НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Клауд-платформа створення, підтримки та моніторингу ETL-процесів (комплексна тема). Загальна частина

Виконав студент IV курсу, групи ІП-81
(шифр групи)
Добрянський Богдан Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

студент IV курсу, групи ІП-81
(шифр групи)
Савастру Станіслав Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник проф., д.т.н., проф. Стеценко І.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент професор кафедри ОТ, д.т.н, доцент, Клименко І.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ –2022

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютеризованих систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2022 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Добрянський Богдан Ігорович
(прізвище, ім'я, по батькові)

Савастру Станіславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Клауд-платформа створення, підтримки та моніторингу ETL-процесів (комплексна тема). Загальна частина

керівник проєкту Стеценко Інна В'ячеславівна, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» червня 2022 р. №1033-С

2. Термін подання студентом проєкту « 19 » червня 2022 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.

4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

5. Перелік графічного матеріалу

1) Схема структурна діяльності

2) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2022 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2022	
2	Аналіз існуючих методів розв’язання задачі	17.03.2022	
3	Постановка та формалізація задачі	23.03.2022	
4	Розробка інформаційного забезпечення	12.04.2022	
5	Алгоритмізація задачі	15.04.2022	
6	Обґрунтування вибору використаних технічних засобів	27.04.2022	
7	Розробка програмного забезпечення	09.05.2022	
8	Налагодження програми	16.05.2022	
9	Виконання графічних документів	21.05.2022	
10	Оформлення пояснювальної записки	28.05.2022	
11	Подання ДП на попередній захист	06.06.2022	
12	Подання ДП рецензенту	12.06.2022	
13	Подання ДП на основний захист	19.06.2022	

Студент _____ Богдан ДОБРЯНСЬКИЙ
(підпис) (ініціали, прізвище)

Студент _____ Станіслав САВАСТРУ
(підпис) (ініціали, прізвище)

Керівник _____ Інна СТЕЦЕНКО
(підпис) (ініціали, прізвище)

Анотація

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 23 таблиці, 10 рисунків та 12 джерел – загалом 53 сторінки.

Дипломний проєкт присвячений автоматизації конфігурування, підтримки та моніторингу виконання ETL-процесів у клауд середовищі.

Мета: підвищення якості та зручності програмного забезпечення для створення, підтримки та моніторингу ETL-процесів за рахунок клауд технологій.

Об'єкт дослідження: програмне забезпечення для створення, підтримки та моніторингу ETL-процесів.

Предмет дослідження: процеси розроблення, модифікації, аналізу, забезпечення якості, впровадження і супроводження програмного забезпечення для створення, підтримки та моніторингу ETL-процесів.

У розділі «Аналіз вимог до програмного забезпечення» було розглянуто загальні положення, виконано аналіз предметної області та сформовано постановку задачі.

Розділ «Модулювання та конструювання програмного забезпечення» присвячений моделюванню, аналізу та конструюванню програмного забезпечення.

У розділі «Аналіз якості та тестування програмного забезпечення» було розглянуто призначення та предмет тестування. Було описано вимоги до середовища тестування та описано тести, що виконувалися над розробленим програмним продуктом.

Розділ «Впровадження та супровід програмного забезпечення» присвячений розгортанню та супроводу програмного забезпечення у різних середовищах.

КЛЮЧОВІ СЛОВА: ETL, CLOUD, WEB APPLICATIONS, DATABASES.

ABSTRACT

The explanatory note to the diploma project contains 23 tables, 10 figures, 12 sources and 53 pages.

The diploma project is dedicated to the automation of configuration, support and monitoring of ETL-processes in the cloud environment.

The aim of the diploma project is to improve the quality and usability of software for creating, maintaining and monitoring ETL processes using cloud technologies.

The object of research: software for creating, maintaining, and monitoring ETL-processes.

The subject of research: processes of development, modification, analysis, quality assurance, implementation, and maintenance of software for the creation, maintenance, and monitoring of ETL-processes.

In the section ‘Analysis of the software requirements’ the general provisions were considered, the analysis of the subject area was performed and the problem statement was formed.

The section ‘Modulation and design of software’ is devoted to modeling, analysis, and design of the software.

In the section ‘Software quality analysis and testing’ the purpose and subject of testing were considered. The requirements for the testing environment were described and the tests performed on the developed software product were described.

The section ‘Software Implementation and Support’ is devoted to the deployment and maintenance of software in different environments.

KEYWORDS: ETL, CLOUD, WEB APPLICATIONS, DATABASES.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-8112.27.045440.01.91	Технічне завдання	11	
3	А4	КПІ.ІП-8112.27.045440.02.81	Пояснювальна записка	52	
4	А4	КПІ.ІП-8112.27.045440.03.34	Керівництво користувача	10	
5	А3	КПІ.ІП-8112.27.045440.04.99	Графічний матеріал	2	
6	А4		Завдання на дипломний проєкт	2	
7	А4	КПІ.ІП-8127.045440.05.81	Пояснювальна записка	53	
8	А4	КПІ.ІП-8127.045440.06.13	Опис програми	39	
9	А4	КПІ.ІП-8127.045440.07.51	Програма тестування	5	
10	А3	КПІ.ІП-8127.045440.08.99	Графічний матеріал	2	
11	А4		Завдання на дипломний проєкт	2	
12	А4	КПІ.ІП-8112.045440.09.81	Пояснювальна записка	79	
13	А4	КПІ.ІП-8112.045440.10.13	Опис програми	44	
14	А4	КПІ.ІП-8112.045440.11.51	Програма тестування	5	
15	А3	КПІ.ІП-8112.045440.12.99	Графічний матеріал	2	

КПІ.ІП-8112.27.045440.00.90

Змін.	Арк.	№ докум.	Підп.	Дата
Розроб.		Добрянський Б.І.		
Розроб.		Савастру С.В.		
Перевір.		Стеценко І.В.		
Н.контр.		Ліщук К.І.		
Затв.		Жаріков Е.В.		

Відомість дипломного проєкту

Літ.	Аркуш	Аркушів
	1	

КПІ ім. Ігоря Сікорського
ФІОТ каф. ІПІ гр. ІП-81

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

КЛАУД-ПЛАТФОРМА СТВОРЕННЯ, ПІДТРИМКИ ТА МОНІТОРИНГУ ETL-ПРОЦЕСІВ (КОМПЛЕКСНА ТЕМА). ЗАГАЛЬНА ЧАСТИНА

Технічне завдання

КПІ.ІП-8112.27.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Інна СТЕЦЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавці:
_____ Богдан ДОБРЯНСЬКИЙ
_____ Станіслав САВАСТРУ

Київ – 2022

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Клауд-платформа створення, підтримки та моніторингу ETL-процесів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку клауд-платформа створення, підтримки та моніторингу ETL-процесів, котра використовується для створення, підтримки та моніторингу ETL-процесів та призначена для користувачів, яким потрібне рішення для зручного проведення та моніторингу міграції даних між базами даних, та яку можна легко встановити у хмару.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки клауд-платформи для створення, підтримки та моніторингу ETL-процесів є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації конфігурування, підтримки та моніторингу виконання ETL-процесів у клауд середовищі.

Метою розробки є підвищення якості та зручності програмного забезпечення для створення, підтримки та моніторингу ETL-процесів за рахунок клауд технологій.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- авторизація та аутентифікація;
- створення, редагування та видалення ETL-процесу;
- конфігурація вхідних джерел даних;
- конфігурація кінцевих джерел даних;
- налаштування таких кроків виконання у ETL-процесі: витягування даних з вхідних джерел, трансформування та фільтрація потоку даних та збереження трансформованих даних до кінцевих сховищ;
- запуск та перегляд результатів моніторингу виконання ETL-процесу з можливістю зупинки;

4.1.1.2 Для адміністратора системи:

- початкова ініціалізація системи зі створенням головного адміністратора;
- створення та адміністрування користувачів платформи;

4.1.2 Додаткові вимоги:

- застосунок має збиратися у Helm-модуль для легкого встановлення до хмари;
- прогрес виконання ETL-процесу має відображатися у реальному часі;
- кожен активний ETL-процес має бути ізольованим та не впливати на виконання інших;

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.2.4 Кожен активний ETL-процес має бути ізольованим та не впливати на виконання інших.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору ..... Intel.

4.4.2.2 Об‘єм ОЗП..... 4 Гб.

4.4.2.3 Кількістьть ядер процесору…… 4

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі:

- дані введені користувачем на екранній формі;
- JSON для запиту до серверної частини застосунку.

4.5.3 Результати повинні бути представлені в наступному форматі:

- відображення на екранній формі;
- JSON відповідь від серверної частини застосунку.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування.

5.4 Графічна частина повинна бути виконана не менше ніж на 3 аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Креслення вигляду екранних форм.

5.4.2 Схема структурна діяльності.

5.4.3 Схема бази даних.

5.4.4 Схема структурна класів серверної частини.

5.4.5 Схема структурна варіантів використання.

5.4.6 Схема структурна класів клієнтської частини.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03	
2.	Розробка технічного завдання	25.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	05.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	15.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	23.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	27.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	таки 09.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	15.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	21.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему:

Клауд-платформа створення, підтримки та моніторингу ETL-процесів (комплексна тема).

Загальна частина

КПІ.ІП-8112.27.045440.02.81

Київ – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTTP	–	HyperText Transfer Protocol, протокол передачі гіпертексту
API	–	Application programming interface, прикладний програмний Інтерфейс
UML	–	Unified Modeling Language, уніфікована мова моделювання
БД	–	База даних.
СУБД	–	Система управління базами даних
ETL	–	Extract, Transform, Load, процес виймання, перетворення та завантаження даних

ВСТУП

На сьогодні, у часи, коли більшість організацій надає перевагу розгортанню своєї інфраструктури у клауді, дуже важливим є адаптація вже звичних та дуже важливих рішень до цього середовища. Одним з них є організація та проведення ETL-процесів, адже міграція даних між сховищами з попередньою з фільтрацією та трансформацією є одною з базових складових для пришвидшення роботи аналітиків або для автоматизації міграції бази даних, наприклад, при переході на нову версію якогось модулю системи, що, як ми знаємо, є невід'ємним елементом процесу розробки. У першому випадку аналітик матиме змогу швидко налаштувати та виконати ETL-процес замість того, щоб під'єднуватися до бази напряму та виконувати усі операції вручну, а потім ще й завантажувати у кінцеву базу. У другому випадку це значно спрощує процедуру міграції між сховищами даних, навіть без змушеної зупинки роботи елементів інфраструктури, що взаємодіють з даними сховищами.

Метою даної роботи є розробка платформи для створення, підтримки та моніторингу ETL-процесів. Вже з назви роботи зрозуміло, що наша основна мета – зробити систему сумісною з клауд середовищем. У нашому випадку система має легко встановлюватися до кластеру Kubernetes та інтуїтивно конфігуруватися. Платформа матиме зручний графічний інтерфейс та систему авторизації на базі ролей.

Можливими користувачами системи можуть стати підприємства, внутрішня інфраструктура яких розгорнута у клауді. Це є оптимальним рішенням, бо така платформа стане повноцінним елементом їхньої інфраструктури. У іншому випадку їм треба буде приєднуватись до сторонніх системи, що скоріше за все призведе до появи вразливості до атак на систему, або ж до значного ускладнення обслуговування такої системи.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення та аналіз предметної області

Кінцевою метою нашої роботи є розробка клауд-платформи для створення, підтримки та моніторингу ETL-процесів. Для початку розберемося з ти, що таке ETL-процес. В загальному випадку під ETL-процесом розуміється процес, який складається з трьох етапів:

- витягування даних (extract);
- трансформація даних (transformation);
- завантаження даних до кінцевого сховища даних (load) [1].

На першому етапі відбувається під'єднання до початкових сховищ даних та витягування з них даних відповідно до початкової конфігурації ETL-процесу. На другому етапі ми проводимо фільтрацію та трансформацію даних, що отримали на першому. На цьому ж етапі ми проводимо поєднання даних з кількох джерел даних, якщо така задача наявна. Наприклад, якщо одна складова даних лежить у одному сховищі, а друга – у іншому сховищі, тоді нам треба зробити поєднання даних за деякою ознакою. На етапі вигрузки даних нам потрібно під'єднатися до цільового сховища даних та завантажити до нього дані.

Існує 2 основні підходи до виконання ETL-процесу:

- написання та мануальне виконання окремих скриптів на для конкретного ETL-процесу;
- використання готового програмного забезпечення, що дозволяє конфігурувати та виконувати ETL-процес за допомогою графічного інтерфейсу[2].

Розглянемо кожний підхід окремо.

Написання скриптів під конкретний ETL-процес звичайно має дуже великий ступінь гнучкості. По-перше, знімається більшість обмежень на джерела даних та кінцеві сховища даних. Головною умовою можливості

роботи з ними стає наявність інструментів роботи з цим джерелом у обраній для написання ETL-процесу мові програмування. Аналогічно покращується гнучкість роботи на етапі трансформації даних. Але ж звичайно є й недоліки у такому підході.

Перший й найголовніший недолік – людина має вміти програмувати та знати мову програмування, що підійде для таких цілей на доволі високому рівні. А далеко не кожна людина, якій потрібні результати роботи ETL-процесу є програмістом.

Другий недолік – складність конфігурації та подальшої модифікації ETL-процесу, що забиратиме величезну кількість часу.

Також значно ускладнюється його виконання. Звичайно, якщо і джерела даних, і сховища даних у межах доступності з комп'ютера користувача, то виконати його можна доволі швидко. Проблеми виникають, коли ці бази даних знаходяться на віддаленому сервері або у хмарі. Тоді користувач буде вимушений заходити до цього серверу або хмари та виконувати ті скрипти там. Такий підхід є занадто складним та ризикованим з точки зору інформаційної безпеки[3].

Ще одним недоліком такого підходу можна виділити неможливість моніторингу стану ETL-процесу.

Усі ці недоліки значно перекриваються переваги у гнучкості. Тож у більшості випадків підхід є неоптимальним. Використовувати його доцільно лише тоді, коли такий ETL-процес є одноразовим, тож його написання та виконання забере менше часу, ніж налаштування програмного забезпечення для автоматичного виконання ETL-процесу.

Тепер розглянемо інший підхід, а саме використання готових програмних рішень для налаштування ETL-процесу. Головною перевагою такого підходу є налаштування за допомогою графічного інтерфейсу. Зазвичай, користувач подібного програмного забезпечення має можливість налаштовувати джерела даних зі списку вже підтриманих. Те ж саме

відноситься й до кінцевих сховищ даних. Далі вже саме програмне забезпечення аналізує обрані джерела даних та пропонує обрати, наприклад, потрібні таблиці. Так само зазвичай налаштовується й етап трансформації. Тобто, ми бачимо вже першу й основну перевагу – користувач не обов'язково має знати якусь мову програмування, щоб налаштувати ETL-процес. Відповідно зменшується час конфігурації та полегшується його модифікація, а графічний інтерфейс також дозволяє візуалізувати елементи ETL-процесу. Те ж саме стосується й процесу його виконання. Використовуючи готове програмне рішення для ETL-процесу ми можемо у реальному часі проводити моніторинг стану процесу під час виконання та зберігати результати виконання[4].

Однак, слід зауважити, що зазвичай такі застосунки є пропрієтарними, платформо-орієнтованими або непристосованими до клауду. Ліквідація цих недоліків є також однією з цілей даної роботи.

1.2 Аналіз успішних IT-проектів

1.2.1 Аналіз відомих технічних рішень

Розглянемо більш детально, які типи рішень для ETL-процесів існують. Такі рішення можна поділити на 3 категорії:

- Enterprise рішення;
- Open-Source рішення;
- Cloud-Based рішення[5].

Також іноді окремо виділяють внутрішні розробки компаній подібних рішень, що вони використовують у своїх бізнес процесах, але, за відсутності публічної інформації про них, робити це ми не будемо. Тож розглянемо кожен тип окремо.

Enterprise рішення для ETL-процесів зазвичай є найбільш досконалими з усіх існуючих, пропонують розвинений графічний інтерфейс та підтримують велику кількість різноманітних типів сховищ даних. Вони розробляються та

підтримуються комерційними організаціями. Однак ціна за використання такого роду застосунків досить велика. Також розробники часто перевантажують рішення різноманітним функціоналом, що робить програму більш складною для користувача та більш вимогливою до системних ресурсів.

Open-Source рішення для ETL-процесів на відміну від попередніх вільні для користування. Також кожен користувач може продивитися код такого рішення, щоб впевнитися у відсутності вразливостей. Однак, такі рішення зазвичай мають меншу кількість функціоналу та не гарантують постійну підтримку та повноту заповнення документації. Хоча в даному випадку обмеженість функціоналу не є значним недоліком, бо основні інструменти, що використовуються найчастіше там присутні. Також в більшості випадків Open-Source рішення можна доволі легко встановити клауду, зібравши перед цим програму у образ Docker.

Cloud-Based рішення для ETL-процесів – рішення, які надаються провайдерами клауду. Головною їх перевагою є швидкість роботи, доступність та гнучкість сервісів, тож система сама буде адаптуватись під потрібну кількість ресурсів для виконання ETL-процесу. Також, якщо користувач тримає свої дані у того ж клауд-провайдера, то мають місце додаткові оптимізації під час виконання ETL-процесів. Значним недоліком такого підходу є те, що Cloud-Based рішення працює лише у середі обраного клауд-провайдера. Тобто, якщо дані для аналізу знаходяться у клауді іншого провайдера, їх треба буде перенести на той, де буде виконуватися ETL-процес.

1.2.2 Аналіз відомих програмних продуктів

Розглянемо найпопулярніші існуючі рішення для виконання ETL-процесів.

Talend Open Studio

Продукт з відкритим кодом для створення та виконання ETL-процесів від компанії Talend.

Переваги:

- відкритий код;
- підтримка всіх найпопулярніших СУБД;
- велика кількість підтримуваних функцій для перетворювання даних;
- гнучка система плагінів;
- генерація коду для кожного створеного ETL-процесу, що дозволяє виконувати його незалежно.

Недоліки:

- не надто зручний та дещо застарілий графічний інтерфейс;
- відсутність веб-версії;
- помилки в згенерованому коді, які часто можливо усунути лише вручну, що фактично нівелює всі переваги автоматичної генерації коду;
- відсутність інтегрованого планування, для нього пропонується використання сторонніх планувальників;
- відсутність інтегрованого моніторингу.

HevoData

Пропрієтарний продукт від компанії Hevo.

Переваги:

- джерелом даних можуть бути не лише СКБД, в якості них можуть виступати різні сервіси, наприклад данні з Shopify, Stripe, Google Drive та багатьох інших CRM та Ecommerce систем або навіть довільного REST API;
- інтуїтивно зрозумілий веб-інтерфейс;
- синхронізація даних в реальному часі;
- інтегроване планування та моніторинг виконання ETL-процесів.

Недоліки:

- пропрієтарне, платне ПЗ;
- централізація, відсутність можливості розгортання в хмарі;

– значно менша (в порівняні з Talend) кількість підтримуваних вихідних сховищ даних;
– відсутність можливості розширення через систему плагінів.

SQL Server Integration Services (SSIS)

Продукт від компанії Microsoft для створення складних бізнес-рішень в області інтеграцій та перетворення даних

Переваги:

– велика кількість вбудованих найчастіше використовуваних задач та перетворювань;
– можливість використання як графічного інтерфейсу, так і коду для створення ETL-процесів;
– вбудований пакетний менеджер, який містить велику кількість як окремих готових компонентів ETL-процесів, так і готові ETL-процеси, завантажені користувачами;
– є частиною MS SQL Server, тобто не вимагає окремої ліцензії[6].

Недоліки:

– пропрієтарне ПЗ;
– замкнутість на постачальнику, переважна більшість функцій так чи інакше пов’язана з іншими продуктами Microsoft, наприклад інтеграція з хмарою Azure, інтеграція з Azure Synapse Analytics, прив’язка всіх вбудованих компонентів до SQL Server; фактично всі переваги проявляються лише при повному використанні стеку Microsoft.

1.3 Аналіз вимог до програмного забезпечення

1.3.1 Розроблення функціональних вимог

Сформулюємо функціональні вимоги до розроблюваної системи. У системі початково передбачується 3 актора: неавторизований користувач, користувач системи та адміністратор системи. Неавторизований користувач передбачувано матиме змогу лише авторизуватися на платформі для подальших дій. Авторизований користувач та адміністратор загалом будуть відрізнятися лише можливістю керувати користувачами. Зобразимо функціональні вимоги за допомогою діаграми варіантів використання на рисунку 1.1.

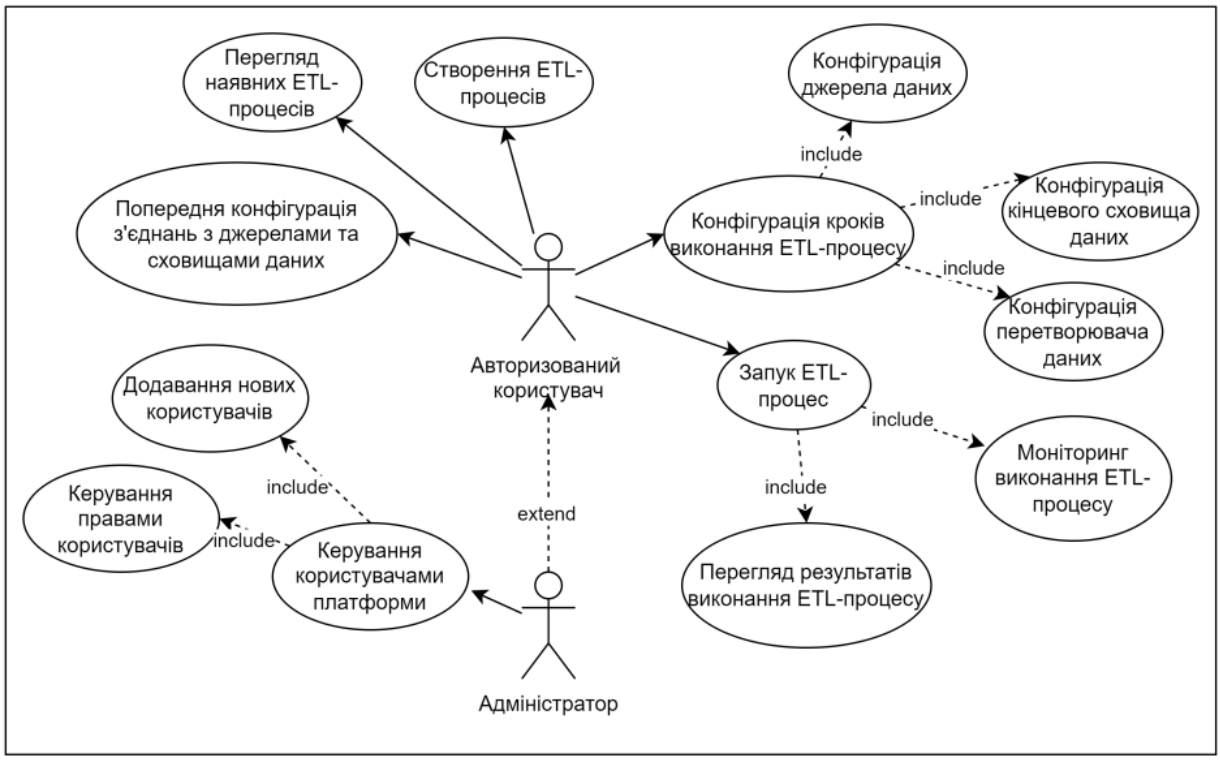


Рисунок 1.1 – Діаграми використання платформи

Більше детально варіанти використання з відповідними функціональними вимогами описані у таблиці 1.1.

Таблиця 1.1 – Варіанти використання з функціональними вимогами

Актор	Варіант використання	Функціональна вимога
Неавторизований користувач	Авторизація у системі	Неавторизований користувач має змогу авторизуватися у системі за наявності у нього акаунту, виданого адміністратором.
Авторизований користувач	Створення ETL-процесу	Користувач може створити ETL-процес, що дасть йому можливість подальшої його конфігурації.
	Перегляд наявних ETL-процесів	Користувач має змогу передивлятися наявні у системі ETL-процеси
	Перегляд кроків виконання ETL-процесу	Користувач відкриває екранну форму, що містить наявні у системі ETL-процеси та обирає потрібний йому зі списку. Користувач отримує екранну форму, на якій зображено кроки виконання ETL-процесу.
	Попередня конфігурація з’єднань за джерелами та сховищами даних	Користувач обирає тип сховища даних. Користувач отримує екранну форму, на якій йому пропонується заповнити потрібні для з’єднання зі сховищем даних параметри.

Продовження таблиці 1.1

		Користувач заповнює потрібні параметри з’єднання. Користувач має змогу перевірити коректність даних та протестувати з’єднання зі сховищем даних перед відправкою запиту на зберігання.
	Конфігурація джерела даних для ETL-процесу	Користувач відкриває екранну форму, що містить наявні у системі ETL-процеси та обирає потрібний йому зі списку. Користувач отримує екранну форму, на якій зображено кроки виконання ETL-процесу.
		Користувач обирає потрібний тип джерела даних. Користувач отримує список з доступними джерелами даних даного типу. Користувач обирає потрібне джерело зі списку. Користувач обирає потрібну таблицю з джерела даних.
	Конфігурація перетворювача даних	Користувач відкриває екранну форму, що містить наявні у системі ETL-процеси та обирає потрібний йому зі списку.

Продовження таблиці 1.1

		Користувач отримує екранну форму, на якій зображено кроки виконання ETL-процесу. Користувач обирає джерело даних для перетворювача з наявних у ETL-процесі. Користувач конфігурує перетворювач даних за допомогою JavaScript коду.
	Конфігурація джерела даних для ETL-процесу	Користувач відкриває екранну форму, що містить наявні у системі ETL-процеси та обирає потрібний йому зі списку. Користувач отримує екранну форму, на якій зображено кроки виконання ETL-процесу.
		Користувач обирає потрібний тип кінцевого сховища даних. Користувач отримує список з доступними сховищами даних даного типу. Користувач обирає потрібне сховище даних зі списку. Користувач отримує список з доступними джерелами даних для завантаження у сховище з тих, що вже присутні у ETL-процесі. Користувач обирає потрібні зі списку.

Продовження таблиці 1.1

	Виконання ETL-процесу	Користувач відкриває екранну форму, що містить наявні у системі ETL-процеси та обирає потрібний йому зі списку. Користувач натискає на кнопку виконання ETL-процесу.
	Моніторинг виконання ETL-процесу	Користувач відкриває екранну форму з історією виконання ETL-процесів та обирає потрібний йому зі списку процесів, що виконуються.
	Перегляд результатів виконання ETL-процесу	Користувач відкриває екранну форму з історією виконання ETL-процесів та обирає потрібний йому зі списку процесів, що вже завершили виконання.
Адміністратор	Перегляд наявних у системі користувачів	Адміністратор відкриває екранну форму, що містить наявних у системі користувачів
	Додавання нових користувачів до системи	Адміністратор відкриває екранну форму для створення користувачів та заповнює потрібні параметри
	Керування правами користувачів	Адміністратор відкриває екранну форму, що містить наявних у системі користувачів. Адміністратор має змогу додавати права користувачам або ж їх відбирати.

Продовження таблиці 1.1

	Видалення наявних користувачів з системи	Адміністратор відкриває екранну форму, що містить наявних у системі користувачів. Адміністратор видаляє непотрібного користувача.

1.3.2 Розроблення нефункціональних вимог

Нефункціональними вимогами є такі вимоги до програмного продукту, що не відносяться до функціонального навантаження та поведінки системи. Сформулюємо нефункціональні вимоги до платформи:

- графічний інтерфейс системи має бути інтуїтивно зрозумілим та зручним у використанні;
- платформа має легко масштабуватись;
- платформа має легко встановлюватись до клауду (кластеру Kubernetes);
- складність додавання підтримки нових типів СУБД для використання у якості джерел та сховищ даних має бути мінімальною;
- усі виконання ETL-процесів мають бути ізольованими один від одного, тож вони не мають впливати на виконання інших ETL-процесів та на коректність роботи самої платформи.

1.4 Постановка задачі

У даній роботі розглядається розробка платформи для роботи з ETL-процесами, що має графічний веб-інтерфейс та пристосована до легкого розгортання у хмарі. Якщо розглядати основний сценарій, що має виконувати платформа, то можна виділити наступні входи та виходи.

Нехай, користувач має доступ до деякої скінченної кількості джерел та сховищ даних різних типів та хоче провести ETL-процес, що передбачає виймання даних з джерел даних, їх фільтрацію, перетворення та завантаження до сховищ даних. Тоді входами нашої платформи є:

- параметри з’єднання з джерелами даних;
- параметри з’єднання з кінцевими сховищами даних;
- конфігурація виймачів, перетворювачів та завантажувачів даних у деякому визначеному форматі.

Так як наступним кроком сценарію роботи є запит користувача на виконання ETL-процесу, то можна виділити наступні виходи розроблюваної платформи:

- результати моніторингу виконання ETL-процесу, що оновлюються у реальному часі;
- дані що були, відфільтровані, трансформовані та вивантажені до кінцевих сховищ відповідно до вхідної конфігурації ETL-процесу;
- звіт про виконання ETL-процесу.

Висновки до розділу

У даному розділі нами було викладено основні поняття щодо ETL-процесів та проаналізовано предметну область. Були розкриті основні підходи для організації ETL-процесів та наведені приклади реалізацій подібних програмних продуктів. Було проаналізовано існуючі технічні рішення та існуючі успішні продукти. В результаті було сформульовано функціональні та нефункціональні вимоги до програмного забезпечення та визначено задачі даної роботи.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

2.1.1 Змістовна постановка задачі

Розроблювана платформа призначена для створення, підтримки та моніторингу ETL-процесів. Платформа повинна мати зручний та адаптивний графічний інтерфейс, має підтримувати не менше 2 типів баз даних, що будуть використовуватися у ролі джерел та сховищ даних. Ці бази даних мають бути різного типу (SQL та NoSQL). Архітектура застосунку має бути побудована таким чином, щоб була можливість додавання підтримки нових баз даних без суттєвих змін у архітектурі та вихідному коді застосунку. Для цього у роботі потрібно використовувати деякі з загальноприйнятих підходів та шаблонів проектування, де вони доречні.

Кінцевим результатом моделювання повинен стати опис загальної архітектури платформи з обґрунтуванням описаних рішень. Користувач платформи, побудованої на основі розробленої моделі, повинен мати можливість виконувати усі операції, що були вказані у вимогах, насамперед операції з ETL-процесами. Також дуже важливою є умова того, що усі ETL-процеси ізольовані один від одного та від самого застосунку, тож не впливатимуть на якість його роботи. Одним з варіантів виконання цієї умови може стати виконання кожного ETL-процесу у окремому системному процесі.

Розроблена модель має бути пристосованою до розгортання у клауд середовищі. Одразу після встановлення платформи до клауду має проводитися її ініціалізація. Під ініціалізацією розуміється в тому числі створення першого користувача системи з правами адміністратора. Після першої авторизації у системі користувачу повинен змінити згенерований пароль на більш безпечний. Платформа має блокувати усі інші дії, що може виконувати авторизований користувач, доки не буде проведена зміна паролю.

2.2 Архітектура програмного забезпечення

2.2.1 Загальні положення по архітектурі програмного забезпечення

Розроблювана платформа глобально складається з двох великих частин: серверної частини та клієнтської частини. Серверна частина відповідає за бізнес логіку застосунку та обробку запитів з клієнтської частини. Клієнтська частина формує запити до серверної частини та відображає користувачу дані, що прийшли у відповідь. Комунікація між клієнтською і серверною частиною відбувається за допомогою протоколу HTTPS та SSE для завантаження даних з серверу у реальному часі.

Для можливості відрізнити процес виконання ETL-процесу від конфігурації ETL-процесу, що зберігається у системі, ми вводимо таке поняття як «ETL-пайплайн» або просто «пайплайн», що є англіцизмом та переводиться як «конвеєр». Це слово надалі позначатиме саме конфігурацію ETL-процесу у нашій платформі.

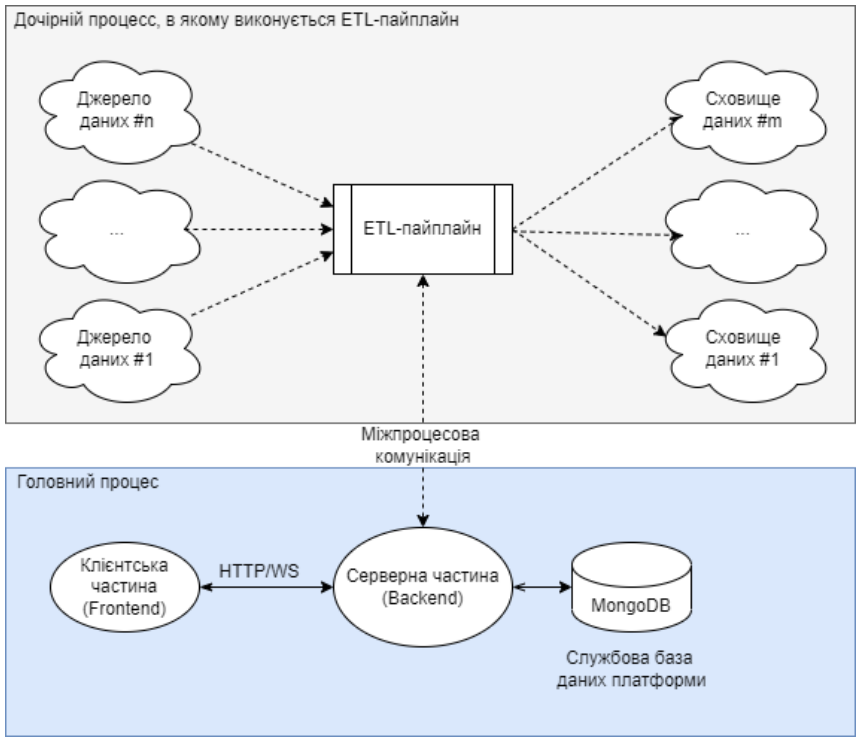


Рисунок 2.1 – Архітектура платформи

Для зберігання даних платформи була обрана NoSQL база даних MongoDB. Ми обрали саме її через те, що у NoSQL базі даних значно легше зберігати багаторівневі структури даних, якими є ETL-пайплайн.

В якості перших підтримуваних баз даних були обрані PostgreSQL[7] та MongoDB[8], бо вони є одними з найпопулярніших представників SQL та NoSQL баз даних відповідно. Виконання пайплайнів у нашій системі буде проходити у окремих процесах. За допомогою обміну повідомленнями між процесами буде виконуватись моніторинг їх виконання з подальшим відображенням користувачу. Архітектура платформи з модулями високого рівню зображена на рисунку 2.1.

2.2.2 Архітектура клієнтської частини

При аналізі описаних вище вимог та з урахуванням того, що клієнтська частина має надавати інтерфейс для виконання всіх підтриманих серверною частиною дій клієнтську частину можна поділити на наступні модулі.

Модуль авторизації повинен відповідати за керування доступу до додатку, містити в собі відображення для входу користувача та зміни його паролю та містити всю пов’язану з цим функціональність, наприклад, перевірка авторизації користувача при переході на певну сторінку.

Модуль кабінету користувача потрібен для створення нових та перегляду існуючих пайплайнів, їх запуску, відображення прогресу їх виконання в реальному часі та перегляду історії виконань.

Модуль редагування пайплайнів потрібен для створення, редагування та видалення входів, перетворювачів та виходів.

Отже маємо, що клієнтська частина складається з 3 основних модулів

- модуль авторизації та аутентифікації;
- модуль кабінету користувача;
- модуль редагування ETL-пайплайнів.

Також вона містить 2 додаткових модулі, а саме:

– базовий модуль, який містить певні компоненти та сервіси, які повторно використовуються у всіх інших модулях;
– модуль імпортованих компонентів, який потрібен для роботи зі сторонніми бібліотеками компонентів для того, щоб не завантажувати одразу всю бібліотеку, якщо необхідна лише невелика кількість компонентів з цієї бібліотеки. Це, в свою чергу, дозволить зменшити загальний розмір клієнтського застосунку.

Архітектура клієнтської частини схематично зображена на рисунку 2.2. Більш детальний опис цій архітектурі надається в індивідуальній частині №2.

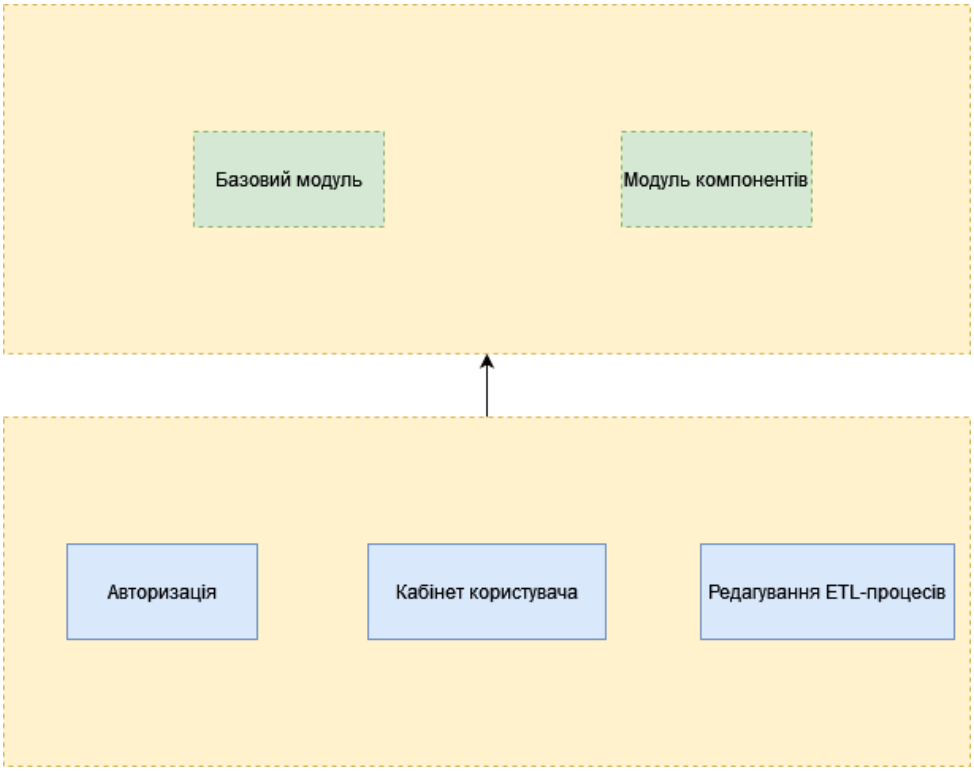


Рисунок 2.2 – Архітектура клієнтської частини

2.2.3 Архітектура серверної частини

Розглянемо можливі варіанти розбиття серверної частини на модулі. Виділення окремих модулів будемо виконувати за логічним призначенням окремого модулю.

По-перше, наша платформа повинна мати логіку авторизації для обмеження доступу неавторизованим користувачам. Навіть у випадку, коли

платформа буде ставитись у внутрішню мережу підприємства, авторизація та аутентифікація необхідні хоча б для фіксації та відокремлення дій різних користувачів. Тож першим модулем має стати модуль авторизації.

Ми вже знаємо, що уся платформа концентрована навколо ETL-процесів. Але очікувана логіка має занадто багато різноманітних операцій, тож створити окремний модуль під усі операції від конфігурації ETL-процесу до його виконання та моніторингу було б нелогічно. Тож необхідна декомпозиція модулю. Увесь процес роботи з ETL-процесами можна поділити на три великі кроки:

- перевірка та налаштування з’єднань з джерелом та сховищем даних;
- створення конфігурації ETL-процесу;
- виконання, моніторинг роботи та перегляд результатів виконання ETL-процесу.

Ми вже бачимо три потенціальні модулі серверної частини. Можна було б на цьому і завершити розбиття, проте, конфігурація ETL-процесу все ще доволі великий модуль, який можна було б розділити на підмодулі. По одному підмодулю можна виділити на кожний крок конфігурації ETL-процесу, так як конфігурації вийчамів, перетворювачів та вивантажувачів є окремими процесами, що значно відрізняються. Тоді серверну частину можна поділити таким чином:

- модуль авторизації;
- модуль виймання даних із джерел;
- модуль фільтрації та трансформації даних;
- модуль збереження даних до сховищ;
- модуль виконання ETL-пайплайну.

Тепер більш детально розглянемо, за що конкретно відповідатиме кожен виділений модуль.

Модуль авторизації відповідає за керування користувачами, їх аутентифікацію та авторизацію у системі. Також він частково відповідальний за ініціалізацію системи за створення початкового адміністратора системи.

Модуль виймання даних із джерел повністю відповідає за взаємодію з джерелами та сховищами даних, як на етапі конфігурації ETL-процесу, так і на етапі виконання ETL-процесу та моніторингу. На етапі конфігурації через цей модуль мають виконуватися дії по вийманню даних про зареєстровані у системі бази даних, що можуть бути використані у ETL-процесі, та дії, що пов’язанні з вийманням інформації про схему бази даних та відповідних таблиць. На етапі виконання та моніторингу через цей модуль мають виконуватись операції ETL-процесу, що потребують комунікації з базами даних. Наприклад, під час виймання початкових даних та під час здереження даних до сховищ.

Три останні модулі, а саме модуль витягнення даних із джерел, модуль трансформації даних та модуль збереження даних до сховища, як вже було відмічено, можна об’єднати у великий модуль більш високого рівня. Цей модуль буде відповідати за конфігурацію ETL-пайплайнів. Усі три модулі так чи інакше взаємодіють один з одним. Кожен з них формує окремий крок виконання пайплайну, а сам пайплайн представляє собою граф без циклів, вузлами якого є кроки виконання. Приклад пайплайну наведений на рисунку 2.3.

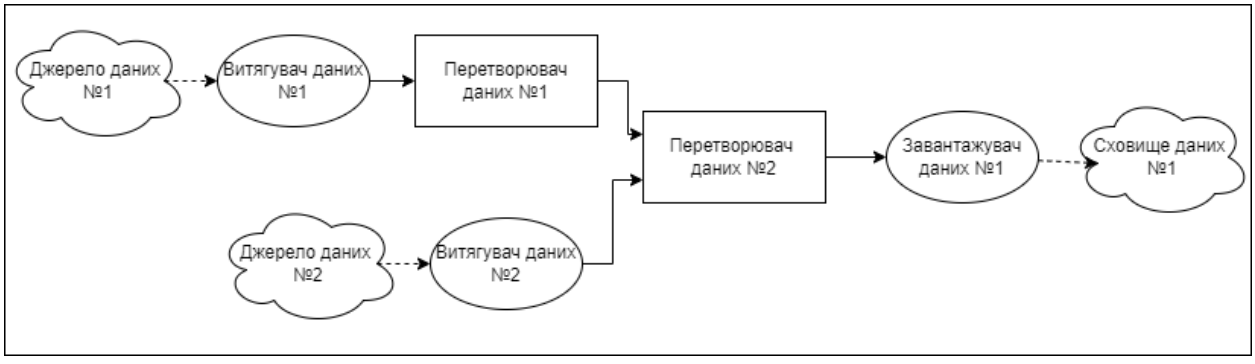


Рисунок 2.3 – Візуалізація ETL-пайплайну

Слід зауважити, що елементи відповідальні за вивантаження даних з джерел та за завантаження даних до сховищ мають спільну рису – кожен з них

взаємодіє з базою даних тим чи іншим чином. В такому випадку доволі очевидним стає рішення виділити керування даними для взаємодії з базами даних окремо від цих кроків. Ми можемо зберігати інформацію про з'єднання з базами даних окремо та при конфігурації обирати джерела даних з тих, що вже присутні у системі. Звичайно, усі параметри, потрібні для з'єднання з базою даних потрібно зберігати у зашифрованому вигляді.

2.3 Конструювання програмного забезпечення

Для розробки серверної частини була обрана платформа Node.js[9] з використанням мови TypeScript[10] та фреймворку NestJS[11]. Цей фреймворк дозволяє будувати серверні застосунки, що легко та швидко масштабуються. Для зберігання службових даних платформи ми обрали NoSQL базу даних MongoDB. Таке рішення є логічним через велику вкладеність структури пайплайну та через те, що в більшості випадків ми нам треба діставати одразу увесь пайплайн з усіма присутніми в ньому елементами, які у свою чергу теж містять потрібні дані.

Для розробки клієнтської частини розробляється веб-додаток на мові програмування TypeScript та фреймворком Angular[12]. Цей фреймворк дозволяє будувати модульні клієнтські застосунки, готові інструменти для реалізації повноцінного розділення даних та їх відображення, готові інструменти для реалізації найбільш застосованих патернів.

2.4 Аналіз безпеки даних

У нашій платформі безпеці даних приділяється велика увага. По-перше, службова база даних платформи буде доступною для під'єднання лише в межах її поду (Pod) у кластері Kubernetes. По-друге, ми маємо зберігати в базі даних вразливу інформацію. Такою в нашому випадку є паролі користувачів та параметри під'єднання до сторонніх баз даних. Для зберігання паролів користувачів ми використовуємо загальновживаний підхід з їх хешування.

Хешування проводиться за допомогою алгоритму bcrypt. У результаті роботи він віддає скомпоновану строку, що містить версію алгоритму, сіль та сам хеш. Сіль є модифікатором початкової строки для зменшення ймовірності колізій двох паролів та ускладнення підбору паролів у разі витоку даних з бази. За потреби перевірки правильності пароля, ми проводимо хешування пароля та порівнюємо з відповідним хешем збереженим у базі даних.

Як вже було вказано, у базі даних ми також зберігаємо параметри під'єднання до баз даних, які використовуються під час конфігурації та виконання ETL-процесів. У цьому випадку хешування вже не підходить, так як нам потрібно мати можливість відновити параметри з'єднання. Для шифрування ми використовуємо алгоритм AES256-GCM, що наразі вважається найоптимальнішим рішенням для шифрування даних. Отримані зашифровані дані та параметри шифрування ми зберігаємо у базі даних. Ключ шифрування ми зберігаємо у змінній оточення, в якому розгорнута платформа. Коли нам потрібні зашифровані параметри з'єднання, ми дешифруємо їх за допомогою ключа та параметрів шифрування, що збережені поряд з зашифрованими даними.

Висновки до розділу

У даному розділі нами була виконана змістовна постановка задачі. Також була описана та обґрунтована загальна архітектура розроблюваної платформи, виділені логічні модулі, на які можна поділити платформу. Також поверхнево були описані архітектурні та програмні рішення по кожному з модулів платформи. Більше детально про архітектуру модулів викладено у індивідуальних частинах роботи.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Призначення та предмет тестування

Призначенням тестування є перевірка якості програмного забезпечення та перевірка відповідності розробленого програмного забезпечення функціональним вимогам.

Предметом тестування є платформа як цільна система. Тестування окремих модулів серверної та клієнтської частин окремо було виконано у відповідних індивідуальних частинах дипломного проєкту. У даному випадку тестування сценаріїв роботи застосунку буде проводитись для працюючих у зв’язці усіх компонентів платформи.

3.2 Функціональність, що підлягає тестуванню

Тестування буде виконуватися таким чином, що функціональні вимоги будуть перевірені у тій послідовності, у якій потенційний користувач буде використовувати платформу. Тоді перевіряються наступні варіанти використання:

- як неавторизований користувач, ініціалізація системи;
- як неавторизований користувач, авторизація у системі;
- як авторизований користувач, зміна паролю для входу до системи;
- як авторизований користувач, створення конфігурації ETL-процесу;
- як авторизований користувач, тестування параметрів з’єднання з базою даних;
- як авторизований користувач, додавання до системи параметрів з’єднання з базою даних;
- як авторизований користувач, виймання переліку збережених параметрів з’єднання;

- як авторизований користувач, додавання виймача даних до ETL-пайплайну;
- як авторизований користувач, генерація шаблону перетворювача ETL-пайплайну;
- як авторизований користувач, додавання перетворювача даних до ETL-пайплайну;
- як авторизований користувач, додавання виймача даних до ETL-пайплайну;
- як авторизований користувач, виконання зконфігурованого ETL-процесу;
- як авторизований користувач, моніторинг процесу виконання ETL-процесу
- як авторизований користувач, перегляд результату виконання ETL-процесу

3.3 Підхід до тестування

У даній роботі для тестування функціональності розробленої платформи був використаний функціональний підхід, що зумовлюється необхідністю перевірити правильність роботи всіх модулів платформи у сукупності.

Під функціональним тестуванням мається на увазі тестування варіантів використання на предмет співпадіння очікуваного результати з отриманим під час проходження того варіанту використання.

3.4 Вимоги до середовища тестування

Тестування має виконуватись на машині з встановленою операційною системою Linux. У середовищі має бути встановлений Docker, в якому буде розгортатись клієнтська та серверна частини платформи та службова база даних. Також з машини, на якій виконується тестування мають бути доступні

бази даних, які будуть використовуватись при тестуванні в якості джерел або сховищ даних.

3.5 Опис тестів

Як вже було вказано, тести йдуть у порядку, в якому виконується головний сценарій роботи з нашою платформою – конфігурування та виконання ETL-процесу. Надалі надано описа тестів з ілюстраціями та отриманими результатами.

Першим кроком під час взаємодії з платформою є ініціалізація платформи та створення головного адміністратора. План та результати тестування наведено у таблицях 3.1 та 3.2.

Таблиця 3.1 – Тестування ініціалізації платформи

Мета тесту	Перевірка правильності роботи ініціалізації платформи
Початковий стан	Неініціалізована платформа
Вхідні дані	-
Схема проведення тесту	Одразу після розгортання платформи зайти на неї
Очікуваний результат	Створено головного адміністратора та надано логін та пароль для нього
Стан програмного продукту після проведення тесту	Створено головного адміністратора. Дані про нього збережено до бази даних.
Результат тесту	Пройдений

Таблиця 3.2 – Тестування неможливості повторної ініціалізації платформи

Мета тесту	Перевірка неможливості повторної ініціалізації платформи
Початковий стан	Ініціалізована платформа з присутнім головним адміністратором.
Вхідні дані	-
Схема проведення тесту	Спробувати виконати повторну ініціалізацію системи.
Очікуваний результат	Запит повертає помилку
Стан програмного продукту після проведення тесту	Новий користувач не створюється
Результат тесту	Пройдений

Наступним кроком після ініціалізації системи є авторизація. План та результати тестування авторизації наведені у таблицях 3.3 та 3.4.

Таблиця 3.3 – Тестування авторизації

Мета тесту	Перевірка правильності роботи авторизації
Початковий стан	Ініціалізована платформа
Вхідні дані	Правильні логін та пароль
Схема проведення тесту	Перейти на сторінку авторизації та ввести логін з паролем
Очікуваний результат	Користувача авторизовано
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.4 – Тестування неможливості авторизації при неправильному логіні або паролі

Мета тесту	Перевірка правильності роботи авторизації
Початковий стан	Ініціалізована платформа
Вхідні дані	Неправильні логін та пароль
Схема проведення тесту	Перейти на сторінку авторизації та ввести логін з паролем
Очікуваний результат	Користувача не авторизовано
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Одразу після авторизації користувач повинен змінити пароль. План та результати тестування цього функціоналу наведені у таблиці 3.5.

Таблиця 3.5 – Тестування функціонування зміни паролю

Мета тесту	Перевірка правильності роботи зміни паролю
Початковий стан	Користувач авторизований у системі
Вхідні дані	Правильний логін, старий пароль та новий пароль
Схема проведення тесту	Перейти на сторінку авторизації та ввести описані вхідні дані
Очікуваний результат	Пароль змінено
Стан програмного продукту після проведення тесту	Змінено пароль користувача
Результат тесту	Пройдений

Нехай користувач хоче створити конфігурацію ETL-процесу. Протестуємо створення конфігурації ETL-процесу. План та результати тестування цього функціоналу наведені у таблиці 3.6.

Таблиця 3.6 – Тестування створення конфігурації ETL-процесу

Мета тесту	Перевірка правильності роботи створення ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Назва конфігурації ETL-процесу
Схема проведення тесту	Перейти на сторінку зі списком присутніх конфігурацій ETL-процесів, натиснути кнопку створення нового ETL-процесу та ввести назву ETL-пайплайну
Очікуваний результат	Повідомлення про успішне створення конфігурації з переходом на сторінку новоствореної конфігурації
Стан програмного продукту після проведення тесту	Порожню конфігурацію ETL-процесу збережено до бази даних.
Результат тесту	Пройдений

Після створення конфігурації ETL-процесу її треба заповнити. Першим кроком є додавання виймача даних. Проте для цього потрібні параметри з’єднання з базою даних. Тобто, потрібно почергово перевірити правильність параметрів з’єднання, зберігти параметри з’єднання у базі даних та переглянути присутні у системі параметри з’єднання, щоб обрати потрібні. План та результати тестування описаного функціоналу наведені у таблицях 3.7 – 3.10.

Таблиця 3.7 – Тестування перевірки параметрів з’єднання з базою даних при правильних введених параметрах

Мета тесту	Тестування перевірки параметрів з’єднання з базою даних при правильних введених параметрах
Початковий стан	Користувач авторизований у системі
Вхідні дані	Тип бази даних та правильні параметри з’єднання
Схема проведення тесту	Перейти на екран додавання параметрів з’єднання. Ввести дані та натиснути кнопку, що виконує тестування з’єднання
Очікуваний результат	Повідомлення про успішність тестового з’єднання з базою даних
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.8 – Тестування перевірки параметрів з’єднання з базою даних при неправильних введених параметрах

Мета тесту	Тестування перевірки параметрів з’єднання з базою даних при неправильних введених параметрах
Початковий стан	Користувач авторизований у системі
Вхідні дані	Тип бази даних та неправильні параметри з’єднання
Схема проведення тесту	Перейти на екран додавання параметрів з’єднання. Ввести дані та натиснути кнопку, що виконує тестування з’єднання
Очікуваний результат	Повідомлення про неуспішність тестового з’єднання з базою даних та список помилок

Продовження таблиці 3.8

Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.9 – Тестування збереження параметрів з’єднання з базою даних

Мета тесту	Тестування збереження параметрів з’єднання з базою даних
Початковий стан	Користувач авторизований у системі
Вхідні дані	Тип бази даних, правильні параметри з’єднання та назва параметрів з’єднання
Схема проведення тесту	Перейти на екран додавання параметрів з’єднання. Ввести дані та натиснути кнопку, що виконує збереження параметрів з’єднання
Очікуваний результат	Повідомлення про успішність збереження параметрів з’єднання та ідентифікатор збережених параметрів ‘єднання
Стан програмного продукту після проведення тесту	До бази даних збережено параметри з’єднання у зашифрованому вигляді
Результат тесту	Пройдений

Таблиця 3.10 – Тестування відображення присутніх параметрів з’єднання з базою даних

Мета тесту	Тестування відображення присутніх параметрів з’єднання з базою даних
Початковий стан	Користувач авторизований у системі

Продовження таблиці 3.10

Вхідні дані	-
Схема проведення тесту	Перейти на екран з присутніми у системі параметрами з’єднання
Очікуваний результат	Список з назв, типів та ідентифікаторів збережених параметрів з’єднання. Параметри з’єднання повертатися не повинні.
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Тож маючи параметри з’єднання з базою даних ми можемо додати виймач даних до конфігурації ETL-процесу. План та результати тестування описаного функціоналу наведені у таблицях 3.11 та 3.12.

Таблиця 3.11 – Тестування додавання правильної конфігурації виймача даних до конфігурації ETL-процесу

Мета тесту	Тестування додавання правильної конфігурації виймача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну, назва конфігурації виймача, ідентифікатор параметрів з’єднання, список назв таблиць
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації виймача до конфігурації ETL-процесу. Ввести вхідні дані.

Продовження таблиці 3.11

Очікуваний результат	Повідомлення про успішність збереження конфігурації виймача та відображення оновленої конфігурації ETL-процесу.
Стан програмного продукту після проведення тесту	До бази даних збережено конфігурацію виймача даних
Результат тесту	Пройдений

Таблиця 3.12 – Тестування неможливості додавання неправильної конфігурації виймача даних до конфігурації ETL-процесу

Мета тесту	Тестування неможливості додавання неправильної конфігурації виймача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну , назва конфігурації виймача, неіснуючий ідентифікатор параметрів з’єднання, список назв таблиць
Схема проведення тесту	Перейти на екран з присутніми у системі параметрами з’єднання
Очікуваний результат	Повідомлення про помилку під час перевірки правильності конфігурації виймача даних
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Наступним після виймачів даних в загальному випадку додаються перетворювачі даних. Слід зауважити, що під час конфігурації перетворювача виконується генерування шаблону коду перетворювача. Тож протестуємо описану функціональність. План та результати тестування описаного функціоналу наведені у таблицях 3.13 - 3.16.

Таблиця 3.13 – Тестування генерації шаблону скрипту перетворювача при правильних да введених даних

Мета тесту	Тестування генерації шаблону скрипту перетворювача при правильних да введених даних
Початковий стан	Користувач авторизований у системі
Вхідні дані	Назва мови програмування скрипту, список ідентифікаторів елементів з числа виймачів та перетворювачів даних
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації перетворювача даних до конфігурації ETL-процесу. Обрати входи перетворювача з числа виймачів та перетворювачів. Натиснути кнопку генерації шаблону скрипту.
Очікуваний результат	Згенерований шаблон скрипту перетворювача
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.14 – Тестування генерації шаблону скрипту перетворювача при правильних да введених даних

Мета тесту	Тестування генерації шаблону скрипту перетворювача при неправильних да введених даних
Початковий стан	Користувач авторизований у системі
Вхідні дані	Назва мови програмування скрипту, список ідентифікаторів елементів з числа виймачів та перетворювачів даних, один з яких не має відповідного йому перетворювача або виймача
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації перетворювача даних до конфігурації ETL-процесу. Обрати входи перетворювача з числа виймачів та перетворювачів. Натиснути кнопку генерації шаблону скрипту.
Очікуваний результат	Отримано повідомлення про помилку
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.15 – Тестування додавання правильної конфігурації перетворювача даних до конфігурації ETL-процесу

Мета тесту	Тестування додавання правильної конфігурації перетворювача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну, назва конфігурації перетворювача, список входів з числа виймачів та

Продовження таблиці 3.15

	перетворювачів, мова скрипту перетворення та правильний скрипт перетворення
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації перетворювача до конфігурації ETL-процесу. Ввести вхідні дані.
Очікуваний результат	Повідомлення про успішність збереження конфігурації перетворювача та відображення оновленої конфігурації ETL-процесу.
Стан програмного продукту після проведення тесту	До бази даних збережено конфігурацію перетворювача даних
Результат тесту	Пройдений

Таблиця 3.16 – Тестування неможливості додавання неправильної конфігурації перетворювача даних до конфігурації ETL-процесу

Мета тесту	Тестування неможливості додавання неправильної конфігурації перетворювача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну, назва конфігурації перетворювача, список входів з числа виймачів та перетворювачів, мова скрипту перетворення та неправильний скрипт перетворення
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації перетворювача до конфігурації ETL-процесу. Ввести вхідні дані.
Очікуваний результат	Повідомлення про помилку під час перевірки правильності конфігурації перетворювача даних

Продовження таблиці 3.16

Стан програмного продукту після проведення тесту	До бази даних збережено конфігурацію перетворювача даних
Результат тесту	Пройдений

Останнім кроком конфігурації ETL-процесу є конфігурація вивантажувачів даних. Перевіримо дану функціональність. План та результати тестування описаного функціоналу наведені у таблицях 3.17 та 3.18.

Таблиця 3.17 – Тестування додавання правильної конфігурації вивантажувача даних до конфігурації ETL-процесу

Мета тесту	Тестування додавання правильної конфігурації вивантажувача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну, назва конфігурації вивантажувача, список ідентифікаторів елементів з числа виймачів та перетворювачів даних
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації вивантажувачів до конфігурації ETL-процесу. Ввести вхідні дані.
Очікуваний результат	Повідомлення про успішність збереження конфігурації вивантажувача та відображення оновленої конфігурації ETL-процесу.
Стан програмного продукту після проведення тесту	До бази даних збережено конфігурацію вивантажувача даних
Результат тесту	Пройдений

Таблиця 3.18 – Тестування неможливості додавання неправильної конфігурації виймача даних до конфігурації ETL-процесу

Мета тесту	Тестування неможливості додавання неправильної конфігурації виймача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор ETL-пайплайну, назва конфігурації вивантажувача, список ідентифікаторів елементів з числа виймачів та перетворювачів даних, один з яких не має відповідного йому перетворювача або виймача
Схема проведення тесту	Перейти на конфігурацію ETL-процесу. Перейти на екран додавання конфігурації вивантажувачів до конфігурації ETL-процесу. Ввести вхідні дані.
Очікуваний результат	Повідомлення про помилку під час перевірки правильності конфігурації вивантажувача даних
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Після конфігурування ETL-процесу наступним кроком є його виконання, моніторинг та перегляд результатів виконання. Протестуємо описану функціональність. План та результати тестування описаного функціоналу наведені у таблицях 3.19 - 3.22.

Таблиця 3.19 – Тестування виконання ETL-процесу з заповненою конфігурацією

Мета тесту	Тестування виконання ETL-процесу з ідентифікатором існуючої конфігурації
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор заповненої конфігурації ETL-процесу
Схема проведення тесту	Перейти на екран з присутніми у системі конфігураціями ETL-процесів. Обрати та запустити на виконання конфігурацію ETL-процес за ідентифікатором
Очікуваний результат	Повідомлення про успішність початку виконання ETL-процесу з можливістю перейти до моніторингу виконання
Стан програмного продукту після проведення тесту	Почалося виконання ETL-процесу. Проміжні метрики виконання зберігаються у базі даних
Результат тесту	Пройдений

Таблиця 3.20 – Тестування виконання ETL-процесу з порожньою конфігурацією

Мета тесту	Тестування неможливості додавання неправильної конфігурації виймача даних до конфігурації ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор порожньої конфігурації ETL-процесу
Схема проведення тесту	Перейти на екран з присутніми у системі конфігураціями ETL-процесів. Обрати та запустити на виконання конфігурацію ETL-процес за ідентифікатором

Продовження таблиці 3.20

Очікуваний результат	Повідомлення про помилку під час запуску ETL-на виконання через те, що він не зконфігурований
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.21 – Тестування моніторингу виконання ETL-процесу

Мета тесту	Тестування моніторингу виконання ETL-процесу
Початковий стан	Користувач авторизований у системі
Вхідні дані	Ідентифікатор процесу виконання ETL-процесу
Схема проведення тесту	Перейти на екран, що містить ETL-процеси, що у процесі виконання. Натиснути на потрібний
Очікуваний результат	Метрики виконання ETL-процесу, що оновлюються у реальному часі
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Таблиця 3.22 – Тестування перегляду історії виконання ETL-процесів

Мета тесту	Тестування перегляду історії виконання ETL-процесів
Початковий стан	Користувач авторизований у системі
Вхідні дані	-
Схема проведення тесту	Перейти на екран, що містить ETL-процеси, що містить історію виконання ETL-процесів.

Продовження таблиці 3.22

Очікуваний результат	Список з історією виконання ETL-процесів
Стан програмного продукту після проведення тесту	Не змінений
Результат тесту	Пройдений

Висновки до розділу

У даному розділі було проведено аналіз якості та тестування розробленої платформи. Було визначено призначення та предмет тестування. Описано обраний підхід до тестування. Було протестовано платформу за допомогою функціонально тестування на відповідність функціональним вимогам. За результатами тестування усі тести пройшли успішно, що може свідчити про якість розробленого програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

4.1.1 Розгортання програмного забезпечення локально

Для розгортання як серверної, так і клієнтської частини необхідно встановити платформу NodeJS версії 16 або вище та пакетний менеджер yarn.

Для запуску серверної частини необхідно змінити .env файл, в якому записані змінні середовища виконання, додавши туди наступні змінні:

- ENC_KEY – ключ для шифрування даних про з'єднання користувачів. Представляє собою шістнадцятковий рядок;
- MONGODB_URI – рядок з'єднання до бази даних mongodb, в якій зберігаються дані, необхідні для роботи серверної частини;
- JWT_SECRET – ключ для підписування та перевірки JWT-токенів авторизації;
- JWT_EXPIRES_IN_SEC – час дії виданого JWT-токену авторизації (в секундах).

```
MONGODB_URI=mongodb://27017
ENC_KEY=5caa1272167beaedf85e70848866f53b0d0dd478211e00f05dc2d148608f0eba
JWT_SECRET=66556A576E5A723475377821412544 2A472D4B6150645367566B597032733576
JWT_EXPIRES_IN_SEC=86_400
```

Рисунок 4.1 – Заповнений файл змінних середовища виконання

Після цього необхідно виконати команду yarn (для встановлювання залежностей) та запустити сервер командою yarn start. За замовчуванням сервер використовує порт 3000.

Для розгортання клієнтської частини достатньо виконати команду yarn (для встановлювання залежностей) та запустити командою yarn start. За замовчуванням клієнт використовує порт 4200.

4.1.2 Розгортання програмного забезпечення на віртуальній машині

Для розгортання сервера на віртуальній машині необхідно виконати такі ж самі дії, як і для локального розгортання, тобто, завантажити NodeJS та yarn, та встановити змінні середовища, проте замість команди yarn start необхідно виконати команду yarn build && yarn start:prod.

Для розгортання клієнту на віртуальній машині необхідно завантажити код та змінити поле baseUrl в файлі environment.production.json, встановивши адресу, на якій розгорнуто сервер. Це посилання буде використовуватися у всіх клієнтах, які виконують запити на сервер. Після цієї зміни необхідно ввести команди ng build –prod та перенаправити застосований сервер (наприклад, nginx) на файл index.html. Варто зазначити, що всі маршрути домену (а не лише порожній маршрут) необхідно перенаправити на index.html, адже клієнтська маршрутизація відбувається динамічно.

Після розгортання клієнтської та серверної частини додаток доступ до додатку буде здійснюватися з того домену (чи IP-адреси), на якому розгорнута клієнтська частина.

4.1.3 Розгортання програмного забезпечення в контейнері

Розгортання додатку в контейнері значно підвищує гнучкість створення, розробки та розгортання додатку, адже при такому розгортанні гарантується, що контейнер буде працювати однаково на всіх машинах. Для контейнеризації додатку застосовується програма Docker – проект з відкритим кодом, який застосовується для розгортання додатків у виді переносних автономних контейнерів. Контейнер працює під управлінням операційної системи, містить файлову систему, мережевий доступ та все інше, що й звичайна віртуальна машина. Проте, на відміну від віртуальної машини, контейнери застосовують ядро операційної системи, на якій вони працюють, що робить їх значно легшими ніж віртуальна машина, тобто, мають менший розмір та швидше запускаються.

Для розгорнення додатку в контейнері застосовуються Docker-файли. Вони містять послідовність команд, яку треба виконати для того, щоб зібрати додаток в контейнер.

```
FROM node:16-alpine as builder

ENV NODE_ENV build

USER node
WORKDIR /home/node

COPY package*.json ./
RUN npm ci

COPY --chown=node:node . .
RUN npm run build \
    && npm prune --production

# ---

FROM node:16-alpine

ENV NODE_ENV production

USER node
WORKDIR /home/node

COPY --from=builder --chown=node:node /home/node/package*.json ./
COPY --from=builder --chown=node:node /home/node/node_modules/ ./node_modules/
COPY --from=builder --chown=node:node /home/node/dist/ ./dist/

CMD ["node", "dist/server.js"]
```

Рисунок 4.2 – Dockerfile для серверної частини

Цей файл містить двоетапну збірку для додатку. Перший етап компілює файли, наступний – запускає. Для створення контейнеру необхідно виконати команду *docker build -t {container_name} .* , де замість *{container_name}* необхідно вказати назву контейнеру. Після цього створений контейнер необхідно запустити за допомогою команди *docker run -p 3000:3000 {container_name},* де container_name – назва створеного контейнеру, 3000:3000 – вхідний та вихідний порт відповідно.

Аналогічні дії необхідно виконати для розгортання клієнтського додатку в контейнері.

Після цього за допомогою команди docker push контейнери можна опублікувати для подальшого використання на віртуальних машинах або в кластерах.

4.1.4 Розгортання програмного забезпечення в кластері

Найбільш практичним варіантом розгортанням додатку є розгортання його в кластері. Для керування кластером зазвичай застосовують оркестратори контейнерів, такі як, наприклад, Kubernetes, RedHat OpenShift або Docker swarm. Оркестратор контейнерів дає можливість координувати взаємодію декількох контейнерів та виконує їх автоматичне масштабування, виділяє контейнерам певні ресурси, створює мережі для обміну даних між контейнерами, займається моніторингом стану контейнерів, їх оновленням, керує доступами до контейнерів і т.д.

В нашому додатку розроблена схема розгорнення в кластері, який керується Kubernetes. Опис процесу оркестрації контейнерів та призначення створених об’єктів Kubernetes виходить за рамки цієї роботи, тому буде коротко описано лише процес розгортання. Існує два способи розгортання нашого додатку в kubernetes: ручна та автоматична.

Для ручного розгортання додатку в кластері необхідно налаштувати змінні оточення в файлі backend/secrets.yaml та frontend/secrets.yaml. Після цього з пристрою, який має доступ до кластеру Kubernetes необхідно виконати команду kubectl apply -f deployments/k8s. Також за допомогою ручного розгортання

Для автоматичного розгортання додатку в кластері застосовується Helm. При цьому в файлі values.yaml необхідно встановити всі змінні, описані вище для всіх типів розгортань, а також необхідно вказати, чи потрібно розгортати базу даних для цього проекту всередині контейнера, встановивши значення логічної змінної deployDatabase. Якщо база даних буде розгорнута в тому ж кластері, то рядок з’єднання не потрібно встановлювати (а навіть якщо його

встановити, він буде ігноруватись). В такому випадку сервер приєднається до розгорнутої в кластері бази даних. Не зважаючи на це, розгорнення бази даних в кластері не рекомендується. Для розгортання необхідно виконати команду *helm install etl-chart helm/ --values helm/values.yaml*

4.2 Робота з програмним забезпеченням

Розгорнуту інструкцію по роботі з програмним забезпеченням описано в документі «Керівництво користувача».

Висновки до розділу

В цьому розділі було детально описано процес розгортання, супроводу та роботи з програмним забезпеченням. Були розглянуті окремо 4 способи розгортання створеного програмного забезпечення, а саме: локальний, на віртуальній машині, в контейнері та в кластері. Всі способи були детально описані окремо для серверного та клієнтського додатку.

ВИСНОВКИ

В результаті виконання дипломного проекту було спроектовано та розроблено платформу створення, підтримки та моніторингу ETL-процесів.

Нами було проаналізовано предметну область. Було розглянуто особливості структури ETL-процесу та етапи його виконання. Також ми розглянули існуючі підходи до організації виконання ETL-процесів, якими є написання скриптів, які виконуватимуть ETL-процес, та використання готових програмних продуктів для цих цілей. Зосередившись на програмних реалізаціях ми розглянули існуючі типи таких програмних продуктів, виділили переваги та недоліки кожного типу. Після цього було виконано огляд успішних продуктів, що відповідають описаним типам.

Наступним було виконано моделювання платформи. Виконано опис компонентів системи та шляхів взаємодії між ними. Такими компонентами стали серверна та клієнтська частина. Кожен з компонентів був розбитий на окремі логічні модулі з описом їх призначення. На основі цього розбиття на модулі було розроблено архітектуру серверної та клієнтської частин.

Було визначено засоби для розробки платформи. Для написання серверної частини було обрано платформу Node.js з використанням мови TypeScript та фреймворку NestJS. В якості службової бази даних платформи було використано MongoDB. Для розробки клієнтської частини було використано також TypeScript і платформу Angular.

Після виконання конструювання серверної та клієнтської частини було проведено функціональне тестування, що перевірило реалізовану платформу на відповідність до функціональних вимог. Під час тестування покроково виконано ETL-процес від процесу його конфігурації до перегляду результатів виконання.

Після тестування було описано процес впровадження та супроводу платформи. Були описані 4 можливі варіанти розгортання платформи, а саме:

– локальне;

– розгортання на віртуальній машині;
– розгортання в контейнері;
– розгортання в кластері.

Отже, під час дипломного проектування нами було розроблено платформу, яка повністю відповідає визначеним вимогам. Дана платформа легко встановлюється до будь-якого середовища, насамперед до клауду, та є рішенням, що може використовуватися різними категоріями користувачів для конфігурації та виконання потрібних їм ETL-процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Ralph., Kimball, (2004). The data warehouse ETL toolkit : practical techniques for extracting, cleaning, conforming, and delivering data. Caserta, Joe, 1965-. Indianapolis, IN: Wiley. ISBN 0764579231. OCLC 57301227.

2) ETL What it is and why it matters. Архів оригіналу за 24 серпня 2017. Процитовано 24 серпня 2017.

3) Zhao, Shirley (2017-10-20). "What is ETL? (Extract, Transform, Load) | Experian". Experian Data Quality. Retrieved 2018-12-12.

4) Theodorou, Vasileios (2017). "Frequent patterns in ETL workflows: An empirical approach". Data & Knowledge Engineering. 112: 1–16. doi:10.1016/j.datak.2017.08.004. hdl:2117/110172.

5) David Loshin. ETL (Extract, Transform, Load) // Business Intelligence. — 2nd. — Morgan Kaufmann, 2012. — 400 p. — ISBN 978-0-12-385890-0.

6) Rankins, Ray; Bertucci, Paul; Jennsen, Paul (December 2002). Microsoft SQL Server 2000 Unleashed (2 ed.). Indiana: Sams. pp. 86–87. ISBN 97z80672324673. OCLC 474621100.

7) PostgreSQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/

8) MongoDB Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/

9) Node.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/uk/docs/

10) TypeScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/

11) NestJS Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nestjs.com/

12) Angular Documentation [Електронний ресурс] – Режим доступу до ресурсу: https:/angular.io/

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1011570263

Дата проверки:
14.06.2022 06:50:37 EEST

Тип проверки:
Doc vs Internet + Library

Дата отчета:
14.06.2022 06:53:46 EEST

ID пользователя:
76913

Название файла: **ІП-81_ДОБРЯНСЬКИЙ_САВАСТРУ_ПЗ**

Количество страниц: **46** Количество слов: **7949** Количество символов: **60819** Размер файла: **368.25 KB** ID файла: **1011441164**

10.6%
Совпадения

Наибольшее совпадение: **8.4%** с источником из Библиотеки (ID файла: **1011441166**)

0.28% Источники из Интернета	2	Страница 48
10.6% Источники из Библиотеки	137	Страница 48

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

КЛАУД-ПЛАТФОРМА СТВОРЕННЯ, ПІДТРИМКИ ТА МОНІТОРИНГУ ETL-ПРОЦЕСІВ (КОМПЛЕКСНА ТЕМА). ЗАГАЛЬНА ЧАСТИНА

Керівництво користувача

КПІ.ІП-8112.27.045440.03.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Інна СТЕЦЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавці:
_____ Богдан ДОБРЯНСЬКИЙ
_____ Станіслав САВАСТРУ

Київ – 2022

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Вхід у систему

Після запуску застосунку необхідно ввести стандартний логін і пароль для адміністратора. Його генерація відбувається залежно від способу розгортання, проте для локального розгортання достатньо відправити POST запит на маршрут /init (рисунок 1.1).

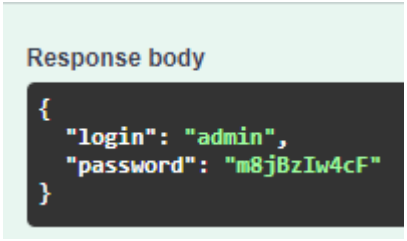


Рисунок 1.1 – Отриманий логін та пароль головного адміністратора

Після цього на сторінці входу користувач матиме можливість ввести цей логін і пароль для входу в додаток (рисунок 1.2).

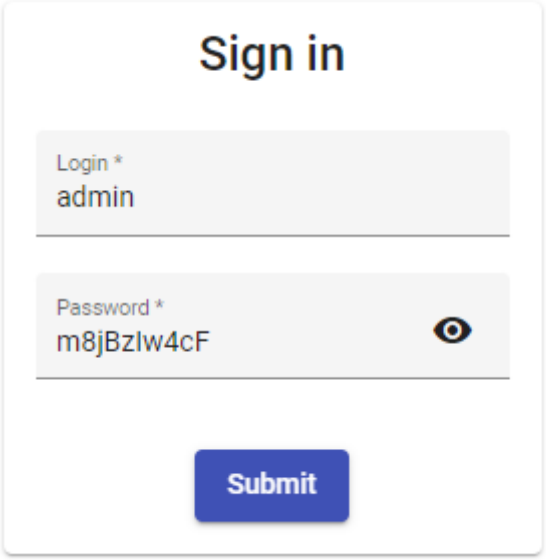


Рисунок 1.2 - Форма входу в додаток.

Після введення логіну та паролю та натисканню на кнопку Submit користувачу буде одразу запропоновано змінити пароль. Для цього необхідно ввести старий пароль, новий пароль та підтвердження нового паролю (рисунок 1.3).

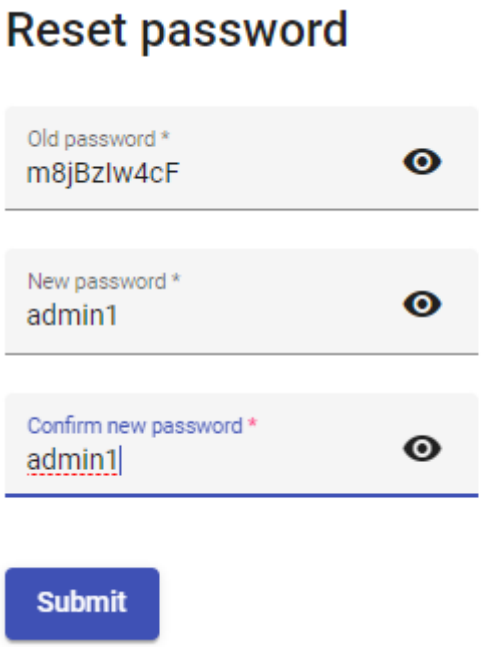


Рисунок 1.3 – Форма зміни паролю

1.2 Створення пайплайну

Після успішної зміни парою користувач побачить головну сторінку застосунку (рисунок 1.4).

Рисунок 1.4 – Головна сторінка

Оскільки користувач не має створених пайплайнів, він повинен натиснути на кнопку Add Pipeline, після чого він побачить вікно для введення назви пайплайну (рисунок 1.5).

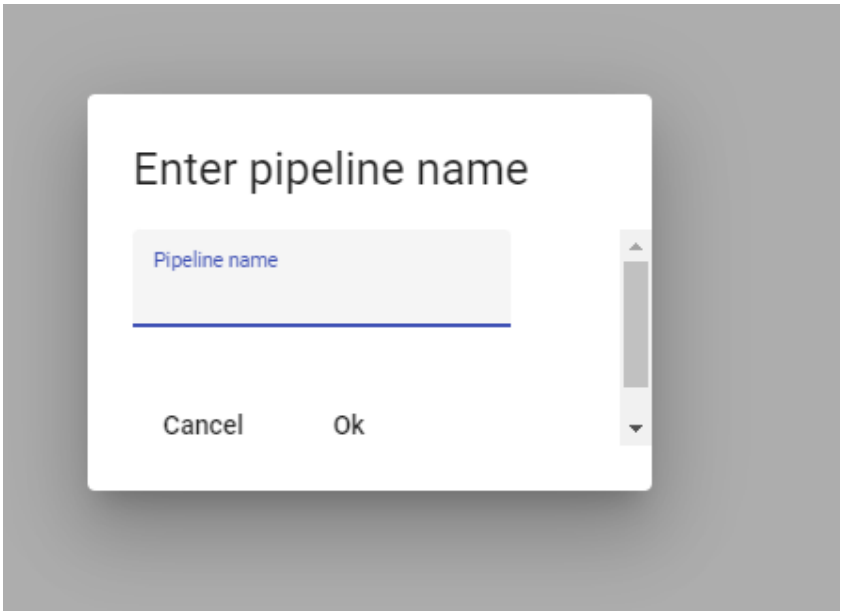


Рисунок 1.5 – Введення назви пайплайну

Після введення назви пайплайну та натискання кнопки OK користувача перенаправить на сторінку редагування пайплайну, де він може налаштувати входи, перетворювачі та виходи (рисунок 1.6).

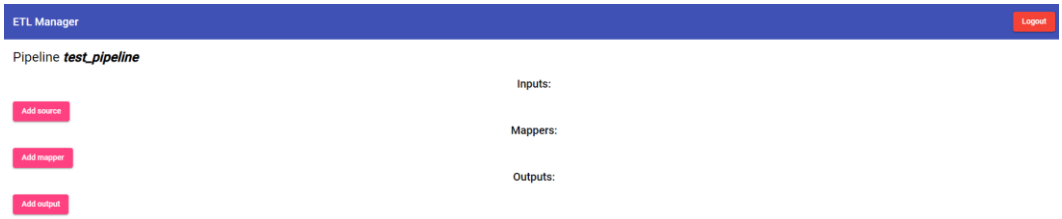


Рисунок 1.6 – Створений пайплайн

Як бачимо, створений пайплайн немає жодного входу, виходу та перетворювача.

Для додавання входу необхідно натиснути на кнопку Add Input. Після цього відкриється вікно додавання входу (рисунок 1.7).

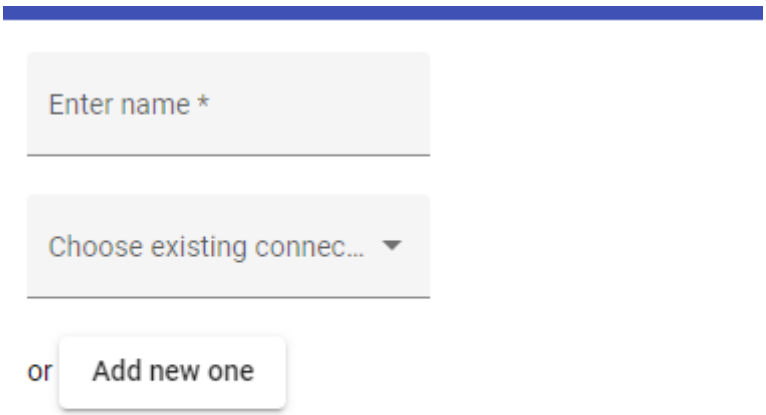


Рисунок 1.7 – Вікно додавання входу

Для того, щоб додати вхід необхідно ввести його назву та обрати джерело даних.

Для додавання джерела даних необхідно натиснути на кнопку Add New one, після чого відкриється вікно додавання джерела даних (рисунок 1.8).

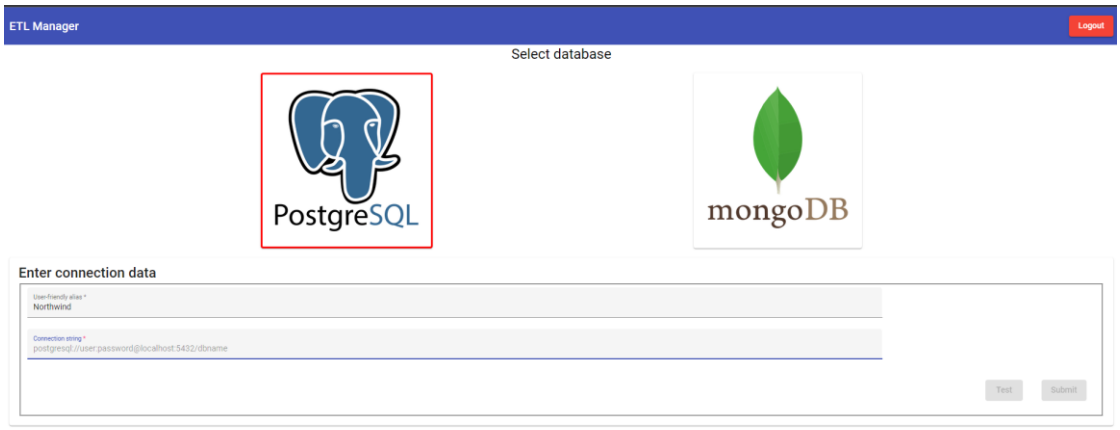


Рисунок 1.8 – Додавання джерела даних

Після введення рядку з’єднання користувача буде повернуто на сторінку додавання входу, де додане джерело даних з’явиться у випадаючому списку, та після обирання цього джерела даних з’являться таблиці (рисунок 1.9).

Рисунок 1.9 – Отримане джерело даних

В цьому списку необхідно обрати необхідні таблиці, які будуть застосовуватися з цього джерела даних для виконання ETL-процесу.

Після цього створений вхід буде доступний користувачу (рисунок 1.10)

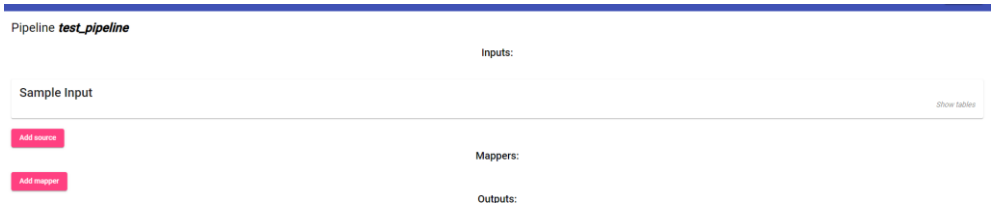


Рисунок 1.10 – Створений вхід

Після цього користувач може додати перетворювач. Для цього необхідно натиснути на кнопку Add Mapper. Відкриється сторінка додавання перетворювача (рисунок 1.11).

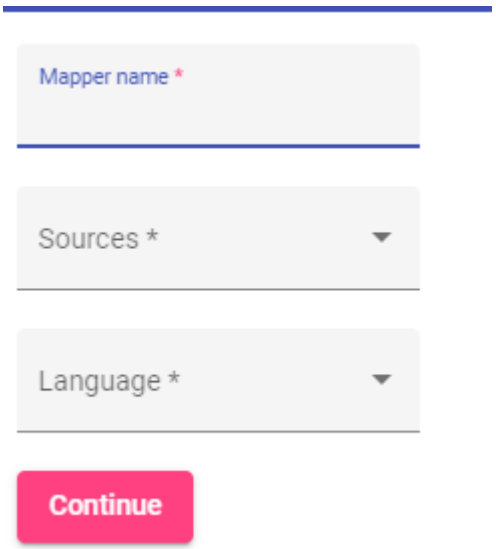


Рисунок 1.11 – Форма додавання перетворювача

В цій формі необхідно ввести назву перетворювача, обрати джерела та вибрати мову програмування для написання коду перетворювача. Після цього, користувач побачить згенерований код (рисунок 1.12).

Рисунок 1.12 – Згенерований код перетворювача

Після написання коду та збереження перетворювача користувач зможе побачити його на головній сторінці (рисунок 1.13).

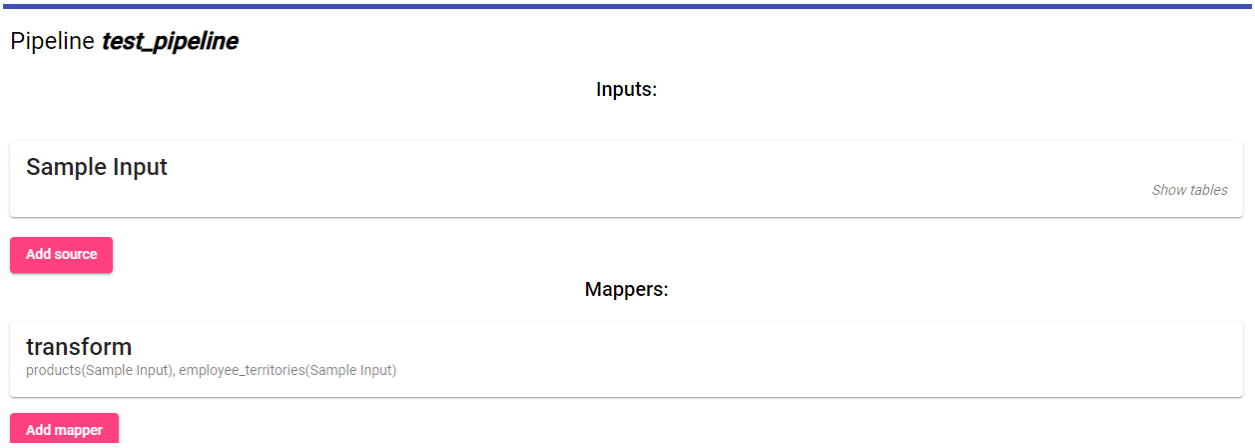


Рисунок 1.13 – Збережений перетворювач

Останнім кроком є додавання виходу ETL-процесу. Для цього необхідно натиснути на кнопку Add Output. Після цього відкриється сторінка додавання виходу (рисунок 1.14)

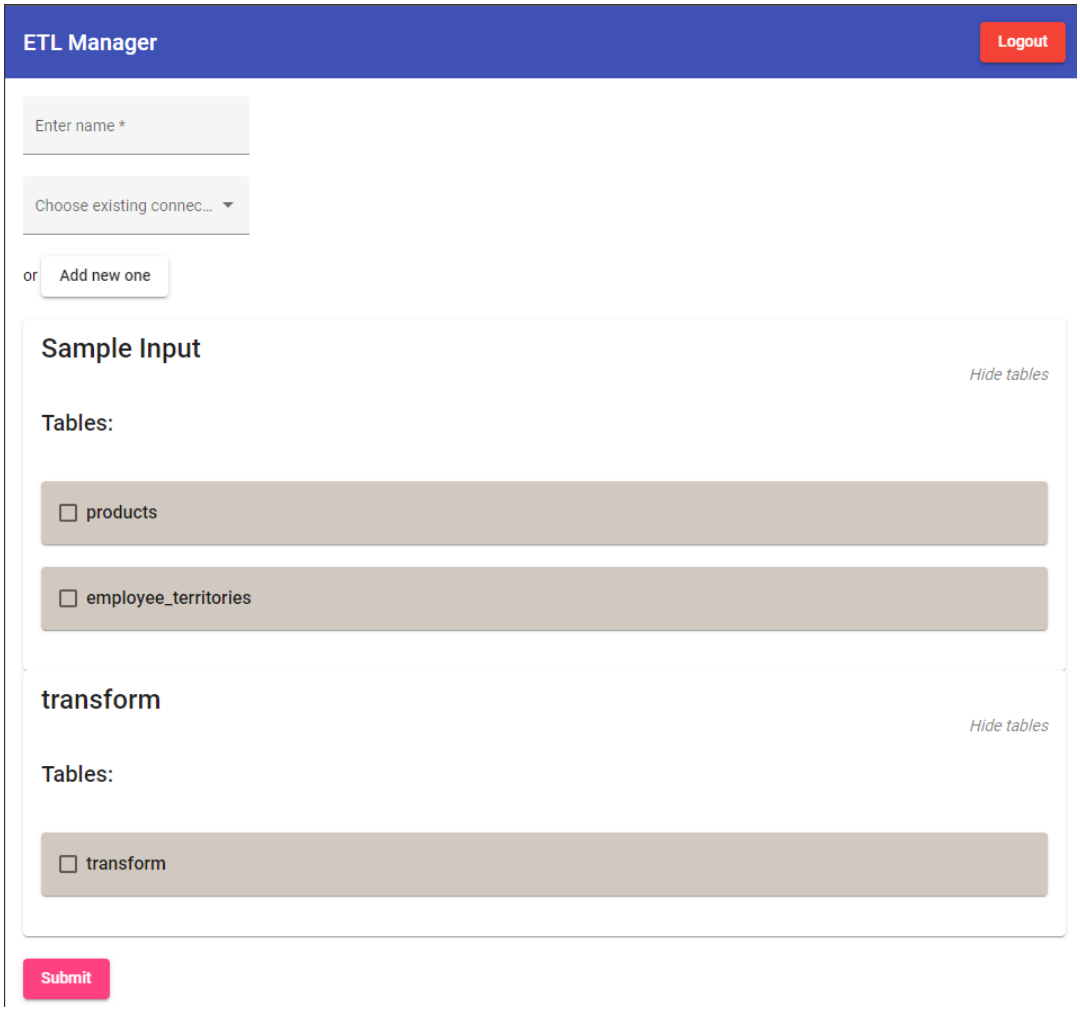


Рисунок 1.14 – Сторінка додавання виходу.

Для додавання виходу необхідно ввести назву виходу, обрати вихідне джерело даних (аналогічно з додаванням входу) та обрати дані для виходу (серед вхідних даних та даних перетворювачів). Після збереження виходу користувача перенаправить на сторінку редагування пайлпайну (рисунок 1.15).

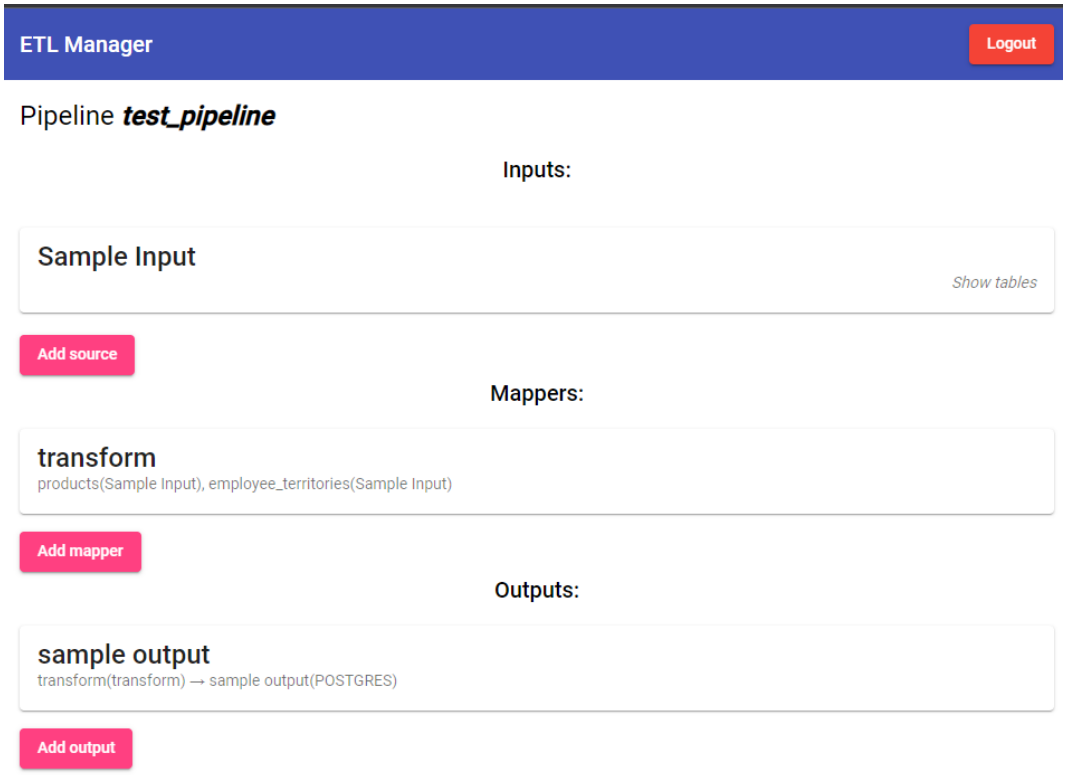


Рисунок 1.15 – Пайплайн зі створеним виходом

1.3 Виконання пайплайну

Після виконання цих кроків створено мінімальний пайплайн, який можна запустити. За необхідності можна додати більше входів, перетворювачів та виходів.

За допомогою кнопки Run можна запустити пайплайн. При цьому прогрес його виконання буде відображатися під назвою.

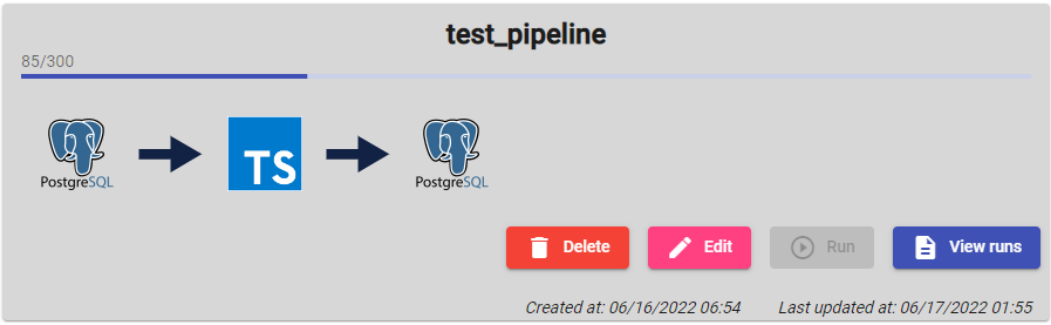


Рисунок 1.16 – Виконання пайплайну

За допомогою кнопки Delete можна також видалити пайплайн. При цьому відкриється вікно, в якому треба буде виконати підтвердження цієї дії (рисунок 1.17).

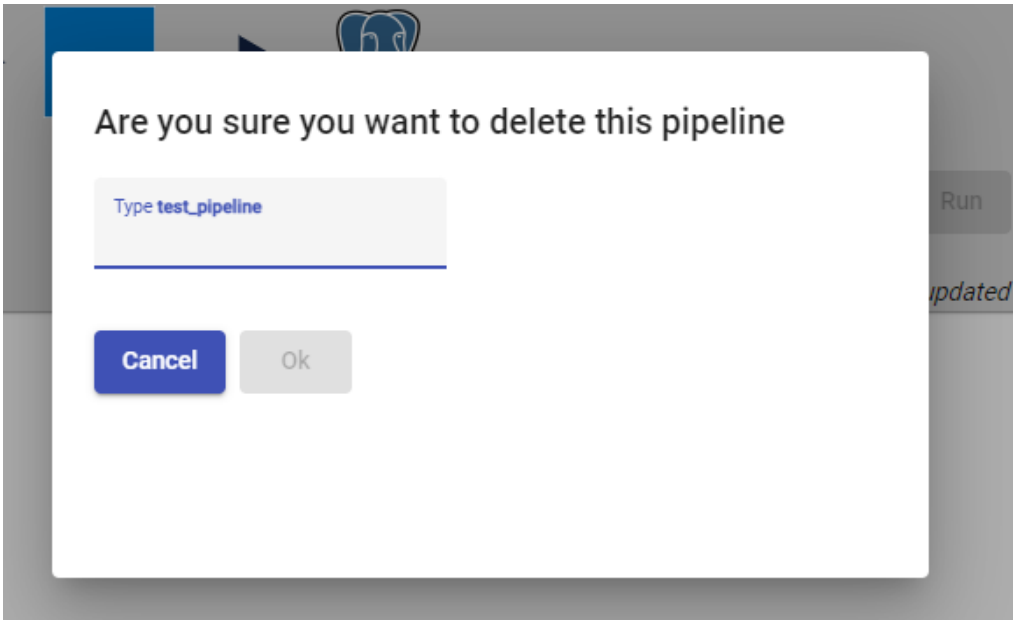


Рисунок 1.17 – Видалення пайплайну

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

КЛАУД-ПЛАТФОРМА СТВОРЕННЯ, ПІДТРИМКИ ТА МОНІТОРИНГУ ETL-ПРОЦЕСІВ (КОМПЛЕКСНА ТЕМА). ЗАГАЛЬНА ЧАСТИНА

Графічний матеріал

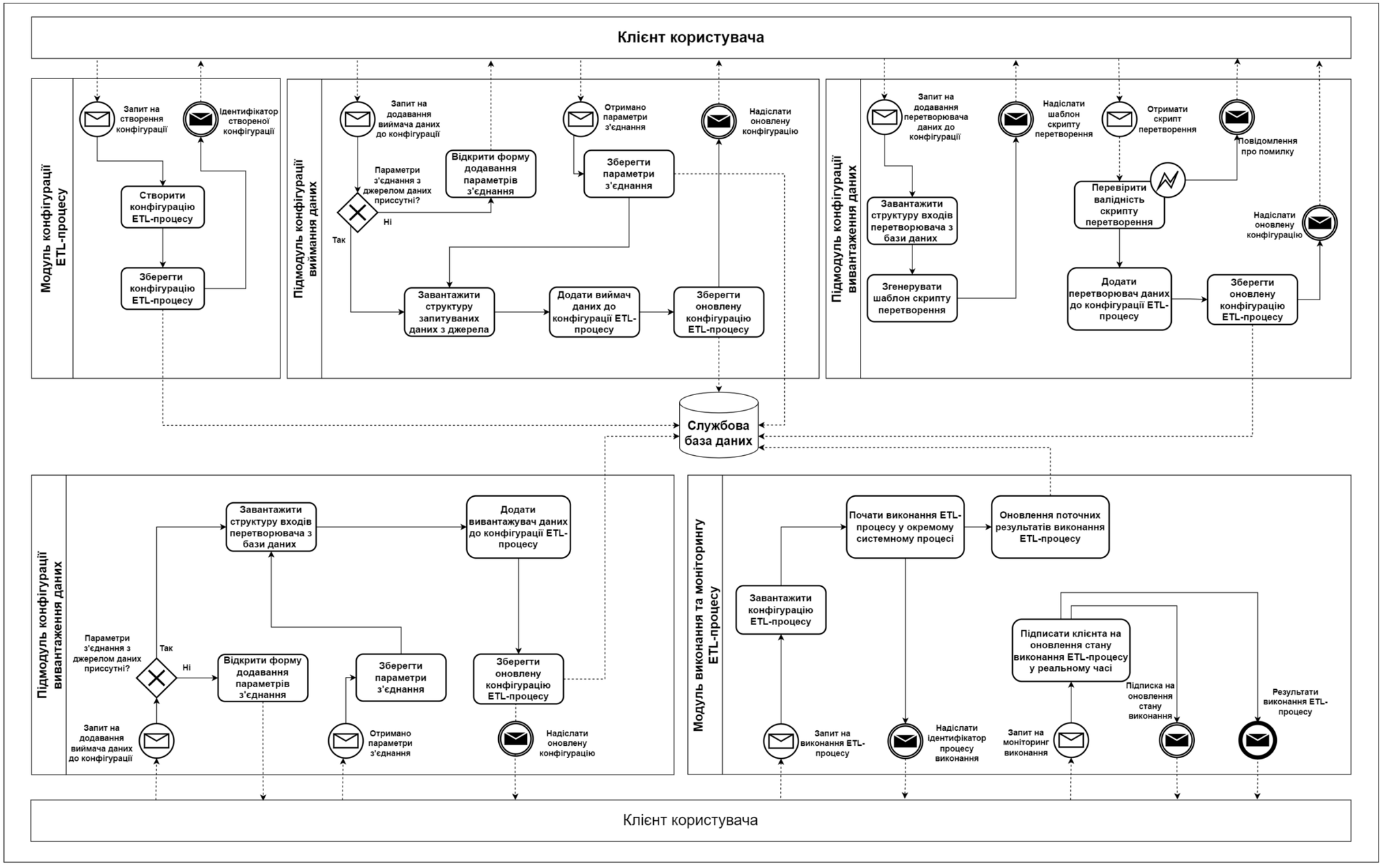
КПІ.ІП-8112.27.045440.04.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Інна СТЕЦЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

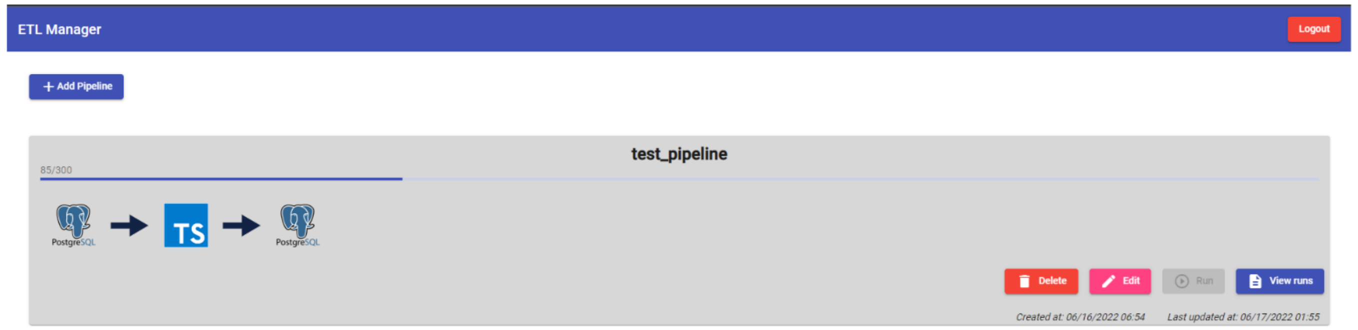
Виконавці:
_____ Богдан ДОБРЯНСЬКИЙ
_____ Станіслав САВАСТРУ

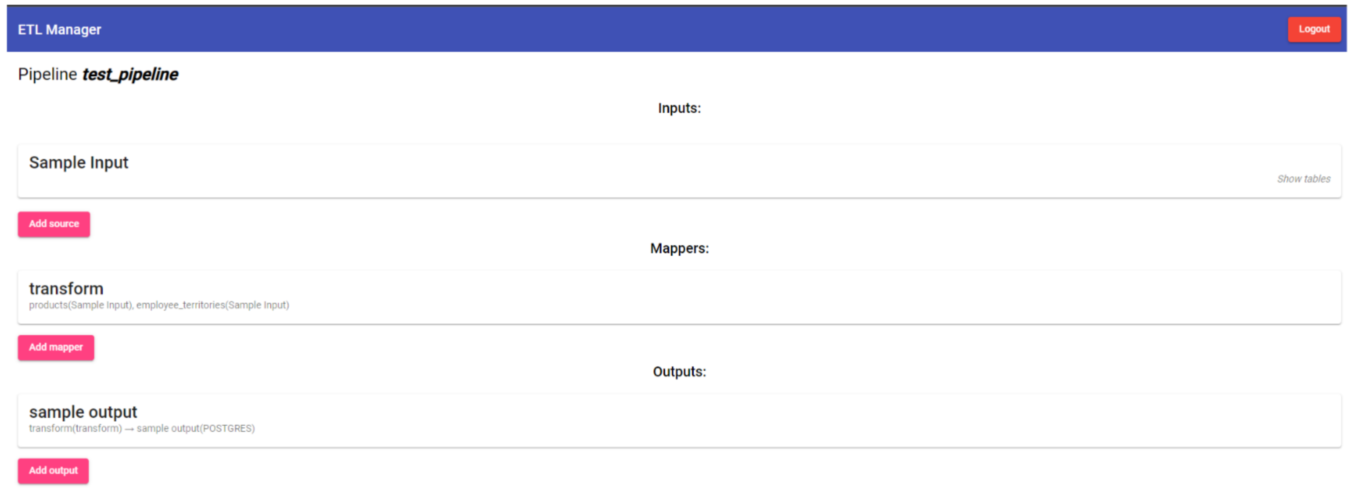
Київ – 2022



					КПІ.ІП-8112.27.045440.04.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності
Розробив		Савастру С.В.			Літера: — Маса: — Масштаб: —
Розробив		Добрянський Б.І.			Аркуш 1 / Аркушів 2
Перевірив		Стеценко І.В.			
Т. кон.					Клауд-платформа створення, підтримки та моніторингу ETL-процесів (комплексна тема) Загальна частина
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-81
Затвердив		Стеценко І.В.			

КПІ.ІП-8112.27.045440.04.99





Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-8112.27.045440.04.99.КЕ		
Розробив		Добрянський Б.І			Креслення вигляду екранних форм	Літера	Маса	Масштаб
Розробив		Савастру С.В.				Аркуш 2	Аркушів 2
Перевірив		Стеценко І.В.					
Т. кон.					Клауд-платформа створення, підтримки та моніторингу ETL-процесів(комплексна тема). Загальна частина	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-81	
Н. кон.		Ліщук К.І.					
Затвердив		Стеценко І.В.					