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

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Система хранения данных на основе Raspberry pi
тема

Руководитель	_____	доцент, канд. техн. наук	О. А. Русанова
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		И. В. Якимов
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____		В. И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
« ___» _____ 2018 г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студенту Якимову Игорю Владимировичу
фамилия, имя, отчество

Группа КИ14-08Б Направление (специальность) 09.03.01
номер код

«Информатика и вычислительная техника»
наименование

Тема выпускной квалификационной работы Система хранения данных на основе Raspberry pi

Утверждена приказом по университету № _____ от _____

Руководитель ВКР О. А. Русанова, доцент кафедры В.Т.
инициалы, фамилия, должность, учебное звание и место работы

Исходные данные для ВКР: задание на бакалаврскую работу

Перечень разделов для ВКР: 1 Анализ предметной области; 2 Проектирование системы; 3 Описание работы приложения

Перечень графического материала: презентация доклада выступления; видео

Руководитель ВКР _____ О. А. Русанова
подпись инициалы и фамилия

Задание принял к исполнению _____ И. В. Якимов
подпись инициалы и фамилия

«____» _____ 2018 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С появлением первых компьютерных сетей появилась потребность в особых устройствах, которые обслуживали бы доступ к ресурсам данной сети. Данными электронными устройствами стали сервера, которые с течением времени представляли собой большие ЭВМ (мейнфреймы), затем персональные компьютеры, а впоследствии и отдельные вычислительные системы. Первые серверы, которые мы знаем в их нынешнем состоянии, появились в конце 80-ых годов прошлого столетия и так и назывались – файловые серверы.

На сегодняшний день существует множество сервисов хранения данных. Особенно популярной стала «облачная» модель хранения, при которой компьютер клиента является лишь терминалом, а все данные и расчеты происходят на других серверах.

Однако для некоторых пользователей хранение данных на сторонних серверах является недопустимым ввиду следующих причин:

1. Отсутствие контроля конфиденциальности в отношении данных в «облаке»;
2. Все данные физически находится далеко от пользователя.

В таком случае решением становится внедрение и обслуживание своего собственного хранилища. Реализация такой системы хранения данных – **цель данной работы.**

Для достижения данной цели необходимо решить следующие **задачи:**

1. разработать сервер;
2. разработать веб-приложение клиента;
3. разработать мобильное приложение клиента.

1 Анализ задания на ВКР

1.1 Требования к системам хранения данных

Система хранения данных служит для получения гарантированного доступа к файлам, которые находятся на сервере, множеству авторизированных пользователей, которое находится на конечном аппаратном сервере. Основными требованием таких систем являются:

1. Легковесность: такие системы не должны иметь сложный функционал, и осуществлять сложные вычислительные процессы.
2. Кроссплатформенность: доступ к системе через мобильные устройства и персональные компьютеры.
3. Наличие системы авторизации и управления пользователями;
4. Возможность работать с внешними устройствами (флеш накопителями или внешними жесткими дисками).
5. Предоставление на каждого пользователя выделенного пространства для хранения данных.
6. Защита доступа к файлам пользователей.

1.2 Обзор существующих аналогов

1.2.1 Сервер на базе FTP протокола

FTP – протокол передачи данных, разработанный на архитектуре «клиент-сервер», позволяет передавать данные любого объема по сети. Также этот протокол имеет реализацию расширения FTPS, добавляющую поддержку криптографических протоколов транспортной безопасности и защищенных сокетов [5]. В данное время существует множество реализаций серверов и клиентов на данном протоколе. Интерфейс сервера FileZilla представлен на рисунке 1:

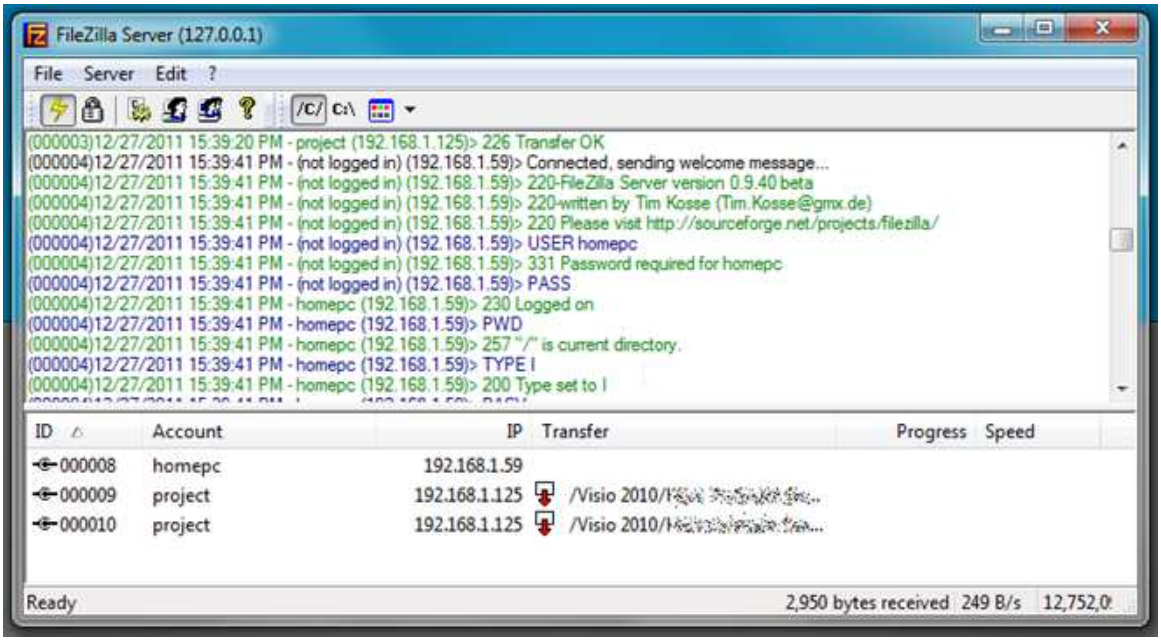

Рисунок 1 – Интерфейс ftp сервера – FileZilla Server

Для решения данной задачи наиболее правильным выбором было бы использовать сервер, реализованный на протоколе ftp, однако для устройств разных операционных систем потребовалось бы дополнительное программное обеспечение в виде FTP клиент-приложения. Также, протокол ftp медленнее работает при работе с несколькими клиентами, чем с одним, так как ftp сервер работает в один поток и данный поток обслуживает множество клиентов, в связи с чем, при большом количестве пользователей может возникнуть регрессии в скорости работы/загрузки. Также, протокол FTP не поддерживает функции сжатия данных и отдаёт файлы «как есть».

1.2.2 Файловый сервер на nodejs

Nodejs – программная платформа, разработанная на движке V8, созданная компанией Google [6]. Сервер, работающий на данной платформе, использует протокол http. Данная платформа идеально подходит для разработки http серверов запросов-ответов и микросервисов для кроссплатформенных клиентов

и веб-приложений архитектуры клиент-сервер [8]. Но для предоставления доступа к файлам сервера на данной платформе имеют свои недостатки.

Например, данная платформа работает в один поток и асинхронно обслуживает несколько клиентов. Таким образом, при достаточно большом обилии клиентов, скорость отправки данных уменьшается. Также, скорость работы серверов на nodejs низкая за счет того, что платформа высокоуровневая и выполняется в виртуальной машине. Помимо всего, реализаций серверов хранения данных на данной платформе невелико.

1.2.3 Файловый хостинг на примере dropfiles.com

Множество файловых хостингов предоставляют пользователям данные через веб-серверы, путём генерации уникальных ссылок на них. В качестве примера рассмотрим сервис dropfiles.com, интерфейс которого представлен на рисунке2:

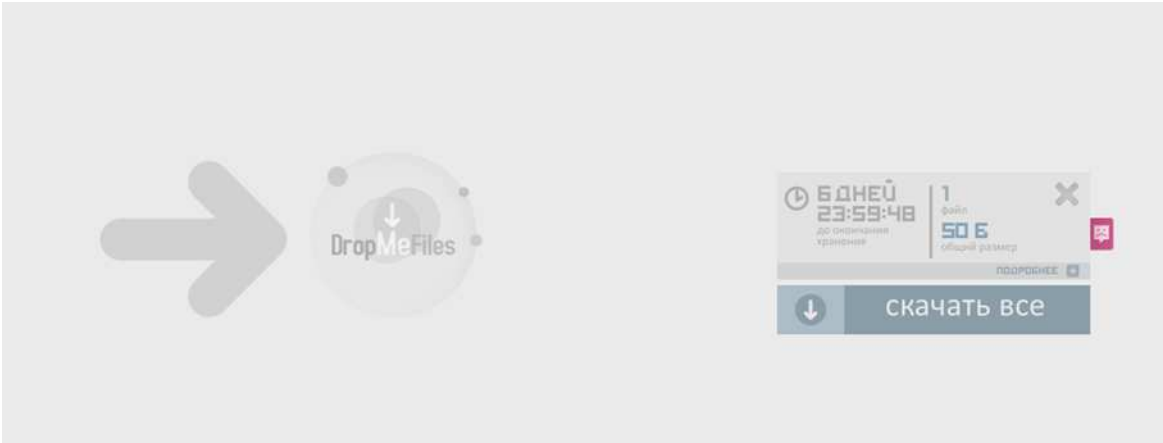

Рисунок 2 – Интерфейс сервиса dropfiles.com

Данный сервис позволяет генерировать ссылки для конкретных данных, загруженных на сервер хостинга [7]. Можно загружать несколько файлов сразу, однако данные хранятся ограниченное время. Также существует ограничение на максимальный размер загружаемых файлов.

1.3 Анализ технического задания, динамической модели системы

Для определения основных сущностей, а также их обязанностей, составлена диаграмма контекста системы, представленная на рисунке 3.

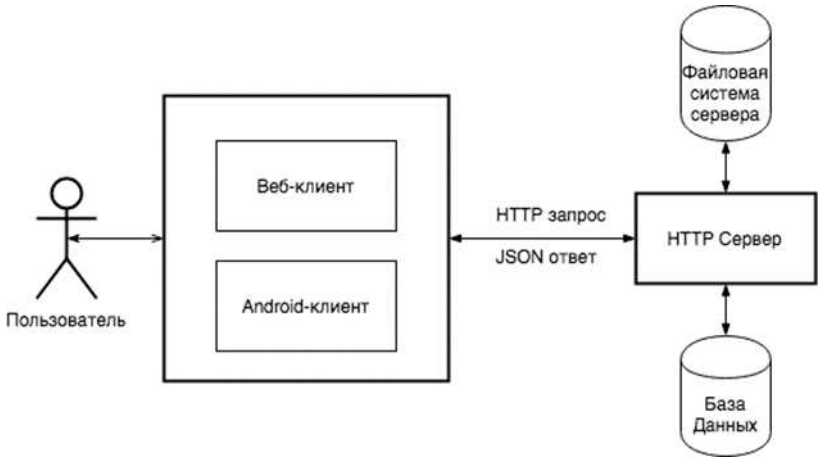

Рисунок 3 – Диаграмма контекста системы

Пользователю предоставляется возможность работы с одной из двух внешних систем на выбор: веб-клиентом или мобильным клиентом, разработанным для ОС Android. В качестве внутренней системы выступает сервер, разработанный на языке программирования go [1]. Связь между клиентом и сервером осуществляется по протоколу HTTP. Сам сервер взаимодействует с двумя хранилищами: базой данных, реализованной на языке SQL, и файловой системой сервера.

Детальные функциональные требования к работе системы сформулированы в виде диаграммы прецедентов пользователей, представленной на рисунке 4.

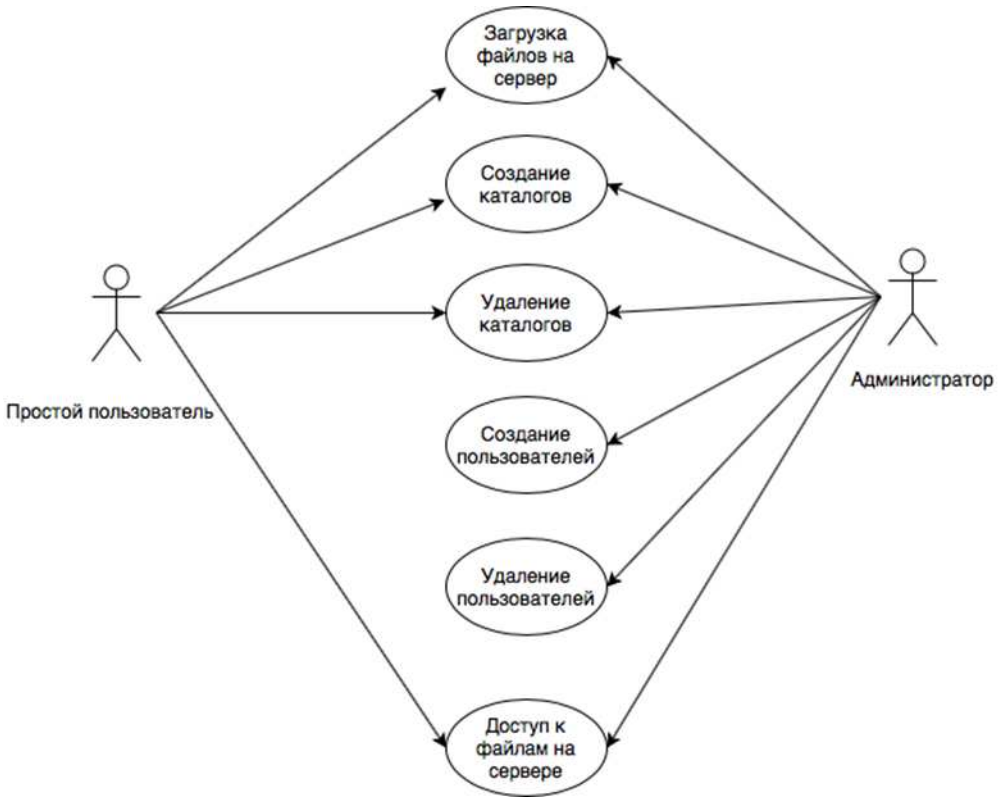

Рисунок 4 – Диаграмма прецедентов

Непривилегированному пользователю доступны следующие действия:

1. загружать файлы на сервер;
2. создавать каталоги;
3. удалять каталоги;
4. скачивать / иметь доступ к файлам на сервере.

Администратор помимо всего этого может создавать и удалять пользователей.

Согласно диаграмме прецедентов, клиентское приложение должно содержать следующие окна:

1. окно авторизации;
2. окно списка файлов и каталогов;
3. окно управления пользователями системы;
4. окно добавления файла;
5. окно добавления каталога.

Диаграмма, отражающая возможные переходы между ними приведена на рисунке 5:

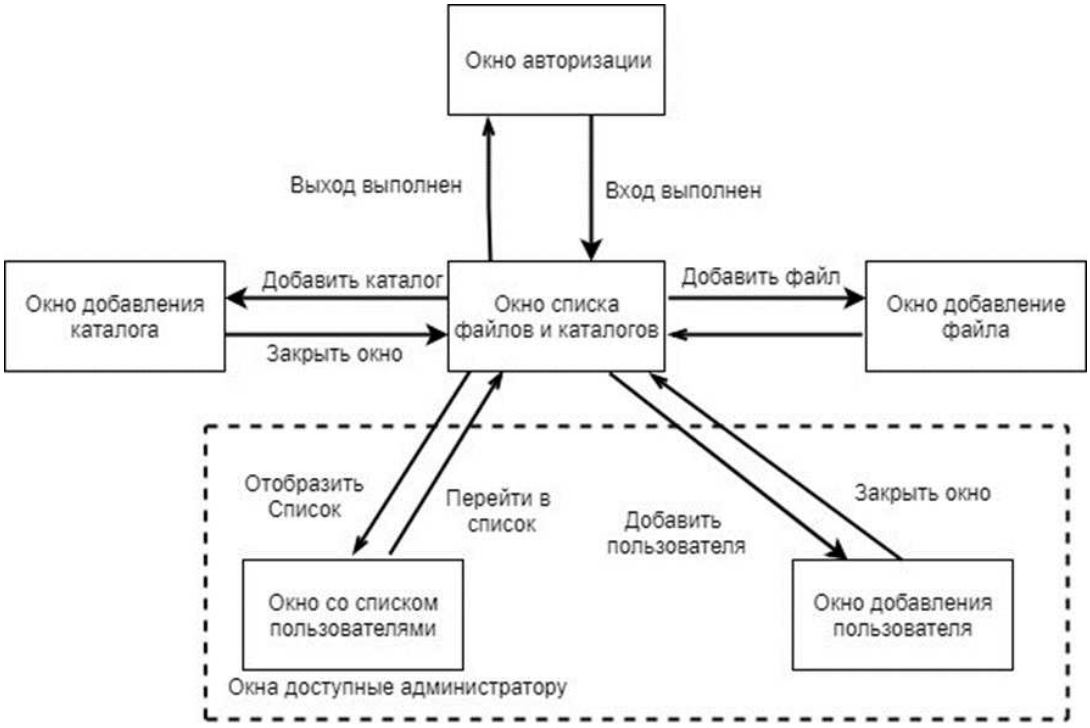

Рисунок 5 – Диаграмма потока экранов

Для каждого из элементов интерфейса разработаны условные макеты, которые представлены на рисунках 5-7:

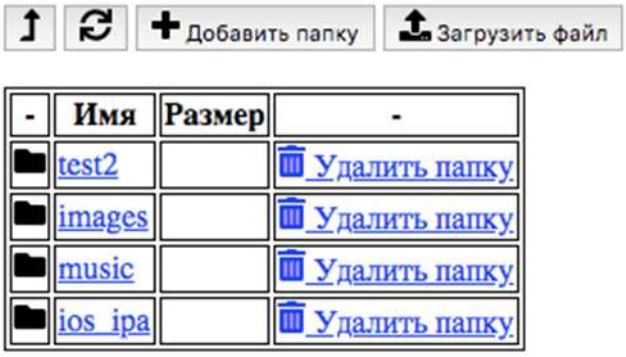

Рисунок 5 – Элемент интерфейса – список файлов и каталогов

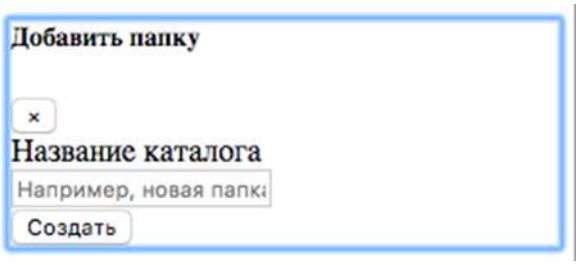

Рисунок 6 – Элемент интерфейса – окно добавления каталога

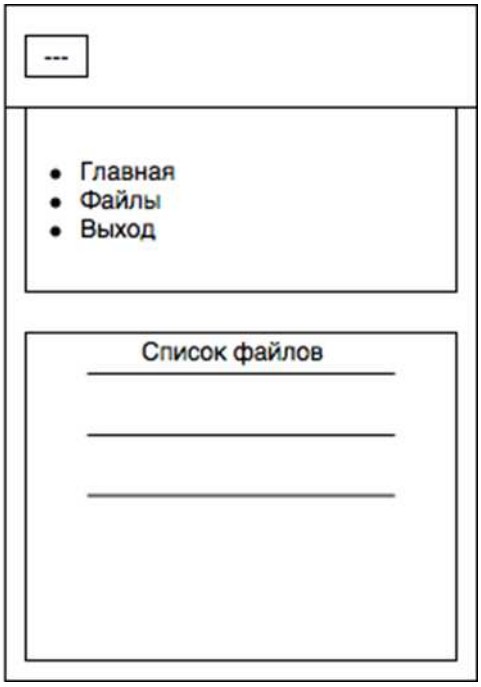

Рисунок 7 – Макет интерфейса мобильной версии – окно со списком файлов

1.4 Выводы по главе

1. Сформированы требования к разрабатываемой системе.
2. Проанализировано техническое задание и выполнен обзор существующих аналогов.
3. Определены основные сущности системы, а также их обязанности.
4. Сформулированы детальные функциональные требования к работе системы в виде диаграммы прецедентов пользователей.
5. Разработан макет интерфейса системы.

2 Проектирование системы

2.1 Структурная схема системы хранения

На основе анализа технического задания была разработана следующая структурная схема работы системы:

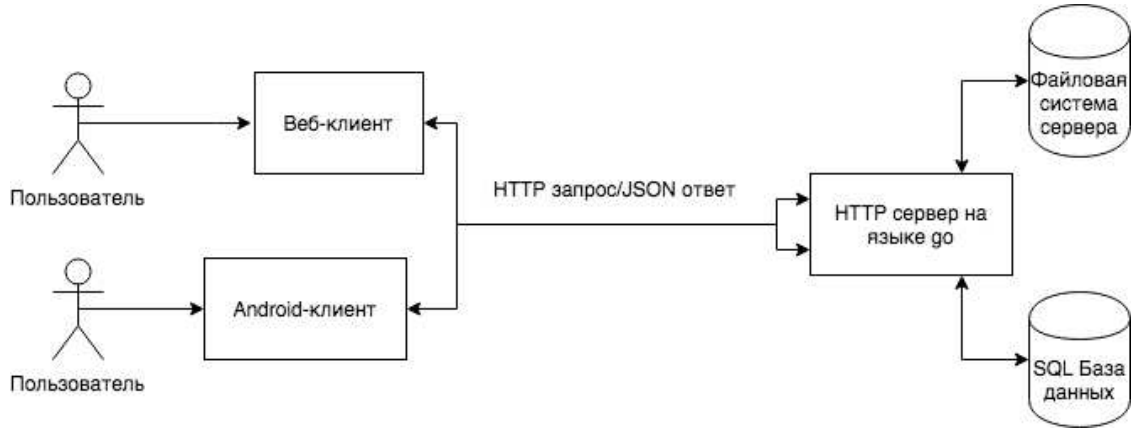

Рисунок 8 – Структурная схема системы хранения файлов

Система состоит из файловой системы сервера, базы данных, 3 программных частей, включающих в себя:

1. веб-клиент;
2. клиент для операционной системы Android;
3. сервер, общающийся с клиентами и работающий с файловой системой сервера.

База данных используется для хранения информации о пользователях и файлах, находящихся на сервере.

Одной из причин хранения информации о файлах стала защита пользовательских данных от несанкционированного доступа по url адресу. Исходя из этого, таблица базы данных, в которой хранится информация о файлах, должна также хранить путь до файла, а файл при загрузке, в свою очередь, должен иметь уникальное наименование для усложнения подбора названия файлов.

База данных содержит данные о местонахождении файлов на сервере и имеет строгую структуру, а также должна обеспечивать высокую скорость доступа к данным в ней. Поэтому выбрана SQL-подобная база данных postgresql [11].

Структура базы данных и связи между таблицами выглядят следующим образом, изображенном на рисунке 9:

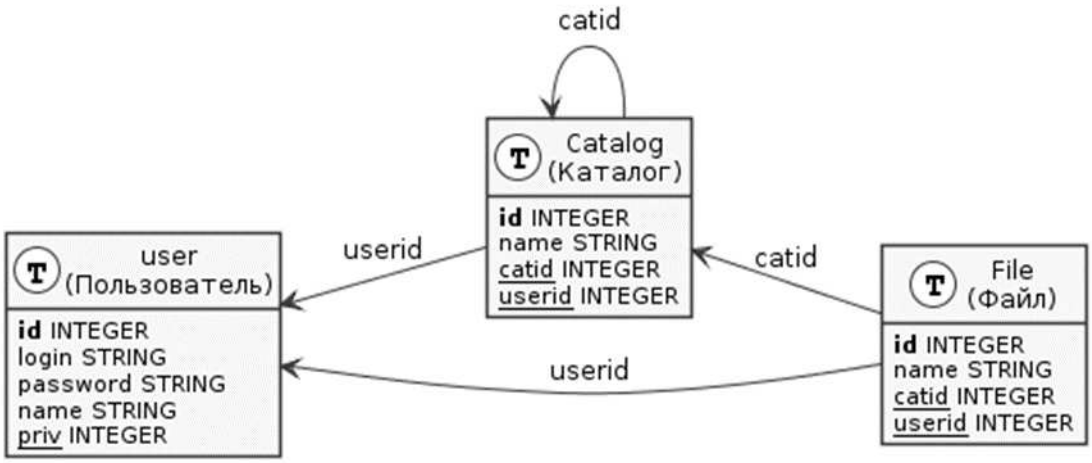

Рисунок 9 – Схема базы данных и связей между таблицами

2.2 Технология разработки системы

Серверная часть будет представлять из себя http сервер, с которым будет взаимодействовать клиентские веб и Android приложения. Для взаимодействия приложения будут отправлять серверу http запросы методами POST, GET, DELETE или UPDATE, хранить данные об авторизации в браузере или на смартфоне клиента, а также сервер будет предоставлять доступ к файлам, находящимся на нём.

2.2.1 Серверное приложение

Для разработки серверной части был выбран язык go, разработанный компанией Google. Данный язык используется, в основном, при разработке

простых мелких веб-сервисов. Для работы с базой данных будет использоваться ORM библиотека gorm [12].

С помощью неё можно проводить миграции в базу данных на основе разработанных моделей таблиц, которые описаны на языке go.

Для работы с бизнес-логикой сервера будет использоваться фреймворк gin-gonic [3]. Данный фремйворк позволяет реализовать логику запросов к серверу через http методы GET/POST/DELETE/UPDATE.

Работа с сервером будет осуществляться через два клиентских приложения.

2.2.2 Веб-приложение

Одним из простых и верных решений при выборе инструментария для веб-приложения является MVVM фреймворк Vuejs [2]. Vuejs позволяет разрабатывать одностраничные приложения (single page application) на языке javascript.

За верстку в приложении будет отвечать фреймворк bootstrap [10]. Он позволяет создавать адаптивные веб-страницы, кросс-браузерные и стандартизированные интерфейсы. Данный фреймворк отлично подходит для разработки под различные размеры экранов, что позволит сделать веб-приложение более адаптированным, как под мобильные, так и под настольные операционные системы.

2.2.3 Android приложение

Для разработки клиентского приложения для Android был выбран язык java. Реализация велась в интегрированной среде разработки Android Studio.

Для выполнения запросов к серверу будет использоваться библиотека android-async-http-client [14]. Она позволяет осуществлять асинхронные запросы

к серверу с помощью уже готовых классов в ней, созданных на основе сокетов tcp/ip.

Отладка приложения будет происходить на устройстве под управлением операционной системы Android версии 5.1, с 8 ядерным процессором на частоте 1.4 ГГц, и с 2 ГБ ОЗУ.

2.3 Алгоритм работы системы

Первоначально происходит авторизация пользователя в системе согласно приведенной далее схеме:

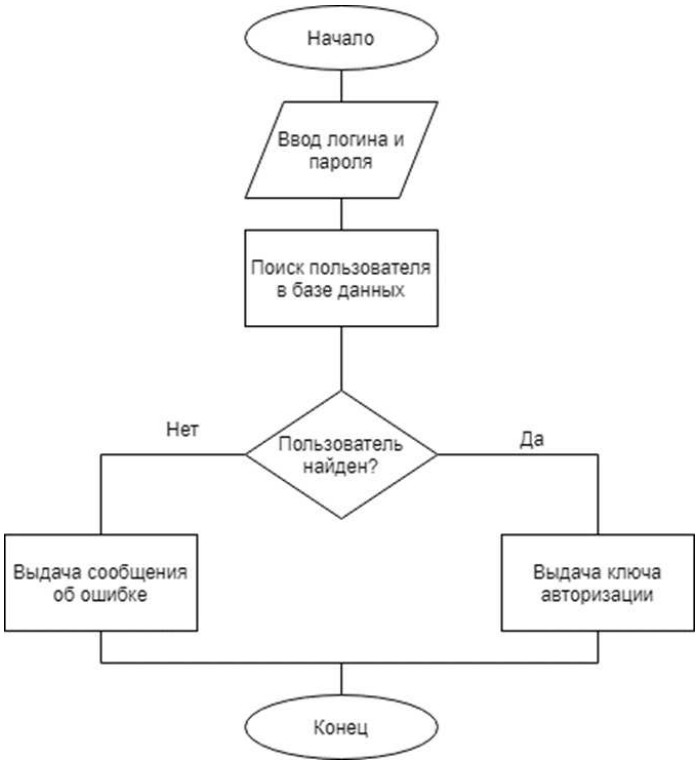

Рисунок 11 – Схема работы авторизации пользователя

При отправке запросов авторизованным пользователем серверное приложение обрабатывает запросы по следующей функциональной схеме:

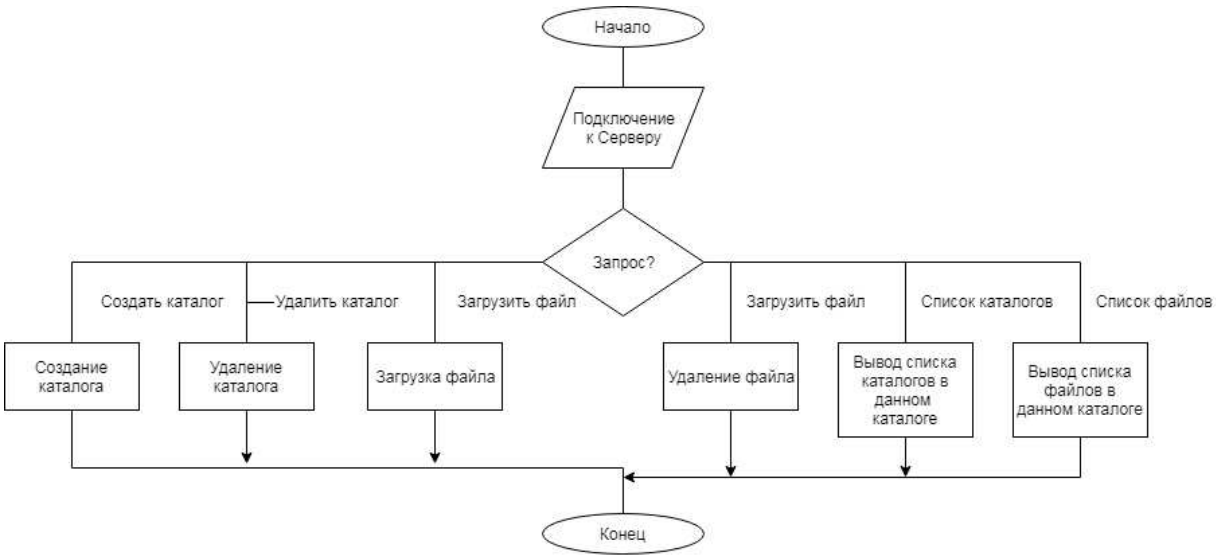

Рисунок 12 – Схема работы серверного приложения для обычного пользователя

Пользователи системы хранятся в отдельной таблице базы данных. За уровень привилегий в таблице отвечает отдельное поле. Привилегий у пользователей два: администратор и простой пользователь.

В отличие от простого пользователя, администратор системы также может создавать, удалять любых пользователей.

2.4 Аппаратный сервер и операционная система

В качестве сервера выступает raspberry pi model b [9]. На данном компьютере установлена операционная система Archlinux Arm с установленным программным обеспечением в виде компилятора языка go и веб-сервера nginx.

2.5 Выводы по главе

На основе требований системы, описанных в первой главе, была разработана архитектура системы. Выбраны технологии для реализации клиентских и серверного приложений.

3 Описание работы приложений

3.1 Описание работы серверного приложения

Серверное приложение представляет из себя http сервер, написанный на языке программирования go [3]. Данный сервер принимает http запросы от клиента и отправляет ответ в формате JSON. Он разработан на архитектуре MVC, которая позволяет разрабатывать серверные приложения, разделяя их на отдельные логические блоки: model, view, controller.

Блок model отвечает за работу с базой данных, в ней происходит работа с orm библиотекой gorm [11]. В нём описана схема структуры базы данных, которая создается при запуске сервера

Пример описания таблицы users в базе данных на языке go в файле models.go:

```
user struct {
        Id       int
        Login    string `json:"login"`
        Password string `json:"password"`
        Name     string `json:"Name"`
        Priv     int    `json:"Priv"`
    }
```

Блок view отвечает за отрисовку интерфейса, но так как у нас используются клиентские приложения, он отсутствует.

Блок controller отвечает за логику серверного приложения. В данном блоке описаны все ответы сервера на всевозможные запрос клиента, исходя из схемы работы серверного приложения.

Для удаления директорий на сервере используется рекурсивная функция deleteCatalogs, которая стирает файлы в категориях и вызывает саму себя для вложенных каталогов. Её реализация выглядит следующим образом:

```
func deleteCatalogs(id int, userid int) {
```

```
func deleteCatalogs(id int, userid int) {
deleteFilesInCatalog(id, userid)

var catalogs []qcatalog

db.Where("catid = ? and userid = ?", id, userid).Find(&catalogs)

for _, iter := range catalogs {

        deleteCatalogs(iter.Id, userid)

}

var tmpcatalog qcatalog

db.Find(&tmpcatalog, id)

db.Delete(&tmpcatalog)

}
```

Для того, чтобы сервер проверял, какой из пользователей в данный момент хочет получить ответ, используется технология javascript web token [13]. При каждом запросе на сервер в заголовок http заносится хеш-ключ, сгенерированный при авторизации пользователя. Хеш-ключ проверяется сервером, с помощью него сервер производит расшифровку данных сессии пользователя и проверяет идентификатор вошедшего пользователя. Полученный хеш-ключ необходимо хранить в приложении в случае успешной авторизации.

После загрузки файлов перед добавлением информации о файле в базу данных генерируется уникальное имя с помощью склеивания текущего имени и случайной хеш-строки.

Хеш-строка генерируется на основе текущего времени. Каждый символ берётся из массива, содержащего латинские буквы и цифры. Позиция элемента, который будет использоваться в итоговом хеше, находится как остаток от деления текущего времени, переведенного в секунды, на длину массива.

3.2 Описание работы веб-клиента

Веб клиент реализован на языке javascript на фрейворке vuejs [2]. Данный фреймворк позволяет создавать приложения путём разработки атомарных компонентов на языке javascript.

Клиентское приложение состоит из нескольких экранов, которые описаны в отдельных файлах компонентов формата vue. Принцип разработки клиентского приложения подразумевает создание вложенных компонент.

Таким образом, самый главный компонент-маршрутизатор, в котором происходит переключение экрана, называется router.vue. В нём описаны переходы между главными экранами приложения.

Пример описания маршрутизатора для vuejs на javascript:

```
export default new Router({
 routes: [
  {
   path: '/',
   name: 'Authorization',
   component: Auth
  },
  {
   path: '/Main',
   name: 'Main',
   component: Main
  }]
```

Соответственно, отдельно описаны компоненты авторизации, главного окна, окна добавления директории и загрузки файла. В совокупности, внешний вид веб-клиента показан на рисунке 13:

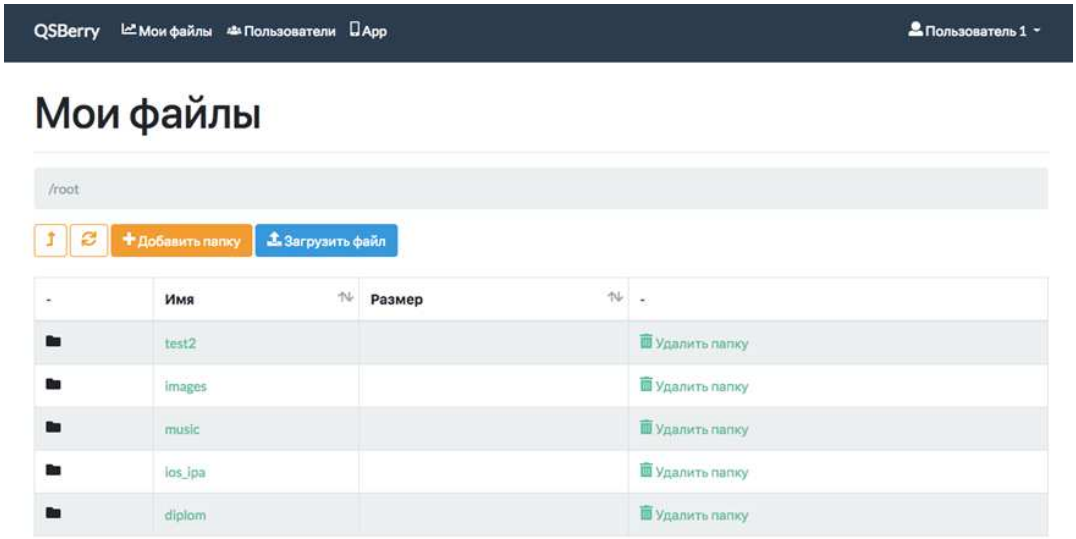

Рисунок 13 – Вид главного экрана веб-клиента

Для того, чтобы интерфейс приложения отображался корректно на устройствах с разным разрешением экрана, в том числе и мобильных, используется фреймворк bootstrap [10]. Он позволяет адаптировать элементы под мелкие разрешения экрана. На рисунке 14 изображено главное окно при мелком разрешении экрана:

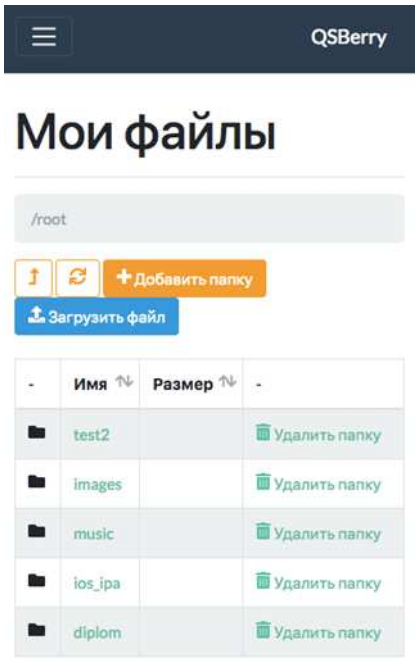

Рисунок 14 – Вид главного экрана при ширине экрана 320px

Для запросов на сервер, приложение использует библиотеку axios. Данная библиотека позволяет отправлять асинхронные запросы на сервер, получать ответ и преобразовывать сразу его в объекты на языке javascript для того, чтобы потом выводить эти данные на интерфейс клиентского приложения.

Хеш-ключ, полученный при авторизации пользователя, хранится в localstorage браузера пользователя.

3.3 Описание работы клиентского приложения на Android

Android приложение состоит из нескольких экранов, верстка которых описывается в файлах формата xml [4].

Рассмотрим файл res/layout/activity_login.xml. В данном файле приведена верстка окна авторизации в приложении. Для работы с элементами интерфейса им задается уникальный идентификатор id. Для доступа к управлению данным элементам и описывания их логики используется java файл src/main/java/loginactivity.java.

Пример описания поля для ввода пароля из файла activity_login.xml:

```
<EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:singleLine="true"
/>
```

Пример кода для получения доступа управления к элементам интерфейса из файла loginactivity.java:

final Button loginbutton = (Button) findViewById(R.id.sign_button);

final AutoCompleteTextView login = (AutoCompleteTextView) findViewById(R.id.login);

final EditText password = (EditText) findViewById(R.id.password);

Для обработки списка файлов в приложении используется отдельный класс FileAdapter, наследуемый от Adapter. Класс Adapter описывает структуры, выводимые в списках, и проводит работу над отдельно выведенным элементом списка.

На следующем изображении можно увидеть, как выглядит отрисованный список файлов в клиентском приложении для мобильных устройств.

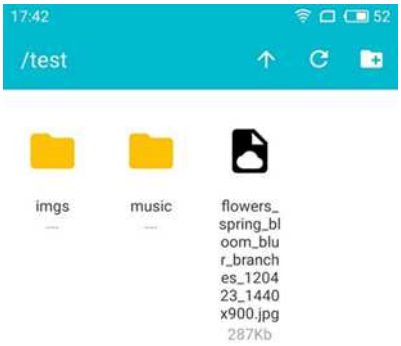

Рисунок 15 – Вид списка файлов клиентского приложения для операционной системы Android

Данные об адресе сервера, хеш-ключе, а также настройке клиента хранятся в статических полях класса Preferences.

Все функции для отправки запросов на сервер также вызываются через статически настроенный клиент, описанный в файле preferences.java. Для внешнего ускорения интерфейса запросы к серверу выполняются асинхронно.

Ответы с сервера в формате JSON преобразуются через стандартные библиотеки и средства языка java.

3.4 Выводы по главе

В соответствии с разработанной архитектурой и моделями базы данных, была разработана система, состоящая из серверного приложения, клиентского веб-приложения и клиентского приложения для системы Android.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана система хранения данных на основе одноплатного компьютера raspberry pi.

На основе требований и макетов были разработаны клиент-серверные приложения, объединенные в систему, позволяющую хранить данные на конечном сервере, основанном на raspberry pi под руководством операционной системы Linux.

СПИСОК СОКРАЩЕНИЙ

ВТ – Вычислительная техника

ИКИТ – Институт космических и информационных технологий

СФУ – Сибирский федеральный университет

MVVM – Model View View-Model

SQL – Structured query language

FTP – File transfer protocol

HTTP – Hyper text transfer protocol

ЭВМ – Электронно-вычислительная машина

ORM – Object-Relational Mapping

URL – Uniform Resource Locator

JSON – Javascript object notation

MVC – Model View Controller

XML – eXtensible Markup Language

ОС – Операционная система

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт языка Go [Электронный ресурс] – Режим доступа: https://golang.org
2. Сайт фреймворка Vuejs [Электронный ресурс] – Режим доступа: https://vuejs.org/
3. Официальный репозиторий фреймворка Gin на github [Электронный ресурс] – Режим доступа: https://github.com/gin-gonic/gin
4. Android Developers [Электронный ресурс] – Режим доступа: https://developers.google.com/android/
5. Что такое FTP [Электронный ресурс] – Режим доступа: https://semantica.in/blog/ftp.html
6. Официальный сайт nodejs [Электронный ресурс] – Режим доступа: http://nodejs.org
7. Файлообменник dropfiles [Электронный ресурс] – Режим доступа: http://dropmefiles.com
8. Express JS [Электронный ресурс] – Режим доступа: http://expressjs.com
9. Официальный сайт raspberry pi [Электронный ресурс] – Режим доступа: https://www.raspberrypi.org/
10. BootstrapVue [Электронный ресурс] – Режим доступа: https://bootstrap-vue.js.org
11. Postgresql [Электронный ресурс] – Режим доступа: https://postgresql.org
12. Библиотека gorm [Электронный ресурс] – Режим доступа: http://doc.gorm.io/
13. JSON WebToken [Электронный ресурс] – Режим доступа: https://jwt.io/
14. Android Async http client [Электронный ресурс] – Режим доступа: http://loopj.com/android-async-http/

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

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Система хранения данных на основе Raspberry pi
тема

Руководитель	_____	доцент, канд. техн. наук	О. А. Русанова
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		И. В. Якимов
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____		В. И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2018