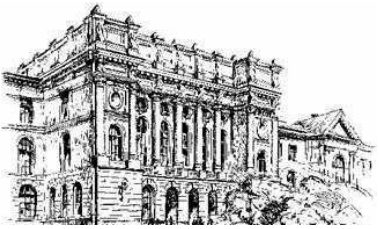
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий



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: 09.03.01 – 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, 65 стр., 12 рис., 2 табл., 19 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

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 являются web-приложением, напоминающем "социальную сеть компании". В таких случаях пользовательский интерфейс бизнес-системы в полном объёме доступен только через web-браузер. Поэтому телефония из браузера стала неотъемлемой частью бизнес-систем. 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 поддерживать 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).

AEC 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 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 поддержку распространённых кодеков G.711 и H.264.

Flash Player 11 (последняя версия на сегодня) поддерживают VoIP алгоритмы: AEC, AJB, 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. 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

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 браузерами 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 протоколу 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 компонентам 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 устранятся.

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 Битрикс24 и amoCRM.[10] А также рассмотрим одну из наиболее распространённых CRM-систем с открытым исходным кодом vtiger CRM.

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. Функция IP-телефонии работает только в браузерах Google Chrome и Firefox.[12]

1.4.3. IP-телефония в vtiger CRM

Телефония в vtiger CRM реализована следующим образом. Все контакты доступны на сайте, однако чтобы позвонить необходимо иметь сторонний софт-фон или же аппаратный телефон. Взаимодействие такого стороннего телефона с CRM-системой выполняет модуль Connector. Он принимает с сервера VoIP-телефонии информацию о звонке и отправляет её в CRM-систему, в которой уже создаётся карточка звонка.

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.

Были рассмотренны и существующие реализации VoIP-телефонии в современных CRM-системах. Видно, что телефония из браузера имеется ещё не во всех CRM-системах, а в некоторых, в которых имеется, работает не во всех браузерах.

2. ПОСТАНОВКА ЗАДАЧИ, ВЫБОР СПОСОБА РЕШЕНИЯ

В современном мире коммуникации через интернет очень удобны. Такие коммуникации особенно будут экономить время, если их встроить в бизнес-системы. Но для бизнес-систем это не столь необходимо, как для CRM-систем.

В CRM-системах звонки занимают одну из центральных ролей для успешного бизнеса. У каждой CRM-системы имеется модуль Звонки, который для каждого звонка создаёт карточку. В ней можно записывать информацию о звонке, определить его значение (сделка, покупка, продажа и т.п.). Такие модули Звонков могут хранить историю о звонках, на основе которой будет составляться статистика, которая полезна управляющему персоналу компании. Для них важно знать сколько сделок совершили операторы, кто из них лучший. Функция записи звонков позволит выявлять плохих и хороших работников.

Из раздела 1.4 понятно, что в бизнес-системах и CRM-системах есть пробелы в удобстве организации VoIP-телефонии в браузере. Поэтому постараемся заполнить эти пробелы разработкой модуля VoIP-телефонии для web-браузера.

2.1. Постановка задачи

Итак, необходимо разработать модуль VoIP-телефонии для web-браузера. Данный модуль будет выполнять функцию телефона (софтфона), который будет доступен прямо со страницы web-приложения.

Так же данный модуль будет решением с отрытым исходным кодом. Это позволит тем кто его использует настраивать его под свои задачи.

Установим следующие требования к модулю:

1. Поддержка современными web-браузерами: Firefox, Google Chrome, Safary, Edge, IE, Opera, Android Browser, Chrome for Android, Bowser.

2. Модуль должен быть универсальным, и легко встраиваться в бизнес-системы и CRM-системы.

3. Модуль должен полностью контролировать состояние звонка.

Обычно софт-фоны поддерживают передачу видео, аудио и мгновенных сообщений. Однако CRM-системах видео-звонки практически не требуется. Поэтому в данной работе ограничимся передачей аудио потока.

Графический интерфейс должен состоять из скользящей кнопки и плавающего окна, которое появляется и скрывается по нажатию на эту кнопку. Плавающее окно должно включать поле для набора номера и кнопки "вызов"и "сброс которые должны меняются во время входящего звонка на кнопки "снять трубку"и "отклонить звонок"соответственно. То есть графический интерфейс должен реагировать на состояние звонка. Приведём ещё один пример. Графический интерфейс должен запрещать пользователю нажимать на кнопку сброса пока звонок ещё не начался.

2.2. Выбор способа решения

Как было рассмотренно в разделе 1.3.2 лучшей на сегодня технологией для организации VoIP-телефона в браузере является WebRTC. Данная технология встроенна в браузер и доступ к её компонентам имеется только с помощью JavaScript, который выполняется на web-странице. Поэтому разработка клиентской части будет реализованна на JavaScript.

Для обмена в VoIP-телефонии существуют два протокола H323 и SIP. По технологии WebRTC имеются только две реализации SIP протокола — это библиотеки sipML5 и JsSIP. Реализации протокола H323 на сегодняшний день нет, поэтому этот протокол мы рассматривать не будем.

2.2.1. Сравнительный анализ библиотек sipML5 и JsSIP

Функциональные возможности библиотек sipML5 и JsSIP представленны в таблице 2.1.[13][14]

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

Обе библиотеки имеют свои преимущества, но для нашей задачи выберем библиотеку sipML5, потому что функциональных возможностей у неё больше и не нужно дополнительно устанавливать Node.js на сервер.

Таблица 2.1. Функциональные возможности библиотек sipML5 и JsSIP

Функциональная возможность	**sipML5**	**JsSIP**
аудио звонки	+	+
видео звонки	+	+
мгновенные сообщения	+	+
статус присутствия в сети	+	+
удержание вызова	+	+
отключение микрофона	+	+
тональный набор (DTMF)	+	+
трансляция экрана	только для Chrome	-
групповой звонок	+	-
перенаправление вызова	+	-

2.3. Резюме

В данном разделе была поставлена задача разработки модуля SIP-телефонии для web-браузера, определены основные требования. Для разработки была выбрана технология WebRTC и библиотека sipML5 для осуществления звонков по протоколу SIP.

3. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ МОДУЛЯ SIP-ТЕЛЕФОНИИ

Перед разработкой, необходимо обдумать на каком уровне находится разрабатываемый модуль SIP-телефонии по отношению к элементам браузера и web-приложению. Так же необходимо определить основные части модуля и его архитектуру.

3.1. Иерархия web-приложения и модуля SIP-телефонии

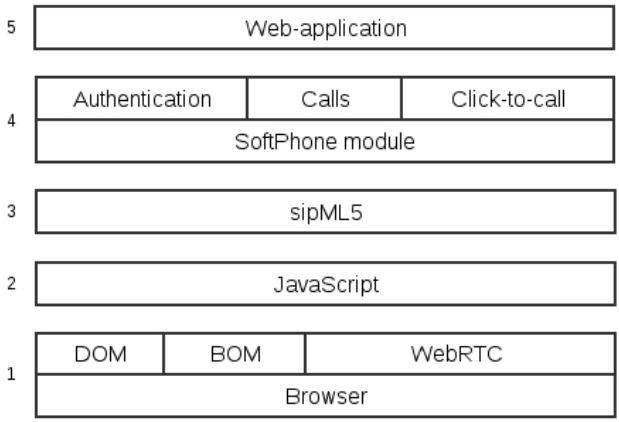


Рисунок 3.1. Иерархия web-приложения, отражающая работу разрабатываемого модуля SIP-телефонии

На рисунке 3.1 изображена иерархия web-приложения, отражающая работу разрабатываемого модуля SIP-телефонии в контексте

взаимодействия с браузером. Поясним назначение каждого уровня:

1. На этом уровне расположены следующие компоненты браузера: объектная модель документа (Document Object Module, DOM), объектная модель браузера (Browser Object Module, BOM) и WebRTC. Объектная модель документа может хранить такие элементы как аудио, которые позволят нашему модулю воспроизводить рингтон и звук гудков. WebRTC предоставляет возможность захватывать аудио с микрофона, кодировать его, и передавать при помощи WebSockets. При приёме аудио WebRTC предоставляет возможность фильтровать его алгоритмами VoIP-телефонии.

2. Данный уровень представляют функции JavaScript, которые реализуют доступ web-страницы к элементам объектной модели документа и WebRTC.

3. На этом уровне располагается выбранная нами библиотека sipML5. Она обеспечивает обмен аудио-данными по протоколу SIP. То есть анализирует пакеты приходящие от WebSockets и если они являются SIP-пакетами, то поступают на обработку.

4. Этот уровень является основной частью разработки. Здесь должны быть реализованы функции, которые легко использовать для звонка. Эти функции будут вызываться SIP-событиями, или по инициативе пользователя, и будут являться обёртками для библиотеки sipML5.

   Здесь же необходимо реализовать интерфейс взаимодействия с

web-приложением (бизнес-системой или CRM-системой). Основными функциями этого интерфейса будут являться: получение данных о SIP-аккаунте (Authentication), оповещения о звонке (Calls), и функция click-to-call, которая при нажатии на номер телефона в web-приложении набирает его.

5. На верхнем модуле располагается web-приложение, которое будет подключать разрабатываемый нами модуль как файл на JavaScript.

3.2. Архитектура модуля SIP-телефонии

На рисунке 3.2 изображена архитектура модуля. Опишем каждую его часть.

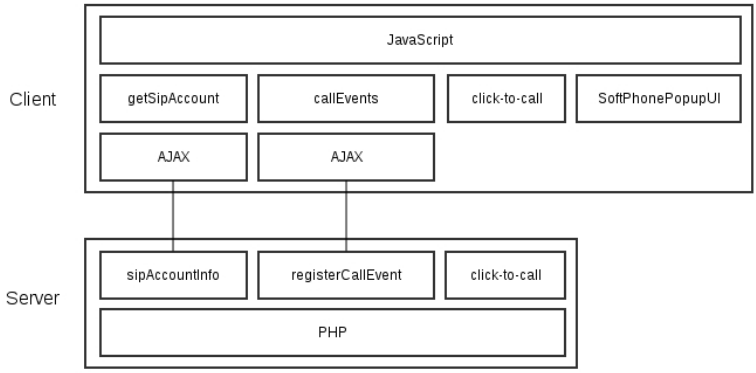


Рисунок 3.2. Архитектура модуля

Основой модуля является его ядро softPhoneJS. Оно обеспечивает взаимодействие с библиотекой sipML5 и отслеживает состояние звон-

ка.

При переключении состояния звонка информация передаётся в модуль CallEvents, который отправляет информацию на сервер, для регистрации звонка. Для передачи данных в этом случае проще всего использовать AJAX.

Однако перед тем как звонить, необходимо получить из CRM-системы данные о SIP-аккаунте. За это отвечает модуль getSipAccount. Данные так же передаются при помощи AJAX.

SoftPhonePopupUI отвечает за графический интерфейс скользящей кнопки и плавающего окна. Должны быть реализованы функции, которые обрабатывают перетаскивание окна мышью.

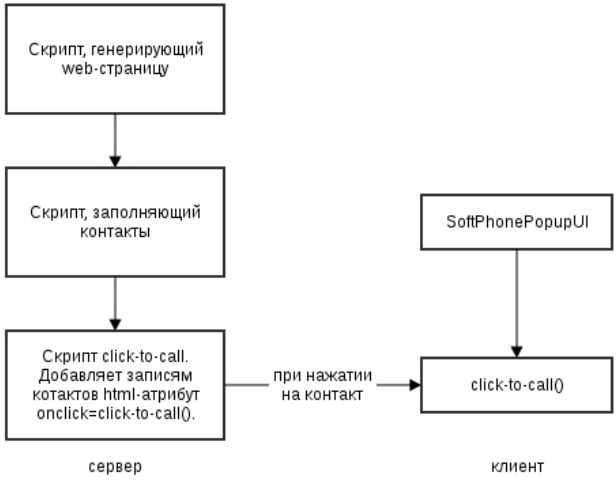


Рисунок 3.3. Реализация click-to-call

Click-to-call сам набирает номер при нажатии на контакт. Однако в этом случае обмен происходит не через AJAX. Реализация функции

click-to-call изображена на рисунке 3.3.

Скрипт, генерирующий web-страницу заполняет её контактами, но перед этим добавляет каждому контакту html-атрибут, который вызывает функцию обработчика при клике по контакту. Обработчиком задаётся функция click-to-call которая реализована на клиенте в модуле SoftPhonePopupUI.

3.3. Резюме

В данном разделе была рассмотренна иерархия web-приложения, определено какое место занимает модуль SIP-телефонии, встроенный в него. Выделены основные части модуля SIP-телефонии, разработана его архитектура, выбраны способы взаимодействия с сервером.

4. РАЗРАБОТКА МОДУЛЯ SIP-ТЕЛЕФОНИИ

В данном разделе описывается конфигурация окружения разработчика: сервер VoIP-телефонии Asterisk, CRM-система vtiger CRM. Приводится описание разработанных компонентов модуля SIP-телефонии для web-браузера.

4.1. Конфигурация окружения разработчика

В качестве первой CRM-системы в которую встроим модуль SIP-телефонии для браузера была выбрана SalesPlatform vtiger CRM, у которой исходный код находится в открытом доступе. В качестве сервера VoIP-телефонии был выбран сервер Asterisk, в связи с тем, что vtiger CRM использует именно его для организации звонков сторонними (не браузерными) приложениями.

Однако работа модуля SIP-телефонии не зависит от того какой сервер соединяет клиентов в сети. Для связи клиентов можно было выбрать любой из серверов который поддерживает WebRTC. Список серверов был приведён в разделе 1.3.2.

На сервер была установлена операционная система Debian GNU/Linux 8.4 (jessie). На эту машину был установлен сервер VoIP-телефонии Asterisk. На ней же установлена CRM-система vtiger CRM.

4.1.1. Настройка сервера VoIP-телефонии Asterisk

По началу был установлен Asterisk 11 - первая версия сервера, поддерживающая WebRTC.[15] При его установке для работы WebRTC необходимо выбрать драйверы chan_pjsip и chan_sip и модули res_srtp, res_crypto и res_http_websocket. Затем был установлен FreePBX 12. FreePBX - это доступная через web-интерфейс графическая оболочка конфигурационных файлов Asterisk.[16]

Через графический интерфейс FreePBX были созданы пользователи, им были присвоены номера, и включена возможность звонить с помощью WebRTC. Однако таким способом настроить работу сервера для работы с WebRTC не удалось. В качестве тестирования работы WebRTC была использована demo-версия браузерного SIP-телефона библиотеки sipML5.

Затем было предположенно, что для работы WebRTC, необходимо установить Asterisk последней версии 13.7.2. Так же был установлен FreePBX 13. Но с такой связкой тоже не удалось осуществить звонок с помощью WebRTC.

Тогда было решено не устанавливать FreePBX, и настроить Asterisk вручную, меняя конфигурационные файлы по инструкции в документации.[15] После такой настройки удалось осуществить звонок с помощью WebRTC.

В итоге 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. Также добавлены номера,

позволяющие звонить при помощи обычных сокетов по порту 5060.

При такой конфигурации сервера звонки могут осуществлятя между:

- двумя клиентами, использующими WebSockets
- одним клиентом, использующим WebSockets и другим клиентом, использующим обычные сокеты
- двумя клиентами, использующими обычные сокеты

4.1.2. Настройка vtiger CRM

Используемой CRM-системой является 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 SIP-телефонии были помещены в корень CRM-системы. У vtiger CRM конечно имеется инструкция добавления новых модулей в систему, но не будем её использовать. Вместо этого сосредоточимся на разработке модуля SIP-телефонии, а не на том, как правильно его подключать.

4.2. Разработка общих компонентов модуля SIP-телефонии

Здесь рассмотрим разработку компонентов модуля, которые будут общими для любой CRM-системы.

Итак, разрабатываемый модуль должен быть легко подключаем к web-приложению. То есть нужно чтобы его можно было подклю-

чить одним файлом JavaScript. 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 - библиотека sipML5.
- softPhoneCore.js - ядро модуля SIP-телефонии, отвечает за аутентификацию клиента на сервере SIP-телефонии, и осуществление звонков. Также в нём отслеживаются события звонка (CallEvents).
- softPhonePopupUI.js - компонент обработки плавающего окна при перетаскивании мышью.
- softPhone.css - разметка плавающего окна.
- softPhone_"web-app".js[1] - компонент, в котором размещается функция getSipAccount() для конкретного web-приложения.
- softPhone_"web-app".php - подмодуль, отправляющий sipAccountInfo клиенту.

Основные функции подмодуля softPhoneCore.js:

- createSipStack() - инициирует конфигурацию SIP-соединения, основываясь на информации полученной getSipAccount().
- register() - аутентификация SIP-клиента на сервере.

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

- 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.

Основные функции подмодуля softPhonePopupUI.js:

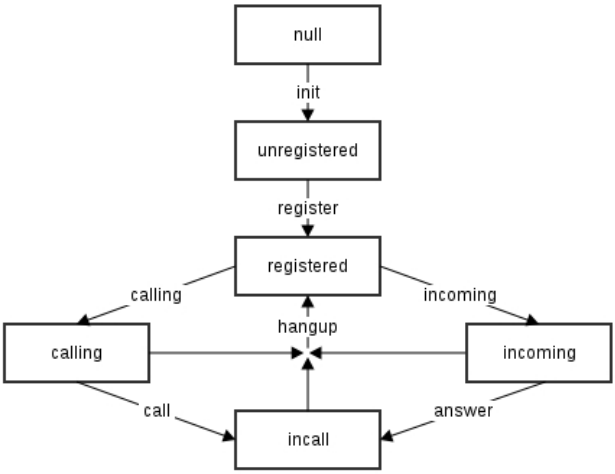


Рисунок 4.1. Конечный автомат, управляющий состоянием SIP-клиента

- show_hide_popup() - функция отображения и скрытия всплывающего окна.
- mouseDown(e), mouseUp() и popupMove(e) - функции перетаскивания всплывающего окна.

4.3. Встраивание модуля SIP-телефонии в CRM-систему

Как уже было выбранно в разделе 4.1 постараемся встроить наш модуль в CRM-систему SalesPlatform vtiger CRM 6.4.

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.[17] В этом случае скрипт будет выполнятся при каждой загрузке страницы.

Общие компоненты уже написаны. Теперь необходимо реализовать компоненты softPhone_vtiger.js и softPhone_vtiger.php конкретно для данной CRM-системы.

Компонент softPhone_vtiger.js реализует функции getSipAccount и callEvents, а компонент softPhone_vtiger.php реализует функции sipAccountInfo и registerCallEvent (см. рис. 3.2).

Функция getSipAccount() осуществляет AJAX-запрос данных о SIP-аккаунте (ip-адрес и порт сервера телефонии, SIP-номер и пароль для текущего пользователя CRM-системы). А компонент 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 компонент 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 трудоёмки для данной работы, 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 данной CRM-системе за управление карточками звонка отвечает модуль PBXManager. В его функцию click-to-call 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.

4.4. Резюме

Рассмотрена конфигурация окружения разработчика. Рассмотрена разработка общих компонентов модуля SIP-телефонии для web-браузера. Так же рассмотренно встраивание модуля в CRM-систему на примере SalesPlatform vtiger CRM 6.4.

5. ТЕСТИРОВАНИЕ, АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В данном разделе рассматривается тестовое окружение и ручное функциональное тестирование. Модульное тестирование не проводилось, потому что в коде в основном создавались обёртки над библиотекой sipML5, а эта библиотека протестирована создателями.

5.1. Тестовое окружение

Тестовое окружение серверной части было приведено в разделе 4.1.

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[18] работает на данных браузерах так же.

Из этого 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 протокола HTTPS.[19] Почему демо-версия sipML5 работает со сбоями в браузере Opera пока не ясно.

5.2. Функциональное тестирование

Автоматизированного функционального тестирования не проводилось из-за трудоёмкости задачи. У Asterisk имеются встроенные средства записывать звонки. При таком тестировании необходимо было бы реализовать тестовый модуль SIP-телефонии, который бы записывал во время звонка две дорожки - одну с микрофона, другую поступающую от собеседника. Помимо этого необходимо было бы создать тестовые сценарии, которые можно сделать например в Selenium IDE. После выполнения этих сценариев, необходимо отправить файлы записи звонка от Asterisk клиенту и сравнить их с файлами записанными тестовым модулем SIP-телефонии.

Поэтому функциональное тестирование проводилось вручную.

Критерии корректной работы модуля SIP-телефонии:

- 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 мо-

мент разговора кнопка звонка должна быть выключена и доступна только кнопка сброса.

- Корректная обработка звонков на несуществующие номера.

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. Так же отслеживался лог-звонка на сервере Asterisk. Полностью прошёл все тестовые сценарии только SIP-модуль, запущенный в браузере Firefox. В его логе звонка ошибок не было (см. листинг 5.1).

Листинг 5.1. Лог звонка на сервере Asterisk, 103 - реализованный SIP-модуль, 101 - SIP-клиент на мобильном устройстве CSipSimple

```
1     -- Registered SIP '101' at 192.168.0.104:59099
2        > Saved useragent "CSipSimple_marvel-10/r2457" for peer 101
3  [Jun 23 07:45:03] NOTICE[9307]: chan_sip.c:27708
       handle_request_subscribe: Received SIP subscribe for peer
       without mailbox: 101
4    == WebSocket connection from '192.168.0.105:58921' for protocol '
       sip' accepted using version '13'
5     -- Registered SIP '103' at 192.168.0.105:58921
6        > Saved useragent "IM-client/OMA1.0 sipML5-v1.0.0.0" for peer
            103
7  [Jun 23 07:46:17] NOTICE[9364]: chan_sip.c:23945
       handle_response_peerpoke: Peer '103' is now Reachable. (32ms /
       2000ms)
8    == Using SIP VIDEO CoS mark 6
9    == Using SIP RTP CoS mark 5
10    -- Executing [101@from-internal:1] NoOp("SIP/103-00000000", "
         webrtc test call") in new stack
11    -- Executing [101@from-internal:2] Dial("SIP/103-00000000", "SIP
         /101") in new stack
12   == Using SIP RTP CoS mark 5
```

```
13      -- Called SIP/101
14      -- SIP/101-00000001 is ringing
15         > 0x7f8f84005fd0 -- Probation passed - setting RTP source
               address to 192.168.0.104:4000
16      -- SIP/101-00000001 answered SIP/103-00000000
17      -- Channel SIP/101-00000001 joined 'simple_bridge' basic-bridge
            <d6898e4f-618e-4bed-afd2-48c4a170cb8e>
18      -- Channel SIP/103-00000000 joined 'simple_bridge' basic-bridge
            <d6898e4f-618e-4bed-afd2-48c4a170cb8e>
19         > 0x7f8f84005fd0 -- Probation passed - setting RTP source
               address to 192.168.0.104:4000
20         > 0x7f8f8800d740 -- Probation passed - setting RTP source
               address to 192.168.0.105:56165
21      -- Channel SIP/101-00000001 left 'simple_bridge' basic-bridge <
            d6898e4f-618e-4bed-afd2-48c4a170cb8e>
22      -- Channel SIP/103-00000000 left 'simple_bridge' basic-bridge <
            d6898e4f-618e-4bed-afd2-48c4a170cb8e>
23    == Spawn extension (from-internal, 101, 2) exited non-zero on 'SIP
         /103-00000000'
24      -- Unregistered SIP '103'
25    == WebSocket connection from '192.168.0.105:58921' closed
```

На рисунках 5.1 - 5.8 изображён корректный графический интерфейс модуля в разных состояниях. Описание состояния читайте в названиях рисунков.

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

5.3. Резюме

Таким образом проводилось тестирование. По ходу разработки, оно помогало выявить следующие функциональные ошибки: некорректное состояние сессии звонка, односторонняя передача голоса и некорректное состояние графического интерфейса.

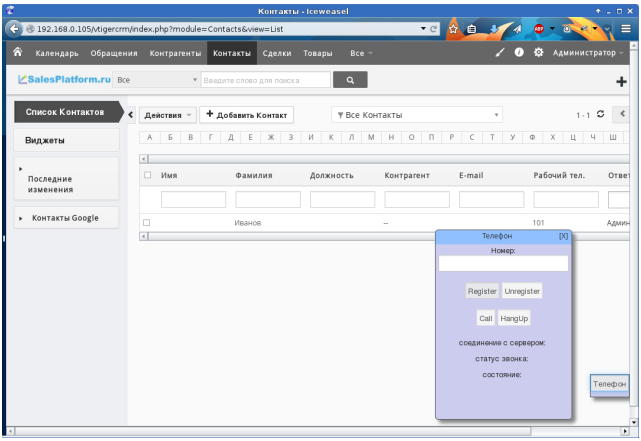


Рисунок 5.1. Модуль в CRM-системе SalesPlatform vtiger CRM 6.4, начальное состояние модуля

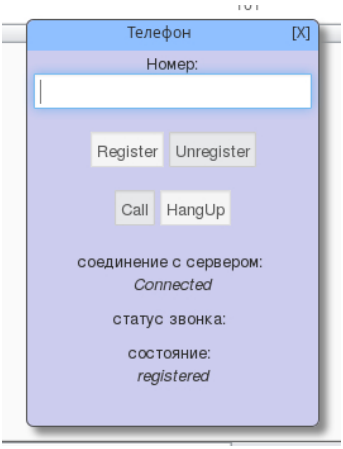


Рисунок 5.2. Состояние после аутентификации на сервере

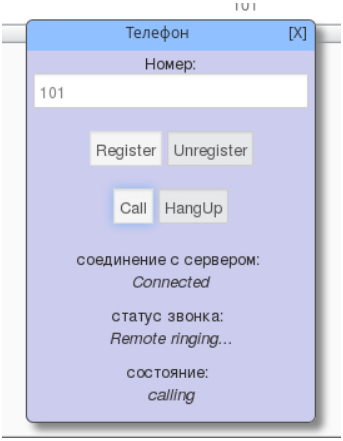


Рисунок 5.3. Состояние исходящего звонка (воспроизводятся гудки)

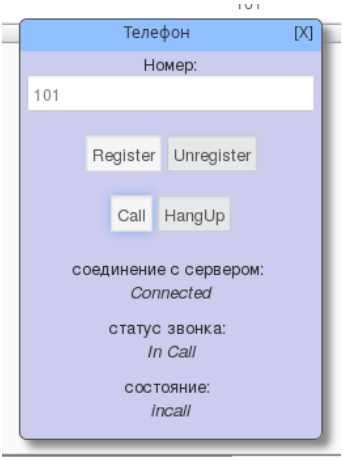


Рисунок 5.4. Состояние в процессе звонка

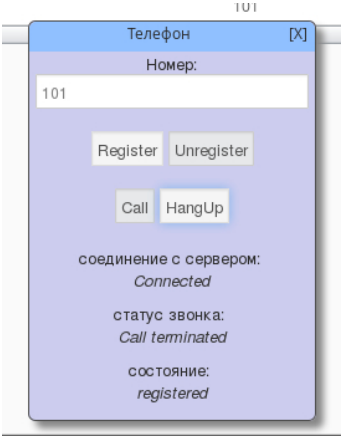


Рисунок 5.5. Состояние завершения звонка, такое же как и состояние после аутентификации на сервере

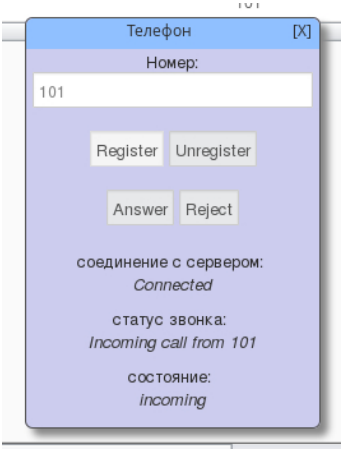


Рисунок 5.6. Состояние входящего звонка (воспроизводится рингтон)

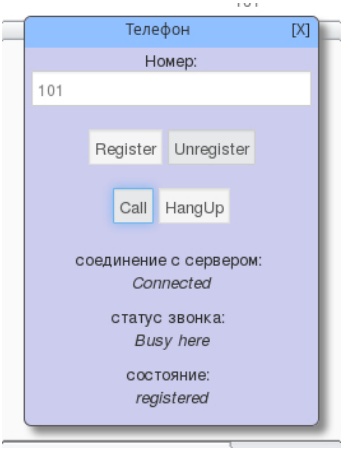


Рисунок 5.7. Вызываемый абонент занят

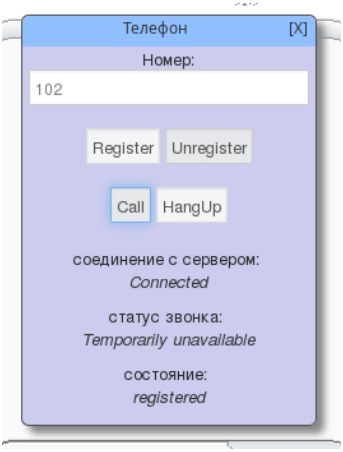


Рисунок 5.8. Вызываемый абонент недоступен

ЗАКЛЮЧЕНИЕ

Проведён анализ предметной области. Поставлена задача разработки модуля SIP-телефонии для web-браузера, решение которой заполняет пробелы VoIP-телефонии в бизнес-системах и CRM-системах. Для решения данной задачи выбрана современная технология WebRTC.

Основная часть модуля SIP-телефонии для web-браузера была реализована. Звонки тестировались из браузера Firefox на сконфигурированном для WebRTC сервере IP-телефонии Asterisk. Остальные браузеры звонили, но с некоторыми неудачами. Это скорее всего связанно с тем, что у технологии WebRTC отсутствует окончательный стандарт. RFC этой технологии находится в черновом варианте.

Также WebRTC гарантирует поддержку большинством браузеров, однако демо-версия библиотеки sipML5 в тестируемых нами браузерах ведёт себя с неудачами.

Большое количество времени было использовано на изучение библиотеки sipML5. Освоение её документации оказалось более сложной задачей, чем ожидалось. Поэтому на внедрение модуля SIP-телефонии в CRM-систему осталось меньше времени и данная задача была выполненна частично.

В итоге реализован модуль SIP-телефонии, который частично встроен в CRM-систему. В дальнейшем необходимо реализовать: серверную часть, которая получает информацию о SIP-аккаунте; регистрацию звонков в CRM-системе; поддержку большим числом браузеров.

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. Bowser - WebRTC on iOSElectronic resource, Ericsson Research. — URL: http://www.openwebrtc.org/bowser/ (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. FreePBXElectronic resource, Sangoma Technologies. — URL: https:

//www.freepbx.org/ (online; accessed: 20.06.2016).

17. vtigerCRM Data Model. — URL: https://www.vtiger.com/products/crm/docs/510/vtigerCRM_DataModel_5.2.1.pdf (online; accessed: 19.06.2016).
18. sipML5 live demoElectronic resource, Doubango Telecom. — URL: https://www.doubango.org/sipml5/call.htm?svn=250 (online; accessed: 19.06.2016).
19. 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 ?>
```