МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

РАБОТА ПРОВЕРЕНА

Рецензент
генеральный директор
ООО «АнимаРендер»
_____ Ю.В. Ситников
“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор
_____ Л.Б. Соколинский
“___”_____ 2017 г.

# РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ "TABLETIME" НА ПЛАТФОРМЕ ANDROID

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.13-005-1382.ВКР

Научный руководитель
Преподаватель кафедры СП
_____ П.Г. Верман

Автор работы,
студент группы КЭ-401
_____ А.А. Вейс

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

*Заведение* – любое место, в котором можно забронировать столик: кафе, ресторан, бар и т.д.

*Пользователь* – человек, использующий мобильное приложение для бронирования столиков в заведениях.

*Менеджер* – представитель заведения, который вносит и изменяет информацию о заведении, а также обрабатывает заявки на бронирование от пользователей.

*Push-уведомление* – небольшое сообщение от приложения, отображаемое на экране.

## АКТУАЛЬНОСТЬ

В настоящее время вследствие увеличения доступности мобильного Интернета наблюдается тенденция к использованию мобильных приложений для совершения таких действий, как заказ, бронирование или покупка каких-либо товаров и услуг. Это позволяет пользователю избежать необходимости самостоятельно совершать телефонный звонок, который может обернуться длительным ожиданием, недоступностью абонента и тратой денежных средств. Кроме того, не всегда пользователь оказывается в условиях тишины и возможности говорить по телефону, что делает использование мобильного приложения удобнее, чем совершение звонка.

## ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью данной работы является разработка Android-приложения «TableTime», которое предоставляет возможность бронирования столиков в заведениях города Челябинск.

Приложение «TableTime» – это приложение, которое позволяет пользователю забронировать столик в заведении из списка заведений, предоставляемого приложением. При изменении статуса брони пользователь получает соответствующее уведомление.

Для разработки приложения необходимо решить следующие задачи:

– выполнить анализ предметной области;
– спроектировать и реализовать сервер системы;
– спроектировать и реализовать Android-приложение;
– провести тестирование.

ОБЗОР ЛИТЕРАТУРЫ

По ссылкам [4–10] представлены проекты, аналогичные разрабатываемому. В работе [3] обоснована целесообразность сотрудничества заведений с подобными приложениями с точки зрения маркетинга. В ресурсах [1, 2, 12, 17, 23, 26] описаны документации систем управлений базами данных, сравнение ORM-решений и механизмы миграций схем баз данных. Ресурсы [11, 13, 14, 21] предоставляют информацию о разработке под платформу Android. По ссылкам [18, 24] описаны методы обеспечения безопасности данных пользователей. По ссылке [15] представлена документация веб-сервера Django, в ссылке [22] – платформы OpenShift, а в [27] – поставщика хостинг-услуг FirstVDS. Ресурсы [16, 20, 25] описывают сервисы, предоставляющие услуги рассылки сообщений пользователям.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, четырех глав, заключения, библиографии и приложения. Объем работы составляет 38 страницы, объем библиографии – 27 источников, объем приложения – 4 страницы.

СОДЕРЖАНИЕ РАБОТЫ

Первая глава «Анализ предметной области» содержит постановку задачи и обзор аналогичных проектов.

Вторая глава «Проектирование системы» содержит описание и анализ требований к мобильному приложению и серверу системы «TableTime», описание архитектур приложения и сервера, а также описание схем баз данных приложения и сервера.

Третья глава «Реализация системы» описывает подробности реализации сервера системы «TableTime» и мобильного приложения для платформы Android.

Четвертая глава «Тестирование системы» посвящена результатам тестирования сервера системы «TableTime» и мобильного приложения для платформы Android.

В заключении описываются основные результаты, полученные при выполнении дипломной работы.

В приложении представлен графический интерфейс пользователя мобильного приложения.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Постановка задачи

Внедрение информационных технологий в сферу ресторанного бизнеса необходимо для решения следующего спектра задач [2]:

– поиск заведения и предоставление информации о заведении: адрес, телефон, часы работы, меню;

– своевременное уведомление пользователей о появлении новых заведений;

– своевременное уведомление пользователей об изменении заведением адреса, времени работы, ценовой политики и т.д.;

– своевременное уведомление пользователей о появлении новых акций и специальных предложений;

– предоставление услуги бронирования столиков;

– автоматизация контроля бронирования столиков;

– автоматизация заказа блюд;

– заказ доставки блюд.

Для заведений участие в проектах, подобных разрабатываемому приложению, имеет как большой недостаток, так и немалую выгоду. Недостаток заключается в том, что сотрудникам необходимо тратить часть рабочего времени на поддержание актуальности предоставляемой информации, т.е. вовремя вносить изменения через интерфейс администратора заведения или сообщать о них технической поддержке проекта. К тому же необходимо следить за новым источником заявок на бронирование столиков. В то же время маркетинговые исследования показывают, что предоставление информации о заведении его официальными источниками и тем более сторонними сервисами является эффективным воздействием на потребителя, повышающим конкурентоспособность заведения [3].

Для обеспечения взаимодействия желающего забронировать столик пользователя и менеджера соответствующего заведения, система «TableTime» должна включать в себя приложения для клиентов под различные

платформы, сервер системы и приложение для менеджеров. В рамках данной работы будут реализованы сервер системы и мобильное приложение под платформу Android.

## 1.2. Обзор аналогов

### 1.2.1. RestOn - бронирование столиков

«RestOn» – одно из первых предлагаемых приложений в магазине Google Play по запросу «бронирование столиков». В описании приложения сказано, что оно предоставляет описание заведений, фотографии заведения, отзывы о заведении, карту, а также предлагает функционал прокладывания маршрута до заведения и брони столика в нем [9]. При входе в приложение открывается форма регистрации, в которой требуется указать номер телефона, электронный ящик, имя и фамилию.

Значительным недостатком приложения является ненадежность системы регистрации, в которой пользователь может не получить смс-сообщение с кодом для подтверждения регистрации.

### 1.2.2. EatOut

С помощью приложения «EatOut» можно посмотреть фотографии заведения и отзывы о нем, проложить маршрут до заведения и забронировать столик [7]. Приложение предоставляет фильтр заведений по широкому спектру параметров, например, по типу заведения и по среднему чеку. Возможность просмотра информации в режиме отсутствия подключения к Интернету не предусмотрена.

### 1.2.3. Toptable

Приложение «Toptable» предоставляет возможности фильтрации списка заведений по типам заведений, поиска заведения по названию, составления списка избранных заведений, бронирования столика [10]. По информации из описания приложения, бронирование столика подтверждается звонком администратора заведения. Какая информация о заведении доступна пользователю осталось неизвестным, потому что список доступных заведений оказался пустым.

### 1.2.4. Столики

В приложении «Столики» можно искать заведение по названию, составлять список избранных заведений, бронировать столик [5]. Загрузка приложения шла достаточно долго, но в итоге, как и Toptable выдало пустой список заведений.

Возможность просмотра информации в режиме отсутствия подключения к Интернету не предусмотрена.

### 1.2.5. ТоМесто

Из приложения «ТоМесто» можно узнать следующую информацию о заведениях: фотографии, меню, отзывы, режим работы, описание, похожие места и т.д. Данное приложение предоставляет возможности поиска заведений и бронирования столиков [6].

При просмотре подробной информации о заведении приложение выводит уведомление о том, сколько человек сейчас просматривает эту страницу, и сколько человек забронировали столик за последний час. Уведомления перекрывают доступ к некоторым элементам управления. Вдобавок периодически появляется уведомление с текстом «не удалось определить ваше местоположение», несмотря на то, что приложение его корректно определило.

Возможность просмотра информации в режиме отсутствия подключения к Интернету не предусмотрена.

### 1.2.6. Афиша–Рестораны

Приложение «Афиша-рестораны» предоставляет категории и фильтры для поиска заведений, их адреса, рейтинги, ценовые категории, обзоры, рекомендации, телефоны, места рядом и т.д. В приложении доступна услуга бронирования столиков, имеются разделы новостей ресторанов, истории бронирования и поиска [4]. Приложение обладает удобным современным и интуитивно понятным интерфейсом.

Возможность просмотра информации в режиме отсутствия подключения к Интернету не предусмотрена.

### 1.2.7. Lipe

Приложение [8] предоставляет множество фильтров для поиска заведений, например, по названию, городу, кухне, типу, ценовой категории, режиму работы, способу оплаты и т.д. При выборе фильтров каждый раз необходимо выбирать желаемый город из неупорядоченного списка городов, что делает работу с приложением медленной. В приложении доступны описания заведений, меню по категориям, отзывы пользователей и т.д.

При любом переходе между экранами приложение ненадолго зависает, подгружая информацию. Восприятие предоставляемой приложением информации затруднено из-за нарушения композиции структурных элементов экранов. В приложении отсутствует возможность просматривать информацию в приложении без подключения к Интернету.

## 1.3. Вывод

Из рассмотренных приложений для города Челябинск актуальными являются приложения «Афиша–Рестораны» и «ТоМесто».

Обзор аналогов показал, что в первую очередь стоит обратить внимание на надежность сервисов, используемых для регистрации новых пользователей.

Пользователю должно быть понятно, почему в приложении нет ни одного доступного заведения, если список заведений оказался пустым, т.е. нужно явно дифференцировать поведение приложения при действительно отсутствии доступных заведений, при недоступном сервере приложения и при случайном единичном сбое в работе системы.

В разрабатываемом приложении должна быть разрешена работа без подключения к Интернету с заранее загруженной информацией.

Кроме того, приложение должно обладать удобным и интуитивно понятным интерфейсом и дизайном, обеспечивающим комфортную работу с приложением.

# 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 2.1. Функциональные требования

Разрабатываемая система для бронирования столиков должна удовлетворять следующим функциональным требованиям:

– система должна предоставлять пользователю возможность регистрации и авторизации с нескольких устройств;

– система должна обеспечивать синхронизацию данных на разных устройствах пользователя;

– система должна позволять пользователю просматривать без подключения к Интернету заранее загруженную информацию;

– система должна предоставлять пользователю возможность менять свои регистрационные данные;

– система должна предоставлять пользователю информацию о заведениях, схему зала заведения, фотографии, меню и акции заведений;

– система должна предоставлять пользователю возможность поиска блюда по названию или по тегам;

– система должна позволять пользователю создавать заявки на бронирование (с указанием даты, времени, столика, произвольного комментария) и отменять их;

– система должна предоставлять менеджеру возможность обработки пользовательских заявок на бронирование: принять или отклонить заявку, или позвонить пользователю на указанный номер телефона;

– система должна позволять менеджеру изменять информацию о заведении, об акциях, о меню.

## 2.2. Нефункциональные требования

В результате анализа функциональных требований и обзора аналогичных проектов, были сформулированы следующие нефункциональные требования:

– приложение должно быть написано на языке Java под платформу Android.

– сервер системы должен иметь REST интерфейс;

– сервер системы должен быть написан на языке Python с использованием платформы Django;

### 2.3. Варианты использования системы

Диаграмма вариантов использования мобильного приложения представлена на рис. 1. С приложением взаимодействуют два актера – неизвестный пользователь и авторизованный пользователь. Неизвестный пользователь – это человек, использующий мобильное приложение, не пройдя процедуру авторизации или регистрации. Авторизованный пользователь – это человек, который успешно прошел процедуру авторизации или регистрации.

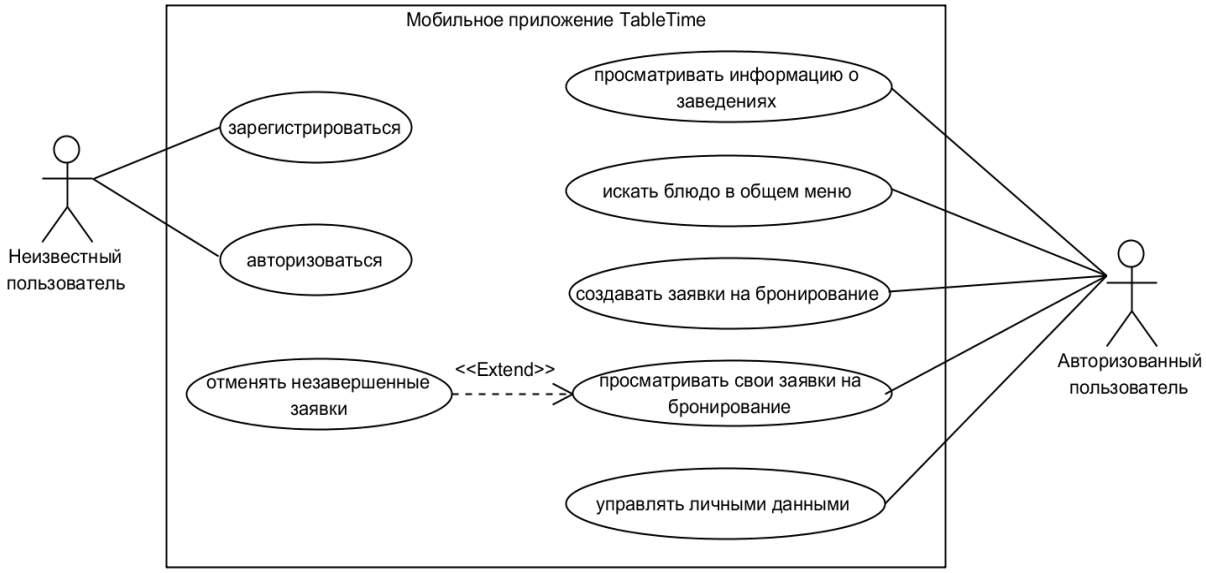


**Рис. 1.** Варианты использования мобильного приложения «TableTime».

Неизвестный пользователь может *зарегистрироваться в приложении,* указав имя, номер телефона и электронный адрес.

Неизвестный пользователь может *авторизоваться в приложении* через номер телефона или электронный адрес.

Авторизованный пользователь может *посмотреть информацию о заведении,* т.е. его адрес, телефон, дни и часы работы, схему зала, меню, фотографии, акции.

Авторизованный пользователь может *искать блюдо в общем меню,* т.е. получить список блюд, найденных по названию или по тегам в меню различных заведений.

Авторизованный пользователь может *создавать заявки на бронирование* с указанием даты, времени, количества персон, столика (по желанию) и произвольного комментария (по желанию).

Авторизованный пользователь может *просматривать свои заявки на бронирование,* отслеживать их статус. Если заявка еще не обработана, или уже принята, но еще не завершена, пользователь может *отменить незавершенную заявку*.

Авторизованный пользователь может *управлять личными данными*, т.е. изменять номер телефона, имя, электронный адрес.

Диаграмма вариантов использования сервера системы представлена на рис. 2. С сервером взаимодействуют три актера: клиентское приложение, приложение для пользователя и приложение для менеджера. Клиентское приложение – приложение, которое обратилось к серверу, но еще не было опознано им как приложение для пользователя или менеджера.

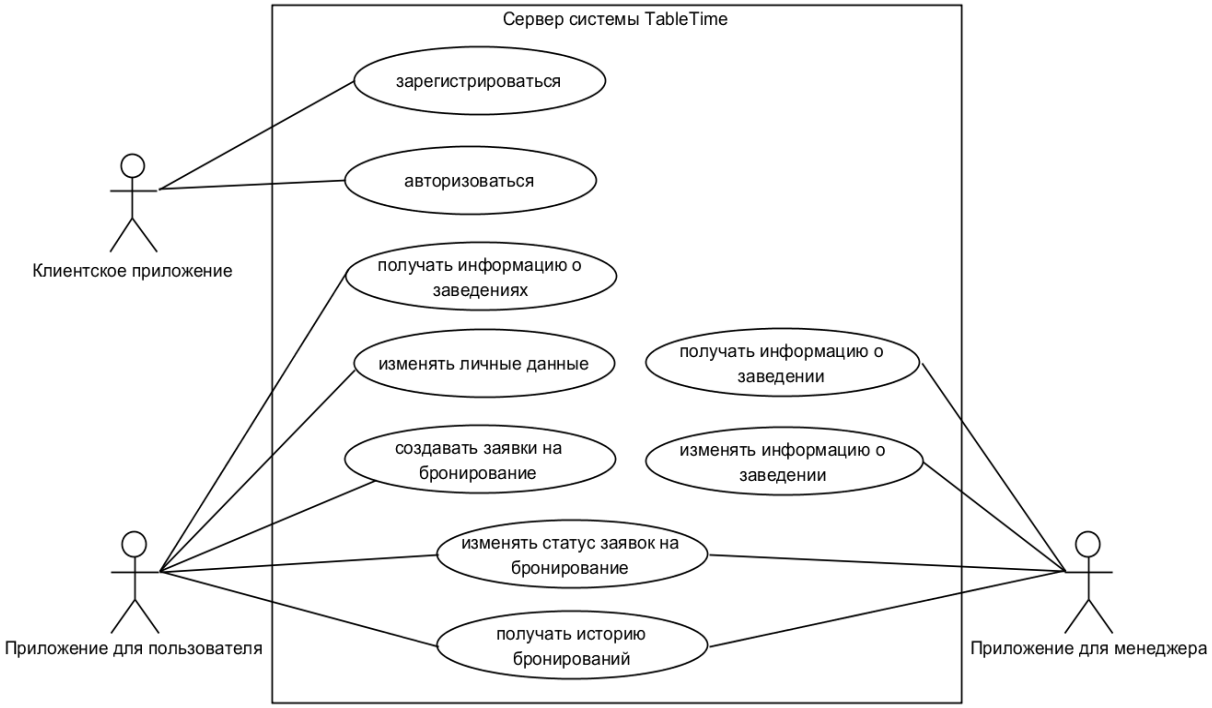


**Рис. 2.** Варианты использования сервера системы «TableTime»

Клиентское приложение может *авторизоваться* или *зарегистрироваться.* Если клиентское приложение попытается зарегистрироваться как приложение для менеджера, ему вернется ответ с сообщением о том, что приложения для менеджеров регистрирует администратор системы.

Приложение для пользователя может *получать информацию о заведениях,* т.е. общую информацию, акции и меню.

Приложение для пользователя может *изменять личную информацию*: номер телефона, имя и электронный адрес.

Приложение для пользователя может *создавать заявки на бронирование* столика в определенном заведении.

Приложение для менеджера может *получать информацию о заведении*, т.е. общую информацию, акции и меню заведения, в котором работает менеджер.

Приложение для менеджера может *изменять информацию о заведении*, добавлять, изменять и удалять акции и фотографии заведения, редактировать пункты меню.

Оба приложения могут *изменять статус заявок на бронирование,* приложение менеджера может установить один из следующих статусов: «принята», «отклонена», «завершена» или «клиент не пришел». Приложение пользователя может установить заявке статус «отменена».

Оба приложения могут *получать историю бронирований*: приложение пользователя получает историю бронирований пользователя, а приложение менеджера получает историю бронирований заведения.

### 2.4. Архитектура разрабатываемой системы

#### 2.4.1. Архитектура сервера системы

На рис. 3 изображена диаграмма компонентов сервера системы «TableTime».

*Диспетчер URL* выполняет привязку URL, по которому к серверу обратилось клиентское приложение, к функции из обработчика запросов приложения пользователя или менеджера.

*Обработчик запросов приложения пользователя* и *обработчик запросов приложения менеджера* содержат логику сбора, преобразования и компоновки данных, предоставленных (или запрошенных) приложением пользователя или менеджера соответственно.

*Сервис авторизации* – компонент, предоставляющий средства принятия решения о том, имеет ли клиентское приложение право на выполнение запрошенного действия.

*Компонент рассылки СМС-сообщений* генерирует тексты СМС-сообщений и содержит конфигурацию для использования стороннего сервиса СМС-рассылок.

*Компонент рассылки электронных писем* генерирует тексты электронных писем и содержит конфигурацию для использования стороннего сервиса рассылки электронных писем.

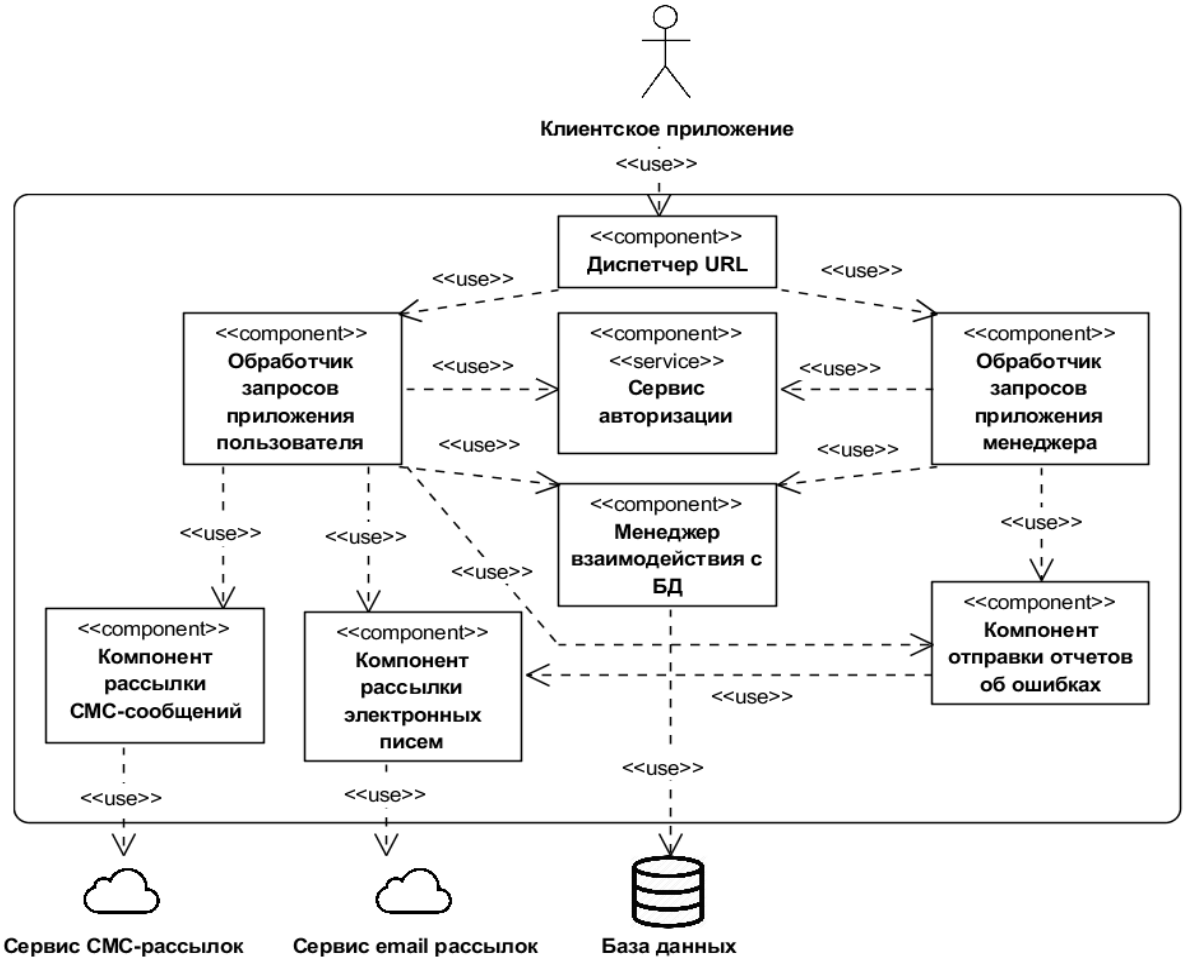


**Рис. 3.** Компоненты сервера системы

*Компонент отправки отчетов об ошибках* – компонент, который в случае некорректного поведения системы создает запись в файле логов, формирует текст отчета об ошибке и при помощи компонента рассылки электронных писем отправляет сформированный отчет разработчику.

*Менеджер взаимодействия с базой данных* отвечает за взаимодействие с базой данных, а также содержит конфигурационные данные для соединения с базой данных. Кроме того, в этом компоненте определены модели ORM (объектно-реляционного проектора).

### 2.4.2. Схема базы данных сервера системы

На рис. 4 представлена схема базы данных сервера системы.

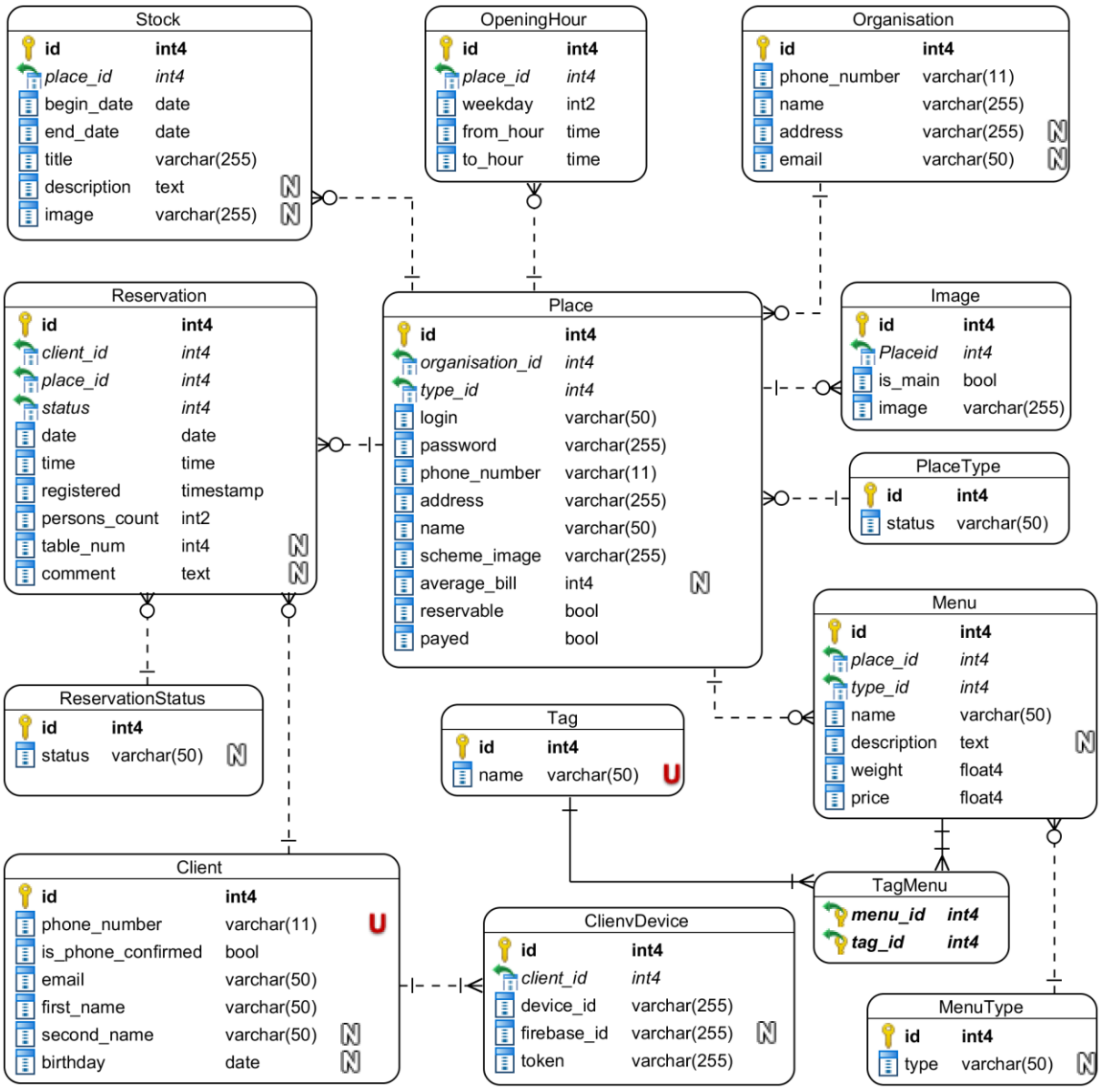


**Рис. 4.** Схема базы данных сервера системы

База данных состоит из следующих таблиц:

– *client_device* – таблица, хранящая данные об устройствах пользователей;

– *client* – таблица, хранящая данные о пользователях приложений;

– *reservation* – таблица, в которой содержатся данные о заявках пользователей на бронирование;

– *reservation_status* – таблица-перечисление, хранящая возможные статусы заявок на бронирование;

– *tag* – таблица, хранящая теги для пунктов меню;

– *menu* – таблица для хранения данных о пунктах меню заведений;

– *tag_menu* – таблица, порожденная связью «многие-ко-многим» между таблицами tag и menu;

– *menu_type* – таблица-перечисление, хранящая категории пунктов меню;

– *place* – таблица, которая хранит данные о заведениях;

– *place_type* – таблица-перечисление, хранящая типы заведений;

– *organisation* – таблица, которая хранит данные об организации-владельце заведения;

– *opening_hour* – таблица, хранящая данные о времени работы заведений;

– *image* – таблица, в которой содержатся URL-пути для доступа к фотографиям заведений;

– *stock* – таблица для хранения специальных предложений заведений.

### 2.4.3. Архитектура Android-приложения

На рис. 5 изображена диаграмма компонентов приложения.

Компонент *Интерфейс пользователя* предназначен для взаимодействия с системой.

*Компонент заведений* предоставляет информацию о заведениях и отвечает за поиск заведений по названию или типу.

*Компонент бронирований* позволяет получать историю бронирований, создавать новые заявки на бронирование, менять статус незавершенных заявок.

*Компонент информации о пользователе* позволяет просмотреть и изменить регистрационные данные пользователя.

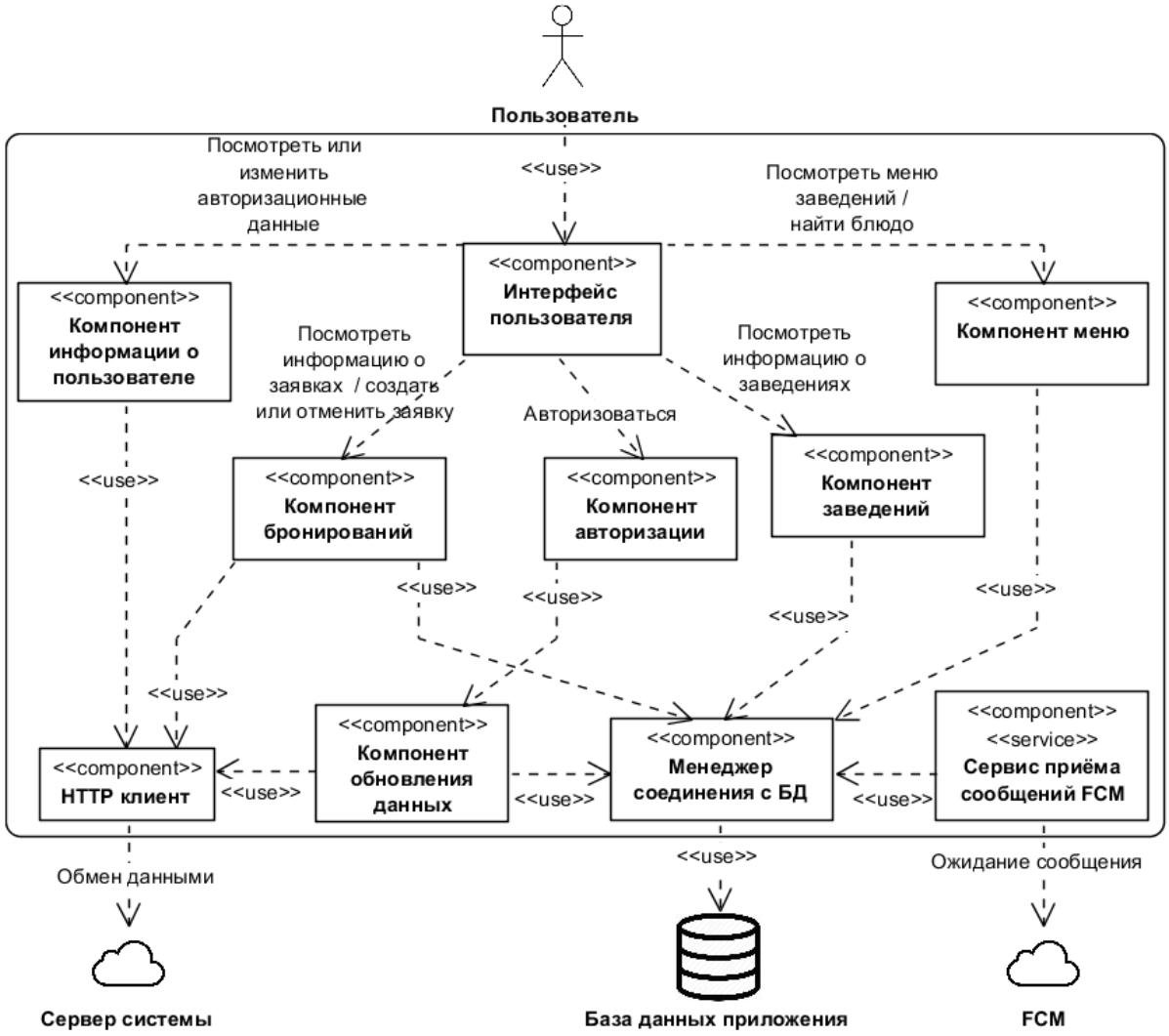


**Рис. 5.** Компоненты Android-приложения

*Компонент меню* предоставляет меню заведения по категориям, а также отвечает за поиск желаемого блюда по названию или тегу по меню всех заведений.

*Компонент авторизации* определяет необходимость авторизации и предоставляет возможность регистрации или авторизации при необходимости.

*Компонент обновления данных* обновляет данные в базе данных приложения, если устройство авторизовано, сервер системы доступен, и прошло определенное время после предыдущего обновления данных.

*HTTP клиент* реализует взаимодействие с сервером системы посредством обращения к REST-интерфейсу сервера.

*Менеджер соединения с базой данных* содержит конфигурационные данные для соединения с базой данных, скрипт инициализации базы данных, а также скрипт создания схемы базы данных.

*Сервис приема сообщений FCM* обрабатывает приходящие сообщения от сервиса Firebase Cloud Messaging *(FCM)* и оповещает пользователя о полученных сообщениях. *Firebase Cloud Messaging* – это бесплатный сервис для отправки сообщений с серверов в приложения для устройств Android, iOS и Chrome [16]. В системе «TableTime» FCM используется для ускорения обмена информацией о заявках на бронирование.

### 2.4.4. Схема базы данных Android-приложения

На рис. 6 представлена схема базы данных Android-приложения.

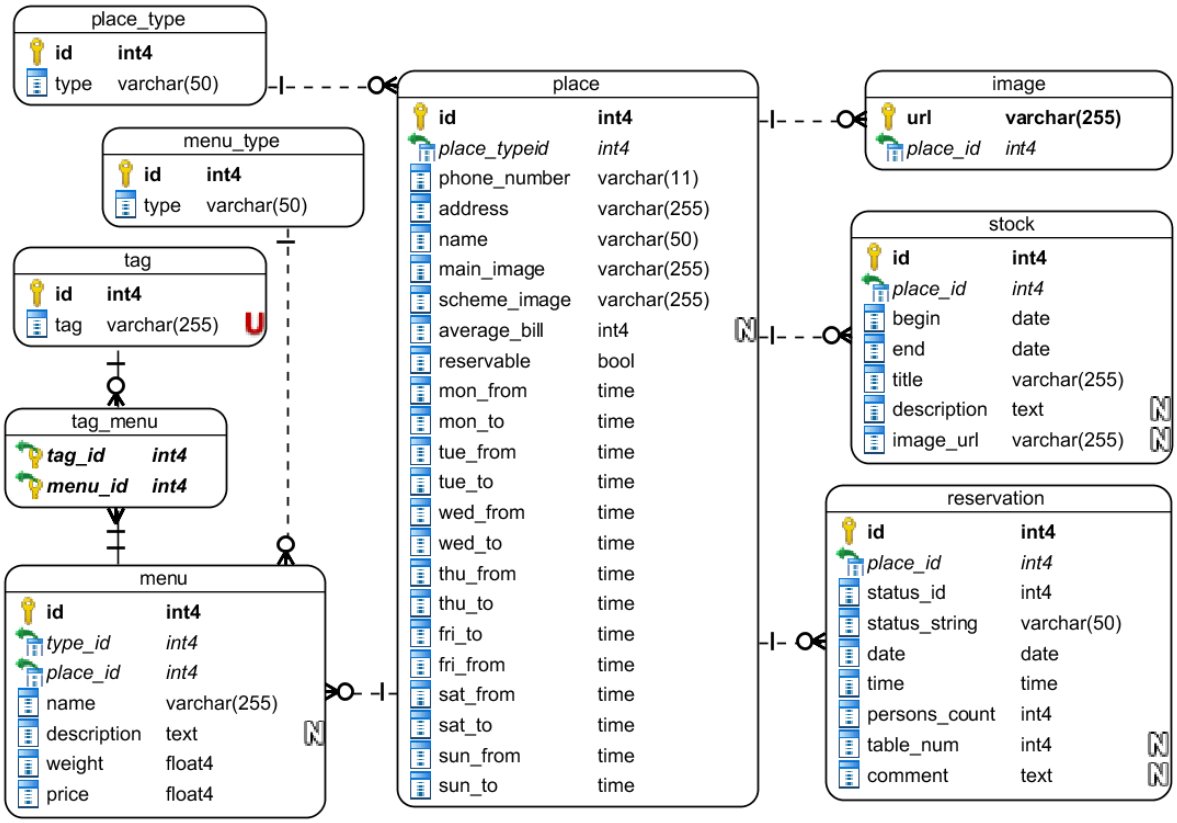


**Рис. 6.** Схема базы данных приложения

Семантика таблиц локальной базы данных в мобильном приложении соответствует семантике одноименных таблиц базы данных сервера системы.

## 2.5. Вывод

В соответствии с требованиями была спроектирована архитектура системы «TableTime». Кроме того, были спроектированы базы данных сервера системы и мобильного приложения под платформу Android.

# 3. РЕАЛИЗАЦИЯ СИСТЕМЫ

## 3.1. Инструменты реализации

Серверная часть системы реализована на высокоуровневом языке программирования *Python* версии 3.5 с использованием веб-фреймворка Django. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное и функциональное. *Django* – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования, похожий на MVC. Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а логика Представления реализуется в Django уровнем Шаблонов (Template). Из-за этого архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV) [15]. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

На сервере данные хранятся в базе данных, созданной с помощью СУБД PostreSQL [23], т.к. она свободно распространяется и интегрируется с фреймворком Django.

Клиентская часть системы реализована для платформы Android, являющейся наиболее популярной мобильной операционной системой в мире [14] на высокоуровневом объектно-ориентированном языке Java. Для локального хранения данных используется база данных, созданная с помощью СУБД SQLite [26], поддержка которой встроена в платформу Android.

Взаимодействие клиентского приложения и сервера реализовано посредством HTTP-запросов от приложения к серверу и FCM-сообщений от сервера.

Пример HTTP-запроса информации о заведениях показан на рис. 7. В ответе на запрос приходит строка в формате JSON, содержащая описание массива объектов заведений.

```
GET /api/places/
Headers: JWT: <JSON Web Token >
Response:
[{
  "reservable": true,
  "id": 5,
  "address": "Академика Королева, 13",
  "scheme_image": "9d2ecead-da33-408b-b8dc-28fa198726e4.jpg",
  "main_image": "34bfc7e4-3609-4074-880f-663222abbfec.jpg",
  "images": [
    "34bfc7e4-3609-4074-880f-663222abbfec.jpg",
    "1115c577-f715-4954-bde5-d3180427c1c9.jpg"],
  "phone_number": "89193501223",
  "opening_time": [{
      "from_hour": "10:00 ",
      "weekday": 1,
      "to_hour": "24:00"
    },
    …
  ],
  "average_bill": null,
  "stocks": []
  "name": "Корizza",
  "type_id": 5
}]
```

**Рис. 7.** Пример запроса на получение информации о заведениях

## 3.2. Реализация сервера системы

### 3.2.1. Работа с базой данных

Работа с базой данных осуществляется при помощи Django ORM, создание и изменение таблиц происходит посредством использования механизма миграций [12], генерируемых Django, по которым создаются SQL-запросы на создание и изменение таблиц базы данных.

### 3.2.2. Отправка сообщений пользователям

При регистрации пользователь указывает номер телефона и адрес электронной почты. На указанный номер телефона высылается сгенерированный сервером код авторизации – строка длиной в 7 символов, состоя-

щая из цифр и прописных букв латинского алфавита. При авторизации в приложении уже зарегистрированного пользователя процедура аналогична: код авторизации высылается на указанный номер телефона или адрес электронной почты.

СМС-рассылка и Email-рассылка осуществляются с использованием сторонних сервисов SMSЦентр [25] и SendGrid [20].

Когда менеджер изменяет статус заявки, пользователю отправляется FCM-сообщение, в котором передаются следующие данные: идентификатор заявки, статус которой изменился, и код нового статуса.

В случае ошибки при отправке любого типа сообщения, сервер делает соответсвующую запись в файле логов и пытается отправить разработчику электронное сообщение об ошибке.

### 3.2.3. Безопасность данных пользователей и заведений

Пароли заведений и коды авторизации пользовательских устройств не хранятся в системе в открытом виде. В базе данных хранятся результаты вычисления криптографической хеш-функции bcrypt от пароля или кода. *bcrypt* – адаптивная криптографическая функция формирования ключа, используемая для защищенного хранения паролей [24]. Для защиты от атак с помощью радужных таблиц bcrypt использует *соль*, т.е. произвольную строку данных, добавляемую к строке пароля.

В случае успешной аутентификации клиентского приложения, сервер высылает приложению сгенерированный JWT, используемый в дальнейшем для авторизации клиентского приложения. *JWT (JSON Web Token)* – это открытый стандарт для обеспечения безопасности передачи пакетов между сторонами [18]. JSON Web Token дословно переводится как веб маркер в формате JSON. Токен (маркер) передается в зашифрованном виде и состоит из трех частей, разделенных точками.

1. Закодированный заголовок, содержащий информацию о типе стандарта («JWT») и об алгоритме шифрования (например, «HS256»).

2. Тело токена, которое содержит метаданные. В приложении «TableTime» тело токена содержит тип клиентского приложения (приложение для пользователя или приложение для менеджера), идентификатор клиентского приложения и дату истечения токена, после которой токен будет считаться недействительным.

3. Цифровая подпись, которая содержит ключ для шифрования всех частей токена.

### 3.2.4. Развертывание тестового и рабочего серверов

Тестовый сервер приложения размещен на облачном хостинге OpenShift, разработанном компанией Red Hat. *OpenShift* – платформа-сервис (PaaS), поддерживающая множество языков и фреймворков, в том числе язык Python 3 и фреймворк Django [22].

В системе «TableTime» происходит сбор персональных данных пользователей: имя, фамилия, телефон, адрес электронной почты, дата рождения пользователя. По законодательству Российской Федерации эти данные должны храниться на территории государства. Поэтому для развертывания рабочего сервера было решено арендовать виртуальный выделенный сервер у российского поставщика хостинг-услуг FirstVDS [27].

В ходе развертывания тестового и рабочего серверов были написаны конфигурационные файлы для веб-сервера nginx (только рабочий сервер) и сервера приложений Django. На оба сервера были установлены зависимости проекта, перечисленные на рис. 8.

```
Bcrypt>=3.1.0
cffi
Django>=1.8.12
Pillow
psycopg2
pycparser
six
sendgrid==1.2
django-cors-headers
pyjwt
rsa
requests
```

**Рис. 8.** Листинг списка зависимостей

### 3.3. Реализация Android-приложения

#### 3.3.1. Поддержка разных версий платформы Android

В каждом Android-проекте необходимо обозначить минимальную поддерживаемую приложением версию платформы Android. Чем меньше версия, тем на большее число устройств будет возможна установка приложения, но будет невозможна или ограничена работа с некоторыми возможностями API более поздних версий платформы [11].

Согласно официальному сайту платформы Android [13], доля Android-устройств версии 4.1 и выше составляет 98,2 % на момент начала апреля 2017 года. Поэтому в качестве минимальной поддерживаемой версии было решено выбрать версию 4.1 (Jelly Bean, API 16).

Для корректной работы приложения на устройствах с разными версиями API были разработаны различные версии некоторых компонентов интерфейса (см пункт 3.3.2).

#### 3.3.2. Реализация пользовательского интерфейса

Дизайн приложения разрабатывался с опорой на принципы Material Design. *Material Design* представляет собой комплексную концепцию создания визуальных, движущихся и интерактивных элементов для различных платформ и устройств от компании Google [21]. Скриншоты пользовательского интерфейса представлены в приложении.

Для навигации по основным экранам приложения используется паттерн *Navigation Drawer* – меню, выдвигающееся с левой стороны экрана и содержащее основные пункты навигации по приложению.

Некоторые возможности Material Design, например, определенные виды анимации и ряд XML-атрибутов элементов доступны только в Android с уровнем API не меньше 21. Для обеспечения совместимости приложений с устройствами под управлением более ранних версий платформы Android создаются разные файлы ресурсов для версий платформы с уровнем API меньше 21 (в каталогах values/, drawable/ и т.д.) и с уровнем 21 и выше (в каталогах values-v21/, drawable-v21/ и т.д.).

Рассмотрим вышеописанный подход на примере определения конкретного стиля *raisedButton*, используемого в приложении для стилизации элементов управления. На рис. 9 приведено определение стиля *raisedButton* для платформы Android с версией API ниже 21, на рис. 10 – с версией 21 и выше. Ресурсы, используемые для определения типа изменения внешнего вида элемента при взаимодействии пользователя с ним, приведены на рис. 11 и рис. 12 соответственно для вышеописанных стилевых файлов.

```
<style name="raisedButton" parent="@style/Widget.AppCompat.Button.Colored">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_gravity">center_horizontal</item>
    <item name="android:background">@drawable/button_selector</item>
</style>
```

**Рис. 9.** Определение стиля raisedButton в файле values/styles.xml

```
<style name="raisedButton" parent="@style/Widget.AppCompat.Button.Colored">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_gravity">center_horizontal</item>
    <item name="android:background">@drawable/ripple</item>
</style>
```

**Рис. 10.** Определение стиля raisedButton в файле values-v21/styles.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/accent_background_dark"
        android:state_pressed="true" />
    <item android:drawable="@drawable/accent_background_dark"
        android:state_focused="true" />
    <item android:drawable="@drawable/accent_background" />
</selector>
```

**Рис. 11.** Листинг drawable/ button_selector.xml

```
<ripple android:color="@color/colorText"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <selector>
            <item android:drawable="@drawable/accent_background_dark"
                android:state_pressed="true" />
            <item android:drawable="@drawable/accent_background_dark"
                android:state_focused="true" />
            <item android:drawable="@drawable/accent_background" />
        </selector>
    </item>
</ripple>
```

**Рис. 12.** Листинг drawable-v21/ripple.xml

### 3.3.3. Работа с базой данных

В процессе реализации мобильного приложения было решено отказаться от использования ORM-библиотек в пользу использования класса-помощника СУБД SQLite *SQLiteOpenHelper* и прекомпилированных SQL-выражений *SQLiteStatement*. Решение обусловлено тем, что приложение поддерживает большой спектр версий платформы Android, тогда как работа некоторых ORM-библиотек нестабильна на платформах с версией API меньше, чем 21 [17]. Кроме того, согласно источнику [1], сохранение и чтение данных происходит быстрее без использования ORM, чем с использованием любого ORM решения, а в реализуемом приложении одной из самых частых операций с базой данных является сохранение большого объема данных, полученных от сервера системы.

### 3.3.4. Прием сообщений сервиса Firebase Cloud Messaging

Когда на сервер системы поступает информация о том, что менеджер изменил статус заявки клиента, на все устройства клиента с помощью сервиса FCM отправляется сообщение с информацией о новом статусе заявки, после чего приложение выводит на экран push-уведомление.

Всего может быть 3 варианта того, как и в каком виде информация об изменении в заявке появится на устройствах клиента.

1. Если в очереди сообщений для пользователей системы находится не больше 100 сообщений, устройства пользователя получают сообщение, содержащее идентификатор заявки, статус которой изменился, и код нового статуса.

2. Если в очереди больше 100 сообщений, устройства пользователя получают сообщение о том, что им были высланы FCM-сообщения, но самих сообщений устройства не получают. В таком случае приложение обновляет статусы незавершенных заявок посредством HTTP-запроса к серверу.

3. Если произошла непредвиденная ситуация (сбой в системе «TableTime», или в сервисе FCM), и на протяжении долгого времени пользова-

тель не получает информацию о состоянии его заявок, то пользователь может запросить принудительное обновление статусов заявок, которое реализовано HTTP-запросом к серверу.

### 3.4. Изменение заявки на бронирование

На рис. 13 через взаимодействие компонентов системы показан процесс изменения заявки на бронирование.

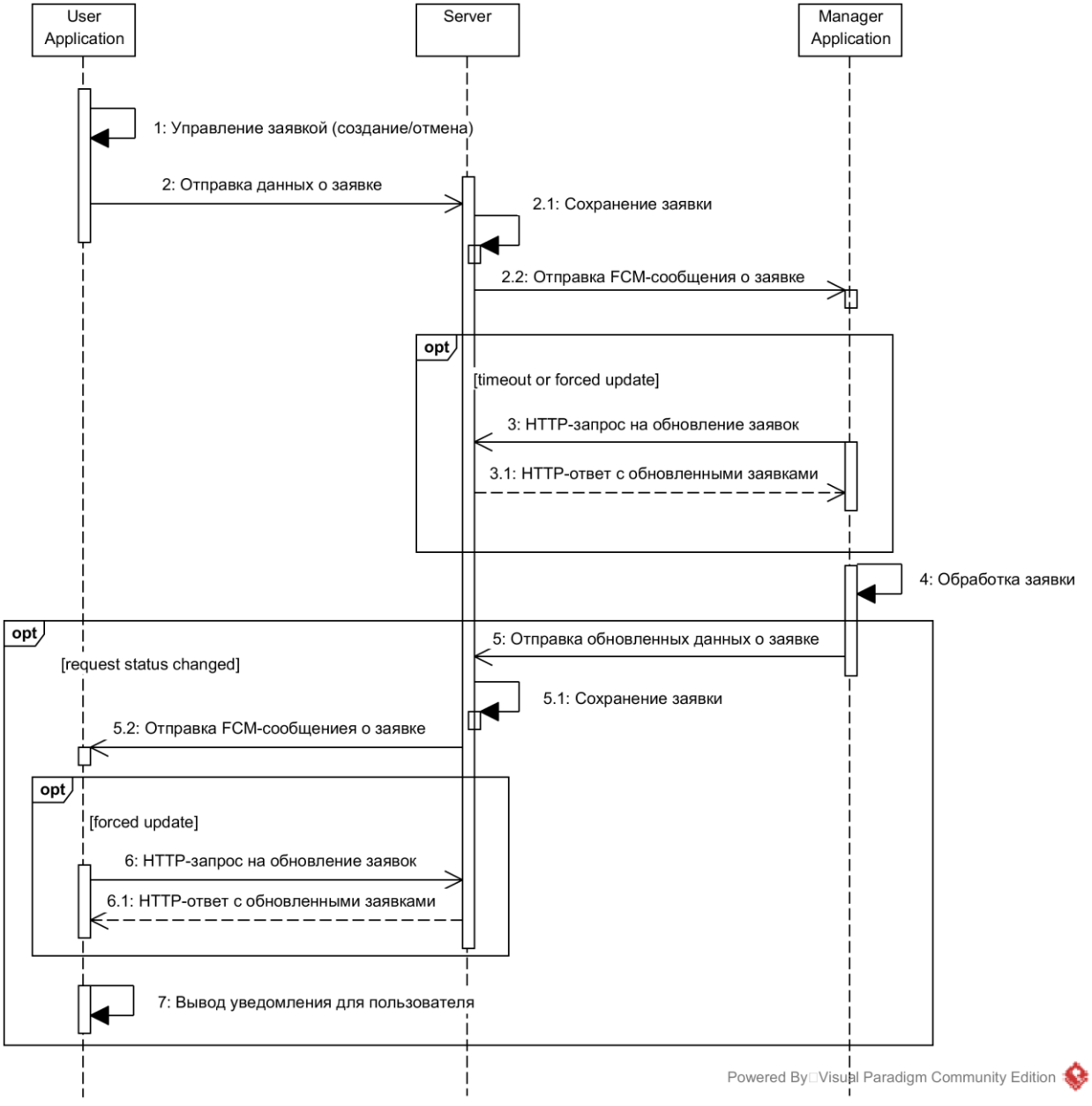


**Рис. 13.** Диаграмма последовательности при изменении заявки

Как только пользователь совершает какое-либо действие со своими заявками (создает новую или отменяет уже существующую), приложение

отправляет данные о созданной или измененной заявке на сервер. Сервер сохраняет полученные данные в базу данных и отправляет приложению менеджера FCM-сообщение с информацией о заявке. В приложении менеджера происходит обработка заявки, и, если в результате обработки ее статус изменился, приложение отправляет данные о заявке на сервер системы. Сервер сохраняет данные и отправляет FCM-сообщение с информацией о заявке приложению пользователя. Приложение пользователя выводит push-уведомление с информацией о смене статуса заявки. Со стороны обоих приложений предусмотрено принудительное обновление статусов заявок посредством HTTP-запросов при долгом отсутствии FCM-сообщений.

### 3.5. Вывод

На основе разработанной архитектуры были реализованы сервер системы «TableTime» и мобильное приложение под платформу Android. Взаимодействие сервера и мобильного приложения реализовано посредством HTTP-запросов и FCM-сообщений.

# 4. ТЕСТИРОВАНИЕ СИСТЕМЫ

Для тестирования системы применялось функциональное тестирование, т.е. тестирование программного обеспечения в целях проверки реализуемости функциональных требований.

Тестирование Android-приложения проводилось как вручную, так и с использованием модульных тестов.

Модульные тесты в Android можно разделить на два типа:

– локальные модульные тесты – тесты, для выполнения которых используется только виртуальная машина Java (JVM). Они предназначены для тестирования бизнес-логики, которая не взаимодействует с операционной системой Android;

– инструментальные модульные тесты – тесты, с помощью которых тестируется логика, использующая Android API. Их выполнение происходит на физическом устройстве или эмуляторе.

Большая часть логики приложения связана со взаимодействием с API системы Android, поэтому по большей части были использованы инструментальные модульные тесты.

Для модульного тестирования использовалась библиотека JUnit [19]. Библиотека предназначена для тестирования программ на языке Java и имеет большое число возможностей: разделение данных и логики теста, группировка тестов, использование аннотаций для описания тестов, обработка исключений, возникших в тесте, и т.д.

Поскольку в процессе реализации приложения было принято решение отказаться от ORM-библиотек, для работы с базой данных было написано большое количество кода, в котором легко допустить ошибки, связанные с использованием прекомпилированных SQL-выражений. В связи с этим было решено уделить особое внимание тестированию компонента, предназначенного для работы с базой данных. Пример модульного теста, проверяющего правильность вставки и чтения данных о заведении приведен на рис. 14.

В приложении часто используется вывод одних и тех же данных в различных форматах. Пример тестирования представления даты представлен на рис. 15.

В табл. 1 приведен протокол ручного тестирования некоторых аспектов работы системы.

```
@Test
    public void insertingPlace_isCorrect(){
        Place place = new Place(
                "Тестовая организация",
                "+70000000000",
                "Кафе",
                1,
                "пр-т Ленина, 75",
                "imagename.jpg");
        List<OpenTime> openTime = new ArrayList<>();
        openTime.addAll(Arrays.asList(
                new OpenTime(1, "08:00:00", "23:00:00"),
                new OpenTime(2, "08:00:00", "23:00:00"),
                new OpenTime(3, "08:00:00", "23:00:00"),
                new OpenTime(4, "08:00:00", "23:00:00"),
                new OpenTime(5, "08:00:00", "23:00:00"),
                new OpenTime(6, "10:00:00", "02:30:00"),
                new OpenTime(7, "10:00:00", "02:30:00")));
        place.setOpenTime(openTime);

        Context appContext = InstrumentationRegistry.getTargetContext();
        DBHelper databaseHelper = DBHelper.getInstance(appContext);

        List<Place> places = new ArrayList<>();
        places.add(place);

        databaseHelper.addOrUpdatePlaces(places);

        Place selectedPlace = databaseHelper.getPlaceById(100);
        assertEquals("10:00:00", select-
edPlace.getOpenTime().get(6).getFromHour());
        assertEquals("Test", selectedPlace.getPlaceName());
    }
```

**Рис. 14.** Модульный тест для проверки корректности вставки в базу данных и чтения из нее данных о заведении

```
    @Test
    public void dateFormatting_isCorrect() {
        Context appContext = InstrumentationRegistry.getTargetContext();
        String dotedDate = formatToDotedDate(appContext, "2017-01-23");
        String noYearDate = formatToNoYearDate(appContext, "2017-01-23");
        assertEquals("23.01.17", dotedDate);
        assertEquals("23 января", noYearDate);
    }
```

**Рис. 15.** Модульный тест для проверки различных форматов вывода даты

**Табл. 1.** Тестирование приложения

| № | Аспект работы | Действия | Результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Регистрация | 1. На экране регистрации ввести имя, телефон и email.<br>2. Нажать на кнопку «зарегистрироваться».<br>3. Ввести полученный в СМС код авторизации. | Выводится сообщение об успешной регистрации. Происходит перенаправление на экран со списком заведений.<br>В базе данных сервера появляется информация о новом пользователе. | Да |
| 2 | Авторизация при помощи мобильного телефона | 1. На экране авторизации ввести номер телефона.<br>2. Нажать на кнопку «авторизоваться».<br>3. Ввести полученный в СМС код авторизации. | Выводится сообщение об успешной авторизации. Происходит перенаправление на экран со списком заведений. | Да |
| 3 | Авторизация при помощи электронного адреса | 1. На экране авторизации нажать на кнопку «вход через email».<br>2. На экране авторизации ввести email.<br>3. Нажать на кнопку «авторизоваться».<br>4. Ввести полученный в СМС код авторизации. | Выводится сообщение об успешной авторизации. Происходит перенаправление на экран со списком заведений. | Да |
| 4 | Создание заявки на бронирование | 1. На экране информации о заведении нажать на кнопку «забронировать».<br>2. Заполнить обязательные поля желаемых даты, времени, количества персон.<br>3. Нажать на кнопку «забронировать». | Происходит сохранение данных о заявке на сервере системы. Выводится сообщение об успешном создании заявки.<br>В базе данных сервера появляется информация о новой заявке. | Да |

Окончание табл. 1

| № | Аспект работы | Действия | Результат | Тест пройден? |
|---|---|---|---|---|
| 5 | Отмена заявки на бронирование | 1. На экране списка текущих заявок на карточке заявки нажать на кнопку «отменить бронь». | Происходит сохранение данных о заявке на сервере системы. Выводится сообщение об успешной отмене заявки. В базе данных сервера изменяется статус заявки. | Да |
| 6 | Изменение персональных данных пользователя | 1. На экране данных о пользователе изменить email.<br>2. Нажать на кнопку «сохранить». | Происходит сохранение обновленных данных на сервере системы. Выводится сообщение об успешном изменении данных. В базе данных сервера изменяется статус заявки. | Да |

В табл. 2 приведен протокол тестирования некоторых аспектов работы компонента сервера, отвечающего за обработку запросов приложения менеджера.

**Табл. 2.** Тестирование сервера

| № | Аспект работы | Действия | Результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Получить информацию о заведении менеджера | 1. Отправить GET-запрос на соответствующий URL, содержащий заголовок с JWT. | Получен JSON, содержащий информацию о запрошенном заведении. | Да |
| 2 | Получить информацию о чужом заведении | 2. Отправить GET-запрос на соответствующий URL, содержащий заголовок с JWT. | Получен JSON с ненулевым кодом ошибки и сообщением «ошибка доступа». | Да |

Окончание табл. 2

| № | Аспект работы | Действия | Результат | Тест пройден? |
|---|---|---|---|---|
| 3 | Изменить информацию о заведении | 1. Отправить PUT-запрос на соответствующий URL, содержащий заголовок с JWT. Тело запроса – JSON с изменяемыми параметрами заведения. | В базе данных изменяются отправленные параметры. Получен JSON с нулевым кодом ошибки и пустым сообщением об ошибке. | Да |
| 4 | Добавить пункт меню заведения | 1. Отправить POST-запрос на соответствующий URL, содержащий заголовок с JWT. Тело запроса – JSON с параметрами нового пункта меню. | В базе данных появляется пункт меню с отправленными параметрами. Получен JSON с нулевым кодом ошибки и пустым сообщением об ошибке. | Да |
| 5 | Изменить пункт меню заведения | 1. Отправить PUT-запрос на соответствующий URL, содержащий заголовок с JWT. Тело запроса – JSON с параметрами пункта меню. | В базе данных изменяются отправленные параметры. Получен JSON с нулевым кодом ошибки и пустым сообщением об ошибке. | Да |
| 6 | Удалить пункт меню заведения | 1. Отправить DELETE-запрос на соответствующий URL, содержащий заголовок с JWT. | В базе данных удаляется выбранный пункт меню. Получен JSON с нулевым кодом ошибки и пустым сообщением об ошибке. | Да |
| 7 | Изменить статус заявки на бронирование | 1. Отправить PUT-запрос на соответствующий URL, содержащий заголовок с JWT. Тело запроса – JSON с идентификаторами заявки и ее нового статуса. | В базе данных изменяется статус заявки. Пользователь, создавший заявку, получает уведомление о смене ее статуса. | Да |

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы бакалавра было разработано мобильное приложение «TableTime» на платформе Android. Код системы составил свыше 1300 строк кода на языке Python, свыше 6500 строк на языке Java и около 2000 строк на языке разметки XML.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

1. Выполнен анализ предметной области и произведен обзор существующих решений.

2. Спроектирован и реализован сервер системы.

3. Спроектировано и реализовано Android-приложение.

4. Проведено тестирование приложения.

НАПРАВЛЕНИЯ ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ

Дальнейшим направлением развития будет создание интерактивной карты столиков заведения и внедрение функционала предварительного заказа блюд.

**ЛИТЕРАТУРА**

1. Ешин С.С. Сравнительный анализ ORM-библиотек для платформы Android на основе критерия производительности. // Информатика: проблемы, методология, технологии материалы XV международной научно-методической конференции (12–13 февраля 2015 г., г. Воронеж). Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2015. – С. 276–280.

2. Кирпичников А.П., Ляшева С.А., Шлеймович М.П., Еремеев Д.Е. Автоматизированная система взаимодействия пользователей с базой данных посредством приложений для мобильных устройств. // Вестник казанского технологического университета, 2015. – Т. 18. – № 3. – С. 235–238.

3. Малышкина Е.А. Совершенствование маркетинговых инструментов в интернет-бизнесе как фактор наиболее эффективного воздействия на потребителя. // Социально-экономические явления и процессы. – Тамбовский государственный университет им. Г.Р. Державина, 2012. – С. 136–146.

4. Приложение в Google Play «Афиша–Рестораны». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.afisha.restaurants (дата обращения: 17.02.2017).

5. Приложение в Google Play «Столики». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.saas.app (дата обращения: 17.02.2017).

6. Приложение в Google Play «ToMесто – лучшие рестораны». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.tomesto.tomesto (дата обращения: 10.02.2017).

7. Приложение в Google Play «EatOut». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.yell.rest (дата обращения: 17.02.2017).

8. Приложение в Google Play «Lipe». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.lipe.android.activity (дата обращения: 25.02.2017).

9. Приложение в Google Play «RestOn – бронирование столиков». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.reston.cs.company.android/ (дата обращения: 09.03.2017).

10. Приложение в Google Play «Toptable». [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.safaroff.toptable (дата обращения: 28.02.2017).

11. Харди Б., Филлипс Б. Программирование под Android. – СПб.: Питер, 2014. – 592 с.

12. Шевченко Д. Версионная миграция структуры базы данных: основные подходы. [Электронный ресурс] URL: https://habrahabr.ru/post/121265/ (дата обращения: 01.05.2017)

13. Android Developers Dashboard. [Электронный ресурс] URL: https://developer.android.com/about/dashboards/index.html (дата обращения: 20.04.2017)

14. Android Development Tool. [Электронный ресурс] URL: https://developer.android.com (дата обращения: 17.02.2017).

15. Django documentation. [Электронный ресурс] URL: https://docs.djangoproject.com/ (дата обращения: 29.03.2017).

16. Firebase Cloud Messaging. [Электронный ресурс] URL: https://firebase.google.com/docs/cloud-messaging/ (дата обращения: 29.03.2017).

17. GreenDao frequently asked questions. [Электронный ресурс] URL: http://greenrobot.org/eventbus/documentation/faq/ (дата обращения: 07.05.2017).

18. JSON Web Tokens. [Электронный ресурс] URL: https://jwt.io/ (дата обращения: 07.03.2017).

19. JUnit – фреймворк для модульного тестирования приложений на языке Java. [Электронный ресурс] URL: http://junit.org/junit4/ (дата обращения: 13.03.2017).

20. Marketing & Transactional Email Service SendGrid. [Электронный ресурс] URL: https://sendgrid.com/ (дата обращения: 20.02.2017).

21. Material design guidelines SendGrid. [Электронный ресурс] URL: https://material.io/guidelines/material-design/introduction.html (дата обращения: 20.03.2017).

22. OpenShift: PaaS by Red Hat. [Электронный ресурс] URL: https://www.openshift.com/ (дата обращения: 20.02.2017).

23. PostgreSQL: The world’s most advanced open source database. [Электронный ресурс] URL: http://www.postgresql.org/ (дата обращения: 20.02.2017).

24. Provos N., Mazieres D. A Future-Adaptable Password Scheme. // USENIX Annual Technical Conference, FREENIX Track, 1999. – С. 81–91.

25. SMSЦентр. [Электронный ресурс] URL: https://smsc.ru/ (дата обращения: 28.02.2017).

26. SQLite database engine. [Электронный ресурс] URL: https://www.sqlite.org/ (дата обращения: 28.02.2017).

27. VDS-хостинг FirstVDS. [Электронный ресурс] URL: https://firstvds.ru/ (дата обращения: 24.04.2017).

## ПРИЛОЖЕНИЕ

### Графический интерфейс приложения

Результаты отображения интерфейса на устройстве OnePlus X, 5", Android 6.0.1.

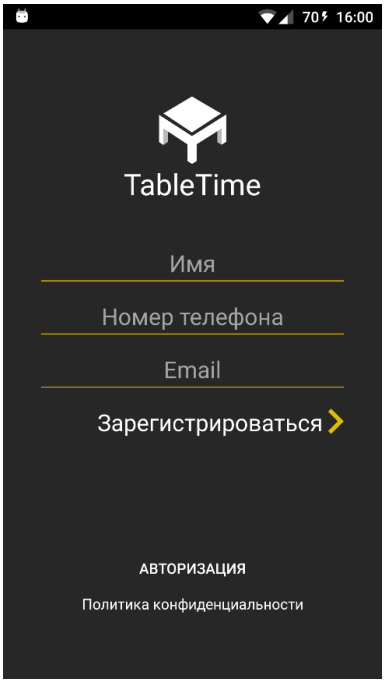


**Рис. 1.** Экран регистрации

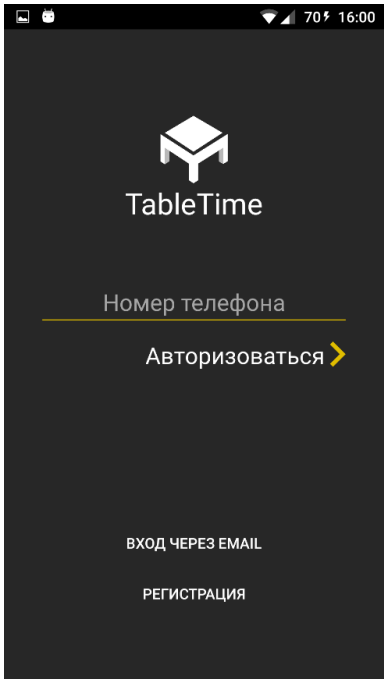


**Рис. 2.** Экран авторизации

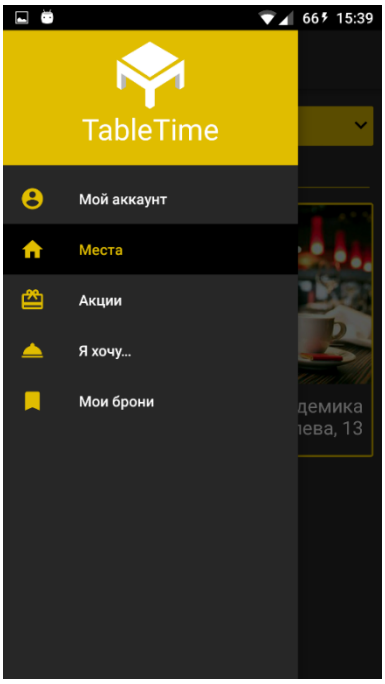


**Рис. 3.** Главное меню

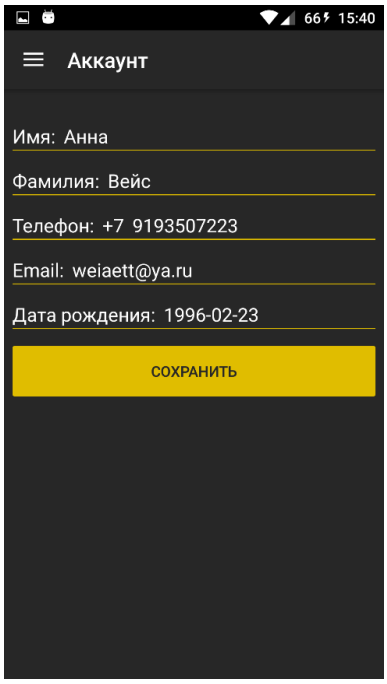


**Рис. 4.** Экран информации о пользователе

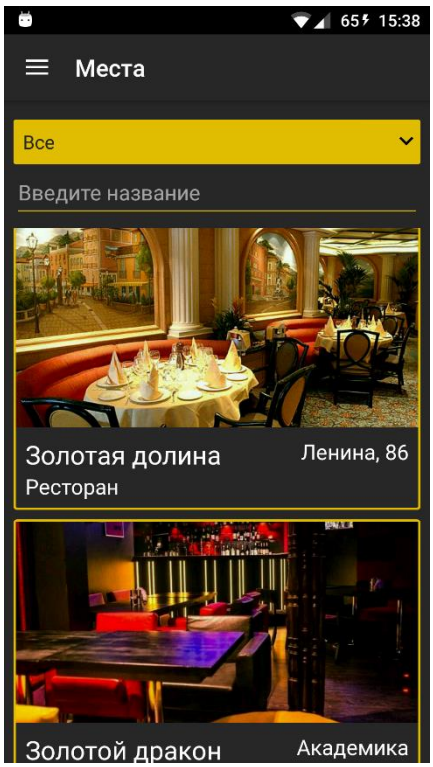


**Рис. 5.** Экран списка доступных заведений

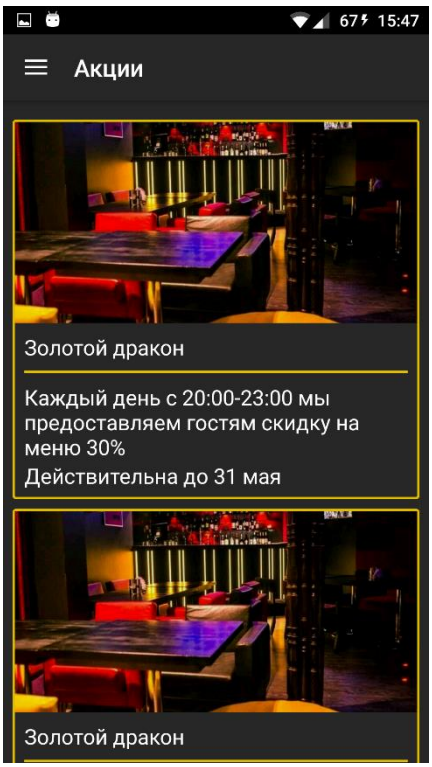


**Рис. 6.** Экран списка специальных предложений


**Рис. 7.** Экран с общей информацией о заведении


**Рис. 8.** Экран меню заведения


**Рис. 9.** Экран с часами работы заведения

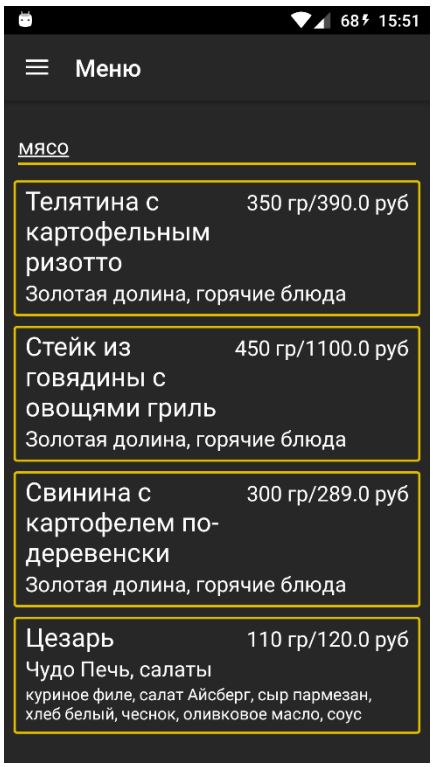


**Рис. 10.** Экран поиска по общему меню заведений

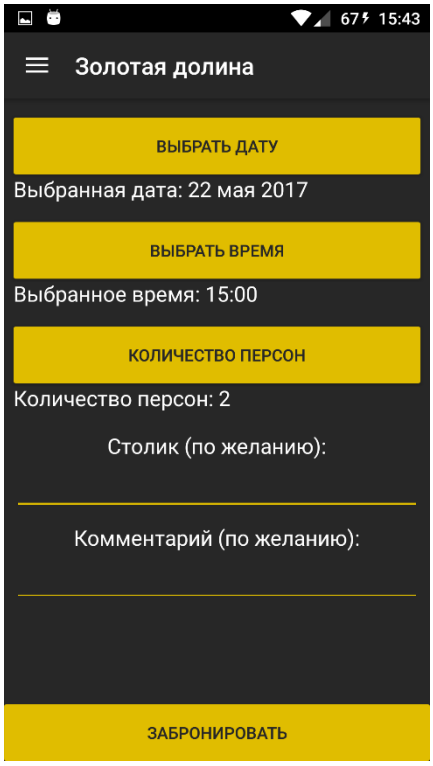


**Рис. 11.** Экран создания заявки на бронирование

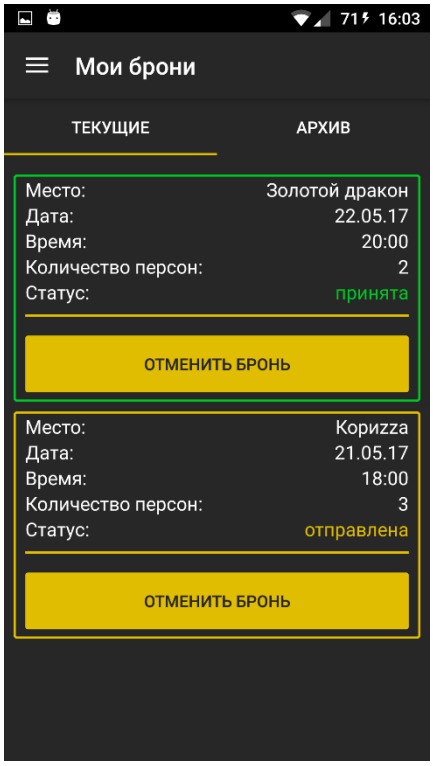


**Рис. 12.** Экран со списком незавершенных заявок на бронирование

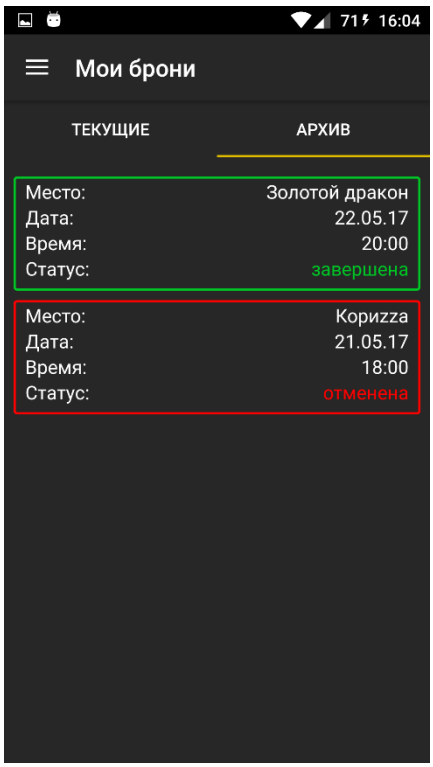


**Рис. 13.** Экран завершенных заявок на бронирование