НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050103 “Програмна інженерія’’

на тему: «Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі»

Виконав: студент 4 курсу, групи ТВз-51

Іванів Андрій Петрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викл. Гайдаржи Володимир Іванович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н. Побіровський Ю.М _____
(посада, 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 – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

Іваніву Андрію Петровичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі»
керівник роботи ст. викл. Гайдаржи В.І.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи персональний комп’ютер під керуванням операційної системи MacOS, мова програмування Typescript та JavaScript.
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) проаналізувати існуючі реєстри інформаційних ресурсів, проаналізувати вимоги до нової системи ведення реєстру, визначити переваги та недоліки сервісів які надають можливість зберігати данні в хмарні сховища, розробити схему об’єктів данних сховища, передбачити методи управління доступу до інформаційних ресурсів в хмарному середовищі, реалізувати графічний веб інтерфейс для авторизації користувачів та можливість перегляду данних з сховища, завантаження нових файлів, створення мульти вложенної ієрархії каталогів.
5. Перелік ілюстративного матеріалу 1. Мета роботи. 2. Електронні інформаційні ресурси. 3. Хмарні технології. 4. Amazon Web Services Simple Storage Service. 5. Amazon Web Services Congito та Identity Access Management. 6. Взаємодія сервісів AWS. 7. Структура проекта. 8. Форма авторизації користувача в систему. 9. Зразок

інтерфейсу завантаження файлів. 10. Зразок інтерфейсу кореневих каталогів системи. 11. Інтерфейс вкладених файлів та каталогів. 12. Висновки.

6. Публікації: XVII Міжнародна науково-практична конференція аспірантів, магістрантів і студентів "Сучасні проблеми наукового забезпечення енергетики", м.Київ, КПІ ім. Ігоря Сікорського, 23-26 квітня 2019 року, VI науково-практична дистанційна конференція молодих вчених і фахівців з розробки програмного забезпечення "Сучасні аспекти розробки програмного забезпечення",м. Київ, КПІ ім. Ігоря Сікорського, 31 травня 2019 року.

Дата видачі завдання "___"_____201___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Програмна реалізація системи		
5.	Оформлення пояснювальної записки		
6.	Захист програмного продукту		
7.	Передзахист		
8.	Захист		

Студент _____ Іванів А.П.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Гайджаржи В.І.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи — проаналізувати існуючі реєстри інформаційних ресурсів, розробити схему об’єктів даних сховища, передбачити методи управління доступу до інформаційних ресурсів в хмарному середовищі, реалізувати графічний веб інтерфейс для можливості перегляду даних сховища, завантаження нових файлів, створення мультивложенної ієрархії каталогів.. Для створення графічного інтерфейсу використано засоби фреймворку Angular 6 та мову програмування TypeScript. В основу вибраної технології покладено використання вбудованого в фреймворк Angular 6 компілятора Angular CLI, призначеного для створення виконуваних модулів програм. Результати дослідження доповідалися на двох наукових конференціях.

Записка містить 45 сторінок, 13 рисунків, 5 додатків і 18 посилань.

Ключові слова: РЕЄСТР ІНФОРМАЦІЙНИХ РЕСУРСІВ, ВЕБ ДОДАТОК, ANGULAR, TYPESCRIPT, AMAZON, SIMPLE STORAGE SERVICE.

ABSTRACT

The purpose of the work is to analyze the existing registries of information resources, to develop data warehouse schemas, to define methods for managing access to information resources in the environment, to implement a graphical web interface for the ability to view data storage, to download new files, creating a multi-cataloged catalog directory. A graphical user interface that uses the Angular 6 framework and TypeScript programming. The basis of the selected technology is the use of the Angular 6 compiler built-in Angular CLI framework, designed to create executable modules of the program. The results of the study were published at two scientific conferences.

The note contains 45 pages, 13 figures, 5 attachments and 18 links.

KEYWORDS: REGISTER OF INFORMATIONAL RESOURCES, WEB APPLICATION, ANGULAR, TYPESCRIPT, AMAZON, SIMPLE STORAGE SERVICE.

ЗМІСТ

ВСТУП

Однією з актуальних проблем інформаційних технологій останнього часу є використання систем електронних інформаційних ресурсів на базі хмарних технологій. Проведене в роботі дослідження було присвячене аналізу можливостей використання сервісів хмари Amazon Web Services для супроводження реєстру інформаційних ресурсів.

Для того щоб інформаційні ресурси були доступні вам або вашій команді на будь-якому комп'ютері або мобільному пристрої, то вам на допомогу можуть прийти хмарні сховища даних.

Хмарне сховище — модель схову даних, де цифрові дані зберігаються в логічні пули, фізичне зберігання охоплює кілька серверів (і часто на різних місцях (локаціях)), фізичне середовище, як правило, належить хостинговим компаніям, вони ж керують цим середовищем. Ці постачальники хмарних систем схову даних відповідають за схов наявної інформації й доступ до неї, та за роботу фізичного середовища. Користувачі купують у постачальників послуг хмарного сховища змогу зберігати там дані.

В умовах формування в країні основ інформаційного суспільства особливої важливості набуває розбудова Систем електронних інформаційних ресурсів, метою яких є створення інструменту, здатного підвищити ефективність використання інформаційних ресурсів та істотно знизити витрати на їх утримання за рахунок використання метаданих щодо інформаційних масивів, баз даних, реєстрів та інших видів інформаційних ресурсів[1].

ПОСТАНОВКА ЗАДАЧІ

Для розробки реєстру інформаційних ресурсів в хмарному середовищі необхідно:

- проаналізувати існуючі реєстри інформаційних ресурсів;
- проаналізувати вимоги до нової системи ведення реєстру;
- визначити переваги та недоліки сервісів які надають можливість зберігати данні в хмарні сховища;
- розробити схему об’єктів даних сховища;
- передбачити методи управління доступу до інформаційних ресурсів в хмарному середовищі;
- реалізувати графічний веб інтерфейс для авторизації користувачів та можливість перегляду даних сховища, завантаження нових файлів, створення мультивложенної ієрархії каталогів.

1. ЗАДАЧА РЕАЛІЗАЦІЇ РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ В ХМАРНОМУ СЕРЕДОВИЩІ

Реєстр інформаційних ресурсів повинен виконувати низку функцій, основними з яких:

- реєстрація та авторизація користувача в системі;
- формування груп доступу до ресурсів;
- завантаження ресурсу в сховище S3
- розбиття реєстру на категорії
- перегляд ресурсу за прямим посиланням;
- вивантаження ресурсу на локальний комп’ютер.

Також однією з основних цілей буда реалізація можливості управління доступом до електронних ресурсів групам користувачів.

1.1. Реєстр інформаційних ресурсів в хмарному середовищі

Електронні інформаційні ресурси — будь-які інформаційні ресурси, для відтворення яких необхідні електронні пристрої, не зважаючи на тип відтворюваної інформації, або її вміст.

Реєстр електронних інформаційних ресурсів — електронна система, в якій записи збираються, організовуються та класифікуються, щоб полегшити їх збереження, вилучення, використання та розпорядження.

Хмарні технології — це технології, які надають користувачам Інтернету доступ до комп’ютерних ресурсів сервера і використання програмного забезпечення як онлайн-сервіса[2].

Хмара — це деякий ЦОД (дата-центр, сервер) або їх мережа, де зберігаються дані та програми, що з'єднуються з користувачами через Інтернет.

Національним інститутом стандартів і технологій США (ANSI) зафіксовані такі обов'язкові характеристики хмарних обчислень:

- Самообслуговування по вимозі (self service on demand), споживач самостійно визначає і змінює обчислювальні потреби, такі як серверний час, швидкість доступу та обробки даних, обсяг збережених даних без взаємодії з представником постачальника послуг;
- Універсальний доступ мережою, послуги доступні споживачам по мережі передачі даних незалежно від використовуваного термінального пристрою;
- Об'єднання ресурсів (resource pooling), постачальник послуг об'єднує ресурси для обслуговування великої кількості споживачів в єдиний пул для динамічного перерозподілу потужностей між споживачами в умовах постійної зміни попиту на потужності; при цьому споживачі контролюють тільки основні параметри послуги (наприклад, обсяг даних, швидкість доступу), але фактичний розподіл ресурсів, що надаються споживачеві, здійснює постачальник (в деяких випадках споживачі все-таки можуть управляти деякими фізичними параметрами перерозподілу, наприклад, вказувати бажаний центр обробки даних з міркувань географічної близькості);
- Еластичність, послуги можуть бути надані, розширені, звужені в будь-який момент часу, без додаткових витрат на взаємодію з постачальником, як правило, в автоматичному режимі;
- Облік споживання, постачальник послуг автоматично обчислює спожиті ресурси на певному рівні абстракції (наприклад, обсяг збережених даних, пропускна спроможність, кількість користувачів, кількість транзакцій), і на основі цих даних оцінює обсяг наданих споживачам послуг.

1.2. Переваги зберігання реєстру інформаційних ресурсів в хмарному середовищі

Основні переваги зберігання реєстру інформаційних ресурсів в хмарному середовищі:

- не потрібні великі обчислювальні потужності ПК - по суті будь-який смартфон, планшет і т.д., при відкритті вікна браузера отримує величезний потенціал;
- відмовостійкість;
- певний рівень безпеки;
- висока швидкість обробки даних;
- економія на покупці софта - всі необхідні програми вже є в сервісі, де будуть працювати додатки;
- ваш власний вінчестер не наповнюється - всі дані зберігаються в мережі.

З точки зору споживача, ці характеристики дозволяють отримати послуги з високим рівнем доступності і низькими ризиками непрацездатності, забезпечити швидке масштабування обчислювальної системи завдяки еластичності без необхідності створення, обслуговування і модернізації власної апаратної інфраструктури[3]. Зручність і універсальність доступу забезпечується широкою доступністю послуг і підтримкою різного класу термінальних пристроїв (персональних комп'ютерів, мобільних телефонів, інтернет-планшетів).

Усі сервіси хмарних сховищ, розглянуті в цій темі, мають папки для робочого столу для Mac і ПК. Це дозволяє користувачам перетягувати файли між хмарою та їх локальним сховищем. Пропускна спроможність : ви можете уникнути надсилання файлів людям і надсилати посилання на одержувачів електронною поштою. Збережені файли можна отримати з будь-якого місця через підключення до Інтернету.

Дуже рекомендується, щоб підприємства мали план екстреного резервного копіювання у випадку кризи. Компанія може використовувати її як резервний план,

надаючи другу резервну копію важливих файлів. Ці записи зберігаються у віддаленому місці і можуть бути отримані через онлайн-з'єднання.

Підприємства та організації можуть часто знижувати річні експлуатаційні витрати за рахунок використання хмарних сховищ; Це коштує близько 3 центів за гігабайт для внутрішнього зберігання даних. Користувачі можуть спостерігати додаткову економію коштів, оскільки не вимагають внутрішньої можливості для віддаленого зберігання інформації.

2. АНАЛІЗ ПРОБЛЕМИ ЗБЕРІГАННЯ ДАНИХ РЕЄСТРУ В ХМАРНОМУ СЕРЕДОВИЩІ ТА УПРАВЛІННЯ ДОСТУПОМ ДО НИХ

AWS, Google Drive і OneDrive - найпопулярніші провайдери хмарних сховищ. Вибір між ними, щоб визначити, який з них кращий постачальник хмарних сховищ , не є легким завданням. У кожного є сильні та слабкі сторони, які не завжди збігаються[4].

Хмарні сховища зазвичай посилаються на службу зберігання об'єктів розміщення, але цей термін розширився, щоб включити інші типи сховища даних, які тепер доступні як служба, наприклад блочне сховище.

Служби сховищ об'єктів, такі як Amazon S3, Oracle Cloud Storage і Microsoft Azure Storage, програмне забезпечення для зберігання об'єктів, як OpenStack Swift, системи зберігання об'єктів, такі як EMC Atmos, EMC ECS і Hitachi Content Platform, а також дослідних проектів розподіленого зберігання, таких як OceanStore і VISION Cloud сховища, які можуть розміщуватися і розгортатися з характеристиками хмарного зберігання.

Доступ до служб зберігання хмар може здійснюватися через спільно розташовану службу хмарних обчислень, інтерфейс програмування веб-служб (API) або програми, що використовують API, такі як хмарне робоче сховище, шлюз хмарного сховища або веб-системи управління контентом.

Cloud - це служба, де дані віддалено підтримуються, керуються та резервуються. Служба дозволяє користувачам зберігати файли в Інтернеті, щоб вони мали доступ до них з будь-якого місця через Інтернет.

Зберігаючи данні на серверах Amazon, Google та низки інших необхідно розуміти що США не мають одного всеосяжного закону для регулювання даних по всій країні. Замість цього вона запровадила специфічні для сектору закони та

нормативно-правові акти, які працюють разом з законодавством на державному рівні, з метою збереження даних громадян, таких як HIPAA .

Постачальник послуг електронної комунікації або послуги віддалених обчислень повинен виконувати зобов'язання цієї глави щодо збереження, резервного копіювання або розкриття вмісту дроту або електронного зв'язку, а також будь-якого запису або іншої інформації, що належить клієнту або абоненту в межах власності такого постачальника , зберігання або контроль, незалежно від того, чи знаходиться таке повідомлення, запис або інша інформація в межах чи поза межами Сполучених Штатів[5].

Якщо ви - резидент США, ваші дані можна отримати, якщо ви розмістите їх за межами США в країні, "з якою США має виконавчу угоду". Конгрес виправдовує це в іншому розділі, стверджуючи наступне: "Своєчасний доступ до електронних даних, що зберігаються постачальниками послуг зв'язку, є важливим компонентом урядових зусиль для захисту громадської безпеки і боротьби з серйозними злочинами, включаючи тероризм".

Акт також надасть іноземним урядам (тим, хто має виконавчу угоду з США) право вимагати інформацію про своїх громадян від американських технологічних компаній. Відповідні технологічні компанії можуть скасувати цей запит протягом 14 днів, якщо вони вважають, що: "Клієнт або абонент не є особою Сполучених Штатів і не проживає в Сполучених Штатах і що необхідне розкриття може створити значний ризик того, що постачальник порушить закони кваліфікованого іноземного уряду".

Тобто потрібно мати на увазі, що на законодавчому рівні є ризики що доступ до електронних ресурсів системи отримають треті лиця.

2.1. Огляд провайдерів хмарних середовищ

Основна механіка спільного використання файлів однакова від одного постачальника до іншого, покладаючись на інтернет-посилання, які вказують на файли та папки, але посилання можуть бути небезпечними, якщо вони потрапляють в чужі руки. Щоб допомогти в управлінні контентом, можна використовувати додаткові функції, що пов'язують файли, такі як паролі та терміни закінчення терміну дії, хоча більшість провайдерів не в змозі це зробити[6].

Об'єкти, що зберігаються у OneDrive , можуть бути спільно використані за допомогою веб-інтерфейсу. До кожної папки та файлу додається кнопка спільного доступу.

Зберігання, синхронізація та обмін є найважливішими елементами будь-якої служби хмарних сховищ, але ви не станете кращими у виставці, дотримуючись основних трюків. Що відрізняє AWS, Google Drive і OneDrive від решти поля - це інтеграція додатків. Вони підвищують продуктивність праці та полегшують співпрацю[7].

Рішення AWS Amazon є світовим лідером, коли мова йде про SMB (Server Message Block) і пропонує розширені функції, такі як управління життєвим циклом і історія версій файлів.

AWS краще, ніж їх конкуренти, тому що вони не намагаються відтворити старі середовища центрів обробки даних. Конкуренти AWS вважають, що хмарні обчислення є комбінацією віртуалізації та автоматизації в існуючих центрах обробки даних. Причина в тому, що хмарні обчислення є зміною парадигми для IT. Це як перехід від мейнфреймів до клієнта / сервера. Весь IT-стек був переосмислений такими інтернет-гігантами, як AMZN, GOOG і Facebook. Цей перероблений IT-стек орієнтований на додатки наступного покоління, масштабованість і надійність. Це більш інтегрований і цілісний підхід до інфраструктури та додатків. І це нічого не схоже на існуючі центри даних.

Створюйте прості, примітивні служби, які є надійними і масштабованими (S3, EC2, SQS), а потім створюйте їх у послуги вищого порядку (RDS, EMR). Бізнес-

модель Amazon Web Services розрахована на показники вартості + ціни, з фіксованою маржею, що, оскільки вони знижують свої витрати, їх ціноутворення також зменшується.

Інженерна культура Amazon має довгострокове (2-3 роки) мислення і дорожні карти, а також широту роботи, що дозволяє швидко збігатися з функціональними випусками, вона зосереджена на простоті, безпеці, стійкості та масштабованості[8].

2.2. Огляд існуючих реалізацій ведення реєстру інформаційних ресурсів в хмарному середовищі

Як приклад реалізації реєстру інформаційних ресурсів було розглянуто додаток на базі Google Drive. Проте через неможливість достатньо гнучко керувати доступом користувачів до ресурсів було прийняте рішення змінити сервіс хмарного середовища.

Платформа Drive надає API, а також клієнтські бібліотеки, приклади, орієнтовані на мову, і документацію, які допоможуть розроблювати додатки, якіінтегруютсья з Диском. • Основна функціональність додатків Drive - завантажити та вивантажити файли на Google Диск. Проте платформа Drive забезпечує набагато більше, аніж просто зберігання, а саме загальний доступ до файлів, текстову індексацію файлів, OCR пошук по зображенням та інше.

Схема метаданих ресурсу (паспорт ресурсу):

• ідентифікаційний номер набору даних;

• найменування набору даних (до 254 символів);

• ключові слова, які відображають основний зміст набору даних;

• стислий опис змісту набору даних (до 4000 символів);

• відомості про мову інформації, яка міститься у наборі даних;

• формат (формати), в якому доступний набір даних;

• дату і час першого оприлюднення набору даних;

• дату і час внесення останніх змін до набору даних;

• гіперпосилання на набір даних (електронний документ для завантаження або інтерфейс прикладного програмування);

Головна відмінність Google Диск полягає в тому, що у синхронізованій папці також знаходяться файли Документів Google. Ресурси, такі як PDF-файли, завантажуються як є, тому їх можна відкрити безпосередньо. Рідні документи Google - це просто посилання на URL, які відкривають файл у веб-переглядачі. Хоча це практично, це означає, що не можна відкривати файли в автономному режимі, і немає реальної резервної копії у випадку проблем із підключенням або сервісом. Це один з головних недолік, але варіант для автоматичного перетворення на робочі формати, такі як TXT, RTF, DOC, ODT, XLS і PDF тут присутні.

3. ЗАСОБИ РОЗРОБКИ

Інтерфейс корстувача було реалізованно за допомогою фреймворка Angular 6 на мові TypeScript в операційній системі macOS Mojave та у середовищі розробки WebStorm. Технологія використання реєстру полягає в наступному: корисувачу необхідно виконати реєстацію в системі, за допомогою сервісу Cognito, доступ до директорій та файлів сховища налаштовується в сервісі IAM. Підчас авторизації користувач отримує ключі доступу та індитифіказії з Cognito User Pool. Після цього, використовуючи ключ ідентифікації користувач отримує тимчасові повноваження AWS з Cognito Identity Pool. Таким чином за допомогою технологій AWS користувач веб-додаткуотримує доступ до операцій над ресурсами хмарного середовища.

3.1. Середовище розробки Web Storm

JetBeans розробив WebStorm як висококласний IDE для проектів веб-розробки. Крім підтримки HTML, CSS і JavaScript, WebStorm також підтримує кілька широко використовуваних фреймворків JavaScript, включаючи Angular, NodeJS, React і Meteor. JetBeans додатково сприяє тому, що WebStorm є саме JavaScript IDE.

Серед основних переваг: допомога для написання коду, дозволяє програмістам аналізувати код, написаний на різних підтримуваних мовах, дозволяє користувачам миттєво перевіряти вплив змін.

Налагодження в WebStorm все зроблено за допомогою графічного інтерфейсу (Рис. 1). Є всі комбінації клавіш для всіх важливих функцій. Деякі з них, звичайно, різні, тому що це різні програми, але для людей, які хотіли б перейти від WebStorm до Visual Studio Code, є навіть плагін для останнього ("intellij-idea-keybindings"). На жаль, деякі ярлики (наприклад, одне з моїх улюблених - CTRL + W для розширення вибору) прекрасно працюють у файлах JS або TypeScript, але не в HTML. Сподіваюся, він буде покращений.

Багато, якщо не всі, розширені функції редактора, такі як численні курсори або імена рефакторингу змінних, підтримуються в Visual Studio Code. Це повнофункціональний редактор.

Такоє він має підтримку GIT і він простіше у використанні. Майже в кожній команді є еквівалент GUI, однак у деяких випадках використання терміналу все ще є необхідним. Одним з найбільш корисних інструментів, на мій погляд, є інструмент вирішення конфліктів злиття.

WebStorm пропонує підтримку управління проектами. Він може, наприклад, бути інтегрований з JIRA, який може допомогти перемикати завдання, реєструвати час, витрачений на конкретне завдання, і т.д. У Visual Code Code таких параметрів немає.

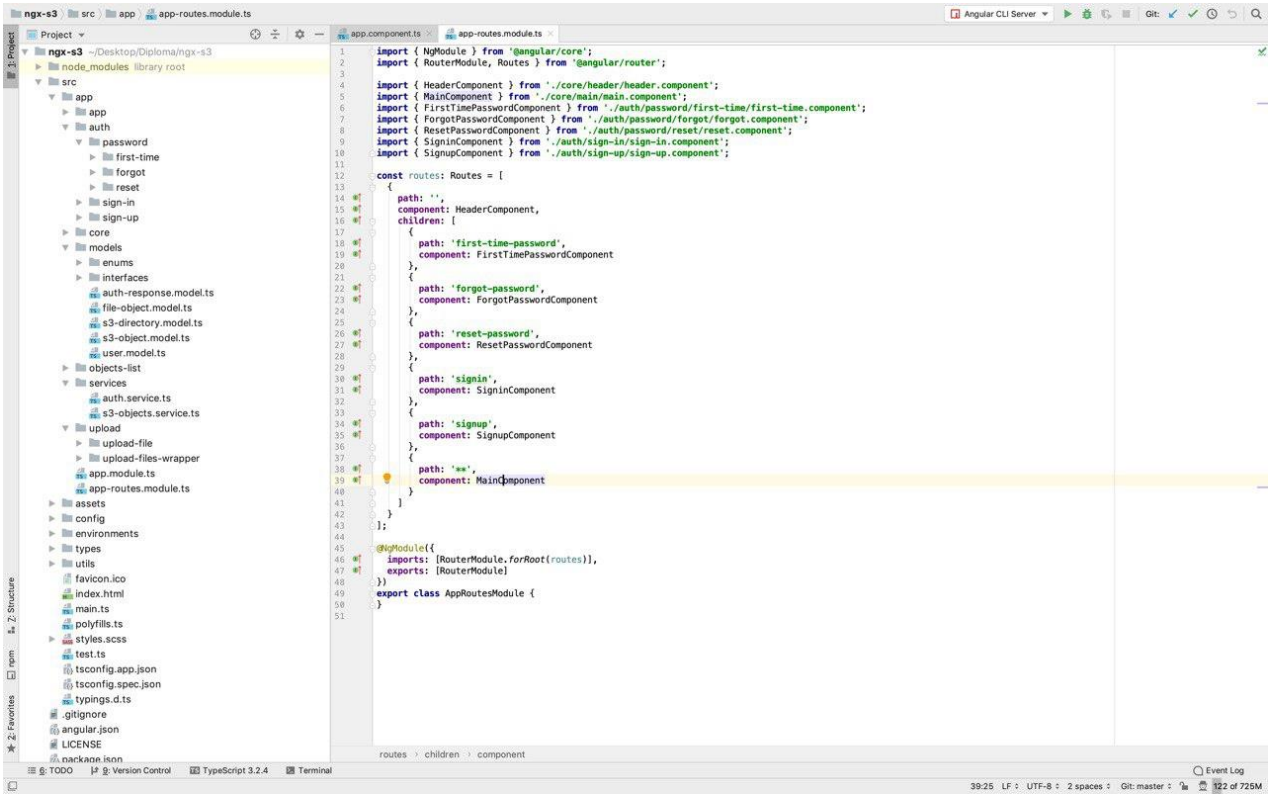

Рис. 1. Інтерфейс середовища розробки WebStorm

Усі налаштування VSC необхідно редагувати вручну. При виборі опції налаштувань у меню просто відкривається файл налаштування у форматі JSON. WebStorm можна повністю налаштувати за допомогою графічного інтерфейсу і мати набагато більше налаштувань для гри.

Наразі в Visual Studio Code одночасно можна відкривати лише до трьох файлів, а лише як один з одним. Показ файлів на вкладках, як і в багатьох інших середовищах розробки, не підтримується, але, за офіційним веб-сайтом, робота над цією функцією вже розпочалася. WebStorm відкриває файли на вкладках і має деякі корисні ярлики для переміщення по них.

Вбудовані інструменти тестування, що входять до складу WebStorm, дозволяють програмістам виконувати модульне тестування без додаткового часу та зусиль[9]. Програмісти можуть перевірити JavaScript-код на стороні клієнта за допомогою програми Karma і протестувати NodeJS з Mocha. Крім того, WebStorm дозволяє їм запускати тестові одиниці безпосередньо в середовищі IDE і переглядати результати тестів у візуальному форматі.

3.2. Мова програмування TypeScript

TypeScript — мову програмування, що компілюється в JavaScript і розроблена спеціально для великих додатків. Вона багато перейняла від таких мов як Java та C#, які є більш дисциплінованими та строгими, ніж динамічно типізований JS.

TypeScript є набором JavaScript, який в першу чергу надає необов'язкові статичні типи, класи і інтерфейси (Рис. 2). Одна з великих переваг полягає в тому, щоб дозволити інтегрованим середовищам розробки забезпечити більш насичене середовище для виявлення поширених помилок під час введення коду. Для великого проекту JavaScript, прийняття TypeScript може призвести до більш надійного програмного забезпечення, в той же час розгортається там, де виконується звичайна програма JavaScript.

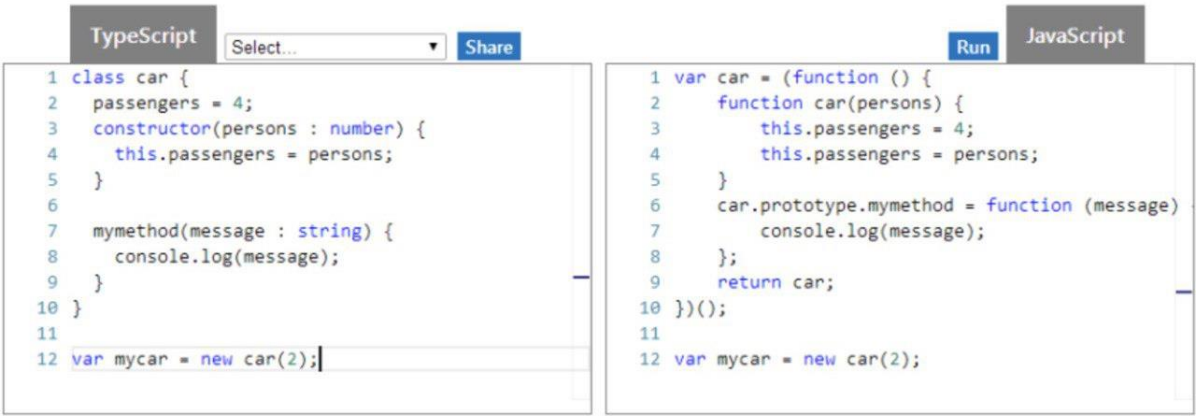

Рис. 2. Транспіляції коду з мови TypeScript в JavaScript

Це відкритий код, але ви отримуєте тільки розумний Intellisense, як ви вводите, якщо ви використовуєте підтримувану IDE. Спочатку це була лише Microsoft Visual Studio (також відзначена в блозі від Miguel de Icaza). У наші дні інші IDE також пропонують підтримку TypeScript .

Схожою технологією є CoffeeScript , але це він має іншу мету. Проте CoffeeScript забезпечує доступність для людей, але TypeScript також забезпечує глибоку читабельність для інструментів за допомогою додаткового статичного набору тексту[10]. Є також Dart, але це повна заміна для JavaScript (хоча вона може створювати код JavaScript).

Із-за статичної типізації код на TypeScript більш передбачуваний і тому його легко дебажити. Завдяки гарній підтримці ООП, модулів та просторів імен, проект, навіть при великому об'ємі, буде впорядкованим. Компілятор на стадії збирання знаходить багато помилок, ще до того як вони потраплять в рантайм і зламають щось. Починаючи з 2 версії Angular написана на TypeScript (далі — TS) і рекомендує його як мову розробки[11].

3.3. Фреймворк реалізації веб-інтерфейсу Angular 6

Angular 6 - найкращий фреймворк з відкритим вихідним кодом для побудови веб- та мобільних додатків за архітектурою мікро-служб і був задуманий як мобільний перший підхід.

У Angular 2, контролери і $scope були замінені компонентами і директивами. Компоненти - це директиви з шаблоном. Вони стосуються перегляду програми та логіки на сторінці. У Angular 2 існує два види директив. Це структурні директиви, які змінюють компонування DOM шляхом видалення та заміни його елементів, а також атрибутивних директив, які змінюють поведінку або зовнішній вигляд елемента DOM.

Angular 6 використовує розширений JavaScript, який дуже оптимізований, що робить додаток і веб-завантаження швидше. Angular2 швидко завантажується за допомогою компонентного маршрутизатора. Це допомагає в автоматичному розбитті коду, тому користувачеві завантажувати код, необхідний для постачальника перегляду. Багато модулів видаляються з кутового ядра, в результаті чого краща продуктивність означає, що ви зможете вибрати і вибрати ту частину, яка вам потрібна. Багато модулів видаляються з кутового ядра, в результаті чого краща продуктивність означає, що ви зможете вибрати і вибрати ту частину, яка вам потрібна.

3.4. Сервіси Amazon Web Services

Amazon Simple Storage Service (Amazon S3) - це сервіс зберігання об'єктів, що пропонує кращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних[12]. Це означає, що нашими клієнтами можуть бути компанії будь-яких розмірів і з будь-яких областей діяльності. Вони можуть використовувати наш сервіс для зберігання і захисту будь-яких обсягів даних в різних ситуаціях, наприклад для забезпечення роботи сайтів, мобільних додатків, для резервного копіювання та відновлення, архівації, корпоративних додатків, пристроїв IoT і

аналізу великих даних. Amazon S3 пропонує прості у використанні інструменти адміністрування, які дозволяють організувати дані і точно налаштувати обмеження доступу відповідно потребами вашого бізнесу або законодавчими вимогами. Amazon S3 забезпечує надійність 99%.

Веб-сервіси Amazon (скорочено AWS) - це набір віддалених обчислювальних послуг (також званих веб-службами), які разом складають платформу хмарних обчислень, що пропонується через Інтернет за допомогою Amazon . com . Найбільш центральними і відомими з цих послуг є Amazon EC2 і Amazon S3.

AWS - це набір продуктів для хостингу, які спрямовані на те, щоб позбутися головного болю від традиційних хостингових рішень. Такі послуги, як Dropbox і сайти, такі як Reddit, використовують AWS. Насправді, ми відчуваємо, що знаходимося в доброму сусідстві, будучи на AWS. Я склав кілька причин для вибору AWS і пояснив їх тут. Тож давайте зануримося і зрозуміємо, чому AWS краще, ніж конкуренція, для великих і малих користувачів.

Amazon взяв освіжаючий підхід до ціноутворення свого хостингу при запуску AWS. Ви платите за те, що ви використовуєте. Це робить багато сенсу для серверної інфраструктури, оскільки трафік має тенденцію бути дуже лопневим, особливо чим більше сайт. Традиційне обладнання, здебільшого, не використовується для 90% його життєвого циклу. AWS допомагає впоратися з цією проблемою, зберігаючи її дешево в повільні часи.

Не можна заперечувати швидкість AWS. Еластичний блок зберігання майже так само швидко, як S3, але надає різні функції. Обчислювальні блоки EC2 надають продуктивність класу Xeon за погодинною ставкою. Надійність краще, ніж більшість приватних центрів обробки даних у світі, і якщо виникає проблема, ви зазвичай залишаєтеся в мережі, але зі зниженою продуктивністю. Це перевірено за допомогою красивої програми Chaos Monkey, де за допомогою цієї програми вона випадково відключає компонент у вашому хмарному середовищі. Тоді ви можете, чи ваша програма ще працює, або якщо вона повністю знищена. Таким чином, у нашому випадку мавпа хаосу збила нашу базу даних і веб-сервер. База даних, яка була службою RDS, відразу ж перейшла на іншу базу даних, використовуючи

функцію Multi AZ, як обіцяв AWS. У сценарії веб-сервера, коли один веб-сервер був знижений, інший веб-сервер був запущений за допомогою функції автомасштабування, тому ми нарешті зробили висновок, що AWS забезпечує високу продуктивність, як вони обіцяли.

Якщо вам коли-небудь доводилося надавати веб-службу, ви дуже добре знаєте цей біль. Традиційні провайдери займають від 48 до 96 годин для надання серверу. Тоді вам доведеться витратити кілька годин на її налаштування і перевірити все. AWS скорочує час розгортання до хвилин. Якщо ви використовуєте свої Amazon Machine Images, ви можете мати машину, розгорнуту і готову прийняти з'єднання в такий короткий проміжок часу. Це важливо, коли, наприклад, ви запускаєте акцію, яка генерує тон трафіку через певні проміжки часу, або просто потрібна гнучкість для задоволення попиту, коли новий продукт запускається.

AWS також надає VPC, яка може використовуватися для розміщення наших послуг у приватній мережі, яка не доступна з Інтернету, але може спілкуватися з ресурсами в одній мережі. Це обмежує доступ до ресурсів таким чином, щоб будь-який зловмисний користувач з Інтернету.До цих ресурсів, розміщених у приватній мережі, можна звертатися за допомогою Amazon VPN або деяких служб з відкритим вихідним кодом, таких як OpenVPN.

Найважливішою особливістю AWS є його гнучкість. Всі служби працюють і спілкуються разом з вашою заявкою, щоб автоматично оцінювати попит і обробляти її відповідно.

У поєднанні з фантастичними API та зображеннями машини Amazon, які ви створюєте, ви можете мати повністю налаштоване рішення, яке забезпечує примірник сервера за 10 хвилин, і готові приймати з'єднання, як тільки він з'явиться в Інтернеті. Тоді ви можете швидко вимкнути випадки, коли вони більше не потрібні, роблячи управління сервером справою минулого.

Сервіс Amazon Cognito дозволяє швидко і просто додавати можливості реєстрації, авторизації і контролю доступу користувачів в мобільні та інтернет-додатки. Amazon Cognito масштабується до мільйонів користувачів і підтримує авторизацію за допомогою соціальних постачальників посвідчень (Facebook, Google,

Amazon), а також постачальників корпоративних посвідчень на основі SAML 2.0[13].

Сервіс AWS Identity and Access Management (IAM) надає можливості безпечного управління доступом до сервісів та ресурсів AWS. Використовуючи IAM, можна створювати користувачів AWS і групи, керувати ними, а також використовувати дозволу, щоб надавати або забороняти доступ до ресурсів AWS. IAM - це можливість аккаунта AWS, яка надається без додаткової оплати. Плата стягується лише за використання створеними користувачами інших сервісів AWS.

4. ОПИС ПОБУДОВАНОЇ МОДЕЛІ РЕЄСТРУ ТА МЕТОДУ РОЗВ'ЯЗАННЯ ПОСТАВЛЕННОЇ ЗАДАЧІ

Побудована модель реєстру передбачає зберігання інформаційних ресурсів та відповідних мета данних у хмарному середовищі Amazon з використання технологій AWS, S3, IAM та Congito.

4.1. Опис побудованого інтерфейсу взаємодії системи ти хмарного середовища AWS

У Cognito є пули користувачів і пули ідентифікації. Пули користувачів для підтримки користувачів і пулів ідентифікаторів призначені для створення тимчасових облікових даних AWS, використовуючи кілька ідентифікаторів веб-сайтів, включаючи ідентифікацію користувача Cognito.

Ми створили пул користувачів у Cognito і пов'язали його з пулом ідентифікаторів. Пул ідентичності надає облікові дані користувачам, що пройшли аутентифікацію, та користувачам, що не пройшли перевірку автентичності, на основі пов'язаних ролей та політик IAM. Тепер будь-який дійсний користувач у нашому користувальницькому пулі Cognito може отримати тимчасові облікові дані AWS (Рис. 3), використовуючи асоційований пул ідентифікації, і використовувати ці тимчасові облікові дані для прямого завантаження файлів на S3.

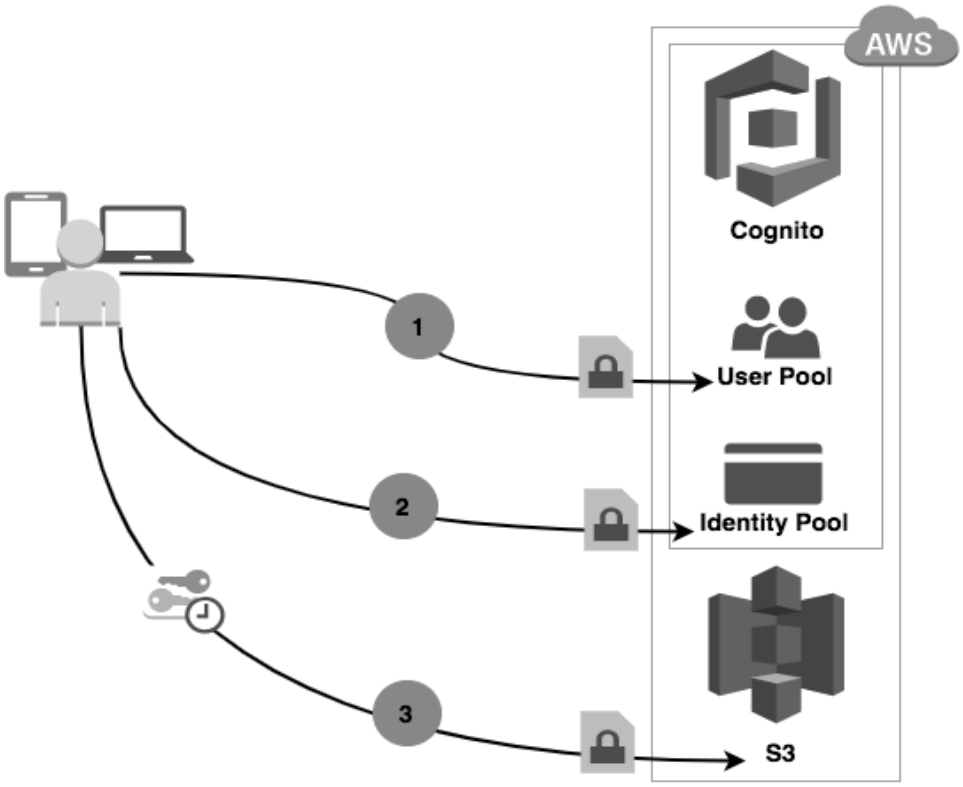

Рис. 3. Схема взаємодії системи та сервісів AWS

Таким чином реалізований звязок між різними сервісами Амазон та програмним продуктом. Данний підхід дозволяє управляти доступом до електронних ресурсів сховища різним користувачам.

4.2. Методи розв’язання поставленної задачі

Для реалізації поставленої задачі було проаналізовано існуючі реєстри інформаційних ресурсів та визначені переваги та недоліки сервісів які надають можливість зберігати данні в хмарних сховищах. В якості постачальника хмарних техноголій булообрано Amazon Web Services через гнучкість систем та можливості взаємодії між ними. Проаналізувавши вимоги до нової системи ведення реєстру інформаційних ресурсів було розроблено схему об’єктів данних сховища. Передбачені методи управління доступу до інформаційних ресурсів в хмарному середовищі Amazon Web Services Simple Storage Service за допомогою сервіса Identity Access Management.

5. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЗАЄМОДІЇ СЕРВІСІВ AMAZON WEB SERVICES І ВЕБ-ДОДАТКУ У ВИПАДКУ РЕАЛІЗАЦІЇ РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ

Реєстр інформаційних ресурсів повинен виконувати низку функцій, основними з яких: реєстація та авторизація користувача в системі, формування груп доступу до ресурсів, завантаження ресурсу в сховище S3, розбиття реєстру на категорії, перегляд ресурсу за прямим посиланням та вивантаження ресурсу на локальний комп’ютер.

Для реалізації всіх поставленних задач була обранна ServerLess архітектура[14]. В якості сервісної частини повністю використовувалися сервіси Amazon Web Services.

5.1. Створення веб-додатку за допомогою Angular 6

Для створення проекту потрібно використовувати Angular CLI. Він полегшує створення коду додатків і бібліотеки, а також виконання різноманітних поточних завдань розробки, таких як тестування, комплектація та розгортання[15].

Angular CLI встановлює необхідні пакети Angular npm та інші залежності. Angular включає в себе сервер, так що ви можете легко створювати і обслуговувати вашу програму локально. Команда “ng serve” запускає сервер, стежить за файли і перебудовує додаток , коли ви зробите зміни в цих файлах. Параметр --open(або просто -o) автоматично відкриває ваш браузер за адресом http://localhost:4200/.

5.2. Підключення бібліотеки AWS SDK і її використання

Щоб почати роботу з AWS в найкоротші терміни, необхідно використовувати AWS SDK для JavaScript у середовищі Node.js. SDK дозволяє спростити написання програмного коду завдяки об'єктам JavaScript для різних сервісів AWS, включаючи

Amazon S3, Amazon EC2, DynamoDB і Amazon SWF (Рис. 4). Окремий завантаження пакет містить в собі бібліотеку класів AWS JavaScript і документацію.

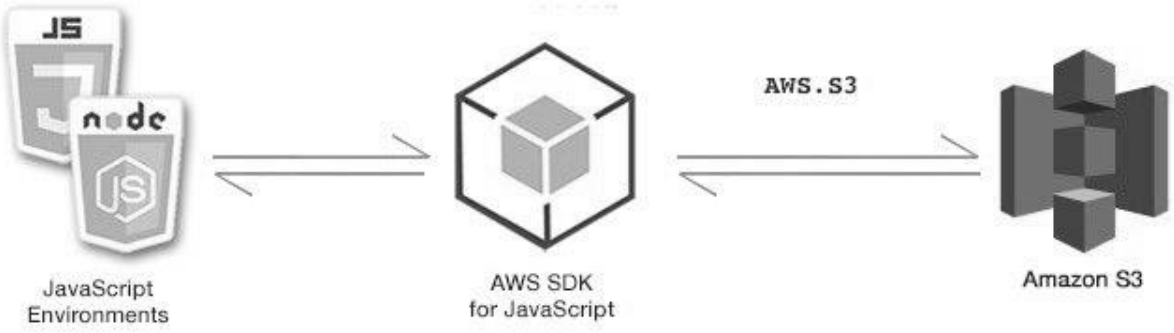

Рис. 4. Схема взаємодії системи, AWS SDK та сервісів AWS

Щоб встановити AWS SDK необхідно виконати команду “npm install aws-sdk”. Для налаштування створіть файл даних доступу за адресою ~ / .aws / credentials в ОС Mac / Linux або C: \ Users \ USERNAME \ .aws \ credentials в ОС Windows, в якому aws_access_key_id = ваш_ключ_доступа, а aws_secret_access_key = ваш_секретний_ключ У керівництві по початку роботи описані інші засоби завантаження даних доступу.У назві сховищ можуть бути присутніми малі літери, цифри і дефіс. Назва кожної мітки повинно починатися і закінчуватися на малу літеру або цифру.

5.3. Реалізація авторизації користувача в системі

Для реєстрації користувача в системі необхідно створити CognitoUserPoolоб'єкт, надавши UserPoolIdта та ClientId, і використовувати ім'я користувача, пароль, список атрибутів і дані перевірки. Приклад графічного інтерфейсу користувача зображений на Рис. 5.

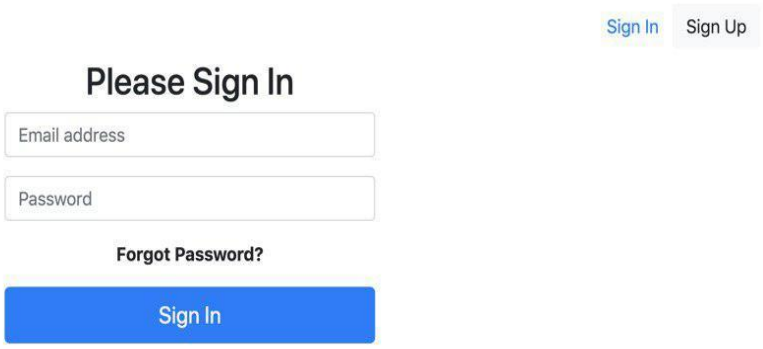

Рис. 5. Форма авторизації користувача в систему

Щоб отримати користувача з локальної пам’яті неаобхідно отримати його ключі доступу за допомогою набору для розробки програмного забеспечення Amazon Web Services Software Development Kit.

5.4. Реалізація управління данними в хмарному середовищі

Зв’язок між веб-додатком та сховищем AWS S3 реалізовано за допомогою AWS SDK та власними сервісами для адаптації роботи з системою. Основними моделями в роботі з сховищеми є модель файла та модель каталогу. Обидві моделі на стороні хмарного сховища преобразуються в єдиний вид – об’єкт сховища[16].

Модель файла на стороні веб-додатку має наступний вигляд:

```
export class S3ObjectModel {
  static acceptedFileTypes = ['jpeg', 'jpg', 'png'];
  key: string;
  url: string;
  year: string;
  month: string;
  day: string;
  size: string;
  name: string;
```

```
  type: string;
  preview: boolean;
}
```

Взаємодія з сховищем S3 для виконання основних задач системи (отримання списку обєктів в сховищі, створення нових каталогів та завантаження файлів) реалізована за допомогою данного сервіса.

6. МЕТОДИКА ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Щоб реалізувати доступ до данних хмари лише авторизованим корисувачам, було розробленно авторизацію.

Після успішної авторизації корисува опиняється в коренемову каталозі системи (Рис. 6), в якій він має змогу побачити групи, до який йому надав доступ адміністратор системи.

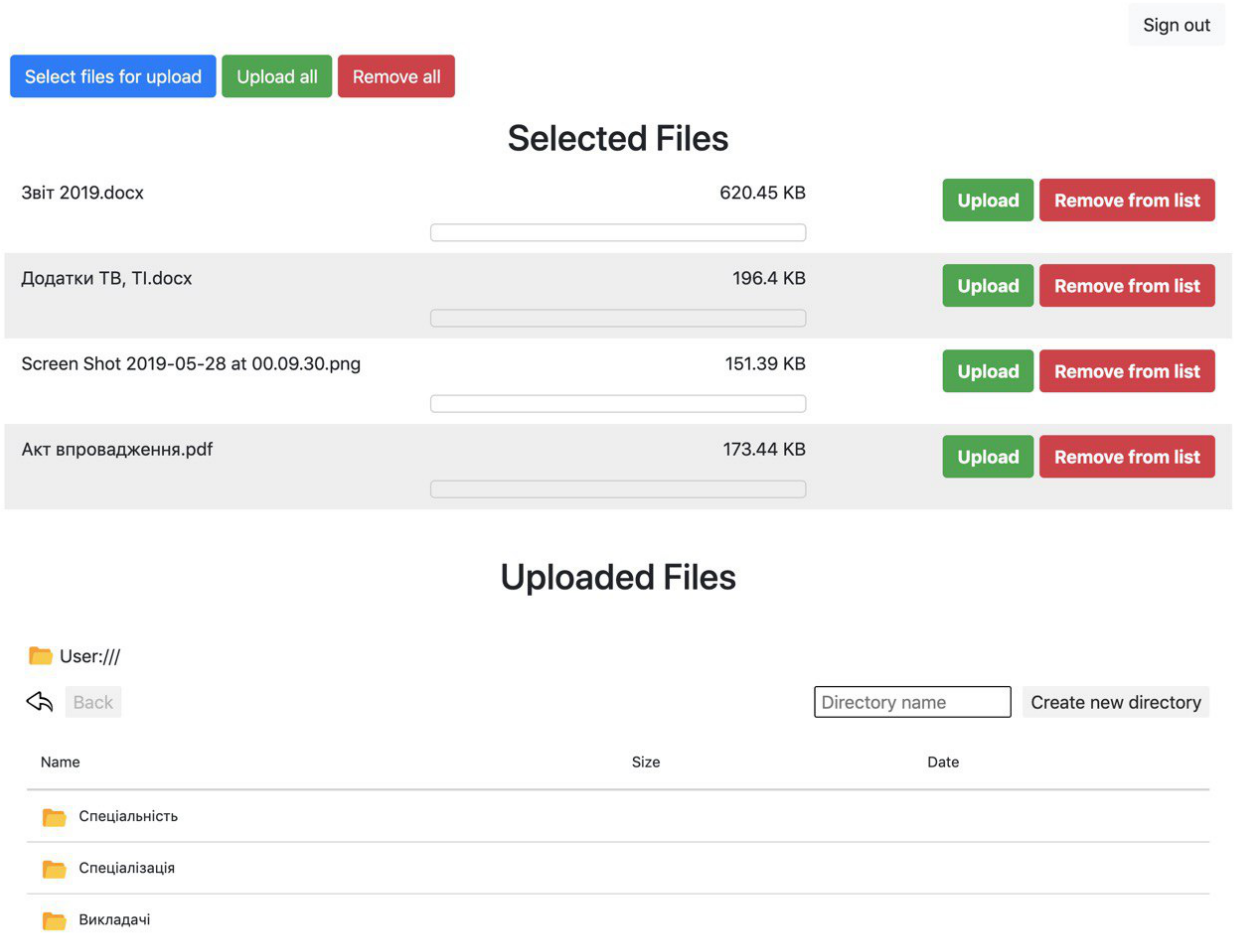

Рис. 6. Інтерфейс користувача в кореневій дерикторії

Після того як користувач зайде в одну з кореневих папок, він має змогу завантажувати файли в сховище, створювати вложені каталоги та переглядати

раніше завантажені файли та каталоги в системі (Рис. 7). Також користувач має змогу повернутися на один рівень вверх по дереву каталогів кнопкою “Back”.

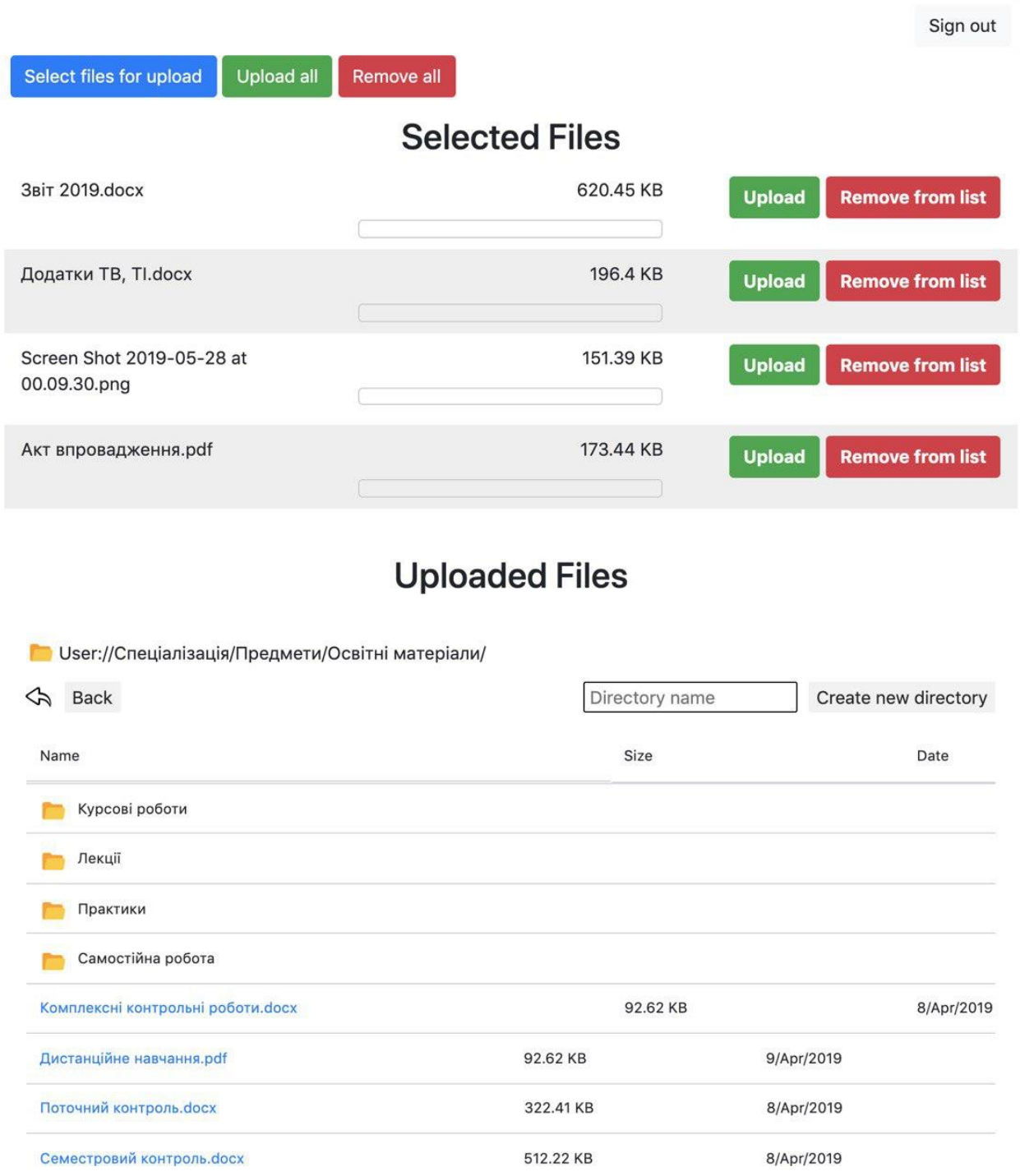

Рис. 7. Інтерфейс відображення файлів та каталогів системи

Для використання реалізованого веб-додатку реєстру інформаційних ресурсів необхідно мати встановленним будь-який Інтернет браузер. Потрібно переконатися, що ваша система відповідає данним вимогам. Хоча це, як правило, не є проблемою,

коли мова йде про вимоги до апаратних засобів, ви можете помітити, що це, наприклад, зовсім інша історія, коли мова йде про підтримувані операційні системи. Користувачі Firefox у Windows 2000, наприклад, помітять, що вони не зможуть оновитись з Firefox 12 на 13 у найближчому майбутньому, оскільки Mozilla знизила підтримку цієї операційної системи, починаючи з цієї версії браузера.

6.1. Технічні вимоги до середовищ використання

На данний момент найпопулярнішим браузером є Google Chrome. Підтримка браузера операційними системами починається з Windows XP SP2, OS X 10.5.6, Ubuntu 10.04, Debian 6, OpenSuse 11.3 та Fedora Linux 14(Рис. 8).

Operating System	Internet Explorer 11	Internet Explorer 10	Internet Explorer 9	Internet Explorer 8	Internet Explorer 7	Internet Explorer 6	Firefox 12+	Safari 4+	Chrome 12+
Windows 8.1 Desktop	✓*	-	-	-	-	-	✓	-	✓
Windows 8 Desktop	-	✓*	-	-	-	-	✓	-	✓
Windows Server 2012 R2	✓*	-	-	-	-	-	✓	-	✓
Windows Server 2012	-	✓*	-	-	-	-	✓	-	✓
Windows 7	-	-	✓*	✓*	-	-	✓	-	✓
Windows 7 SP1	✓*	✓*	✓*	✓*	-	-	✓	-	✓
Windows Server 2008 SP2	-	-	✓	✓	✓	-	✓	-	✓
Windows Server 2008 R2 SP1	✓*	-	✓*	✓*	-	-	✓	-	✓
Windows Vista SP2	-	-	✓	✓	✓	-	✓	-	✓
Windows Server 2003 SP2, Windows XP SP3	-	-	-	✓	✓	-	✓	-	✓
Macintosh OS 10.5.7+ (Intel-based)	-	-	-	-	-	-	✓	✓	-

Рис. 8. Підтримка браузерів операційними системами

Мінімальними вимогами для процессорів, які запускаються браузер на операційні системі Windows є Internet Explorer 8. Він потребує 233 Mhz процесор;

мінімум 64 MB RAM (для Windows XP), рекомендовано 512 MB RAM; мінімум 150 MB (Windows XP), 70 MB (Windows Vista) вільного місця на диску.

В свою чергу браузер Google Chrome потребує процесор Pentium 4 для операційної системи Windows, та прецессор Intel для операційної системи Mac ; мінімум 128 MB RAM; мінімум 100 MB вільного місця на диску.

6.2. Інструкція по налаштуванню середовища

Щоб створити свій обліковий запис Amazon перейдіть на домашню сторінку веб-служб Amazon https://aws.amazon.com/ та виберіть "Sign Up" . Примітка: якщо нещодавно ви ввійшли в AWS, кнопка може мати текст “Sign In to the Console”. Введіть дані свого облікового запису, а потім виберіть “Continue”. Важливо: переконайтеся, що ви правильно ввели інформацію про обліковий запис, особливо адресу електронної пошти. Якщо неправильно ввести адресу електронної пошти, ви не зможете отримати доступ до свого облікового запису. Якщо кнопка ” Create a new AWS account" не відображається, спочатку виберіть “Sign in to a different account” , а потім виберіть " Create a new AWS account" . Виберіть Особистий або Професійний тарифні плани. Примітка: особисті рахунки та професійні рахунки мають однакові функції та набір інструментів. Введіть інформацію своєї компанії або особисту інформацію (Рис. 9). Прочитайте та прийміть угоду з клієнтом AWS https://aws.amazon.com/agreement/. Примітка: Переконайтеся, що ви прочитали та зрозуміли умови Угоди з клієнтами AWS. Нажміть на кнопку “Create Account and Continue” .

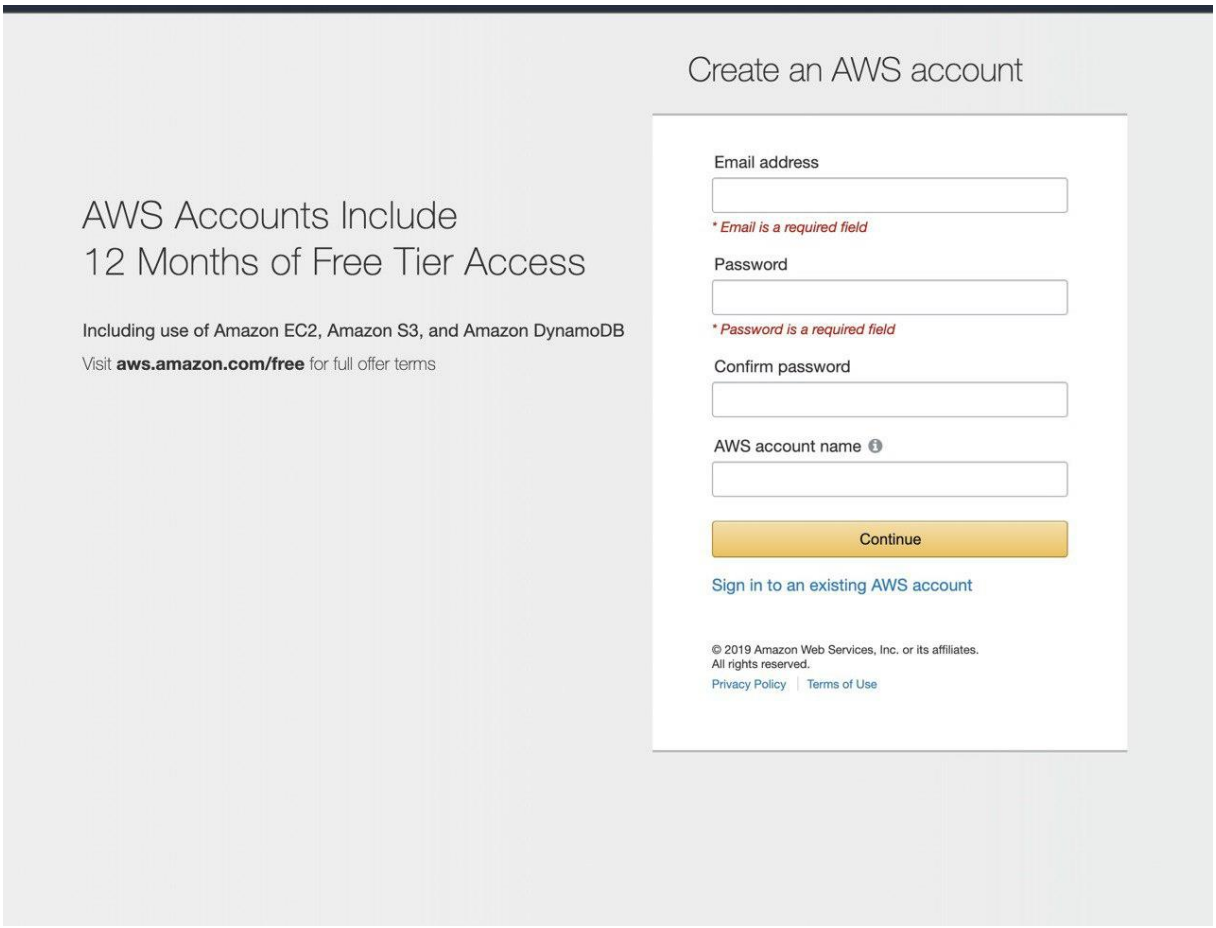

Рис. 9. Форма реєстрації AWS

Ви отримали повідомлення електронної пошти, щоб підтвердити, що ваш обліковий запис створено. Ви можете увійти до нового облікового запису, використовуючи надану адресу електронної пошти та пароль. Однак ви не можете користуватися послугами AWS, доки не завершите активацію свого облікового запису. Щоб активувати свій акаунт на сторінці “Payment Information “ введіть інформацію про спосіб оплати, а потім виберіть “Secure Submit” . Примітка. Якщо ви хочете використовувати іншу адресу для облікового запису AWS, виберіть “Use a new address”, перш ніж нажати “Secure Submit” .

Перевірте свій номер телефону. Виберіть, чи потрібно перевірити свій обліковий запис за допомогою текстового повідомлення (SMS) або голосового виклику. Виберіть код країни або регіону зі списку. Введіть номер телефону, до якого ви зможете дістатися протягом наступних хвилин. Введіть код, що відображається в captcha. Коли ви будете готові, оберіть “Contact me”. Через кілька

хвилин автоматизована система зв'яжеться з вами. Примітка: якщо ви вирішили підтвердити свій обліковий запис за допомогою SMS, виберіть “Send SMS” натомість.

Введіть PIN-код, отриманий за допомогою текстового повідомлення або голосового дзвінка, а потім нажміть кнопку “Continue”.

Виберіть план підтримки AWS. На сторінці “Select a Support Plan” виберіть один з доступних планів підтримки. Опис доступних планів підтримки та їх переваг дивитися у розділі Порівняння планів підтримки AWS https://aws.amazon.com/premiumsupport/features/.

Дочекайтеся активації облікового запису, Після вибору плану підтримки, сторінка підтвердження вказує, що ваш обліковий запис активовано. Облікові записи зазвичай активуються протягом декількох хвилин, але процес може тривати до 24 годин.

Ви можете увійти до свого облікового запису AWS протягом цього часу. На домашній сторінці AWS може відображатися кнопка, яка відображає "Повна реєстрація" протягом цього часу, навіть якщо ви виконали всі дії в процесі реєстрації. Коли ваш обліковий запис буде повністю активовано, ви отримаєте електронний лист із підтвердженням. Після отримання цього листа ви маєте повний доступ до всіх служб AWS.

Щоб створити сховище AWS S3 увійдіть до консолі керування AWS і відкрийте консоль Amazon S3 на сторінці https://console.aws.amazon.com/s3/ . Нажміть кнопку “Create bucket” .

У полі “Bucket name” введіть унікальне ім'я DNS для вашого нового сегмента (Рис. 10). Назва повинна бути унікальною для всіх існуючих імен сегментів у Amazon S3. Після того як ви створили сховище, ви не зможете змінити ім'я, тому вибирайте розумно.

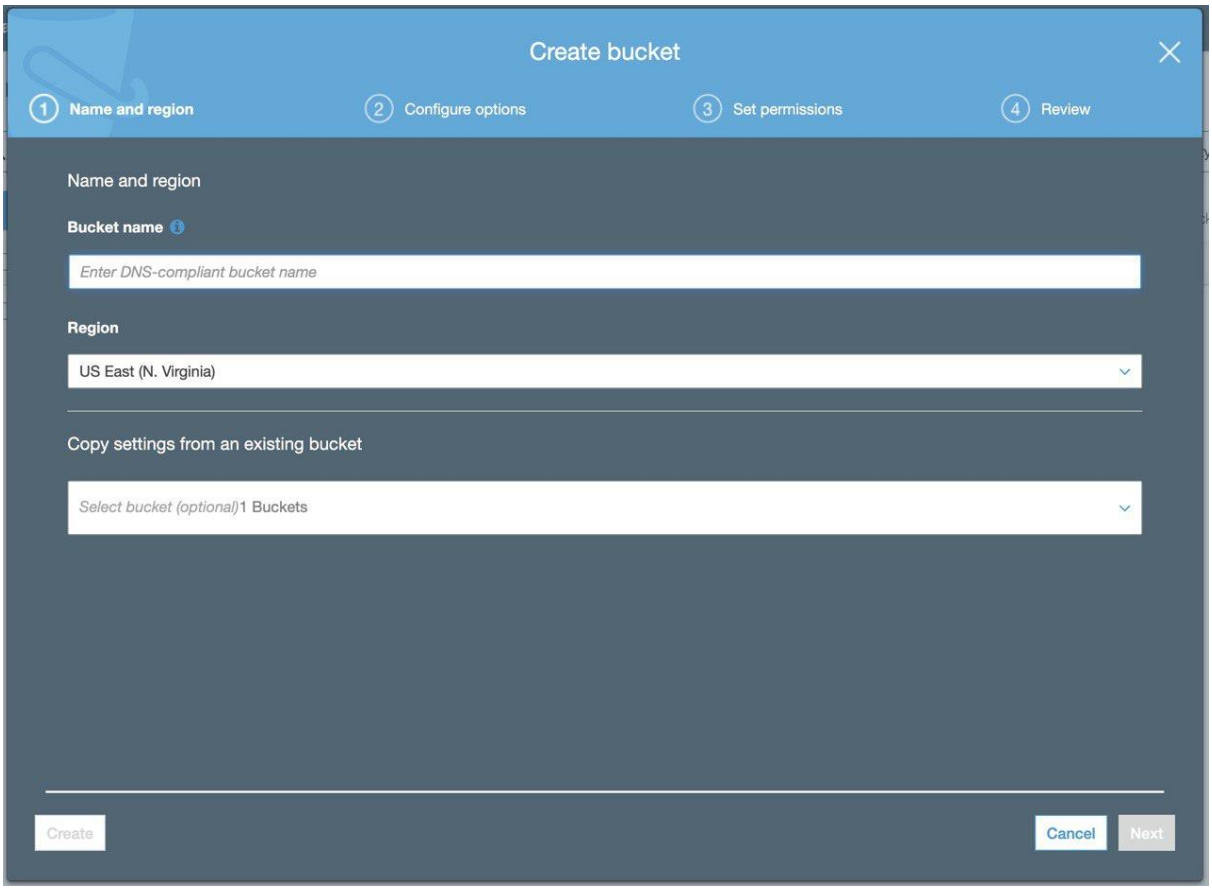

Рис. 10. Вікно створення сховища AWS S3

Виберіть назву таку, яка відображатиме об'єкти у сегменті, оскільки назва URL-адреси відображається в URL-адресі, який вказує на об'єкти, які ви збираєтеся розмістити у вашому сегменті. Для отримання відомостей про іменування сегментів дивитися Rules for bucket naming https://docs.aws.amazon.com/AmazonS3/latest/dev//BucketRestrictions.html#bucketnamingrules в керівництві розробників служби Amazon Simple Storage Service .

Для регіону виберіть US West (Oregon) як регіон, де потрібно розміщувати сегмент. Нажміть кнопку " Create", після цього в списку сховищ Amazon S3 зявиться новий запис.

Увімкніть CORS на сховищі S3 з наступними правами CORS (Рис. 11):

<? xml version = " 1.0 " encoding = " UTF-8 " ?>

< CORSConfiguration xmlns = " http://s3.amazonaws.com/doc/2006-03-01/ " >

< CORSRule >

< AllowedOrigin > * </ AllowedOrigin >

< AllowedMethod > PUT </ AllowedMethod >

< AllowedMethod > GET </ AllowedMethod >

< AllowedMethod > POST </ AllowedMethod >

< AllowedMethod > DELETE </ AllowedMethod >

< MaxAgeSeconds > 3000 </ MaxAgeSeconds >

< ExposeHeader > ETag </ ExposeHeader >

< AllowedHeader > * </ AllowedHeader >

</ CORSRule >

</ CORSConfiguration >

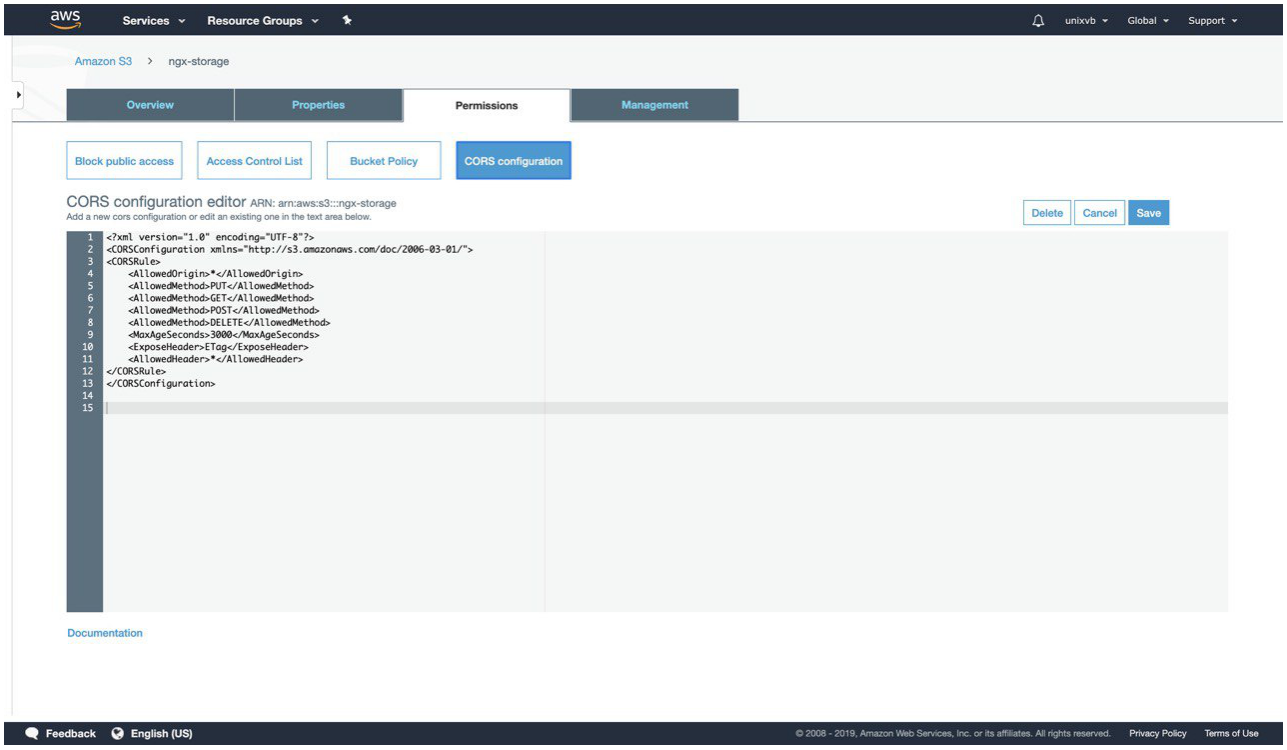

Рис. 11. Налаштування доступів CORS до сховища AWS S3

Після цього користувачам системи буде відкритий доступ до електроних ресурсів сховища через сервіс контролю доступу Amazon Web Services Identity Access Management.

Для того щоб інсталювати імплементований проект локально необхідно:

1. встановити на машину систему контролю версії Git;
2. відкрити командний рядок cmd;

3. перейти в папку, в якій буде встановленно проект командою “cd назва_папки”;
4. зкопіювати проект з приватного репозиторія GitHub виконавши команду “git clone https://github.com/melnikmariya/ngx-s3.git”;
5. перейти в папку проекту командою “cd ngx-s3”;
6. інсталювати всі залежності додатку виконавши команду “npm install”;
7. додати рядок “127.0.0.1 ngx-s3.auth.us-east-1.amazoncognito.com” в файл C:/Windows/System32/drivers/etc/hosts (Рис. 12);

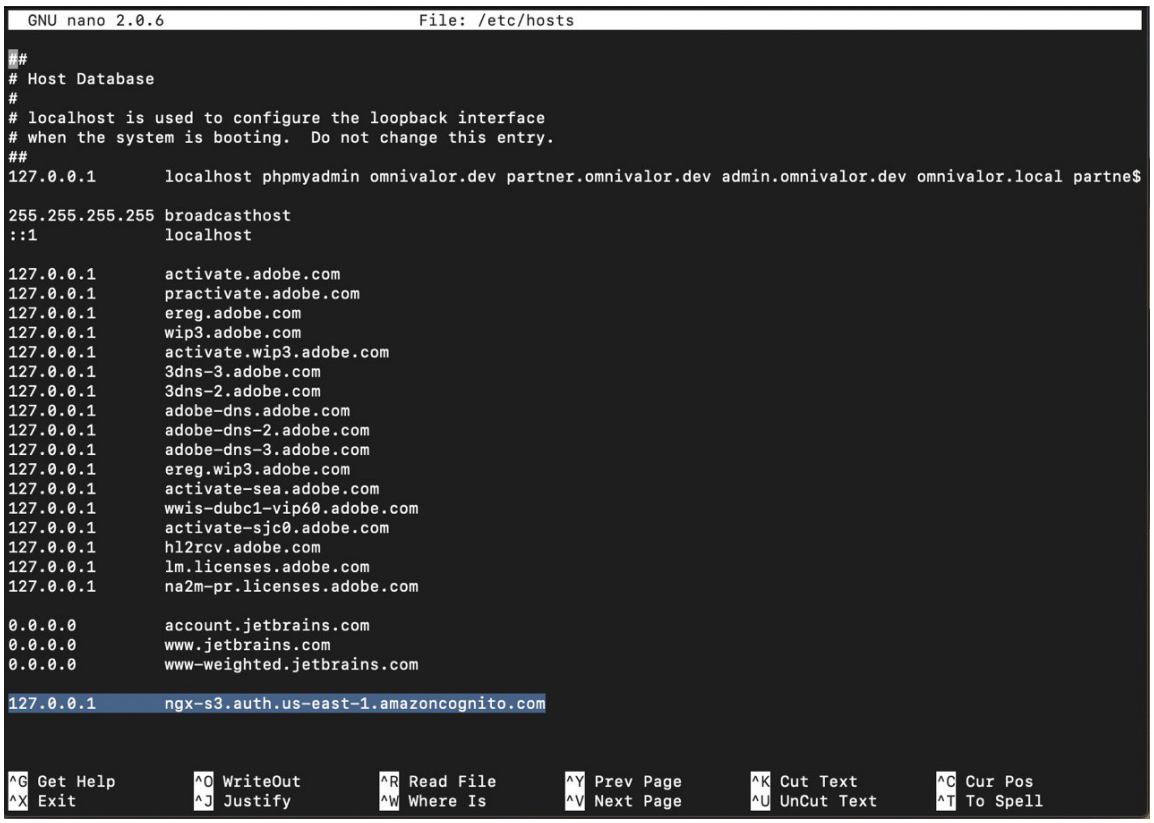

Рис. 12. Налаштування файлу hosts

8. запустити додаток командою “ng serve” в папці з проектом;
9. перейти за посиланням http://ngx-s3.auth.us-east-1.amazoncognito.com:8001/ в браузері.

Після цього у вас локально буде запущений графічний інтерфейс користувача з всіма можливостями системи.

6.3. Загальний опис взаємодії веб-додатку з сервісами AWS

Технологія використання реєстру полягає в наступному:

1. корисувач виконує реєстацію в системі;
2. адміністатору необхідно активувати обліковий запис в сервісі Cognito;
3. адміністатору необхідно надати доступ до директорії в сервісі IAM[18];
4. підчас авторизації користувач отримує ключі доступу та індитифіказії з Cognito User Pool (Рис. 13. Зв’язок 1);
5. використовуючи ключ ідентифікації користувач отримує тимчасові повноваження AWS з Cognito Identity Pool (Рис. 13. Зв’язок 2).

Таким чином корисувач за технологією AWS отримує доступ до операцій над ресурсами (Рис. 13. Зв’язок 3).

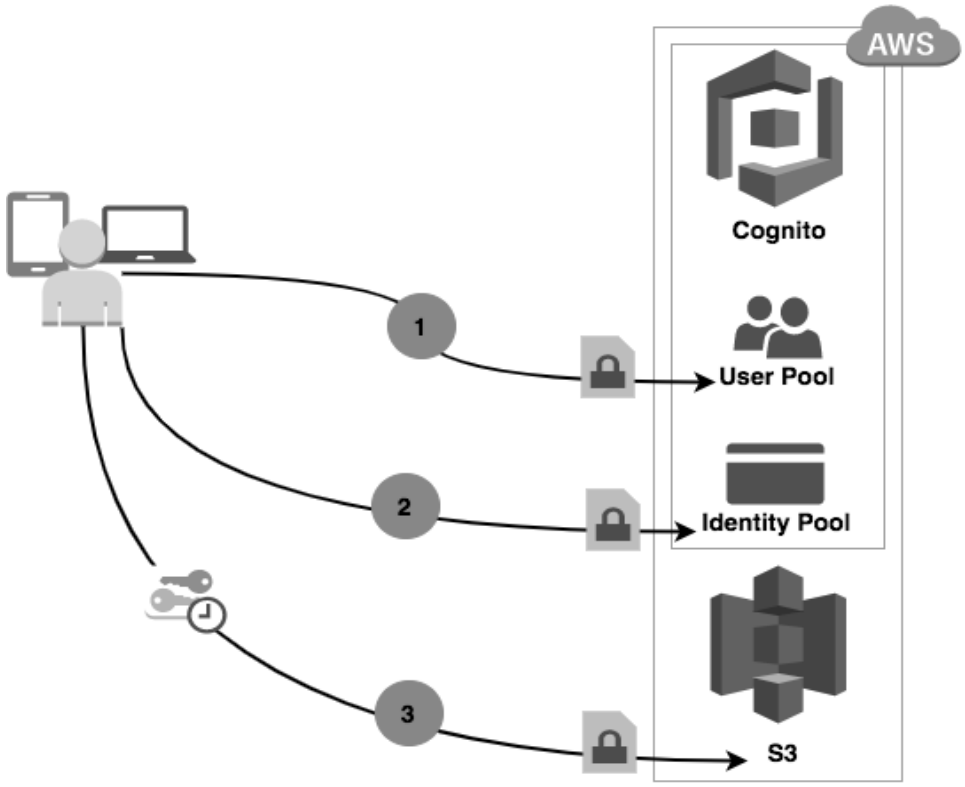

Рис. 13. Схема взаємодії системи та сервісів AWS

На прикладі електронних ресурсів кафедри створенно систему взаємодії веб інтерфейсу з хмарними сервісами AWS для зберігання електронних ресурсів та управління доступом до них.

ВИСНОВКИ

Проаналізовано існуючі реєстри інформаційних ресурсів, визначенно їх переваги та недоліки. Проаналізовано вимоги до нової системи ведення реєстру інформаційних ресурсів. Визначені переваги та недоліки сервісів які надають можливість зберігати данні в хмарних сховищах. Розроблено схему об’єктів данних сховища, а саме файлів та каталогів. Передбачені методи управління доступу до інформаційних ресурсів в хмарному середовищі Amazon Web Services Simple Storage Service за допомогою сервіса Identity Access Management.

З використанням фреймворку Angular 6 та мови програмування TypeScript реалізовано графічний веб інтерфейс для авторизації користувачів та можливість перегляду данних з сховища, завантаження нових файлів, створення вложенної ієрархії каталогів.

Реалізовані можливості програмного продукту повністю задовільняють поставлені задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петров В.В., Нестеренко О.В., Монастирецький М.Г., Шагалов В.Ю. Національні ін-формаційні ресурси. Проблеми формування, розвитку, управління і використання // Реєстрація, зберігання і оброб. даних. — 2001. — Т. 3, № 2. — С. 38–49.
2. Дубова Н. SOA: подходы к реализации // Открытые системы. — 2004. — № 6. — С. 37–41.
3. Ньюкомер Э. Web-сервисы. Для професионалов. — СПб.: Питер, 2003. — 256 с.
4. Нестеренко О.В. Технології інтеграції інформаційних ресурсів інформаційно-аналітичних систем органів державної влади // Науково-технічна інформація. — 2001. — № 4. — С. 3–6.
5. Еременко Т.В. Каталогизация ресурсов Интернета (Опыт библиотек США) // Науч. и техн. б-ки. — 2002. — № 5. — С. 53–67.
6. OCLC [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.oclc.org/en/home.html?redirect=true.
7. Данілін А.В. Стандарти і єдина архітектура інформаційних технологій. — Microsoft Press, 2003. http://www.microsoft.com/Ukraine/Government/Analytics/IntegrationTechnologies/Standards.mspx.
8. SQL Server [Електронний ресурс] // Microsoft. – 2017. – Режим доступу до ресурсу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2017-editions.
9. Флэнаган Д. SAML 2.0 / Дэвид Флэнаган., 2011. – 492 с. – (Питер).
10. TIOBE Index [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.tiobe.com/tiobe-index/.
11. Гибкая разработка веб-приложений / Сэм Руби., 2014. – 448 с. – (Для профессионалов).

12. Krasner G. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80 / G. Krasner, S. Pope // 1988 Article / G. Krasner, S. Pope. – Denville: SIGS Publications, 1988. – (Journal of Object-Oriented Programming). – С. 26–49.
13. The MIT License (MIT) [Електронний ресурс] – Режим доступу до ресурсу: https://opensource.org/licenses/MIT.
14. An Introduction to ERB Templating [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://www.stuartellis.name/articles/erb/
15. Проектування інформаційних систем: Посібник / За ред. В.С. Пономаренка. — К.: Академія, 2002. — 450 с.
16. Воропаева В.Я. Метод оценки влияния промышленно-ориентированных программных комплексов на загрузку информационных систем производственных предприятий // Наукові праці ДонНТУ. Серія: “Електротехніка і енергетика”. — 2015. — №1(17). — С. 98-103.
17. Причины возникновения проблем совместимости программного обеспечения [Електронний ресурс] – Режим доступу до ресурсу: http://helpiks.org/7-46217.html
18. Шабан М.Р. Использование COM-технологии при проведении Экспертизы на соответствие требованием нд тзи / М.Р. Шабан, М.П. Марковская, Ф.Г. Кислов, А.С. Потенко //Моделювання та інформаційні технології. — 2015. — Вип. 75. — С. 56-59.

ДОДАТОК А

Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б 12-1	auth.service.ts	Основний сервіс взаємодії веб додатку з AWS Cognito
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б 12-2	s3-objects.service.ts	Основний компонент взаємодії веб додатку з AWS S3

ДОДАТОК Б

Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі

Лістинг програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 5

Київ 2019

Текст програми взаємодії з AWS Cognito

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { BehaviorSubject } from 'rxjs';
import { AuthenticationDetails, CognitoUser, CognitoUserAttribute, CognitoUserPool, CognitoUserSession } from 'amazon-cognito-identity-js';
import { AWSError, CognitoIdentityCredentials, config as AWSConfig } from 'aws-sdk';
import { cognitoConfig } from '../../config/cognito';
import { SignupDataInterface } from '../models/interfaces/signup-data.interface';
import { AuthStatusCodeEnum, AuthStatusCodeType } from '../models/enums/auth-status-code.enum';
import { UserModel } from '../models/user.model';

@Injectable({ providedIn: 'root' })
export class AuthService {
 private static userPoolLoginKey = `cognito-idp.${cognitoConfig.userPool.region}.amazonaws.com/${cognitoConfig.userPool.UserPoolId}`;

 public cognitoAwsCredentials: CognitoIdentityCredentials;
 public currentStatus$ = new BehaviorSubject<AuthStatusCodeType>(AuthStatusCodeEnum.signedOut);

 private userPool = new CognitoUserPool(cognitoConfig.userPool);
 private previousAppParams: any;
 private signupData: SignupDataInterface = {};

 constructor(private router: Router) {}

 private getCognitoUser(username?: string) {
  username = username || this.signupData.username;
  if (!username) {
   return undefined;
  }
  return new CognitoUser({
   Username: username,
   Pool: this.userPool
  });
 }

 signUp(user: SignupDataInterface, callback?: (err: Error, statusCode: string) => void) {
  const authService = this;

  this.userPool.signUp(
   user.username,
   user.password,
   [new CognitoUserAttribute({ Name: 'email', Value: user.username })],
   null,
   function (err, result) {
    authService.currentStatus$.next(AuthStatusCodeEnum.signedIn);
    authService.signupData = {};
    callback(err, AuthStatusCodeEnum.signedIn);
   });
 }

 signIn(user: SignupDataInterface, callback?: (err: Error, statusCode: AuthStatusCodeType) => void) {
  const authService = this;
  const username = user.username || this.signupData.username;
  const password = user.password || this.signupData.password;
  if (!username || !password) {
   callback(new Error('AuthenticationDetails are incomplete.'),
    AuthStatusCodeEnum.uncompletedSignInData);
   return;
  } else {
   this.signupData.username = username;
   this.signupData.password = password;
  }

  const cognitoUser = this.getCognitoUser(username);
  const auth = new AuthenticationDetails({ Username: username, Password: password });
  cognitoUser.authenticateUser(auth, {
   onSuccess: function (authResult) {
    authService.currentStatus$.next(AuthStatusCodeEnum.signedIn);
    authService.signupData = {};
    callback(null, AuthStatusCodeEnum.signedIn);
   },

   onFailure: function (err) {
    authService.currentStatus$.next(AuthStatusCodeEnum.unknownError);
    if (err.code === 'UserNotFoundException' || err.code === 'NotAuthorizedException') {
     callback(err, AuthStatusCodeEnum.noSuchUser);
    } else {
```

```
        callback(err, AuthStatusCodeEnum.unknownError);
      }
    },
    newPasswordRequired: function (userAttributes, requiredAttributes) {
      if (!user.newPassword) {
        const newNoNewPasswordError = new Error('First time logged in but new password is not provided');
        if (callback) {
          authService.currentStatus$.next(AuthStatusCodeEnum.newPasswordRequired);
          callback(newNoNewPasswordError, AuthStatusCodeEnum.newPasswordRequired);
          return;
        } else {
          throw newNoNewPasswordError;
        }
      }
      if (authService.signupData) {
        userAttributes = Object.assign(userAttributes, authService.signupData.additionalData);
      }
      delete userAttributes.email_verified;
      cognitoUser.completeNewPasswordChallenge(user.newPassword, userAttributes, this);
    }
  });
}

forgotPassword(username: string, callback: (error: Error, statusCode: string) => void) {
  const authService = this;
  this.signupData.username = username;
  const cognitoUser = this.getCognitoUser(username);
  cognitoUser.forgotPassword({
    onSuccess: function () {
      authService.currentStatus$.next(AuthStatusCodeEnum.verificationCodeRequired);
      callback(null, AuthStatusCodeEnum.verificationCodeRequired);
    },
    onFailure: function (err) {
      authService.currentStatus$.next(AuthStatusCodeEnum.unknownError);
      if (err.name === 'UserNotFoundException') {
        callback(err, AuthStatusCodeEnum.noSuchUser);
      } else {
        callback(err, AuthStatusCodeEnum.unknownError);
      }
    },
    inputVerificationCode: function (data) {
      authService.currentStatus$.next(AuthStatusCodeEnum.verificationCodeRequired);
      callback(null, AuthStatusCodeEnum.verificationCodeRequired);
    }
  });
}

confirmPassword(verficationCode: string, newPassword: string, callback: (error: Error, statusCode: string) => void) {
  if (!this.signupData.username) {
    callback(new Error('Username is Empty.'), AuthStatusCodeEnum.uncompletedSignInData);
    return;
  }
  const authService = this;
  const cognitoUser = new CognitoUser({
    Username: this.signupData.username,
    Pool: this.userPool
  });
  cognitoUser.confirmPassword(verficationCode, newPassword, {
    onSuccess: () => {
      authService.currentStatus$.next(AuthStatusCodeEnum.passwordChanged);
      callback(null, AuthStatusCodeEnum.success);
    },
    onFailure: (err: Error) => {
      authService.currentStatus$.next(AuthStatusCodeEnum.unknownError);
      callback(err, AuthStatusCodeEnum.unknownError);
    }
  });
}
signout() {
  const currentUser = this.userPool.getCurrentUser();
  if (currentUser) {
    currentUser.signOut();
  }
  this.currentStatus$.next(AuthStatusCodeEnum.signedOut);
  this.router.navigate(['/signin']);
}
private getCurrentCognitoUser(callback: (err1?: Error, cognitoUser?: CognitoUser, groups?: string[]) => void) {
  const cognitoUser = this.userPool.getCurrentUser();
  if (cognitoUser) {
```

```
    cognitoUser.getSession((err: Error, session: CognitoUserSession) => {
      if (session && session.isValid()) {
        const groups = session.getIdToken().decodePayload()['cognito:groups'];
        if (!this.cognitoAwsCredentials || this.cognitoAwsCredentials.needsRefresh()) {
          this.updateAWSCredentials(session.getIdToken().getJwtToken(), cognitoUser.getUsername(), (err2) => {
            if (err2) {
              callback(err2);
            } else {
              callback(undefined, cognitoUser, groups);
            }
          });
        } else {
          callback(undefined, cognitoUser, groups);
        }
      } else {
        callback(undefined, undefined);
      }
    });
  } else {
    callback(undefined, undefined);
  }
 }
 getCurrentUser(callback: (err?: Error, user?: UserModel) => void) {
   this.getCurrentCognitoUser((err, cognitoUser, groups) => {
     if (cognitoUser && cognitoUser.getUsername()) {
       const identityId = this.cognitoAwsCredentials ? this.cognitoAwsCredentials.identityId : undefined;
       callback(undefined, new UserModel(true, cognitoUser.getUsername(), identityId, groups));
     } else {
       callback(undefined, UserModel.default);
     }
   });
 }
 private updateAWSCredentials(sessionToken: string, username: string, callback: (err?: Error) => void) {
   const logins = {};
   logins[AuthService.userPoolLoginKey] = sessionToken;
   this.cognitoAwsCredentials = new CognitoIdentityCredentials(
     {
       IdentityPoolId: cognitoConfig.identityPool.id,
       Logins: logins,
       LoginId: username
     },
     {
       region: cognitoConfig.userPool.region
     }
   );
   // call refresh method in order to authenticate user and get new temp credentials
   this.cognitoAwsCredentials.refresh((err: AWSError) => {
     if (err) {
       callback(err);
     } else {
       AWSConfig.credentials = this.cognitoAwsCredentials;
       callback(null);
     }
   });
 }

 setPreviousAppParams(params: any) {
   this.previousAppParams = params;
 }
}
```

Текст програми взаємодії з AWS S3

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';
import { S3 } from 'aws-sdk';
import { S3Factory } from '../../utils';
import { s3Config } from '../../config/s3';

export const DIVIDER = '/';

@Injectable({ providedIn: 'root' })
export class S3ObjectsService {

 constructor() {
   this.region = s3Config.defaultRegion;
```

```
  }

  private readonly region: string;
  private signedUrlExpire = 60 * 5;

  public changes$ = new Subject<boolean>();

  private static relativeFolder(folder: string) {
    return folder.substr(1) + DIVIDER;
  }

  private static generateKey(folder: string, name: string) {
    return S3ObjectsService.relativeFolder(folder) + name;
  }

  public list(folder: string) {
    return S3Factory.getS3(this.region).listObjectsV2({
      Bucket: s3Config.buckets[this.region],
      Prefix: S3ObjectsService.relativeFolder(folder),
      Delimiter: DIVIDER
    }).promise();
  }

  public uploadFile(folder: string, file: File, progressCallback: (error: Error, progress: number, speed: number) => void) {
    const s3Upload = S3Factory.getS3(this.region).upload({
      Key: S3ObjectsService.generateKey(folder, file.name),
      Bucket: s3Config.buckets[this.region],
      Body: file,
      ContentType: file.type
    });
    s3Upload.on('httpUploadProgress', this.handleS3UploadProgress(progressCallback));
    s3Upload.send(this.handleS3UploadComplete(progressCallback));
    return s3Upload;
  }

  public createFolder(relativePath: string, name: string) {
    return S3Factory.getS3(this.region).upload({
      Key: S3ObjectsService.generateKey(relativePath, name) + DIVIDER,
      Bucket: s3Config.buckets[this.region],
      Body: ''
    });
  }

  public getSignedUrl(key: string) {
    return S3Factory.getS3(this.region).getSignedUrl('getObject', {
      Bucket: s3Config.buckets[this.region],
      Key: key,
      Expires: this.signedUrlExpire
    });
  }

  private handleS3UploadProgress
  (progressCallback: (error: Error, progress: number, speed: number) => void) {
    let uploadStartTime = new Date().getTime();
    let uploadedBytes = 0;
    return (progressEvent: S3.ManagedUpload.Progress) => {
      const currentTime = new Date().getTime();
      const timeElapsedInSeconds = (currentTime - uploadStartTime) / 1000;
      if (timeElapsedInSeconds > 0) {
        const speed = (progressEvent.loaded - uploadedBytes) / timeElapsedInSeconds;
        const progress = Math.floor((progressEvent.loaded * 100) / progressEvent.total);
        progressCallback(undefined, progress, speed);
        uploadStartTime = currentTime;
        uploadedBytes = progressEvent.loaded;
      }
    };
  }

  private handleS3UploadComplete(
    progressCallback: (error: Error, progress: number, speed: number) => void) {
    return (error: Error, data: S3.ManagedUpload.SendData) => {
      if (error) {
        progressCallback(error, undefined, undefined);
      } else {
        progressCallback(error, 100, undefined);
      }
    };
  }
}
```

ДОДАТОК В

Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі

Опис програмного коду

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 9

Київ 2019

АНОТАЦІЯ

Даний додаток містить опис двох основних програмних модулів розроблених для роботи реєстру електронних інформаційних ресурсів в хмарному середовищі AWS. Створені програмні системи реалізують взаємодію з сервісами AWS Cognito і AWS S3 та виконують такі завдання:

— авторизація користувача в системі;

— отримання даних сховища;

— завантаження електронних ресурсів в сховище;

— створення дерикторій в сховищі.

Програма отримує дані через елементи керування розташовані на графічному інтерфейсі веб додатку реалізованого за допомогою фреймворку Angular 6.

При розробці обох програмних систем використовувалась мова TypeScript у середовищі програмування IntelliJ IDEA WebStorm.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис двох програмних модулів системи для роботи з хмарними сервісами Amazon Web Services. У додатку Б міститься програмний код даних модулів.

Програмний продукт працює в будь-якій операційній системі (Windows, MacOS та Linux) і потребує встановленого на ПК будь-якого сучасного браузера та сервера NodeJS.

При розробці програмного додатку використовувалась мова TypeScript з фреймворком Angular 6 та середовище розробки IntelliJ IDEA WebStorm.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений веб додаток виконує завдання авторизації користувача в системі за допомогою сервіса AWS Cognito, перегляду та додавання електронних ресурсів хмари AWS S3, надання контрольованого доступу до ресурсів за допомогою сервіса AWS IAM.

Також розроблений програмний продукт може використовуватися в якості учбових матеріалів при опрацюванні технологій взаємодії веб додатків та хмарних сервісів Amazon.

Функціональні обмеження на використання сервісів полягає лише в фізичному діапазоні об’єму завантажених електроних даних.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для реалізації взаємодії веб додатку з сервысами Amazon знадобилося розробити веб сервіси. Основним з інструментів для створення сервісів були можливості Angular CLI.

Всі сервіси були написанні на мові програмування TypeScript за допомогою фреймворку Angular 6.

Такоєж до проекту підключаються зовнішні залежності, які управляються менеджером залежностей npm.

При запуску додатка спочатку з’являться вграфічний інтерфейс для введення даних, далі данні передаються до сервісів, сервіси відправляють запити до Amazon, після отримання відповіді обробленні данні відображаються на графічному інтерфейсі.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для забезпечення повноцінної роботи та досягнення високої ефективності роботи створення додатку для демонстрації роботи системи з сервісами Amazon у рамках реалізації реєстру електронних ресурсів було обрано IntelliJ IDEA WebStorm який показав себе надійною та гнучкою середою розробки програм.

Розроблений додаток працюють в будь-якій операційній системі (Windows, MacOS та Linux) і потребує встановленого на ПК будь-якого сучасного юраузера та серверну частину NodeJS.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Розроблений веб додаток не потребує інсталяції, достатньо запустити виконувану команду “ng serve” в кореневій папці проекту.

Після запуску додатка користувач в браузері отримає доступ до графічного інтерфейсу програми, звідки може виконувати визначений функціонал системи.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розробленого додатку є інформація яка зчитується з форм авторизації та реєстрації, файли та назви дерикторій які користувач обирає при завантажені в хмару Amazon.

Вхідні дані для елемента input можуть бути будь-якими строками, файли для завантаження будь-якого формату.

Вихідними даними є список з завантаженими електронними ресурсами та посилання на них в хмарі

ДОДАТОК Г

Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі

Акт впровадження

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 2

Київ 2019

“Затверджую”
В.о. зав. кафедри АПЕПС
КПІ ім. Ігоря Сікорського

к.т.н. Коваль О.В.

“4” червня 2019 р.

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи освітньо-кваліфікаційного рівня “бакалавр”

Іванів Андрія Петровича

Іванів А.П. у процесі виконання дипломної роботи на тему “ Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі” розробив програмний додаток, що включає в себе користувацький графічний інтерфейс для завантаження електронних ресурсів в хмеру, що був успішно реалізований у рамках виконання бакалаврської роботи.

Розроблений реєстр інформаційних ресурсів виконує низку функцій, основними з яких є: реєстація та авторизація користувача в системі, формування груп доступу до ресурсів, завантаження ресурсу в сховище AWS S3, розбиття реєстру на категорії, перегляд ресурсу за прямим посиланням та вивантаження ресурсу на локальний комп’ютер.

Комплекс розробляється у процесі виконання науково-дослідної роботи на тему “*Технологія поб удови динамічних реєстрів електронних інформаційних ресурсів та засобів їх ефективної обробки у датацентрах гетерогенної структури*“, що виконується на кафедрі АПЕПС ТЕФ у Лабораторії комп’ютерного моделювання динамічних процесів і систем.

Результати виконання дипломної роботи впроваджені в лабораторії “Комп’ютерного моделювання динамічних процесів ти систем”.

Зав. нав.-наук. лабораторії В.І. Гайдаржи

ДОДАТОК Д

Інструментальні засоби супроводження реєстру інформаційних ресурсів в хмарному середовищі

Результати апробації

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 12

Київ 2019

Результати роботи були публіковані в двох конференціях: XVII Міжнародна науково-практична конференція аспірантів, магістрантів і студентів “Сучасні проблеми наукового забезпечення енергетики”, м.Київ, КПІ ім. Ігоря Сікорського, 23-26 квітня 2019 року, VI науково-практична дистанційна конференція молодих вчених і фахівців з розробки програмного забезпечення “Сучасні аспекти розробки програмного забезпечення”,м. Київ, КПІ ім. Ігоря Сікорського, 31 травня 2019 року.