Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные Системы
кафедра
УТВЕРЖДАЮЗав. кафедрой ИС
_____ ВиденинС. А.
подпись инициалы, фамилия
« _____» _____ 2016г.

ДИПЛОМНЫЙ ПРОЕКТ

230201.65 Информационные системы и технологии

Разработка мобильного социального приложения для продуктового магазина

Пояснительная записка

Руководитель Н.В. Молокова
подпись, дата
Выпускник В.А. Тюкпеев
подпись, дата
Нормоконтролер Ю. В. Шмагрис
подпись, дата

Красноярск 2016

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информационные системы»

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ С. А. Виденин
« _____» _____ 2016г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме дипломного проекта

Студенту ТюкпеевуВадиму Анатольевичу
Группа: ЗКИ10-02 Специальность: 230201.65 «Информационные системы и технологии»
Тема выпускной квалификационной работы: «Разработка мобильного социального приложения для продуктового магазина.»
Утверждена приказом по университету № 4043/с от 24.03.2016.
Руководитель ВКР: Н. В. Молокова, кандидат технических наук кафедры «Информационные системы» ИКИТ СФУ.
Исходные данные для ВКР: список требований к разрабатываемой системе, методические указания научного руководителя, учебные пособия, электронные ресурсы.
Перечень разделов ВКР: общие сведения, проектирование ИС, описание ИС.
Перечень графического материала: презентация, выполненная в Microsoft Office PowerPoint 2013.

Руководитель ВКР _____ Н.В. Молокова
(подпись)

Задание принял к исполнению _____ В. А. Тюкпеев
(подпись)

« ___ » _____ 20__ г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка мобильного социального приложения для продуктового магазина» содержит 58 страницы текстового документа,28 иллюстрации,14 использованных источников.

ИНФОРМАЦИОННАЯ СИСТЕМА, ПРОЕКТИРОВАНИЕ, МОБ-ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, СУБД.

Цель проекта - разработка мобильного социального приложения для продуктового магазина, помощью которой будет осуществляться взаимодействие пользователя и персонала продуктового магазина непосредственно через мобильное приложение организации.

Для достижения поставленной цели были выделены и выполнены следующие задачи:

- Выбор платформы и устройства;
- Выбор программных средств и среды разработки;
- Проектирование базы данных;
- Разработка интерфейса проекта;
- Реализация мобильного приложения.

Результатом дипломного проекта является мобильное приложение, позволяющая повысить взаимодействия организации с пользователями.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В информационный век возможность свободно передавать и воспринимать информацию, является главной концептуальной идеей развития современного общества. Одной из проблем которого является выбор нужной информации, сейчас чтобы получить полезную информацию, необходим поиск нужных источников, организация материала, полученного из различных источников, представление информации должным образом, оценка качества, создание, решение, формулирование имеющийся информации.Все это делают получение полезной информации действительно трудновыполнимой задачей.

В современных условиях развития необходимо обеспечивать людей качественной обработанной и полезной информацией, для удовлетворения информационной потребностью интересующей человека в определенной предметной области. В современной разработке программного обеспечения, возникает вопрос выбора платформы или устройства, так как сейчас у пользователя большой выбор устройств для получение информации. По этому важным этапом разработки информационных систем, является проектированием, на котором должны будут получены различные ответы касаемые конечного потребителя программного обеспечения.

Целью данного дипломного проекта является разработка мобильногосоциального приложение для продуктового магазина, с помощью которой будет осуществляться взаимодействие пользователя и местной сети магазинов.

Для достижения поставленной цели были выделены и выполнены следующие задачи:

- Выбор платформы и устройства;
- Выбор программных средств и среды разработки;
- Проектированиебазыданных;
- Разработкаинтерфейсапроекта;

- Реализация мобильного приложения.

Основной задачей проектируемой информационной системы является обеспечение связи между пользователями и персоналом технической поддержки местной сети магазинов. Данная информационная система будет использоваться потенциальными клиентами и персоналом местных магазинов, программное средствопредназначенодля передачиполезной информации дляпользователей.

В первой главе данной дипломной работы излагаются основные сведения об информационной системе, описывается общая структура информационной системы, даются определения основным понятиям. Большая часть главы посвящена выбору платформы и устройства, также выбор средств разработки и обзору используемых при разработке технологий. Так же в этой главе рассмотрены понятия баз данных и систем управления базами данных.

Во второй части дипломного проекта выявлены требования к системе, которые послужили основой при ее проектировании. В этой части была создана структура проекта, разработан интерфейс и проведено моделирование.

В третьей главе приведен сценарий работы пользователя с системой, рассмотрены средства обеспечения информационной безопасности системы.

Раздел 1 Общие сведения

1.1 Структура информационной системы

Под информационной системой понимают любую объединенную систему для сбора, структурирования, хранения и передачи информации.

Общая структура информационной системы может представить, как объеденные подсистемы.

Подсистема — это часть системы, выделенная по какому-либо признаку.

Структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем (рисунок 1). Среди обеспечивающих подсистем обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение. [21]

Рисунок 1 - Структура информационной системы

Цель подсистемы информационного программного средства состоит в своевременном создании и выдачикак полезной, так и верной информации для принятия решений.

Информационное обеспечение — это объеденная система кодировании и классификации информации, документации, информационных схем,

распространяющихся в организации, также является методологией проектирований баз данных.

При создании информационных систем необходимо учитывать два аспекта:

- Изучение информации, циркулирующих в организации;
- Проектирование и создание баз данных для обслуживания потребностей организации.

Схема информационных потоков, показывают движения информации и ее объемы,места возникновения первичной информации и использования результатной информации. За счет анализа структуры подобных схем можно выработать меры по совершенствованию всей системы управления.

Построение схем информационных потоков, позволяющих выявить объемы информации и провести ее детальный анализ, обеспечивает:

- Исключение дублирующей и неиспользуемой информации;
- Классификацию и рациональное представление информации.

Методология построения баз данных базируется на теоретических основах их проектирования. Основные идеи методологии можно рассмотреть, как два последовательно реализуемых этапов.

На первом этапе происходит обследование всех функциональных подразделений фирмы с целью:

- Понять специфику и структуру ее деятельности;
- Определить информационные объекты и соответствующий состав реквизитов (параметров, характеристик), описывающих их свойства и назначение.

Второй этап заключается в построении концептуальной информационно-логической модели данных для обследованной на первом этапе сферы деятельности. В этой модели должны быть установлены и оптимизированы все связи между объектами и их реквизитами. Информационно-логическая модель является фундаментом, на котором будет создана база данных.

Для создания информационного обеспечения необходимо:

- Ясное понимание целей, задач, функций всей системы управления организацией;
- Выявление движения информации от момента возникновения и до ее использования на различных уровнях управления, представленной для анализа в виде схем информационных потоков,
- Совершенствованиесистемыдокументооборота;
- Наличие и использование системы классификации и кодирования;
- Владение методологией создания концептуальных информационно-логических моделей, отражающих взаимосвязь информации;
- Создание массивов информации на машинных носителях, что требует наличия современного технического обеспечения.

Техническое обеспечение — это комплекс технических средств, предназначенных для работы информационной системы, а также соответствующая документация на эти средства и технологические процессы

Комплекс технических средств составляют:

- Компьютерылюбыхмоделей;
- Устройства сбора, накопления, обработки, передачи и вывода информации;
- Устройства передачи данных и линий связи;
- Оргтехника и устройства автоматического съема информации;
- Эксплуатационныематериалы.

Документацией оформляются предварительный выбор технических средств, организация их эксплуатации, технологический процесс обработки данных, технологическое оснащение.

Математическое и программное обеспечение — совокупность математических методов, моделей, алгоритмов и программ для реализации целей, и задач информационной системы, а также нормального функционирования комплекса технических средств.

К средствам математического обеспечения относятся:

- Средствамоделированияпроцессовуправления;
- Типовыезадачиуправления;
- Методы математического программирования, математической статистики, теории массового обслуживания и др.

В состав программного обеспечения входят общесистемные и специальные программные продукты, а также техническая документация.

К общесистемному программному обеспечению относятся комплексы программ, ориентированных на пользователей и предназначенных для решения типовых задач обработки информации. Они служат для расширения функциональных возможностей компьютеров, контроля и управления процессом обработки данных.

Специальное программное обеспечение представляет собой совокупность программ, разработанных при создании конкретной информационной системы. В его состав входят пакеты прикладных программ, реализующие разработанные модели разной степени адекватности, отражающие функционирование реального объекта.

Техническая документация на разработку программных средств должна содержать описание задач, задание на алгоритмизацию, экономико-математическую модель задачи, контрольные примеры.

Организационное обеспечение — совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой в процессе разработки и эксплуатации информационной системы.

Организационное обеспечение реализует следующие функции:

- Анализ существующей системы управления организацией, где будет использоваться ИС, и выявление задач, подлежащих автоматизации;
- Подготовку задач к решению на компьютере, включая техническое задание на проектирование ИС и технико-экономическое обоснование ее эффективности;

- Разработку управленческих решений по составу и структуре организации, методологии решения задач, направленных на повышение эффективности системы управления.

Правовое обеспечение — совокупность правовых норм, определяющих создание, юридический статус и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации. Главной целью правового обеспечения является укрепление законности. В состав правового обеспечения входят законы, указы, постановления государственных органов власти, приказы, инструкции и другие нормативные документы министерств, ведомств, организаций, местных органов власти.

В правовом обеспечении можно выделить общую часть, регулирующую функционирование любой информационной системы, и локальную часть, регулирующую функционирование конкретной системы.

Правовое обеспечение этапов разработки информационной системы включает нормативные акты, связанные с договорными отношениями разработчика и заказчика и правовым регулированием отклонений от договора.

Правовое обеспечение этапов функционирования информационной системы включает:

- Статусинформационнойсистемы;
- Права, обязанности и ответственность персонала;
- Правовые положения отдельных видов процесса управления;
- Порядок создания и использования информации.

1.2 Средства разработки

В настоящее время информационные технологии интенсивно развиваются и стремительно внедряются в разные сферы человеческой деятельности. Статичное веб-пространство становиться динамичным, потому

что пользователю нужно получать информацию мгновенно. Для разработки динамичных веб-приложений стандартом становятся такие языки, как HTML, язык гипертекстовой разметки, CSS, язык описания стилейиJavaScript,сценарный язык программирования.

HTML представляет собой набор команд, описывающих структуру документа. Этот язык разметки позволяет выделить в документе отдельные логические части такие как заголовки, абзацы, таблицы, списки, но не задает конкретные атрибуты форматирования. Конкретный вид форматирования определяет сам браузер при чтении документа.

CSS является формальным языком описания внешнего вида документа, написанного с использованием языка разметки. CSS используется веб-разработчиками для определения таких параметров страницы, как цвета, шрифты, расположение отдельных блоков и других аспектов представления внешнего вида.

Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

JavaScript — это язык программирования, позволяющий сделать веб-страницу интерактивной, то есть реагирующей на действия пользователя. Последовательность инструкций (называемая программой, скриптом или сценарием) выполняется интерпретатором, встроенным в обычный браузер. Иными словами, код программы внедряется в HTML-документ и выполняется на стороне клиента. Для выполнения программы даже не нужно перезагружать страницу.

В качестве среды для разработки проекта была выбрана WebStorm от компании JetBrains (рисунок 2). Программа включает в себя набор всех необходимых компонентов для создания интерактивных веб-сайтов. WebStorm, подходит для разработки, как клиентской, так и серверной частей приложения. Главное достоинство WebStorm – это удобный и умный редактор JavaScript, HTML и CSS, который также поддерживает новые технологии и языки. WebStorm делает разработку проще и удобней, обеспечивая подсветку и автодополнение кода, его анализ по ходу редактирования, быструю навигацию, а также предоставляет разработчику мощные инструменты отладки и интеграцию с системами управления версиями. WebStorm понимает структуру проекта и код, обнаруживает возможные проблемы еще до того, как проект открыли в браузере, и предлагает их решение. Среда разработки WebStorm создана на основе платформы IntelliJ IDEA, которая прежде использовалась в качестве плагина в VisualStudio.

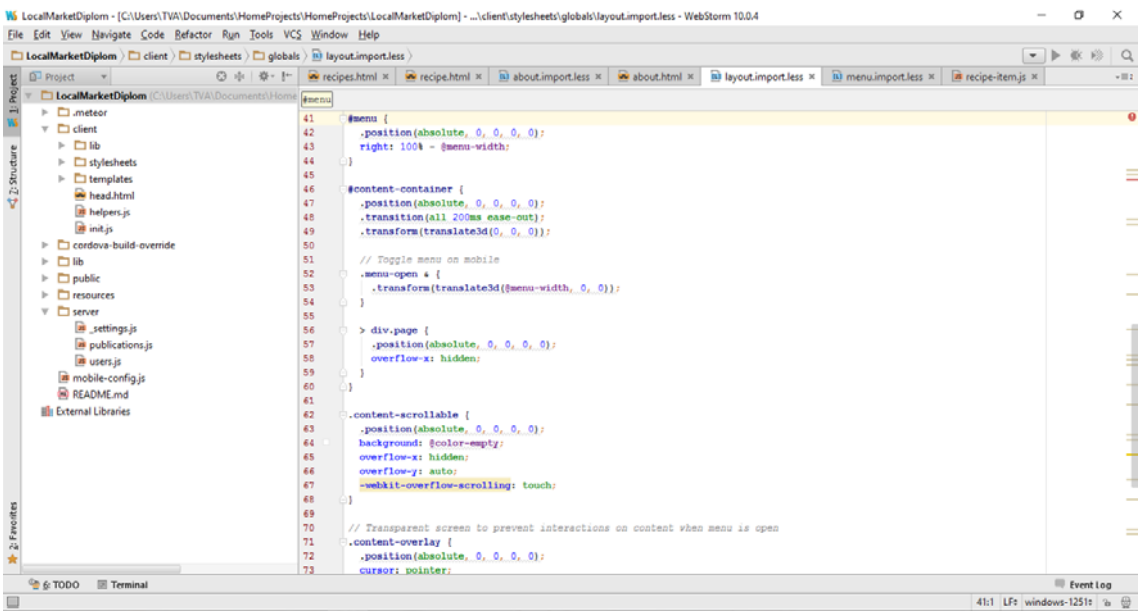


Рисунок 2 - Интерфейс WebStorm

Еще одним программным средством разработки стал фреймворкMeteor.

Фреймворк — в информационных системах, структура программной системы; программное обеспечение, упрощающее процесс разработки и

осуществляющее объединение разных компонентов большого программного проекта. В отличие от библиотек, которые объединяют набор подпрограмм близкой функциональности, фреймворк содержит в себе большое количество разных по назначению библиотек.

Meteor — это веб-фреймворк для создания приложений реального времени. Meteor можно представить, как прослойку между интерфейсом приложения и его базой данных, которая следит за их синхронизацией.

Основнымипринципами Meteor являются:

- Чистый JavaScript. Можно создавать приложения только на JavaScript, так как одни и те же API доступны и на клиенте, и на сервере, в том числе API базы данных. Таким образом один и тот же код можно использовать, как на клиенте так и на сервере;
- Живое обновлений страницы. Шаблоны обновляются при изменении данных в базе, благодаря протоколу DDPданные легко передаются с серверной части в клиентскую;
- Мощная синхронизация данных. Можно писать клиентский код, как будто он работает на сервере и имеет прямой доступ к базе данных;
- Компенсация времени задержки. Когда пользователь вносит изменение обновления данных на экране происходят немедленно — без ожидания отклика сервера. Если сервер отклоняет запрос или выполняет его по-другому на клиенте вносятся исправления для синхронизации с тем, что фактически произошло;
- Полностью автономные зависимости приложения. Одна команда компилирует все приложение в единый архив;
- Умные пакеты. Умные пакеты Meteor — это маленькие программы, которые могут внедрить код на клиенте или на сервере.

В настоящее время большинство приложений на Meteor в качестве системы управления базами данных используют MongoDB, которая работает с

нереляционными базами данных, потому что она поддерживается лучше всего, но на будущее запланирована поддержка и других баз данных.

1.3Базы данных

Сегодня трудно себе представить информационную систему, которая не имела бы базу данных. Концепции и технологии баз данных складывались постепенно и всегда были тесно связанны с развитием систем автоматизированной обработки информации.

База данных — это совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ.

Организация структуры БД формируется исходя из следующих соображений:

- Адекватность описываемому объекту(на уровне концептуальной и логической модели);
- Удобство использования для ведения учёта и анализа данных (на уровне физической модели).

В процессе разработки проекта, использовались базы NoSQL. Термин «NoSQL», в информатике, обозначает ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых реляционных.

NoSQL подход решает проблемы распределения базы данных, проблему строгой структуры данных, присущей реляционной базе данных и упрощают работу с большими данными.

Представителями нереляционных баз данных являются базы данных на основе графов, хранилища «ключ-значение», документо-ориентированные базы данных, колоночные хранилища данных.

Базы данных на основе графов применяются для задач, в которых данные имеют большое количество связей, например, социальные сети.

Хранилища «ключ-значение» являются простейшим типом хранилищ данных, которые используют ключ для доступа к значению. Такие хранилища используются для хранения изображений, создания специализированных файловых систем, в качестве кэшей для объектов;

Документо-ориентированные базы данных служат для хранения иерархических структур данных. Модель данных обладает гибкой структурой и позволяет хранить слабоструктурированные данные;

В колоночных хранилищах данных, данные хранятся в виде разреженной матрицы, строки и столбы которой используются как ключи. Типичным применением этого вида является веб-индексирование, а также задачи, связанные с большими данными, с пониженными требованиями к согласованности данных.

Документо-ориентированная модель данных является самым распространённым представителем нереляционных моделей, которая лежит в основе системы управления базами данныхMongoDB. Данная модель нацелена на хранение иерархических структур данных. Документо-ориентированная модель оперирует понятиями документ, что является аналогом кортежа в реляционной модели, и коллекция, представляющая набор документов.

Документо-ориентированные базы данных имеют гибкую структуру, которая обеспечивает ряд важных особенностей:

- Отсутствие схемы. Документо-ориентированным базам данных не нужна предопределенная структура данных, которую нужно хранить в базе данных. В реляционных базах данных сначала определяется структура таблиц, в которых будут храниться данные, для чего требуется знать наперед содержание, возможные значения и структуру информации. В документо-ориентированной базе данных информация храниться в коллекциях, состоящих из документов произвольного формата.

- Документо-ориентированная модель предусматривает хранение данных в агрегированном, денормализованном виде. Один документ коллекции может храниться информацию о нескольких связанных сущностях «реального мира». Такая модель хранения данных обосновывается выигрышем в производительности, за счет уменьшения количества запросов к базе данных. Высокий уровень денормализации данных приводит к проблемам обновления данных в нескольких документах, а иногда и коллекциях, что увеличивает вероятность допустить ошибку при работе с данными. Таким образом, сущности, которые встречаются несколько раз, выделяют в отдельные коллекции.
- Гибкая структура данных позволяет описывать сущности предметной области любой сложности.

1.4СУБД

База данных предполагает наличие комплекса программных средств, обслуживающих эту базу данных и позволяющих использовать содержащуюся в ней информацию. Такие комплексы программ называют СУБД. Система управления базами данных (СУБД) — это программное обеспечение, с помощью которого пользователи могут создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

Обычно, современная СУБД содержит следующие компоненты:

- Ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
- Процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;

- Подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;
- Сервисные программы, обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

Основными функциями СУБД являются:

- Управление данными во внешней памяти;
- Управление данными в оперативной памяти с использованием дискового кэша;
- Журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- Поддержка языков баз данных (язык определения данных, язык манипулирования данными);
- Определение данных. Определить, какая именно информация будет храниться в базе данных, задать свойства данных, их тип (например, число цифр или символов), а также указать, как эти данные связаны между собой. В некоторых случаях есть возможность задавать форматы и критерии проверки данных;
- Обработка данных. Данные могут обрабатываться различными способами. Можно выбирать любые поля, фильтровать и сортировать данные. Можно объединять данные с другой, связанной с ними, информацией и вычислять итоговые значения. Обращение к базам данных осуществляется с помощью СУБД.

Одним из решений для хранения и обработки данных является СУБД MongoDB.

СУБД MongoDB реализует документ-ориентированную модель данных.

Документ — это набор, состоящий из имен и значений свойств. Значение может быть простым типом, массивом или другим документом.

Ключевое отличие реляционной модели от документ-ориентированной в том, что при использовании реляционной модели данные раскладываются по нескольким таблицам с фиксированной структурой, что приводит к необходимости выполнения соединений для работы с взаимосвязанными данными из разных таблиц. Документ-ориентированная модель представляет данные в агрегированной форме, то есть работать с объектом как с единым целым: все данные, относящиеся к сущности, хранятся в одном объекте базы данных.

1.5Вывод по разделу 1

В ходе ознакомления со структурой информационных систем были изучены основные аспекты построения и функционирования информационных систем. В результате обзора средств разработки был определен набор инструментов, способствующих проектированию и разработке информационной системы. В качестве средств разработки были выбраны: языки HTML, CSS иJavaScript, так как они являются стандартными языками при разработке в веб-пространстве, программное обеспечение WebStorm было выбранно в качестве среды разработки за счет инструментария, который упрощает процесс разработки и написания кода, и Meteor, в качестве платформы разработки (фреймворка) из-за возможности восприятия языка JavaScript как на стороне клиента, так и на стороне сервера. Так же были рассмотрены базы данных NoSQL и СУБД MongoDB.

Раздел 2Проектирование ИС

2.1 Формирование требований

В ходе проектирования социального мобильного приложение для продуктового магазина, были сформированы следующие задачи:

- Упростить процесс взаимодействия организации с клиентами и партнерами через Интернет;
- Создать инструмент распространения информации об организации;
- Объединение клиентов, предоставляя возможность взаимодействия;
- Реализовать «канал» обмена информацией с удаленными пользователями.

2.1.1 Требования к структуре приложения

Требованиями кструктуре приложения являются:

- Главная страница приложения должна показывать последнюю новость, последние загруженные фотографии пользователей,краткий список добавленных рецептов;
- Главное меню должно быть скрыто от пользователя, и появляется только при нажатии кнопки, после нажатия появляется меню;
- Каждый пункт меню должен соответствовать конкретному разделу программы или переводить пользователя на другойраздел;
- Выводить список рецептов, описание рецепта, фотография, ингредиентыж;
- Возможность добавлении рецептов в закладки;
- Возможность пользователем создавать фотографии с помощью приложения;
- Авторизация с помощью социальной сети «Твиттер».

В приложении должны присутствовать следующие разделы:

- Главная;
    - Новости;
    - Последнее приготовленное блюдо;
    - Рекомендованные рецепты;
- Что приготовили (Список последних приготовленных блюд);
- Рецепты;
    - Список всех рецептов;
    - Описание рецепта;
    - Ингредиенты;
    - Фотографии, созданные пользователями;
- Закладки;
- О нас (Виды деятельности, Контакты).

2.1.2 Требования к мобильному приложению

Мобильное приложениедолжно отвечать следующим требованиям:

- Иметь простой интерфейс;
- Предусматривать разделение пользователей на два ролевых модуля (администратор и пользователь);
- Работать в режиме реального времени.

Клиент организации должен иметь возможность:

- Просматривать информацию в системе без регистрации;
- Оставлять фотографии с приготовленными блюдами в системе;
- Регистрироваться в приложении с помощью «Твиттера»;
- Добавлять понравившиеся блюда в закладки;
- Просматривать последние приготовленные блюд.

Администратор системы должен иметь возможность:

- Добавлять новость в системе;

- Добавлять новые рецепты;
- Редактировать рецепты;
- Удалять загруженные фотографии пользователей;
- Блокировать пользователей;
- Просматривать список новых сообщений от пользователей;
- Удалять сообщения пользователей.

2.1.3 Технические требования

Технические требования ко всему проекту:

- Поддержка наиболее популярных операционных систем, таких как AppleIOS, GoogleAndroid, Blackberry, WindowsPhone;
- Использовать стандартные для разработки веб-приложений языки, HTML, CSS, JavaScript.

2.2 Проектирование структуры приложения и разработка интерфейса

Чтобы облегчить разработку нашего приложения на метеор, был примененодин из подходов проектирование программного обеспечения, называемый «outside-in»описанный в книге «GrowingObject-OrientedSoftwareGuidedbyTests», этот подход поощряет, начинать разработку с интерфейса пользователя, а уже потом разрабатывать логику всего приложения.

Это означает, что в первую очередь нужно подумать, как будет разработан, клиентская часть приложения и как будет выглядеть интерфейс пользователя.

2.2.1 Файловая структура приложения

Прежде чем приступить к проектирование и разработке клиентской части приложения, нужно создать правильную файловую структурой приложения в Метеор(рисунок 3).

Метеор имеет специальные директории, которые имеют ряд правил:

- Код на стороне сервера исполняется только в директории «/server»;
- Клиентская часть исполняется только в директории «/client»;
- Все остальные файлы используются и на клиенте, и на сервере;
- Статичные файлы (Изображений, файлов, шрифтов и т.п.) в директории «/public»;
- Файлы в директории /libзагружаются первыми.

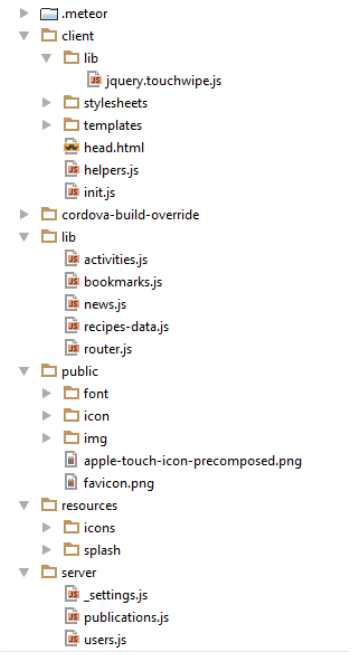


Рисунок 3 –Файловая структура приложения

2.2.2 Структура разделов приложения

В ходе проектирования была разработана структурараздeлов приложения (рисунок 4). На главной странице сайта будут находится разделы «Что

приготовили», «Рецепты», «Закладки», «О нас». Чтобы раскрыть сущность каждого из разделов, внутри них будут использоваться различные элементы, такие как блоки, таблицы или списки.

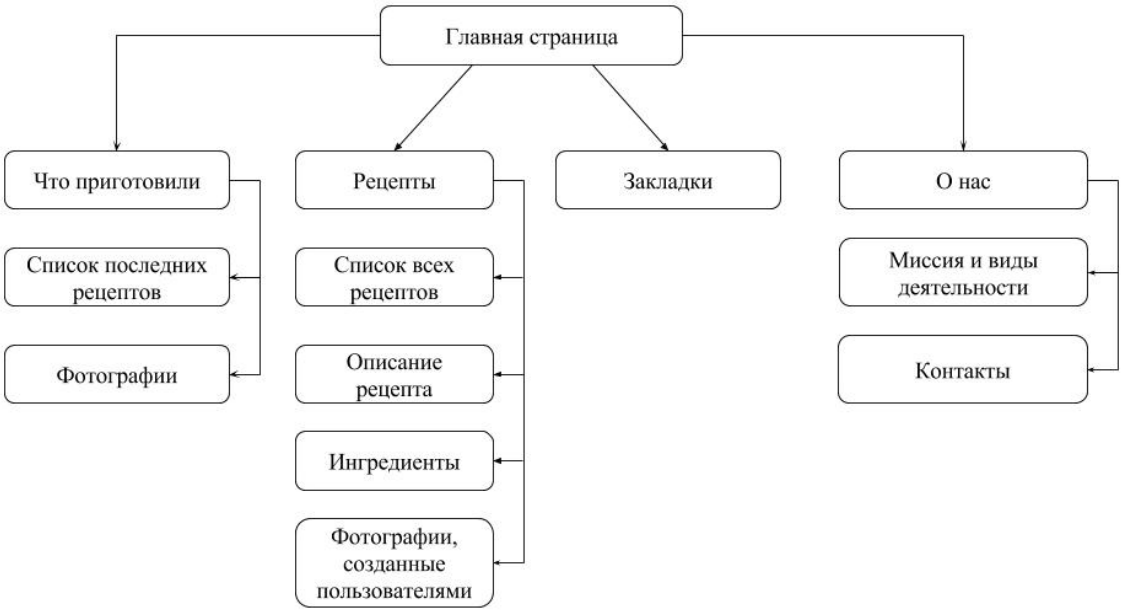


Рисунок 4 - Внутренняя структура приложения

На следующем этапе проектирования главной страницы приложения, была создана семантическая структура приложения, с помощью HTMLэлементов которые разграничены по тегам. На этом этапе необходимо учитывать, что злоупотребление разметкой приводит к разрастанию приложения, усложнению их стилевого оформления и сопровождения.

Теги HTML образуют иерархическую структуру, называемую объектной моделью документа (DOM). DOM — это способ представления объектов, которые определяются через HTML, а затем взаимодействуют интерактивно с помощью сценарного языка, например, JavaScript. Теги HTML определяют элементы DOM — сущности, находящиеся в модели.

Спецификация HTML5 предоставляет новые семантические теги, описывающие содержащийся в них контент. Так как многие разработчики

включают в свои страницы боковые панели, заголовки, завершители и секции, в спецификацию HTML5 были включены новые теги, предназначенные для деления страницы на логические области. Среди таких элементов:

- «header», определение заголовка страницы или раздела;
- «footer», определение завершителя страницы или раздела;
- «nav», определение области навигации страницы или раздела;
- «section», определение логической области страницы.

Для описания основных элементов приложения была создана семантическая разметка (рисунок 5). Семантическая разметка представляет собой макет будущего приложения. Некоторые элементы находятся внутри других. Таким образом,видны взаимоотношения, определяющие, какие элементы будут потомками других элементов в DOM, и это поможет примерно представить, каким будет код HTML.

Первымразделом нашего приложения, является главная страница, которая была структурирована различнымиэлементами, основой которойявляется «body», который представляет собой контейнер для элементов «header» и «section». Элементы «header» и «section» тоже представляют собой контейнеры для более простых элементов, таких как заголовки «h2», списки «ul» и таблицы «table». Элемент «section» используется для определения разделов страницы. В первом разделе страницы будет содержаться изображение «img» с логотипом социального мобильного приложения для продуктовых магазина, и отобразим дату и время внутри элемента «span». Во втором будет представлена последняя новость.Следующий раздел будет содержать последнее приготовленное блюдо пользователем, и фотография самого блюда. В четвертом разделе планируется разместить список любимых рецептов.В последней секции будет создана ссылка на весь список рецептов.

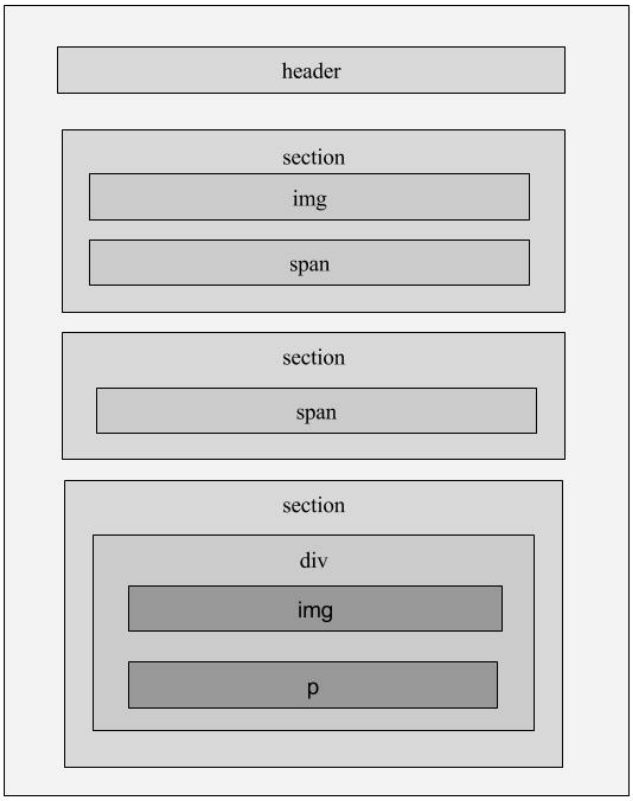


Рисунок 5–Пример семантической разметки страницы приложения

После идентификации основных элементов страницыее можнопредставить в виде древовидная диаграммы, которая определят содержимое этих элементов

Эта диаграмма создает ясное представление о содержимом DOM. Кроме того, она упрощает представление взаимоотношений. Элементы DOM, находящиеся в нижней части дерева рассматриваются, как потомки тех элементов, что находятся выше, если существует путь, связывающий их. Непосредственные потомки называются дочерними элементами, а элементы, находящиеся над дочерними, являются по отношению к последним родительскими элементами. Например, на диаграмме все элементы являются потомками элемента «html», а элемент «ul» является потомком элемента «header». Элемент «ul» имеет четыре дочерних элемента (это пункты списка, элементы «li»).

Для облегчения написания приложения на метеор, используется шаблонизатор «Handlebars», использование шаблоназаторов на стороне клиента, облегчает написание шаблонов, разделяет разметку от кода, делая код понятным и простым для сопровождения.

Handlebarsможно использовать на стороне сервера, но в основном его используют как клиентский шаблонизатор, он принимает JSON данные, из которых генерирует HTML-теги. Основное преимущество шаблонизатора ограничение написание логики и разметки в одном файле, не позволяя разработчикам добавлять произвольный JavaScriptвместе с разметкой.

Рассмотрим один из примеров формирования шаблона HTML, с использованием Handlebars (рисунок 6).

```
<template name="about">
  <div class="page about">
    {{> nav title='О нас' black=true}}

    <div class="content-scrollable">
      <div class="bg-image about">
        <h1 class="title-about">
          <img src="img/app/logo-local.svg" alt="logo-local"/>
        </h1>
      </div>
      <div class="description-about">
        <p>Находитесь где угодно, подключайтесь к сообществу местных магазинов, чтобы получить последние новости, подобр
          анных рецептов , и делиться своими впечатлениями с местными жителями. Вы можете посмотреть список любимых реце
          птов магазина, как вдохновение для готовки, закладки для избранных рецептов, публикация изображений после того
          как вы приготовите блюдо дома, просмотр фотографий которые сделали другие пользователи.
        </p>
      </div>

    </div>
  </div>
</template>
```

Рисунок 6–Пример HTML страницы написанный с помощью Handlebars

После того, как была спроектирована структура приложения и создана структураHTML, необходимо было разработать приятный интерфейс. Внешний вид элементов приложения определялсяза счеткаскадной таблиц стилей.

С помощью CSS можно применить стиль к веб-страницам, чтобы придать им определенный внешнийвид. Работа каскадных таблиц основана на подключении стилей к объектной модели документа. Используя CSS, можно быстро и просто изменить стиль любого элемента.Один из способов

добавления стилей к веб-странице заключается во вставкетребуемых для этого инструкций в заголовок страницы внутри элемента «head»(рисунок 7).

```
<head>
  <meta charset="utf-8">
  <title>Местный магазин</title>
  <meta name="description" content="Получить последние новости, просмотр волшебных рецептов , и делиться фотографиями с
  <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1" />
  <link rel="shortcut icon" type="image/png" href="/favicon.png?123" sizes="16x16 32x32 64x64">

  <link rel="apple-touch-icon" sizes="120x120" href="apple-touch-icon-precomposed.png">
</head>
```

Рисунок 7– Элемент «head»

Meteorавтоматически минифицируетCSS, по этому стили лучше всего хранить в директории «/client» (рисунок 8), минификация файлов это простой подход в уменьшение размеров CSS, JS, HTMLфайлов. В процессе минификации, будут удалены все комментарии к коду, переносы строк, лишние табы и пробельные символы(рисунок 9). Минифицирование файлов очень важная операция так как экономит примерно 20% объема данных, такая операция ускоряет загрузку всего приложения.

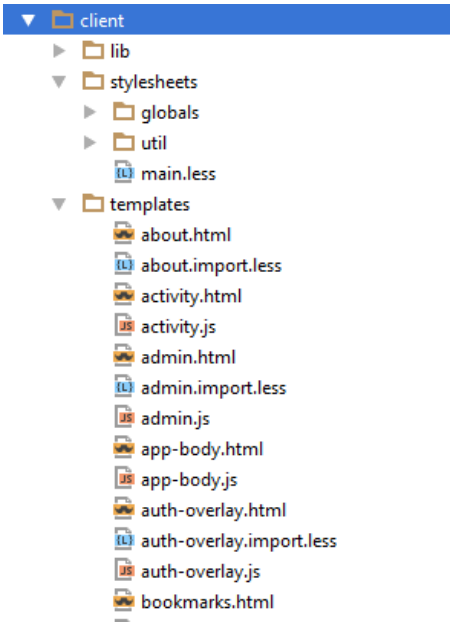
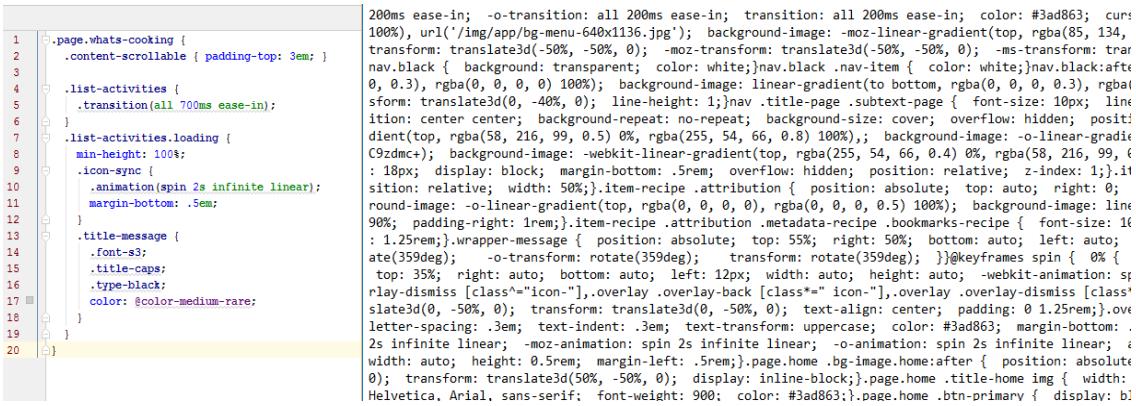


Рисунок 8–Таблица стилей хранится в директории «/client»



```
.page.whats-cooking {
  .content-scrollable { padding-top: 3em; }

  .list-activities {
    .transition(all 700ms ease-in);
  }
  .list-activities.loading {
    min-height: 100%;
    .icon-sync {
      .animation(spin 2s infinite linear);
      margin-bottom: .5em;
    }
    .title-message {
      .font-s3;
      .title-caps;
      .type-black;
      color: @color-medium-rare;
    }
  }
}
```

Рисунок 9–Пример минифицированого файла.

Для упрощение написание каскадных таблиц стилей, был использован метаязык Less на основе CSS. Less расширяет каскадную таблицу стилей, увеличивая уровни абстракций кода, позволяя писать стили профессионального уровня пример кода на языке Less (рисунок 10).

```
// Core
@color-primary: #3ad863; // blood orange
@color-secondary: @color-well; //
@color-tertiary: @color-medium-well; //
@color-complementary: #000; //

// Social Colors
@color-twitter: #5195f5;

// Alert
@color-negative: #ff3642; //error, alert


// Greyscale
@color-empty: white;
@color-raw: #f8f8f8;
@color-raw-alt: #f2f2f2;
@color-rare: #eee;
@color-medium-rare: #ccc;
@color-medium: #aaa;
@color-medium-well: #666;
@color-well: #555;
@color-full: #333;
```

Рисунок 10–Less позволяет использовать переменные.

Lesscовместим со всеми версиями CSS, решает проблему громоздких таблиц стилей, которых тяжело поддерживать в больших проектах. Также Lessпредоставляет огромное количество функций, которые на данный момент

не доступны в самом CSS, некоторые из этих функций такие ка переменные, вложенности, наследование появятся только в новых спецификациях 4 уровня. Но благодаря препроцессорам и постпроцессорам можно использовать сейчас.

2.2.3 Разработка основных страниц приложения

Главная страница содержит большое количество информативных блоков, которые ведут на другие разделы приложения, поэтому было уделено огромное внимание каждому разделу на этой странице. «Заголовок» страницы состоит из элемента «section»и имеет серый фон, «section»содержит элементы страницы, такие как логотип, время, и блок новостей, на данный момент из базы выводится последняя опубликованная новость(рисунок 11).

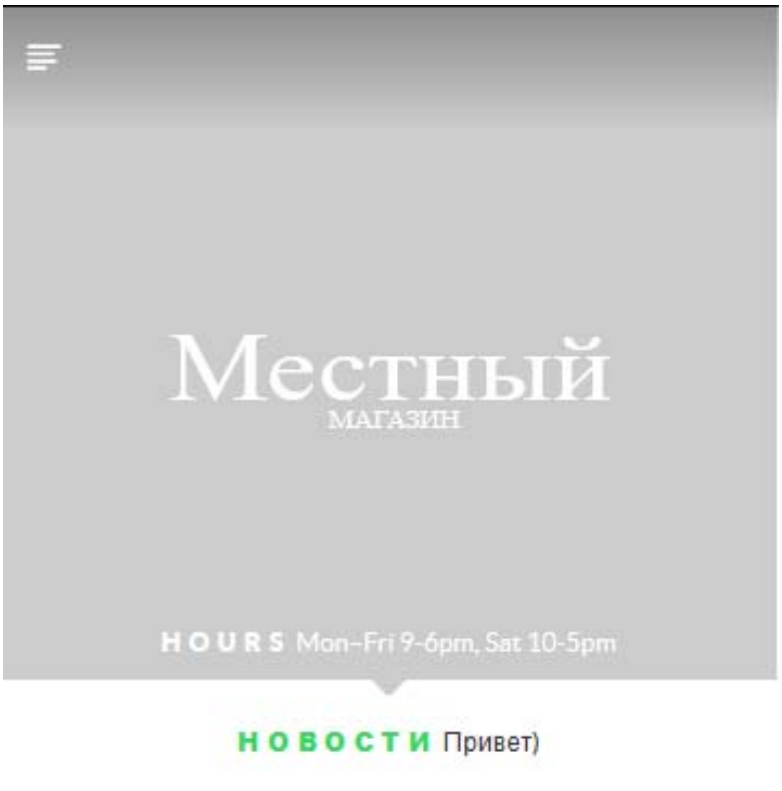


Рисунок 11 - Раздел «Заголовок»

Раздел «Что сейчас готовят?» (рисунок 12) состоит из заголовка«h2»,«section»и ссылки ведущаяна страницу в которой опубликованы

все последние блюда приготовленными пользователями. В элементе «section» выводится последний элемент из списка опубликованных блюд, выводится имя пользователя, название блюда, комментарий,и изображение «img». Таким образом были представлены возможности публикации фотографий пользователям, и просмотр всех фотографий, сделанными пользователями на специальной странице.

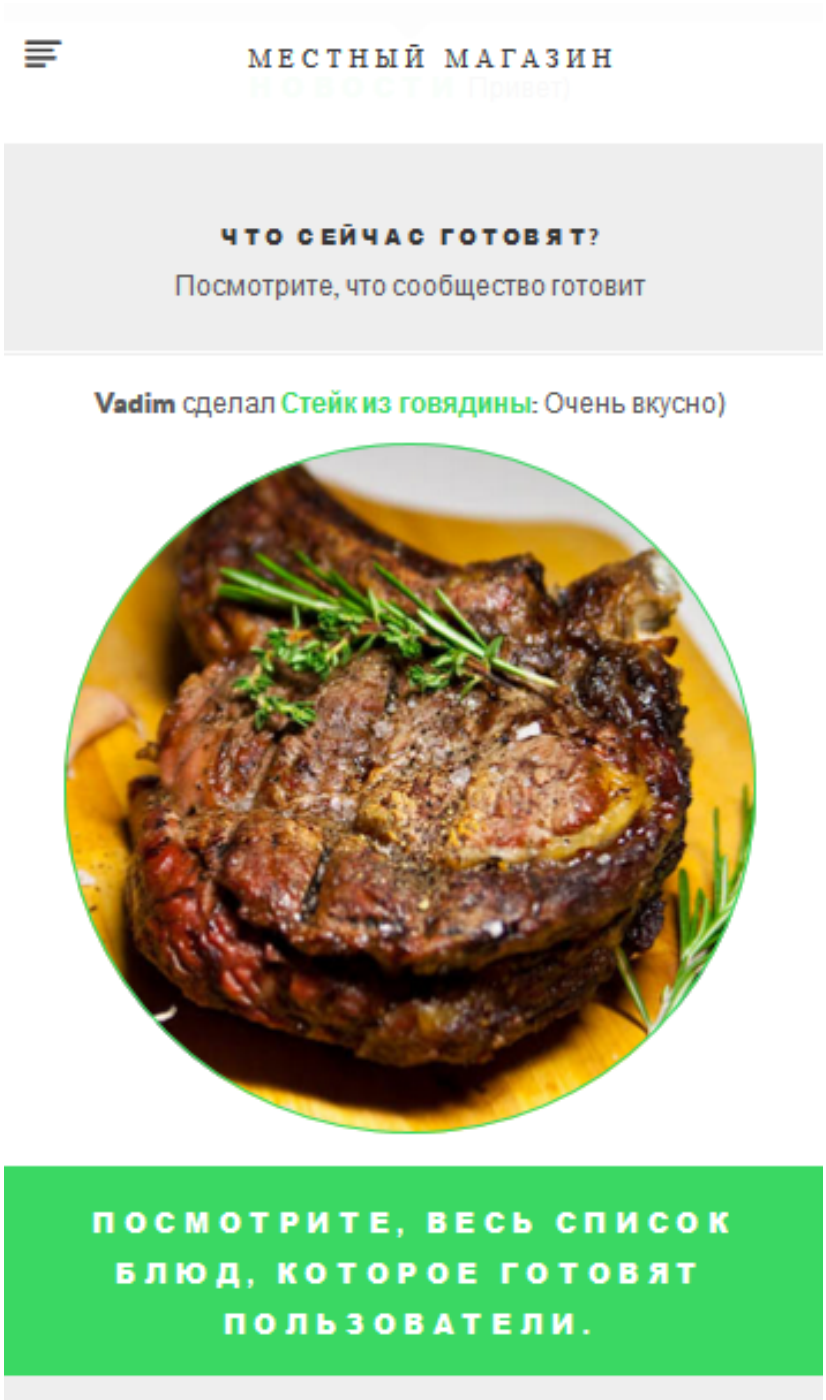


Рисунок 12 - Раздел «Как опробовать»

Раздел «Наши любимые рецепты» (рисунок 13) состоит из заголовка, параграфа и список ссылок внутри которых находятся изображения, ссылки ведут на подборку рецептов, внизу страницы находится ссылка, указывающая на страницу которая содержит все рецепты в базе.

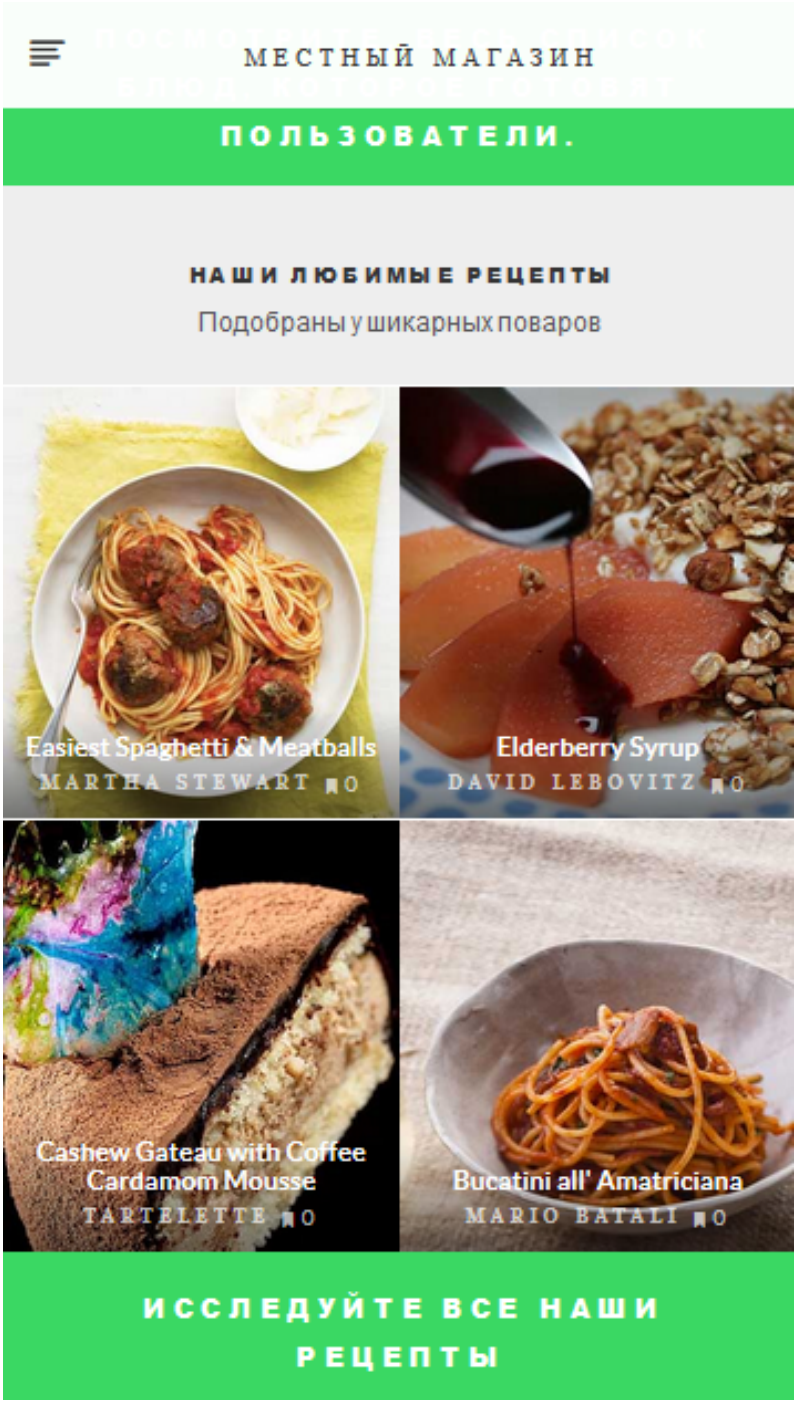


Рисунок 13 - Раздел «Как опробовать»

HTML структура основных страниц приложения (рисунок 14) формируется с помощью стандартных элементов HTML и шаблоназатораHandlebars.

```
    <section class="callout-news">
      <span class="title-callout">Новости</span>
      {{latestNews.text}}
    </section>

    <h2 class="list-title">Что сейчас готовят?</h2>
    <div class="list-subtitle">Посмотрите, что сообщество готовит</div>
    <section class="list-activities">
      {{#each activities}}
        {{> activity}}
      {{else}}
        <div class="wrapper-message">
          <div class="title-message">Никто не готовит еще!</div>
          <div class="subtitle-message">Добавте что вы сегодня приготовили по рецепту, и он будет отображаться здесь.</
        </div>
      {{/each}}
    </section>
    <a href="{{pathFor 'feed'}}" class="btn-primary">Посмотрите, весь список блюд, которое готовят пользователи.</a>

    <h2 class="list-title">Наши любимые рецепты</h2>
    <div class="list-subtitle">Подобраны у шикарных поваров</div>
    <section class="list-recipes">
      {{#each featuredRecipes}}
        {{> recipeItem recipe=this size='small'}}
      {{/each}}
    </section>

    <a href="{{pathFor 'recipes'}}" class="btn-primary">Исследуйте все наши рецепты</a>
  </div>

  </div>
</template>
```

Рисунок 14 - Структура «Главной страницы»

Раздел «О компании» после применения CSS-правилк элементам страницы показан на (рисунок 15).

Одним из решений для настроек стиля определенного элемента являетсяназначение формирующему его коду HTML идентификатора. Например, для фона секции «О компании» было задано CSS-правило «присвоить серый цвет»

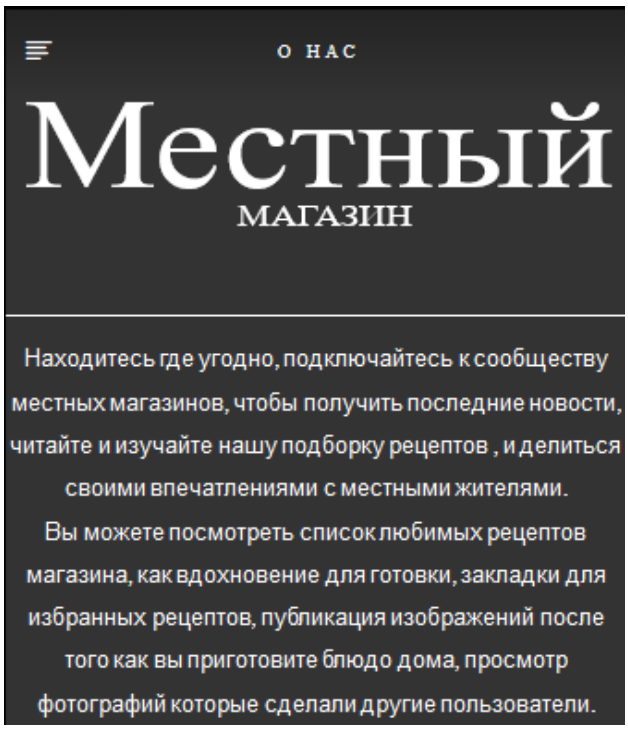


Рисунок 15 - Раздел «О компании»

Внутренняя HTML структура меню сайта представлена в виде элементов ссылок «a» (рисунок 16).

```
<div id="container" class="{{menuOpen}} {{overlayOpen}}">
  <section id="menu" class="overthrow">
    <a href="{{pathFor 'home'}}" class="{{activePage 'home'}}">
      <span class="wrapper-menu-item">
        <span class="icon-home"></span>
        <span class="title-menu">Главная</span>
      </span>
    </a>
    <a href="{{pathFor 'feed'}}" class="{{activePage 'feed'}}">
      <span class="wrapper-menu-item">
        <span class="icon-rss"></span>
        <span class="title-menu">Что готовят?</span>
      </span>
    </a>
    <a href="{{pathFor 'recipes'}}" class="{{activePage 'recipes' 'recipe'}}">
      <span class="wrapper-menu-item">
        <span class="icon-category"></span>
        <span class="title-menu">Рецепты</span>
      </span>
    </a>
    <a href="{{pathFor 'bookmarks'}}" class="{{activePage 'bookmarks'}}">
      <span class="wrapper-menu-item">
        <span class="icon-bookmark-hollow"></span>
        <span class="title-menu">Закладки</span>
      </span>
    </a>
    <a href="{{pathFor 'about'}}" class="{{activePage 'about'}}">
      <span class="wrapper-menu-item">
        <span class="icon-question"></span>
        <span class="title-menu">О нас</span>
      </span>
    </a>
  </section>
```

Рисунок 16 - Структура «меню» приложения

Если нужно применить один и тот же стиль ко многим элементам можно указать класс для управлениявсеми этими элементами, как например было сделано для меню приложения (рисунок 17).

```
a {
  box-shadow: rgba(129,206,200,.1) 0 1px 0 0;
  color: rgba(255,255,255,.4);
  display: block;
  height: 20%;
  position: relative;
  text-align: center;

  &:hover,
  &:active,
  &.active {
    color: @color-empty;
  }

  .wrapper-menu-item {
    .font-s2;
    .type-bold;
    .position(absolute, 50%, auto, auto, 50%);
    .transform(translate3d(-50%, -50% ,0));
    display: block;
  }

  [class^="icon-"],
  [class*=" icon-"] {
    .font-m1;
    display: block;
    margin-bottom: 5%;
  }
```

Рисунок 17 - Правило для элементов «меню» приложения

Таким образом, стиль, определенный для тега «a», долженприменяться к содержимому данного элемента (и любых других элементов, относящихсяк этому элементу).

Меню сайта содержит такие пункты, как «Главная», «Что готовят?», «Рецепты»,«Закладки», «О компании».Кэлементу «header», в котором находится список меню применены правила CSS, которые позволяют зафиксировать его вверху страницы. Каждый пункт в меню соответствует названию отдельного раздела и перемещается от одного раздела к другому при помощи их идентификаторов (рисунок 18).

Рисунок 18–«Меню» сайта

2.3 Проектирование серверной частимобильного приложения

В зависимости от предметной области информационные системы могут различаться по своим функциям, архитектуре, реализации. Однако, они обладают некоторыми общими свойствами:

- Информационные системы предназначены для сбора, хранения и обработки информации. Поэтому в основе любой из них лежит среда хранения и доступа к данным;
- Информационные системы ориентируются на конечного пользователя, не обладающего высокой квалификацией в области применения вычислительной техники. Поэтому клиентские приложения информационной системы должны обладать простым, удобным, легко осваиваемым

интерфейсом, который предоставляет конечному пользователю все необходимые для работы функции, но в то же время не дает ему возможность выполнять какие-либо лишние действия.

Таким образом, исходя из вышеописанных свойств информационных систем, помимо программной составляющей, для проекта необходимо:

- Спроектировать базу данных, предназначенную для хранения информации;
- Разработать графический интерфейс пользователя.

Для раскрытия функций модуля была разработана диаграмма вариантов использования Use-case. Диаграмма Use-Case рассматривается как главное средство для первичного моделирования динамики системы, используется для выяснения требований заказчика к разрабатываемой системе, фиксации этих требований в форме, которая позволит проводить дальнейшую разработку.

Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования.

Актером или действующим лицом называется любая сущность, взаимодействующая с системой извне.

В результате проектирования приложения, была создана диаграмма вариантов использования (рисунок 19).

Варианты использования служат для описания сервисов, которые система предоставляет актеру. Варианты использования приложения(рисунок 20) прежде всего основываются на определенных видах действий.

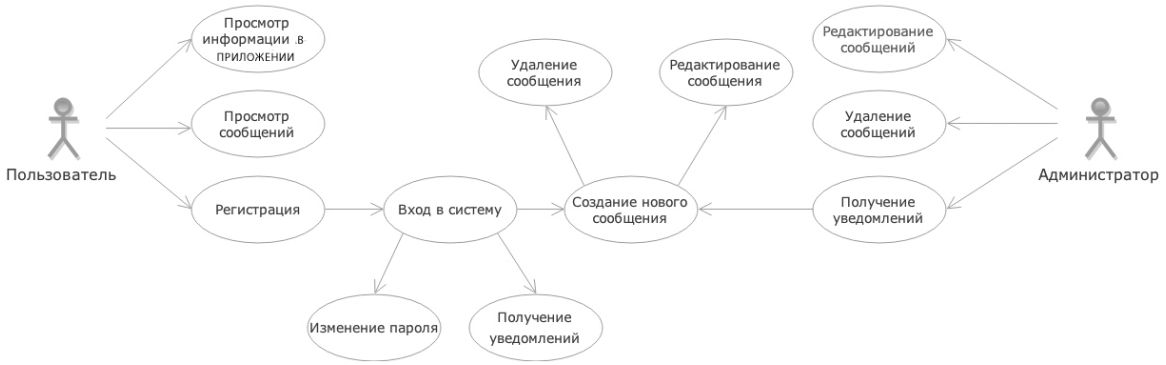


Рисунок 19–Use-case диаграмма использования

В системе присутствует два актера (пользователь и администратор системы). Для пользователя и администраторав системе доступны следующие виды действий.

Пользователь может:

- Просматривать информацию организации;
- Зарегистрироваться;
- Оставлять сообщение;
- Добавлять комментарии к сообщениям других пользователей;
- Редактировать свое сообщение;
- Изменять свой пароль;
- Получать уведомления.

Администратор может:

- Редактировать сообщения пользователей;
- Создавать сообщения;
- Удалять сообщения пользователей;
- Добавлять комментарии к сообщениям пользователей;
- Редактировать свое сообщение;
- Изменять свой пароль;
- Получать уведомления;
- Просмотреть список новых сообщений от пользователей.

Актор	Вариант использования	Формулировка
Пользователь	Просмотр информации	Этот вариант использования позволяет пользователю просматривать все разделы без регистрации
Пользователь	Просмотр сообщений	У пользователя есть возможность просматривать сообщения других пользователей без регистрации
Пользователь	Регистрация	У пользователя есть возможность зарегистрироваться, при этом, пользователь должен будет ввести свое имя и пароль
Пользователь	Вход в систему	Этот вариант использования предполагает авторизацию пользователя в системе
Пользователь	Изменение пароля	Пользователь имеет возможность изменить свой пароль
Пользователь	Получение уведомлений	Этот вариант использования предполагает, что пользователь будет получать уведомления о новых комментариях
Пользователь	Удаление сообщения	У пользователя есть возможность удалять свое сообщение
Пользователь	Создание нового сообщения	Этот вариант использования позволяет пользователю создавать новое сообщение
Пользователь	Редактирование сообщения	У пользователя есть возможность изменять свое сообщение
Администратор	Редактирование сообщений	У администратора есть возможность редактировать любое сообщение
Администратор	Удаление сообщений	Этот вариант использования позволяет администратору удалять сообщения пользователей
Администратор	Получение уведомлений	Этот вариант использования предполагает, что администратор будет получать уведомления о созданных пользователями сообщениях
Администратор	Создание нового сообщения	У администратора есть возможность создавать сообщения
Администратор	Редактирование сообщения	Этот вариант использования позволяет администратору изменять свои сообщения
Администратор	Удаление сообщения	Этот вариант использования позволяет администратору удалять свои сообщения

Рисунок 20 – Варианты использования

2.4Проектирование базы данных

Существую следующие этапы проектирования баз данных:

- Концептуальное проектирование — сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия: исследование предметной области, выявление фрагментов пользовательского представления (каждый из которых характеризуется, информационными объектами и связями между ними), моделирование и интеграция всех представлений. После этого получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели «сущность-связь».
- Логическое проектирование — преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

На этапе концептуального проектирования базы данных была создана модель сущность-связь, часто называемая также ER-моделью.

На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных.

Одним из основных понятий ER-диаграммы являются сущность.

Сущность — это реальный или виртуальный объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

В ходе концептуального проектирования была построена ER-модель базы данных (рисунок 21). В качестве сущностей модели выступают:

- Приложение;
- Модуль технической поддержки;
- Клиент;

- Пользователи;
- Администратор;
- Сообщения;
- Комментарии;
- Уведомления.

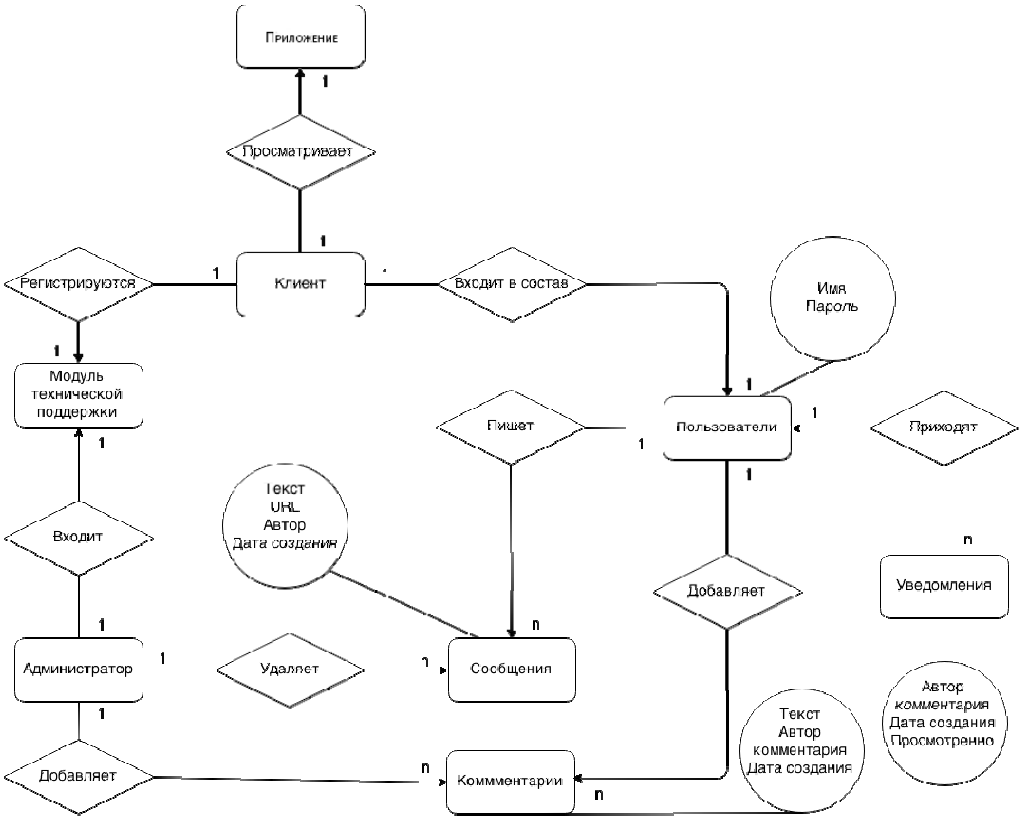


Рисунок 21 - ER- модель базы данных

Логическая структура базы данных должна быть спроектирована и описана для конкретной СУБД. При разработке модуля технической поддержки использовалась СУБД MongoDB.

Если рассматривать понятие базы данных в контексте MongoDB, то его можно определить, как логическое и физическое собрание коллекций.

В MongoDB не существует явного способа создать базу данных. База создается автоматически при записи в принадлежащую ей коллекцию.

Коллекции — это контейнеры структурно или концептуально схожих документов.

Модель устройства базы данных в MongoDB (рисунок 22). представляет собой набор «баз данных», которые состоят из «коллекций». «Коллекции» состоят из «документов». Каждый «документ» состоит из «полей». «Коллекции» могут быть проиндексированы, что улучшает производительность выборки и сортировки. Получение данных из MongoDB сводится к получению «указателя», который передает эти данные, если они нужны.

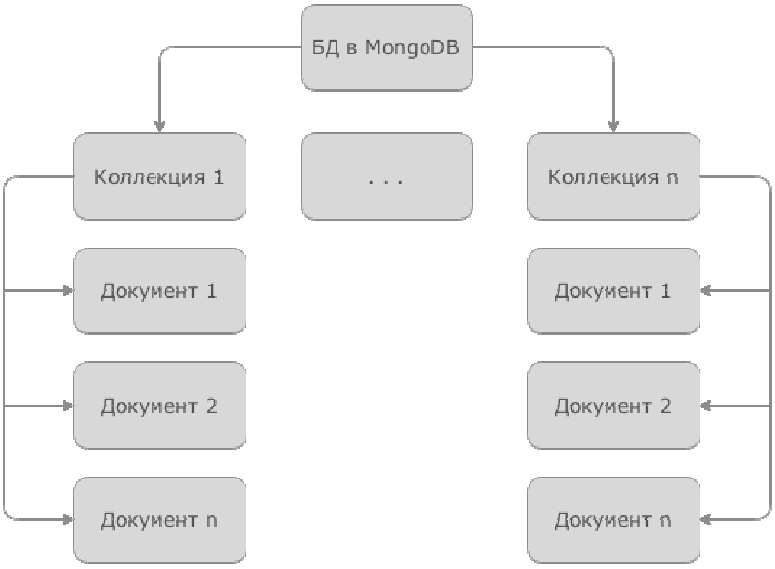


Рисунок 22 - Модель устройства базы данных в MongoDB

После анализа основных функций модуля технической поддержки, представленных в диаграмме вариантов использования, было определено какие коллекции будут использованы в базе данных.

В логической структуре базы данных (рисунок 23), все коллекции будут связаны между собой. Например, коллекция сообщений будет связана с коллекцией уведомлений, потому что, как только зарегистрированный пользователь напишет комментарий к сообщению другого пользователя, то этот пользователь получит уведомление. Так же коллекция сообщений связана с комментариями, так как к каждому сообщению будут добавляться

комментарии других пользователей. Коллекция сообщений связана с коллекцией пользователей системы, таким образом каждый, кто создает сообщение, считается его автором.

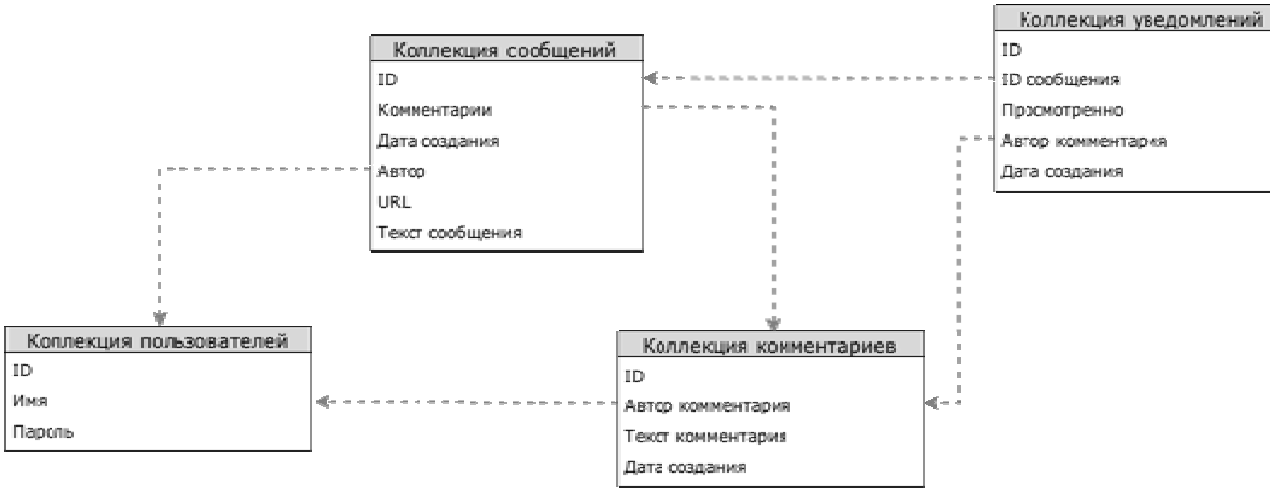


Рисунок 23 - Логическая структура базы данных

MongoDB является документо-ориентированной системой, в которой центральным понятием является документ. Документ можно представить, как объект, хранящий некоторую информацию. В некотором смысле он подобен строкам в реляционных СУБД, где строки хранят информацию об отдельном элементе.

Так как в системе будет механизм авторизации, базе данных нужна коллекция пользователей, атрибутами которой будут:

- ID;
- Имя;
- Пароль.

Пользователи системы будут обмениваться сообщениями, значит необходима коллекция сообщений, атрибутами которой будут:

- ID;
- Комментарии;
- Дата создания;
- Автор;

- URL;
- Текст сообщения.

Атрибуты коллекции уведомлений:

- ID;
- ID сообщения;
- Дата создания;
- Автор;
- Просмотрено.

Атрибуты коллекции комментариев:

- ID;
- Дата создания;
- Автор комментария;
- Текст комментария.

Физический уровень содержит всю информацию необходимую для реализации базы данных. MongoDBдокументо-ориентированная база данных, хранящая данные в JSONформате. Однако MongoDBявляется объектной базой, нежели чистым хранилищем типа ключ значение.

Физический уровень модели зависит от выбранного сервера. Множество поставщиков веб-сервисов предлагают многопользовательские хранилища типа ключ-значение. Большинство из них удовлетворяют критериям, перечисленным выше, однако каждое обладает своими отличительными качествами и отличается стандартами. Преимущество таких хранилищ данных перед реляционной СУБДдешевизна и легко интегрируется с мобильными сервисами.

2.5Вывод по разделу 2

В разделе рассмотрен процесс проектирования элементов информационной системы приложения. На первом этапе были определены основные требования к проекту. Следующим этапом стала разработка структуры приложения. Для того, чтобы определить спроектировать серверную часть была построена диаграмма вариантов использования. На этапе проектирования базы данных была создана ER-модель, определяющая ключевые сущности системы. На основе ER-модель было спроектировано представление данных на логическом уровне.

Раздел 3 Описание ИС

3.1Сценарий работы пользователя с системой

На главной страницеприложения (рисунок 24), пользователь сразу сможет просматривать последние новости, опубликованные фотографии других пользователям, просматривать блюда, загруженные администраторами, при этом он не сможет создавать свои собственные фотографии, сохранять в закладки понравившиеся рецепты, пока не зарегистрируется.

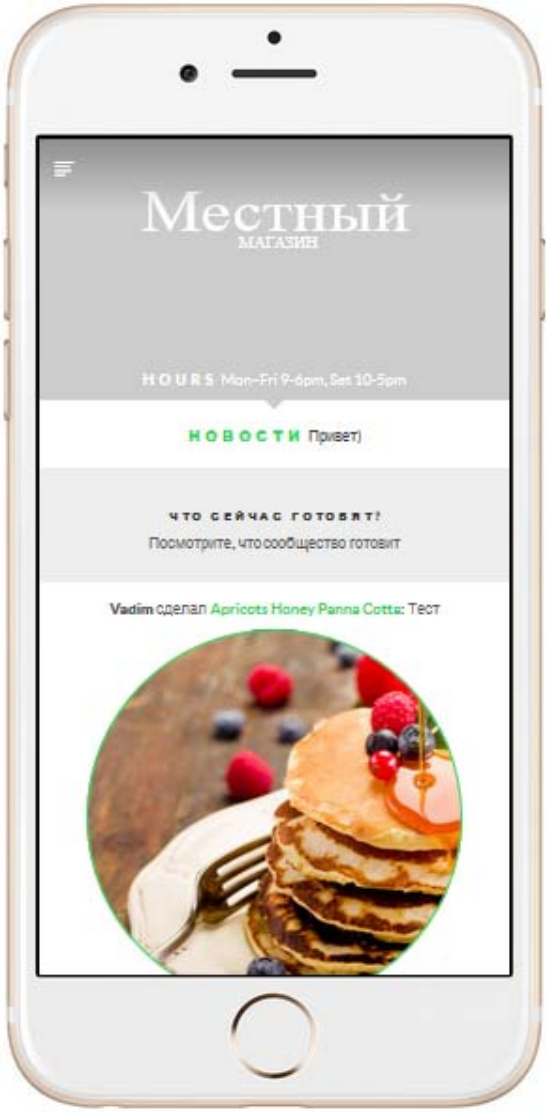


Рисунок 24–Главная страница

Чтобы оставить сообщение пользователю системы нужно зарегистрироваться или войти в систему. Если пользователь уже зарегистрирован, то он может войти в систему просто введя свое имя и пароль от социальной сети «Twitter» в форму авторизации (рисунок 25).

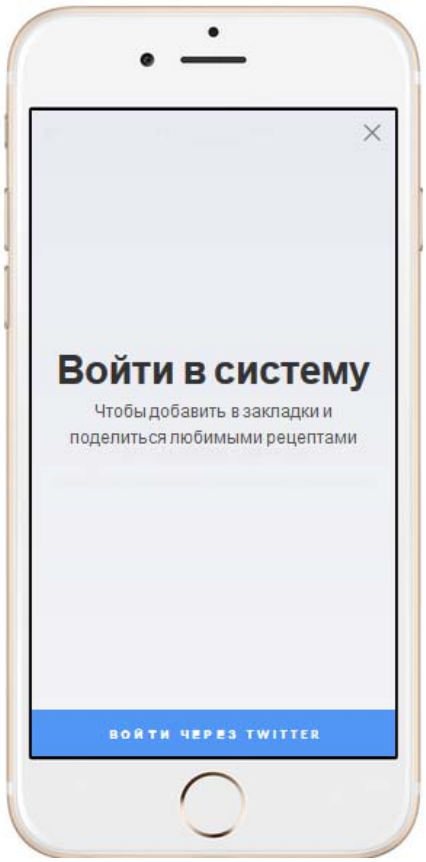


Рисунок 25 - Форма авторизации

Теперь, когда пользователь вошел в систему, он можетпросматривать рецепты рисунок (26). Выбрав один из рецептов пользователь может знакомится с ингредиентами и советами в готовке.

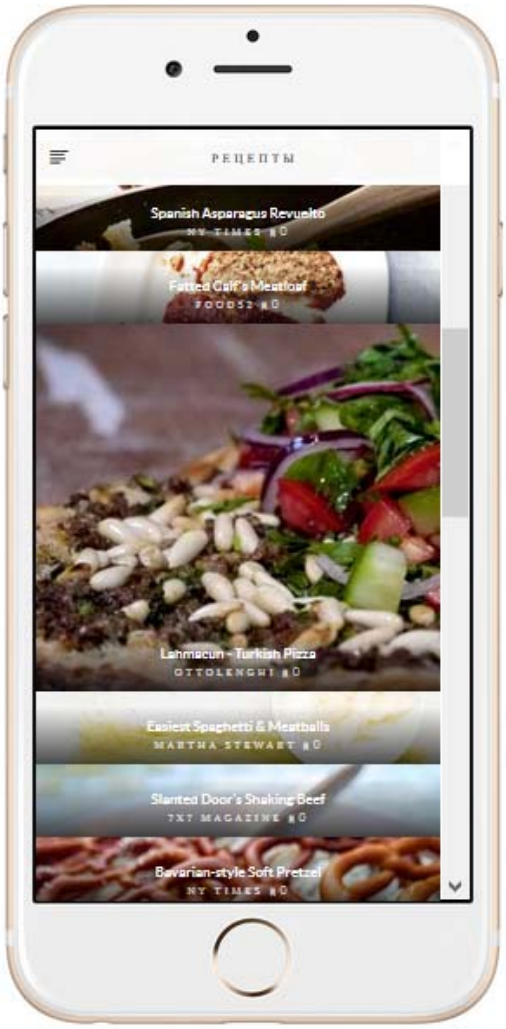


Рисунок 26–Просмотр список рецептов

Еще одной возможностью для пользователя системы, является добавление рецептов в закладки. Для этого нужно перейти на вкладку с рецептом, а затем нажать на кнопку «Добавить в закладки». Понравившийся рецепт попадет на страницу закладок, на этой странице хранятся все сохранённые рецепты пользователя (рисунок 27).

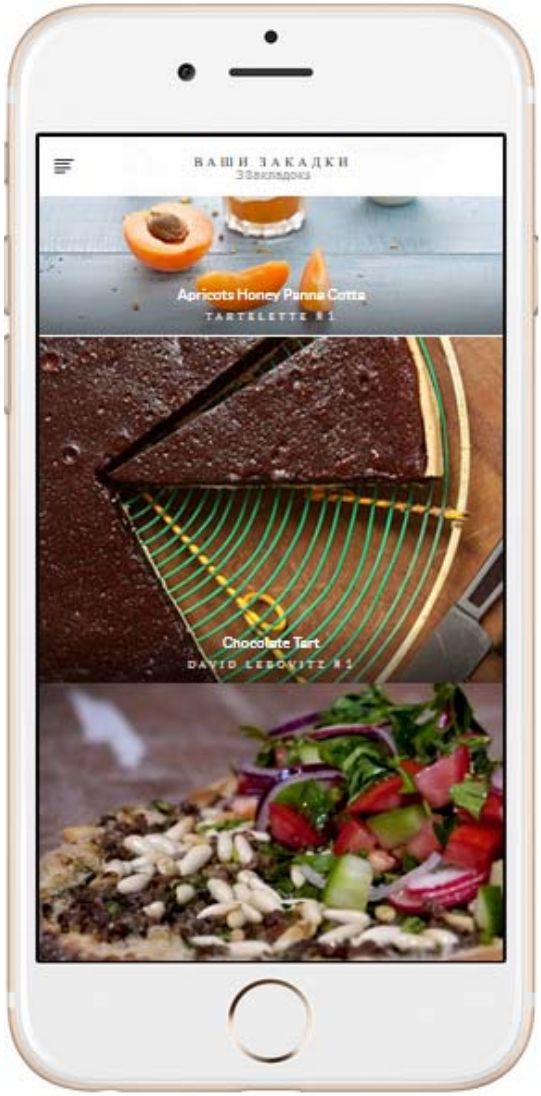


Рисунок 27–Страница закладок

После авторизации в системе, пользователи могут создавать изображения, с помощью встроенной камеры в мобильном устройстве. Сделанные фотографии можноопубликовать в социальных сетях. Все фотографии пользователей хранятся на странице «Что готовится»(рисунок 28).

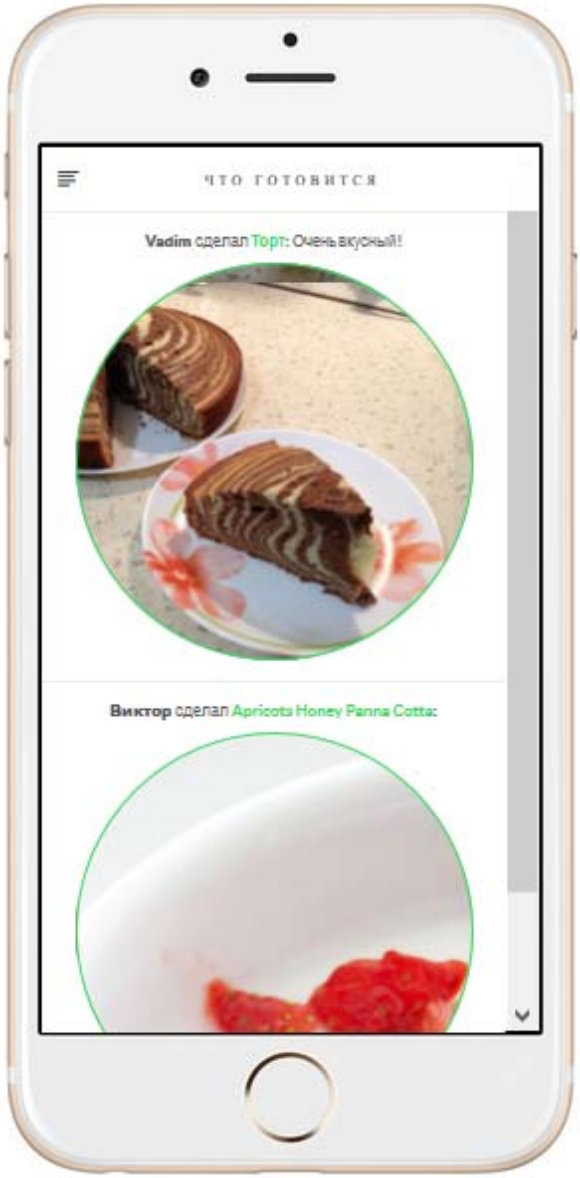


Рисунок 28 - Создание комментария

3.2Информационная безопасность ИС

Информационная безопасность системы обеспечивается за счет безопасности платформы Meteor. Безопасность веб-приложения, созданного с помощью Meteor, реализуется за счет установки и удаления специальных «Умных пакетов».

Умные пакеты — это функциональные модули, которые можно с легкостью добавлять или удалять с помощью командной строки. Умные пакеты

могут включать код, исполняемый на стороне сервера или клиента, интерфейсы пользователей и многое другое.

Например, пакет «insecure» разрешает серверу не проверять правила доступа перед считыванием или изменением данных. Этот пакет устанавливается по умолчанию, открывая доступ всем клиентам. Удаление этого пакета отменяет возможность изменения данных на сервере для клиентов.

В платформе Meteor имеются «Умные пакеты», упрощающие добавление возможности входа и авторизации пользователя в системе. В пакете «accounts» применяется комплексный подход; он включает интерфейс пользователя, серверную базу данных и необходимые клиент-серверные API.

Пакет «account-password» поддерживает создание пользователя и вход в систему с использованием запомненной комбинации имени пользователя и пароля. Реализация использует протокол защищенной передачи пароля и поэтому незашифрованные текстовые пароли между клиентом и сервером не передаются.

Таким образом в приложении присутствуют такие элементы безопасности, как:

- Система аутентификации;
- Код, допускающий изменение данных только со стороны персонала технической поддержки.

3.3Оценка качества ИС

С точки зрения ИСО/МЭК 9126[20], качество информационной системы можно определить, как совокупную характеристику программного средства с учётом следующих составляющих:

- Функциональные возможности — способность программного средства обеспечивать решение задач, удовлетворяющих сформулированные

потребности пользователей при применении комплекса программ в заданных условиях;

- Функциональная пригодность — набор и описания субхарактеристики и ее атрибутов, определяющие назначение, номенклатуру, основные, необходимые и достаточные функции программного средства;
- Правильность — способность программного средства обеспечивать правильные или приемлемые для пользователя результаты и внешние эффекты;
- Способность к взаимодействию — свойство программных средств и их компонентов взаимодействовать с одной или большим числом компонентов внутренней и внешней среды;
- Защищенность — способность компонентов программного средства защищать программы и информацию от любых негативных воздействий;
- Надежность работы информационных систем определяется надежностью функциональных компонентов, общего программного обеспечения, комплексов технических и инженерных средств;
- Сопровождаемость — приспособленность программного средства к модификации и изменению конфигурации и функций;
- Практичность — свойства программного средства, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях;
- Эффективность — свойства программного средства, обеспечивающие требуемую производительность решения функциональных задач, с учетом количества используемых вычислительных ресурсов в установленных условиях;
- Мобильность — подготовленность программного средства к переносу из одной аппаратно-операционной среды в другую.

После анализа критериев качества информационных систем, можно сказать, что для спроектированной системы мобильного приложения будут следующими:

- Функциональные возможности. Функциональные возможности удовлетворяют сформулированные потребности пользователей при применении информационной системы, но они малы. Таким образом, функциональные возможности средние.
- Функциональная пригодность. Информационная система имеет все необходимые функции, исходя из технического задания. Таким образом, функциональная пригодность высокая.
- Правильность. Программное средство обеспечивает правильные или приемлемые результаты и внешние эффекты. Таким образом, правильность высокая.
- Защищенность. Так как защиту системы можно регулировать с помощью средств фреймворкаMeteor, то, защищенность можно определить, как среднюю.
- Надежность. Проект использует оптимальное количество функций, что положительно сказывается на надежности. Таким образом, надёжность средняя.
- Сопровождаемость. Так как система разработана с использование фреймворкаMeteor, то ее можно легко модифицировать и увеличивать ее функционал и масштаб. Таким образом, сопровождаемость высокая.
- Практичность. Система имеет простой и понятный интерфейс. Благодаря этому, она будет удобна для пользователей. Таким образом, практичность высокая.
- Эффективность. Информационная система использует мало вычислительных ресурсов при выполнении своих задач и функций. Таким образом, эффективность высокая.

- Мобильность. Так как информационная система технической поддержки реализована с помощью средств веб-разработки, то проблем с ее возможным переносом не возникнет.

3.5 Вывод по разделу3

Вначале раздела был представлен сценарий работы пользователя с системой. Таким образом были продемонстрированы все возможности пользователя в системе. В разделе «Информационная безопасность ИС»были описаны пути обеспечения информационной безопасности с помощьюфреймворкаMeteor.Также в главе был проведен анализ качества информационной системы, у большинства критериев качества преобладают высокие показатели.

ЗАКЛЮЧЕНИЕ

Результатом выполненной работы является мобильное социальное приложение для продуктового магазина.

В данной работе представлен процесс проектирования системы. В начале проектирования системы были поставлены задачи, поэтапное решение которых привело к ее построению и реализации.

В ходе проектирования системы были определенысредствадля ее разработки. Анализ этих средств помог установитьпреимущества разработки и определить особенности подобныхсредств.

В результате выполнения следующего этапа проектирования системыбыли сформированы требованияк проекту. Формулировка требований и их утверждение привели к началу этапа построения системы. На этом этапе был создан интерфейс проекта, разработана база данных и построена диаграмма вариантов использования.

Следующим этапом стало описание системы. На этом этапе описывается сценарийдействий пользователя при взаимодействии с системой. Также были определены средстваобеспечения безопасности системы и проведен анализ ее надежности.

Спроектированная система позволит сократить время пользователю, и повысить взаимодействияорганизации с пользователями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 30.12.2013. – Красноярск: ИПК СФУ, 2013. – 60 с.

2. Штайнер, Г. HTML/XML/CSS: пер с англ. / Г. Штайнер – Москва : Лаборатория базовых знаний, 2005. – 512 с.

3. Угринович, Н.Д.Практикум по информатике и информационным технологиям / Н.Д. Угринович, Л.Л.Босова, Н.И.Михайлова.– Москва:Издательство "БИНОМ. Лаборатория знаний", 2002. – 400 с.

4. Ульман, Д. Основы реляционных баз данных / Д. Ульман, Д. Уид – : Москва : Лори, 2006. – 384 с.

5. Шелдон, Р. MySQL. Базовый курс для начинающих: пер с англ. / Р.Шелдон, Дж.Мойе – Москва : Издательский дом «Вильямс», 2007. – 880 с.

6. Шаши, Ш. Основы построения баз данных: пер с англ. / Ш. Шаши. – Москва : КУДИЦ-ОБРАЗ, 2004. – 336 с.

7. Нильсен, Я. Веб-дизайн: книга Якоба Нильсена: пер с англ. / Я. Нильсен. — Санкт-Петербург :Символ-Плюс, 2000. – 512 с.

8. Глушаков, С.В.Программирование Web-страниц / С.В.Глушаков, И.А.Жакин, Т.С.Хачиров.– Москва :ООО "Издательство АСТ"; Харьков: "Фолио", 2003. – 398 с.

9. Хомоненко, А. Д. Основы современных компьютерных технологий: учеб.пособие / ред.: А.Д. Хомоненко – Санкт-Петербург : КРОНА принт, 1998. – 496 с.

10. Хомоненко, А. Д. Базы данных: учебник для высших учебных заведений. – 4-е изд., доп. и перераб. / под ред. проф.: А.Д. Хомоненко – Санкт-Петербург : КОРОНА принт, 2004. – 736 с.

11. Румянцев, Д. Сам себе Web-программист. Практикум создания качественного Web-сайта: учебное пособие / Д. Румянцев. – Москва :ИНФРА-М, 2001. – 207 с.

12. Зандстра, М. PHP. Объекты, шаблоны и методики программирования: пер. с англ. / Зандстра, М. – Москва : Вильямс, 2015. – 576 с.

13. Контеров, Д. PHP 5 / Д.Котеров. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2016. – 1104 с.

14. Скляр, Д. PHP. Рецепты программирования: пер. с англ. / Скляр Д., Трахтенберг А. – Санкт-Петербург : Питер, 2015. – 784 с.

15. Колесниченко, Д. PHP и MySQL. Разработка веб-приложений / Д.Колесниченко. – 5-е изд. – Санкт-Петербург : БХВ-Петербург, 2015. – 592 с.

16. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS: пер. с англ. / Р.Никсон. – Санкт-Петербург : Питер, 2015. – 688 с.

17. Закас, Н. JavaScript. Оптимизация производительности: пер. с англ. / Н.Закас. – Москва : Символ-Плюс, 2012. – 256 с.

18. Дронов, В. Macromedia Dreamweaver MX. Наиболее полное руководство:учебное пособие / В. Дронов. – Санкт-Петербург :БХВ-Петербург, 2003. – 728 с.

19. Флэнаган, Д. JavaScript. Подробное руководство: пер. с англ. / Д.Флэнаган. – Москва : Символ-Плюс, 2009. – 992 с.

20. Белунцов, В. Новейший самоучитель по разработке Web-страниц:учебное пособие / В.Белунцов. – Москва :Издательство "ДЕСС КОМ", 2000. – 448 с.