

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-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

Эта страница специально оставлена пустой.

РЕФЕРАТ

Отчет, 56 стр., 11 рис., 1 табл., 18 ист., 1 прил.

СОФТ-ФОН, VOIP-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-браузера и встроен в CRM-систему.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	9
1. ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ РЕАЛИЗАЦИИ SIP-ТЕЛЕФОНИИ ДЛЯ WEB-БРАУЗЕРА	10
1.1. Подход к реализации телефонии для web-браузера на Java	10
1.2. Подход к реализации телефонии для web-браузера на Flash	12
1.3. Подход к реализации телефонии для web-браузера на WebRTC	13
1.3.1. Краткая историческая справка о возможностях браузера передавать информацию	14
1.3.2. Обзор технологии WebRTC	14
1.4. Обзор существующих реализаций IP-телефонии в web-браузере	17
1.4.1. IP-телефония в amoCRM	17
1.4.2. IP-телефония в Битрикс24	18
1.5. Резюме	18
2. ПОСТАНОВКА ЗАДАЧИ, ВЫБОР СПОСОБА РЕШЕНИЯ	19
2.1. Постановка задачи	19
2.2. Выбор способа решения	19
3. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ МОДУЛЯ	22

4. РАЗРАБОТКА	24
4.1. Разработка подмодулей, не зависящих от web-приложения	24
4.2. Разработка подмодулей, зависящих от web-приложения, на примере SalesPlatform vtiger CRM 6.4	27
5. ТЕСТИРОВАНИЕ, АНАЛИЗ ПОЛУЧЕННЫХ РЕ- ЗУЛЬТАТОВ	29
5.1. Тестовое окружение	29
5.2. Функциональное тестирование	30
ЗАКЛЮЧЕНИЕ	36
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	37
ПРИЛОЖЕНИЕ А. ЛИСТИНГИ	40

СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

AEC	Acoustic Echo Cancellation, эхоподавление
AGC	Automatic Gain Control, автоматическая регулировка усиления
AJAX	Asynchronous Javascript And Xml, Асинхронный JavaScript и Xml
AJB	Adaptive Jitter Buffer, буфер выравнивания задержек передачи
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
ICE	Interactive Connectivity Establishment, Установление интерактивного соединения
JRE	Java Runtime Environment
RTP	Real-time Transport Protocol, безопасный протокол передачи данных в реальном времени
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
WebRTC	Real-Time Communications, коммуникации в реальном времени

ВВЕДЕНИЕ

Определенной вехой в истории телекоммуникаций является IP-телефония (Voice over IP, VoIP), позволившая передавать "голос" по IP-сетям. По сравнению с аналоговой телефонией VoIP позволяет сократить расходы на разговоры, обеспечивает мобильность пользователей, а также позволяет передавать видео. В связи с этим IP-телефония получила большое распространение.

Многие бизнес-системы для звонков используют VoIP. Разработчикам бизнес-систем просто записывать информацию о таких звонках: имена собеседников, время начала и завершения звонка, длительность, а также сам звонок в аудио-файле. Такая информация полезна менеджерам, которые могут анализировать работу операторов.

Бизнес-система обычно является web-приложением, напоминающим "социальную сеть компании". В таких случаях пользовательский интерфейс бизнес-системы в полном объеме доступен только через web-браузер. Однако организовать VoIP-телефонию в браузере является непростой задачей.

Большинство бизнес-систем использует виртуальные серверы VoIP-телефонии. Тогда браузерные VoIP-телефоны реализованы под конкретный сервер VoIP-телефонии и приходится платить за виртуальный сервер.

Было бы гораздо удобнее, если был модуль VoIP-телефонии для web-браузера, который можно подключить к любой бизнес-системе.

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 поддержка захвата аудио с микрофона. Среда выполнения для Java (Java Runtime Environment, JRE) широко распространена и обычно уже установлена в Windows и Linux системах.[1] Java код выполняется на JRE установленной на компьютере или в расширении браузера, захватывает аудио с микрофона и отправляет его на сервер по протоколу передачи данных в реальном времени (Real-time Transport Protocol, 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: эхоподавление (Acoustic Echo Cancellation, AEC), автоматическая регулировка усиления (Automatic Gain Control, AGC), буфер выравнивания задержек передачи (Adaptive Jitter Buffer, AJB) и подавление шума (Noise suppression).

АЕС 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. АJB 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, АЕС) 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 процессора клиента.

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 алгоритмы обработки 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

Для обмена данными в этом подходе необходим Adobe Flash Player и Adobe Media Server. Для передачи голоса используется протокол RTMFP, работающий по UDP. Имеется поддержка распространённых кодеков G.711 и H.264.

Flash Player 11 (последняя версия на сегодня) поддерживают VoIP алгоритмы: АЕС, АJB, симметричный алгоритм блочного шифрования AES. Шифрование AES защищает трафик между браузером и сервером от посторонних. Передачу данных осуществляется по протоколу RTMFP, работающего по UDP.[2]

У подхода к реализации VoIP-телефонии на Flash есть следующие преимущества:

1. Поддерживается большинством браузеров
2. Привычная технология для разработчиков – Action Script 3
3. Качественная передача аудио и видео

Недостатки данного подхода:

1. Требуется промежуточный сервер (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

У современных браузеров имеется множество возможностей, предоставляющих огромное количество инструментов для разработки. Прежде чем переходить к технологии Web-коммуникаций в реальном времени (Web Real-Time Communications, WebRTC) необходимо рассмотреть какие есть инструменты разработки у браузера для сетевой передачи данных.

1.3.1. Краткая историческая справка о возможностях браузера передавать информацию

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 каждый раз при получении новых данных от сервера. Эта задержка возникает из-за необходимости установления HTTP-соединения. Это создавало проблемы для создания web-приложений реального времени.[3]

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.[4] 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.[5]

1.3.2. Обзор технологии WebRTC

WebRTC - проект с открытым исходным кодом, предназначенный для организации передачи потоковых данных между браузером или другими поддерживающими его приложениями по технологии точка-точка.[6]

Технология WebRTC имеет продуманную архитектуру, избавленную от ошибок и недостатков, выявленных в плагинах браузера, которые существовали до неё. Технологические возможности WebRTC: безопасный протокол передачи данных в реальном времени (Secure Real-time Transport Protocol, SRTP), протокол датаграмм безопасности транспортного уровня (Datagram Transport Layer Security, DTLS), установление интерактивного соединения (Interactive Connectivity Establishment, ICE), утилиты прохождения сессий для NAT (Session Traversal Utilities for NAT, STUN), AEC, AGC, AJB, аудиокодек Opus, видеокодек VP8.

Набор используемых в WebRTC технологий очень большой. SRTP и DTLS обеспечивают защиту трафика между WebRTC узлами. ICE и STUN помогают преодолеть NAT.[7] AEC, AGC и AJB работают для того чтобы сделать аудио и видео качественным – без лагов и задержек. Кодеки Opus и VP8 хорошо подходят для глобального Интернета, где скорость соединения может резко изменяться.

Также нужно отметить, что подходы к реализации VoIP-телефонии в браузере, рассмотренные ранее (Java и Flash) требуют дополнительной установки ПО. WebRTC – это единственная технология, которая является встроенной в браузер. То есть в самом браузере имеются компоненты WebRTC. С помощью JavaScript к этим элементам можно обращаться прямо со страницы. Сегодня уже достаточно большое количество web-браузеров поддерживают WebRTC, подробная информация приведена в таблице 1.1.[8]

Преимущества технологии WebRTC:

1. Все алгоритмы обработки звука VoIP телефонии

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

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[9]

2. Поддерживается не всеми браузерами

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

1.4. Обзор существующих реализаций

IP-телефонии в web-браузере

Рассмотрим существующие на сегодняшний день реализации IP-телефонии в web-браузере. В частности рассмотрим самые рейтинговые CRM-системы.[10]

1.4.1. IP-телефония в amoCRM

В amoCRM имеется виджет UIS, который предоставляет доступ к облачному серверу IP-телефонии ATC UIS. Виджет работает по технологии WebRTC, и в связке с сервером ATC UIS реализованы следующие возможности[11]:

- запись звонков
- коллтрекинг
- сценарии обработки вызовов
- голосовое меню и почта
- виртуальный факс

Так же имеется виджет OnlinePBX, который использует виртуальную АТС OnlinePBX. Помимо этих двух виджетов имеется бесплатный виджет Asterisk. Все эти виджеты по функциональности почти не отличаются.

1.4.2. IP-телефония в Битрикс24

Телефония в Битрикс24 реализована с помощью технологии WebRTC. Телефонию эту можно использовать бесплатно, однако при бесплатном использовании имеется ограничение на число возможных абонентов а также нет возможности записи разговора. Коллтрекинг отсутствует, но зато имеется возможность видео звонков.[12]

1.5. Резюме

В данном разделе был проведён анализ существующих подходов для реализации VoIP-телефонии в браузере. На сегодняшний день лучшим является подход, использующий в качестве инструмента технологию WebRTC. В нём реализованы необходимые для VoIP-телефонии алгоритмы, а так же не требуется установка дополнительного программного обеспечения. Так же были рассмотрены существующие реализации VoIP-телефонии.

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.[13][14]

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 имеются 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-

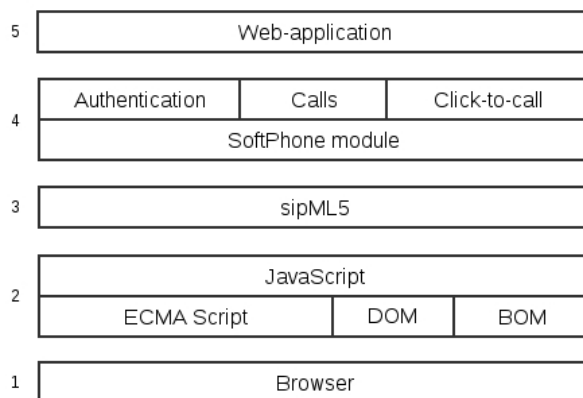


Рисунок 2.1. Структура взаимодействия модулей

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.

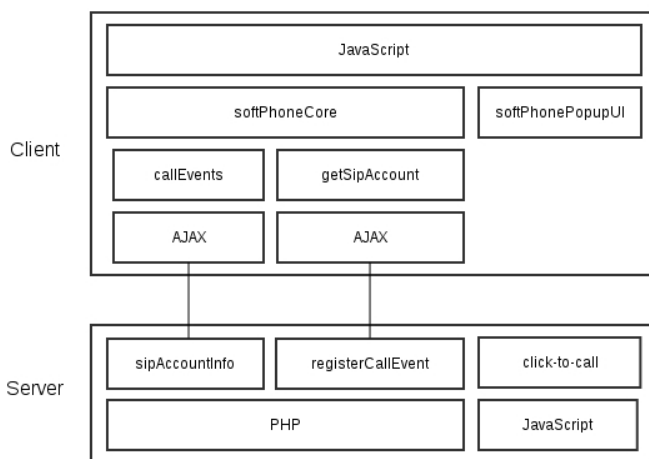


Рисунок 3.1. Архитектура модуля

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 подключаться модуль

Основные функции подмодуля `softPhoneCore.js`:

- `createSipStack()` - иницирует 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 сделано 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.

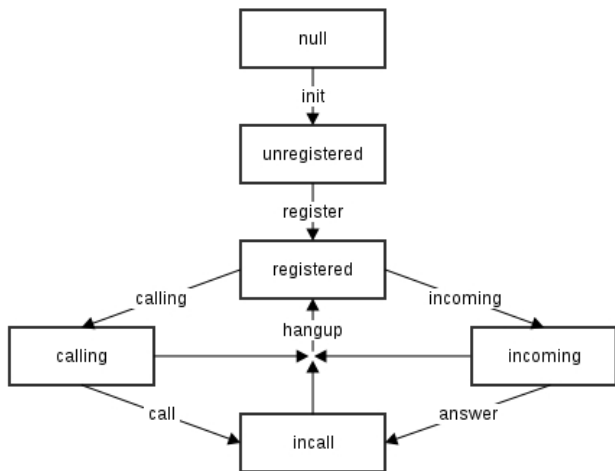


Рисунок 4.1. Конечный автомат, управляющий состоянием SIP-клиента

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 создается 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.[15] 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.[16] В этом случае скрипт будет выполняться при каждой загрузке страницы.

Тестирование проводилось на браузерах Icedweasel 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[17] 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`. [18]

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.

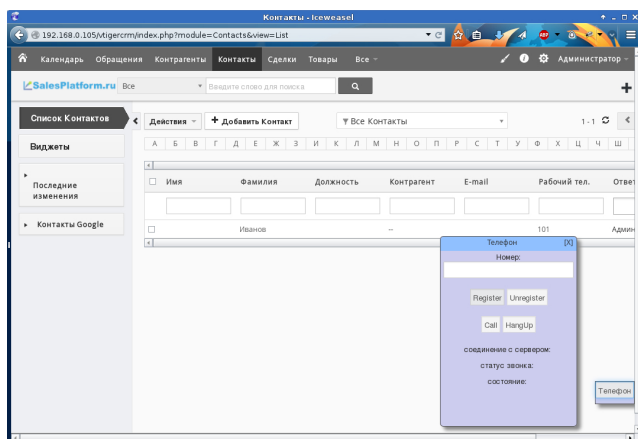


Рисунок 5.1. Модуль в CRM-системе SalesPlatform vtiger CRM 6.4, начальное состояние модуля

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

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

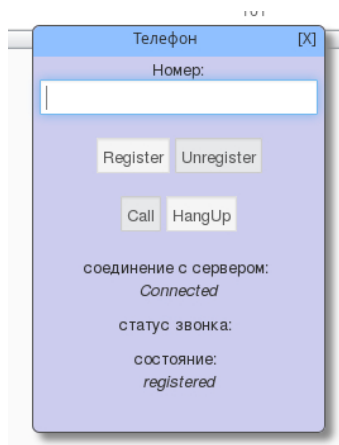


Рисунок 5.2. Состояние после аутентификации на сервере

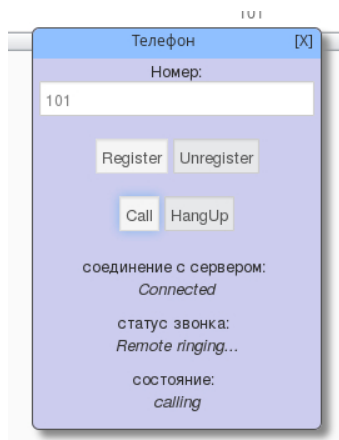


Рисунок 5.3. Состояние исходящего звонка (воспроизводятся гудки)

фического интерфейса.

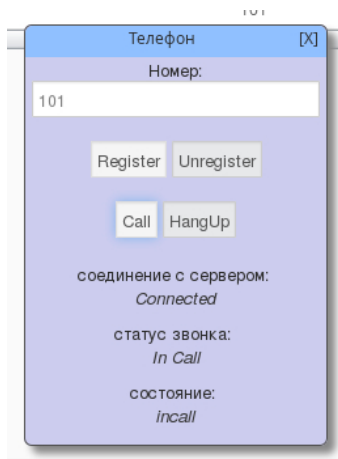


Рисунок 5.4. Состояние в процессе звонка

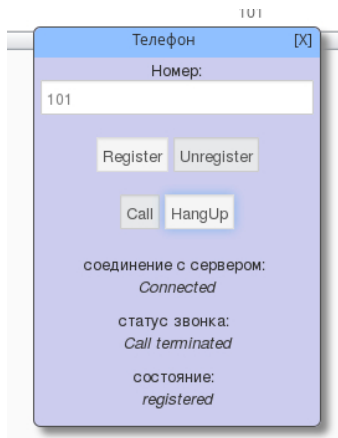


Рисунок 5.5. Состояние завершения звонка, такое же как и состояние после аутентификации на сервере

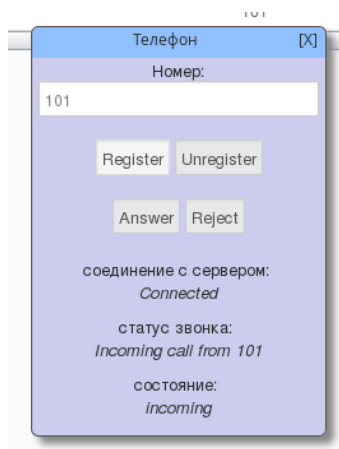


Рисунок 5.6. Состояние входящего звонка (воспроизводится рингтон)

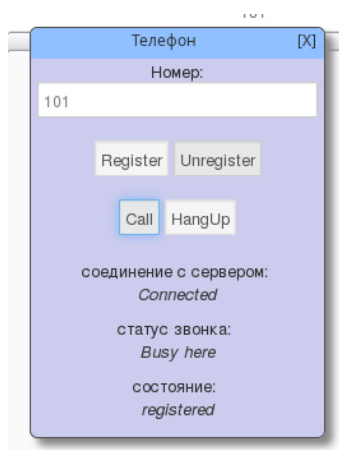


Рисунок 5.7. Вызываемый абонент занят

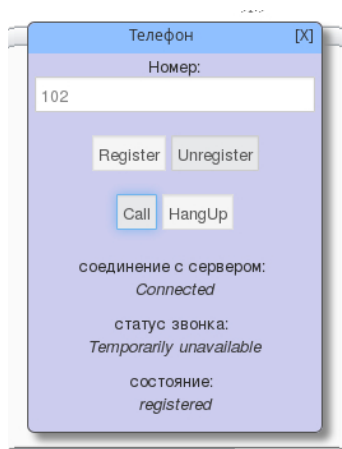


Рисунок 5.8. Вызываемый абонент недоступен

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 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. 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).
2. Flash Media ServerElectronic resource, Adobe Systems Incorporated. — URL: <http://www.adobe.com/products/adobe-media-server-family.html> (online; accessed: 20.06.2016).
3. Introducing WebSockets: Bringing Sockets to the WebElectronic resource, HTML5 Rocks. — URL: <http://www.html5rocks.com/en/tutorials/websockets/basics/> (online; accessed: 19.06.2016).
4. 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).
5. Browser - WebRTC on iOSElectronic resource, Ericsson Research. — URL: <http://www.openwebrtc.org/browser/> (online; accessed: 19.06.2016).
6. WebRTCElectronic resource, WebRTC. — URL: <https://sites.google.com/site/webrtc/home> (online; accessed: 16.06.2016).
7. Всё, что вы хотели знать о протоколе SIP [Электронный ресурс], Системный администратор. — URL: <http://samag.ru/archive/article/2017> (дата обращения: 16.06.2016).

8. Can i use... Support tables for HTML5, CSS3, etcElectronic resource, Can i use. — URL: <http://caniuse.com/#feat=websockets> (online; accessed: 19.06.2016).
9. 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).
10. Рейтинг CRM-систем (управление отношениями с клиентами) 2016 [Электронный ресурс], Тэглайн. — URL: <http://tagline.ru/crm-systems-rating/> (дата обращения: 20.06.2016).
11. Онлайн CRM система - amoCRM [Электронный ресурс], amoCRM. — URL: <http://www.amocrm.ru/> (дата обращения: 20.06.2016).
12. amoCRM. — Телефония в Битрикс24. — URL: https://dev.1c-bitrix.ru/learning/course/?COURSE_ID=77&CHAPTER_ID=02564&LESSON_PATH=6907.2564 (дата обращения: 20.06.2016).
13. World's first HTML5 SIP clientElectronic resource, Doubango Telecom. — URL: <https://www.doubango.org/sipml5/> (online; accessed: 19.06.2016).
14. the JavaScript SIP library JsSIPElectronic resource, JsSIP. — URL: <http://jssip.net/> (online; accessed: 19.06.2016).
15. Digium, Inc. — Asterisk WebRTC Support. — URL: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+WebRTC+Support> (online; accessed: 19.06.2016).
16. vtigerCRM Data Model. — URL: <https://www.vtiger.com/>

- products/crm/docs/510/vtigerCRM_DataModel_5.2.1.pdf (online; accessed: 19.06.2016).
17. sipML5 live demoElectronic resource, Doubango Telecom.— URL: <https://www.doubango.org/sipml5/call.htm?svn=250> (online; accessed: 19.06.2016).
 18. 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

```
1  var server_ip;
2  var server_port;
3  var sip_number;
4  var sip_password;
5
6  $.ajax({
7      url: "softPhone.html", cache: false, success: function(html){
8          $( "body" ).append( html );
9
10         getSipAccount();
11     }
12 });
```

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

```
1      <script src="SIPml-api.js?svn=230" type="text/javascript" defer
2          ></script>
3
4      <script src="softPhone_vtiger.js" type="text/javascript" defer
5          ></script>
6
7      <script src="softPhoneCore.js" type="text/javascript" defer></
8          script>
9
10     <script src="softPhonePopupUI.js" type="text/javascript" defer
11         ></script>
12
13     <link rel="stylesheet" type="text/css" href="softPhone.css">
14
15     <div id="popup">
16         <div id="popup_bar">
17             Телефон
18             <span id="btn_close"> [X] </span>
19         </div>
20     </div>
```

```

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

```

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

```
1  #popup{
2      display:none;
3      background-color:#cccccc;
4      position:fixed;
5      box-shadow: 6px 6px 5px #888888;
6      border-radius:6px;
7      border:1px solid #4f4f4f;
8      text-align:center;
9      z-index: 1;
10 }
11
12 #popup_bar{
13     width:100%;
14     background-color:#90c0ff;
15     position:relative;
16     border-radius:6px 6px 0 0;
17     text-align:center;
18     height:24px;
19     cursor:move;
20 }
21
22 #btn_close{
23     float:right;
24     cursor:pointer;
25     padding-right:6px;
26 }
27
28 #txtRegStatus{
29     font-style: italic;
30 }
31
32 #txtCallStatus{
33     font-style: italic;
34 }
35
36 #txtState{
37     font-style: italic;
```

```

38 }
39
40 #aside_popup_phone {
41     display: block;
42     background-color: #cccccc;
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:
77                         onSipEventSession } // optional: '*' means all
78                         events
79                     });
80                 registerSession.register();
81                 break;
82             }
83             case 'i_new_call': // incoming audio/video call
84             {
85                 if (callSession) {
86                     // do not accept the incoming call if we're already
87                     'in call'
88                     e.newSession.hangup(); // comment this line for
89                     multi-line support
90                 }
91                 else {
92                     callSession = e.newSession;
93                     // start listening for events
94                     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         ),
196         events_listener: { events: '*', listener:
197             onSipEventSession } // optional: '*' means all
198             events
199     });
200     callSession.call(document.getElementById("phone_popup_number
201     ").value.toString());
202 }
203 else {
204     txtCallStatus.innerHTML = txtCallStatus.value = 'Connecting
205     ...';
206     console.log("call = " + txtCallStatus.value);
207     callSession.accept();
208     finalState("incall");
209 }
210 console.log("call 2");
211 }
212 // terminates the call (SIP BYE or CANCEL)
213 function hangup() {
214     if (callSession) {
215         txtCallStatus.innerHTML = txtCallStatus.value = 'Terminating
216         the call...';
217         console.log("call = " + txtCallStatus.value);
218         callSession.hangup({events_listener: { events: '*', listener
219             : onSipEventSession }});
220     }
221 }
222
223 function finalState(action){
224     switch (state)
225     {
226         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 A.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($(window).height() * 30 /
51                 100) + "px";
52             popup.style.left = Math.round($(window).width() * 70 /
53                 100) + "px";
54             popup.style.width = "230px";
55             popup.style.height = "320px";
56             is_load_popup = true;
57         }
58         popup.style.display = "block";
59         is_show_popup = true;
60     } else {
61         popup.style.display = "none";
62         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 + ":" +
12                         server_port + " // " + sip_number + " " +
13                         sip_password);
14         }
15     });
16 }

```

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'=>$port, '
10     sip_number'=>$sip_number, 'sip_password'=>$sip_password);
11 $data = json_encode($sip_settings);
12 echo $data;

```

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