TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Tarkvaratehnika õppetool

Ärihoone liftide kaugjuhtimise mobiilirakendus

Bakalaureusetöö

Üliõpilane: Vitali Eller
Üliõpilaskood: 104549IAPB
Juhendaja: Kaarel Allik

Tallinn
2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

……. ……

(*kuupäev)* *(allkiri)*

Ärihoone liftide kaugjuhtimise mobiilirakendus

Annotatsioon

Antud bakalaureusetöö eesmärk on luua mobiilirakendus, mis lihtsustab ärihoone külalise identifitseerimist. Rakendus võimaldab teatada residendile külalise kohaletulekust. Ärihoone töötajal on võimalik näha külalist, rääkida temaga, vastata temale tekstisõnumiga, anda külastajaga tegelemine sekretärile üle, keelduda külastajaga kohtumast või saata lift külalise järele.

Eesmärgini jõudmiseks kirjeldatakse esmalt nii funktsionaalsed kui ka mitmefunktsionaalsed süsteeminõuded, lokaalse andmebaasi arhitektuur. Seejärel kirjeldatakse rakenduse ja API koostööd, võrreldakse probleemi lahendamiseks sobivaid tehnoloogiaid ja esitatakse rakenduse prototüüp.

Töö käigus on saavutatud kõik püstitatud eesmärgid. Töö tulemusena on valminud kaks rakendust iOS mobiiliplatvormile: üks tahvelarvutitele ning teine mobiiltelefonidele. Autori tehtud töö näitab, et kirjeldatud süsteem on efektiivne vahend info jagamiseks ärihoone töötaja ja külastaja vahel.

Lõputöö on kirjutatud vene keeles ning sisaldab teksti 51 lk, 4 peatükki, 21 joonist, 2 tabeli.

Mobile Remote Control Application for Elevators in Office Blocks

Abstract

The aim of this thesis is to create mobile application simplifying identification of business buildings' customers. This application makes it possible to notify the resident about coming visitors. Employee can see the customer, talk with him/her, reply with text message, redirect the call to secretary, decline the call, and also call the lift to the customer.

In order to achieve the aim this work discusses both functional and nonfunctional system requirements, architecture of local database. It also describes the relation with API; compares the best appropriate technologies for solution of this problem; and demonstrates the model of the application.

During the work all the aims set are achieved. As a result, two mobile applications for IOS platform are created – one for tablets and another for mobile phones. The work carried through by author demonstrated that such system is an effective tool for customers' identification, as well as for calling the lift distantly.

Present work is written on Russian language and consists of 51 pages, four chapters, 21 pictures and 2 tables.

Мобильное приложение для удалённого управления лифтом коммерческого здания

Аннотация

Целью данной дипломной работы является создание мобильного приложения, которое упрощает идентификацию посетителя коммерческого здания. Приложение позволяет оповестить резидента о том, что к нему кто-то пришел. Сотрудник может увидеть того, кто пришел, поговорить с гостем, ответить текстовым сообщением, перенаправить вызов секретарю, отклонить вызов, а также отправить за посетителем лифт.

Для достижения цели в работе описываются как функциональные, так и не функциональные требования к системе, архитектура локальной базы данных. Описывается взаимодействие с API. Проводится сравнение наиболее подходящих технологий для решения данной проблемы. Демонстрируется макет приложения.

В ходе работы все поставленные цели были достигнуты. В результате было создано два мобильных приложения для платформы iOS, одно для планшетов, второе для мобильных телефонов. Проделанная автором работа показала, что данная система являться эффективным инструментом для идентификации посетителя здания, а также для дистанционного вызова лифта.

Данная работа написана на русском языке и состоит из 51 страниц, 4 главы, 21 иллюстрацию, 2 таблиц.

Список сокращений и терминов

iOS	Операционная система для смартфонов, электронных планшетов и носимых проигрывателей, разрабатываемая и выпускаемая американской компанией Apple. [1]
API	**Application programming interface** Набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. [2]
iPad	Планшет, выпускаемый компанией Apple
iPhone	Смартфон, выпускаемый компанией Apple
SMS	Технология, позволяющая осуществлять приём и передачу коротких текстовых сообщений с помощью сотового телефона. [19]
Core Data	Контейнер хранилище данных при разработке приложений для iOS. [6]
Нефункциональные требования	**Non-functional requirements** Требования, которые определяют критерии работы системы в целом, а не отдельные сценарии поведения [20]
Функциональные требования	**Functional requirement** Требования объясняющие задачи или действия, которые должны быть выполнены [20]

Прецеденты системы	**Use cases** Обобщённые сведения о некоторых отношениях между вариантами использования, субъектами и системами [22]
VoIP	Телефонная связь по протоколу IP [21]
Push уведомления	**Push Notifications** Один из способов распространения информации (контента), когда данные поступают от поставщика к пользователю, на основе установленных параметров [10]
HTTPS	**HyperText Transfer Protocol Secure** Расширение протокола HTTP, поддерживающее шифрование.[23]
ОС	**Operating system** Комплекс взаимосвязанных программ, предназначенных для управления ресурсами вычислительного устройства и организации взаимодействия с пользователем.[24]
Шаблон проектирования или **паттерн**	**Design pattern** В разработке программного обеспечения — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста. [25]
MVC	**Model-view-controller** Схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента [26]

«Одиночка»	**Singleton**
	Паттерн гарантирует, что существует только один экземпляр данного класса. [12]
«Категория»	**Category**
	Мощный механизм, позволяющий добавлять методы к существующим классам без наследования. [12]
«Наблюдатель»	**Observer**
	Один объект уведомляет другие объекты об изменениях своего состояния. [12]
Сокеты	**Sockets**
	Программный интерфейс для обеспечения обмена данными между процессами. [27]
UI/ ГИП	**User interface/ Графический интерфейс пользователя**
	Разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений [5]
UX	**User experience**
	Ощущения, возникающие у человека при непосредственном взаимодействии с объектами окружающего мира. [28]
SSD	**Solid-state drive**
	Компьютерное немеханическое запоминающее устройство на основе микросхем памяти.[29]

HDD	**Hard disk drive** Запоминающее устройство (устройство хранения информации) произвольного доступа, основанное на принципе магнитной записи.[30]
Android	Операционная система для смартфонов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков, очков Google, телевизоров и других устройств.[31]
XML	**Extensible Markup Language** Расширяемый язык разметки [32]
«Нативное» приложение	**Native application** Приложение, которое было разработано для использования на конкретной платформе.

Список иллюстраций

Рисунок 1. Общая архитектура **24**
Рисунок 2. Примеры графического интерфейса **26**
Рисунок 3. Пример сохранения полученных данных в модель **26**
Рисунок 4. Пример получения данных из модели. **27**
Рисунок 5. iPhone Core Data модель **27**
Рисунок 6. iPad Core Data модель **27**
Рисунок 7. Схема создания OpenTok сессии **28**
Рисунок 8. Общая схема работы Push уведомлений **29**
Рисунок 9. Метод отправки текстового сообщения **30**
Рисунок 10. Инициализации паттерна “Одиночка” на примере класса SocketManager **31**
Рисунок 11. Принцип MVC **32**
Рисунок 12. Пример категории для класса UILabel **32**
Рисунок 13. Пример графического интерфейса iPad приложения **33**
Рисунок 14. Пример графического интерфейса iPhone приложения **34**
Рисунок 15. Пример запроса - вызов лифта **35**
Рисунок 16. Макет окна авторизации **40**
..... **40**
Рисунок 17. Макет окон поиска сотрудника **40**
Рисунок 18. Макет окна вызова сотрудника **41**
Рисунок 19. Макет окон авторизации и восстановления пароля **41**
Рисунок 20. Макет основного окна приложения (Main view) **42**
Рисунок 21. Макет окна «Входящий вызов» **42**

Список таблиц

Таблица 1. iPad Core Data 49

Таблица 2. iPhone Core Data 50

Содержание

Autorideklaratsioon **2**

Annotatsioon **3**

Abstract **4**

Аннотация **5**

Список сокращений и терминов **6**

Список иллюстраций **10**

Список таблиц **11**

Содержание **12**

Введение **14**

1. Постановка проблемы **15**

1.1 Описание проблемы и обзор похожих решений **15**

1.2 Постановка задач **15**

1.3 Обзор работы **16**

2. Анализ **17**

2.1 Прецеденты системы **17**

2.2 Пользователи **20**

2.3 Требования к системе **21**

2.3.1 Функциональные требования 21

2.3.1.1 iPad 21

2.3.1.2 iPhone 21

2.3.2 Нефункциональные требования 22

3. Архитектура системы **24**

3.1 Общая архитектура **24**

3.2 Клиентская часть **24**

3.2.1 Общая характеристика клиентской части 24

3.2.2 Графический интерфейс 25

3.2.3 Core Data 26

3.2.4 OpenTok API 28

3.2.5 Apple Push Notification Service 28

3.2.6 Sockets 29

3.2.7 Паттерны проектирования 30

3.3 iPad приложение **32**

3.3.1 Общая характеристика 32

3.3.2 Примеры графического интерфейса 33

3.4 iPhone приложение **33**

3.4.1 Общая характеристика 33

3.4.2 Примеры графического интерфейса 34

3.5 CocoaPods **34**

3.6 Удаленный вызов лифта **35**

3.7 Авторизация и аутентификация **35**

3.7.1 Access token 35

3.7.2 Refresh token 36
3.8 Серверная часть 36
3.8.1 KONE API 36
3.8.2 Общая характеристика 36
4. Реализация 38
4.1 Сравнение и выбор наиболее подходящих технологий 38
4.1.1 Выбор платформы 38
4.1.2 Выбор языка программирования 39
4.2 Макет iPad приложения 40
4.3 Макет iPhone приложения 41
Заключение 43
Kokkuvõte 44
Список используемой литературы 46
Дополнение 1 49

Введение

Целью данной дипломной работы является создание мобильного приложения, которое решает проблему взаимодействие работника здания и посетителя, позволяет в реальном времени оповещать работника о том, что к нему пришли. В свою очередь работник может ответить, голосом, текстовым сообщением, перенаправить вызов секретарю, отменить вызов или отправить лифт за посетителем. Не редкость когда человек приходит в здание впервые, он не знает на каком этаже находится нужная ему компания, иногда он знает только имя человека, к которому пришел, а в лобби не у кого спросить. Для решения данной проблемы используются указатели и различные информативные стенды. Также часто встречается, что доступ в лифт разрешен только работникам. Данное мобильное приложение, комплексное решение данных проблем.

Мобильное приложение разработано в компании Mobile Digital OÜ, автор данной работы является сотрудником данной компании и ведет непосредственную разработку мобильного приложения.

1. Постановка проблемы

1.1 Описание проблемы и обзор похожих решений

На данный момент существует очень мало мобильных приложений, которые позволяют вызывать лифт дистанционно. Существует много решений того, как мог бы посетитель связаться с работником здания, но нет единого решения для всех этих задач. Ниже приведены примеры уже существующих решений.

Первая категория приложений – это приложения для управления лифтами. Одним из примеров такого рода приложений является KONE RemoteCall [3]. Приложение позволяет удаленно вызвать лифт, программа дает возможность наблюдать в реальном времени за перемещением лифтов. Но в данном приложении нет функционала связи между работником и гостем. Для этого придется использовать какой-то другой способ.

Вторая категория - это приложения, которые решают вопрос, как связаться с работником. Если посетитель знает мобильный телефон человека, к которому он пришел, он может позвонить. Или же гость может узнать информацию у консьержа в лобби. Недостатки:

- Гость может не знать номера телефона.
- В лобби никого нет, и нет возможности спросить.
- Если работник не может ответить, например он на совещании, он будет вынужден отправить SMS ответ.
- Если работник не отвечает, гость может растеряться и не знать, что ему делать дальше и может уйти.

1.2 Постановка задач

Целью данной дипломной работы является создание прототипа приложения, которое решает проблему связи между посетителем здания и работником, позволяет в реальном времени оповещать работника о том, что у него посетитель. При помощи приложения можно наблюдать за передвижение лифтов в реальном времени, а так же удаленно вызывать лифт.

Данное приложение значительно упростит ориентирование в лобби. Посетителю не придется обращаться к консьержу или искать нужную компанию в длинном списке. Регистрация в системе необходима только для работников и для планшетов iPad. Все зарегистрированные пользователи будут отображаться в списке работников.

Задачами данной работы являются:

- Анализ требований системы
- Описание общей архитектуры
- Написание прототипа приложений, как для iPad, так и для iPhone

1.3 Обзор работы

Данная работа делится на 3 основные части.

В первой части проводится анализ требований системы. Описываются прецеденты системы, функциональные и нефункциональные требования.

Во второй части описывается архитектура системы, структура iPad и iPhone приложений. Описание графического интерфейса, структура Core Data. Описываются процессы авторизации и аутентификации.

В третьей части проводится анализ технологий, наиболее подходящих для реализации данной системы, иллюстрируется макет приложения.

2. Анализ

2.1 Прецеденты системы

Основными прецедентами использования данной системы являются:

- Просмотр списка фирм
- Просмотр списка работников
- VoIP звонок
- Голосовой ответ
- Ответ текстовым сообщением
- Перенаправление вызова секретарю
- Отправить лифт
- Вызов лифта
- Наблюдение за передвижением лифтов в настоящем времени

Прецеденты системы:

Название: Просмотр списка фирм

Действующие лица: Посетитель/неавторизованный пользователь

Устройство: iPad

Описание: Посетитель/неавторизованный пользователь может просматривать список названий фирм, которые находятся в здании. Имена компаний упорядочены в алфавитном порядке. Существует возможность быстрого поиска по первой букве названия фирмы. Выбрав из списка нужную компанию, пользователь переходит к списку работников данной компании. Также имеется возможность переключиться на список всех работников здания, которые зарегистрированы в системе.

Название: Просмотр списка работников

Действующие лица: Посетитель/неавторизованный пользователь

Устройство: iPad

Описание: Посетитель/неавторизованный пользователь может просматривать список работников фирмы. Имена работников также упорядочены в алфавитном порядке. Существует возможность быстрого поиска по первой букве имени сотрудника. Выбрав из списка нужного сотрудника компании, пользователь переходит к экрану VoIP вызова.

Название: VoIP звонок

Действующие лица: Посетитель/неавторизованный пользователь, сотрудник/авторизованный пользователь

Устройство: iPad, iPhone

Описание: Посетитель имеет возможность связаться с сотрудником используя VoIP звонок. Во время звонка передается видео изображение посетителя, передача звука изначально запрещена. Вызов сотрудника длится 30 секунд, до того, как вызов прекратится. Как только сотрудник компании отвечает на вызов, отсчет останавливается.

Название: Голосовой ответ

Действующие лица: Посетитель/неавторизованный пользователь, сотрудник/авторизованный пользователь

Устройство: iPad, iPhone

Описание: У авторизованного пользователя есть возможность ответить на звонок используя микрофон. Для этого сотрудник должен нажать и удерживать кнопку «Говорить». В этот момент обе стороны могут разговаривать между собой.

Название: Ответ текстовым сообщением

Действующие лица: Сотрудник/авторизованный пользователь

Устройство: iPhone

Описание: Авторизованный пользователь имеет возможность ответить на входящий вызов текстовым сообщением. Доступно три варианта текстовых сообщений, которые пользователь может изменять.

Название: Перенаправление вызова секретарю

Действующие лица: Сотрудник/авторизованный пользователь

Устройство: iPhone

Описание: Сотрудник имеет возможность перенаправить звонок секретарю компании, в случае если он не хочет или не может говорить. Также звонок будет автоматически перенаправлен в случае если пользователь принял вызов, но в течении 15 секунд не дал ответа посетителю. Еще одна возможность автоматически перенаправить вызов, в настройках приложения выбрать «Нет на месте». В таком случае звонок будет сразу направлен секретарю.

Название: Отправка лифта

Действующие лица: Сотрудник/авторизованный пользователь

Устройство: iPhone

Описание: Сотрудник имеет возможность отправить лифт за посетителем. В данном случае посетитель увидит сообщение о том, какой лифт за ним прибудет, на каком он этаже и на какой этаж ему надо ехать.

Название: Вызов лифта

Действующие лица: Сотрудник/авторизованный пользователь

Устройство: iPhone

Описание: Авторизованный пользователь имеет возможность удаленно, указав с какого и на какой этаж. Существует возможность выбрать необходимую группу лифтов, если в здании их несколько.

Название: Наблюдение за передвижением лифтов в настоящее время

Действующие лица: Сотрудник/авторизованный пользователь

Устройство: iPhone

Описание: Авторизованный пользователь может наблюдать в реальном времени где находятся лифты.

2.2 Пользователи

В данной системе существует 2 вида пользователей:

- Неавторизованный пользователь, посетитель, которому не нужна авторизация для использования приложения.
- Авторизованный пользователь, сотрудник, совершивший вход в систему используя свой аккаунт.

Неавторизованные пользователи имеют возможность просматривать список фирм и список сотрудников, совершать VoIP звонок.

Авторизированные пользователи могут наблюдать за передвижением лифтов, вызывать лифт, принимать VoIP вызов, отвечать на него голосом, текстовым сообщением, перенаправлять вызов секретарю, а также могут отправить лифт за посетителем.

2.3 Требования к системе

2.3.1 Функциональные требования

2.3.1.1 iPad

- Посетитель имеет возможность пользоваться приложением без регистрации аккаунта, но приложение должно быть авторизированно в системе сотрудником.
- Возможен просмотр списка фирм.
- Возможен просмотр списка всех сотрудников здания, авторизированных в iPhone приложении.
- Неавторизированный пользователь может начать VoIP вызов.
- В случае отсутствия ответа, перенаправление вызова секретарю компании.
- Неавторизированный пользователь получает ответ от сотрудника.
- Без доступа в интернет система не активна до того времени, как соединение не будет восстановлено.
- При интернет соединении все запросы направляются на сервер и полученные данные обрабатываются: отображаются графически или же сохраняются локально.

2.3.1.2 iPhone

- Пользоваться приложением может только авторизованный пользователь.
- При выборе двух этажей, с какого этажа и на какой, вызывается лифт.
- Возможность выбирать язык интерфейса – эстонский, русский или английский.
- Пользователь может изменять текстовые сообщения.
- Пользователь может указывать статус «Нет на месте»
- Возможность выбрать «Этаж по умолчанию»
- Авторизованный пользователь мгновенно получает оповещение о входящем VoIP вызове.
- Можно перенаправить входящий вызов секретарю в том случае, если авторизованный пользователь сам не является секретарем.

- Мгновенный ответ текстовым сообщением
- Отправка лифта за посетителем, система определяет с какого этажа был вызов и лифт движется с данного этажа на «Этаж по умолчанию»
- При интернет соединении все запросы направляются на сервер и полученные данные обрабатываются: отображаются графически или же сохраняются локально.
- При отсутствии интернет соединения возможно использовать только те функции, которые не связаны с API.

2.3.2 Нефункциональные требования

- Совместимость
    - Приложения должны работать на мобильных устройствах iOS, начиная с версии 8.4
- Быстродействие
    - Ответ с сервера должен приходить мгновенно
    - При наличии интернета Push уведомления должны приходить в течении 5 секунд.
    - При обновлении данных с сервера, приложение должно оставаться доступным для пользователя.
- Защищенность
    - Использование HTTPS[8]
    - Без access token невозможно получить данные с сервера
- Надежность
    - Не должно быть ошибок, которые приведут к закрытию приложения.
    - Отсутствие исключений.

- Простота
    - Простой и понятный UI
    - Продуманный UX
    - Три языка интерфейса – эстонский, русский, английский.

3. Архитектура системы

3.1 Общая архитектура

Данная система использует архитектуру “Клиент-Сервер”. Модель функционирования такой системы заключается в следующем: клиент делает запрос серверу, сервер (серверная часть) получает запрос, выполняет его и отсылает результат клиенту (клиентская часть). [4]

Клиентской частью системы являются iOS приложения (iPhone, iPad).

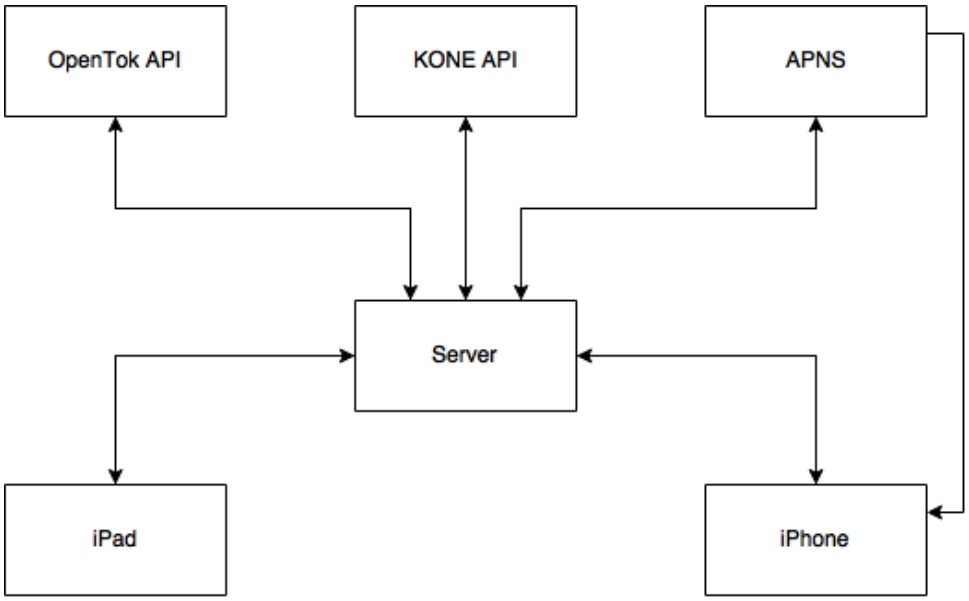

Рисунок 1. Общая архитектура

3.2 Клиентская часть

3.2.1 Общая характеристика клиентской части

iOS приложения являются клиентской частью данной системы. Клиентская часть запускается из класса AppDelegate. AppDelegate является основной точкой входа в приложение, данный класс содержит методы, которые отвечают за жизненный цикл приложения. Класс отвечающий за запросы – BaseRequest. Данный класс является родительским классом для всех запросов к серверу. Все контроллеры приложения наследуются от класса BaseViewController. Используются следующие паттерны проектирования – MVC, Singletone, Observer. Далее будут расписаны основные технологии и паттерны клиентской части.

3.2.2 Графический интерфейс

Графический интерфейс пользователя (ГИП/UI)- разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений [5]. При создании интерфейса придерживались следующих правил:

- **Простота** - чем меньше кнопок, тем проще пользователю пользоваться приложением.
- **Уникальность** – графический интерфейс должен запомниться пользователю.
- **Отклик** – приложение должно откликаться на любое действие пользователя, чтобы подтвердить взаимодействие с ним.
- **Большой палец** – Как правило пользователь держит смартфон в одной руке и в основном взаимодействует с приложением большим пальцем. Интерфейс должен быть продуман так, чтобы все действия можно было совершать только большим пальцем.
- **Внимание к деталям**

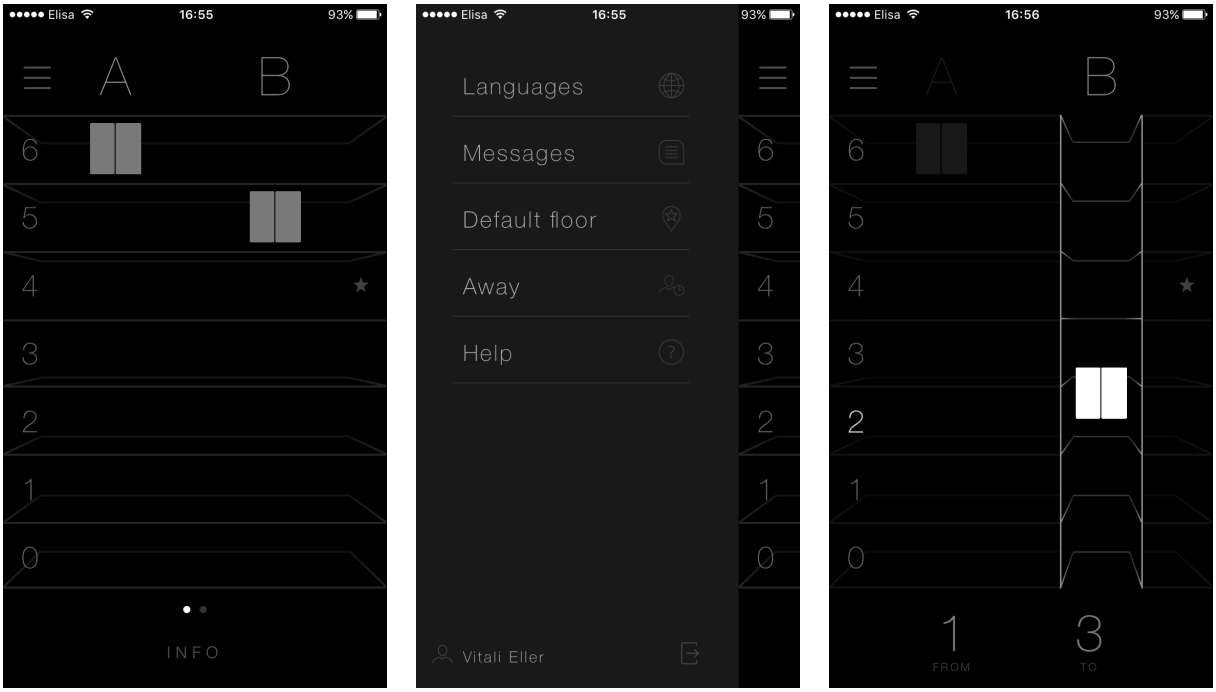

Рисунок 2. Примеры графического интерфейса

3.2.3 Core Data

Core Data представляет из себя контейнер хранилище данных при разработке приложений для iOS. Они хранятся в бинарном виде (так называемом Core Data формате: xcdatamodeld).[6] Для упращения работы с Core Data автор выбрал библиотеку RHManagedObject [7]. Данная библиотека значительно упрощает работу с моделями данных. Согласно документации Apple [8], получение объекта из Core Data занимает примерно 10 строк кода, используя библиотеку RHManagedObject это можно сделать в одну строку.

Пример сохранения объекта в модель:

```
- (BOOL)parseResponseArray:(NSArray *)array
{
    [Person deleteWithPredicate:[NSPredicate predicateWithFormat:@"companyID=nil"] error:nil];

    for (NSDictionary *companyDict in array) {

        Person *person = [Person newEntityWithError:nil];

        if (person) {
            [person setEmail:[companyDict getString:@"email"]];;
            [person setName:[companyDict getString:@"name"]];
            [person setPhone:[companyDict getString:@"phone"]];
        }
    }

    [Person commit];

    return YES;
}
```

Рисунок 3. Пример сохранения полученных данных в модель

Пример получения объекта из модели:

```
Person *selectedPerson = [Person getWithPredicate:[NSPredicate predicateWithFormat:@"name = %@", personName]  error:nil];
```

Рисунок 4. Пример получения данных из модели.

В данной системе используется две модели Core Data:

- iPhone Model – модель данных для iPhone приложения. В данной модели хранятся данные о пропущенных вызовах.

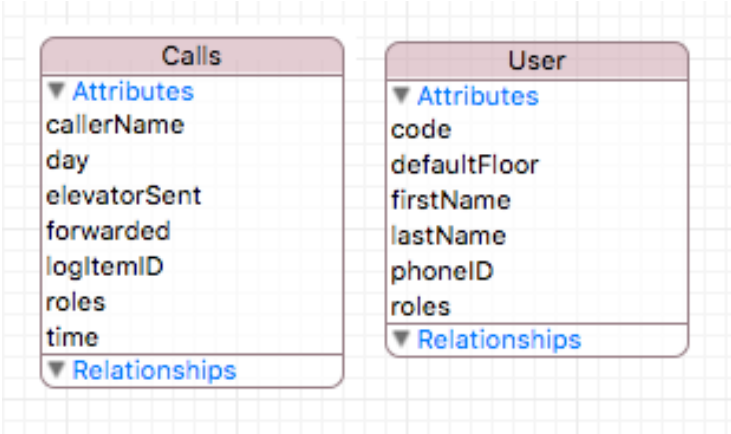

Рисунок 5. iPhone Core Data модель

- iPad Model – модель данных для iPad приложения. В модели хранятся данные о всех фирмах и о сотрудниках.

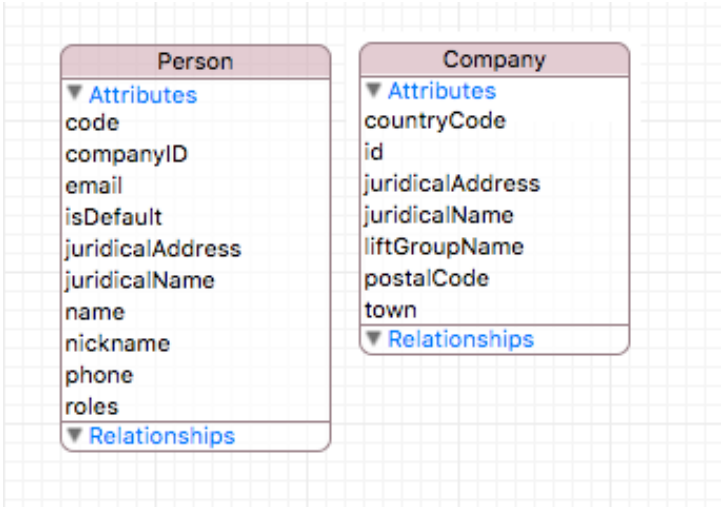

Рисунок 6. iPad Core Data модель

3.2.4 OpenTok API

OpenTok является лидирующей WebRTC платформой для аудио и видео чатов, передачи текстовых сообщений, а также предоставляет возможность screen share [9]. В данной системе используются аудио и видео чат. При начале VoIP вызова, iPad приложение отправляет запрос на сервер для создания сессии, сервер возвращает дынные сессии: `apiKey, sessionId, token`. Далее используя полученные данные, iPad подключается к сессии и отправляет эти параметры абоненту с помощью Push Notification. Получив уведомление, абонент подключается к сессии.

Рисунок 7. Схема создания OpenTok сессии

3.2.5 Apple Push Notification Service

Push-уведомления - один из способов распространения информации (контента), когда данные поступают от поставщика к пользователю, на основе установленных параметров. Пользователь же в свою очередь либо отвергает, либо принимает данные. [10]

Для того, чтобы уведомление отобразилось на экране, приложение не должно быть запущенно, именно это и является основным преимуществом данной технологии. Нет необходимости беспрерывной проверки событий в фоновом режиме, эту задачу выполняет серверная часть.

Данная технология используется в приложении для отправки VoIP вызова. Так как вызов должен доходить до абонента не зависимо от того, включено у него приложение или нет, автор выбрали именно эту технологию. Также push уведомления помогают экономить заряд батареи и трафик. В отличии от Socket сообщений, push уведомления доходят до пользователя даже в том случае, если телефон находится в спящем режиме.

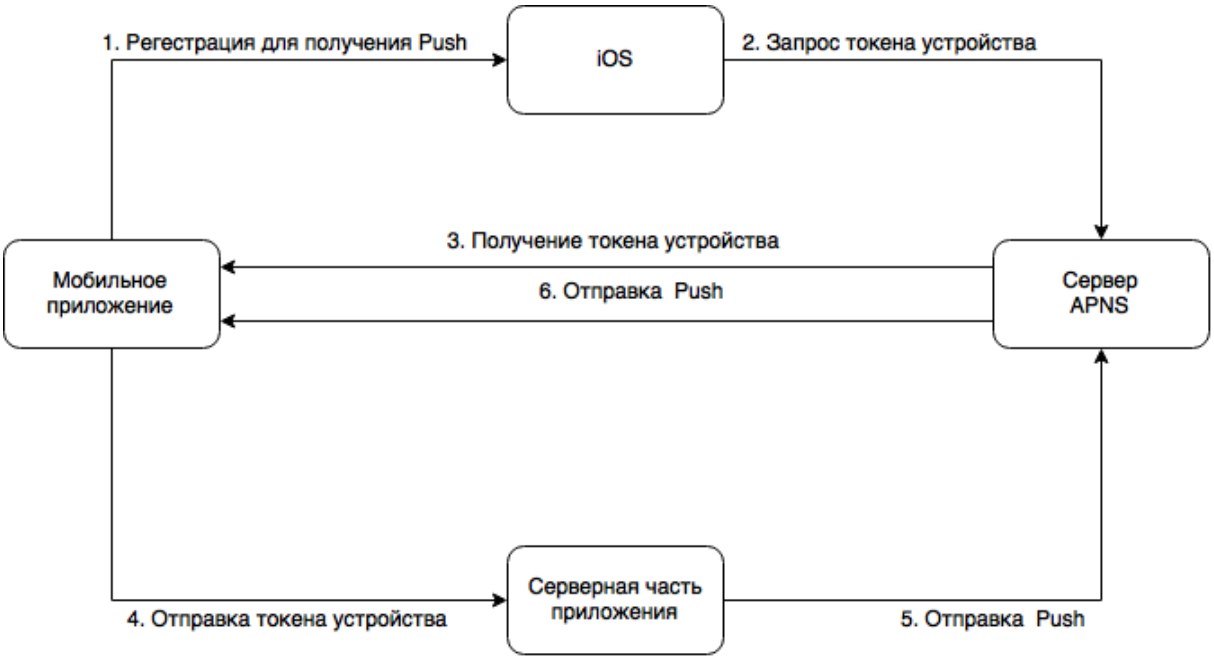

Рисунок 8. Общая схема работы Push уведомлений

3.2.6 Sockets

Сокеты - программный интерфейс для обеспечения обмена данными между процессами. [11] Сокеты делятся на два вида – клиентские и серверные. Серверный сокет прослушивает определенный порт, а клиентский подключается к серверу. После успешной установки соединения сервер и клиент начинают обмениваться информацией. В данной работе информацией обмениваются iPad и iPhone приложения. Примером использования сокетов является возможность ответа на VoIP вызов текстовым сообщением. Одни из главных минусов является тот факт, когда устройство на платформе iOS находится в спящем режиме, ОС блокирует доступ приложениям в интернет, для сохранения заряда батареи. В этом случае, для обмена информацией используются Push уведомления.

В данном приложении синхронизация между устройствами нужна на протяжении всего VoIP вызова, для избежания нагрузки на сервер и отправки многочисленных запросов, было решено использовать сокеты.

```
- (void)sendMessage:(NSString*)message
{
    NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                          [self generateRandomID],  @"id",
                          @"message", @"type",
                          [FDKeychain itemForKey:kCodeKey forService:kKeychainServiceKey error:nil], @"deviceId",
                          [FDKeychain itemForKey:@"iPadCode" forService:kKeychainServiceKey error:nil], @"receiverId",
                          message, @"message",
                          nil];
    [self sendDictionary:dict];
    [self close];
}

- (void)sendDictionary:(NSDictionary*)dictionary
{
    NSError *error;
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dictionary
                                                       options:0
                                                         error:&error];

    NSString *JSONString = [[NSString alloc] initWithBytes:[jsonData bytes] length:[jsonData length] encoding:NSUTF8StringEncoding];

    NSData *plainData = [JSONString dataUsingEncoding:NSUTF8StringEncoding];

    NSString *pubkey = @"PUBLIC KEY";

    NSData *ret = [RSA encryptData:plainData publicKey:pubkey];

    NSData*base64 = [ret base64EncodedDataWithOptions:0];

    NSString *myString = [[NSString alloc] initWithData:base64 encoding:NSUTF8StringEncoding];

    if (!jsonData) {
        NSLog(@"JSON error: %@", error);
    } else {
        NSData *data = [[NSData alloc] initWithData:[myString dataUsingEncoding:NSASCIIStringEncoding]];
        [_outputStream write:[data bytes] maxLength:[data length]];
    }

}
```

Рисунок 9. Метод отправки текстового сообщения

3.2.7 Паттерны проектирования

Паттерн — это образец решения некоторой проблемы. Решение, которое можно повторить в другом проекте. Паттерны упрощают понимание кода. Они помогают писать слабо-связанный код, такой код, в котором можно легко модифицировать компоненты, или заменить целый компонент на его аналог, почти не трогая остальную часть проекта.[12] В проекте используются:

- Порождающие паттерны – Одиночка (singleton).

Одиночка – данный паттерн гарантирует, что существует только один экземпляр данного класса. [12]

```objc
static SocketManager *sharedManager = nil;

#pragma mark -
#pragma mark Singletone

+ (void) initialize
{
    if (self == [SocketManager class])
    {
        sharedManager = [[self alloc] init];
    }
}

-(id) init
{
    self = [super init];

    if (self)
    {

    }

    return self;
}

+ (SocketManager *)sharedManager
{
    return sharedManager;
}
```

Рисунок 10. Инициализации паттерна “Одиночка” на примере класса SocketManager

- Структурные паттерны – MVC, Category

MVC (Model-View-Controller) – является одним из наиболее используемых паттернов.

Модель – содержит данные об объекте и механизмы манипуляций над ним. [12] Одним из примеров модели в данном приложении является класс Lift.

View – Отвечает за визуальное отображение модели. [12] Представлением модели Lift, является класс LiftView.

Controller – координирует всю работу. Имеет доступ к данным модели, отображает их в представлениях, подписывается на события и манипулирует данными. [12]

Рисунок 11. Принцип MVC

«Категория» (Category) — очень мощный механизм, позволяющий добавлять методы к существующим классам без наследования. [12]

```
@implementation UILabel (SpaceBetweenLetters)

- (void)setLabelText:(NSString*)text withSpace:(float)space
{
    if (self.attributedText ){
        self.attributedText = nil;
    }

    NSMutableAttributedString *attributedString = [[NSMutableAttributedString alloc] initWithString:text];
    [attributedString addAttribute:NSKernAttributeName
                             value:@(space)
                             range:NSMakeRange(0, attributedString.length)];

    self.attributedText = attributedString;
}

@end
```

Рисунок 12. Пример категории для класса UILabel

- Поведенческие паттерны – Наблюдатель (observer)

В паттерне «Наблюдатель» один объект уведомляет другие объекты об изменениях своего состояния. Объектам, связанным таким образом, не нужно знать друг о друге. [12] Push-уведомления являются хорошим примером данного подхода проектирования.

3.3 iPad приложение

3.3.1 Общая характеристика

Данное приложение служит для навигации посетителя в здании, а так же для связи между посетителем и резидентом здания. Посетитель выбирает из списка нужную компанию, далее сотрудника, к которому он направляется. Также данное приложение помогает направлять пользователя к нужному лифту, в случае если посетитель выбрал из списка компанию, но данная группа лифтов не предоставляет доступ к выбранной компании, приложение подскажет, к какой группе идти. Кроме этого, существует возможность информирование посетителя о том, что лифт отправлен, на каком он

этаже и какой лифт ему ожидать. Основной целью является – VoIP вызов сотрудника. Приложение отправляет Push уведомление, тем самым информируя резидента здания о том, что к нему посетитель.

3.3.2 Примеры графического интерфейса

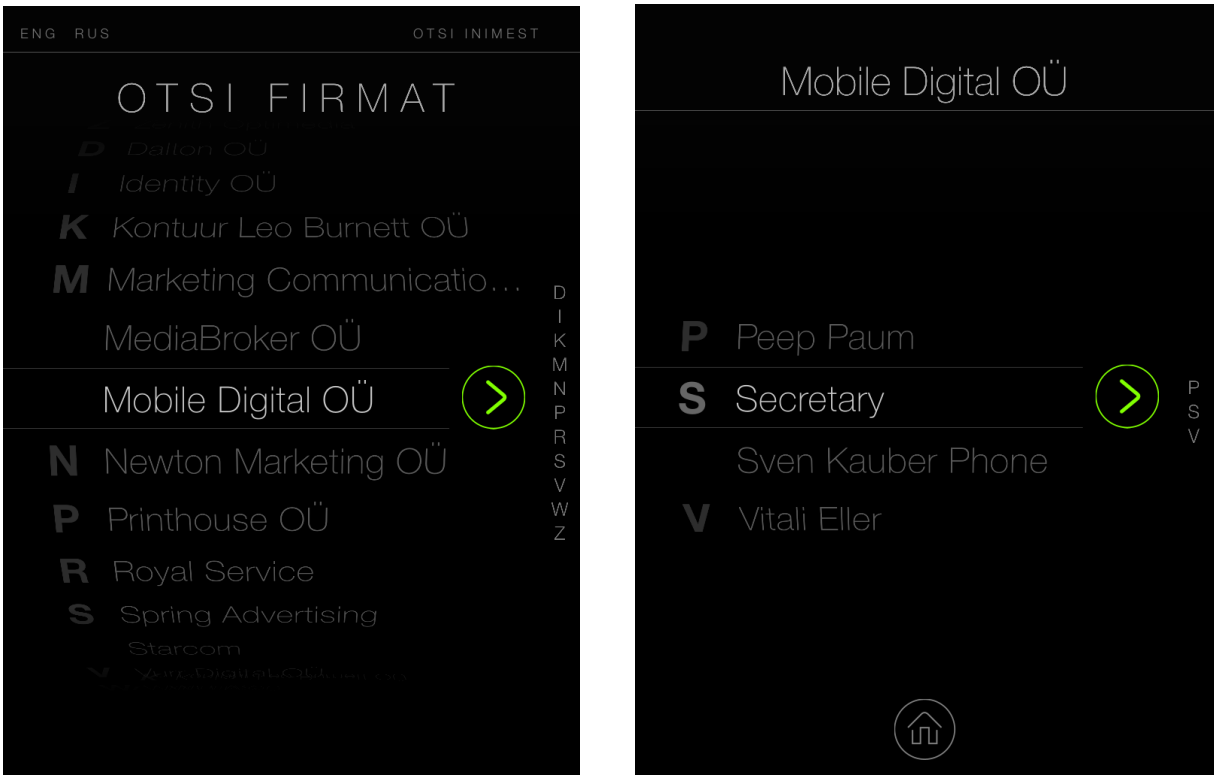

Рисунок 13. Пример графического интерфейса iPad приложения

3.4 iPhone приложение

3.4.1 Общая характеристика

Основной целью данного приложения является прием VoIP вызова и информирования резидента здания о том, что к нему посетитель. Так же одной из ключевых возможностей является дистанционное управление лифтами. Авторизированный пользователь имеет возможность принять или отклонить вызов, перенаправить вызов секретарю, ответить текстовым сообщением, поговорить с посетителем, отправить лифт, а так же вызвать лифт для личного передвижения между этажами.

Общение iPhone приложения с iPad приложением производится по средствам Push уведомлений, а так же при помощи сокетов (sockets). В первую очередь, устройство получает Push уведомление от iPad, дальнейшее общение происходит по средствам сокетов.

3.4.2 Примеры графического интерфейса

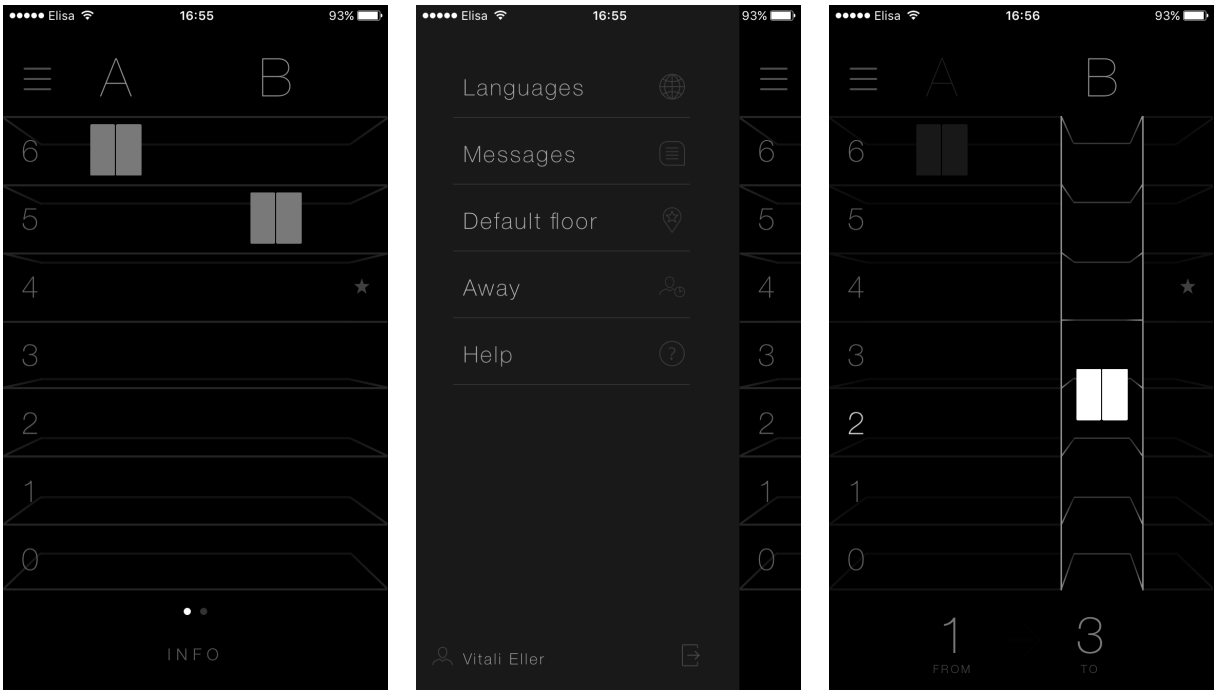

Рисунок 14. Пример графического интерфейса iPhone приложения

3.5 CocoaPods

CocoaPods [13] – очень мощный и популярный инструмент среди iOS и Mac OS X разработчиков для подключения сторонних библиотек в проект, которые значительно упрощают разработку мобильных приложений. Все подключаемые библиотеки необходимо добавлять в Podfile. Podfile – это спецификация, где обозначены все зависимости для проекта. [14]

CocoaPods имеет следующие преимущества:

- Позволяет просмотреть список подключенных библиотек в одном месте
- Позволяет легко следить за версиями библиотек
- Позволяет быстро обновить все библиотеки

На момент написания работы автор использует CocoaPods 0.39.0.

3.6 Удаленный вызов лифта

Одной из главныйх функций данной системы является возможность удаленного вызова лифта. При успешной авторизации, клиентское приложение получает xuuid для каждой группы лифтов, к которым данный пользователь имеет доступ. Далее для вызова лифта отправляется запрос с параметрами : source_floor, destination_floor, xuuid.

- source_floor – текущий этаж
- destination_floor – этаж, на который пользователь желает попасть.
- xuuid - уникальный идентификатор группы лифтов

После отправки запроса, серверная часть перенаправляет полученные параметры в KONE API и уже там отдается команда лифтам. Прямой связи серверной части и лифтов нет. Все запросы, связанные с лифтами, обрабатывает KONE API.

```
- (instancetype)initWithSourceFloor:(NSString *)sourceFloor and:(NSString*)destinationFloor xuuid:(NSString*)xuuid
{

    self = [super authenticatedRequestWithData:@{@"source_floor" : sourceFloor,
                                                  @"destination_floor":destinationFloor,
                                                  @"xuuid":xuuid} andMethod:@"elevator/move"];

    if (self) {
        self.serializer = [AFHTTPRequestSerializer serializer];
        self.requestAPIUrl = kApiSharedBaseUrl;
    }

    return self;
}
```

Рисунок 15. Пример запроса - вызов лифта

3.7 Авторизация и аутентификация

3.7.1 Access token

После успешной авторизации приложение клиента получает access token. Access token - некий ключ (обычно просто набор символов), предъявление которого является пропуском к защищенным ресурсам. Обращение к ним в самом простом случае происходит по HTTPS с указанием в заголовках или в качестве одного из параметров. [15] Полученный access token сохраняется в связке ключей (Keychain). Keychain — функция (другими словами — технология), с помощью которой, в одном месте операционных системах Mac OS X и iOS, в защищённом виде, сохраняются личные данные пользователя (логины и пароли). [16] Далее access token используется для аутентификации запросов от пользователя. Access token имеет длительность жизни 24

часа, спустя сутки данный ключ становится не активным и для его обновления необходим refresh token. Кроме того, приложение сохраняет дату и время, когда был получен ключ авторизации. Это необходимо для того, чтобы в дальнейшем своевременно его обновить.

3.7.2 Refresh token

Также как и access token, refresh token приходит с ответом после успешной авторизации. Он также сохраняется в связке ключей. Срок жизни refresh token-а составляет 10 дней. После авторизации, в случае, если приложение активно, в фоновом режиме каждый час происходит проверка, сколько часов еще действителен ключ авторизации, сравнивая нынешнее время с тем временем, когда был получен access token. Иначе, если приложение было какое-то время не активно, то при активации приложения, проверяется действителен ли access token. В случае, если ключ доступа не активен, приложение отправляет запрос на его обновления, используя refresh token. В том случае, когда приложение активно, access token будет обновлен за 3 часа до истечения срока действия. При обновлении ключа доступа, также обновляется и refresh token.

3.8 Серверная часть

3.8.1 KONE API

KONE - финская компания, специализирующаяся на производстве лифтов, кранов, эскалаторов, траволаторов и пассажирских подъемников.[17] Данное приложение основано на API, предоставленное компанией KONE. Серверная часть обращается к KONE API, обрабатывает полученные данные и возвращает их клиентской части. Данная система будет работать только с лифтами произведенными компанией KONE.

3.8.2 Общая характеристика

Серверная часть расположена на облачном сервисе DigitalOcean. Данный сервис был выбран по следующим причинам:

- SSD вместо HDD
- Быстрая и легкая настройка
- Готовые пакеты

Операционная система сервера - Debian GNU / Linux 8.1(Jessie) с архитектурой x64.

Веб сервер - Nginx 1.6.2 I/O.

База данных - MariaDB 10.0.16. MariaDB - ответвление СУБД MySQL.

PHP 5.6.9, PHP-FPM (FastCGI Process Manager) – оптимален для сервисов с высокой посещаемостью.

Socket-server: Ratchet Websockets for PHP.

Сервис отправки e-mail – MailGun.

4. Реализация

4.1 Сравнение и выбор наиболее подходящих технологий

4.1.1 Выбор платформы

Одной из главных проблем, в начале разработки является выбор платформы iOS или Android. С какой платформы начать разработку? Основные критерии:

- Быстрая и удобная среда разработки

В данной категории выбор был сделан в пользу iOS, так как Xcode намного удобнее, быстрее и мощнее, чем Eclipse.

- Симулятор устройства

Важным фактором является возможность быстрой сборки проекта на симуляторе для проверки. Симулятор Android очень медленный и зачастую выдает ошибки, на его загрузку и настройку уходит очень много времени. В ту очередь, как симулятор iOS запускается нажатием одной кнопки.

- Дизайн интерфейса

Xcode имеет встроенный interface builder, который позволяет быстро и удобно собрать интерфейс для iOS приложений. В Android необходимо писать XML файлы, содержащие инструкции по отрисовке интерфейса.

- Стабильность платформы

Смартфоны и планшеты компании Apple значительно надежнее, чем устройства на платформе Android. Даже старый iPhone работает достаточно быстро и без зависаний, в то время как аппараты Android примерно после двух лет службы начинают зависать и тормозить. По аппаратной надежности iPhone также впереди всех производителей: по исследованию компании Strategy Analytics, устройства Apple почти в три раза надежнее смартфонов Samsung и в пять – Nokia. [18]

- Целевая аудитория

Основными пользователями данной системы являются офисные работники. По статистике, бизнесмены и офисные рабочии используют смартфоны на платформе iOS, так как iPhone считается статусной вещью.

Исходя из перечисленных пунктов, было принято решение начинать разработку под iOS, и уже потом для Android.

4.1.2 Выбор языка программирования

Основными языками программирования, для разработки «нативных» приложения для iOS являются Objective-C и Swift. Каждый из перечисленных языков обладает своими плюсами и минусами.

Плюсы языка Swift:

- Он более читаемый
- Безопаснее чем Objective-
- Меньше кода
- Более современный язык программирования

Плюсы языка Objective-C:

- Проверен и оттестирован
- Много обучающего материала
- Большое количество сторонних библиотек

Выбор был сделан в пользу языка Objective-C, так как данный язык проверен временем и на нем написано более миллиона приложений. Также в интернете есть огромное количество сторонних библиотек на Objective-C, которые значительно упрощают разработку приложений.

Swift значительно набирает обороты, и в ближайшем будущем вытеснит Objective-C.

4.2 Макет iPad приложения

Макет iPad приложения наглядно иллюстрирует, как выглядит приложение (см. Рисунок 16, Рисунок 17, Рисунок 18). Для создания макета использовался инструмент Balsamiq Mockups.

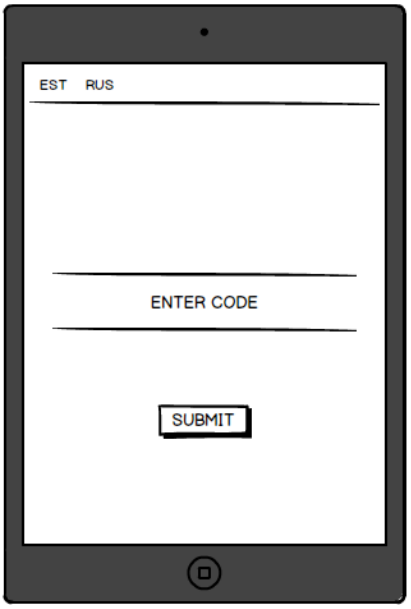

Рисунок 16. Макет окна авторизации

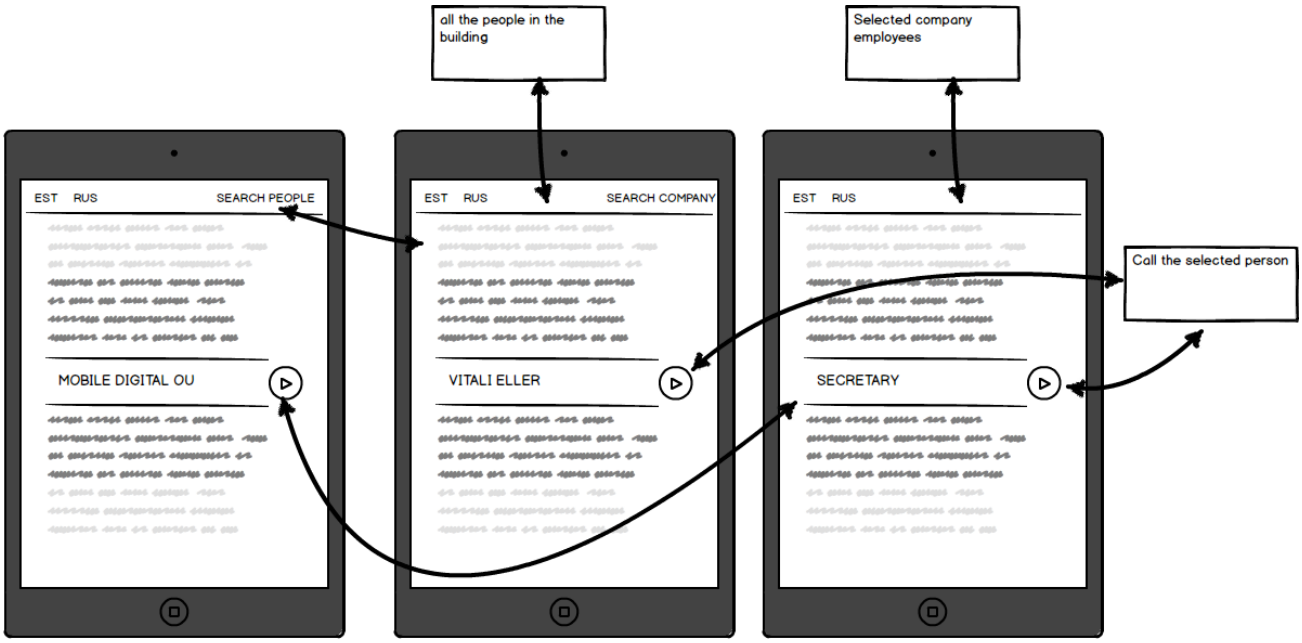

Рисунок 17. Макет окон поиска сотрудника

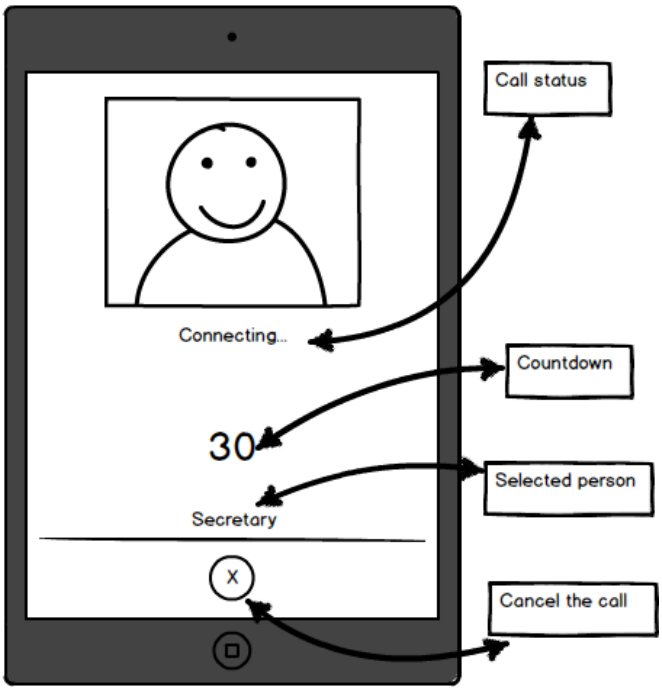

Рисунок 18. Макет окна вызова сотрудника

4.3 Макет iPhone приложения

Макет iPhone приложения наглядно иллюстрирует, как выглядит приложение (см. Рисунок 19, Рисунок 20, Рисунок 21). Для создания макета использовался инструмент Balsamiq Mockups.

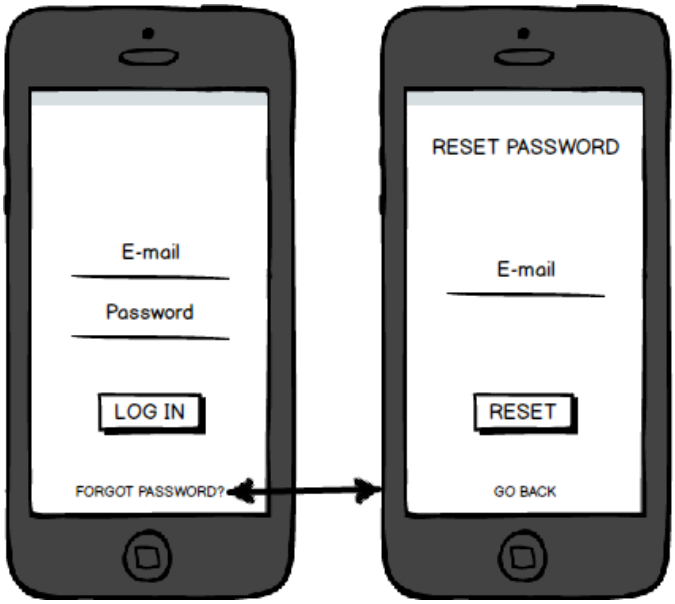

Рисунок 19. Макет окон авторизации и восстановления пароля

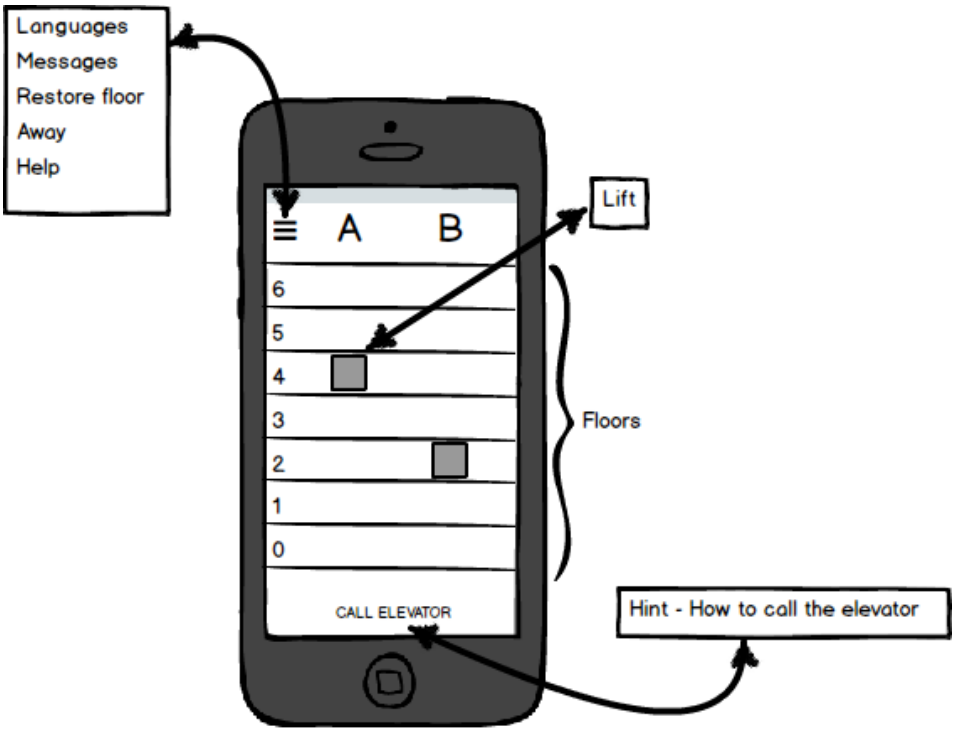

Рисунок 20. Макет основного окна приложения (Main view)

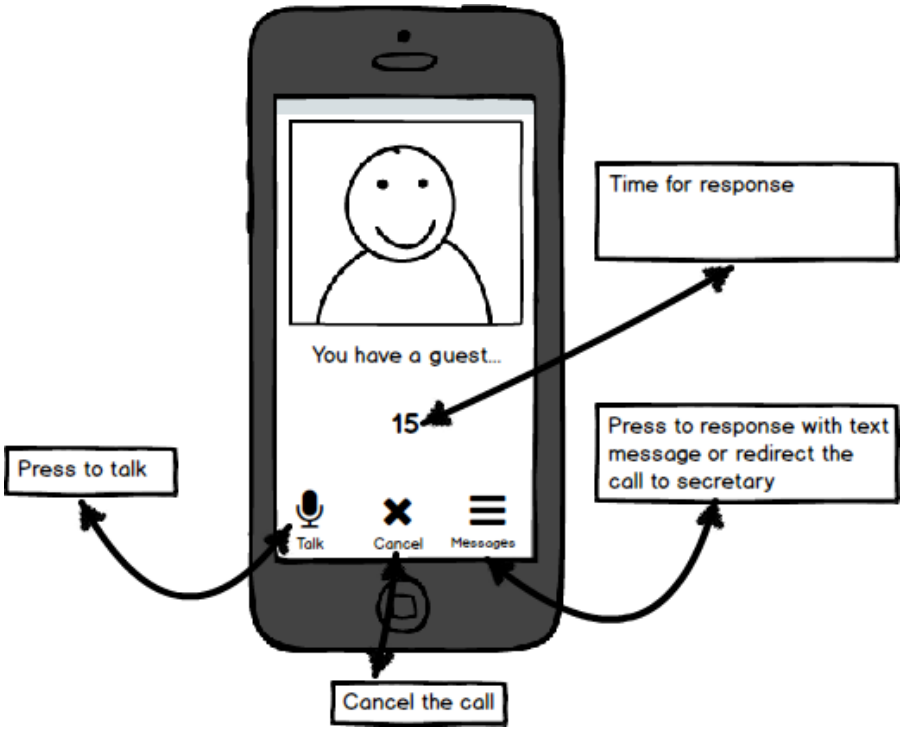

Рисунок 21. Макет окна «Входящий вызов»

Заключение

В ходе работы все поставленные цели были достигнуты: были описаны прецеденты системы, функциональные и нефункциональные требования, архитектура клиентской части, также была описана общая информация о серверной части. Были описаны основы работы push уведомлений, основные паттерны проектирования, используемые в проекте, структура для хранения данных - Core Data, а также Sockets. Были рассмотрены основные принципы, на которых основан графический интерфейс пользователя. Также было проведено сравнение языков программирования для iOS – Swift и Objective-C. Как результат, была создана программа для идентификации посетителя и для удаленного вызова лифта. Прототип приложения готов и в будущем планируется дальнейшее развитие.

Проделанная автором работа показала, что данное приложение является эффективным инструментом для идентификации посетителя, а так же для удаленного вызова лифта. В ходе разработки данного приложения, автор использовал знания полученные во время обучения в университете, а также получен дополнительный опыт в разработке программного обеспечения для мобильных устройств.

Kokkuvõte

Töö käigus on täidetud kõik püstitatud eesmärgid: esmalt on kirjeldatud süsteemi kasutamise stsenaariumid, funktsionaalsed ja mittefunktsionaalsed nõuded, kliendipoolse osa arhitektuur, serveripoolse osa puudutav informatsioon. Seejärel on antud ülevaade push-teavituste tööpõhimõtetest, projektis kasutatud projekteerimise mallidest, andmete hoidmise struktuurist Core Data ning Sockets. On vaadeldud kasutajaliidese loomise põhiprintsiipe, samuti on võrreldud iOS platvormile rakenduste loomiseks kasutatavaid programmeerimiskeeli Swift ja Objective-C. Töö tulemusena on valminud programm, mille ülesandeks on ärihoone külalise identifitseerimine ja lifti kaugväljakutse. Prototüüp on valmis ning autoril on plaanis tulevikus rakenduse edasiarendamine.

Autori tehtud töö näitab, et projekteeritud rakendus on efektiivne vahend ärihoone külalise identifitseerimiseks ning lifti kaugväljakutseks. Rakenduse arendamise jooksul autor kasutas teadmisi, mis on saadud ülikoolis õppimise ajal, ning sai väärtusliku kogemuse edaspidiseks mobiilrakenduste arendamiseks.

Summary

During the work all the aims set were achieved: there were described the precedents of the system, functional and nonfunctional system requirements, architecture of customers part, as well as the general information about server-side. The work also discussed the basis of push notifications, general designing patterns used in the project, structure for data storage – Core Data, and also Sockets. Moreover, there was the discussion provided about the general principles on which customers' graphical interface was based. Also, IOS programming languages, Swift and Objective-C, were compared. As a result, there was a program created for customers' identification and distant lift call. The prototype is ready and further development is assumed.

The work carried through by the author showed that this application is an effective tool for customers' identification and also for distant lift call. During the development of this project, author used knowledge obtained during the university studies, as well as additional experience in development of mobile software.

Список используемой литературы

[1] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/IOS [Дата посещения 23.11.2015]

[2] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/API [Дата посещения 23.11.2015]

[3] «KONE» [В Интернете] Available: http://cdn.kone.com/www.kone.se/Images/kone-remotecall-factsheet.pdf?v=1 [Дата посещения 24.11.2015]

[4] «Компьютерная газета A-Z» [В Интернете] Available: http://www.nestor.minsk.by/kg/2000/06/kg00612.html [Дата посещения 25.11.2015]

[5] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Графический_интерфейс_пользователя [Дата посещения 26.11.2015]

[6] «iMaladec» [В Интернете] Available: http://www.imaladec.com/story/coredata [Дата посещения 26.11.2015]

[7] «GitHub» [В Интернете] Available: https://github.com/chriscdn/RHManagedObject [Дата посещения 26.11.2015]

[8] «Apple» [В Интернете] Available: https://developer.apple.com/library/tvos/documentation/Cocoa/Conceptual/CoreData/FetchingObjects.html [Дата посещения 26.11.2015]

[9] «TokBox» [В Интернете] Available: https://tokbox.com [Дата посещения 27.11.2015]

[10] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Технология_Push [Дата посещения 28.11.2015]

[11] «Habrahabr» [В Интернете] Available: http://habrahabr.ru/post/149077/ [Дата посещения 30.11.2015]

[12] «Habrahabr» [В Интернете] Available: http://habrahabr.ru/post/202960/ [Дата посещения 30.11.2015]

[13] «Cocoapods» [В Интернете] Available: https://cocoapods.org [Дата посещения 03.12.2015]

[14] «Habrahabr» [В Интернете] Available: http://habrahabr.ru/post/261711/ [Дата посещения 03.12.2015]

[15] «Habrahabr» [В Интернете] Available: http://habrahabr.ru/company/mailru/blog/115163/ [Дата посещения 08.12.2015]

[16] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Связка_ключей_iCloud [Дата посещения 08.12.2015]

[17] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Kone [Дата посещения 08.12.2015]

[18] «MacDigger» [В Интернете] Available: http://www.macdigger.ru/iphone-ipod/ios-protiv-android-10-preimushhestv-operacionnoj-sistemy-apple.html [Дата посещения 11.12.2015]

[19] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/SMS [Дата посещения 15.12.2015]

[20] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Анализ_требований [Дата обращения: 15.12.2015]

[21] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/IP-телефония [Дата обращения: 15.12.2015]

[22] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Прецедент_(UML) [Дата обращения: 15.12.2015]

[23] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/HTTPS [Дата обращения: 15.12.2015]

[24] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Операционная_система [Дата обращения: 15.12.2015]

[25] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Шаблон_проектирования [Дата обращения: 19.12.2015]

[26] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Model-View-Controller [Дата обращения: 19.12.2015]

[27] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Сокет_(программный_интерфейс) [Дата обращения: 19.12.2015]

[28] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Опыт_взаимодействия [Дата обращения: 19.12.2015]

[29] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Твердотельный_накопитель [Дата обращения: 19.12.2015]

[30] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Жёсткий_диск [Дата обращения: 20.12.2015]

[31] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/Android [Дата обращения: 20.12.2015]

[32] «Wikipedia» [В Интернете] Available: https://ru.wikipedia.org/wiki/XML [Дата обращения: 20.12.2015]

Дополнение 1

Таблица 1. iPad Core Data

Таблица	Название колонки	Описание	Пример
Person	code	Уникальный код устройства резидента здания	1234vitali
	companyID	Порядковый номер компании	1
	email	Адрес электронной почты резидента здания	vitali@mdigital.ee
	isDefault	Является ли данная персона первой в списке по умолчанию	YES
	juridicalAddress	Адрес компании	Laeva 2
	juridicalName	Имя компании	Mobile Digital OU
	name	Имя и фамилия сотрудника	Vitali Eller
	nickname	Имя отображающиеся в списке сотрудников	Secretary
	phone	Мобильный телефон сотрудника	54320696
	role	Роли резидента здания	Admin
Company	countryCode	Код страны	EE
	id	Порядковый номер компании	1

	juridicalAddress	Адрес компании	Laeva 2
	juridicalName	Имя компании	Mobile Digital OU
	liftGroupName	Имя группы лифтов, к которым у компании есть доступ	Duplex
	postalCode	Почтовый индекс компании	10613
	town	Город	Tallinn

Таблица 2. iPhone Core Data

Таблица	**Название колонки**	**Описание**	**Пример**
Calls	callerName	Имя работника, кому адресован звонок	Vitali Eller
	day	Дата звонка	28.12.2015
	elevatorSended	Был ли отправлен лифт	YES
	forwarded	Был ли перенаправлен вызов секретарю	YES
	logItemID	Порядковый номер записи	1
	roles	Роль сотрудника, кому адрисован вызов	Admin
	time	Время звонка	13:44:12

User	code	Уникальный код устройства резидента здания	1234vitali
	defaultFloor	Этаж по умолчанию	4
	firstName	Имя работника	Vitali
	lastName	Фамилия работника	Eller
	roles	Роли сотрудника	Admin