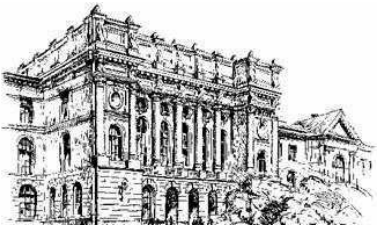
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий



ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Модуль SIP-телефонии для веб-браузера**

Студент гр. 43501/4 В.С. Филиппов

Санкт-Петербург – 2016

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Работа допущена к защите
зав. кафедрой
_____ В.М. Ицыксон
«____» _____ 2016 г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Модуль SIP-телефонии для веб-браузера**

Направление: 230100 – Информатика и вычислительная техника

Выполнил студент гр. 43501/4 _____ В.С. Филиппов
Научный руководитель,
ст. преп. _____ А.В. Зозуля

Санкт-Петербург – 2016

Эта страница специально оставлена пустой.

РЕФЕРАТ

Отчет, 53 стр., 11 рис., 1 табл., 14 ист., 1 прил.

СОФТ-ФОН, IP-ТЕЛЕФОНИЯ, WEB-ПРИЛОЖЕНИЕ

Бакалаврская работа посвящена осуществлению телефонных звонков из браузера. Рассмотрены существующие решения в данной области. Сформулированы требования к программному модулю, осуществляющего звонки из браузера. Такой модуль будет удобен в бизнес-системах и web-приложениях. Разработана архитектура модуля. Написан программный код. Проведено функциональное тестирование. В результате был разработан модуль SIP-телефона для web-броузера, и встроен в CRM-систему.

СОДЕРЖАНИЕ

СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

AEC	Acoustic Echo Cancellation, эхоподавление
AES	Advanced Encryption Standard, один из симметричных алгоритмов блочного шифрования
AGC	Automatic Gain Control, автоматическая регулировка усиления
AJAX	Asynchronous Javascript And Xml
AJB	Adaptive Jitter Buffer, буфер выравнивания задержек передачи
AS3	Action Script 3
CRM	Customer Relationship Management, управление взаимоотношениями с клиентами
DTLS	Datagram Transport Layer Security, протокол датаграмм безопасности транспортного уровня
DTMF	Dual-Tone Multi-Frequency, двухтональный многочастотный аналоговый сигнал
FCS	Flash Communication Server
ICE	Interactive Connectivity Establishment
JRE	Java Runtime Environment
NAT	Network Address Translation, трансляция сетевых адресов

RTC	Real-Time Communications, коммуникации в реальном времени
SDK	Software Development Kit, комплект средств разработки
SIP	Session Initiation Protocol
SRTP	Secure Real-time Transport Protocol, безопасный протокол передачи данных в реальном времени
STUN	Session Traversal Utilities for NAT, утилиты прохождения сессий для NAT
VoIP	Voice over IP, IP-телефония
ПО	программное обеспечение

ВВЕДЕНИЕ

В последнее время очень популярными стали web-приложения, например социальные сети, игры, онлайн-редакторы (документов, изображений и видео), прямые видео-трансляции и многие-многие другие.

Около 15 лет назад клиент просматривал web-страницы, только переходя с одной на другую. Примерно в 2005 году, появился способ сделать страницы динамичными с помощью AJAX. С тех пор, почти весь обмен по HTTP инициировался клиентом разными способами, например каким-нибудь действием, или периодическим опросом сервера на получение новых данных. Однако при таком обмене появляется задержка на установление HTTP-соединения каждый раз при получении новых данных от сервера. Это создавало проблемы для создания web-приложений реального времени.[1]

Около 5 лет назад появилась новая технология, которая позволила обмениваться двум сторонам асинхронно и симметрично. Это полнодуплексный протокол WebSocket, который работает поверх TCP. Уже в 2009 году вышла первая версия браузера, поддерживающая стандарт.[2] А в 2012 году браузер Bowser начал первым в мире, поддерживать технологию WebRTC, которая использует WebSocket и позволяет совершать аудио и видео звонки прямо из браузера.[3]

1. ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ РЕАЛИЗАЦИИ SIP-ТЕЛЕФОНИИ ДЛЯ WEB-БРАУЗЕРА

Для разработки модуля телефонии для web-браузера необходимо сначала проанализировать существующие способы. Рассмотрим их в хронологическом порядке.

1.1. Подход к реализации телефонии для web-браузера на Java

Временем появления телефонии для браузера можно считать момент, когда в Java апплетах появилась поддерживать захвата аудио с микрофона. JRE широко распространена и обычно уже установлена в Windows и Linux системах.[4] Java код выполняется на JRE установленной на компьютере или в расширении браузера, захватывает аудио с микрофона и отправляет его на сервер по протоколу RTP. Такой апплет должен быть подписан, и при его запуске пользователя спросят, желает ли он запустить подписанный апплет от данного производителя, который имеет доступ к функциям сетевого обмена, доступ к микрофону и т.п.

Преимущества данного подхода:

1. Поддерживается большинством браузеров
2. Возможность прямого взаимодействия с сервером по RTP

3. Доступность JRE для конечного пользователя

К сожалению, в Java есть проблемы с обработкой звука в реальном времени. А это почти всё алгоритмы, которые должны быть у каждого VoIP-телефона: AEC, AGC, AJB и Noise suppression (подавление шума).

Эхоподавление позволяет использовать динамики так, чтобы собеседник не слышал собственных слов, которые предаются обратно с динамиков на микрофон. AGC регулирует громкость так, чтобы не было слишком тихо или слишком громко. AJB устраняет большую задержку в передаче и "choppy audio"– прерывистый неразборчивый звук.

Все эти алгоритмы теоретически можно реализовать на Java, но это проблемно. Во-первых, реализовать универсальные и производительные алгоритмы (например, AEC) достаточно сложно. Во-вторых, реализация таких алгоритмов на Java может работать в несколько раз медленнее, чем на C/C++, а это может сказаться с большим расходом ресурсов клиентского CPU.

Производители Java апплетов с функцией звонков реализуют собственные обработчики звука или используют уже существующие решения на C/C++. Они используют в апплете библиотеки, которые берут на себя обработку вышеописанных алгоритмов. В результате Java апплет имеет стандартные VoIP функции для обеспечения качественного звонка со всеми VoIP алгоритмами.

Таким образом, подход к реализации VoIP-телефонии на Java имеет два недостатка:

1. Сложность реализации алгоритмов обработки звука для каждой платформы

2. Отсутствие кроссплатформенности - алгоритмы обработки звука должны быть реализованы на всех платформах, или используемые библиотеки должны быть кроссплатформенными

3. Необходимо устанавливать JRE

Довести DSP до отличного качества или купить соответствующие разработки может позволить себе не каждый вендор. То же касается поддержки различных кодеков для аудио и видео.

1.2. Подход к реализации телефонии для web-браузера на Flash

Начиная с 6 версии Flash Player умел взаимодействовать с FCS MX 1.0 и обмениваться с сервером потоками аудио данных. Он умел захватывать аудио и кодировать его с помощью кодека NellyMoser, и видео и кодировать его с помощью кодека Sorenson Spark. В качестве транспорта для аудио и видео в Flash Player 6 использовался протокол RTMP, который сегодня имеет открытую спецификацию, опубликованную Adobe. До полноценной VoIP-телефонии тогда было еще очень далеко. Но платформа делала свое дело и передавала звук и видео от одного плеера к другому через сервер.

Однако в связке Flash Player 6 + FCS MX 1.0 была задержка звука, она также осталась в следующих версиях сервера, включая последнюю Adobe Media Server. Причина в том, что RTMP протокол

работает поверх TCP, а потому не приспособлен для полноценного VoIP. Для приложений реального времени лучше использовать UDP.

Проблему с UDP в Flash Player решили в 10 версии: ввели поддержку нового протокола RTMFP и функцию AEC. В 11 версии Flash Player добавили поддержку кодеков G.711 и H.264. В AS3 API так же имеются AJB для кодеков G.711 и Speex.

Итак, VoIP алгоритмы, которые поддерживает Flash Player 11: AEC, AJB, AES шифрование. Шифрование AES защищает трафик между браузером и сервером от посторонних.

Но у Flash Player есть небольшая проблема. В документации Adobe AS3 сказано, что RTMFP поддерживает три режима: надежная доставка, частично-надежная доставка, ненадежная доставка. Но есть только два флага для аудио и видео которые принимают либо "true"либо "false". "False"описывается как режим частичной доставки. В итоге, получается, что ненадежную доставку включить не удаётся, а при передачи звука она наиболее важна. Частичная доставка – это TCP ретрансмиты, которые происходят очень ограниченное время, но этого хватает, чтобы испортить звук в нестабильной сети. Такие ретрансмиты вызывают дрожание, которые портит поток. AJB на принимающей стороне не может справится с таким большим разбросом. Решением может оказаться добавление ненадёжной доставки на уровне протокола на серверной стороне.

Таким образом, у подхода к реализации VoIP-телефонии на Flash есть следующие преимущества:

1. Поддерживается большинством браузеров

2. Привычная технология для разработчиков – AS3
3. Качественная передача аудио и видео

Однако имеются и недостатки:

1. Требует промежуточного сервера (не поддерживает открытые UDP протоколы, такие как RTP/SRTP)
2. Отсутствие AGC
3. Необходимо устанавливать Flash Player

1.3. Подход к реализации телефонии для web-браузера на WebRTC

WebRTC - проект с открытым исходным кодом, предназначенный для организации передачи потоковых данных между браузерами или другими поддерживающими его приложениями по технологии точка-точка.[5]

Технология WebRTC имеет продуманную архитектуру, избавленную от ошибок и недостатков, выявленных в плагинах браузера, которые существовали до неё. Технологические возможностях WebRTC: SRTP, DTLS, ICE, STUN, AEC, AGC, AJB, аудиокодек Opus, видеокодек VP8.

Набор используемых в WebRTC технологий больше похож на VoIP SDK. SRTP и DTLS обеспечивает защиту трафика между WebRTC узлами. ICE и STUN помогают преодолеть NAT.[6] AEC, AGC и AJB работают для того чтобы сделать аудио и видео качественным – без лагов и задержек. Кодеки Opus и VP8 хорошо подходят для

Таблица 1.1. Поддержка WebRTC web-браузерами

Web-браузер	**Версия**
IE	11
Edge	13
Firefox	45
Chrome	29
Safari	9.1
Opera	38
iOS Safari	8.4
Opera Mini	-
Android Browser	4.4
Chrome for Android	50

глобального Интернета, где скорость соединения может неожиданно падать.

Однако надо отметить, что подходы к реализации VoIP-телефонии в браузере, рассмотренные ранее (Java и Flash) требуют дополнительной установки ПО. WebRTC – это единственная технология, которая является родной для браузера. Сегодня уже достаточно большое количество web-браузеров поддерживают WebRTC, подробная информация приведена в таблице 1.1.[7]

Преимущества технологии WebRTC:

1. Все алгоритмы обработки звука

2. Технология встроена в браузер
3. Совместимость с традиционными VoIP
4. Реализован на популярном среди web-разработчиков языке JavaScript
5. Поддерживается многими серверами IP-телефонии (Asterisk, FreeSWITCH, Kamailio, OverSIP, OfficeSIP и др.)

Недостатки технологии WebRTC:

1. RFC ещё не разработан, на сегодняшний день существует черновик[8]
2. Поддерживается не всеми браузерами

Как мы видим преимуществ у подхода для разработки модуля SIP-телефонии на основе технологии WebRTC больше. Недостатки же в ближайшее время будут устранятся.

2. ПОСТАНОВКА ЗАДАЧИ, ВЫБОР СПОСОБА РЕШЕНИЯ

2.1. Постановка задачи

В современном мире коммуникации через интернет становятся очень удобными, и экономят время. Такие коммуникации особенно будут экономить время, если их встроить в CRM-системы.

В связи с этим задача будет написать модуль SIP-телефона для web-браузера. Такой модуль должен быть кросс-браузерным, и иметь возможность без сложностей встраиваться в любое web-приложение, например, CRM-систему.

Обычно софт-фоны поддерживают передачу видео, аудио и мгновенных сообщений. Однако в данной работе ограничимся только передачей аудио потока.

2.2. Выбор способа решения

Для реализации данной задачи, будем использовать технологию WebRTC (см. главу 1). Существуют две реализации технологии WebRTC по протоколу SIP на JavaScript: библиотека sipML5 и библиотека JsSIP.[9][10]

Размер библиотеки JsSIP 130kb. Размер библиотеки sipML5 чуть больше 1Mb. Однако JsSIP требует установки Node.js на сервере.

Обе библиотеки довольно неплохие и имеют следующие преимущества:

1. хорошая документация
2. возможность осуществления аудио и видео звонков
3. возможность отправки мгновенных сообщений
4. поддержка статуса присутствия в сети
5. функция удержания вызова
6. функция отключения микрофона
7. тональный набор (DTMF)

Помимо этого в библиотеке sipML5 имеется функции трансляции экрана (пока что только для Google Chrome), перенаправления вызова и группового звонка.

Хоть и обе библиотеки довольно хороши, для нашей задачи выберем библиотеку sipML5, потому что функциональность у неё больше и не нужно дополнительно устанавливать Node.js на сервер.

На рисунке 2.1 под номером 4 изображён разрабатываемый модуль. У которого имеется должно быть ядро, которое выполняет функции звонков, и модули подключения к конкретной CRM-системе: Authentication, Calls, Click-to-call.

Ядро должно содержать код, который оборачивает библиотеку sipML5 и предоставляет функции аутентификации на сервере и звонка. Так же должно включать независимую от библиотеки часть, которая контролирует состояние сессий аутентификации и звонка.

Модуль Authentication должен выполнять запрос данных о SIP-аккаунте пользователя CRM-системы. Модуль Calls должен регистри-

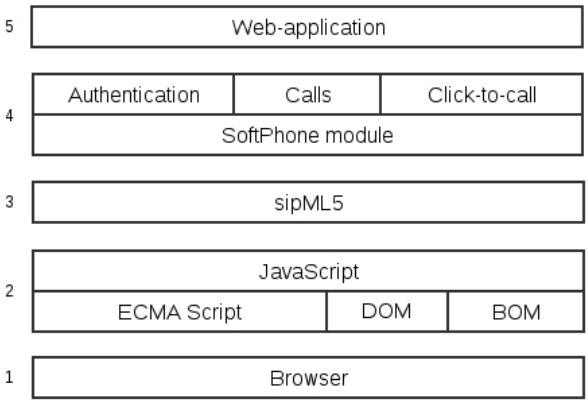


Рисунок 2.1. Структура взаимодействия модулей

ровать звонки в CRM-системе. Модуль Click-to-call должен открывать софт-фон по клику на номер телефона, и заполнять им поле вызова.

Графический интерфейс должен состоять из скользящей кнопки и плавающего окна, которое появляется и скрывается по нажатию на эту кнопку. Плавающее окно должно включать кнопки вызова и сброса, которые меняются во время входящего звонка на кнопки снятия трубки и отклонения звонка соответственно.

3. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ МОДУЛЯ

На рисунке 3.1 изображена архитектура модуля. Опишем каждую его часть.

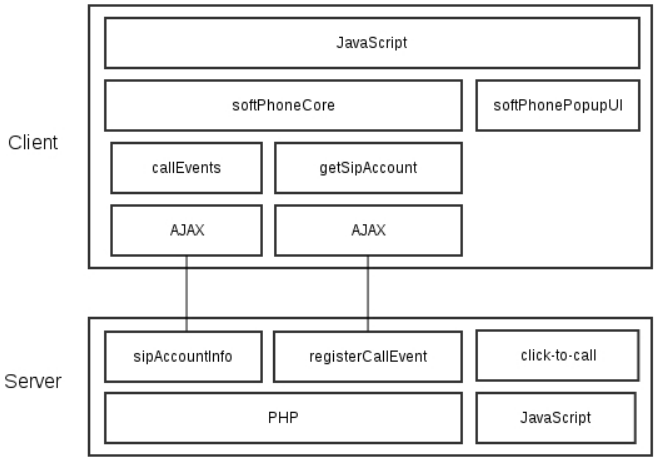


Рисунок 3.1. Архитектура модуля

Основой модуля является его ядро softPhoneJS. Ядро обеспечивает взаимодействие с sipML5 и переключает состояние звонка. При переключении состояния информация передаётся в модуль CallEvents, который является зависимым от web-приложения, и далее отправляет информацию на сервер, для регистрации звонка.

Только перед тем как звонить, необходимо получить из CRM-системы данные о SIP-аккаунте. За это отвечает модуль

getSipAccount.

SoftPhonePopupUI отвечает за графический интерфейс скользящей кнопки и плавающего окна. Так же графический интерфейс реагирует на состояние звонка, например, запрещая пользователю нажимать на кнопку сброса пока звонок ещё не начался.

Click-to-call реализован на сервере, так как он должен добавлять функцию обработчика на html-страницу, получаемую от сервера. Обработчик вызывается при нажатии на номер телефона на странице web-приложения и заполняет поле номера на плавающем окне.

4. РАЗРАБОТКА

4.1. Разработка подмодулей, не зависящих от web-приложения

Разрабатываемый модуль должен быть легко подключаем к web-приложению. Поэтому было решено сделать файл softPhone.js, который подключает модуль softPhone.html AJAX-запросом. SoftPhone.html включает в себя разметку кнопок плавающего окна, подключает аудио-файлы звонка и гудков, а также подключает остальные подмодули:

- SIPml-api.js - подмодуль библиотеки sipML5.
- softPhoneCore.js - подмодуль, отвечающий за аутентификацию клиента на сервере SIP-телефонии, и осуществление звонков. Также в нём отслеживаются события звонка (CallEvents).
- softPhonePopupUI.js - подмодуль обработки плавающего окна при перетаскивании мышью.
- softPhone.css - подмодуль разметки плавающего окна.
- softPhone_"web-app".js[1] - подмодуль, в котором размещается функция getSipAccount() для конкретного web-приложения.
- softPhone_"web-app".php - подмодуль, отправляющий sipAccountInfo клиенту.

[1] здесь "web-app"означает название web-приложения, к которому будет подключатся модуль

Основные функции подмодуля softPhoneCore.js:

- createSipStack() - инициирует конфигурацию SIP-соединения, основываясь на информации полученной getSipAccount().
- register() - аутентификация SIP-клиента на сервере.
- unregister() - деаутентификация SIP-клиента на сервере, выполняется при закрытии web-страницы, если не было сделанно вручную.
- onSipEventStack(e) - обработчик событий управляющей созданием или завершением SIP-сессий. Сессия аутентификации создаётся после аутентификации, а сессия звонка создаётся при входящем или исходящем звонке. Такие события поступают от сервера.
- onSipEventSession(e) - обработчик событий SIP-сессий. Обрабатываются события ответа на звонок и сброса. Такие события поступают от собеседника.
- call() - обработчик кнопки вызова.
- hangup() - обработчик кнопки сброса.
- finalState(action) - функция конечного автомата, управляющего состоянием SIP-клиента (см. рисунок 4.1). В состоянии calling мы слышим гудки, а в состоянии incoming мы слышим рингтон. Также здесь происходит включение и выключение кнопок Call и HangUp.

- startRingTone(), stopRingTone(), startRingbackTone() и stopRingbackTone() - функции воспроизведения и останова воспроизведения рингтона и гудков.

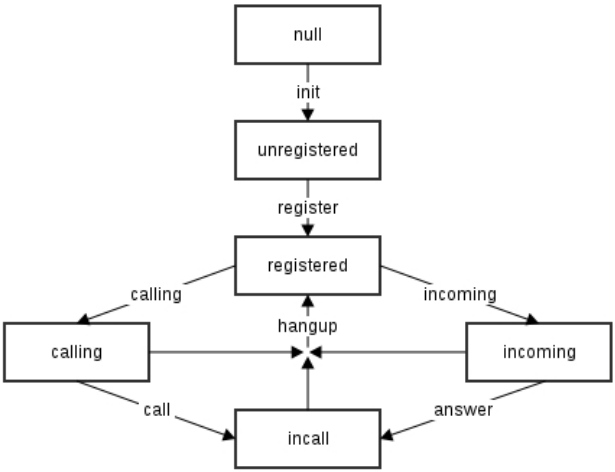


Рисунок 4.1. Конечный автомат, управляющий состоянием SIP-клиента

Основные функции подмодуля softPhonePopupUI.js:

- show_hide_popup() - функция отображения и скрытия всплывающего окна.

- mouseDown(e), mouseUp() и popupMove(e) - функции перетаскивания всплывающего окна.

4.2. Разработка подмодулей, зависящих от web-приложения, на примере SalesPlatform vtiger CRM 6.4

Модуль софт-фона будет очень полезен для CRM-систем. Операторам компаний будет намного удобнее осуществлять звонки из CRM-системы прямо в браузере и создавать комментарии к звонкам. В CRM-системе SalesPlatform vtiger CRM 6.4 имеется модуль PBXManager, который осуществляет звонки при помощи стороннего настольного или мобильного приложения, или же аппаратного SIP-телефона. При осуществлении вызова через CRM-систему, система сообщает серверу SIP-телефонии, что происходит вызов, тогда стороннее приложение или аппаратный SIP-телефон начинает звонить. Во время звонка в CRM-системе создаётся карточка звонка.

Подмодули softPhone_vtiger.js и softPhone_vtiger.php были разработаны для CRM-системы SalesPlatform vtiger CRM 6.4. Подмодуль softPhone_vtiger.js содержит функцию getSipAccount(), которая осуществляет AJAX-запрос данных о SIP-аккаунте (ip-адрес и порт сервера телефонии, SIP-номер и пароль для текущего пользователя CRM-системы). А подмодуль softPhone_vtiger.php отвечает на этот запрос. Для этого он анализирует данные о сессии пользователя, и, обращаясь к другим модулям CRM-системы, получает информацию о SIP-аккаунте текущего пользователя.

Так же подмодуль softPhone_vtiger.js должен генерировать и отправлять данные о состоянии звонка, а softPhone_vtiger.php должен их получать и генерировать карточки звонков, которые заполняются

операторами.

Однако эти задачи очень трудоёмки, так как исходный код данной CRM-системы очень большой, и поэтому выполнены не были. Получение информации о SIP-аккаунте было прописано только для одного пользователя. То есть любой пользователь, кто использует разработанный нами софт-фон, авторизуется за одного и того же пользователя на SIP-сервере. Так же не было реализовано создание карточек звонков в CRM-системе.

В функцию click-to-call модуля PBXManager было добавленно несколько строк кода, заполняющих поле набираемого номера в нашем модуле. То есть осуществив patch PBXManager, легко можно добавить click-to-call в разрабатываемый модуль.

5. ТЕСТИРОВАНИЕ, АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1. Тестовое окружение

На сервере установлена операционная система Debian GNU/Linux 8.4 (jessie). На сервере установлен сервер IP-телефонии Asterisk 13.7.2. В конфигурационных файлах настроены звонки с номерами формата XXX, и добавлены внутренние номера, позволяющие звонить при помощи web-сокетов по порту 8088. Для этого создан DTLS-сертификат, и включена поддержка icesupport.[11] Также добавлены номера, позволяющие звонить при помощи обычных сокетов по порту 5060.

На этом же сервере установлена CRM-система SalesPlatform vtiger CRM 6.4, которая работает на сервере Apache 2.4.10, в связке с PHP 5.6.22-0+deb8u1 и MySQL Server 5.5.49-0+deb8u1. Файлы разрабатываемого модуля помещены в корень CRM-системы. Чтобы подключить JavaScript файл к данной CRM-системе, необходимо добавить в базу данных CRM-системы в таблицу vtiger_links запись с полями: linkurl - название файла (в нашем случае softPhone.js), linktype - HEADERSCRIPT, tabid - 0.[12] В этом случае скрипт будет выполнятся при каждой загрузке страницы.

Тестирование проводилось на браузерах Iceweasel 38.7.1, Mozilla Firefox 47.0, в них звонок работает хорошо. Также проводилось тестирование в браузерах Google Chrome 50.0.2661.75, Opera 38.0.2220.31, но на них не удалось полностью управлять звонком: иногда голос передавался только в одну сторону, иногда не удавалось его завершить.

Однако следует заметить, что демо-версия sipML5 клиента[13] работает на данной тестовой системе так же. В связи с этим можно сделать вывод, Google Chrome ведёт себя так из-за нововведения в версии 47, которое запрещает функцию getUserMedia() если используется протокол http, и позволяет её только для https.[14]

5.2. Функциональное тестирование

Требования к работе модуля:

- Аутентификация и деаутентификация на сервере.
- Сессия аутентификации должна корректно переходить из одного состояния в другое.
- Корректные инициализация вызова и воспроизведение рингтона и гудков.
- Сессия звонка должна корректно переходить из одного состояния в другое.
- При звонке передача голоса должна быть в обе стороны.
- В графическом интерфейсе в определённые состояния звонка должны быть неактивны определённые кнопки. Например, в момент разговора кнопка звонка должна быть выключена и доступна только кнопка сброса.
- Корректная обработка звонков на несуществующие номера.

Функциональное тестирование проводилось вручную. Проверялись инициализация вызова как входящего, так и исходящего. Проверялось завершение вызова, инициируемое как нашим клиентом, так и клиентом собеседника. Завершение вызова проверялось как в момент разговора, так и в момент до снятия трубки.

На рисунках 5.1 - 5.8 изображён корректный графический интерфейс модуля в разных состояниях.

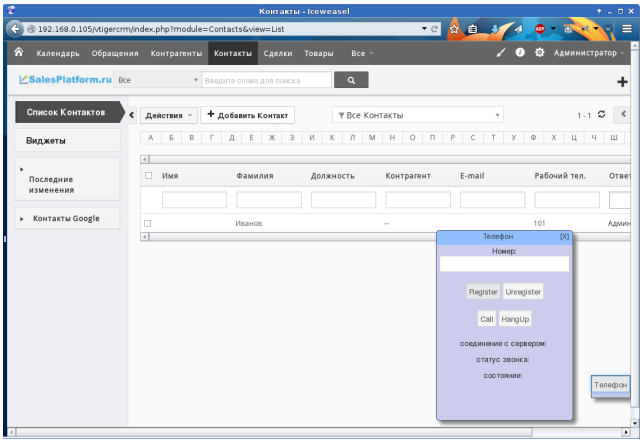


Рисунок 5.1. Модуль в CRM-системе SalesPlatform vtiger CRM 6.4, начальное состояние модуля

На рисунке 5.1 видно кнопку "телефон". Это скользящая кнопка, выполняет функцию скрытия и отображения плавающего окна софтфона.

Тестирование проводилось по ходу разработки, помогало выявить следующие функциональные ошибки: некорректное состояние сессии звонка, односторонняя передача голоса и некорректное состояние гра-

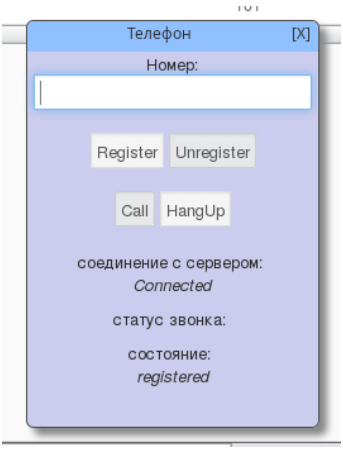


Рисунок 5.2. Состояние после аутентификации на сервере

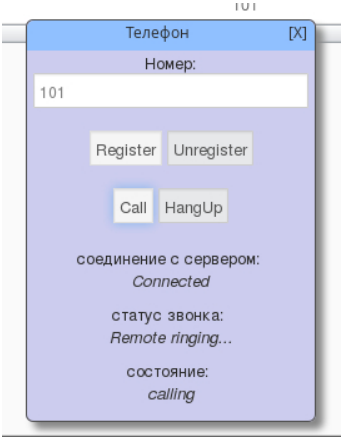


Рисунок 5.3. Состояние исходящего звонка (воспроизводятся гудки)

фического интерфейса.

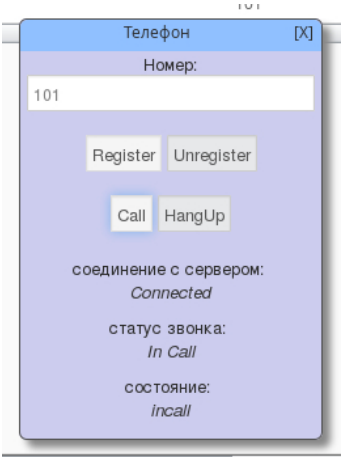


Рисунок 5.4. Состояние в процессе звонка

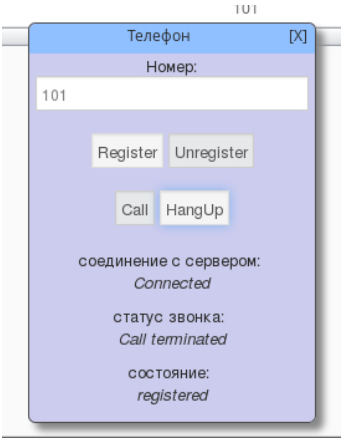


Рисунок 5.5. Состояние завершения звонка, такое же как и состояние после аутентификации на сервере

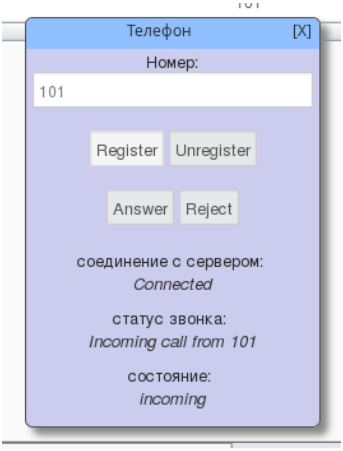


Рисунок 5.6. Состояние входящего звонка (воспроизводится рингтон)

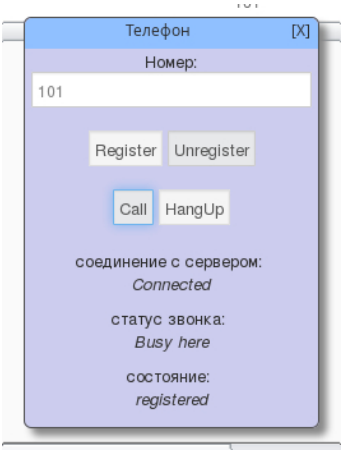


Рисунок 5.7. Вызываемый абонент занят

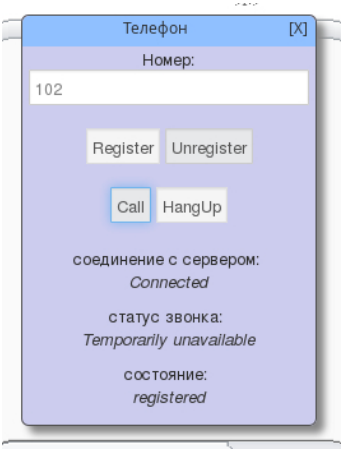


Рисунок 5.8. Вызываемый абонент недоступен

ЗАКЛЮЧЕНИЕ

Пока что главной проблемой технологии WebRTC является её новизна. Пока что не удалось написать полностью кросс-браузерный код разработанного софт-фона. Было потрачено огромное количество времени на то, чтобы разобраться как использовать sipML5 для звонков. Документация оказалась менее подробной чем ожидалось. В связи с этим утерянным временем не удалось осуществить внедрение данного модуля в CRM-систему.

Однако основная часть модуля была написана. Звонки тестировались из браузера Firefox на сконфигурированном для WebRTC сервере IP-телефонии Asterisk. Остальные браузеры звонили, но с некоторыми неудачами. И это скорее всего вина не наша, так как демо-версия библиотеки sipML5 в тестируемых нами браузерах ведёт себя аналогично.

В итоге разработана alfa-версия модуля, встраиваемого в CRM-систему, который нужно доделывать. А именно необходимо реализовать реализовать серверную часть, которая получает информацию о SIP-аккаунте; регистрацию звонков в CRM-системе; доделать ядро модуля так, чтобы звонки осуществлялись в почти любом современном браузере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Introducing WebSockets: Bringing Sockets to the WebElectronic resource, HTML5 Rocks. — URL: http://www.html5rocks.com/en/tutorials/websockets/basics/ (online; accessed: 19.06.2016).
2. Introducing WebSockets: Bringing Sockets to the WebElectronic resource, Chromium Blog. — URL: http://blog.chromium.org/2009/12/web-sockets-now-available-in-google.html (online; accessed: 19.06.2016).
3. Bowser - WebRTC on iOSElectronic resource, Ericsson Research. — URL: http://www.openwebrtc.org/bowser/ (online; accessed: 19.06.2016).
4. WebRTC, Flash RTMFP, Java Applet – три ведущих технологии для браузерных VoIP звонков [Электронный ресурс], CNews Клуб. — URL: http://club.cnews.ru/blogs/entry/webrtc_flash_java_ (дата обращения: 16.06.2016).
5. WebRTCElectronic resource, WebRTC. — URL: https://sites.google.com/site/webrtc/home (online; accessed: 16.06.2016).
6. Всё, что вы хотели знать о протоколе SIP [Электронный ресурс], Системный администратор. — URL: http://samag.ru/archive/article/2017 (дата обращения: 16.06.2016).
7. Can i use... Support tables for HTML5, CSS3, etcElectronic resource, Can i use. — URL: http://caniuse.com/#feat=websockets (online; accessed: 19.06.2016).

8. Bergkvist Adam, Burnett Daniel C., Jennings Cullen et al. WebRTC 1.0: Real-time Communication Between BrowsersElectronic resource // Internet Requests for Comments, W3C. — 2016. — May. — URL: https://www.w3.org/TR/2016/WD-webrtc-20160531/ (online; accessed: 16.06.2016).
9. World's first HTML5 SIP clientElectronic resource, Doubango Telecom. — URL: https://www.doubango.org/sipml5/ (online; accessed: 19.06.2016).
10. the JavaScript SIP library JsSIPElectronic resource, JsSIP. — URL: http://jssip.net/ (online; accessed: 19.06.2016).
11. Digium, Inc. — Asterisk WebRTC Support. — URL: https://wiki.asterisk.org/wiki/display/AST/Asterisk+WebRTC+Support (online; accessed: 19.06.2016).
12. vtigerCRM Data Model. — URL: https://www.vtiger.com/products/crm/docs/510/vtigerCRM_DataModel_5.2.1.pdf (online; accessed: 19.06.2016).
13. sipML5 live demoElectronic resource, Doubango Telecom. — URL: https://www.doubango.org/sipml5/call.htm?svn=250 (online; accessed: 19.06.2016).
14. Chrome 47 WebRTC media recording, secure origins & proxy handlingElectronic resource, Google Developers. — URL: https://developers.google.com/web/updates/2015/10/chrome-47-webrtc (online; accessed: 19.06.2016).

ПРИЛОЖЕНИЕ А

ЛИСТИНГИ

Листинг А.1. Главный файл softPhone.js

```
var server_ip;
var server_port;
var sip_number;
var sip_password;

$.ajax({
    url: "softPhone.html",cache: false,success: function(html){
        $( "body" ).append( html );

        getSipAccount();
    }
});
```

Листинг А.2. Файл html-разметки softPhone.html

```
    <script src="SIPml-api.js?svn=230" type="text/javascript" defer
        ></script>
    <script src="softPhone_vtiger.js" type="text/javascript" defer
        ></script>
    <script src="softPhoneCore.js" type="text/javascript" defer></
        script>
    <script src="softPhonePopupUI.js" type="text/javascript" defer
        ></script>
    <link rel="stylesheet" type="text/css" href="softPhone.css">

    <div id="popup">
        <div id="popup_bar">
            Телефон
            <span id="btn_close"> [X] </span>
        </div>
        <p>
```

```
13              Номер: <input id="phone_popup_number" type="text" size="
                    16">
14          </p>
15          <p>
16              <input type="button" id="btnRegister" value="Register"
                    onclick='register();' />
17              <input type="button" id="btnUnRegister" value="
                    Unregister" onclick='unregister();' disabled/>
18          </p>
19          <p>
20              <input type="button" disabled id="btnCall" value="Call"
                    onclick='call();' />
21              <input type="button" disabled id="btnHangUp" value="
                    HangUp" onclick='hangup();' />
22          </p>
23          <div style="font-size:10pt;">
24          <p>
25              соединение с сервером: 
26              <label id="txtRegStatus"></label>
27          </p>
28          <p>
29              статус звонка: 
30              <label id="txtCallStatus"></label>
31          </p>
32          <p>
33              состояние: 
34              <label id="txtState"></label>
35          </p>
36          </div>
37      </div>
38
39      <div id="aside_popup_phone">
40          <input type="button" id="btnShowHidePopup" value="Телефон"
                onclick='show_hide_popup();' />
41      </div>
42      <audio id="audio_remote" autoplay="autoplay" />
43      <audio id="ringtone" loop src="sounds/ringtone.wav" />
44      <audio id="ringbacktone" loop src="sounds/ringbacktone.wav" />
```

Листинг А.3. Файл стилей html-разметки softPhone.css

```
#popup{
    display:none;
    background-color:#ccccee;
    position:fixed;
    box-shadow: 6px 6px 5px #888888;
    border-radius:6px;
    border:1px solid #4f4f4f;
    text-align:center;
    z-index: 1;
}

#popup_bar{
    width:100%;
    background-color:#90c0ff;
    position:relative;
    border-radius:6px 6px 0 0;
    text-align:center;
    height:24px;
    cursor:move;
}

#btn_close{
    float:right;
    cursor:pointer;
    padding-right:6px;
}

#txtRegStatus{
    font-style: italic;
}

#txtCallStatus{
    font-style: italic;
}

#txtState{
    font-style: italic;
```

```
38 }
39
40 #aside_popup_phone {
41     display:block;
42     background-color:#ccccee;
43     position:fixed;
44     right:0px;
45     bottom:50px;
46     box-shadow: 6px 6px 5px #888888;
47     border:1px solid #4f4f4f;
48     z-index: 1;
49 }
```

Листинг А.4. Файл ядра softPhoneCore.js

```
1  var sipStack;
2  var registerSession;
3  var callSession = null;
4
5
6  var state = "null";//null, unregistered, registered, calling,
       incoming, incall
7
8  function createSipStack(){
9      console.log("createSipStack 1");
10     sipStack = new SIPml.Stack({
11             realm: server_ip,
12             impi: sip_number,
13             impu: 'sip:' + sip_number + '@' + server_ip + '
                   192.168.0.105',
14             password: sip_password,
15             websocket_proxy_url: 'ws://' + server_ip + ':' +
                   server_port + '/ws',
16             ice_servers: "[{ url: 'stun:stun.l.google.com:19302'}]",
17             enable_rtcweb_breaker: false,
18             events_listener: { events: '*', listener:
                   onSipEventStack }, // optional: '*' means all
                   events
19             sip_headers: [ // optional
```

```
20                      { name: 'User-Agent', value: 'IM-client/OMA1.0
                            sipML5-v1.0.0.0' },
21                      { name: 'Organization', value: 'TODO' }
22              ]
23          }
24      );
25      console.log("createSipStack 2");
26  }
27
28  var readyCallback = function(e){
29      createSipStack();
30      console.info('sipml5 is ready');
31  };
32  var errorCallback = function(e){
33      console.error('Failed to initialize the engine: ' + e.message);
34  }
35
36  window.onbeforeunload = function () {
37      unregister();
38      localStorage['window_count'] = +localStorage['window_count'] -
            1;
39      document.getElementById('window_count_label').innerHTML =
            localStorage['window_count'];
40      console.log("onbeforeunload " + localStorage['window_count']);
41  };
42
43  // sends SIP REGISTER request to login
44  function register(){
45      //from onload begin
46      //init finalState
47      state = "null";
48      txtState.innerHTML = state;
49      finalState("init");
50      //init SIPML5
51      SIPml.init(readyCallback, errorCallback);//createSipStack
52      //from onload end
53
54      console.log("register 1");
55      sipStack.start(); //event 'started', register
```

```
56      finalState("register");
57      console.log("register 2");
58  }
59
60  // sends SIP REGISTER (expires=0) to logout
61  function unregister() {
62      console.log("unregister 1");
63      if (sipStack) {
64          sipStack.stop(); // shutdown all sessions
65      }
66      finalState("unregister");
67      console.log("unregister 2");
68  }
69
70  function onSipEventStack(e){
71      console.log("onSipEventStack 1, type = " + e.type);
72      switch (e.type) {
73          case 'started':
74          { //register
75              registerSession = sipStack.newSession('register', {
76                  events_listener: { events: '*', listener:
                        onSipEventSession } // optional: '*' means all
                        events
77              });
78              registerSession.register();
79              break;
80          }
81          case 'i_new_call': // incoming audio/video call
82          {
83              if (callSession) {
84                  // do not accept the incoming call if we're already
                        'in call'
85                  e.newSession.hangup(); // comment this line for
                        multi-line support
86              }
87              else {
88                  callSession = e.newSession;
89                  // start listening for events
90                  callSession.setConfiguration({
```

```
91                          audio_remote: document.getElementById('
                                audio_remote'),
92                          events_listener: { events: '*', listener:
                                onSipEventSession } // optional: '*'
                                means all events
93                      });
94
95                  var sRemoteNumber = (callSession.
                        getRemoteFriendlyName() || 'unknown');
96                  txtCallStatus.innerHTML = txtCallStatus.value = "
                        Incoming call from " + sRemoteNumber;
97                  console.log("call = " + txtCallStatus.value);
98
99                  finalState("incoming");
100             }
101             break;
102         }
103         case 'stopping': case 'stopped': case 'failed_to_start':
                case 'failed_to_stop':
104         {
105             var bFailure = (e.type == 'failed_to_start') || (e.type
                    == 'failed_to_stop');
106             oSipStack = null;
107             oSipSessionRegister = null;
108             oSipSessionCall = null;
109
110             finalState("unregister");
111
112             txtCallStatus.innerHTML = txtCallStatus.value = "";
113             console.log("call = " + txtCallStatus.value);
114             txtRegStatus.innerHTML = txtRegStatus.value = bFailure ?
                     "Disconnected: " + e.description + "" : "
                    Disconnected";
115             console.log("reg = " + txtRegStatus.value);
116             break;
117         }
118         case 'starting': default: break;
119     }
120     console.log("onSipEventStack 2");
```

```
121 }
122
123 function onSipEventSession(e){
124     console.log("onSipEventSession 1");
125     console.info('session event = ' + e.type);
126     switch (e.type)
127     {
128         case 'connecting': case 'connected':
129         {
130             var bConnected = (e.type === 'connected');
131             if (e.session === registerSession) {
132                 txtRegStatus.innerHTML = txtRegStatus.value = e.
                        description;
133                 console.log("reg = " + txtRegStatus.value);
134             }
135             else if (e.session === callSession) {
136                 if (bConnected) {
137                     finalState("call_or_answer");
138                 }
139                 txtCallStatus.innerHTML = txtCallStatus.value = e.
                        description;
140                 console.log("call = " + txtCallStatus.value);
141             }
142             break;
143         } // 'connecting' | 'connected'
144         case 'terminating': case 'terminated':
145         {
146             if (e.session == registerSession) {
147                 callSession = null;
148                 registerSession = null;
149
150                 txtRegStatus.innerHTML = txtRegStatus.value = e.
                        description;
151                 console.log("reg = " + txtRegStatus.value);
152             }
153             else if (e.session == callSession) {
154                 txtCallStatus.innerHTML = txtCallStatus.value = e.
                        description;
155                 console.log("call = " + txtCallStatus.value);
```

```
156                 callSession = null;
157             }
158             if (e.type === 'terminated'){
159                 finalState("hangup");
160                 txtCallStatus.innerHTML = txtCallStatus.value = e.
                        description;
161                 console.log("call = " + txtCallStatus.value);
162             }
163             break;
164         } // 'terminating' | 'terminated'
165         case 'i_ao_request':
166         {
167             if(e.session == callSession){
168                 var iSipResponseCode = e.getSipResponseCode();
169                 if (iSipResponseCode == 180 || iSipResponseCode ==
                        183) {
170                     finalState('calling');
171                     txtCallStatus.innerHTML = txtCallStatus.value =
                            'Remote ringing...';
172                     console.log("call = " + txtCallStatus.value);
173                 }
174             }
175             break;
176         }
177     }
178     console.log("onSipEventSession 2");
179 }
180
181 function eventsListener(e){
182     console.info('session event = ' + e.type);
183 }
184
185 function call(){
186     console.log("call 1" + " " + document.getElementById("
            phone_popup_number").value.toString());
187     if (callSession === null) {
188         console.log("new call");
189         if (document.getElementById("phone_popup_number").value.
                toString() === ""){
```

```
190             alert("введите номер");
191             return;
192         }
193         callSession = sipStack.newSession('call-audio', {
194                 audio_remote: document.getElementById('audio_remote'
                        ),
195                 events_listener: { events: '*', listener:
                        onSipEventSession } // optional: '*' means all
                        events
196             });
197         callSession.call(document.getElementById("phone_popup_number
                ").value.toString());
198     }
199     else {
200         txtCallStatus.innerHTML = txtCallStatus.value = 'Connecting
                ...';
201         console.log("call = " + txtCallStatus.value);
202         callSession.accept();
203         finalState("incall");
204     }
205     console.log("call 2");
206 }
207
208 // terminates the call (SIP BYE or CANCEL)
209 function hangup() {
210     if (callSession) {
211         txtCallStatus.innerHTML = txtCallStatus.value = 'Terminating
                the call...';
212         console.log("call = " + txtCallStatus.value);
213         callSession.hangup({events_listener: { events: '*', listener
                : onSipEventSession }});
214     }
215 }
216
217
218 function finalState(action){
219     switch (state)
220     {
221         case "null":
```

```
222         {
223             if (action === "init"){
224                 btnCall.value = 'Call';
225                 btnHangUp.value = 'HangUp';
226                 btnCall.disabled = true;
227                 btnHangUp.disabled = true;
228                 btnRegister.disabled = false;
229                 btnUnRegister.disabled = true;
230                 state = "unregistered";
231             }
232             break;
233         }
234         case "unregistered":
235         {
236             if (action === "register") {
237                 btnCall.value = 'Call';
238                 btnHangUp.value = 'HangUp';
239                 btnCall.disabled = false;
240                 btnHangUp.disabled = true;
241                 state = "registered";
242                 btnRegister.disabled = true;
243                 btnUnRegister.disabled = false;
244             }
245             break;
246         }
247         case "registered":
248         {
249             if (action === "calling") {
250                 btnCall.value = 'Call';
251                 btnHangUp.value = 'HangUp';
252                 btnCall.disabled = true;
253                 btnHangUp.disabled = false;
254                 state = "calling";
255                 startRingbackTone();
256             }
257             if (action === "incoming") {
258                 btnCall.value = 'Answer';
259                 btnHangUp.value = 'Reject';
260                 btnCall.disabled = false;
```

```
261                 btnHangUp.disabled = false;
262                 state = "incoming";
263                 startRingTone();
264             }
265             break;
266         }
267         case "calling":
268         {
269             if (action === "call_or_answer") {
270                 btnCall.value = 'Call';
271                 btnHangUp.value = 'HangUp';
272                 btnCall.disabled = true;
273                 btnHangUp.disabled = false;
274                 state = "incall";
275                 stopRingbackTone();
276             } else
277             if (action === "hangup") {
278                 btnCall.value = 'Call';
279                 btnHangUp.value = 'HangUp';
280                 btnCall.disabled = false;
281                 btnHangUp.disabled = true;
282                 state = "registered";
283                 stopRingbackTone();
284             }
285             break;
286         }
287         case "incoming":
288         {
289             if (action === "call_or_answer") {
290                 btnCall.value = 'Call';
291                 btnHangUp.value = 'HangUp';
292                 btnCall.disabled = true;
293                 btnHangUp.disabled = false;
294                 state = "incall";
295                 stopRingTone();
296             } else
297             if (action === "hangup") {
298                 btnCall.value = 'Call';
299                 btnHangUp.value = 'HangUp';
```

```
300                 btnCall.disabled = false;
301                 btnHangUp.disabled = true;
302                 state = "registered";
303                 stopRingTone();
304             }
305             break;
306         }
307         case "incall":
308         {
309             if (action === "hangup") {
310                 btnCall.value = 'Call';
311                 btnHangUp.value = 'HangUp';
312                 btnCall.disabled = false;
313                 btnHangUp.disabled = true;
314                 state = "registered";
315             }
316             break;
317         }
318     }
319     if (action === "unregister"){
320         btnCall.value = 'Call';
321         btnHangUp.value = 'HangUp';
322         btnCall.disabled = true;
323         btnHangUp.disabled = true;
324         btnRegister.disabled = false;
325         btnUnRegister.disabled = true;
326         state = "unregistered";
327         stopRingTone();
328         stopRingbackTone();
329     }
330     txtState.innerHTML = state;
331     console.log("state = " + state);
332 }
333
334 function startRingTone() {
335     try { ringtone.play(); }
336     catch (e) { }
337 }
338 function stopRingTone() {
```

```
339     try { ringtone.pause(); }
340     catch (e) { }
341 }
342 function startRingbackTone() {
343     try { ringbacktone.play(); }
344     catch (e) { }
345 }
346 function stopRingbackTone() {
347     try { ringbacktone.pause(); }
348     catch (e) { }
349 }
```

Листинг А.5. Файл обработки событий графического интерфейса softPhonePopupUI.js

```
1  var is_show_popup = false;
2  var is_load_popup = false;
3
4  (function(){
5  //popup part
6      var popup = document.getElementById("popup");
7      var popup_bar = document.getElementById("popup_bar");
8      var btn_close = document.getElementById("btn_close");
9
10     var offset = {x: 0, y: 0};
11
12     popup_bar.addEventListener('mousedown', mouseDown, false);
13     window.addEventListener('mouseup', mouseUp, false);
14
15     function mouseUp()
16     {
17         window.removeEventListener('mousemove', popupMove, true);
18     }
19
20     function mouseDown(e) {
21         offset.x = e.clientX - popup.offsetLeft;
22         offset.y = e.clientY - popup.offsetTop;
23         window.addEventListener('mousemove', popupMove, true);
```

```
24      }
25
26      function popupMove(e) {
27          popup.style.position = 'fixed';
28          var top = e.clientY - offset.y;
29          var left = e.clientX - offset.x;
30          popup.style.top = top + 'px';
31          popup.style.left = left + 'px';
32      }
33
34      window.onkeydown = function (e) {
35          if (e.keyCode == 27) { // if ESC key pressed
36              btn_close.click(e);
37          }
38      }
39
40      btn_close.onclick = function (e) {
41          popup.style.display = "none";
42          is_show_popup = false;
43      }
44
45  }());
46
47  function show_hide_popup(){
48      if (is_show_popup === false){
49          if (is_load_popup === false){
50              popup.style.top = Math.round($(window).height() * 30 /
                    100) + "px";
51              popup.style.left = Math.round($(window).width() * 70 /
                    100) + "px";
52              popup.style.width = "230px";
53              popup.style.height = "320px";
54              is_load_popup = true;
55    }
56          popup.style.display = "block";
57          is_show_popup = true;
58      } else {
59          popup.style.display = "none";
60          is_show_popup = false;
```

```
61      }
62  }
```

Листинг А.6. Файл подключения модуля к SalesPlatform vtiger CRM 6.4 softPhone_vtiger.js, клиентская часть

```
1   function getSipAccount(){
2       $.ajax({
3           url: "softPhone_vtiger.php",
4           dataType: 'json',
5           cache: false,
6           success: function(json){
7               server_ip = json.server_ip;
8               server_port = json.server_port;
9               sip_number = json.sip_number;
10              sip_password = json.sip_password;
11              console.log("server info: " + server_ip + ":" +
                    server_port + " // " + sip_number + " " +
                    sip_password);
12          }
13      });
14  }
```

Листинг А.7. Файл подключения модуля к SalesPlatform vtiger CRM 6.4 softPhone_vtiger.php, серверная часть

```
1   <?php
2
3   $ip = '192.168.0.105';
4   $port = '8088';
5
6   $sip_number = '103';
7   $sip_password = '103pas';
8
9   $sip_settings = array ('server_ip'=>$ip, 'server_port'=>$port, '
        sip_number'=>$sip_number, 'sip_password'=>$sip_password);
10  $data = json_encode($sip_settings);
11  echo $data;
```

```
12 error_log("here");
13 exit();
14
15
16 ?>
```