НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Створення інформаційних сервісів для підтримки навчального процесу на платформі Docker

Виконав: студент 4 курсу, групи ДА-32
(шифр групи)

Приходько Владислав Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н Гіоргізова-Гай В.Ш. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний доцент, к.е.н. Рощина Н. В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Приходько Владислав Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Створення інформаційних сервісів для підтримки навчального процесу на платформі Docker ,
керівник роботи Гіоргізова-Гай Вікторія Шалвівна, к.т.н ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету: від « 10 » травня 2017 р. №1477-с
2. Термін подання студентом роботи: 12.06.2017

3. Вихідні дані до роботи: Платформа Docker. Відкрите програмне забезпечення. Підтримка розмежування прав доступу користувачів і протоколу LDAP. Підтримка ієрархії папок у файловому сховищі. Підтримка розповсюджених браузерів.

4. Зміст роботи:
Види віртуалізації серверів та принципи організації платформи Docker.
Обґрунтування вибору програмного забезпечення для створення інформаційних сервісів.
Реалізація інформаційних сервісів в мережі кафедри СП.
Тестування роботи інформаційних сервісів.
Функціонально-вартісний аналіз системи.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):

1. Платформа Docker і види віртуалізації серверів – плакат_____
2. Вибір ПЗ для організації інформаційних сервісів – плакат _____
3. Архітектура системи інформаційних сервісів – плакат_____
4. Приклади роботи інформаційних сервісів – плакат_____

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
економічний	Рощина Н. В., доцент, к.е.н.		

7. Дата видачі завдання 20.01.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	20.01.17	
2	Збір інформації	28.02.17	
3	Вивчення принципів побудови інформаційних сервісів	15.03.17	
4	Проведення порівняльної характеристики компонентів інформаційного сервісу	10.04.17	
5	Розробка технічного завдання	20.04.17	
6	Встановлення програмних модулів на сервер	30.04.17	
7	Тестування інформаційних сервісів	10.05.17	
8	Оформлення дипломної роботи	1.06.17	
9	Отримання допуску до захисту та подача роботи в ДЕК	12.06.17	

Студент _____ В.С. Приходько
(підпис) (ініціали, прізвище)

Керівник роботи _____ В.Ш. Гіоргізова-Гай
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Приходька Владислав Сергійовича на тему: «Створення інформаційних сервісів для підтримки навчального процесу на платформі Docker»

Метою дипломної робити є створення і налаштування системи інформаційних сервісів – хмарного сховища і online офісу на платформі Docker для потреб навчального процесу на кафедрі СП.

В роботі розглянуто види віртуалізації ресурсів в інформаційних технологіях та особливості організації контейнерів на платформі Docker. На основі критеріїв, що висувалися до застосування сервісів у навчальному процесі, проведено порівняльну характеристику відкритих програмних рішень для компонентів системи – хмарних сховищ і офісів. В ході виконання роботи було обґрунтовано вибір програмних рішень для компонентів інформаційних сервісів, розроблено загальну архітектуру системи, розглянуто варіанти доступу до системи користувача, що знаходиться поза межами кафедри СП, проведено налаштування системи, тестування її роботи і підготовку до встановлення на фізичний сервер кафедри.

Загальний обсяг роботи: 98 сторінки, додаток на 6 сторінок, 44 рисунків, 8 таблиць та 19 посилань.

Ключові слова: платформа Docker, інформаційні сервіси, хмарне сховище, хмарний офіс, контейнер, архітектура системи..

АННОТАЦИЯ

бакалаврской дипломной работы Приходько Владислава Сергеевича на тему: «Создание информационных сервисов для поддержки учебного процесса на платформе Docker»

Целью дипломной работы является создание и настройка системы информационных сервисов - облачного хранилища и online офиса на платформе Docker для нужд учебного процесса на кафедре СП.

В работе рассмотрены виды виртуализации ресурсов в информационных технологиях и особенности организации контейнеров на платформе Docker. На основе критериев, которые выдвигались к применению сервисов в учебном процессе, проведена сравнительная характеристика открытых программных решений для компонентов системы - облачных хранилищ и офисов. В ходе выполнения работы было обосновано выбор программных решений для компонентов информационных сервисов, разработана общая архитектура системы, рассмотрены варианты доступа к системе пользователя, находящегося за пределами кафедры СП, проведено настройку системы, тестирование ее работы и подготовку к установке на физический сервер кафедры.

Общий объем работы: 98 страницы, приложение на 6 страниц, 44 рисунков, 8 таблиц и 19 ссылок.

Ключевые слова: платформа Docker, информационные сервисы, облачное хранилище, облачный офис, контейнер, архитектура системы.

ABSTRACT

to the bachelor thesis by Prykhodko Vladyslav Serhiyovych on: «Creating information services to support the learning process on the Docker platform»

The aim of the thesis is to create and configure system of information services, i.e. cloud storage and an online office on the Docker platform for the needs of the educational process at the CAD department.

The paper considers virtualization resources types in information technologies and organization features of containers on the Docker platform. Based on the criteria that were put forward to the application of services in the learning process, comparative characteristics of open software solutions for the components of the system - cloud storage and offices were carried out. In the course of the work, the choice of software solutions for the components of information services was justified, a general architecture of the system was developed, options for accessing the system of the user outside the CAD department were considered. In addition, we configured the system, tested and prepared it for installation on a physical server of the department.

The diploma includes 98 pages, 44 illustrations, 8 tables, 19 references and 1 appendix (6 pages).

Keywords: Docker platform, information services, cloud storage, cloud office, container, system architecture.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Хост – комп'ютерний сервер, що містить ресурс і надає доступ у форматі клієнт-сервер.

ОС – операційна система.

ІТ – інформаційні технології.

CLI – Command Line Interface.

API – Application Programming Interface.

LXC – LinuX Containers, система віртуалізації на рівні операційної системи.

LDAP - Lightweight Directory Access Protocol, полегшений протокол доступу до каталогів.

WebDAV – Web-based Distributed Authoring and Versioning, це набір розширень та доповнень до протоколу HTTP, які дозволяють користувачам спільно редагувати та керувати файлами на веб-серверах.

2FA – Two-factor authentication, двох етапна автентифікація.

CMS – Content Management System, система керуванням вмістом.

NFS – Network File System, протокол мережевого доступу до файлової системи.

SQL – Structured query language, декларативна мова програмування для взаємодії користувача з базами даних.

AMQP – Advanced Message Queuing Protocol, протокол передачі повідомлень між компонентами системи.

RESP – Redis Serialization Protocol, прокол для взаємодії користувача з кешем REDIS.

DN – Distinguished Name, унікальне ім’я об’єкту в каталозі LDAP.

CSR – Certificate Signing Request, це зашифрований запит на випуск сертифіката, який містить детальну інформацію про домен та організацію.

ВСТУП

Останнім часом у галузі інформаційно-комунікаційних технологій спостерігається бурхливий розвиток хмарних технологій. Відповідно до цього виникають численні хмарні сервіси, що все частіше застосовуються у різних сферах людської діяльності. Їх використовують у науці, освіті, бізнесі тощо. Одним із найпоширеніших подібних сервісів являються хмарні сховища даних.

Хмарне сховище даних – це модель он-лайн сховища, в якому дані зберігаються на численних розподілених у мережі серверах, що надаються в користування клієнтам, в основному, третьою стороною.

Таким чином, замість розміщення файлів на носіях зовнішньої пам’яті (або на вінчестерах комп’ютерів) інструменти і результати роботи поступово переносяться та розміщуються у хмарному сховищі даних або у “хмарі“.

Основною різницею між хмарним сховищем даних та звичайними носіями даних є: синхронізація даних між різними комп’ютерами, резервне копіювання файлів з комп’ютера у “хмару”, спільна робота певної групи осіб з окремими файлами та папками.

При зберіганні даних у “хмарі” через робочий комп’ютер можна отримати доступ до них з будь-якого іншого пристрою, наприклад, з планшета, смартфона або домашнього ноутбука. Всі зміни з файлами (редагування, копіювання, сортування або вилучення) будуть автоматично відображені на всіх пристроях. Спільний доступ можна надавати окремим файлам і папкам. Це дуже зручно, коли група осіб працює над спільним проектом. Наприклад, якщо один з членів команди завантажує файли в спільну папку, то інші учасники одразу ж отримують до нього доступ. При цьому не потрібно відправляти файл кожному з членів проекту персонально [1].

Але не всі данні можна зберігати на серверах третьої сторони і всі сервіси для хмарного зберігання даних є платними, а для безкоштовного використання відділяються сховища розмір яких не перевищує 20 ГБайт, тому вони підходять

для створення тільки особистих сховищ. Для вирішення цих проблем з’явився напрямок створення приватних сховищ. Розмір приватних сховищ обмежують тільки параметрами сервера, але тим самим накладають на власника відповідальність за збереження даних у сховища.

Останні роки прослідкується тренд переходу навчання з традиційних форм до мультимедійних і кількість електронного матеріалу щорічно збільшується з шаленою швидкістю, тому виникають проблеми централізовано зберігання і передачі учням матеріалу. При цьому виникає проблема у захисті інтелектуальної праці.

Метою даної роботи є створення інформаційних сервісів для підтримки навчального процесу, а саме обміну даними між студентами та викладачами.

Розроблена система включає приватне хмарне сховище і хмарний офіс, який інтегровано до сховища, для полегшення роботи з документами і можливостям робити невеликі правки.

Необхідно було створити гнучку систему, що дозволяла би легко адаптуватися під потреби навчального процесу, та потреби кафедри.

Таким чином, тема даного дипломного проекту направлена на вирішення конкретних задач, і тому є актуальною.

1 ВИДИ ВІРТУАЛІЗАЦІЇ СЕРВЕРІВ ТА ПРИНЦИПИ ОРГАНІЗАЦІЇ ПЛАТФОРМИ DOCKER

Згідно зі статистикою середній рівень завантаження процесорних потужності у серверів під управлінням Windows не перевищує 10%, у Unix-систем цей показник краще, але тим не менше в середньому не перевищує 20%. Низька ефективність використання серверів пояснюється широко застосовуваним з початку 90-х років підходом "один додаток - один сервер", тобто кожен раз для розгортання нової програми компанія купує новий сервер. Очевидно, що на практиці це означає швидке збільшення серверного парку і як наслідок - зростання витрат на його адміністрування, енергоспоживання та охолодження, а також потреба в додаткових приміщеннях для установки все нових серверів і придбання ліцензій на серверну ОС.

Віртуалізація ресурсів фізичного сервера дозволяє гнучко розподіляти їх між додатками, кожне з яких при цьому "бачить" тільки призначені йому ресурси і "вважає", що йому виділено окремий сервер, тобто в даному випадку реалізується підхід "один сервер - кілька додатків", але без зниження продуктивності, доступності та безпеки серверних додатків. Крім того, рішення віртуалізації дають можливість запускати в розділах різні ОС за допомогою емуляції їх системних викликів до апаратних ресурсів сервера.

В основі віртуалізації лежить можливість одного комп'ютера виконувати роботу декількох комп'ютерів завдяки розподілу його ресурсів за кількома середах. За допомогою віртуальних серверів і віртуальних настільних комп'ютерів можна розмістити кілька ОС і кілька додатків в єдиному місці розташування. Таким чином, фізичні і географічні обмеження перестають мати будь-яке значення. Крім енергозбереження та скорочення витрат завдяки більш ефективному використанню апаратних ресурсів, віртуальна інфраструктура забезпечує високий рівень доступності ресурсів, більш ефективну систему

управління, підвищену безпеку і вдосконалену систему відновлення в критичних ситуаціях.

У широкому розумінні поняття віртуалізації є приховування справжньої реалізації будь-якого процесу або об'єкта від істинного його уявлення для того, хто ним користується. Продуктом віртуалізації є щось зручне для використання, насправді, має більш складну або зовсім іншу структуру, відмінну від тієї, яка сприймається при роботі з об'єктом. Іншими словами, відбувається відділення уявлення від реалізації чого-небудь. Віртуалізація покликана абстрагувати програмне забезпечення від апаратної частини.

У комп'ютерних технологіях під терміном "віртуалізація" зазвичай розуміється абстракція обчислювальних ресурсів і надання користувачеві системи, яка "інкапсулює" (приховує в собі) власну реалізацію. Простіше кажучи, користувач працює з зручним для себе поданням об'єкта, і для нього не має значення, як об'єкт влаштований в дійсності [2].

Наведемо основні переваги технологій віртуалізації:

1) Ефективне використання обчислювальних ресурсів
2) Скорочення витрат на інфраструктуру
3) Зниження витрат на програмне забезпечення
4) Підвищення гнучкості і швидкості реагування системи
5) Несумісні додатки можуть працювати на одному комп'ютері
6) Підвищення доступності додатків і забезпечення безперервності роботи підприємства
7) Можливості легкої архівації
8) Підвищення керованості інфраструктури

1.1 Види віртуалізації серверів та інші види віртуалізації в інформаційних технологіях

Сьогодні, говорячи про технології віртуалізації, як правило, мають на увазі віртуалізацію серверів, так як остання стає найбільш популярним рішенням на

ринку IT. Віртуалізація серверів має на увазі запуск на одному фізичному сервері декількох віртуальних серверів. Віртуальні машини або сервера є програми, запущені на хостовій операційній системі, які емулює фізичні пристрої сервера. На кожній віртуальній машині може бути встановлена операційна система, на яку можуть бути встановлені додатки і служби. Типові представники це продукти VmWare (ESX, Server, Workstation) і Microsoft (Hyper-V, Virtual Server, Virtual PC).

Використання x86-віртуалізації почалося в кінці 90-х з робочих станцій: одночасно зі збільшенням числа версій клієнтських ОС постійно зростала і кількість людей (розробників ПЗ, фахівців з технічної підтримки, експертів), яким потрібно було на одному ПК мати відразу кілька копій різних ОС.

Віртуалізація для серверної інфраструктури стала застосовуватися трохи пізніше, і пов'язано це було, перш за все, з вирішенням завдань консолідації обчислювальних ресурсів. Але тут відразу сформувалося два незалежних напрямки:

- підтримка неоднорідних операційних середовищ (в тому числі, для роботи успадкованих додатків). Цей випадок найбільш часто зустрічається в рамках корпоративних інформаційних систем. Технічно проблема вирішується шляхом одночасної роботи на одному комп'ютері декількох віртуальних машин, кожна з яких включає екземпляр операційної системи. Але реалізація цього режиму виконувалася за допомогою двох принципово різних підходів: повної віртуалізації і паравіртуалізації;
- підтримка однорідних обчислювальних середовищ на увазі ізоляцію служб в рамках одного примірника ядра операційної системи (віртуалізація на рівні ОС), що найбільш характерно для хостингу додатків провайдерами послуг. Звичайно, тут можна використовувати і варіант віртуальних машин, але набагато ефективніше створити ізольовані контейнерів на базі одного ядра однієї ОС.

Початок масового застосування технологій x86-віртуалізації стартував в 2004-2006 рр. і був пов'язаний з початком їх масового застосування в корпоративних системах. Відповідно, якщо раніше розробники в основному займалися створенням технологій виконання віртуальних середовищ, то тепер на перший план стали виходити завдання управління цими рішеннями і їх інтеграції в загальну корпоративну IT-інфраструктуру.

Безліч побутових проблем і проблеми розробки технологій віртуалізації пов'язані з подоланням успадкованих особливостей програмно-апаратної архітектури x86. Найчастіше в комп'ютерних технологіях виділяють такі види віртуалізації:

1) Віртуалізація платформ (або серверів).
   - Повна віртуалізація (рис. 1). Використовуються не модифіковані екземпляри гостьових операційних систем, а для підтримки роботи цих ОС служить загальний шар емуляції їх виконання поверх хостовій ОС, в ролі якої виступає звичайна операційна система. Така технологія застосовується, зокрема, в VMware Workstation, VMware Server (колишній GSX Server), Parallels Desktop, Parallels Server, MS Virtual PC, MS Virtual Server, Virtual Iron. До переваг даного підходу можна зарахувати відносну простоту реалізації, універсальність і надійність рішення; всі функції управління бере на себе хост-ОС. Недоліки – високі додаткові накладні витрати на апаратні ресурси, відсутність врахування особливостей гостьових ОС, менша, ніж потрібно, гнучкість у використанні апаратних засобів.

Рисунок 1 – Повна віртуалізація

- Паравіртуалізація (рис. 2). Модифікація ядра гостьової ОС виконується таким чином, що в неї включається новий набір API, через який вона може безпосередньо працювати з апаратурою, що не конфліктує з іншими віртуальними машинами. При цьому немає необхідності використовувати повноцінну ОС в якості хостового ПО, функції якого в даному випадку виконує спеціальна система, що отримала назву гіпервізор. Саме цей варіант є сьогодні найбільш актуальним напрямком розвитку серверних технологій віртуалізації і застосовується в VMware ESX Server, Xen (і рішеннях інших постачальників на базі цієї технології), Microsoft Hyper-V. Переваги даної технології полягають у відсутності потреби в хостовій ОС - ВМ, встановлюються фактично на "голе залізо", а апаратні ресурси використовуються ефективно. Недоліки - в складності реалізації підходу і необхідність створення спеціалізованої ОС-гіпервізора.

Рисунок 2 – Паравіртуалізація [2]

- Віртуалізація на рівні ядра ОС (рис. 3). Цей варіант передбачає використання одного ядра хостовій ОС для створення незалежних паралельно працюють операційних середовищ. Для гостьового ПО створюється тільки власне мережеве та апаратне оточення. Такий варіант використовується в Virtuozzo (для Linux і Windows), OpenVZ (безкоштовний варіант Virtuozzo) і Solaris Containers. Переваги – висока ефективність використання апаратних ресурсів, низькі накладні технічні витрати, відмінна керованість, мінімізація

витрат на придбання ліцензій. Недоліки - реалізація тільки однорідних обчислювальних середовищ.

Рисунок 3 – Віртуалізація на рівні ОС [2]

- Віртуалізація з апаратною підтримкою. Даний варіант передбачає, що апаратне обладнання забезпечує архітектурну підтримку, яка полегшує створення монітору віртуальних машин і дозволяє запускати гостьові ОС ізольовано. Гостьова система не залежить від архітектури хостовой платформи і реалізації платформи віртуалізації. Переваги – висока ефективність використання апаратних ресурсів, спрощення розробки платформ віртуалізації за рахунок надання апаратних інтерфейсів управління і підтримки віртуальних гостьових систем, незалежність гостьової системи від архітектури хостовій платформи і реалізації платформи віртуалізації. Недоліки – збільшення витрат на покупку серверів, потрібно враховувати на етапі вибору обладнання. Приклади платформ віртуалізації, з апаратною підтримкою: KVM, VMware Workstation, VMware Fusion, Hyper-V, Windows Virtual PC, Xen, Parallels Desktop для Mac, Oracle VM Server для SPARC, VirtualBox і Parallels Workstation.

2) Віртуалізація додатків (рис. 4). Віртуалізація додатків с має на увазі застосування моделі сильної ізоляції прикладних програм з керованим взаємодією з ОС, при якій віртуалізується кожен екземпляр додатків, все його основні компоненти: файли (включаючи системні), реєстр,

шрифти, INI-файли, COM-об'єкти, служби. Додаток виповнюється без процедури інсталяції в традиційному її розумінні і може запускатися прямо з зовнішніх носіїв (наприклад, з флеш-карт або з мережевих папок). З точки зору IT-відділу, такий підхід має очевидні переваги: прискорення розгортання настільних систем і можливість управління ними, зведення до мінімуму не тільки конфліктів між додатками, а й потреби в тестуванні додатків на сумісність. Дана технологія дозволяє використовувати на одному комп'ютері, а точніше в одній і тій же операційній системі кілька несумісних між собою додатків одночасно. Віртуалізація додатків дозволяє користувачам запускати один і той же заздалегідь конфігурований додаток або групу додатків з сервера. При цьому додатки будуть працювати незалежно один від одного, не вносячи жодних змін в операційну систему. Фактично саме такий варіант віртуалізації використовується в Sun Java Virtual Machine, Microsoft Application Virtualization (раніше називалося Softgrid), Thinstall (на початку 2008 р увійшла до складу VMware), Symantec / Altiris.

Рисунок 4 – Віртуалізація додатків [2]

3) Віртуалізація уявлень (рис. 5) Віртуалізація уявлень має на увазі емуляцію інтерфейсу користувача. Тобто користувач бачить додаток і працює з ним на своєму терміналі, хоча насправді додаток виконується на віддаленому сервері, а користувачеві передається лише картинка віддаленого додатки. Залежно від режиму роботи користувач може

бачити віддалений робочий стіл і запущене на ньому додаток, або тільки саме вікно програми.

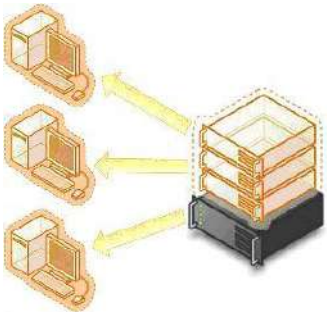

Рисунок 5 – Віртуалізація уявлень [2]

4) Віртуалізація даних – це представлення даних як абстрактного слою, незалежного від базових систем баз даних, структур і сховищ.
5) Мережева віртуалізація – це створення віртуальної мережі, адресуючи простір мережі в середині або через існуючі підмережі.
6) Віртуалізація пам'яті дозволяє об’єднати ресурси оперативної пам’яті з мережевих систем в єдиний адресний простір пам’яті.
7) Віртуалізація сховищ – це повне абстрагування логічного сховища даних від фізичного сховища [3].

1.2 Організація платформи Docker

Docker відкрита платформа для розробки, публікації, і запуску додатків. Docker дає можливість відокремити програми від інфраструктури щоб можливо було швидко зробити розгортання. З Docker, можливо керувати інфраструктурою так само просто як і додатками. Використовуючи методологію докер для вивантаження, встановлення і тестування можна істотно скоротити час між написанням коду і його здаванням в експлуатацію.

Докер забезпечує можливість упаковки і запуску додатка в ізольованому середовищі званим контейнером. Ізоляція і необхідний рівень безпеки дозволяють запускати безліч контейнерів на заданому хості. Через легкий

характер контейнерів, які працюють без додаткового навантаження гіпервізора, можна запустити більше контейнерів ніж на віртуальній машині [4].

Сама платформа називається Docker Engine, яка представляє клієнт-серверний додаток з трьома головними компонентами (рис. 6):

- Сервер працює у фоновому режимі (демон).
- REST API, який використовують програми для взаємодії з сервером.
- Інтерфейс командного рядка (CLI) клієнт.

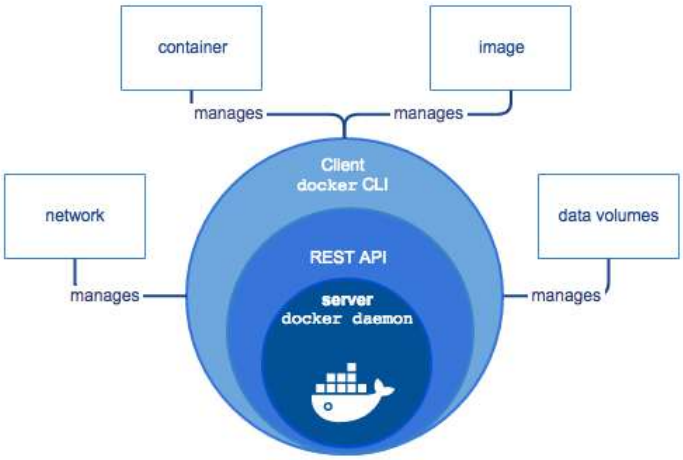

Рисунок 6 – Склад платформи Docker [5]

CLI використовує Docker REST API для керування або взаємодії з демоном Докер за допомогою сценаріїв або безпосередньо команд CLI. Багато інших додатків Docker засновані на використанні API і CLI

Демон створює і управляє об'єктами Docker, такими як образи, контейнери, мережі та сховища даних.

1.2.1 Архітектура платформи Docker Engin

Docker використовує архітектуру клієнт-сервер (рис. 7). Docker клієнт спілкується з демоном Docker, який бере на себе створення, запуск, розподіл контейнерів. Обидва, клієнт і сервер можуть працювати на одній системі, також можна підключити клієнт до віддаленого демона docker. Клієнт і сервер спілкуються через сокет або через RESTful API [5].

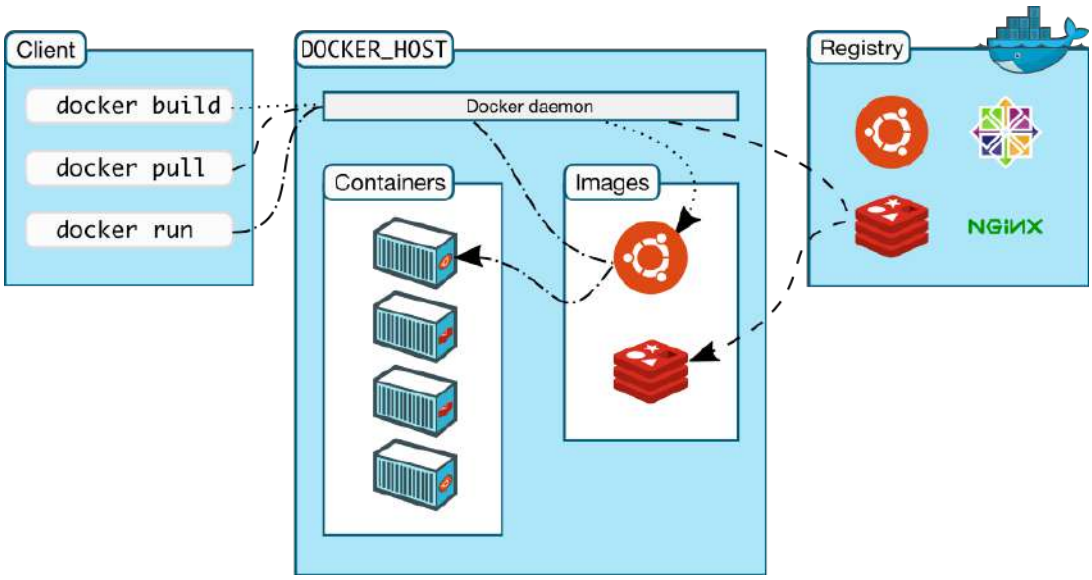

Рисунок 7 – Архітектура платформи Docker [5]

Як показано на рис. 7, демон запускається на хост-машині. Користувач не взаємодіє з сервером на пряму, а використовує для цього клієнт. Docker-клієнт - головний інтерфейс до Docker системи. Він отримує команди від користувача і взаємодіє з docker-демоном. Демон Docker працює на хост-машині. Користувач використовує клієнт Docker для взаємодії з демоном.

1.2.2 Головні компоненти платформи Docker Engine

Щоб розуміти, як працює docker, потрібно знати про три основні його компоненти:

- образи (images)
- реєстр (registries)
- контейнери

Docker-образ - це read-only шаблон з набором інструкцій для створення контейнера. Наприклад, образ може містити операційну систему Ubuntu з веб-сервером NGINX і додатком на ній. Образи використовуються для створення контейнерів. Docker дозволяє створювати нові образи, оновлювати існуючі, або завантажувати і використовувати образи, які були створені іншими користувачами.

Реєстр - це бібліотека образів. Він може бути публічним або приватним і може розташовуватися на одному сервері з демоном, клієнтом або на окремому сервері. Реєстри - це компонента поширення.

Контейнери схожі на директорії. У контейнерах міститься все, що потрібно для роботи програми. Кожен контейнер створюється з образу. Контейнери можуть бути створені, запущені, зупинені або видалені. Кожен контейнер ізольований і є безпечною платформою для додатка. Контейнери - це компонента роботи [6].

Docker сервіс надає режим swarm за допомогою якого можна обмежувати кількість примірників процесів образу. Можна вказати кількість паралельних завдань реплік для запуску, і менеджер swarm гарантує, що навантаження розподілиться рівномірно між робочими вузлами.

1.2.3 Переваги контейнеризації

Контейнери несуть в собі багато привабливих переваг як для розробників, так і для системних адміністраторів.

Деякі з найбільш привабливих переваг перераховані нижче.

1) Абстрагування хост-системи від контейнеризованих додатків. Контейнери були задуманими повністю стандартизованими. Це означає, що контейнер з'єднується з хостом або чим-небудь зовнішнім по відношенню до нього за допомогою певних стандартизованих інтерфейсів. Контейнеризований додаток не повинен покладатися або якимось чином залежати від ресурсів або архітектури хоста, на якому він працює. Це спрощує припущення про середовищі виконання програми в процесі розробки. Аналогічно, з точки зору хосту, кожен контейнер являє собою "чорний ящик".
2) Простота масштабування. Одним з переваг абстрагування між операційною системою хоста і контейнерами є те, що при правильному проектуванні програми, масштабування може бути простим і прямолінійним. Сервіс-орієнтована архітектура в комбінації з

контейнеризованими додатками забезпечує основу для легкого масштабування. Розробник може запустити кілька контейнерів на своїй робочій машині, при цьому та ж система може бути горизонтально масштабувати, наприклад, на тестовому майданчику. Коли контейнери запускаються в експлуатацію, вони знову можуть бути масштабовані.

3) Простота управління залежностями і версіями додатка. Контейнери дозволяють розробнику програми або компоненту додатку, з усіма його залежностями і далі працювати з ними як з єдиним цілим. Хосту не треба турбуватися про залежності, необхідних для запуску конкретного додатку. Якщо хост може запустити Docker, він може запустити будь-який Docker-контейнер. Це робить легким управління залежностями і також спрощує управління версіями програми. Хост-системи більше не повинні відповідати за управління залежностями додатки, тому що, за винятком випадків залежності одних контейнерів від інших контейнерів, все залежне повинно міститися в самому контейнері.
4) Надзвичайно легкі, ізольовані середовища виконання. Не дивлячись на те, що контейнери не надають такого ж рівня ізоляції та управління ресурсами, як технології віртуалізації, вони мають надзвичайно легке середовище виконання. Контейнери ізольовані на рівні процесів, працюючи при цьому поверх одного і того ж ядра хосту. Це означає, що контейнер не включає в себе повну операційну систему, що призводить до практично миттєвого його запуску. Розробник може легко запустити сотні контейнерів зі своєю робочою машини без будь-яких проблем.
5) Спільно використовувані шари. Контейнери легкі ще і в тому сенсі, що вони зберігаються "пошарово". Якщо кілька контейнерів засновані на одному і тому ж шарі, вони можуть спільно використовувати цей

базовий шар без дублювання, що призводить до мінімальному завантаженні дискового простору в наступних образах.

6) Можливість компонування та передбачуваність. Docker-файли дозволяють користувачам задати конкретні дії, необхідні для створення нового образу контейнера. Це дозволяє задавати налаштування середовища виконання так, як нібито це код, при бажанні зберігаючи ці настройки в системі контролю версій. Однакові Docker-файл, зібраний в одному і тому ж оточенні, завжди створить ідентичні образи контейнеру [7].

1.3 Висновки

Docker - чудовий інструмент, як для невеликих так і для масштабних проектів з великою кількістю одночасно працюючих завдань.

Docker базується на технологіях namespaces і cgroups (перша забезпучує ізоляцію, друга - угруповання процесів і обмеження ресурсів), тому в плані віртуалізації він мало чим відрізняється від звичних нам LXC / OpenVZ. Та ж швидкість роботи, ті ж методи ізоляції, засновані на механізмах ядра Linux, але він надає зручне API для взаємодії з ними, що дозволяє розгорнути повноцінне віртуальне оточення і запустити в ньому додаток так само просто, як, наприклад, перезапустити веб-сервер.

Docker відрізняється тим, що в контейнер загортається тільки додаток з його залежностями. А працює він поверх платформи встановленої всередині операційної системи. Тим самим звільняючи від необхідності включати гостьову ОС, економивши тим самим до 5 ГБайт розміру, і незалежність запуску контейнеру від середовища його розробки. Це робить Docker вкрай автоматизованим і універсальним.

Отже проаналізувавши платформу Docker можна виділити такі переваги перед іншими рішеннями віртуалізації рівня ОС:

1) Docker використовують універсальний формат образів. Це означає, що всі образи можна без будь-яких проблем перенести на іншу машину або передати для використання іншими користувачами.
2) Образ може служити базою для інших образів. В Docker вважається нормою використовувати безліч шарів для формування кінцевого образу. Для створення нового образу беруть базового образу Ubuntu, потім додають Apache, щоб створити мікросервіс Ubuntu + Apache.
3) При спільний розробці образ можна керувати версіями, так само як це робиться в GIT.
4) У Docker велике спільнота підтримки і велика екосистема, яка включає серйозну кількість інструментів масштабування, групування, моніторингу, розгортання та управління контейнерами.

2 АНАЛІЗ ВХІДНИХ ДАНИХ

Змістом дипломної роботи є організація інформаційних сервісів для забезпечення навчального процесу у мережевій інфраструктурі кафедри СП. Для цього необхідно встановити хмарне сховище даних і хмарний офіс та організувати доступ до цих сервісів.

Програмне забезпечення повинно мати можливість встановлення на ЕОМ під управлінням ОС Linux та працювати на платформі Docker.

Функціональні вимоги

Так як система складається з двох складових, то розглянемо функціональні вимоги для кожної складової окремо.

До хмарного сховища даних висуваються такі вимоги:

1) Хмарне сховище має відноситися до відкритого програмного забезпечення
2) Підтримка розмежування прав доступу
3) Підтримка ієрархії папок
4) Підтримка протоколу LDAP
5) Підтримка розповсюджених форматів зображення та відео
6) Хмарне сховище повинно бути підтримуваним розробниками і мати велику спільноту користувачів

До хмарного офісу висуваються такі вимоги:

1) Хмарний офіс має відноситися до відкритого програмного забезпечення
2) Можливість відкривати і редагувати, такі формати: doc, docx, odt, xls, xlsx, ods, ptt, pttx, odp
3) Створювати документи, таких форматів: docx, xlsx, pttx
4) Підтримка показу формату PDF
5) Максимальна відповідність функціоналу і інтерфейсу десктопних редакторів

6) Хмарний офіс повинен бути підтримуваним розробениками і мати велику спільноту користувачів.

Вимоги користувачів

1) Зручний і інтуїтивний інтерфейс користувача
2) Доступ до основного функціоналу за два кліки мишкою
3) Наявність документації для користувача
4) Наявність мобільного додатку або мобільної версії сайту
5) Наявність десктопного клієнта
6) Підтримка розповсюджених браузерів (Firefox, Chrome, Opera, Safari, IE, Microsoft Edge)

Системні вимоги

1) Сервер:
   - ОС: Linux (Debian 7, SUSE Linux Enterprise Server 11 SP3 & 12, Red Hat Enterprise Linux/CentOS 6.5 and 7 (7 is 64-bit only), Ubuntu 14.04 LTS, 16.04 LTS)
   - RAM: 6 GB
   - CPU: 2 GHz, 3 Core
   - HDD: 60GB
2) База даних: MySQL/MariaDB 5.5+ або PostgreSQL
3) Веб-сервер: Apache 2 (mod_php, php-fpm) або Nginx (php-fpm)
4) PHP 5.6+
5) Docker engine 1.8+
6) Веб браузер:
   - IE11+
   - Microsoft Edge
   - Firefox 14+
   - Chrome 18+
   - Safari 7+
   - Opera 21+

7) Мобільний додаток:
   - iOS 7+
   - Android 4+
8) Десктопний клієнт:
   - Windows XP SP3
   - Windows 7+
   - Mac OS X 10.7+ (64-bit only)
   - Linux (CentOS 6.5, 7 (7 is 64-bit only), Ubuntu 12.04 LTS, 14.04 LTS, 14.10, Fedora 20, 21, openSUSE 12.3, 13, Debian 7 & 8).

Вимоги до зовнішніх інтерфейсів

1) Відкрите з’єднання для протоколу LDAP
2) Відкрите з’єднання для протоколів HTTP, HTTPS
3) Відкрите з’єднання для протоколу WebDAV

2.1 Висновки

Під час написання даного розділу було проведено аналіз вимог до інформаційних сервісів для забезпечення навчального процесу. Під час аналізу було виділено такі групи вимог: функціональні, користувачів, системні та зовнішніх інтерфейсів. Аналіз був проведений для подальшого полегшення вибору складових і побудови інформаційної системи, яка відповідатиме вхідним вимогам.

3 ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СЕРВІСІВ

Виходячи з аналізу вхідних даних, які були розглянуті в попередньому розділи, потрібно проаналізувати ринок відкритого і вільного програмного забезпечення, і також безкоштовні версії комерційних продуктів.

3.1 Вибір хмарного сховища даних

У сучасному світі є багато рішень для зберігання даних, як і на власних серверах так і серверах компаній, які представляють це як послугу.

Але на ринку можна виділити, такі рішення:

3.1.1 Хмарне сховище Owncloud

Напевно, самий популярний проект, що дозволяє організувати власне сховище файлів для обміну даними між користувачами. Причому за можливостями він давно обігнав Dropbox, оскільки крім надсилання файлів користувач отримує ще календар, закладки, адресну книгу (з угрупованням за категоріями), список справ TODO і так далі. Реалізовано шифрування файлів, після активації даної можливості інформацію не може бути переглянута навіть адміністратор. Можливий контроль версій файлів (як back-end використовується Git, при нестачі простору старі редакції автоматично видаляються), установка квот і обмежень на максимальний розмір файлів. Кошик дозволяє відновлювати файли і каталоги, віддалені через веб-інтерфейс. Користувач може переглядати PDF- і ODF-файли, малюнки в фотогалереї, прослуховувати музику. Передбачено редагування текстових файлів за допомогою онлайн-редактора. Доступна синхронізація файлів, календаря та адресної книги з мобільним пристроєм або ПК і з іншими системами, що підтримують протокол remoteStorage. Система повнотекстового пошуку, заснована на движку Apache

Lucene, дозволяє шукати не тільки по іменах файлів, але і по їх вмісту. Інтерфейс сховища показано на рис. 8.

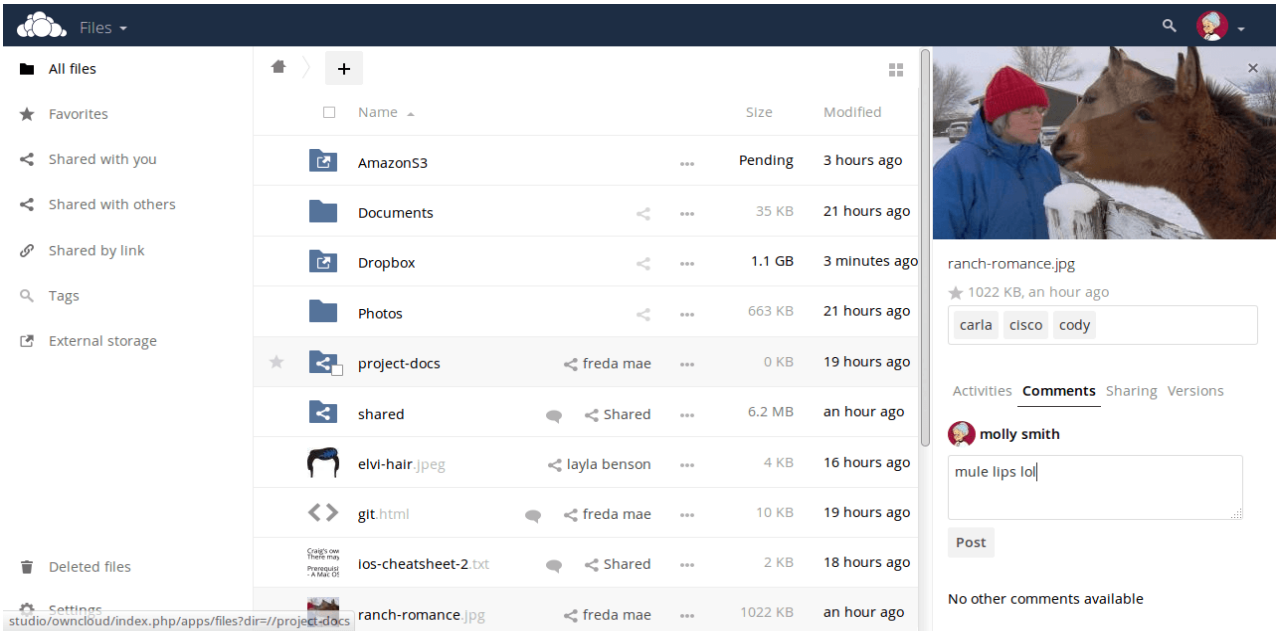

Рисунок 8 – Інтерфейс сховища Owncloud [8]

Базові можливості легко розширити за допомогою плагінів, частина з них надається самим проектом, доступні розробки третіх сторін. Велику колекцію плагінів можна знайти в репозиторії. Тут знаходимо модуль, перевіряючий збережені файли на наявність вірусів (за допомогою ClamAV), модуль для організації музичного сервера, що дозволяє прослуховувати музичну колекцію з будь-якого пристрою в мережі, сховище подкастів і відеороликів з доступом через веб-інтерфейс або медіаплеєр. Таким же чином додається підтримка OpenID і LDAP, а також робота з зовнішніми сховищами Dropbox, Swift, FTP, SFTP, Google Docs, S3 і WebDAV.

Доступ до даних надається як для зареєстрованих на сервері користувачів ownCloud (позначаються як загальні / Shared), так і без реєстрації для анонімного відвідувача (у вигляді прямого посилання). Реалізована можливість відправки повідомлень іншим користувачам через стандартний механізм нотифікації KDE (Open Collaboration Services API, спочатку проект розвивався під егідою KDE).

Для доступу використовується веб-браузер або WebDAV, KDE KIO-Slaves, за допомогою яких можна підключити сховище у вигляді мережевого диска. Інтерфейс системи локалізована і організований логічно і просто, тому з його освоєнням не повинно виникнути проблем у користувача з будь-яким рівнем підготовки. Розроблено клієнти ownCloud Desktop Client і Mobile Clients, що дозволяють синхронізувати дані з настільною системою під управлінням Windows, Linux і OS X або мобільним пристроєм Android (доступний в двох версіях - платної і безкоштовної) або iOS (iPhone / iPad / iPod). Крім цього, в мережі інтернет можна знайти велику кількість розширень і додатків App Store, що дозволяють зробити роботу з ownCloud ще більш зручною. Наприклад, для файлових менеджерів Dolphin, Nautilus, Finder і Explorer доступні модулі інтеграції з ownCloud.

Кілька серверів ownCloud можуть взаємодіяти між собою, забезпечуючи автоматичне резервне копіювання і міграцію даних користувача на інший сервер. Продукт швидко розвивається, новий реліз виходить регулярно кожні три місяці.

Написаний ownCloud на PHP і JavaScript, як СУБД можна використовувати SQLite, MySQL або PostgreSQL. Для розгортання підійде стандартний LAMP- або WAMP-сервер, а сам процес досить тривіальний [8].

На жаль, проект має довгу історію зломів - в різний час в коді ownCloud дослідники знаходили численні критичні уразливості (виконання довільного PHP-коду на сервері, отримання повного доступу до календарів інших користувачів та інше) [9].

3.1.2 Хмарне сховище Nextcloud

Nextcloud - альтернатива програмного забезпечення для хмарних обчислень, яка дає вам повний контроль над вашими даними. Він призначений для людей і організацій з багатьма користувачами. Це відносно молодий проект, який відокремився від аналогічного проекту ownCloud. Інтерфейс користувача хмари показано на рис. 9.

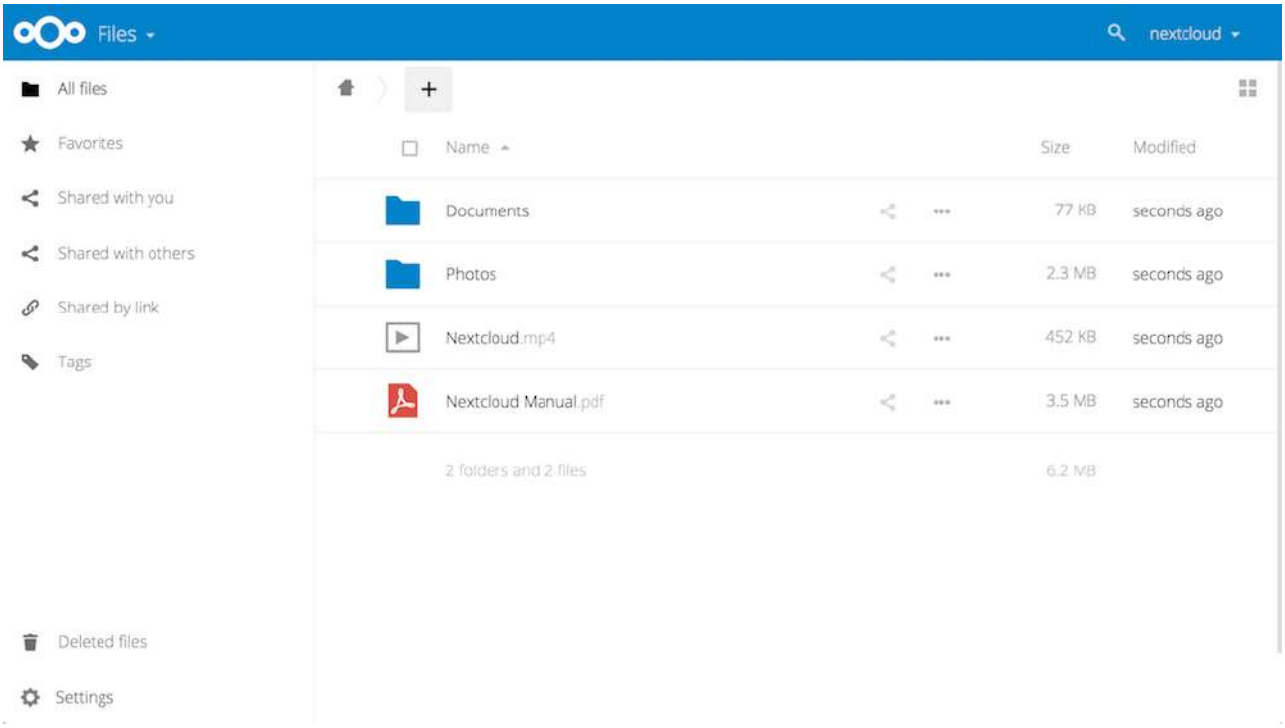

Рисунок 9 – Головна сторінка Nextcloud

На стороні безпеки Nextcloud надає документацію, яка пояснює кращі методи забезпечення безпеки. Небезпечні чи суперечливі функції строго відкидаються, а обов'язковий код перевіряється ще двома рецензентами, щоб переконатися, що все куленепробивне.

Крім того, система автентифікації Nextcloud складається з двох етапів. Активні сеанси можуть бути анульовані через список, шляхом видалення користувача в настройках адміністратора або шляхом зміни паролів. Адміністратори можуть включати або відключати двухетапну автентифікацію для користувачів в командному рядку.

У той час як інші пропрієтарні служби, такі як Dropbox, пропонують також 2FA, Nextcloud дозволяє вам «примусово» включати або відключати 2FA для конкретних членів організації. З невеликою кількістю налаштувань ви також можете створити свого власного постачальника 2FA, щоб уникнути блокування за допомогою спеціального додатку 2FA.

Nextcloud також пропонує шифрування на стороні сервера, яка не включене з коробки (щоб забезпечити більш зручний інтерфейс для новачків).

Сервери Nextcloud шифрують видалені дані, але ваше локальне сховище працює без шифрування за умовчанням. Важливо знати, що шифрування збільшує розмір файлу на 35%, що не обов'язково є порушником транзакції, якщо ви стурбовані безпекою.

Після включення шифрування на стороні сервера, його неможливо відключити в панелі управління адміністратора. Тому обов'язково потрібно створювати резервні копії ключів для користувачів шифрування. Якщо ключ втрачений, ці дані більше неможливо буде відкрити.

Ще одна з функцій Nextcloud - це федеративний обмін, який монтує загальні файли з віддалених серверів Nextcloud (або інших серверів, які його підтримують) для створення власного кластеру хмари. Уявіть собі, що в якості папок вам пропонується співпрацювати і мати доступ до Dropbox або Google Диску. Однак в цьому випадку він використовує відкритий протокол, який сумісний з багатьма різними постачальниками послуг (наприклад, ownCloud). Mounted Shares надає прямий доступ до папки, на яку вас запрошено, тому ваш простір на сервері не використовуватиметься для власної копії файлів.

Це дозволяє децентралізувати спільне використання файлів, незалежно від того, де зберігаються дані, з гнучкими правами адміністрування для установки індивідуальних прав користувачів. Ви можете використовувати ідентифікатор федеративної хмари, щоб ділитися файлами з іншими користувачами Nextclouders на основі вашого імені користувача (наприклад: username@example.com/nextcloud) [10].

З інформаційним додатком для серверу можна перевірити продуктивність вашого серверу з Nextcloud. Це може стати в нагоді для усунення неполадок або поліпшення на вашому сервері. Додаток є частиною установки Nextcloud, тому не потрібно нічого встановлювати вручну, щоб використовувати його. Він дозволяє відстежувати:

- Завантаження процесора і використання пам'яті
- Кількість активних користувачів з плином часу
- Кількість акцій в різних категоріях

- Статистика зберігання
- Параметри сервера, такі як версія PHP, тип і розмір бази даних, обмеження пам'яті і багато іншого.

3.1.3 Хмарне сховище Pydio

Pydio - рішення, яке виросло за п'ять років з файл-менеджера, який використовується для управління файлами на веб-сервері, в повноцінну платформу рівня підприємства для обміну даними між користувачами за допомогою веб-інтерфейсу, iOS- і Android-клієнта або протоколу WebDAV. Можливе просте створення міні-сайту, на якому будуть публікуватися списки розміщених документів. Доступно попередній перегляд для більшості поширених форматів (аудіо, відео, PDF, офісні документи). У разі зміни каталогу або файлу зацікавлені користувачі отримують сповіщення. Доступ до файлів можуть отримати як зареєстровані, так і анонімні користувачі.

Веб-інтерфейс локалізований (хоча і не повністю), побудований логічно і зрозуміло (рис. 10). Зліва зібрані всі ресурси (папка, загальні та закладки), вгорі панель дій (показуються тільки доступні), праворуч виводиться докладна інформація про вибраний файл. Самі файли відображаються у вікні посередині. Вид відображення можна змінити, непотрібні блоки прибрати. Деякі дії над файлами викликаються за допомогою контекстного меню. У загальному робота з Pydio нагадує настільний додаток.

Можлива автентифікація засобами Active Directory / LDAP, HTTP, CAS, FTP, OTP і іншими. Поділ прав засновано на ролях, які застосовуються до користувачів і груп. Адмініструвати сервер можуть кілька людей, яким чітко задаються права. Адміністратор має можливість слідкувати за діяльністю користувачів в режимі реального часу.

Забезпечується шифрування протягом сеансу HTTPS і даних на рівні файлової системи за допомогою EncFS. У червні 2013 року професійним агентством безпеки у Франції було проведено аудит Pydio, в результаті якого вразливості, специфічні для веб-додатків, не було виявлено.

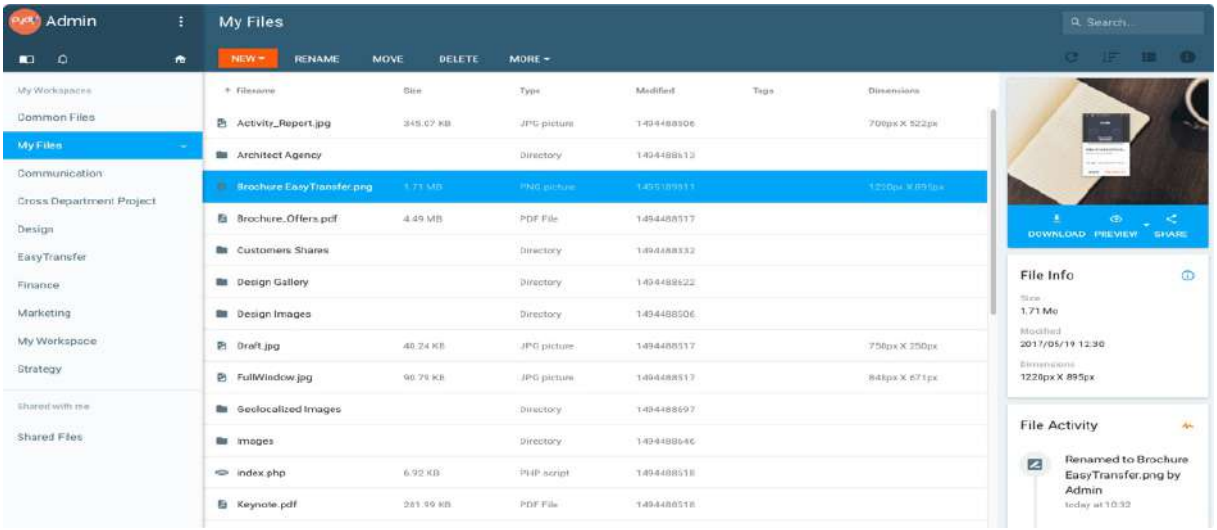

Рисунок 10 – Інтерфейс Pydio [11]

Також доступні плагіни (Bridges), що дозволяють інтегрувати Pydio в популярні CMS, що базуються на PHP, - Drupal, WordPress і Joomla. Доступний відповідне API, тому даний список легко розширити. Плагін Pydio for Filelink для Mozilla Thunderbird дозволяє автоматично замінювати великі вкладенні файли в повідомленні, генерованим посиланням на сховище Pydio.

Модульність дозволяє при необхідності наростити можливості системи і зібрати систему під конкретні потреби. Наприклад, забезпечити доступ до інших джерел даних (файлова система, FTP, SFTP, Samba, Amazon S3, Dropbox, HPCloud, IMAP, POP і так далі), перевіряти файли антивірусом. Також за допомогою плагінів підключається текстові і офісні редактори (через веб-сервіс Zoho), реалізується можливість перегляду зображень, програвання аудіо- і відеофайлів і багато іншого. Поведінка деяких модулів можна налаштовувати більш тонко, але для цього файли налаштування доведеться редагувати вручну.

Основні плагіни поставляються разом з архівом Pydio, інші доступні на офіційному сайті сзовища. Розробити свій плагін не так вже й складно, проект надає всю необхідну документацію і демонстраційні плагіни, які можна використовувати як основу для власних [11].

Для індексації та для швидкого пошуку по сховищу використовується бібліотека Apache Lucene.

Написаний Pydio з використанням HTML, PHP, Ajax і JavaScript. Використовуються стандартні драйвери файлової системи, тому сервер легко переносити і масштабувати.

3.1.4 Хмарне сховище Seafile

Наймолодший продукт огляд - Seafile. Перші версії були представлені в кінці 2012 року, але до релізу 1.3 інтерфейс користувача був тільки китайською мовою, тому популярність він лише починає набирати. У Seafile реалізовані не тільки функції зберігання і синхронізації даних, а й елементи спільної роботи з контентом. Користувач може створювати будь-яку кількість бібліотек (по суті, окремий віртуальне сховище) і відкривати доступ для груп або контактів без обмежень. Допущені користувачі через бібліотеку обмінюються файлами. У разі змін фалу передбачена можливість відправки сповіщень. При створенні бібліотеки можливо активувати доступу по паролю з кодуванням. Якщо ввімкнути функцію шифрування документа «закривається» до відправки на сервері (його можуть переглядати тільки допущені користувачі), підтримується HTTPS. На рівні бібліотеки також реалізовано відстеження версій (за замовчуванням 60 днів, можна змінити число зберігання днів, зберігати всю історію або відключити зовсім), доступ до попередніх версій файлу, відновлення видаленого файлу, аудит (хто і коли зробив зміни). Інтерфейс користувача є простим і інтуїтивним (рис. 11). Підтримується попередній перегляд основних типів файлів, обговорення інформації з учасниками групи, функції ведення списків завдань і управління проектами, і персональне Wiki. Ще одним плюсом є менше навантаження на сервер, в порівнянні з іншими учасниками огляду.

Доступ до даних можливий як через веб-інтерфейс, так і за допомогою клієнтів Seafile для Windows, Linux, Mac OS X, Android і iOS.

Код проекту написаний на мові Python і розповсюджується під ліцензією GPLv3, для зберігання метаданих використовується SQLite. Версія Community Edition серверної частини пропонується безкоштовно для Linux і Raspberry Pi,

для Windows, ціна складає 150 доларів. Також є Pro Edition з великими можливостями: доступ по WebDAV, функції пошуку, по електронній пошті оповіщення та інше [12].

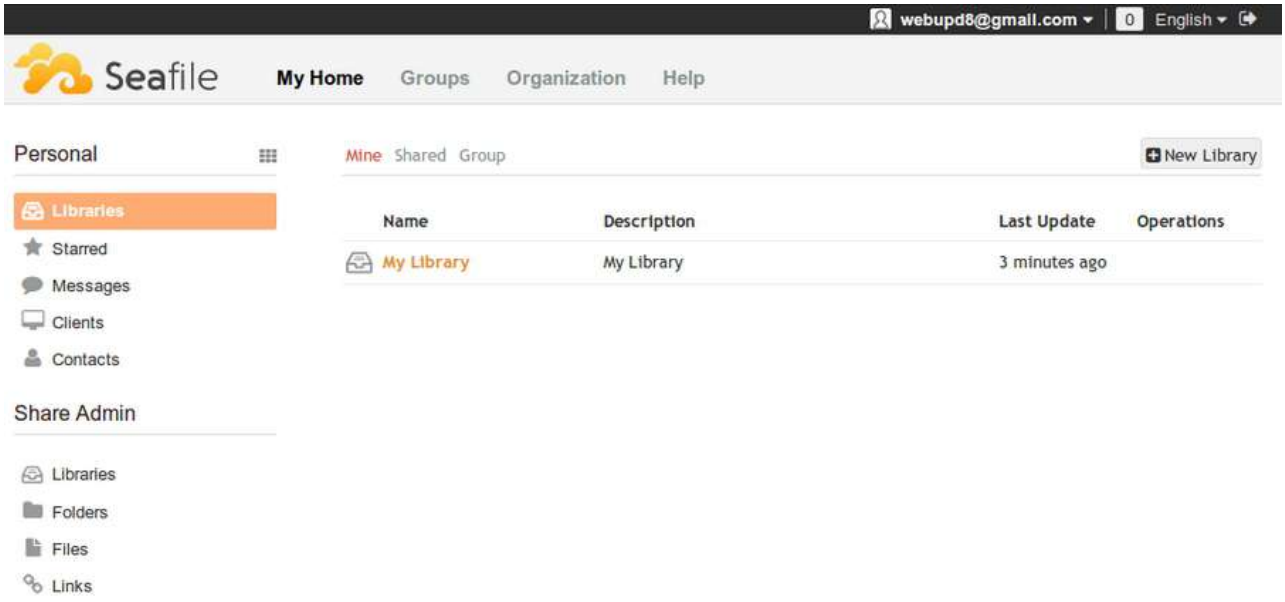

Рисунок 11 – Інтерфейс Seafile [12]

Основні можливості хмарних сховищ, було зведено у загальну таблицю 1, для побудови таблиці були використанні дані, тільки відкритих і безкоштовних рішень продуктів розглянутих вище.

Зробивши аналіз доступних хмарних сховищ для інформаційних сервіс для забезпечення навчального процесу, можна зробити висновок, що лідером є рішення від Nextcloud, він забезпечує багатий функціонал і відкритість системи, тим самим забезпечивши собі велику спільноту користувачів, які перейшли з Owncloud. Owncloud є цікавим проектом, але весь необхідний функціонал є платним, тому для вирішення завдання не підходить. Pydio – другий лідер у порівнянні за його лаконічність і швидкість роботи, але знову деякий потрібний функціонал є платним, тому він не підходить. Seafile є цікавим сховищем, написаним на Python, але його головним недоліком є недостатній функціонал і погана документація.

Таблиця 1 – Порівняння хмарних сховищ

Параметр	Owncloud	Nextcloud	Pydio	Seafile
Мобільний додаток	+	+	+	+
Контроль версій	+	+	+	+
Підтримка версій та дій	+	+	+	+
Попередній перегляд	+	+	+	-
Підтримка сторонніх додатків	+	+	+	-
Перегляд PDF, зображень, відео	+	+	+	-
Звіт у реальному часі	+	+	-	-
Закриття активних сесій	+	+	-	-
Завантаження перетягуванням файлу	+	+	-	-
Захист паролем спільних посилань	-	+	+	-
Підтримка протоколу LDAP	-	+	-	-
Зміна емблеми та кольору інтерфейсу	-	+	-	-
Наявність груп користувачів	-	+	-	-

3.2 Вибір хмарного офісу

Онлайнові офіси вже давно переросли статус «гидких каченят» і сьогодні є повноцінним робочим інструментом сотень тисяч користувачів. Більш того, завдяки своєму хмарному походженню ці веб-додатки часто пропонують навіть більш цікавий набір функцій, ніж їх десктопні конкуренти. У цьому огляді викладена інформація про найпоширеніші офіси: автономні рішення та рішення, принцип роботи яких заснований на моделі платформа як послуга. Також проведено порівняння офісів, визначені їх переваги та недоліки [13].

3.2.1 Хмарний офіс Google Docs

Комплект офісних програм від компанії Google є на сьогоднішній день безумовним лідером в цій категорії програмного забезпечення і пропонує повний набір функцій, який необхідний для роботи. У його комплект включені три додатки, що відповідають за створення і редагування текстових документів (рис. 12), електронних таблиць і презентацій відповідно. До недавнього часу до складу пакета входив і сервіс хмарного зберігання Google Drive, однак зараз компанія виділила його в окремий, хоча і як і раніше міцно пов'язаний з офісом сервіс.

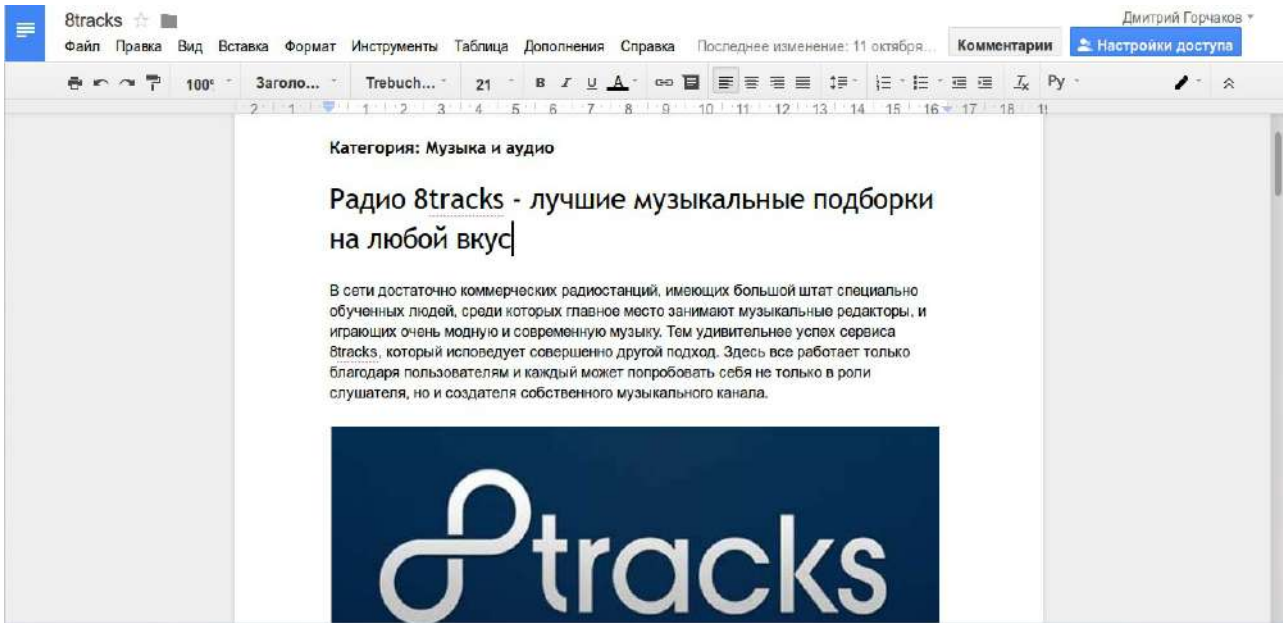

Рисунок 12 – Вигляд редактора текстових документів Google [14]

Спроба описати можливості офісних програм Google заздалегідь приречена на невдачу, так як навіть просте їх перерахування займе не одну сторінку. Тим більше, що робота над ними не припиняється ні на хвилину і щотижня з'являється якесь оновлення, що вводить підтримку нових функцій або усуває наявні помилки. Досить сказати, що функціонал додатків Google повністю перекриває потреби звичайного офісного працівника. Якщо ви не працюєте з файлами з надмірно складним форматуванням або занадто громіздкими таблицями з безліччю формул, то ніяких незручностей ви не відчуєте.

Окремим абзацом необхідно відзначити чудові здібності цього офісного пакету до спільної роботи. Завдяки пропрацював функцій спільного доступу, коментування, одночасного редагування ви зможете легко організувати роботу над документами віддалених співробітників або навіть цілої фірми.

Google Docs підтримують більшість популярних форматів файлів, у тому числі DOC, XLS, ODT, ODS, RTF, CSV, PPT і так далі, так що ви можете сміливо використовувати його для редагування вже готових файлів. Ніяких проблем, як правило, не виникає і при відкритті документів, створених за допомогою онлайнових додатків Google, в інших програмах [14].

3.2.2 Хмарний офіс Microsoft Office Online

Компанія Microsoft не стала винаходити велосипед, а пішла тим самим шляхом, який пройшов кілька років раніше Google. Ні, не можна сказати, що Microsoft Office Online зроблений за тим самим принципом, що і Google Docs, все ж досвід створення великих продуктів Word і Excel так просто не забувається, але загальні тенденції простежуються.

До складу Microsoft Office Online входять три додатки - Word Online, Excel Online і PowerPoint Online. Представляти кожен додаток окремо абсолютно немає необхідності, так як ви напевно знайомі з їх прообразами зі звичайного Microsoft Office. Тут ми бачимо схожий стрічковий інтерфейс, знайомі клавіатурні поєднання, звичні прийоми роботи. Все зроблено для того, щоб користувач офісних продуктів компанії не зазнав дискомфорту при переході в хмари.

Але деякі необхідні функцій ви на своїх місцях все ж не знайдете, так як можливості веб-додатків поки ще сильно відстають від звичайних Word, Excel і PowerPoint. Так, ви зможете створювати в Word Online красиві документи (рис. 13), але тільки в тому випадку, якщо навчитеся обходитися вбудованими стилями, тому що власні стилі тут створювати не можна. Так, Excel Online добре справляється навіть зі складними файлами XLS і XLSX, але про вкладені макросах доведеться забути. PowerPoint Online виглядає майже як справжній

редактор презентацій, але можна з упевненістю сказати, що будь-яка спроба створення більш-менш складного проекту приречена на провал [15].

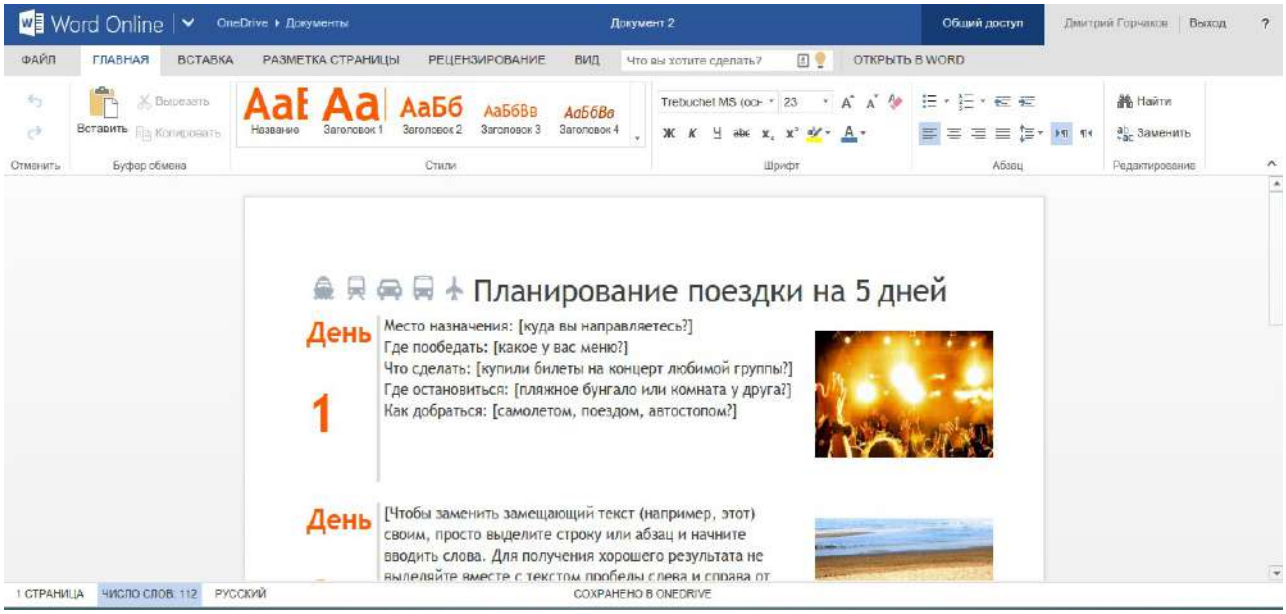

Рисунок 13 - Інтерфейс Microsoft Word Online [15]

В цілому, не хочеться лаяти Microsoft, адже їх онлайновий офіс за останнім часом зробив величезний крок вперед, але працювати ще доведеться дуже багато. В даний час Microsoft Office Online цілком підійде для роботи з простими документами, але не більше.

3.2.3 Хмарний офіс Zoho Office

Поки продукти Google і Microsoft сперечаються за звання найкращого офісного пакету, компанія Zoho цілком може собі дозволити стежити зі сторони за цією метушнею. Мало хто знає, але саме вона є першопрохідцями в сфері розробки онлайнових офісів і батьком найдосконалішого комплекту веб-додатків для роботи з документами. Zoho пропонує більше 25 різних програм і сервісів, які призначені як для корпоративного, так і індивідуального використання. Є серед них текстовий редактор Zoho Writer (рис. 14), програма для роботи з таблицями Zoho Sheet і інструмент для створення презентацій Zoho Show.

Інтерфейс цих сервісів є своєрідним мікс з стрічкового інтерфейсу всім відомого офісу і власних напрацювань. Він досить простий, інтуїтивно зрозумілий і досить гарний, так що працювати в такому офісі суцільне задоволення. За кількістю доступних функцій програми від Zoho не поступаються офісного пакету Google і точно перевершують Microsoft Office Online. Тут є все необхідне для форматування документів, перевірка правопису, повноцінна робота з малюнками, формулами і таблицями, автоматичне збереження, інструментарій для публікації і спільної роботи над документами.

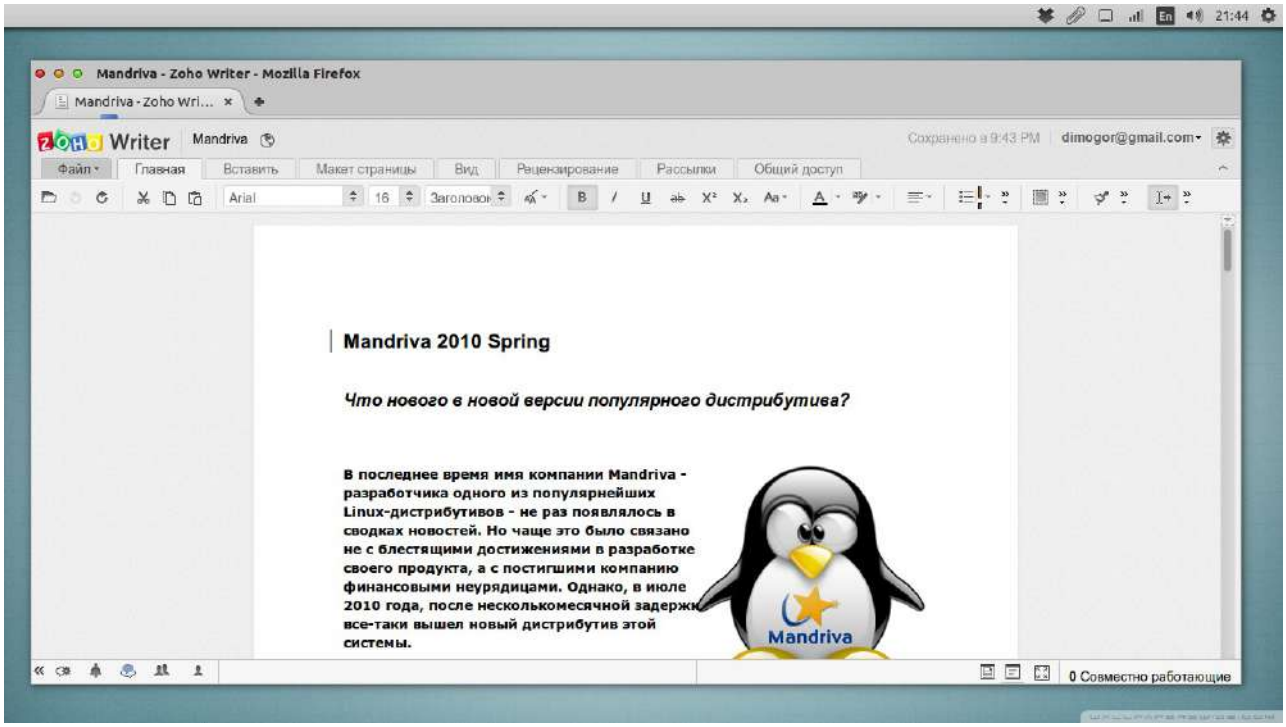

Рисунок 14 - Інтерфейс Zoho Writer [16]

Для зберігання файлів використовується безкоштовне сховище Zoho Docs, в якому кожному користувачеві виділяється по 5 Гб. Тут можна зручно розсортувати файли по папках, завантажити їх на свій комп'ютер, надати спільний доступ або відправити по електронній пошті. До речі, для роботи з Zoho Docs можна завантажити спеціальний клієнт, версії якого є для всіх операційних систем. Крім цього, ви можете налаштувати інтеграцію Zoho з хмарним сервісом Dropbox або Box і працювати зі збереженими там документами [16].

3.2.4 Хмарний офіс OnlyOffice

Якщо ви думаєте, що розробку повноцінного комплекту веб-додатків для офісу можуть собі дозволити тільки великі і потужні компанії, то продукт латвійської компанії Ascensio System SIA розвіє це переконання. Він містить всі компоненти для повного перенесення роботи в хмару, в тому числі набір інструментів для управління проектами та просту CRM. Але нас цікавить перш за все пакет офісних додатків, що складається з Document Editor, Spreadsheet Editor і Presentation Editor.

Інтерфейс програм OnlyOffice (рис. 15) найбільше нагадує сильно перероблений Microsoft Office 2003, тобто ніяких стрічкових меню та інших нововведень. Багатьом таке рішення припаде до смаку, тим більше, що розробники добре попрацювали над удосконаленням вже знайомих інструментів.

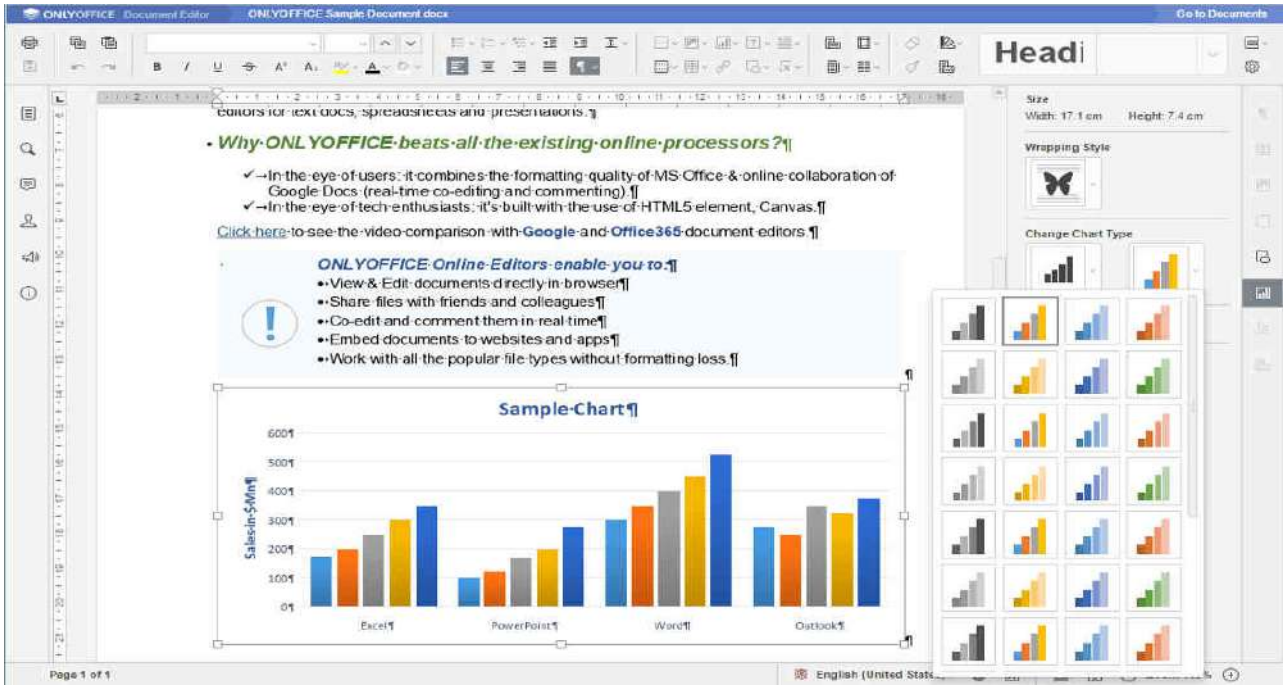

Рисунок 15 – Інтерфейс Onlyoffice Document Editor

У верхній частині вікна редагування знаходиться основна панель інструментів, зліва - невелика панель з кнопками пошуку, чату, коментування, а праворуч - кнопки для роботи з таблицями, картинками, кліп артом і діаграмами.

З точки зору функціональності OnlyOffice цілком придатний до роботи над тестовими документами, електронними таблицями і презентаціями, проте відсутність деяких звичних можливостей будуть змушувати користувача збентежено чесати в потилиці. Так табличний редактор страждає від відсутності автодоповнення, а в редакторі презентацій не спостерігається навіть будь-якого натяку на ефекти переходу і анімації.

Але проблему функціоналу вони вирішили найкращим відкриттям документу у порівнянні з попередніми рішеннями. При відкритті складних документів (в яких присутні інші елементи: графіки, колонтитули тощо), які були створенні у десктопних редакторах, форматування документу зберігається і залишається можливості його редагування. Наприклад при відкритті документу в Google Docs з графіками, графіки перетворюються на зображення, яке вже користувач не зможе відредагувати в онлайн офісі.

Інструменти для спільної роботи в рамках OnlyOffice складаються з вбудованого чату, системи коментування та можливості тонкого налаштування прав доступу до файлів. З цієї точки зору немає ніяких нарікань - все реалізовано грамотно, зручно, зрозуміло. Дуже також порадувала можливість інтеграції цього хмарного офісу з такими сервісами як Dropbox, Box, OneDrive, а також імпорт документів з Google Drive і Zoho Docs.

Однією з проблем даного офісу є відсутність локалізацій, тому він зараз працює тільки з англійською мовою, але дана проблема вирішується наявністю документації на російській мові, розробники обіцяють додати вибір мови інтерфейсу у наступній версії продукту [17].

Всі основні можливості хмарних офісів зведено у таблицю 2 для їх порівняння.

Безсумнівним лідером серед офісних веб-додатків, як і раніше, є Google Docs. Його можливості з лишком перекривають потреби більшості користувачів, але деякий його функціонал є обмеженим і реалізованим не повність, наприклад шрифти можна добавляти тільки з бібліотеки Google Fonts. Не варто скидати з рахунків продукт Zoho Office. Його відточена роками якість поможе

користувачеві вам працювати з документами буквально будь-якої складності. Продукт компанії Microsoft поки дещо відстає від лідерів, але в компанії докладають великих зусиль для його розвитку, так що в найближчому майбутньому все ще може змінитися. Але один з самих молодих офісів OnlyOffice, є найбільш перспективним і функціональним продуктом. Його не достатки перекриваються добре реалізованим функціоналом. Даний продукт відповідає усім вимогам, які були виставленні у попередньому розділі.

Таблиця 2 – Порівняння можливостей хмарних офісів

Можливість	Google Docs	MS Office Online	Zoho Docs	OnlyOffice
Розширене форматування шрифту	+	+	+	+
Розширене форматування абзацу	+	+	+	+
Розширене форматування таблиці	-	-	+	+
Зображення	+	+	+	+
Автофігури	+/-	-	+	+
Діаграми	+/-	-	-	+
Формули	+	+	+	+
Створення та редагування стилів	+	-	+	+
Розриви розділів	-	+	-	+
Колонтитули, номери сторінок	+	+	+	+
Кольорові схеми	+	+	-	+
Додавання нового шрифту	+/-	-	-	+/-
Автозаповнення таблиці	+	+	+	-
Обмеження у безкоштовних версіях	-	+/-	+	-

3.3 Висновки

Отже, згідно з вимогами до складових інформаційних сервісів для підтримки навчального процесу, який було зроблено у попередньому розділу, і на основі аналізу можливих варіантів реалізації сервісів було обрано: для хмарного сховища продукт Nextcloud, а для хмарного офісу – продукт OnlyOffice.

Nextcloud підтримує найбільший функціонал у безкоштовній версії і має велику спільноту користувачів. Так як увесь код є відкритим і добре документованим, то розширити функціонал, в якому може виникнути потреба у майбутньому, не складе труднощів.

Для хмарного офісу було обрано продукт OnlyOffice, так як в даній роботі розробляється автономна система для інформаційних сервісів. Він має простий інтерфейс та достатній для вимог кафедри функціонал. Якщо розглядати варіант побудови напів автономної системи, в якій би використовувався хмарний офіс, встановлений на сторонніх серверах, то найкращим вибором можна вважати Google Docs. Він є добре документованим і має зручний API для його підключення. Недоліком такого рішення є проблеми, які можуть виникнути при зміні ліцензійної угоди чи блокування домену доступу до даного сервісу.

4 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНИХ СЕРВІСІВ В МЕРЕЖІ КАФЕДРИ СП

У цьому розділі буде розглянуто основні архітектури хмарних обчислень в залежності від навантаження на систему. Та буде розроблено архітектуру інформаційних сервісів для підтримки навчального процесу. Після буде надана повна інструкція по створенню і налаштуванню інформаційних сервісів.

4.1 Архітектура компонентів хмарного сховища

Так як система складається з двох сервісів, то ми повинні розглядати дві різні архітектури, але основні принципи побудови залишаються не змінними, тому розглянемо загальні принципи побудови для хмарного сховища, а для хмарного офісу використаємо аналогічну модель.

Типова схема хмарного сховища наведена на рис. 16:

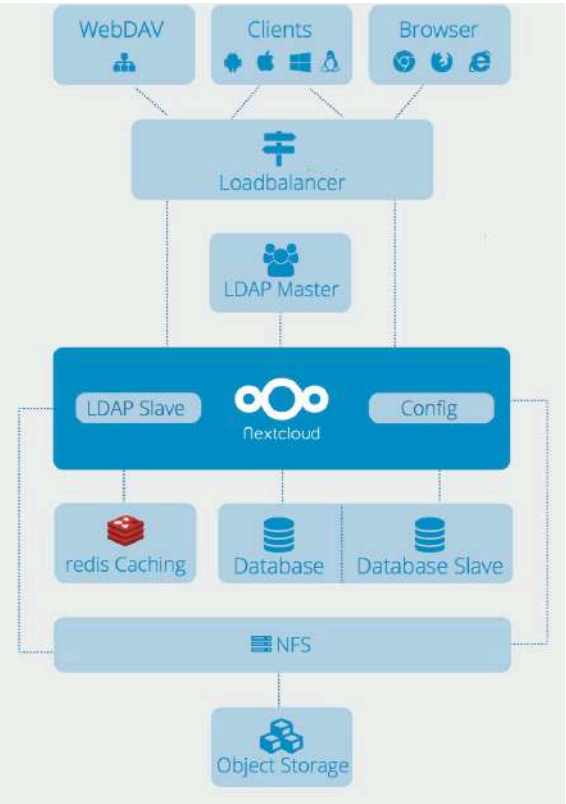

Рисунок 16 – Схема взаємодії компонентів хмарного сховища [18]

Хмарне сховище складається з таких компонентів:

1) Loadbalancer – сервіс, який регулює завантаженість основних сервісів.
2) Redis Caching – сервіс, для кешування даних.
3) LDAP – сервіс, для централізованого зберігання облікових записів.
4) Database – база даних, в якій зберігаються дані сервісу.
5) NFS – мережева файлова система, яка надає доступ до файлів користувачів, що зберігаються у файловому сховищі.

Для доступу користувача застосовується один із інтерфейсів доступу:

- Веб-браузер
- Клієнти написанні для різних платформ
- Протокол доступу WebDAV

Після того, як користувач зайшов через один із інтерфейсів проходить процедуру автентифікації. Для спрощення керування Nextcloud інтегрується в наявну інфраструктуру обробки облікових записів підтримуючи протокол LDAP, на сервері LDAP задається група для користувача і квота доступної пам’яті.

Після успішної процедури автентифікації користувач потрапляє до свого сховища. Всі дані необхідні для функціювання сайту зберігаються у базі даних, для пришвидшення доступу використовують кеш-сервер Redis. Для зберігання файлів користувачів використовується локальне сховище, яке використовує віртуалізацію сховищ. Також можливо використати мережеву файлову систему з такими сховищами: GFS2, Windows Network Drive, CIFS, Red Hat Storage, IBM Elastic Stor або сховище об’єктів, сумісне с SWIFT и S3 [18].

4.2 Архітектура системи в залежності від навантаження

В залежності від навантаження, яке будуть створювати користувачі та кількості даних, яка буде зберігатися у хмарному сховищі, схема взаємодії компонентів (див. рис. 16), буде змінювати свій вигляд і розміщення компонентів. Виділяють такі групи користувачів:

- ***Невеликі робочі групи або відділу***

В даному випадку розуміють групу до 250 людей працюючих одночасно і розмір всіх зберігаючих даних до 10 Тб. Веб-сервер, база даних і сховище даних зберігаються на одній машині і підключаються до LDAP-сервісу, який знаходиться на іншому сервері. Схема цього підключення наведена на рис. 17.

Рисунок 17 – Архітектура для малої групи користувачів [10]

- ***Середні робочі групи чи малі підприємства***

В даному випадку група до 1000 людей працюючих одночасно і розмір і розмір всіх зберігаючих даних до 200Тб. В даному випадку перед входом систему стоїть балансер навантаження, який розподіляє навантаження між двома веб-серверами, кеш, LDAP і файлове сховище винесенні на окремі сервери. База даних розподілена між двома серверами і працює у режимі Master-Slave. Схема зображена на рис. 18.

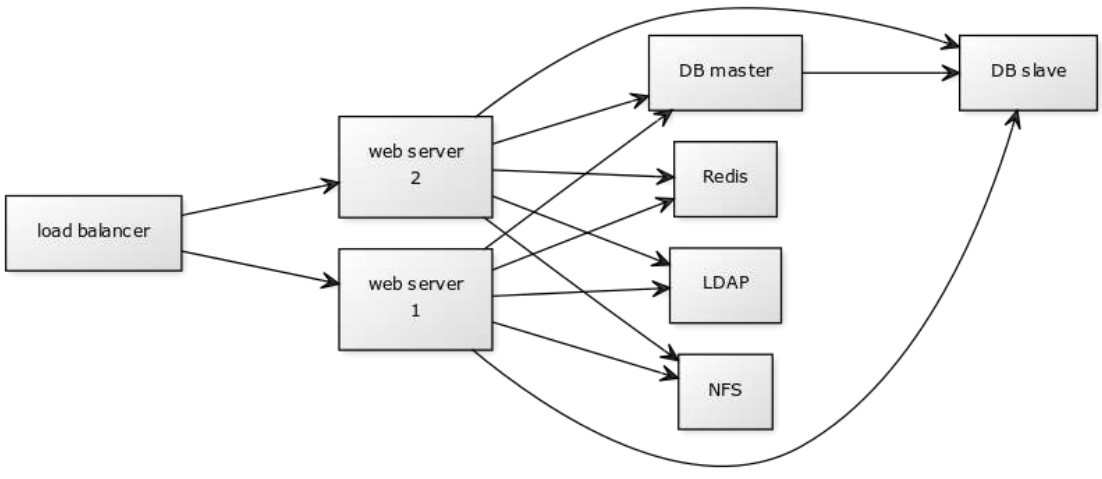

Рисунок 18 – Архітектура для середньої групи користувачів [10]

- ***Великі підприємства і постачальники послуг***

В даному випадку група до 100000 людей працюючих одночасно і розмір і розмір всіх зберігаючих даних до 1Пб. В даному випадку перед входом систему

стоїть два балансери навантаження, який розподіляють навантаження між чотирма веб-серверами і на кожному сервері є локальна копія LDAP (slave LDAP), файлове сховище винесено на окремий сервер. База даних має свій власний балансер навантаження і працює у режимі Master-Slave. Кеш розподілений між двома серверами [19]. Схема зображена на рис. 19.

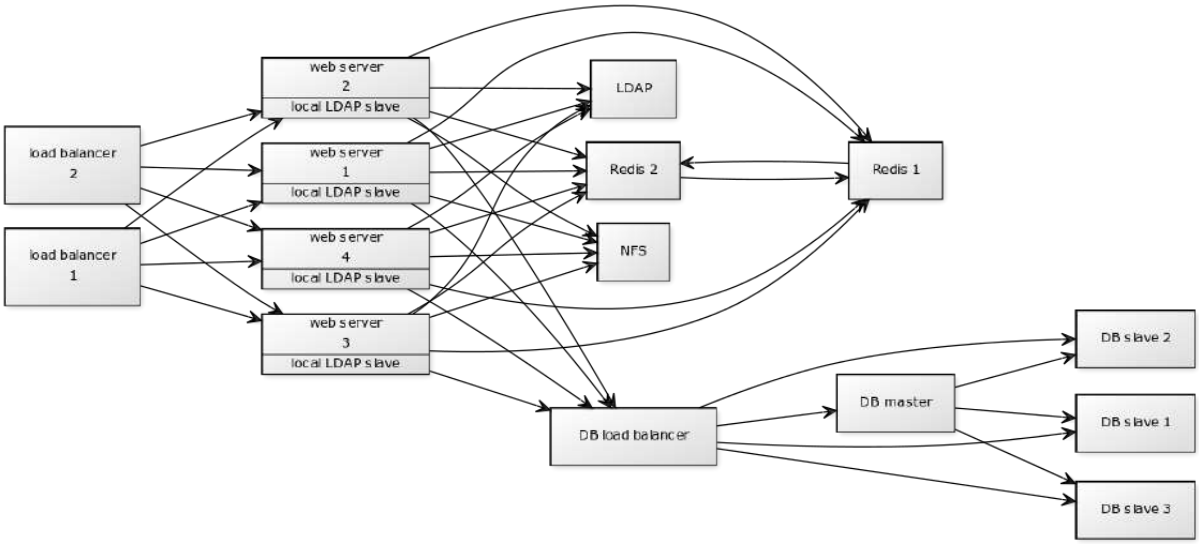

Рисунок 19 – Архітектура для великої групи користувачів [10]

4.3 Архітектура компонентів інформаційних сервісів

Так як на кафедрі невелика кількість студентів і кількість матеріалу <10 ТБ, то буде використовувати архітектуру для малої кількості користувачів.

Розглянемо можливі варіанти підключення системи, до кафедральної веб-сторінки:

- *Встановлення перенаправлення на веб-сторінці кафедри*

  Даний метод є самим простим у реалізації, і дозволяє отримувати доступ до системи, як і через веб-сторінку кафедри так і через пряме посилання. Мінусами даного підходу є те, що для кожного нового інформаційного сервісу буде потрібне окреме доменне ім’я та авторизація користувача повинна бути організована на кожному сервісі окрема.

- *Розробка модулю для веб-сторінки з авторизацією у системі*
  Даний підхід вимагає написання модулю для веб-сторінки на мові PHP. Модуль представляє доступ до сховища, і авторизацію на ньому. Перевагою даного методу є більша захищеність, також відпадає необхідність в окремому доменному імені для звернення. Недоліком даного підходу, що авторизація користувач буде робити на кожним сервісом окремо.
- *Розробка модулю з авторизацією у системі на веб-сторінки кафедри*
  Даний метод вимагає доопрацювання кафедральної веб-сторінки, з додаванням авторизації через протокол LDAP і зміну даних на сервері LDAP і також написання модулю для веб-сторінки. При даному підході користувач один раз авторизувавшись на веб-сторінці кафедри буде мати доступ до всіх інформаційних сервісів.

Для підключення до веб-сторінки кафедри виберемо перенаправлення, як самий простий і швидкий варіант реалізації. З поставлених вимог, що інформаційні сервіси для забезпечення навчального процесу, мають бути реалізовані на платформі Docker, будемо використовувати готові образи для реалізації. На рис. 20 наведена схема взаємодії компонентів системи. На схемі показана архітектура системи з усіма протоколами, по яких відбувається обмін даними. У системі тільки в контейнері Web-server Nextcloud є зв’язок типу “контейнер-хост”, тобто він виступає мостом між віртуальною мережею і мережею хоста.

Архітектура розроблена на основі віртуальних мереж, що підтримуються платформою Docker при взаємодії контейнерів. Контейнери всередині мережі взаємодіють за зв’язком типу “контейнер-контейнер”. Тому створимо дві віртуальні мережі для взаємодії контейнерів інформаційних сервісів. Перша мережа, що виділена зеленим кольором, слугує для взаємодії контейнерів хмарного офісу. Друга мережа, що виділена червоним кольором, слугує для взаємодії контейнерів хмарного сховища. Стрілки, які з’єднують контейнери одного кольору, вказують на те що порти для взаємодії відкриті тільки в межах

даної мережі. Стрілки, які з’єднують контейнери різного кольору, вказують на те, що порти для взаємодії відкриті для доступу з інших мереж.

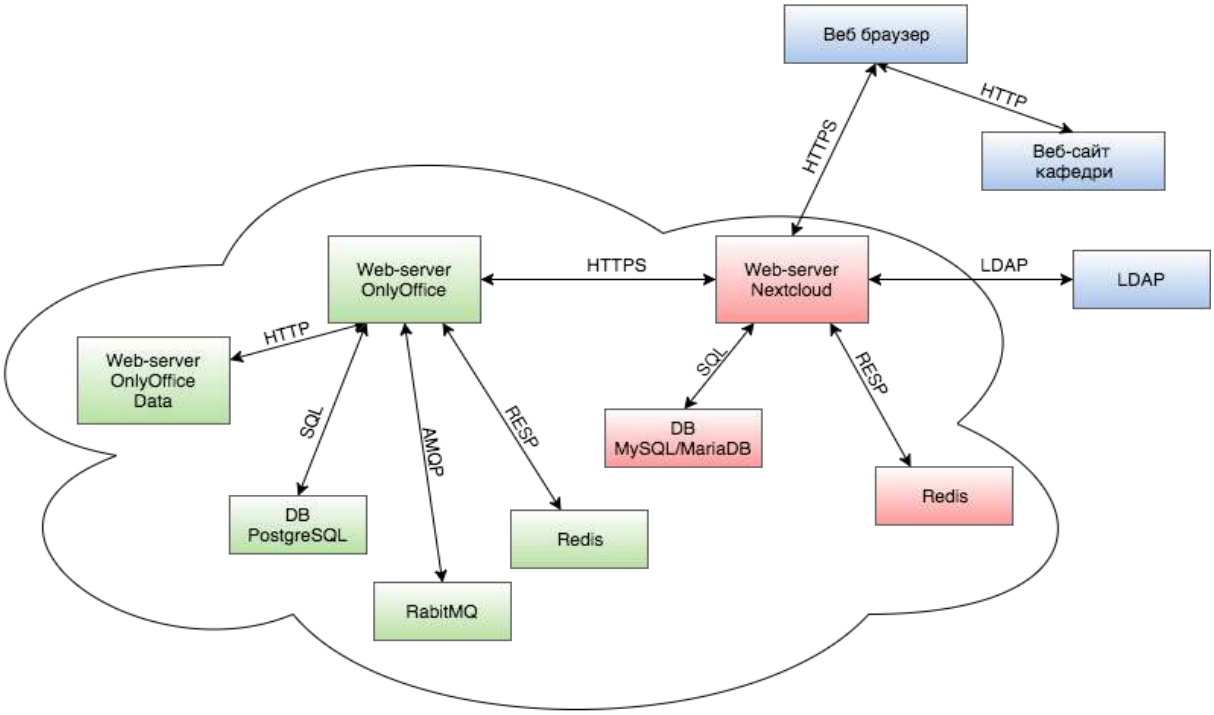

Рисунок 20 – Архітектура інформаційних сервісів на платформі Docker

Система побудована з врахуванням можливості горизонтального масштабування. Для масштабування потрібно буде додати контейнер з HAProxy. Для хмарного сховища, масштабування полягає у нарощенні кількості контейнерів з Web-server Nextcloud. А для масштабування хмарного офісу, було створено контейнер Web-Server OnlyOffice Data, який створений з того самого образу, що і контейнер Web-Server OnlyOffice. Необхідність у ньому виникла із специфіки роботи хмарного офісу, а саме можливості одночасної праці декількох користувачів над одним документом. Тому необхідно мати спільне сховище даних для всіх контейнерів, а не локальні в кожному, бо неможливо гарантувати потрапляння всіх користувачів, які працюють над спільним документом на один веб-сервер.

Контейнери взаємодіють через відповідні протоколи взаємодії, розглянемо їх схему див. рис. 20.

Користувач, відкриваючи веб-сторінку кафедри, надсилає запит через протокол HTTP і отримуючи у відповідь веб-сторінку кафедри. Після цього на сторінці він обирає пункт “інформаційні сервіси” і перенаправляється на домене ім’я хмарного сховища, надсилає до його запит через протокол HTTPS і отримує у відповідь веб-сторінку авторизації хмарного сховища, де проходить автентифікацію.

Для автентифікації користувача використовується централізований сервіс прав користувачів, з яким йде спілкування через протокол LDAP, після підтвердження прав користувача перевіряється, чи є про його інформація в кеші Redis, через протокол RESP. Якщо інформація є, то згідно даної інформації формується веб-сторінка користувача з його даними, якщо інформації немає, то запит для її отримання надсилається до бази даних.

Коли користувач хоче звернутися до хмарного офісу для створення нового документу чи редагування старого, запит надсилається до веб-серверу OnlyOffice через протокол HTTPS. Після чого завантажується сторінка з онлайн редактором документу.

При завантаженні онлайн редактору з початку в кеші Redis з початку перевіряється, чи хтось редагує даний документ. Якщо так, то підтягуються данні з бази даних, якщо документу немає на сервері, то додаються данні до кешу Redis і бази даних зі сховища. Також використовується сервіс RabbitMQ, для обробки черги задач конвертації і їх результатів, також він використовується як шина подій. Даний сервіс виступає у ролі брокеру повідомлень, з яким йде спілкування за протоколом AMQP.

4.4 Встановлення інформаційних сервісів

У даному підрозділі надана детальна інструкція по встановленню і налаштуванню інформаційних сервісів для підтримка навчального процесу на платформі Docker.

4.4.1 Встановлення і налаштування платформи Docker

Тепер перейдемо до самого встановлення сервісу, інструкція наведена для системи Ubuntu 16.04 LTS:

1) По-перше, оновимо базу даних пакетів:
   $ sudo apt-get update
2) Встановлення платформи Docker. Додамо ключ GPG для офіційного репозиторію Docker до системи:
   $ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
3) Включимо репозиторій Docker в APT джерел:
   $ sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'
4) Оновим бази даних пакетів з пакетами Docker:
   $ sudo apt-get update
5) Встановлення Docker в систему:
   $ sudo apt-get install -y docker-engine
6) Встановимо нову версію docker-compose:
   $ sudo curl -o /usr/local/bin/docker-compose -L "https://github.com/docker/compose/releases/download/1.13.0/docker-compose-$(uname -s)-$(uname -m)"
7) Далі ми надаємо права доступу для docker-compose:
   $ sudo chmod +x /usr/local/bin/docker-compose

4.4.2 Встановлення інформаційних сервісів

Для спрощення розповсюдження всі необхідні файли завантажено на GitHub, потрібно тільки завантажити їх:

1) Для того, щоб завантажити документи з git-репозиторію, необхідно встановити git:
   $ sudo apt-get install git

2) Перейдемо в парку де буде встановлюватися система і скопіюємо файли з репозиторію GitHub, в яких міститься конфігурація файл docker-compose, з усіма зв’язками і налаштуваннями системи:
   $ git clone https://github.com/vvlladd28/cloud.git .
3) Запустимо систему на встановлення через docker-compose, буде завантажено образи з репозиторію Docker Hub і створено контейнери з усіма їх зв’язками:
   $ sudo docker-compose up -d

4.4.3 Налаштування HTTPS з’єднання з системою

1) Створимо закриті ключи сертифікатів
   $ openssl genrsa -out nextcloud.key 2048
   $ openssl genrsa -out onlyoffice.key 2048
2) Створимо запит на підпис сертифікатів (CSR)
   $ openssl req -new -key nextcloud.key -out nextcloud.csr
   $ openssl req -new -key onlyoffice.key -out onlyoffice.csr
3) Підпишемо сертифікати за допомогою закритого ключа і CSR
   $ openssl x509 -req -days 365 -in nextcloud.csr -signkey nextcloud.key -out nextcloud.crt
   $ openssl x509 -req -days 365 -in onlyoffice.csr -signkey onlyoffice.key -out onlyoffice.crt
4) Встановлення SLL-сертифікатів
   $ mkdir -p ./nextcloud/data/certs
   $ cp nextcloud.key ./onlyoffice/data/certs/
   $ cp nextcloud.crt ./onlyoffice/data/certs/
   $ chmod 400 ./onlyoffice/data/certs /nextcloud.key
   $ mkdir -p ./onlyoffice/data/certs
   $ cp onlyoffice.key ./onlyoffice/data/certs
   $ cp onlyoffice.crt ./onlyoffice/data/certs
   $ chmod 400 ./onlyoffice/data/certs/onlyoffice.key

4.4.4 Налаштування інформаційних сервісів

В нас є працюючий сервіс, до якого є доступ по IP-адресі, для доступу по доменному імені потрібно його прописати на DNS сервері. Для цього зверніться до системних адміністраторів. Але налаштування самого хмарного і його зв’язку з хмарним офісом, ще не зроблено.

Для налаштування інформаційних сервісів перейдіть по IP-адресі серверу чи доменному іменні, тоді відкриється сторінка рис. 21 де потрібно ввести логін для адміністратора та його пароль.

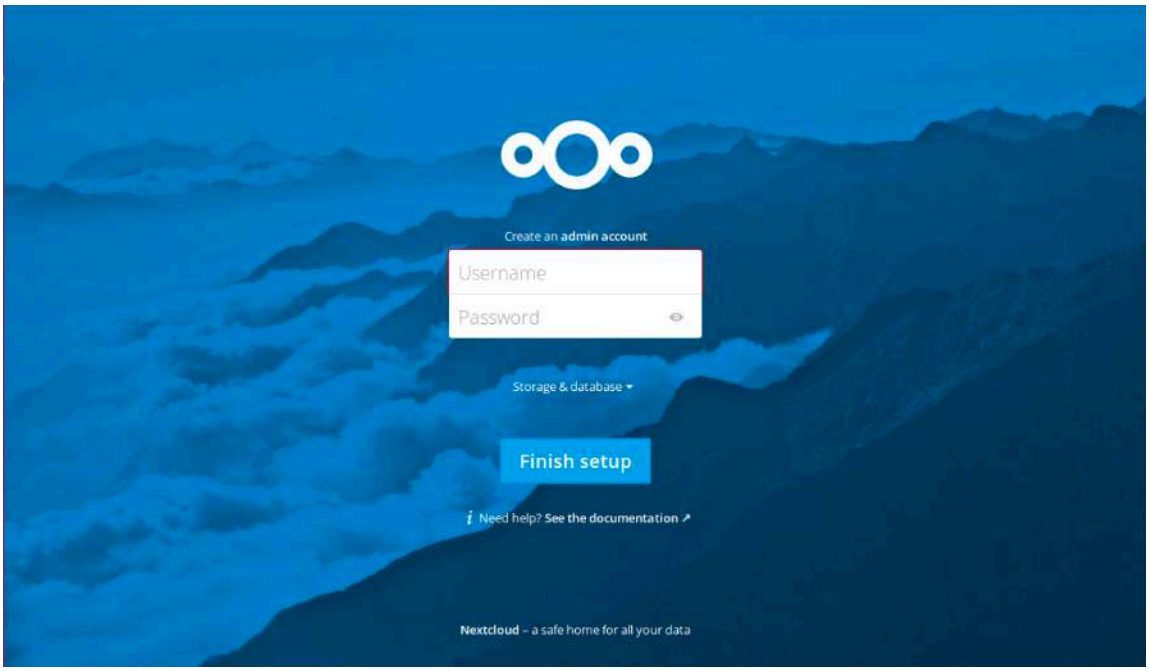

Рисунок 21 – Сторінка першого запуску хмарного сховища

При натисканні на напис “Storage & Database” (рис. 22) відкрається розширене меню налаштування.

В даному меню потрібно прописати такі дані:

- Data folder – шлях до місця зберігання даних користувачів.
- Configure the database – вибрати базу даних, що буде використовуватися (MySQL/MariaDB), та заповнити такі поля:
  - Database user – ім’я користувача для бази даних
  - Database password – пароль користувача базою даних
  - Database name – назва бази даних на сервері

- Database domen – адрес серверу з базою даних

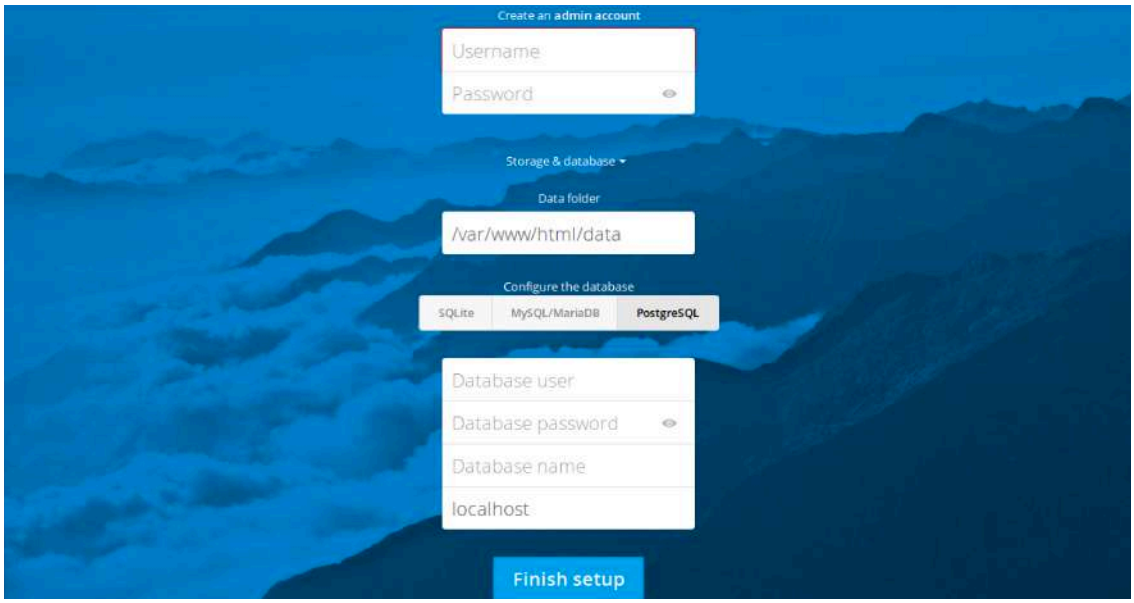

Рисунок 22 – Меню розширеного налаштування

Після введення всіх даних потрібно натиснути кнопку “Finish setup”, після буде завантажено вікно з головною сторінкою хмарного сховища (рис. 23). Також буде показане вікно з привітанням та посиланнями для завантаження офіційних клієнтів для різних платформ для більш комфортнішого використання сервісу. Також в сховищі буде знаходиться файл з інструкцією для користувача по використанню сховищем даних “Nextcloud Manual.pdf” та демонстраційні файли роботи з функціоналом.

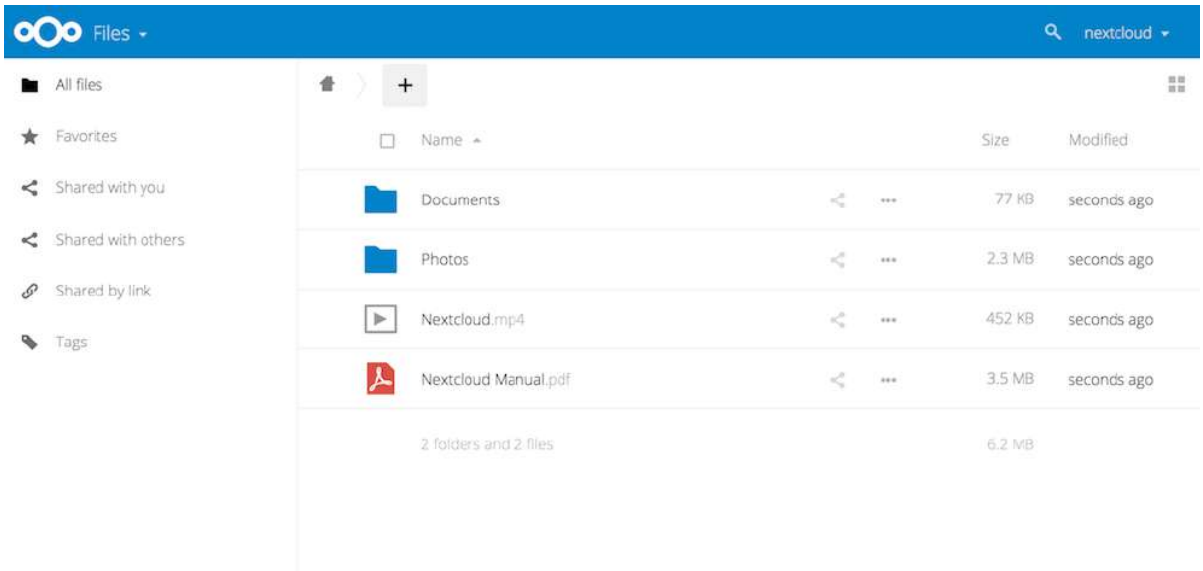

Рисунок 23 – Головна сторінка хмарного сховища

Для подальшого налаштування активуємо можливості, які за замовченням є не доступні. Натиснувши на кнопку “Files”->”Apps”, потрапляємо у менеджер додатків, він доступний тільки адміністраторам. У правом бічному меню обираємо пункт “Not enabled” (рис. 24).

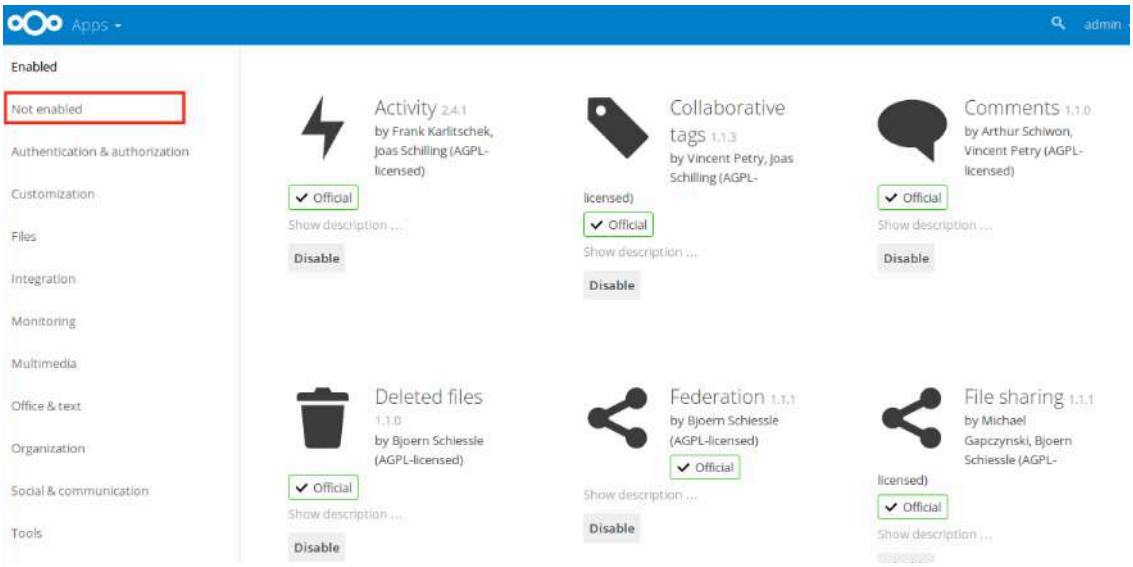

Рисунок 24 – Менеджер додатків

У менеджері додатків шукаємо такі додатки “LDAP user and group backend” і “Onlyoffice” під назвою натискаємо на кнопку “Enabled”, щоб вони стали доступні. Після переходимо у панель адміністратора, натиснути ім’я користувача –> Admin (рис. 25).

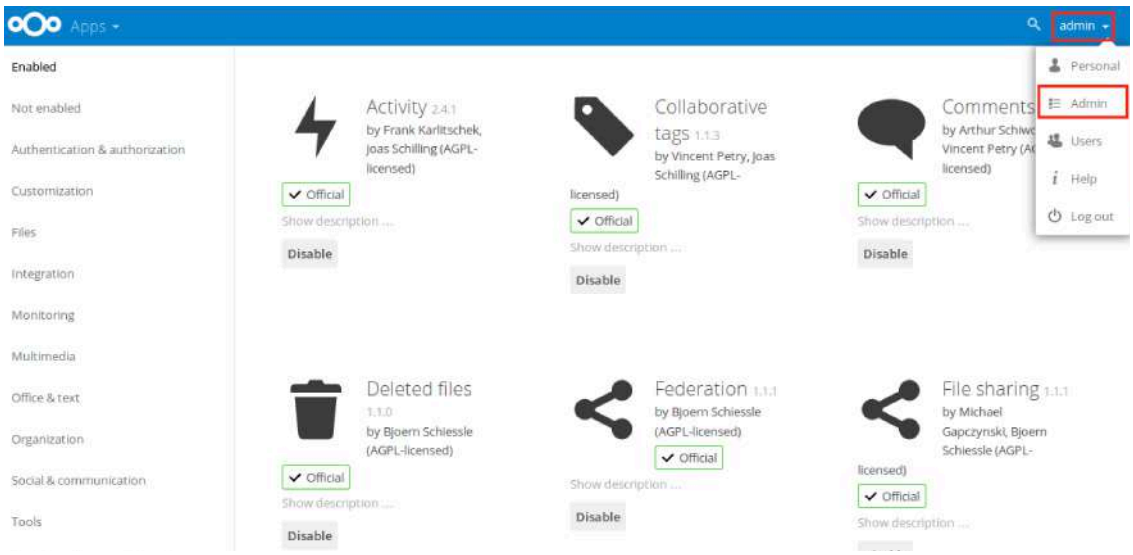

Рисунок 25 – Перехід в панель адміністратора

В панелі адміністратора злускаємось вниз сторінки до пункту ONLYOFFICE (рис. 26). В даному пункті потрібно прописати в полі Document Editing Service address ip-адресу, дописавши в кінці номер порту 81 (в кінці дописати :81) чи доменне ім’я хмарного сховища і натиснути кнопку “Save”.

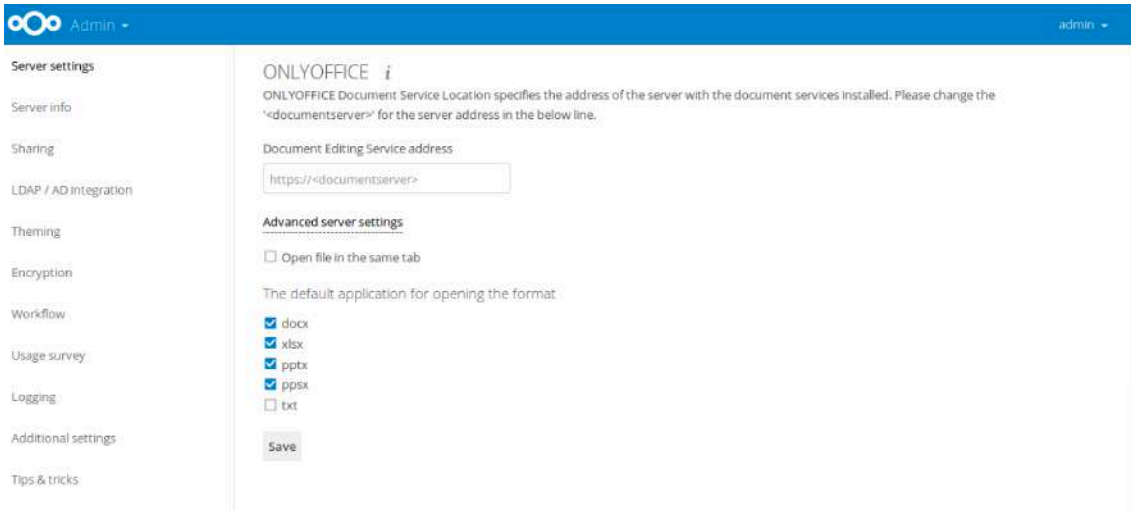

Рисунок 26 – Пункт налаштування Onlyoffice

4.5 Висновки

У даному розділи було описано ряд типових архітектур розподілених систем в залежності від їх навантаження. На основі підрахунків кількості користувачів та приблизного обсягу інформації, яку потрібно зберігати, було запропоновано архітектуру, з відповідає потребам забезпечення навчального процесу в мережевій інфраструктурі кафедри СП. Взаємодія компонентів системи відбувається за допомогою зв’язків контейнерів, в яких вони знаходяться. Це дозволяє нам винести компоненти системи на окремі сервери, не змінюючи системи.

В розділі надана покрокова інструкція встановлена платформи Docker, інструкція по копіюванню файлів з GitHub, і запуску зібраної системи за допомогою надбудови docker-compose над платформою Docker, що звільняє нас

від необхідності всі данні і зв’язки контейнерів вводити через командний рядок. Далі надана інструкція, по налаштуванню хмарного сховища і підключення хмарного офісу до нього. Результатом виконання всіх цих простих дій, ми отримаємо робочий інформаційний сервіс для підтримки навчального процесу.

5 ТЕСТУВАННЯ ІНФОРМАЦІЙНИХ СЕРВІСІВ

5.1 Методи тестування

Тестування програмного забезпечення - процес виявлення помилок у програмному забезпеченні (ПЗ). На жаль, існуючі на сьогоднішній день методи тестування ПО не дозволяють однозначно і повністю усунути всі дефекти і помилки і встановити коректність функціонування аналізованої програми особливо в закритих приватних програмах. Тому всі існуючі методи тестування діють в рамках формального процесу перевірки досліджуваного або розроблюваного ПЗ.

Такий процес формальної перевірки або верифікації може довести, що дефекти відсутні, з точки зору використовуваного методу. (Тобто немає ніякої можливості точно встановити або гарантувати відсутність дефектів у програмному продукті з урахуванням людського фактора, присутнього на всіх етапах життєвого циклу ПЗ).

Існує безліч підходів до вирішення завдання тестування і верифікації ПЗ, але ефективне тестування складних програмних продуктів - це процес найвищою мірою творчий, що не зводиться до слідування строгим і чітким процедурам або створення таких.

Тестом називається інформація, яка складається з вихідних даних, спеціально підібраних для налагоджують програму, і відповідних їм еталонних результатів (не тільки остаточних, але і проміжних), використовуваних надалі для контролю правильності роботи програми.

Вихідні дані необхідно підбирати з метою виявлення помилок, такий підхід помітно підвищить якість програми.

Тестування - процес деструктивний (тобто зворотний творчому, конструктивному).

Основні принципи тестування:

1) Опис передбачуваних значень вихідних даних або результатів має бути необхідною частиною тестового набору. Помилкові, але правдоподібні результати можуть бути визнані правильними, якщо результати тесту не були заздалегідь визначені.
2) Слід уникати тестування програми тільки її автором.
3) Необхідно досконально вивчати результати застосування кожного тесту.
4) Тести для неправильних і непередбачених вхідних даних слід розробляти так само ретельно, як для правильних і передбачених. Так як цілком імовірно, що тести, що представляють невірні і неправильні вхідні дані, володіють більшою виявляє здатність, ніж тести, відповідні коректним вхідним даним.
5) Необхідно перевіряти не тільки, робить програма то, для чого вона призначена, але і ні робить вона те, що не повинна робити.
6) Не слід викидати тести, навіть якщо програма вже не потрібна.

Контроль програми зводиться до того, щоб підібрати систему тестів, отримання правильних результатів для якої гарантувало б правильну роботу програми і для інших вихідних даних з області, зазначеної в розв'язуваної задачі.

Тестування системи проводилося двома методами:

- функціональне тестування;
- метод контрольних тестів (Test - випробування, перевірка).

Функціональне тестування, або тестування методом чорного ящика дозволяє отримати комбінації вхідних даних, що забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорний ящик», чия поведінка можна визначити тільки дослідженням його входів і відповідних виходів.

Тестування методом «чорного ящика» має на увазі, що тестувальник має доступ до ПЗ тільки через ті ж інтерфейси, що і замовник або користувач, або

через зовнішні інтерфейси, що дозволяють іншого комп'ютера або іншому процесу підключитися до системи для тестування.

Для реалізації методу контрольних тестів повинні бути виготовлені або заздалегідь відомі еталонні результати, на підставі звірки з якими одержуваних тестових результатів, можна було б зробити висновок про правильність роботи програми на даному тесті.

Еталонні результати для обчислювальних завдань можна отримати, здійснюючи обчислення вручну, застосовуючи результати, отримані раніше на іншому комп'ютері або за іншою програмою, або, використовуючи відомі факти, властивості, фізичні закони або системи автоматизованого тестування і налагодження.

При невпорядкованому тестуванні вихідні дані, що імітують зовнішнє середовище, випадковим чином генеруються у всьому діапазоні можливої зміни параметрів. При цьому багато значення вихідних даних характеризуються малою імовірністю виявлення помилок і не виправдовують витрати на виконання тестування. Крім того, можлива поява логічно суперечливих даних. У той же час дані, найбільш важливі з позиції реального використання програм і можливостей виявлення помилок, можуть виявитися неохопленими в процесі тестування.

5.2 Функціональне тестування системи

В інформаційні сервісах є великий функціонал, але найчастіші сценарії користування наведенні на рис. 27. В системі є дві ролі користувач і адміністратор. Адміністратор може надавати квоти на розмір сховищ користувачів, та назначати групу до користувача, визначивши права, які доступні користувачеві і виконувати всі дії, що доступні користувачеві. Користувач може створювати, редагувати і видаляти папки, завантажувати документи, надавати права доступу, і для конкретного користувача, і для існуючої групи користувачів.

Рисунок 27 - Діаграма варіантів використання системи

Для перевірки правильності роботи інформаційних сервісів виконаємо тести роботи користувача по найчастішим сценаріях використання:

- Створення папки. Папки створюються просто натиснувши на кнопку “+” (рис. 28).

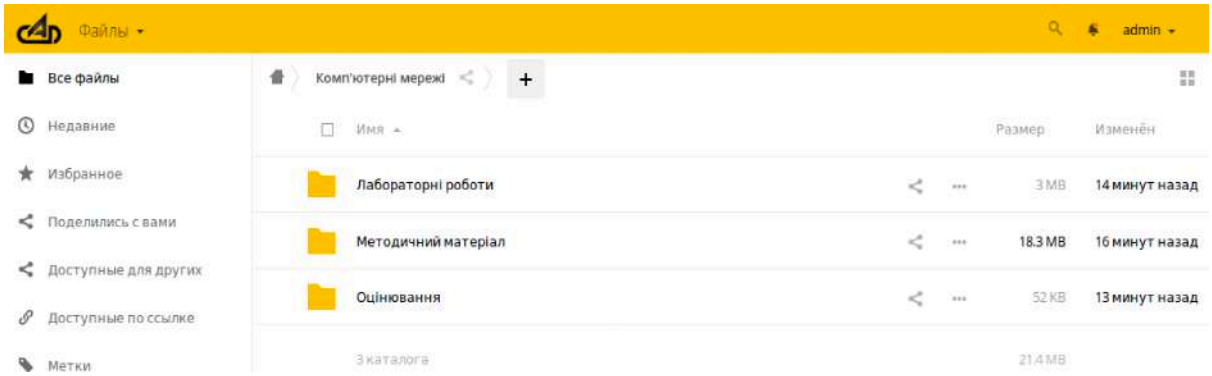

Рисунок 28 – Приклад створеної теки

- Завантаження файлу. Файл можливо завантажити чи через кнопку “+” чи перетягуванням файлу у область веб-сторінки. При виводі детальної інформації про файл ми бачимо маленьке зображення документу, і інформацію про нього. Також в цьому пункті можна передивитися історії змін, залишити та прочитати коментарі та управляти правами доступу до файлу іншим користувачам (рис. 29).

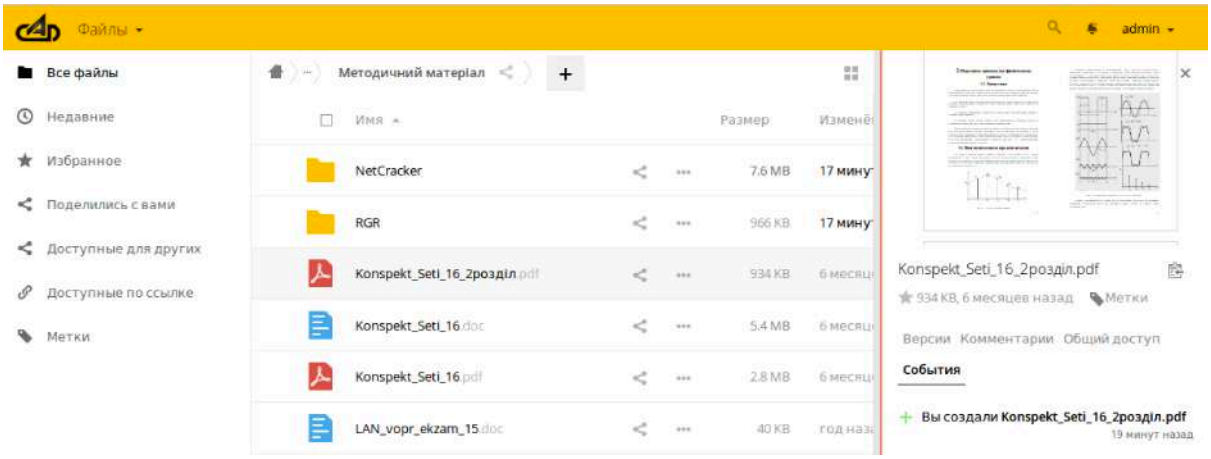

Рисунок 29 – Приклад виводу інформації про файл

- Перегляд зображень. Зображення відкривається на весь екран (рис. 30).

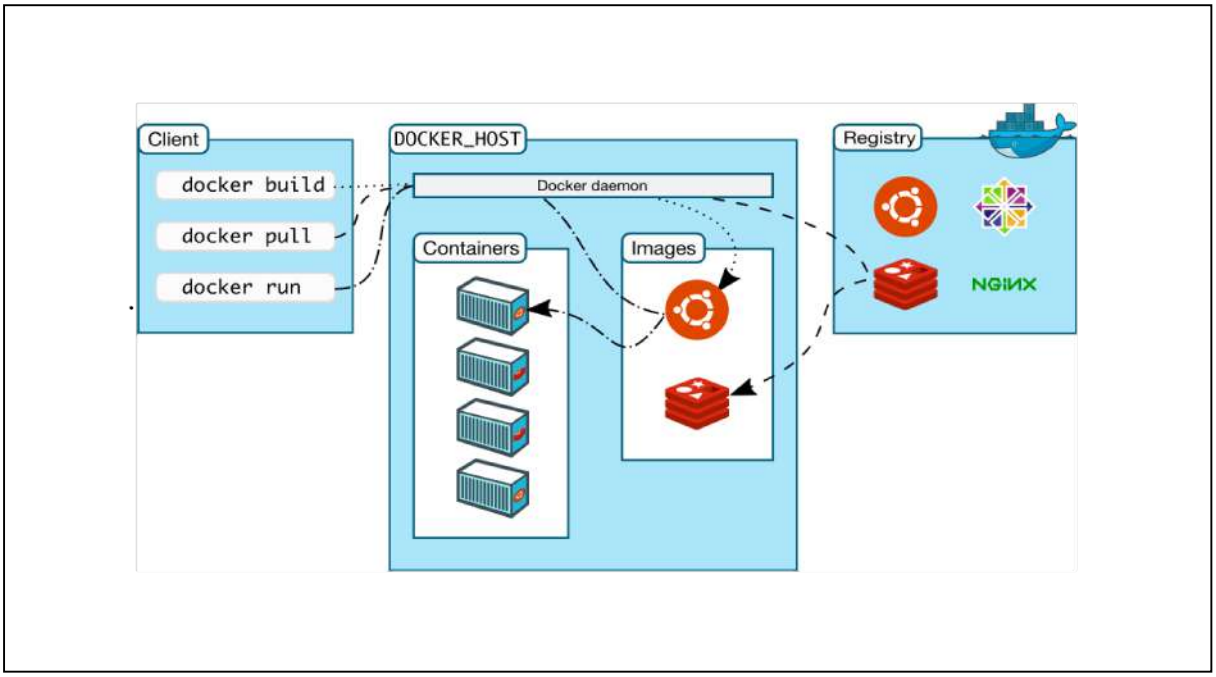

Рисунок 30 – Приклад відкриття зображення

- Відкриття PDF документу (рис. 31). Файли pdf викриваються в стандартному перегляді, але його функціонал більше ніж в Google Drive. Файл відкривається в останньому місці, де він був відкритий, також в документі можливо створювати закладки та швидко переходити в початок чи кінець документу. Приємним бонусом буде те, що можливо робити поворот окремих сторінок за і проти годинникової стрілки на 90 градусів.

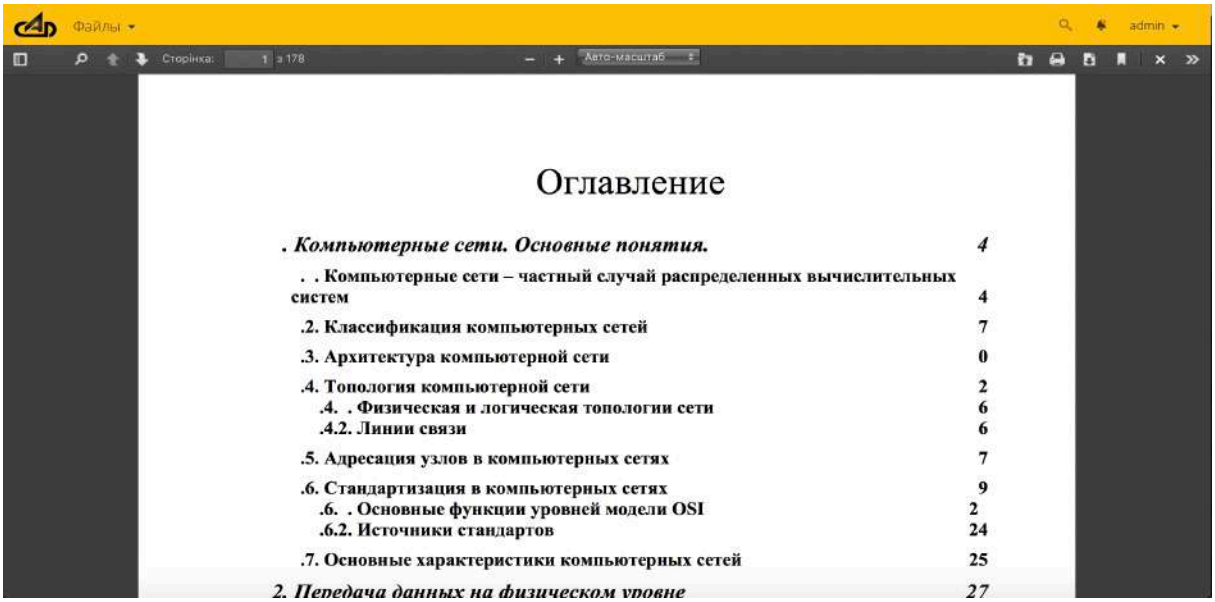

Рисунок 31 – Приклад роботи переглядача PDF файлів

- Створення нового документу. Новий документ створюється так само як і нова тека, по натисканню на кнопку “+” і введення назви документу. На рис. 32 продемонстровано вікно, яке відкривається після створення нової презентації.

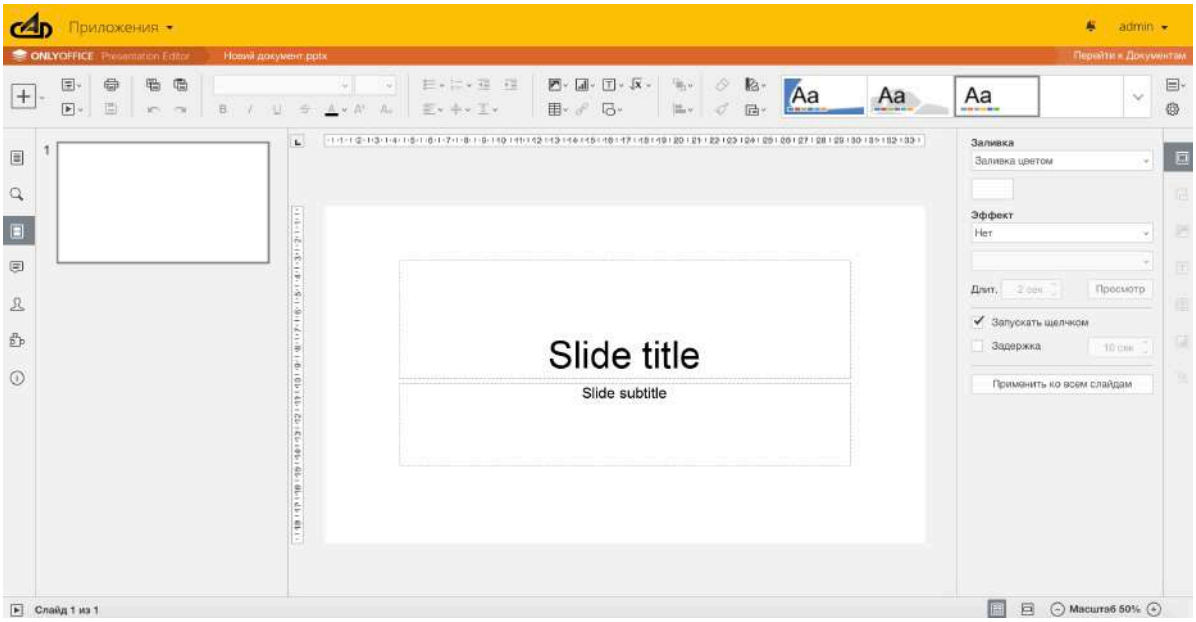

Рисунок 32 – Вікно OnlyOffice Presentation

- Перегляд та редагування документів docx та xlsx (рис. 34). Для тесту було відкрито документ Word з методичним матеріал з комп’ютерних

мереж (рис. 33) та документ Excel з оцінками та відвідуванням по курсу комп’ютерні мережі (рис. 34), та перевірено їх відкриття.

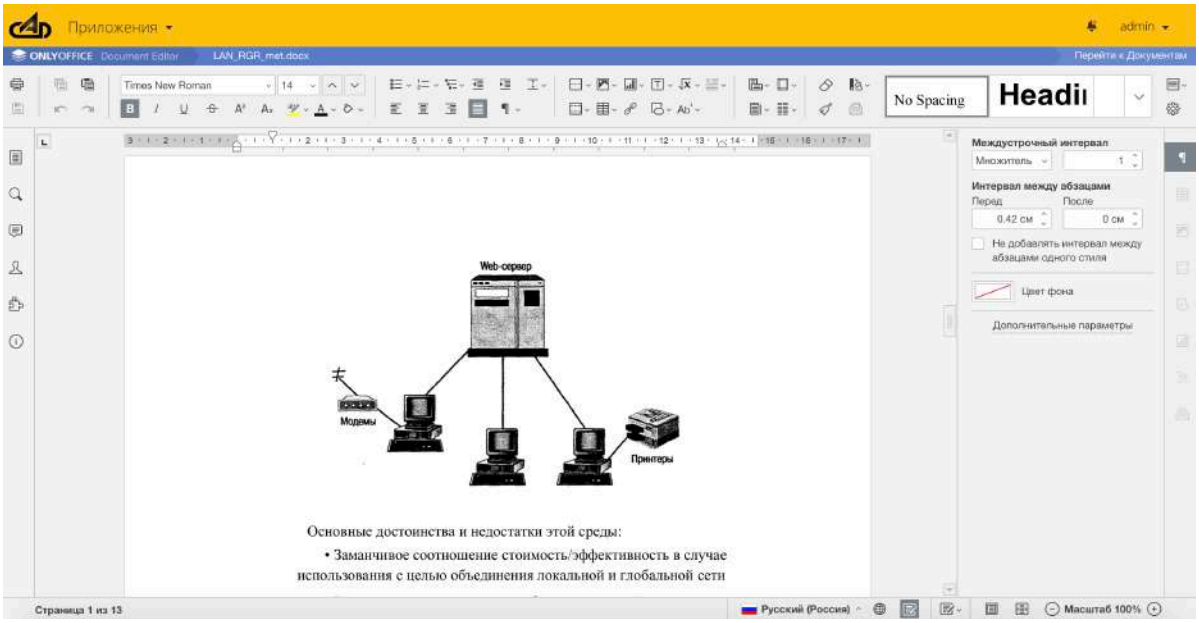

Рисунок 33 – Відкриття документу Word

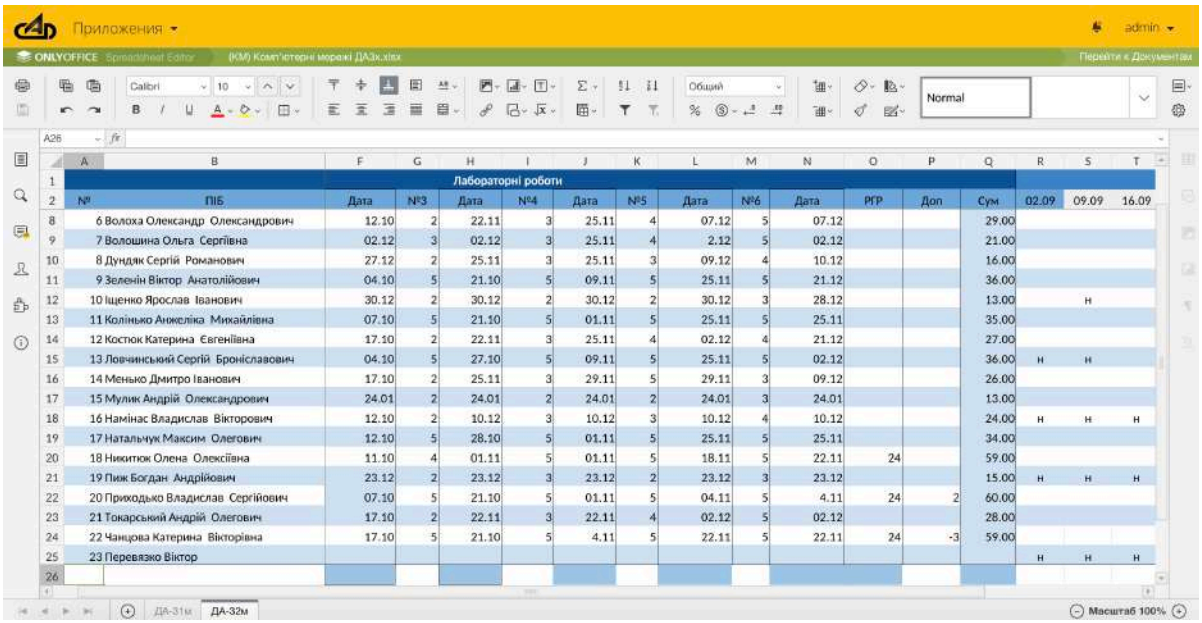

Рисунок 34 – Відкриття документу Excel

- Надання прав на перегляд файлів (рис. 35). Для того щоб надати права на перегляд файлів, то на вкладці “Общий доступ” потрібно прописати в поле імена користувачів чи назви груп, для яких даний доступ назначений. Біля документу появиться надпис “Общий доступ”, який

означає, що до документу є спільний доступ. А документ з’явиться у користувачів, для яких він відкритий.

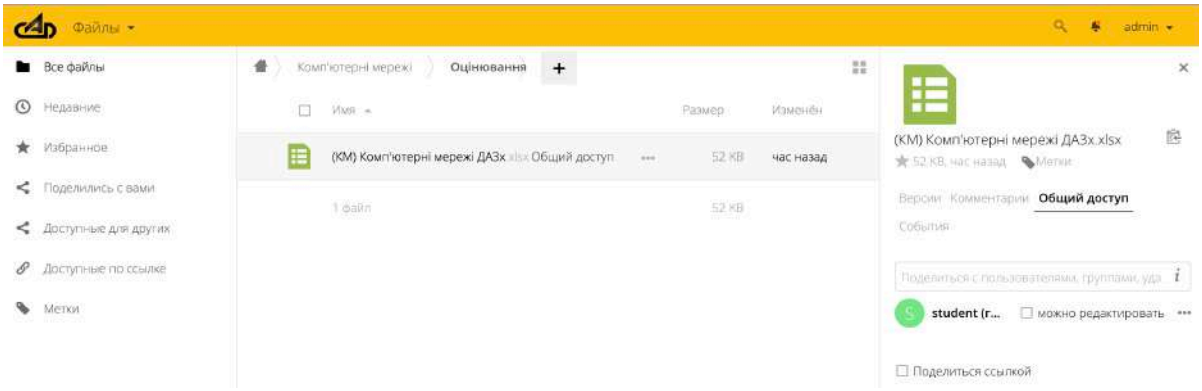

Рисунок 35 – Вікно надання спільного доступу.

Якщо документ відкритий тільки на читання, то вікно перегляду змінить свій вигляд і стане, як на рис. 36.

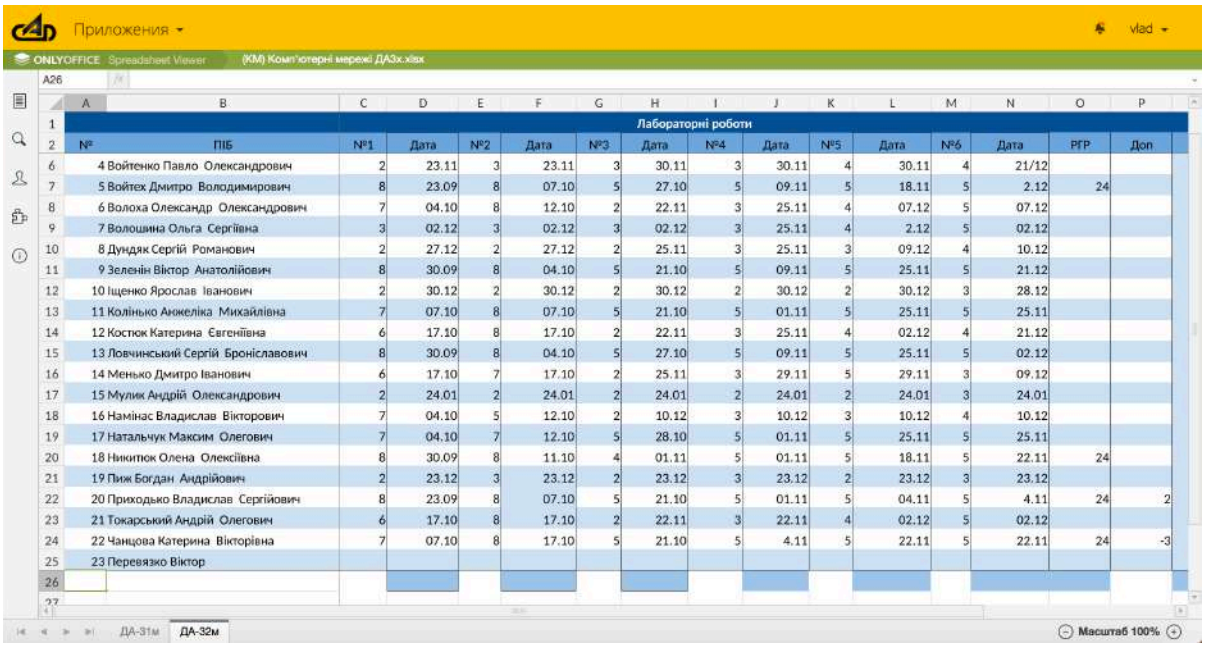

Рисунок 36 – Вікно перегляду документу в режимі читання

- Надання прав на редагування файлів та тек (рис. 37). Налаштування прав спільного доступу є дуже гнучким, і підтримує такі можливості: редагування, створення, зміну, видалення і права на дальше поширення доступу.

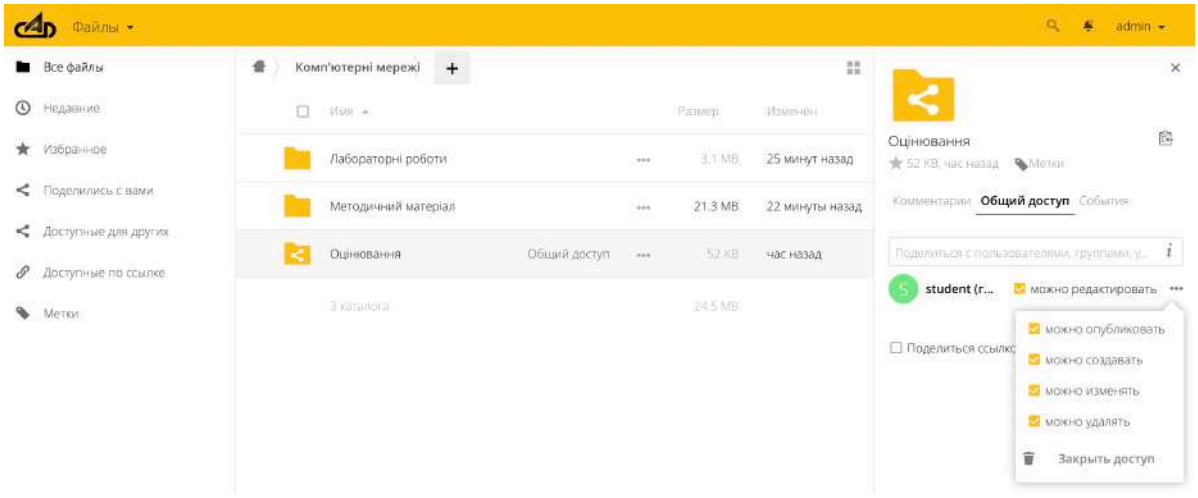

Рисунок 37 – Доступні права при спільному доступі

- Інформаційний сервіс підтримує одночасну роботу над документами декількох користувачів. Всі зміни відображаються у реальному часі та підписуються іменем користувача, який зробив ці зміни.
- Надання прав на доступ до файлів і тек, через посилання (рис. 38). При доступі через посилання відкривається для неавторизованих користувачів спрощення версія сховища (рис. 39) і доступний тільки онлайн перегляд зображень, для більшого захисту посилання можливо захистити паролем і вказати дату, до якого воно буде дійсне, також доступне налаштування посилання, тільки для завантаження файлів.

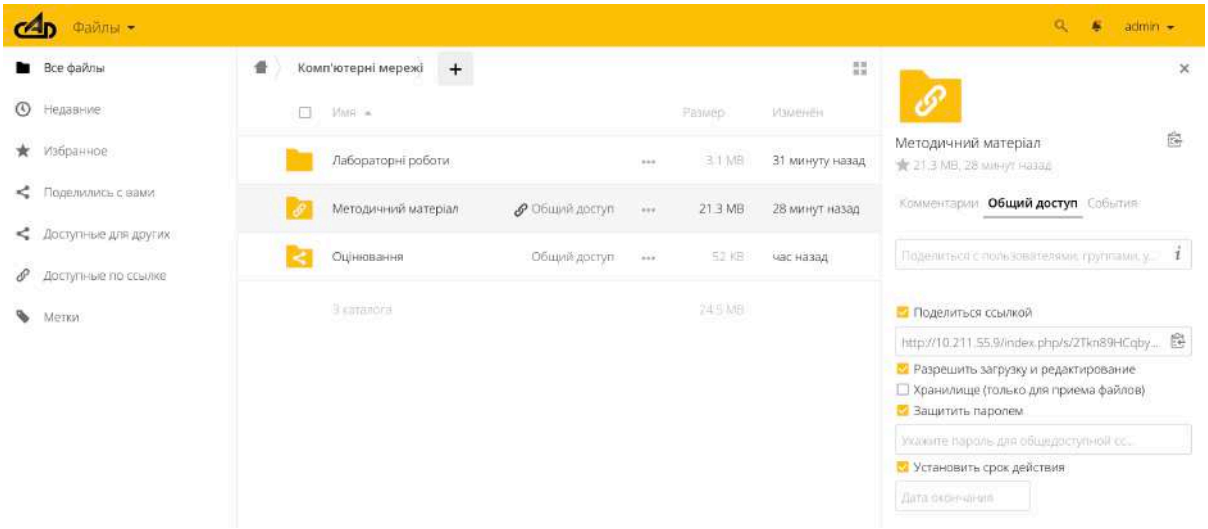

Рисунок 38 – Налаштування доступу за посиланням

Рисунок 39 – Вигляд сховища при переході за посиланням

5.3 Висновки

В цьому розділі наведено результати функціонального тестування інформаційний сервіс для підтримки навчального процесу. Тест був зроблений на сценаріях використання, які найчастіше будуть використовуватися на кафедрі СП. Паралельно з тестами давався опис функціоналу і приклади того, як змінюється вигляд вікон залежно від певних налаштувань. Всі тести інформаційний сервіс пройшов успішно.

6 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ СИСТЕМИ

У даному розділі проводиться оцінка основних характеристик системи, призначеного для створення інформаційних сервісів для забезпечення навчального процесу. Інформаційні сервіси будуть створенні і пряцюватимуть на платформі Docker.

Система представляє собою web-сторінку, яку можна відкрити через мобільний додаток або у браузерах, таких як Google Chrome, Mozilla Firefox, Opera та інші.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі

оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

6.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– система повинен функціонувати на персональних комп'ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об'ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

6.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – обрання складових інформаційної системи для підтримки навчального процесу з встановлення на фізичний сервер 26 корпусу

НТУУ «КПІ». Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір хмарного сховища даних;

F_2 – вибір оптимальної СКБД;

F_3 – вибір хмарного офісу.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) хмарне сховище Pydio;

б) хмарне сховище Nextcloud;

Функція F_2:

а) база даних MySQL;

б) база даних PostgreSQL.

Функція F_3:

а) хмарний офіс Onlyoffice;

б) хмарний офіс Google Docs.

6.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 40). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 3).

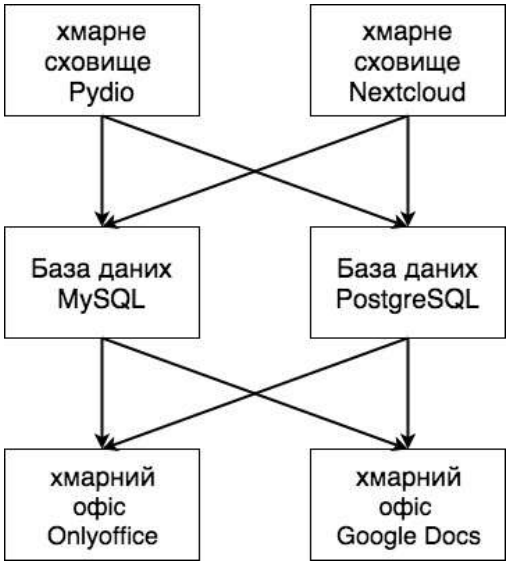

Рисунок 40 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 3 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Відкритий програмний код, велика кількість функціоналу доступна після встановлення, код на безпеку перевіряється як мінімум двома людьми, швидке збільшення функціоналу.	В деяких місцях складний інтерфейс, відсутнє меню для правої кнопки миші .
	Б	Відкритий програмний код, швидко дійний, інтерфейс подібний до класичного файлового менеджеру код на безпеку перевірений аудиторською компанією.	Частина потрібного фунціоналу знаходиться у платній версії, невелика кількість додатків.
F2	А	Надійність, внесення змін без перезапуску, безкоштовність, просте адміністрування, підтримка великої кількості стовбців в таблиці, підтримка мовою PHP	Необхідність додаткової інсталяції, низький рівень користувацької підтримки, повільний
	Б	Безкоштовність, наявність доброї документації, підтримка різних форматів SQL, підтримка JSON формату, немає обмеження на максимальну довжину рядку даних	Відсутність вкладених запитів, велика кількість перевірок даних.

Таблиця 3 (закінчення)

1	2	3	4
F3	*А*	Відкритий програмний код, встановлюється на власному сервері, гнучкий налаштуванні, широкий функціонал	Інтерфейс англійською мовою, доповнення функціоналу не йде швидкими темпами.
	Б	Безкоштовність, частий вихід нових функцій, велика стійкість до навантаження, надійна безпека.	Зміна API чи правил використання, залежність від інтернет каналу, документи відкривають не точно.

На основі аналізу позитивно-негативної матриці робимо висновок, що при обрані програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки хмарне сховище, потрібне з достатнім функціоналом, гарної підтримкою, незалежність від компаній та потребуватиме мінімум затрат для встановлення, то обираємо варіант Б.

Функція *F2:*

Вибір СКБД не відіграє велику роль у даному продукту, тому що відсутні великі дані, які будуть зберігатися у БД, тому вважаємо варіанти А і Б гідними розгляду.

Функція *F3:*

Оскільки хмарний офіс, повинен максимально точно відкривати документи та мати можливість для додавання нових шрифтів чи модифікування коду, тому обираємо варіант А.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2а – F3а
2. F1б – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

6.2 Обґрунтування системи параметрів ПП

6.2.1 Опис параметрів

На підставі даних про основні функції, що повинен мати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – час відповіді на запит клієнта;

– *X2* – об’єм даних, що передається по мережі;

– *X3* – об’єм оперативної пам’яті, що використовується;

– *X4* – кількість функцій, доступних для користувачів.

X1: Відображає час, що необхідний серверу для обробки запиту надісланого клієнтом до серверу, створення і відправлення відповіді на запит користувача.

X2: Відображає загальний розмір даних, що передаються клієнтові через мережу Інтернет та розмір даних, які переданні по внутрішній мережі кафедри СП.

X3: Відображає кількість оперативної пам’яті, яка необхідно буде для стабільної роботи та швидкої обробки запитів клієнта надісланих на інформаційних сервісів.

X4: Відображає кількість функцій сервісу, що будуть доступні користувачеві.

6.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію інформаційної системи як показано у табл. 4.

Таблиця 4 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Час відповіді на запит клієнта	X1	с	10	4	2
Об’єм даних, що передається по мережі	X2	МБ	200	140	80
Об’єм оперативної пам’яті, що використовується	X3	ГБ	22	14	10
Кількість функцій, доступних для користувачів	X4	Шт.	10	16	22

За даними таблиці 4 будуються графічні характеристики параметрів – рис. 41 – рис. 44.

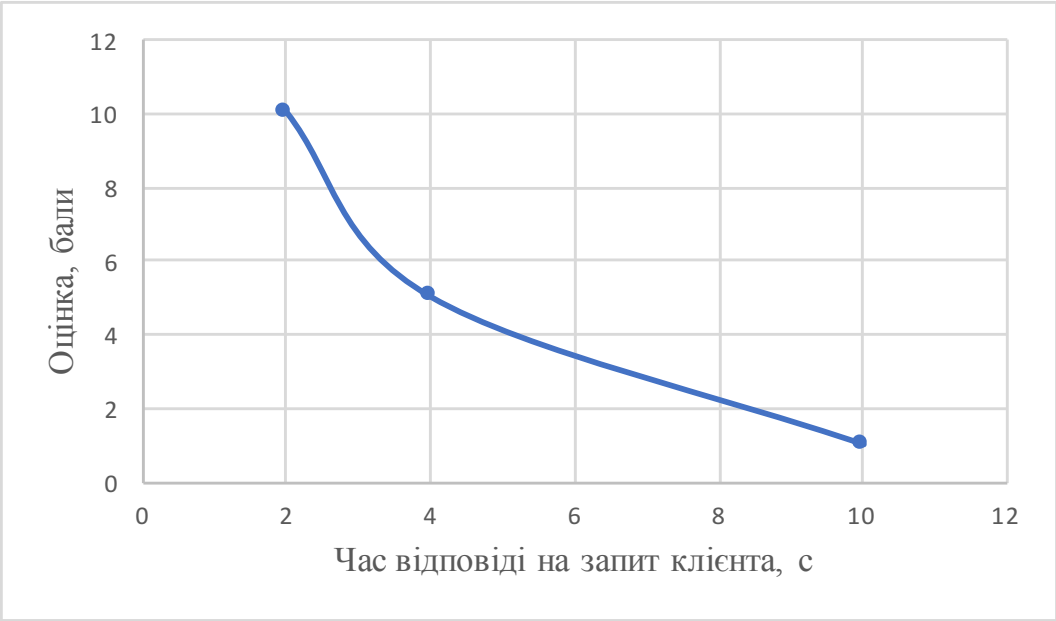

Рисунок 41 – X1, час відповіді на запит клієнта

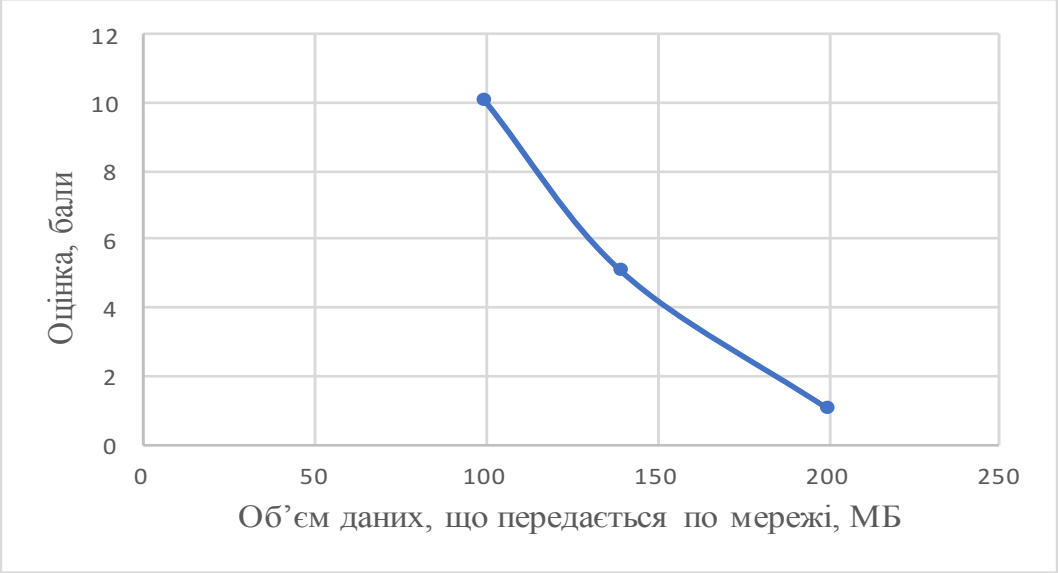

Рисунок 43 – X2, об’єм даних, що передається по мережі

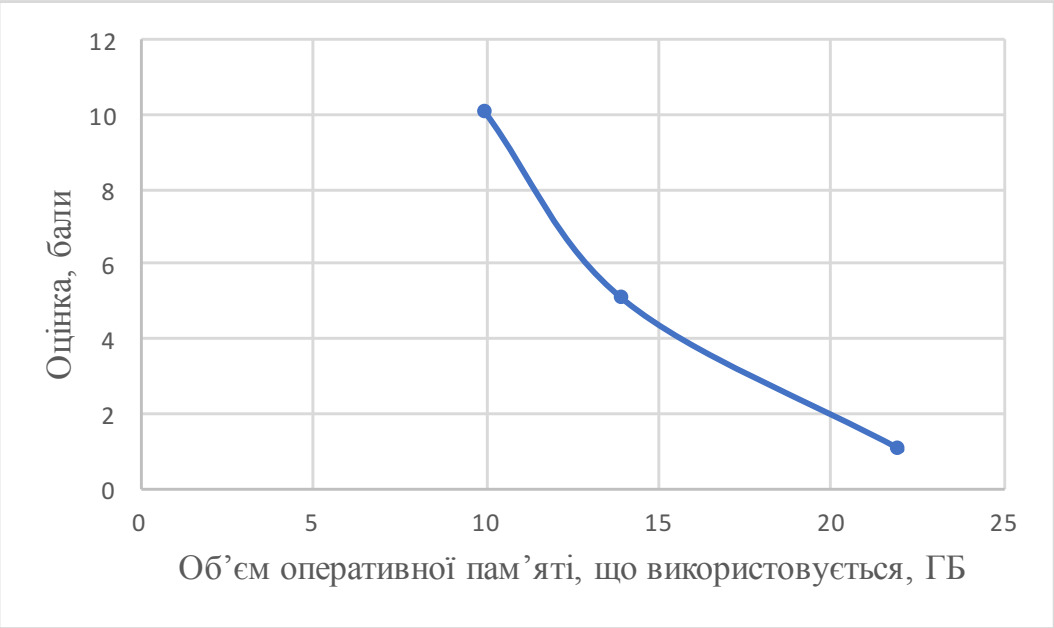

Рисунок 42 - X3, об’єм оперативної пам’яті, що використовується

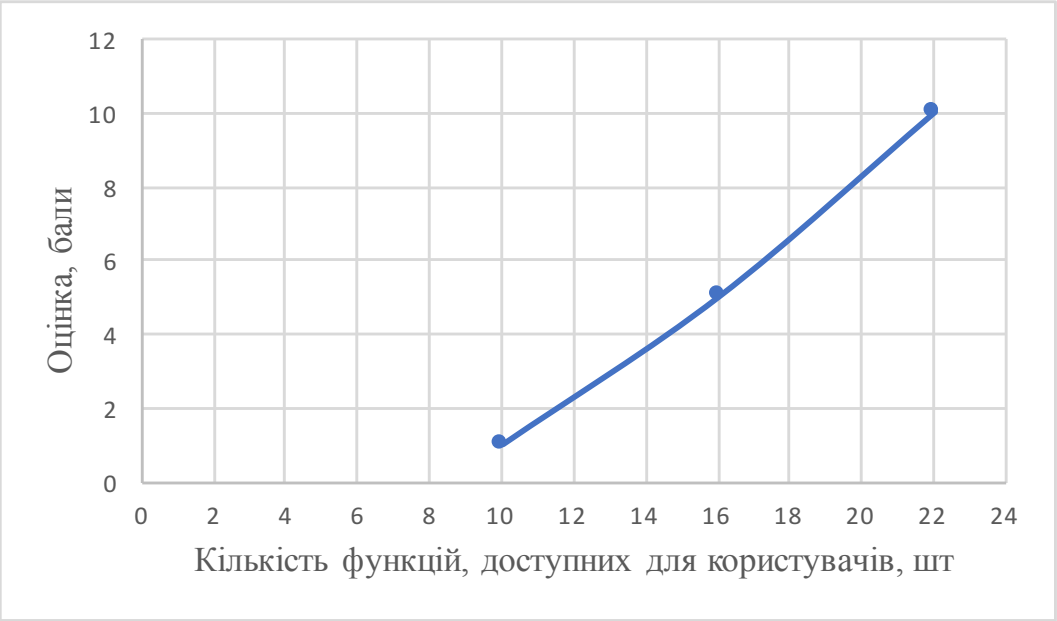

Рисунок 44 – X4, кількість функцій, доступних для користувачів

6.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – вибір програмного продукту, який має найбільш зручний та функціональний набір можливостей при низький необхідності у ресурсах.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.

Таблиця 5 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Час відповіді на запит клієнта	с	1	2	1	2	1	2	2	11	-6,5	42,25
X2	Об’єм даних, що передається по мережі	МБ	2	1	2	1	2	1	1	10	-7,5	56,25
X3	Об’єм оперативної пам’яті, що використовується	ГБ	3	3	4	3	4	4	3	24	6,5	42,25
X4	Кількість функцій, доступних для користувачів	шт.	4	4	3	4	3	3	4	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_i = 17{,}5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i\text{-}T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3\text{-}n)} = \frac{12 \cdot 201}{7^2(4^3\text{-}4)} = 0{,}8 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 6.

Таблиця 6 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	<	>	<	>	<	<	<	0,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	<	>	<	<	>	>	1,5

Числове значення, що визначає ступінь переваги *і*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{ де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i^{'}}{\sum_{i=1}^{n} b_i^{'}}, \text{де } b_i^{'} = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 7, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 7 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1,0	0,5	1,5	1,5	4,5	0,281	20,25	0,293	91,125	0,288
X2	1,5	1,0	1,5	1,5	5,5	0,344	30,25	0,438	166,375	0,527
X3	0,5	0,5	1,0	1,5	3,5	0,219	12,25	0,178	42,875	0,136
X4	0,5	0,5	0,5	1,0	2,5	0,156	6,25	0,091	15,625	0,049
Всього:					16	1	69	1	316	1

6.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X1(час відповіді на запит клієнта) та X4 (Кількість функцій, доступних для користувачів) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3(об'єм оперативної пам'яті, що використовується) буде найкращим у випадку обрання у F2 варіанта Б і становитиме 14, для варіанту А це значення буде 16.

Абсолютне значення параметра X2(об'єм даних, що передається по мережі) буде найкраще при обрані варіанту А 140, а при обрані варіанту Б 150.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 8):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j},$$

де n – кількість параметрів; $K_{вi}$ – коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 8 – Розрахунок показників рівня якості варіантів реалізації ПП

Основні функції	Варіант реалізації функції	Параметр	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	4	4,8	0,288	1,38
F2	А	X2	140	5,2	0,527	2,74
		X3	16	3,9	0,049	0,19
	Б	X2	150	4,3	0,527	2,27
		X3	14	4,2	0,049	0,21
F3	А	X4	16	4,9	0,136	0,67

За даними з таблиці 8 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 1,38 + 2,74 + 0,19 + 0,67 = 4,98

K_{K2} = 1,38 + 2,27 + 0,21 +0,67 = 4,53

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

6.4 Економічний аналіз варіантів розробки ПП

Програмний продукт, що буде встановлюватись на сервер кафедри вже розроблено, проте необхідно увесь рік тримати під контролем сервер, на якому буде встановлено програмне забезпечення. Отже

T_1 = 365 людино-днів на рік.

T_I = 365 · 8 = 2920 людино-годин;

Обидва варіанти однаково трудоємні

В адмініструванні беруть участь два адміністратори з окладом 7500 грн. Визначимо зарплату за годину за формулою:

$$\text{С}_\text{ч} = \frac{\text{М}}{T_m \cdot t} \text{ грн.},$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$\text{С}_\text{ч} = \frac{7500 + 7500}{2 \cdot 21 \cdot 8} = 44{,}64 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$\text{С}_\text{ЗП} = \text{С}_\text{ч} \cdot \text{Т}_i \cdot \text{К}_\text{Д},$$

де $\text{С}_\text{ч}$– величина погодинної оплати праці програміста; Т_i – трудомісткість відповідного завдання; $\text{К}_\text{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$С_{ЗП}$ = 44,64 * 2920 · 1,2 = 156418,56 грн на рік.

Відрахування на соціальний внесок незалежно від ризику становить 22%:

І. $С_{ВІД} = С_{ЗП} · 0.22$ =156418,56 * 0.22 = 34412,08 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Працюватиме одна електрона обчислювальна машина, котра працюватиме цілодобово:

$С_Г = 12·М·К_З$ = 12 · 7500· 0,2 = 18000 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г · (1+ К_З)$ = 12000· (1 + 0.2) = 21600 грн.

Відрахування на соціальний внесок становить 22%:

$С_{ВІД} = С_{ЗП} · 0.22$ = 14400 * 0,22 = 4752 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 29000 грн.

$С_А = К_{ТМ} · К_А · Ц_{ПР}$ = 1.15 · 0.25 · 29000 = 8337,5 грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} · Ц_{ПР} · К_Р$ = 1.15 · 29000 · 0.05 = 1667,5 грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = Д_К · t_З · К_В$ = 365 · 24 · 1 = 8760 годин,

де $Д_К$ – календарна кількість днів у році; t –кількість робочих годин електронної обчислювальної машини в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} · N_С · К_З · Ц_{ЕН}$ =8760 · 0,75 · 1 · 1,94 = 12745,8 грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 29000 \cdot 0,67 = 19430$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 21600 + 4752 + 8337,5 + 1667,5 + 12745,8 + 19430 = 68532,8$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 68532,8 / 8760 = 7,82$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

$С_М = 7,82 * 2920 = 22844,27$

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

$С_Н = 156418,56 * 0,67 = 104800,44$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

$С_{ПП} = 156418,56 + 34412,08 + 22844,27 + 104800,44 = 318475,35$ грн.;

6.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1} = 4,98 / 318475,35 = 1,56 \cdot 10^{-5}$;

$К_{ТЕР2} = 4,53 / 318475,35 = 1,42 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1,56 \cdot 10^{-5}$.

6.6 Висновки

В даному розділі було проведено функціонально-вартісний аналіз системи інформаційних сервісів, яке буде розроблено в рамках дипломного проекту і

буде встановлено на сервер кафедри. Процес аналіз складався можна розділити на дві частини.

У першій проведено дослідження системи з технічної точки зору: були визначено основні параметри, що повинні бути головними при обранні кращої реалізації. На основі обчислених значень параметрів, а також оцінок експертів було обчислено коефіцієнт технічного рівня, який у другій частині допоміг визначити найкращий варіант з техніко-економічної точки зору.

У другій частині виконувалося економічне обґрунтування альтернативних варіантів реалізації. Порівняння обраних варіантів реалізації робилося з урахуванням витрат на заробітні плати, електроенергії, накладні витрати та трудомісткістю завдання.

Після виконання функціонально-вартісного аналізу програмного комплексу, що буде встановлюватися на сервер кафедри, що залишилися після проведення першої частини аналізу було виявлено, що перший варіант є найбільш оптимальним для реалізації. Його показник техніко-економічного рівня якості $К_{ТЕР1} = 1,56 \cdot 10^{-5}$;

Цей варіант реалізації програмного продукту має такі параметри:

– хмарне сховище Nextcloud;

– база даних MySQL;

– хмарний офіс Onlyoffice.

Даний варіант реалізації є найбільш продуктивним, за рахунок використання сучасних складових з дизайном, який використовується роками. Набір функціоналу інформаційних сервісів для підтримки навчального процесу відповідає вимогам виставлених кафедрою.

ВИСНОВКИ

В ході дипломної роботи було проаналізовано, розроблено, налаштовано і протестовано інформаційні сервіси для підтримки навчального процесу.

В першому розділі було розглянуто види віртуалізації та принципи організації платформи Docker. Перевагами використання контейнерів є найбільш раціональне використовуються ресурсів серверів, що дає можливість запускати більшу кількість додатків, чим при використанні інших видів віртуалізації серверів. Контейнери допомагають абстрагуватися від хост-системи і розробляти логіку додатків, додаючи можливість швидкого горизонтального масштабування додатку.

У другому розділі було сформовано вимоги до системи, які висувають користувачі та системні адміністратори. Всі вимоги, згідно їх характеру, було розділено на чотири групи: функціональні, користувачки, системні та зовнішніх інтерфейсів.

У третьому розділі було виконане порівняння і вибір складових інформаційних сервісів. При порівняні хмарних сховищ найкращим виявився продукт Nextcloud, який надає користувачу найбільший функціонал при відкритому коді. Сховище Pydio є прямим конкурентом Nextcloud, але частина його функціоналу є платною, тому при виборі безкоштовного сховища, вибір залишається за Nextcloud. Сховище OwnCloud, яке раніше було лідером ринку поступово втрачає свої позиції через суперечки у команді і переходу частини працівників до Nextcloud, тому з часом дана система зникне з ринку. Сховище Seafile сховище може конкурувати з Nextcloud, але воно відносно нове та не популярне.

Зробивши порівняння хмарних офісів, можна виділити двох лідерів Google Docs і OnlyOffice. Кожне рішення підходить для певної реалізації системи. Так як будується повністю автономна система, то було обрано OnlyOffice. Google Docs найкраще підходить для змішаного типу систем, коли окремі сервіси

працюють на сторонні провайдера. Офіс Microsoft Office Online дещо відстає від лідерів ринку, але видно, але в компанії докладають великих зусиль для його розвитку, так що в найближчому майбутньому все ще може змінитися. А хмарний офіс Zoho Office є простим редактором, хоч і з найкращою якістю, яка була відточена роками.

В четвертому розділі було розглянуто ряд типових архітектур розподілених системи в залежності від їх навантаження і запропоновано архітектуру системи інформаційних сервісів, яка відповідає потребам забезпечення навчального процесу на кафедрі СП. Також було розглянуто варіанти підключення системи до кафедрального сайту. Був вибраний варіант реалізації через перенаправлення, оскільки він є найбільш простим в реалізації і потребує мінімальної зміни кафедрального сайту. В розділі описана архітектура системи інформаційних сервісів з протоколами зв'язку, віртуальними мережами та взаємодією з мережею хоста.

Матеріал четвертого розділу є досить детальною інструкцією зі всіма кроками, командами та полями для заповнення, які можуть бути корисним при встановленні і налаштуванні інформаційних сервісів для підтримки навчального процесу на платформі Docker.

У п'ятому розділі було проведено функціональне тестування інформаційних сервісів з виконанням типових дій користувача. Всі тести було пройшли успішно.

В майбутньому дану роботу можна розвивати в напрямку додавання нових інформаційних сервісів. Також можливо розширити сфери використання хмарного офісу, а саме для автоматичного створення документів-звітів чи його використання для конвертації документів LaTeX. Хмарне сховище можна розвивати у напрямку створення єдиного розподіленого сховища на території НТУУ "КПІ ім. Ігоря Сікорського" чи з розширенням функціоналу використовувати для електронного документу обігу на кафедрі. Також можливий розвиток у напрямку створення модулю для веб-сторінки кафедри,

який наддасть можливість єдиного входу до інформаційних сервісів кафедри, які будуть розширятися.

На даний момент хмарні технології знаходяться на піки популярності і велика кількість сервісів переходять у хмари, тому розвиваючи даний напрямок можливо не відставати від тенденцій світу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шишкін В.М. Безпека хмарних обчислень – проблеми та можливості ризик-аналізу [Текст] / В.М. Шишкін// Міжнародна наукова конференція “Автоматизовані системи управління та сучасні інформаційні технології”: тези доп. – Tbilisi: Publication House “Technical University”, 2011. – С. 142.
2. Технологии виртуализации [Электронный ресурс] – Режим доступу: http://www.intuit.ru/studies/courses/673/529/lecture/11915. – Дата доступу: 24.05.2017.
3. Виртуализация: определение и виды [Электронный ресурс] – Режим доступу: https://habrahabr.ru/sandbox/40157/ – Дата доступу: 24.05.2017.
4. Знакомимся с основными возможностями Docker [Электронный ресурс] – Режим доступу до ресурсу: https://xakep.ru/2015/06/01/docker-usage/ – Дата доступу: 24.05.2017.
5. Офіційний сайт документації Docker [Електронний ресурс] – Режим доступа: https://docs.docker.com/ – Дата доступа: 24.05.2017.
6. Введение в Docker [Электронный ресурс] – Режим доступу: http://docker.cool/docs/docker-engine/docker-overview/ – Дата доступа: 24.05.2017.
7. Виртуализация процесса разработки [Электронный ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/docker/ – Дата доступа: 24.05.2017.
8. Офіційний сайт OwnCloud [Електронний ресурс] – Режим доступа: https://owncloud.com – Дата доступа: 25.05.2017.
9. Поднимаем сервис для хранения и синхронизации конфиденциальных данных [Электронный ресурс] – Режим доступу до ресурсу: https://xakep.ru/2014/09/24/cloud-crime/ – Дата доступа: 25.05.2017.
10. Офіційний сайт Nextcloud [Електронний ресурс] – Режим доступа: https://nextcloud.com – Дата доступа: 25.05.2017.

11. Офіційний сайт Pydio [Електронний ресурс] – Режим доступа: https://pydio.com – Дата доступа: 25.05.2017.
12. Офіційний сайт Seafile [Електронний ресурс] – Режим доступа: https://www.seafile.com/en/home/ – Дата доступа: 25.05.2017.
13. Какой офис выбрать? Облачный! [Электронный ресурс] – Режим доступу до ресурсу: http://itc.ua/articles/kakoy-ofis-vyibrat-oblachnyiy/ – Дата доступа: 25.05.2017.
14. Офіційний сайт Google Docs [Електронний ресурс] – Режим доступа: https://docs.google.com – Дата доступа: 25.05.2017.
15. Офіційний сайт Microsoft Office Online [Електронний ресурс] – Режим доступа: https://products.office.com/uk-ua/office-online/ – Дата доступа: 25.05.2017.
16. Офіційний сайт Zoho Docs [Електронний ресурс] – Режим доступа: https://www.zoho.com/docs/ – Дата доступа: 25.05.2017.
17. Офіційний сайт OnlyOffice [Електронний ресурс] – Режим доступа: https://www.onlyoffice.com – Дата доступа: 25.05.2017.
18. Nextcloud Solution Architecture. Bring data back under control of IT. – Germany: Nextcloud GmbH, 2017. – 18 с.
19. Офіційний сайт документації Nextcloud [Електронний ресурс] – Режим доступа: https://docs.nextcloud.com – Дата доступа: 26.05.2017.

ДОДАТОК А

ЛІСТИНГ КОНФІГУРАЦІЇ СИСТЕМИ

docker-compose.yml

```
version: '2'
services:
  nextcloud:
    build: .
    container_name: nextcloud
    links:
      - nextcloud-mariadb
      - nextcloud-redis
    depends_on:
      - onlyoffice-documentserver
    restart: always
    networks:
      - nextcloud
      - onlyoffice
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./nextcloud/apps:/var/www/html/apps
      - ./nextcloud/data:/var/www/html/data
      - ./nextcloud/config:/var/www/html/config
      - /var/run/docker.sock:/tmp/docker.sock:ro

  nextcloud-mariadb:
    container_name: nextcloud-mariadb
    image: mariadb
    environment:
      - MYSQL_ROOT_PASSWORD=qwerty78
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_PASSWORD=qwerty67
    restart: always
    networks:
      - nextcloud
    volumes:
      - ./nextcloud/db:/var/lib/mysql
```

```
nextcloud-redis:
  container_name: nextcloud-redis
  image: redis
  restart: always
  networks:
    - nextcloud
  expose:
    - '6379'

onlyoffice-documentserver-data:
  container_name: onlyoffice-documentserver-data
  image: onlyoffice/documentserver:latest
  environment:
    - ONLYOFFICE_DATA_CONTAINER=true
    - POSTGRESQL_SERVER_HOST=onlyoffice-postgresql
    - POSTGRESQL_SERVER_PORT=5432
    - POSTGRESQL_SERVER_DB_NAME=onlyoffice
    - POSTGRESQL_SERVER_USER=onlyoffice
    - RABBITMQ_SERVER_URL=amqp://guest:guest@onlyoffice-rabbitmq
    - REDIS_SERVER_HOST=onlyoffice-redis
    - REDIS_SERVER_PORT=6379
  stdin_open: true
  restart: always
  networks:
    - onlyoffice
  volumes:
     - ./onlyoffice/data:/var/www/onlyoffice/Data
     - ./onlyoffice/log:/var/log/onlyoffice

onlyoffice-documentserver:
  container_name: onlyoffice-documentserver
  image: onlyoffice/documentserver:latest
  depends_on:
    - onlyoffice-documentserver-data
    - onlyoffice-postgresql
    - onlyoffice-redis
    - onlyoffice-rabbitmq
  environment:
    - ONLYOFFICE_DATA_CONTAINER_HOST=onlyoffice-documentserver-data
    - BALANCE=uri depth 3
    - EXCLUDE_PORTS=443
    - HTTP_CHECK=GET /healthcheck
    - EXTRA_SETTINGS=http-check expect string true
  stdin_open: true
```

```
    restart: always
    networks:
      - onlyoffice
      - nextcloud
    ports:
      - '81:80'
      - '444:443'
    volumes_from:
     - onlyoffice-documentserver-data

  onlyoffice-redis:
    container_name: onlyoffice-redis
    image: redis
    restart: always
    networks:
     - onlyoffice
    expose:
      - '6379'

  onlyoffice-rabbitmq:
    container_name: onlyoffice-rabbitmq
    image: rabbitmq
    restart: always
    networks:
     - onlyoffice
    expose:
      - '5672'

  onlyoffice-postgresql:
    container_name: onlyoffice-postgresql
    image: postgres:9.5
    environment:
      - POSTGRES_DB=onlyoffice
      - POSTGRES_USER=onlyoffice
    networks:
      - onlyoffice
    restart: always
    expose:
      - '5432'
    volumes:
      - .onlyoffice/db:/var/lib/postgresql

networks:
  nextcloud:
    driver: 'bridge'
```

```
  onlyoffice:
    driver: 'bridge'
```

Dockerfile

```
FROM nextcloud:latest
COPY ./onlyoffice /var/www/html/custom_apps/onlyoffice
COPY docker-entrypoint.sh /root/entrypoint.sh
RUN chmod +x /root/entrypoint.sh
ENTRYPOINT ["/root/entrypoint.sh"]
CMD ["apache2-foreground"]
```

entrypoint.sh

```
#!/bin/bash
set -e

# version_greater A B returns whether A > B
function version_greater() {
    [[ "$(printf '%s\n' "$@" | sort -V | head -n 1)" != "$1" ]];
}

installed_version="0.0.0~unknown"
if [ -f /var/www/html/version.php ]; then
    installed_version=$(php -r 'require "/var/www/html/version.php"; echo
"$OC_VersionString";')
fi
image_version=$(php -r 'require "/usr/src/nextcloud/version.php"; echo
"$OC_VersionString";')

if version_greater "$installed_version" "$image_version"; then
```

```
    echo "Can't start Nextcloud because the version of the data ($installed_version) is higher than the docker image version ($image_version) and downgrading is not supported. Are you sure you have pulled the newest image version?"
    exit 1
fi

if version_greater "$image_version" "$installed_version"; then
    if [ "$installed_version" != "0.0.0~unknown" ]; then
        su - www-data -s /bin/bash -c 'php /var/www/html/occ app:list' > /tmp/list_before
    fi

    rsync -a --delete --exclude /config/ --exclude /data/ --exclude /custom_apps/ /usr/src/nextcloud/ /var/www/html/

    if [ ! -d /var/www/html/config ]; then
        cp -arT /usr/src/nextcloud/config /var/www/html/config
    fi

    if [ ! -d /var/www/html/data ]; then
        cp -arT /usr/src/nextcloud/data /var/www/html/data
    fi

    if [ ! -d /var/www/html/custom_apps ]; then
        cp -arT /usr/src/nextcloud/custom_apps /var/www/html/custom_apps
        cp -a /usr/src/nextcloud/config/apps.config.php /var/www/html/config/apps.config.php
    fi

    if [ "$installed_version" != "0.0.0~unknown" ]; then
```

```
        su - www-data -s /bin/bash -c 'php /var/www/html/occ upgrade --no-app-disable'

        su - www-data -s /bin/bash -c 'php /var/www/html/occ app:list' > /tmp/list_after
        echo "The following apps have beed disabled:"
        diff <(sed -n "/Enabled:/,/Disabled:/p" /tmp/list_before) <(sed -n
"/Enabled:/,/Disabled:/p" /tmp/list_after) | grep '<' | cut -d- -f2 | cut -d: -f1
        rm -f /tmp/list_before /tmp/list_after
    fi
fi

cp -R /var/www/html/custom_apps/onlyoffice /var/www/html/apps/onlyoffice
chown -R www-data:www-data /var/www/html
rm -rf /var/www/html/custom_apps/onlyoffice

exec "$@"
```