastāv no sekojošā ierīču kopuma:

- **Vārteja (*gateway*)** –pie IP-tīkla un pie telefonu tīkla (PBX/PSTN) pieslēgta ierīce. Šī ir galvenā IP-telefonijas arhitektūras sastāvdaļa, kas savieno telefonu tīklu ar IP tīklu un kuras funkcijas ir:
  - fiziska saskarne ar telefonu tīklu un IP tīklu,
  - balss saspiešana, pakotēšana un atjaunošana,
  - balss pakotņu pārsūtīšana,
  - savienošanās uzstādīšana un pārtraukšana ar PBX/PSTN izsaucēju abonentu,
  - savienošanās uzstādīšana ar attālināto vārteju;
- **Dispečers (*gatekeeper*)** – pie IP-tīkla pieslēgta ierīce, kas satur visu IP-telefonijas darbības loģiku. Tās funkcijas ir:
  - abonenta autentifikācija un autorizācija,
  - izsaukumu sadalījums starp vārtejām;

- **Monitors (*administration manager*)** – neobligāts pie IP-tīkla pieslēgts modulis, kas ir izmantots citu tīkla ierīču attālinātai konfigurēšanai un administrēšanai.

Tātad, IP-tīkla arhitektūra ir sekojoša: pie IP-tīkla pieslēgtas vārtejas nodrošina saskarni abonentam un izpilda balss saspiešanas, pakotēšanas un atjaunošanas funkcijas; vārteju sadarbību nodrošina dispečeri; savukārt monitori var būt izmantoti vārteju, dispečeru un citu tīkla ierīču attālinātai konfigurēšanai un administrēšanai.

2.3. Esošu telefonijas platformu pārskats

Šajā nodaļā pieminēsim dažas populāras atvērta pirmkoda PBX telefonijas risinājumus un to atšķirības[10]:

- **Asterisk** – izpilda visas tradicionālās telefonijas sistēmas funkcijas un daudz vairāk;
- **sipX** – Asterisk-bāzēts risinājums, kas strādā tikai ar SIP protokolu;
- **Elastix** – piedāvā ērtu tīmekļa saskarni izsaukuma centra izveidošanai, kontrolēšanai, atskaišu veidošanai, IVR konfigurēšanai;
- **FreeSWITCH** – koncentrē uzmanību uz modularitāti, vairākplatformu atbalstu, mērogojamību un stabilitāti;
- **OpenPBX** (**CallWeaver**) – modulāra arhitektūra, iebūvēts TDM karšu atbalsts;
- **CallButler** – Windows-bāzēta atvērta pirmkoda PBX, implementēta uz .NET platformas;
- **CoreDial VoiceAxis** – projektēta priekš PBX menedžmenta, tarifikācijas un apgādes. Brīnišķīgs rīks Asterisk lietotājam mazos un lielos uzņēmumos, lai iegūtu mērogojamu, efektīvu un rentablu PBX risinājumu;

- **Fonality trixbox Pro** – izmanto tā saukto hibrīd-hostētu arhitektūru un piedāvā bezmaksus VoIP zvanus un monitoringu, kā arī citi tradicionāli risinājumi.

3. ASTERISK PAMATI

Asterisk – tā ir platforma telefonijai ar atklāto pirmkodu, kas bija izstrādāta izpildei Linux operētājsistēmā. Asterisk ir slavens ar savām konfigurēšanas iespējām un stingru atbilstību standartiem. Neviena cita privātā atzara centrāle nepiedāvā tikpat plašas iespējas dažādu projektu realizācijai un savu jaunievedumu izveidošanai, kā arī integrāciju ar citām biznesa tehnoloģijām. [1] Asterisk piedāvā ļoti daudz standarta iebūvētu funkciju, tādu kā:

- balss pasts;
- mūzikas atskaņošana gaidīšanas laikā;
- konferenču veidošana;
- izsaukumu novietošanā stāvvietā (*call parking*);
- izsaukumu rindošana un aģenti;
- IVR sistēma;
- telefonsarunu ierakstīšanas sistēma;
- lietotāju pārvaldība;

un daudz citu. Tiek piedāvātas arī integrācijas iespējas ar tādām interesantām un revolucionārām sistēmām kā tekstu pārvēršana runā un balss atpazīšana.

Telekomunikāciju vēsturē vēl nekad nebija sistēmas, kura tik perfekti apmierināja jebkuras kategorijas biznesa vajadzības. Asterisk risina problēmas ar sistēmas pārnesamību un ierobežotu funkcionalitāti – tas ir tiešām ļoti pārliecinošs un iespaidīgs projekts, kas ļauj daudz efektīvāk veikt visas standarta ATC funkcijas.

Asterisk izvēles galvenās priekšrocības priekš biznesa ir:

- **Ekonomiskums** – lai izveidotu savu sistēmu, kas apkalpos telefonu izsaukumus, nav nepieciešams veikt kapitālus ieguldījums aparatūrā;

- **Risinājumu ekskluzivitāte** – pateicoties atklātai arhitektūrai, var viegli, ātri un elastīgi veikt integrāciju ar praktiski jebkuru informācijas sistēmu.

Asterisk pielietošanas sfēras var būt sekojošas:

- Privātā atzara centrāle;
- Izsaukumu centru organizācija;
- IVR sistēmu organizēšana;
- Konferenču organizēšana;
- Balss pasta sistēmas;
- Tehniskais un informācijas atbalsta dienests;
- Uzziņu-informāciju sistēmas „Karstā līnija";
- Klientu automatizētas informēšanas sistēmas;
- u.c.

Nākamajās nodaļās tiek sīkāk apskatīti Asterisk piedāvātās iespējas un rīki. Īpaši detalizēti tiek izskatītas tādas Asterisk sastāvdaļas kā dialplāns un AGI, jo tie būs svarīgākie izstrādājamas Infoline sistēmas komponenti.

3.1. Dialplāns

Asterisk sistēmas sirds ir dialplāns, kas nosāka visu ienākošu un izejošu zvanu apstrādes likumus – instrukcijas, kurām sekos Asterisk. Atšķirība no tradicionālām telefonijas sistēmām, Asterisk dialplāns ir ļoti elastīgs un pilnīgi konfigurējams.

Dialplāns atrodas konfigurācijas datnē extensions.conf un sastāv no četriem kopā strādājošiem elementiem:

- **Konteksti (*contexts*)**– papildu numuru grupas ar nosaukumiem**.** Konteksts ir viens no kanālu aprakstošiem parametriem – dialplāna punkts, no kura sākās

savienojumu apstrāde, kas izpildās caur šo kanālu. Tā ir viena no svarīgākajām daļām Asterisk konfigurācijā.

Konteksta galvenās funkcijas ir**:**

- *izolācija:* konteksti ļauj izolēt dažādas dialplāna daļas, neļaujot tām savstarpēji iedarboties – papildu numurs, definēts viena konteksta ietvaros, ir pilnīgi izolēts no cita konteksta papildu numuriem, ja nav noteikts citādi;
- *drošība*: konteksta pareiza lietošana ļauj nodrošināt specifiskas funkcijas tikai izvēlētiem abonentiem, ierobežojot citus – piemēram, aizliegt viņiem tālsatiksmes zvanus.

- **Papildu numuri (*extensions*) –** telekomunikācijās šīs termins nozīmē ciparu identifikatoru, kas ir piešķirts līnijai, kas ved pie konkrēta tālruņa [1], tomēr Asterisk sistēmā tam ir plašāka nozīme – tie nosaka soļu secību, kas izpildīs Asterisk pēc šī līnijas izsaukšanas – tātad, papildu numuri nosaka, kas notiks ar zvanu pēc tā apstrādes atbilstoši dialplānam;
- **Prioritātes (*priorities*) –** numuri, kas nosaka lietotņu izpildes secību, un kas ir definēti konkrētam papildu numuram. Numerācija obligāti sākas ar 1;
- **Lietotnes (*applications*)–** izpilda noteiktu darbību ar kanālu, piemēram, savienošanas pārtraukumu vai melodijas atskaņošanu.

Lai labāk saprastu dialplāna koncepciju (jo tās saprašana ir kritiski svarīga lieta darbam ar Asterisk), izskatīsim vieglu piemēru [1]:

```
[incoming]
exten=>1234,1,Answer()
exten=>1234,n,Playback(labdien-pasaule)
exten=>1234,n,Hangup()
```

Šeit [*incoming*] ir konteksta nosaukums priekš ienākošiem izsaukumiem, *exten* => ir papildu numuru sintakse, kam seko papildu numuru nosaukums vai numurs, un lietojums, kas ir atdalīti ar komatiem. Pieņemsim, ka mums ir nokonfigurēts kanāls, un visi izsaukumi

tiek nosūtīti uz *incoming* kontekstu. Tad, zvanot uz papildu numuru *1234*, Asterisk vispirms atbildēs uz izsaukumu – to liks darīt *Answer*() lietotne, kas atbild kanālam, kas pieņēma izsaukumu; tad Asterisk atskaņos ierakstīto skaņu datni *labdien-pasaule* ar *Playback*() lietotni, un beigās pārtrauks savienojumu ar *Hangup*() lietotni.

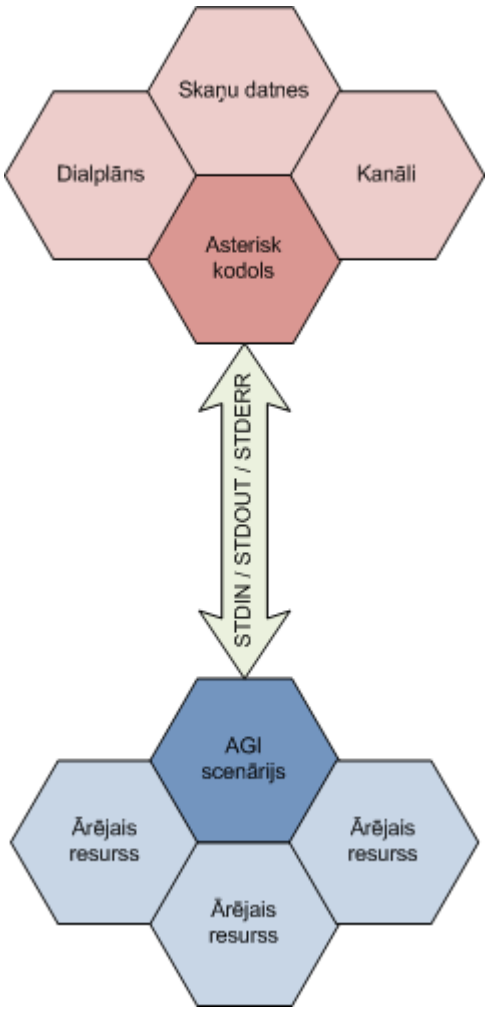
Dialplāns ir ļoti jaudīgs un elastīgs instruments: apskatītais statiskā dialplāna piemērs veica tikai vienu darbību visiem izsaukumiem, bet realitātē tas ir līdzīgs programmēšanas valodai, un tajā ir zarošanas nosacījumi, lietotāja ievades apstrāde, mainīgie, instrumenti, kas līdzīgi regulārām izteiksmēm, operatori, makrokomandas un daudzi citi lietderīgi instrumenti.

3.2. AGI (*Asterisk Gateway Interface*)

AGI – Asterisk vārtejas saskarne – ir sistēma, kas definē standarta saskarni, ar kuras palīdzību ārējas programmas var pārvaldīt Asterisk dialplānu. Visbiežāk AGI scenāriji tiek izmantoti, lai realizētu paplašinātu loģiku, savienotos ar relācijas datu bāzēm un realizētu pieeju citiem ārējiem resursiem. Kontroles nodošana ārējam AGI scenārijam ļauj Asterisk sistēmai realizēt uzdevumus, kuru izpilde būtu neiespējama citā gadījumā. [1]

AGI scenārijus, kas paplašina Asterisk funkcionalitāti, var rakstīt praktiski jebkurā programmēšanas valodā. AGI scenāriju palaišana notiek ar lietotnes AGI() izsaukšanu no dialplāna, kuras arguments ir AGI scenārija nosaukums.

Ir svarīgi pieminēt, ka AGI nepiedāvā programmēšanas API – sadarbība un datu apmaiņa ar Asterisk notiek pa STDIN (standarta ievade), STDOUT (standarta izvade) un STDERR (standarta kļūda) kanāliem: lai iegūtu informāciju no Asterisk, AGI scenārijs lasa STDIN kanālu, lai nosūtītu informāciju Asteriskam, AGI scenārijs raksta datus STDOUT kanālā, un lai iesūtītu atkļūdošanas informāciju Asterisk konsolē, AGI scenārijs raksta datus STDERR. Vizuāli šī komunicēšana ir attēlota Zīm. 1.



Zīm. 1 AGI scenāriju komunicēšana ar Asterisk

Palaižot AGI scenāriju, Asterisk ieraksta STDIN kanālā savus mainīgos ar to vērtībām, un beigās raksta tukšu rindu, signalizējot, ka tas beidza datu pārraidi. Pēc katras AGI scenārija komandas Asterisk sistēmai, tā atgriež atbildi, kura AGI scenārijam ir jānolasa.

Dažas lietderīgas Asterisk AGI komandas, kuras tiek izmantotas izstrādājamā Infoline sistēmā:

- ANSWER – atbild uz izsaukumu, kas ienāk kanālā;
- GET DATA 1 [2] [3] – noskaņo skaņas datni ar nosaukumu *1* un pieņem DTMF secību ar maksimālu garumu *3*, ar gaidīšanas laiku *2*. Atgriež ievadīto vērtību no izsaucošā abonenta;

- WAIT FOR DIGIT 1 – gaida DTMF kodu ievadīšanu laikā *1* (sekundēs). Atriež ciparu ASCII vērtību, ja tā bija ievadīta;
- SET
    - CALLERID 1 – maina tekoša kanāla izsaucoša abonenta identifikatoru uz numuru *1*,
    - CONTEXT 1 – nosaka kontekstu *1*, kurā tiks turpināta izpilde pēc AGI scenārija izpildes beigām,
    - EXTENSION 1– maina papildu numuru *1*, kuram pāries izpilde pēc AGI scenārija izpildes beigām,
    - PRIORITY 1 – maina prioritāti, no kuras turpinās izpilde pēc AGI scenārija izpildes beigām;
- STREAM FILE 1 2 [3] – noskaņo skaņas datni ar nosaukumu *1*, kuru var pārtraukt ar izejošu kodu ievadīšanu, kas ir specificēti parametrā *2*. Ja ir specificēts nobīdes fragments *3*, tad atskaņošana sākas ar to;
- HANGUP [1]– beidz savienošanu pa tekošo vai pa noradīto *1* kanālu.

Dažām programmēšanas valodām ir jau gatavas bibliotēkas un platformas darbam ar Asterisk AGI, kas piedāvā daudzus gatavus risinājumus un efektīvus rīkus. No [9] avota var lejupielādēt palīglīdzekļus Java, Pascal, Perl, PHP, Python, Ruby, C, .NET un pat Haskell valodām.

3.3. AMI *(Asterisk Manager Interface)*

AMI – Asterisk pārvaldes saskarne – ir jaudīga programmas saskarne, kas ļauj ārējām programmām pārvaldīt un kontrolēt Asterisk sistēmu. [1] Var teikt, ka tas ir pretstats AGI, kas ļauj Asterisk sistēmai palaist ārējo programmu.

Viens no populārākajiem AMI uzdevumiem ir integrācija ar esošām sistēmām un lietojumprocesiem. Tas tiek arī pielietots lietotnēs, piemēram, lai automātiski izsauktu numuru, vai izdarīt kādas izmaiņas Asterisk konfigurācijas datnēs.

Savienošanās ar AMI notiek caur tīkla portu, kuru noklausās AMI. Pēc veiksmīgas autentifikācijas klienta programma var sūtīt komandas Asterisk sistēmai, un saņemt atbildes kopā ar informāciju par sistēmas statusu patstāvīgu atjaunošanu.

Datu apmaiņa notiek ar paketēm, kuras satur informāciju formā „atslēga: vērtība”.

3.4. Citas Asterisk iespējas

Lai nodemonstrētu, cik patiešām ir jaudīgs Asterisk, šajā nodaļā ir apskatīti dažādas gan standarta, gan neparastas iespējas, kuras varētu būt lietderīgas gan izsaukuma centra organizēšanai, gan privātā atzara centrāles izveidei.

3.4.1. Asterisk IVR (*Interactive voice response*)

IVR – Astersik interaktīva balss reaģēšana – ir tehnoloģija, kura ļauj datoram uztvert balss un DTMF tastatūras ievadu. IVR sistēmas ļauj klientiem piekļūt kompānijas datubāzei, izmantojot telefona tastatūru vai balss atpazīšanu, ar kuras palīdzību viņi var noskaidrot interesējošu informāciju, ja sekos instrukcijām. IVR sistēmas sniedz informāciju ar iepriekš ierakstīto vai dinamiski ģenerēto audio uzziņu palīdzību. [8]

IVR sistēmas tiek bieži izmantotas izsaukšanas centros, jo tie ļauj veikt lielu klientu numuru maršrutēšanu un efektīvi noslogot operatorus. Zvanot tādā centrā, izsaucošais abonents noklausās sākotnēju balss uzziņu, kas piedāvās viņam izvēli ar visiem pieejamiem pakalpojumiem un to atbilstošiem numuriem.

IVR sistēmas var tikt realizētas Asterisk dialplāna scenāriju rakstīšanas valodā ar audio ierakstu atskaņošanas un lietotāja ievades saņemšanas komandām.

Asterisk piedāvā arī Video IVR iespēju, kas ļauj izmantot lietotnes, spēles, video tērzēšanu, skatīties video, audio un grafiskus attēlus savā telefona caur 3G video zvanu, neko nepielādējot un neuzstādot – visa lietojumprogrammas loģika atrodas serverī.

3.4.2. Balss pasts

Balss pasts ir viena no plašāk izmantotām funkcijām telefonijā. Tas ir ļoti elastīgi un plaši konfigurējams Asterisk ietvaros, bet šeit apskatīsim tikai nelielu daudzumu no visām iespējām [1]:

- Neierobežotas ar izmēru un apsargātas ar paroli balss pasta kastes, kuras satur pastu mapes, lai varētu ērti organizēt balss pasta ziņojumus;
- Standarti un speciāli sveicinājumi;
- Dažādi sveicinājumi atkarībā no stāvokļiem „aizņemts" un „nav pieejams";
- Iespēja saistīt vairākus telefonus ar vienu balss pasta kasti, un vairākas balss pasta kastes ar vienu telefonu.
- Paziņojumi par jaunu balss ziņojumu pa e-pastu, ar iespēju pievienot e-pastam balss ziņojumu kā skaņas datni;
- Balss pasta pārsūtīšana un apraides sūtīšana;
- Jauna balss pasta indikators daudziem telefonu tipiem (gaismas indikators vai tonālais signāls).

3.4.3. Karsti galdi (*Hot desks*)

Karstu galdu uzdevums ir ekonomiski organizēt darba vietas uzņēmumā, kur darbinieki strādā nenormētu laiku un visu laiku mainās vietām. Šī sistēma ļauj darbiniekiem ielogoties savā kontā no jebkura telefona, piekļūt savam balss pastam, utt.

3.4.4. Zvanu novietošana stāvvietā (*Call parking*)

Ar šo funkciju palīdzību var pārvest ienākošu izsaukumu gaidīšanas stāvoklī, lai tas varētu tikt pieņemts citā papildu numurā – tas ir efektīva metode nodrošināt iespēju pārslēgt izsaucošos abonentus starp sistēmas lietotājiem.

3.4.5. Konferenču organizēšana

Audio konferenču organizēšana notiek ar lietotnes *MeetMe*() palīdzību, kas nodrošina iespēju komunicēt vairākiem abonentiem vienlaikus. Galvenās *MeetMe*() funkcijas ir [1]:

- Iespēja izveidot konferenci, kas ir pasargāta ar paroli;
- Konferenču administrēšana – iespēja izslēgt skaņu, konferences bloķēšana, dalībnieku izslēgšana no konferences;
- Statisku vai dinamisku konferenču izveide.

3.4.6. Mūzikas atskaņošana gaidīšanas laikā

Protams, ka jebkura PBX piedāvā šādu iespēju. Bet Asterisk ir viena no nedaudziem, kas ļauj atskaņot MP3 formāta datnes. Pie tam tradicionālajās ATC gaidīšanas mūzika nāk no viena un tā paša avota un spēlē vienmēr, pat ja abonents nav gaidīšanas režīmā; tajā pat laikā Asterisk ļauj atskaņot katram abonentam savu melodiju.

3.4.7. Teksts - runā (*Festival*)

Festival – ir populārs mehānisms ar atklāto pirmkodu, kas ļauj atskaņot tekstu runas formā. Galvenā *Festival* izmantošanas ideja ir tāda, ka dialplāns var padot tekstu Festivālam, kas „lasīs" to izsaucošam abonentam. [1].

4. PROBLĒMAS RISINĀJUMS

4.1. Projektēšana

Jebkuras lielas sistēmas projektējumā viena no svarīgākajām daļām ir sistēmas prasmīga sadalīšana moduļos, katram no kuriem ir sava funkcionalitāte. Šai moduļu pieejai ir daudz priekšrocību:

- tā ļauj efektīvāk strādāt komandā, jo katrs ir atbildīgs tikai par savu moduli;
- tā atvieglo projektu testēšanu, jo katru moduli var pārbaudīt atsevišķi un kļūdas atrašanās vieta ir viegli atrodama;
- tā ļauj kontrolēt projekta sarežģītību un abstrahēties no zemāka līmeņa grūtībām [2];
- tā ļauj vieglāk ieviest jaunas modifikācijas, bez nepieciešamības pārbūvēt visu sistēmu;
- tā ļauj sadalīt projekta resursus un atkārtoti tos izmantot citos projektos.

Uzdevums bija nokonstruēt tādu sistēmu, uz kuras bāzes pēc tam varētu ātri uzbūvēt līdzīgas informāciju-uzziņu sistēmas, un viegli uzturēt un ieviest jaunu funkcionalitāti jau esošā risinājuma. Tāpēc tika nolemts atteikties no Asterisk piedāvātā IVR risinājuma, kā arī integrācijas ar datu bāzes, un tika nolemts izveidot savus universālus apmetņus (*wrappers*) gan Asterisk funkcionalitātei, gan Oracle datubāzes funkcionalitātei, kas tiks izmantoti galvenajā AGI skriptā – tas padarītu sistēmas analīzi, modificēšanu, testēšanu un atkļūdošanu daudz vieglāku, jo programmatūras loģika būtu vieglāk uztverama. Šī pieeja lāva atrisināt vienu ļoti svarīgu uzdevumu – tā ļāva iztikt bez nepieciešamības visu laiku turēt galvā visus projekta līmeņus un deva iespēju koncentrēties tikai uz vienu pašlaik izstrādājamo daļu, kas, autoraprāt, bija viens no galvenajiem projekta veiksmes kritērijiem.

Protams, varētu iztikt arī ar iekšējiem Asterisk risinājumiem – ja tas būtu vienreizējs un vienkāršāks uzdevums un laiks būtu stipri ierobežots, tad būtu vērts izmantot Asterisk esošos dialplānu rīkus. Bet orientācija bija tieši uz kvalitāti un skaistu, atkārtoti izmantojamu un viegli modificējamu risinājumu. Funkcionalitāte bija diezgan sarežģīta, un

dialplānu pieeja varēja rast daudz tehnisko grūtību un apgrūtināt sarežģītības kontrolēšanu [2].

Par nepieciešamām projekta raksturiezīmēm, kas liktas tā pamatā, tika uzskatīti [2]:

- **Minimāla sarežģītība** – sarežģītību var samazināt, sadalot sistēmu apakšsistēmās. Projektam jābūt vieglam un saprotamam. Veiksmīgs projekts ir tāds, kurā var strādāt ar programmas fragmentu, droši ignorējot lielāko citu fragmentu daļu.

- **Uzturēšanas vieglums** – projektējot sistēmu, ir jādomā par programmētājiem, kas uzturēs šo produktu – jādomā par viņiem kā par savu auditoriju un jāprojektē sistēma tā, lai tās darbība būtu uzskatama.

- **Paplašināmība** – šī īpašība ļauj modificēt un uzlabot sistēmu, neizjaucot tās galveno struktūru. Viena fragmenta mainīšana nedrīkst ietekmēt pārējos fragmentus. Un visvairāk iespējamu izmaiņu ieviešanai jāpieprasa vismazākās pūles.

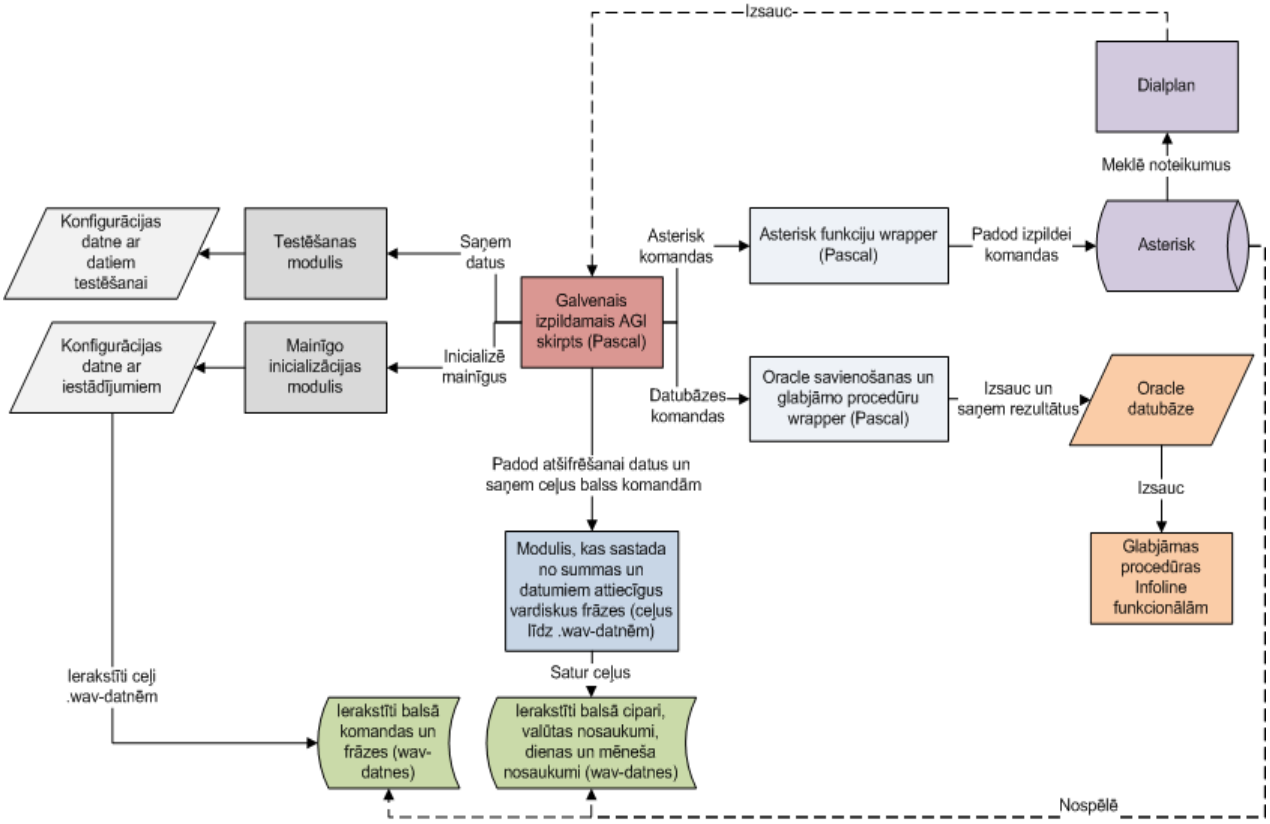
- **Vāja saistība** (*loose coupling*) – saistību daudzumam starp programmas moduļiem jābūt pēc iespējas mazākām, kas atvieglos integrāciju, testēšanu un uzturēšanu. Tādai projektēšanai var izmantot abstrakciju, iekapsulēšanu un informācijas slēpšanu.

- **Atkārtotas izmantošanas iespēja** – jāprojektē sistēma tā, lai tās fragmentus varētu viegli izmantot arī citās sistēmās.

- **Minimāla, bet pilna funkcionalitāte** – lieku daļu trūkums sistēmā. Jābūt visam, kas tiek pieprasīts, un ne vairāk.

- **Stratifikācija** – dekompozīcijas līmeņu sadalīšana, kas ļauj izpētīt sistēmu jebkurā atsevišķā līmenī un iegūt noteiktu priekšstatu par to. Jāprojektē sistēma tā, lai to varētu izstudēt atsevišķos līmeņos, ignorējot citus līmeņus.

Balstoties uz šiem noteikumiem, uzziņu-informatīvas sistēmas Infoline risinājums tika projektēts, kā ir redzams Zīm. 2. Tika plaši izmantota abstrakcijas koncepcija: sistēma sastāv no vairākiem moduļiem, katrs no kuriem nodrošina savu funkcionalitāti.



Zīm. 2 Infoline sistēmas projektējums

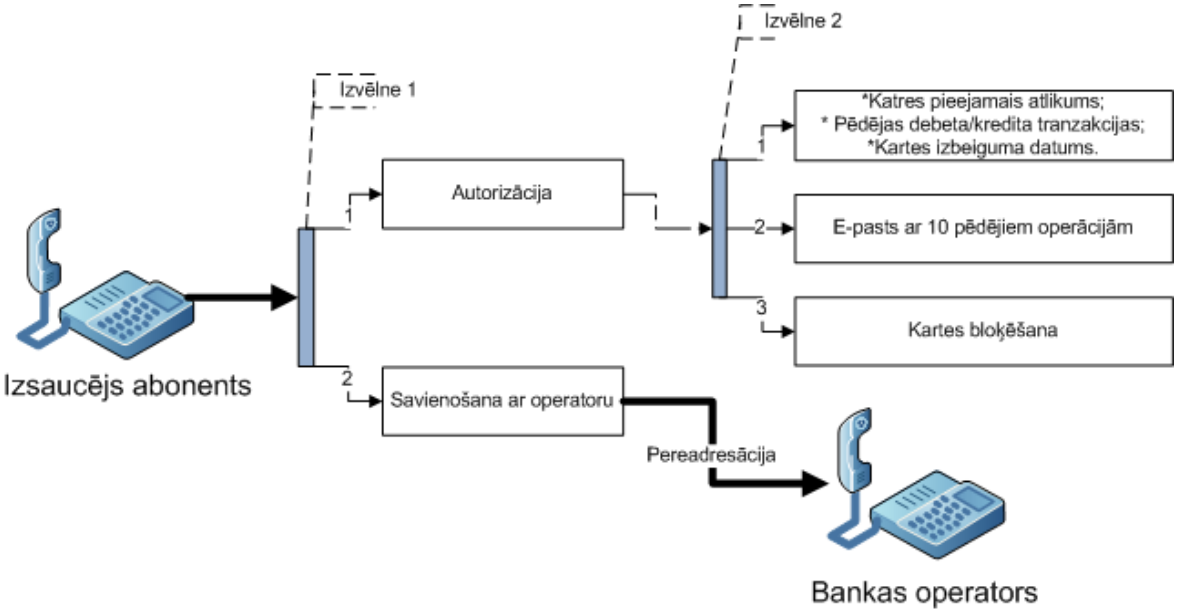
Galvenās sistēmas sastāvdaļas ir:

- Asterisk apmetnis, kas pārvaldīja Asterisk dialplānu un nodrošināja ērtu saskarni tā pamatfunkcijām;
- Oracle funkciju apmetnis, kas komunicēja ar Oracle datubāzi caur ODBC un nodrošināja saskarni tajā glabāto procedūru izsaukšanai;
- Balss modulis, kas pārvērta naudas summas un datumus attiecīgajos balss ziņojumos;
- Testēšanas modulis, kas salīdzināja datubāzes izsaukumu rezultātus ar rezultātiem, kas nolasīti no konfigurācijas datnes testpiemēriem;

- Inicializācijas modulis, kas inicializēja visus mainīgos;
- Galvenais izpildāmais scenārijs, kas, izmantojot visus pārējos moduļus, realizēja galveno funkcionalitāti.

Sistēmas sirds ir Pascal valodā rakstīts AGI scenārijs, kurš tika izsaukts no Asterisk, kad ienākošais zvans notiek uz dialplānā noteikto Infoline numuru. Izmantojot citus sistēmas moduļus, skripts realizē visu galveno funkcionalitāti – mainīgo inicializāciju, savienošanos ar Oracle datubāzi un tās procedūru izsaukšana, Asterisk iespēju izmantošana, atskaņojot izsaucošam abonentam ierakstīto balss informāciju un saņemot no viņa ievadu kā nospiestus uz telefona ciparus, un naudas summu un datumu pārvēršana attiecīgos balss ziņojumos. Katrs no moduļiem ir sīkāk aprakstīts nākamajā nodaļā.

Nākošais solis bija noplānot izvēlnes, lai tās būtu loģiskā secībā un viegli pieejamas lietotājam. Rezultātu var redzēt Zīm. 3.



Zīm. 3 Izsaucošā abonenta izvēlne

Ņemot vērā klientam nepieciešamo funkcionalitāti, tika izstrādāta tāda izvēlne, kur pirmajā līmenī var vai nu savienoties ar operatoru, vai nu autorizēties un pāriet uz pamatizvēlni, kurā ir pieejama visa informācija par karti, e-pasta saņemšana par operācijām un kartes bloķēšana, ja tā ir aktīva.

No manas automatizētas izziņas sistēmas izvēlņu izstrādāšanas pieredzes ir ļoti svarīgi ievērot sekojošas lietas, kuri bija veiksmīgi realizēti Infoline sistēmā:

- izvēlnes pārejām jābūt loģiskām un secīgām;
- lietotājam no paša pirmā izvēlnes līmeņa jāsaka, kādas operācijas vai operāciju kategorijas ir paslēptas zem katras opcijas, it sevišķi kad ir daudz izvēlnes līmeņu;
- obligāti jāļauj izsaucošam abonentam savienoties ar operatoru kļūdu vai neskaidrības gadījumā;
- visām balss izvēlnēm jābūt pēc iespējas īsākām, skaidrākām un konkrētākām, lai nepārslogotu lietotāju ar lieku informāciju un tajā pašā laikā dotu viņam visu nepieciešamo informāciju;
- pēc iespējas samazināt opciju daudzumu, apvienojot vienā vietā visu, ko var, lai nesajauktu lietotāju;
- zeltais IVR sistēmu būvēšanas likums saka, ka izvēlnei nedrīkst būt vairāk par piecām opcijām, kā arī nedrīkst būt vairāk par diviem izvēlnes līmeņiem;
- labai sistēmai jārūpējas par pastāvīgiem lietotājiem – tiem jāļauj ievadīt izvēlnes opcijas vai datus pēc iespējas ātrāk, neklausoties līdz balss ziņojuma beigām;
- paredzēt atgriešanās iespēju, lai lietotājs vienmēr varētu tikt iepriekšējā izvēlnē (ir pieņemts atgriešanai izmantot ‘*’ simbolu);
- paredzēt atkārtošanas iespēju gadījumam, ja lietotājs nepareizi ievadīja kādus datus (visbiežāk atkārtošanai izmanto ‘*’ simbolu);
- paredzēt speciālu simbolu, kas nozīmē datu ievadīšanas beigas (ir pieņemts apstiprināšanai izmantot ‘#’ simbolu);

- paredzēt pieļautu atkārtošanas skaitu kā priekš datu ievadīšanai, tā arī priekš opcijas izvēlei;
- paredzēt lietotāja ievades gaidīšanas laiku – noildzi, pēc kuras notiek vai nu atgriešanās izvēlnē, vai nu zvana pabeigšana.

4.2. Realizācija

Šajā nodaļā ir aprakstīti visi moduļi, kuri tika īsi apskatīti projektēšanas nodaļā. Datubāzes daļa ir apskatīta tikai shematiski, jo darba autors nepiedalījās tās izveidē.

4.2.1. Dialplāns

Kā jau bija minēts teorētiskajā daļā, dialplāns nosaka, kā Asterisk apstrādās visus ienākošus un izejošus izsaukumus. Lai izmantotu AGI scenāriju dialplanā, jāizsauc lietotne AGI() ar argumentu, kas ir norāde uz AGI scenārija vārdu. Galīga produkcijas versija izskatās šādi:

```
[incoming-moscow]
exten => 2986,1,AGI(/opt/infoline/infoline.sh)
```

Kur „2986” ir Infoline uzziņu-informācijas sistēmas telefona papildu numurs, „1” ir instrukcijas numurs jeb prioritāte, un „AGI(/opt/infoline/infoline.sh)” ir skripta izsaukšana – darbība, kas izpildās pie izsaukuma.

4.2.2. AGI izpildāmais scenārijs

Galvenā scenārija izsaukšana notiek no dialplānā definēta scenārija infoline.sh. To nevarēja izsaukt tieši, jo pirms tam bija jāeksportē vairāki mainīgie komandu interpretatora vidē, lai tie varētu tikt izmantoti kā statiski mainīgie vēlāk izpildāmajā galvenajā Infoline scenārijā.

Infoline.sh izskats:

```
#!/bin/bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/oracle:/opt/infoline
export TNS_ADMIN=/opt/infoline
export INFOLINE_CONF=/opt/infoline/infoline.ini
export INFOLINE_TEST=/opt/infoline/infoline.test.ini
/opt/infoline/infolineMain.exe
```

Pēdējā rindā tiek izsaukts Infoline galvenais scenārijs, kura pirmkods ir iekļauts 8.1 pielikumā. Tas ir Infoline sirds – tas iekļauj visu sistēmas loģiku un izsauc pārējus

projektēšanas laikā atdalītus moduļus. Šī pieeja ļāva padarīt sistēmu viegli lasāmu, saprotamu un modificējamu.

AGI galvenais scenārijs iekļāva sevī visu augsta līmeņa loģiku un realizēja visu pieprasītu funkcionālu.

4.2.3. Asterisk apmetne (*wrapper*)

Šī moduļa ietvaros tika realizēta klase, kas pārvalda Asterisk dialplānu caur standarta ievadu-izvadu kanāliem, un kas piedāvā viegli saprotamu, modificējamu un ērtu saskarni.

Asterisk klases funkcijas ir sekojošas:

- Inicializācija – visu Asterisk padoto mainīgo nolasīšana no standarta ievades kanāla un to saglabāšana;
- Atbildēšana uz zvanu – atbild uz izsaukumu, kas ienāk kanālā;
- Izvēlnes realizācija – atskaņo balss ziņojumu ar visām pieejamām opcijām un gaida lietotāja ievadi noteiktu laiku, pēc kura atkārto balss ziņojumu. Atgriež lietotāja izvēlēto opciju;
- Lietotāja ievadītu datu ar tonālu ievadi nolasīšana – atskaņo balss ziņojumu ar instrukcijām, ko un kā ievadīt, un gaida lietotāja ievadi, kura beidzas ar apstiprinošu simbolu, noteiktu laiku, pēc kura vēlreiz atkārto instrukcijas. Atgriež lietotāja ievadītos datus;
- Zvanu iniciēšana – uzstādina tekoša kanāla izsaucošā abonenta identifikatoru, kontekstu, papildu numuru un maina prioritāti, tādējādi iniciējot zvanu no izsaucoša abonenta abonentam uz papildu numura, kas atrodas iedotā kontekstā;
- Skaņu datnes atskaņošana – atskaņo skaņas datni vai skaņas datņu sarakstu;
- Izsaukuma pārtraukšana – beidz savienošanu, pēc izvēles pirms tam atskaņojot ziņojumu.

Lai neapgrūtinātu galvenā scenārija uztveri, tika nolemts neizmantot pa tiešo šo klasi un uzrakstīt tam kārtēju apmetni, kas tiek realizēts tādu pašu funkcionālu procedūru veidā, kas pieņems rādītāju uz Asterisk klasi.

4.2.4. Datubāzes daļa

Saskarni ar Oracle datubāzi un tajā glabātām procedūrām nodrošina Oracle apmetņa modulis, kas komunicē ar datubāzi caur ODBC. Modulis iekļauj visu vajadzīgo funkcionalitāti:

- savienošanu un atvienošanu no datubāzes;
- lietotāja pieteikšanos un atteikšanos;
- informācijas iegūšanu par lietotāja kontu;
- e-pasta sūtīšanu par pēdējām operācijām;
- pēdējo kļūdu apraksta saņemšanu.

4.2.5. Datumu un summu būvēšanas modulis

Modulis, kas no datumiem un naudas summām atgriež sarakstu ar ceļiem līdz attiecīgajām audio datnēm, spēlējot kuras pēc kārtas, tiks sastādīta to balss interpretācija.

Ir svarīgi pieminēt, kā vairākumam autoram pazīstamu automātisko uzziņu sistēmu šī ir vāja vieta – datumi un summas vārdiem skan ļoti nedabiski, jo nav ievērotas pareizas pauzes, balss intonācijas augstums dažādās frāzes vietās, kā arī lietvārdu dzimtes, skaitļi un locīšanas likumi.

4.2.5.1. Datumi

Vispirms studijā tika ierakstīti visi iespējami dienu numuri, mēnešu nosaukumi un gadi lokatīvā, no kuriem varēja uzbūvēt, piemēram, tādu frāzi kā „trīsdesmit pirmajā maijā divi tūkstoši desmitajā gadā” ar dabiskām pauzēm starp vārdiem, kuras tika ņemtas vērā skaņas apstrādes procesā. Šeit jāpiebilst, ka slikta balss ierakstīšanas kvalitāte var sabojāt pat labāku IVR sistēmu, tāpēc profesionālai balss ierakstīšanai bija veltīts daudz laika un resursu. Audio ieraksti tika nosaukti pēc viena likuma un rūpīgi sakārtoti.

Pēc tam atliek tikai dešifrēt dienas, mēnešus un gadus no padotā datuma un, zinot ceļus līdz audio ierakstiem, atbilstoši salikt tos kopā.

4.2.5.2. Summas vārdiem

Ar summām gāja ne tik viegli, jo bija jāņem vērā visus sarežģītus gramatikas likumus un intonācijas. Katram vārdam bija ierakstīti vairāki varianti intonāciju: ar paaugstināšanu, ar pazemināšanu un tādiem vārdiem kā „simts", „tūkstotis" utt. arī ar taisno. Tie tika mainītas atkarībā no vārda atrašanas vietas – sākumā intonācijai jābūt augošai, vidū – taisnai, un beigās – pazeminātai.

4.2.6. Inicializācijas modulis

Lai nepārslogotu galveno Infoline scenāriju, tika nolemts izdalīt mainīgo inicializāciju atsevišķā modulī. Tas ielasa iestatījumus no konfigurācijas datnes, kas satur ceļus uz visām izvēlņu balss instrukcijām un ierakstītām frāzēm un pauzēm, no kurām tiek būvēti teikumi ar lietotājam vajadzīgo informāciju, kā arī datubāzes savienošanas parametrus, noildžu laikus, mēģinājumu skaitu izvēlnēm un lietotāju datu ievadam, u.c. Visa tā informācija tika uzglabāta globālajos mainīgos.

4.2.7. Testēšanas modulis

Infoline scenārijs var strādāt arī testēšanas režīmā, kurā visiem datubāzes procedūru izsaukumiem tiks atgriezts rezultāts no izvēlētā testpiemēra. Visi testpiemēri ir aprakstīti testēšanas konfigurācijas datnē. Tā ir veiksmīga pieeja, jo tā ļāva notestēt programmatūru, abstrahējoties no datubāzes funkcionalitātes un tās procedūru atgrieztajām vērtībām: piemēram, bija svarīgi notestēt balss instrukcijas priekš dažādiem datumiem un dažādām naudas summām dažādās valūtās – katru reizi mainīt datus lietotāju tabulās būtu ilgi un neefektīvi, kā arī tas apgrūtinātu pārbaudes uz nepareizo rezultātu atgriešanu vai kļūdu rašanos procedūrā, bet šī pieeja to atviegloja, jo ļāva visu to izdarīt, vienkārši pievienojot testpiemērus.

Vienu piemēru var apskatīt zemāk:

```
[Test Cases]
;test case list - executing the first enabled one
Case1    = 1
Case2    = 0
Case3    = 0
```

```
;--------------------------
[Case1 Login]
Card    = 111111
Doc     = 111111
Error   = 0
Result  = 0

[Case1 Logout]
Error    = 0
Result   = 0

[Case1 AccInfo]
CardStatus    = 1
CardExpDate   = 12.12.2009
AccAmt        = 99999999.12
AccCcy        = 810
;
LastDRDate    = 11.11.2007
LastDRAmt     = 11.11
LastDRTrAmt   = 111.11
LastDRTrCcy   = 810
;
LastCRDate    = 10.10.2007
LastCRAmt     = 10.10
LastCRTrAmt   = 100.10
LastCRTrCcy   = 810
;
Error        = 0
Result       = 0

[Case1 SetStatus]
NewStatus    = 0
Error        = 0
Result       = 0

[Case1 EMail]
Error        = 0
Result       = 0
```

Kā redzams, katrai iespējamai lietotāja darbībai ir definēti rezultāti, reakcija uz kuriem tika rūpīgi pārbaudīta un izpētīta. Pēc tām tika ieviestas rezultātu pārbaudes, atkarībā no kuru rezultāta tika vai nu padota iegūtā informācija, vai nu paziņots par mēģinājuma neveiksmīgumu un vai nu pārslēdza zvanu uz operatoru, vai nu paziņoja par neiespējamību iegūt vajadzīgo informāciju un atgriezās iepriekšējā izvēlnē.

5. NĀKOTNES PLĀNI

Es biju ļoti pārsteigta ar ļoti plašām Asterisk pielietošanas iespējām un ar Asterisk ierobežotu pielietošanu reālajā dzīvē. Balstoties uz to, man radās daudz biznesa ideju, kuras varētu veiksmīgi realizēt un pārdot. Asterisk ir brīnišķīga un revolucionāra tehnoloģija ar neierobežotām iespējām biznesa uzlabošanai un automatizācijai, un kamēr ir ļoti maz profesionālu, kuri pārzina abas tās puses – telefoniju un datorzinātnes - šo gandrīz tukšu nišu varētu veiksmīgi aizņemt un realizēt visdažādākus telefonijas projektus: no tādiem pieprasītiem kā izsaukumu centra organizēšana, līdz tādiem ekskluzīviem kā automātiskā sistēma ar balss atpazīšanu, kura sniedz aktuālu informāciju par noteiktu tematiku, piemēram, lidostas vai sabiedriskā transporta sarakstu, fotoradaru atrašanas vietas, noteiktu preču pieejamību veikalos, automātiskā atrašanas-pazušanas dienesta izveidošana, informācija par pakalpojumiem, par precīzu laiku, biļešu pasūtīšana, viesnīcu rezervēšana un daudz citu.

Es noteikti turpināšu interesēties par Asterisk platformu un mēģināt atrast ar to saistītus projektus, kā arī domāt par iespēju veidot interesantus gatavus risinājumus, kas varētu ieinteresēt potenciālus pircējus.

Runājot par realizēto Infoline sistēmu, tai ir ļoti daudz attīstīšanas plānu.

Pirmkārt, tagad es strādāju pie jaunas funkcionalitātes ieviešanas, kas ļaus caur Infoline sistēmu veikt maksājumus pa telefonu. Būs pieejami divu veidu maksājumi:

- no internetbanka standarta veidnēm;
- no lietotāja izveidotajām veidnēm.

Standarta veidnēm būs ierakstīti pilni nosaukumi, kuri tiks atskaņoti lietotāja izvēlnē. Ar lietotāju veidnēm vēl nav saprotams, kā ērtāk piedāvāt iespēju izvēlēties kādu no tām – ar numuriem pēc kārtas, vai sintezēt to nosaukumus balsī. Vēl nav atrisināta problēma ar veidņu parametru ievadīšanu – ideālā gadījumā lietotājam jāļauj ievadīt tikai summu, bet dažreiz ar to nepietiek, jo ir jāievada papildus rekvizītus.

Otrkārt, šī sistēma ieinteresēja arī mūsu Latvijas Norvik banku, un tagad iet runā par to adoptēšanu un ieviešanu arī Latvijā. Bet tā kā tajā nekad netika lietots Asterisk un pietrūkst piemērotu speciālistu, tas vēl ir pārrunu stadijā.

6. REZULTĀTI UN SECINĀJUMI

Darba rezultātā tika izpētīta Asterisk telefonijas platforma un daudzas no tās piedāvātām iespējam. Šīs zināšanas palīdzēja veikt banka automātiskas informācijas-uzziņu sistēmas Infoline projektēšanu un izstrādi.

Tas bija pirmais darba autora projekts šajā sfērā, tāpēc tā realizācija aizņēma vairāk par trijiem mēnešiem. Ļoti daudz laika tika veltīts Asterisk koncepciju pētīšanai, it sevišķi dialplānam un AGI. Ir svarīgi ar tāda tipa jauniem uzdevumiem neķerties klāt tūlītējai realizācijai, bet rūpīgi visu izpētīt un vispirms pamēģināt kaut ko vienkāršu, lai labāk saprastu koncepcijas un pārliecināties, kā viss strādā tā, kā tu gaidi no sistēmas. Pirms projekta uzsākšanas tika uzrakstīts vairums dialplānu un AGI scenāriju – un var secināt, ka šīs pieejas gaitā iegūtā pieredze deva plašāku Asterisk izpratni un pārliecinātību par risinājuma veiksmīgumu. Rezultātā izveidotā Infoline sistēma strādā Krievijas Norvik bankās vairāk kā gadu, un, pateicoties rūpīgai testēšanai, līdz šīm brīdim ar to nav bijis nekādu problēmu.

No Infoline sistēmas risinājuma var secināt, ka visas projektēšanai izvirzītas prasības tika veiksmīgi izpildītas. Sistēma ir ļoti pārdomāti un rūpīgi sadalīta apakšsistēmas, kas atbilst minimālas sarežģītības prasībai. Šo sistēmu ir viegli uzturēt un paplašināt, ieviešot nepieciešamas izmaiņas nepieciešamos moduļos un papildinot galvenā scenārija loģiku. Saistība starp moduļiem ir minimāla – galvenais scenārijs izmanto visus pārējus moduļus, kuri nav saistīti savā starpā. Priekš atkārtotas izmantošanas tāda tipa sistēmas izveidošanai ir perfekti noderīga Asterisk apmetne, kas realizē lielāko daļu no Asterisk funkcionalitātes un kurai var viegli pievienot trūkstošo funkcionalitāti; kā arī modulis, kas pārvērš summas un datumus vārdos. Sistēmai nav nekādas liekas funkcionalitātes. Stratifikācijas noteikums arī izpildās, jo sistēmas dekompozīcija ļauj izpētīt katru no līmeņiem atsevišķi.

Asterisk pētīšanas gaitā darba autors secināja, ka neskatoties uz tā plašām iespējam, kurām bieži nav analogu standarta PBX sistēmās, un ekspluatācijas ekonomiskumu, Asterisk tiek ļoti reti izmantots, kas, visticamāk, ir dēļ konfigurācijas sarežģītības, jo tas apvieno divas ļoti dažādas sfēras: telefoniju un datoriku, un to prasmīga izmantošana savos projektos pieprasa daudz zināšanu par abām sfērām. Secinājumā gribētos teikt, ka es esmu

ļoti pateicīga šim projektam, kas deva man iespēju iepazīties ar Asterisk un ļāva iegūt daudz pieredzes uz tās bāzēto sistēmu projektēšanā un realizācijā. Telefonijas sistēmu izveidošana ir ļoti aktuāls un pieprasīts uzdevums, un es noteikti saistīšu savu nākotnes karjeru ar šo uzdevumu.

7. IZMANTOTĀ LITERATŪRA

1. **Madsen L., Smith J., Van Meggelen J.,**, *Asterisk: The Future of Telephony, Second Edition*, New York: O'Reilly Media, 2007, 608 p.

2. **McConnel S**., *Code Complete*, New York: Microsoft Press, 2004, 960 p.

3. **Ļimoņina J.,** Mobilais serviss: banku sistēma īsziņu sūtīšanai: kvalifikācijas darbs, LU Datorikas fakultāte, Rīga: Latvijas Universitāte, 2008, 53 lpp.

4. Asterisk jaunumi un konsultācijas [atsauce 31.05.2010]. Pieejams: http://www.asteriskguru.com/

5. *Asterisk as IVR server* [atsauce 31.05.2010]. Pieejams: http://docs.digium.com/misc/white-paper-ivr-a20.pdf

6. *Asterisk Reference Information* [atsauce 31.05.2010]. Pieejams: http://www.asterisk.org/docs

7. *Interactive Voice Response, Asterisk IVR Designing, Asterisk IVR Configuration Asterisk Services* [atsauce 31.05.2010]. Pieejams: http://www.asteriskservice.com/asterisk/services/asterisk-ivr-designing.html

8. *IVR*, Wikipedia – The Free Encyclopedia [atsauce 31.05.2010]. Pieejams: http://en.wikipedia.org/wiki/Ivr

9. *Reference guide to all things VOIP* [atsauce 31.05.2010]. Pieejams:http://www.voip-info.org/

10. *Top 10 Open Source PBX Software*, VOIP Today, [atsauce 31.05.2010]. Pieejams: http://www.voiptoday.org/index.php?option=com_content&view=article&id=414:top-10-open-source-pbx-software&catid=53:general&Itemid=101

11. *История IP-телефонии*, [atsauce 31.05.2010]. Pieejams: http://iptelephony.report.ru/material.asp?MID=1780

8. PIELIKUMI

8.1. Pielikums. Galvenā izpildāmā AGI scenārija pirmkods

```
{$MODE DELPHI}

program infoline;

{$DEFINE TEST_CASE}

uses SysUtils, Classes, db, infolineInit;

{$INCLUDE infoline_agi.inc}
{$INCLUDE infoline_dict.inc}
{$INCLUDE infoline_db.inc}

const
    RES_FAILED         = -1;
    WRONG_ACC_CURRENCY     = 9999;
    c64   = 64;
    c2k   = 2048;     //buffer sizes

type
    char64 = packed array [0..pred(c64)] of Char;     //small arrays
(variables)
    char2k = packed array [0..pred(c2k)] of Char;     //large arrays
(dates, amounts)

var
    asterisk,
    session:     Pointer;
    uniqueID,
    callerID:     char64;

//============================================================================
//--------------------------Logging-------------------------------------
type
TReason = (rsInfo, rsWarning, rsError);

procedure WriteToLog(funcName: AnsiString; reason: TReason;
            aMessage : AnsiString = ''); overload;
const cMsg : array[TReason] of AnsiString = ('INFO', 'WARNING', 'ERROR');
var outputFile:     Text;
    hasError:     LongBool;
    errCode:     LongInt;
    errMsg:         char2k;
    mainMsg:     AnsiString;
begin
    //writing only information suitable for current log level
    case reason of
    rsInfo:
        if (LOG_LEVEL < 7) then Exit;
    rsWarning:
        if (LOG_LEVEL < 3) then Exit;
```

```
    rsError:
        if (LOG_LEVEL < 1) then Exit;
    end;

    //msg time, reason and caller module
    mainMsg := Format('[%s]: %s: "%s"',
    [FormatDateTime('dd/mm/yy hh:mm:ss', Now), cMsg[reason], funcName]);

    //getting oracle error, if there's any
    case reason of
        rsError:
        begin
        DB_Ora_ErrorDesc(session, hasError, errCode, @errMsg, c2k);
        if (hasError) then
        mainMsg :=  Format('%s'#10#9'OraError: %d'#10#9'Message: %s',
            [mainMsg, errCode, errMsg]);
    end;
    end;

    //sender's log  message
    if (aMessage <> '') then mainMsg := mainMsg + #10#9 + aMessage;
    mainMsg := mainMsg + #10;

    //change log path according to unique asterisk id variable
    AssignFile(outputFile, Format('%s/%s.log', [PATH_TO_LOG_FILE,
uniqueID]));
    while (true) do
    begin
    try
        Append(outputFile);
        break;
    except
    end;
    try
        Rewrite(outputFile);
        break;
    except
        Exit;
    end;
    end;

    try
    //writing result message to file
    WriteLn(outputFile, mainMsg);
    except
    end;
    Close(outputFile);
end;

procedure WriteToLog(funcName: AnsiString; reason: TReason;
            aFormat: AnsiString; aArgs: array of const); overload;
begin
    WriteToLog(funcName, reason, Format(aFormat, aArgs));
end;

//=============================================================================
//------------------------------Helpers----------------------------------
```

```
procedure PlaySound(sound: AnsiString); forward;

procedure FInit(ret: LongInt; sound: AnsiString = '');
begin
    //destroying session with db
    if Assigned(session) then
    begin
    if (ret <> 0) then DB_Logout(session, ret);
    DB_Disconnect(session);
    DB_Release(session);
    end;

    //destroying connection with asterisk
    if Assigned(asterisk) then
    begin
    if (ret <> 0) then Agi_Hangup(asterisk)
    else if (sound > '') then PlaySound(sound);
    Agi Release(asterisk);
    end;
    Halt(ret);
end;

procedure PlaySound(sound: AnsiString);
begin
    //playing one separate sound
    if not (Agi_Play_Sound(asterisk, PChar(sound)) = agiOK) then
    begin
    WriteToLog('PlaySound', rsWarning, 'Can''t play sound: %s', [sound]);
        Finit(1);
    end;
end;

//=============================================================================
//--------------------------Getting Data -------------------------------
function GetData(mainSound: AnsiString; prefixSound: AnsiString = ''):
AnsiString;
var cResult : char64;
begin
    //helper function: returns user's input, ended by '#' sign
    case Agi_Read_Value(asterisk, BUTTON_OK, BUTTON_RETRY,
TRY_COUNT_INPUT, TIMEOUT_INPUT,
                @cResult, c64, PChar(mainSound), PChar(prefixSound)) of
    agiTimeout:
        begin
        WriteToLog('GetData: AgiReadValue', rsWarning,
            'Timeout on: %s %s', [prefixSound, mainSound]);
        Finit(1);
        end;
    agiError:
        begin
        WriteToLog('GetData: AgiReadValue', rsWarning,
            'Error on: %s %s', [prefixSound, mainSound]);
        Finit(1);
        end;
    end;
    Result := StrPas(@cResult);
end;
```

```
function GetMenuData(menuKeys, mainSound: AnsiString; prefixSound:
AnsiString = ''): LongInt;
var i: LongInt;
    menuData: char64;
begin
    //helper function: returns key from the set, pressed by user
    Result := 0;
    for i := 0 to TRY_COUNT_MENU do
    begin
        case Agi_Read_Menu(asterisk, PChar(menuKeys), TRY_COUNT_INPUT,
TIMEOUT_INPUT,
            @menuData, c64, PChar(mainSound), PChar(prefixSound)) of
        agiTimeout:
        begin
            WriteToLog('GetMenuData: AgiReadMenu', rsInfo,
            'Timeout on: %s %s', [prefixSound, mainSound]);
            Finit(1);
        end;
        agiError:
        begin
            WriteToLog('GetMenuData: AgiReadMenu', rsInfo,
            'Error on: %s %s', [prefixSound, mainSound]);
            Finit(1);
        end;
            agiOK:
            begin
            try
            Result := StrToInt(StrPas(@menuData));
        except
            WriteToLog('GetMenuData: AgiReadMenu', rsInfo,
            'Invalid result: %s', [StrPas(@menuData)]);
            Result := -1;
        end;
        exit;
        end;
    end;
    prefixSound := '';
    end;
end;

//=============================================================================
//----------------------------Authorization-----------------------------------
type
TAccountRec = record
    //---------account info-----------
    isAccOK:                Boolean;
    accAmount,
    lastDRAmount, lastDRTrAmount,
    lastCRAmount, lastCRTrAmount:    Double;
    lastDRTrCurrency, lastCRTrCurrency,
    accCurrency, cardStatus:        LongInt;
    cardExpDate,lastDRDate, lastCRDate:    TDateTime;
    //----------authorization---------
    isAuthOK: Boolean;
    cardCode, passCode: AnsiString;
end;
```

```
{type TAuthorizationRec = record
    isOK: Boolean;
    cardCode, passCode: AnsiString;
end;}

procedure AccountInfo(var env); cdecl; forward;

procedure Authorize(var env);    cdecl;
var ret: LongInt;
    resAuth: Boolean;
begin
    //trying to authorize
    resAuth:= DB_Login(session,
    PChar(TAccountRec(env).cardCode), PChar(TAccountRec(env).passCode),
@callerID, ret);

    //authorization failed: writing to log
    if not(resAuth) then
    WriteToLog('DB_Login', rsError,
        'Authorization failed with params:'#10#9'CallerID =
%s'#10#9'PassCode = %s'#10#9'CardCode = %s',
        [callerID, TAccountRec(env).passCode,
TAccountRec(env).cardCode]);

    TAccountRec(env).isAuthOK := resAuth and (ret = 0);

    //authorization successful: getting account data
    if (TAccountRec(env).isAuthOK) then
    AccountInfo(env);
end;

function Authorization(var accRec: TAccountRec): Boolean;
var i:              LongInt;
    prefixSound:    AnsiString;
//    authRec:        TAuthorizationRec;
begin
    Result := false;
    prefixSound := '';

    //3 tries to enter valid pass/card codes for authorization
    for i := 1 to TRY_COUNT_MENU do
    begin
    //getting passport code
        accRec.passCode := GetData(SOUND_AUTH_PASS, prefixSound);
        prefixSound := SOUND_ERROR;
    if (Length(accRec.passCode) <> DOC_DIGIT_COUNT) then continue;

    //getting card code
        accRec.cardCode := GetData(SOUND_AUTH_CARD);
        if (Length(accRec.cardCode) <> CARD_DIGIT_COUNT) then continue;

    //getting authorization for entered data and information about
account,
    //in case authorization was successful. meanwhile playing "wait"
sound
    agi_set_callback(asterisk, @Authorize, accRec);
```

```
    PlaySound(SOUND_OPERATOR);

    //authorization succeed
        if (accRec.isAuthOk) then
        begin
            Result := true;
            exit;
        end;

        //authorization failed
        prefixSound := SOUND_AUTH_ERR;
        if (i = TRY_COUNT_MENU) then PlaySound(prefixSound);
    end;
end;

//=========================================================================
//-------------------------Level 2-----------------------------------
procedure AccountInfo(var env); cdecl;
var res: Boolean;
    ret: LongInt;
begin
    //getting account information: status, balance, transactions,
expiration date
    res := DB_Acc_Info_Pas(session,
    TAccountRec(env).cardStatus,    TAccountRec(env).cardExpDate,
    TAccountRec(env).accAmount,    TAccountRec(env).accCurrency,
    TAccountRec(env).lastDRDate,    TAccountRec(env).lastDRAmount,
    TAccountRec(env).lastDRTrAmount,TAccountRec(env).lastDRTrCurrency,
    TAccountRec(env).lastCRDate,    TAccountRec(env).lastCRAmount,
    TAccountRec(env).lastCRTrAmount,TAccountRec(env).lastCRTrCurrency,
ret);
    WriteToLog('DB_Acc_Info_Pas', rsInfo,
    'Status: %d '#10#9'Amount: %n'#10#9'Cur: %d'#10#9'Date: %s'#10#9'Ret:
%d', [
        TAccountRec(env).cardStatus,
        TAccountRec(env).accAmount,
        TAccountRec(env).accCurrency,
        DateToStr(TAccountRec(env).cardExpDate),
        ret
    ]
    );

    if not(res) then
    WriteToLog('DB_Acc_Info_Pas', rsError
    {$IFDEF TEST}
        ,'Status: %d '#10#9'Amount: %n'#10#9'Date: %n'#10#9'Ret: %d',
        [TAccountRec(env).cardStatus, TAccountRec(env).accAmount,
        TAccountRec(env).cardExpDate, ret]
    {$ENDIF}
    );

    TAccountRec(env).isAccOK := res and (ret = 0);
end;

function C2_PlayAviableAmountAndGetStatus(authNeeded: Boolean; aShortMode
: boolean = false): LongInt;
    //---------------------------
```

```
    procedure PlayTransaction(date: TDateTime;
            accAmount, trAmount:     Double;
            accIndex, trCurrency:     LongInt;
            soundDate, soundAmount:    AnsiString);
    var currencyIndex: LongInt;
    transactionAmount: Double;
    buffer: char2k;
    begin
    //helper procedure: playing last transaction's date and amount

        //if transaction currency ISO doesn't exist - play in account
currency
        if Dict_Iso_To_Index(trCurrency, currencyIndex) then
            transactionAmount := trAmount
        else begin
            //if account currency is invalid either - not playing
amount/date at all
            WriteToLog('PlayTransaction', rsWarning,
            'Can''t convert transaction currency with ISO: %d Amt:%f',
[trCurrency,trAmount]);
            if (accIndex = WRONG_ACC_CURRENCY) then exit;
        transactionAmount := accAmount;
        currencyIndex := accIndex;
        end;

    //play last transaction's date
    Dict_Ru_Play_Date_Pas(PChar(PATH_TO_DICT), date, @buffer, c2k);
        PlaySound(soundDate);
    PlaySound(buffer);

    //play last transaction's amount
    Dict_Ru_Play_Amount_Index(PChar(PATH_TO_DICT), transactionAmount,
        currencyIndex, @buffer, c2k);
    PlaySound(soundAmount);
    PlaySound(buffer);
    end;
    //-------------------------
    function IsZero(balance: Double): Boolean;
    const eps = 1.0E-2;    //1.1E-10;
    begin
        Result := (abs(balance) < eps);
    end;
    //---------------------------
var accRec:         TAccountRec;
    accCurrencyIndex:    LongInt;
    buffer:         char2k;
    isBlocked:         Boolean;
begin
    Result := RES_FAILED;

    //getting account data together with authorization
    if (authNeeded) then
    begin
        if not Authorization(accRec) then Exit;
        AccountInfo(accRec);
    end else
    begin
```

```
        //getting account data without authorization
        agi_set_callback(asterisk, @AccountInfo, accRec);
        PlaySound(SOUND_OPERATOR);
    end;

    //can't get data, system is unaviable
    if not (accRec.isAccOK) then
    begin
        PlaySound(SOUND_NO_CONNECTION);
        WriteToLog('PlayAviableAmount', rsWarning, 'Can''t get account
data', []);
        Exit;
    end;

    Result := accRec.cardStatus;
    isBlocked := (accRec.cardStatus = CARD_STATUS_BLOCKED);

    //0. get account currency index, if doesn't exist - not playing
    if not Dict_Iso_To_Index(accRec.accCurrency, accCurrencyIndex) then
    begin
    accCurrencyIndex := WRONG_ACC_CURRENCY;
    WriteToLog('PlayAviableAmount', rsWarning,
        'Can''t convert account currency with ISO: %d',
[accRec.accCurrency]);
    end;

    //1. building phrases about amounts according to card status
    if (isBlocked) then PlaySound(SOUND_C_BLOCKED);
    if IsZero(accRec.accAmount) then
    begin    //1.1. balance is zero
    if (isBlocked) then PlaySound(SOUND_BLOCK_BAL_ZERO)
        else PlaySound(SOUND_AVAIL_BAL_ZERO);
    end else
    if (accCurrencyIndex <> WRONG_ACC_CURRENCY) then
    begin    //1.2. balance is valid
    if (isBlocked) then PlaySound(SOUND_BLOCK_MOD)
        else PlaySound(SOUND_AVAIL_MOD);
    Dict_Ru_Play_Amount_Index(PChar(PATH_TO_DICT), accRec.accAmount,
        accCurrencyIndex, @buffer, c2k);
    PlaySound(buffer);
    end;

    if (aShortMode) then Exit;

    //2. last debit transaction
    if not(IsZero(accRec.lastDRDate)) then
    PlayTransaction(accRec.lastDRDate,
        accRec.lastDRAmount, accRec.lastDRTrAmount, accCurrencyIndex,
        accRec.lastDRTrCurrency, SOUND_DB_DATE, SOUND_DB_AMOUNT);

    //3. last credit transaction
    if not(IsZero(accRec.lastCRDate)) then
    PlayTransaction(accRec.lastCRDate,
        accRec.lastCRAmount, accRec.lastCRTrAmount, accCurrencyIndex,
        accRec.lastCRTrCurrency, SOUND_CR_DATE, SOUND_CR_AMOUNT);

    //4. play: card expiration date
```

```
    Dict_Ru_Play_Date_Pas(PChar(PATH_TO_DICT), accRec.cardExpDate,
@buffer, c2k);
    PlaySound(SOUND_EXPIRATION);
    PlaySound(buffer);
end;

//---------------block card----------------
type
TBlockCardRec = record
    isOK:    Boolean;
    resState:    LongInt;
    cardCode:    char64;
    cardStatus:    LongInt;
end;

procedure BlockCard(var env); cdecl;
var ret: LongInt;
    res: Boolean;
begin
    //setting card status to "Blocked"
    res :=  DB_Set_Card_Status(session, @TBlockCardRec(env).cardCode,
        CARD_STATUS_BLOCKED, TBlockCardRec(env).cardStatus, ret);

    //T0-DO: manage block authorization (ret?)

    if not(res) then WriteToLog('DB_Set_Card_Status', rsError);
    TBlockCardRec(env).isOK := res and (ret = 0);
    TBlockCardRec(env).resState := ret;
end;

procedure C1_SwitchToOperator; forward;

procedure C2_BlockCard;
var blockRec:    TBlockCardRec;
    codeLength:    LongInt;
begin
    //one try to enter valid card code to confirm card blocking,
otherwise - exit
    case Agi_Read_Value(asterisk, BUTTON_OK, BUTTON_RETRY, 1,
TIMEOUT_INPUT,
    @blockRec.cardCode, c64, PChar(SOUND_BLOCK_VAL)) of
        agiError:
        begin
                WriteToLog('BlockCard: AgiReadValue', rsWarning);
            Finit(1);
        end;
    agiOK: ;
        else Exit;
    end;

    codeLength := StrLen(@blockRec.cardCode);
    if (codeLength = 0) then
    Exit
    else if (codeLength <> CARD_DIGIT_COUNT) then
    begin
        PlaySound(SOUND_ERROR);
    Exit;
```

```
    end;

    agi_set_callback(asterisk, @BlockCard, blockRec);
    PlaySound(SOUND_OPERATOR);

    //TO-DO: manage blockRec.resState ?

    if (blockRec.isOK) then
        PlaySound(SOUND_BLOCK_OK)    //success
    else begin
        PlaySound(SOUND_BLOCK_ERR);    //failure
        C1_SwitchToOperator;
    end;
end;

//-----------------send mail---------------
procedure SendMail(var env); cdecl;
var ret: LongInt;
    res: Boolean;
begin
    res := DB_Statement_Email(session, '', ret);
    if not(res) then WriteToLog('DB_Statement_Email', rsError);
    Boolean(env) := res and (ret = 0);
end;

procedure C2_SendEmailWithOperations;
var isSent: Boolean;
begin
    //sending email with last 10 operations
    agi_set_callback(asterisk, @SendMail, isSent);
    PlaySound(SOUND_OPERATOR);
    if (isSent) then
    PlaySound(SOUND_EMAIL_OK)    //success
    else PlaySound(SOUND_EMAIL_ERR);    //failure
end;

//============================================================================
//---------------------------Level 1-----------------------------------
procedure C1_Menu;
var cardStatus: LongInt;
    ret:    LongInt;
    menuKeys:    Array[Boolean] of AnsiString = ('1,2,3,*', '1,2,*');
    soundChoice: AnsiString;
    isBlocked:    Boolean;
begin
    //authorizing and playing information about account
    cardStatus := C2_PlayAviableAmountAndGetStatus(true,true);
    if (cardStatus = RES_FAILED) then Exit;

    //selecting menu according to card status
    isBlocked := (cardStatus = CARD_STATUS_BLOCKED);
    if (isBlocked) then
    soundChoice := SOUND_CHOICE_3
    else soundChoice := SOUND_CHOICE_2;

    while (true) do
    begin
```

```
    //pause = 2sec
    agi_wait(asterisk, 1);
        case  GetMenuData(menuKeys[isBlocked], soundChoice) of
            1: C2_PlayAviableAmountAndGetStatus(false,false);
            2: C2_SendEmailWithOperations;
            3: C2_BlockCard;
        else break;
        end;
        agi_wait(asterisk, 1);
    end;

    //logging out before going to previous menu
    if not DB_Logout(session, ret) then
    WriteToLog('Menu1: Logout', rsError);
end;

procedure C1_SwitchToOperator;
begin
    PlaySound(SOUND_OPERATOR);

    //redirecting call to operator
    if (Agi_Make_Call(asterisk, @callerID, PChar(OPERATOR_NUMBER),
    PChar(OPERATOR_CONTEXT)) <> agiOK) then
    begin
    WriteToLog('SwitchToOperator', rsError,
        'Failed to switch caller [%s] to operator', [callerID]);
    Finit(1);
    end;

    Finit(0);
end;

//---------------db connection------------
procedure ConnectToDB(var env); cdecl;
begin
    //initializing session and connection to db
    session := DB_Init(PChar(PATH_TO_TEST));
    Boolean(env) := DB_Connect(session, PChar(DB_CONNECT_PARAMS));

    if not(Boolean(env)) then
    WriteToLog('DB_Connect', rsError,
        'Can''t connect to DB with params: %s', [DB_CONNECT_PARAMS]);
end;

//=============================================================================
//==============================main=========================================
//=============================================================================

var isConnected:    Boolean;
    prefixGreeting:    AnsiString;

begin
    //reading all variables from settings file
    if not LoadSettingsForInfoLine then
    begin
    WriteToLog('Main module', rsError, 'Can''t load settings from ini',
[]);
```

```
    Finit(1);
    end;

    //initializing asterisk
    asterisk := Agi_Init();
    if not Assigned(asterisk) then
    begin
    WriteToLog('Main module', rsError, 'Can''t initialize asterisk', []);
    Finit(1);
    end;
    Agi_Answer(asterisk);

    //initializing asterisk variables
    Agi_Variables(asterisk, 'agi_callerid', @callerID, c64);
    Agi_Variables(asterisk, 'agi_uniqueid', @uniqueID, c64);
    WriteToLog('Main module', rsInfo,
    'Starting session with:'#10#9#9'uniqueID = %s'#10#9#9'callerID = %s',
    [uniqueID, callerID]);

    //connecting to db while playing "greeting" and setting connection
status
    agi_set_callback(asterisk, @ConnectToDB, isConnected);

    //main menu - if cannot connect to db allow only to switch to
operator
    prefixGreeting := SOUND_GREETING;
    while true do
    begin
    case GetMenuData('1,2,*', SOUND_CHOICE_1, prefixGreeting) of
        1: if (isConnected)
        then C1_Menu
        else PlaySound(SOUND_NO_CONNECTION);
        2: C1_SwitchToOperator;
        else Finit(1);
    end;
    prefixGreeting := '';
    end;

    FInit(1);
end.
```

DOKUMENTĀRĀ LAPA

Bakalaura darbs „Asterisk tehnoloģiju izmantošana Infoline sistēmas izstrādē" izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai:

Autors: Jekaterina Ļimoņina _____

Rekomendēju darbu aizstāvēšanai:

Vadītājs: M.Dat. Vadims Potehins _____

Recenzents: _____

Darbs iesniegts Datorikas fakultatē: _____

Metodiķe: Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____

Komisijas sekretārs: Uldis Straujums _____