Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

РАБОТА ПРОВЕРЕНА
РЕЦЕНЗЕНТ,
_____
_____
«____»_____2017 Г.

ДОПУСТИТЬ К ЗАЩИТЕ
ЗАВЕДУЮЩИЙ КАФЕДРОЙ,
Д.Ф.-М.Н., ДОЦЕНТ
_____С.А.ЗАГРЕБИНА
«____»_____2017 Г.

Разработка Android-приложения для предприятия мобильного экспресс-питания

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2017.067.05.000 ПЗ

Нормоконтролер,
Доцент каф. МиКМ, к.ф.-м.н.,
_____Т.А. Макаровских
_____2017 г.

Руководитель проекта,
Доцент каф. МиКМ, к.ф.-м.н.,
_____Т.А. Макаровских
_____2017 г.

Автор работы
Студент группы ЕТ-485
_____Р.В. Байтимиров
_____2017 г.

Челябинск, 2017

Дипломная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»
Направление «Прикладная математика и информатика»

ДОПУСТИТЬ К ЗАЩИТЕ
ЗАВЕДУЮЩИЙ КАФЕДРОЙ,
Д.Ф.-М.Н., ДОЦЕНТ
_____С.А.ЗАГРЕБИНА
«____»_____2017 г.

ЗАДАНИЕ
на выпускную квалификационную работу студента
Байтимирова Романа Викторовича
Группа ЕТ-485

1 Тема работы

Разработка Android-приложения для предприятия мобильного экспресс-питания утверждена приказом по университету от «___»_____20__г. №_____.

2 Срок сдачи студентом законченной работы

«___»_____20__г.

3 Исходные данные к работе

Данные о предприятии мобильного экспресс-питания: данные о клиентах, данные о продукции, бизнес-план предприятия мобильного экспресс питания.

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов)

4.1 Обзор мобильных приложение для предприятий мобильного экспресс-питания

4.2 Схема базы данных, информацию о точках продажах, о личной информации клиентов, о заказах клиентов

4.3 Программная реализация приложения для предприятия мобильного экспресс-питания

4.4 Тестирование разработанного мобильного приложение

4.5 Разработка бизнес-плана для предприятия мобильного экспресс-питания, с учетом введения разработанного мобильного приложения

5 Перечень графического материала

5.1 Титульный слайд

5.2 Цель и задачи работы

5.3 Сравнение аналогов разрабатываемого приложения

5.4 Описание базы данных

5.5 Структурная схема приложения

5.6 Разработка мобильного приложения

5.7 Тестирование мобильного приложения

5.8 Разработка бизнес-плана для предприятия

5.9 Основные результаты и выводы

6 Дата выдачи задания «___»_____20__г.

Руководитель работы _____/Т.А. Макаровских/

Задание принял к исполнению_____/Р.В. Байтимиров/

КАЛЕНДАРНЫЙ ПЛАН

Наименование разделов выпускной квалификационной работы	Срок выполнения разделов работы	Отметка о выполнении руководителя
1. Сбор материалов и литературы по теме работы	01.02.2017 – 15.02.2017	
2. Обзор и сравнение аналогов разрабатываемого приложения	16.02.2017 – 24.02.2017	
3. Разработка архитектуры системы	25.02.2017 – 09.03.2017	
4. Разработка базы данных	10.03.2017 – 15.03.2017	
5. Разработка мобильного приложения	16.03.2017 – 12.04.2017	
6. Тестирование системы	13.04.2017 – 18.04.2017	
7. Разработка бизнес-плана для предприятия	19.04.2017 – 25.04.2017	
8.Подготовка пояснительной записки	26.04.2017 – 16.05.2017	
9. Оформление пояснительной записки	17.05.2017 – 28.05.2017	
10. Проверка работы руководителем, исправление замечаний	28.05.2017 – 02.06.2017	
11. Получение отзыва руководителя	02.06.2017	
12. Подготовка графического материала	17.05.2017 – 02.06.2017	
13. Нормоконтроль	05.06.2017	
14. Рецензирование, представление зав. кафедрой	17.07.2017	

Заведующий кафедрой _____/С.А. Загребина /

Руководитель работы _____/Т.А. Макаровских/

Студент _____/Р.В. Байтимиров/

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

АННОТАЦИЯ

Байтимиров Р.В. Разработка Android-приложения для предприятия мобильного экспресс-питания / Р.В. Байтимиров. – Челябинск: ЮУрГУ, Институт естественных и точных наук, 2017. – 66 с., 14 ил., 4 табл., библиогр. список – 12 названий

В дипломной работе были рассмотрены приложения для предприятий мобильного экспресс-питания. Также был произведён сравнительный анализ рассмотренных приложений.

Было разработано Android-приложение для предприятия мобильного экспресс-питания на основе Ionic Framework с использованием Firebase Realtime Database. Было произведено тестирование разработанного приложения для предприятий мобильного экспресс-питания.

Был рассмотрен бизнес-план предприятия мобильного экспресс-питания в период с 01.01.2017 по 01.05.2017 до внедрения мобильного приложения. Также был составлен бизнес-план для предприятия мобильного экспресс-питания в период с 01.05.2017 по 01.09.2017, с условием внедрения разработанного мобильного приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Быстрая еда, приготовленная из полуфабрикатов, пользуется большим спросом как в крупных мегаполисах, так и в небольших провинциальных городах. Она отличается приятным вкусом, доступной ценой и высокой калорийностью. Самый выгодный формат заведения быстрого питания — предприятие мобильного экспресс-питания.

Небольшой передвижной фургончик при грамотном подходе будет приносить больше прибыли, чем стационарное предприятие экспресс-питания. Днем можно торговать возле деловых центров или учебных заведений, а вечером перемещаться в парки или скверы.

Для дополнительного потока клиентов и автоматизации принятия онлайн-заказов идеальным решением будет создание мобильного приложения.

В связи с бурным развитием экспресс-питания в Российской Федерации **актуальность** исследования в данном направлении является актуальной задачей.

Так как предприятие мобильного экспресс-питания относится к разряду малого бизнеса, следовательно, чтобы стать доступнее для конечного пользователя, ставится цель создания мобильного приложения.

Объектом исследования является предприятие мобильного экспресс-питания, оказывающее услуги в передвижных фургонах.

Предметом исследования является мобильное приложение, позволяющее осуществлять онлайн-заказ в предприятии мобильного экспресс-питания.

Целью работы является разработка мобильного приложения для предприятия мобильного экспресс-питания, которое поддерживает поиск ближайших точек продаж, оформление онлайн-заказа, и обеспечение передачи данных о заказах между мобильным устройством и базой данных.

В работе поставлены и решены следующие **задачи**:

1) обзор и сравнение существующего программного обеспечения для предприятий мобильного экспресс-питания;

2) реализация Android-приложения, обрабатывающего онлайн-заказы пользователей и отправку их на сервер, а также отображение полученной с сервера информации;

3) прогнозирование бизнес-плана для предприятия мобильного экспресс-питания с учётом мобильного приложения.

Информационная база исследования

- Ionic2. Build Mobile Apps with HTML5. – Режим доступа: https://www.joshmorony.com/tag/ionic2/
- Documentation Apache Cordova. – Режим доступа: https://cordova.apache.org/docs/en/latest/
- Documentation Firebase. – Режим доступа: https://firebase.google.com/docs/

Структура и объем работы

Работа состоит из введения, четырех разделов, заключения, библиографического списка и приложения. Объем работы составляет 66 страниц, объем библиографии – 12 источников.

В первой главе происходит обзор и сравнение существующих приложений для предприятий мобильного экспресс-питания с разрабатываемым приложением.

Во второй главе описываются отличия реляционных моделей БД и NoSQL БД. Составлена структурная схема приложения с описанием каждого экрана.

В третьей главе описывается архитектура приложения. Описывается выбранный способ разработки мобильного приложения. Подробно описывается разработка одного экрана приложения. Разбираются все остальные подключенные модули и библиотеки проекта.

В четвертой главе приведен бизнес-план предприятия до внедрения разрабатываемого приложения. Учитываются все затраты на разработку приложения и все маркетинговые мероприятия. Разрабатывается бизнес-план для предприятия мобильного экспресс-питания, с учетом внедрения мобильного приложения.

В заключении на основании разработанного приложения и полученных данных о бизнес-плане сделаны общие выводы.

1 ОБЗОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СУЩЕСТВУЮЩИХ ПРЕДПРИЯТИЙ МОБИЛЬНОГО ЭКСПРЕСС-ПИТАНИЯ

Рассмотрим мобильные приложения на базе операционной системы Android из магазина приложений Google Play, которые предназначены для предприятий мобильного экспресс-питания.

1.1 Приложения для предприятий мобильного экспресс-питания

В ходе поиска не было найдено аналогов в Российской Федерации. Увеличив границы поиска, были найдены приложения для предприятий мобильного экспресс-питания, основными из которых являются:

1) Furgoneta;
2) StreetEat;
3) Kater.

Далее был произведён обзор каждого из приложений для подготовки к сравнению.

Мобильное приложение «Furgoneta»

Мобильное приложение «Furgoneta» принадлежит одноимённому бренду, предприятию мобильного экспресс-питания «Furgoneta» (Украина, Киев) [1]. Приложение является близким по функциям по сравнению с разрабатываемым приложением.

С помощью данного мобильного приложения можно найти точки продаж, просматривать меню, производить заказ. Так же предусмотрена доставка.

При включении приложения пользователь видит главный экран (рис. 1), на котором указаны основные пункты меню, необходимые для ознакомления с предприятием, продукцией предприятия, также для оформления заказа.

Для того, чтобы найти точки продаж следует сделать следующую последовательность действий (рис. 2):

1) с главной страницы перейти на боковую панель;
2) перейти по ссылке «Наши адреса»;
3) выбрать нужный адрес, и нажать на кнопку «На карте» для открытия карты.

Рисунок 1 — Главный экран мобильного приложения «Furgoneta»

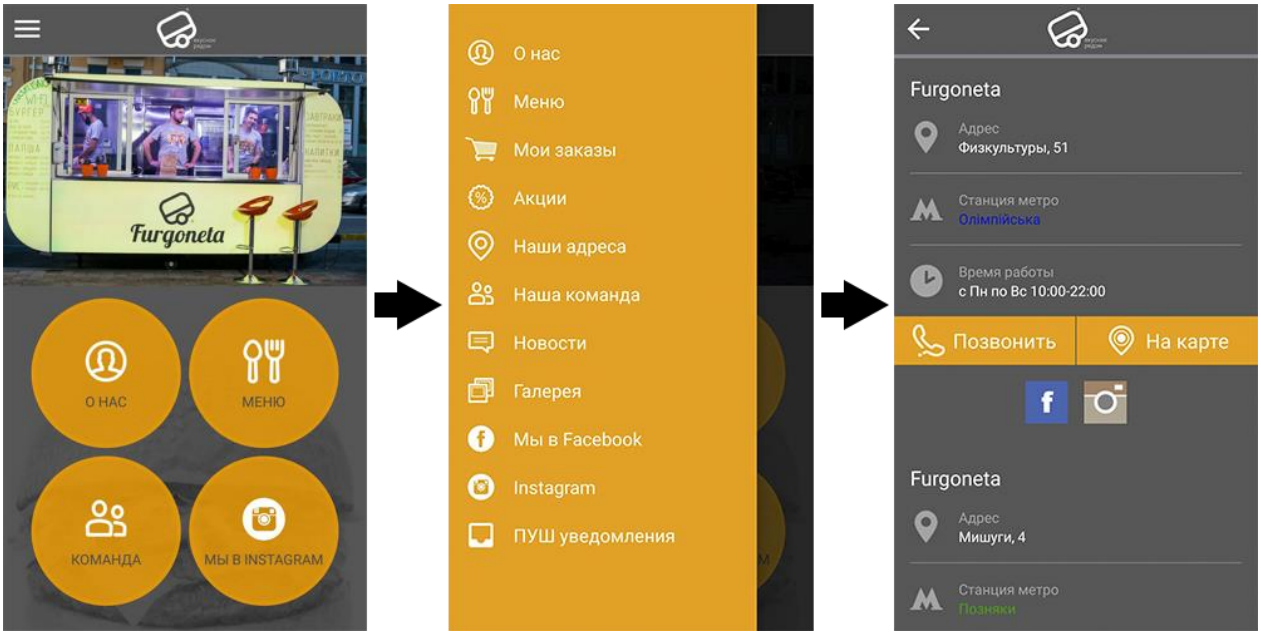

Рисунок 2 — Порядок действий для поиска точки продаж

Мобильное приложение «StreetEat»

Мобильное приложение «StreetEat» является агрегатором предприятий мобильного экспресс-питания для таких стран, как Италия, Великобритания, Франция и Испания [2].

Данное приложение объединяет информацию о нескольких предприятиях мобильного экспресс-питания и имеет единый пользовательский интерфейс.

С помощью данного приложения можно найти ближайшие точки продаж разных брендов в зависимости от предоставляемой продукции. Так как приложение является агрегатором, то нельзя произвести онлайн-заказ.

На главном экране мобильного приложения расположен список поддерживаемых стран (рис. 3).

Рисунок 3 — Главный экран мобильного приложения «StreetEat»

Для поиска точки продаж следует выполнить следующую последовательность действий (рис. 4):

1) на главном экране выбрать страну;
2) выбрать блюдо.

После выполнения данных действий, будет показана карта с точками продаж различных предприятий мобильного экспресс-питания.

Рисунок 4 — Порядок действий для поиска точек продаж

Мобильное приложение «Kater»

Мобильное приложение «Kater» является единой системой заказов для предприятий мобильного экспресс-питания в США [3].

Приложение объединяет в себе разные предприятия мобильного экспресс-питания, в каждом из которых можно сделать онлайн-заказ, используя данное мобильное приложение. При создании онлайн-заказа клиент в праве назначить готовность сделанного заказа на удобное для него время и забрать его самостоятельно.

При входе в мобильное приложение пользователь видит список компаний, личный кабинет, корзину заказа (рис. 5).

Для поиска ближайших точек продаж следует на главном экране нажать на кнопку «Map» (рис. 6). Будет показана карта с ближайшими точками продаж всех предприятий мобильного экспресс питания.

Рисунок 5 — Главный экран мобильного приложения «Kater»

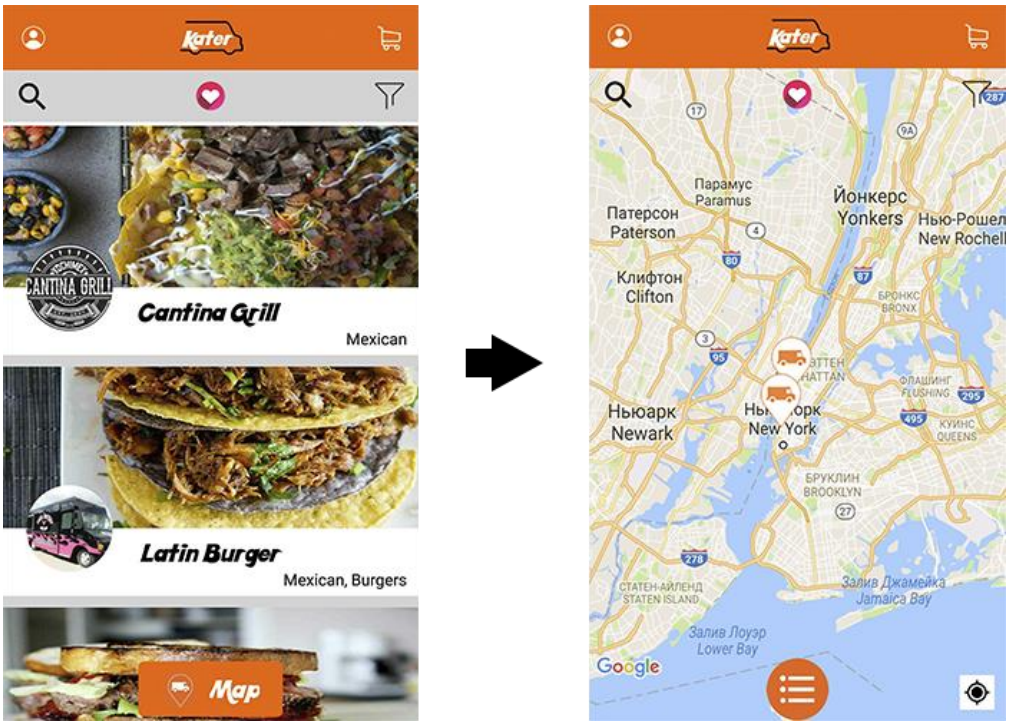

Рисунок 6 — Порядок действий для поиска точек продаж

1.2 Описание компании и разрабатываемого приложения

Компания «FastFOOD» является предприятием мобильного экспресс-питания в г. Челябинске. Предприятие оказывает услуги в сфере экспресс-питания с помощью передвижных фургонов.

Потребителями данной компании выступают любые слои населения, основными из которых являются:

1) люди, которые в связи с быстрым темпом жизни, могут позволить себе только быстрые перекусы (например, работники офисов, таксисты);

2) люди подросткового возраста.

Всего предприятий мобильного экспресс-питания в г. Челябинске насчитывается не более 5 компаний, каждая из которых имеет узкую специализацию — изготовление и продажа бургеров. В отличии от конкурентов компания «FastFOOD» владеет фургонами, вмещающими в себя большое количество оборудования, тем самым расширяя ассортимент блюд.

Основной инновацией предприятия является создание мобильного приложения, в котором указывается текущее местоположение точек продаж. Кроме того, в разрабатываемом приложение предлагается сделать онлайн-заказ на определённое время, получение которого обслуживается вне очереди.

Разрабатываемое приложение для компании «FastFOOD» должно быть создано на базе операционной системы Android и поддерживаться устройствами начиная с версии 4.4 и выше.

Разрабатываемое приложение должно иметь следующие функции:

1) возможность регистрации в приложении;
2) просмотр точек продаж на карте в реальном времени;
3) просмотр меню для каждого из фургонов;
4) возможность создания онлайн-заказа для выбранного фургона.

1.3 Сравнение разрабатываемого приложения с аналогами

В качестве критериев для сравнения выступают следующие функции, свойства и особенности приложений:

1) местоположение целевой аудитории приложения (страна/город);
2) стоимость лицензии;
3) отсутствие привязанности приложения к определённой компании;
4) возможность регистрации;
5) местонахождение точек экспресс-питания на карте;
6) просмотр продукции;

7) возможность совершения онлайн-заказа;

8) доставка заказа;

9) возможность самовывоза заказа в указанное клиентом время.

Данные о сравнении разрабатываемого приложения и аналогов представлены в таблице 1.

Таблица 1 — Сравнение разрабатываемого приложения и аналогов

№ п/п	Критерии сравнения	Приложения			
		Разрабатываемое приложение	Furgoneta	StreetEat	Kater
1	Местоположение целевой аудитории приложения	Россия, Челябинск	Украина, Киев	Италия, Велико-британия, Франция, Испания	США
2	Стоимость лицензии	Бесплатная	Бесплат-ная	Бесплат-ная	Бесплат-ная
3	Отсутствие привя-занности приложе-ния к определенной компании	–	–	+	+
4	Возможность реги-страции	+	–	–	+
5	Местонахождение точек экспресс-пи-тания на карте	+	+	+	+
6	Просмотр продук-ции	+	+	+	+
7	Возможность совер-шения онлайн-за-каза	+	+	–	+

Окончание таблицы 1

8	Доставка заказа	–	+	–	–
9	Возможность самовывоза заказа в указанное клиентом время	+	+/–	–	+

Выводы по главе один

В ходе анализа мобильных приложений данной отрасли было выявлено, что в Российской Федерации приложение данного функционала не реализовано.

Исходя из проведённого сравнения найденных мобильных приложений предприятий мобильного экспресс-питания и разрабатываемого приложения, можно утверждать, что создание мобильного приложения для предприятия мобильного экспресс-питания в Челябинске актуально.

2 СХЕМА БАЗЫ ДАННЫХ И СТРУКТУРНАЯ СХЕМА ПРОГРАММЫ

2.1 Базы данных для разрабатываемого приложения

База данных (БД) — объективная форма представления совокупности данных, систематизированных так, чтобы эти данные были найдены и обработаны с помощью ЭВМ.

Для управления, изменения и администрирования базы данных используется система управления базами данных (СУБД). Каждая СУБД поддерживает различные модели и структуры БД.

Реляционная модель БД

Одной из самых популярных моделей базы данных является реляционная модель. Реляционная модель предоставляет способы четкого структурирования данных. В ней используются таблицы с колоннами и рядами. У каждого ряда таблицы (записи) есть уникальный номер (первичный ключ), а колонны служат полями данных (рис. 7).

Рисунок 7 — Пример таблицы реляционной БД

Реляционные модели БД используют схемы — строго описанные структуры данных. Схема базы данных включает в себя описание содержания, структуры и ограничения целостности.

NoSQL БД

В NoSQL базах структура данных не регламентирована — в отдельной строке или документе можно добавить произвольное поле без предварительного декларативного изменения структуры всей таблицы. Таким образом, если появляется необходимость поменять модель данных, то единственное достаточное действие — отразить изменение в коде приложения.

NoSQL БД не основываются на одной модели, а каждая база данных в зависимости от целей использует различные модели.

В зависимости от модели данных можно выделить четыре типа хранилищ:

1) хранилище «ключ-значение»;

2) хранилище семейств колонок;

3) документо-ориентированные СУБД;

4) базы данных на основе графов.

Более подробно рассмотрим документо-ориентированную модель. Данная модель основывается на текстовом формате обмена данными JSON. Базы данных такого формата состоят из ключей и данных. Ключ может содержать в себе другие ключи, которые в свою очередь будут иметь данные. Таким образом образуются подузлы и узлы (рис. 8). В данном случае узлом является «customers», подузлом является «customers_one», ключи «firstName», «birthday», «location» и их данные составляют поля данных (рис. 9).

Рисунок 8 — Пример узла NoSQL БД

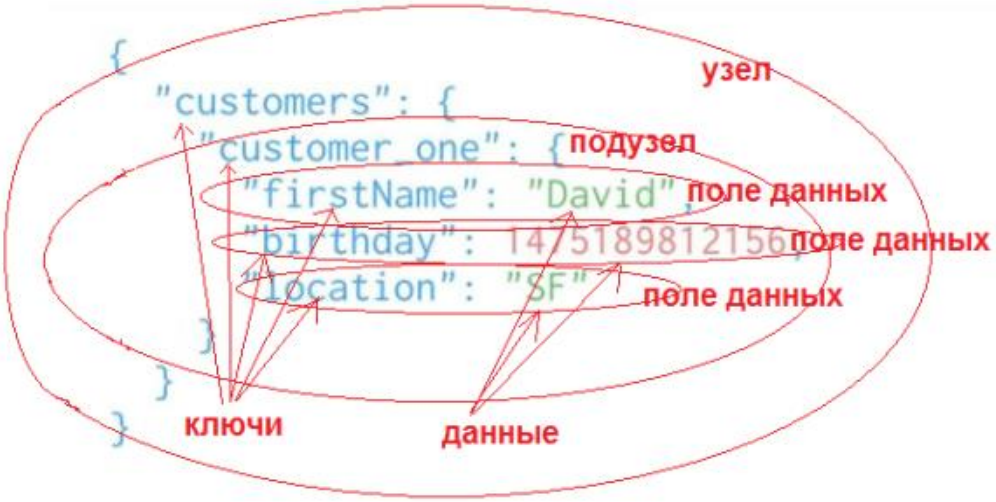

Рисунок 9 — Разбор узла на подузлы и поля данных

Преимуществами NoSQL БД перед реляционными моделями БД являются:

1) большая скорость чтения в распределенной среде;

2) возможность хранить физически объекты в том виде, в каком с ним работает приложение;

3) возможность разработки базы данных без задания схемы;

4) более простые способы горизонтального масштабирования (создание кластера из нескольких машин);

5) родная поддержка атомарности на уровне записей.

2.2 База данных разрабатываемого приложения

Для хранения и синхронизации данных разрабатываемого приложения была выбрана облачная база данных Firebase Realtime Database от Firebase. Данная база данных относится к NoSQL БД, основывающейся на документо-ориентированной модели данных. База данных позволяет работать с данными, которые хранятся как JSON и синхронизируются в реальном времени.

Данные хранящиеся в базе данных могут храниться в любом виде, без привязанности к определенному типу данных. При необходимости установить ограничения устанавливаются правила безопасности.

Существуют следующие правила безопасности:

1) .read — определяет кто может читать данные из базы данных;

2) .write — определяет кто может записывать данные в базу данных;

3) .validate — определяет какому типу данных принадлежит поле;

4) .indexOn — используется для более быстрого поиска данных, путем исключения узлов из поиска.

База данных разрабатываемого приложения содержит в себе несколько узлов: узел пользователей и узел фургонов.

В узле пользователей «users» находятся подузлы, ключ которых равен уникальному ключу зарегистрированного пользователя, с личными данными о пользователе, данными о текущем заказе и выполненных заказах. Структура узла «users» приведена на рисунке 10.

Рисунок 10 — Структура узла «users»

В узле «car» находятся подузлы для каждой из точки продаж. В подузлах хранятся данные о точках продаж предприятия, таких как, расположение точки продаж, меню каждого фургона, в которое входят товары, и заказы клиентов.

2.3 Реализация базы данных разрабатываемого приложения

Для работы с Firebase Realtime Database необходимо подключить приложение к Firebase с помощью конфигурации аккаунта (листинг 1).

Подключение базы данных происходит путем инициализации конфигурации аккаунта в приложение с помощью модуля «AngularFireModule».

Листинг 1 — Конфигурации аккаунта Firebase

```
export const FIREBASE_CONFIG = {
  apiKey: "AIzaSyACI6DKPG8vh61eT1gSdGR5UNyFT3LUmoA",
  authDomain: "fastfood-1495630596176.firebaseapp.com",
  databaseURL: "https://fastfood-1495630596176.firebaseio.com",
  projectId: "fastfood-1495630596176",
  storageBucket: "fastfood-1495630596176.appspot.com",
  messagingSenderId: "59783860433"
};
```

На примере узла «users» рассмотрим добавление, чтение, обновление и удаление данных в Firebase Realtime Database.

Рассмотрим добавление данных в базу данных на примере регистрации пользователей (листинг 2).

Листинг 2 — Добавление данных в Firebase Realtime Database

```
result = this.afAuth.auth.createUserWithEmailAnd|
Password(user.email, user.password);
firebase.database().ref('users').child(result.uid).set({
  uid: result.uid,
  email: user.email
});
```

В переменной «result» хранится объект с данными о только что зарегистрированном пользователе — адрес электронной почты, пароль, уникальный идентификатор. Происходит подключение к базе данных с помощью метода «database()». С помощью метода «ref()» выбирается, если ранее был создан, или создается, если нет, узел «users». Добавляется подузел, ключ которого совпадает для каждого пользователя с уникальным ключом пользователя, данный ключ берётся из переменной result. Для добавления данных используется метод «set()». При использовании метода «set()» все подузлы и ключи полностью перезаписываются.

Аналогично происходит обновление и удаление данных, используя методы «update()» и «remove()», соответственно.

Рассмотрим чтение данных из базы данных на примере профиля пользователя (листинг 3).

Листинг 3 — Чтение данных из Firebase Realtime Database

```
var userId = firebase.auth().currentUser.uid;
return firebase.database().ref('/users/' + userId)
.once('value').then(function(snapshot) {
  name.innerText = snapshot.val().name;
  phone.innerText = snapshot.val().phone;
  email.innerText = snapshot.val().email;
});
```

В переменной «userId» содержится уникальный идентификатор пользователя.

Аналогично записи происходит подключение к узлу «users» с подузлом равному уникальному идентификатору пользователя. С помощью метода «once» устанавливается слушатель вида «данные», возвращающий данные о пользователе, которые записываются для последующих действий в переменные.

2.4 Структурная схема мобильного приложения

Мобильное приложение состоит из восьми экранов:

1) экран авторизации;
2) экран регистрации;
3) экран ввода личной информации после регистрации;
4) главный экран;
5) экран профиля;
6) экран просмотра активных и выполненных заказов;
7) экран меню;
8) экран онлайн-заказа.

На рисунке 11 показана структурная схема разрабатываемого приложения.

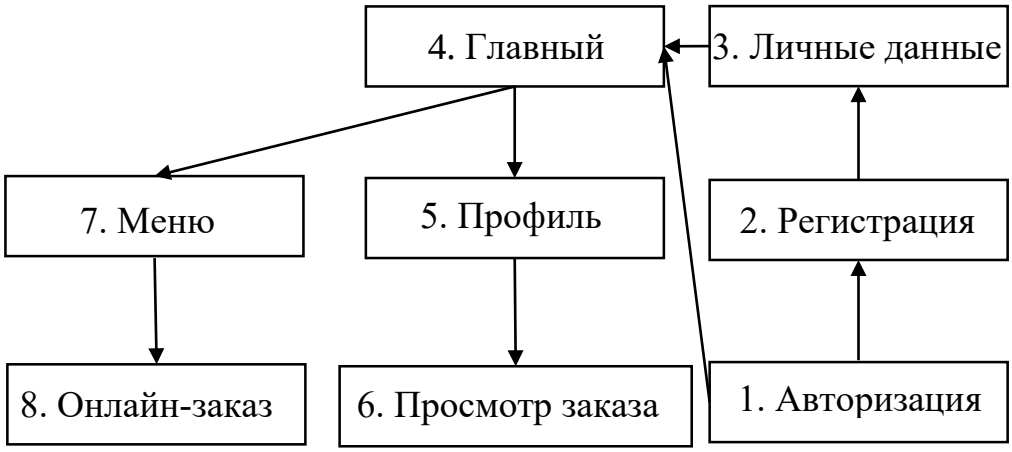

Рисунок 11 — Структурная схема разрабатываемого приложения

На экране авторизации расположены поля ввода почты и пароля пользователя. При совпадении введённых данных с данными пользователей в базе данных, пользователь переходит на главный экран, иначе пользователь уведомляется о невозможности авторизации. Так же предоставляется возможность перейти на экран регистрации для создания нового аккаунта.

На экране регистрации расположены поля ввода почты и пароля для совершения регистрации в приложении. При некорректном вводе данных, либо при совпадении введённых данных с данными зарегистрированных аккаунтов пользователей, пользователь уведомляется о невозможности продолжить регистрацию. После успешной регистрации пользователь перемещается на экран ввода личных данных.

На экране ввода личных данных уже зарегистрированного пользователя просят ввести личные данные, такие как имя и телефон, для связи с фургонами. После успешного окончания регистрации пользователь попадает на главный экран.

Экраны авторизации, регистрации и ввода личных данных расположены на рисунке 11.

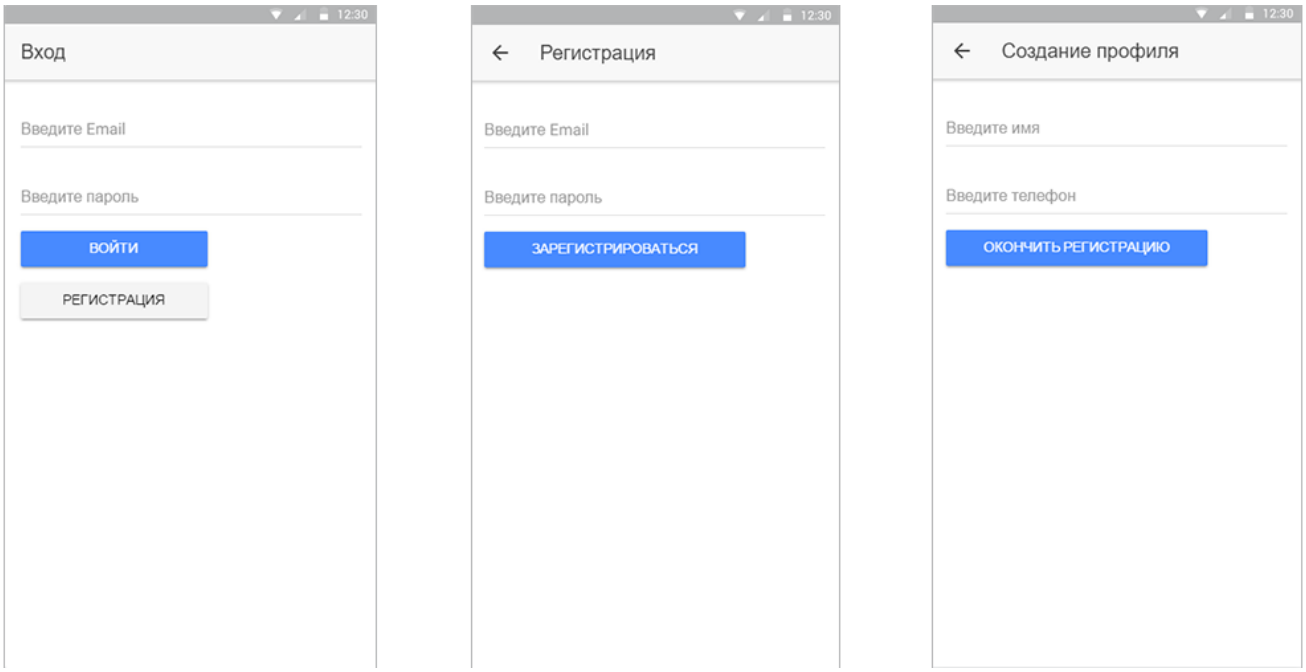

Рисунок 11 — Экраны авторизации, регистрации, ввода личных данных

На главном экране пользователь видит карту с текущим местоположением пользователя и точек продаж. В отличии от мобильных приложений конкурентов это позволяет пользователю сразу же после входа в приложение найти подходящий для него фургон, и перейти в меню выбора товаров. Пользователю предоставляется выбор перейти в меню товаров или в профиль пользователя.

В профиле пользователя хранятся личные данные пользователя и активные и выполненные онлайн-заказы. Пользователь может просмотреть любой заказ.

На экране просмотра заказа пользователь видит детали выбранного заказа, такие как, дата заказа, время заказа, заказанную продукцию, а также стоимость данного заказа.

Главный экран, экран профиля и экран заказа расположены на рисунке 12.

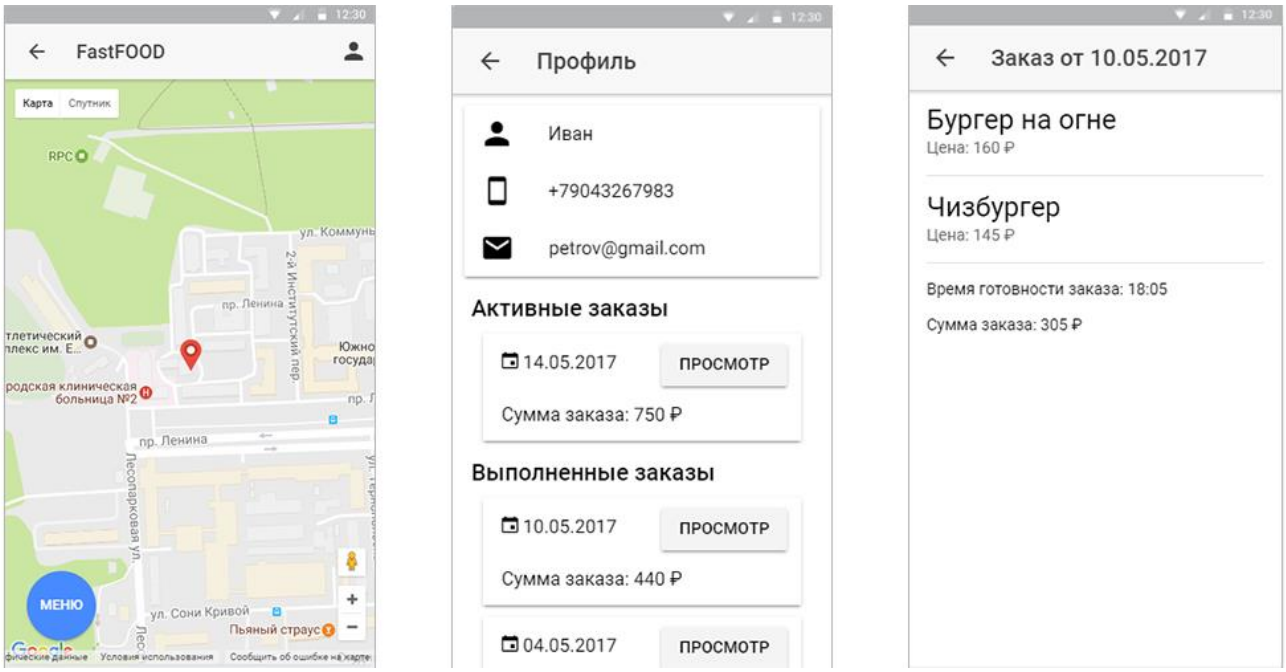

Рисунок 12 — Главный экран, экран профиля и экран заказа

На экране меню товаров пользователю предлагается выбрать подходящий по местоположению фургон, после чего пользователь выбирает интересующий его товар. При выборе товара, данный товар добавляется в корзину. После окончания выбора товаров пользователь переходит в корзину.

На экране онлайн-заказа пользователь видит выбранные им товары, которые при необходимости он может удалить, а также стоимость каждого товара и общую стоимость заказа. Пользователь должен выбрать время готовности заказа и при желании оставить комментарий к заказу. После подтверждения заказа, заказ добавляется в профиль к пользователю, а также к выбранному фургону.

Экран меню и экран онлайн-заказа представлен на рисунке 13.

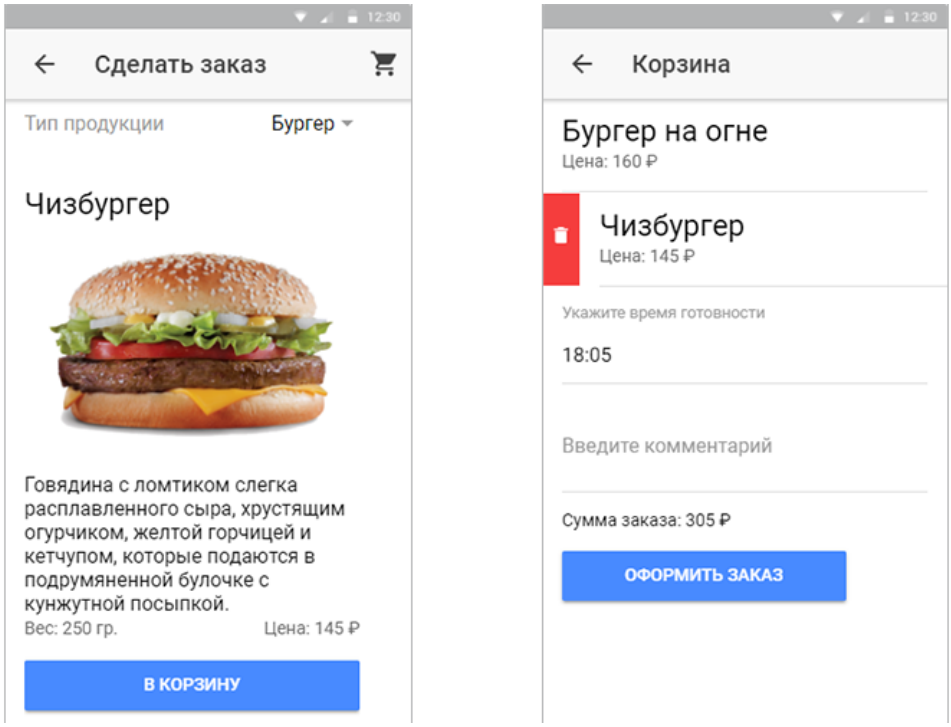

Рисунок 13 — Экран меню и экран оформления онлайн-заказа

Выводы по главе два

В ходе сравнения разных моделей баз данных была выбрана Firebase Realtime Database. Данная база данных идеально подходит для оформления онлайн-заказов, т.к. все изменения происходят мгновенно, как на стороне базы данных, так и на стороне клиента.

Описанные восемь экранов приложения выполняют полный функционал приложения, поставленный в задачах работы.

3 ОПИСАНИЕ РАЗРАБОТКИ

Мобильное приложение для предприятия мобильного экспресс-питания построена по клиент-серверной архитектуре (рис. 14) и состоит из двух компонентов:

1) Android-приложение;
2) Сервер и база данных Firebase.

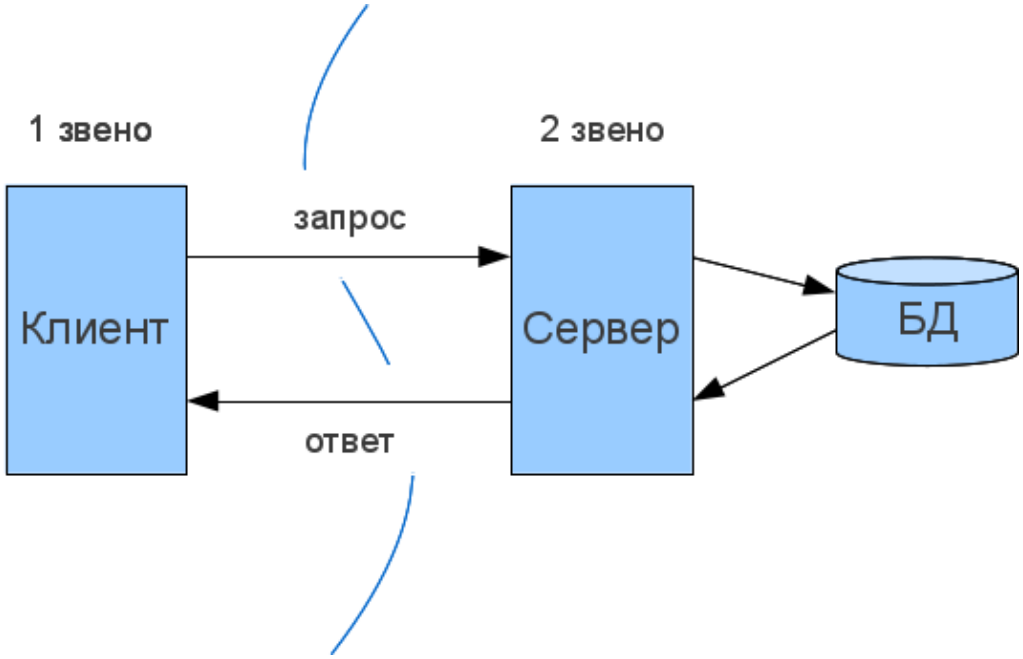

Рисунок 14 — Клиент-серверная архитектура

Клиентская часть выступает в роли мобильного приложения. Разрабатываемое мобильное приложение представляет собой пользовательский интерфейс системы. Приложение показывает текущее местоположение точек продаж, отображает личные данные пользователя, показывает меню в каждой точке продаж, также предоставляет возможность совершить онлайн-заказ.

Мобильное приложение передает на сервер информацию о пользователе, информацию об активных и выполненных заказах.

3.1 Разработка клиентской части

Для разработки мобильного приложения использовался Ionic Framework.

Ionic Framework — это SDK (software development kit — комплект средств разработки) с открытым исходным кодом для разработки гибридных мобильных приложений с использованием web-технологий, созданный на основе AngularJS, SASS, Apache Cordova.

Приложение, созданное с помощью данной технологии, является кроссплатформенным. Это обеспечивается за счет преобразования CSS, HTML и JavaScript в код, который любая платформа воспринимает как элемент web.

Для загрузки необходимых библиотек и модулей понадобиться Node.js. Для построения конечного мобильного приложения будут необходимы Java SDK и Android SDK.

Так как создание гибридных приложений основано на web-технологиях, следовательно, каждый экран будет состоять из файла разметки, файла стилизации разметки и файла скриптов.

В корне проекта файл config.xml отвечает за основные настройки созданного приложения. Файл index.html является связующим звеном всех компонентов приложения.

3.2 Разработка одного экрана приложения

Распишем создание одного экрана на примере главного экрана.

Для создания нового экрана необходимо в командной строке создать страницу в проекте. Далее произведем разметку главного экрана с помощью стандартных тегов HTML или тегов Ionic Framework (листинг 3). Теги Ionic Framework по умолчанию стилизованы под разрабатываемую операционную систему, поэтому с ними работать гораздо удобнее.

Листинг 3 — Разметка главного экрана

```
<ion-header>
  <ion-navbar>
    <ion-title>FastFOOD</ion-title>
    <ion-buttons end>
      <button ion-button icon-only (click)="openUserProfile()">
         <ion-icon name="md-person"></ion-icon>
       </button>
    </ion-buttons>
  </ion-navbar>
</ion-header>
<ion-content class="no-scroll">
  <div id="map"></div>
  <button ion-button class="btn-menu" (click)="openMainMenu
  Page()">Меню</button>
</ion-content>
```

Экран состоит из двух основных блоков — «header» и «content». В блоке header по умолчанию располагаются кнопка навигации «Назад» и название разрабатываемой страницы. На данном экране также добавлена кнопка перехода на экран профиля.

Блок «content» по умолчанию пустой. В нём располагается всё содержимое экрана. В данном случае это карта и кнопка перехода на экран меню.

Для срабатывания кнопок перехода на экран профиля и на экран меню необходимо указать обработчик нажатия «click».

Следующим шагом будет стилизация разметки свойствами CSS (листинг 4).

Листинг 4 — Стилизация разметки главного экрана

```
page-home {
  #map {
    height: 100%;
    width: 100%;
  }
  .btn-menu {
    position: absolute;
    bottom: 20px;
    left: 20px;
    border-radius: 1000px;
    width: 70px;
    height: 70px;
  }
  .no-scroll .scroll-content{
    overflow: hidden;
  }
}
```

В Ionic Framework существуют CSS свойства, прописываемые непосредственно в тегах HTML, значения данных свойств по умолчанию заданы. Такой подход используется для ускорения разработки экранов и создания одного стиля экранов. Например, <ion-content padding-left> будет означать, что блок «content» будет иметь padding слева равный 16px. Значение по умолчанию также можно изменять.

Последним шагом будет написание скриптов для корректной работы экрана, переходов на следующие экраны. В файле скриптов подключаются сторонние библиотеки, модули и API (application programming interface), также прописываются функции и методы.

Для подключения сторонних модулей необходимо скачать и установить их с помощью Node.js. Далее подключить их непосредственно в самом файле скриптов (листинг 5).

Листинг 5 — Подключаемые модули

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, ToastController } from 'ionic-angu-
lar';
import { AngularFireAuth } from "angularfire2/auth";
```

Для главного экрана необходимо отображение карт Google Maps. Подключение происходит с помощью Google Maps JavaScript API (листинг 6). Функция

«googleMap()» инициализирует карты. С помощью метода «getCurrentPosition()» получаем текущие координаты пользователя в виде широты и долготы. Далее инициализируем маркер пользователя и отображаем его на карте.

Листинг 6 — Подключение карт Google Maps

```
googleMap() {
  let that = this;
  let userMarker: any;

  this.map = new google.maps.Map(document.getElementById('map'), {
    zoom: 16
  });
  this.imageUserMarker = '../../assets/img/usermarker-1.png';
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(function(position) {
      var pos = {
        lat: position.coords.latitude,
        lng: position.coords.longitude
       };
       userMarker = new google.maps.Marker({
         position: pos,
         map: that.map,
         icon: that.imageUserMarker
       });
       that.map.setCenter(pos);
    }, function() {
       this.handleLocationError(true, that.infoWindow,
      that.map.getCenter());
    });
  } else {
    this.handleLocationError(false, that.infoWindow,
    that.map.getCenter());
  }
}
```

Для перехода с одного экрана на другой используется метод «push()» с помощью компонента контроллера навигации «NavController» (листинг 7).

Листинг 7 — Переход на новый экран

```
openMainMenuPage() {
  this.navCtrl.push('MainMenuPage');
}
openUserProfile() {
  this.navCtrl.push('UserProfilePage');
}
```

Функция «ionViewDidLoad()» является стандартной функцией Ionic Framework, эта функция срабатывает при загрузке экрана.

Всплывающие уведомления в приложении осуществляются с помощью контроллера уведомлений «ToastController» и метода «create()» (листинг 8).

Листинг 8 — Создание всплывающих уведомлений

```
ionViewDidLoad() {
  this.afAuth.authState.subscribe(data => {
    if (data && data.email && data.uid) {
      this.toast.create({
        message: `Добро пожаловать в FastFOOD, ${data.email}`,
        duration: 3000
       }).present();
    }
    else {
      this.toast.create({
         message: `Не была пройдена аутентификация`,
         duration: 3000
      }).present();
    }
  });
}
```

На рисунке 15 изображен главный экран после входа в аккаунт. Пользователь видит уведомление, созданное с помощью контроллера уведомлений «ToastController».

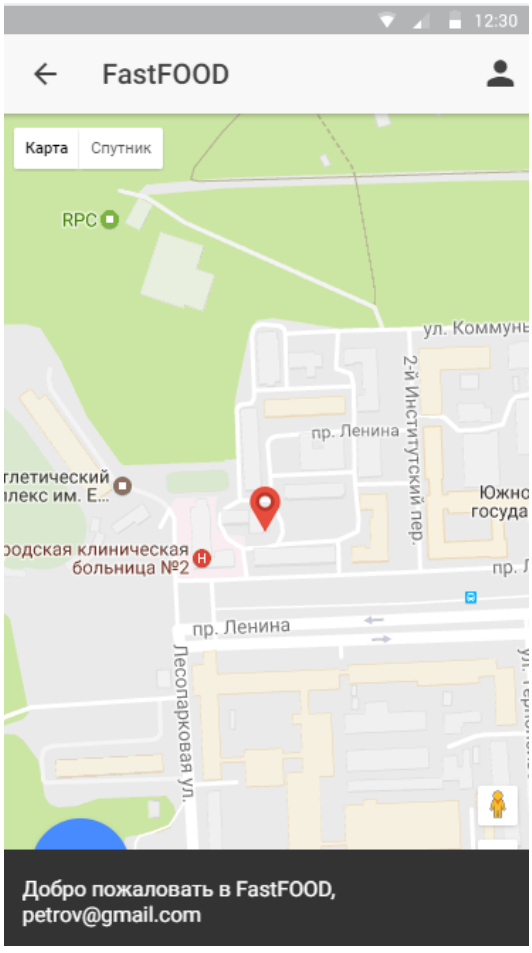

Рисунок 15 — Главный экран мобильного приложения

Создание дальнейших экранов аналогично созданию главного экрана.

3.3 Подключенные модули и библиотеки в проекте

Для подключения к базе данных Firebase Realtime Database используются следующие модули:

1) AngularFireModule — предназначен для инициализации Firebase в проекте;

2) AngularFireAuthModule — предназначен для авторизации пользователей;

3) AngularFireDatabaseModule — предназначен для подключения к базе данных.

Для авторизации пользователей в базе данных используется библиотека «AngularFireAuth» и метод «signInWithEmailAndPassword()». Данный метод сравнивает введённые данные почты и пароля с данными, хранящимися на сервере.

Также библиотека «AngularFireAuth» используется и для регистрации пользователей, с помощью метода «createUserWithEmailAndPassword()». Данный метод сравнивает данные почты и пароля с данными, хранящимися на сервере, если они различны, то создается новый пользователь.

При выборе типа продукции происходит сортировка товаров. В данном случае происходит запрос, содержащий дочерние элементы, соответствующие указанному значению «equalTo».

Листинг 9 — Сортировка по заданному значению

```
filterMainMenuItemByCar(tagCar: string) {
  this.mainMenu = this.db.list('mainMenuList', {
    query: {
      orderByChild: 'tagCar',
      equalTo: tagCar
    }
  });
  this.color = tagCar;
}
```

Так же можно производить сортировку с определёнными ограничениями. Например, использование «startAt()», «endAt()» позволяет выбирать произвольные начальные и конечные точки для наших запросов.

Диалоговые окна создаются с помощью контроллера «AlertController» и метода «create()» (рис. 14). В диалоговых формах можно располагать как обычный текст, так и поля для ввода данных, для последующей отправки на сервер.

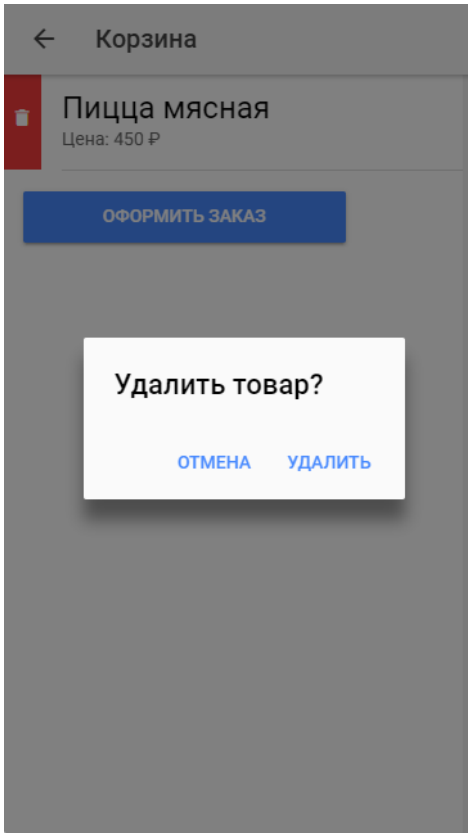

Рисунок 14 — Диалоговое окно удаления товара

Выводы по главе три

Клиент-серверная архитектура самая подходящая для данного типа приложения, так как все данные хранятся на сервере, также все вычисления находятся на сервере, следовательно, требования к клиенту в плане ЭВМ снижается.

Подробное описание разработки одного экрана приложения даёт полное описание работы Ionic Framework, взаимодействия всех его частей.

5 БИЗНЕС-ПЛАН ПРЕДПРИЯТИЯ МОБИЛЬНОГО ЭКСПРЕСС-ПИТАНИЯ

4.1 Бизнес-план до внедрения мобильного приложения

Рассмотрим проект бизнес-плана компании, для которой разрабатывается мобильное приложение, в период с 01.01.2017 по 01.05.2017. В разделе «Движение денежных средств» можно увидеть выручку от реализации продукции.

Таблица 2 — Раздел бизнес-плана до внедрения мобильного приложения

Движение денежных средств	Единица измерений показателей	01.01	01.02	01.03	01.04	01.05
Выручка от реализации	тысяч рублей	**227,5**	**248,5**	**273**	**297,5**	**325,5**
Итого приток	тысяч рублей	227,5	248,5	273	297,5	325,5
Себестоимость без амортизации	тысяч рублей	209,925	224,075	235,45	246,825	259,825
Расходы без прибыли	тысяч рублей	2,46375	3,49125	5,46862	7,446	9,70419
Итого отток	тысяч рублей	206,012	227,205	240,533	253,886	269,089
Поток денежных средств	тысяч рублей	**21,487**	**21,294**	**32,466**	**43,613**	**56,410**
Поток денежных средств нарастающим итогом	тысяч рублей	23,4874	44,781	77,2481	120,862	177,27

Отрасль мобильного экспресс-питания отличается своей сезонностью. Так весенне-осенний период является более прибыльным в сравнении с другими. Это подтверждается изменением выручки от реализации.

4.2 Бизнес-план после внедрения мобильного приложения

Затраты на разработку мобильного приложения в корпоративном секторе относятся к инвестиционным. Инвестиционные затраты — совокупные расходы, которые связаны с реализацией бизнес-проектов [4].

В данном случае затраты для реализации бизнес-проекта по разработке мобильного приложения будут следующие:

1) заработная плата разработчикам;
2) маркетинговые затраты.

Так как средство разработки позволяет разработать приложение гибридным и кроссплатформенным, то стоимость разработки приложения под другие операционные системы будет минимальной.

Уделим особое внимание маркетинговым затратам. Для увеличения популярности мобильного приложения производятся следующие мероприятия:

1) контекстная реклама в социальных сетях;
2) реклама в других приложения;
3) листовки с акциями.

Маркетинговые мероприятия и стоимость их проведения отражены в таблице 3.

Таблица 3 — Маркетинговые мероприятия и стоимость их проведения

Маркетинговое мероприятие	Стоимость проведения, руб./мес.
Контекстная реклама в социальных сетях	20000
Реклама в других приложениях	15000
Листовки с акциями	5000

Предполагается, что с внедрением мобильного приложения, выручка предприятия повысится в период с 01.05.2017 по 01.09.2017 месяцев до 30% (таблица 4).

Таблица 4 — Бизнес-план после внедрения мобильного приложения

Движение денежных средств	Единица измерений показателей	01.05	01.06	01.07	01.08	01.09
Выручка от реализации	тысяч рублей	**297,08**	**318,36**	**360,36**	**385**	**378**
Итого приток	тысяч рублей	297,08	318,36	360,36	385	378
Себестоимость без амортизации	тысяч рублей	246,23	264,91	284,41	295,85	292,6
Расходы без прибыли	тысяч рублей	6,277	6,667	10,11	12,157	11,659
Итого отток	тысяч рублей	244,95	271,181	293,862	307,623	304,372
Поток денежных средств	тысяч рублей	**52,125**	**47,178**	**66,497**	**77,376**	**73,627**
Поток денежных средств нарастающим итогом	тысяч рублей	72,125	119,303	185,800	263,177	336,805

Выводы по главе четыре

Бизнес-план предприятия говорит о сезонности данной сферы бизнеса.

Несмотря на маркетинговые мероприятия предполагается, после внедрения мобильного приложения, увеличение выручки от реализации.

В качестве показателей движения денежных средств после внедрения мобильного приложения были взяты предполагаемые значения в период с 01.05.2017 по 01.09.2017.

ЗАКЛЮЧЕНИЕ

В ходе проведённой работы был совершён поиск конкурентных приложений для предприятий мобильного экспресс-питания, который выявил, что на территории Российской Федерации нет аналогичных приложений. Расширив поиск были отобраны 3 мобильных приложения схожего функционала с разрабатываемым, был произведён их обзор и сравнение с разрабатываемым приложением.

Так же был проведён выбор модели базы данных для разрабатываемого приложения. Выбранная база данных Firebase Realtime Database является NoSQL БД и располагается на серверах Firebase.

Была составлена и подробно описана структурная схема разрабатываемого приложения.

Был выбран Ionic Framework как средство разработки, так как структура основывается на web-технологиях. Так же приложения является кроссплатформенным. В приложении осуществлено регистрация и аутентификация пользователей, поиск текущего местоположение пользователя и точек продаж, а также просмотр меню каждой точки продаж и возможность сделать онлайн-заказ.

Оценка бизнес-плана компании и оценка затрат на разработку и внедрение мобильного приложения позволяют предположить движение денежные средств на период с 01.05.2017 по 01.09.2017. В ходе которого наблюдается увеличение прибыли компании.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Официальный сайт street food в Киеве. – Режим доступа: http://furgoneta.com.ua/
2. Официальный сайт StreetEat. – Режим доступа: http://www.streeteat.it/it/home/
3. Официальный сайт приложения Food Trucks. – Режим доступа: http://www.eatkater.com/
4. Инвестиционные затраты в бизнес-планировании. – Режим доступа: http://www.openbusiness.ru/business-planning/planirovanie/investitsionnye-zatraty-v-biznes-planirovanii/
5. Ionic2. Build Mobile Apps with HTML5. – Режим доступа: https://www.joshmorony.com/tag/ionic2/
6. Documentation Apache Cordova. – Режим доступа: https://cordova.apache.org/docs/en/latest/
7. Базы данных SQL, NoSQL и различия в моделях баз данных. – Режим доступа: http://devacademy.ru/posts/sql-nosql/
8. Documentation Firebase. – Режим доступа: https://firebase.google.com/docs/
9. Ionic Framework – Developers. – Режим доступа: https://ionicframework.com/developers/
10. Ionic Framework. // Документация Ionic Framework – Режим доступа: http://www.ionic-framework.ru/dokumentaciya/
11. Сайт Константина Кочетова Firebase.INFO – Режим доступа: https://firebase-info.com/
12. Развитие рынка фаст-фуда в России. – Режим доступа: http://allbest.ru/marketing/9000349467.html

ПРИЛОЖЕНИЯ

Приложение А

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Разработка Android-приложения для предприятия мобильного экспресс-питания

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2017.067.05.000 ПЗ

Нормоконтролер,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Руководитель работы,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Автор работы
Студент группы ЕТ-485
_____Р.В. Байтимиров
_____2017 г.

Челябинск, 2017

1. ВВЕДЕНИЕ

1.1. Наименование программного изделия

Полное наименование программы — «Разработка Android-приложения для предприятия мобильного экспресс-питания». Краткое наименование программы — приложение экспресс-питания.

1.2. Область применения

Приложение экспресс-питания предназначено для нахождения точек продаж предприятий мобильного экспресс-питания. Так же приложение экспресс-питания может использоваться для просмотра меню и для онлайн-заказа продукции.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Документ, на основании которого ведется разработка

Разработка ведётся на основании задания выпускной квалификационной работы.

2.2. Организация, утвердившая этот документ, и дата его утверждения

Задание на ВКР утверждено руководителем работы, к.ф.-м.н., доцентом кафедры МиКМ Макаровских Т.А.

2.3. Наименование темы разработки

Наименование темы разработки — FastFOOD.

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Разработка является частью выпускной квалификационной работы.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1.Требования к функциональным характеристикам

4.1.1. Состав выполняемых функций

4.1.1.1. Приложение экспресс-питания должно позволять зарегистрировать пользователя в системе.

4.1.1.2. Приложение экспресс-питания должно предоставлять пользователю данные о местонахождении точек продаж предприятия мобильного экспресс-питания.

4.1.1.3. Приложение экспресс-питания должно предоставлять пользователю данные о меню продукции каждой точки продаж предприятия мобильного экспресс-питания.

4.1.1.4. Приложение экспресс-питания должно предоставлять пользователю возможность сделать онлайн-заказ продукции. Данные о заказе должны быть переданы на веб-сервер.

4.1.1.5. Приложение экспресс-питания должно уведомлять пользователя о готовности заказа.

4.1.2. Организация входных и выходных данных

Организация входных и выходных данных должна соответствовать ПРИЛОЖЕНИЮ 2.

В процессе работы приложения экспресс-питания входной информацией должны являться выбранные пользователем данные о заказе.

4.2. Требования к надежности

4.2.1. Требования к надежному функционированию

Программа должна нормально функционировать при бесперебойной работе ЭВМ. При возникновении сбоя в работе аппаратуры восстановление нормальной работы программы должно производиться после:

1) перезагрузки операционной системы;

2) запуска исполняемого файла программы.

Уровень надёжности программы должен соответствовать технологии программирования, предусматривающей:

1) инспекцию исходных текстов программы;

2) автономное тестирование модулей программы;

3) тестирование сопряжений модулей программы;

4) комплексное тестирование программы.

4.2.2. Время восстановление после отказа

Время восстановления после отказа должно состоять из:

1) времени запуска пользователем исполняемого файла программы;

2) времени повторного ввода потерянных данных.

4.3. Условия эксплуатации

Программа должна храниться в виде двух маркированных копий: эталонной и рабочей. Периодическая перезапись информации должна осуществляться согласно нанесенной маркировке.

4.4. Требования к составу и параметрам технических средств

Программа должна корректно работать на следующем или совместимом с ним оборудовании:

1) смартфон с операционной системой Android.

4.5. Требования к информационной и программной совместимости

4.5.1. Требования к информационным структурам на входе и выходе

Требования к информационным структурам на входе и выходе определены в п. 4.1.2.

4.5.2 Требования к методам решения

Выбор методов решения осуществляется без согласования с заказчиком.

4.5.3 Требования к языкам программирования

Язык программирования выбирается разработчиком без согласования с заказчиком.

4.5.4. Требования к программным средствам, используемым программой

Для работы приложения экспресс-питания необходим смартфон с операционной системой Android 5.1.1 или выше.

4.6. Требования к маркировке и упаковке

Диски с эталонным и рабочими экземплярами программы должны иметь маркировку, состоящую из надписи «Дипломная работа. Байтимиров Р.В. ЕТ-485, 2017», надписи «эталон» или «рабочая», даты последней перезаписи программы. Упаковка должна соответствовать условиям хранения диска. На упаковке должны быть указаны условия транспортирования и хранения диска.

4.7. Требования к транспортированию и хранению

Условия транспортирования и хранения диска должны соответствовать п. 4.6.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать следующие документы:

1) технический проект программы по ГОСТ 19.404-79 в машинописном исполнении;

2) описание программы по ГОСТ 19.402-78 на компакт-диске;

3) текст программы по ГОСТ 19.401-78 на компакт-диске;

4) руководство программиста по ГОСТ 19.504-79 на компакт-диске в виде файла README.TXT.

Пояснительная записка «Технический проект программы» должна содержать следующие разделы.

1. Раздел «Входные данные» (характер, организация входных данных).

2. Раздел «Выходные данные» (характер и организация выходных данных).

3. Раздел «Описание логической структуры» при технологии структурного программирования должен включать следующие материалы:

- описание связей программы с другими программами;
- описание внутренних массивов и переменных, которые используются в межмодульном обмене данными;
- схема иерархии программы (приводится рисунок);
- расшифровка наименования модулей (приводится таблица с перечнем наименований модулей в алфавитном порядке с указанием выполняемой каждым модулем функции);
- описание функционирования программы с учетом её модульного деления (приводится словесное описание выполнения программы с учетом вызовов модулей);
- описание модулей программы (подраздел заполняется на основе паспортов модулей).

4. Раздел «Используемые технические средства» (типы ПК, на которых возможно выполнение программы; устройства, используемы при выполнении программы).

5. Раздел «Вызов и загрузка» (виды носителей программы, их используемый объем; способы вызова программы с соответствующих носителей данных; входные точки в программу – запуск программы).

6. Раздел «План мероприятий по разработке и внедрению программы» (план мероприятий разрабатывается для реализации программы коллективом программистов; планом должны быть предусмотрены контрольные временные точки реализации; приводится состав тестов и принципы для каждой из контрольных точек).

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели должны определяться заказчиком без участия исполнителя.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Разработка программы должна выполняться по следующим этапам:

1) разработка, согласование и утверждение технического проекта программы с пояснительной запиской – 5 недель;

2) разработка рабочего проекта программы с комплексным тестированием – 6 недель;

3) приемка-сдача с исправлением обнаруженных недостатков в программе и программной документации – 2 недели.

4) внедрение.

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Виды испытаний

Испытания программы и верификация документации должны производиться в организации заказчика с привлечением сторонних экспертов. Проверочные тесты должны готовиться заказчиком.

8.2. Общие требования к приемке

Приемка программы должна осуществляться заказчиком. Программа должна считаться годной, если она удовлетворяет всем пунктам данного технического задания.

Приложение Б

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Разработка Android-приложения для предприятия мобильного экспресс-питания

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2017.067.05.000 ПЗ

Нормоконтролер,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Руководитель работы,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Автор работы
Студент группы ЕТ-485
_____Р.В. Байтимиров
_____2017 г.

Челябинск, 2017

1. ОБЩИЕ СВЕДЕНИЯ

Мобильное приложение «FastFOOD» предназначено для решения следующих задач:

1) просмотр информации о местонахождении точек продаж, меню продукции;
2) регистрация пользователей;
3) оформление онлайн-заказов пользователями.

2. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Выпадающий список – элемент управления, представляющий из себя надпись, нажатие на которую открывает список возможных вариантов.

Всплывающее уведомление – короткое сообщение, отображаемо в нижней части экрана мобильного телефона короткий промежуток времени.

Диалоговое окно – элемент управления, отображаемы поверх текущего окна, предназначенный для ввода некоторый данных, либо требующий от пользователя принятия некоторого решения.

3. ПРАВИЛО ВВОДА ВХОДНЫХ ДАННЫХ

Корректность ввода большинства данных контролируются приложением. Ввод некорректных данных зачастую невозможен. Исключением является поле ввода адреса электронной почты на экране регистрации. Введенный адрес должен быть существующим, в противном случае окончание регистрации будет невозможным. При этом, соответствие введённого текста шаблону адреса электронной почты также контролируется приложением.

4. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ И РАБОТА С ПРОГРАММОЙ

Пользователю мобильного приложения «FastFOOD» предлагается графический интерфейс, состоящий из стандартных элементов управления ОС Android:

- списков;
- полей ввода данных;
- выпадающих списков;
- кнопок;
- надписей;
- диалоговых окон.

На рисунке 1 приведен пример экрана, содержащего надписи, одну кнопку и кнопку, вызывающее диалоговое окно.

На рисунке 2 приведен пример вызываемого диалогового окна кнопкой на рисунке 1.

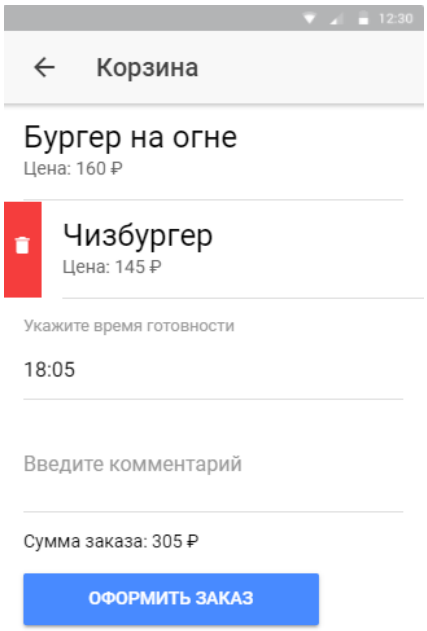

Рисунок 1 — Пример экрана

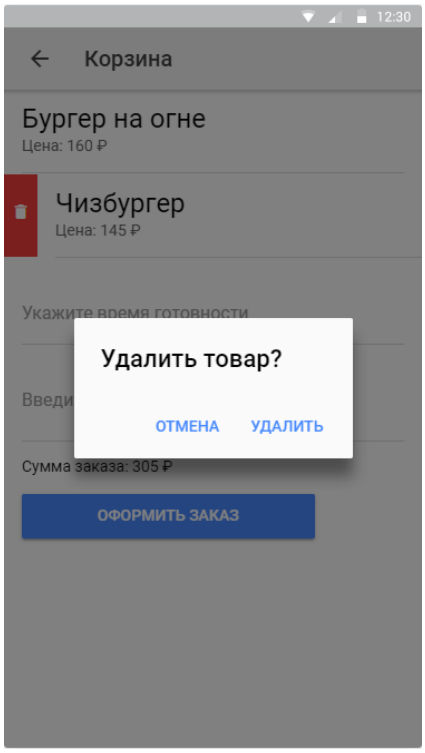

Рисунок 2 — Пример диалогового окна

5. РАБОТА С ПРОГРАММОЙ

Ниже приведены способы решения типовых задач.

РЕГИСТРАЦИЯ В СИСТЕМЕ

На экране авторизации необходимо нажать на кнопку регистрация, расположенную ниже кнопки авторизации. Далее, заполнить следующие поля:

- email — адрес электронной почты пользователя (один адрес может принадлежать только одному пользователю системы);
- пароль — пароль, который будет использоваться при авторизации в системе.

Далее необходимо нажать на кнопку зарегистрироваться. Будет сделан перевод на следующий экран для окончания регистрации — ввода имени и телефона пользователя.

ОФОРМЛЕНИЕ ОНЛАЙН-ЗАКАЗА

На экране с меню продукцией необходимо выбрать вид продукции. Далее необходимо выбрать интересующие товары и нажать на кнопку «В КОРЗИНУ».

Для окончания оформления онлайн-заказа необходимо перейти в корзину и сделать следующие пункты:

- указать время готовности;
- добавить комментарий к заказу при необходимости.

По выполнению этих шагов появится всплывающее уведомление об успешности оформления онлайн-заказа.

Приложение В

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Разработка Android-приложения для предприятия мобильного экспресс-питания

ИСХОДНЫЙ КОД ПРОГРАММЫ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2017.067.05.000 ПЗ

Нормоконтролер,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Руководитель работы,
Доцент каф. МиКМ, к.ф.-м.н., доцент
_____Т.А. Макаровских
_____2017 г.

Автор работы
Студент группы ЕТ-485
_____Р.В. Байтимиров
_____2017 г.

Челябинск, 2017

Приложение В.1 — Файл confix.xml

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="io.ionic.starter" version="0.0.1" xmlns="http://www.w3.org/ns/widg-
ets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>FastFOOD</name>
    <description>FastFOOD app.</description>
    <content src="index.html" />
    <access origin="*" />
    <allow-navigation href="http://ionic.local/*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <preference name="webviewbounce" value="false" />
    <preference name="UIWebViewBounce" value="false" />
    <preference name="DisallowOverscroll" value="true" />
    <preference name="android-minSdkVersion" value="16" />
    <preference name="BackupWebStorage" value="none" />
    <preference name="SplashMaintainAspectRatio" value="true" />
    <preference name="FadeSplashScreenDuration" value="300" />
    <preference name="SplashShowOnlyFirstTime" value="false" />
    <preference name="SplashScreen" value="screen" />
    <preference name="SplashScreenDelay" value="3000" />
    <platform name="android">
        <allow-intent href="market:*" />
        <icon density="ldpi" src="resources/android/icon/drawable-ldpi-icon.png"
/>
        <icon density="mdpi" src="resources/android/icon/drawable-mdpi-icon.png"
/>
        <icon density="hdpi" src="resources/android/icon/drawable-hdpi-icon.png"
/>
        <icon density="xhdpi" src="resources/android/icon/drawable-xhdpi-
icon.png" />
        <icon density="xxhdpi" src="resources/android/icon/drawable-xxhdpi-
icon.png" />
        <icon density="xxxhdpi" src="resources/android/icon/drawable-xxxhdpi-
icon.png" />
        <splash density="land-ldpi" src="resources/android/splash/drawable-land-
ldpi-screen.png" />
        <splash density="land-mdpi" src="resources/android/splash/drawable-land-
mdpi-screen.png" />
        <splash density="land-hdpi" src="resources/android/splash/drawable-land-
hdpi-screen.png" />
        <splash density="land-xhdpi" src="resources/android/splash/drawable-
land-xhdpi-screen.png" />
        <splash density="land-xxhdpi" src="resources/android/splash/drawable-
land-xxhdpi-screen.png" />
        <splash density="land-xxxhdpi" src="resources/android/splash/drawable-
land-xxxhdpi-screen.png" />
        <splash density="port-ldpi" src="resources/android/splash/drawable-port-
ldpi-screen.png" />
        <splash density="port-mdpi" src="resources/android/splash/drawable-port-
mdpi-screen.png" />
        <splash density="port-hdpi" src="resources/android/splash/drawable-port-
hdpi-screen.png" />
        <splash density="port-xhdpi" src="resources/android/splash/drawable-
port-xhdpi-screen.png" />
        <splash density="port-xxhdpi" src="resources/android/splash/drawable-
port-xxhdpi-screen.png" />
        <splash density="port-xxxhdpi" src="resources/android/splash/drawable-
port-xxxhdpi-screen.png" />
    </platform>
```

```
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
        <icon height="57" src="resources/ios/icon/icon.png" width="57" />
        <icon height="114" src="resources/ios/icon/icon@2x.png" width="114" />
        <icon height="40" src="resources/ios/icon/icon-40.png" width="40" />
        <icon height="80" src="resources/ios/icon/icon-40@2x.png" width="80" />
        <icon height="120" src="resources/ios/icon/icon-40@3x.png" width="120"
/>
        <icon height="50" src="resources/ios/icon/icon-50.png" width="50" />
        <icon height="100" src="resources/ios/icon/icon-50@2x.png" width="100"
/>
        <icon height="60" src="resources/ios/icon/icon-60.png" width="60" />
        <icon height="120" src="resources/ios/icon/icon-60@2x.png" width="120"
/>
        <icon height="180" src="resources/ios/icon/icon-60@3x.png" width="180"
/>
        <icon height="72" src="resources/ios/icon/icon-72.png" width="72" />
        <icon height="144" src="resources/ios/icon/icon-72@2x.png" width="144"
/>
        <icon height="76" src="resources/ios/icon/icon-76.png" width="76" />
        <icon height="152" src="resources/ios/icon/icon-76@2x.png" width="152"
/>
        <icon height="167" src="resources/ios/icon/icon-83.5@2x.png" width="167"
/>
        <icon height="29" src="resources/ios/icon/icon-small.png" width="29" />
        <icon height="58" src="resources/ios/icon/icon-small@2x.png" width="58"
/>
        <icon height="87" src="resources/ios/icon/icon-small@3x.png" width="87"
/>
        <splash height="1136" src="resources/ios/splash/Default-568h@2x~iph-
one.png" width="640" />
        <splash height="1334" src="resources/ios/splash/Default-667h.png"
width="750" />
        <splash height="2208" src="resources/ios/splash/Default-736h.png"
width="1242" />
        <splash height="1242" src="resources/ios/splash/Default-Landscape-
736h.png" width="2208" />
        <splash height="1536" src="resources/ios/splash/Default-Land-
scape@2x~ipad.png" width="2048" />
        <splash height="2048" src="resources/ios/splash/Default-Land-
scape@~ipadpro.png" width="2732" />
        <splash height="768" src="resources/ios/splash/Default-Land-
scape~ipad.png" width="1024" />
        <splash height="2048" src="resources/ios/splash/Default-Por-
trait@2x~ipad.png" width="1536" />
        <splash height="2732" src="resources/ios/splash/Default-Por-
trait@~ipadpro.png" width="2048" />
        <splash height="1024" src="resources/ios/splash/Default-Por-
trait~ipad.png" width="768" />
        <splash height="960" src="resources/ios/splash/Default@2x~iphone.png"
width="640" />
        <splash height="480" src="resources/ios/splash/Default~iphone.png"
width="320" />
    </platform>
    <engine name="android" spec="^6.2.3" />
    <plugin name="cordova-plugin-console" spec="^1.0.5" />
    <plugin name="cordova-plugin-device" spec="^1.1.4" />
    <plugin name="cordova-plugin-splashscreen" spec="^4.0.3" />
    <plugin name="cordova-plugin-statusbar" spec="^2.2.2" />
    <plugin name="cordova-plugin-whitelist" spec="^1.3.1" />
    <plugin name="ionic-plugin-keyboard" spec="^2.2.1" />
</widget>
```

Приложение В.2 — Файл index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="UTF-8">
  <title>FastFOOD</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-
scale=1.0, maximum-scale=1.0, user-scalable=no">
  <meta name="format-detection" content="telephone=no">
  <meta name="msapplication-tap-highlight" content="no">
  <link rel="icon" type="image/x-icon" href="assets/icon/favicon.ico">
  <link rel="manifest" href="manifest.json">
  <meta name="theme-color" content="#4e8ef7">
  <script src="cordova.js"></script>
  <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD-
mOUN_WEi63W7QgUnbHXYQoVS6KQAts1c"></script>
  <script src="https://unpkg.com/querybase@0.6.0"></script>
  <link href="build/main.css" rel="stylesheet">
</head>
<body>
  <ion-app></ion-app>
  <script src="build/polyfills.js"></script>
  <script src="build/main.js"></script>
</body>
</html>
```

Приложение В.3 — Файл app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { ErrorHandler, NgModule } from '@angular/core';
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';
import { AngularFireModule } from "angularfire2";
import { AngularFireAuthModule } from "angularfire2/auth";
import { AngularFireDatabaseModule } from 'angularfire2/database';
import { MyApp } from './app.component';
import { FIREBASE_CONFIG } from './app.firebase.config';
@NgModule({
  declarations: [
    MyApp
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    AngularFireModule.initializeApp(FIREBASE_CONFIG),
    AngularFireAuthModule,
    AngularFireDatabaseModule
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
export class AppModule {}
```

Приложение В.4 — Файл login.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Вход</ion-title>
  </ion-navbar>
</ion-header>
<ion-content text-center class="no-scroll">
      <ion-item padding-left padding-right>
            <ion-label floating>Введите Email</ion-label>
            <ion-input type="text" [(ngModel)]="user.email" clearInput></ion-in-
put>
      </ion-item>
      <ion-item padding-left padding-right>
            <ion-label floating>Введите пароль</ion-label>
            <ion-input type="password" [(ngModel)]="user.password" clear-
Input></ion-input>
      </ion-item>
      <button ion-button margin-left margin-top class="btn-login"
(click)="login(user)">Войти</button>
      <button ion-button color="light" margin-left margin-top class="btn-regis-
ter" (click)="register()">Регистрация</button>

</ion-content>
```

Приложение В.5 — Файл login.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, ToastController } from 'ionic-an-
gular';
import { User } from '../../models/user'
import { AngularFireAuth } from "angularfire2/auth";
@IonicPage()
@Component({
  selector: 'page-login',
  templateUrl: 'login.html',
})
export class LoginPage {
      user = {} as User;
  constructor(private afAuth: AngularFireAuth, private toast: ToastController,
      public navCtrl: NavController, public navParams: NavParams) {
  }
  async login(user: User) {
      try   {
            const result = await this.afAuth.auth.signInWithEmailAndPass-
word(user.email, user.password);
            if (result) {
                  this.navCtrl.push('HomePage');
            }
    }
    catch (e) {
      this.toast.create({
        message: `Неправильный Email или Пароль`,
        duration: 3000
      }).present();
    }
  }
  register() {
      this.navCtrl.push('RegisterPage');
  }
}
```

Приложение В.6 — Файл register.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Регистрация</ion-title>
  </ion-navbar>
</ion-header>
<ion-content text-center class="no-scroll">
      <ion-item padding-left padding-right>
            <ion-label floating>Введите Email</ion-label>
            <ion-input type="text" [(ngModel)]="user.email" clearInput></ion-in-
put>
      </ion-item>
      <ion-item padding-left padding-right>
            <ion-label floating>Введите пароль</ion-label>
            <ion-input type="password" [(ngModel)]="user.password" clear-
Input></ion-input>
      </ion-item>

      <button ion-button margin-left margin-top ion-button class="btn-register"
(click)="register(user)">Зарегистрироваться</button>

</ion-content>
```

Приложение В.7 — Файл register.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, ToastController } from 'ionic-an-
gular';
import { User } from '../../models/user';
import { AngularFireAuth } from "angularfire2/auth";
import firebase from 'firebase';
@IonicPage()
@Component({
  selector: 'page-register',
  templateUrl: 'register.html',
})
export class RegisterPage {
      user = {} as User;
  constructor(private afAuth: AngularFireAuth, private toast: ToastController,
      public navCtrl: NavController, public navParams: NavParams) {
  }
  async register(user: User) {
      try   {
            const result = await this.afAuth.auth.createUserWithEmailAndPass-
word(user.email, user.password);
      this.navCtrl.push('CreateProfilePage');
      firebase.database().ref('users').child(result.uid).set({
        uid: result.uid,
        email: user.email
      });
            }
            catch (e) {
                  this.toast.create({
        message: `Неправильный Email или Пароль`,
        duration: 3000
      }).present();
            }
  }
}
```

Приложение В.8 — Файл create-profile.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Создание профиля</ion-title>
  </ion-navbar>
</ion-header>
<ion-content class="no-scroll">
      <ion-item padding-left padding-right>
            <ion-label floating>Введите имя</ion-label>
            <ion-input type="text" [(ngModel)]="user.name" clearInput></ion-in-
put>
      </ion-item>
      <ion-item padding-left padding-right>
            <ion-label floating>Введите телефон</ion-label>
            <ion-input type="tel" [(ngModel)]="user.phone" clearInput></ion-in-
put>
      </ion-item>
      <button ion-button margin-left margin-top ion-button class="btn-register"
(click)="register(user)">Окончить регистрацию</button>
</ion-content>
```

Приложение В.9 — Файл create-profile.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import firebase from 'firebase';
import { User } from '../../models/user';
import { AngularFireAuth } from "angularfire2/auth";

@IonicPage()
@Component({
      selector: 'page-create-profile',
      templateUrl: 'create-profile.html',
})
export class CreateProfilePage {
      user = {} as User;
      constructor(private afAuth: AngularFireAuth,
            public navCtrl: NavController, public navParams: NavParams) {
      }
      register (user: User) {
            var userID = firebase.auth().currentUser;
            if (userID) {
                  firebase.database().ref('users').child(userID.uid).update({
                        name: user.name,
                        phone: user.phone
                  })
                  this.navCtrl.push('HomePage');
            }
      }
}
```

Приложение В.10 — Файл home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>FastFOOD</ion-title>
    <ion-buttons end>
          <button ion-button icon-only (click)="openUserProfile()">
                  <ion-icon name="md-person"></ion-icon>
            </button>
          </ion-buttons>
```

```
  </ion-navbar>
</ion-header>
<ion-content class="no-scroll">
      <div id="map"></div>

  <button ion-button class="btn-menu" (click)="openMainMenuPage()">Меню</button>
</ion-content>
```

Приложение В.11 — Файл home.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, ToastController } from 'ionic-an-
gular';
import { AngularFireAuth } from "angularfire2/auth";

declare var google;

@IonicPage()
@Component({
      selector: 'page-home',
      templateUrl: 'home.html',
})
export class HomePage {

      map: any;
      infoWindow: any;
      imageUserMarker: any;
      imageCarMarkerBlue: any;
      imageCarMarkerRed: any;
      imageCarMarkerGreen: any;

      constructor(private afAuth: AngularFireAuth, private toast: ToastControl-
ler,
            public navCtrl: NavController, public navParams: NavParams) {

            let that = this;
            setTimeout(function () {
                  that.googleMap();
            }, 2000)
      }
      ionViewDidLoad() {
            this.afAuth.authState.subscribe(data => {
                  if (data && data.email && data.uid) {
                        this.toast.create({
                              message: `Добро пожаловать в FastFOOD,
${data.email}`,
                              duration: 3000
                        }).present();
                  }
                  else {
                        this.toast.create({
                              message: `Не была пройдена аутентификация`,
                              duration: 3000
                        }).present();
                  }
            });
      }
      googleMap() {
            let that = this;
            let userMarker: any;
            let imageCarMarkerBlue: any;
            let imageCarMarkerRed: any;
            let imageCarMarkerGreen: any;
            this.map = new google.maps.Map(document.getElementById('map'), {
```

```
                zoom: 16
            });
            this.imageUserMarker = '../../assets/img/usermarker-1.png';
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(function(position) {
                    var pos = {
                        lat: position.coords.latitude,
                        lng: position.coords.longitude
                    };
                    userMarker = new google.maps.Marker({
                        position: pos,
                        map: that.map,
                        icon: that.imageUserMarker
                    });
                    that.map.setCenter(pos);
                }, function() {
                    this.handleLocationError(true, that.infoWindow,
that.map.getCenter());
                });
            } else {
                // Browser doesn't support Geolocation
                this.handleLocationError(false, that.infoWindow,
that.map.getCenter());
            }
      }
      handleLocationError(browserHasGeolocation, infoWindow, pos) {
            infoWindow.setPosition(pos);
            infoWindow.setContent(browserHasGeolocation ?
                'Error: The Geolocation service failed.' :
                'Error: Your browser doesn\'t support geolocation.');
      }
      openMainMenuPage() {
            this.navCtrl.push('MainMenuPage');
      }
      openUserProfile() {
            this.navCtrl.push('UserProfilePage');
      }
}
```

Приложение В.12 — Файл main-menu.html

```
<ion-header>

      <ion-navbar>
            <ion-title>Сделать заказ</ion-title>
            <ion-buttons end>
                <!--<button ion-button icon-only (click)="addMainMenuItem()">
                    <ion-icon name="md-add"></ion-icon>
                </button>-->
                <button ion-button icon-only (click)="openOrder()">
                    <ion-icon name="md-cart"></ion-icon>
                </button>
            </ion-buttons>
      </ion-navbar>

</ion-header>


<ion-content>
      <ion-list padding-right>
            <ion-item>
                <ion-label>Выберите фургон</ion-label>
                <ion-select id="color" [(ngModel)]="tagCar" (ionChange)="fil-
terMainMenuItemByCar(tagCar)">
```

```
                        <ion-option value="blue">Синий</ion-option>
                        <ion-option value="red">Красный</ion-option>
                  </ion-select>
            </ion-item>
            <ion-item>
                  <ion-label>Тип продукции</ion-label>
                  <ion-select [(ngModel)]="tagItem" (ionChange)="filterMainMenu-
ItemByItem(tagItem)">
                        <ion-option value="burger">Бургер</ion-option>
                        <ion-option value="pizza">Пицца</ion-option>
                  </ion-select>
            </ion-item>
      </ion-list>
      <ion-list>
            <ion-item-sliding *ngFor="let item of mainMenu | async">
                  <ion-item padding-right>
                  <h1 style="white-space: normal;">{{item.title}}</h1>
                        <img src="{{item.image}}" alt="">
                        <h2 style="white-space: normal;">{{item.descrip-
tion}}</h2>
                        <p class="weight">Вес: <span>{{item.weight}}</span>
гр.</p>
                        <p class="cost">Цена: <span> {{item.cost}}</span> ₽</p>
                        <button ion-button block margin-top style="height: 40px;
font-size: 1.4rem" (click)="addMainMenuItemToOrder(item.$key)">В КОРЗИНУ</but-
ton>
                  </ion-item>
                  <ion-item-options style="border: 0;" side="right">
                        <button ion-button color="secondary" (click)="editMain-
MenuItem(item)">
                              <ion-icon name="md-create"></ion-icon>
                        </button>
                  </ion-item-options>
                  <ion-item-options style="border: 0;" side="left">
                        <button ion-button color="danger" (click)="deleteMain-
MenuItem(item.$key)">
                              <ion-icon name="md-trash"></ion-icon>
                        </button>
                  </ion-item-options>
            </ion-item-sliding>
      </ion-list>
</ion-content>
```

Приложение В.13 — Файл main-menu.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, AlertController } from 'ionic-an-
gular';
import { AngularFireDatabase, FirebaseListObservable } from 'angularfire2/data-
base';
import firebase from 'firebase';
import { AngularFireAuth } from "angularfire2/auth";

@IonicPage()
@Component({
  selector: 'page-main-menu',
  templateUrl: 'main-menu.html',
})
export class MainMenuPage {

      mainMenu: FirebaseListObservable<any[]>;
  color: string;
  typeItem: string;
```

```
  constructor(public navCtrl: NavController, public navParams: NavParams, public
alertCtrl: AlertController, public db: AngularFireDatabase) {
      this.mainMenu = db.list(' ');
    this.color = '';
    this.typeItem = '';
  }

  openOrder() {
    this.navCtrl.push('OrderPage');
  }

  addMainMenuItemToOrder(itemID) {
    var userID = firebase.auth().currentUser;
    if (userID) {
      firebase.database().ref('mainMenu-
List').child(this.color).child(itemID).once('value', function(snapshot) {
        firebase.database().ref('users').child(userID.uid).child('or-
der').push(snapshot.val());
      });
    }
  }

  filterMainMenuItemByCar(tagCar: string) {
    this.mainMenu = this.db.list('mainMenuList', {
      query: {
        orderByChild: 'tagCar',
        equalTo: tagCar
      }
    });
    this.color = tagCar;
  }

  filterMainMenuItemByItem(tagItem: string) {
    this.mainMenu = this.db.list('mainMenuList/' + this.color, {
      query: {
        orderByChild: 'tagItem',
        equalTo: tagItem
      }
    });
    this.typeItem = tagItem;
  }

  addMainMenuItem() {
      let prompt = this.alertCtrl.create({

    title: "Добавить товар",

      inputs: [
            {
                  name: 'title',
                  placeholder: "Название",
            },
            {
                  name: 'description',
                  placeholder: "Описание"
            },
            {
                  name: 'image',
                  placeholder: "Ссылка на картинку"
            },
      {
        name: 'weight',
        placeholder: "Вес"
      },
      {
```

```
      name: 'cost',
      placeholder: "Цена"
    },
    {
      name: 'count',
      placeholder: "Количество"
    },
    {
      name: 'tagItem',
      placeholder: "Тег продукта"
    },
    {
      name: 'tagCar',
      placeholder: "Тег фургона"
    }
    ],
    buttons: [
          {
                text: "Отмена"
          },
          {
                text: "Сохранить",
                handler: data => {
                      this.mainMenu.push({
                            title: data.title,
                            description: data.description,
                            image: data.image,
          weight: data.weight,
          cost: data.cost,
          count: data.count,
                            tagItem: data.tagItem,
          tagCar: data.tagCar
                      })
                }
          }
    ]
    })
    prompt.present();
}

editMainMenuItem(item) {
  let prompt = this.alertCtrl.create({
  title: "Редактировать товар",
  inputs: [
    {
      name: 'title',
      placeholder: "Название",
      value: item.title
    },
    {
      name: 'description',
      placeholder: "Описание",
      value: item.description
    },
    {
      name: 'image',
      placeholder: "Ссылка на картинку",
      value: item.image
    },
    {
      name: 'weight',
      placeholder: "Вес",
      value: item.weight
    },
    {
```

```
      name: 'cost',
      placeholder: "Цена",
      value: item.cost
    },
    {
      name: 'count',
      placeholder: "Количество",
      value: item.count
    },
    {
      name: 'tagItem',
      placeholder: "Тег продукта",
      value: item.tagItem
    },
    {
      name: 'tagCar',
      placeholder: "Тег фургона",
      value: item.tagCar
    }
  ],
  buttons: [
    {
      text: "Отмена"
    },
    {
      text: "Сохранить",
      handler: data => {
        let newTitle:String = item.title;
        let newDescription:String = item.description;
        let newImage:String = item.image;
        let newWeight:String = item.weight;
        let newCost:String = item.cost;
        let newCount:String = item.count;
        let newTagItem:String = item.tagItem;
        let newTagCar:String = item.tagCar;

        if(data.title != '') {
          newTitle = data.title;
        }

        if(data.description != '') {
          newDescription = data.description;
        }

        if(data.image != '') {
          newImage = data.image;
        }

        if(data.weight != '') {
          newWeight = data.weight;
        }

        if(data.cost != '') {
          newCost = data.cost;
        }

        if(data.count != '') {
          newCount = data.count;
        }

        if(data.tagItem != '') {
          newTagItem = data.tagItem;
        }

        if(data.tagCar != '') {
```

```
            newTagCar = data.tagCar;
          }
          this.mainMenu.update(item.$key, {
            title: newTitle,
            description: newDescription,
            image: newImage,
            weight: newWeight,
            cost: newCost,
            count: newCount,
            tagItem: newTagItem,
            tagCar: newTagCar
          })
        }
      }
    ]
    })
    prompt.present();
  }

  deleteMainMenuItem(itemID):void {
    let prompt = this.alertCtrl.create({
    title: "Удалить товар?",
    buttons: [
      {
        text: "Отмена",
      },
      {
        text: "Удалить",
        handler: data => {
          this.mainMenu.remove(itemID);
        }
      }
    ]
    })
    prompt.present();
  }
}
```

Приложение В.14 — Файл order.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, AlertController } from 'ionic-an-
gular';
import { AngularFireDatabase, FirebaseListObservable } from 'angularfire2/data-
base';
import { AngularFireAuth } from "angularfire2/auth";
import firebase from 'firebase';

@IonicPage()
@Component({
  selector: 'page-order',
  templateUrl: 'order.html',
})
export class OrderPage {

      order: FirebaseListObservable<any[]>;

  constructor(private afAuth: AngularFireAuth,
      public navCtrl: NavController, public navParams: NavParams, public
alertCtrl: AlertController, public db: AngularFireDatabase) {
      var userID = firebase.auth().currentUser;
      this.order = db.list('users/'+ userID.uid +'/order');
  }
```

```
  deleteOrderItem(itemID):void {
    let prompt = this.alertCtrl.create({
    title: "Удалить товар?",
    buttons: [
      {
        text: "Отмена",
      },
      {
        text: "Удалить",
        handler: data => {
          this.order.remove(itemID);
        }
      }
    ]
    })
    prompt.present();
  }

}
```

Приложение В.15 — Файл order.html

```
<ion-header>

      <ion-navbar>
            <ion-title>Корзина</ion-title>
      </ion-navbar>

</ion-header>


<ion-content class="no-scroll">
      <ion-list>
            <ion-item-sliding *ngFor="let item of order | async">
                  <ion-item padding-right>
                        <h1 style="white-space: normal;">{{item.title}}</h1>
                        <p class="cost">Цена: <span> {{item.cost}}</span> ₽</p>
                  </ion-item>
                  <ion-item-options style="border: 0;" side="left">
                        <button ion-button color="danger">
                              <ion-icon name="md-trash" (click)="deleteOrder-
Item(item.$key)"></ion-icon>
                        </button>
                  </ion-item-options>
            </ion-item-sliding>

            <ion-item padding-left padding-right>
                  <ion-label floating>Укажите время готовности</ion-label>
                  <ion-datetime displayFormat="HH:mm"></ion-datetime>
            </ion-item>

            <ion-item padding-left padding-right>
            <ion-label floating>Введите комментарий</ion-label>
            <ion-textarea></ion-textarea>
            </ion-item>
            <p padding-left>Сумма заказа: <span> {{item.sum}}</span> ₽</p>
            <button ion-button block margin-top margin-left margin-right
style="height: 40px; font-size: 1.4rem; width: 70%">ОФОРМИТЬ ЗАКАЗ</button>
      </ion-list>
</ion-content>
```