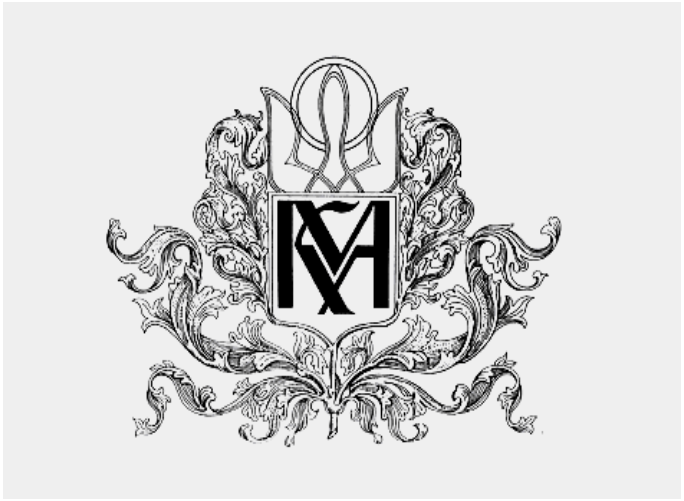
Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики



ПОБУДОВА АРХІТЕКТУРИ ВІРТУАЛЬНОЇ КІМНАТИ ДАНИХ

Текстова частина

магістерської роботи

за спеціальністю „Інженерія Програмного забезпечення”

Керівник магістерської роботи
к.н., доц. М.В. Почебут

_____
(підпис)

“____” _____ 2021 р.

Виконав студент Д.В. Міхов
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Декан кафедри інформатики, д.т.н., доц.

_____ А.М. Глибовець

“____” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на магістерську роботу

Студенту: Міхов Д.В. факультету інформатики 2 курсу МП

Розробити: Архітектуру програмного забезпечення для віртуальної кімнати даних

Зміст текстової частини до магістерської роботи:

Індивідуальне завдання

Вступ

1. Що таке Virtual Data Room та вибір підходу створення програмного забезпечення

2. Проектування архітектури застосунку для віртуальної кімнати даних

3. Реалізація вимог якісних атрибутів в цільовій архітектурі рішення

Висновки

Список літератури

Дата видачі “____” _____ 2021 р.

Керівник к.н., доц. М.В. Почебут _____

Завдання отримав Д.В. Міхов _____

Тема: ПОБУДОВА АРХІТЕКТУРИ ВІРТУАЛЬНОЇ КІМНАТИ ДАНИХ

Календарний план виконання роботи:

№ п/п	Назва етапу дипломної роботи	Термін виконання етапу
1.	Отримання завдання на дипломну роботу.	02.11.2020
2.	Огляд літератури за темою роботи.	15.11.2020
3.	Аналіз існуючих підходів до проектування архітектури	15.01.2021
4.	Визначення та опитування стейкхолдерів. Збір та класифікація вимог	25.02.2021
5.	Аналіз вимог, визначення ключових якісних атрибутів	15.03.2021
6.	Проектування архітектури	25.04.2021
7	Написання пояснювальної записки	04.05.2021
8.	Коригування роботи згідно з коментарями наукового керівника	15.05.2021
9	Остаточне оформлення пояснювальної записки та слайдів	26.05.2021
10.	Захист курсової роботи	18.06.2021

Студент Д.В. Міхов

Керівник М.В. Почебут

“_____”_____

Зміст

Анотація

У даній роботі розглянуті підходи до проектування архітектури програмного забезпечення, досліджен досвід адресування архітектурно значущих вимог до програмного забезпеченя. Визначен склад стейкхолдерів проекту та їх інтереси; виявлені якісні атрибути, які впливають на архітектуру рішення; запропоновані представлення відповідні до точок зору стейкхолдерів; розроблена архітектура рішення.

Ключові слова: ISO 42010, архітектура, SEI, RUP, SAD, TOGAF, ASR, QA, нефункціональні вимоги.

Вступ

Актуальність теми.

Ефективна архітектурна документація є однією з умов успіху в розробці програмного забезпечення. Універсальної форми або методології не існує, а кількість фреймворків проектування збільшується через розвиток технологій та еволюцію стандарта ISO 42010, який вперше з'явився в 2011 році. На еволюцію стандарта в тому числі впливають існуючи фреймворки – робоча група розробників стандарту досліджує близько 80 фреймворків, короткий опис яких є доступним на сайті розробників стандарту (http://www.iso-architecture.org/ieee-1471/afs/frameworks-table.html). Така різноманітність пояснюється тим, що задача стандарту забезпечити оптимальну трансформацію інтересів стейкхолдерів в зрозумілу архітектурну документацію, при цьому склад таких стейкхолдерів, складність задач, контекст існуючих рішень може суттєво впливати на вимоги до архітектури та її документування.

В даній роботі розглядається проект нового IT рішення, тобто не має обмежень від попередніх або існучих систим. Таким чином є можливість розробити власний полегшений підхід до проектування архітектури рішення.

Мета дослідження. Розробити архітектуру програмного забезпечення для віртуальної кімнати даних використовуючи власний підхід.

Завдання дослідження. Провести аналіз існуючих практик щодо проектування архітектури програмного забезпечення та стандарту ISO 42010. Дослідити існуючи тактики, які адресують вимоги якісних атрибутів, умови їх використання, та вибрати ті, які відповідають вимогам проекта.

Об'єкт дослідження. Розробка програмного забезпечення для віртуальної кімнати даних.

Предмет дослідження. Проектування архітектури програмного забезпечення.

Джерела дослідження. Електронні ресурси, електронні версії друкованої літератури, стандарт ISO 42010.

Наукова новизна одержаних результатів. Наукова новиза полягає в створенні і використанні власного підходу до проектування архітектури нового програмного забезпеченя.

Практичне значення одержаних результатів. Розроблена архітектура рішення використовується для розробки програмного забезпечення для віртуальної кімнати даних. Використаний підхід проектування архітектури відповідає критеріям стандарту ISO 42010 для фреймворків та може бути використаний для формального опису нового фреймворку.

1. Що таке Virtual Data Room та вибір підходу створення програмного забезпечення.

1.1. Навіщо ще одна VDR.

Віртуальна кімната даних, далі VDR – це IT рішення, що дозволяє організувати контрольований та захищений доступ до документів для певного набору користувачів. Ріешення VDR прийшли на заміну традиційному підходу організації кімнати даних, що є складовою частиною процесу due diligence (перевірка) при продажі бізнесу (merges&acquisitions або M&A транзакції) або вихід компаній на публічні ринки капіталу (IPO). З розвитком технологій та глобалізації бізнесу рішення VDR також почали широко використовувати в інших ситуаціях, які потребують надання доступу третім особам до великої кількості конфіденційних документів і де наслідки від витоку інформації можуть бути чутливими та комерційно значущими. Наприклад, при проведенні аудиту, оформленні патентів або ліцензій, складних судових процесах, тощо. Існує досить багато рішень щодо організації VDR, які відрізняються функціоналом, технічними характеристиками, комерційною моделлю, рівнем зручності для користувачів, засобами захисту інформації. Перелік найбільш поширених провайдерів надан в Додатку 1.

Проте більшість таких рішень з’явились в 2000-х та мають відносно обмежену зручність для колаборації при аналізі інформації, тому зважаючи на це і на власний досвід компанія, яка консультує з питань M&A, вирішила розробити рішення для власних потреб та для просування на ринку.

1.2. Огляд процесу розработки програмного забезпечення

Зважаючи на високі вимоги до безпеки та захисту інформації та розміри потенційних збитків у разі неналежного користування інформації, яка може бути розміщена в репозиторії провайдера сервісу, було вирішено організувати розробку рішення виходячи з найкращих практик побудови програмного забезпечення.

Брейнстормінг Аналіз		Повні затрати Аналіз ROI		Побудова рішення		План впроваждення Випуск для користування	
Ідея	**Вимоги**	**Бюджет**	**Дизайн**	**Розробка**	**Тестування**	**Впровадження**	**Підтримка**
	Вимоги бізнесу та користувачів Системні вимоги		Архітектура рішення Інтерфейси Сценарії для користувача		План тестування Тестове оточення Тести		План прийому Передача знань Моніторинг та підтримка

Виділяють 8 основних етапів життевого циклу розробки програмного забезпечення – дослідження, визначення вимог, побудова бюджету, побудова архітектури, розробка, тестування, впровадження, підтримка.

Фокус даної роботи на дизайні архітектури рішення та тим етапам, що передують в межах необхідних для розробки архітектури.

1.3. Чи так важлива архітектура і які бувають її види

Концепція архітектури програмного забезпечення вперше з’явилась на початку 1970-х в досліжденнях Edsger Dijkstra та David Parnas, які вказали, що структура комп’ятерних систем безпосередньо впливають на таких систем і вибір правильної структури надзвичайно важливий. Далі, в 90-х основні аспекти архітектури програмного забезпечення як дисципліни були визначені на підставі досліджень засобів опису архітектури, стилів, патернів, документацій та основних методів. При цьому слід зауважити, що не має консенсусу щодо визначення поняття архітектури, тому наведемо визначення декількох авторитетних джерел:

- ISO/IEC

  Архітектура програмного забезпечення – це фундаментальні концепти або властивості системи в своєму середовищі втілені в своїх елементах, відносинах між ними та принципами дизайну та розвитку.

- Software Engineering Institute (SEI)

  Архітектура програмного забезпечення – це сукупність структур необхідних для опису системи, які включають елементи програмного забезпечення, відносин між ними та властивості обох.

- Mictosoft Application Architecture Guide

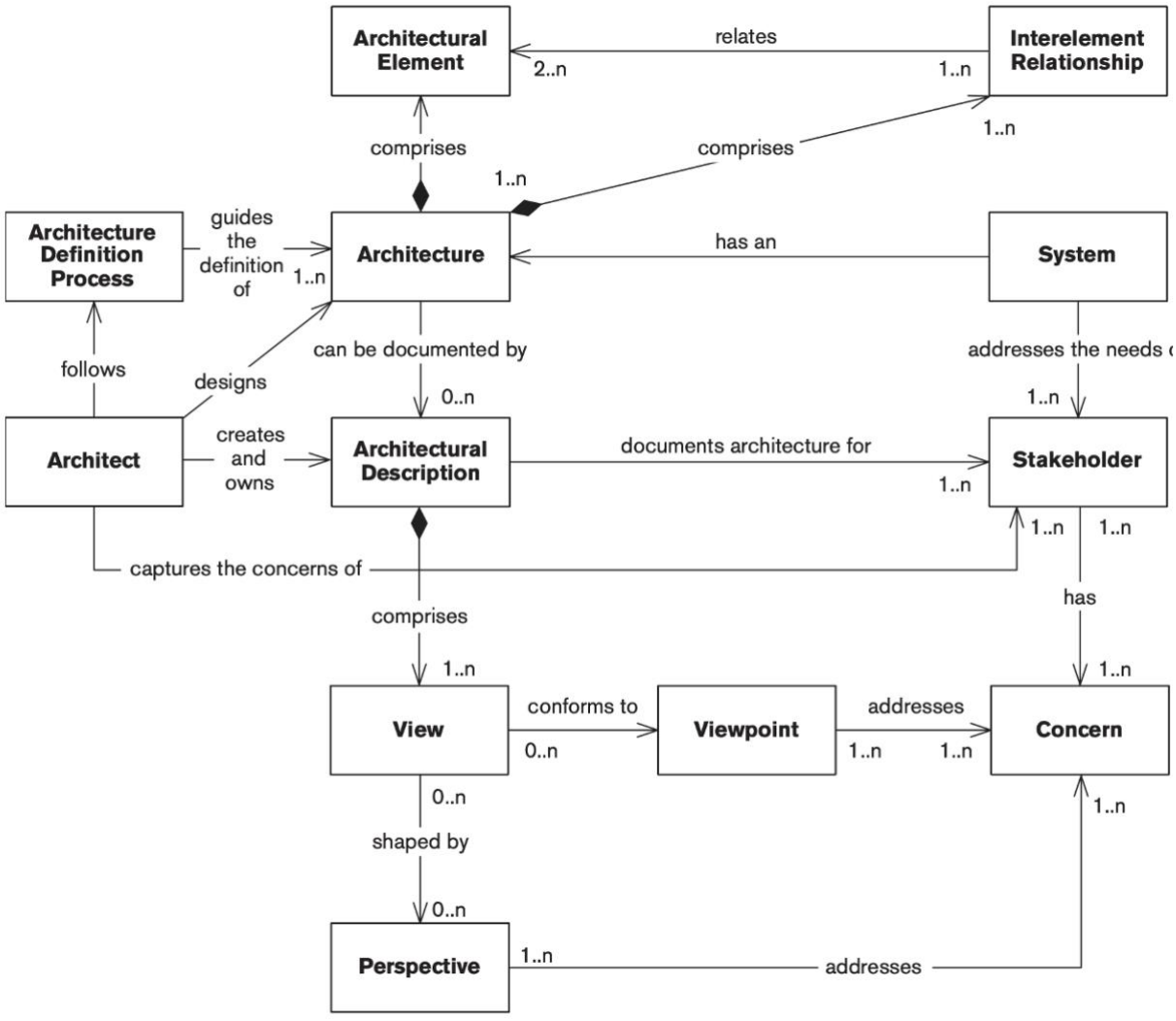
  Архітектура програмного забезпечення – це процес визначення структурованного рішення, яке відповідає усім бізнесовим, технічним та операційним вимогам. Він включає серію рішень заснованих на широкому колі факторів, де кожне таке рішення може мати значний вплив на якість.

Важливість архітектури виходить з того, що будь-яка система або програмне забезпечення має архітектуру не зважаючи на те описана вона чи ні. При цьому кожна система має лише одну архітектуру, яка може бути представлена різними засобами. Між іншим з цього випливає наступний

висновок – хоча кожна система має архітектуру, не кожна має архітектуру, яка якісно відображена в архітектурній документації.

Розвиток наукових та практичних досліджень стосовно створення архітектурної документації для програмноо забезпечення втілився в розробку стандарту ISO 42010, який визначає що таке архітектурний фрейворк та спеціальні вимоги для таких фреймворків. Стандарт ISO 42010 вперше був затверджений та опублікований в 2011 році. Наприкінці 2020 року був закінчен перегляд стандарту та направлен секретаріату ISO для затвердження.

Як зображено на Рисунку 1, архітектор програмного забезпечння виконує ключову роль в цьому процесі, використовуючи загальновживані принципи та фреймворки для ефективної коммунікації архітектурного рішення.



Риснок 1.

Архітектури також розрізняють за типами, хоча єдиної класифікації за типами не існує. Зауважимо, що серед інших виділяють IT архітектуру підприємства, яка має стратегічний характер та має на меті:

- оптимізувати наявні сервіси та системи,
- скоротити надлишковість,
- стандартувати сервіси та системи,
- покращити ефективність,
- скоротити IT ризики.

В той час, як архітектура рішення вирішує тактичні і конкретні задачі з фокусом на:

- успішне впровадження рішення,
- дизайн та розробку ефективного рішення,
- своєчасне виявлення та управління технічними ризиками,
- скорочення техничних ризиків.

1.4. Этапи підготовки архітектури рішення та вимоги до документації

Завданням побудови архітектури рішення є трансформація вимог до програмного забезпечення в архітектуру, що описує верхньорівневу структуру та ідентифікує її компоненти.

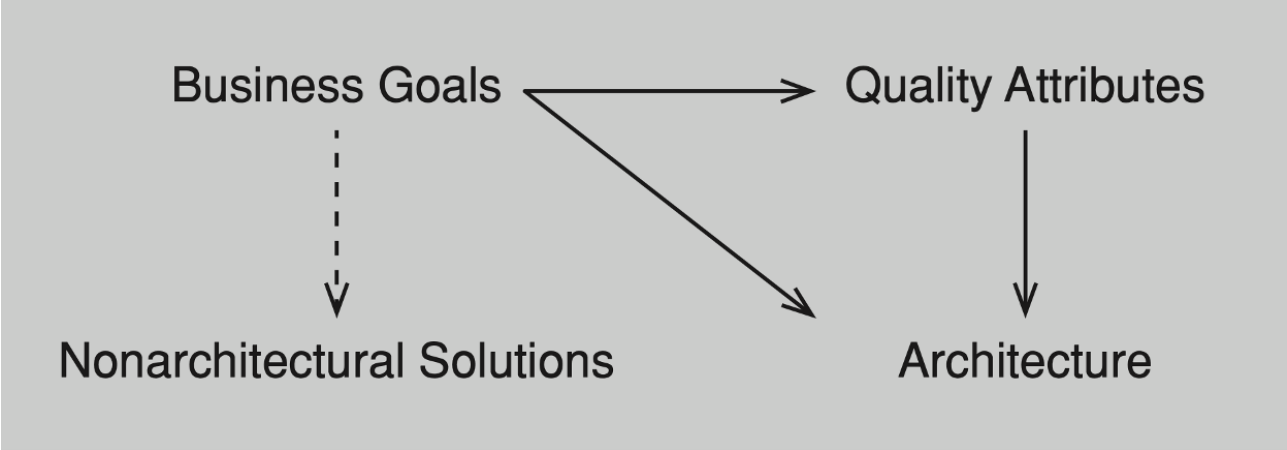


Рисунок 2.

Таким чином для побудови архітектури рішення необхідно

- проаналізувати бізнесові цілі та драйвери;
- визначити стейкхолдерів, їхні інтереси та вимоги;
- зібрати та пріорітизувати архітектурно важливі вимоги (architecturally significant requirements, ASRs);
- визначити головні якісні атрибути (quality attributes, QAs) та вибрати відповідні тактики для їх досяжнення;
- розробити цільову архітектуру та задокументувати її.

Щодо архітектурної документації, то існує велика кількість шаблонів Software Architecture Document (SAD), які допомогають якісно структурувати контент для потреб різних користувачів, в тому числі від визнаних організацій, як наприклад Software Engineering Institute (SEI), The Open Group, Rational Software Corporation (зараз входить до IBM).

- Структура SAD відповідно до шаблону **Software Engineering Institute** має такі елементи
    - Огляд документації
    - Опис вхідної інформації для архітектури
    - Представлення
        - Назва представлення

        - Зображення
        - Елементи каталогу
        - Контекстна діаграма
        - Можливості змін
        - Посилання на вхідні вимоги

- **The Open Group** запропонували The Open Group Architecture Framework (TOGAF) для визначення підходів до розробки, планування, впровадження и управління IT архітектурою підприємства. Структура SAD згідно TOGAF має наступний вид:
    - Резюме
    - Історія версій, зміст, задачі
    - Цілі, вимоги та обмеження, архітектурні принципи
    - Базовая архітектура
    - Обгрунтування підходів проектування
    - Цільова архітектура
    - GAP аналіз
    - Оцінка впливу

- **Rational Software Corporation** одні з перших запропонували ітеративний підход до розробки програмного забезпечення – Rational Unified Process (RUP). Відповідно до RUP документація по архітектурі складається з:

    - Введення
    - Зображення архітектури
    - Цілі та обмеження архітектури
    - Use-case представлення
    - Логічні представлення
    - Процесні представлення
        - Представлення розгортання
        - Представлення безпеки
        - Операції
        - Представлення даних
        - Продуктивність та масштабовність
        - Якість

Як можна побачити всі підходи вимагають ретельного аналізу цілей та обмежень при цьому різняться в наборі представлень, які презентують

спроектовану архітектур, тому при виборі структури документації серед іншого слід враховувати складність рішення та круг стейкхолдерів, які будуть використовувати документацію при подальшій розробці застосунку. В наступному розділі на базі дослідження вимог стейкхолдерів ми визначемось з переліком представлень, якими будемо описувати архітектуру для цього проекту, та викладемо результати проектування.

2. Проектування архітектури застосунку для віртуальної кімнати даних

2.1. Дослідження вимог стейколдерів, обмежень та визначення припущень.

На першому етапі визначили стейкхолдерів проекту, яких для цілей даної роботи об’єднали в наступні ролі – спонсор проекту, архітектор, розробники, тестувальникм, IT фахівці замовника, клієнтські менеджери замовника, топ-менеджмент замовника. На підставі спілкування з стейкхолерами та анкетування визначили основні функціональні та нефункціональні вимоги, перелік проблем які їх хвилють.

Функціональні вимоги:

- Веб застосунок для використання на комп’ютерах та мобільних пристріях
- Створення кімнати даних
- Завантаження документи та визначити їх структуру в зручний спосіб
- Підтримка форматів .jpeg, .docx, .pdf, .pptx, .xlsx
- Додавання користувачів з доступами різних рівней
- Мультіфакторна аутентифікація користувачів
- Відкриття доступу до кімнати
- Моніторинг активності користувачів
- Повідомлення у реальному часі
- Підтримка наступних каналів для повідомлень – email, sms.
- Закриття кімнати даних

Нефункціональні вимоги:

- Підтримка 1 міліона зареєстрованих користувачів з можливістю збільшити кількість
- Підтримка 10000 кімнат з можливістю збільшити кількість
- Підтримка останіх версій сучасних веб браузерів
- Система має бути доступна протягом робочих часів
- Всі документи мають зберігатися зашифрованими та передаватись через захищені канали
- Рівень захисту має бути на рівні вимог до захисту для фінансових установ

Обмеження:

- Хмарне рішення на AWS
- Мікросервісна архітектура
- Контейнерезація

Таблиця 1. Перелік інтересів груп стейкхолдерів

Стейкхолдер/ роль	**Фокус/Інтереси**	**Представлення**
Спонсор проекту	Проект може вийти за рамки бюджету	Project roadmap, business canvas, work-breakdown structure, product backlog
IT Архітектор	Не всі ключові вимоги вказані Захист даних	Функціональні та нефункціональні вимоги, ключові use cases, business canvas, опитувальники та чеклисти
Розробники	Складність реалізації вимог безпеки та продуктивності Неконтрольовані зміни до беклогу	Logical diagram, development diagram, deployment diagram, sequence diagram, backlog
Тестувальники	Чіткі вимоги, налаштування оточення для тестування може бути складним Суттєва невідповідність оточень для розробки, тестів та релізів	Deployment diagram, logical diagram, sequence diagram, результати тестування
IT фахівці замовника	Рішення може не відповідати вимогам безпеки	Project roadmap, functional decomposition, deployment diagram

Клієнтські менеджери замовника	Які конкурентні переваги Невпевнені в функціональності платформи Чи достатньо інструментів для клієнтської підтримки	Use cases, project roadmap, product backlog, functional decomposition
Топ-менеджмент замовника	Коли вийде на ринок	product roadmap, зворотній звя’язок від клієнтів, план продаж

2.2. Визначення переліку архітектурно важливих вимог (ASRs). Аналіз та пріоритезація якісні атрибутів (QA)

На підставі зібраних вимог були вибрані такі, що потенційно важливі з точки зору архітектури рішення, та оцінені з огляду рівня впливу на бізнес та архітектуру.

Таблиця 2. Оцінка впливу архітектурно значущих вимог

Тип	Вимога (ASR)	Вплив на бізнес (low/ medium/ high)	Вплив на архітек-туру (low/ medium/ high)
Функціональні	Доступ згідно ролей: • Admin – суперкористувач, може додавати/видаляти користувачів, конфігурувати систему, не має доступу до даних, не може редагувати матрицю дозволів • Client admin – завантажує документи, адмініструє матрицю доступів, користувачів VDR • Користувач VDR – доступ на читання • Менеджер клієнта – аналітика та звітність без доступу до VDR • Клієнтська підтримка – доступ до логів, алертів без доступу до VDR	H	M
	Інтуітивне адміністрування дозволів для доступу	H	L
	Масове завантаження	H	L
	Захищене зберігання інформації закритих кімнат	M	M

	Видача пошуку сперше пропонує найкращі результати	H	L
	Сторення нотанок та їх поширення серед інших користувачів	H	M
	Повідомлення про внесення змін до даних	H	L
	Вся інформація відображається з водяними знаками	H	L
Quality Attributes			
Продуктивність	Система має завантажувати документи швидше ніж 50 Mb/s	M	M
Доступність	Система має бути доступна 99.95% часу	M	H
	Система має автоматично відновлюватись	L	H
Захист	Користувач має бути аутентифікований	M	H
	Антивірусний захист забезпечує, щоб завантажені файли не заражені будь-якими відомими вірусами	M	M
	Доступ до документів визначається матрицею прав доступу	M	M
	Можливе налаштування доступу для певних IP адрес	M	L
	Вся інформація має бути зашифрованою	H	M
Масштабовність	Збільшення кількості одночасних сесій з 1000 до 100 000 не має погіршити продуктивність	H	H
	Система має забезпечити одночасний доступ до одного документу/кімнати для 1000 користувачів	M	M
Об'єм	Cloud native AWS application	M	M

	Мікросервісна архітектура	L	M
	Microsoft stack	M	M

В результаті пріоритизації вимог при проектуванні архітектури вирішено зосередитись на наступних якісних атрибутах:

- Масштабовність (Scalability)
- Доступність (Availability)
- Продуктивність (Performance)
- Безпека (Security)

На Рисунку 3 дерево корисності відображає визначені архітектурно значущі вимоги та метрики і їх цільові значення, які мають бути забезпечені при побудові рішення.

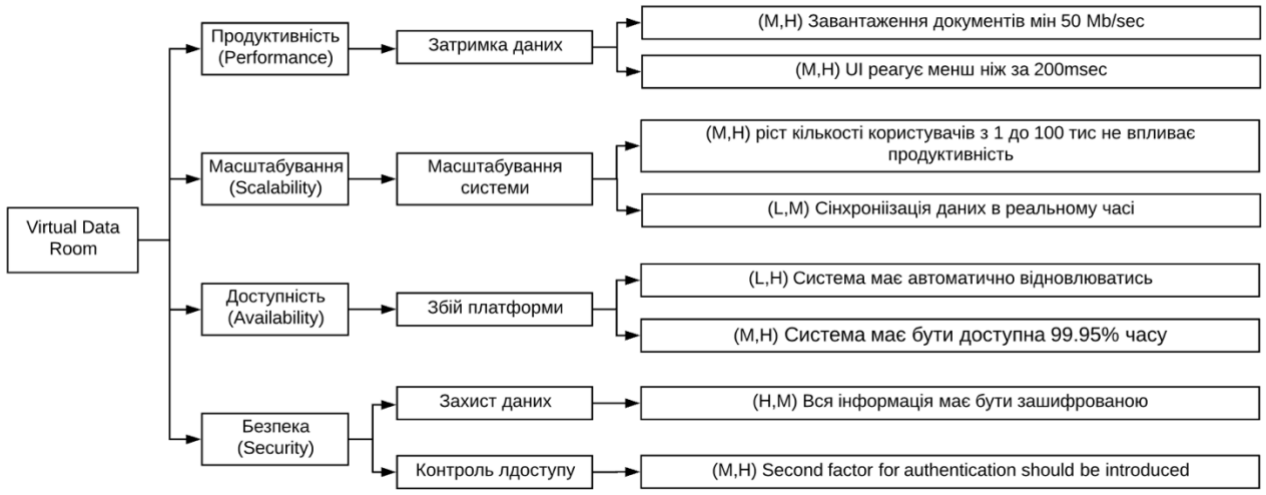


Рисунок 3. Дерево корисності

2.3. Огляд та вибір представленнь для опису рішення

Для адресування проблем та інтересів, які є пріоритними для стейкхолдерів, в стандарті ISO 42010 існує поняття «точки зору» визначене як робочий продукт, який встановлює правила конструювання, інтерпретації та використання архітектурного представлення для структурування певних системних інтересів. При цьому кожній точки зору відповідає одне представлення. Кожний фреймоврк побудови архітектурної документації має свій набір представлень і відповідних точок зору. Наприклад, фреймворк RUP включає такі представлення:

- Use case view (використання)
- Logical view (логічний)
- Implementation view (реалізації)
- Process view (процесний)
- Deployment view (розгортання)

Розанскі і Вудс в своїй роботі визначили 6 найбільш вживаних представлень та проаналізували їх релевантність для різних типах систем (дивись Таблицю 3):

- Контекстне (Context)– зображує відносини, залежності та взаємодії між системою та оточенням
- Функціональне (Functional) – зображує функціональні елементи системи при її виконанні, їх відповідальність, інтерфейси та є ключовим при проектуванні більшості архітектур.
- Інформаційне (Information) – описує як система зберігає, обробляє, управляє та розповсюджує інформацію для того, щоб відповісти на ключові питання щодо змісту, структури, затримки, посилань та міграції даних.
- Паралельності (Concurrency) – зображує паралельні структури системи та пов’язує з ними функціональні елементи для того, щоб чітко ідентифікувати частини системи, які можуть виконуватись паралельно та як вони координуються та контролюються.
- Розробки (Development) – описує архітектуру, яка підтримує процес розробки програмного забезпечення, та представляє інтерес для тих учасників, які залучені у побудові, тестуванні, супроводі системи.
- Розгортання (Deployment) – описує оточення де система буде розгортатися та залежності, які система має від елементів такого оточення.

- Операційне (Operational) – зображує як система буде використовуватись, адмініструватись та підтримуватись в виконавчому оточенні.

Таблиця 3. Релевантність застосування представлень

	OLTP Information System	Calculation Service/ Middleware	DSS/MIS System	High-Volume Web Site	Enterprise Package
Context	High	Low	High	Medium	Medium
Functional	High	High	Low	High	High
Information	Medium	Low	High	Medium	Medium
Concurrency	Low	High	Low	Medium	Varies
Development	High	High	Low	High	High
Deployment	High	High	High	High	High
Operational	Varies	Low	Medium	Medium	High

Рішення для віртуальної кімнати даних можна віднести до групи High-Volume Web Site, для якої пріоритеними є наступні представлення – функціональне, розоробки та розгортання, які будуть представлені в наступних розділах.

2.3.1. Логічний представлення (Logical view)

Логічний представлення описує яку функціональність для кінцевого користувача забезпечує система та складається з наступних елементів

- Admin panel – підтримує функції адміністратора
- User – керує користувачами системи
- Support – забезпечує функції щодо підтримки користувачів
- Notification – керування повідомленнями
- Permission management – управління доступами
- Room and Document management – управління документами
- Monitoring – моніторинг дій в системі
- Archive – створення архіву
- Analytics – бізнес аналітика
- Search – пошукові інструменти

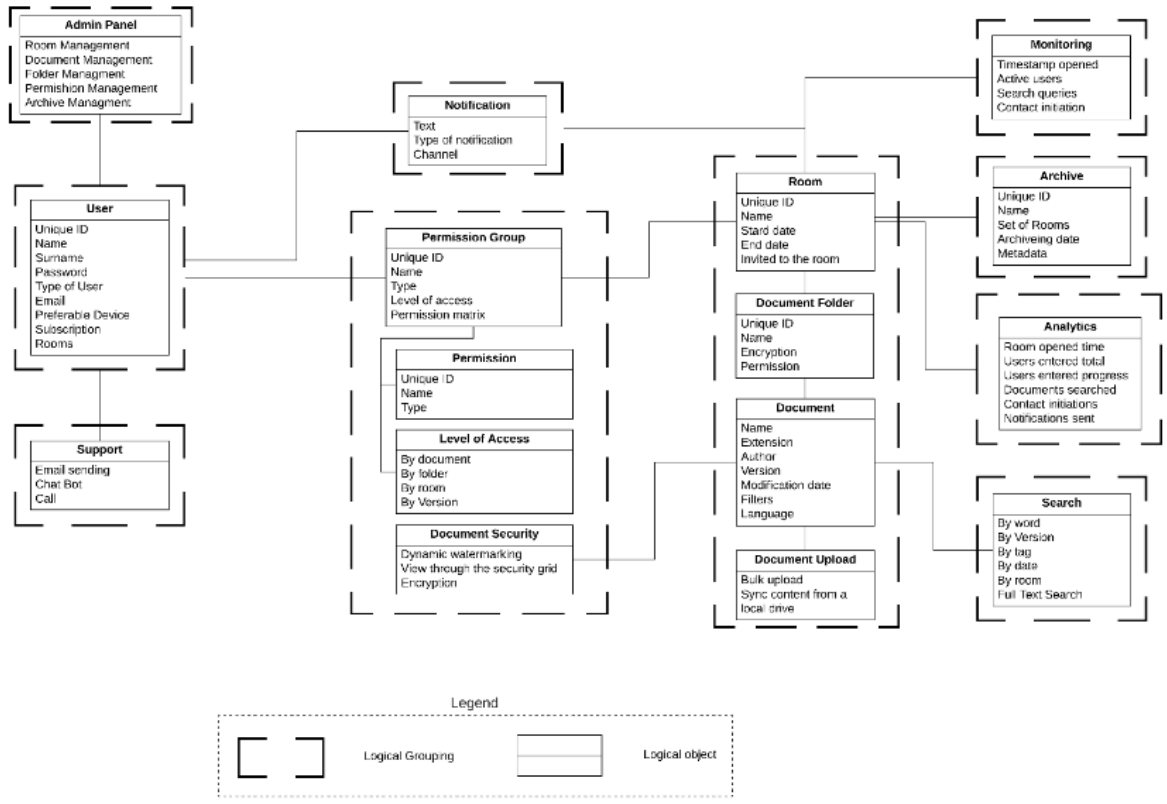


Рисунок 4. Логічне представлення

2.3.2. Представлення розробки (Development view)

Представлення розробки демонструє систему як її бачать програмісти та показує організацію системи на рівні модулів. Архітектура рішення віртуальної кімнати даних складаєть з додаткових завнішніх сервісів та 5 шарів:

- Data Layer
- Data Access Layer
- Business Layer
- Presentation Layer
- External Services: External Libraries and Monitoring system

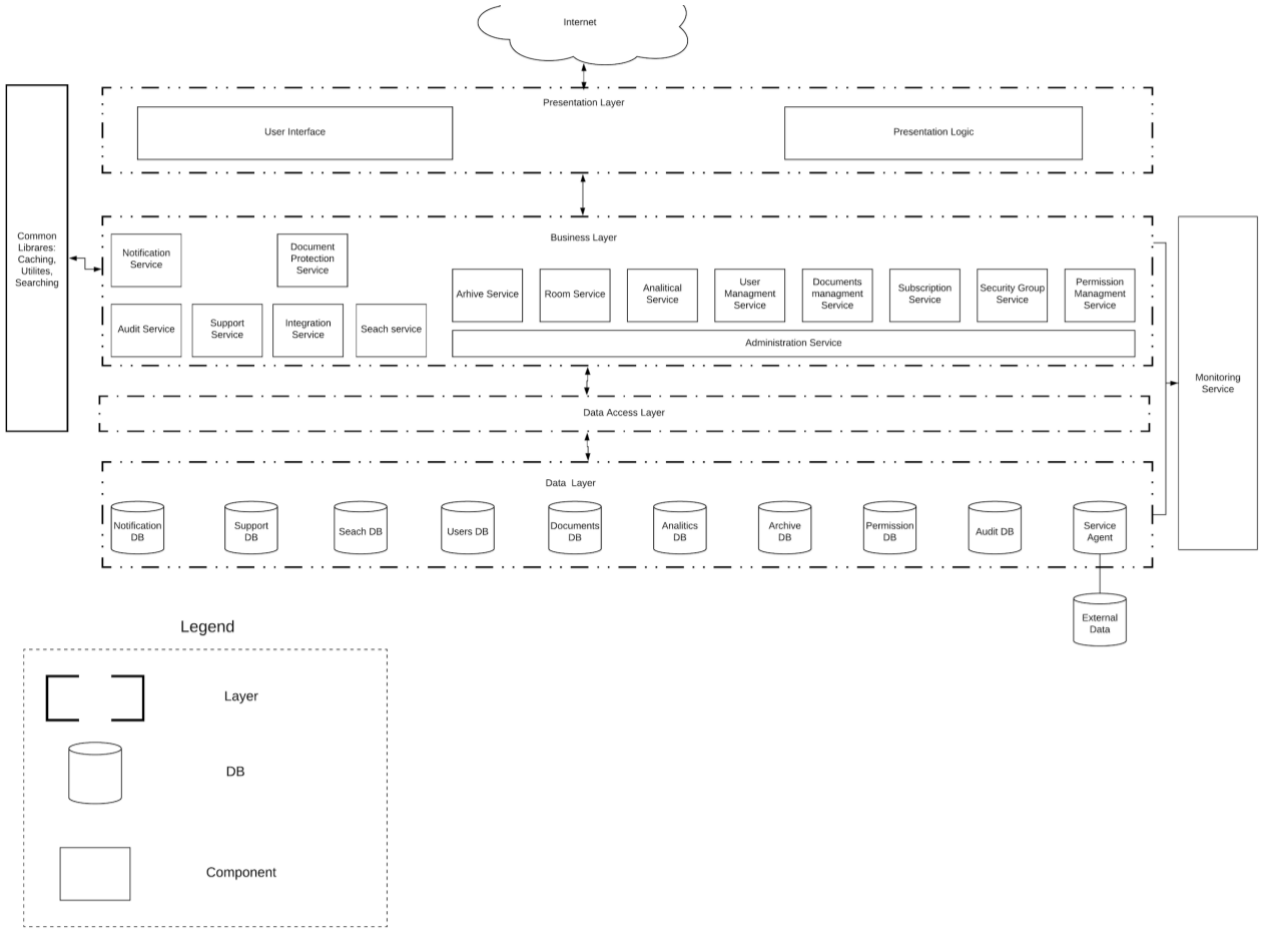


Рисунок 5. Представлення розробки

2.3.3. Представлення розгортання (Deployment view)

З урахуванням вимог щодо використання хмарних сервісів AWS вибран сценарій розгортання інфраструктури на базі сервісу Amazon CloudFormation, який дозволяє створити інфраструктуру в Amazon.

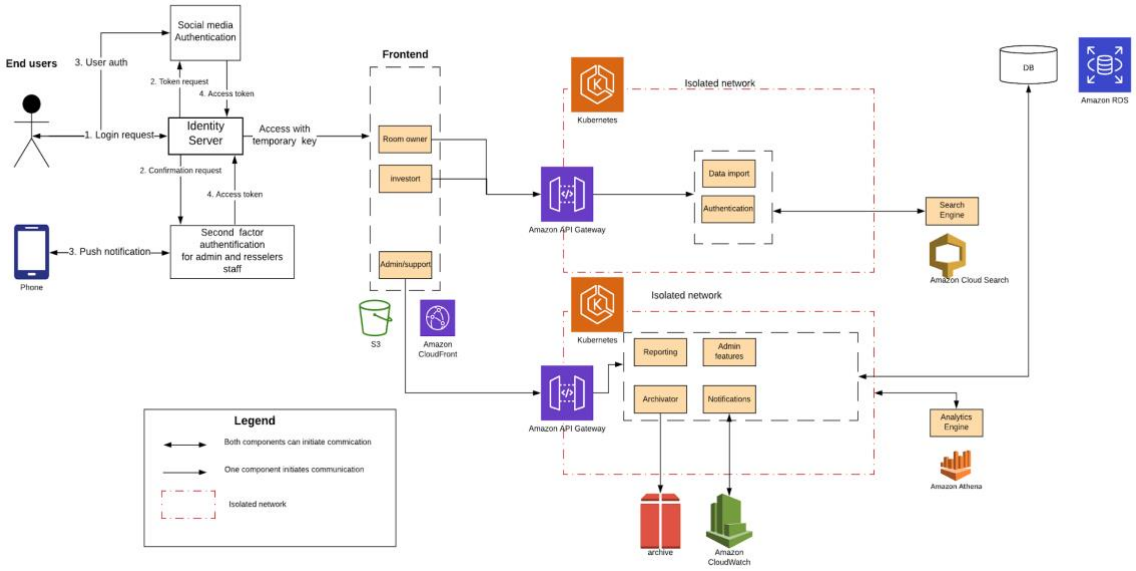


Рисунок 6. Представлення розгортання

3. Реалізація вимог якісних атрибутів в цільовій архітектурі рішення

Раніше були визначені якісні атрибути та метрики, які дозволяють кількісно міряти та тестувати відповідні атрибути на відповідність до цільових значень. Тобто за допомогою таких атрибутів можна визначити наскільки система задовільняє вимогам стейкхолдерів. В цьому розділі ми розглянемо сценарії рішень, які дозволяють досягти тих чи інших якісних показників, та визначемо дизайни відповідних рішень за допомогою поширених архітектурних тактик.

3.1. Доступність (Availability) є ознакою застосунку, що вказує на його спроможність виконувати свої задачі, та також пов’язана з таким якісними атрибутами, як захист, продуктивність та стійкість то збоїв. Тактики щодо доступності можуть бути сгруповані наступним чином – виявлення збою, відновлення після збою та запобігання збою. Більш детальний перелік тактик представлен на Рисунку 7.

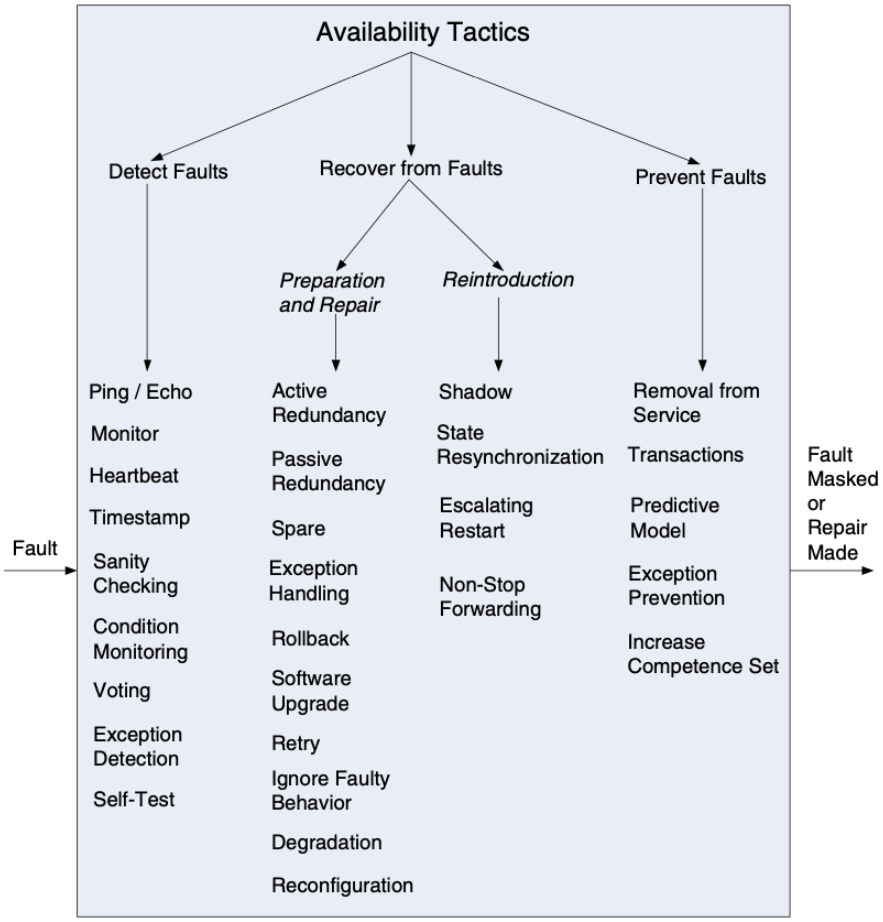


Рисунок 7. Тактики досягнення доступності

Можна сказати, що доступність будується на фундаменті атрибуту надійності (reliability) та здатності відновитися у разі збою. В свою чергу надійність – це ознака системи бути працездатною протягом часу, яка вимірюється ймовірністю, що система не буде мати збоїв та буде працювати згідно вимог протягом визначеного часу.

Для досягнення надійності та доступності застосували патерн event sourcing в комбінації з мікросервісним стилем:

Мікросервісна архітектра виключає збій системи через збій окремого елементу (single point of failure) та забезпечує плавне згортання системи у разі збоїв. Зважаючи на розподілений характер системи побудованої на мікросервісах координація та виконання довгих бізнес-процессів требує додаткових зусиль.

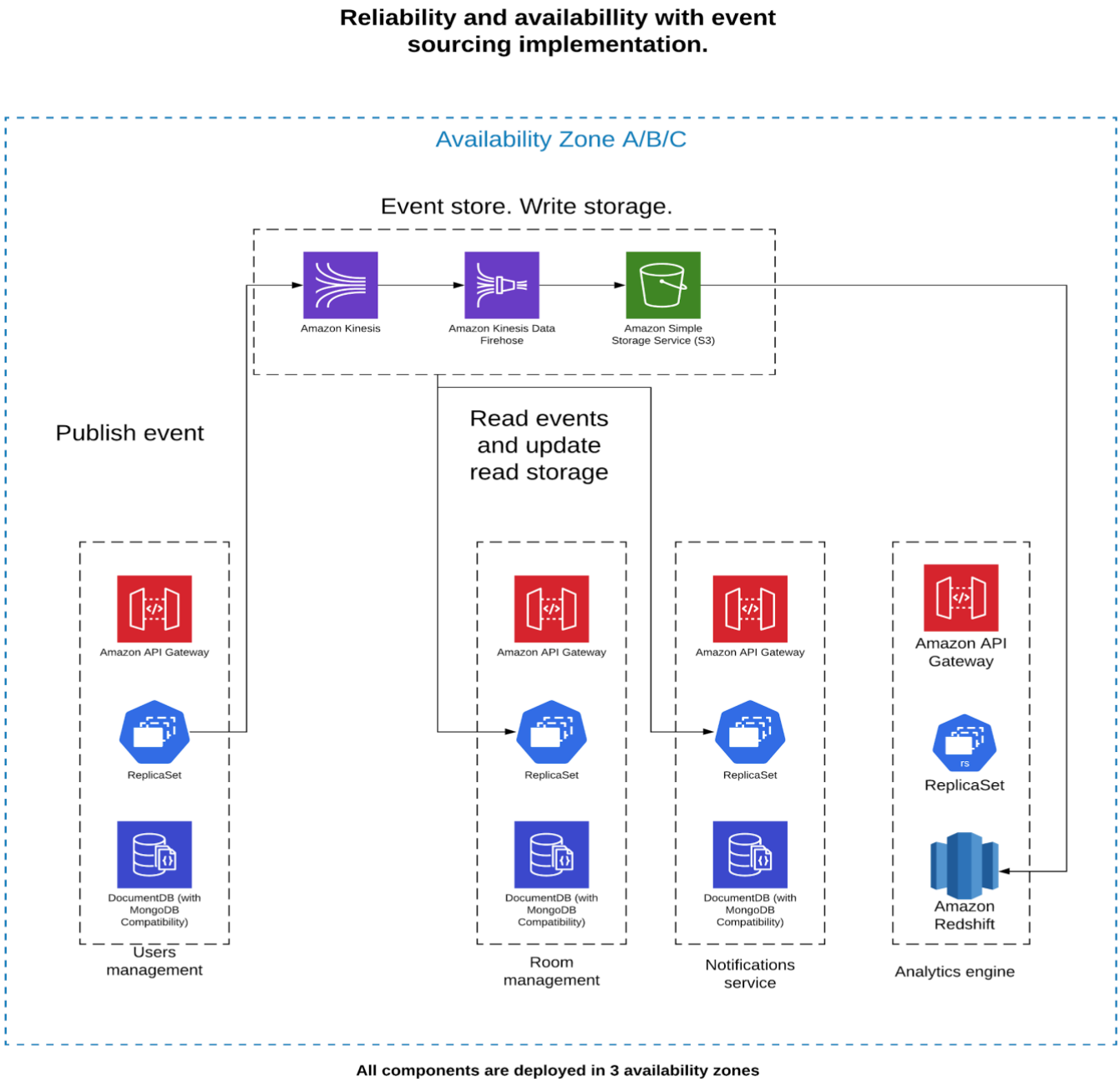


Рисунок 8. Імплементація event-sourcing

Event-sourcing патерн застосований до розподілених систем гарантує кінцеву узгодженність системи та надійну обробку довготривалих процесів.

Вся активність системи буде зберігатися як незмінні події в найбільн найдійному AWS сховище – S3, що дозволить відновити стан системи в будь який час.

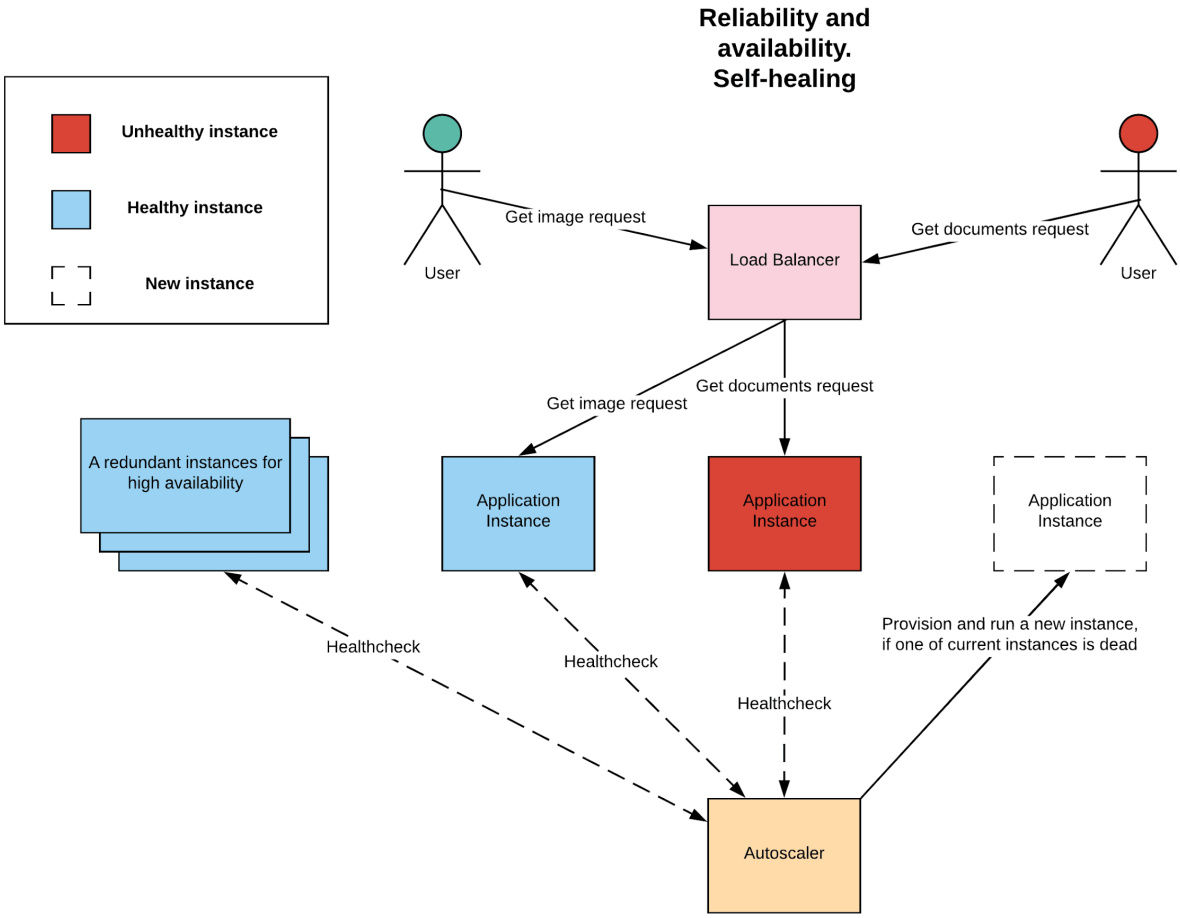


Рисунок 9. Імплементація автоматичного відновлення

Контроль стану системи та автомасштабування забезпечують можливість автоматичного відновлення системи – автоматичний старт нового екземпляру застосунку замість того, що має збої, деталі на Рисунку 9.

Сховище Amazon S3 забезпечує 99.999999999% довговічність даних, що робить його ідеальним вибором для зберігання подій.

Amazon DocumentDB – NoSQL сховище, що має широкі можливості для фізичного розбиття даних застосунку на розділи. Відпвідно вся інформація, що стосується одної кімнати даних, має зберігатися в одному розділі. У разі якщо

буде якийсь збій у DocumentDB, це може зашкодити лише незначній кількості кімнат з даними.

Застосування event sourcing патерну для рішення побудованого на потоковому стеку Amazon Kinesis дозволяє досягти надійності. Система буде кінцево узгодженою у разі, якщо деякі сервіси зазнають збої. Збереження стану системи на S3 робить втрату даних майже неможливою. Завдяки використанню компонент AWS загальна доступність системи дорівнює 99.63% в одній зоні доступності (дивись Таблицю 4). 4Розміщення всіх компонентів системи в декількох регіонах збільшує доступність до 99.99%.

Таблиця 4. Перелік компонентів AWS та їх SLAs

Назва	**Доступність в одному регіоні**
API Gateway	99.95%
DocumentDB	99.99%
Amazon EKS	99.9%
Amazon Kinesis	99.9%
Amazon Kinesis Firehorse	99.9%
Amazon S3	99.99%
Загальна доступність	99.63%

3.2. Масштабовність та продуктивність (Scalability & Perfomance)

Продуктивність (performance) в першу чергу стосується часу та можливостям програмного забезпечення витримати вимого своєчасної обробки процесів та запитів. Для вебзастосунків кількість запитів може сягати дестяки мільйонів, тому продуктивність як правило є центральним якісним атрибутом. Метриками продуктивності можуть бути пропускна здатність, час реакції, затримка. Рішення, що дозволяють досягти продуктивності, поділяють на 2 групи – контроль запитів на ресурси та управління ресурсами, дивись Рисунок 10.

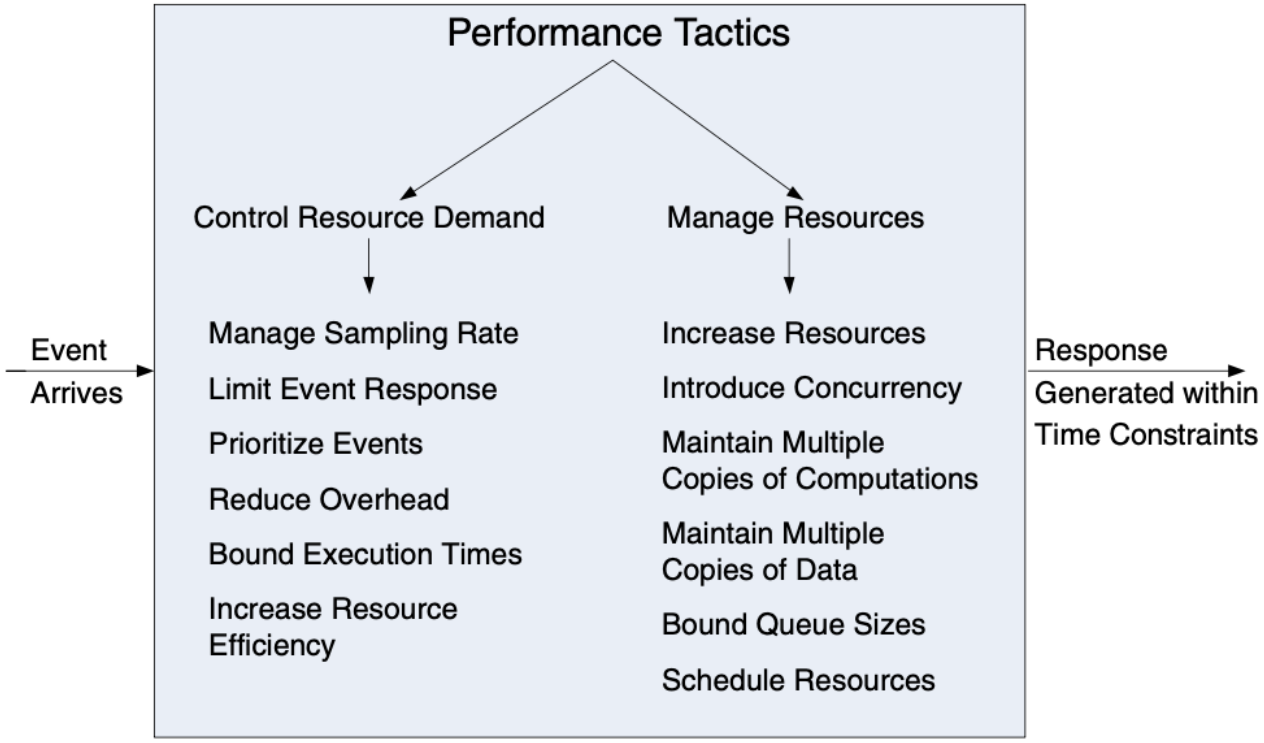


Рисунок 10. Тактики досяжнення продуктивності

Продуктивність тісно пов’язана з масштабовністю, яка часто асоціюється з хмарними обчисленнями, лямбда архітектурами, мікросервісами. Існує багато визначень масштабовності, але їх поєднує те, що вони в свій спосіб описуть властивість системи витримати збільшене навантаження без погіршеня продуктивності.

Ебот і Фішер в своій роботі виділили 50 правил масштабовності, які найбільш застосовані при побудові інформаційних вебсистем. Правило 19 говорить, що послаблючи ACID властивості узгодженості можна досягти значної гнучкості при масштабуванні. Таким рішенням є BASE архітектркура (Basically Available, Soft state, and Eventually consistent), яка дозволяє базам даних стати узгодженою з часом, що часто використвується в noSQL базах даних.

Для досягнення показників масштабовності та продуктивності використали патерни CQRS та event sourcing. Мікросервісна архітектура дозволяє масштабувати будь-який мікросервіс. Крім того, ми відокремили операції читання та запису, що дозволяє розділити відповідні запити, наприклад, коли відкривається кімната даних і система отримує багато запитів на читання від різних користувачів. В той же час event sourcing дозовляє не зберігати всі стани кімнати, а відновити іх при необхідності застосовуючи відповідну послідовність подій.

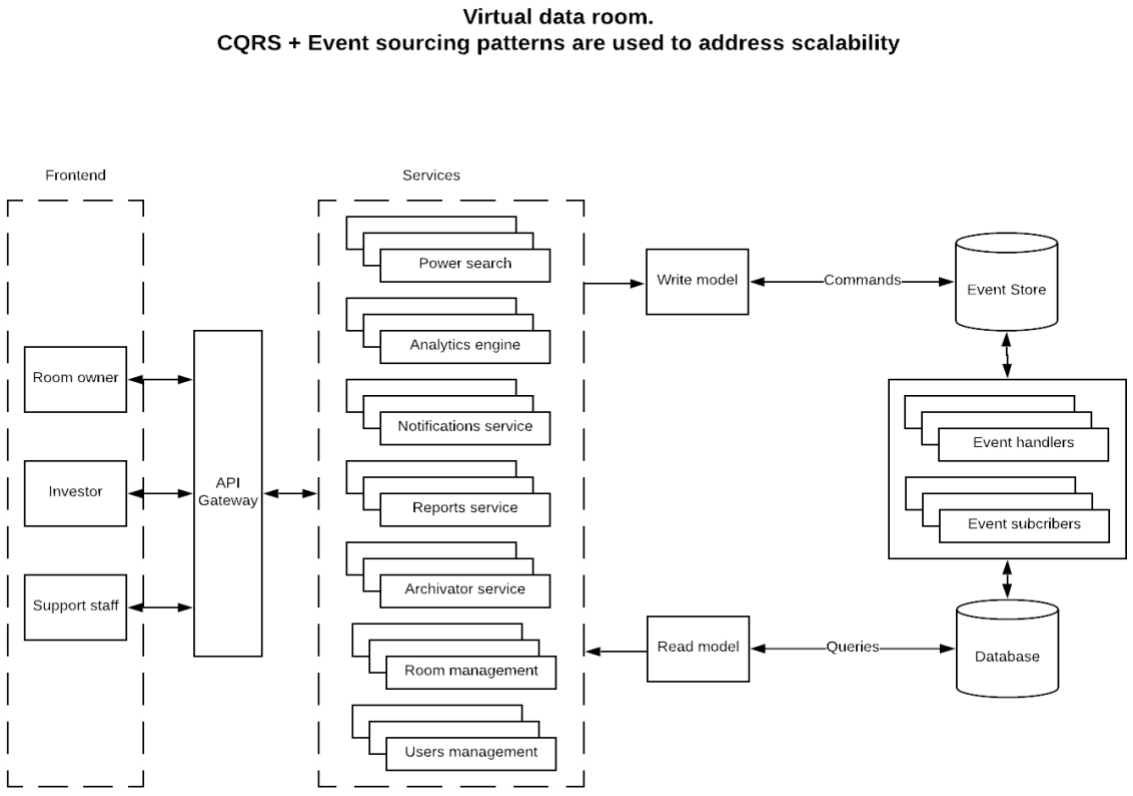


Рисунок 11. Імплементація масштабовності

3.3. Безпека (Security)

Безпека – це міра можливості системи захистити інформацію від несанкційного доступу в той час коли, авторизовані користувачі та системи мають такий доступ. Найбільш простий підхід до опису захисту має три характеристики – конфіденційність (confidentiality), цілісність (integrity), доступність (availability), або CIA.

Software Engineering Institue дає широкий перелік тактик, які можуть бути застосовані для досяжнення нефункціональних вимог до захисту, які поділяються на чотири групи – виявлення атаки, супротив атаці, відповідь на атаку та відновлення після атаки.

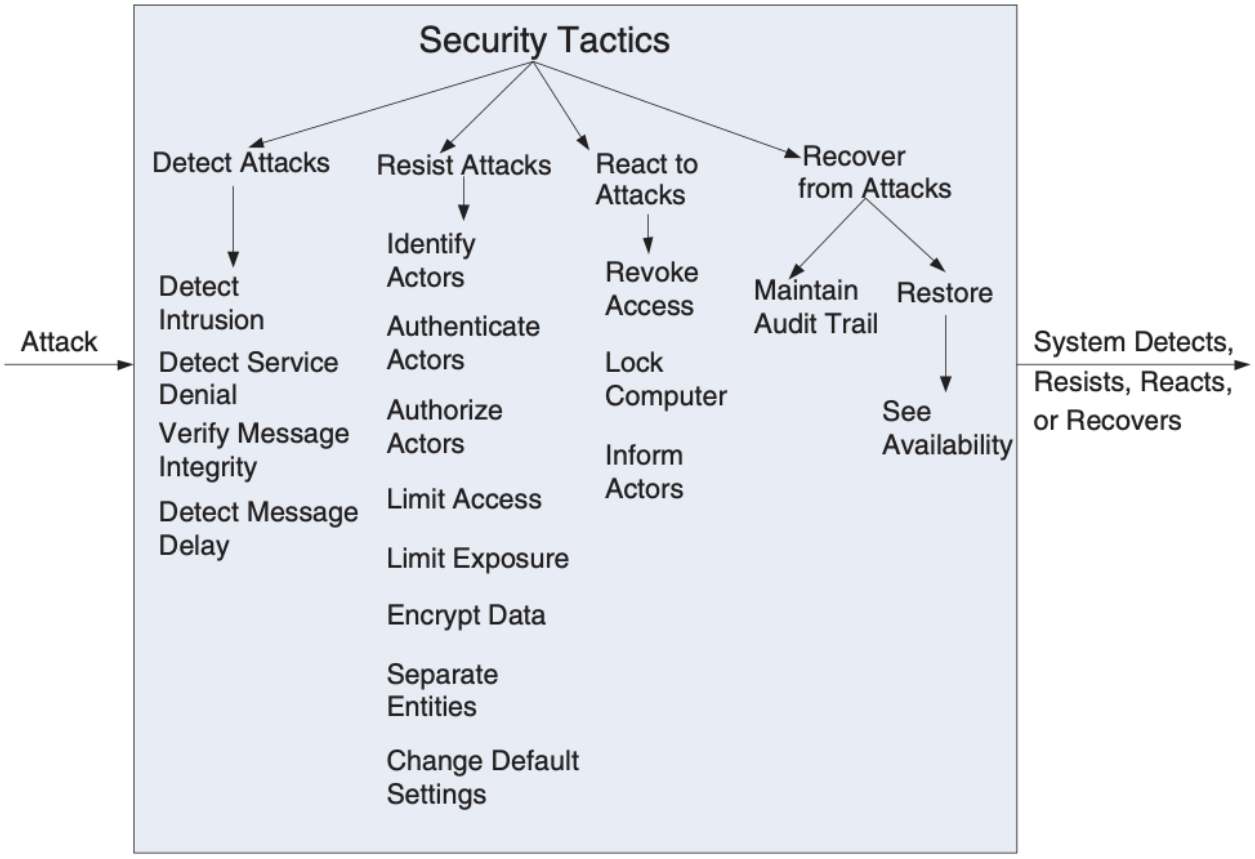


Рисунок 12. Тактики досягнення безпеки

Згідно вимог користувачі мають доступ до даних тільки після мультіфакторної аутентицікації. Для вирішення цієї задачі ми використали патерн федеративної ідентичності (federated identity), в якому доступ обробляється мікросервісом аутентифікації, який в свою чергу має доступ до довіренного провайдера ідентифікатора (токена). Такий провайдер обробляє запит через сервіс аутентифікації соціальних мереж і, у разі успіху, повертає токен мікросервісу. На цьому етапі звичайний користувач отримує доступ до системи, в той час як адміністратор має підтвердити запит на доступ за допомогою другого фактора.

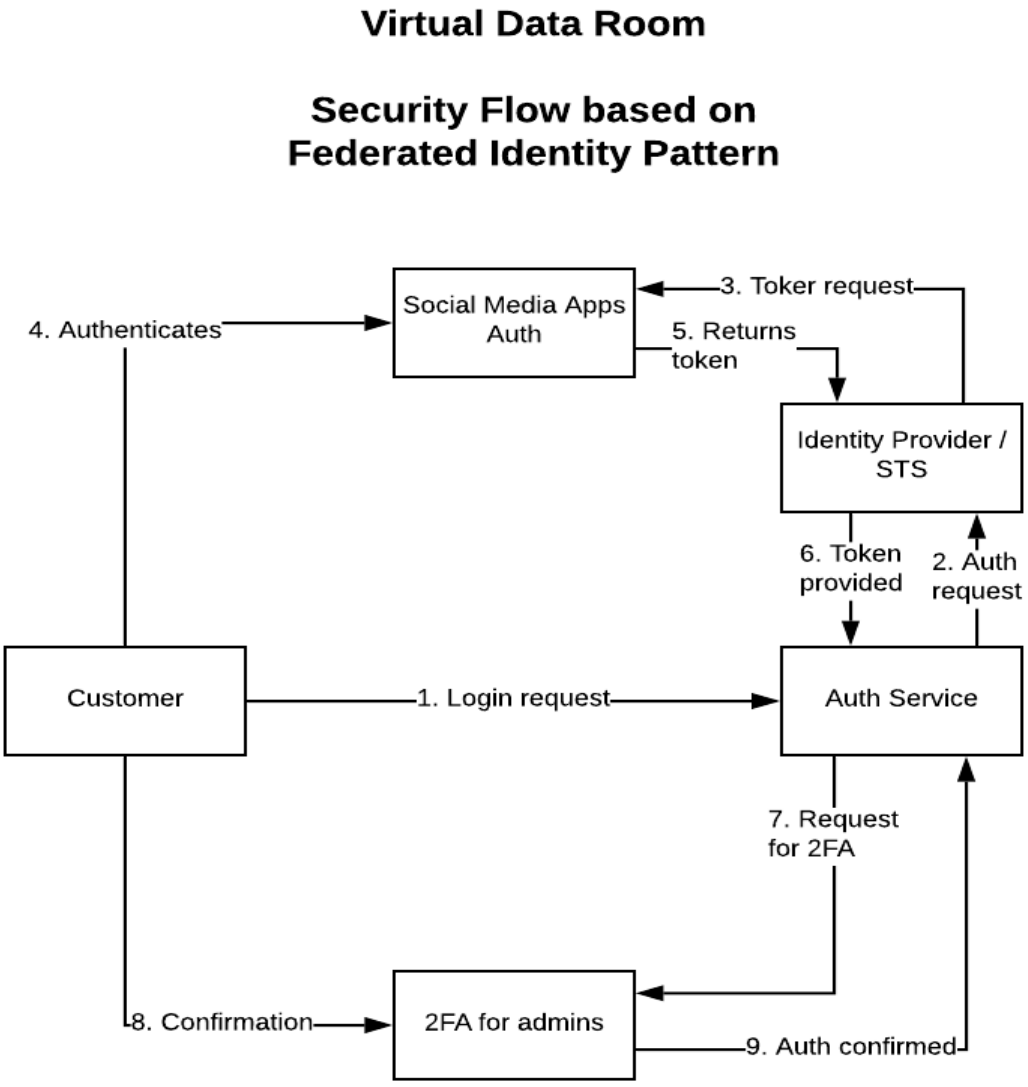


Рисунок 13. Імплементація мультіфакторної аутентифцікації.

Висновки

В першому розділі роботи сформульована задача, визначена важливість якісної архітектурної документації при побудові програмного забезпечення, проведен огляд підходів до процесу розробки архітектури рішення, структури архітектурної документації.

Другий розділ висвітлює аналіз вимог стейкхолдерів, оцінку нефункціональних вимог в розрізі якісних атрибутів, іх пріоритизацію та вибір вимог, які мають найбільший вплив на архітектуру рішення. В данном розділі також розглянули підходи до наборів представлень, які мають адресувати інтереси стейкхолдерів, та їх релевантність до різних типів задач. На підставі аналізу визначили набір представлень, достатній для документування архітектури решення віртуальної кімнати даних, та зробили дизайн таких представлень.

В третьому розділі були розглянуті можливі способи адресування інтересів стейкхолдерів, виражених через метрики якісних атрибутів, які мають найбільший потенційний вплив на архітектуру рішення. На підставі аналізу тактик та кращих практик (патернів) розроблені архітектурні рішення для досяжнення необхідної доступності, продуктивності, масштабовністі та безпеки даних.

Дана робота була зосереджена аспектах розробки програмного забезпечення, які мають безпосередній вплив на архітектуру рішення, тому поза межами цієї роботи, але в рамках проекту були також детально розглянуті і підготовлені use-case сценарії, ієрархічна структура робот, бюджет, роадмеп проекту та конвеєр розробки (Додаток 3), що разом дозволило перейти до наступного етапу – розробки програмного забезпечення.

З іншого боку підход до проектування архітектури рішення, який був застосован в данном проекті, потенційно може використувати як фреймворк для інших проектів. Згідно стандарту ISO 42010 фреймворк відповідає міжнародному стандарту, якщо він включає

1. Інформацію, яка його описує
2. Один або більше інтересів
3. Один або більше стейкхолдерів, які мають такі інтереси
4. Одну або більше точок зору (включаючи їх специфікації згідно міжнародного стандарту)
5. Правила зв’язку, які пов’язують точки зору (згідно п.5.7. стандарту)
6. Умови для застосування (згідно п.6.1. стандарту)
7. Узгодженість фреймворку з положеннями концептуальної моделі стандарту.

Список літератури

1. Architecture Frameworks: TOGAF, ArchiMate, Zachman & DoDAF. [Електронний ресурс]. Режим доступу: http://grahamberrisford.com/00EAframeworks/TOGAF%20with%20ArchiMate,%20DoDAF%20and%20Zachman.htm
2. Best Virtual Data Rooms. [Електронний ресурс]. Режим доступу: https://dataroom-providers.org
3. Datasite Diligence. [Електронний ресурс]. Режим доступу: https://www.datasite.com/us/en/products/overview/diligence.html
4. Functional and Nonfunctional Requirements: Specification and Types. [Електронний ресурс]. Режим доступу: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/
5. John Olamendy. Documenting the Software Architecture. [Електронний ресурс]. Режим доступу: https://johnolamendy.wordpress.com/2011/06/01/documenting-the-software-architecture/
6. ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description. [Електронний ресурс]. Режим доступу: https://www.iso.org/standard/50508.html
7. ISO/IEC 25010. [Електронний ресурс]. Режим доступу: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
8. ISO/IEC/IEEE 42010 Website. [Електронний ресурс]. Режим доступу: http://www.iso-architecture.org/ieee-1471/index.html
9. Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice (SEI Series in Software Engineering). *Addison-Wesley Professional.* 2012.
10. Mark Richards. Software Architecture Patterns. *O'Reilly Media, Inc*. 2015.
11. Martin Abbot. Scalability Rules: 50 Principles for Scaling Web Sites. *Addison-Wesley Professional.* 2011.

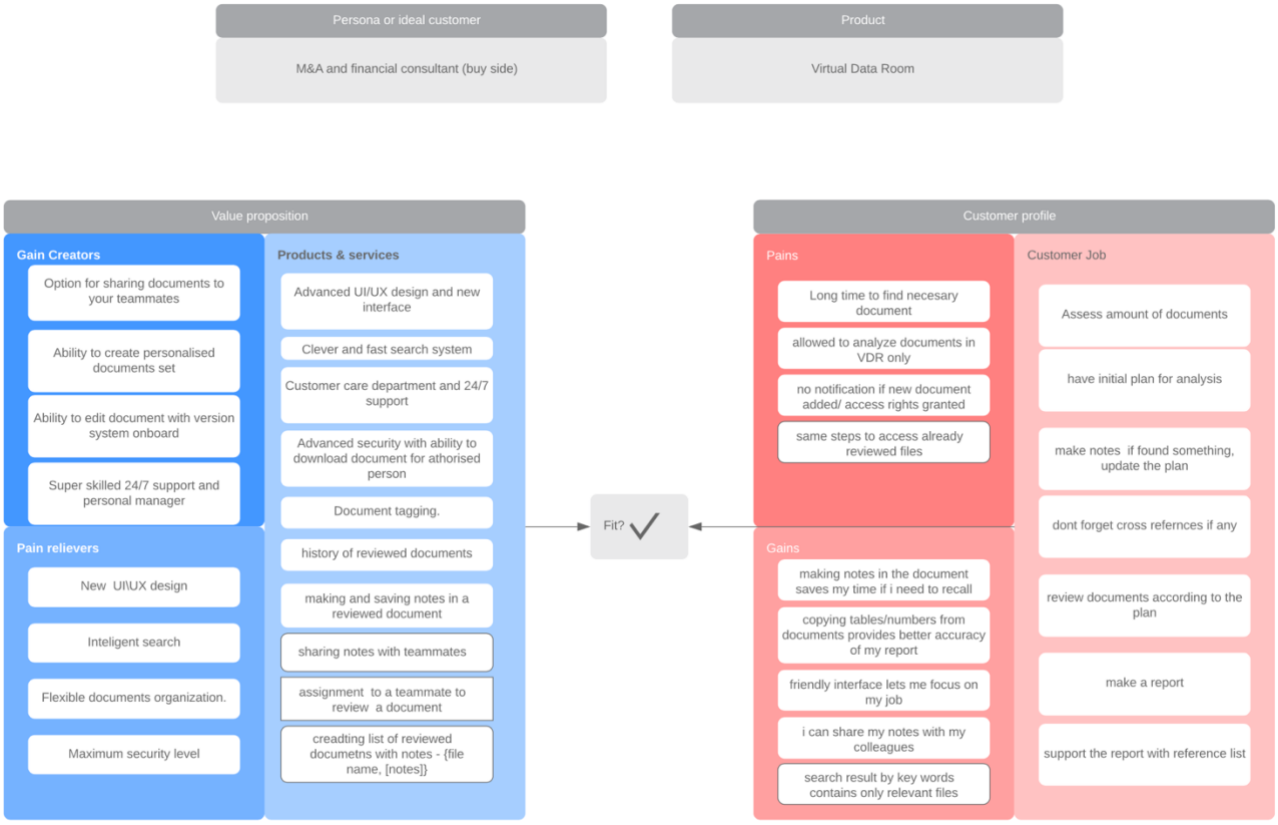
12. Microsoft® Application Architecture Guide. *Microsoft Press.* 2009
13. Mohamed Alladin. Software Architecture - The Difference Between Architecture and Design. [Електронний ресурс]. Режим доступу: https://codeburst.io/software-architecture-the-difference-between-architecture-and-design-7936abdd5830
14. Nick Rozanski, Eoin Woods. Software System Architecture. *Addison-Wesley.* 2012
15. Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford. Documenting Software Architectures. *Addison-Wesley.* 2010.
16. Paul Preiss. The Biggest Problems With ISO 42010 and IEEE 1471. [Електронний ресурс]. Режим доступу: https://itabok.iasaglobal.org/the-biggest-problems-with-iso-42010-and-ieee-1471/
17. Rational Unified Process: Overview. [Електронний ресурс]. Режим доступу: https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/
18. Software Architecture Document. RUP template. [Електронний ресурс]. Режим доступу: https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/webtmpl/templates/a_and_d/rup_sad.htm
19. Software Architecture Document Template. [Електронний ресурс]. Режим доступу: https://wiki.sei.cmu.edu/confluence/display/SAD/Software+Architecture+Documentation+Template
20. The C4 model for visualising software architecture. [Електронний ресурс]. Режим доступу: https://c4model.com
21. The Best Virtual Data Room Service Providers Comparison. [Електронний ресурс]. Режим доступу: https://dealroom.net/resources/virtual-data-room-providers-comparison
22. The TOGAF Standard, Version 9.2 Overview. [Електронний ресурс]. Режим доступу: https://www.opengroup.org/togaf
23. Vijini Mallawaarachchi. 10 Common Software Architectural Patterns in a nutshell. [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013

24. Views and Beyond Documentation Template. [Електронний ресурс]. Режим доступу: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=519381
25. Виртуальные комнаты данных iDeals. [Електронний ресурс]. Режим доступу: https://ru.idealsvdr.com

Додаток 1. ТОП 10 рішень VDR згідно dataroom-providers.org

1	**iDeals**	**4.92**/ 5.0	iDeals virtual data room proposes a wide range of customization and collaboration tools. The easy to set up software and its user-friendly interface ensure any online deals or document management services are smooth. The iDeals VDR allows the user to generate graphs and customs reports on user activity. The VDR is widely used by companies of all sizes.
2	**Firmex**	**4.81**/ 5.0	Firmex virtual data room specializes in highly secured cloud services for online deals and file-sharing. Firmex has gained the reputation of being a secure and helpful platform. This VDR provider is among the top VDR providers because of its high-quality standards and flexible approach to clients' needs. The provider offers different software packages suited to the needs of both small and large businesses.
3	**Intralinks**	**4.72**/ 5.0	Intralinks is trusted by many large companies and well-known global brands. It is an experienced traditional virtual data room provider with high-quality standards and representatives in several countries. It is a true giant in the data room industry and its target market is large sized businesses. Intralinks' VDR solution might feel not flexible enough for younger entrepreneurs that prefer agile software.
4	**Citrix**	**4.59**/ 5.0	Citrix Systems is a reliable company providing cloud services for online business deals and employee workflow needs. Citrix Workspaces is virtual data room software for online deals and includes all tools necessary for document management. Clients can use additional Citrix Systems software to cover process needs.
5	**WatchDox**	**4.51**/ 5.0	BlackBerry Workspaces (fka WatchDox by Blackberry) is a VDR provider that may be a bit difficult to use but redeems itself with the easily-available user and admin guides, trainings, and documentation for users. This VDR has a wide range of advanced tools for virtual space customization. This VDR provides services for businesses of all sizes.
6	**Merrill**	**4.32**/ 5.0	Datasite is an updated software product of Merrill Corporation. Datasite is a VDR that provides excellent solutions for complex deals. Merrill also provides quality extra services that help in consulting, content administration, and other processes required for profitable corporations.
7	**Onehub**	**4.23**/ 5.0	Onehub offers multiple cloud solutions and secure services for business document flow. Pricing for the Onehub data room may be on the high side but it guarantees the best technical support and software product quality.
8	**DealRoom**	**4.17**/ 5.0	DealRoom is an M&A due diligence SaaS. The software is expensive, pricing wise, but it does provide advanced online tools for both the buy-side and sell-side of the M&A process. The software also has features that tackle each aspect of the M&A deal lifecycle.
9	**Digify**	**4.05**/ 5.0	Digify is a relatively young virtual data room provider. Despite its late entry into the VDR market, it proposes all basic virtual data room tools and is well known to be a user-friendly platform targeting small-sized businesses.
10	**Donnelley**	**3.90**/ 5.0	Donnelley Financial Solutions Group's Venue provides basic cloud instruments for business deals done online. Venue features tools for digital document flow and file sharing.

Додаток 2. Value proposition



Додаток 3. Конвеєр розробки програмного забезпечення для віртуальної кімнати даних.