

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіотехнічних систем літальних апаратів
Кафедра комп'ютерних систем, мереж та кібербезпеки

КУРСОВИЙ ПРОЕКТ (РОБОТА)

з Проектування МПС
(назва дисципліни)

на тему: Сервіс хмарного зберігання файлів

Студента (ки) 4 курсу 545 групи
напряму підготовки комп'ютерна інженерія

Бесчетнікова І. О.
(прізвище та ініціали)

Керівник доц. Галькевич О. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії _____
(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

Содержание

ВВЕДЕНИЕ	3
1. ПОСТАНОВКА ЗАДАЧИ	4
1.1. Анализ предметной области.....	4
2. ПРОЕКТИРОВАНИЕ.....	11
2.1. Выбор архитектуры.....	11
2.2. Уровень web-приложения.....	11
2.2.1. Расширяемость	13
2.2.2. HTTP и HTML.....	14
2.3. Уровень сервиса.....	16
2.4. Уровень хранилища данных.....	16
2.5. UML –диаграмма использования.....	18
2.6. Проектирование графического интерфейса: проектирование представлений Web-приложения и интерфейса программы для устройств- серверов	25
3. РАЗРАБОТКА.....	31
3.1. Разработка диаграммы классов.....	31
3.2. Поля, методы и свойства	34
3.3. Разработка алгоритмов.....	39
4. ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ.....	43
ВЫВОДЫ.....	46
СПИСОК ЛИТЕРАТУРЫ.....	47
ПРИЛОЖЕНИЕ А Техническое задание	48
ПРИЛОЖЕНИЕ Б Руководство клиента	54
ПРИЛОЖЕНИЕ В Руководство администратора	56

ВВЕДЕНИЕ

Пояснительная записка курсового проекта включает в себя:

- страниц – 60;
- рисунков – 26;
- таблиц – 31;
- приложений – 3;
- источников – 5.

Целью данной курсовой работы является разработка приложения “Сервис облачного хранения файлов”.

Задачами курсовой работы являются:

- 1) реализация алгоритма загрузки и выгрузки файлов любых форматов и размеров;
- 2) реализация облачной инфраструктуры;
- 3) защита и шифрование данных пользователей;

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Анализ предметной области

Согласно закону Мура, количество транзисторов на кристалле процессора каждые 2 года растет, и кривая, характеризующая данное поведение, имеет экспоненциальный характер^[1].

Компьютерные устройства и компьютерное оборудование с каждым днем становится производительнее, а также требует больше объема памяти, на которой будет храниться гигантское количество по меркам конца XX и начала XXI столетия информации. Это проблема решалась и до сих пор решается разными способами: увеличение объема памяти на устройстве, использование легко переносимых флэш-накопителей, разделение данных между двумя или несколькими схожими устройствами, способных передавать информацию кабельным или радиоволновым путем. Однако эту актуальную проблему призваны легко решить сервисы облачного хранения данных, без каких-либо существенных денежных затрат для конечного пользователя.

Облачное хранилище или «Облако» – это онлайн-хранилище, которое распределяет все пользовательские данные на множестве удаленных друг от друга серверов. Об организации данного сервиса никто не может знать, кроме специалистов, принимавших участие в создании комплекса ПО для реализации этого проекта, поэтому это повышает надежность всей системы. Хакерские атаки должны быть направлены на множество точек на земном шаре, чтобы нарушить работу всего комплекса серверов, впоследствии чего теряются или же попадают в руки злоумышленников все пользовательские личные данные. Однако даже это будет осуществить непросто, в особенности, если у системы есть мощные ответные механизмы защиты и высоконадежные, отказоустойчивые, компоненты.

Помимо хранения информации, данная инфраструктура позволяет проводить самые разнообразные операции с данными, манипуляция которыми зачастую необходима нам в тех случаях, когда нет возможности скачать все необходимое на устройство, используемое в данный момент, или же, которое не содержит тех необходимых инструментов, которые предоставляет сервис облачного хранения данных: редактирование, конвертирование, архивирование, шифрование и т.д. Поэтому «облако» концептуально поддерживается всеми видами устройств, имеющих выход во всемирную сеть.

На сегодняшний день существует множество сервисов облачного хранения данных. Все они имеют как преимущества, так и недостатки, которые далее будут рассмотрены.

GoogleDrive

Один из самых популярных сервисов облачного хранилища, преимуществом которого является многофункциональный web-интерфейс, в котором доступны следующие операции:

- редактирование и создание документов в режиме совместного доступа;
- просмотр мультимедийного контента;
- расширение почти всех существующих надстроек;
- поиск документов;
- управление правами доступа.

На фоне всех других подобных сервисов можно выделить следующие преимущества и недостатки:

Таблица 1.1 – Преимущества и недостатки сервиса GoogleDrive

+	–
Неограниченное пространство под хранение изображение и фотографий.	Выделенное свободное пространство будет использоваться другими смежными сервисами.
Гибкий web-интерфейс, позволяющий настраивать рабочую среду под нужды пользователя.	Windows Desktop версия приложения не обладает какими-либо преимуществами.
При регистрации сразу предоставляется 15 ГБ бесплатного дискового пространства.	

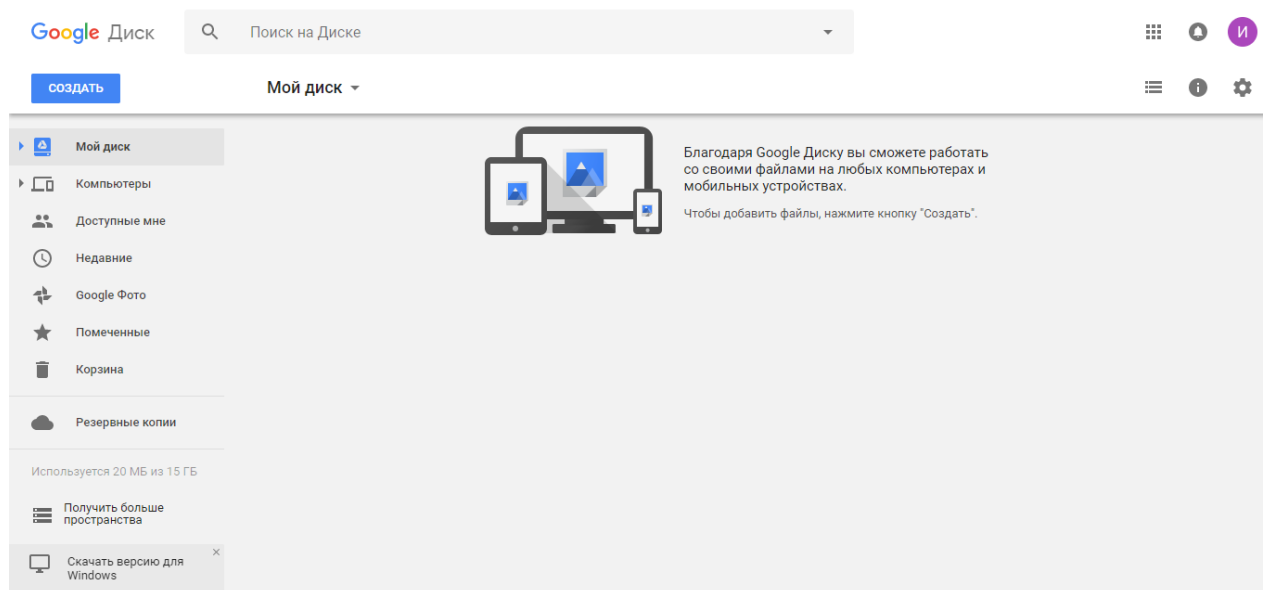


Рисунок 1.1 – Web-интерфейс сервиса GoogleDrive

Microsoft OneDrive

Надежное онлайн-хранилище от компании Microsoft, изначально интегрированное в ОС Windows 10, что дает возможность работать со своими данными на локальном компьютере с дальнейшей автосинхронизацией с «облаком». Отличительная особенность данного сервиса – интеграция с пакетом Office 365, дающая ряд преимуществ при работе с документами разного типа: Word, Excel, PowerPoint, Visio, Access, Publisher, OneNote и т.д.

Основные характеристики сервиса:

- бесплатное место: 5 ГБ;
- максимальный размер одного файла: 10 ГБ;
- максимальный объем пространства: 5 ТБ;
- совместимость с мобильными ОС: поддерживается на Android выше 4.0, iOS выше 9.0, Windows Phone 7 / 8, на Symbian Belle и на MeeGo 1.2.

Преимущества и недостатки сервиса:

Таблица 1.2 – Преимущества и недостатки сервиса OneDrive

+	–
Интеграция с пакетом Office 365.	Отсутствие дополнительных функций.
Высокая скорость и стабильная работа.	Бесплатный тариф предоставляет всего 5 ГБ свободного дискового пространства.
Скидки при покупке дискового пространства с Office 365.	

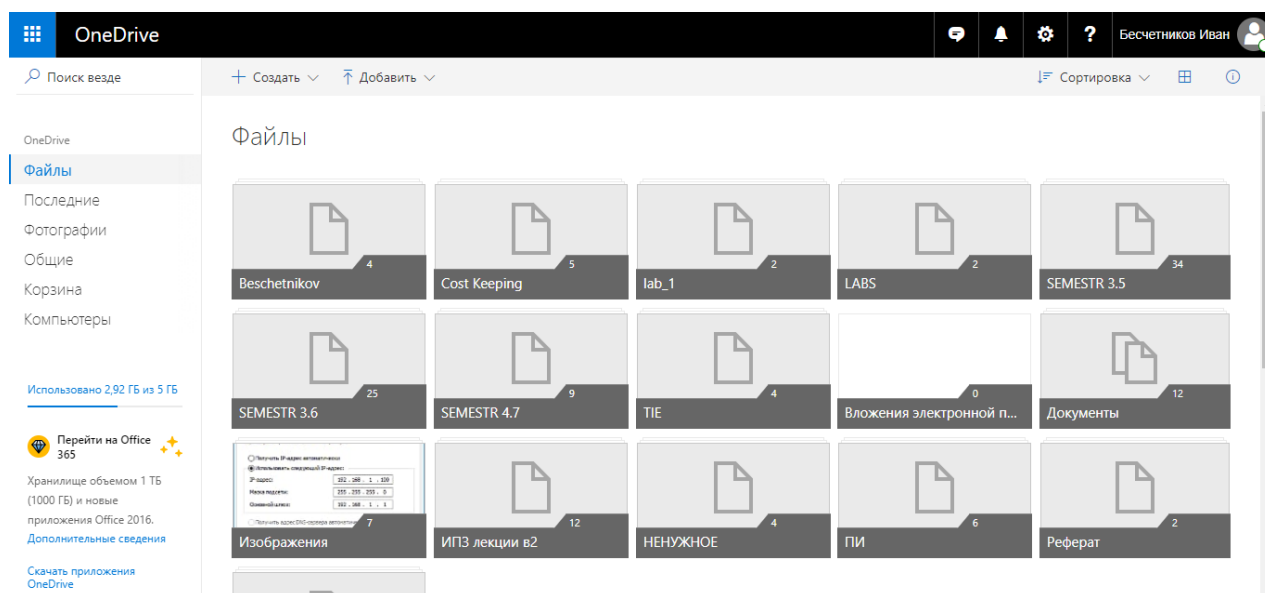


Рисунок 1.2 – Web-интерфейс сервиса OneDrive

Dropbox

Является одним из самых первых сервисов «облачной» архитектуры, который за все свое долгое время существования и развития успел доработать каждый узел инфраструктуры. Dropbox, как и большинство других популярных сервисов, предоставляет довольно функциональный и удобный web-интерфейс. Также он работает по модели Freemium, позволяющий при создании аккаунта выбрать объем свободного дискового пространства с дальнейшим увеличением, но при внесении оплаты.

Основные характеристики сервиса:

- бесплатное место: 2 ГБ;
- максимальный размер одного файла через web-интерфейс – 20 ГБ, через Windows Desktop – без ограничений;
- максимальный объем пространства: 1 ТБ;
- совместимость с мобильными ОС: поддерживается на Windows, macOS, Linux; на мобильных ОС Android, iOS, Windows Phone и BlackBerry.

Преимущества и недостатки сервиса:

Таблица 1.3 – Преимущества и недостатки сервиса Dropbox

+	–
Гибкий и функциональный web-интерфейс.	Максимально количество допустимых функций распространяется только для бизнес-клиентов.
Удобство в использовании на клиентских версиях.	Бесплатный тариф предоставляет всего 2 ГБ свободного дискового пространства.
256-битное шифрование AES и шифрование SSL.	
Лучшая среди аналогов технология синхронизации.	
Интеграция с Microsoft Office 365.	
Неограниченное восстановление файлов и журнал версий.	
Ссылки доступа с паролем и сроком действия.	
Возможность настраивать и управлять уровнями доступа.	

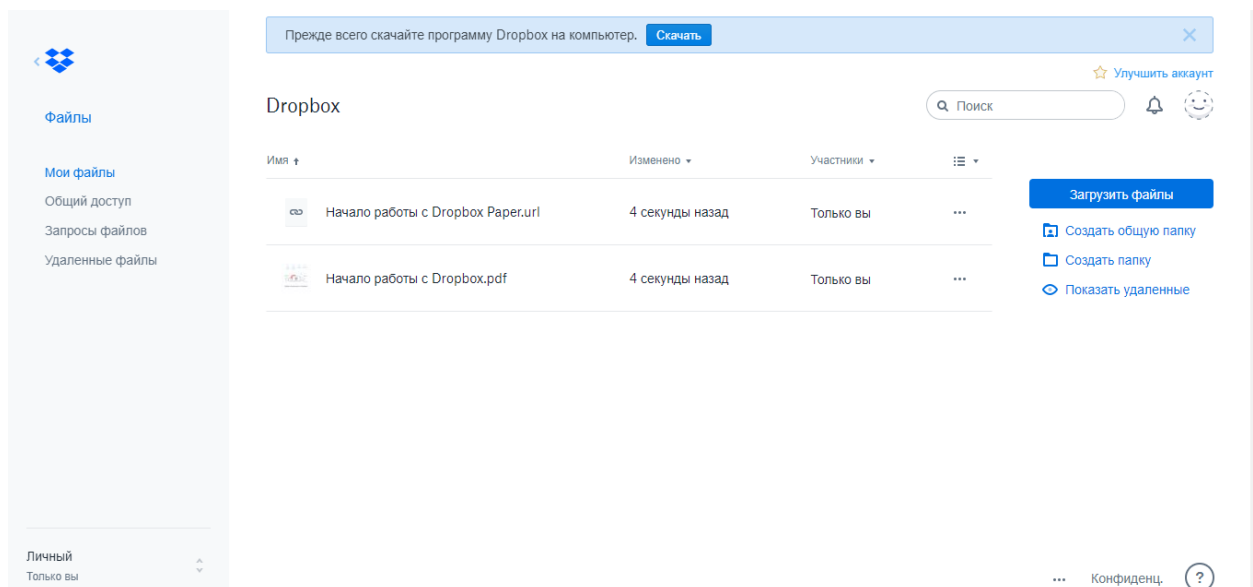


Рисунок 1.3 – Web-интерфейс сервиса Dropbox

Mega

Перспективный облачный файлообменник, предоставляющий максимально удобный набор инструментов для пользователей, которые не предпочитают выкладывать деньги за пользование небольшим объемом памяти под хранение личных данных. Такой сервис также предоставляет пользователям возможность делиться своими личными файлами по схеме «friend-to-friend», используя для этих целей надежный криптографический алгоритм шифрования AES, причем открытый ключ известен только тому пользователю, которому станет доступна ссылка на файл обмена.

«Mega» позиционируется как сервис, защищающий личные данные пользователей, поэтому используется сквозное шифрование.

Данный web-сервис существует в трех вариантах для использования: web-приложение, Windows Desktop приложение и мобильное приложение.

Основные характеристики сервиса:

- бесплатное место: 50 ГБ;
- максимальный размер одного файла через web-интерфейс – 10 ГБ, через десктопное приложение – без ограничений;
- максимальный объем пространства: 4 ТБ;
- совместимость с мобильными ОС: поддерживается на Windows, macOS, Linux; на мобильных ОС Android, iOS, Windows Phone и BlackBerry.

Преимущества и недостатки сервиса:

Таблица 1.4 – Преимущества и недостатки сервиса Dropbox

+	–
Плагины для браузеров.	Не высокая скорость доступа.
Интегрированный чат.	Существуют лимиты на трафик.
2048-битное шифрование AES и шифрование SSL.	
Предоставляется большой объем бесплатного хранилища – 50 ГБ.	

Сравнительная оценка некоторых параметров, существующих популярных облачных сервисов:

Таблица 1.5 – Сводная таблица для популярных облачных сервисов

Критерии оценки	Google Drive	Microsoft OneDrive	Mega	Dropbox
Максимально допустимый объем выделяемой памяти (ТБ)	30	5	4	1
Максимальный размер одного файла (ГБ)	5000	10	10	20
Шифрование данных	–	–	+	+
Интеграция с офисным пакетом документов от Microsoft	+	+	–	–
Совместимость с мобильными ОС	+	+	+	+
Синхронизация данных на ПК	+	+	–	+
Редактирование и создание документов в режиме совместного доступа	+	+	–	+
Просмотр мультимедийного контента	+	–	–	–
Большое разнообразие дополнительных функций	+	–	+	–

2. ПРОЕКТИРОВАНИЕ

2.1. Выбор архитектуры

В ходе проектирования инфраструктуры сервиса была выбрана трехуровневая архитектура:

- Уровень web-приложения;
- Уровень сервиса;
- Уровень хранилища данных.

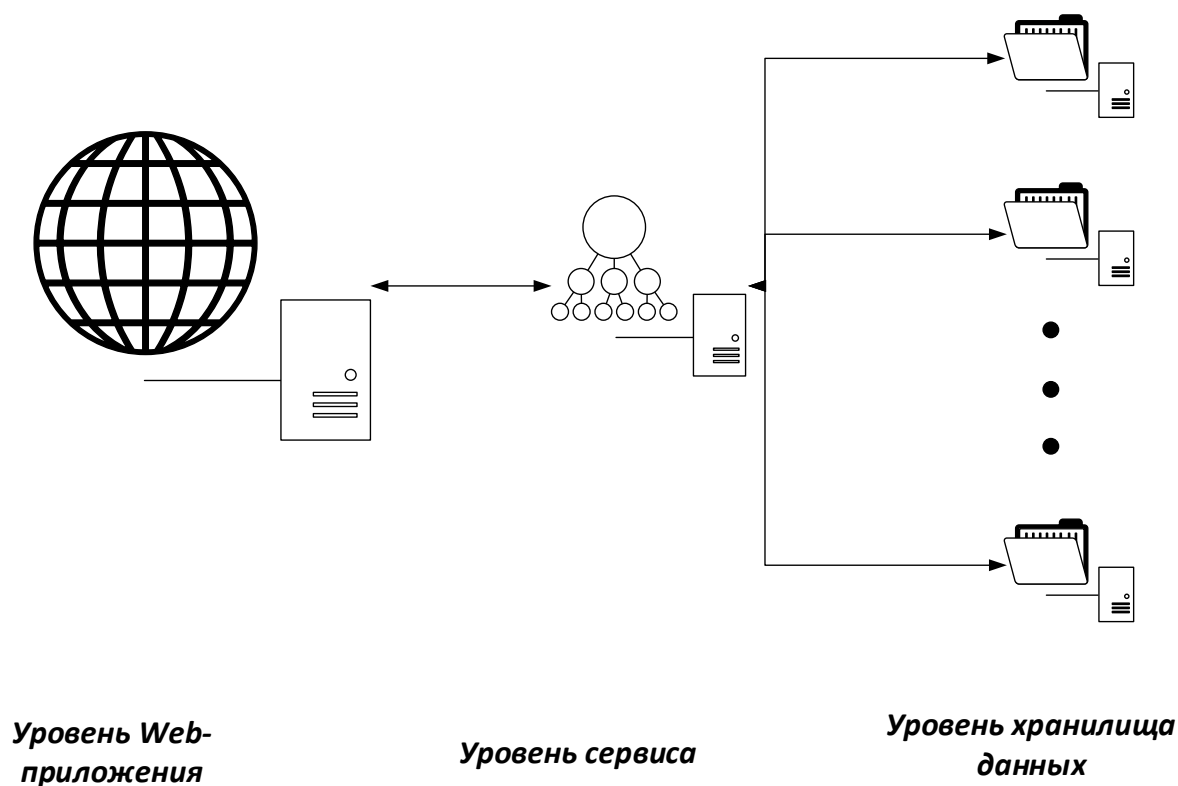


Рисунок 2.1 – Трехуровневая архитектура проекта

2.2. Уровень web-приложения

Уровень web-приложения представляет из себя приложение, которое является главным во всем проекте и служит для контроля всех пользовательских действий: регистрация, авторизация, аутентификация, валидация. В случае неуспеха выполнения данных процессов, будет выброшена системой соответствующая ошибка, которую пользователь будет обязан исправить, следуя полученной инструкции от браузера или web-приложения.

Для реализации этого узла системы был использован ASP.NET MVC 5 фреймворк для создания web-приложений. [\[2\]](#)

ASP.NET MVC 5 – это высоконадежная инфраструктура web-приложения, которая призвана обеспечить создание легковесной надежной архитектуры проекта, достигающееся путем заранее реализованных полноценных отказоустойчивых и протестированных модулей. Она также поддерживает расширяемость проектов, тестируемость, поэтому сопровождение становится легче и проще.

Эта платформа имеет в наличии отличный инструментарий, позволяющий создавать простые, а вместе с ним и гибкие модули. Встроенные вспомогательные методы HTML генерируют ясный и удобочитаемый код разметки, соответствующий всем современным стандартам описания графических web-интерфейсов. Высокопроизводительная маршрутизация дает возможность создавать удобные, читабельные URL-адреса, благодаря чему программист волен придумывать любые осмысленные имена страницам, или же имена, которые скрывают некоторую логику работы всего web-приложения.

Проекты, написанные на ASP.NET MVC 5 поддерживают расширяемость, а это значит, что каждый отдельный модуль или модуль, встраиваемый в систему можно проверить в модульном тестировании, а интеграционные тесты будут отслеживать состояние всей системы в фоновом режиме работы.

Данная платформа для разработки web-приложений использует архитектурный шаблон MVC. К преимуществам этого шаблона можно отнести следующее:

- Пользователь взаимодействует с приложением в соответствии с естественным циклом: пользователь совершает некоторое действие, в ответ на которое модель изменяет свое состояние и рендерит определенное представление, что хорошо вписывается в логику работы запросов и ответов протокола HTTP.
- Ряд web-приложений нуждаются в разделении модулей всего проекта: HTML-представления, компоненты по улучшению web-интерфейса, базы данных, бизнес-логика, сервисы и т.д. Все это отлично вписывается в концепцию MVC. [\[3\]](#)

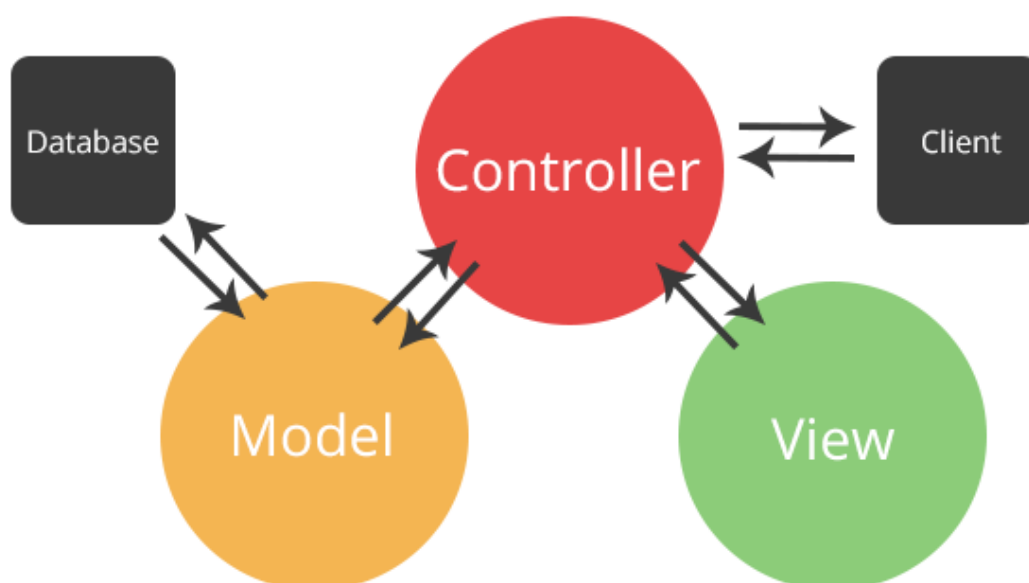


Рисунок 2.2 – Архитектура MVC-проектов

Инфраструктура ASP.NET MVC Framework реализует этот шаблон, причем с доработанными и улучшенными некоторыми сценариями использования данного паттерна.

2.2.1. Расширяемость

Инфраструктура MVC Framework построена таким образом, компоненты которой являются взаимно независимыми и представляются в виде абстрактных базовых классов. Поэтому существуют следующие схемы использования встроенных компонентов:

- Полное использование встроенных компонентов;
- Частичное использование встроенных компонентов с некоторой корректировкой, реализуя собственный класс с наследованием от абстрактного класса или интерфейса всех его методов и свойств;
- Полная замена существующего компонента собственнореализованным.

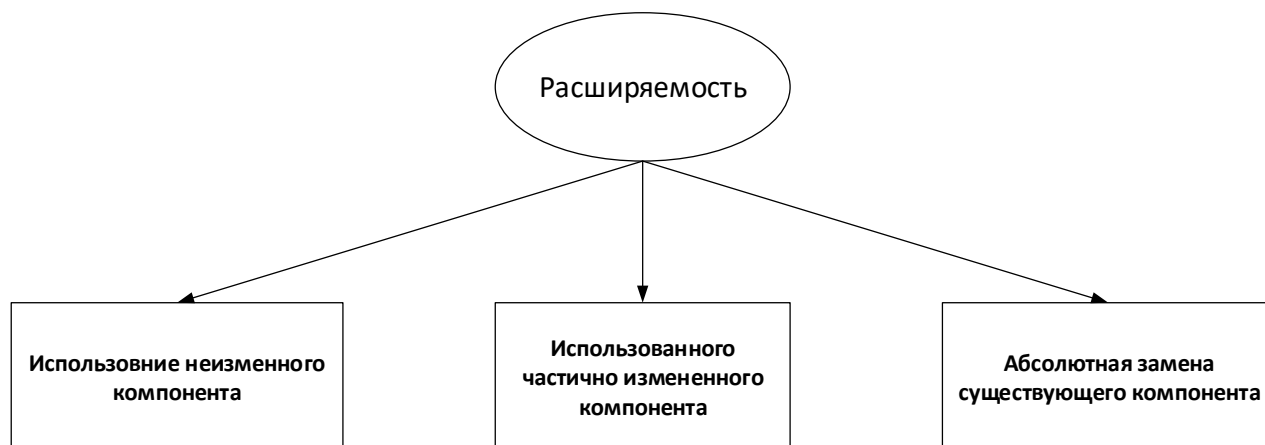


Рисунок 2.3 – Сценарии использования встроенных компонентов ASP.NET MVC Framework

2.2.2. HTTP и HTML

Инфраструктура ASP.NET MVC Framework генерирует ясную и простую разметку, при использовании HTML-хелперов. Однако помимо простой генерации HTML-разметки, она способна на построение элегантных, легко обрабатываемых блоков кода-разметки, которые будут в значительно большей степени лучше отрендерены, нежели чем бесчисленное множество сложных блоков, которыми трудно управлять. Генерируемый код вдобавок будет оформлен красивыми и «дружелюбными» стилями CSS и плавной анимацией JavaScript.

С другой стороны, если требуется создание более сложных элементов управления, такие как DatePicker, Clock, CascadeMenu и т.д., то можно воспользоваться встроенными библиотеками, которые значительно ускоряют и упрощают процесс разработки таких элементов. Например, библиотеки jQuery и Bootstrap CSS уже в последних версиях сред разработок, таких как Visual Studio и Code Studio, имеются, и значатся, как встроенные компоненты, эффективность которых оспаривать никто не станет.

Так же, в отличие от страниц WEB.FORMS страницы ASP.NET MVC Framework значительно меньше по размеру, даже несмотря на то, что скорость соединения с web-сайтами возросла в разы, тем не менее это повышает комфорт использования для конечного пользователя и ускоряет процесс запуска самого web-приложения. Это достигается за счет отсутствия View State-данных страницы, с которыми в основном работает платформа WEB.FORMS.

ASP.NET MVC тесно связана с HTTP-протоколом. ^[4] Это позволяет контролировать запросы и ответы, возникающие между сервером и клиентом. Благодаря этому, можно точно корректировать каждый момент работы двух сторон. А при использовании AJAX-запросов не будут нужны никакие

автоматические отправки обратного вызова, т.к. она проста и не использует состояние клиентской стороны.

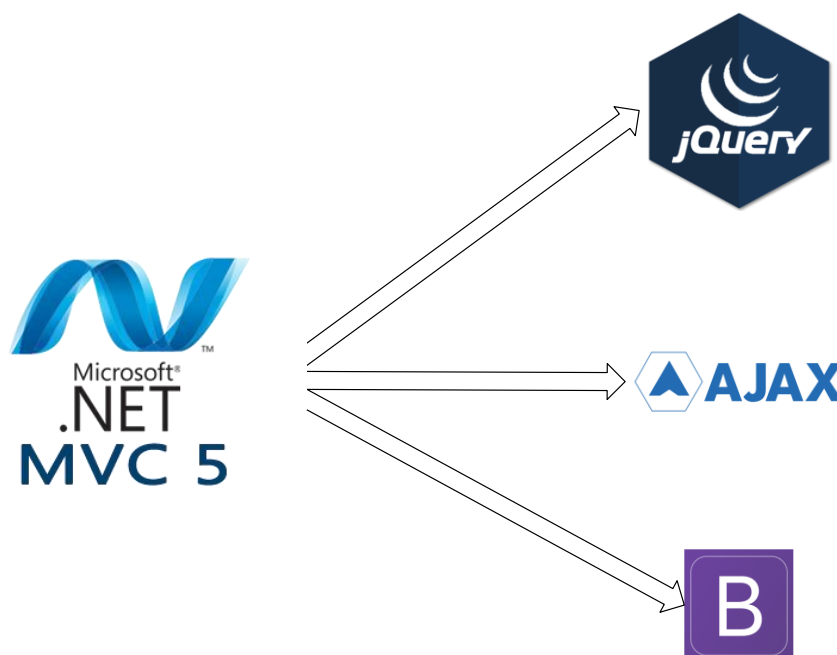


Рисунок 2.4 – Библиотеки, встроенные по умолчанию в ASP.NET MVC 5

Тестируемость

Разделение ответственности между всеми частями web-приложения – то, что позволяет данный фреймворк, делая сопровождение и тестирование отдельных частей всего приложения гораздо проще и удобнее. Однако на этом возможности ASP.NET MVC Framework платформы не ограничены. Для каждого модуля компонентно-ориентированного проекта всей системы, она обеспечивает структурированность, которая так необходима при использовании модульного тестирования.

Помимо модульного тестирования, существуют также средства, призванные симитировать как действия пользователя, так и взаимодействие с ним в графическом интерфейсе приложения, обработка которых автоматически осуществляется всей инфраструктурой, и нет больше необходимости следить за изменениями структуры HTML-разметки, или же отслеживать генерируемые HTML-элементы и идентификаторы CSS.

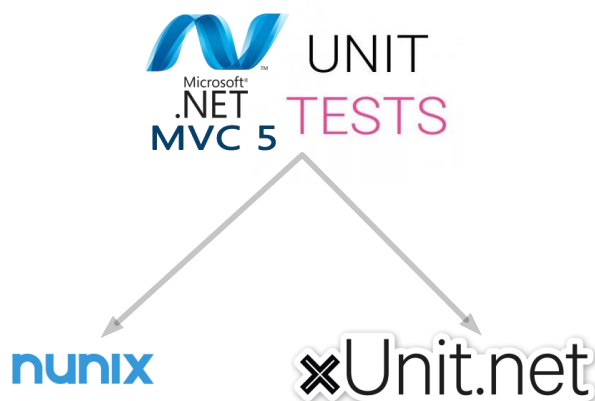


Рисунок 2.5 – Интегрированные инструменты модульного тестирования

2.3. Уровень сервиса

Уровень сервиса. Данный уровень является интерфейсом по обмену данными между web-приложением и серверными приложениями. Его работа осуществляется независимо от работы web-приложения, однако уровень web-приложения может получать некоторую информацию о процессе транзакции по извлечению или загрузке файлов.

Уровень хранилища данных. На данном уровне оперируются один или группа файлов, которые следует сохранить или извлечь из памяти устройства. При извлечении данные передаются уровню, с которым он непосредственно связан – уровень сервиса. В случае возникновения ошибок, приложение этого уровня должно сообщить приложению сервиса об этом, а также ее название. Тогда уровень сервиса в обязательном порядке предпримет попытку по устранению ошибки, если, конечно, она логическая и не связана с аппаратной составляющей серверов-хранилищ. Примерами таких ошибок могут выступать:

- Потеря пакетов данных при передаче или извлечении данных;
- Временная недоступность сервера-хранилища;
- Нехватка свободной памяти;

2.4. Уровень хранилища данных

Уровень хранилища данных. Данный уровень представляет из себя множество приложений, установленных на устройствах, способных хранить, получать и извлекать информацию из памяти. Это может быть любой ПК, любая рабочая станция или любой гаджет, который работает под управлением четырех операционных систем: Windows 10 (любая редакция), Android (любая

версия), iOS (любая версия) и Windows Phone 8.1. Такая интероперабельность достигается за счет использования Xamarin. [\[5\]](#)



Рисунок 2.6 – кроссплатформенность Xamarin

Xamarin – это инструмент разработки кроссплатформенного ПО, позволяющий создавать приложения под управлением платформы .NET, которая в свою очередь предоставляет возможность использовать такие языки программирования как C#, F# и Visual Basic; а также приложения, написанные на языке программирования C++.

Несомненно, это удобное решение для проектов, в которых реализуется кроссплатформенность за достаточно короткие сроки. К преимуществам данного выбора можно отнести следующее:

- создание унифицированного кода для устройств, работающих под управлением разных ОС;
- предоставляется возможность обращаться к нативным API разных платформ;
- использование в качестве языка программирования – C#, который прост в понимании и использовании.

А к недостаткам данного выбора можно отнести следующее:

- приложения выходят менее производительными;
- приложения используют больше памяти, чем, если бы они были написаны на Java или C++ для отдельно взятой платформы.

В данном проекте использовались Windows 10 и Android платформы. Android с пакетами для двух версий: 4.2 и 7.1.

2.5. UML –диаграмма использования

Функции, реализованные в проекте, можно представить с помощью диаграммы прецедентов.

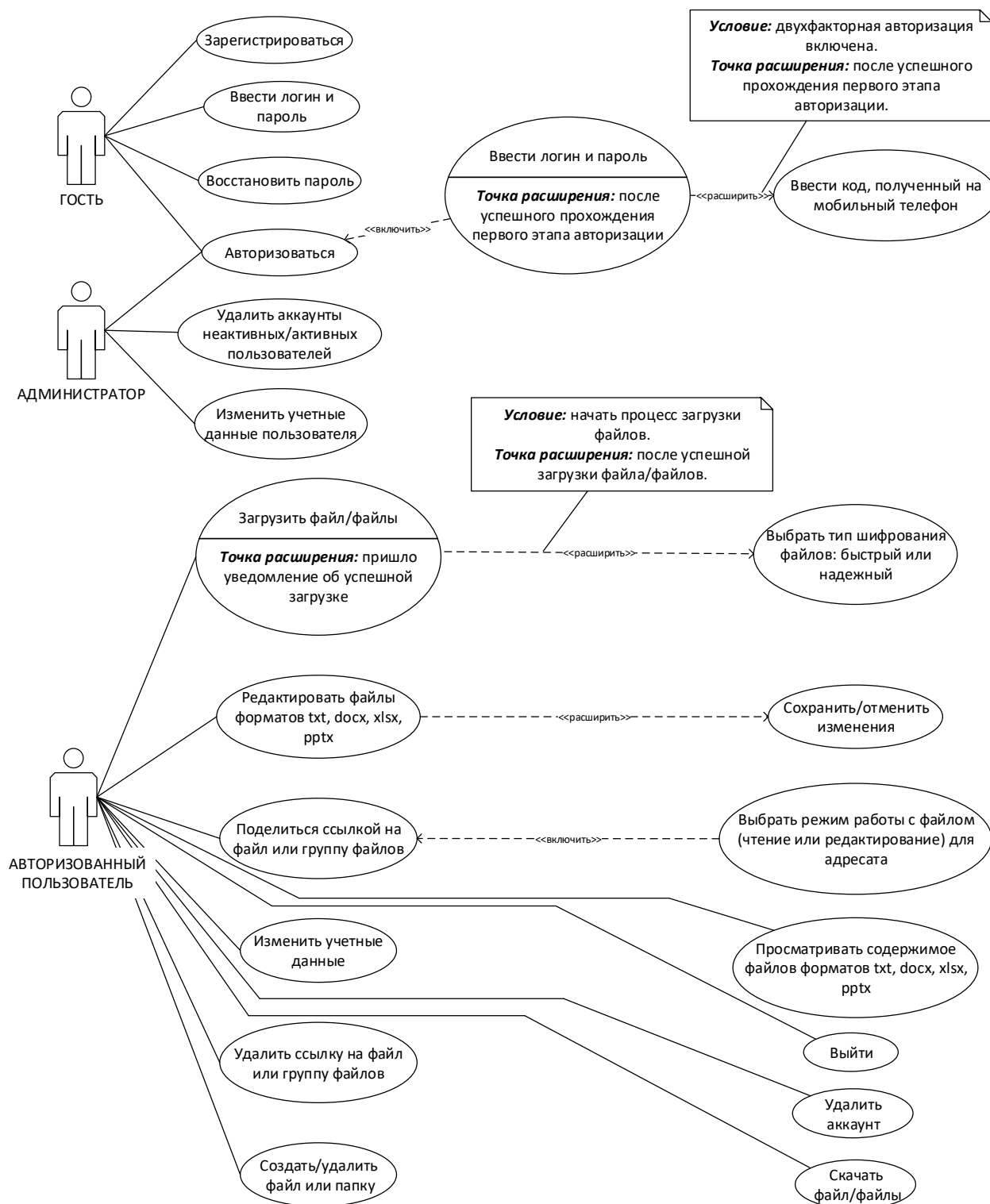


Рисунок 2.7 – UML-диаграмма использования. Взаимодействие пользователя с интерфейсом программы и ее функциями.

Действующее лицо «Гость»: данный актер представляет из себя пользователя, который не авторизован или не имеет зарегистрированной учетной записи в этом сервисе.

Действующее лицо «Авторизованный пользователь»: данный актер представляет из себя пользователя, который вошел в систему, пройдя этап авторизации или же регистрации.

Действующее лицо «Администратор»: данный актер представляет из себя пользователя, уполномоченное по администрированию учетными записями других пользователей.

Таблица 2.1 – Прецедент «Зарегистрироваться»

Название прецедента	«Зарегистрироваться»
Описание	Данный прецедент позволяет регистрироваться на данный сервис.

Таблица 2.2 – Прецедент «Авторизоваться»

Название прецедента	«Авторизоваться»
Описание	Данный прецедент позволяет авторизоваться.

Таблица 2.3 – Прецедент «Добавить запись»

Название прецедента	«Добавить запись»
Описание	Данный прецедент позволяет добавить строку-запись, в которой можно осуществить следующие операции: ввести сумму, выбрать категорию, добавить категорию, выбрать дату, добавить комментарий к записи.

Таблица 2.4 – Прецедент «Восстановить пароль»

Название прецедента	«Восстановить пароль»
Описание	Данный прецедент позволяет восстановить доступ к аккаунту, в случае если пользователь забыл пароль.

Таблица 2.5 – Прецедент «Удалить аккаунты неактивных пользователей»

Название прецедента	«Удалить аккаунты неактивных пользователей»
Описание	Данный прецедент позволяет удалять администратору аккаунты тех пользователей, а также на свое усмотрение, которые на протяжении длительного времени не пользовались данным сервисом.

Таблица 2.6 – Прецедент «Изменить учетные данные пользователя»

Название прецедента	«Изменить учетные данные пользователя»
Описание	Данный прецедент позволяет администратору изменять учетные данные пользователя, такие как логин, пароль, номер мобильного телефона и другое, если пользователю не удастся войти в свой аккаунт.

Таблица 2.7 – Прецедент «Изменить учетные данные пользователя»

Название прецедента	«Изменить учетные данные пользователя»
Описание	Данный прецедент позволяет администратору изменять учетные данные пользователя, такие как логин, пароль, номер мобильного телефона и другое, если пользователю не удастся войти в свой аккаунт.

Таблица 2.8 – Прецедент «Удалить аккаунт»

Название прецедента	«Удалить аккаунт»
Описание	Данный прецедент позволяют пользователю при необходимости удалить собственный аккаунт.

Таблица 2.9 – Прецедент «Выйти»

Название прецедента	«Выйти»
Описание	Данный прецедент позволяет выйти из учетной записи.

Таблица 2.10 – Прецедент «Удалить файл или папку»

Название прецедента	«Удалить файл или папку»
Описание	Данный прецедент позволяет пользователю удалять существующую папку или файл.

Таблица 2.11 – Прецедент «Создать файл или папку»

Название прецедента	«Создать файл или папку»
Описание	Данный прецедент позволяет пользователю создать папку или файл.

Таблица 2.12 – Прецедент «Удалить ссылку на файл или группу файлов»

Название прецедента	«Удалить ссылку на файл или группу файлов»
Описание	Данный прецедент позволяет удалять пользователю ранее созданную ссылку на файл или группу файлов для лиц, получивших доступ к ним.

Таблица 2.13 – Прецедент «Поделиться ссылкой на файл или группу файлов»

Название прецедента	«Поделиться ссылкой на файл или группу файлов»
Описание	Данный прецедент позволяет генерировать ссылку на файл или группу файлов для доверенных пользователей.

Таблица 2.14 – Прецедент «Изменить учетные данные»

Название прецедента	«Изменить учетные данные пользователя»
Описание	Данный прецедент позволяет пользователю изменять некоторые учетные данные: логин, пароль, ФИО, аватар, номер мобильного телефона и т.д.

Таблица 2.15 – Прецедент «Просматривать содержимое файлов форматов docx, xlsx, pptx, txt»

Название прецедента	«Просматривать содержимое файлов форматов docx, xlsx, pptx, txt»
Описание	Данный прецедент позволяет пользователю просматривать содержимое файлов с такими расширениями.

Таблица 2.16 – Прецедент «Редактировать файлы форматов docx, xlsx, pptx, txt»

Название прецедента	«Редактировать файлы форматов docx, xlsx, pptx, txt»
Описание	Данный прецедент позволяет редактировать файлы с такими расширениями.

Таблица 2.17 – Прецедент «Сохранить изменения»

Название прецедента	«Сохранить изменения»
Описание	Данный прецедент позволяет пользователю сохранить все внесенные изменения в файле при его редактировании.

Таблица 2.18 – Прецедент «Скачать файл/файлы»

Название прецедента	«Скачать файл/файлы я»
Описание	Данный прецедент позволяет скачивать файл или группу файлов из сервиса на используемое в данный момент устройство пользователем.

Таблица 2.19 – Прецедент «Загрузить файл/файлы»

Название прецедента	«Загрузить файл/файлы»
Описание	Данный прецедент позволяет пользователю загружать файлы или группу файл в сервис из устройства, используемое в данный момент времени.

2.6. Проектирование графического интерфейса: проектирование представлений Web-приложения и интерфейса программы для устройств-серверов

Пользовательский интерфейс был реализован с использованием HTML, CSS и движка представлений Razor. `_ViewStart` – представление, визуализирующееся при запуске приложения. Оно не содержит визуальных элементов, а устанавливает представление шаблона (`_Layout`). Представление `_Layout` определяет шаблон страниц авторизации и регистрации приложения. Всего в приложении имеется порядка 30 представлений.

Страница авторизации предлагает ввести существующие данные учетной записи пользователя в соответствующие поля для ввода: Email и Password.

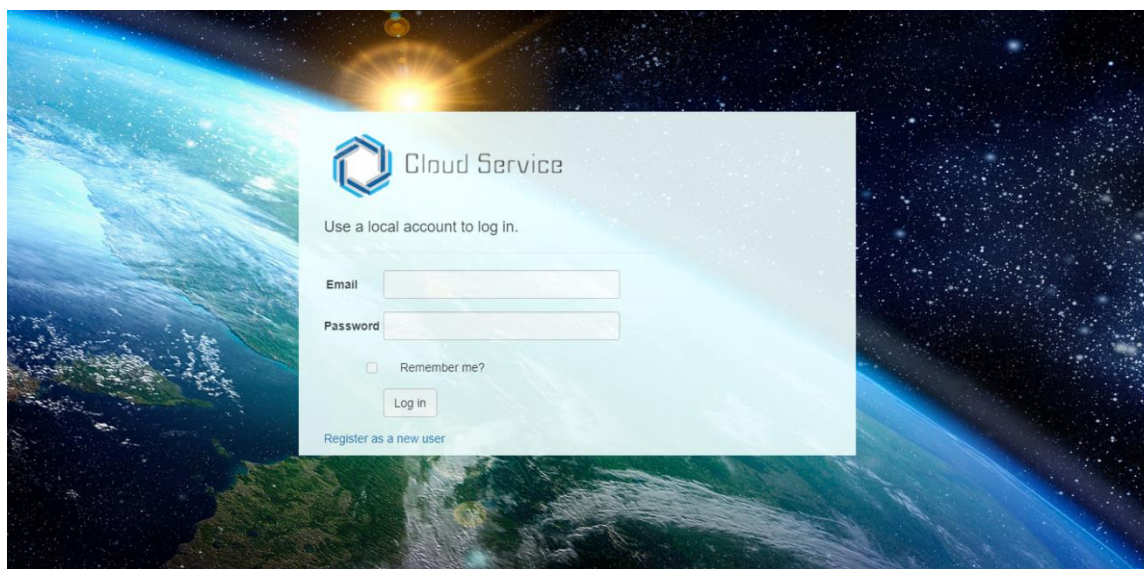


Рисунок 2.8 – Страница авторизации.

Если же пользователь является Гостем в системе, то он может перейти на страницу регистрации по соответствующей ссылке ниже, а затем зарегистрироваться, предварительно заполнив все необходимые поля.

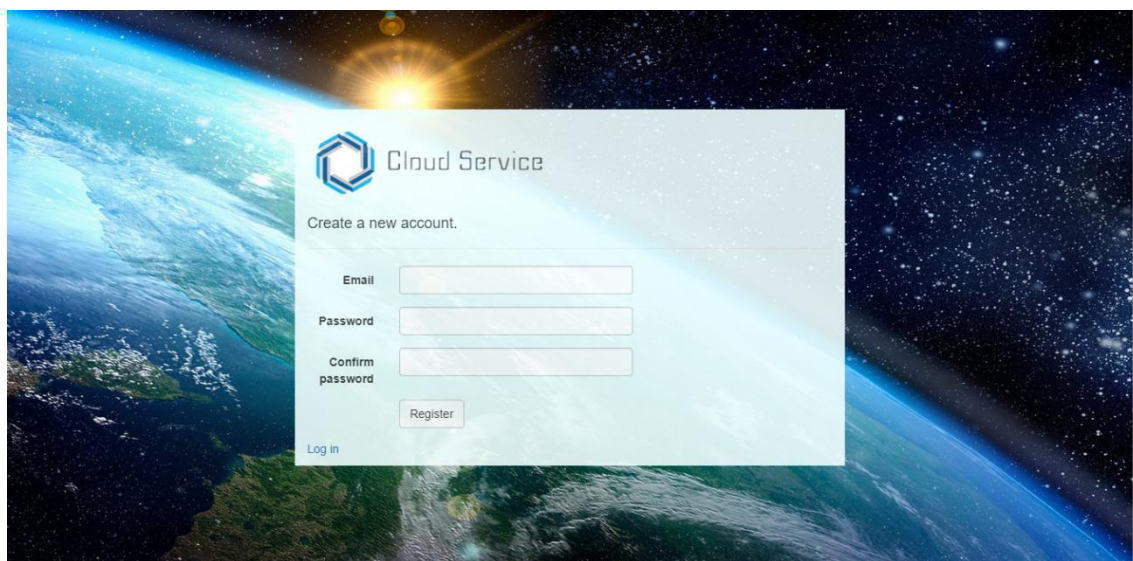


Рисунок 2.9 – Страница регистрации.

Представление `_Layout2` определяет шаблон всех остальных страниц. В верхней части второго шаблона находится логотип, который является ссылкой на домашнюю страницу приложения, а также множество ссылок на частичные представления, необходимые по работе с полем, содержащим все ранее загруженные пользовательские файлы и папки. При необходимости, есть две управляющие ссылки учетной записью, которые позволят деавторизоваться и настроить параметры аккаунта



Рисунок 2.10 – Панель управления для авторизованного пользователя.

Используя эту навигационную панель, можно загружать и выгружать файлы или папки в облачном хранилище. Для того, чтобы загрузить файл или группу файлов, а также папку или группу папок в хранилище сервера, предназначена кнопка «Download», при нажатии которой выводится небольшое контекстное меню с выбором объектов, выгружаемых на сервер: файлы или папки.



Рисунок 2.11 – Кнопка загрузки файлов на сервер, а также кнопка скачивания.

Для того, чтобы скачать файлы с сервера, предназначена кнопка «Upload», которая расположена левее, чем кнопка «Download». Она сработает только в том случае, если пользователь выбрал необходимые файлы, отметив их флажками.

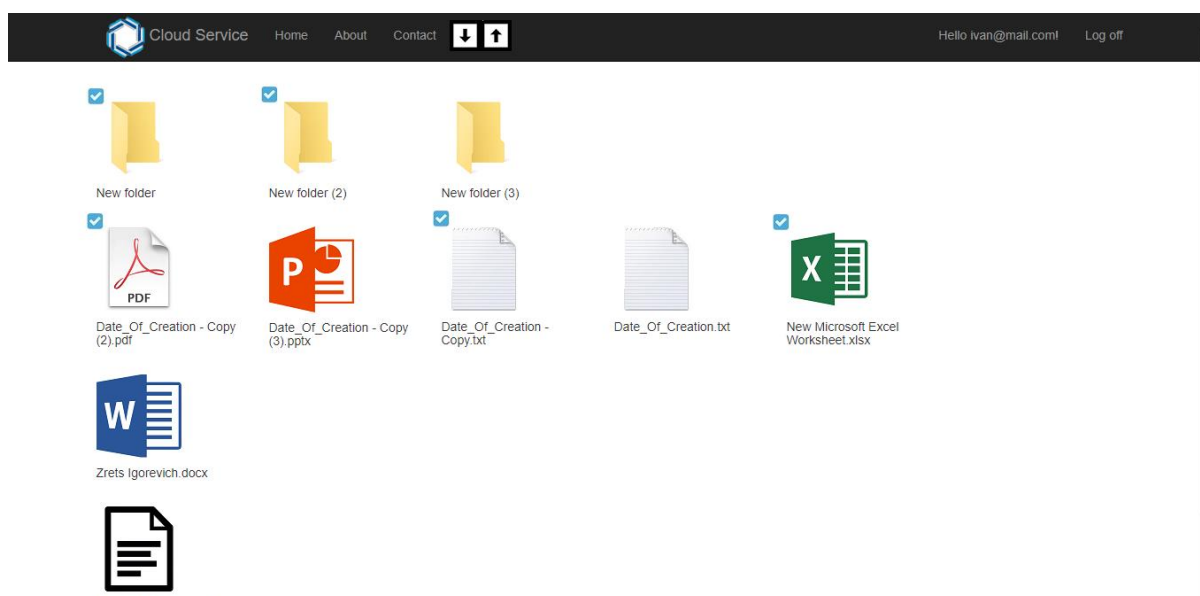


Рисунок 2.12 – Страница со всеми личными файлами и папками.

Затем сервис автоматически их соберет в архив, который и будет скачан в конечном итоге пользователем на ПК или гаджет.

Т.к. в проекте не менее 30 как частичных, так и простых страниц, то далее будут рассмотрены основные, которые демонстрируют выполнение основного функционала приложением.

DisplayFiles (FileController)

Частичное представление, в котором все файлы и папки отображаются в отсортированном виде, поэтому пользователю будет легче ориентироваться в данном информационном пространстве.

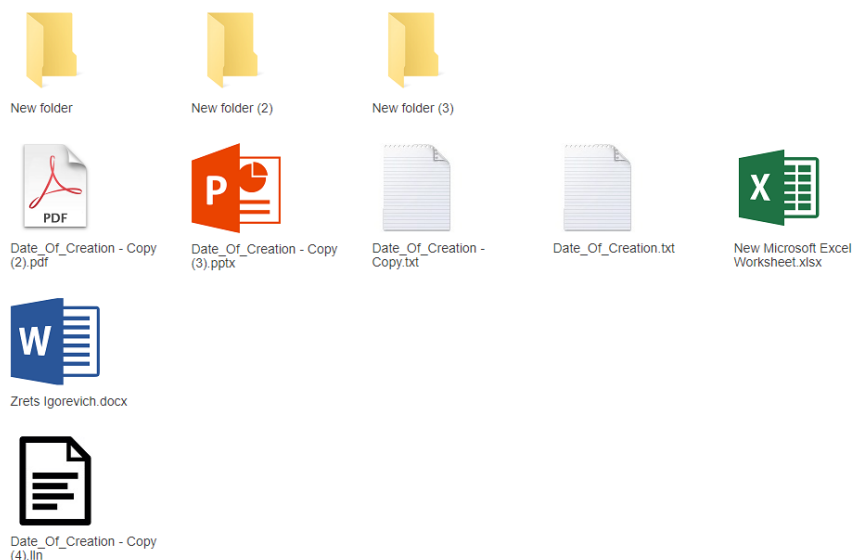


Рисунок 2.13 – Результат работы метода DisplayFiles для контроллера FileController.

Все папки всегда располагаются в самой верхней части этой среды, а все файлы – ниже. Также для увеличения удобства в поиске необходимого файла/файлов файлы, которые входят в состав пакета офисных документов Microsoft, такие как Word, Excel, PowerPoint, а также стандартные документы расширений .txt и .pdf, они будут располагаться выше, нежели файлов, нераспознанных сервисом.

Index (ManageController)

Это представление позволяет авторизованному пользователю настраивать свою учетную запись:

- изменить логин;
- изменить пароль;
- добавить номер мобильного телефона;
- включить или выключить двухфакторную авторизацию.

Чтобы перейти на эту страницу, следует нажать на ссылку в правом верхнем углу собственного логина.

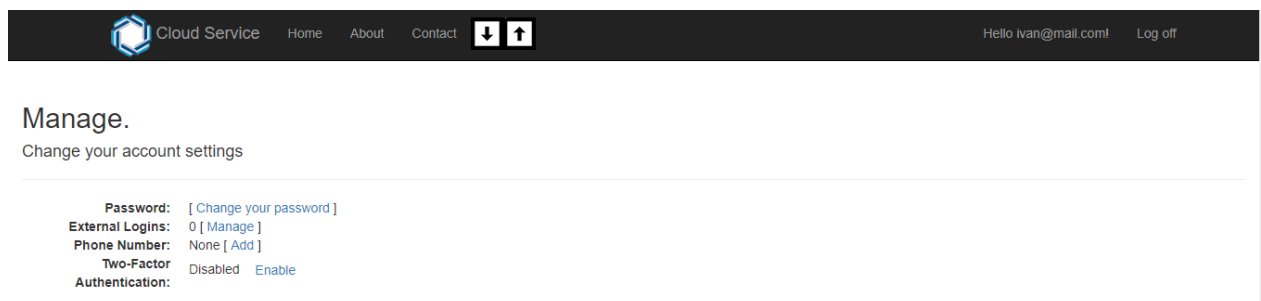
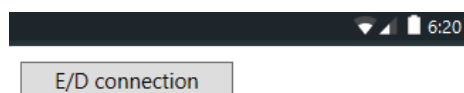


Рисунок 2.14 – Результат работы метода Index для контроллера ManageController.

Разрабатываемый графический интерфейс для устройств-серверов представляет из себя единственное окно приложения минималистичного вида, которое состоит всего из трех частей: кнопка активации и деактивации соединения с сервисом по обмену данными, стек статуса работы всего приложения и стек ошибок.



Status Stack

Error Stack



Рисунок 2.15 – Интерфейс программы для серверного устройства.

Кнопка активации и деактивации соединения служит для того, чтобы уведомить сервис о том, что устройство готово к обмену данными, как, например, при запуске приложения; или же для того, чтобы прервать соединение, в случае некоторых технических неполадок.

Стек статуса выводит краткую информацию о времени работы всего приложения, количество использованного трафика в процессе обмена данными, количество полученных файлов и количество переданных сервису файлов.

Стек ошибок выводит информацию об ошибках, возникающих в процессе работы приложения, поэтому в случае обнаружения оператором конфликтов между web-приложением и устройствами-серверами, он может получить необходимую информацию об ошибке здесь. Следует заметить, что ошибки (кроме технических) не прерывают работу всего приложения.

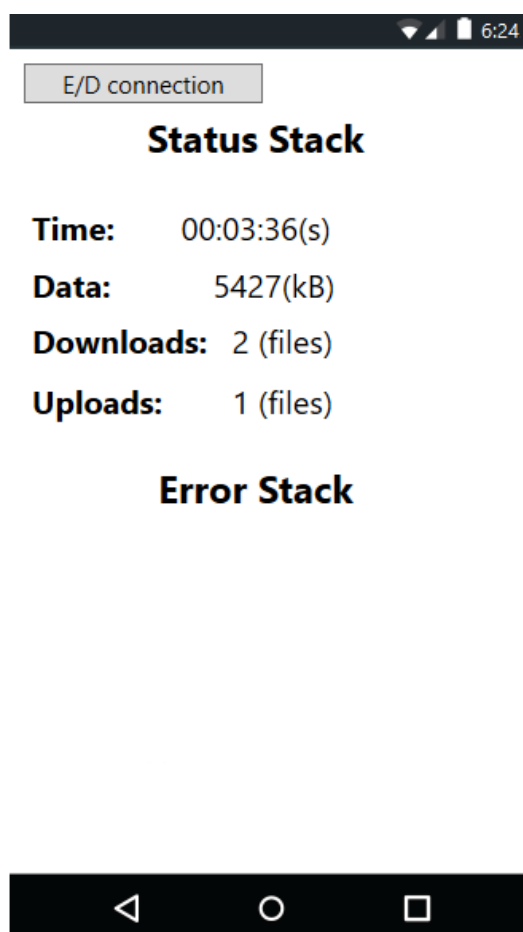


Рисунок 2.16 – Интерфейс программы для устройства-сервера в режиме работы.

3. РАЗРАБОТКА

3.1. Разработка диаграммы классов

При анализе поставленной задачи были выделены три основные сущности, представленные тремя проектами, в которых были созданы классы и интерфейсы для их взаимодействия: UI (HomeController, AccountController, ManageController, MobileServerController, FileController, MainPage), BusinessLogic (Service1, StateObject, MainPage, FileModel, FileNFolders,), DAL (ApplicationDbContext, ApplicationUser, ApplicationUserManager).

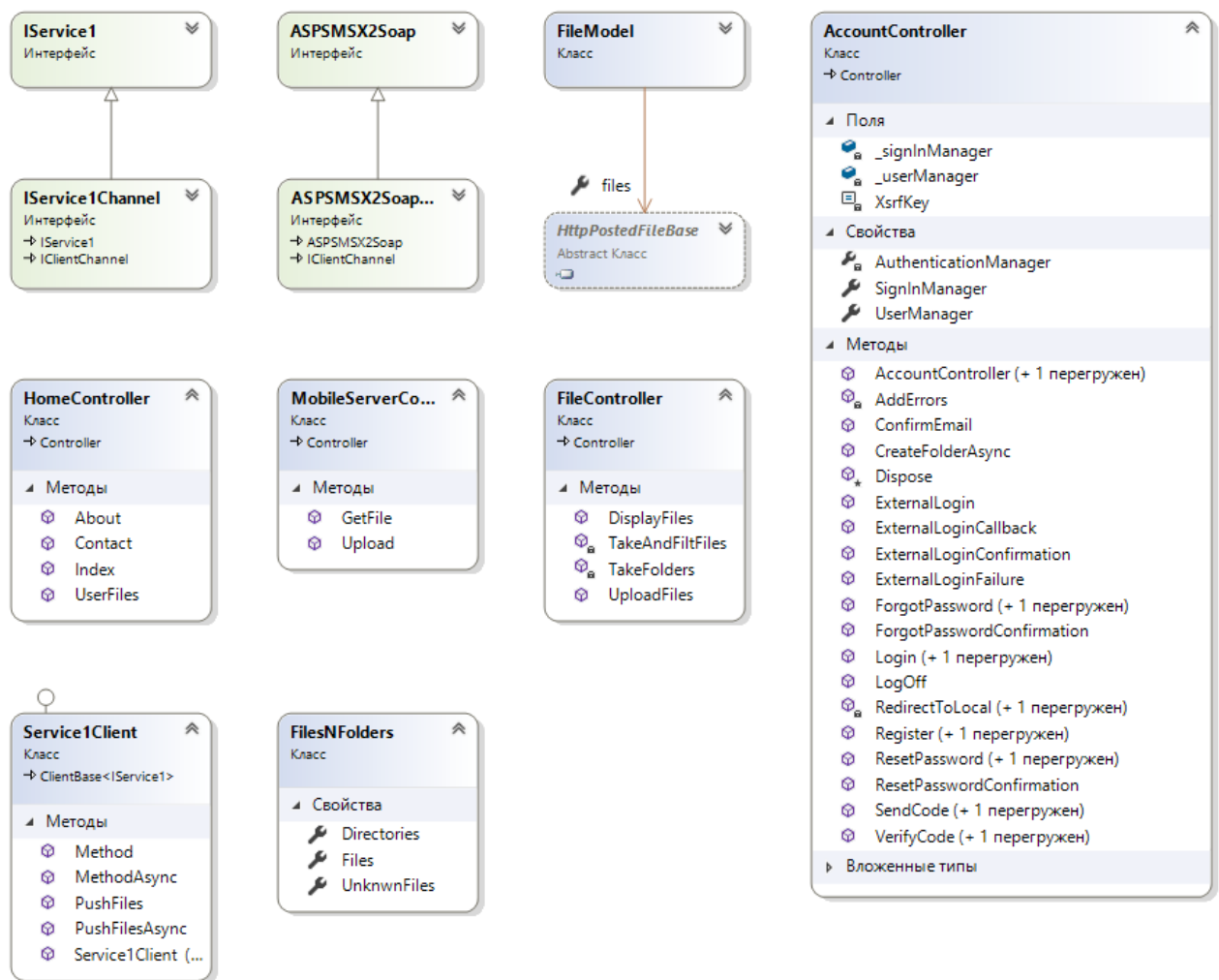


Рисунок 3.1 – Диаграмма классов web-приложения.

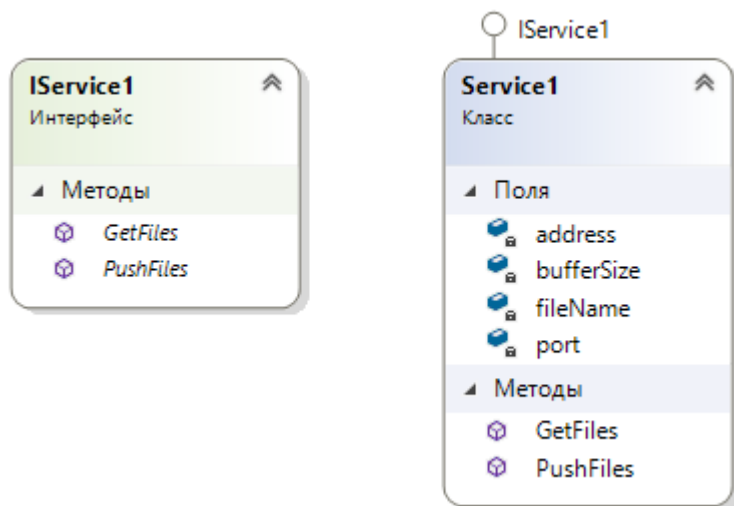


Рисунок 3.2 – Диаграмма классов сервиса.

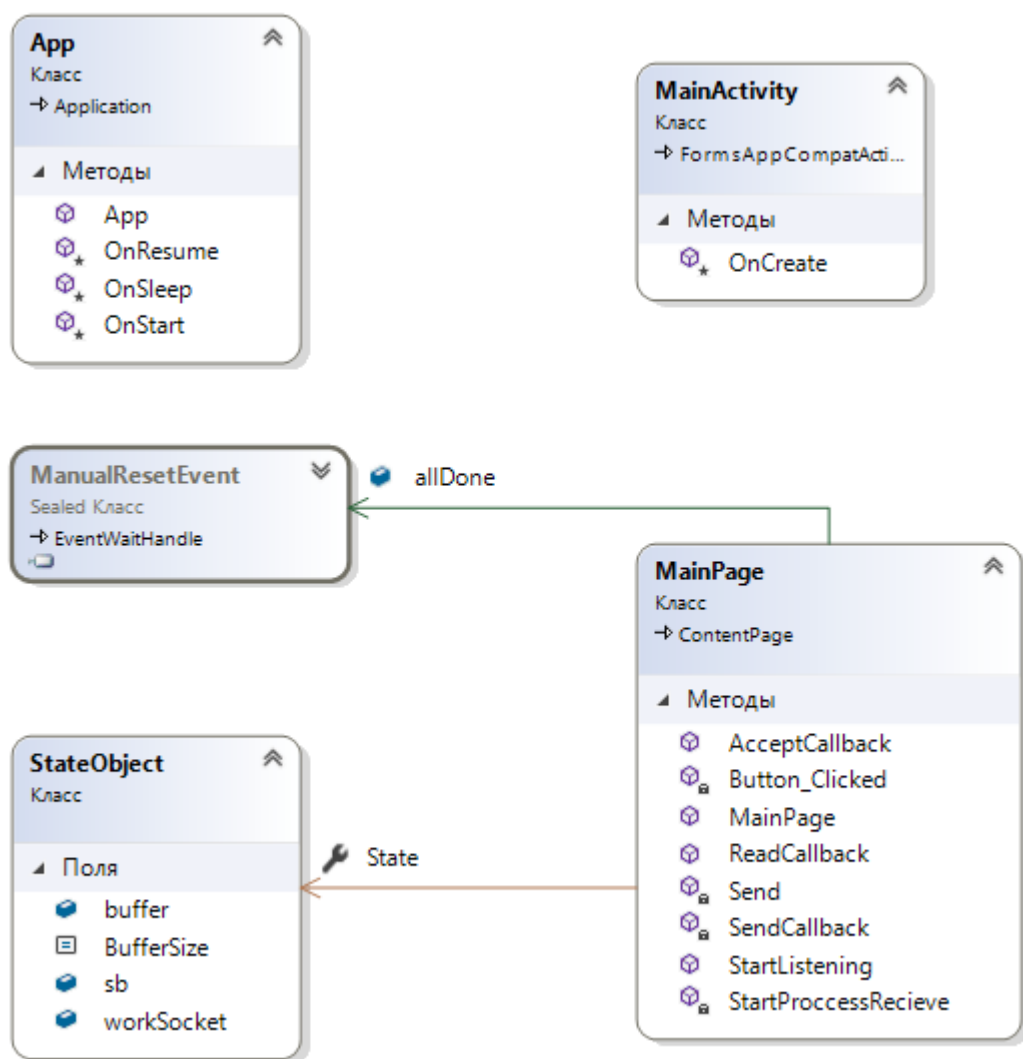


Рисунок 3.3 – Диаграмма классов сервера.

Класс HomeController – контроллер, отвечающий за основную маршрутизацию в приложении: вывод стартовой страницы, основной страницы авторизованного пользователя с его личными файлами; вывод страницы с контактной информацией.

Класс AccountController – контроллер, отвечающий за создание и управление аккаунтом пользователя: регистрация, авторизация, изменение логина или пароля, включение двухфакторной авторизации и т.п.

Класс FileController – контроллер, отвечающий за сортировку файлов по типу, редактирование, сохранение изменений в файле.

Класс MobileServerController – контроллер, отвечающий за загрузку файлов на сервер, предварительно передав их сервису, скачивание файлов, а также шифрование и дешифрование файлов.

Класс FilesNFolders – вспомогательный класс, представляющий собой объект всех пользовательских файлов и папок, содержащий такие свойства: количество файлов, папок; типы файлов, количество страниц.

Класс ServiceClient – класс, отвечающий за связывание удаленной службы и web-приложения.

Класс Service1 – класс на стороне службы, принимающий и передающий файлы клиенту, а также распределяющий файлы между всеми устройствами-серверами по определенному алгоритму.

Класс App – класс, отвечающий за жизненный цикл функционирования мобильного приложения.

Класс MainActivity – класс, объект которого каждый раз создается при запуске приложения на Android-устройстве; в котором имеются все необходимые методы и свойства для нормальной работы приложения непосредственно на данной ОС.

Класс MainPage – класс, который отвечает за извлечение необходимых файлов из хранилища, удаление, а также передачу их службе.

Класс StateObject – вспомогательный класс, который отвечает за формирование и прием пакетов данных при передаче и получении файлов.

3.2. Поля, методы и свойства

Методы и поля, использованные в проекте Cloud Service, WebService и Mobile Server

В таблицах приведено описание(назначение) методов и полей следующих классов:

Таблица 3.1 – Методы HomeController

Название	Описание
Методы CostKeeping	
public ActionResult Index	Запрос GET на рендеринг главной web-страницы и возврат ее в виде ответа пользователю
public ActionResult UserFiles	Запрос GET на рендеринг web-страницы с личными файлами и папками, а также возврат ее в виде ответа пользователю

Таблица 3.2 – Поля, методы и свойства AccountController

Название	Описание
Поля AccountController	
private ApplicationSignInManager _signInManager	Объект класса авторизованных пользователей
private ApplicationUserManager _userManager	Объекты класса зарегистрированных пользователей
Методы AccountController	
public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)	Конструктор, инициализирующий объекты классов ApplicationUserManager и ApplicationSignInManager
public ActionResult Login(string returnUrl)	Метод, вызываемый методом GET. Служит для вывода частичного представления логирования
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)	Асинхронный метод логирования, вызываемый запросом POST. Служит для получения модели данных LoginViewModel
public ActionResult Register()	Метод, вызываемый запросом GET. Служит для вывода частичного представления регистрации
public async Task<ActionResult> Register(RegisterViewModel model)	Асинхронный метод регистрации, вызываемый методом POST. Служит для получения модели данных RegisterViewModel

public Task CreateFolderAsync(string name)	Асинхронный метод для создания личной папки пользователя на сервере, вызывающийся после успешного выполнения метода Register
Свойства AccountController	
public ApplicationSignInManager SignInManager	Свойство, получающее или возвращающее объект класса ApplicationSignInManager
public ApplicationUserManager UserManager	Свойство, получающее или возвращающее объект класса ApplicationUserManager

Таблица 3.3 – Методы FileController

Название	Описание
Методы FileController	
public ViewResult DisplayFiles(string userName)	Метод, выводящий частичное представление отображения файлов и папок
public ActionResult UploadFiles(HttpPostedFileBase[] files)	Метод, вызывающийся запросом POST, получающий загружаемые пользователем файлы и папки на сервер
private List<FileInfo> TakeAndFiltFiles(List<FileInfo> files, int itemPerRow)	Метод, сортирующий файлы при выводе частичного представления
private List<DirectoryInfo> TakeFolders(List<DirectoryInfo> files, int itemPerRow)	Метод, сортирующий папки при выводе частичного представления

Таблица 3.4 – свойства FilesNFolders

Название	Описание
Своства FilesNFolders	
public List<FileInfo[]> Files	Содержит все пользовательские файлы
public List<FileInfo[]> UnknwnFiles	Содержит все пользовательские файлы с неизвестным расширением
public List<DirectoryInfo[]> Directories	Содержит все пользовательские папки

Таблица 3.5 – Методы ManageController

Название	Описание
Методы ManageController	
public async Task<ActionResult> Index(ManageMessageId? message)	Асинхронный метод, вызываемый запросом GET. Изменяет все пользовательские учетные данные и уведомляет об этом пользователя
public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)	Асинхронный метод, вызываемый запросом POST. Удаляет старый логин и устанавливает новый
public ActionResult AddPhoneNumber()	Метод, вызываемый запросом GET. Выводит частичное представление для привязки мобильного номера к аккаунта
public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)	Асинхронный метод, вызываемый запросом POST. Привязывает номер мобильного телефона к аккаунту

Таблица 3.6 – Поля и методы MobileServerController

Название	Описание
Методы MobileServerController	
public string Upload(IEnumerable<HttpPostedFileBase> uploads)	Передаёт загруженные пользователем файлы сервису
public string Download(IEnumerable<HttpPostedFileBase> uploads)	Получает пользовательские файлы от сервиса

3.3. Разработка алгоритмов

В соответствии с разработанной диаграммой классов необходимо спроектировать такие классы и методы:

- 1) HomeController;
- 2) AccountController;
- 3) ManageController;
- 4) FileController;
- 5) MobileController;
- 6) PushFiles;
- 7) GetFiles;
- 8) StartListening;
- 9) И др.

Ниже приведены несколько алгоритмов работы некоторых методов из списка:

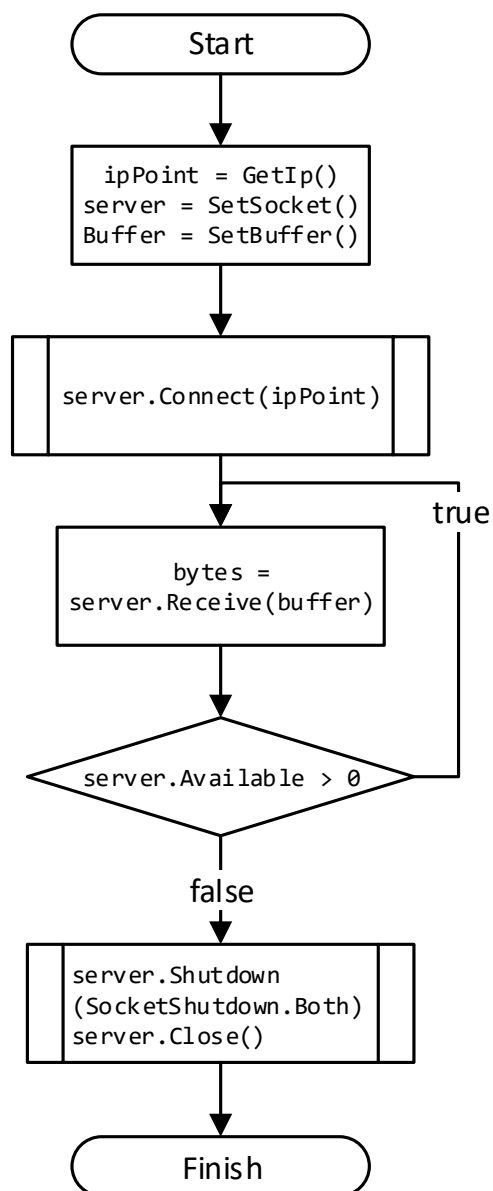


Рисунок 3.4 – Схема принципа работы метода PushFiles, производящий передачу файлов службе, до тех пор, пока есть данные и сервис работоспособен.

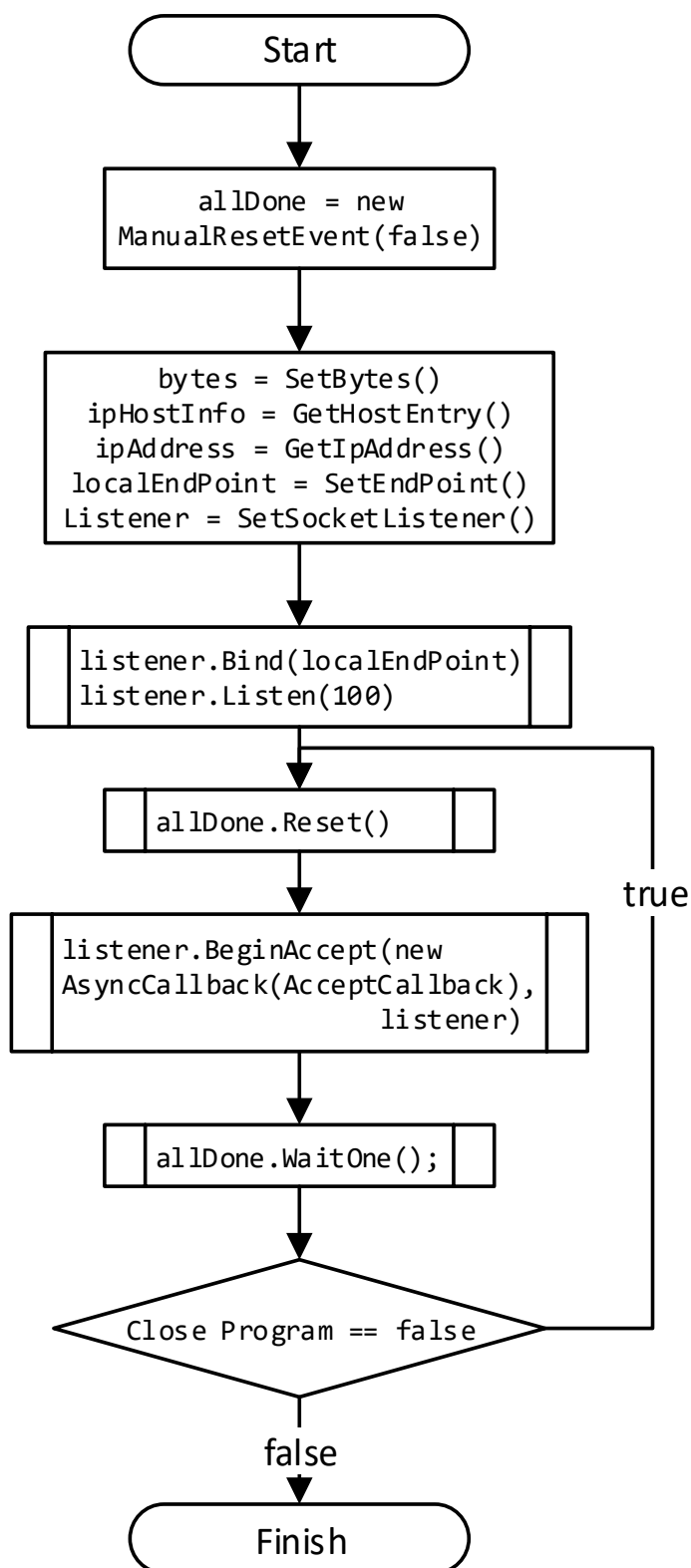


Рисунок 3.5 – Схема принципа работы метода `StartListening` на серверной стороне, который «прослушивает» входящие подключения от службы.

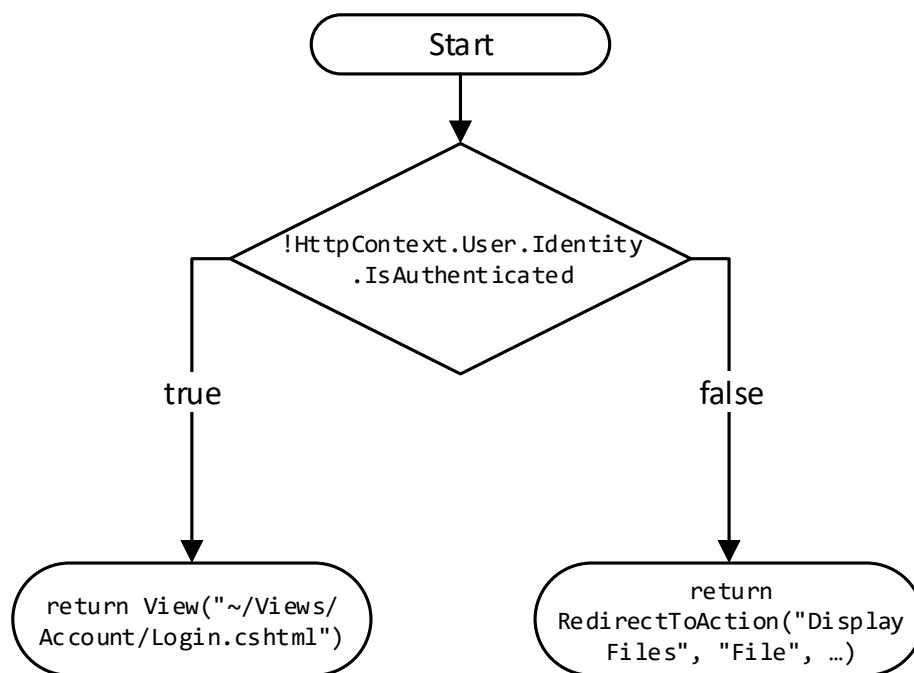


Рисунок 3.6 – Схема принципа работы метода `Index` для контроллера `HomeController`, проверяющий, авторизован ли пользователь и, в соответствии с полученными результатами, перенаправляет пользователя на определенную страницу.

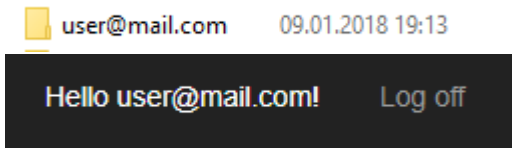
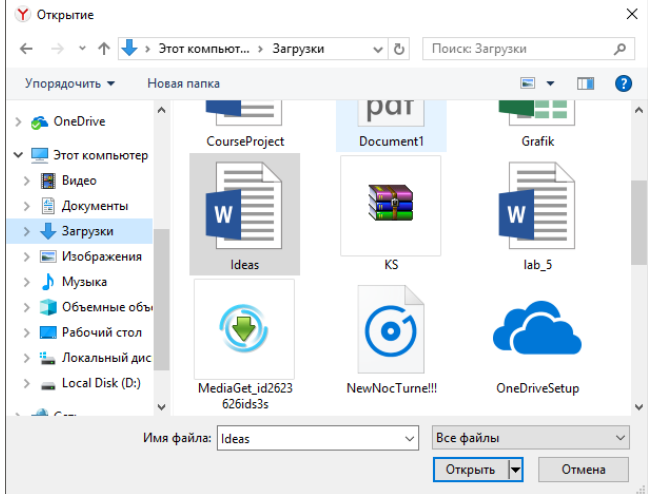
4. ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

Верификация проводилась путем проверки реализации требований технического задания:

1. Регистрация, авторизация пользователя.
2. Загрузка файла на сервер.
3. Создание, удаление, переименование файла.

Результаты проделанной работы, которые подтверждают, что программа правильно выполняет задачи приведены ниже:

Таблица 4.1 – Набор тестов

Название теста	Результат теста	Скриншот с результатом
Регистрация пользователя Гостя	Пользователь успешно зарегистрирован	
Загрузка файла на сервер	Файл успешно загружен	

Переименование папки	Папка успешно переименована	
Переименование файла	Файл успешно переименован	
Удаление папки	Папка успешно удалена	

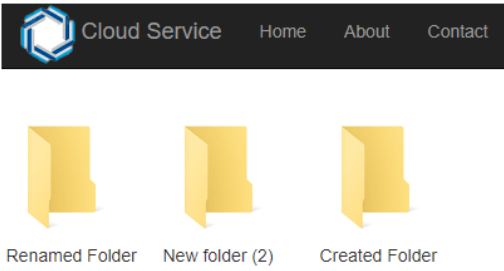
Создание папки	Папка успешно создана	
----------------	-----------------------	--

Таблица 4.2 – Набор Unit-тестов

Тестируемый метод	Описание метода	Скриншот с результатом
CreatingUser StorageFolder_ CreatedUser StorageFolder	Создание папки под хранение личных файлов для только что зарегистрировавшего пользователя.	
DownloadigFiles_ DownloadedFiles	Загрузка файлов на сервер.	
UploadingFiles_ UploadedFiles	Скачивание файлов из сервера.	
CreatingFolder_ CreatedFolder	Создание простой папки.	
RomovingFile_ RemovedFile	Удаление файла.	
IsEqualHashesOfL oginAndPassword _CheckHashesOfL oginAndPassword	Одностороннее шифрование пользовательских учетных данных (логин и пароль) hash-функцией.	
IsEncryptedDataOf File_CheckDataOf File	Шифрование содержимого файла алгоритмом RSA.	

ВЫВОДЫ

При выполнении данной курсовой работы был реализован проект, позволяющий загружать, выгружать файлы, папки, также организовывать рабочее пространство с данными. Разработанная программа верифицирована и протестирована.

Данное приложение выполняет:

- 1) регистрацию, авторизацию пользователей;
- 2) создание, удаление, изменение файлов, папок;
- 3) загрузку файлов, папок в сервера;
- 4) выгрузку файлов, папок из сервера;
- 5) удаление аккаунта;
- 6) настраивание учетных данных;

Для дальнейшей модернизации проекта можно:

- 1) улучшить оптимизацию приложения;
- 2) добавить учетную запись для администратора;
- 3) добавить функцию «поделиться» для файлов и папок;
- 4) добавить чат для двух или группы пользователей;
- 5) добавить шифрование файлов;

СПИСОК ЛИТЕРАТУРЫ

1. WIKIPEDIA [Электр. ресурс] – Режим доступа:
https://ru.wikipedia.org/wiki/Закон_Мура
2. ASP.NET MVC FRAMEWORK – WIKIPEDIA [Электр. ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ASP.NET_MVC_Framework
3. ASP.NET MVC 5 – METANIT.COM [Электр. ресурс] – Режим доступа:
<https://metanit.com/sharp/mvc5/>
4. XAMARIN [Электр. ресурс] – Режим доступа:
<https://www.xamarin.com/platform>
5. PROFESSORWEB [Электр. ресурс] – Режим доступа:
<https://professorweb.ru/>
6. WIKIPEDIA [Электр. ресурс] – Режим доступа:
<https://ru.wikipedia.org/wiki/Model-View-Controller>

ПРИЛОЖЕНИЕ А Техническое задание

1 Введение

1.1 Наименование программы

Разрабатываемый комплекс программ называется «Сервис облачного хранилища файлов».

1.2 Краткая характеристика области применения

Этот сервис предназначен для хранения, редактирования файлов на удаленных серверах, а также для их обмена. Он организован по «облачному» принципу.

2 Основание для разработки

2.1 Основание для проведения разработки

Основание для разработки Программы - задание на курсовой проект кафедры "Компьютерных систем, сетей и кибербезопасности" Национального аэрокосмического университета им. Н.Е. Жуковского "ХАИ" по дисциплине «Проектирование МПС».

2.2 Наименование и условное обозначение разработки

Наименование темы разработки – «Сервис облачного хранения файлов».

Условное обозначение темы разработки (шифр темы) – «СОХФ-01».

3 Назначение разработки

3.1 Функциональное назначение разработки

Функциональным назначением сохранение, загрузка, выгрузка файла/файлов разных форматов, а также предоставление возможности редактировать их (файлы форматов docx, xlsx, pptx, txt).

3.2 Эксплуатационное назначение

Сервис предназначен для хранения файлов на удаленных серверах, которые неизвестны пользователю и представляются единым сервером, с которым и осуществляется работа. Доступ к данным происходит за счет авторизации пользователя посредством ранее созданного аккаунта. Авторизованное лицо так же может редактировать файлы форматов docx, xlsx, pptx, txt и создавать ссылку на один или группу файлов, с выбранным режимом доступа, и делиться ею для с доверенными лицами. Безопасность личных данных учетной записи возложена на сервис, который будет шифровать их и при извлечении дешифровать.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Общие сведения, назначения и состав сервиса СОХФ-01

СОХФ-01 – это программный комплекс средств, предназначенных для организации облачного хранилища файлов всех форматов и размерами, не свыше 5 Гб. Данный сервис будет доступен только авторизованным пользователям, у которых есть своя учетная запись с личными файлами. Не авторизованным пользователям следует пройти регистрацию, чтобы получить доступ к хранилищу, выделенного сервисом. Он позволяет хранить абсолютно любые файлы, редактировать файлы форматов docx, xlsx, pptx; делиться файлом или группой файлов при помощи генерируемой ссылкой, которая неявно включает параметры режима доступа к ним (режима чтения и режим редактирования).

СОХФ-01 состоит из главного сервера, сервиса и множества серверов-хранилищ. В обязанности главного сервера входит:

- Регистрация пользователей;
- Авторизация пользователей;
- Аутентификация пользователей;
- Верификация получаемых данных;
- Вывод графического интерфейса web-приложения;
- Получение данных от клиентской стороны;
- Передача данных сервису;
- Получение данных от сервиса.

Назначением сервиса является передача и получение данных как от web-приложения, так и от устройств-серверов. Причина, по которой этот компонент отделен от других в том, что это повышает надежность и отказоустойчивость, а также защищенность всей системы в целом.

Т.к. концепция «облака» подразумевает использование множество удаленных друг от друга серверов, то в данном случае используются сервера, на которых будут храниться все пользовательские данные, предварительно зашифрованные web-приложением.

4.1.2 Требования к составу выполняемых функций

Сервис должен выполнять следующие функции:

- a) Регистрация, авторизация, аутентификация пользователей в системе;
- b) Деавторизация пользователей;
- c) Валидация данных, получаемых от пользователей;
- d) Наделять пользователей при регистрации в системе ролью «user»;
- e) Удалять аккаунты пользователей, наделенных ролью «admin», пользователем с ролью «senioradmin», а также аккаунты, имеющих роль «user»;
- f) Удалять аккаунты пользователей, наделенных ролью «user», пользователем с ролью «admin»;

- g) Загружать и выгружать файлы;
- h) Сохранять пользовательские файлы на работоспособных и в то же время «активных» устройствах-серверах;
- i) Шифровать и дешифровать файлы в соответствии с выбранным пользователем режимом: режим повышенной надежности (алгоритм RSA) и режим быстрой работы с файлами (алгоритм DES).
- j) Ограничивать размер загружаемых файлов. Если пользователь будет пытаться загрузить файл, размером свыше 5 ГБ, отклонить запрос;
- k) устанавливать или разрывать соединение с сервисом из приложения на устройстве-сервере;
- l) Отображать краткую статистику по обмену данными на устройстве-сервере;
- m) В случае возникновения программной ошибки на устройстве-сервере, продолжить функционирование приложения и отобразить сведения о возникшей ошибке;
- n) Искать, сортировать, группировать, создавать файлы в web-приложении;
- o) Предоставлять возможность просматривать или редактировать содержимое файлов форматов docx, xlsx, pptx, txt, pdf;
- p) Изменять параметры созданного аккаунта: почта, номер мобильного телефона, двухфакторная авторизация, имя, возраст, аватар (графическое представление пользователя в системе);
- q) Создавать учетную запись после успешной регистрации пользователя;
- r) Удалять учетную запись.

4.1.3 Требования к организации входных данных

Входные данные при регистрации: электронная почта и номер мобильного телефона должны быть существующими.

Входные данные при авторизации: учетные данные зарегистрированного пользователя.

Входные данные при загрузке файлов: размер файла должен быть не свыше 5 ГБ.

Входные данные при редактировании содержимого файла: файл должен представляться форматом или docx, или xlsx, или pptx, или txt.

Входные данные при просмотре содержимого файла: файл должен представляться форматом или docx, или xlsx, или pptx, или txt, или pdf.

4.1.4 Требования к организации выходных данных

Выходные данные при регистрации: оповещение об успешной или неуспешной регистрации.

Выходные данные при авторизации: переход на страницу личного кабинета, либо оповещение о возникшей ошибке.

Выходные данные при загрузке файлов: оповещение об успешно проведенной операции либо о возникшей ошибке.

Выходные данные для отредактированного (но изменения которого не были зафиксированы пользователем) файла до наступления момента автосохранения: неизменившееся содержимое файла.

Выходные данные для отредактированного (но изменения которого не были зафиксированы пользователем) файла после наступления момента автосохранения: сохраненный файл, с изменившимся или неизменившимся содержимым.

Выходные данные для отредактированного (изменения которого были зафиксированы пользователем) файла до наступления момента автосохранения: сохраненный файл, с изменившимся или неизменившимся содержимым.

Выходные данные для отредактированного (изменения которого были зафиксированы пользователем) файла после наступления момента автосохранения: сохраненный файл, с изменившимся или неизменившимся содержимым.

4.1.5 Требования к временным характеристикам

Максимально допустимое время загрузки файла/файлов на сервер за один сеанс: 7 часов.

Максимально допустимое время скачивания файла/файлов за один сеанс: 7 часов.

4.1.6 Требования к безопасности

Сервис должен гарантировать безопасность учетных данных пользователя (логин, пароль, адрес электронной почты, номер мобильного телефона), а также личных данных.

4.2 Требования к надежности

4.2.1 Перечень мероприятий по обеспечению надежного (устойчивого) функционирования сервиса

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий:

- a) организация бесперебойного питания технических средств;
- b) использование лицензионного программного обеспечения;
- c) допустимый диапазон температуры помещения, в котором эксплуатируются устройства-сервера: от +15 до +35 °С;
- d) допустимый диапазон температуры помещения, в котором хранятся устройства-сервера: от +10 до +45 °С;

е) ограничение доступа к помещению, в котором находятся устройства-сервера;

ф) адекватное, рациональное и исключительное наделение ролью «admin» учетных записей сотрудников.

4.2.2 Требования к обеспечению надежного функционирования сервиса

Во процессе работы Программы возможны отказы по ее вине, других программных средств или аппаратного обеспечения.

Возможные отказы в работе Программы по ее вине не должны приводить к потере данных в файлах, используемых программой или пользователем в момент отказа.

Безопасность данных в случае аварийных ситуаций, которые не вызваны Программой, должна обеспечиваться средствами ОС.

Не допускается требование перезагрузки ОС после отказа Программы.

4.2.3 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), нефатальным сбоем (не крахом) операционной системы, не должно превышать столько-то минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

4.2.4 Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3 Условия эксплуатации

4.3.1 Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

4.3.2 Требования к видам обслуживания

См. Требования к обеспечению надежного (устойчивого) функционирования программы.

4.3.3 Требования к численности и квалификации персонала

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее двух штатных единиц - системный администратор и конечный пользователь программы (оператор).

Системный администратор должен иметь высшее профильное образование и сертификаты компании-производителя операционной системы. Задачи, выполняемые системным администратором:

- а) поддержание работоспособности технических средств;
- б) установка (инсталляция) и поддержание работоспособности системных программных средств - операционной системы;
- с) установка (инсталляция) программы.

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

4.4 Требования к составу и параметрам технических средств

К составу технических средств должны входить:

- Сервер либо стационарный компьютер:
 - а) Процессор Intel Core i5 или Intel Xeon E3 1240v3 и выше;
 - б) Память 8 Гб DDR3 и больше;
 - с) Дисковая система 240 Гб SSD или HDD и больше;
 - д) ОС Windows 8 или Windows Server 2012 и выше;
 - е) IIS 8.0 и выше.
- Устройства под управлением следующих ОС на выбор: IOS, Android, Windows Phone.

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к информационным структурам и методам решений

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

4.5.2 Требования к исходным кодам и языкам программирования

Для разработки web-приложения должны использоваться следующие языки программирования: C# (7.0 и выше) – разработка инфраструктуры серверной части и JavaScript – разработка «дружелюбного» графического интерфейса приложения, в среде разработки Visual Studio 2016 и выше.

Для разработки сервиса также должен использоваться C# 7.0 и выше, в среде разработки Visual Studio 2016 и выше.

Для разработки серверного приложения должен использоваться должен использоваться C# версии не ниже 4.0, в среде разработки Xamarin Studio.

4.5.3 Требования к программным средствам, используемым сервисом

Web-приложение должно работать под управлением ОС Windows 8 и выше или Windows Server 2012 и выше, а также допускаются соответствующие пакеты обновлений.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы. Допускается применение соответствующего пакета обновления.

Для запуска web-приложения должен быть предустановлен веб-сервер IIS Express или IIS 5.0, один из которых следует корректно сконфигурирован в соответствии с расположением папки проекта web-приложения и ОС.

Параметры ОС настроены таким образом, чтобы исполнение программы было возможным. Для этого как минимум должны быть правильно настроены переменные окружения среды выполнения web-приложения и корректно установлены права пользователя.

4.6 Специальные требования

Специальные требования не предъявляются.

5 Требования к программной документации

5.1 Предварительный состав программной документации

В результате разработки сервиса должна быть представлена следующая программная документация:

- a) техническое задание;
- b) схемы алгоритмов;
- c) тексты программ;
- d) пояснительная записка;
- e) руководство оператора;

Кроме программного обеспечения на диске обязательно должна находиться пояснительная записка в электронном виде, содержащая весь комплект документации, предусмотренный в п. 5.1 настоящего технического задания.

ПРИЛОЖЕНИЕ Б Руководство клиента

1. Условия работы приложений Web-приложение

В состав технических средств должно входить любое десктопное или мобильное устройство, с наличием любого браузера и доступом в Интернет.

2. Использование

Чтобы иметь возможность пользоваться сервисом, необходимо пройти этап регистрации или авторизации. Если пользователь не имеет ранее созданную учетную запись, ему следует зарегистрироваться, перейдя на соответствующую страницу регистрации, кликнув по ссылке, скриншот которой приведен в П.Б.1.

Для регистрации следует заполнить 3 поля: Email, Password и Confirm Password.

Email – поле, в которое надо ввести существующий адрес электронной почты.

Password – поле, в которое надо ввести придуманный пароль, необходимый также и при дальнейших авторизациях.

Confirm Password – поле, в которое надо ввести пароль, объявленный в «Password». Это необходимо, чтобы убедиться в том, что случайно допущенная опечатка пользователем в пароле не будет сохранена системой и избавит от перерегистрации или прохождения процесса восстановления пароля (ссылка на эту функцию показана на скриншоте П.Б.2. Появляется после неудачно пройденного этапа авторизации).

Для пользователей, имеющих ранее созданную учетную запись, следует заполнить поля Email и Password. В П.Б.3 приведен скриншот со страницей авторизации.

На «домашней» странице пользователю представляется весь основной функционал по манипуляции личными файлами и папками (П.Б.4). Чтобы загрузить файл или группу файлов, следует нажать на кнопку «Upload», которая выглядит, как на скриншоте в П.Б.5. Если же надо скачать файлы, необходимо выбрать целевые, т.е. отметить их флажками, как показано на скриншоте П.Б.6, а затем нажать на кнопку «Download» (П.Б.7).

Чтобы создать, удалить или переименовать файл/папку, необходимо кликнуть правой кнопкой мыши по выбранному элементу, после чего появится небольшое контекстное меню с перечнем этих функций. Скриншот контекстного меню показан в П.Б.8.

Перейти на страницу управления личным аккаунтом можно кликнув по ссылке, которая является логином (пример такой ссылки показан на скриншоте П.Б.9).

На странице управления личным аккаунтом отображены 4 ссылки, которые активируют ту или иную функцию, в зависимости от того, чем пользователь хочет воспользоваться: изменить пароль, установить другой

способ авторизации (через внешние сервисы), привязать номер мобильного телефона или же включить двухфакторную авторизацию. Скриншот с функциями показан в П.Б.10.

Для того, чтобы деавторизоваться, предназначена ссылка «Log off» (П.Б.11).

ПРИЛОЖЕНИЕ В Руководство администратора

1. Сервер под web-приложение:

В состав технических средств должно входить любое десктопное или лэптопное устройство с минимальными техническими характеристиками (CPU – 2.4Ghz, RAM – 4 GB, HDD – 1 GB).

2. Серверные устройства под хранение файлов:

В состав технических средств должно входить любое мобильное устройство с минимальными техническими характеристиками (CPU – 1.4Ghz, RAM – 512 MB, SSD – 1 GB).

3. Установка web-приложения

Для установки приложений выполните следующие шаги:

- a) скачать файл-архив;
- b) распаковать в любой удобной директории;
- c) убедиться в наличии на конечном устройстве платформы .NET Framework с версией 4.5.2 и выше;
- d) убедиться в наличии IIS Express 10.0 и выше;
- e) В корневой директории IIS Express, в папке config, открыть файл applicationhost.config любым инструментом по редактированию текстовых файлов. В начало секции <sites> добавить следующий код:

```
<site name="Cloud Storage" id="1">
  <application path="/" applicationPool="Clr4IntegratedAppPool">
    <virtualDirectory path="/" physicalPath="..." />
  </application>
  <bindings>
    <binding protocol="http" bindingInformation=":57460:localhost"/>
  </bindings>
</site>
```

Вместо троеточия в свойстве physicalPath прописать путь к разархивированному приложению;

- f) Запустить IIS Express.

4. Установка сервиса

- a) скачать файл-архив;
- b) распаковать в любой удобной директории;
- c) убедиться в наличии платформы на конечном устройстве платформы .NET Framework с версией 4.5.2 и выше;
- d) убедиться в наличии IIS Express 10.0 и выше;
- e) убедиться в наличии Visual Studio не ниже 15;
- f) запустить visual studio;
- g) собрать решение и проект;
- h) запустить проект «Service1»

5. Установка приложения для устройств-серверов:

- a) скачать файл-архив;
- b) распаковать в любой удобной директории;
- c) убедиться в наличии платформы на конечном устройстве платформы .NET Framework с версией 4.5.2 и выше;

- d) убедиться в наличии IIS Express 10.0 и выше;
- e) убедиться в наличии Visual Studio 17 с предустановленным набором инструментов Xamarin;
- f) запустить visual studio;
- g) собрать решение и проект;
- h) подсоединить мобильное устройство к ПК по USB-кабелю.
- i) На мобильном устройстве с ОС Android установить в настройках режим «Отладка по USB»
- j) Выбрать проект для мобильного устройства: в случае ОС Android – MobileServer.Android, в случае ОС WindowsPhone 8.1 – MobileServer.WP, в случае ОС IOS – MobileServer.IOS.
- k) Для целевого проекта убедиться в наличии строго необходимых библиотек под определенную версию ОС мобильного устройства, а также убедиться в том, что в настройках запускаемого проекта выбраны эти библиотеки.
- l) Запустить проект.

6. Возможные сообщения администратору при не правильном выполнении каких-то действий в программе

Если в процессе работы программы возникают некоторые ошибки, сбой или приложение просто не может запуститься, и, если вам не удастся разрешить программные конфликты самостоятельно, то опишите подробно проблему в сообщении, а затем отправьте на следующий адрес электронной почты: ivanbesch@outlook.com.

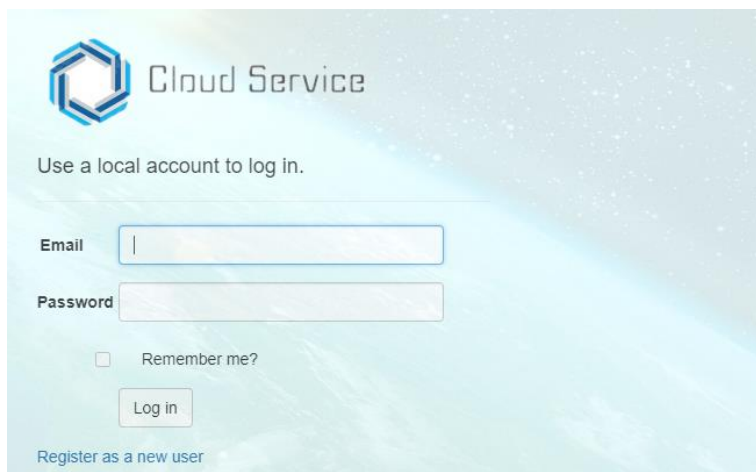
П.Б.1. Ссылка на страницу регистрации.

[Register as a new user](#)

П.Б.2. Ссылка на восстановление пароля.

[Restore password](#)

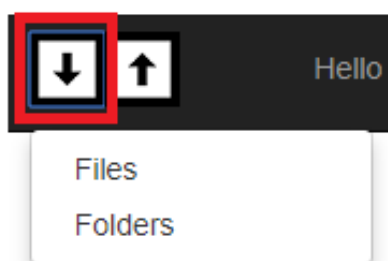
П.Б.3. Страница авторизации.



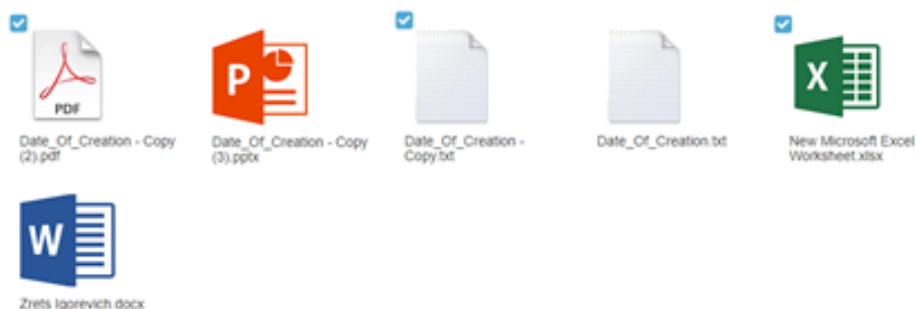
П.Б.4. Панель функций и содержимое хранилища пользователя.



П.Б.5. Кнопка загрузки на сервер файлов/папок.



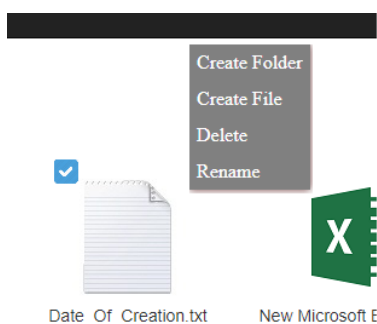
П.Б.6. Файлы, отмеченные флажками.



П.Б.7. Кнопка скачивания.



П.Б.8. Контекстное меню при нажатии правой кнопки мыши.



П.Б.9. Ссылка-логин на учетные данные.

Hello [ivan@mail.com!](#)

П.Б.10. Страница управления учетными данными.

Manage.

Change your account settings

Password: [\[Change your password \]](#)
 External Logins: 0 [\[Manage \]](#)
 Phone Number: None [\[Add \]](#)
 Two-Factor Authentication: Disabled [Enable](#)

П.Б.11. Ссылка на деавторизацию.

[Log off](#)