Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ВІРТУАЛЬНА КІМНАТА ДАНИХ ЯК СХОВИЩЕ КОНФІДЕНЦІЙНИХ КОРПОРАТИВНИХ ДОКУМЕНТІВ

Текстова частина до курсової роботи
за спеціальністю „Програмна інженерія”

Керівник курсової роботи

д. т. н., д. Глибовець А.М.

_____
(підпис)
“____” _____ 2020 р.

Виконав студент
Торба Т.В.

“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
проф., д.ф-м.н.
_____Гороховський С.С.
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту 1-го курсу магістерської програми "Комп'ютерні науки" факультету інформатики Торбі Тетяні Вікторівні

ТЕМА: «Віртуальна кімната даних як сховище конфіденційних корпоративних документів»

Вихідні дані:
-
Зміст ТЧ до магістерської роботи:
Зміст
Анотація
Вступ
1 Дослідження галузі
2 Проектування архітектури
3 Розробка мікро сервісу
4 Результати
Висновки
Список літератури
Додатки

Дата видачі „___” _____ 2020 р. Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Календарний план виконання роботи:

Тема : «Віртуальна кімната даних як сховище конфіденційних корпоративних документів»

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання роботи	Примітка
1	Отримання завдання на курсову роботу	11.04.2020	
2	Огляд джерел за темою роботи	16.04.2020	
3	Написання вступу та змісту	20.04.2020	
4	Розробка власного підходу	22.04.2020	
5	Проектування архітектури	24.04.2020	
6	Написання курсової роботи	26.04.2020	
7	Остаточне оформлення та слайдів	11.05.2020	
8	Захист курсової роботи	22.05.2020	

Зміст

АНОТАЦІЯ

У роботі розглянуто базові поняття про віртуальні кімнати даних, проведено аналіз конкурентів для визначення ключових недоліків та переваг існуючих продуктів. Проаналізовано основні типи архітектурних стилів для побудови віртуальної кімнати даних.

Спроектовано систему віртуальної кімнати даних з використання сучасних архітектурних стилів та підходів та побудовано безпечний мікро сервіс з AWS Lambda та ASP.NET Core.

Проаналізовано переваги на недоліки отриманого сервісу.

ВСТУП

Актуальність теми. Сучасний світ та людство в цілому активно та цілеспрямовано рухається в віртуальний світ, який спрощує багато процесів, як і в повсякденному житті так і бізнесі. Не став і винятком фінансовий світ, який потребує зберігати велику кількість конфіденційних документі в одному місті.

До появи Інтернету файли зберігалися в фізичних кімнатах даних. Ці кімнати потребували додаткових витрат, таких як оренда приміщень, велика кількість паперу, персонал та інше . Витрати також були з боку професіоналів, які хотіли скористатися такими кімнатами для перегляду фінансових документів. З боку користувачів потрібно було не тільки витратити час та кошти, для того щоб дістатися до кімнати, також потрібно було записуватися заздалегідь та переглядати велику купу документів.

З появою Інтернету, цей неафективний підхід був спростований і процес був перенесений до віртуальною кімнати даних.

Мета дослідження. Метою даної роботи є дослідження та використання сучасних архітектурних стилів для проектування програмного забезпечення для віртуальної кімнати даних.

Для досягнення мети передбачаються такі кроки:

- Дати визначення поняттю віртуальної кімнати даних
- Розглянути конкурентів та визначити їх переваги та недоліки
- Визначити ключових користувачів та сформувати ключові вимоги до віртуальної кімнати даних
- Проаналізувати існуючи архітектурні стилі та визначити ключові характеристики
- Побудувати архітектуру віртуальної кімнати даних

- Побудувати сервіс, який відобразить переваги вибраної архітектури
- Проаналізувати недоліки та переваги отриманого результату

Об’єкт дослідження. Концепція віртуальної кімнати даних як сховище конфіденційних корпоративних документів.

Предмет дослідження. Можливість використання нових архітектурних стилів для побудови віртуальної кімнати даних.

Джерела дослідження. Електронні ресурси та книжкові видання.

РОЗДІЛ І. ДОСЛІДЖЕННЯ ГАЛУЗІ

1.1 Віртуальна кімната даних та її можливості

Віртуальна кімната даних - це безпечне інтернет-сховище даних, яке використовується для зберігання та розповсюдження даних. Віртуальні кімнати даних використовуються, коли є необхідність суворої конфіденційності даних, з можливістю їх поширення великому колу користувачів.

Віртуальні кімнати даних мають багато переваг перед фізичними кімнатами даних. Такі переваги як цілодобова доступність даних з будь-якого пристрою та будь-якого місця в світі, безпека управління даними та економічна ефективність. Віртуальні кімнати даних використовуються в багатьох галузях, включаючи біотехнологію, інформаційні технології та телекомунікації, інвестиційна банківська справа, бухгалтерський облік, уряд, бізнес-брокери тощо. [1]

Як приклад використання можна взяти процес злиття та поглинання бізнесу. За рік у всьому світі відбувається близько 40 000 угод в сфері злиття та поглинання. Під час такої угоди нові потенційні інвестори повинні мати можливість переглядати фінансові документи компанії, що є предметом ділової угоди. Однак ім’я компанії або будь-які інші ідентифікаційні дані слід зберігати в таємниці, щоб уникнути спекуляцій. Зазвичай інвестиційні банки полегшують подібний тип угод, допомагаючи компаніям у створенні угод, належним чином організовуючи документи та запрошуючи потенційних інвесторів, які можуть бути зацікавлені в цьому. Рішення віртуальної кімнати даних використовуються лише для обміну фінансовими документами та співпраці над ними. Сама угода укладається, підписується та завершується поза віртуальною кімнатою даних.

1.2 Навіщо використовують віртуальну кімнату даних

Для всіх користувачів інтернету віртуальні кімнати даних стали нормою, змінивши колись кабінети або носії фізичних даних. Фізичні кабінети даних та фізичні носії даних мали свої обмеження та були трудомісткими та незручними для залучених сторін. З розвитком Інтернету, кімната фізичних даних та носіїв стала застарілою концепцією.

Нижче перераховані найпоширеніші види використання віртуальних кімнат даних:

- Корпоративне зберігання документів
- Дошка комунікацій
- Безпечний обмін документами
- Злиття або придбання компаній
- Збір коштів
- Первісне публічне розміщення
- Стратегічні партнерства
- Аудити
- Збільшення венчурного капіталу
- Управління інтелектуальною власністю[8]

В своїй роботі я хочу зосередитись на корпоративному зберігання документів, да на перше місце виходять такі поняття як безпека поширення документів для користувачів всього світу.

1.3 Аналіз конкурентів на ринку

Основними конкурентами на світовому ринку віртуальних кімнат даних є:

- iDeals Virtual Data Room
- Citrix ShareFile
- Merrill DataSite
- Box Virtual Data Room
- CapLinked
- Intralinks Dealspace
- Brainloop Secure Dataroom

Виходячи матриці конкурентного аналізу (рис. 1.1 та рис 1.2) можна зазначити, що основними елементами, яким надають найбільшу увагу є безпека середовища та інтелектуальний пошук завантажених документів. Тобто важливо не тільки завантажити документи, але важливо також мати методи, які допоможуть знайти документи, які користувач потребує.

Щодо ціни, то вона залежить від постачальника. Користувач сплачує базові функції, а додаткові потребують додаткових коштів. Деякі постачальники дають змогу купити необмежену кількість планів за передбачувану ціну.

Середовище віртуальних кімнат дуже конкурентне і розвивається швидкими темпами, що дає змогу користувачу вибирати саме ту систему, яка задовольняє конкретно його потреби.

Якщо спробувати описати віртуальну кімнату даних з точки зору бізнесу, то вона буди виглядати як вказано на рис. 1.3

Віртуальна кімната даних	Безкошновна спроба продукту	Цілодобова підтримка клієнтів	Підтримка різних платформ	Пітримка декількох мов	Просте завантаження файлів	Набір інструментів захисту	Просте управління	Розумний пошук	Зовнішній вигляд	Перетворення типів файлів	Завантаження зашифрованих файлів	Ціна
Ansarada	5	5	3	3	3	5	3	5	3	3	3	3
iDeals Virtual Data Room	5	1	5	5	5	5	5	1	5	0	0	3
Citrix ShareFile	5	3	1	1	1	5	2	1	1	1	0	3
Merrill DataSite	3	2	3	3	3	5	5	2	4	2	0	3
Box Virtual Data Room	3	3	4	3	2	5	4	1	4	2	5	3
CapLinked	3	2	0	3	2	4	3	2	2	4	4	3
Intralinks Dealspace	5	5	5	5	5	5	5	2	5	4	5	5
Brainloop Secure Dataroom	3	5	4	2	4	4	4	2	2	2	2	1

Рисунок 1.1 Матриця конкурентного аналізу

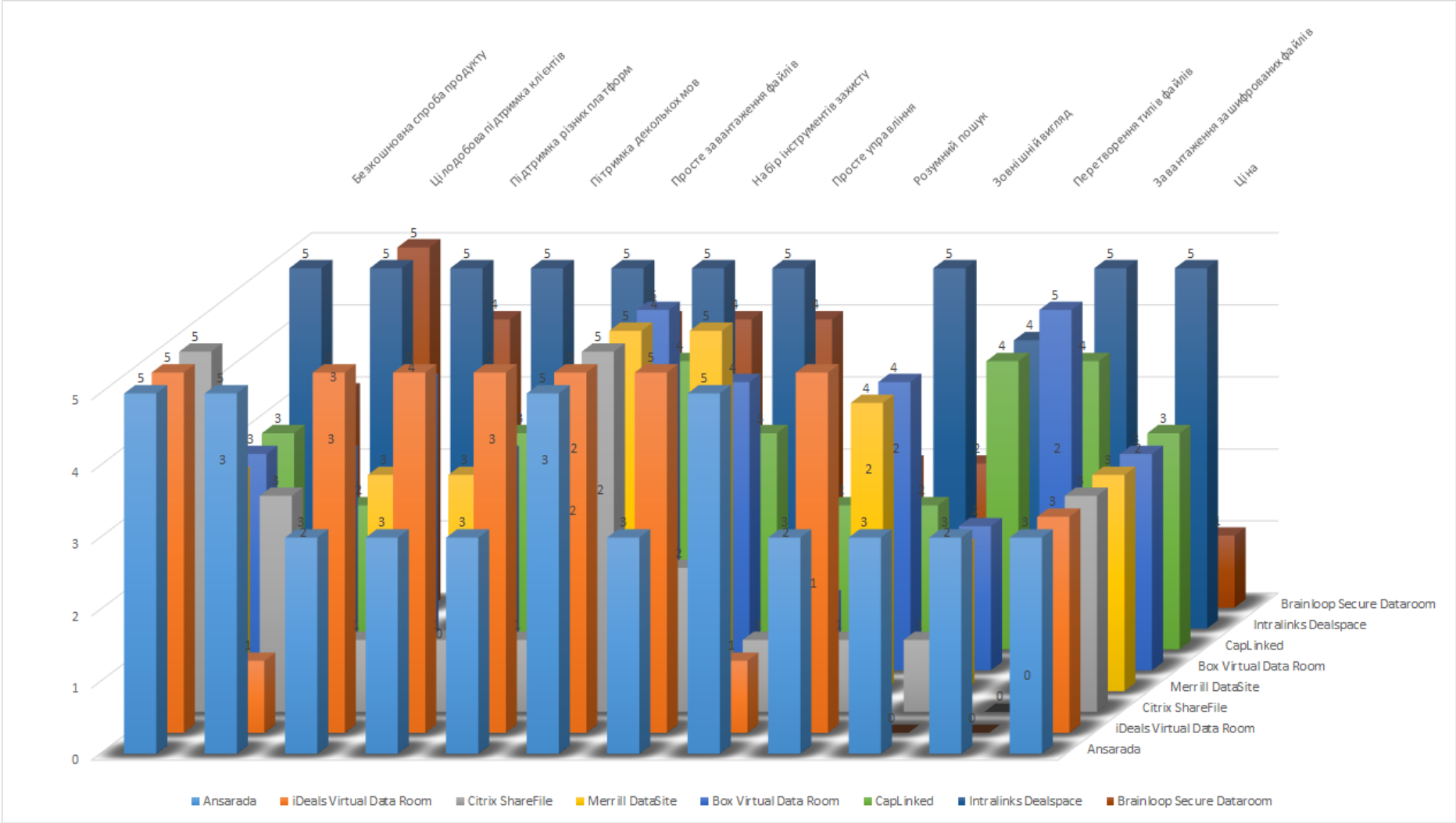

Рисунок 1.2 Матриця конкурентного аналізу

Рисунок 1.3 Бізнес модель

1.4 Ключові користувачі продукту та налаштування основного процесу

Для того щоб правильно сформувати наші вимоги до віртуальної кімнати, нам необхідно визначити ключових користувачів продукту та основний робочий процес. Ключовими особами, які приймають участь в процесі обробки корпоративних документів можуть бути:

- Фінансовий директор приватної компанії
- Аналітик фінансового відділу
- Кінцевий користувач, якому необхідно проглядати документи

Що до основного робочого процесу, він виглядає наступним чином:

1. Створення кімнати даних;

2. Завантаження та структурування документів приватної компанії найбільш ефективно;

3. Додавання нових користувачів до кімнати даних;

4. Надавання певним користувачам доступ до певних документів;

5. Запуск кабінету даних. Усі користувачі повинні отримувати доступ одночасно, щоб уникнути непорозумінь;

6. Контроль діяльності користувачів: хто є найактивнішим та зацікавленим а, хто не переглянув жодного документа тощо;

7. Закінчення і закриття кімнати даних.

Налаштування процесу відносно користувачів виглядає наступним чином:

1. Створіть кабінет даних -> Фінансовий директор приватної компанії

2. Завантажте та структуруйте документи приватної компанії найбільш ефективно -> Фінансовий директор приватної компанії, Аналітик фінансового відділу

3. Додайте нових потенційних користувачів до кімнати даних -> Фінансовий директор приватної компанії, Аналітик фінансового відділу

4. Надайте доступ до певних документів для певних потенційних користувачів -> Фінансовий директор приватної компанії, Аналітик фінансового відділу

5. Запустіть кабінет даних. Усі потенційні користувачі повинні отримати доступ одночасно, щоб уникнути непорозумінь -> Аналітик фінансового відділу

6. Мати можливість контролювати діяльність користувачів: хто є найактивнішим та зацікавленим; хто не переглянув жодного документа тощо -> Фінансовий директор приватної компанії, Аналітик фінансового відділу

7. Перегляд ієрархії документів, попередній перегляд, завантаження доступних файлів -> Фінансовий директор приватної компанії, Аналітик фінансового відділу, Кінцевий користувач, якому необхідно проглядати документи

8. Перегляд дозволених документів з моменту останнього входу на сайт -> Кінцевий користувач, якому необхідно проглядати документи

9. Закінчити і закрити кімнату даних -> Аналітик фінансового відділу[7]

РОЗДІЛ ІІ. ПРОЄКТУВАННЯ АРХІТЕКТУРИ

2.1 Аналіз існуючих архітектурних стилів

Архітектурні стилі дають поняття, як організувати програмний код. Це найвищий рівень деталізації, і він визначає шари, модулі високого рівня програми та спосіб взаємодії цих модулів і шарів один з одним, а також відносини між ними.

Приклади архітектурних стилів:

- Архітектура класної дошки
- Клієнт-серверна архітектура
- Архітектури, побудовані навколо бази даних (Database-centric architecture)
- Розподілені обчислення
- Подієва архітектура
- Front end та back end
- Неявні виклики (Implicit invocations)
- Монолітний застосунок (Monolithic application)
- Peer-to-peer
- Пайпи і фільтри (англ. Pipes and filters)
- Plugin
- Representational State Transfer
- Rule evaluation
- Пошуко-орієнтована архітектура
- Сервіс-орієнтована архітектура
- Shared nothing architecture
- Software componentry
- Space based architecture
- Структурована
- Багаторівнева [2]

З точки зору актуальності та поширеності, використовуються монолітна (рис. 2.1) та мікро сервісна архітектура (рис. 2.2)

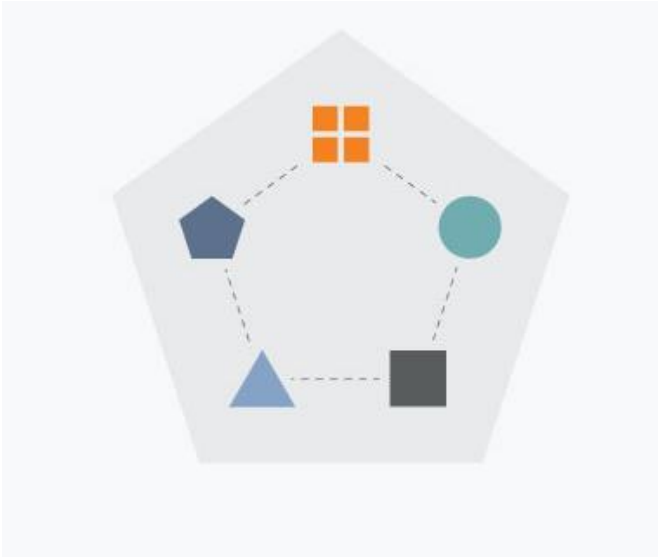

Рисунок 2.1 Монолітна архітектура

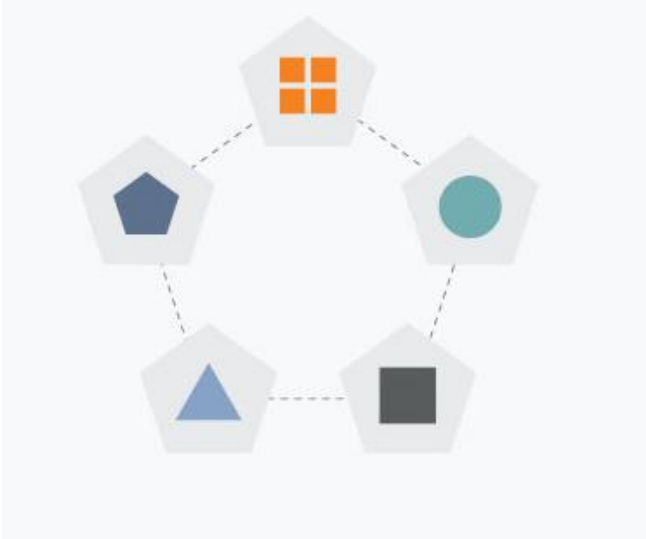

Рисунок 2.2 Мікро сервісна архітектура

Якщо порівнювати ці дві архітектури то можна сказати що монолітна архітектура найкраще підходить для простих, легких програмних продуктів. Якщо ми говоримо про комплексне рішення, яке потребую масштабованості та відмово стійкості, тоді вибір схиляться до мікро сервісної архітектури.

Ця архітектура дозволяє, щоб кожен компонент знаходився у власному виконавчому файлі або веб-сервісі. Це означає, що компоненти можуть працювати на виділеному обладнанні, а окремі частини функціонально можуть бути збільшені та використані з більшою точністю.[5]

2.2 Архітектура віртуальної кімнати даних

Для опису архітектури було вибрано 4+1 Модель. Модель опису архітектури програмних систем була запропонована Філіппом Крухтеном, в даний час професором інженерії програмного забезпечення в Університеті Британської Колумбії в 1995 році. Модель складається з логічного представлення (рис 2.1), діаграми процесу (рис. 2.2), діаграми розгортання (рис. 2.3) та фізичне представлення (рис. 2.4.).

Дані діаграми дають змогу розглянути архітектуру віртуальної кімнати даних з різних боків .[6]

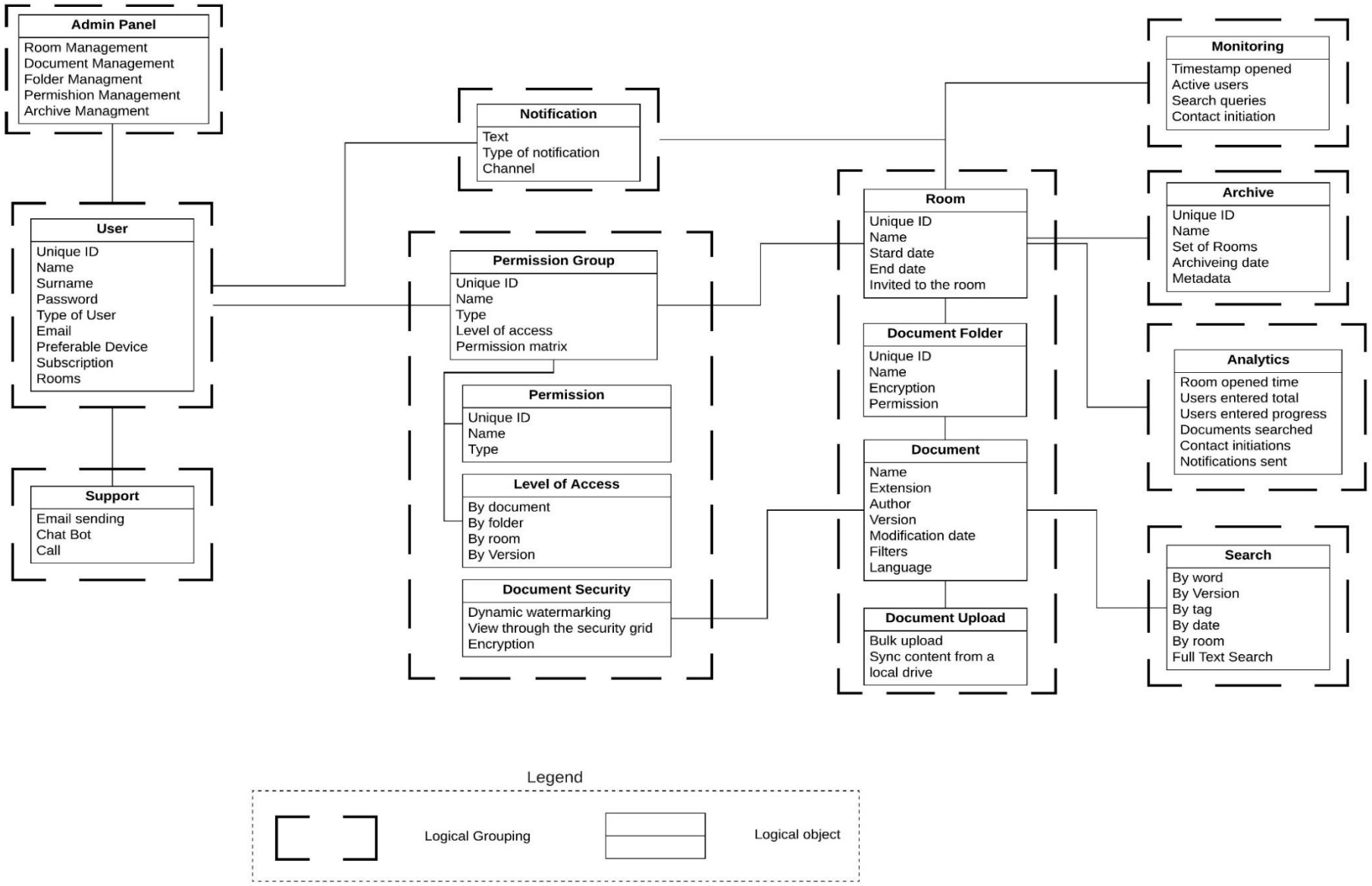

Рисунок 2.1 Логічне представлення

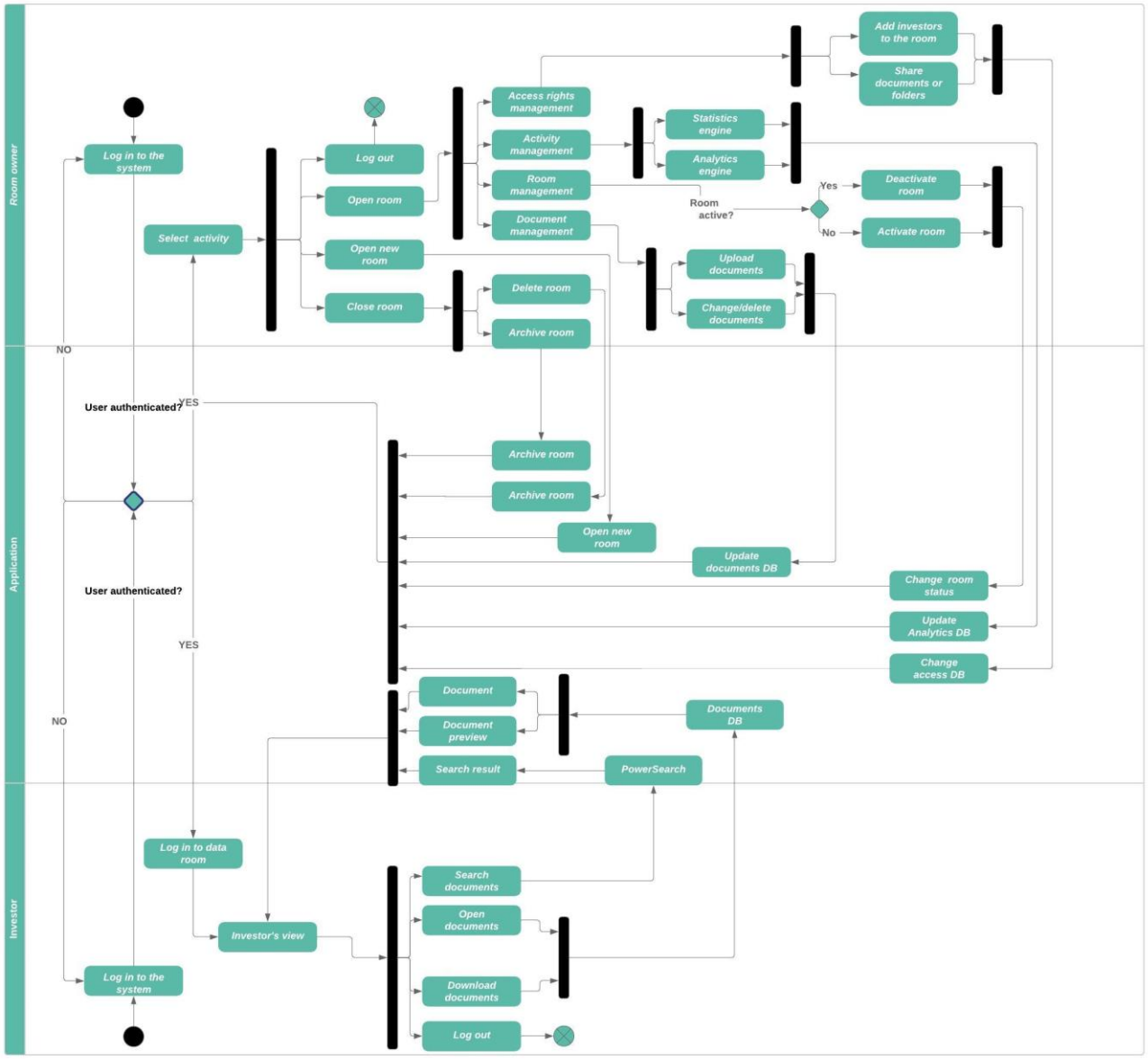

Рисунок 2.2 Діаграма процесу

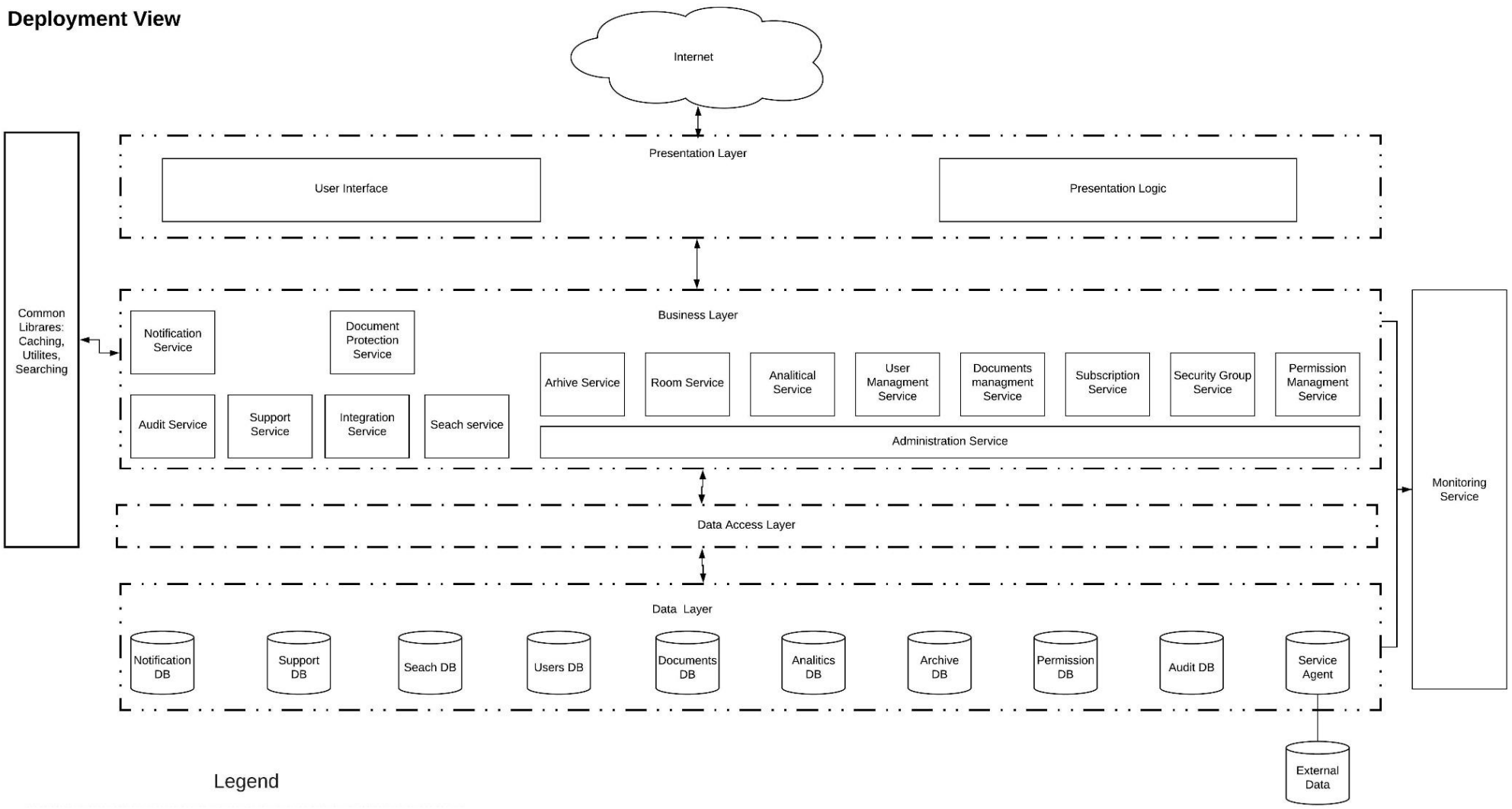

Рисунок 2.3 Діаграма розгортання

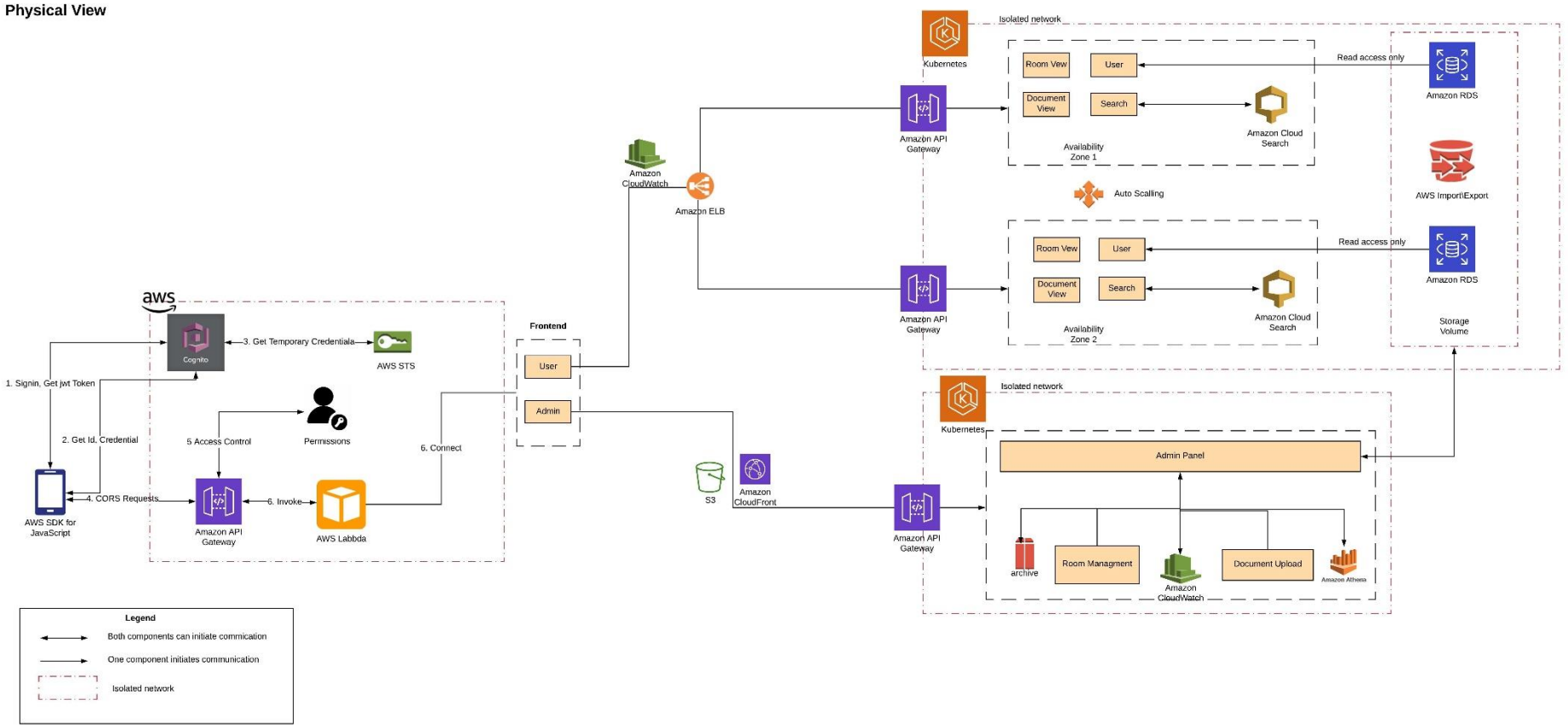

Рисунок 2.3 Фізичне представлення

РОЗДІЛ ІІІ. РОЗРОБКА МІКРОСЕРВІСУ ВИКОРИСТОВУЮЧИ AWS LAMBDA

3.1 Програмування мікро сервісу авторизації

Віртуальна кімната даних це комплексне рішення, яке повинно бути масштабоване та безпечне. Основним елементові віртуальної кімнати є безпека середовища, яка забезпечується в першу чергу авторизацією користувача.

Для того, щоб сервіс авторизації був універсальний та задовольняв потреби користувача був вибраний Okta фреймворк, який додає додатковий прошарок до безпеки сервісу.

Для програмування сервісу авторизації було встановлено Microsoft Visual Studio та використані облікові записи в AWS та Okta. Для роботи необхідно додати AWS Toolkit for Visual Studio та Newtonsoft.Json пакети на основі AWS Serverless Application (.NET Core). [4]

Для того щоб розпочати роботу с сервісом необхідно сформувати ключ для роботи с Okta та налаштувати його не сервісі (рис. 3.1).

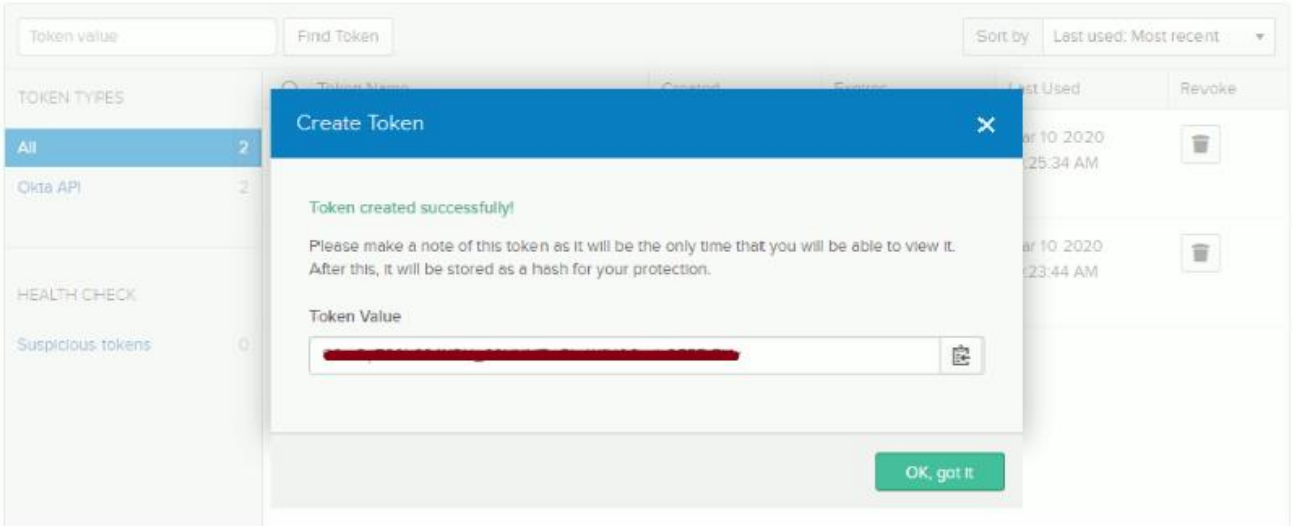

Рисунок 3.1 Налаштування ключа

Після необхідно зробити базові налаштування сервісу для роботи та приєднати Okta бібліотеку до сервісу (рис. 3.2).

```
1  namespace AuService.Model
2  {
       Ссылок: 3
3      public class OktaSetting
4      {
           ссылка: 1 | 0 исключения
5          public string Domain { get; set; }
           ссылка: 1 | 0 исключения
6          public string Token { get; set; }
7      }
8  }
9
```

Рисунок 3.2 Клас налаштувань сервісу

Далі для відправки даних необхідно налаштувати класс OktaAuRequest. Цей клас використовується для відправки даних через API Окта за адресою:

https: // {yourOktaDomain} / api / v1 / authn.

Екземпляр цього класу буде передавати Okta з іменем користувача та паролем.

```
namespace AuService.Model
{
    ссылка: 1
    public class OktaAuRequest
    {
        ссылка: 1 | 0 исключения
        public string OktaUsername { get; set; }
        ссылка: 1 | 0 исключения
        public string OktaPassword { get; set; }
    }
}
```

SerController відповідає тільки за одну дію авторизації. Метод на вхід приймає облікові дані, які передаються в мікро сервіс (рис. 3.3.).

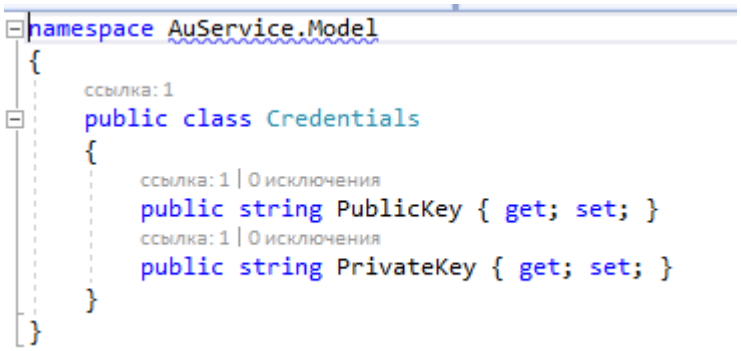

Рисунок 3.3. Клас довірених даних.

В модель також додані класи які відповідають за обробку сесій.

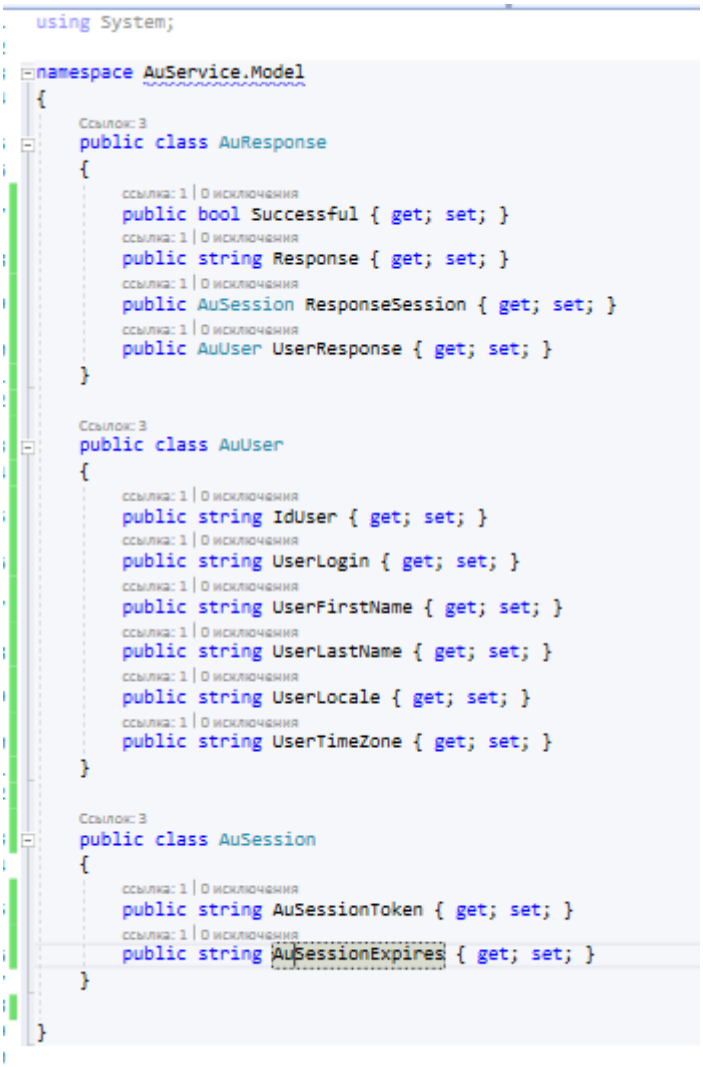

Рисунок 3.4. Допоміжні класи обробки

Для того щоб додати кроки автентифікації в наш основний сервіс було додано в контролер код авторизації, де передача даних проходить через JSON (рисунок 3.5).

```
var ret = new AuResponse();

string url = string.Format("{0}/api/v1/authn", this._settings.Value.Domain);

var data = new OktaAuRequest() { OktaUsername = input.PublicKey, OktaPassword = input.PrivateKey };

var json = new StringContent(JsonConvert.SerializeObject(data), Encoding.UTF8, mediaType: "application/json");
```

Рисунок 3.5. Ініціалізація

І останнім кроком базових налаштувань будо додано конфігураційні дані для сервісу в файл AppSettings.

Рисунок 3.6 AppSettings файл

Основна реалізація представлена в контролері (рис. 3.7) та передає базові дані про користувача, такі як ім’я, часовий пояс та його місце знаходження.

```
[HttpPost]
Ссылок: 0 | 0 запросов | 0 исключения
public async Task<AuResponse> Authenticate(Credentials input)
{
    var ret = new AuResponse();
    var url = string.Format("{0}/api/v1/authn", this._settings.Value.Domain);
    var data = new OktaAuRequest() { OktaUsername = input.PublicKey, OktaPassword = input.PrivateKey };
    var json = new StringContent(JsonConvert.SerializeObject(data), Encoding.UTF8, mediaType: "application/json");
    using (var client = new HttpClient())
    {
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(scheme: "SSWS", parameter: this._settings.Value.Token);
        using (var res = await client.PostAsync(url, json))
        using (var content = res.Content)
        {
            var responseJson = await content.ReadAsStringAsync();
            Console.WriteLine(responseJson);
            dynamic responseObj = JsonConvert.DeserializeObject(responseJson);
            if (responseObj.status == "SUCCESS")
            {
                _session = new AuSession()
                {
                    AuSessionToken = responseObj.sessionToken,
                    AuSessionExpires = responseObj.expiresAt
                };
                _user = new AuUser()
                {
                    IdUser = responseObj._embedded.user.id,
                    UserLogin = responseObj._embedded.user.login,
                    UserLocale = responseObj._embedded.user.locale,
                    UserTimeZone = responseObj._embedded.user.timeZone,
                    UserFirstName = responseObj._embedded.user.firstName,
                    UserLastName = responseObj._embedded.user.lastName
                };
            }
        }
    }
```

Рисунок 3.7 Основний метод авторизації

За допомогою Virtual Studio сервіс можна розгорнути на виділеному просторі в AWS та використовувати для авторизації користувачів віртуальної кімнати даних.

ВИСНОВКИ

Уході виконання курсової роботи було описано та охарактеризовано сторонні постачальники віртуальних кімнат даних, виявлено та проаналізовано основні характеристики та вимоги на які потрібно звернути увагу. Основна та найважливіша характеристика це безпека даних, яка забезпечується на різних рівнях програмного продукту. Також у роботі було розглянуто різні типи архітектурних стилів та підходів, проаналізовано їх переваги та недоліки в розрізі актуальності та масштабованості.

На основі проаналізованої інформації були висунуті основні вимоги до віртуальної кімнати даних, наведено чотири архітектурних схеми, які дали змогу представити продукт с різних точок зору.

Для пришвидшення процесу розробки було використано сторонні бібліотеки та API.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Quora [Електронний ресурс] – 2019. – Режим доступу: https://www.quora.com/What-is-the-future-of-virtual-data-rooms
2. Herbertograc [Електронний ресурс] – 2017. – Режим доступу: https://herbertograca.com/2017/07/28/architectural-styles-vs-architectural-patterns-vs-design-patterns/
3. Citeseerx [Електронний ресурс] – 2018. – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.683.3956&rep=rep1&type=pdf
4. Okta [Електронний ресурс] – 2019. – Режим доступу: https://developer.okta.com/blog/2019/03/21/build-secure-microservices-with-aspnet-core
5. Medium [Електронний ресурс] – 2019. – Режим доступу: https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59
6. MuleSoft [Електронний ресурс] – 2019. – Режим доступу: https://www.mulesoft.com/resources/api/microservices-vs-monolithic
7. AWS [Електронний ресурс] – 2020. – Режим доступу: https://eu-west-2.console.aws.amazon.com/console/home?region=eu-west-2#
8. SecureDocs [Електронний ресурс] – 2020. – Режим доступу: https://www.securedocs.com/blog/virtual-data-room-information