НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
"___"_____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності **123 «Комп'ютерна інженерія»**

на тему:
«Система управління документами з використанням технології Java EE»

Виконала:

студентка IV курсу, групи КВ-91

Мамай Вероніка Володимирівна _____ _____
(підпис)

Керівник ст. викл. Дробязко І. П. _____ _____
(підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент _____ _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма

«Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис)

«_____» _____2022 р.

ЗАВДАННЯ
на дипломний проєкт студента

Мамай Вероніки Володимирівни

1. Тема проєкту «Система управління документами з використанням технології Java EE», керівник проєкту - старший викладач Дробязко І. П.,

затверджені наказом по університету від «31» травня 2023 р. №2107-С

2.Термін подання студентом проєкту _____30.05.2023 р._____

3. Вихідні дані до проєкту _____Див. Технічне завдання_____

4. Зміст пояснювальної записки:

- Аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- Вибір технологій і засобів розроблення системи;
- Опис середовища розробки системи;
- Опис розробленої системи управління документами.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

- Архітектура системи. Схема структурна;
- Залежності між модулями. Схема структурна;

- Діаграма системи. Схема структурна;
- Додавання нового файлу до системи. Схема алгоритму;
- Презентація дипломного проєкту.

---

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання 4.11.2022

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою роботи	04.11.2023	
2.	Розроблення та узгодження технічного завдання	15.12.2023	
3.	Розроблення структури системи	20.01.2023	
4.	Розроблення та програмна реалізація системи	09.03.2023	
5.	Тестування системи	10.05.2023	
6.	Підготовка матеріалів текстової частини проєкту	12.05.2023	
7.	Підготовка матеріалів графічної частини проєкту	16.05.2023	
8.	Оформлення документації дипломного проекту	19.05.2023	
9.	Попередній огляд матеріалів дипломного проєкту на кафедрі	30.05.2023	

Студент _____ Вероніка МАМАЙ
(підпис)

Керівник проєкту _____ Ірина ДРОБЯЗКО
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (56 с., 37– рис.).

Об’єкт розробки – система управління документообігом.

В ході виконання дипломного бакалаврського проєкту:

- проведено аналіз існуючих рішень;
- обрано технології та засоби розробки системи;
- розроблено структуру системи, виконано програмну реалізацію;
- здійснено тестування системи;

Розроблена система забезпечує наступні функціональні можливості:

- завантаження файлів до глобального репозиторію;
- можливість створення, запису метаданих відповідно до типу даних; можливість пошуку файлів за метаданими;
- можливість створення ролі, типу файлу, директорії та доступу до них;
- можливість використання системи як браузерного додатку;
- можливість зв'язку з іншими системами обробки документів за допомогою адреси API;

У процесі розробки використовувалася технологія JEE із сервером додатків Jboss/Wildfly як серверним рівнем і Google Angular SPA для зовнішнього рівня. Як сховище метаданих використовувалася база даних PostgreSQL, а як сховище двійкових файлів Min.IO відповідно до протоколу Amazon S3. Система безпеки заснована на системі KeyCloak, яка служить для управління ідентифікацією.

Впровадження системи дозволить підвищити ефективність роботи з документами в установах, покращити організацію ведення документації.

Ключові слова: СИСТЕМА ДОКУМЕНТООБІГУ, БАЗА ДАНИХ, JAVA, МЕТАДАНІ, ANGULAR.

ABSTRACT

An explanatory note is attached to the final work (56 p., 37 figures).

The subject of the study is a document management system.

During the implementation of the bachelor's diploma project:

- an analysis of existing solutions was carried out;

- technologies and tools for system development are selected;

- system structure was designed, software was implemented;

- system tests were carried out;

The developed system provides the following functionalities:

- uploading files to the global repository;

- the ability to create and save metadata depending on the type of document; the ability to search for files based on dynamically created metadata;

- the ability to define access levels based on dynamically created roles, directories and file types;

- access to the system using a browser application;

- the ability to integrate with other systems using the Rest API;

In the development process, JEE technology was used with the Jboss/Wildfly application server as the backend layer and Google Angular SPA for the frontend layer. A PostgreSQL database was used as a metadata store and Min.IO as a binary file store in accordance with the Amazon S3 protocol. The security system is based on the KeyCloak system that serves as identity management.

The implementation of the system will improve the efficiency of work with documents in institutions, and will improve the organization of document management.

Keywords: DOCUMENT SYSTEM, DATABASE, JAVA, METADATA, ANGULAR.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.002 ТЗ	Система управління	4		
			документами з			
			використанням технології			
			Java EE			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Система управління	2		
			документами з			
			використанням технології			
			Java EE			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Система управління	56		
			документами з			
			використанням технології			
			Java EE			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Архітектура	1		
			системи			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Мамай В.В.			Система управління документами з використанням технології Java EE Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-91	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.006 Д2	Залежності між	1		
			модулями			
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Діаграма системи	1		
			Схема структурна			
	А4	ІАЛЦ.045440.008 Д4	Додавання нового	1		
			файлу до системи			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			
			Презентація			

					ІАЛЦ.045440.001 ОА	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

					ІАЛЦ. 045440.002 ТЗ		
Змін	Арк.	№ докум.	Підпис	Дата			
Розробив		Мамай В.В.			*Система управління документами з використанням технології Java EE* ***Технічне завдання***	Літ.	Аркуш 1 / Аркушів 56
Перевірив		Дробязко І.П.					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-91	
Затвердив		Романкевич В.О.					

НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Система управління документами з використанням технології Java EE».

Галузь застосування: системи документообігу для застосувань різного призначення.

ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення системи управління документами, яка шляхом збереження бінарних файлів в глобальному репозиторії та використання метаданих для пошуку, дозволяє підвищити ефективність роботи з документами.

ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та інші джерела інформації за темою розробки.

ТЕХНІЧНІ ВИМОГИ

1. Вимоги до програмного продукту, що розробляється

- можливість додавання файлу та метаданих до нього;
- можливість додавати нові ролі, директорії, типи документів, словники;
- можливість редагування ролі, директорії, типи документів, словники;
- авторизація за допомогою логіну та паролю за допомогою Keycloak;

- можливість роботи у якості користувача та адміністратора;
- зберігання файлів у глобальному репозиторії;
- наявність зручного інтерфейсу користувача.

2. Вимоги до демонстраційної версії системи VMP визначені для одного ПК, на якому будуть встановлені всі компоненти:

- сервер додатків Wildfly;
- база даних PostgreSQL;
- Keycloak;
- сервер S3 Min.IO;
- веб-браузер Google Chrome;

3.Вимоги до програмного та апаратного забезпечення користувача:

- Процесор з архітектурою x64: 4 ядра, 8 потоків, 3,6 ГГц, наприклад, Intel 10-го покоління i3-10100;
- Оперативна пам'ять - мін. 16 ГБ;
- HDD/SSD - мін. 64 ГБ для операційної системи та програмних компонентів;
- Операційна система: Windows 10 або Linux, наприклад, Ubuntu.

3. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою роботи	04.11.2023
2.	Розроблення та узгодження технічного завдання	15.12.2023
3.	Розроблення структури системи	20.01.2023
4.	Розроблення та програмна реалізація системи	09.03.2023
5.	Тестування системи	10.05.2023
6.	Підготовка матеріалів текстової частини проєкту	12.05.2023
7.	Підготовка матеріалів графічної частини проєкту	16.05.2023
8.	Оформлення документації дипломного проекту	19.05.2023
9.	Попередній огляд матеріалів дипломного проєкту на кафедрі	30.05.2023

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.004 ПЗ	Система управління	56		
			документами з			
			використанням технології			
			Java EE			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Архітектура	1		
			системи			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Залежності між			
			модулями	1		
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Діаграма системи	1		
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Мамай В.В.			Система управління документами з використанням технології Java EE Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-91	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Додавання нового	1		
			файлу до системи			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			
			Презентація			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.045440.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: «Система управління документами з використанням технології Java EE »

Київ – 2023 року

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.045440.005 Д1. Архітектура системи. Схема структурна

ІАЛЦ 045440.006 Д2. Залежності між модулями. Схема структурна

ІАЛЦ.045440.007 Д3. Діаграма системи. Схема структурна

ІАЛЦ.045440.008 Д4. Додавання нового файлу до системи. Схема алгоритму

Додаток 3. Презентація проєкту

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД — Система управління базою даних;

API — Application Programming Interface – Прикладний програмний інтерфейс;

DA — Digital Assets — Цифрові активи;

DMS — Document Management System — Електронний документообіг;

DOM — Document Object Model — Об'єктна модель документа;

DTO — Data Transfer Object — Об'єкт передачі даних;

IDE — Integrated Development environment — Інтегроване середовище розробки;

JAX-RS — Jakarta RESTful Web Services;

JDK Java Development Kit;

JPA — Java Persistence API;

JPQL — Java Persistence Query Language — Платформно-незалежна об'єктно-орієнтована мова запитів;

JRE — Java Runtime Environment;

JVM — Java Virtual Machine — Віртуальна машина Java;

MVC — Model-view-controller — Модель–представлення–контролер;

OCR — Optical Character Recognition — Оптичне розпізнавання тексту;

ORM — Object-Relational Mapping — Об'єктно-реляційне відображення;

OS — Open-Source;

PIM — Product Information Management — Управління інформацією про продукт;

POM — Project Object Model;

REST — Representational State Transfer – Передача репрезентативного стану;

SDK — Software Development Kit;

SPA — Single Page Application — Односторінковий застосунок;

ВСТУП

У сучасному світі зберігання та захист інформації стають все більш актуальною проблемою. За допомогою різноманітних цифрових пристроїв інформація поширюється та стає доступною будь-де в світі. Проте можливий несанкціонований доступ до даних хакерами та викрадення і розміщення їх у вільному доступі.

Для зберігання даних використовують різні підходи, кожен з яких має свої переваги та недоліки. Наприклад, зберігання даних на сервері може бути корисним для збереження конфіденційної інформації, приватних даних, а також для захисту від зловмисного програмного забезпечення.

Наразі використання серверів як місця зберігання даних втрачає свою актуальність, здебільшого це залишається важливим для галузей, де необхідна перш за все конфіденційність та швидкість доступу. Значною перевагою при використанні сервера є можливість самостійно обирати методи інтеграції з іншими системами, без обмежень вибору версій. На відміну від хмарних серверів, у даному випадку краще зберігається конфіденційність, адже керування доступом відбувається на стороні локального серверу, тоді як на хмарі неможливо спрогнозувати витік інформації.

Головними перевагами хмарних серверів є зручність зберігання та доступу до інформації. Через оптимізацію застарілих систем зберігання можна підвищити ефективність роботи, завантажувати файли багатьох типів та отримувати до них доступ з різних пристроїв з поділом на різні рівні доступу. Популярними хмарними сховищами є: Dropbox, Mega,OneDrive, iCloud та інші.

Також, окрім зберігання даних, важливо мати добре оптимізоване середовище для управління ними. Найважливішим є швидкість доступу без потреби довгого пошуку серед десятків, сотень, а то й тисяч файлів. Для цього можна використовувати метадані (назва файлу, дата створення, розмір тощо). Існує багато систем збереження файлів, які дозволяють додавати метадані до

них, наприклад, Dropbox, Google Drive, Tropy та інші. Проте вони орієнтовані виключно на збереження даних у власних хмарних сервісах.

Завданням дипломного проєкту є розробка системи зберігання файлів та управління ними, де, окрім можливості додавання метаданих до окремих файлів, додається можливість вибору між місцем їх зберігання, а також додаткова можливість інтеграції з різними системами. Це зробить систему більш гнучкою та універсальною для використання в застосуваннях різного призначення.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Аналіз засобів для управління DA

На сьогоднішній день існує багато програм, додатків чи браузерних розширень, здатних виконувати функцію зберігання різноманітних даних, головною ціллю яких є надання можливості централізувати місцезнаходження файлів, адже при збільшенні кількості інформації суттєво ускладняється можливість управління і збільшується ризик зробити помилку, наприклад, загубити потрібну версію одного й того самого документу, або переплутати версії. Існує багато систем які здатні допомогти в керуванні, зберіганні та обміні різних файлів. Основна функція це надання користувачам доступу до централізованої бібліотеки в якій доступні функції збереження та обміну файлами без підключення до локального серверу. Розглянемо популярні варіанти таких програм.

Bynder — зручна вебплатформа для управління DA (цифрові активи фотографії, зображення, відео, документи, пакети даних, електронні книжки, презентації, анімації тощо), що позиціонує себе як помічника для “подолання хаосу зростаючого вмісту”. З безперечних переваг варто зазначити сучасний та приємний, інтуїтивно зрозумілий, інтерфейс, який можна побачити на рис. 1.

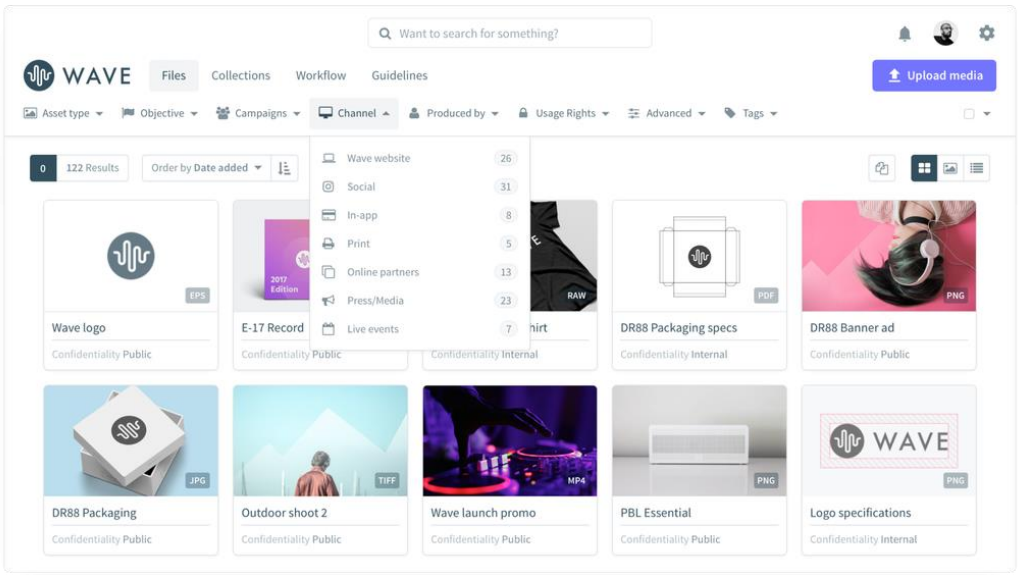


Рисунок 1 – Вигляд інтерфейсу Bynder

Всі дані знаходяться в одному центральному місці зберігання для цілої компанії або бренду, що суттєво оптимізує спільну працю, включно з файлами які можуть бути доступні для користувачів з усього світу. Присутня функція пошуку з використанням фільтрів, а також можливість надавати права користувачам. Для безпеки використовується двофакторна аутентифікація.

Вебплатформа платна, окрім основних функцій можна купити дві дорожчі версії які розширені модулями творчої студії, робочого менеджменту, аналітикою, студією редагування відео, та іншими.

Має можливість інтеграції з : Adobe Experience Manager, Google, Microsoft Active Directory, WordPress та інші.

Acquia DAM (Widen) — вебплатформа розрахована на використання фірмами як головного місця зберігання матеріалу (фотографії, відео, брошури, логотипи тощо), яка спрощує організацію та доставку продукту на ринок. Окрім управління DA має також PIM. PIM — Product Information Management або управління інформацією про продукт, у даному випадку мова йде про систему управління інформацією, яка є необхідна для маркетингу та продажу продуктів через канали дистрибуції. Інтерфейс наведено на рис. 2.

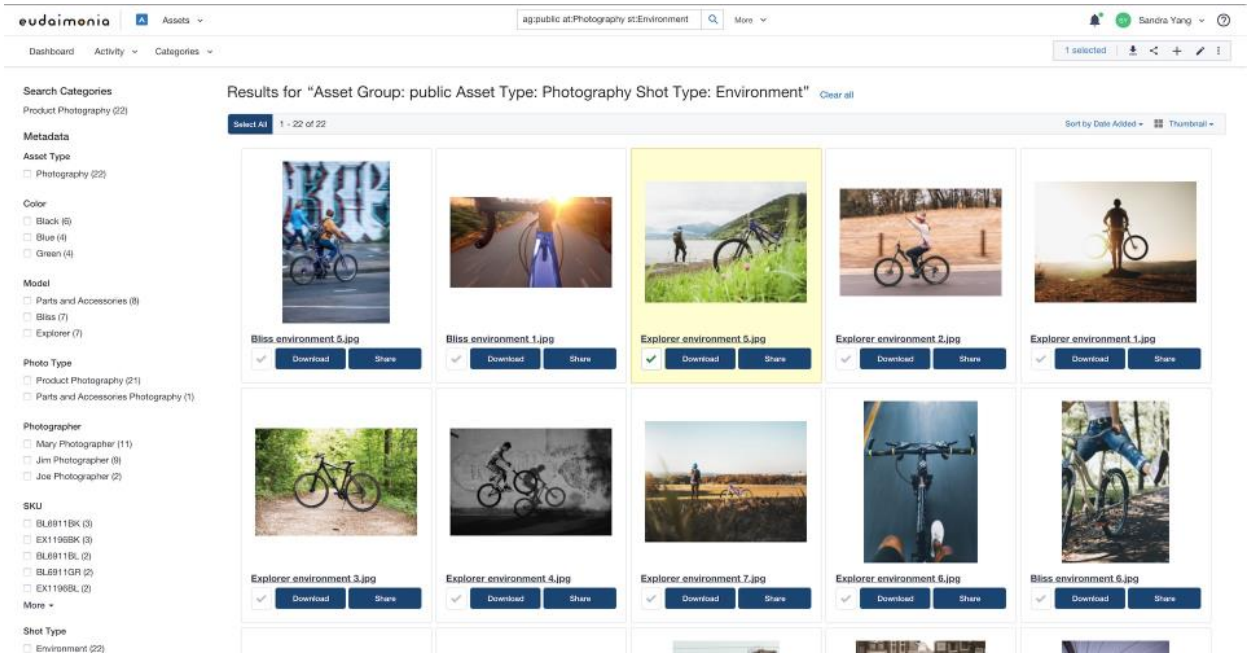


Рисунок 2 – Вигляд інтерфейсу Acquia DAM

З істотних переваг є функція інтелектуального пошуку з гнучкими моделями метаданих, які допомагають класифікувати дані, і можливість розрізнити рівні допуску наданням прав різним групам користувачів. Кількість ролей з користувацьким рівнем доступу не обмежена. Автоматичне надання позначок фотографіям чи зображенням за допомогою штучного інтелекту.

З інших функцій наявний попередній перегляд у високій якості, доступ в окремо зазначений час, контроль версій, історія активності. Окрім вебплатформи також є мобільні додатки на iOS та Android. Завантаження та редагування може використовувати необмежена кількість людей з функцією додавання водного знаку.

Має можливість безкоштовної інтеграції, які розроблені і підтримується безпосередньо Acquia, з: Adobe Workfront, Brightcove, GRIN, Google Drive, Google Sheets, Wistia, Dropbox та інші.

Партнерська інтеграція надається не у всіх рівнях підписки, але можуть бути підключені як окреме налаштування, зокрема з: Adobe Creative Cloud, Microsoft Office 365, ShotFlow, Shutterstock, Visme, Mediafly та інші.

Brandfolder — інтуїтивно зрозуміле програмне забезпечення для управління DA, з широкими можливостями для користувача, не тільки збереження, управління та обмін даними, а також аналітика різних форматів. Підтримує, окрім звичних документів та зображень, 3D-рендеринг та відео з розширенням 8К.

З особливостей, варто зазначити запатентовану технологію з використанням штучного інтелект — Brand Intelligence, яка запатентована Brandfolder. Алгоритми аналізують файл, визначають теги, за якими можна буде його знайти, що суттєво спрощує пошук. Автоматично надає загальні метадані, заповнює теги, має можливість створювати свої інтелектуальні колекції з заданими тегами.

Аналітика з використанням штучного інтелекту надає детальний аналіз даних. Виявляє два однакових файли при завантаженні, а також зазначає коли маємо ідентичні або схожі теги. Є можливість виявлення конкретних елементів

на відео за допомогою технології Brand Intelligence, яка аналізує кожен кадр і автоматичні надає детальні теги, що продемонстровано на рис. 3.

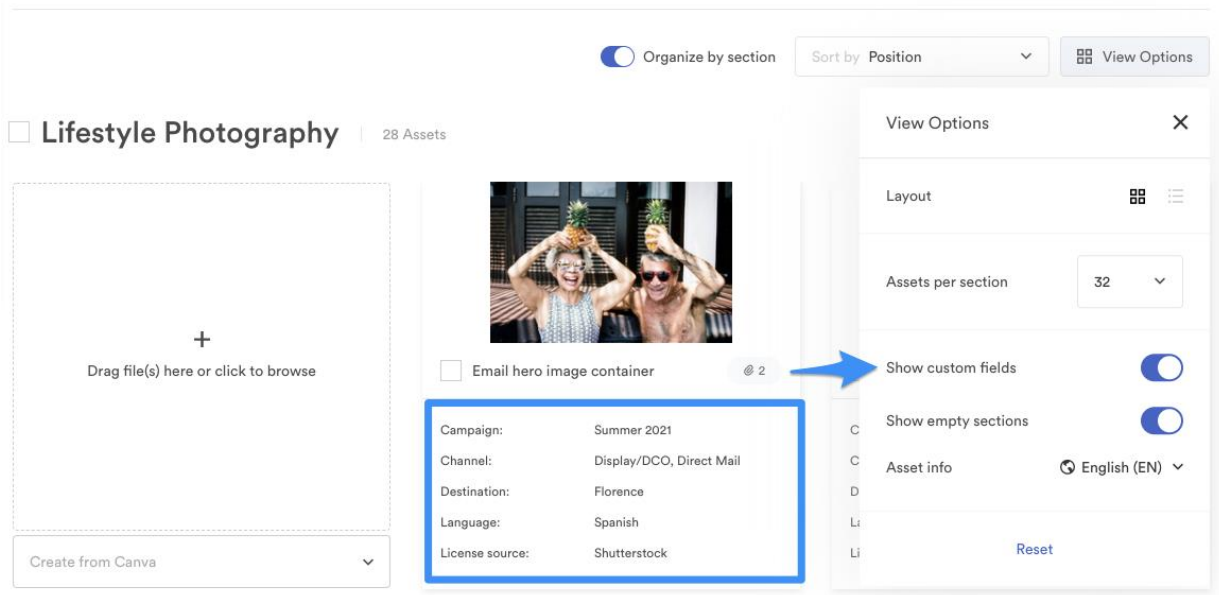


Рисунок 3 – Вигляд інтерфейсу Brandfolder разом з метаданими

Має можливість інтеграції з: Adobe Illustrator, Adobe InDesign, Adobe Photoshop, Canva, Chrome Extension, Figma, Google Analytics, Google Apps, IBM, Microsoft Office, WordPress та інші.

4ALLPORTAL — програмне забезпечення з потужними функціями текстового пошуку та OCR (Optical Character Recognition або оптичне розпізнавання тексту — технологія яка допомагає при пошуку тексту у зображеннях, розпізнає текст у зображеннях таких форматів як: GIF, JPG, PNG та інші).

Можливість завантажувати фото, відео, документи в одному місці, надавати окремим файлам опис, теги, які можна буде використати для цільового пошуку, а також наявна можливість розширеного пошуку за типом файлу, статусу тощо. При пошуку також може бути проаналізована інформація з тексту чи з зображення, яке містить текст.

Користувача можна наділити різними правами відповідно до ролі, що створює багаторівневий контроль, доступ до окремих файлів, інформації,

метаданих тощо. Також окрім надання ролі, можливий обмежений за часом доступ до файлу.

До файлів можна приписати завдання, певну роботу та виконавця для неї. Даний файл буде зазначено особливою червоною позначкою, яка вказує на процес зміни у файлі, тобто роботу у даний момент часу (можна побачити на рис. 4).

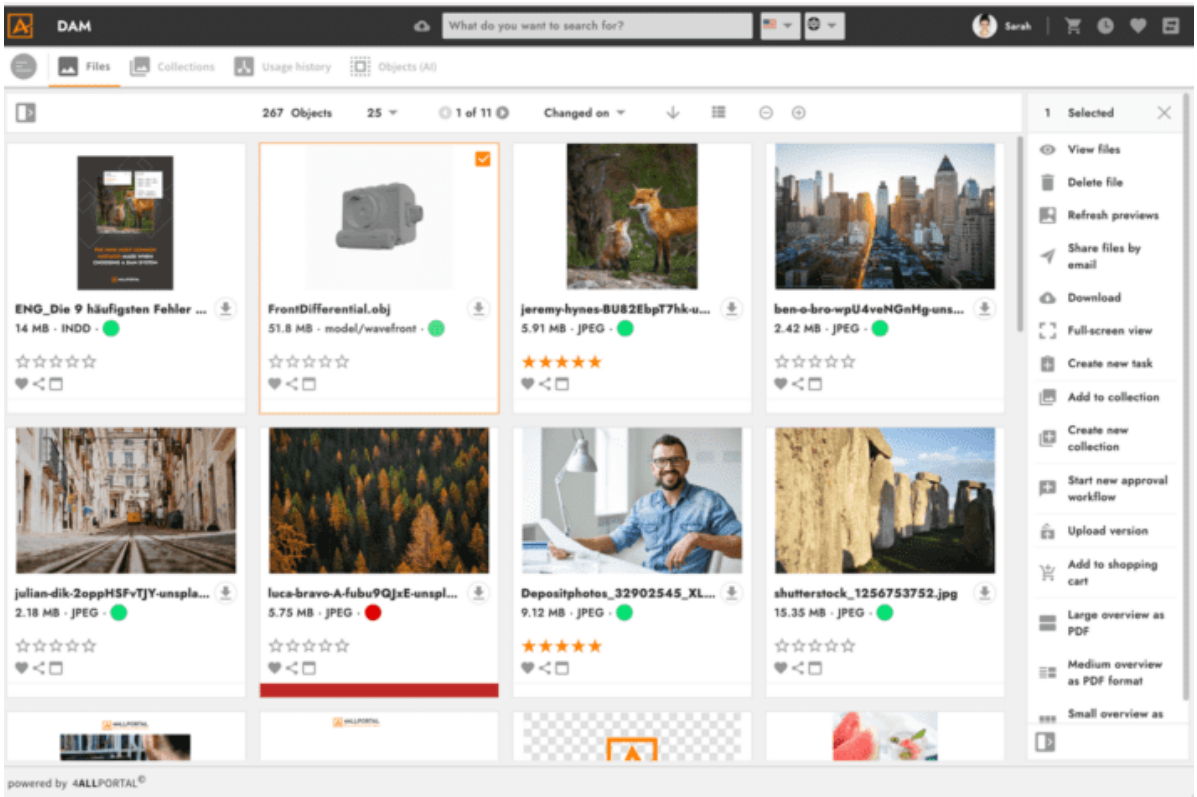


Рисунок 4 – Вигляд файлу який знаходиться в процесі обробки

Для підвищення ефективності є функція відстеження всіх змін, які вносять члени команди, система автоматично оновлює версію для всіх користувачів, а також залишає можливість порівняння та повернення до попередньої версії.

Надає можливість зберігання як на хмарному сервісі, так і на особистому сервері.

Має можливість інтеграції з: Adobe Creative Cloud, Adobe Illustrator, Adobe InDesign, Adobe Photoshop, Dropbox, JIRA, MacOS, Microsoft Office, Microsoft Teams, Prometheus, Social Media (YouTube, Facebook, Instagram, LinkedIn, тощо),WordPress та інші.

1.2 Аналіз засобів для управління DMS

З проблемою організації управління та збереження документів з дотриманням відповідних стандартів в сучасному світі має справу кожна фірма, яка веде яку-небудь електронну документацію. Невеличка помилка при великих масштабах документообігу може призвести до страшного безладу в бюрократичній сфері. Для вирішення таких проблем існують так звані системи управління документами — DMS (Document Management System — електронний документообіг). Такі системи зазвичай створюються під конкретні компанії і призначені для зберігання з можливістю ведення історії версій, надання доступу різним користувачам системи з правами читання, запису, змін тощо.

Також в таких системах для зручності подальшого пошуку наявна можливість автоматичного надання метаданих чи запит про них у користувача при додаванню файлу, найчастіше такі дані це час додавання, назва користувача який здійснює додавання чи тип документу який буде доданий. З внесенням змін також з'являється потреба в історії змін, для збереження версій, щоб не виникало проблеми при роботі більш ніж однієї людини над одним файлом, та версія над якою будуть працювати завжди була актуальною. Для зручності подальшого пошуку окрім стандартного індексу для документів, в деяких системах є вбудована технологія OCR, що забезпечує пошуку по ключовим словам навіть у не текстових файлах.

Для зберігання може бути використано як сервер, хмарний сервіс так і СУБД (система управління базою даних). Також популярним є рішення інтеграції з різними програмами управління документами, для зручності створення нового, чи внесення змін в існуючий, без потреби виходити з DMS та шукати даний документ на комп'ютері, з можливим попереднім завантаження потрібного файлу. Такий підхід дозволяє суттєво заощадити час.

Розглянемо популярні варіанти таких систем.

Wrike — програмне забезпечення для організації робочого процесу, яке також допомагає організувати спільне користування документами та медіафайлами, має вбудовану систему управління.

Для зручності вся потрібна інформація може бути закріплена на головному екрані, а всі пов'язані з цим файли зберігаються в одному місці, що спрощує доступ до них та подальше редагування. Управління головним екраном та робочими процесами і документами відбувається за допомогою конструктора віджетів, що продемонстровано на рис. 5

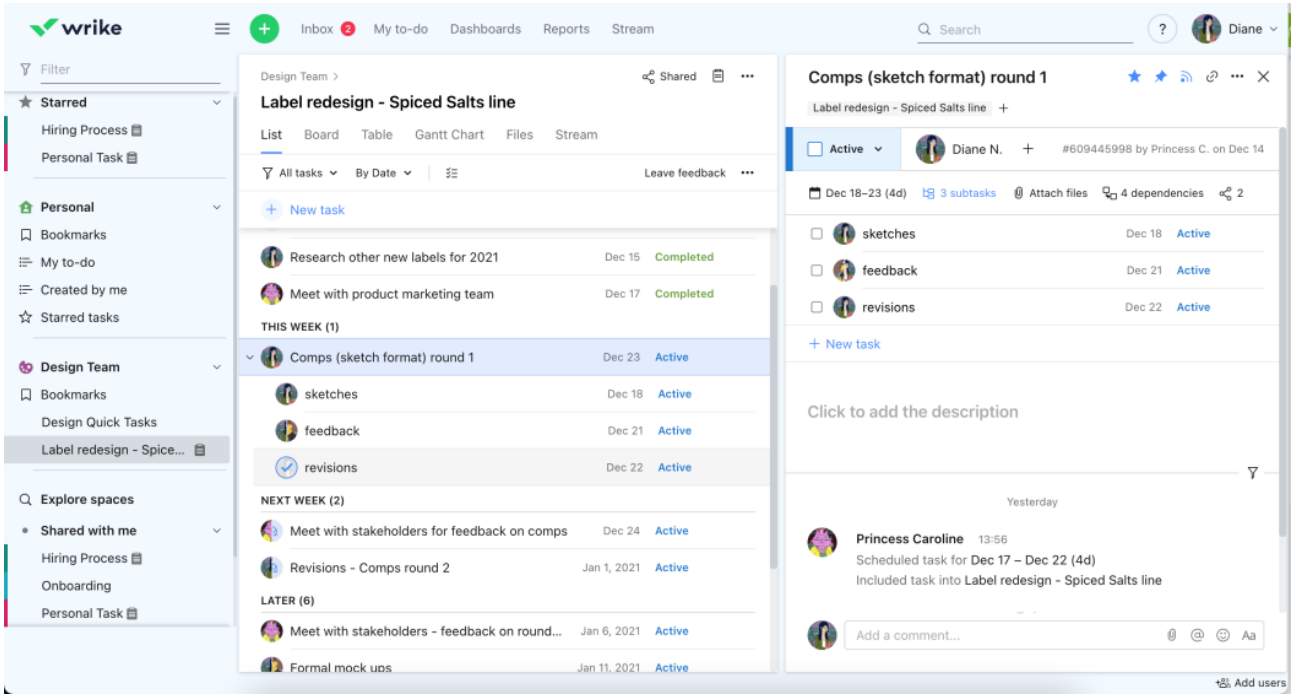


Рисунок 5 – Вигляд інтерфейсу екрану робочого процесу

Для швидкого пошуку можливо створити ієрархію задач, з розширенням до підзадачі. Наявна функція надання доступу до файлу іншим членам команди. Для більшої оптимізації програма має можливість відправлення інтелектуальних повідомлень при ризику відставання від плану.

Додавання можна зробити з комп'ютера чи вказати посилання на інші офісні інструменти, як-от GoogleDrive або DropBox, або створити файл всередині програми. Збереження відбувається на особистому хмарному просторі , що дозволяє керувати документами з будь-якого місця.

Revver — програмне забезпечення для управління документами та робочими процесами. Підходить як для зберігання документів та організації спільної праці так і для відображення та автоматизації процесу праці та управління контрактами і життєвими циклами.

Має можливість створення каталогів (та їх подальшого заглиблення) по шаблонам для створення узгодженої файлової структури яка підходить для даного типу документації. Можливий попередній перегляд файлів з каталогу, наприклад, перегляд завантаженого PDF-файлу на рис. 6.

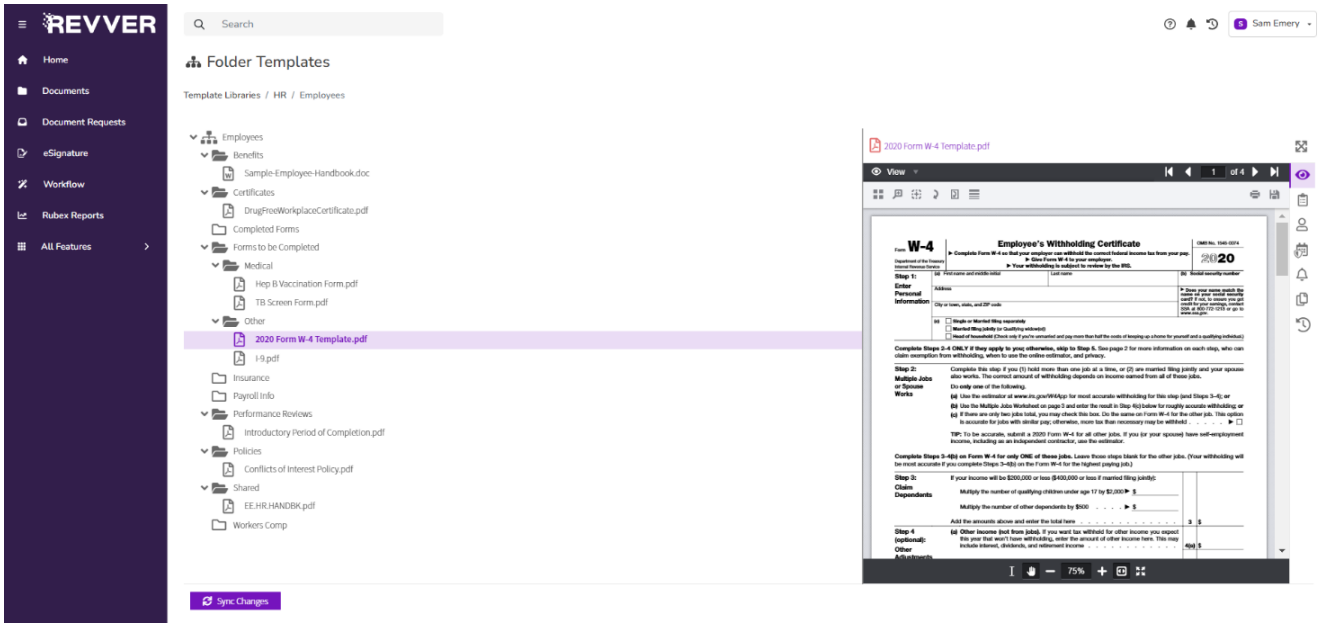


Рисунок 6 – Вигляд інтерфейсу Revver з переглядом файлу

У стандартних типів зображень, текстових документів, PDF-файлів є можливість автоматично створювати метадані, а також згрупувати за ними. Функція пошуку, як проста так і розширена з уживанням метаданих, для більшої ефективності використовується збереження результатів пошуку для швидкого доступу.

Для організації спільної роботи групи користувачів над одним документом є можливість додавання коментарів, запис активності документу для збереження та відслідковування оновлень, а також встановлення етапів узгодження. Управління робочим процесом дає змогу контролювати прогрес від простої маршрутизації документів і до багатоетапних процесів.

Модуль Revver Reports на основі завантажених документів у каталогах може автоматично створити бізнес-аналітику з наочними графіками та діаграмами, а звіт перетворити у формати PDF, HTML або CSV.

DocuWare — комплексна система управління документами, яка була розроблена для управління цифровими документами та їх систематизації.

Окрім основних функцій зберігання, пошуку та виймання документу має можливість попереднього перегляду файлів та легку систему автоматизування процесів сканування та архівації. Універсальні функції дають змогу отримати доступ з будь-якого приладу який має доступ до інтернету, приклад інтерфейсу зображений на рис. 7

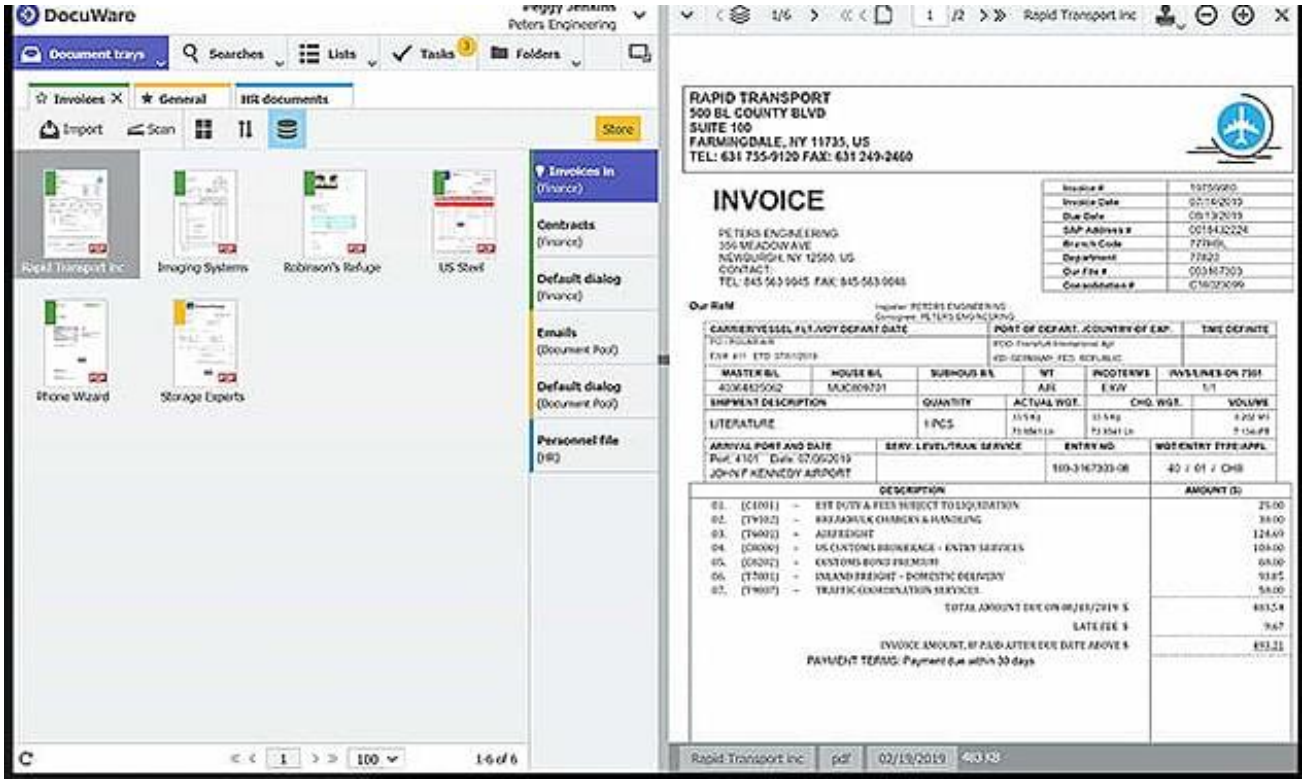


Рисунок 7 – Вигляд інтерфейсу DocuWare

Має можливість інтеграції з великою кількістю різноманітних систем, наприклад: управління взаємовідносинами з клієнтами чи споживачами, включно з функціями збору, зберігання й аналіз інформації, що до них відноситься, планування ресурсів підприємства, корпоративна інформаційна система, призначена для автоматизації обліку та керування.

1.3 Аналіз засобів збереження інформації

Отже, підсумовуючи вищезазначені приклади, варто звернути увагу на місце збереження файлів.

Для виконання основної функції — зберігання файлів, треба проаналізувати можливості. Головний вибір, з яким стикаються розробники, чи буде інформація знаходитись на локальному сервері, чи в мережі віддалених серверів, розміщених в Інтернеті, тобто на хмарному сервісі.

On-premises software — тобто локальне програмне забезпечення, яке реалізує збереження даних в мережі інфраструктури організації, яка найчастіше фізично знаходиться в приміщенні фірми. Основною перевагою цього варіанту є безпека, адже локальна система цілком незалежна, побудована за міжмережевим екраном, який контролює мережевий трафік згідно з областями безпеки та прав користувачів, які отримують доступ безпосередньо через підключення до локального сервера.

Для підтримання функціонування програмного забезпечення потрібен постійний контроль та технічне обслуговування, оновлення системи тощо. Принципова перевага в тому, що компанія має вільний простір при виборі методів інтеграції структури з іншими системами. Але варто зазначити, що в такому випадку треба враховувати як складність і трудомісткість процесу, так і додаткові витрати на обслуговування, підтримку безпеки, розширення серверів та підтримання безперервної роботи і врахування такої можливості, як, наприклад, відключення світла.

При необхідності розширення варто враховувати додаткові витрати, як-от кошт техніки, обладнання і час роботи техніків, та час який буде потрібен на реалізацію.

Локальне забезпечення доступне тільки тим користувачам які знаходяться фізично на місці роботи, тобто це викликає обмеження лише організацією де розміщено програмне забезпечення, і ускладнює роботу з клієнтами.

Software as a service — також відоме як програмне забезпечення на вимогу, чи вебхостингове програмне забезпечення, яке є моделлю ліцензування та доставки програмного забезпечення реалізоване на підставі бізнес-моделі за підпискою, тобто клієнти платять не за викуплену програму, а за її використання. У даному випадку розглядаються послуги хмарного постачальника для розміщення програми/даних в центрі обробки даних постачальника. Підписка може бути як помісячна, так і річна, а такий постачальник хмарних сховищ як pCloud надає можливість викупити сховище у довічний доступ. Такий тип оплати дозволить заощадити на техніці і спеціалістах, а також дозволяє без додаткових витрат за потребою зменшити кількість пам'яті. Відомі постачальники таких послуг : Google Диск, Microsoft OneDrive, pCloud, Icedrive, Dropbox тощо.

Головна перевага цього методу це те, що часу на додаткове збудування та інсталяцію не потрібно, послуга доступна від моменту покупки. Також не потребує багато часу на багатомасштабне впровадження, тільки на навчання та адаптацію працівників, інтеграцію нового програмного забезпечення з іншими елементами системи. Але незважаючи на швидкість, з якою можна отримати вже готове рішення, воно буде загальним, а не пристосованим індивідуально до потреб користувача.

Такий підхід робить можливим організацію віддаленої роботи легше, для доступу до файлів потрібне лише стабільне інтернет-з'єднання. Майже кожне хмарних рішень підтримують дозвіл на доступ користувачів з-поза організації, і не обмежує доступ.

Безпека реалізована через постачальника, і може мати можливість надання адміністративних прав для встановлення різних рівнів доступу для кінцевого користувача, щоб забезпечити конфіденційну інформацію організації. Через те що доступ до таких систем реалізується через Інтернет, вони мають вищі ризики злому.

Також над оновлення та виправленням можливих помилок працює постачальник, і зазвичай додаткова оплата не налічується, але при будь-яких

проблемах виправити їх одразу та особисто не вийде, єдине що можна зробити це написати провайдеру та чекати коли він все виправить.

Не можна однозначно сказати який варіант краще, адже при використанні фізичного сервера компанія хоча і має можливість широкого контролю, та відсутність посередників управління своїми даними, але за це бере на себе зобов'язання по його конфігурації, обслуговуванню, повністю контрольоване розширення чи оновлення та вирішенню можливих проблем. Таке рішення підходить для банків, фінансових та державних установ. В той час як хмарні сервери зручні саме тим, що не потрібно витрачати час та гроші на контролювання працездатності, оновлення, розширення тощо. Також головна їх перевага це доступність у всіх місцях країни та світу, єдина вимога це мати стабільне підключення інтернету.

1.4 Обґрунтування теми дипломного проекту та постановка задачі

Підсумовуючи проведений вище аналіз існуючих програм для збереження та управління файлами, а також розглянувши методи збереження цих файлів, варто зазначити, що через те що не існує єдиного рішення, яке б підходило для різних компаній, то напрямок розробки таких систем для управління DA, та DMS тільки зростають. Пояснюється це тип, що потреби на використання електронного документообігу зростають. Універсального рішення не існує, адже різні компанії працюють з різними документами. Також і вимоги до зберігання даних теж відрізняються, наприклад, державні підприємства, які повинні дотримуватися суворої конфіденційності та не можуть мати доступу до Інтернету будуть використовувати локальне збереження, коли як для інших підприємств є важлива можливість віддаленої роботи з клієнтом, що буде реалізовано через хмарні сервери.

Виходячи з вищезазначеного, потрібна така система, яка б надавала можливість при додаванні документу створювати метадані з урахуванням потреб

користувача щодо різних місць зберігання цих даних. Також створювані метадані не є стандартними, оскільки залежатимуть від користувача.

Отже варто сформулювати функції які потрібно реалізувати:

- Створення метаданих, типів документу, ролей, директорій;
- Додавання метаданих до типу документу;
- Додавання до ролей доступні директорії, та типи документів, які можна додати;
- Додавання метаданих (згідно з типом) до файлу;
- Додавання файлу до глобального репозиторія; Можливість додавання файлів як через користувацький інтерфейс так і за допомогою API для можливості інтеграції з іншими системами;
- Можливість входу до системи як користувач з наданими правами (залежно від потреб: додавання, читання, внесення змін або видалення документів);

2 ВИБІР ТЕХНОЛОГІЙ І ЗАСОБІВ РОЗРОБЛЕННЯ СИСТЕМИ

Проєкт системи необхідно розділити на три логічні частини:

1) Front end — частина проєкту, яка відповідає за користувацький інтерфейс;

2) Back end — внутрішня частина реалізації;

3) База даних.

Такий поділ дозволяє реалізувати принцип separation of concerns – розділення відповідальності, тобто поділ програми на окремі розділи, кожен з яких призначений для реалізації окремих задач. Такий підхід спрощує роботу поділом великої задачі на розділ малих взаємопов'язаних задач, підвищує надійність і дозволяє використання окремих фрагментів повторно.

2.1 Вибір фреймворку для front end розробки

Фреймворк, за допомогою якого розроблений користувацький інтерфейс проєкту, Angular — відомий фреймворк з OS(Open-Source тобто програмне забезпечення з відкритим початковим кодом), який розробляється під керівництвом Angular Team у компанії Google. Головна причина його використання це інструмент Angular CLI якій значно полегшує та пришвидшує роботу з проєктами. З його використанням можна створити початковий проєкт, конфігурацію файлів для розгортання, шаблонні файли компонентів, навігацій та серверів, автоматично створити файли для тестування тощо. Створити новий проєкт можна тільки за допомогою однієї команди **ng new** + назва проєкту. Angular CLI автоматично створить проєкт, встановить залежності та повністю приготує для подальшої роботи. Angular не потребує часових витрат на налаштування архітектури проєкту, всі компоненти налаштовуються автоматично, тобто надається повноцінна архітектура та MVC (Model-view-controller тобто Модель–представлення–контролер, архітектура, яка складається з трьох модулів: модель даних, інтерфейс користувача та модуль керування).

Фреймворк відмінно підходить для маніпуляцій з великим обсягом даних та знижує ймовірність проблем, пов'язаних із неграмотно підготовленою архітектурою.

Також суттєва перевага Angular це Material UI, якій полегшує роботу з налаштуванням користувацького інтерфейсу, і хоча робить вигляд біль шаблонізованим, але і не потрібно багато часу для налаштування. Підтримує одностороннюю і двосторонню прив'язку даних. Двостороння прив'язка даних дозволяє при зміна у введенні інтерфейсу користувача, призведе до зміни у стані моделі, і навпаки.

Окрім цього велика і докладна документація, використання Typescript, Model-View-ViewModel, яка дозволяє одночасну роботу над одним фрагментом кільком людям, використовуючи одні й ті самі дані, інкрементний DOM (Document Object Model — Об'єктна модель документа), який зберігає DOM сам в собі, тобто не використовує додаткову пам’ять в процесі, що зменшує використання пам’яті.

2.2 Вибір мови back end розробки

Для розробки back end в першу чергу потрібно вибрати мову програмування. Вибираючи між двома мовами високого рівня C++ та Java, варто врахувати їх переваги та недоліки.

З переваг C++ однозначно швидкість роботи, свобода яка надається програмісту, підтримка парадигм узагальнення, об'єктно-орієнтованої, абстракції даних та інше. Також мова має так званий syntactic sugar - можливості синтаксису, які роблять код більш зрозумілим при читанні, а програмування в цілому легшим, при цьому не впливаючи на роботу програми.

Але недоліки значно переважають, головні з них це потреба постійно слідкувати за фрагментами пам'яті які були виділені динамічно, а також суттєві недоліки ієрархії класів, коли можливе наслідування від більш ніж одного класу,

що ускладнює роботу та може призвести до логічних помилок. А головний недолік це процес компіляції, який робить мову більш залежною від платформи.

Головна перевага Java — наявність JVM, Java Virtual Machine або віртуальна машина Java, програми яка слугує для запуску та виконання програм, що зробило її кросплатформенною і дозволяє запускати програми на різних операційних системах та на різних приладах. Початковий код JVM компільований у байт-код, а після він виконується за допомогою Bytecode Interpreter у самій JVM, яка виконує роль процесора, що розуміє інструкції на байт-код.

Спростити роботу допомагає наявність у JVM Garbage Collector (автоматичне управління пам'яттю, для запобігання витоку), що дозволяє зосередитися на розв'язанні поставленої задачі, алгоритмах та функціях, а не витрачати час на контроль пам'яті, що пришвидшує процес розробки.

Велика кількість стандартних бібліотек які доступні в процесі компіляції і їх не потрібно завантажувати додатково, а також, для програм, які написані цією мовою є дуже багато OS бібліотек, які надалі розвиваються, та мають гарну підтримку через спільноту, і відповідь на проблему можна знайти в Google, або запитати на платформах типу Stack Overflow. З переваг Java також простіший синтаксис, що спрощує написання та розуміння коду, і суворо типізованість мови, що допомагає уникнути багатьох помилок.

2.3 База даних та інструменти для роботи з базами даних

Як система управління базами даних була використана PostgreSQL з OS, підтримує багатоплатформність, як звиклий реляційний підхід (інформація зберігається у постаті запису які пов'язані між собою), так і об'єктний, тобто зберігання даних в постаті об'єкту, атрибутів, клас тощо.

PostgreSQL підтримує багато різних типів, зокрема, булеві значення, бінарні, грошові, геометричні та інші. Також підтримує складні та складені запитання, які містять в собі одночасно читання, запис і перевірку. Підтримує

написання функцій різними мовами програмування, наприклад: C, C++, Java, Python, PHP тощо. Підтримує також і менш популярні мови програмування, наприклад: Delphi, Lisp.

Можливість роботи над одними даними кількома особами, що реалізовано за допомогою MVCC — Multiversion Concurrency Control, технологія полягає у тому, що користувачі не працюють напряму над даними, а тільки з так званим снапшотем, що є “знімком” даних і тільки після зафіксованої транзакції надходять до початкової бази даних. Відповідає принципам ACID - Atomicity, Consistency, Isolation, Durability — атомарність, узгодженість, ізольованість, міцність, що робить її прогнозованою та надійною, зменшує ризики конфліктів системи.

Як клієнт та інструмент управління базою даних було використано DBeaver. Зручна програма з інтуїтивно зрозумілими інтерфейсом, має можливість роботи зі структурами та моделями баз даних, а також автоматично створені діаграми до кожної таблиці, які відповідають за зв’язок між сутностями.

Для створення таблиць було ужито Liquibase — бібліотеки OS, для роботи з міграціями баз даних (виконання команд визначення даних — створення таблиць, ключів та унікальних значень, індексів тощо), можливість відстеження змін. Така реалізація позбавляє потреби створення бази даних безпосередньо у коді, а виносить все в окремий файл.

3 ОПИС СЕРЕДОВИЩА РОЗРОБКИ СИСТЕМИ

Першим кроком є встановлення SDK — Software Development Kit — набору інструментів для розробників, орієнтованих на платформу, що містить в собі вже готовий набір для розробки, наприклад, компілятор, відладгоджувач та бібліотеку тощо. Для даної розробки в мові Java було використане рішення, яке є доступне на сайті фірми Oracle, яка займається розбудовою та підтриманням мови. Пакет JDK(Java Development Kit) включає, серед іншого: компілятор Java і віртуальну машину JRE (Java Runtime Environment) , яка повинна бути підібрана відповідно до архітектури процесора та операційної системи. У даному випадку було встановлено рішення для 64-розрядного середовища Windows. Вже після встановлення середовищі JDK завантаження та налаштування інших компонентів - Eclipse IDE, Angular, Visual Studio Code, Maven, Keycloak, WildFly.

Для розробки використано інтегроване OS середовище розробки програмного коду Eclipse IDE. Eclipse вибрано за легкість встановлення та використання, має можливість підключення до різних баз даних, великий час існування, а отже велика спільнота та кількість інформації в інтернеті у випадку пошуку розв’язання проблеми чи оптимізації, легка інтеграція з Git та Maven, а також безкоштовне використання, на відміну від IntelliJ IDEA Ultimate. Завантажити можна з офіційного сайту.

Окрім середовища для розробки у back end, потрібне середовище для front end, для якого обрано фреймворк Angular, інструкція для завантаження доступна на офіційному сайті. Для роботи з фреймворком буде використаний безкоштовний редактор вихідного коду - Visual Studio Code, який містить також функції підтримку відладгоджувача, підсвічування синтаксису та має багато розширень які можна швидко встановити не виходячи з редактору. Для завантаження доступний на офіційному сайті.

Для автоматизації збірки використаний інструмент Maven створений Apache Software Foundation. Для побудови використовує інструкції описані в

файлах POM кожного з модулі і самостійно встановлює залежність від зовнішніх модулів, компонентів та визначає порядок збірки, також можливе виконання компіляції коду та пакетування з використанням заздалегідь зазначених цілей. Завантажити Maven можливо з офіційного сайту використовуючи інструкції, дотримуючись вимог які потрібні для коректної роботи. При використанні операційної системи Windows варто звернути увагу на додаткові вимоги в окремому розділі, що містяться також на сайті.

Keycloak OS інструмент для налаштування рівня безпеки в програмі, що переносить все управління правами, доступом і логування за програму. Keycloak надає можливість самостійного управління профілями користувачів, прив'язки до користувачів атрибутів, ролей чи токенів. Має можливість інтеграції, наприклад з Google, GitHub, Facebook, Twitter.

В моменті логування користувач буде перенесений на екран логування Keycloak, і вся автентифікації користувача відбувається в незалежній, від основної, програмі. Після вдалого залогування, до основної програми повертається токен, якій містить інформацію яка описує користувача, тобто авторизаційні дані, на підставі яких розпізнаємо права користувача. Завантажити можна з офіційного сайту де міститься також і інструкція.

WildFly — OS сервер додатків на платформ Java EE, розроблений компанією JBoss. Він призначений для виконання різних завдань, таких як вебсервер, сервер додатків та багато іншого. WildFly є одним із найпопулярніших серверів додатків у світі і одним з найвідоміших для Java EE. Обрано було за високу продуктивність, що забезпечує ефективну роботу завдяки оптимізації внутрішньої архітектури. Покращує масштабованість програми, що досягнена за допомогою кластеризації та розподіленої обробки і дозволяє обробляти велику кількості даних та запитів. Має широкий спектр можливостей та налаштувань, що дозволяє підлаштовувати роботу під потреби конкретного проєкту і керувати його роботою. Завантажити можна з офіційного сайту, але підібрати версію відповідно до версії платформи JEE.

4 ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ УПРАВЛІННЯ ДОКУМЕНТАМИ

На рис. 8 зображено загальну архітектуру системи та її основні елементи, що складають екосистему VMP (назви програми “VMP” - Veronika Mamai Program`s).

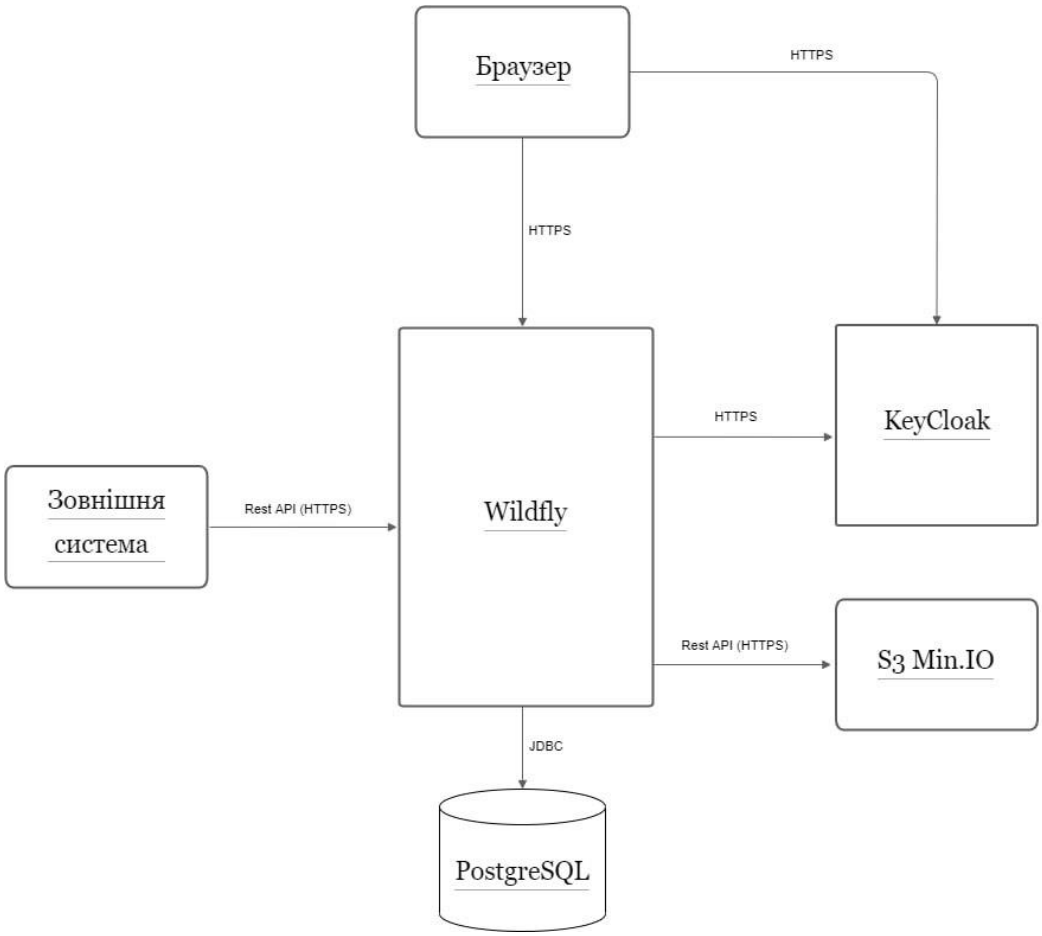


Рисунок 8 – Архітектура системи

Доступ до програми здійснюється через браузер, з користувацького інтерфейсу, також можливий шляхом інтеграції із іншою зовнішньою системою.

При тестуванню використовувався додаток безкоштовний HTTP-клієнт для тестування Postman.

Програма використовує базу даних для зберігання метаданих документів і конфігурації системи. Система безпеки (в частині аутентифікації та авторизації) перенесена в окремий і не залежний від основної програми додаток Keycloak.

4.1 Конфігурація та підготовка проєкту

Головний проєкт Maven складається з 7 основних підпроектів (а також передбачено два додаткових модулі, які потрібні для побудови docker image). Префікс “Vmp-” наданий від назви програми “VMP” - Veronika Mamai Program`s. API — Application Programming Interface – прикладний програмний інтерфейс.

Модулі проєкту та їх призначення

Api — частина проєкту, яка відповідає за встановлення кінцеві точки вебсервісу згідно зі специфікацією JAX-RS (Jakarta RESTful Web Services специфікація API Jakarta EE). Використовується для надання підтримки у створенні вебслужб відповідно до архітектурного шаблону REST — Representational state transfer або передача репрезентативного стану. Також відповідає за створення вебдодатку і інтегрує в собі модуль ui. Кінцевим результатом роботи цього модулю є цілий веб додаток спакований до архіву типу WAR.

Api-ext — подібно, як модуль api, містить в собі кінцеві точки REST, але передбачені для інтеграції з іншими системами. Структура кінцевих точок REST, тобто доступні методи API та формати обміну даними, в цьому модулі були визначені за допомогою інструменту Swagger/OpenAPI. На основі визначення, що міститься у файлі формату JSON, плагін «swagger-maven-plugin» генерує класи об’єктів, т.зв. DTO — Data Transfer Object — об'єкт передачі даних та скелет реалізації для серверної частини API, що надається програмою. Специфікація OpenAPI — це інструмент, який документує створений API, а також дозволяє генерувати аналогічні об’єкти на стороні клієнта API. Інструменти для генерації коду в стандарті OpenAPI доступні для найпопулярніших мов (Java, C#, JavaScript/Typescript, Python тощо), що дозволяє легко інтегрувати системи, розроблені в різних технологіях. Приклад визначення об’єкта, який використовується для обміну даними в рамках спільного API. На рис. 9 показано визначення об’єкта Document із вбудованими атрибутами (Id,

size, content_type...) і список динамічних атрибутів, описаних далі у файлі з класом DocumentAttribute.

```
    "definitions": {
        "Document": {
            "type": "object",
            "description": "Документ який зберігається у VMP",
            "properties": {
                "id": {
                    "type": "integer",
                    "format": "int64",
                    "description": "Ідентифікатор файлу"
                },
                "fileName": {
                    "type": "string",
                    "description": "Ім'я файлу"
                },
                "contentType": {
                    "type": "string",
                    "description": "Тип вмісту файлу"
                },
                "size": {
                    "type": "integer",
                    "format": "int32",
                    "description": "Розмір файлу"
                },
                "directory": {
                    "type": "string",
                    "description": "Каталог"
                },
                "documentType": {
                    "type": "string",
                    "description": "Тип документа"
                },
                "content": {
                    "type": "string",
                    "description": "Вміст файлу закодовано за допомогою base64"
                },
                "attributes": {
                    "type": "array",
                    "items": {
                        "$ref": "#/definitions/DocumentAttribute"
```

Рисунок 9 – Опис об’єкту Document

Визначення знаходиться у файлі ExternalAPI.json у даному модулі, що описує одне з доступних кінцевих точок Rest. Також в даному файлі міститься опис відповідей на виклик API. На рис. 10 фрагмент коду зі способом отримання метаданих документа.

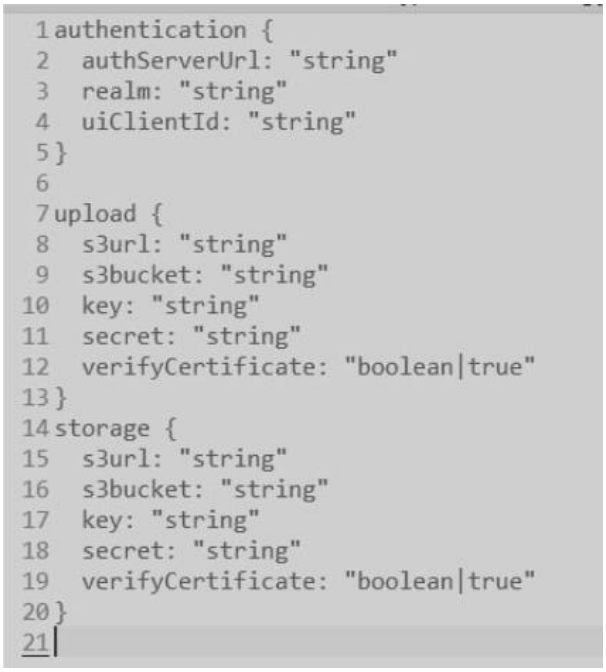
```
"paths": {
    "/files/{id}": {
        "get": {
            "operationId": "retrieveDocument",
            "summary": "Отримати файл",
            "description": "Ця операція отримує файл, що зберігається у VMP",
            "tags": [
                "document"
            ],
            "parameters": [
                {
                    "name": "id",
                    "description": "Ідентифікатор документа",
                    "required": true,
                    "type": "integer",
                    "format": "int64",
                    "in": "path"
                },
                {
                    "name": "data",
                    "description": "Позначте, чи включати дані документа",
                    "required": true,
                    "type": "boolean",
                    "in": "query"
                }
            ],
            "responses": {
                "200": {
                    "description": "Success",
                    "schema": {
                        "$ref": "#/definitions/Document"
                    }
                },
                "401": {
                    "description": "Unauthorized",
                    "schema": {
                        "$ref": "#/definitions/Error"
                    }
                },
                "500": {
                    "description": "Internal Server Error",
                    "schema": {
                        "$ref": "#/definitions/Error"
                    }
```

Рисунок 10 – Опис відповіді на виклик API

У разі успіху (http код 200) повертається об’єкт класу Document. Крім того, якщо під час виклику функції встановлено параметр «data» = «true», вміст двійкового файлу, перетвореного за допомогою коду Base64, повертатиметься разом із метаданими.

App — це модуль, якій відповідає за створення архіву типу EAR в який згрупує всі компоненти програми. Буде розгорнутий в сервері WildFly.

Config — встановлення конфігурації програми. Конфігурація побудована з використанням OS бібліотеки Typesafe Config. На рис. 11 показано специфікацію формату конфігурації, що записана у specification.conf даного модуля.



```
authentication {
  authServerUrl: "string"
  realm: "string"
  uiClientId: "string"
}

upload {
  s3url: "string"
  s3bucket: "string"
  key: "string"
  secret: "string"
  verifyCertificate: "boolean|true"
}
storage {
  s3url: "string"
  s3bucket: "string"
  key: "string"
  secret: "string"
  verifyCertificate: "boolean|true"
}
```

Рисунок 11 – Специфікація формату конфігурації

В системі створено три групи параметрів, перший з яких відповідає за встановлення з'єднання з Keycloak та два з репозиторієм бінарних файлів у стандарті S3.

Для збереження файлів створено два окремих репозиторії. Upload призначений для тимчасового збереження, запис до цього буде відбуватися при додаванні файлу у вікні додавання файлу та метаданих (де також обирається директорія та тип файлу). В даному репозиторії файли будуть зберігатися тимчасово, звідки будуть видалені за визначений проміжок часу (можна визначити при налаштуванні MinIO).

Другий репозиторій storage призначено для довгострокового зберігання, до нього файл буде збережений при закінченні форми додавання документу разом з вибраною директорією та типом файлу, а також з правильно виконаними метаданими (якщо такі є). Конфігураційні дані будуть подані до серверу в моменті старту у файлі application.conf. Файл виконується за допомогою додаткового параметру для JVM „-Dconfig.file=C:\vmp\ma-vmp\etc\config\application.conf". Для даного проєкту подано наступні значення параметрів конфігурації, які можна побачити у файлі application.conf на рис. 12.

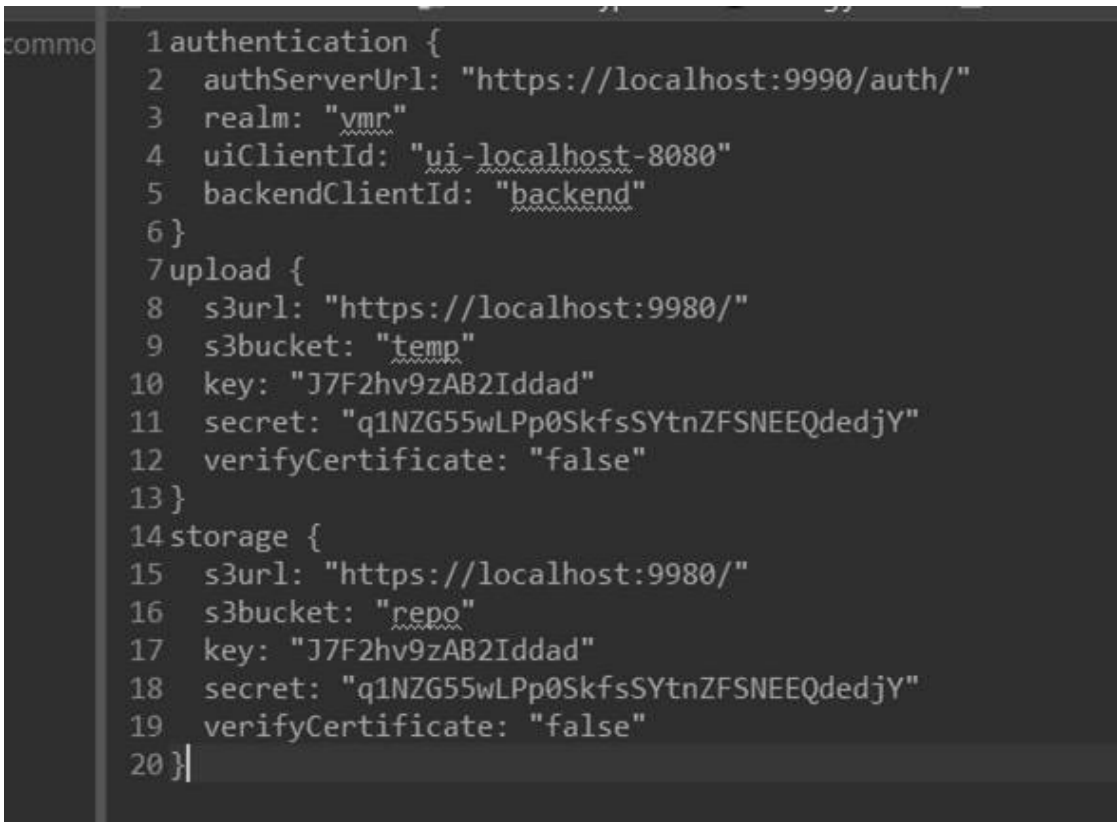


Рисунок 12 – Визначення параметрів конфігурації

Ejb — логіка програми, містить головні біни сесії програми.

Jpa-model — відповідає за роботу з базою даних, в цьому модулі знаходяться класи зіставлення структури реляційної бази даних на класи у вигляді об’єктів-клас Java.

Ui — частина яка містить в собі елементи програми типу SPA (Single Page Application — односторінковий застосунок), Angular який виповнює функцію інтерфейсу користувача частина front end.

На початку роботи приготовано структуру проекту, створюючи модулі в вигляді підпроектів Maven, всі компоненти задекларовані у pom.xml в корні проєкту. POM — Project Object Model або об’єктна модель проекту, XML-файл, який містить інформацію про залежності, конфігурації та іншу важливу інформацію про проект і слугує для опису проєкту на підставі якого будуть виконуватися всі операції Maven. В головному тегу project записана вся інформація про проєкт, така як версія POM, унікальні ідентифікатори groupId и artifactId, properties. Оголошення залежностей між модулями проєкту, між залежностями до зовнішніх бібліотек, зі специфікаціями плагінів, які будуть

встановлені в процесі побудови програми. На рис. 13 в тегу modules задекларовані всі модулі проєкту.

```
43
44    <modules>
45        <module>modules/jpa-model</module>
46        <module>modules/config</module>
47        <module>modules/ejb</module>
48        <module>modules/api</module>
49        <module>modules/api-ext</module>
50        <module>modules/app</module>
51        <module>modules/ui</module>
52        <module>images/platform</module>
53        <module>images/app</module>
54    </modules>
55
```

Рисунок 13 – Декларування модулів проєкту

При збірці проєкту важливу частину роботи виконує так званий reactor, частина ядра Maven, яка збирає всі доступні модулі для збірки, потім сортує у правильному порядку, в залежності від вказання того, який модуль потребує перед збіркою вже готового іншого. На рис. 14 наведено залежність між модулями програми.

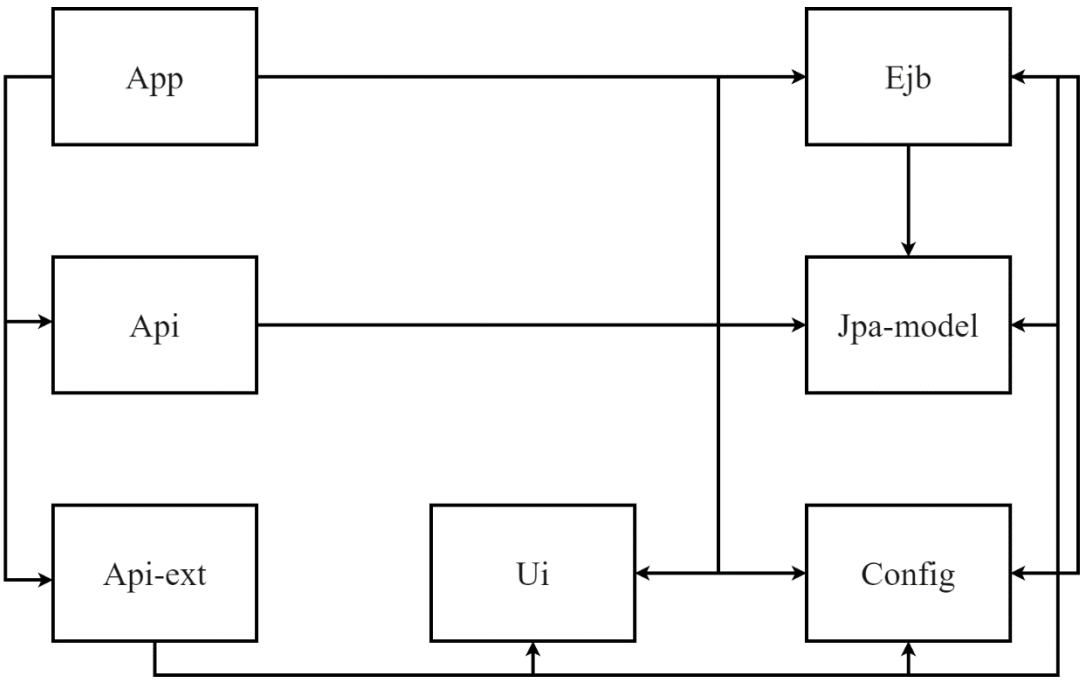


Рисунок 14 – Залежності між модулями

Також reactor враховує оголошення плагінів, які можуть бути додатковими модулями, та їх залежностей від інших модуля у збірці. Останнім кроком є збірка проєкту в визначеному порядку, який можна побачити в інформаційних повідомленнях у консолі при початку роботи Maven (рис. 15).

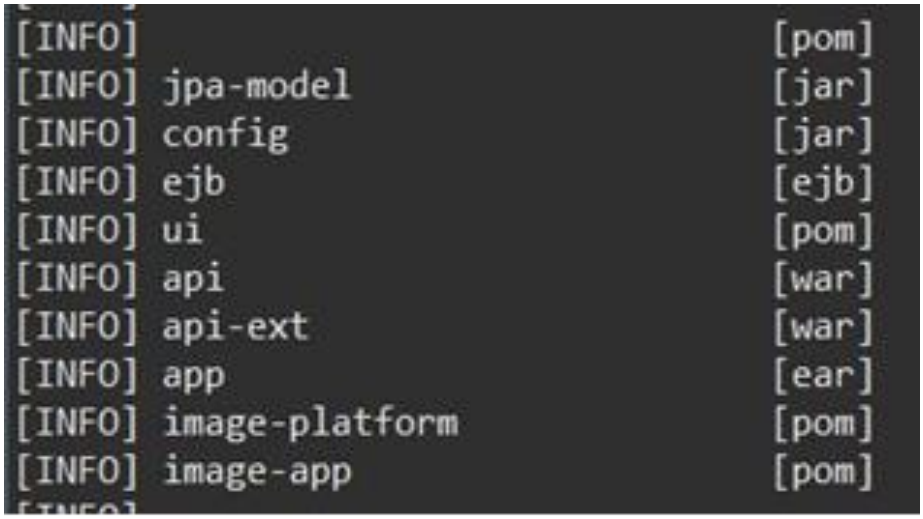


Рисунок 15 – Визначений Maven порядок збірки проєкту

4.2 Створення бази даних та об’єктів зіставлення Java класів

4.2.1 Проектування та створення бази даних

Кожна таблиця має колони creator, created, modified, які відповідають послідовно назві користувача, який додавав запис до таблиці, час коли він був доданий вперше та час останньої зміни запису.

На рис. 16 зображена таблиця roles з двома допоміжними штучно створеними таблицями для зв’язку з іншими таблицями.

Таблиця roles містить запис назви ролі користувача, яку показано в системі у колоні display_name, колона domain_group_name відповідає за приписання користувача до групи, обидві колони повинні бути унікальними, отже створено додатково унікальний index для них. Права надаються згідно з записом в Keycloak. Ця таблиця пов'язана з таблицями document_types та directories, для зв'язку створені окремі допоміжні таблиці role_document_types та role_directories.

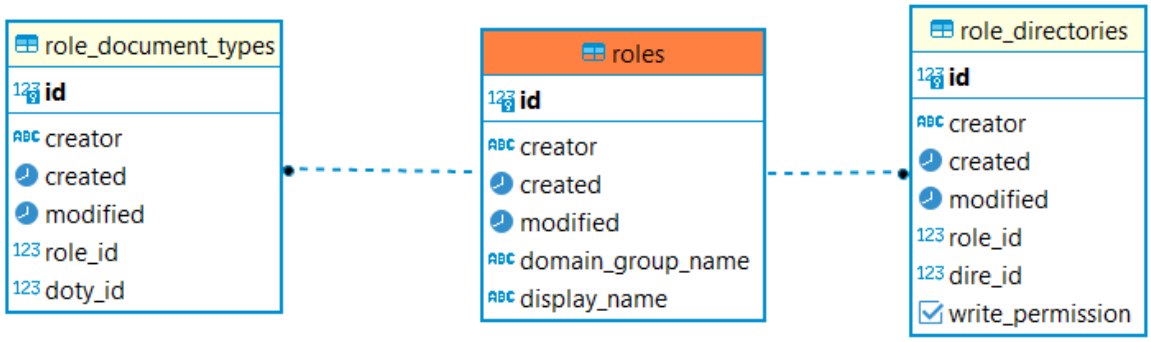


Рисунок 16 – Вигляд таблиці roles та зв'язків

Записи у першій відповідають за надання дозволу, якого саме типу документи можна додати. Назва для референцій до інших таблиць складається або з чотирьох літер + „_id” на кінці. Це може бути або перших літер назви таблиці, якщо це одне слово, як role_id від **role**s, або якщо з двох слів то перші дві літри кожного, як у doty_id від **do**cument_**ty**pes. За цим принципом названі всі колони референцій. Таблиця role_document_types містить унікальний index для поєднання колон role_id та doty_id.

Записи у role_directories відповідає за директорію у яку користувач може додавати файли, а також містить інформацію про дозвіл запису у колоні write_permission. Ця таблиця має унікальний index на поєднання колон role_id та dire_id.

Користувач має можливість створювати типи файлі які він бажає додати, ця інформація міститься в таблиці document_types, колони name, де буде записане ім’я у системі, може не співпадати з display_name — ім’я яке буде видно користувачу через інтерфейс. Для колон display_name та name створений унікальний index.

Таблиця пов'язана з трьома іншими штучними таблицями, як-от: document_type_attributes, documents, role_document_types. Таблиця document_types разом зі зв’язками на рис. 17.

Штучно створена додаткова таблиця document_type_attributes відповідає за зв'язок з таблицею attribute_definitions, де будуть створені динамічні метадані, що будуть приписані до відповідного типу документу.

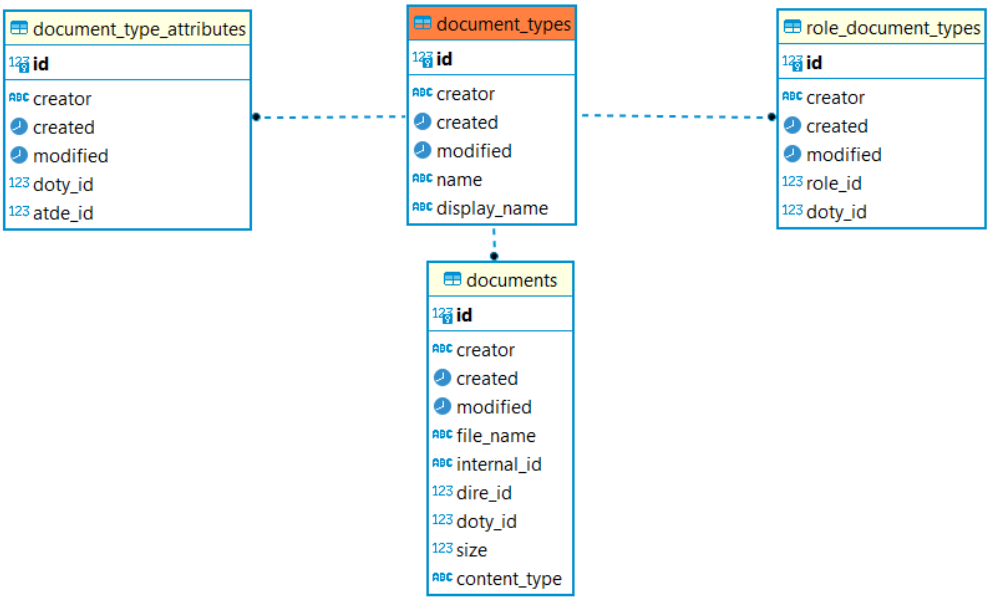


Рисунок 17 – Зображення таблиці document_types та зв'язків

Варто звернути увагу на такі стовпці як:

- shown — булева колона, створена для відображення можливості приховати раніше визначені та використані атрибути. Призводить до видалення атрибута з доступних, але без його фізичного видалення з бази даних;
- required — прапорець для валідаторів, які перевіряють при збереженні файлу чи він обов'язковий, якщо атрибут не заповнений — блокує кнопку "зберегти" або відхиляє запит через API;
- display_priority — порядкова позиція у відображенні на формі метаданих;
- data_type — тип атрибуту: L — словник, B — boolean, T — текст, N — число, D — дата;
- input_max_length — максимальна кількість знаків для тексту;
- input_validation_rule — вказівка на конкретне правило перевірки, наприклад, перевірка номера телефону або формату номера паспорта чи ІПН;
- input_validation_regexp — регулярний вираз;
- input_min_value та input_max_value — мінімальне і максимальне значення;

- input_precision — вказівка для числа, скільки знаків після коми дозволено для дробового числа. Перевіряємо за таким правилом: ^[0-9]+([,.][0-9]{0,${definition.inputPrecision}})?$;
- lide_id — id до таблиці list_values.

На рис. 18 зображена таблиця та пов'язані з нею, де будуть зберігатися метадані в залежності від їх типу. Створено унікальний індекс на поєднанню atde_id та doty_id.

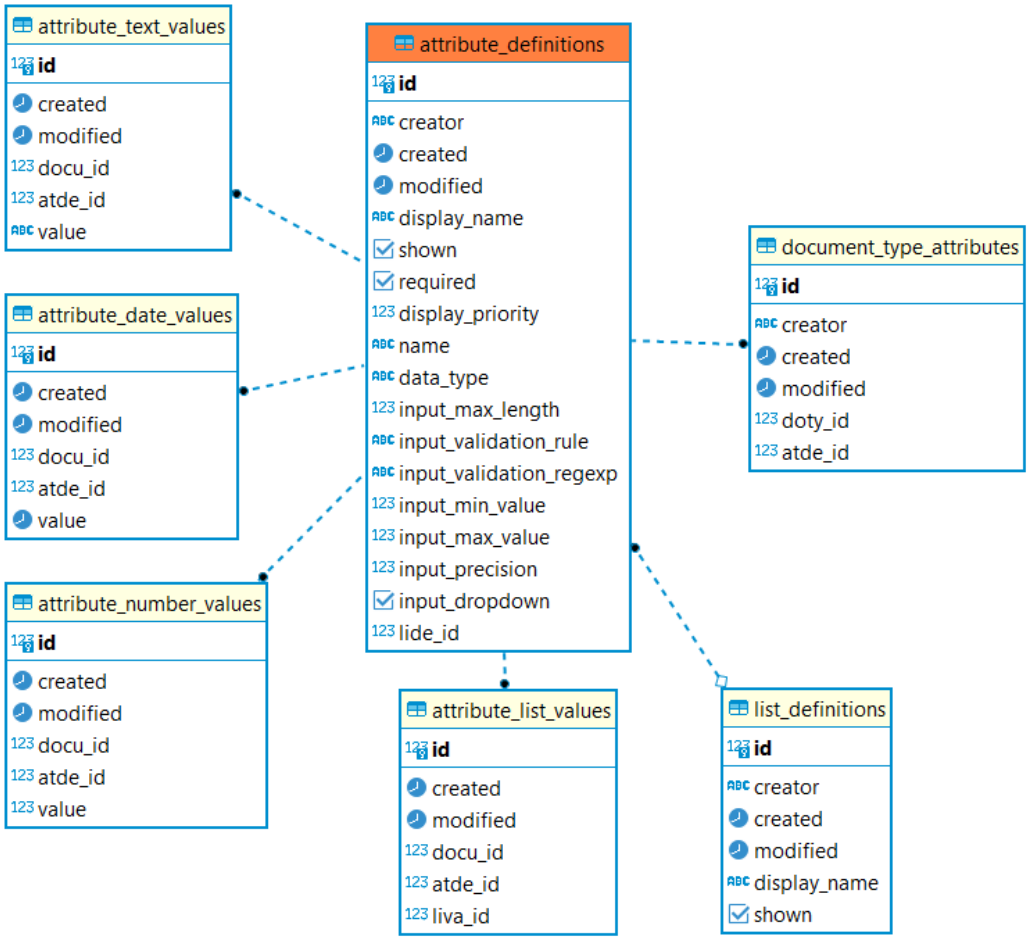


Рисунок 18 – Таблиця attribute_definitions та зв'язків

У таблиці attribute_text_values будуть збережені метадані типу текст, де саме значення буде міститись у value, atde_id відповідає за attribute_definitions де описане метадане, а docu_id відповідає за document_types, створено унікальний index на поєднанню колон docu_id та atde_id. Майже аналогічно з таблицями attribute_date_values, attribute_number_values, list_values тільки перша призначена для метаданих зі значеннями дат, друга для цифр та булевих типів, а

list_values затомість колони value має liva_id, де міститься референція до list_values, значення словника.

Таблиця list_definitions містить назви словників, окрім стандартних колон, має також колону swhon. Має зв'язок з attribute_definitions та list_values, де міститься значення словника. Створений унікальний index на display_name.

Для користувача повинна бути доступна директорія, яка може бути виділена відповідно до його ролі, щоб розділити простір де будуть знаходитися файли. На рис. 19 зображена таблиця directories та пов’язані з нею role_directories та documents.

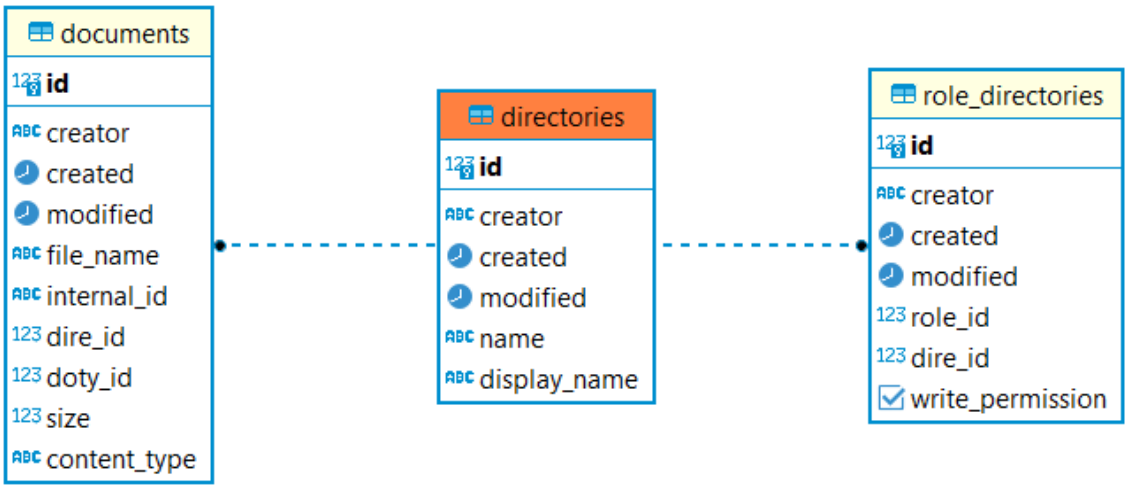


Рисунок 19 – Таблиця directories та зв'язків

Ця таблиця містить дві колони для назви директорії — name та display_name, для запису робочої назви директорії у системі чи внутрішніх документах компанії та назви яку буде видно користувачу відповідно. Для обох колон створено index, для дотримання унікальності.

Таблиця documents містить інформацію про збережений файл, окрім size (розмір файлу) та file_name (назва завантаженого файлу) присутні наступні колони:

- internal_id — відображає унікальний ідентифікатор, який надається при збереженні до хмари/серверу;
- dire_id — id таблиці directories, до якої збережено файл;

- doty_id — id таблиці document_types, де зазначений створений через користувача тип файлу;
- content_type — вид та розширення файлу.

Створено унікальний index на internal_id та поєднанню file_name і dire_id (не можна додати два файли з однаковою назвою в одну директорію). Окрім таблиць directories та document_types, має пов'язання з таблицями де міститься значення метаданих відповідно до типів, що продемонстровано на рис. 20.

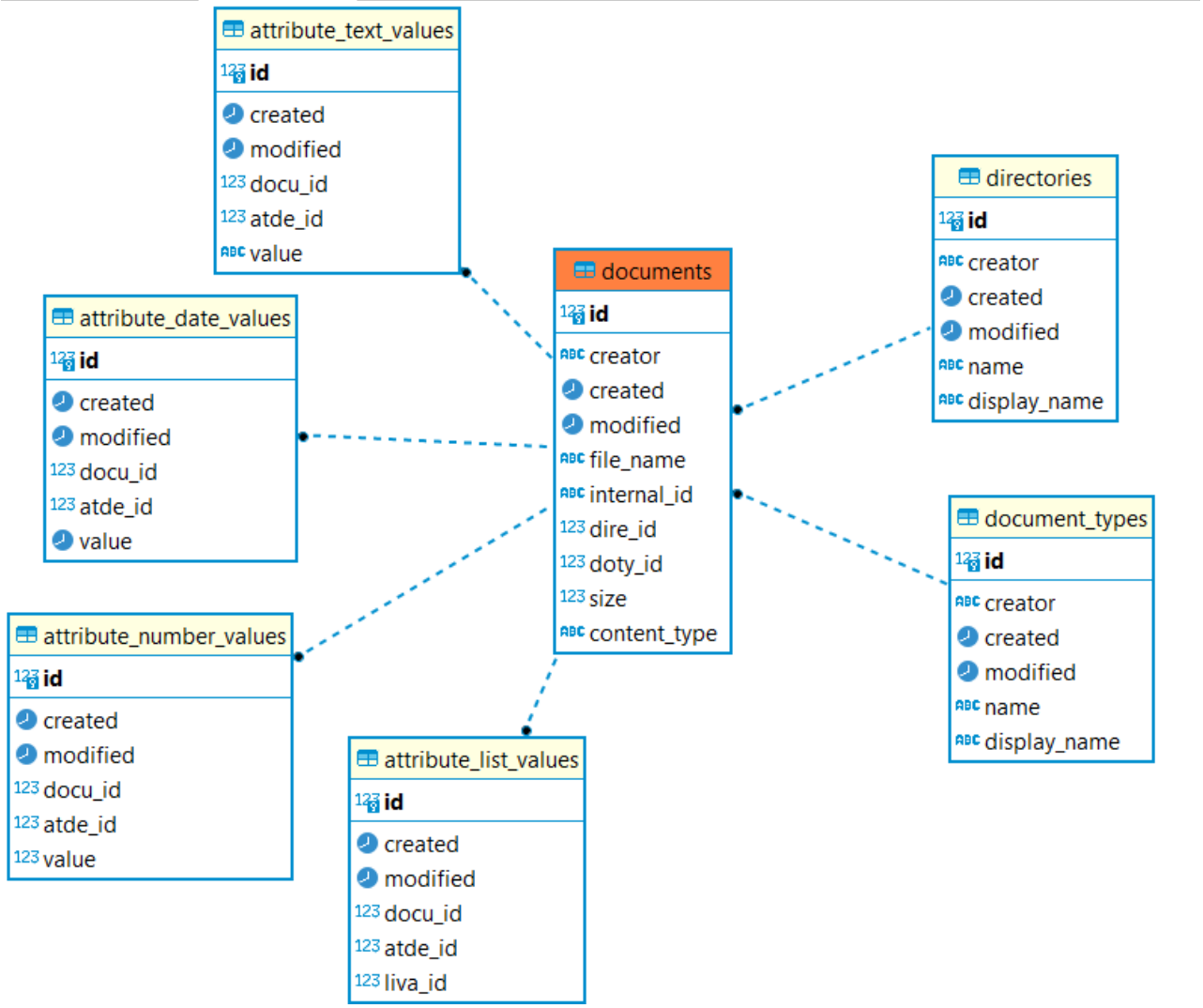


Рисунок 20 – Таблиця documents та зв'язків

Повну діаграму зв'язку сутності для системи можна побачити на рис. 21.

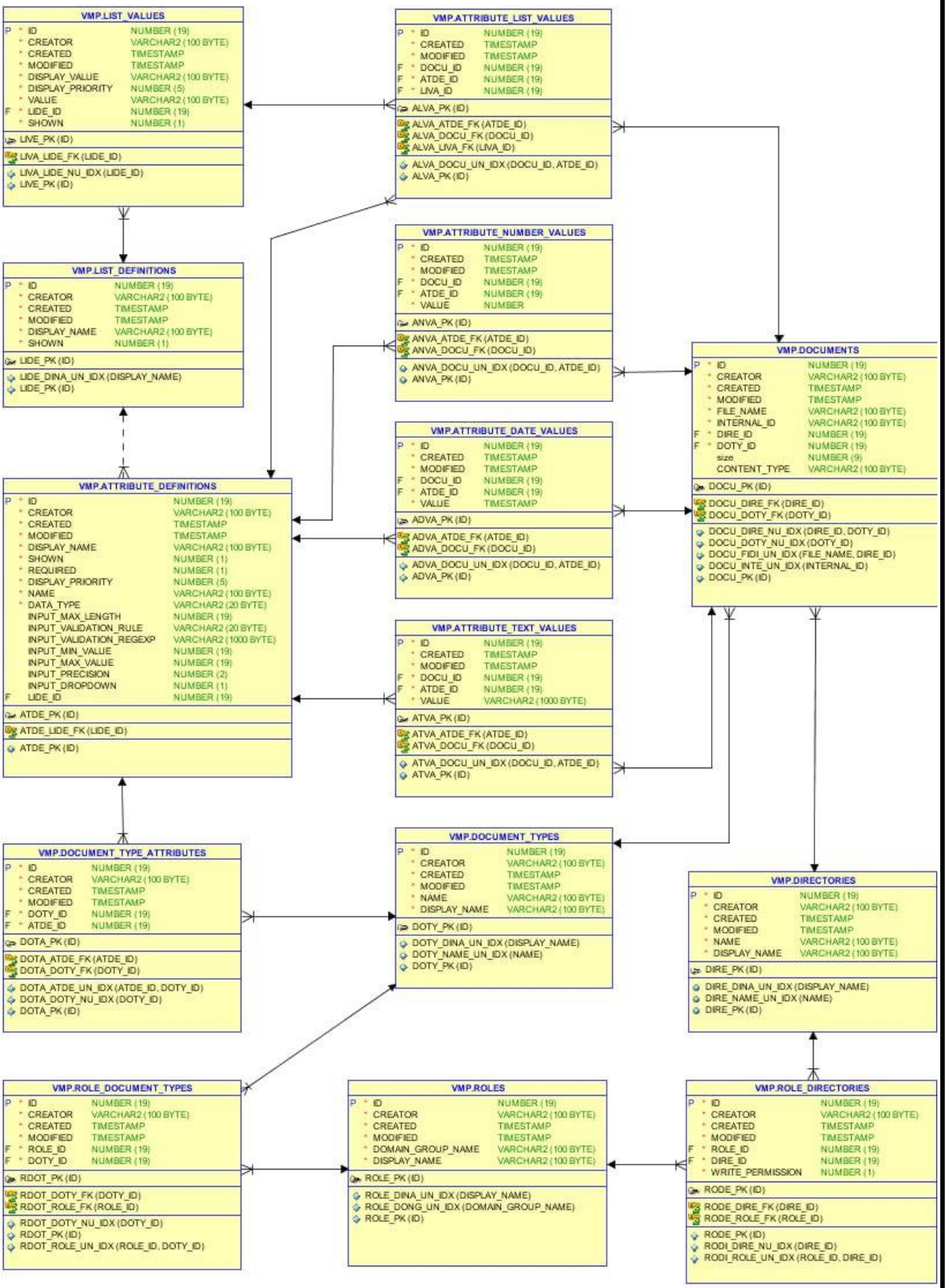


Рисунок 21 – Діаграма системи

Таблиці у базі даних створені за допомогою бібліотеки Liquibase, ціла схема описана у файлі model-changelog.xml в модулі jpa-model. При першому запуску Liquibase автоматично утворить таблицю DatabaseChangeLog, де будуть записані всі changeset, що потрібні для відслідковування змін, та таблицю DatabaseChangeLogLock для контролю одночасного запуску на одній і тій самій базі. На рис. 22 створення таблиці roles у model-changelog.xml.

```
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/n

    <changeSet author="mami" id="vmp-1-1">
        <createTable tableName="roles">
            <column name="id" type="number(19)">
                <constraints primaryKey="true" primaryKeyName="role_pk" />
            </column>
            <column name="creator" type="varchar2(100)">
                <constraints nullable="false" />
            </column>
            <column name="created" type="timestamptz">
                <constraints nullable="false" />
            </column>
            <column name="modified" type="timestamptz">
                <constraints nullable="false" />
            </column>
            <column name="domain_group_name" type="varchar2(100)">
                <constraints nullable="false" />
            </column>
            <column name="display_name" type="varchar2(100)">
                <constraints nullable="false" />
            </column>
        </createTable>
    </changeSet>
    <changeSet author="mami" id="vmp-1-2">
        <createSequence sequenceName="roles_seq" />
    </changeSet>
    <changeSet author="mami" id="vmp-1-3">
        <createIndex indexName="role_dina_un_idx"
            tableName="roles" unique="true">
            <column name="display_name" />
        </createIndex>
    </changeSet>
```

Рисунок 22 – Приклад створення таблиці у модулі jpa-model

ChangeSet з id “vmp-1-1” описує таблицю, яку треба створити, “vmp-1-2” відповідає за створення секвенції для неї, “vmp-1-3” накладання унікального значення на колону display_name.

Такий спосіб створення схеми передбачає використання, як Oracle так і PostgreSQL, але варто зауважити різницю при створенні колони з буліновським значення. При використанні Oracle неможливо створити колону зі значеннями

boolean, для цих цілей використовується тип CHAR(1) з обмеженням використання значень тільки “Y” та “N” або NUMBER(1) з двома можливими значеннями “1” та “0”. Для уникнення проблем додатково створена перевірка на використання Oracle, і якщо було використано — накладається обмеження на ті колони, які повинні бути boolean. Приклад такої перевірки на рис. 23.

```
<changeSet author="mami" id="dmg-1-9-1">
    <preConditions onFail="MARK_RAN">
        <dbms type="oracle" />
    </preConditions>
    <sql>
        ALTER TABLE role_directories
        ADD CONSTRAINT rode_wrpe_CHK CHECK
        (write_permission IN (1, 0))
    </sql>
</changeSet>
```

Рисунок 23 – Перевірка СУБД для створення колони типу boolean

4.2.2 Генерація зіставлення Java класів з таблицями бази даних

Для створення зіставлень було використано JPA Tools, зручний плагін для генерації зіставлень або таблиць по вже існуючому зіставленню, окрім PostgreSQL підтримує також Oracle, MySQL, Ingres, Max DB та інші.

На початку було створено з'єднання з базою даних. Після отримання повідомлення про успішне з'єднання наступним кроком є обрання таблиць для яких буде створено зіставлення. Для даної системи потрібні всі таблиці, створені в базі даних, окрім допоміжних databaseChangeLog та databaseChangeLogLock.

У класах зіставлення генератор первинного ключа SequenceGenerator буде створений тільки з назвою. Для автоматичного генерування sequenceName (ім'я об’єкта послідовності бази даних, з якого буде отримане значення первинного ключа) після кроку вибору таблиць обрано секвенцію як генератор ключів, а назва буде створена від назви таблиці з написаним на кінці “_seq”, що видно на рис. 24 разом з іншими налаштування.

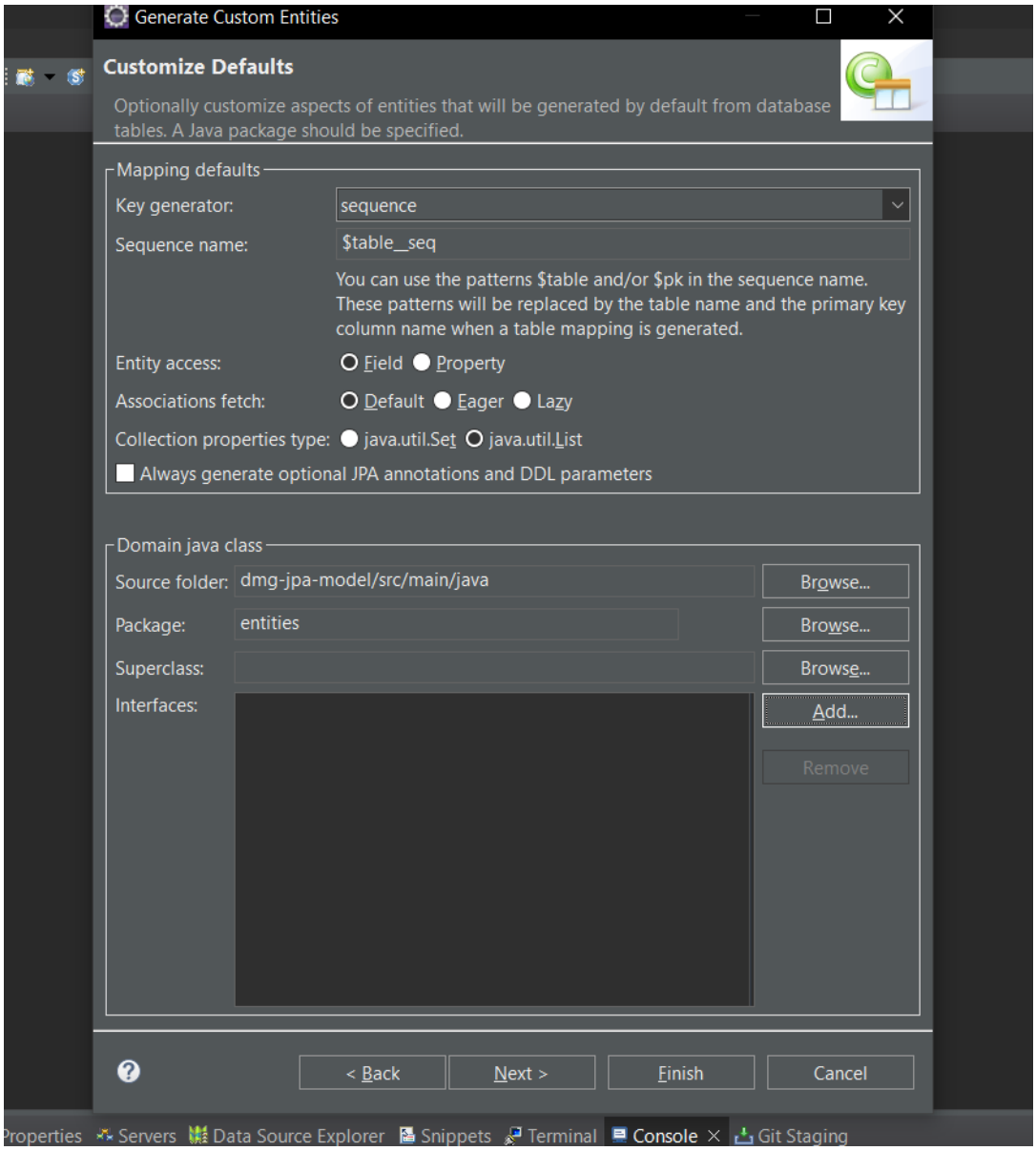


Рисунок 24 – Налаштування для генерації зіставлень

Вже після створення зіставлень за замовчуванням кожний наступний елемент в секвенції буде з різницею у 50, але щоб цього уникнути додано до SequenceGenerator allocationSize = 1, тоді секвенція була з кроком 1. На рис. 25 зображено, як буде виглядати остаточний генератор.

```
24
25●    @Id
26     @SequenceGenerator(name = "ROLES_ID_GENERATOR", sequenceName = "ROLES_SEQ")
27     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "ROLES_ID_GENERATOR")
28     private long id;
```

Рисунок 25 – SequenceGenerator у класі зіставлення таблиці Roles

Також генератор трактує number тип з бази даних як BigDecima, що було у багатьох випадках замінено в залежності від змінних, їх розміру, типу та необхідності на Integer/int/Long/long. Так змінні, які повинні завжди мати значення (наприклад, всі головні ключі Id) повинні бути типу long або int, тоді, як змінна, яка може бути відсутня замінена на Integer/Long. Це пов'язано з тим, що у мові програмування Java значення null може бути надане об'єкту класу-обгортки, а у випадку з примітивним типом даних це призведе до помилки NullPointerException.

Для класи AttributeDefinition зміна dataType відповідає за тип метаних. При зіставленні тип з бази даних VARCHAR(20) буде генератором перетворений у String, що не зовсім відповідає потребам, отже додано анотацію @Enumerated(EnumType.STRING) і надано тип DataType (enum створений у модулі jpa-model в пакеті DTO, рис. 26).

```
3  public enum DataType {
4      /** BOOLEAN */
5      B,
6      /** DATE */
7      D,
8      /** LIST */
9      L,
10     /** NUMBER */
11     N,
12     /** TEXT */
13     T
14 }
```

Рисунок 26 – DataType enum

У кожній класі присутні лінії які відносяться до колон created та modified, для уникнення повторення вони розширені новоствореною класою SimpleBaseEntity, що слугує для зіставлення цих колон. Колона created має додатково у анотації визначення зіставлення updatable = false (відповідає чи включено стовпець до інструкцій SQL UPDATE, що згенеровані persistence provider). У класі присутні функції get, updateModified для приписання оновленого значення змінної modified, а також дві функції з анотаціями. Перша це prePersist з @PrePersist, анотація відповідає за те, що функція, перед якою вона

вказана, буде викликана при створенні нового об'єкта. У середині цієї функції змінним created та modified присвоєно значення функції Instant.now() (для отримання поточного миттєвого часу від системного годинника всесвітнього координованого часу).

Друга функція preUpdate з анотацією @PreUpdate. Слід зазначити, що така функція викликається, лише якщо дані фактично змінено, тобто якщо існує фактичний оператор оновлення SQL для виконання. У даній функції modified присвоєно значення функції Instant.now(). На рис.27 можна побачити класу SimpleBaseEntity.

```
10 @MappedSuperclass
11 public abstract class SimpleBaseEntity {
12
13     @Column(name = "CREATED", updatable = false)
14     private Instant created;
15
16     @Column(name = "MODIFIED")
17     private Instant modified;
18
19     public Instant getCreated() {
20         return created;
21     }
22
23     public Instant getModified() {
24         return modified;
25     }
26
27     public void updateModified() {
28         this.modified = Instant.now();
29     }
30
31     @PrePersist
32     protected void prePersist() {
33         created = Instant.now();
34         modified = Instant.now();
35     }
36
37     @PreUpdate
38     protected void preUpdate() {
39         modified = Instant.now();
40     }
41 }
```

Рисунок 27 – Клас SimpleBaseEntity

4.3 Опис функцій програми

4.3.1 Опис всіх функцій

Додаток передбачає наявність наступних рівнів допуску, для яких будуть доступні функціональні можливості відповідно до діаграми використання основних функцій на рис. 28. Різниця між користувачем системи та звичайним користувачем є в тому, що перший у випадку інтеграції з іншою системою, а другій користувач уживає браузера. З погляду Keycloak різниці в методі доступу не має.

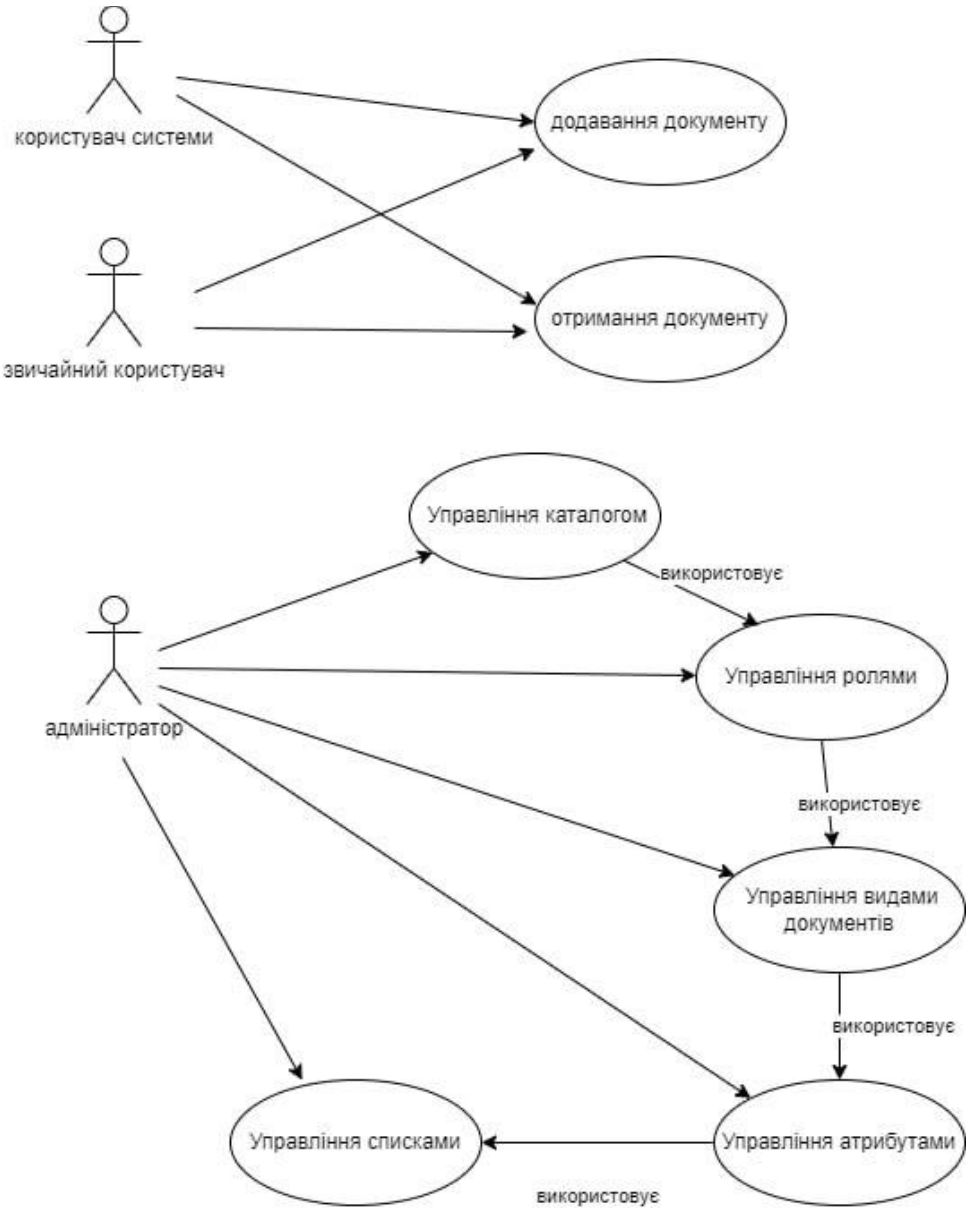


Рисунок 28 – Діаграма доступних функцій відповідно до прав ролей.

Список функцій:

1. Додавання та попередній перегляд документів — функціональні можливості, доступні звичайних користувачам (доступ через графічний інтерфейс програми) і системним користувачам.

Під час додавання документів через інтерфейс користувача вхід і авторизація відбуваються до того, як відобразиться форма додавання документів. У цьому випадку програма надає лише каталоги та типи файлів, до яких користувач має дозволи, які залежать від призначеної ролі. Користувач вибирає один із доступних каталогів і тип файлу. Потім відображаються елементи форми, пов'язані з динамічними атрибутами, які приписані до вибраного типу файлу. Вміст полів форми перевіряється відповідно до визначень окремих атрибутів. Правильно заповнена форма активує кнопку «зберегти».

У разі запису через API також виникає необхідність ідентифікації користувача, що здійснюється шляхом самостійного входу в систему KeyCloak і передачі т.зв. токен носія відповідно до стандарту OAuth. Інформація про обраний каталог і тип файлу передається одночасно з даними авторизації, тому необхідно перевірити узгодженість всього вмісту запиту, включаючи правильність заданих значень атрибутів.

На відміну від графічного інтерфейсу, де ці перевірки виконуються у інтерфейсі програми, тут ці операції виконуються на сервері. На рис. 29 продемонстровано алгоритм додавання файлу (разом з метаданими звиклими, та динамічними, якщо вони є) включно з перевірками правильності заданих атрибутів, і детально буде описаний у пункті 4.3.2 Додавання файлу.

2. Додавання та редагування ролей — адміністративна функція, яка полягає у створенні та зміні ролей, які відповідають ролям, створеним і призначеним користувачам у Keycloak. На екрані редагування ролі ми визначаємо ім'я, що діє в програмі, і технічне ім'я, що може бути надане відповідно до документація, а потім пов'язуємо його з роллю в Keycloak. Крім того, на екрані ми вибираємо типи документів, до яких дана роль матиме доступ.

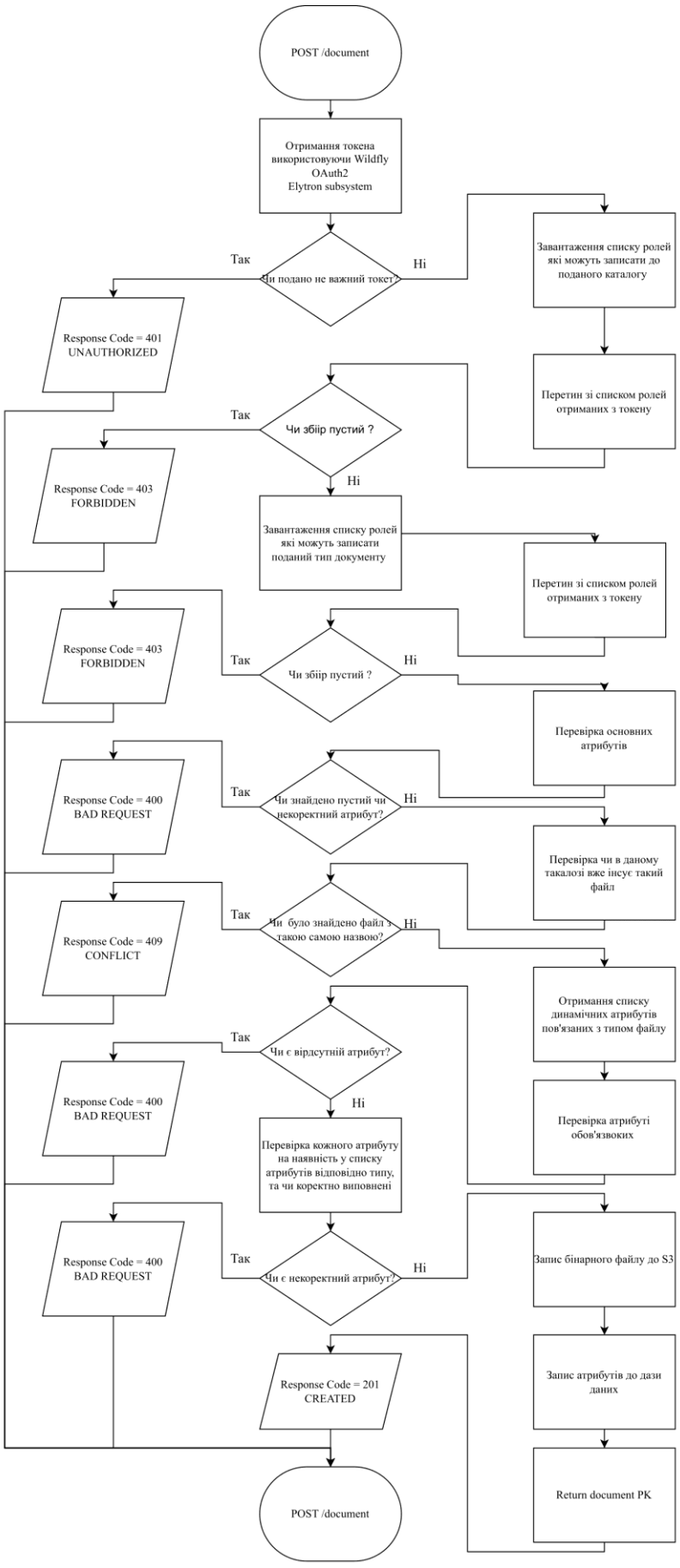


Рисунок 29 – Алгоритм додавання нового файлу та метаданих до нього

3. Додавання та редагування словників. Каталоги виконують у системі дві функції:

А) Систематизація — ви можете розділити збережені документи тематично по різним каталогам;

Б) Безпека — управління доступом через вказані ролі, які мають незалежні права на запис і читання файлів у каталозі.

Створення нового каталогу пов'язане із зазначенням ролей, які мають права запису або лише читання в даному каталозі. Варто звернути увагу на те, що функція запису завжди використовується разом з правами на читання.

4. Керування типами документів включає функцію їх додавання та модифікації разом із зазначенням списку атрибутів, які використовуються для створення набору метаданих, що описують певний тип документа.

5. Керування доступними атрибутами, які використовуються для створення метаданих документа. Залежно від вибраного типу атрибута список ознак, що описують атрибут, може змінюватися:

А) Date і Boolean (дата та логічні типи) — лише основна інформація: ім'я та обов'язковість;

Б) String (тобто значення текстове) — довжина рядка, правило перевірки або регулярний вираз, який потрібно перевірити;

В) Number (цифри) — мінімальне та максимальне значення, точність (кількість знаків після коми);

Г) List (словник) — вимагає вказівки словника з допустимими значеннями.

6. Управління словниками — створення та редагування словників, які використовуються для створення атрибутів типу List та редагування їх вмісту.

Разом з можливістю приховування використаних атрибутів (старі, непотрібні при додаванню нових документів, значення, які неможливо видалити через їх використання в раніше створених документах).

4.3.2 Додавання файлу

Інформація про користувача, який увійшов у систему (відповідно до заданого токена), отримується за допомогою класу SecurityDomain, який надається як частина реалізації протоколу OAuth2 на сервері додатків Wildfly — рис. 30 рядок 48. На основі цієї інформації ми створюємо набір ролей, які має увійшовши в систему користувач.

```
40      public Set<String> getRoles() {
41          var roles = new HashSet<String>();
42          var identity = getIdentity();
43          identity.getRoles().forEach(r -> roles.add(r));
44          return roles;
45      }
46
47      private SecurityIdentity getIdentity() {
48          return SecurityDomain.getCurrent().getCurrentSecurityIdentity();
49      }
```

Рисунок 30 – Читання ролі користувача

На наступному кроці для заданого каталогу з бази даних завантажується список ролей, які мають дозвіл на запис у вказаний каталог. Отриманий набір ролей перетинається з набором ролей, якими володіє користувач. Якщо результатом є порожній набір, програма повертає Response Code = 403 FORBIDDEN.

Таким самим чином відбувається з заданим типом файлу. Завантажуються з бази даних усі ролі, які мають право використовувати даний тип файлу, а потім перетинається з набором ролей, якими володіє користувач. Порожній набір результатів змушує програму повертати Response Code = 403 FORBIDDEN.

Для перетину множин ми використовується метод retainAll(Collection<?> c) у класі java.util.Set.

На наступному кроці перевіряється, чи всі вбудовані атрибути документа надано у вмісті запиту та чи є наданий вміст файлу дійсним кодом Base64. Невиконання вищевказаних умов завершує обробку запиту, повертаючи у відповідь код помилки 400 — Bad Request.

На наступному кроці ми перевіряється, чи вже є файл із вказаною назвою в системі VMP у даному каталозі. Якщо файл уже існує, програма повертає код помилки 409 — CONFLICT.

Останнім кроком є перевірка вмісту розділу динамічних атрибутів. Для цього завантажується з бази даних список визначень атрибутів, пов’язаних із певним типом документа.

На першому кроці перевіряється, чи містяться у поданих атрибутах, наданих в запиті, всі необхідні атрибути. З цією метою для кожного визначення атрибута, де горить прапор REQUIRED, ми перевіряємо наявність атрибута в запиті. Відсутність будь-якого обов'язкового атрибута завершує обробку з кодом помилки 400 — Bad Request.

Потім ітеруються всі надані атрибути в запиті та зіставляються їх визначення атрибутів зі списку атрибутів, доступних для даного типу файлу. Відсутність визначення означає, що в запиті було вказано недійсний атрибут, і програма негайно повертає код помилки 400 — Bad Request. Якщо було можливо зіставити визначення атрибута з даним значенням атрибута, ми перевіряємо правильність заданого значення відповідно до перевірок, встановлених для даного визначення, наприклад, довжина текстового поля, максимальне числове значення, відповідність тексту до подано регулярний вираз. Невиконання будь-якої з умов призводить до повернення коду помилки 400 — Bad Request.

Після виконання наведених вище перевірок програма записує двійковий файл у сховище файлів Min.IO за допомогою клієнтської бібліотеки S3, наданої Amazon.

На останньому крокі створюються об’єкти документа та пов’язані об’єкти, що представляють динамічні атрибути:

- AttributeNumberValue:
- AttributeDateValue:
- AttrlbuteListValue:
- AttributeBooleanValue:

- AttrlibuteTextValue.

Вони зберігаються в базі даних PostgreSQL за допомогою EntityManager.

Ідентифікатор присвоюється новому документу за допомогою секвенції бази даних, що забезпечує його унікальність. Цей ідентифікатор повертається в заголовку відповіді з кодом 201 — Created.

```
        return Response.created(uriInfo.getRequestUriBuilder().path("{id}")
                .build(document.getId())).build();
    }
```

Рисунок 31 — Створення відповіді http зі статусом 201

4.3.4 Пошук по метаданих

У системі VMP документи описуються за допомогою метаданих/ атрибутів, кількість і тип яких змінюються з часом. Це атрибути, які динамічно визначаються протягом життя системи, і їх доступність і необхідність заповнення, під час додавання нового файлу, додатково залежить від типу. Це означає, що створення пошукової системи, яка враховує динамічний список атрибутів, невідомих під час проектування та побудови програми, є досить складним завданням.

По-перше, в інтерфейсі користувача має бути створена панель, у якій користувач буде вибирати атрибути, за якими він бажатиме шукати документи. Для кожного з цих атрибутів з’являється відповідний елемент форми, до якого можна ввести значення даного атрибута. Таким чином, відповідно для:

• Числового чи символьного тип - це буде INPUT TEXT

• Тип дати – селектор дати

• Тип списку – однозначний селектор елемента зі списку

Для демонстрації створено три атрибути: дата, логічний та текстовий, назва яких повторює їх тип. Вибрано почергово з панелі пошуку ці три метадані та надано значення. Результат доданих критеріїв пошуку на рис. 32

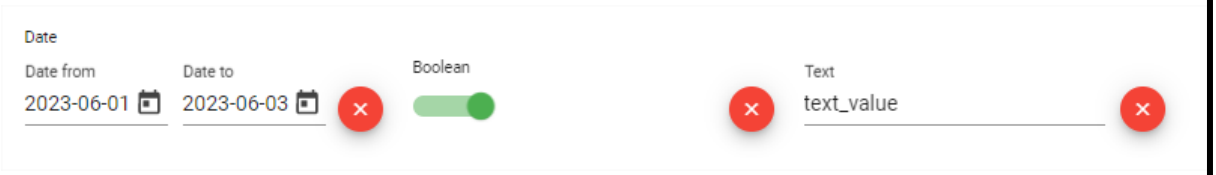


Рисунок 32 – Приклад пошуку за метаданими трьох типів

Після натискання кнопки «Пошук» окрім критеріїв на основі вбудованих атрибутів: ID, ім’я, тип файлу, каталог, також будуть надіслані динамічні критерії на основі наведеної вище динамічної форми.

Приклад запиту, надісланого з інтерфейсу користувача програми:

http://localhost:4200/api/documents/search?Date=2023-06-01%3A2023-06-03&Boolean=true&Text=text_value&limit=10&offset=0&orderBy=%2Bid

Наприклад, параметр дати, тобто діапазон часу, надсилається у форматі "назва_атрибута=РРРР-ММ-ДД:РРРР-ММ-ДД". На стороні серверної частини цей формат має бути декодований у пару об’єктів java.time.ZonedDateTime за допомогою регулярного виразу:

```
java.util.regex.Pattern PATTERN = Pattern.compile("(\\d{4}-\\d{2}-\\d{2})?:(\\d{4}-\\d{2}-\\d{2})?");
```

У зв'язку з тим, що список імен динамічних атрибутів невідомий під час побудови системи, тому критерії повинні бути приведені до форми карти (ключ -> значення). Відповідний код REST Endpont було підготовлено таким чином, щоб передати параметр MultivaluedMap<String, String> частині логіки відповідальної за пошук, приклад на рис. 33.

Наразі система допускає лише однозначні атрибути, але для майбутньої можливості розширення багатозначними атрибутами, була вибрана класа MultivaluedMap, замість класичної карти з унікальними ключами java.util.Map.

У доступному в модулі ejb класі DocumentRepository, який реалізує логіку, пов’язану зі зберіганням метаданих документів, підготовлено функцію, яка за допомогою JPA Criteria Builder динамічно створюватиме відповідний запит до бази даних. Ключовим елементом логіки є функція, яка додає умову компонента до побудованого SQL-запиту на основі одного елемента фільтра, вона зображена на рис. 34.

```java
    @GET
    @Path("/search")
    @Produces(MediaType.APPLICATION_JSON)
    public Response searchDocument(@Context UriInfo uriInfo) {
        MultivaluedMap<String, String> queryParameters = uriInfo.getQueryParameters();
        Map<String, String> filters = new HashMap<String, String>();
        String orderBy = "";
        int limit = -1;
        int offset = 0;
        for (String key : queryParameters.keySet()) {
            switch (key) {
            case "orderBy":
                orderBy = queryParameters.getFirst(key);
                break;
            case "limit":
                limit = Integer.parseInt(queryParameters.getFirst(key));
                break;
            case "offset":
                offset = Integer.parseInt(queryParameters.getFirst(key));
                break;
            default:
                Optional.ofNullable(queryParameters.getFirst(key))
                        .ifPresent(v -> filters.put(key, v.trim().toLowerCase()));
            }
        }
        ResultsPage<DocumentSummaryDTO> results = repository.
                searchDocuments(filters, orderBy, limit, offset, false);
        return Response.ok(results).build();
    }
```

Рисунок 33 – Функція пошуку

```java
    private void addAttributeFilter(Map.Entry<String, String> filter, CriteriaBuilder cb, AbstractQuery<?> cq,
            Root<Document> cqr, List<Predicate> predicates, DocumentAttributeDefinitionDTO attrDef) {
        switch (filter.getKey()) {
        case "document_type":
            prefixIgnoreCase(cb, cqr, cqr.get(Document_.documentType).get(DocumentType_.displayName),
                    filter.getValue(), predicates);
            break;
        default:
            Object filterValue = attrDef.getDataType() == DataType.D || attrDef.getDataType() == DataType.B ? null
                    : DocumentDetailsService.getValueObject(attrDef.getDataType(), filter.getValue());

            switch (attrDef.getDataType()) {
            case B:
                Root<AttributeNumberValue> bRoot = cq.from(AttributeNumberValue.class);
                predicates.add(cb.equal(cqr.get(Document_.id), bRoot.get(AttributeNumberValue_.documentId)));
                predicates.add(cb.equal(bRoot.get(AttributeNumberValue_.attributeDefinitionId), attrDef.getId()));
                predicates.add(
                        cb.equal(bRoot.get(AttributeNumberValue_.value), filter.getValue().equals("true") ? 1 : 0));
                break;
            case D:
                Root<AttributeDateValue> dRoot = cq.from(AttributeDateValue.class);
                predicates.add(cb.equal(cqr.get(Document_.id), dRoot.get(AttributeDateValue_.documentId)));
                predicates.add(cb.equal(dRoot.get(AttributeDateValue_.attributeDefinitionId), attrDef.getId()));
                dateRange(cb, cqr, dRoot.get(AttributeDateValue_.value), filter.getValue(), predicates);
                break;
            case L:
                Root<AttributeListValue> lRoot = cq.from(AttributeListValue.class);
                predicates.add(cb.equal(cqr.get(Document_.id), lRoot.get(AttributeListValue_.documentId)));
                predicates.add(cb.equal(lRoot.get(AttributeListValue_.attributeDefinitionId), attrDef.getId()));
                predicates.add(cb.equal(lRoot.get(AttributeListValue_.livaId), filterValue));
                break;
            case N:
                Root<AttributeNumberValue> nRoot = cq.from(AttributeNumberValue.class);
                predicates.add(cb.equal(cqr.get(Document_.id), nRoot.get(AttributeNumberValue_.documentId)));
                predicates.add(cb.equal(nRoot.get(AttributeNumberValue_.attributeDefinitionId), attrDef.getId()));
                predicates.add(cb.equal(nRoot.get(AttributeNumberValue_.value), filterValue));
                break;
            case T:
                Root<AttributeTextValue> tRoot = cq.from(AttributeTextValue.class);
                predicates.add(cb.equal(cqr.get(Document_.id), tRoot.get(AttributeTextValue_.documentId)));
                predicates.add(cb.equal(tRoot.get(AttributeTextValue_.attributeDefinitionId), attrDef.getId()));
                prefixIgnoreCase(cb, cqr, tRoot.get(AttributeTextValue_.value), filterValue.toString(), predicates);
            default:
                break;
```

Рисунок 34 – Функція додавання критеріїв вибору для динамічного фільтрів

У цій функції на першому кроці відокремлюються фільтри на основі вбудованих атрибутів, у рядку 161 створюється критерій типу документа.

У подальшій частині, залежно від типу атрибута, створюються відповідні компоненти мови JPQL (Java Persistence Query Language). Для кожного типу атрибута необхідно визначити критерій, що з'єднує основну таблицю документа з таблицею, що зберігає даний атрибут (рядки 172, 179, 185, 191 і 197 на рис. 30).

Для згаданого вище атрибута типу «Дата» була надана додаткова допоміжна функція, рис. 35 , яка додає відповідні предикати залежно від того, які компоненти діапазону були надані.

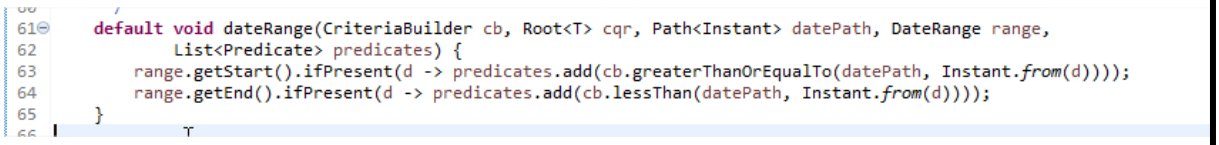


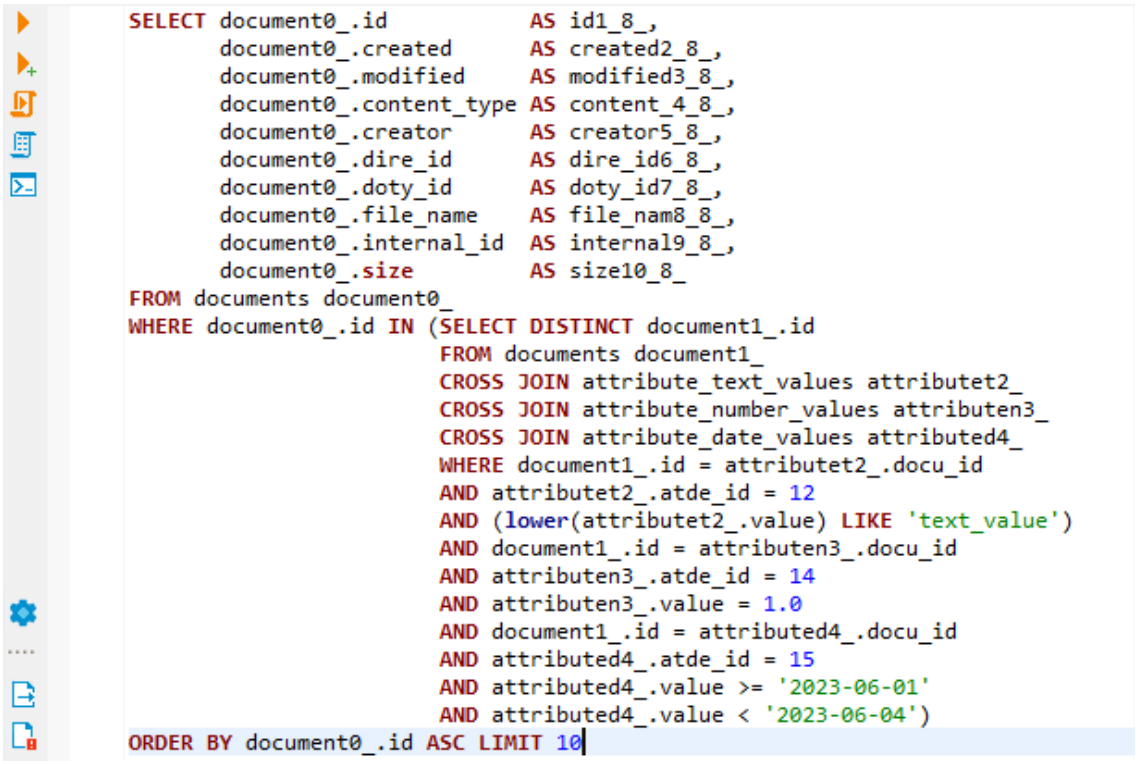
Рисунок 35 – Функція для роботи з датами різного діапазону

Останнє, що потрібно зробити, це перетворити значення фільтра на тип об'єкта, який відповідає типу стовпця в базі даних:

- Логічне значення -> filter.getValue().equals("true") ? 1:0;
- Дата -> перетворення на тип DateRange за допомогою згаданого шаблону регулярного виразу;
- Список -> перетворення в тип Long - Long.parseLong(значення);
- Число -> перетворення в тип BigDecimal - new BigDecimal(значення);
- Текст -> без перетворення;

Запит JPQL, виконаний таким чином, генерує SQL-запит до бази даних. Для значень на рис.28 приклад запитання зображений на рис. 36.

Особливо варто відзначити умову AND attrbuted4_.value < '2023-06-04'. У фільтрі було вибрано верхній діапазон дати «2023-06-03», що вказує на те, що нас цікавлять усі документи, створені до цієї дати. Тому дату верхнього діапазону було навмисно зміщено, щоб також включити документи, створені о 23:59:59. Для цього була створена допоміжна функція nextMidnight, що зображена на рис. 37.



```
SELECT document0_.id           AS id1_8_,
       document0_.created      AS created2_8_,
       document0_.modified     AS modified3_8_,
       document0_.content_type AS content_4_8_,
       document0_.creator      AS creator5_8_,
       document0_.dire_id      AS dire_id6_8_,
       document0_.doty_id      AS doty_id7_8_,
       document0_.file_name    AS file_nam8_8_,
       document0_.internal_id  AS internal9_8_,
       document0_.size         AS size10_8_
FROM documents document0_
WHERE document0_.id IN (SELECT DISTINCT document1_.id
                        FROM documents document1_
                        CROSS JOIN attribute_text_values attributet2_
                        CROSS JOIN attribute_number_values attributen3_
                        CROSS JOIN attribute_date_values attributed4_
                        WHERE document1_.id = attributet2_.docu_id
                        AND attributet2_.atde_id = 12
                        AND (lower(attributet2_.value) LIKE 'text_value')
                        AND document1_.id = attributen3_.docu_id
                        AND attributen3_.atde_id = 14
                        AND attributen3_.value = 1.0
                        AND document1_.id = attributed4_.docu_id
                        AND attributed4_.atde_id = 15
                        AND attributed4_.value >= '2023-06-01'
                        AND attributed4_.value < '2023-06-04')
ORDER BY document0_.id ASC LIMIT 10
```

Рисунок 36 – генероване SQL-запитання до бази даних

```
119    public static ZonedDateTime nextMidnight(LocalDate ld) {
120        return ld.atStartOfDay(ZoneId.systemDefault()).plus(1, ChronoUnit.DAYS);
121    }
```

Рисунок 37 – Допоміжна функція для дати

ВИСНОВОК

Метою дипломного проєкту є створення системи управління документами, яка шляхом збереження бінарних файлів в глобальному репозиторії та використання метаданих для пошуку, дозволяє підвищити ефективність.

В роботі проведено аналіз існуючих систем для збереження цифрових активів та системи електронного документообігу, визначено їх переваги та недоліки, а також сформульовано основні вимоги до розроблення системи.

На основі аналізу можливих технологій створено систему управління документами з використанням технології Java EE, серверном додатків Jboss/Wildfly для серверного рівня та Angular SPA для зовнішнього рівня. Як сховище метаданих використовувалася база даних PostgreSQL, а як сховище двійкових файлів – Min.IO відповідно до протоколу Amazon S3.

Розроблена система має наступні функціональні можливості:

- можливість додавання файлу та метаданих нього;
- можливість додавати нові ролі, директорії, типи документів, словники;
- можливість редагування ролі, директорії, типи документів, словники;
- авторизація за допомогою логіну та паролю за допомогою Keycloak;
- можливість роботи у якості користувача та адміністратора;
- зберігання файлів у глобальному репозиторії;
- наявність зручного інтерфейсу користувача.

Розроблена система є доволі універсальною і може бути застосовна у багатьох сфера, що мають електронний документообіг, адже динамічну систему метаданих можна пристосувати для будь-якої галузі

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Сієрра К. Head First. Java : навчальний посібник / К. Сієрра, Б. Бейтс. – Вид. 2-ге, – Харків : Фабула, 2022. – 720 р. – ISBN 978-617-522-033-7.

2. Робсон Е. Head First Патерни проєктування. / Е. Робсон, Е. Фрімен. – Харків : Фабула, 2020. – 688 с. – ISBN 978-617-096-159-4.

3. Шилдт Г. Java: руководство для начинающих. / Г. Шилдт. – Вид. 9-те, – Київ : Наук. світ, 2023. – 752 с. – ISBN 978-617-550-100-9.

4. Дашнер С. Вивчаємо Java EE. Сучасне програмування для великих підприємств / С. Дашнер. – Санкт-Петербург : Питер Прес, 2018. – 384 с. – ISBN 978-5-4461-0774-2.

5. Фримен А. Angular для профессионалов. / А. Фримен. – Москва : Питер, 2018. – 800 с. – ISBN 978-544-610-451-2.

6. Марухленко А. Разработка защищённых интерфейсов Web-приложений : навч. посіб. / А. Марухленко, Л. Марухленко, М. Ефремов. – Берлін : DirectMEDIA, 2021. – 174 с. – ISBN 978-544-991-676-1.

7. Бауэр К. Java Persistence API и Hibernate / К. Бауэр, Г. Кинг, Г. Грегори. – Москва : ДМК Пресс, 2017. – 632 с. – ISBN 978-597-060-180-8.

8. Maven Documentation [Електронний ресурс] // Maven Apache Maven. – Режим доступу: https://maven.apache.org/guides/index.html

9. Java(TM) EE 8 Specification APIs [Електронний ресурс] // Java EE. – Режим доступу: https://javaee.github.io/javaee-spec/javadocs/