МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«Розробка сервісу збереження файлів з використання технологій Amazon Web Services»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1218

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

С.О. Вдовенко
(ініціали та прізвище)

Керівник доцент кафедри прогамної інженерії, к.ф.-м.н Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

Лісняк А.О.

(підпис) _____

_____

« __ » _____ 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Вдовенко Станіславу Олександровичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка сервісу збереження файлів з використання технологій Amazon Web Services

керівник роботи (проекту) _____

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи 1. Постановка задачі.

2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	17.07.2019	
2.	Збір вихідних даних.	23.07.2019	
3.	Обробка методичних та теоретичних	03.08.2019	
	джерел.		
4.	Розробка першого розділу.	05.08.2019	
5.	Розробка другого розділу.	17.08.2019	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	27.12.2019	
7.	Захист кваліфікаційної роботи.	15.01.2020 (заочне: 10.01.2020)	

Студент _____ _____
(підпис) (ініціали та прізвище)

Керівник роботи _____ _____
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка сервісу збереження файлів з використання технологій Amazon Web Services»: 44 с., 30 рис., 8 джерел.

БАНКІВСЬКІ КАРТИ, КОНТЕЙНЕР, ЗОБРАЖЕННЯ, СИСТЕМА ПРИЙОМУ И ОБРОБКИ ЕЛЕКТРОННИХ ПЛАТЕЖІВ, СХОВИЩЕ.

Об’єкт дослідження – сфера обробки та збереження користувацьких зображень до хмарних сховищ та баз даних, а також системи прийому и обробки електронних платежів для проведення фінансових операцій у всесвітній мережі інтернет.

Мета роботи: дослідити існуючі програмні рішення для обробки та збереження користувацьких зображень до хмарних сховищ та баз даних, а також системи прийому и обробки електронних платежів для проведення фінансових операцій у всесвітній мережі інтернет. Порівняти представлені рішення, а також розробити на їх основі додаток, що буде надавати доступ до їх функціонала, а також розширювати його новою бізнес логікою.

Метод дослідження – порівняти та аналіз представлені рішення для обробки та збереження користувацьких зображень до хмарних сховищ та баз даних, а також системи прийому и обробки електронних платежів.

У наш час дуже багато компаній та звичайних людей переходять до збереження своїх файлів у мережі інтернет за допомогою хмарних технології, що служать чудової альтернативою збереженню на своїх комп’ютерах, бо забезпечує безпеку даних, у разі поломки жорсткого диску чи іншого обладнання на вашу комп’ютері. Хмарні сховища, у свою чергу, надаюсь можливість зроби резервну копію усіх ваших файлів, а також надають доступ до них з будь якого пристрою, чи місця. Усе, що вам потрібно для доступу до ваших файлів, це наявність інтернету.

SUMMARY

Master’s Qualification Thesis «Development of the File Storage Service using Amazon Web Services»: 44 pages, 30 figures, 8 references.

SYSTEM OF ACCEPTANCE AND PROCESSING OF ELECTRONIC PAYMENTS, CONTAINER, STORAGE, IMAGE, BANK CARDS.

The object of study is the scope of processing and storing user images to cloud storage and databases, as well as the system for receiving and processing electronic payments for financial transactions on the Internet.

The aim of the study to explore existing software solutions for processing and storing custom images to cloud storage and databases, as well as systems for receiving and processing electronic payments for financial transactions on the Internet. Compare the solutions presented and develop an application that will give them access to their functionality, as well as expand it with new business logic.

The methods research are compare and analyze the solutions presented for processing and storing custom images to cloud storage and databases, as well as the system for receiving and processing electronic payments.

Nowadays, a lot of companies and ordinary people are moving to save their files on the Internet with cloud technology, which is a great alternative to saving on their computers because it provides data security in case of breakdown of hard disk or other equipment on your computer. to the user. Cloud storage, in turn, gives me the ability to back up all your files and access them from any device or place. All you need to access your files is Internet availability. Therefore, the development of such services is a promising area for business development. With the advent of internet banking, we can use a variety of electronic payment processing and processing systems in our applications that allow us to expand our functionality through financial investment.

ЗМІСТ

Завдання на кваліфікаційну роботу…… 2

Реферат…… 4

Summary…… 5

Вступ…… 7

1 Огляд технологій ……..…… 8

1.1 Node.js…….…….. 8

1.2 Веб-сервіси Amazon …… 12

1.1.2 Влаштування Amazon Web Services …….... 13

1.2.2 Платформні послуги……... 15

1.3 Порівняння сервісів для збереження файлів…… 15

1.3.1 Dropbox…….……..... 15

1.3.2 Gdrive……. 17

1.3.3 Порівняння…….… 17

1.4 Порівняння електронних систем прийому й обробки платежів 20

1.4.1 Stripe……..…….. 20

1.4.2 Amazon Pay.…….…….. 20

1.4.3 PayPal……..…….. 21

1.5 Аналіз вимог.....…….…. 23

2 ПРОЕКТУВАННЯ ДОДАТКУ …….……. 25

3 РОЗРОБКА ДОДАТКУ…… 29

3.1 Розробка сервісу завантаження зображень…… 29

3.2 Розробка сервісу прийому и обробки електронних платежів 36

Висновки…….. 43

Перелік посилань……....……... 44

ВСТУП

У наш час дуже багато компаній та звичайних людей переходять до збереження своїх файлів у мережі інтернет за допомогою хмарних технології, що служать чудової альтернативою збереженню на своїх комп'ютерах, бо забезпечує безпеку даних, у разі поломки жорсткого диску чи іншого обладнання на вашу комп'ютері. Хмарні сховища, надаюсь можливість зроби резервну копію усіх ваших файлів, а також надають доступ до них з будь якого пристрою, чи місця. Усе, що вам потрібно для доступу до ваших файлів, це наявність інтернету. Також з поширенням інтернет – банкінгу ми можемо використовувати у своїх додатках різноманітні системи прийому и обробки електронних платежів, що дозволяють розширювати функціонал за допомогою фінансових інвестицій.

Метою роботи є Розробка сервісу збереження файлів з використання технологій Amazon Web Services.

Для доставлення поставленої мети, сформульовано наступні задачі

1) дослідити сучасні методи роботи з зображення та зберігання їх у хмарних сховищах, а також системи прийому й обробки електронних платежів;
2) спроектувати додаток для збереження файлів;
3) розробити додаток для збереження файлів на основі технологій Amazon Web Services.

Об'єктом роботи є процес розробки додатку сервісу збереження файлів.

Предметом роботи є сервіс збереження файлів.

Структурно робота складається з трьох частин. У першій частині було розглянуто сучасні технології збереження зображень у хмарних сховищах, а також системи прийому й оброки електронних платежів. У другій частині було спроектовано додаток. У третій частині було розроблено додаток, а також описані функції які були створені у ходи виконання кваліфікаційної роботи.

1 ОГЛЯД ТЕХНОЛОГІЙ

Додаток було розроблено за допомогою мови програмування Node.js. Вибір було зроблено через швидкість платформи та велику кількість готових рішень для систем прийому и обробки електронних платежів. А саме сервісу для прийому и обробки електронних платежів Stripe. Також поєднання цих двох технологій дуже гарно описане та задокументовано у мережі інтернет. Що може значно пришвидшити роботу, а також спростити розробку та роботу над помилками і програмними помилками.

1.1 Node.JS

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників [9].

Node.js має наступні властивості:

- Асинхронна одно-нитева модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- JavaScript Google V8;

Для керування модулями використовується пакетний менеджер npm (node package manager).

JavaScript однопоточний, а асинхронний опис не є частиною самої мови; замість цього воно базується на основі його в браузері (або в середовищи

програмізації) і доступно через браузерні API. Теперь посмотрим на длинный ответ (див .рис. 1.1)

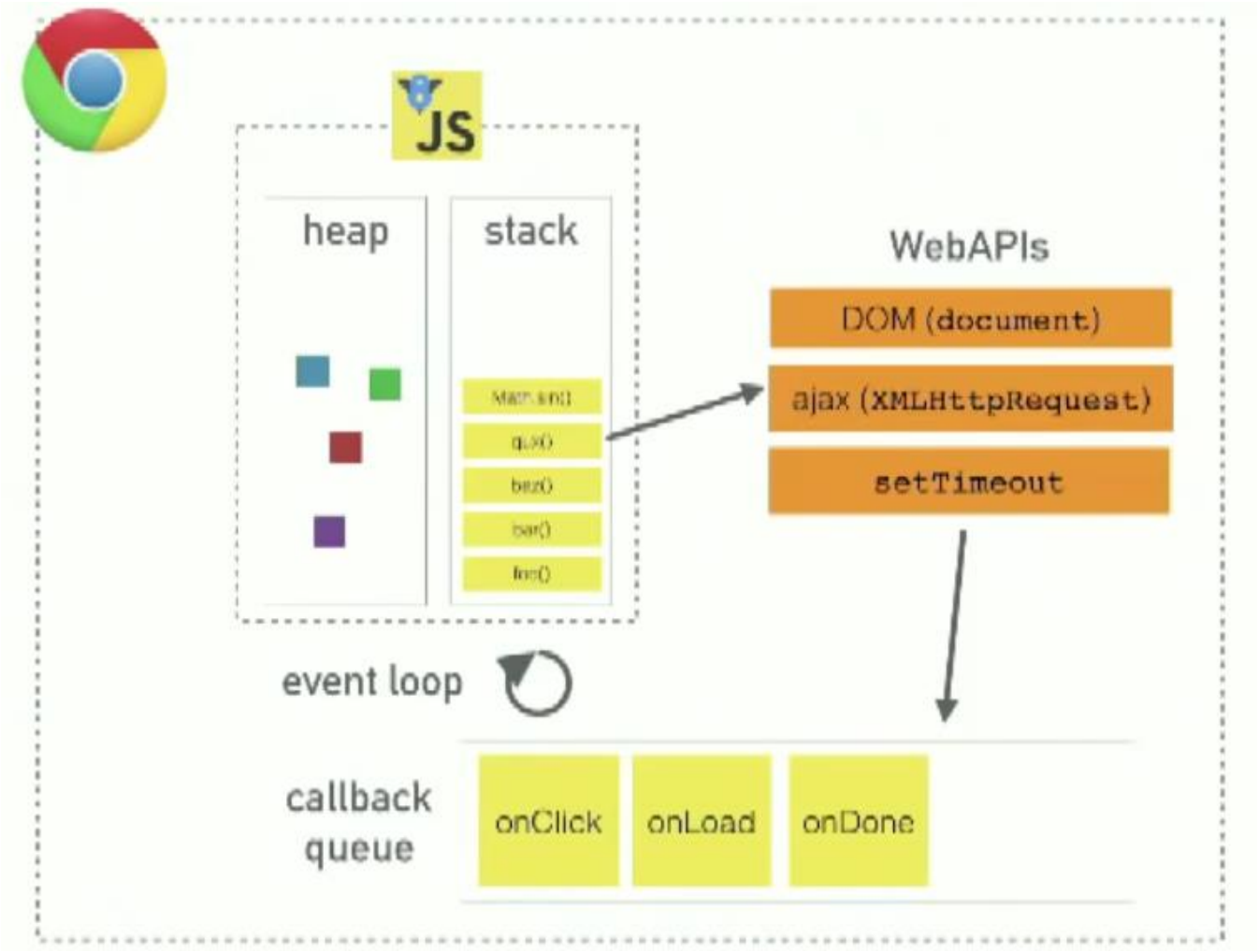


Рисунок 1.1 – Схема роботи event loop

Heap (купа) – об'єкти зібрані в купу, яка є ні що інше, як назва для найменш структурованої частини пам'яті.

Stack (стопка, стек) – репрезентація єдиного потоку виконання JavaScript-коду. Виклики функцій поміщаються в стек (про це нижче).

Цикл подій (Event Loop) – це те, що дозволяє Node.js виконувати неблокіруючіх операції введення/виведення шляхом вивантаження операцій в ядро системи, коли це можливо.

Оскільки більшість сучасних ядер є багато-ядерними, вони можуть обробляти кілька операцій, які виконуються у фоновому режимі. Коли одна з

цих операцій завершується, ядро повідомляє Node.js, що відповідна цієї операції функція зворотного виклику (далі для простоти буде використаний термін «коллбек») може бути додана в чергу опитування, щоб в кінцевому підсумку бути виконаною.

Коли Node.js запускається, вона ініціалізує цикл подій, обробляє наданий на вхід код (або переходить в REPL, який не розглядається в цьому документі), який може виконувати виклики асинхронного API, налаштовувати таймери або викликати process.nextTick(). Потім починається обробка циклу подій.

Розташована нижче діаграма спрощено показує порядок виконання операцій в циклі подій.

Кожна фаза має FIFO чергу коллбеков для виконання. Хоча кожна фаза є по-своєму особливою, зазвичай, коли цикл подій входить в дану фазу, вона буде виконувати будь-які операції, що відносяться до цієї фазі, а потім виконувати функцію зворотного виклику в черзі цієї фази, поки черга не буде вичерпана, або максимальну кількість функцій зворотного виклику НЕ буде оброблено. Коли черга вичерпана або досягнута межа функцій зворотного виклику, цикл подій переміститься на наступну фазу і так далі.

Так як будь-яка з цих операцій може планувати більше операцій, а нові події, оброблені на етапі опитування, ставляться ядром в чергу, нові події опитування можуть бути поставлені в чергу під час обробки поточних подій опитування. В результаті довго виконуються функції зворотного виклику можуть дозволити фазі опитування тривати набагато довше, ніж встановлений поріг таймера. Детальніше дивіться в розділах таймери і опитування.

Примітка: Між реалізаціями в Windows і Unix / Linux існує невелика різниця, але це не важливо для цього демо. Найважливіші частини описуються тут. Насправді, існує сім або вісім кроків, але цікаві нам, які фактично використовує Node.js вказані вище.

Browser or Web API's (браузерні або веб API) – вбудовані в браузер і здатні надавати дані з браузера і навколишнього комп'ютерного середовища і давати можливість виконувати з ними корисні і складні речі.

Вони не є частиною мови JavaScript, але вони побудовані на його основі і надають вам супер сили, які можна використовувати в JavaScript коді. Наприклад Geolocation API надає доступ до декількох простих конструкцій JavaScript, які використовуються для отримання даних про місцезнаходження, так що ви можете, скажімо, відобразити своє місце розташування на Google Map. У фоновому режимі браузер використовує низькорівневий код (наприклад C ++) для зв'язку з обладнанням GPS пристрої (або будь-яким іншим, доступним для визначення даних про місцезнаходження), отримання даних про місцезнаходження і повернення їх в середу браузера для використання в вашому коді. Але знову, ця складність абстрагована від вас за допомогою API.

1.2 Веб-сервіси Amazon

Веб-сервіси Amazon (AWS) – комерційне публічне рішення, підтримує і розвиває компанію Amazon з 2006 року. Поставляє готові рішення та послуги, як за інфраструктурними моделями (віртуальні сервери, ресурси зберігання), так і платформенний рівень (облачні бази даних, облачне використання програмного забезпечення, облачні безсерверні вироби, засоби розробки).

У значній мірі (на платформі Google Cloud Platform) вперше почали формуватися концепції регіональних вибірок у цілому, і визначили основні направлення для розробки публічних моделей різних рівнів. Довгий час було крупним у мірі за випускним публічних хмарних сховищах, у другій половині 2010-го годів увійшов за цим показником Azure від Microsoft, при цьому надійне домінування у сегментальних інфраструктурних та платформних послуг.

Область розміщення в декількох географічно розроблених центральних обробних даних, об'єднаних у групи за географічною близькістю, називаються «регіонами», всередині регіону реалізується декілька «доступних можливостей» (англ. Зона доступності), всередині яких забезпечується висока доступність розміщення основних служб; станом на 2019 рік діють 60 зон доступності в 20 регіонах. Підписчики можуть вибирати регіон і зону доступності, а також надають можливість організувати реплікацію даних і передати запропоновані між зонами доступності.

У хмарному просторі обладнання СУБД різних категорій. Доступні NoSQL-системи – Amazon SimpleDB, DynamoDB, резидентна СУБД ElastiCache, графова СУБД Neptune. У сервісі Amazon Relational Database Service (RDS) підписки можуть використовувати всі обласні бази під управлінням популярних реляційних СУБД MySQL, Oracle Database, Microsoft SQL Server і PostgreSQL, також доступні масштабні реферативні СУБД Amazon Aurora, сумісні з MySQL і Postgre. Аналітична масово-

паралельна реляційна СУБД ParAccel, адаптована для обласної інфраструктури, представлена під торговою маркою Amazon Redshift.

1.2.1 Влаштування Amazon Web Services

Велика кількість сервісів Amazon надає широкий спектр послуг та інструментів для створення комплексних додатків та проводи велику кількість обчислень за короткий час. А також розміщувати на серверах Amazon свої додатки. Після розміщення ці серверні машину можуть змінюватися для більшої продуктивності, а також пропускної здатності. Нижче проведено список цих сервісів (див. рис. 1.2).

Служба Amazon Athena дозволяє провести аналіз даних у Amazon S3, використовуючи стандартний SQL, для роботи в інших робочих системах не потрібно, а підписи можуть містити лише ті дані, що містяться в запитах.

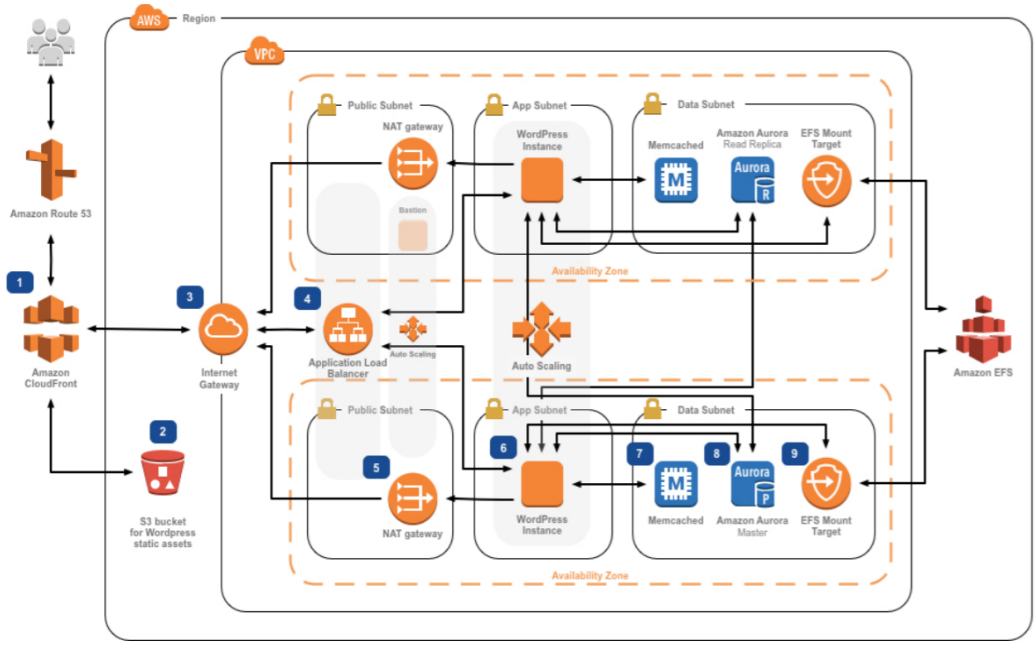


Рисунок 1.2 – Схема сервісів Amazon

Служба Elastic MapReduce зможе підписати створені Hadoop-кластери, які мають відповідну екосистему продуктів класу «найбільших даних» (в тому

численні Spark, Hive, HBase, Presto). Інструмент QuickSight надає підписник можливо візуального аналізу даних, розміщених в службі AWS. Служба Amazon Elasticsearch забезпечує хмарну доступність для переходу до пошукової системи Elasticsearch та в Kibana. Служба Amazon Machine Learning забезпечує підпис доступних інструментів для машинного навчання.

1.2.2 Платформні послуги

Окремий сервіс класу, що підтримує програмне забезпечення, – анонсоване повідомлення Amazon Kinesis (близько до можливості Apache Kafka), служба очевидних SQS та служба повідомлення SNS.

Середнє розширення запропонованого в парадигме безсерверних вичислень - AWS Lambda; Служба еластичних Кубернетів надає можливість розробити запропоновані в контейнерній інфраструктурі під управлінням Кубернети.

1.3 Порівняння сервісів для збереження файлів

1.3.1 Dropbox

Dropbox – файловий хостинг компанії Dropbox Inc., що включає персональне облачне зберігання, синхронізацію файлів та програму-кліент.

Dropbox дозволяє користувачеві створити спеціальну папку на своїх комп'ютерах, яка Dropbox синхронізує таку систему, яка має належне вміст, незалежне від того часу, яке пристрою використовується для прогляду. Файли, розміщені в цій папці, також доступні через веб-сайт Dropbox та мобільні додатки. Dropbox працює за моделями Freemium, в якому користувачі мають можливість створити безкоштовний аккаунт із заданим кількістю вільних просторів, у той час як для збільшення обсягу аккаунта необхідна платформа підпису.

Dropbox підтримує Windows, macOS, Linux; мобільні ОС Android, iOS, Windows Phone та BlackBerry; веб-браузери; також є неофіційні порти на MeeGo та Symbian.

Dropbox дозволяє користувачеві розміщувати файли на віддалених серверах за допомогою клієнтів або за допомогою веб-інтерфейсу через

браузер. При встановленні кліентського програмного забезпечення Dropbox на комп'ютері створює синхронізовану папку. Хотя головний акцент технологій працює на синхронізації та обробці інформації, Dropbox виконує історію завантаження, щоб після видалення файлів з сервера була можливість зберегти дані. Так само проводиться історія файлів, яка доступна для періоду останніх 30 днів, маючи на увазі цю доступну функцію безстрокових історій файлів «Pack-Rat».

Історія файлів ведеться за принципами різної кодирування, щоб сэкономити місце, займаючись файлами. У джерелах даних записується лише відмінна одна версія версії файлу від іншої. Файли, завантажені через кліент, не мають обмежень на розмір, но файли, завантажені через веб-інтерфейс, обмежені 20 ГБ. Також можна випустити файли для публічного доступу через папку «Публічне», що дозволяє використовувати сервіс у своїх файлах. У версії 0.8.x також з'явилася можливість подання в загальному доступному папці в «Моїй Dropbox» для наступного доступу через так званий «Shareable link», що є через веб-інтерфейс. Для спільної роботи над проектом сервісу є можливість створити «Спільний» папок для загального доступу до лицьових, які мають різні навчальні записи на сервісі. Доступна автоматична синхронізація файлів і папок і зберігання версії з можливістю відказу. Для користувачів Dropbox Professional та Dropbox Business доступна функціональність Smart Sync, що забезпечує економічне місце на найвищому дисплеї та відображає лише назву та інформацію про файли, які не завантажують їх вміст.

У відмінній від ряду аналогів, Dropbox не використовує шифрування даних на сторонній службі, що, в приватності, зробив можливим інцидент 19 липня 2011 року, коли з-за помилок в обновленному програмному забезпеченні сервера в ході четвірних годин був загнаний вхід у будь-який аккаунт з використанням любого пароля.

Сервіс пропонує безкоштовно 2 ГБ для зберігання даних, які можна збільшити безкоштовно до 16 ГБ, доступні нові користувачі або ж отримають

кілька гігабайт після виконання завдань (встановлення додатків Dropbox на мобільний телефон і т. Д.). А також можна купити 1 ТБ.

Є офіційний пакет SDK для створення власних пропозицій під Dropbox із використанням популярних мов та платформ Swift, Objective-C, Python, JavaScript, Java, HTTP, .NET.

1.3.2 GDRIVE

Google Диск – це сервіс зберігання, редагування та синхронізації файлів, розроблений компанією Google.

Його функції включають зберігання файлів в Інтернеті, загальний доступ до них і спільне редагування. До складу Google Диску входять Google Документи, Таблиці та Презентації - набір офісних додатків для спільної роботи над текстовими документами, електронними таблицями, презентаціями, кресленнями, веб-формами та іншими файлами. Загальнодоступні документи на Диску індексуються пошуковими системами.

1.3.3 Порівняння

Зберігання файлів і в Dropbox або Google диску є досить простим. Всі три сервісу надають апі для швидко і легкого доступу до можливостей цих платформ. Однак, основною відмінністю є кількість вільного місця для зберігання зображень і документів. Dropbox надає всього 2гб безкоштовного простору. Google диск 15Гб. AWS надає близько 25 гб безкоштовно, однак має обмеження на кількість запитів в день і місяць. За гнучкості використання Dropbox поступається Google диску і AWS сервісів, а платформа від Google має велику кількість рівнів захисту від злому і крадіжки вашої інформації, що є плюсом, але через це зростає поріг входження для новачків.

Також Dropbox надає менший функціонал по роботі з типу файлів їх розмірами та іншими параметрами ніж інші платформи, що ускладнює процес

завантаження і скачування зображень. Але все три сервісу надають можливість обробити права доступу як до окремих файлів так і до папок в цілому.

У свою чергу сервіси AWS тісно взаємопов'язані між собою, що дозволяє налаштувати групи доступу не тільки до файлів, але і редагуванню файлів і папок безпосередньо через сайт AWS. Однак, сервіси AWS можна використовувати безкоштовно. Для роботи з сервісу потрібно створити обліковий запис і прив'язати до нього кредитну карту, що так само, як і в випадку, з Google диском ускладнює вивчення цих технологій для початківців.

За певну плату, яка може бути різною в залежності від обраного паку послуг та апаратного забезпечення сервіси AWS можуть бути розширені до необхідних розмірів.

Отже, основними перевагами веб-сервісів від Amazon є:

1) гнучкість у налаштуванні;
2) можливість розширення апаратного забезпечення;
3) додаткові сервіси для пришвидшення розробки;
4) гарна документація.

Основними недоліками є:

1) обов’язкове надання банківських даних;
2) високий рівень входження для початківців;
3) висока ціна.

Основними перевагами сервісу DropBox є:

1) швідкість підключення та налаштування;
2) можливість разширення вмісту сховища.

Недоліки:

1) погана документація;
2) платне розширення;
3) слабке апаратне забезпечення.

Окрім сервісів для зберігання зображень користувачів у системі використовується платіжна система Stripe. Вона служить для реалізації

підписки на послуги, що надаються у системі. Та є універсальним інструментом для роботи з банківськими картами та іншими видами фінансових операцій у веб додатках, та інших сервісах чи додатках. Про цю систему, а також про її переваги і недоліки ми більш детально дізнаємося у наступному підрозділі кваліфікаційної роботи.

1.4 Порівняння електронних систем прийому й обробки платежів

1.4.1 Stripe

Stripe створює найпотужніші та гнучкі інструменти для інтернет-комерції. Незалежно від того, чи створюєте ви послугу підписки, ринок під замовлення, магазин електронної комерції або платформу для краудфандингу, ретельно розроблені API та неперевершена функціональність Stripe допоможуть вам створити найкращий можливий продукт для своїх користувачів. Мільйони найбільш інноваційних технологічних компаній у світі масштабують швидше та ефективніше, будуючи свій бізнес на Stripe.

За допомогою сервісу, додавання платежів на будь яку бізнес платформу може бути простим. Клієнти на з понад 30 країн, можуть приймати платежі в Інтернеті та особисто, і швидко виплачувати. Також для система прийому и обробки електронних платежів Stripe дозволяє своїм клієнтам управляти платежами або створити власний досвід від кінця до кінця.

Смуга була побудована для платформ. Додайте нові рядки доходу для вашої платформи та своїх клієнтів, приймаючи особисті платежі, підтримуючи підписки та рахунки-фактури, і дозвольте Stripe допомогти в регулюванні та дотриманні правил.

1.4.2 Amazon Pay

Також у процесі підготовки перед розробкою додатку була розглянута система від компанії Amazon, яка має назву Amazon Pay.

Amazon Pay – це послуга з обробки онлайн-платежів, яка належить Amazon. Запущена в 2007 році. Amazon Pay використовує споживчу базу Amazon.com і фокусується на наданні користувачам можливості оплачувати їхні рахунки в Amazon на веб-сайтах зовнішніх торговців. З січня 2019 року послуга доступна в Австрії, Бельгії, Кіпрі, Німеччині, Данії, Іспанії, Франції,

Угорщині, Люксембурзі, Республіці Ірландія, Індії, Італії, Японії, Нідерландах, Португалії, Швеції, Великобританії, США[5].

У цієї системи також є не погона документація, але менш розгорнута ніж у Stripe, також вона менш поширена і тісно пов’язана з іншими сервісами Амазону, шо у випадку додатку, що був у розробці є негавиною рисою. Також, я і у випадку з сервісами, що були описані вище Amazon Pay потребує, при створенні акаунту розробника, підключення платіжної картки.

1.4.3 PayPal

PayPal – міжнародна електронна платіжна система. Це найпоширеніший у світі спосіб розрахунків в Інтернеті. Нині PayPal використовують понад 230 мільйонів людей у 190 країнах світу. PayPal виступає посередником між продавцем і покупцем, забезпечуючи надійність оплати платіжними картками VISA, MasterCard, American Express та інших платіжних систем.

PayPal – найбільша електронна платіжна система, яку використовує для розрахунків 164 мільйони клієнтів у всьому світі. В Україні PayPal є популярним інструментом для оплати покупок у зарубіжних інтернет-магазинах. Ви можете підключити до платіжної системи PayPal одну або декілька банківських карток ПриватБанку для оплати своїх покупок в Інтернеті.

Отже після розгляду усіх вище перерахованих систем ми можемо скласти наступну картину. Перевагами системи для прийому и обробки електронних платежів Stripe, порівнянно з іншими розглянутими вище системами для прийому и обробки електронних платежів може приймати в обробку будь-які кредитні картки чи дебітові карти, у свою чергу PayPal та Amazon Pay можуть працювати лише з картами, що були створені у цих системах. Також однією із переваг є гнучкість у налаштуванні зовнішнього вигляду форми, що служить для введення та обробки персональних даних користувача на веб-сторінці додатку чи на екрані мобільного додатку. У системі прийому и обробки електронних платежів Stripe є великий набір вбудованих стилів для налаштування зовнішнього виду форми введення номерів карт користувачів, а також є дуже гарна документація, що дозволяє у повній мірі змінити усі налаштування зовнішнього виду форми спираючись на потреби конкретно обраного стилю та зовнішнього види додатку, що розроблюється. Також є багато видів форм обробки даних персональних кредитних та дебетових карт користувачів, що відповідають за обробку коректності введених даних та відправки їх на сервери системи прийому и обробки електронних платежів Stripe, для зняття коштів та отримання результату транзакції.

Платіжна система PayPal, в порівнянні з іншими не представлена в Україні і не може використовуватися у нашій країні, але чудово працює у інших країнах світу. Цей факт є суттєвим недоліком і може значно погіршити процес розробки та підтримку додатку, а також роботу по тестуванню продукту під час передачі кінцевому клієнту.

Тому під час порівняння цих трьох систем прийому и обробки електронних платежів була обрана саме система Stripe, через свою гнучкість у налаштуванні зовнішнього види та можливостей оброки даних для різних банківських карток чи дебетових карт.

1.5 Аналіз вимог

Отже, резюмуючи все вищесказане, ми можемо зробити висновок, що для розробки великих бізнес-додатків сервіси AWS підходять найкраще. Вони надають гнучкість у виборі апаратного забезпечення і налаштування прав доступу і робочого оточення, а так само надають ряд додаткових сервісів для спрощення і прискорення як роботи програми так і його розробки. А також система прийому и обробки електронних платежів Stripe[4]. Яка має ряд переваг перед конкурентами

А саме:

1) гнучкість у налаштуванні зовнішнього вигляду;
2) гарна документація;
3) популярність платформи та наявність великої кількості готових рішень.

Тож у процесі розробки додатку будуть використовуватися ці засобі для роботи з файлами користувачів и платіжними операціями у системі, розробку якої ми будемо розглядати у наступних розділах дипломної роботи.

Таким чином, на основі аналізу технологій, що був проведений вище, ми можемо зробити висновок, що оптимальним вибором для роботи буде система прийому та обробки електронних платежів Stripe, що відповідає усім вимогам, що будуть описані нижче, а також веб-сервіси Amazon[7]. Отже, ми визначилися, що для створення додатку будуть використовуватися система прийому та обробки електронних платежів Stripe, а також веб-сервіси Amazon для забезпечення надійного сховища для файлів користувачів. До додатку, що буде розроблено можна висунути наступні вимоги:

1) гнучкість у налаштуванні доступу до сховища користувачів;
2) можливість завантажувати та зберігати зображення;
3) можливість розширення розміру сховища;
4) створення та підтримка тарифних планів, що будуть використовуватися у системі;

5) простий та зрозумілий для користувачів інтерфейс роботи з банківськими картами;

6) захист особистих банківських даних користувачів;

7) захист усіх фінансових операцій у системі.

2 ПРОЕКТУВАННЯ ДОДАТКУ

Користувач може використовувати систему по різному, нижче приведено варіанти використання (див. рис. 2.1). У реєстрації є два етапи. На першому користувач вносить необхідні дані, що є важливими для роботи з системою, що буде розроблятися у ході виконання кваліфікаційної роботи. На другому етапи користувач доповню свій профіль додатковою інформацією.

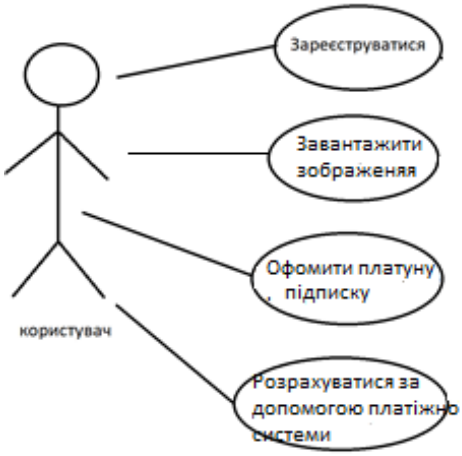


Рисунок 2.1 – Діаграма варіантів використання

Для початку роботи користувач повинен зареєструватися в системі. Без цього використовувати систему не можливо. При першому вході в систему користувач може використовувати деякі функції безкоштовно, але після певної кількості завантажень він повинен підписатися на один із тарифних планів для продовження роботи. Після вибору одного з тарифних планів, що присутні у системі користувач може оплатити їх використовуючи систему прийому та обробки електронних платежів Stripe. Та виконати оплату будь-якою із своїх карток.

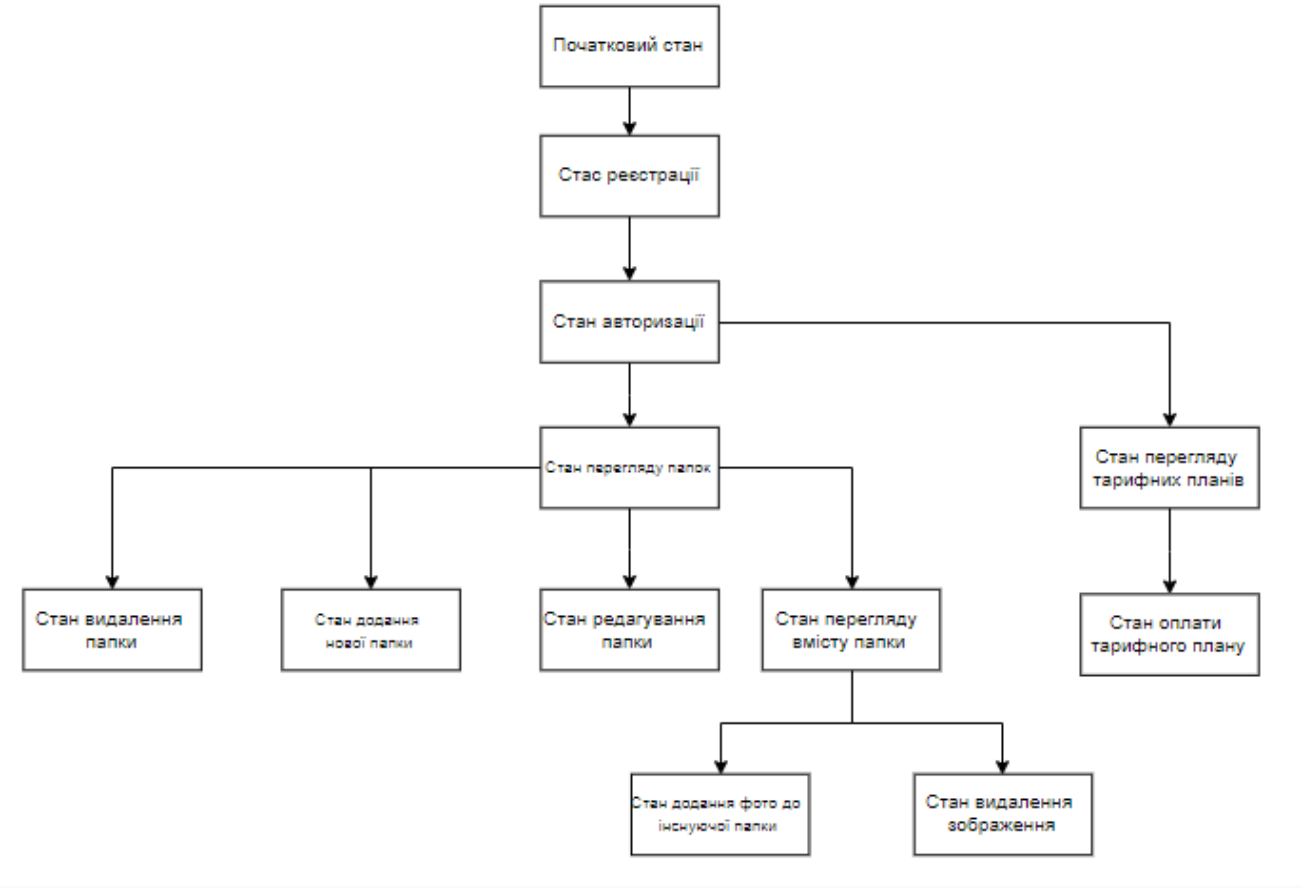


Рисунок 2.2 – Діаграма станів

На схемі вище описані стани додатку, у яких може перебувати застосунок. Першим етапом є реєстрація, яка проходить у два етапи. Спочатку користувач вводить дані, що є обов’язковими для роботи з системою, після чого він може додати більше інформації до свого профілю. Після реєстрації користувач може використовувати систему. Тобто, завантажувати зображення і створювати папки. Але після 30 днів пільгового періоду він повинен підписати на один з тарифних планів, для продовження роботи. Користувач може переключатися між станами, повертатися до перегляду тарифних планів, або завантаження зображень чи створення папок за власним бажання. Для цього на клієнтській частині передбачено кнопки, які викликають функції обробники.

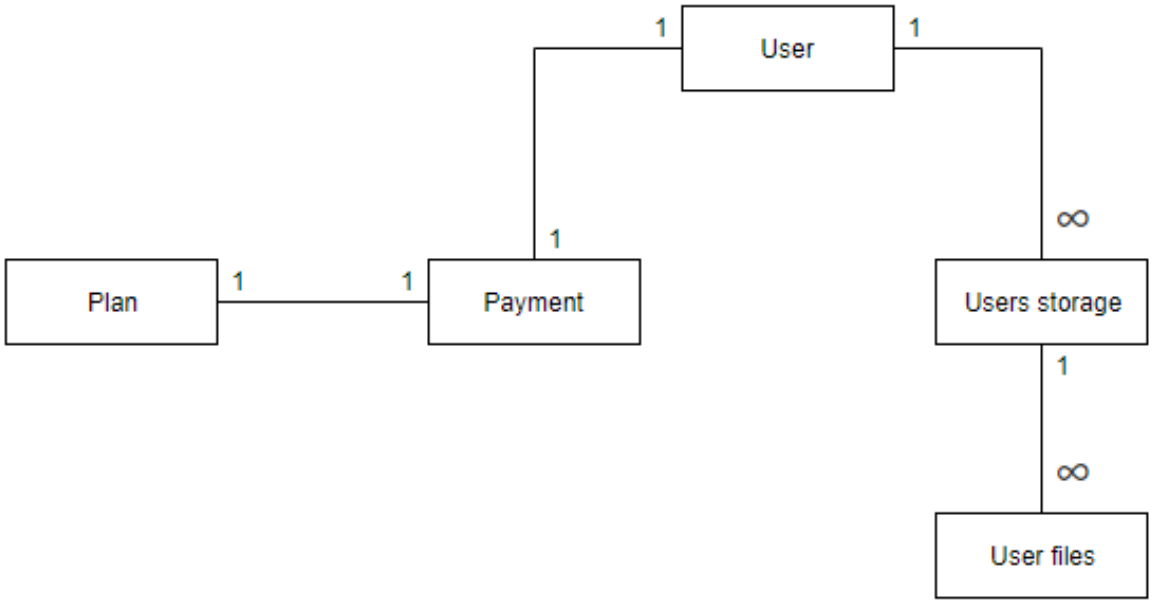


Рисунок 2.3 – ER-діаграма

На схемі віще зображена структура бази даних, що використовується у додатку. Для роботи нам необхідний запис про користувача, а також список усіх доступних планів у системі. Після вибору одно з тарифних планів користувач проводить оплату, інформація про це зберігається у таблиці платежів. Після оплати для користувача відкривається можливість створювати свою папки, а також додавати до них файли.

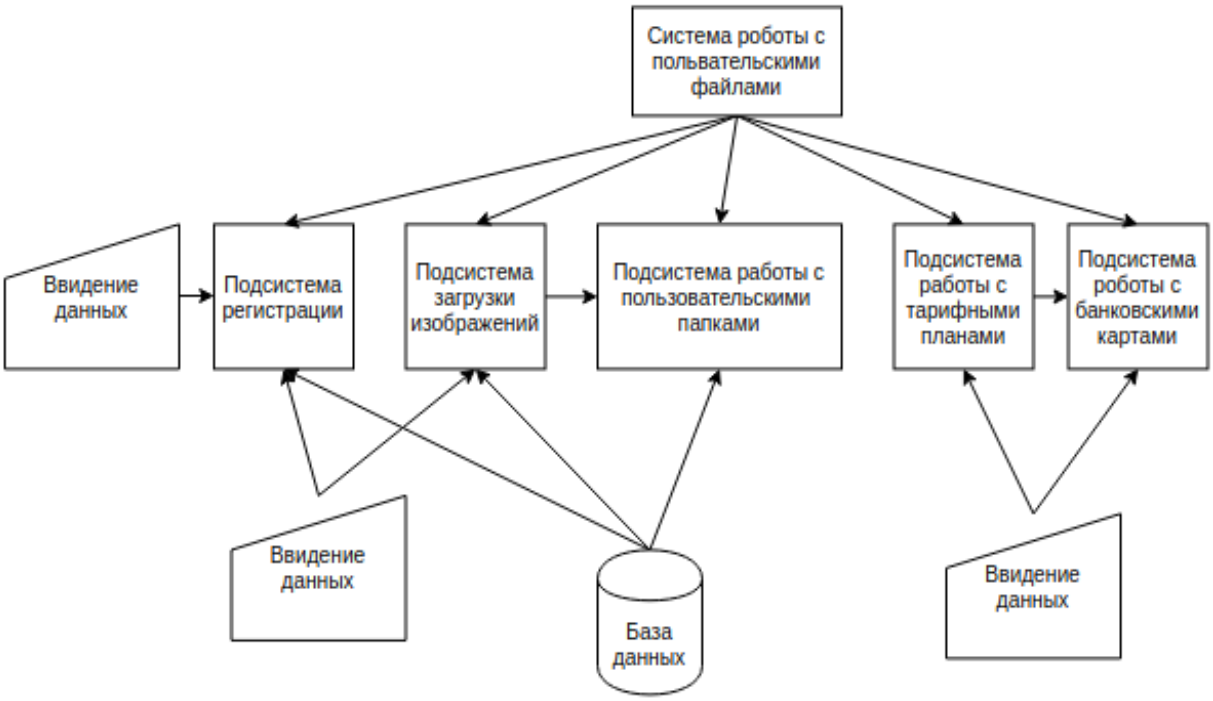


Рисунок 2.4 – Функціональна діаграма

Користувач починає роботу з додатком з введення своїх реєстраційних даних. За їх оброку та запис відповідає підсистема реєстрації. Наступним етапом роботи може буди завантаження файлів до хмарного сховища, за це відповідає підсистема завантаження зображень. Для цього користувачеві також необхідно ввести ім’я паки до якої буде збережено зображення, а також завантажити саме зображення на клієнтській частині для відправки його в функцію обробник у серверній часті додатку. Для створення папок використовується підсистема створення папок, у якій користувач повинен зазначити ім’я папки, що буде створена та збережена до хмарного сховища, яке розміщено на серверах Amazon. Також у додатку буде розроблено підсистему прийому та оброки електронних платежів, яка буду обробляти фінансові операції у системі та записувати інформацію про них. Усі операції збереження та виводу даних проходять з використанням бази даних, що служить для збереження інформації про користувачів, а також посилання на їх зображення та папки у контейнері Amazon S3.

3 РОЗРОБКА ДОДАТКУ

Однією з основних функцій програми є робота з одними ізорабраженіями файлами, а так само забезпечення доступу до додатка і оплата послуг сервісу через систему платних підписок на функціонал надається системою.

3.1 Розробка сервісу завантаження зображень

Отже, першим етапом ми повинні підключитися до сервісів AWS. Для цього нам потрібно створити акаунт у сервісі IAM. Після цього створити нове сховище, яке називається контейнер. При створенні об’єкту підключення до сервісів ми маємо переди усі три ключі, що ми отримали у процесі налаштування сервісів (див. рис. 3.1)

```
const s3 = new AWS.S3({
    accessKeyId: config.IAM_USER_KEY,
    secretAccessKey: config.IAM_USER_SECRET,
    Bucket: config.BUCKET_NAME
});
```

Рисунок 3.1 – Об’єкт підключення

Також ми повинні додати нашу ip-адресу до списки дозволених у налаштування сервісів. Після цього ми можемо починати роботу по створенню додатку для роботи с S3. При відправленні зображення з клієнтської частини додатку ми маємо обробити вхідний запит та отримати об’єкт файлу з запиту. Для цього використовуються спеціалізовані бібліотеки, що створенні спеціально для роботи з одним або масивами зображень. У нашому додатку ми будемо працювати лише з одним файлом, без можливості завантажувати

масив зображень. Для відправки файлу використовується спеціальним вид запиту form-data, що містить у соби спеціальне поле, у яке ми розмінюємо наше зображення для відправки. Нижче наведено приклад отримання зображення тіла запиту (див. рис. 3.2)

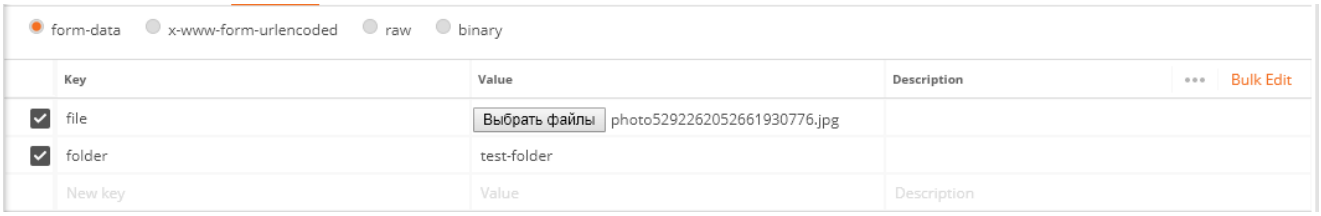


Рисунок 3.2 – Приклад запиту до серверу

На зображенні видно, що ми передаємо два параметри: перший з них це сам файл, що ми будемо завантажувати, а також ім’я папки, у яку ми будемо завантажувати це зображення. Це усе, що нам потрібно відправити для продовження роботи. Нижче наведено приклад функції, що обробляє запит та отримує усі необхідні зміні для завантаження зображення.

```
const form = new multiparty.Form();
form.parse(req, (err, fields, files) => {
    if(err) return console.log(err);
    const file = files.file[0];
    const fileName = file.originalFilename;
    const folderId = fields.folderId[0]
    const folderName = fields.folderName[0]
```

Рисунок 3.3 – Робота з файлами

На рисунку вище ми створюємо об’єкт форми, що буде працювати з формою у запиті від клієнтської частини та обробляємо її для отриманні зображення та усіх необхідних нам мета-даних, що ми будемо використовувати при роботі з цим файлом. Ми зберігаємо усі потрібні данні у

змінні для подальшого завантаження у сервіс. Наступним етапом ми завантажуємо усе у контейнер за допомогою об’єкту з’єднання, що ми створили вище (див. рис.3.4)

```
const photoKey = `${folderName}/${fileName}`;
s3.upload({
  Bucket: config.BUCKET_NAME,
  Key: photoKey,
  Body: fs.readFileSync(file.path),
  ACL: 'public-read'
}, (err, data)=> {
```

Рисунок. 3.4 – Завантаження до сервісу S3

Перше, що нам потрібно зробити для завантаження зображення, це створити унікальний ключ, що буде відповідати за ідентифікацію зображення, що ми додали. Для цього ми використовуємо ім’я папки, у яку клієнт хоче завантажити його зображення та ім’я файлу, шо ми отримали з мета-даних, що були передані разом з цим файлом з клієнтської частини додатку. Та зберігаємо це у спеціальну змінну. Тепер ми можемо завантажити зображення до сервісу S3. Для цього використовується метод upload, що поставляєтеся у бібліотеці AWS. У його конструктор треба передати наступні параметри:

1) ім'я контейнеру, у який ми будемо зберігати зображення;
2) ключ зображення, який ми створили для ідентифікації;
3) об'єкт файлу, що був отриманий з клієнтської частики;
4) правову мітку, шо буде ключ, за допомогою якого ми зможемо сортувати зображення на публічні та приватні, а також клієнт зможе завантажувати свої зображення.

Останнім кроком при завантаженні зображення є збереження даних, що повертає нам бібліотека при успішному завантаженні зображення, або обробка помилки, що сталася у процесі завантаження. Для збереження ми будемо

використовувати базу даних. Нижче розглянемо приклад роботи з даним отриманими з сервісу та запис їх до бази (див. рис. 3.5).

```
}, (err, data)=> {
  if (err) return console.log('There was an error uploading your photo: ', err.message);
  console.log(data, 'data')
  const newImg = new models.images({
    path: photoKey,
    folder: folderId,
    link: data.Location
  })
  newImg.save();
  res.json({message:'Successfully uploaded photo.'})
});
```

Рисунок 3.5 – Збереження до бази даних

У базу даних ми зберігаємо ключ, який ми створили на початку процесу роботи з зображеннями, також ідентифікатор папки, у яку користувач завантажив своє зображення, та посилання, по якому ми можемо отримати зображення для перегляду на сторінці браузеру чи мобільного пристрою, або для завантаження цього зображення на наш пристрій.

Також користувачі можуть зберігати свою зображення у папках, для кращого сортування зображень. Для цього є спеціально розроблений сервіс, що дозволяє створювати користувацькі папки, для збереження зображень, а також надає можливість налаштовувати права доступу до цих папок. Нижче наведено приклад функції, що створює ці папки (див. рис. 3 6)

```
createAlbum = async albumName => {
    console.log('album', albumName);
    try {
        if (albumName.indexOf('/') !== -1) return console.log('Album names cannot contain slashes.')
        const foldersList = await models.folders.findOne({name: albumName});
        if(!foldersList){
            const albumKey = encodeURIComponent(albumName) + '/';
            const test = await s3.putObject({Key: albumKey, Bucket: config.BUCKET_NAME, ACL: 'bucket-owner-full-control'}).promise();
            if(test) return await new models.folders.create({name: albumName})
        }else return ({message:'Folder alredy exists'});
    } catch (error) {
        return ({message: error.message});
    }
};
```

Рисунок 3.6 – Функція створення папки

Функція приймає ім’я папки, що вводить користувач на стороні клієнту у формі. Та перевіряє його на наявність символів, що не допускаються у імені папки. Після цього ми перевіряємо наявність такої папки у конкретного користувача, щоби виключити можливість створення папок з однаковими іменами. Якщо ім’я вільне ми додаємо до нього символ косої лінії. Після цих дій ми можемо додати нову папку до контейнеру у сервіс Amazon. Для цього ми використовує вже існуюче підключення до сервісу с три, що було створено для інших функцій у системі та додаємо цю папку з вказанням ключа, що виступає ім’я цієї папки, а також ім’я контейнеру, що дозволяє використовувати декілька різних контейнерів для підтримки системи та керування файлами користувачів (див. рис.3.7).

```
await s3.putObject({Key: albumKey, Bucket: config.BUCKET_NAME, ACL: 'bucket-owner-full-control'}).promise();
```

Рисунок 3.7 – Додання нової папки до сервісів Амазон

Окрім імен контейнеру та папки користувача ми також додаємо до об’єкту, що передається у якості параметрів спеціальний ключ, що є необхідним для встановлення прав доступу до цієї папки користувачем. У нашому випадку це повний контроль за станом і вмістом цієї папки, також існують інші види ключів, наприклад тільки перегляд, або тільки запис.

Якщо усе пройшло успішно ми можемо зберегти ім’я цієї папки до нашою бази даних та повернути повідомлення про успішність створення користувачу (див. рис.3.8).

```
if(test) return await new models.folders.create({name: albumName})
```

Рисунок 3.8 – Збереження до бази даних нової папки

Тепер коли користувач хоче додати нове зображення до своєї папки він зможе переглянути повний список своїх папок, що були створені раніше. Для цього використовується наступна функція (див. рис. 3.9)

```
getAlbums = async () =>{
    try {
        const albums = await models.folders.find();
        return albums
    } catch (error) {
        return ({message: error.message});
    }
}
```

Рисунок 3.9 – Функція для отримання списку усіх папок користувача

Якщо користувач хоче переглянути усі свої зображення, що збережені у папці він може це зробити по натисканню на будь-яку зі своїх папок, для цього існує спеціальна функція, що звертається до папки за допомогою ключа до вибраної користувачем папки (див. рис. 3.10)

```
getAlbumPhotos = async albumName => {
    try {
        const photos = await s3.getObject({ Bucket: albumName }).promise();
        return photos
    } catch (error) {
        return error
    }

}
```

Рисунок 3.10 – Функція отримання усіх зображень з папки

На стороні користувача вони відображаються у вигляді списку. З кнопкою для видалення зображення. Також користувач може видалити і саму папку. Але при видалені папки також буде видалено і усі зображення, що були

завантажені до цієї папки. Нижче зображено приклад функції, що служить для видалення папок користувачів (див. рис. 3.11).

```
deleteAlbum = async (albumName, id) => {
    const albumKey = encodeURIComponent(albumName) + '/';
    try {
        const deletedFolder = await s3.deleteObjects({Delete: {Objects: [{Key: albumKey}]}, Bucket: config.BUCKET_NAME}).promise();
        console.log(deletedFolder)
        if(deletedFolder) {
            await models.images.deleteMany({folder : id});
            await models.folders.findOneAndDelete({name:albumName});
        }
    } catch (error) {
        return ({message: error.message});
    }
}
```

Рисунок 3.11 – Функція видалення папки користувача

Для початку ми видаляємо папку з контейнеру Amazon, після успішного видалення ми видаляємо запис про цю папку з нашої бази даних. Для цього у параметрах ми передаємо ім’я папки та її ідентифікатор у базі даних.

Також ми можемо видалити окремо одне з зображень з папки. Для цього ми використовуємо функцію (див. рис. 3.12).

```
deletePhoto = async(folderName, photoKey)=> {
  try {
    console.log(folderName, photoKey,'sss');
    let path = `${folderName}/${photoKey}`;
    if(!folderName && photoKey) return ({message:'folder name and photo key is required'});
    const deleteObject = await s3.deleteObject({Key: path, Bucket: config.BUCKET_NAME}).promise();
    if(deleteObject){
      const isdelete = await models.images.deleteOne({path});
      console.log(isdelete);
      return isdelete? ({message: 'Successfully deleted photo.', isdelete}) : ({message: 'Some error'});
    }
  } catch (error) {
    return error
  }
}
```

Рисунок 3.12 –Функція видалення зображення

Спочатку ми знову, як і у випадку з додаванням зображення, отримуємо повних шлях до файлу. Для цього ми використовуємо ім’я папки та зображення, а також додаємо між ними символ косої лінії. Після цього ми

можемо використовувати цей ключ, для видалення зображення з контейнеру. Останнім кроком ми видаляємо посилання на це зображення з бази даних.

3.2 Розробка сервісу прийому и обробки електронних платежів

Для роботи з системою користувач може обрати один з планів, що створені у системі, та підписатися на них. Оплата проходить за допомогою банківської карти, та функціоную за на основі системи прийому и обробки електронних платежів Stirpe. Нижче представлено функції для створення плану для створеного продукту (див. рис. 3.13).

```
app.post('/plans', async (req, res)=>{
    const {id} = req.body;
    const plan = await stripe.plans.create({
        currency: 'usd',
        interval: 'month',
        product: id,
        nickname: 'Pro Plan',
        amount: 4000,
    });

    res.json(plan)
})
```

Рисунок 3.13 – Функція створення нового плану

Для створення плану ми маємо передати період на який буде діяти тарифний план, валюту, що буде використовуватися для оплати підписки, продукт, що виступає планом, а також ім’я тарифного плану. Також треба передати кількість грошей, що буде стягуватися за кожний новий період.

Також ми повинні створити продукт, на який буде підписуватися користувач (див. рис.3.14).

```
app.get('/prod', async (req, res)=>{
    const product = await stripe.products.create({
        name: 'My SaaS Platform',
        type: 'service',
    });
    res.json(product)
})
```

Рисунок 3.14 – Продукт, що буде використаний для підписок

Цей продукт буде використаний для підписки. Для цього ми отримуємо унікальний ідентифікатор цього продукту. І передаємо його до плану, у функцію, що була описана вище. Наступним етапом користувач повинен виконати платіж, для цього він повинен ввести номер своєї карти у клієнтському додатку після цього ми відправляємо запит до api Stripe, яке повертає нам ідентифікатор користувача, а також усі данні по його карті. Цей ідентифікатор використовується у наступній функції, що буде використовуватися для сплати плати за користування планом (див. рис. 3.15)

```
app.post('/charge', async(req, res)=>{

    const {amount, token, stripeEmail}  = req.body;

    const charge = await stripe.charges.create({
        amount,
        currency: 'usd',
        plan : 'plan_Ddhm7j3dEs0gW5',
        source: token
    })
    if(charge) res.json(charge)

});
```

Рисунок 3.15 – Функція оплати підписки

Після успішного підтвердження платежу ми зберігаємо інформацію про цей платіж. Тепер користувачі можуть використовувати систему для зберігання зображень.

Для входу в систему потрібно зареєструватися та зайти у свій акаунт. Для цього існують функції, що обробляють дані, що вводить користувач та записують їх до бази даних, якщо усі переданні поля відповідають вимогам системи (див. рис.3.16)

```
static login (params)  {
    let email = params.email.toLowerCase();
    let userId;
    return User.findOne({where: {email}})
        .then(user => {
            if (!user) return null;
            return crypto.compare(params.password, user.password)
                .then(() => {
                    return crypto.getToken(user.id, user.email, user.role)
                })
                .then(token => {
                    return Token.create({userId: user.id, token})
                })
                .then(tokenData => {
                    if (!tokenData) return ({message: 'error in login!'});
                    return {
                        id: userId,
                        email: params.email,
                        token: tokenData.token
                    }
                })
                .catch((e) => {
                    if (e.message === 'Validation error') return Token.findOne({where: {userId: user.id}})
                        .then(token => {
                            if(token) return ({email: params.email, token: token.token, id: userId})
                        })
                    return(e.message)
                })
        })
};
```

Рисунок 3.16 – Функція входу до системи

Першим етапом ми переводимо електронну пошту користувача до нижнього регістру для того, щоб уникнути повторів у базі даних, та перевіряємо наявність користувача з такою електронною поштою у нашій базі даних. Після цього ми генеруємо унікальний ідентифікатор, що буде служити для однозначної ідентифікації особо користувача у системі. Для подальшого використання ми записуємо цей унікальний ідентифікатор до іншої таблиці у нашій базі даних, якщо він був успішно створений. Останнім кроком, коли ми зберегли новий унікальний ідентифікатор до бази даних, ми повертаємо об’єкт з усією необхідною інформацією про користувача на клієнтську сторону. Якщо ж виникають помилки – повертаємо повідомлення з текстом помилки. Також, у процесі може виявитися, що користувач з такою електронною поштою вже зареєстрований у системі, у такому випадку ми повертаємо данні про нього без створення нового унікального ідентифікатора та запису його у базу даних, тобто використовуємо вже існуючий запис.

Для реєстрації у системі використовується наступна функція, що першим етапом переводить електронну пошту користувача до нижнього регістру для того, щоб уникнути повторів у базі даних, та перевіряємо наявність користувача з такою електронною поштою у нашій базі даних (див. рис. 3.17). Якщо все проходить успішно - повертає електронну адресу та ім’я користувача.

```
static async create (name, email, password) {
    email = email.toLowerCase();
    const user = await User.create({name, email, password: crypto.getHash(password)});
    if (!user) return ('Error while signing up');
    return ({email, name});
}
```

Рисунок 3.17 – Функція створення нового користувача

Після проходження першого кроку реєстрації новий користувач може продовжити процес створення свого акаунту. Для цього використовується наступна функція, що буде розглянута нижче (див. рис.3.18).

```
static async signupSecondStep (id, params) {
    const user = await User.findById(id);
    if (!user) return ('User isn`t found');
    if (user.secondStepRegistration) return ("You were successfully signed up!");
    const userContact = await Contact.findOne({
        where: {linkedInProfileLink: params.linkedInProfileLink}, include: [
            {model: User}
        ]
    });
    if (userContact && userContact.User) return false;
    const contact = await Contact.upsert({
        linkedInProfileLink: params.linkedInProfileLink,
        name: params.name,
        title: params.title
    }, {returning: true});
    params.contactId = contact[0].id;
    params.secondStepRegistration = true;
    const sendTo = await SendTo.findOne({where: {email: user.email}});
    if (sendTo) sendTo.updateAttributes({contactId: params.contactId});
    return user.updateAttributes(params);
}
```

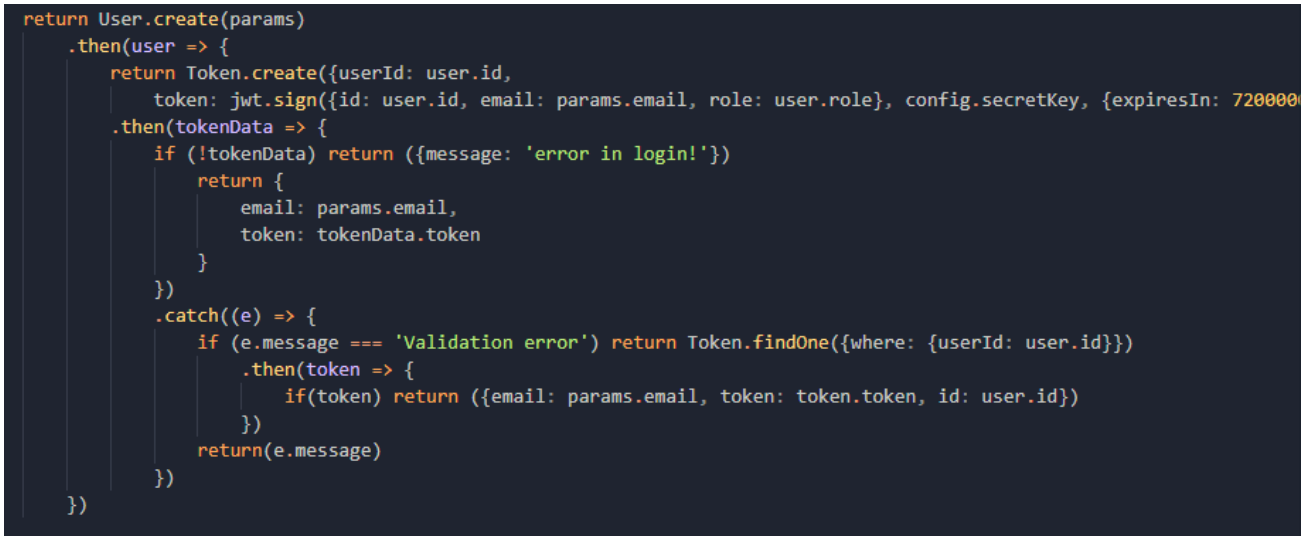
Рисунок 3.18 – Функція другого кроку реєстрації

Окрім звичайної реєстрації через електронну адресу та пароль для входу у систему користувачі можуть увійти через свій акаунт у Google, або інших соціальних мережах. Для цього існує ще один метод реєстрації, що буде описано нижче (див. рис. 3.19).

```
static loginGoogle (params) {
    return User.findOne({where: {googleId: params.googleId}})
        .then(user => {
            if (user) {
                return Token.create({userId: user.id,
                    token: jwt.sign({id: user.id, email: params.email, role: user.role}, config.secretKey, {expiresIn: 7200000})})
                    .then(tokenData => {
                        if (!tokenData) return ({message: 'error in login!'})
                        return {
                            email: params.email,
                            token: tokenData.token
                        }
                    })
                    .catch((e) => {
                        if (e.message === 'Validation error') return Token.findOne({where: {userId: user.id}})
                            .then(token => {
                                if(token) return ({email: params.email, token: token.token, id: user.id})
                            })
                        return(e.message)
                    })
            }
```

Рисунок 3.19 – Реєстрація через Google акаунт

Для реєстрації нам необхідно отримати унікальний ідентифікатор користувача від Google. Після чого отримані дані ми записуємо до нашої бази даних. Якщо все пройшло успішно ми можемо створити унікальний ідентифікатор для нашого додатку. У тіло цього ідентифікатору ми записуємо усю необхідну інформацію про користувача, що може знадобитися у процесі роботи користувача з додатком. У випадку успішного збереження до нашої бази даних ми повертаємо цей унікальний ідентифікатор та електронну адресу користувача до клієнтської частини нашого додатку. Якщо такий користувач вже існував у системі ми шукаємо вже існуючий унікальний ідентифікатор цього користувача у нашій базі даних і повертаємо його до клієнтської частини. Якщо цей користувач реєструєтеся вперше ми повинні додати новий запис до бази даних і виконати усі етапи, що були описані вище(див. рис 3.20).



```
return User.create(params)
    .then(user => {
        return Token.create({userId: user.id,
            token: jwt.sign({id: user.id, email: params.email, role: user.role}, config.secretKey, {expiresIn: 720000
        .then(tokenData => {
            if (!tokenData) return ({message: 'error in login!'})
                return {
                    email: params.email,
                    token: tokenData.token
                }
            })
            .catch((e) => {
                if (e.message === 'Validation error') return Token.findOne({where: {userId: user.id}})
                    .then(token => {
                        if(token) return ({email: params.email, token: token.token, id: user.id})
                    })
                return(e.message)
            })
    })
```

Рисунок 3.20 – Створення запису нового користувача на основі акаунтку Google

ВИСНОВКИ

У ході написання кваліфікаційної роботи були розглянуті сучасні технології для завантаження та зберігання користувацьких зображень. Такі як: Dropbox, Google drive і Amazon S3. Буде проведено аналіз цих сервісів для виявлення того, який підходить найкраще для виконання поставлених вимог. Також були розглянуті системи прийому і обробки електронних платежів. У ході аналізу були розглянуті такі сервіси: PayPal, Stripe, Amazon Pay.

Під час написання другого розділу були сформульовані вимоги до додатку, що буде розроблятися. На основі цих вимог буде розроблятися додаток.

Отже, у ході виконання кваліфікаційної роботи було розроблено додаток, що дозволяє зберігати користувацькі зображення і файли у хмарному сховищі, що надається сервісами компанії Amazon. Для цього використовувалися пакети для розробників, що надають доступ до усього функціоналу веб-сервісів Amazon. На основі цього пакету були розроблені сервіси для роботи з зображеннями, що завантажу користувач, а також створення папок для кращого сортування користувацьких файлів. Для цього створювався контейнер, у якому користувач міг створювати свої папки. Після цього, за допомогою пакету користувачеві надались права до цих папок. Також у ході роботи над кваліфікаційною роботою було спроектовано і розроблено сервіс для роботи з банківськими та онлайн картами на основі системи прийому й обробки електронних платежів Stripe. Були розроблені функції для обробки запиту на стягування коштів за тарифними планами, що були попередньо створені в системі, а також таблиця у базі даних, що буде зберігати усю інформацію про платежі користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1 Янг А., Мек Б., Кантелон М Node.js в дії. 2018. Питер. 250 с.

2. Девід Херрон. Node.js Розробка серверних веб-додатків на JavaScript. ДМК Пресс. 144 с.

3. Браун Ітан. Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript. Керівництво. 300 с.

4. Кантелон М. Цифровая обработка сигналов:методы и средства. ISBN, 2014. – 416 с.

5. Ван Тассел Д. Стиль, разработка, эффективность отладка и испытание программ. М.: Мир, 2010. 320 с.

6. Вологдин Е.И. Паттерны проектирования / Е.И. Вологдин.М.: Санкт-Петербург, 2012. 96 с.

7. JavaScript inf9o. URL : https://javascript.info/ (дата звернення: 19.04.2018).

8 Express. URL : https://expressjs.com/ (дата звернення: 23.04.2018).

9. Node.js documentation. URL : https://nodejs.org/dist/latest-v8.x/docs/api/ (дата звернення: 28.04.2018).