Форма № Н-6.01

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)

м. Харків – 2018 рік

Зміст

**[ВСТУП](#_Toc516415111)** [3](#_Toc516415111)

[1. ПОСТАНУВАННЯ ЗАВДАННЯ 5](#_Toc516415112)

[1.1. Аналіз предметної області 5](#_Toc516415113)

[2. ПРОЕКТУВАННЯ 14](#_Toc516415115)

[2.1. Вибір архітектури 14](#_Toc516415116)

[2.2. Рівень web-додатку 15](#_Toc516415117)

[2.2.1. Розширюваність 17](#_Toc516415120)

[2.2.2. HTTP и HTML 18](#_Toc516415121)

[2.2.3. Здатність до тестування 20](#_Toc516415122)

[2.3. Рівень сервісу 21](#_Toc516415123)

[2.4. Рівень сховища даних 22](#_Toc516415124)

[2.5. UML – діаграма використання 23](#_Toc516415125)

[2.6. Проектування графічного інтерфейсу: проектування сторінок Web-додатка та вікна android-додатка 28](#_Toc516415126)

[3. РОЗРОБКА 38](#_Toc516415127)

[3.1. Розробка діаграми класів 38](#_Toc516415128)

[3.2. Поля, методи та властивості 43](#_Toc516415129)

[3.3. Розробка алгоритмів 51](#_Toc516415131)

[4. ВЕРИФІКАЦІЯ І ТЕСТУВАННЯ ПРОГРАМИ 55](#_Toc516415132)

[ВИСНОВКИ 59](#_Toc516415133)

[СПИСОК ЛІТЕРАТУРИ 61](#_Toc516415138)

**[ДОДАТОК А Технічне завдання](#_Toc516415145)** [63](#_Toc516415145)

**[ДОДАТОК Б Керівництво клієнта](#_Toc516415146)** [72](#_Toc516415146)

**[ДОДАТОК В Керівництво адміністратора](#_Toc516415147)** [74](#_Toc516415147)

# **ВСТУП**

Пояснювальна записка курсового проекту включає в себе:

– сторінок – 76;

– малюнків – 26;

– таблиць – 31;

– додатків – 3;

– джерел – 12.

Метою даної курсової роботи є розробка комплексу програм “Сервіс хмарного зберігання файлів”.

Завданнями дипломної роботи є:

1. реалізація алгоритму завантаження та вивантаження файлів будь-якого формату і розміру;
2. реалізація хмарної інфраструктури;
3. захист та шифрування даних користувачів;
4. ПОСТАНУВАННЯ ЗАВДАННЯ
   1. Аналіз предметної області

Згідно з законом Муру, кількість транзисторів на кристалі процесору кожні 2 роки росте, і крива, що характеризує дану поведінку, має експоненціальний характер [1].

Комп’ютерні пристрої и комп’ютерне обладнання з кожним днем стає продуктивніше, а також потребує більше об’єму пам’яті, на котрій буде зберігатися гігантська кількість за мірками кінця ХХ і початку ХХІ століття інформації. Ця проблема вирішувалась та досі вирішується різними способами: збільшення об’єму пам’яті на пристрої, використання флеш-накопичувачів, які легко транспортувати, поділення даних між двома або декількома схожими пристроями, здатних передавати інформацію кабельним або радіохвильовим шляхом. Однак цю актуальну проблему призвані легко розрішити сервіси хмарного зберігання даних, без будь-яких істотних фінансових затрат для кінцевого користувача.

Хмарне сховище або «Хмара» – це онлайн-сховище, яке розподіляє усі дані користувача на безлічі віддалених один від одного серверів. Про організацію даного сервісу ніхто не може знати, зокрема спеціалістів, які приймають участь у створенні комплексу ПО для реалізації цього проекту, отже це підвищує надійність усієї системи. Хакерські атаки повинні бути спрямовані на безліч об’єктів, зазвичай розташованих в різних регіонах планети, щоби порушити роботу всього комплексу серверів, після чого втрачаються або потрапляють у руки зловмисників усі особисті дані користувачів. Однак навіть це буде непросто здійснити, особисто, якщо у системі є потужні відповідні механізми захисту та високонадійні, відмовостійкі компоненти.  
 Крім зберігання інформації, дана інфраструктура дозволяє проводити найрізноманітніші операції з даними, маніпуляція якими найчастіше необхідна нам у тих випадках, коли нема можливості завантажити усе необхідне на пристрій, який використовується в даний момент, або яке не має інструментарію, який надає сервіс хмарного зберігання даних: редагування, конвертування, архівування, шифрування і т.д. Тому «хмара» концептуально підтримується усіма видами пристроїв, маючих доступ до всесвітньої мережі.

На сьогодні існує безліч сервісів хмарного зберігання даних. Усі вони мають як переваги, так і недоліки, котрі далі будуть розглядані.

**GoogleDrive**

Один з найпопулярніших сервісів хмарного сховища, перевагою якого є багатофункціональний web-інтерфейс, а також в якому доступні такі операції:

* редагування та створення документів в режимі спільного доступу;
* перегляд мультимедійного контенту;
* розширення майже всіх існуючих надбудов;
* пошук документів;
* управління правами доступу.

На тлі всіх інших подібних сервісів можна виділити наступні переваги і недоліки:

Таблиця 1.1 – Переваги та недоліки сервісу GoogleDrive

|  |  |
| --- | --- |
| **+** | **–** |
| Необмежений простір під зберігання зображення і фотографій. | Виділений вільний простір буде використовуватися іншими суміжними сервісами. |
| Гнучкий web-інтерфейс, що дозволяє налаштовувати робоче середовище під потреби користувача. | Windows Desktop версія програми не має будь-яких переваг. |
| При реєстрації відразу надається 15 ГБ безкоштовного дискового простору. |  |

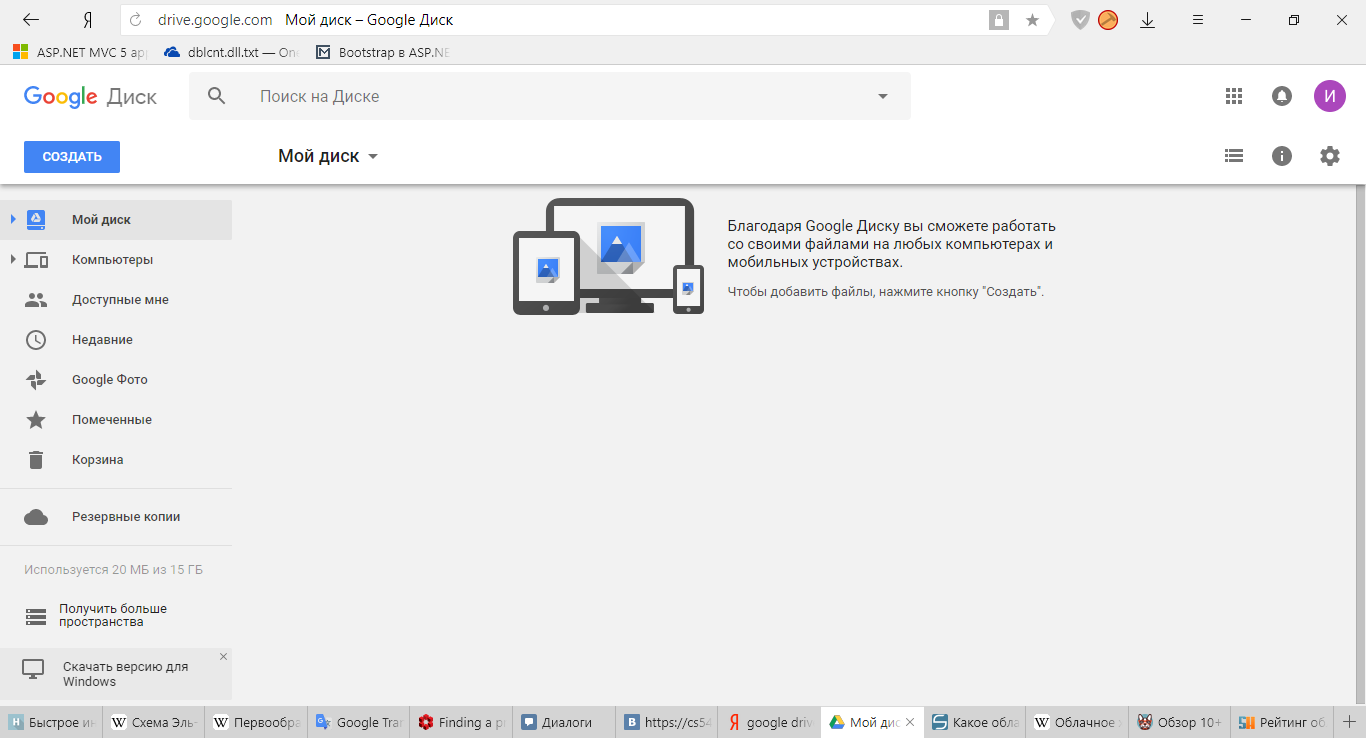


Рисунок 1.1 – Web-інтерфейс сервісу GoogleDrive

**Microsoft OneDrive**

Надійне онлайн-сховище від компанії Microsoft, яке за замовчуванням інтегроване в ОС Windows 10, що дає можливість працювати зі своїми даними на локальному комп'ютері з подальшою автосінхронізацією з «хмарою». Відмінна особливість даного сервісу - інтеграція з пакетом Office 365, що надає ряд переваг при роботі з документами різного типу: Word, Excel, PowerPoint, Visio, Access, Publisher, OneNote і т. д.

Основні характеристики сервісу:

* безкоштовне місце: 5 ГБ;
* максимальний розмір одного файлу: 10 ГБ;
* максимальний обсяг простору: 5 ТБ;
* сумісність з мобільними ОС: підтримується на Android вище 4.0, iOS вище 9.0, Windows Phone 7/8, на Symbian Belle і на MeeGo 1.2.

Переваги та недоліки сервісу:

Таблиця 1.2 – Переваги та недоліки сервісу OneDrive

|  |  |
| --- | --- |
| **+** | **–** |
| Інтеграція з пакетом Office 365. | Відсутність додаткових функцій. |
| Висока швидкість та стабільна робота. | Безкоштовний тариф надає усього 5 ГБ вільного дискового простору. |
| Знижки при покупці дискового простору з Office 365. |  |

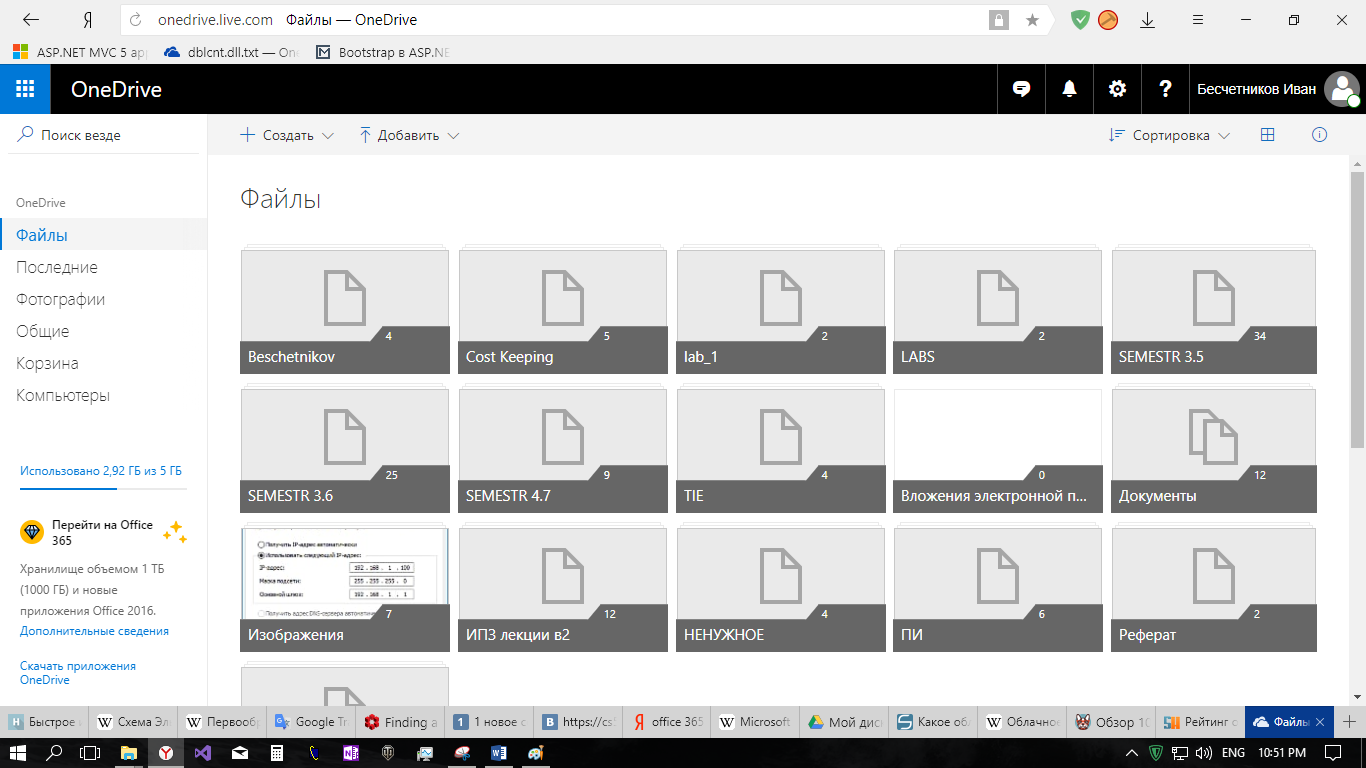


Рисунок 1.2 – Web-інтерфейс сервісу OneDrive

**Dropbox**

Є одним з найперших сервісів «хмарної» архітектури, який за весь свій довгий час існування і розвитку встиг доробити кожен вузол інфраструктури. Dropbox, як і більшість інших популярних сервісів, надає досить функціональний і зручний web-інтерфейс. Також він працює по моделі Freemium, що дозволяє при створенні аккаунта вибрати обсяг вільного дискового простору з подальшим збільшенням, але тільки при внесенні оплати.

Основні характеристики сервісу

* безкоштовне місце: 2 ГБ;
* максимальний розмір одного файлу через web-інтерфейс - 20 ГБ, через Windows Desktop - без обмежень;
* максимальний обсяг простору: 1 ТБ;
* сумісність з мобільними ОС: підтримується на Windows, macOS, Linux; на мобільних ОС Android, iOS, Windows Phone і BlackBerry

Переваги та недоліки сервісу:

Таблиця 1.3 – Переваги та недоліки сервісу Dropbox

|  |  |
| --- | --- |
| **+** | **–** |
| Гнучкий та функціональний web-інтерфейс. | Максимальна кількість допустимих функцій поширюється тільки для бізнес-клієнтів. |
| Зручність у використанні на клієнтських версіях. | Безкоштовний тариф надає всього 2 ГБ вільного дискового простору. |
| 265-бітне шифрування AES та шифрування SSL. |  |
| Найкраща серед аналогів технологія синхронізації. |  |
| Інтеграція з Microsoft Office 365. |  |
| Необмежене відновлення файлів та журналу версій. |  |
| Посилання доступу з паролем та терміном дії. |  |
| Можливість налаштовувати і управляти рівнями доступу. |  |

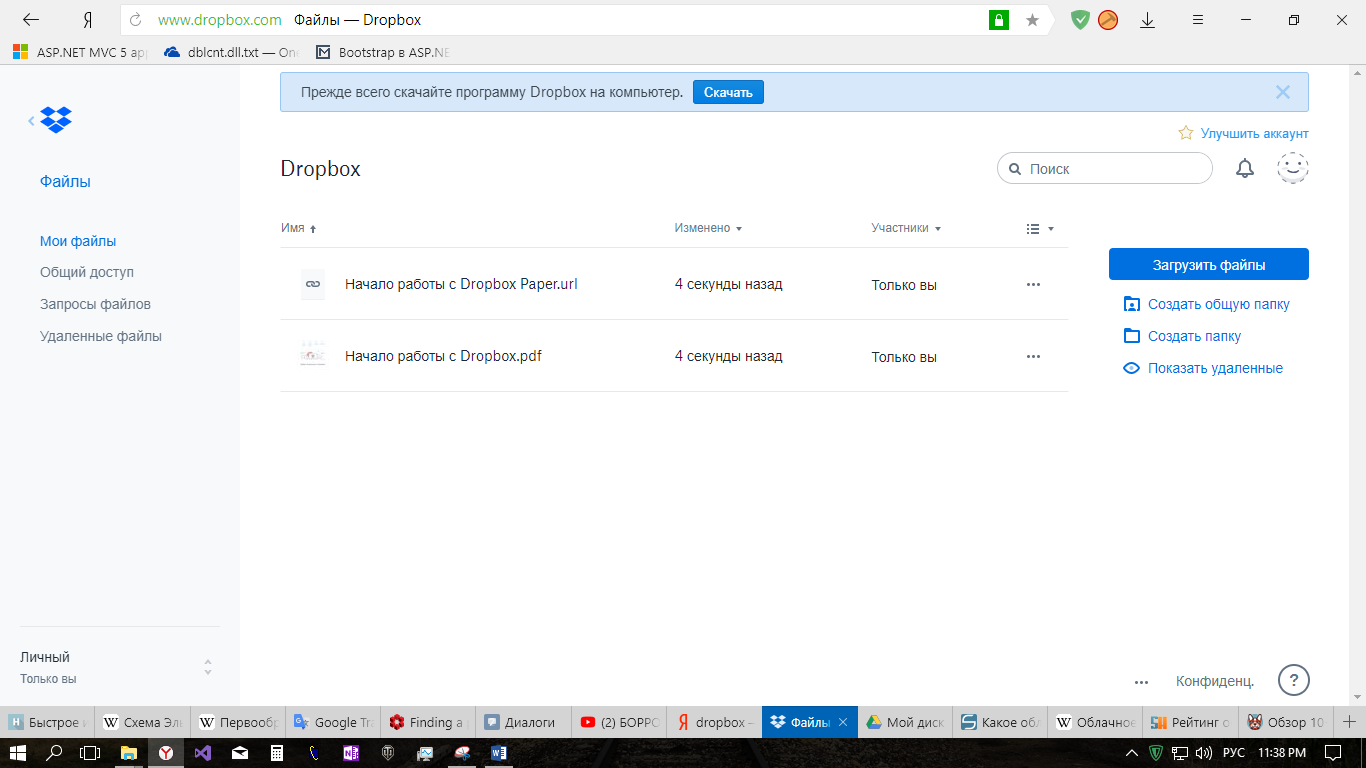


Рисунок 1.3 – Web-інтерфейс сервісу Dropbox

**Mega**

Перспективний хмарний файлообмінник, що надає максимально зручний набір інструментів для користувачів, які не люблять викладати гроші за користування невеликого обсягу пам'яті під зберігання особистих даних. Такий сервіс також надає користувачам можливість ділитися своїми особистими файлами за схемою «friend-to-friend», використовуючи для цих цілей надійний криптографічний алгоритм шифрування AES, причому відкритий ключ відомий тільки тому користувачеві, якому стане доступна посилання на файл обміну.

«Mega» позиціонується як сервіс, що захищає особисті дані користувачів, тому використовується наскрізне шифрування.

Даний web-сервіс існує в трьох варіантах для використання: web-додаток, Windows Desktop додаток і мобільний додаток.

Основні характеристики сервісу:

* безкоштовне місце: 50 ГБ;
* максимальний розмір одного файлу через web-інтерфейс - 10 ГБ, через десктопних програм - без обмежень;
* максимальний обсяг простору: 4 ТБ;
* сумісність з мобільними ОС: підтримується на Windows, macOS, Linux; на мобільних ОС Android, iOS, Windows Phone і BlackBerry.

Переваги та недоліки сервісу:

Таблиця 1.4 – Переваги та недоліки сервісу Dropbox

|  |  |
| --- | --- |
| **+** | **–** |
| Плагіни для браузерів. | Не висока швидкість доступу. |
| Інтегрований чат. | Існують ліміти на трафік. |
| 2048-бітне шифрування AES і шифрування SSL. |  |
| Надається великий обсяг безкоштовного сховища – 50 ГБ. |  |

Порівняльна оцінка деяких параметрів, що існують популярних хмарних сервісів:

Таблиця 1.5 – Зведена таблиця для популярних хмарних сервісів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерії оцінки** | **Google Drive** | **Microsoft OneDrive** | **Mega** | **Dropbox** |
| Максимально допустимий обсяг виділеної пам'яті (ТБ) | 30 | 5 | 4 | 1 |
| Максимальний розмір одного файлу (ГБ) | 5000 | 10 | 10 | 20 |
| Шифрування даних | – | – | + | + |
| Інтеграція з офісним пакетом документів від Microsoft | + | + | – | – |
| Сумісність з мобільними ОС | + | + | + | + |
| Синхронізація даних на ПК | + | + | – | + |
| Редагування та створення документів в режимі спільного доступу | + | + | – | + |
| Перегляд мультимедійного контенту | + | – | – | – |
| Велика розмаїтість додаткових функцій | + | – | + | – |

1. ПРОЕКТУВАННЯ
   1. Вибір архітектури

В ході проектування інфраструктури сервісу була обрана трирівнева архітектура:

* рівень web-додатку;
* рівень сервісу;
* рівень сховища даних.



Рисунок 2.1 – Трирівнева архітектура проекту

* 1. Рівень web-додатку

Рівень web-додатки являє собою додаток, який є головним у всьому проекті і служить для контролю всіх призначених для користувача дій: реєстрація, авторизація, автентифікація, валідація. У разі неуспіху виконання даних процедур, буде викинута системою відповідна помилка, яку користувач буде зобов'язаний виправити, слідуючи отриманої інструкції від браузера або web-додатка.

Для реалізації цього вузла системи був використаний ASP.NET MVC 5 фреймворк для створення web-додатків [2].

ASP.NET MVC 5 - це високонадійна інфраструктура web-додатка, яка покликана забезпечити створення легковагій надійної архітектури проекту, досягається шляхом заздалегідь реалізованих повноцінних відмовостійких і протестованих модулів. Вона також підтримує розширюваність, тестування проектів, тому супровід стає легшим і простішим.

Ця платформа має в наявності відмінний інструментарій, що дозволяє створювати прості, а разом з ним і гнучкі модулі. Вбудовані допоміжні методи HTML генерують ясний та чіткий код розмітки, який відповідає всім сучасним стандартам опису графічних web-інтерфейсів. Високопродуктивна маршрутизація дає можливість створювати зручні, читабельні URL-адреси, завдяки чому програміст вільний придумувати будь-які осмислені імена сторінок, або ж імена, які приховують деяку логіку роботу усього web-додатка.

Проекти, написані на ASP.NET MVC 5 підтримують розширюваність, а це значить, що кожен окремий модуль або модуль, вбудований в систему можна перевірити в модульному тестуванні, а інтеграційні тести будуть відслідковувати стан всієї системи в фоновому режимі роботи.

Дана платформа для розробки web-додатків використовує архітектурний шаблон MVC. До переваг цього шаблону можна віднести наступне:

* користувач взаємодіє з додатком відповідно до природного циклу: користувач робить деяку дію, у відповідь на яку модель змінює свій стан і рендерить певне уявлення, що добре вписується в логіку роботи запитів і відповідей протоколу HTTP.
* Ряд web-додатків потребують поділу модулів всього проекту: HTML-сторінки, компоненти щодо поліпшення web-інтерфейсу, бази даних, бізнес-логіка, сервіси і т. д. Все це відмінно вписується в концепцію MVC [3].

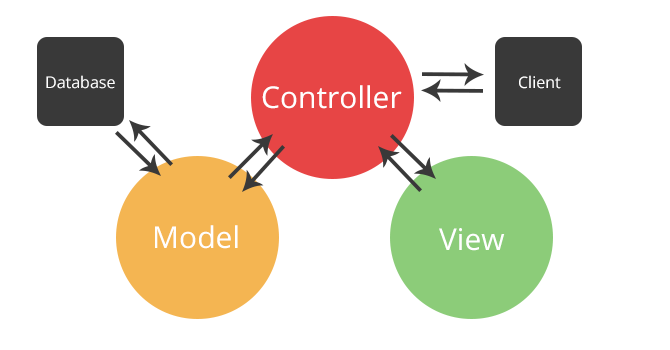


Рисунок 2.2 – Архітектура MVC-проектів

Інфраструктура ASP.NET MVC Framework реалізує цей шаблон, причому з доробленими і поліпшеними деякими сценаріями використання даного патерну.

* + 1. Розширюваність

Інфраструктура MVC Framework побудована таким чином, компоненти якої є взаємно незалежними і представляються у вигляді абстрактних базових класів. Тому існують такі схеми використання вбудованих компонентів:

* повне використання вбудованих компонентів;
* часткове використання вбудованих компонентів з деяким коректуванням, реалізуючи власний клас з успадкуванням від абстрактного класу або інтерфейсу всіх його методів і властивостей;
* повна заміна існуючого компонента власним.



Малюнок 2.3 – Сценарії використання вбудованих компонентів ASP.NET MVC Framework

* + 1. HTTP и HTML

Інфраструктура ASP.NET MVC Framework генерує ясну і просту розмітку, при використанні HTML-хелперів. Однак крім простої генерації HTML-розмітки, вона здатна на побудову елегантних, легко оброблюваних блоків коду-розмітки, які будуть в значно більшій ступені краще рендерити, аніж незліченна безліч складних блоків, якими важко керувати. Генерований код до того ж буде оформлений красивими і «доброзичливими» стилями CSS і плавною анімацією JavaScript.

При обробці запитів фреймворк ASP.NET MVC спирається на систему маршрутизації, яка зіставляє всі вхідні запити з певними в системі маршрутами, які вказують який контролер і метод повинен обробити такий запит. Вбудований маршрут за умовчанням передбачає триланкову структуру: контролер/дію/ параметр.

Завдяки підтримці розширеній маршрутизації ASP.NET, цей потужний компонент зіставлення URL-адрес дозволяє створювати додатки з зрозумілими URL-адресами, які можна використовувати в пошуку. URL-адреси не повинні містити розширення імен файлів і призначені для підтримки шаблонів іменування URL-адрес, які забезпечують адресацію, оптимізовану для пошукових систем (SEO) і для передачі репрезентативного стану (REST)

З іншого боку, якщо виникає потреба у створенні більш складних елементів управління, таких як DatePicker, Clock, CascadeMenu і т. д., то можна скористатися вбудованими бібліотеками, які значно прискорюють і спрощують процес розробки цих елементів. Наприклад, бібліотеки jQuery і BootStrap CSS вже в останніх версіях середовищ розробок, таких як Visual Studio і Code Studio, є, і вважаються вбудованими компонентами, ефективність яких оскаржувати ніхто не стане.

Так само, на відміну від сторінок WEB.FORMS сторінки ASP.NET MVC Framework значно менше за розміром, навіть незважаючи на те, що швидкість з'єднання з web-сайтами зросла в рази, проте це підвищує комфорт використання для кінцевого користувача і прискорює процес запуску самого web-додатки. Це досягається за рахунок відсутності View State-даних сторінки, з якими в основному працює платформа WEB.FORMS.

ASP.NET MVC тісно пов'язана з HTTP-протоколом [4]. Це дозволяє контролювати запити і відповіді, що виникають між сервером та клієнтом. Завдяки цьому, можна точно коригувати кожен момент роботи двох сторін. А при використанні AJAX-запитом не будуть потрібні ніякі автоматичні відправки зворотного виклику, тому що вона проста і не використовує стан клієнтської сторони [5].



Рисунок 2.4 – Бібліотеки, вбудовані за замовчуванням в ASP.NET MVC 5

* + 1. Здатність до тестування

Поділ відповідальності між усіма частинами web-додатки – те, що дозволяє даний фреймворк, роблячи супровід і тестування окремих частин всієї програми набагато простіше і зручніше. Однак на цьому можливості ASP.NET MVC Framework платформи не обмежені. Для кожного модуля компонентно-орієнтованого проекту всієї системи, вона забезпечує структурованість, яка так необхідна при використанні модульного тестування [6].

Крім модульного тестування, існують також засоби, покликані зімітувати як дії користувача, так і взаємодію з ним в графічному інтерфейсі додатка, обробка яких автоматично здійснюється всією інфраструктурою, і немає більше необхідності стежити за змінами структури HTML-розмітки, або ж відстежувати генеруються HTML-елементи і ідентифікатори CSS.



Малюнок 2.5 – Інтегровані інструменти модульного тестування

* 1. Рівень сервісу

Даний рівень є інтерфейсом з обміну даними між web-додатком і серверними додатками. Його робота здійснюється незалежно від роботи web-додатка, проте рівень web-додатка може отримувати деяку інформацію про процес транзакції по вилученню або завантаженню файлів, яку сервіс «витягне» або обробить.

На цьому рівні також оперуються один або група файлів, які слід зберегти або витягти з пам'яті пристрою. Під час вилучення дані передаються рівню, з яким він безпосередньо пов'язаний - рівень сервісу. У разі виникнення помилок, додаток цього рівня має повідомити додатком сервісу про це, а також її назву. Тоді рівень сервісу в обов'язковому порядку зробить спробу щодо усунення помилки, якщо, звичайно, вона не потребує безпосереднього втручання спеціалістів. Прикладами таких помилок можуть виступати:

* втрата пакетів даних при передачі або отриманні даних;
* тимчасова недоступність сервера-сховища;
* відсутність вільної пам'яті;

Додатковими обов’язками цього рівня – шифрування та дешифрування бажаних користувачем даних, що забезпечує зменшення навантаження на систему загалом та її збільшення продуктивності.

* 1. Рівень сховища даних

Даний рівень представляє з себе безліч ftp-серверів, встановлених на пристроях, здатних зберігати, видаляти, отримувати та передавати файл або файлі. Це може бути будь-який ПК, будь-яка робоча станція або гаджет, який працює під управлінням операційної системи Android з 17 рівнем API [7].

На додаток до програми, що забезпечує обмін даними по ftp-протоколу, на пристрій встановлюється додаток, що відстежує зміну файлової системи, тому що ftp-сервер не має додаткових інструментів, здатних відповідати на запит від сервісу про кількість вільної пам'яті, в іншому випадку система викине відповідну помилку, і програма екстрено завершить свою роботу

* 1. UML – діаграма використання

Функції, які реалізовані в проекті, можна уявити за допомогою діаграми використання.



Малюнок 2.7 – UML-діаграма використання. Взаємодія користувача з інтерфейсом програми і її функціями.

Дійова особа "Гість": даний актор являє собою користувача, який не авторизований або не має зареєстрованого облікового запису в цьому сервісі.

Дійова особа "Авторизований користувач»: даний актор являє собою користувача, який увійшов в систему, пройшовши етап авторизації або реєстрації.

Дійова особа "Адміністратор»: даний актор являє собою користувача, уповноваженого адмініструванням облікових записів інших користувачів

Таблиця 2.1 – Прецедент «Зареєструватися»

|  |  |
| --- | --- |
| Назва прецеденту | «Зареєструватися» |
| Опис | Даний прецедент дозволяє реєструватися на даний сервіс. |

Таблиця 2.2 – Прецедент «Авторизуватись»

|  |  |
| --- | --- |
| Назва прецеденту | «Авторизуватись» |
| Опис | Даний прецедент дозволяє авторизуватись. |

Таблиця 2.3 – Прецедент «Відновити пароль»

|  |  |
| --- | --- |
| Назва прецеденту | «Відновити пароль» |
| Опис | Даний прецедент дозволяє відновити доступ до аккаунту, в разі якщо користувач забув пароль. |

Таблиця 2.4 – Прецедент «Видалити акаунти користувачів»

|  |  |
| --- | --- |
| Назва прецеденту | «Видалити акаунти користувачів» |
| Опис | Даний прецедент дозволяє на свій розсуд видаляти адміністратору акаунти користувачів. |

Таблиця 2.5 – Прецедент «Змінити облікові дані користувача»

|  |  |
| --- | --- |
| Назва прецеденту | «Змінити облікові дані користувача» |
| Опис | Даний прецедент дозволяє адміністратору змінювати облікові дані користувача, такі як логін, пароль, номер мобільного телефону та інше, якщо користувачеві не вдається увійти в свій аккаунт. |

Таблиця 2.6 – Прецедент «Видалити аккаунт»

|  |  |
| --- | --- |
| Назва прецеденту | «Видалити аккаунт» |
| Опис | Даний прецедент дозволяють користувачеві при необхідності видалити власний аккаунт. |

Таблиця 2.7 – Прецедент «Вийти»

|  |  |
| --- | --- |
| Назва прецеденту | «Вийти» |
| Опис | Даний прецедент дозволяє вийти з облікового запису. |

Таблиця 2.8 – Прецедент «Видалити файл або папку»

|  |  |
| --- | --- |
| Назва прецеденту | «Видалити файл або папку» |
| Опис | Даний прецедент дозволяє користувачеві видаляти існуючу папку або файл. |

Таблиця 2.9 – Прецедент «Створити файл або папку»

|  |  |
| --- | --- |
| Назва прецеденту | «Створити файл або папку» |
| Опис | Даний прецедент дозволяє користувачеві створити папку або файл. |

Таблиця 2.10 – Прецедент «Змінити облікові дані»

|  |  |
| --- | --- |
| Назва прецеденту | «Змінити облікові дані» |
| Опис | Даний прецедент дозволяє користувачеві змінювати деякі облікові дані: логін, пароль, ПІБ, аватар, номер мобільного телефону і т. д. |

Таблиця 2.11 – Прецедент «Переглядати вміст файлів форматів docx, xlsx, pptx, txt»

|  |  |
| --- | --- |
| Назва прецеденту | «Переглядати вміст файлів форматів docx, xlsx, pptx, txt» |
| Опис | Даний прецедент дозволяє користувачеві переглядати вміст файлів з такими розширеннями. |

Таблиця 2.12 – Прецедент «Завантажити файл/файли»

|  |  |
| --- | --- |
| Назва прецеденту | «Завантажити файл/файли» |
| Опис | Даний прецедент дозволяє користувачеві завантажувати файли або групу файл на сервіс з пристрою, що використовується в даний момент часу. |

Таблиця 2.13 – Прецедент «Завантажити файл/файли з сервера»

|  |  |
| --- | --- |
| Назва прецеденту | «Завантажити файл/файли з сервера» |
| Опис | Даний прецедент дозволяє скачувати файл або групу файлів з сервісу на пристрій, який використовується в даний момент. |

* 1. Проектування графічного інтерфейсу: проектування сторінок Web-додатка та вікна android-додатка

Інтерфейс був реалізований з використанням HTML, CSS та движка представлень Razor. Велика частина CSS-стилів, які повсюдно використовувалися у всіх представленнях в проекті, – відносяться до CSS-фреймворку Bootstrap 4.

Дана бібліотека стилів має в наявності велику кількість HTML- CSS-шаблонів, завдяки яким процес створення web-інтерфейсу проходить простіше і швидше. Також вона надає ряд готових, протестованих скриптів, які роблять сайт інтуїтивно зрозумілим і зручним.

 \_ViewStart - представлення, що візуалізується під час запуску програми. Воно не містить візуальних елементів, а встановлює представлення шаблону (\_Layout). \_Layout визначає шаблон сторінки авторизації, реєстрації, управління обліковим записом, виробничі приміщення з індивідуальними файлами і папками користувача і т. д.

 Всього в додатку є близько 30 представлень, 3 з яких є шаблонами: \_LayoutEntrance, \_LayoutMaster і \_LayoutAdmin.

\_LayoutEntrance - шаблон, що визначає основний вид HTML-сторінок при реєстрації і авторизації.

 \_LayoutMaster - шаблон, який визначає основний вид для більшої частини HTML-сторінок, що відображають і дозволяють налаштовувати обліковий запис користувача.

\_LayoutAdmin - шаблон, який визначає основний вид для HTML-сторінок, які тільки адміністратор буде бачити; дозволяють адмініструвати облікові записи всіх користувачів.

Сторінка авторизації пропонує ввести існуючі дані облікового запису користувача у відповідні поля для введення: Email і Password; або ж скористатися авторизацією через одну з раннє створених облікових записів в популярному сервісі, такому як Google або Facebook.

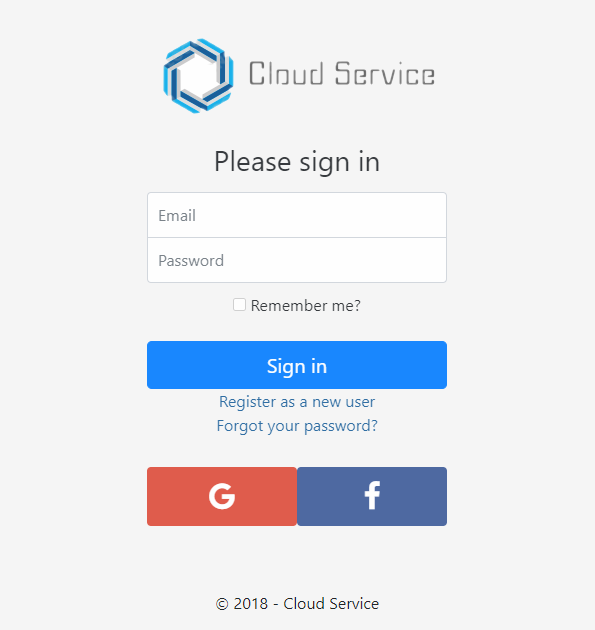


Рисунок 2.8 – Сторінка авторизації

Якщо ж користувач є Гостем в системі, то він може перейти на сторінку реєстрації на відповідне посилання нижче, а потім зареєструватися, попередньо заповнивши усі необхідні поля.

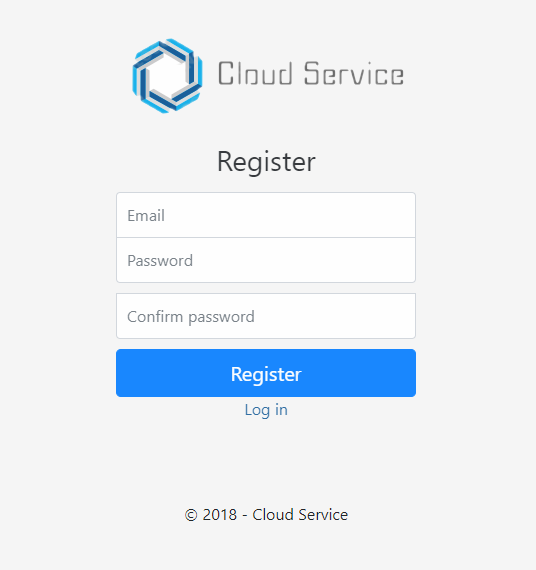


Рисунок 2.9 – Сторінка реєстрації

Представлення \_LayoutMaster визначає шаблон всіх сторінок, пов'язаних з роботою авторизованого користувача. У верхній частині другого шаблону знаходиться логотип даного сервісу, який є посиланням на домашню сторінку програми. Праворуч від логотипу знаходяться 2 кнопки, які запускають процес завантаження файлу/файлів на сервер і скачують звідти відповідно. У разі, якщо користувачеві треба зробити налаштування облікового запису, йому слід перейти на сторінку налаштувань, натиснувши на посилання, що представляє із себе встановлене для користувача ім'я, а правіше від неї знаходиться кнопка виходу з системи.



Рисунок 2.10 – Панель управління для авторизованого користувача

Використовуючи цю навігаційну панель, можна завантажувати і вивантажувати файли або папки в хмарному сховищі. Для того, щоб завантажити файл або групу файлів, а також папку або групу папок в сховище сервера, призначена кнопка «Upload» у вигляді «хмари» із стрілкою, направленою вгору, при натисканні якої починається процес завантаження попередньо вибраних файлів, з встановленої користувачем директорії на пристрої, на сервер.



Рисунок 2.11 – Кнопка завантаження файлів на сервер

Для того, щоб завантажити файли з сервера, призначена кнопка «Download», яка розташована лівіше, ніж кнопка «Upload». Вона спрацює тільки в тому випадку, коли користувач вибрав необхідні файли, зазначивши їх прапорцями.

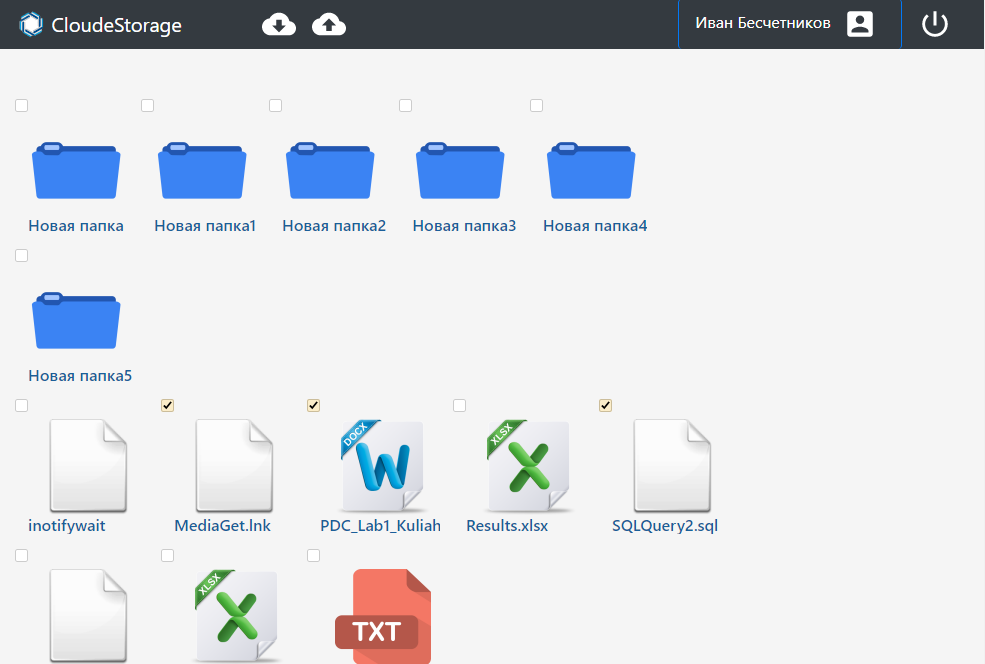


Рисунок 2.12 – Сторінка з усіма особистими файлами і папками

Потім сервіс автоматично їх збере в архів, який і буде викачаний в підсумку користувачем на ПК або гаджет.

Оскільки в проекті не менше 30 як часткових, так і простих сторінок, то далі будуть розглянуті основні, які демонструють виконання основного функціоналу додатком.

**DisplayFiles (FileController)**

Часткове представлення, в якому всі файли і папки відображаються в відсортованому вигляді, тому користувачеві буде легше орієнтуватися в цьому інформаційному просторі.

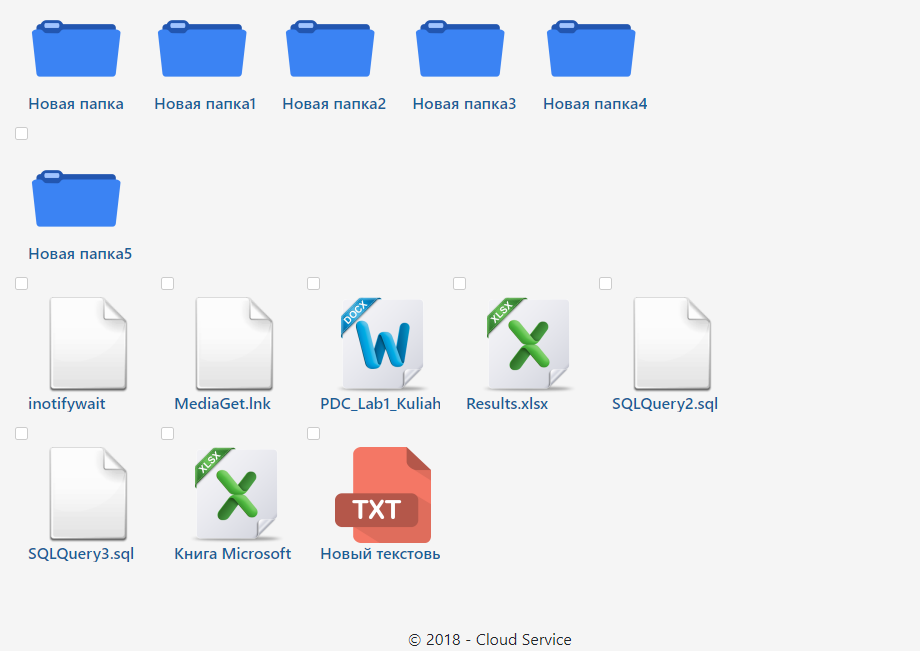


Рисунок 2.13 – Результат роботи методу DisplayFiles для контролера FileController

Всі папки завжди розташовуються у самій верхній частині цього середовища, а всі файли – нижче.

Створення, видалення, перейменування, переміщення папок, файлів проводиться за допомогою контекстного меню, що з'являється після натискання правою кнопкою миші по документу.

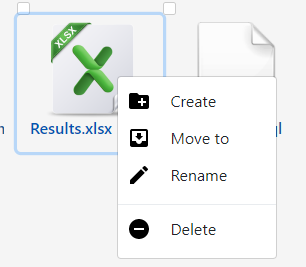


Рисунок 2.14 – Контекстне меню для створення деяких операцій над документом

**DisplayUsers (FileController)**

Часткове представлення адміністратора, яке відображає усіх користувачів системи. Ненаділеному роллю «Admin» користувачу, ця сторінка не буде доступна і при спробі до неї перейти, ввівши необхідний для цього маршрут в рядок запиту, буде викинута помилка додатком.

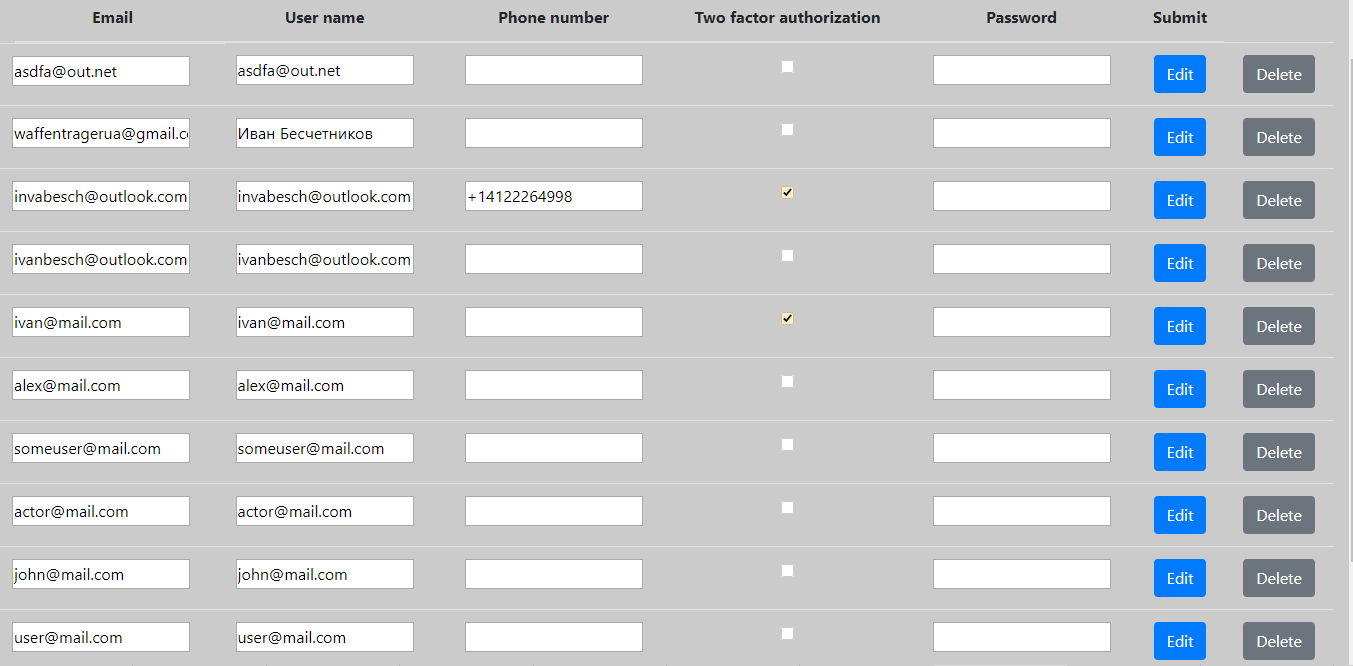


Рисунок 2.15 – Часткове представлення для адміністрування облікових записів

**Index (ManageController)**

Це представлення дозволяє авторизованому користувачеві налаштовувати свій профіль:

* змінити логін;
* змінити пароль;
* додати номер мобільного телефону;
* включити або вимкнути двохфакторну авторизацію;
* видалити аккаунт.

Щоб перейти на цю сторінку, слід натиснути на посилання у правому верхньому куті власного логіна.

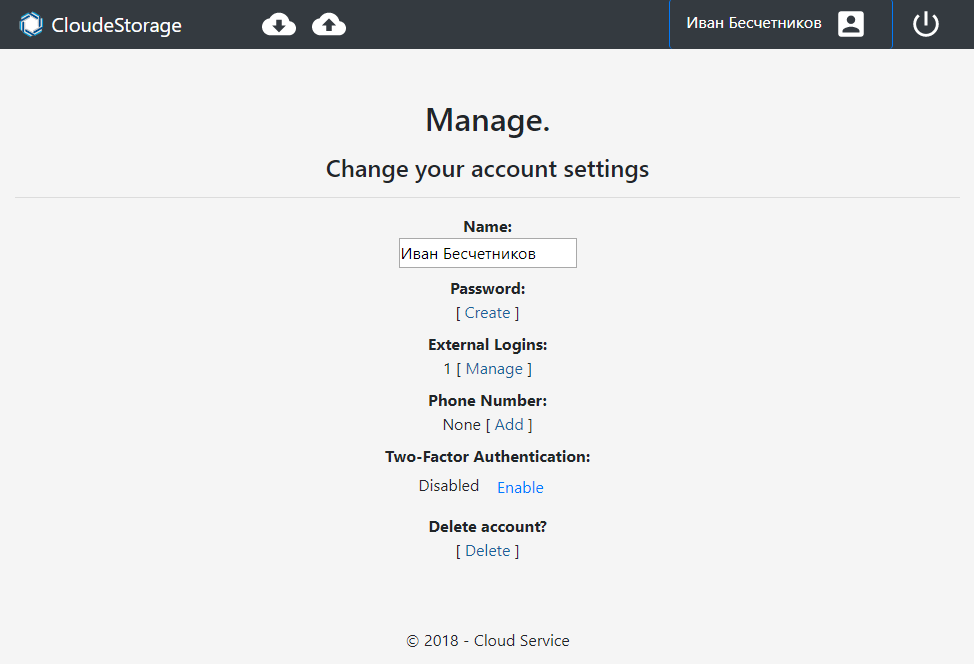


Рисунок 2.14 – Результат роботи методу Index для контролера ManageController

Розроблюваний графічний інтерфейс для пристроїв-серверів, що відстежує зміни в файлової системи пристрою, представляє з себе єдине вікно додатка мінімалістичного виду, яке містить дві кнопки: активації і дезактивації роботи даної служби.



Рисунок 2.15 – Інтерфейс програми FDMonitorService

Щодо FTP-сервера для даного виду пристроїв використовується вже готовий додаток FTP-server команди розробників Banana Studio, який в повному обсязі задовольняє всі потреби WCF-служби для обміну файлами [8].

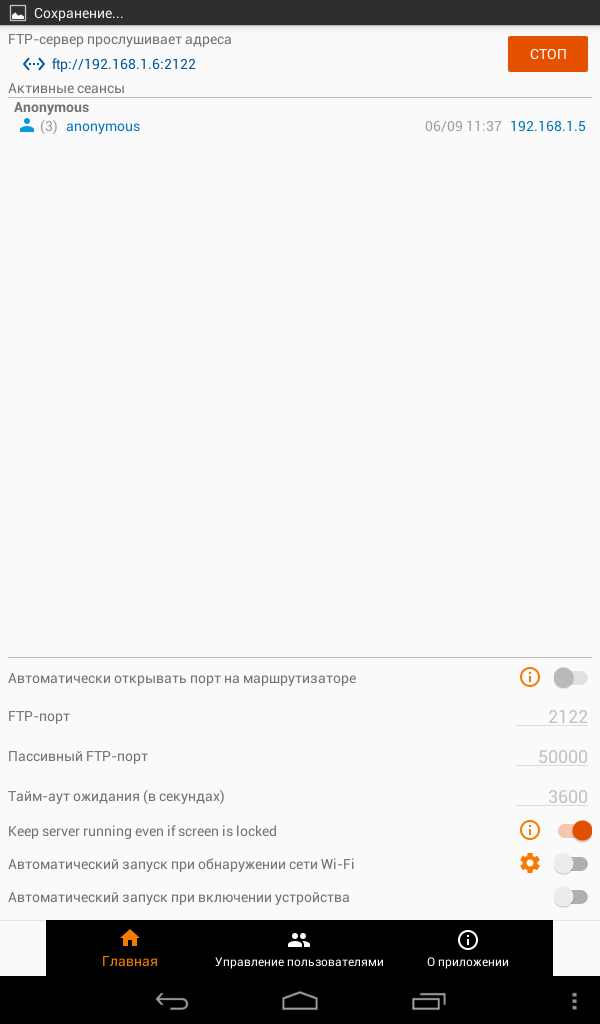


Рисунок 2.16 – Інтерфейс програми FTP-server

1. РОЗРОБКА
   1. Розробка діаграми класів

При аналізі поставленого завдання були виділені три основні сутності, які представлені трьома проектами, а також в яких були створені класи і інтерфейси для їх взаємодії: UI (HomeController, AccountController, ManageController, FileController, MainPage), BusinessLogic (FtpOperations, LoadBalancer, FileNFolders,), DAL (ApplicationDbContext, ApplicationUser, ApplicationUserManager, FileToSave, FileToDelte, FileToDisplay, FileToDownload, FileToUpload, Registration, DeleteAccount).

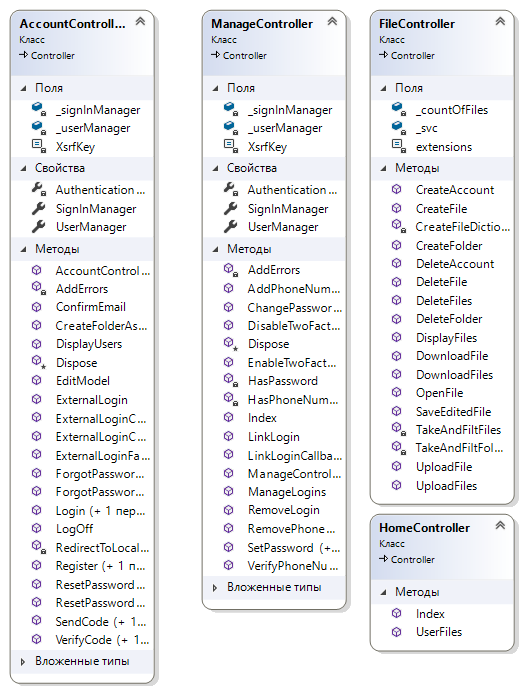


Рисунок 3.1 – Діаграма класів-контролерів web-додатка

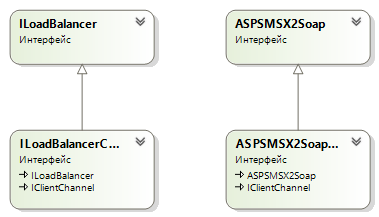


Рисунок 3.2 – Діаграма інтерфейсів служб: служба взаємодії з серверами і служба двохфакторної авторизації

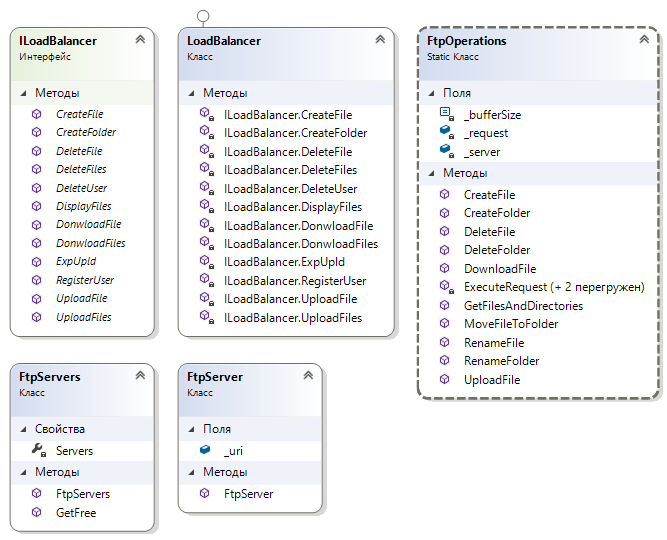


Рисунок 3.3 – Діаграма класів сервісу. Класи взаємодії web-додатка та серверів

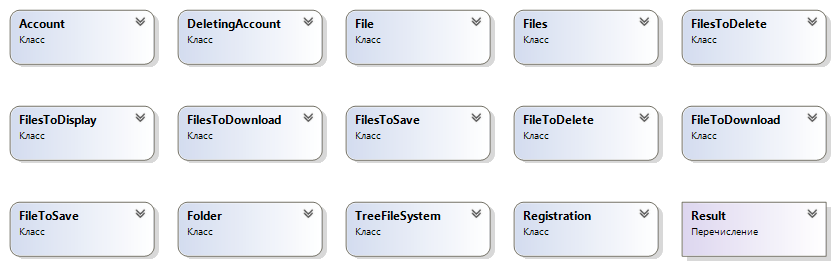


Рисунок 3.4 – Діаграма класів сервісу. Класи, що містять методи, націлені на виконання конкретного завдання або завдань над одним єдиним об'єктом, і які відповідають SOLID-принципам [9]

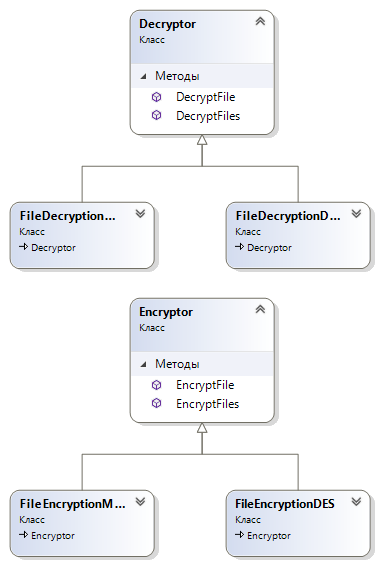


Рисунок 3.5 – Діаграма класів сервісу. Клас шифрування і дешифрування даних

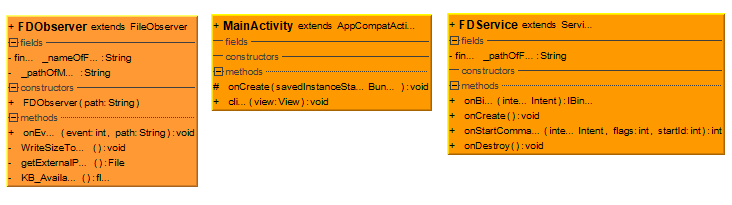


Рисунок 3.6 – Діаграма класів служби моніторингу файлової системи

Клас HomeController – контролер, який відповідає за відображення «домашньої» сторінки, тобто сторінки з одними файлами і папками.

Клас AccountConrtoller – контролер, який відповідає за створення і управління профілем користувача: реєстрація, авторизація, видалення і створення аккаунтів.

Клас ManageConrtoller – контролер, який відповідає за зміну всіх налаштувань облікового запису, а також покликаний взаємодіяти з базами даних.

Клас FileController – контролер, який відповідає за завантаження, скачування, видалення, створення, редагування, зберегти і сортування по типу файлів, папок на стороні сервера.

Клас LoadBalancer – клас, завдяки якому web-додаток взаємодіє з сервісом.

Клас FtpOperations – клас, що містить усі методи, що виробляють відповідні операції на стороні сервера/серверів.

Класи FtpServer і FtpServers – об'єкти прив'язки сервера і серверів відповідно.

Класи FileToSave, FileToDelte, FileToDisplay, FileToDownload, FileToUpload, Registration, DeleteAccount і т.д. – класи, що містять методи, які робляють елементарні операції на стороні сервера.

Класи Encryptor і Decryptor – об'єкти для шифрування і дешифрування даних.

Клас FDObserver – об'єкт відстеження файлової системи.

Клас FDService – об'єкт, що запускає службу, яка виконує усі операції класу FDObserver.

* 1. Поля, методи та властивості

**Методи і поля, використані в проекті Cloud Service, LoadBalancerSvc і FDObserver**

У таблицях наведено опис (призначення) методів і полів наступних класів:

Таблиця 3.1 – Методи HomeController

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи HomeController** | |
| public ActionResult Index | Запит GET на рендеринг головною web-сторінки і повернення її у вигляді відповіді користувачу |

Таблица 3.2 – Поля, методи и властивості AccountController

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Поля AccountController** | |
| private ApplicationSignInManager \_signInManager | Об'єкт класу авторизованих користувачів |
| private ApplicationUserManager \_userManager | Об'єкти класу зареєстрованих користувачів |
| **Методи AccountController** | |
| public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager ) | Конструктор, ініціалізуючий об'єкти класів ApplicationUserManager і ApplicationSignInManager |
| public ActionResult Login(string returnUrl) | Метод, викликає методом GET. Виведення часткового уявлення Login |
| public async Task<ActionResult> Login(LoginViewModel model, string returnUrl) | Асинхронний метод авторизації, викликає запитом POST. Служить для отримання моделі даних LoginViewModel |
| public ActionResult Register() | Метод, викликає запитом GET. Виведення часткового уявлення реєстрації |
| public async Task<ActionResult> Register(RegisterViewModel model) | Асинхронний метод реєстрації, викликається методом POST. Служить для отримання моделі даних RegisterViewModel |
| public Task CreateFolderAsync(string name) | Асинхронний метод для створення особистої папки користувача на сервері, викликає після успішного виконання методу Register |
| public ActionResult DisplayUsers() | Метод, що викликає часткове уявлення з адміністрування користувачів. |
| public async Task<ActionResult> EditModel(ApplicationUser user, string submit) | Асинхронний метод для збереження всіх змін, внесених адміністратором в обліковому записі. |
| **Властивості AccountController** | |
| public ApplicationSignInManager SignInManager | Властивість, яка отримує або повертає об'єкт класу ApplicationSignInManager |

|  |  |
| --- | --- |
| public ApplicationUserManager UserManager | Властивість, яка отримує або повертає об'єкт класу ApplicationUserManager |

Таблиця 3.3 – Методи FileController

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи FileController** | |
| public ViewResult DisplayFiles(string userName) | Метод, що викликає часткове уявлення відображення файлів і папок |
| public ActionResult DeleteAccount(string name) | Метод, що виробляє видалення облікового запису користувача |
| public ActionResult CreateAccount(string name) | Метод, який створює обліковий запис для користувача і його особисту папку на стороні сервера |
| public ActionResult DownloadFile(string name) | Метод для скачування користувачем файлу |
| public ActionResult DownloadFiles(List<string> list) | Метод для скачування користувачем файлів |
| public ActionResult UploadFile(Stream file) | Метод для завантаження файлу на сервер |
| public ActionResult UploadFiles(HttpPostedFileBase[] files) | Метод для завантаження файлів на сервер |
| public ActionResult DeleteFile(string name) | Метод, що видаляє файл на сервері |

|  |  |
| --- | --- |
| public ActionResult DeleteFiles(List<string> list) | Метод, що видаляє файли на сервері |
| public ActionResult CreateFile(string name) | Метод, що створює файли на сервері |
| public ActionResult SaveEditedFile(string name) | Метод, який зберігає всі внесені зміни користувачем у файлі |
| public ActionResult OpenFile(string name) | Метод, що викликає часткове уявлення для відображення вмісту офісного типу файлу |
| public ActionResult CreateFolder(string name) | Метод, який створює папку на сервері |
| public ActionResult DeleteFolder(string name) | Метод, який видаляє папку на сервері |
| private List<FileInfo[]> TakeAndFiltFiles(List<FileInfo> files, int itemPerRow | Метод, сортують файли при виведенні часткового уявлення |
| private List<DirectoryInfo[]> TakeFolders(List<DirectoryInfo> files, int itemPerRow) | Метод, який сортує папки при виведенні часткового уявлення |

Таблиця 3.4 – властивості FilesNFolders

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Властивості FilesNFolders** | |
| public List<FileInfo[]> Files | Містить всі призначені для користувача файли |
| public List<DirectoryInfo[]> Directories | Містить всі призначені для користувача папки |

Таблиця 3.5 – Методи ManageController

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи ManageController** | |
| public async Task<ActionResult> Index(ManageMessageId? message) | Асинхронний метод, який викликається запитом GET. Змінює всі призначені для користувача облікові дані і повідомляє про це користувача |
| public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey) | Асинхронний метод, який викликається запитом POST. Видаляє старий логін і встановлює новий |
| public ActionResult AddPhoneNumber() | Метод, який викликається запитом GET. Виводить часткове уявлення для прив'язки мобільного номера до облікового запису |
| public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model) | Асинхронний метод, який викликається запитом POST. Прив'язує номер мобільного телефону до аккаунту |

Таблиця 3.6 – Поля та методи LoadBalancerSvc

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи LoadBalancerSvc** | |
| bool ILoadBalancer.CreateFile(string file) | Метод, який створює файл на сервері |
| bool ILoadBalancer.CreateFolder(string localPath) | Метод, який створює папку на сервері |
| int ILoadBalancer.DeleteFile(string localPath) | Метод, який видаляє файл на сервері |
| bool ILoadBalancer.DeleteFiles(string localPath) | Метод, що видаляє файли на сервері |
| bool ILoadBalancer.DeleteUser(string mail) | Метод, що видаляє папку-користувача на сервері |
| List<string> ILoadBalancer.DisplayFiles(string path) | Метод, що передає набір файлів і папок в представлення для обраної директорії |
| Stream ILoadBalancer.DonwloadFile(string localPath) | Метод для скачування файлу з сервера |
| Stream ILoadBalancer.DonwloadFiles(string localPath) | Метод для скачування файлів з сервера |
| bool ILoadBalancer.RegisterUser(string mail) | Метод, який створює папку-користувача на сервері |
| bool ILoadBalancer.UploadFile(Stream stream, string localPath) | Метод для завантаження файлу на сервер |
| bool ILoadBalancer.UploadFiles(Stream stream, string localPath) | Метод для завантаження файлів на сервер |

Таблиця 3.7 – Поля та методи MainActivity

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи MainActivity** | |
| protected void onCreate(Bundle savedInstanceState) | Ініціалізатор вікон |
| public void click(View view) | Обробник події натискання кнопки |

Таблиця 3.8 – Поля та методи FDService

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Методи FDService** | |
| public void onCreate() | Ініціалізатор служби |
| public int onStartCommand(Intent intent, int flags, int startId) | Запускає службу |

Таблиця 3.9 – Поля та методи FDObserver

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| **Поля FDObserver** | |
| private final String \_nameOfFile | Файл, в який буде проводитися запис о кількісті вільної пам'яті на пристрої |
| private String \_pathOfMem; | Містить шлях до директорії |
| **Методи FDObserver** | |
| public FDObserver(String path) | Конструктор класу |
| public void onEvent(int event, @Nullable String path) | Обробник події зміни файлової системи |
| private void WriteSizeToFile() | Метод запису в файл |
| private float KB\_Available() | Метод розрахунку кількості вільної пам'яті в кілобайтах |

* 1. Розробка алгоритмів

Відповідно до розробленої діаграми класів необхідно спроектувати такі класи і методи:

1. HomeController;
2. AccountController;
3. ManageController;
4. FileController;
5. FDObserver;
6. LoadBalancerSvc;
7. DisplayFiles;
8. DisplayUsers;
9. DownloadFile;
10. UploadFile;
11. Encrypt;
12. Decrypt;
13. та інші.

Нижче наведено кілька алгоритмів роботи деяких методів зі списку:



Рисунок 3.4 – Схема принципу роботи методу обробника події onChangeFS зміни файлової системи на пристрої



Рисунок 3.5 – Схема принципу роботи методу DisplayFiles на серверній стороні, який отримує список файлів і папок, за вказаним шляхом



Рисунок 3.6 – Схема принципу роботи методу Register для контролера AccountController, реєструючого користувача в системі [10]

1. ВЕРИФІКАЦІЯ І ТЕСТУВАННЯ ПРОГРАМИ

Верифікація проводилася шляхом перевірки реалізації вимог технічного завдання:

1. Реєстрація, авторизація користувача.
2. Завантаження файлу на сервер.
3. Створення, видалення, перейменування файлу.

Результати виконаної роботи, які підтверджують, що програма правильно виконує завдання наведені нижче:

Таблиця 4.1 – Набір тестів

|  |  |  |
| --- | --- | --- |
| Назва тесту | Результат тесту | Скріншот з результатом |
| Реєстрація користувача Гостя | Користувач успішно зареєстрований |  |
| Завантаження файлу на сервер | Файл успішно завантажений |  |
| Перейменування папки | Папка успішно перейменована |  |
| Перейменування файлу | Файл успішно перейменовано |  |

|  |  |  |
| --- | --- | --- |
| Видалення файлу | Файл успішно видалений |  |
| Створення папки | Папка успішно створена |  |

Таблиця 4.2 – Набір Unit-тестів

|  |  |  |
| --- | --- | --- |
| Метод, що тестувався | Опис методу | Скріншот з результатом |
| CreatingUser StorageFolder\_ CreatedUser StorageFolder | Створення папки під зберігання особистих файлів для щойно зареєструвався користувача. |  |
| DownloadigFiles\_DownloadedFiles | Завантаження файлів на сервер. |
| UploadingFiles\_ UploadedFiles | Викачування файлів з сервера. |
| CreatingFolder\_ CreatedFolder | Створення простої папки. |
| RomovingFile\_ RemovedFile | Видалення файлу. |
| IsEqualHashesOfLoginAndPassword\_CheckHashesOfLoginAndPassword | Одностороннє шифрування призначене для облікових даних користувача (логін і пароль) hash-функцією. |  |
| IsEncryptedDataOfFile\_CheckDataOfFile | Шифрування вмісту файлу алгоритмом RSA. |

ВИСНОВКИ

При виконанні даної дипломної роботи був реалізований проект, що дозволяє завантажувати, вивантажувати файли, папки, а також організовувати робочий простір з даними. Розроблена програма верифіцірована і протестована.

Web-додаток, який був розроблений, – має зручний, мінімалістичний, інтуїтивно зрозумілий інтерфейс, який дозволяє користувачеві легко орієнтуватися в своєму просторі з папками і файлами, що було досягнуто за допомогою фреймворка стилів Bootstrap 4. Також завдяки цьому інструменту і мінімальної кількості анімаційних об'єктів сторінки займають набагато менше обсягу пам'яті в наслідок чого завантаження сторінок, що ініціюється користувачем в браузері, відбувається швидше.

Кількість web-сторінок для користувача, який не є особою, уповноваженою з адміністрування облікових записів, зведено до мінімуму: головна сторінка та сторінка налаштувань облікового запису. Кількість web-сторінок для адміністратора – 3, де дві – такі ж, як і у простого користувача, наділеного роллю «user», а третя надає список всіх призначених для користувача облікових записів, редагуючи які немає необхідності переходити на нову сторінку додатка, що досягається за допомогою редагованих текстових полів введення.

Так як ASP.NET MVC 5 має в наявності ряд допоміжних компонентів, що полегшують процес побудови і модернізації логіки сценаріїв серверної частини, то необхідність у створенні точно таких інструментів або подібних була відсутня. До них відносяться ASP.NET Identity, що відповідає за авторизацію, реєстрацію та автентифікацію користувачів в системі, і Open Web Interface for .NET (OWIN), об'єкти якого ідентифікують як самих користувачів, так їх дії, що, безсумнівно, спрощує та робить більш надійним, менш уразливим web-додаток від несанкціонованих дій з боку користувачів в мережі [11].

З метою зменшення загального навантаження на систему, web-додаток передає частину своєї відповідальності службі, яка, як передбачається, розміщується на іншому сервері; як наслідок – швидка, чуйна, а також відмовостійка робота усього комплексу. Важливим аспектом щодо збільшення продуктивності web-додатки – распараллелізація роботи методів за допомогою бібліотек асинхронного TAP і многопоточного TPL програмування [12].

Тестування було розділено на 3 етапи: тестування методів, класів, компонентів (додаток ASP.NET MVC - Unit-тести, з використанням Moq); тестування всієї програми (окремо взятий рівень архітектури комплексу) та тестування всього комплексу, де web-додаток і сервіс запускалися на одному ПК, як локальні процеси, а також один мобільний пристрій, під управління ОС Android з 17 рівнем API. Всі тести були пройдені успішно.

Цей додаток виконує:

1. реєстрацію, авторизацію користувачів;
2. створення, видалення, перейменування файлів, папок;
3. завантаження файлів, папок в сервера;
4. вивантаження файлів, папок з сервера;
5. вилучення користувача;
6. настроювання облікових даних.

Для подальшої модернізації проекту можна:

1. поліпшити оптимізацію додатки;
2. додати функцію «поділитися» для файлів і папок;
3. додати чат для двох осіб чи груп;
4. редагування файлів docx, exlsx, pptx, txt.

СПИСОК ЛІТЕРАТУРИ

1. WIKIPEDIA [Електр. ресурс] – Режим доступу:  
   <https://ru.wikipedia.org/wiki/Закон_Мура>

# ASP.NET MVC FRAMEWORK – WIKIPEDIA [Електр. ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/ASP.NET_MVC_Framework>

# ASP.NET MVC 5 – METANIT.COM [Електр. ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/>

# XAMARIN [Електр. ресурс] – Режим доступу: <https://www.xamarin.com/platform>

# PROFESSORWEB [Електр. ресурс] – Режим доступу: <https://professorweb.ru/>

# WIKIPEDIA [Електр. ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/Model-View-Controller>

1. MICROSOFT [Електр. ресурс] – Режим доступу:

<https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>

1. MICROSOFT [Електр. ресурс] – Режим доступу:

<https://docs.microsoft.com/en-us/aspnet/aspnet/overview/owin-and-katana/an-overview-of-project-katana>

1. ANDROID [Електр. ресурс] – Режим доступу:

<https://www.android.com/intl/ru_ru/versions/jelly-bean-4-3/>

1. BOOTSTRAP [Електр. ресурс] – Режим доступу:

<https://getbootstrap.com/docs/4.0/>

1. MSDN [Електр. ресурс] – Режим доступу:

<https://msdn.microsoft.com/ru-ru/library/ms182532.aspx>

1. PROFESSORWEB [Електр. ресурс] – Режим доступу:

<https://professorweb.ru/my/ASP_NET/mvc/level2/2_5.php>

# **ДОДАТОК А Технічне завдання**

1. **Вступ** 
   1. **Найменування програми**Розроблюваний комплекс програм називається «Сервіс хмарного сховища файлів».
   2. **Коротка характеристика області застосування**Цей сервіс призначений для зберігання, редагування файлів на віддалених серверах, а також для їх обміну. Він організований за «хмарним» принципом.
2. **Підстава для розробки** 
   1. **Підстава для проведення розробки**Підстава для розробки Програми – завдання на дипломну роботу кафедри "Комп'ютерних систем, мереж і кібербезпеки" Національного аерокосмічного університету ім. Н.Є. Жуковського "ХАІ".
   2. **Найменування та умовне позначення розробки**Найменування теми розробки - «Сервіс хмарного зберігання файлів».

Умовне позначення теми розробки (шифр теми) - «СХЗФ-01».

1. **Призначення розробки**
   1. **Функціональне призначення розробки**Функціональним призначенням збереження, завантаження, вивантаження файлу/файлів різних форматів, а також надання можливості переглядати вміст їх (файли форматів docx, xlsx, pptx, txt).
   2. **Експлуатаційне призначення**

Сервіс призначений для зберігання файлів на віддалених серверах, які невідомі користувачеві і представляються єдиним сервером, з яким і здійснюється робота. Доступ до даних відбувається за рахунок авторизації користувача за допомогою раннє створеного аккаунта. Авторизоване особа так само може переглядати вміст файлів форматів docx, xlsx, pptx, txt. Безпека особистих даних облікового запису покладено на сервіс, який буде шифрувати їх і при добуванні дешифрувати.

1. **Вимоги до програми або програмного виробу** 
   1. **Вимоги до функціональних характеристик** 
      1. **Загальні відомості, призначення та склад сервісу СХЗФ-01**

СХЗФ-01 - це програмний комплекс засобів, призначених для організації хмарного сховища файлів всіх форматів і розмірами, не більше 2 ГБ. Даний сервіс буде доступний тільки авторизованим користувачам, у яких є свій обліковий запис з особистими файлами. Авторизованим користувачам слід пройти реєстрацію, щоб отримати доступ до сховища, виділеного сервісом. Він дозволяє зберігати абсолютно будь-які файли, переглядати вміст файлів форматів docx, xlsx, pptx.

СХЗФ-01 складається з головного сервера, сервісу і безлічі серверів-сховищ. В обов'язки головного сервера входить:

* Реєстрація користувачів;
* Авторизація користувачів;
* Автентифікація користувачів;
* Верифікація одержуваних даних;
* Висновок графічного інтерфейсу web-додатки;
* Передача даних сервісу;
* Отримання даних від сервісу.

Призначенням сервісу є передача і отримання даних як від web-додатку, так і від пристроїв-серверів. Причина, по якій цей компонент відділений від інших в тому, що це підвищує надійність і відмовостійкість, а також захищеність усієї системи в цілому.

Оскільки концепція «хмари» має на увазі використання безліч віддалених один від одного серверів, то в даному випадку використовуються сервера, на яких будуть зберігатися всі призначені для користувача дані, попередньо зашифровані web-додатком.

* + 1. **Вимоги до складу виконуваних функцій**

Сервіс повинен виконувати наступні функції:

1. реєстрація, авторизація, автентифікація користувачів в системі;
2. деавторизація користувачів;
3. валідація даних, одержуваних від користувачів;
4. наділяти користувачів при реєстрації в системі роллю «user»;
5. видаляти акаунти користувачів, наділених роллю «admin», користувачем з роллю «senioradmin», а також облікові записи, що мають роль «user»;
6. видаляти акаунти користувачів, наділених роллю «user», користувачем з роллю «admin»;
7. завантажувати і вивантажувати файли;
8. зберігати призначені для користувача файли на працездатних і в той же час «активних» пристроях-серверах;
9. шифрувати і дешифрувати файли відповідно до обраного користувачем режимом: режим підвищеної надійності (алгоритм RSA) і режим швидкої роботи з файлами (алгоритм DES).
10. обмежувати розмір файлів, що завантажуються. Якщо користувач намагатиметься завантажити файл, розміром понад 2 ГБ, відхилити запит;
11. встановлювати або розривати з'єднання з сервісом з програми на пристрої-сервері;
12. у разі виникнення програмної помилки на пристрої-сервері, продовжити функціонування програми та відобразити відомості про виниклу помилку;
13. шукати, сортувати, групувати, створювати файли в web-додатку;
14. надавати можливість переглядати вміст файлів форматів docx, xlsx, pptx, txt, pdf;
15. змінювати параметри створеного аккаунта: пошта, номер мобільного телефону, двохфакторна авторизація, ім'я, вік, аватар (графічне представлення користувача в системі);
16. створювати обліковий запис після успішної реєстрації користувача;
17. видаляти обліковий запис.
    * 1. **Вимоги до організації вхідних даних**

Вхідні дані при реєстрації: електронна пошта і номер мобільного телефону повинні бути існуючими.

Вхідні дані при авторизації: облікові дані зареєстрованого користувача.

Вхідні дані при завантаженні файлів: розмір файлу повинен бути не більше 2 ГБ.

Вхідні дані при перегляді вмісту файлу: файл повинен представлятися форматом або docx, або xlsx, або pptx, або txt, або pdf.

* + 1. **Вимоги до організації вихідних даних**

Вихідні дані при реєстрації: оповіщення про успішну чи неуспішною реєстрації.

Вихідні дані при авторизації: перехід на сторінку особистого кабінету, або сповіщення про виниклу помилку.

Вихідні дані при завантаженні файлів: оповіщення про успішно проведену операцію або про виниклу помилку.

* + 1. **Вимоги до тимчасових характеристик**

Максимально допустимий час завантаження файлу/файлів на сервер за один сеанс: 7 годин.

Максимально допустимий час завантаження файлу/файлів за один сеанс: 7 годин.

* + 1. **Вимоги до безпеки**

Сервіс повинен гарантувати безпеку облікових даних користувача (логін, пароль, адресу електронної пошти, номер мобільного телефону), а також особистих даних.

* 1. **Вимоги до надійності** 
     1. **Перелік заходів щодо забезпечення надійного (сталого) функціонування сервісу**

Надійне (стійке) функціонування програми має бути забезпечене виконанням Замовником сукупності організаційно-технічних заходів:

1. організація безперебійного живлення технічних засобів;
2. використання ліцензійного програмного забезпечення;
3. допустимий діапазон температури приміщення, в якому експлуатуються пристрої-сервера: від +15 до +35 ° С;
4. допустимий діапазон температури приміщення, в якому зберігаються пристрої-сервера: від +10 до +45 ° С;
5. обмеження доступу до приміщення, в якому знаходяться пристрої-сервера;
6. адекватне, раціональне і виняткове наділення роллю «admin» облікових записів співробітників.
   * 1. **Вимоги до забезпечення надійного функціонування сервісу**

У процесі роботи Програми можливі відмови з її вини, інших програмних засобів або апаратного забезпечення.

Можливі відмови в роботі Програми з її вини не повинні призводити до втрати даних в файлах, які використовуються програмою або користувачем в момент відмови.

Безпека даних в разі аварійних ситуацій, які не викликані Програмою, повинна забезпечуватися засобами ОС.

Не допускається вимога перезавантаження ОС після відмови Програми.

* + 1. **Час відновлення після відмови**

Час відновлення після відмови, викликаного збоєм електроживлення технічних засобів (іншими зовнішніми факторами), нефатальним збоєм (не крах) операційної системи, не повинно перевищувати стількох-то хвилин за умови дотримання умов експлуатації технічних і програмних засобів.

Час відновлення після відмови, викликаного несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинно перевищувати часу, необхідного на усунення несправностей технічних засобів і переустановлення програмних засобів.

* + 1. **Відмови через некоректні дії оператора**

Відмови програми можливі внаслідок некоректних дій оператора (користувача) при взаємодії з операційною системою. Щоб уникнути виникнення відмов програми за вказаною вище причини слід забезпечити роботу кінцевого користувача без надання йому адміністративних привілеїв.

* 1. **Умови експлуатації** 
     1. **Кліматичні умови експлуатації**

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

* + 1. **Вимоги до видів обслуговування**

Див. Вимоги до забезпечення надійного (сталого) функціонування програми.

* + 1. **Вимоги до чисельності та кваліфікації персоналу**

Мінімальна кількість персоналу, необхідного для роботи програми, має становити не менше двох штатних одиниць - системний адміністратор і кінцевий користувач програми (оператор).

Системний адміністратор повинен мати вищу профільну освіту та сертифікати компанії-виробника операційної системи. Завдання, що виконуються системним адміністратором:

1. підтримання працездатності технічних засобів;
2. установка (інсталяція) і підтримку працездатності системних програмних засобів - операційної системи;
3. установка (інсталяція) програми.

Кінцевий користувач програми (оператор) повинен володіти практичними навичками роботи з графічним призначеним для користувача інтерфейсом операційної системи.

Персонал повинен бути атестований на II кваліфікаційну групу з електробезпеки (для роботи з конторським обладнанням).

* 1. **Вимоги до складу і параметрів технічних засобів**

До складу технічних засобів повинні входити:

* сервер або стаціонарний комп'ютер:
  1. процесор Intel Core i5 або Intel Xeon E3 1240v3 і вище;
  2. пам'ять 8 ГБ DDR3 і більше;
  3. дискова система 240 ГБ SSD або HDD і більше;
  4. ОС Windows 8 або Windows Server 2012 і вище;
  5. IIS 8.0 і вище.
* Пристрій під керуванням ОС Android.
  1. **Вимоги до інформаційної та програмної сумісності**
     1. **Вимоги до інформаційних структур і методів рішень**

Вимоги до інформаційних структур на вході і виході, а також до методів рішення не пред'являються.

* + 1. **Вимоги до вихідного коду і мов програмування**

Для розробки web-додатки повинні використовуватися наступні мови програмування: C # (7.0 і вище) - розробка інфраструктури серверної частини і JavaScript - розробка «дружнього» інтерфейсу застосунку, в середовищі розробки Visual Studio 2016 і вище.

Для розробки сервісу також повинен використовуватися С # 7.0 і вище, в середовищі розробки Visual Studio 2016 і вище.

Для розробки серверного додатка повинен використовуватися Аndroid API 17.

* + 1. **Вимоги до програмних засобів, які використовуються сервісом**

Web-додаток повинен працювати під управлінням ОС Windows 8 і вище або Windows Server 2012 і вище, а також допускаються відповідні пакти оновлень.

Системні програмні засоби, що використовуються програмою, повинні бути представлені ліцензійної локалізованої версією операційної системи. Допускається застосування відповідного пакета оновлень.

Для запуску web-додатки повинен бути встановлений веб-сервер IIS Express або IIS 5.0, один з яких слід коректно налаштований відповідно до розташування папки проекту web-додатки і ОС.

Параметри ОС налаштовані таким чином, щоб виконання програми було можливим. Для цього як мінімум повинні бути правильно налаштовані змінні оточення середовища виконання web-додатки і коректно встановлені права користувача.

* 1. **Спеціальні вимоги**

Спеціальні вимоги не пред'являються.

1. **Вимоги до програмної документації**
   1. **Попередній склад програмної документації**

В результаті розробки сервісу повинна бути представлена наступна програмна документація:

* 1. технічне завдання;
  2. схеми алгоритмів;
  3. тексти програм;
  4. пояснювальна записка;
  5. керівництво оператора;

Крім програмного забезпечення на диску обов'язково повинна перебувати пояснювальна записка в електронному вигляді, що містить весь комплект документації, передбачений в п. 5.1 цього технічного завдання.

# **ДОДАТОК Б Керівництво клієнта**

1. **Умови роботи web-додатку**

До складу технічних засобів повинно входити будь десктопне або мобільний пристрій, з наявністю будь-якого браузера і доступом в Інтернет.

1. **Використання**

Щоб мати можливість користуватися сервісом, необхідно пройти етап реєстрації або авторизації. Якщо користувач не має раніше створений обліковий запис, йому слід зареєструватися, перейшовши на відповідну сторінку реєстрації, кликнувши на посилання, скріншот якої наведено в П.Б.1.

Для реєстрації слід заповнити 3 поля: Email, Password і Confirm Password.

Email - поле, в яке треба ввести існуючу адресу електронної майже.

Password - поле, в яке треба ввести придуманий пароль, необхідний також і при подальших авторизації.

Confirm Password - поле, в яке треба ввести пароль, оголошений в «Password». Це необхідно, щоб переконатися в тому, що випадково допущена помилка користувачем в паролі не буде збережена системою і позбавить від перереєстрації або проходження процесу відновлення пароля (посилання на цю функцію показана на скріншоті П.Б.2. З'являється після невдало пройденого етапу авторизації).

Для користувачів, що мають раннє створений обліковий запис, слід заповнити поля Email і Password. У П.Б.3 наведено скріншот зі сторінкою авторизації.

На «домашньої» сторінці користувачеві представляється весь основний функціонал по маніпуляції особистими файлами і папками (П.Б.4). Щоб завантажити файл або групу файлів, слід натиснути на кнопку «Upload», яка виглядає, як на скріншоті в П.Б.5. Якщо ж треба завантажити файли, необхідно вибрати цільові, тобто відзначити їх прапорцями, як показано на скріншоті П.Б.6, а потім натиснути на кнопку «Download» (П.Б.7).

Щоб створити, видалити або перейменувати файл / папку, необхідно натиснути правою кнопкою миші по вибраному елементу, після чого з'явиться невелике контекстне меню з переліком цих функцій. Скріншот контекстного меню показаний в П.Б.8.

Перейти на сторінку управління особистим обліковим записом можна кликнувши на посилання, яка є логіном (приклад такого посилання показаний на скріншоті П.Б.9).

На сторінці управління особистим обліковим записом відображені 4 посилання, які активують ту чи іншу функцію, в залежності від того, чим користувач хоче скористатися: змінити пароль, встановити інший спосіб авторизації (через зовнішні сервіси), прив'язати номер мобільного телефону або ж включити двохфакторну авторизацію. Скріншот з функціями показаний в П.Б.10.

Для того, щоб деавторизуватися, призначена посилання «Log off» (П.Б.11).

# **ДОДАТОК В Керівництво адміністратора**

1. **Сервер під web-додаток:**

До складу технічних засобів повинно входити будь десктопне або лептопне пристрій з мінімальними технічними характеристиками (CPU - 2.4Ghz, RAM - 4 GB, HDD - 1 GB).

1. **Серверні пристрої під зберігання файлів:**

До складу технічних засобів повинно входити будь-який мобільний пристрій з мінімальними технічними характеристиками (CPU - 1.4Ghz, RAM - 512 MB, SSD - 1 GB).

1. **Установка web-додатка**

Для установки додатків виконайте наступні кроки:

1. скачати файл-архів;
2. розпакувати у будь-якій зручній директорії;
3. переконатися в наявності на кінцевому пристрої платформи .NET Framework з версією 4.5.2 і вище;
4. переконатися в наявності IIS Express 10.0 і вище;
5. у кореневій директорії IIS Express, в папці config, відкрити файл applicationhost.config будь-яким інструментом з редагування текстових файлів. На початок секції <sites> додати наступний код:  
    <site name="Cloud Storage" id="1">

<application path="/" applicationPool="Clr4IntegratedAppPool">

<virtualDirectory path="/" physicalPath="..." />

</application>

<bindings>

<binding protocol="http" bindingInformation=":57460:localhost"/>

</bindings>

</site>

1. замість трьох крапок у властивості physicalPath прописати шлях до розпакувати з додатком;
2. запустити IIS Express.
3. **Установка сервісу**
4. скачати файл-архів;
5. розпакувати у будь-якій зручній директорії;
6. переконатися в наявності платформи на кінцевому пристрої платформи .NET Framework з версією 4.5.2 і вище;
7. переконатися в наявності IIS Express 10.0 і вище;
8. переконатися в наявності Visual Studio не нижче 15;
9. запустити visual studio;
10. зібрати рішення і проект;
11. запустити проект «LoadBalancerSvc».
12. **Установка додатків для пристроїв-серверів**
13. скачати файл-архів;
14. розпакувати у будь-якій зручній директорії;
15. встановити, слідуючи інструкціям інсталятора.
16. скачати ftp-server banana studio з Play Market.
17. відкрити програму і встановити порт: 2121.
18. натиснути кнопку «запуск».
19. **Можливі повідомлення адміністратору при неправильному виконанні якихось дій в програмі**

Якщо в процесі роботи програми виникають деякі помилки, збої або додаток просто не може запуститися, і, якщо вам не вдається вирішити програмні конфлікти самостійно, то опишіть докладно проблему в повідомленні, а потім надішліть на наступну адресу електронної пошти: ivanbesch@outlook.com.

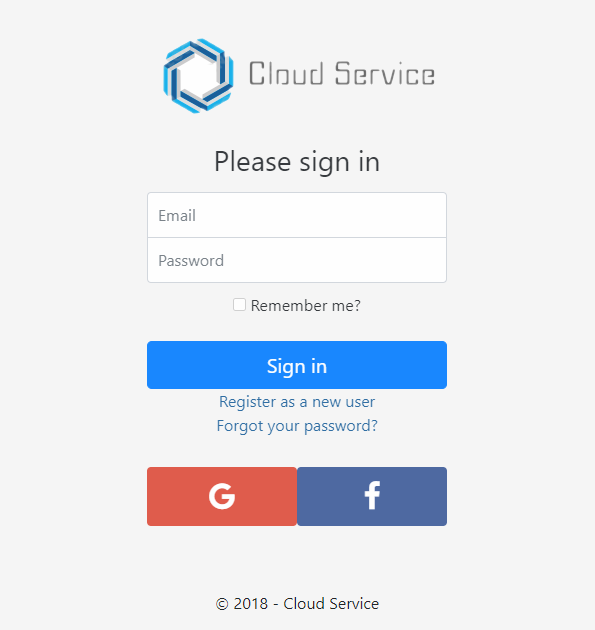
**П.Б.1. Посилання на сторінку реєстрації.**



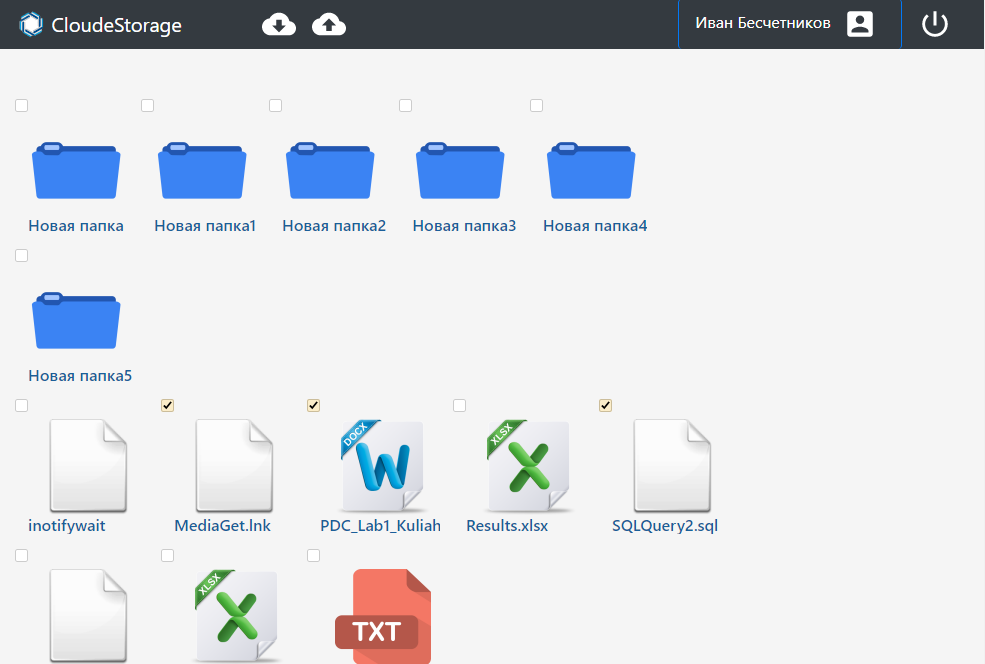
**П.Б.2. Посилання на відновлення пароля.**



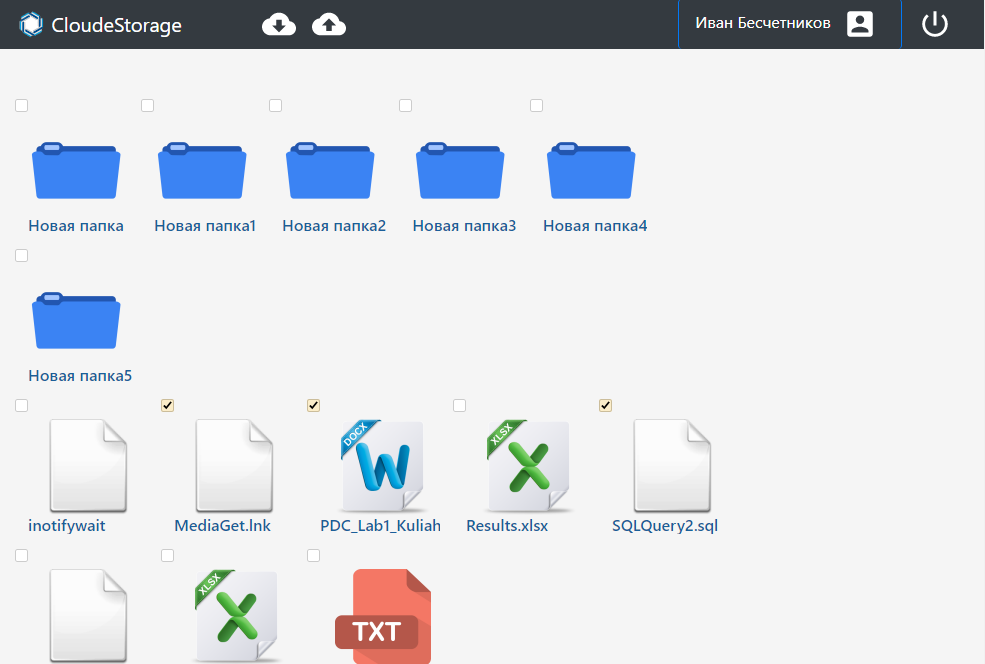
**П.Б.3. Сторінка авторизації.**



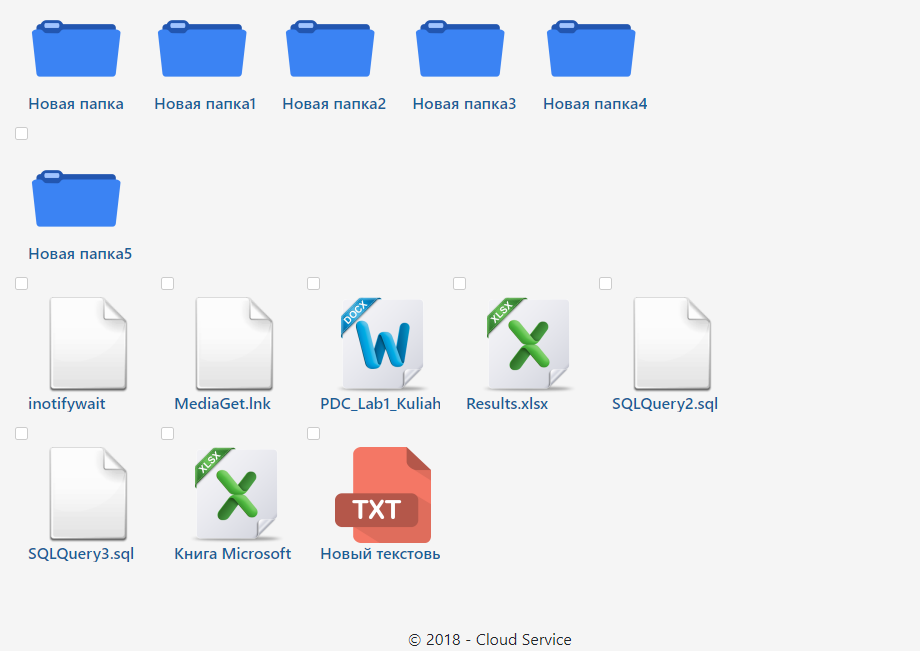
**П.Б.4. Панель функцій і вміст сховища користувача.**



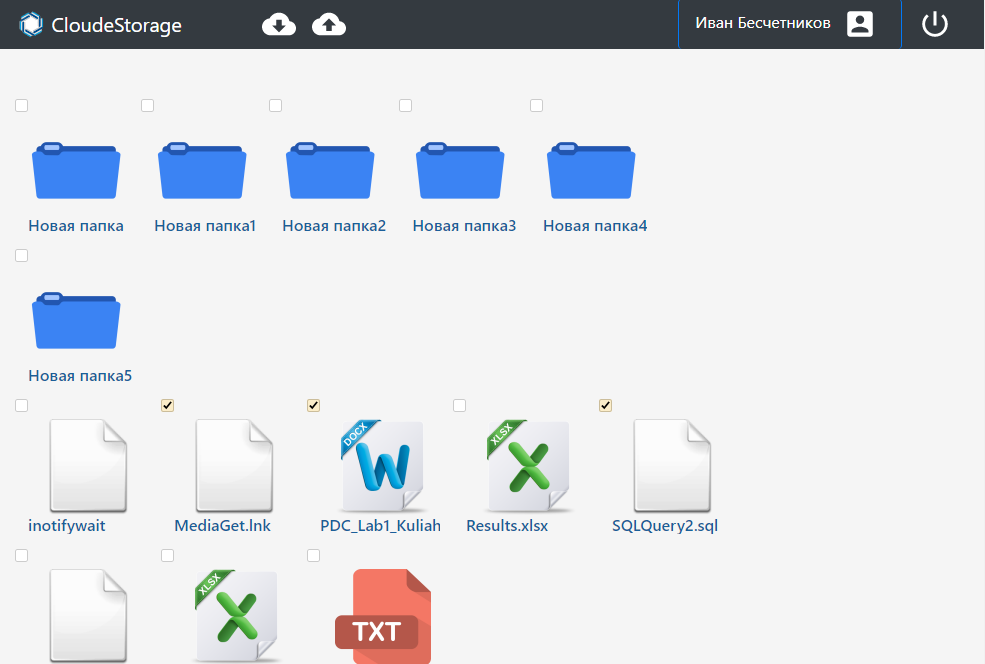
**П.Б.5. Кнопка завантаження на сервер файлів/папок.**



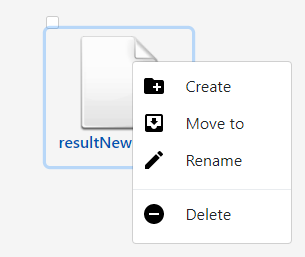
**П.Б.6. Файли, відмічені прапорцями.**



**П.Б.7. Кнопка скачування.**



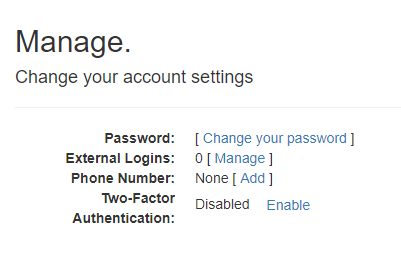
**П.Б.8. Контекстне меню при натисканні правої кнопки миші.**



**П.Б.9. Посилання-логін на облікові дані.**



**П.Б.10. Сторінка управління обліковими даними.**



**П.Б.11. Посилання на деавторизацію.**

