МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп’ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

17 травня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 – Комп'ютерні науки,

освітньо-наукової програми «Інформатика»

на тему: «Інформаційна технологія проєктування прикладних програмних інтерфейсів з підвищеними вимогами до інформаційної безпеки»

здобувача групи ІН.м-21.н Корольова Дмитра Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Дмитро КОРОЛЬОВ

_____

(підпис)

Керівник

асистент кафедри комп’ютерних наук,

к.м.-ф.н. Ольга ШУТИЛЄВА _____

(підпис)

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 – Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-21н Корольова Дмитра Сергійовича

1. Тема роботи: «Інформаційна технологія проєктування прикладних програмних інтерфейсів з підвищеними вимогами до інформаційної безпеки»
затверджую наказом по СумДУ від «08» березня 2024 року № 0234-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 17 травня 2024 року
3. Вихідні дані до кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Стандартний генетичний алгоритм. 3) Розробка інформаційної технології та програмного забезпечення 4) Аналіз результатів роботи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6. Консультанти до проекту (роботи) із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «04» березня 2024 р.
Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Стандартний генетичний алгоритм*		
3	*Розробка інформаційної технології та програмного забезпечення*		
4	*Аналіз результатів роботи*		
5	*Оформлення кваліфікаційної магістерської роботи*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 75 стор., 46 рис., 3 таблиць, 1 додаток, 20 джерел.

Обґрунтування актуальності теми роботи: У сучасному цифровому середовищі безпека прикладних програмних інтерфейсів (API) стає дедалі важливішою, оскільки через мережу Інтернет обмінюється велика кількість даних. Контроль доступу до API гарантує, що лише уповноважені користувачі мають доступ до обмежених ресурсів, а захист від зловмисного вмісту повідомлень вберігає систему від атак.

Об'єкт дослідження: Процес управління безпекою API.

Мета роботи: Визначення та розробка ефективних моделей управління безпекою прикладних програмних інтерфейсів. Розробка систем протоколювання та аудиту для виявлення та аналізу спроб несанкціонованого доступу або інших безпекових інцидентів, аналізу та зберігання логів безпеки.

Методи дослідження: Аналіз сучасних підходів до управління безпекою API, експериментальне моделювання та оцінка ефективності застосованих методів.

Результати: Розроблено та впроваджено модель управління безпекою API, яка враховує сучасні виклики кібербезпеки та дозволяє покращити безпеку API для електронної комерції. Розроблено систему протоколювання та аудиту для виявлення та аналізу спроб несанкціонованого доступу або інших безпекових інцидентів, аналізу та зберігання логів безпеки.

API, БЕЗПЕКА, APIGEE X, ПЛАТІЖНА СИСТЕМА, ЛОГУВАННЯ

ЗМІСТ

ВСТУП

Актуальність. Безпека API є критично важливою в сучасному цифровому середовищі, де велика кількість даних обмінюється через мережу Інтернет. Контроль доступу до API гарантує, що лише уповноважені користувачі можуть отримувати доступ до ресурсів, що мають обмежений доступ. Захист від зловмисного вмісту повідомлень вберігає систему від атак, таких як впровадження зловмисного коду або перехоплення даних.

Об'єкт дослідження. Процес управління безпекою API.

Предмет дослідження. Інформаційна технологія управління безпекою API з порталом для розробників на базі платформи Apigee X.

Суперечність. У роботі вирішується суперечність між необхідністю швидкої та надійної безпеки API та відсутністю моделей які враховують сучасні виклики кібербезпеки.

Гіпотеза дослідження. Якщо використати платформу Apigee X та доповнити загальну модель безпеки API механізмами Advanced Security, то можна покращити безпеку API.

Новизна результатів. На відміну від існуючої моделі орієнтовану на загальну модель безпеки Resource Owner Password запропоновано додаткову модель безпеки Client Credentials з використанням алгоритмів управління трафіком.

Структура роботи. Загальне ознайомлення, теоретичні та експериментальні дослідження, обробка інформації, впровадження результатів.

Впровадження. Результати роботи будуть використані впровадження моделі безпеки API в сфері електронної комерції.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність API для електронної комерції

Прикладні програмні інтерфейси або загальновживана абревіатура API – це спосіб зв’язку між програмами в межах платформи, а також між платформами та іншими інтегрованими системами. API – це спосіб для програм спілкуватися один з одним. API дозволяють одній програмі, наприклад платформі електронної комерції, надавати послуги іншим програмам. Ці послуги можуть включати вміст каталогу, керування замовленнями, інформацію про ціни та дані про клієнтів. Розробники створюють і публікують API, щоб допомогти підключити онлайн-торговців і покупців через веб-сайти та платформи, відіграючи ключову роль у перетворенні даних у корисну бізнес-інформацію та інтеграції з іншими службами для простоти використання [1].

Інтеграція є важливою частиною систем електронної комерції. Найпоширенішим використанням API є інтеграція системи електронної комерції з веб-сайтом продавця, щоб відображати інформацію про продукти та транзакції відвідувачам веб-сайту. Інше використання – синхронізація даних із серверними системами, наприклад, обмін даними з постачальником послуг доставки щодо замовлень і відправлень для оптимізації логістики або оновлення даних клієнтів у системі керування взаємовідносинами з клієнтами (CRM).

Використання API дає змогу онлайн-продавцям автоматизувати та координувати такі функції, як виставлення рахунків, виконання замовлень, маркування, керування кур’єрською доставкою, підтримка клієнтів, рекомендації та стимулювання збуту. [1]

API електронної комерції дає змогу взаємодіяти із системами в екосистемі електронної комерції, щоб відображати відповідну інформацію відвідувачам веб-сайту та синхронізувати діяльність електронної комерції із зовнішніми системами. Вебсайти електронної комерції отримують численні

запити, які вони повинні обробляти та відповідати на них одночасно. Як правило, API електронної комерції використовують архітектуру REST, вимагаючи запитів HTTP для доступу до інформації на сервері.

REST API має легкий дизайн і може координуватися майже з будь-яким інструментом. Він підтримує низку кросплатформенних розробок, мов програмування та процесів автентифікації безпеки.[2]

Більшість платформ електронної комерції, таких як Shopify, Magento та WooCommerce, надають API, які дозволяють інтегрувати їхні можливості з вебсайтом або внутрішніми системами. Деякі з найвідоміших API електронної комерції на ринку включають такі платформи, як Amazon API, Shopify API та eBay. Крім функцій інтеграції, ці API також є ефективними інструментами для моніторингу та аналітики даних.

Переваги API:

- Можливість повторного використання – API використовують сервісно-орієнтований підхід, який дає змогу повторно використовувати різні програми та платформи. Багаторазові програми допомагають скоротити час і зусилля, витрачені на проект, підвищуючи продуктивність розробника.
- Масштабованість – основною функцією API є підключення до цільової системи без модифікації оригінальної системи, обладнання чи програми. API дозволяє створити з’єднання за допомогою кількох рядків коду, пропонуючи масштабованість для більш складних транзакцій.
- Розширюваність – API дозволяє різним програмним компонентам взаємодіяти один з одним, отримуючи та надсилаючи запити. Це розширює початкову функціональність програми або веб-сайту.

Щоб краще зрозуміти, як це працює, найкраще розглянути підхід до підключення на основі API, розділивши API на три типи:

- Systems APIs – ваші основні системи, такі як ERP або CRM, як правило, важкодоступні, оскільки їхні інтерфейси підключення є приватними та складними. Системні API дозволяють обійти цю складність.

- Process APIs – ці API розбивають дані, взаємодіючі з ними та формуючи їх як в одній системі, так і в різних системах. Вони не залежать від вихідних систем, з яких дані походять, або каналів, куди дані мають бути доставлені.

- Experience APIs – тепер дані обмінюються між каналами, кожен з яких може отримати доступ до тих самих даних, але в різних формах. Інтерфейси API Experience дозволяють змінювати конфігурацію даних, щоб їх було легше використовувати аудиторії – усе із загального джерела даних замість того, щоб налаштовувати окремі інтеграції точка-точка для кожного каналу.

Ця стратегія на основі API уніфікує дані для ефективнішого здійснювання багатоканальної комерції, забезпечуючи таким чином клієнтам узгоджену взаємодію з покупками в усіх каналах. Це також дає змогу використовувати API, щоб узгоджено відповідати потребам, що змінюються, і керувати ними [3].

Актуальність API для електронної комерції визначається рядом факторів:

- **Інтеграція з іншими сервісами** – API дозволяють електронним магазинам легко інтегрувати свої системи з різноманітними сервісами, такими як платіжні шлюзи, системи доставки, аналітичні інструменти тощо.

- **Автоматизація процесів** – За допомогою API можна автоматизувати багато процесів в електронній комерції, таких як оновлення товарного асортименту, відстеження стану замовлень та взаємодія з клієнтами.

- **Покращення користувацького досвіду** – API дозволяють створювати персоналізовані та інноваційні функції, що підвищують зручність та привабливість електронного магазину для клієнтів.

- **Швидкість реагування на зміни** – API роблять електронну комерцію більш гнучкою, що дозволяє швидко адаптуватися до змін у вимогах ринку та технологічних новацій.

- **Безпека та надійність** – Використання API може покращити рівень безпеки, оскільки дозволяє забезпечити аутентифікацію та авторизацію взаємодії між різними системами.

Загалом, використання актуальних API у електронній комерції є важливим елементом стратегії розвитку, дозволяючи підприємствам бути більш конкурентоспроможними та ефективними на ринку.

1.2. Платформа Apigee X для управління API

Apigee X – це платформа для управління API, надана Google Cloud, яка надає інструменти для створення, розгортання та управління API. Додатково платформа включає в себе такі можливості як аналітика та заходи безпеки.

Зокрема, Apigee X дозволяє розробникам створювати API та легко налаштовувати їх параметри. Інструменти аналітики забезпечують можливість відстежувати трафік та оцінювати ефективність API. Окрім цього, платформа враховує аспекти безпеки, забезпечуючи аутентифікацію та авторизацію для захисту від несанкціонованого доступу.

Apigee X є масштабованою платформою, придатною для використання в різних розмірах та типах підприємств. Її можна легко інтегрувати з іншими сервісами Google Cloud, що робить її важливим елементом для розробників, які використовують електронну комерцію та інші веб-сервіси в своїх проектах.[4]

Apigee пропонує високопродуктивні API-проксі для створення послідовного, надійного інтерфейсу для служб на стороні сервера. Шар проксі дозволяє детально керувати безпекою, обмеженням швидкості, квотами, аналітикою та іншими параметрами для всіх служб.

Apigee підтримує REST, gRPC, SOAP та GraphQL, надаючи гнучкість для впровадження будь-якого архітектурного стилю API.

Apigee надає проксі-шар API, який знаходиться між ваших служб на стороні сервера та внутрішніми чи зовнішніми клієнтами, які хочуть використовувати інші служби. Apigee пропонує різноманітні політики, які дозволяють додавати безпеку, управління трафіком, медіацію даних, розширення та інші функції до проксі-шару API для стійкого та корпоративного управління API. [5]

Apigee призначено для користувачів двох основних типів та унікальних викликів управління API, з якими вони стикаються:

- **Виробники API** – будують та управляють API, які викривають служби на стороні сервера.
- **Споживачі API** – використовують дані, надані API, у своїх клієнтських додатках. [5]

Apigee архітектура:

Щоб забезпечити зв’язок між VPC, ми використовуємо мережевий піринг VPC. Мережевий піринг дозволяє з’єднувати внутрішні IP-адреси в двох мережах віртуальної приватної хмари (VPC) незалежно від того, належать вони до одного проекту чи однієї організації Google Cloud. Після завершення етапу однорангового зв’язку між двома VPC можливий зв’язок.

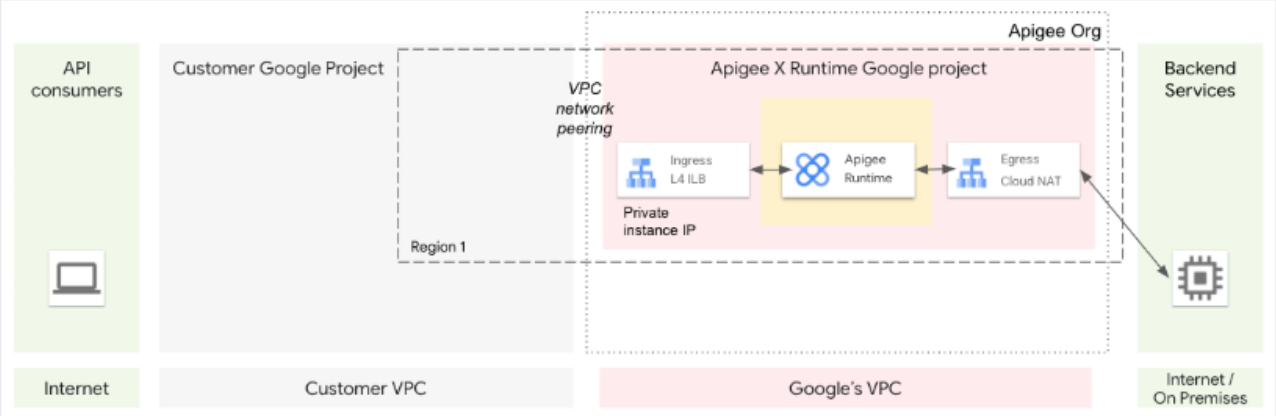

Рисунок 1.1 - Мережевий піринг VPC забезпечує зв’язок між VPC.

1.3. Огляд можливих альтернатив для оптимізації менеджменту та безпеки API

Розглянемо можливі альтернативи, вказуючи на їхні переваги та недоліки, щоб допомогти зробити обґрунтований вибір у контексті конкретних потреб та стратегій розвитку. Розглянемо деякі з важливих аспектів, пов'язаних із застосуванням різних підходів до оптимізації управління та безпеки API.

Таблиця 1.1 - Альтернативи для оптимізації менеджменту та безпеки API

Альтернативні підходи	Переваги	Недоліки
Apigee X Є повноцінною хмарною платформою для управління API від Google Cloud. Вона надає широкий спектр інструментів для створення, тестування, моніторингу та захисту API. Використання Apigee X дозволяє забезпечити високий рівень безпеки та ефективності управління API.	Забезпечення готового та комплексного рішення, висока масштабованість, інтеграція з екосистемою Google Cloud	Необхідність адаптації внутрішніх процесів та можливі труднощі при інтеграції з існуючими системами
Внутрішня розробка API Розробка власних рішень для управління API без залучення сторонніх платформ. Цей підхід може включати створення власних інструментів та реалізацію кастомних рішень	Повний контроль над розробкою та можливість адаптації під конкретні потреби компанії	Значна затрата часу та ресурсів на розробку, можливі обмеження функціональності в порівнянні з готовими рішеннями
Впровадження заходів безпеки окремо Використання окремих інструментів та сервісів для забезпечення безпеки API, не повністю переходячи на Apigee X. Це може включати використання спеціалізованих засобів для моніторингу, автентифікації, та захисту	Гнучкість у виборі інструментів безпеки, можливість вибирати лише ті рішення, які потрібні	Потребує інтеграції та координації різних інструментів, менш ефективніший у порівнянні з інтегрованими рішеннями

Проаналізуємо різні альтернатив управління API, з фокусом на показниках якості, що визначають їхню ефективність та придатність для конкретних потреб. Розглядаючи такі аспекти, як продуктивність, безпека та вартість реалізації, ми можемо здійснити обґрунтований вибір у підході до оптимізації управління та безпеки API, враховуючи вимоги та стратегічні цілі.

Таблиця 1.2 - Огляд показників якості для альтернатив управління API

	Apigee X	**Внутрішня розробка API**	**Впровадження заходів безпеки окремо**
Продуктивність	Висока продуктивність завдяки готовій хмарній платформі	Залежить від рівня оптимізації та якості розробки	Залежить від ефективності та сумісності обраного інструментарію
Безпека	Високий рівень захисту від атак, автентифікація та авторизація, керування доступом та захист даних.	Повний контроль над заходами безпеки, але потребує власної розробки та аудиту	Може забезпечити високий рівень безпеки, але вимагає інтеграції та координації
Вартість реалізації	Вартість може бути вищою, але забезпечує готові рішення та підтримку	Залежить від обсягу та складності розробки	Витрати залежать від обраного набору інструментів

На основі аналізу можливих альтернатив для оптимізації менеджменту та безпеки API, а також показників якості для альтернатив управління API, можна зробити наступні висновки щодо використання Apigee X:

- **Продуктивність та гнучкість** – Apigee X виступає як готова та високопродуктивна хмарна платформа, що надає широкий спектр функцій для управління API. Здатна підтримувати різні архітектурні стилі API (REST, gRPC, SOAP, GraphQL) робить Apigee X гнучким рішенням для різноманітних потреб.
- **Безпека** – висока безпека Apigee X може забезпечувати захист від загроз та вразливостей, що є важливим фактором при виборі серед альтернатив для управління API.
- **Вартість реалізації** – Apigee X може мати вищу вартість реалізації порівняно з іншими альтернативами, проте це компенсується готовими рішеннями та підтримкою, які надає Google Cloud.
- **Надійність** – Apigee X забезпечує стійкий інтерфейс для управління та споживання API, дозволяючи внесення змін до бекенду без перебоїв у роботі API.

На підставі цих висновків можна визначити Apigee X як потужну та безпечну платформу для управління API з високою продуктивністю та гнучкістю. Вибір між Apigee X.

1.4. Аналіз ризиків безпеки API

У світі швидко розвиваючих технологій, де взаємодія між різними системами та сервісами стає необхідністю, безпека API відіграє важливу роль у захисті від потенційних загроз та атак. При впровадженні та управлінні API, ризики безпеки стають важливою складовою стратегії та вимагають уважного аналізу.

Інтерфейси програмного інтерфейсу (API) створено спеціально для обміну найціннішими даними та послугами компанії. Це робить їх прибутковою мішенню

Зловмисники націлені на вразливості бізнес-логіки у API. Але оскільки API унікальні, їм потрібні дні, тижні чи навіть місяці, щоб перевірити та вивчити API. Вони використовують «низькі та повільні» методи, які WAF, шлюзи та інші традиційні інструменти не можуть виявити, що робить вас уразливими.[6]

У минулому організації вважали, що належна автентифікація для взаємодії з API була достатнім стримуючим фактором, щоб направити зловмисників в інше місце. Дані Salt Labs показують, що 78% атак походять від, здавалося б, законних користувачів, які зловмисно досягли належної автентифікації.

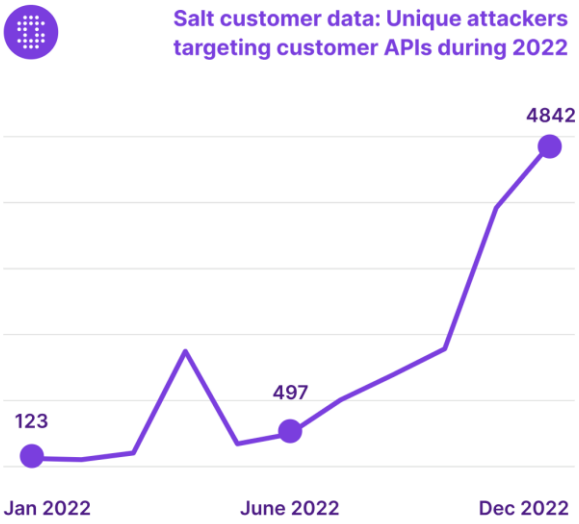

Рисунок 1.2 - Кількість унікальних атак API за 2022 рік

Проаналізуємо основні ризики безпеки API в контексті різних підходів до управління API. Розглянемо потенційні загрози та вразливості, що можуть виникнути при використанні стратегій управління API. [7]

Таблиця 1.3 - Ризики безпеки OWASP API

Ризик	**Опис**
Порушена автентифікація	Механізми автентифікації часто реалізуються неправильно, що дозволяє зловмисникам скомпрометувати токени автентифікації або використовувати недоліки реалізації, щоб тимчасово або назавжди присвоїти ідентифікаційні дані інших користувачів. Порушення здатності системи ідентифікувати клієнта/користувача порушує загальну безпеку API. API вразливе, якщо: 1) Дозволяє надсилати облікові дані, коли зловмисник використовує грубу силу зі списком дійсних імен користувачів і паролів. 2) Дозволяє зловмисникам виконувати атаку грубою силою на той самий обліковий запис користувача, не використовуючи механізм перевірки коду/блокування облікового запису. 3) Дозволяє слабкі паролі. 4) Надсилає конфіденційні дані автентифікації, такі як маркери автентифікації та паролі в URL-адресі. 5) Дозволяє користувачам змінювати адресу електронної пошти, поточний пароль або виконувати будь-які інші конфіденційні операції, не запитуючи підтвердження пароля. 6) Не перевіряє автентичність токенів. 7) Приймає непідписані/слабо підписані маркери JWT (

	{"alg":"none"}) 8) Не перевіряє термін дії JWT. 9) Використовує простий текст, незашифровані або слабко хешовані паролі. 10) Використовує слабкі ключі шифрування. 11) Інші мікросервіси можуть отримати до нього доступ без автентифікації 12) Використовує слабкі або передбачувані маркери для забезпечення автентифікації
Неправильна конфігурація безпеки	API та системи, які їх підтримують, зазвичай містять складні конфігурації, призначені для того, щоб зробити API більш настроюваними. Інженери програмного забезпечення та DevOps можуть пропустити ці конфігурації або не дотримуватися найкращих практик безпеки, коли йдеться про конфігурацію, відкриваючи двері для різних типів атак. API вразливе, якщо: 1) Відповідне посилення безпеки відсутнє в будь-якій частині стеку API або якщо є неправильно налаштовані дозволи для хмарних служб 2) Немає останніх патчів безпеки або системи застаріли 3) Увімкнена непотрібні функції (наприклад, дієслова HTTP, функції журналювання) 4) Існують розбіжності в тому, як сервери в ланцюжку серверів HTTP обробляють вхідні запити 5) Безпека транспортного рівня (TLS) відсутня 6) Директиви безпеки чи керування кеш-пам’яттю не надсилаються клієнтам 7) Політика спільного використання ресурсів між джерелами (CORS) відсутня або налаштована неправильно 8) Повідомлення про помилки містять трасування стека або розкривають іншу конфіденційну інформацію
Неналежне управління документацією	API, як правило, відкривають більше кінцевих точок, ніж традиційні веб-додатки, що робить правильну й оновлену документацію надзвичайно важливою. Правильна інвентаризація хостів і розгорнутих версій API також важлива для пом’якшення таких проблем, як застарілі версії API та відкриті кінцеві точки налагодження. API вразливе, якщо: 1) Призначення хосту API незрозуміле, і немає чітких відповідей на наступні запитання 2) У якому середовищі працює API (наприклад, виробництво, постановка, тестування, розробка)? 3) Хто повинен мати мережевий доступ до API (наприклад, публічний, внутрішній, партнери)? 4) Яка версія API працює? 5) Документація відсутня або існуюча документація не оновлена. 6) Для кожної версії API немає плану виходу на пенсію. 7) Інвентар хоста відсутній або застарілий.
Небезпечне використання API	Розробники, як правило, більше довіряють даним, отриманим від сторонніх API, ніж введеним користувачами, тому, як правило, застосовують слабкіші стандарти безпеки.

	Щоб скомпрометувати API, зловмисники шукають інтегровані сторонні служби замість того, щоб намагатися скомпрометувати цільовий API безпосередньо. API вразливе, якщо: 1) Взаємодіє з іншими API через незашифрований канал; 2) Не перевіряє належним чином і не очищає дані, зібрані з інших API, перед їх обробкою або передачею до наступних компонентів; 3) Сліпо дотримується перенаправлень; 4) Не обмежує кількість ресурсів, доступних для обробки відповідей сторонніх служб; 5) Не реалізує тайм-аути для взаємодії зі сторонніми службами;

1.5. Постановка задачі

Метою роботи є:

1. Розробка та впровадження інформаційної технології управління безпекою API.
2. Протоколювання та аудит для виявлення та аналізу спроб несанкціонованого доступу або інших безпекових інцидентів для платіжної системи з використанням платформи Apigee X.
3. Створення девелоперського порталу.

2. АЛГОРИТМИ ТА ЗАХОДИ БЕЗПЕКИ ДЛЯ API

2.1. Управління трафіком

Часто потрібно обмежувати швидкість надходження запитів до API, особливо для публічних API. API можуть бути перевантажені, якщо дозволити трафіку перевищувати місткість бекенд-сервісу. Обмеження швидкості для API може зменшити трафік до розумного рівня і дозволити успішно обслуговувати запити. Це допоможе забезпечити підтримку продуктивності додатків, не дозволяючи певним додаткам або користувачам споживати більшість пропускної здатності. Також важливо переконатися, що додатки не споживають більше ресурсів, ніж дозволено. Основна ідея полягає в обмеженні швидкості, з якою дозволяється надсилати запити до API.

Apigee надає дві політики, які дозволяють оптимізувати управління трафіком для мінімізації затримок для додатків, забезпечуючи при цьому стабільність бекенд-сервісів. Кожен тип політики вирішує конкретний аспект управління трафіком.

SpikeArrest:

Політика SpikeArrest захищає від перепадів трафіку. Ця політика обмежує кількість запитів, оброблених API-проксі та відправлених на бекенд, захищаючи від витрат продуктивності та періодів простою.

Цю політику слід використовувати для запобігання раптових вибухів трафіку, спричинених зловмисниками, які намагаються завадити обслуговуванню за допомогою атак типу " denial-of-service" (DOS), або через неправильні додатки клієнтів.

Quota:

Політика накладає обмеження на споживання ресурсів клієнтськими додатками, підтримуючи розподілений «лічильник», який обліковує вхідні запити. Лічильник може враховувати виклики API для будь-якого ідентифікованого суб’єкта, включаючи додатки, розробників, API-ключі,

токени доступу і т. д. Зазвичай для ідентифікації клієнтських додатків використовуються ключі API. Ця політика є обчислювальною витратною, тому для високотрафікованих API її слід налаштовувати на довші інтервали часу, такі як день чи місяць.

2.2. Ідентичність, аутентифікація та авторизація

Коли API обробляють користувацькі дані, надзвичайно важливо захищати ці дані, автентифікуючи та авторизуючи користувачів додатків.

Apigee надає політики для допомоги в автентифікації та авторизації. Політики Basic Authentication, OAuth 2.0, JSON Web Token можуть бути використані для забезпечення автентифікації та авторизації API. Apigee-проксі можуть бути використані для інтеграції з іншими постачальниками безпеки або інтеграції з власними службами токенів.

Розглянемо поняття ідентичності, аутентифікації та авторизації в контексті API.

Ідентичність додатка або користувача для конкретного запиту. Деяка частина запиту використовується для ідентифікації того, хто викликає або який додаток викликає. Додатки зазвичай ідентифікуються за допомогою API-ключів. API-ключі передаються разом з кожним запитом. Це надає доступ до захищених API і також дозволяє відстежувати використання API за допомогою додатка. Користувачі можуть бути ідентифіковані за допомогою інформації, такої як ім'я користувача чи номер облікового запису.

Аутентифікація – це процес перевірки облікових даних для підтвердження ідентичності. Для додатків обліковими даними є ключ та секрет додатка. Облікові дані користувача часто включають ім'я користувача та пароль. Також в якості облікових даних може використовуватися клієнтський сертифікат.

Авторизація – це визначення, що користувач або додаток має право робити. Авторизація має сенс тільки у випадку, якщо користувач чи додаток

був автентифікований. Токени часто представляються як доказ авторизації. Стандарти авторизації API включають OAuth 2.0 та SAML 2.0.

OAuth – це фреймворк авторизації, який дозволяє користувачам або клієнтам надавати доступ до ресурсів сервера іншій сутності, не передаючи при цьому облікові дані. Це ключова особливість OAuth: ви можете давати доступ до ресурсів, не передаючи облікові дані користувача чи клієнта, які, як правило, використовуються для доступу до ресурсів на сервері ресурсів. На даний момент існують дві версії OAuth: 1.0 та 2.0. Версія 1.0 є застарілою. Версія 2.0 - це поточний стандарт.

OAuth-токени доступу видаються для надання обмеженого доступу до конкретних ресурсів протягом визначеного періоду часу і можуть бути анульовані користувачем, який надав дозвіл, або сервером, який видав токен. Обмежений доступ до конкретних ресурсів означає, що токен не повинен надавати повний доступ до ресурсів користувача. Токени доступу є дійсними протягом обмеженого часу. Обмеження часу для токена доступу обмежує його вразливість у випадку його компрометації. Токени доступу також можна анулювати, зробивши їх недійсними. Анульований токен не має доступу. Ануляцію може ініціювати як користувач, який спочатку надав дозвіл на доступ до ресурсів користувача, так і сервер.

Access token захищають ресурси протягом обмеженого періоду часу. OAuth-токени доступу є непрозорими рядками. В токені немає кодованої або зашифрованої інформації: токен доступу - це просто випадковий набір символів. Це означає, що всю цікаву інформацію про токен зберігається на сервері авторизації.

Refresh token використовується для повторної авторизації наданого доступу, дозволяючи додатку отримати токен доступу без облікових даних користувача. Refresh token використовуватиметься, коли попередній токен доступу закінчиться та буде недійсним. Так само, як access token, токен оновлення є непрозорим рядком.

2.3. Типи дозволів OAuth

Типи дозволів OAuth визначають різні сценарії авторизації, які OAuth призначено обробляти.

Client Credentials Grant Type:

Клієнт може запросити токен доступу, використовуючи лише свої облікові дані клієнта (або інші підтримувані засоби аутентифікації), коли клієнт запитує доступ до захищених ресурсів, які перебувають під його контролем, або тих, що належать іншому власнику ресурсів із попередньо узгодженими умовами з сервером авторизації (метод, який виходить за рамки цієї специфікації).

Тип дозволу на облікові дані клієнта може бути використаний тільки конфіденційними клієнтами.[8]

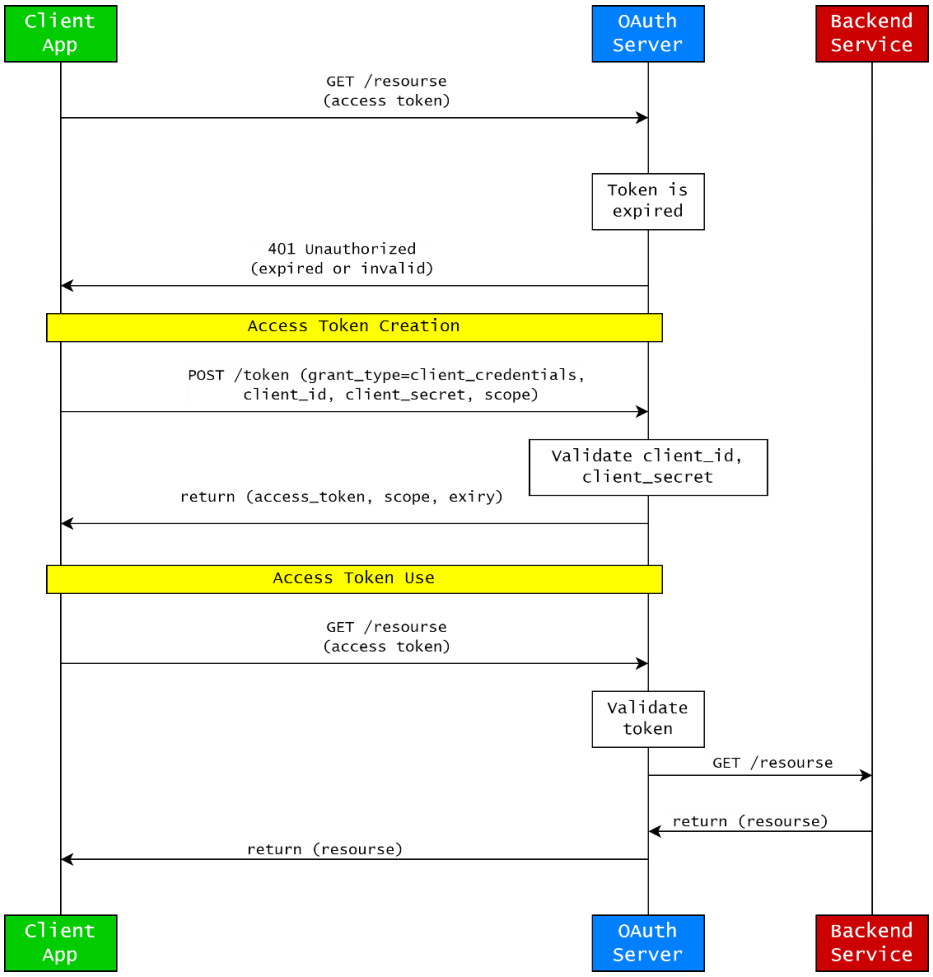

Рисунок 2.1 - Client Credentials Grant Type Flow

Потік, включає такі кроки:

- Клієнт автентифікується на сервері авторизації та запитує токен доступу з ендпоінту токенів.
- Сервер авторизації аутентифікує клієнта і, якщо він є валідним, виділяє токен доступу.

Resource Owner Password Grant Type:

Тип дозволу на використання облікових даних власника ресурсів підходить у випадках, коли власник ресурсів має довіру до клієнта, такого як операційна система пристрою або додаток з високими привілеями. Сервер авторизації повинен виявити особливу обережність при увімкненні цього типу дозволу та дозволити його лише тоді, коли інші потоки не є можливими.

Цей тип дозволу підходить для клієнтів, які можуть отримати облікові дані власника ресурсів (ім'я користувача та пароль, зазвичай за допомогою інтерактивної форми). Також його використовують для міграції існуючих клієнтів, які використовують пряму аутентифікацію за допомогою таких схем, як HTTP Basic Authentication до OAuth, перетворюючи збережені облікові дані в доступний токен.[8]

Потік, включає такі кроки:

- Власник ресурсу надає клієнту своє ім'я користувача та пароль.
- Клієнт запитує токен доступу з ендпоінту токенів сервера авторизації, включаючи отримані від власника ресурсу облікові дані. Під час здійснення запиту клієнт автентифікується на сервері авторизації.
- Сервер авторизації аутентифікує клієнта та перевіряє облікові дані власника ресурсу, і якщо вони валідні, виділяє токен доступу.

Password Client та Resource Owner Password Grant Types в Apigee X

Password Grant може бути використаний при доступі до облікових даних користувача. Він відповідний лише для довіреного додатка, який надається компанією, яка вже має доступ до даних.

Перший користувач для Password Grant - це конфіденційний і довірений додаток, який використовує API. Клієнт повинен бути конфіденційним, щоб він

міг захищати секрет клієнта, і йому повинно бути довірено, оскільки пароль користувача буде передано через додаток. Проксі Apigee генерує токени і відхиляє запити до backend, якщо наданий токен не є дійсним. Наступний в процесі взаємодії залучений користувач додатку, який надає дозвіл на доступ до своїх захищених ресурсів. Іншим учасником є сервер авторизації, який відповідає за перевірку облікових даних користувача та повертає деталі користувача. Останнім учасником є знову backend-сервіс, який обслуговує захищені ресурси.

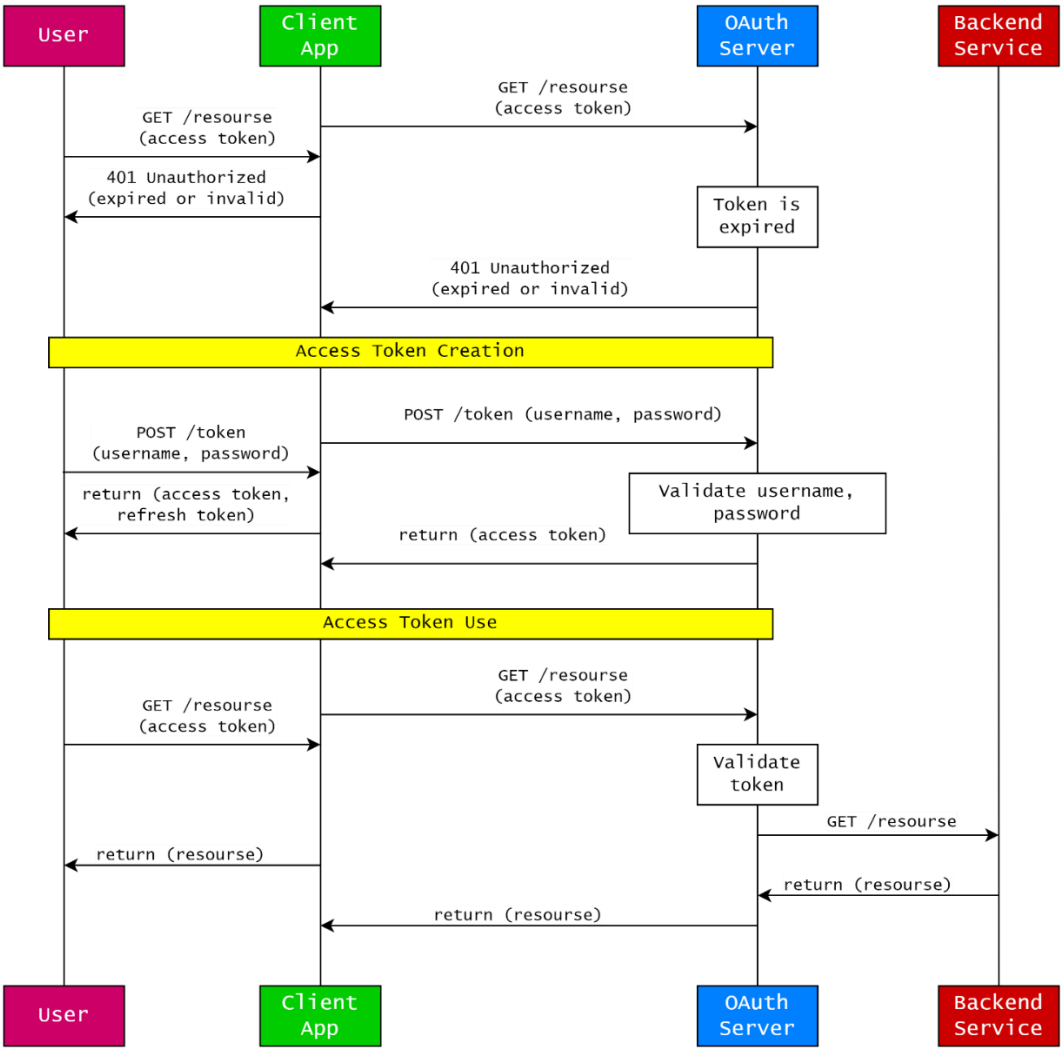

Рисунок 2.2 - Resource Owner Password Grant Type Flow

Додаток повинен бути довіреним і конфіденційним. Користувач повинен довіряти додатку, оскільки ім'я користувача та пароль будуть надані додатку.

Тепер взаємодіє сервер авторизації. Сервер авторизації зберігає імена користувачів та паролі для користувачів.

У POST-запиті до проксі Apigee OAuth, крім типу гранту, ідентифікатора та секрет клієнта, ми також передаємо ім'я користувача та пароль користувача, який надає доступ до захищених ресурсів (див. Рисунок 2.3 -).

Після того як проксі підтвердить ідентифікатор та секрет клієнта, необхідно аутентифікувати дані користувача. Apigee викликає точку авторизації сервера авторизації з ім'ям користувача та його паролем. Сервер авторизації перевіряє ім'я користувача та пароль і повертає деталі користувача, якщо дані користувача є вірними. Якщо дані користувача не є вірними, сервер авторизації повертає помилку, і проксі Apigee OAuth повертає помилку назад до додатку.

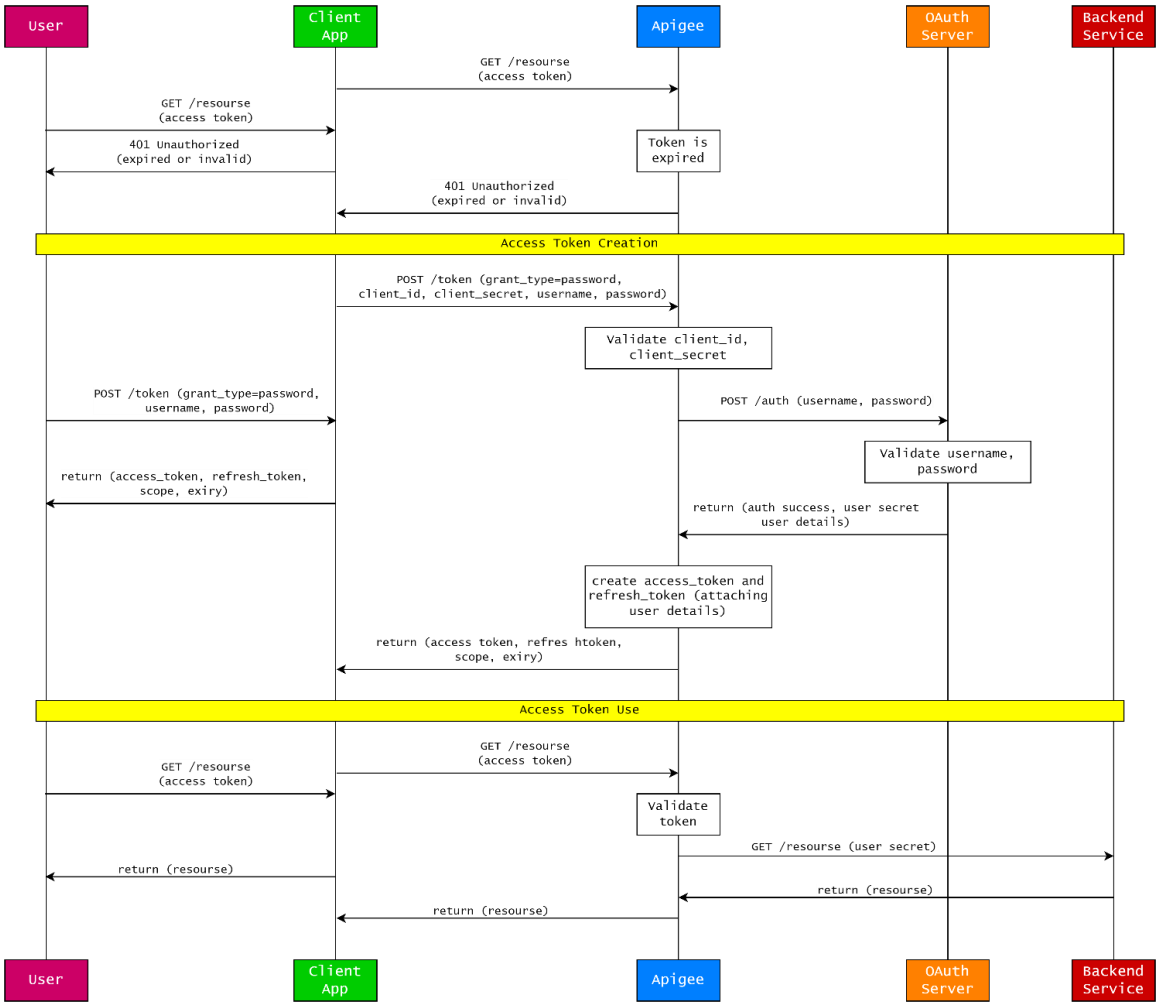

Рисунок 2.3 - Схема реалізація Password Client та Credentials Grant Types в Apigee X

Якщо аутентифікація користувача пройшла успішно, проксі OAuth створює токен доступу та токен оновлення для додатку, який надає доступ до захищених ресурсів (див. Рисунок 2.4 -). Зазвичай проксі також додає власні атрибути до токенів доступу та оновлення. Ці атрибути містять будь-які деталі користувача, які повертаються від сервера авторизації і які буде потрібно бекенд-сервісу. Ці власні атрибути доступні при перевірці токену. [9–14]

Отримана інформація така ж, як і для типу отримання доступу для клієнтів, за винятком того, що відповідь містить токен оновлення.

Токен використовується програмою так само, як і для типу отримання доступу для клієнтів. Під час перевірки токену проксі Apigee має доступ до деталей користувача, які були додані як користувацькі атрибути. Деталі користувача можуть бути відправлені на бекенд-сервіс, якщо це потрібно.

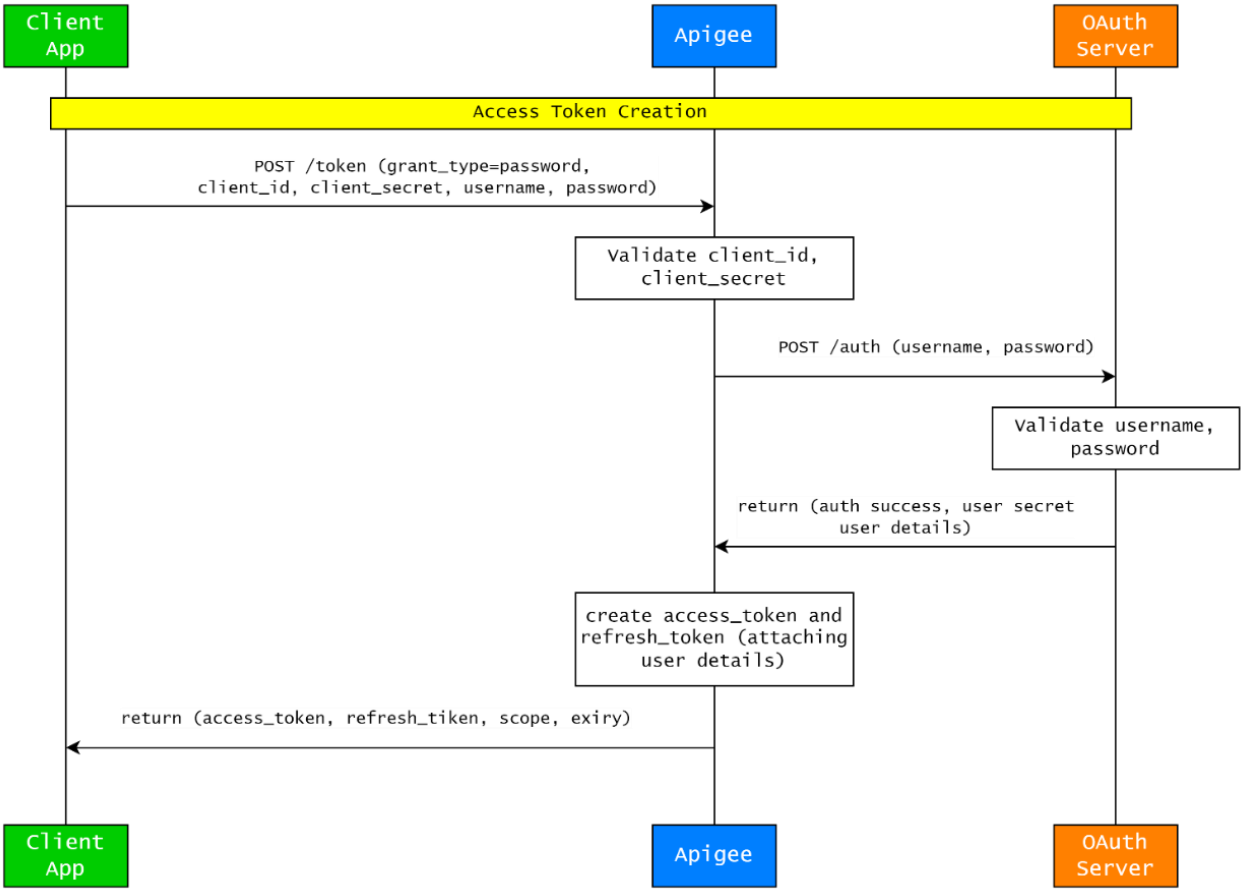

Рисунок 2.4 - Схема створення токену

2.4. Протоколювання та аудит

Для логування використовується політик та додається до проксі-сервісу, який обробляє запити і відповіді між клієнтом та сервером.

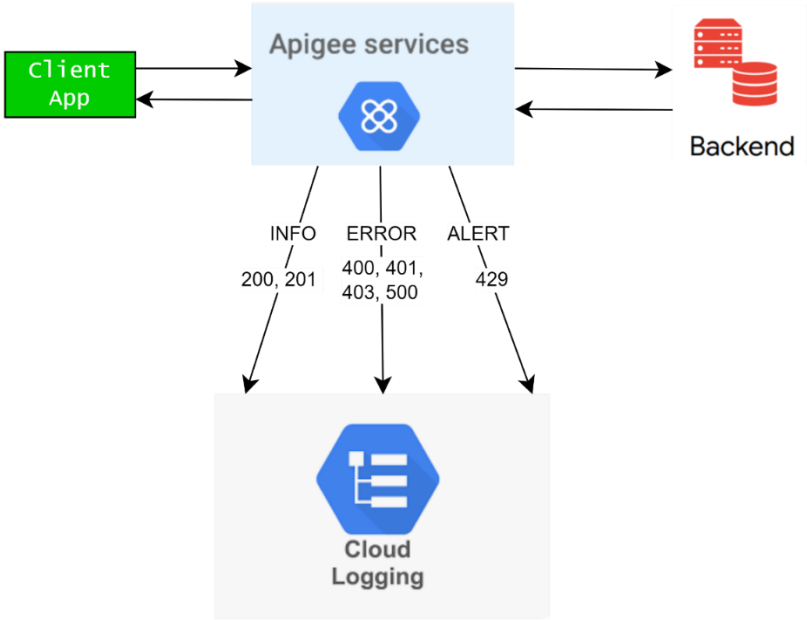

Рисунок 2.5 - Схема логування

Налаштування якої включає наступні кроки:

- налаштування щодо того, які типи повідомлень будуть логуватися, які поля будуть включені в кожний запис журналу, та куди буде зберігатися журнал.
- запит або відповідь проходить через проксі, політика Message Logging записує інформацію про цей обмін у журнал. Це може включати заголовки, тіло запиту або відповіді, інформацію про час та інші метадані.
- залоговані дані можна аналізувати для виявлення патернів, аномалій, або для відслідковування діагностичної інформації. Це може бути корисно для виявлення проблем або для вдосконалення продуктивності системи.
- управління журналом включає в себе збереження, пошук, фільтрацію та видалення записів журналу згідно з вимогами безпеки та збереження даних.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Apigee API Proxy розроблений для реалізації інформаційної технології для розробки прикладних програмних інтерфейсів з використанням моделі Client Credentials та Resource Owner Password Grant Types в Apigee X для підвищених вимог до інформаційної безпеки.

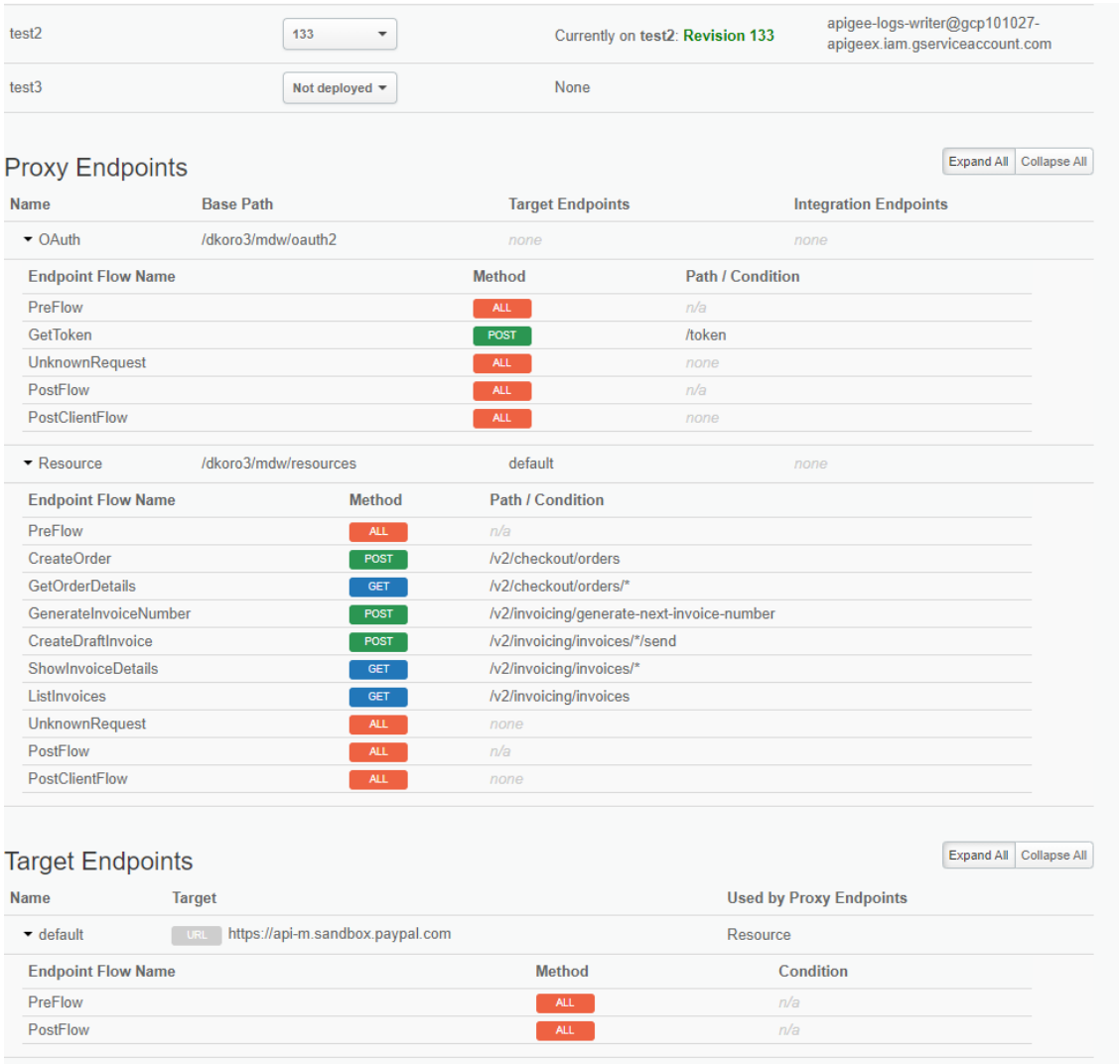

Рисунок 3.1 - Загальний вигляд Overview Apigee Proxy

3.2. Реалізація OAuth сервісу

Основні елементи конфігурацій OAuth проксі-сервера (див. додаток) включають.

ProxyEndpoint - визначає точку доступу до проксі-сервера, яка обробляє HTTP запити та відповіді.

BasePath - базовий шлях, який визначає URL, за яким проксі-сервер буде доступний.

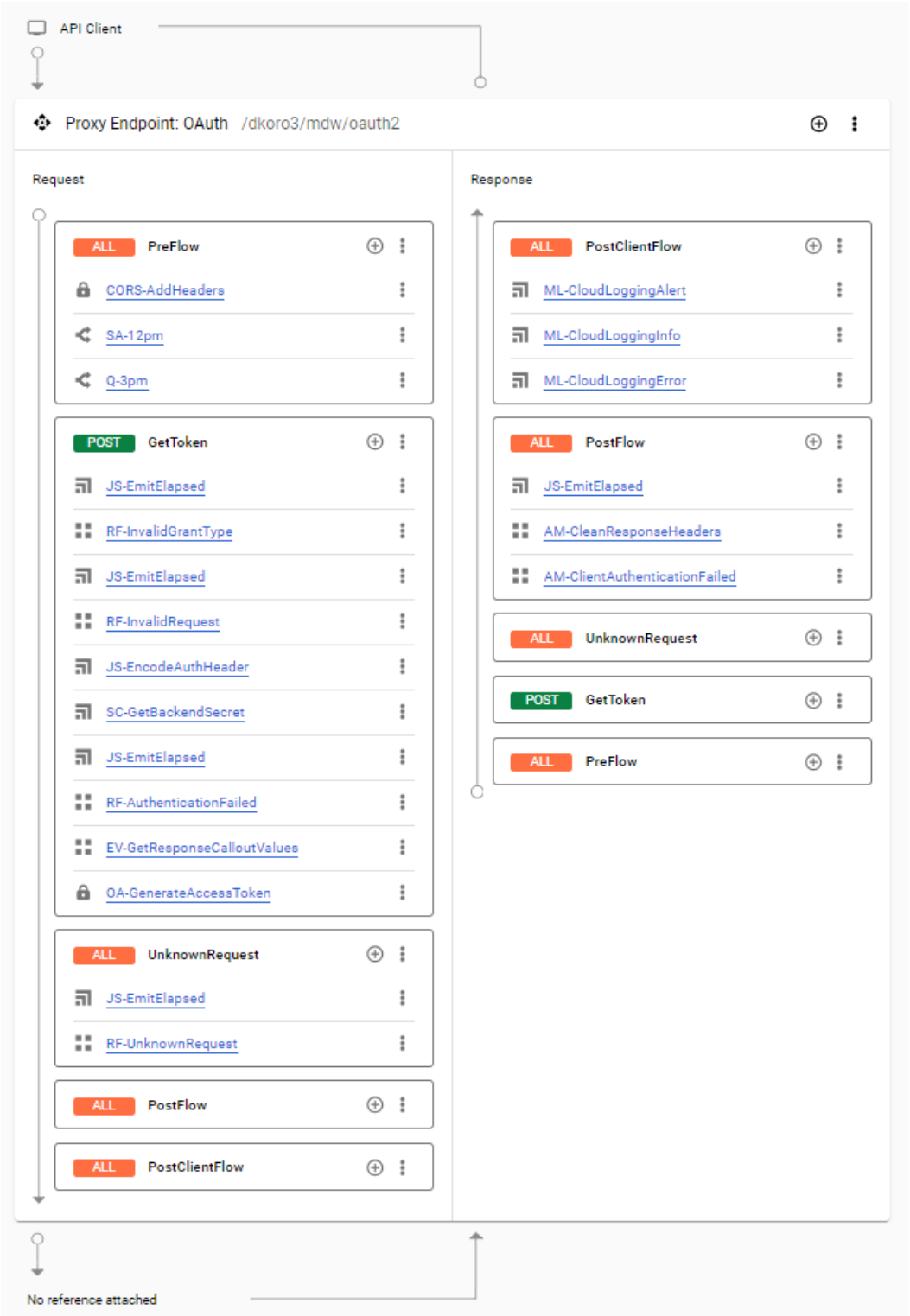

Рисунок 3.2 - Схема реалізації OAuth сервісу

FaultRules - визначає правила для обробки помилок, які можуть виникати під час обробки запитів. Ця фаза включає два FaultRules:

1) SpikeArrest-OAuth - правило викликається, якщо відбувається перевищення обмеження на кількість запитів. У цьому випадку, викликається крок "AM-SpikeArrestErrorMessage", який відповідає за відображення повідомлення про помилку.

2)

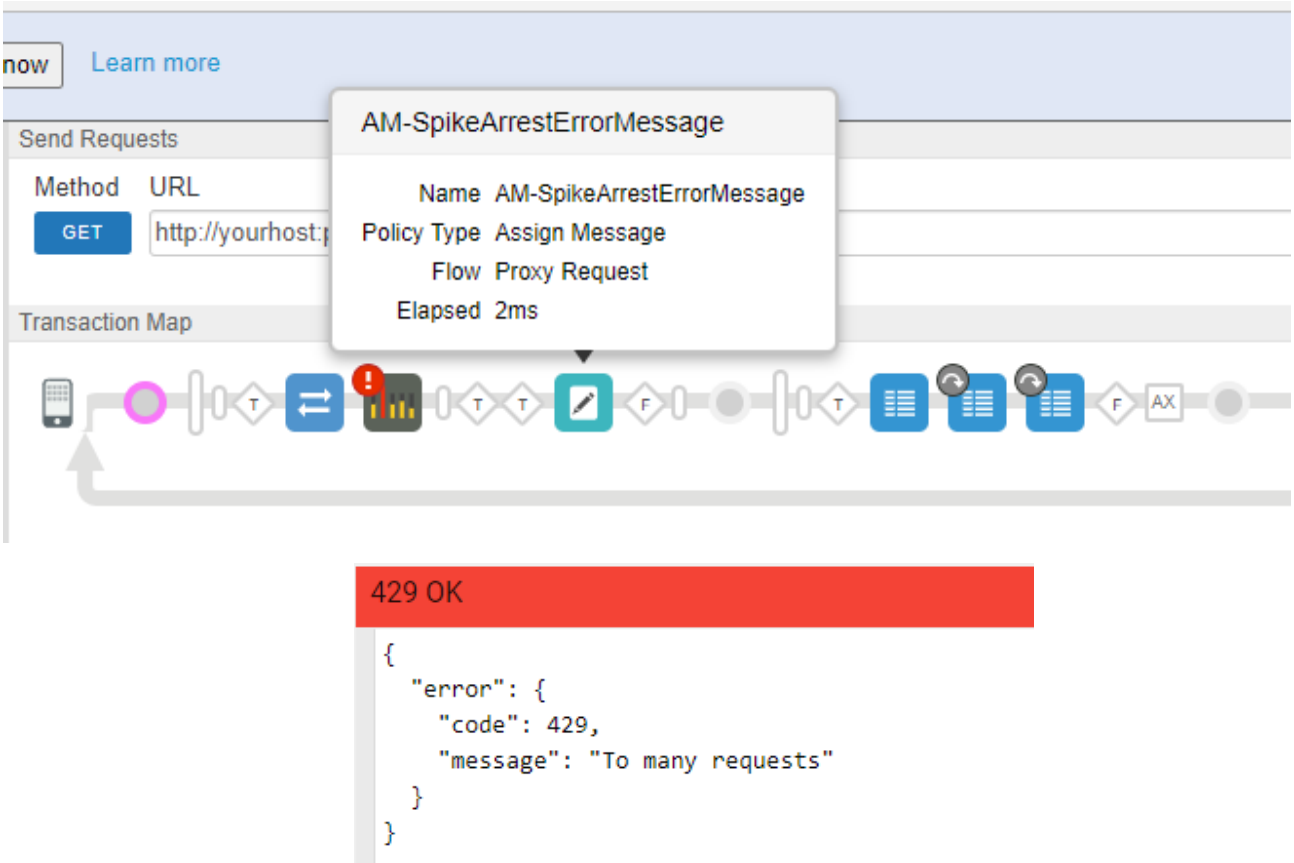

Рисунок 3.3 - Обробка помилки при різкому сплеску трафіку

3) Quota-OAuth - правило викликається, якщо відбувається перевищення обмеження на кількість запитів. У цьому випадку, викликається крок "AM-QuotaErrorMessage", який відповідає за відображення повідомлення про помилку.

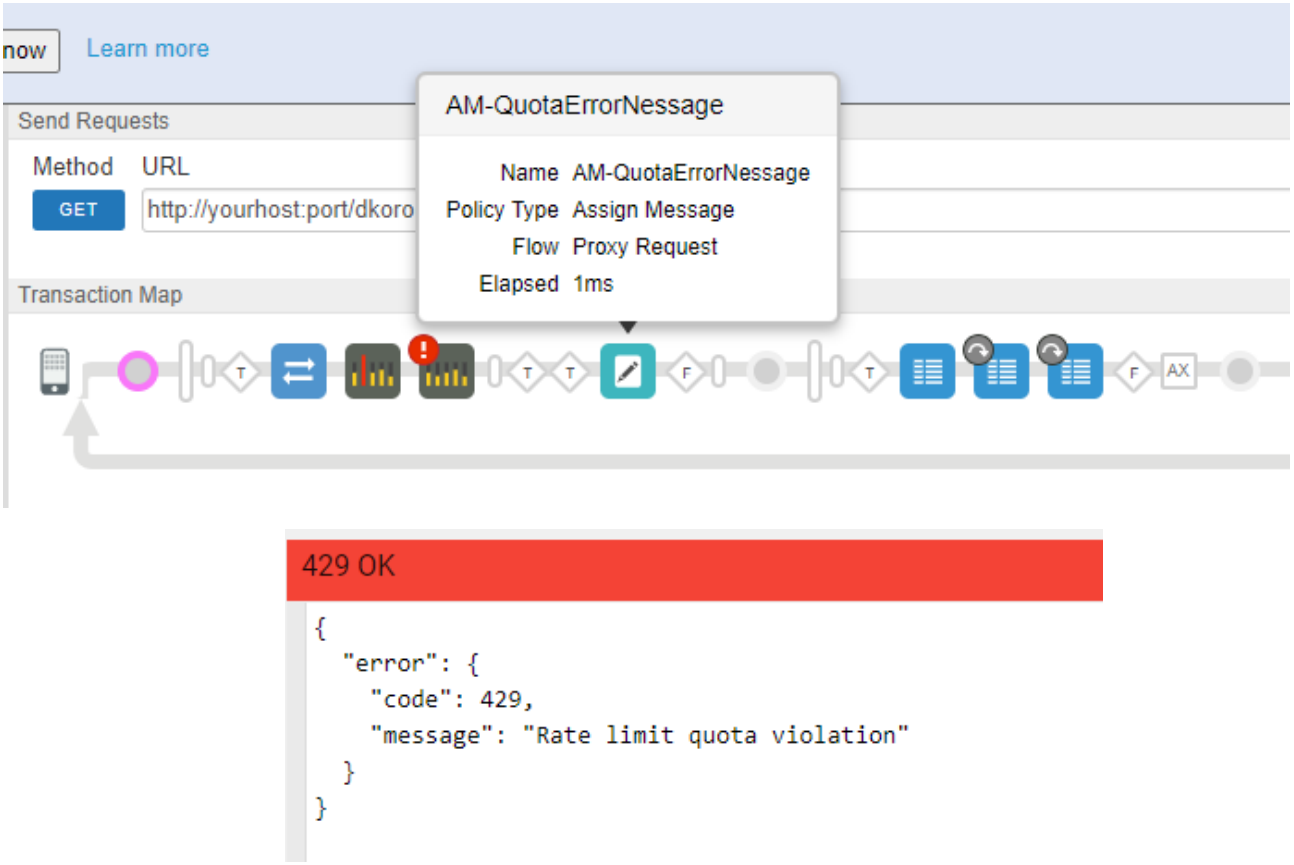

Рисунок 3.4 - Обробка помилки при перевищенню квоти трафіку

Ці правила дозволяють проксі-серверу ефективно обробляти помилки, що виникають під час обробки запитів, та надають можливість відображати користувачам інформацію про помилки, які виникають.

PreFlow - це фаза обробки, яка виконується перед тим, як запит буде переданий до цільового сервісу. У цьому конфігураційному файлі PreFlow має три кроки:

1) CORS-AddHeaders - крок додає заголовки, необхідні для виконання Cross-Origin Resource Sharing (CORS) в HTTP-відповідях. Це дозволяє запитувачам з одного джерела отримувати доступ до ресурсів з іншого джерела.

2) SA-12pm - крок виконується для обмеження кількості запитів, дозволяє надсилати максимум 12 запитів на хвилину зі швидкістю один запит на 5 секундний інтервал. Якщо протягом 5 секундного інтервалу надходить більше одного запиту, такі запити блокуються. Запити допускаються, якщо кількість запитів менше налаштованого обмеження швидкості 12 за хвилину.

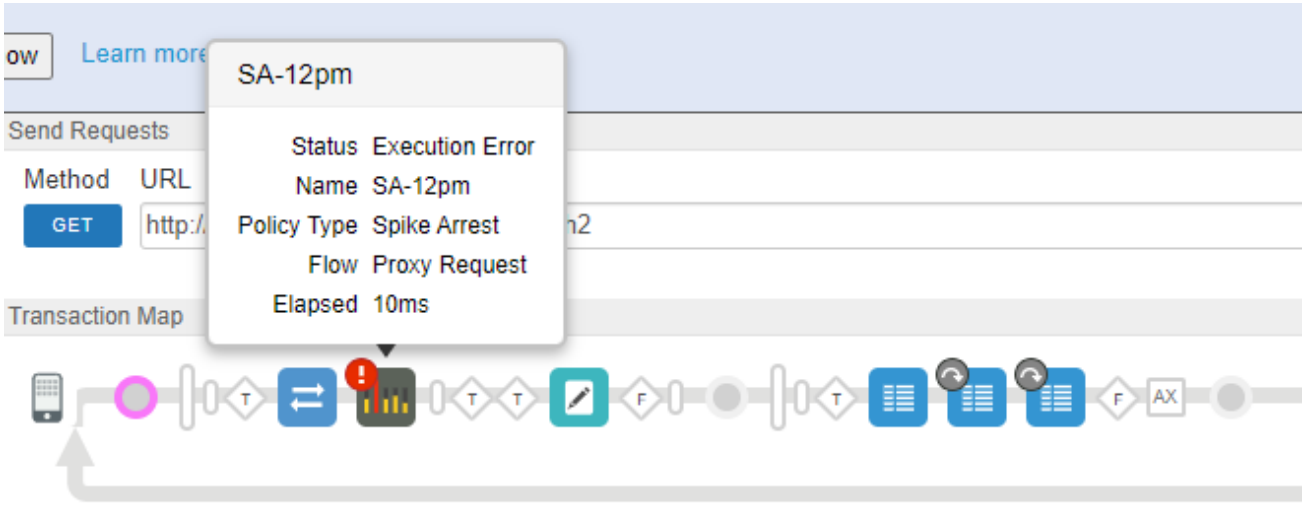

Рисунок 3.5 - Спрацювання захисту API при різкому сплеску трафіку

3) Q-3pm - крок виконується для обмеження кількості запитів, які можуть бути оброблені в певний проміжок часу. У цьому випадку, "3pm" вказує на обмеження до 3 запитів на хвилину.

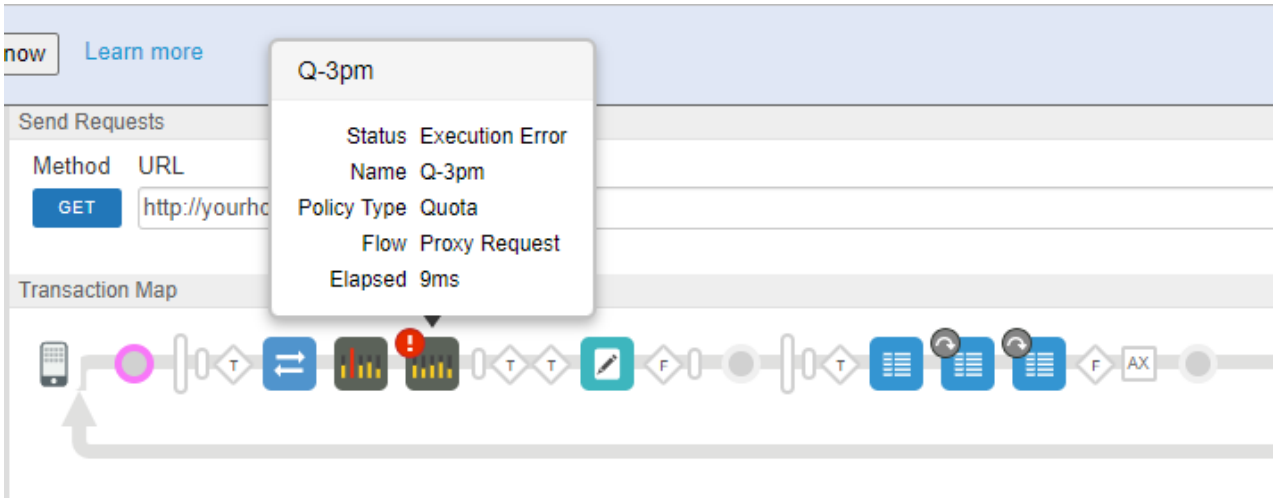

Рисунок 3.6 - Спрацювання захисту API при перевищені квоти трафіку

Ці кроки дозволяють проксі-серверу контролювати кількість запитів, які можуть бути оброблені в певний проміжок часу, та забезпечити безпечну обробку запитів.

PostFlow - це фаза обробки, яка виконується після того, як відповідь від цільового сервісу повернулася до проксі-сервера. У цьому конфігураційному файлі PostFlow має два кроки:

1) AM-CleanResponseHeaders - крок виконується для очищення заголовків відповіді від службової інформації, яка може бути небезпечною для безпеки.

2) AM-ClientAuthenticationFailed - крок виконується, якщо аутентифікація клієнта не вдалася. Відправляє спеціальну відповідь, щоб повідомити клієнта про помилку аутентифікації.

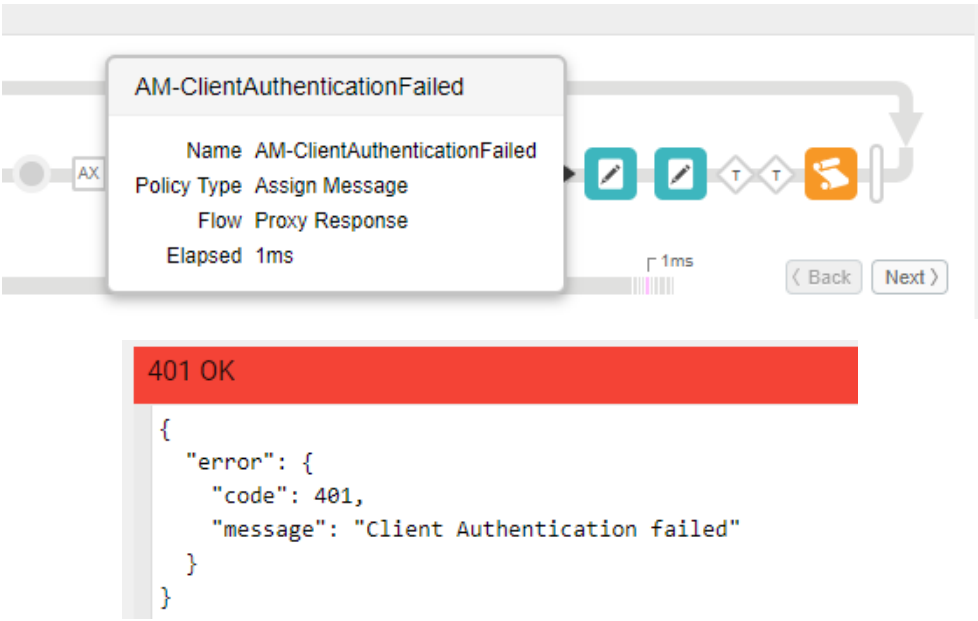

Рисунок 3.7 - Обробка помилки аутентифікації клієнта

Flows - це набір кроків, які виконуються для обробки конкретного запиту. Проект має два різних потоки:

- **GetToken** - потік призначений для видачі токенів OAuth2.0, зазвичай для запитів типу grant_type=password. Він має наступні кроки:

1) RF-InvalidGrantType - перевіряє, чи відповідає тип надання токена вимогам.

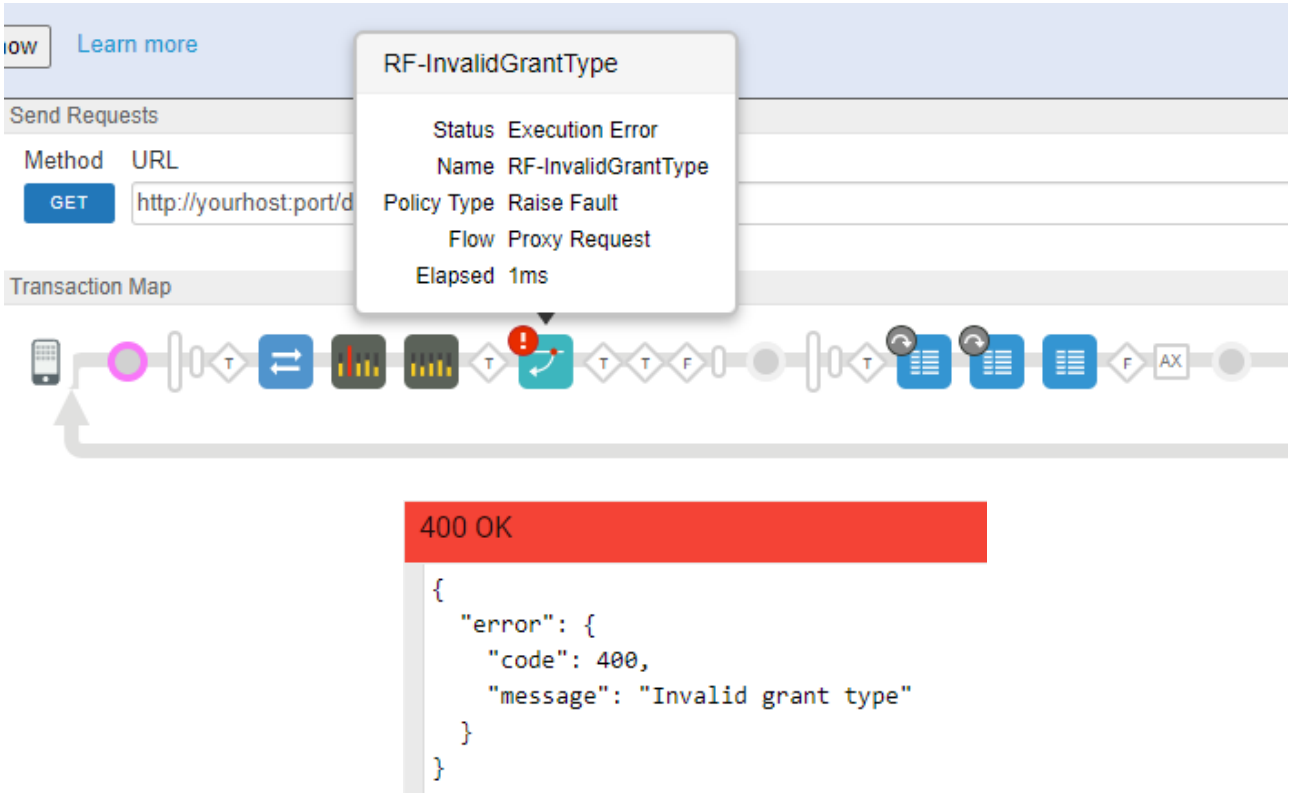

Рисунок 3.8 - Обробка помилки Invalid Grant Type

2) RF-InvalidRequest - перевіряє, чи заповнені всі обов'язкові поля запиту.

Рисунок 3.9 - Обробка помилки Invalid Request

3) JS-EncodeAuthHeader - кодує заголовок авторизації у base64 для використання в запиті отримання секретного ключа для доступу до бекенд-сервісу.

Рисунок 3.10 - Крок кодування заголовку авторизації

4) SC-GetBackendSecret - отримує секретний ключ для доступу до бекенд-сервісу.

5) RF-AuthenticationFailed - перевіряє, чи була успішна аутентифікація клієнта.

Рисунок 3.11 - Обробка помилки аутентифікації користувача

6) EV-GetCalloutValues - отримує значення з виклику бекенд-сервісу.

7) OA-GenerateAccessToken - генерує та повертає токен доступу.

Рисунок 3.12 - Отримання токена доступу

- **UnknownRequest** - потік виконується, якщо запит не відповідає жодному із визначених потоків. Він має один крок:

1) RF-UnknownRequest - повертає відповідь про помилку, оскільки запит не розпізнаний.

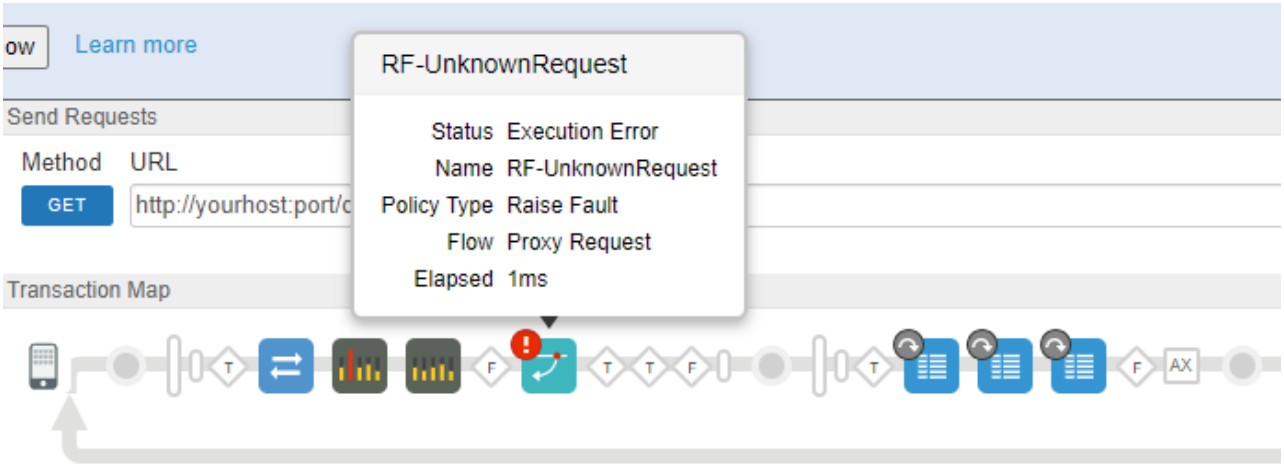

Рисунок 3.13 - Обробка помилки Unknown Request

Ці потоки виконуються в залежності від умови, яка визначена в кожному потоці.

PostClientFlow - потік виконується після того, як відправлено відповідь клієнту. Він має три кроки:

1) ML-CloudLoggingAlert – виконує логування, якщо відповідь має код помилки 429 (заборонено).

2) ML-CloudLoggingInfo – виконує логування, якщо відповідь має код успіху 200.

3) ML-CloudLoggingError – виконує логування, якщо відповідь має код помилки 500, 400 або 401, 403, 404.

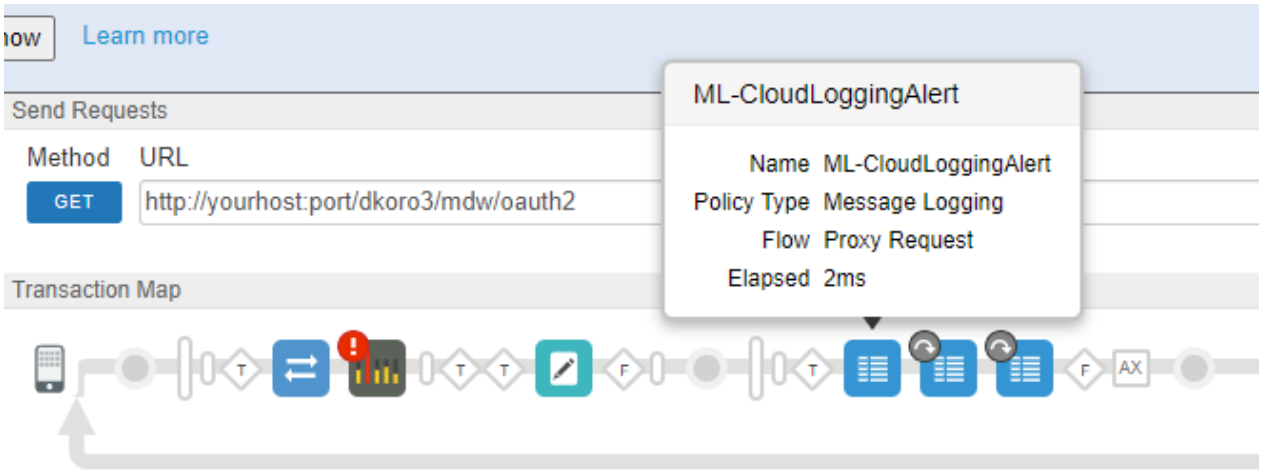

Рисунок 3.14 - Логування Alert у Cloud Logging

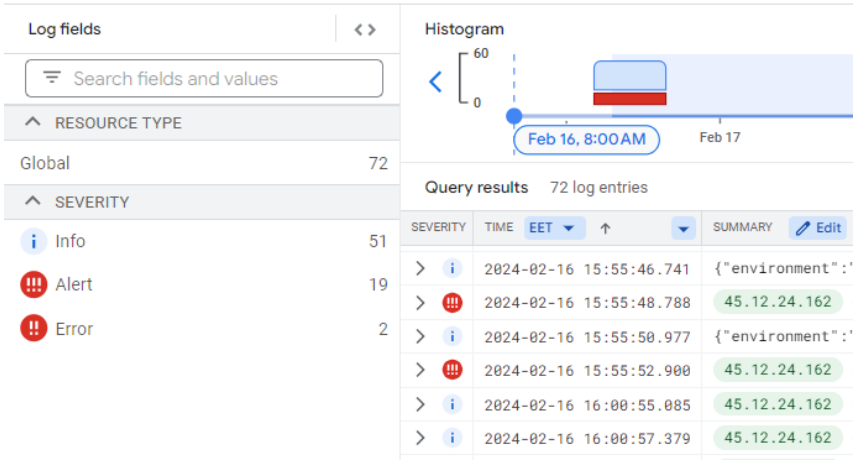

Рисунок 3.15 - Реєстрація Alert, Errors та Info повідомлень у Cloud Logging

Ці кроки виконуються в залежності від умови, яка визначена в кожному кроці.

RouteRule - це правило маршрутизації, яке визначає, які запити будуть оброблені цим проксі-сервером.

3.3. Реалізація Recourses сервісу

Цей сервіс використовується для обробки запитів до ресурсів через API. Розроблені основні потоки (фази).

FaultRules - це набір правил, які визначаються для обробки помилок, які виникають під час обробки запитів. В даному випадку, в ProxyEndpoint для ресурсів, є чотири таких правила:

1) SpikeArrest - правило викликається, якщо відбувається перевищення обмеження на кількість запитів. У цьому випадку, викликається крок "AM-SpikeArrestErrorMessage", який відповідає за відображення повідомлення про помилку.

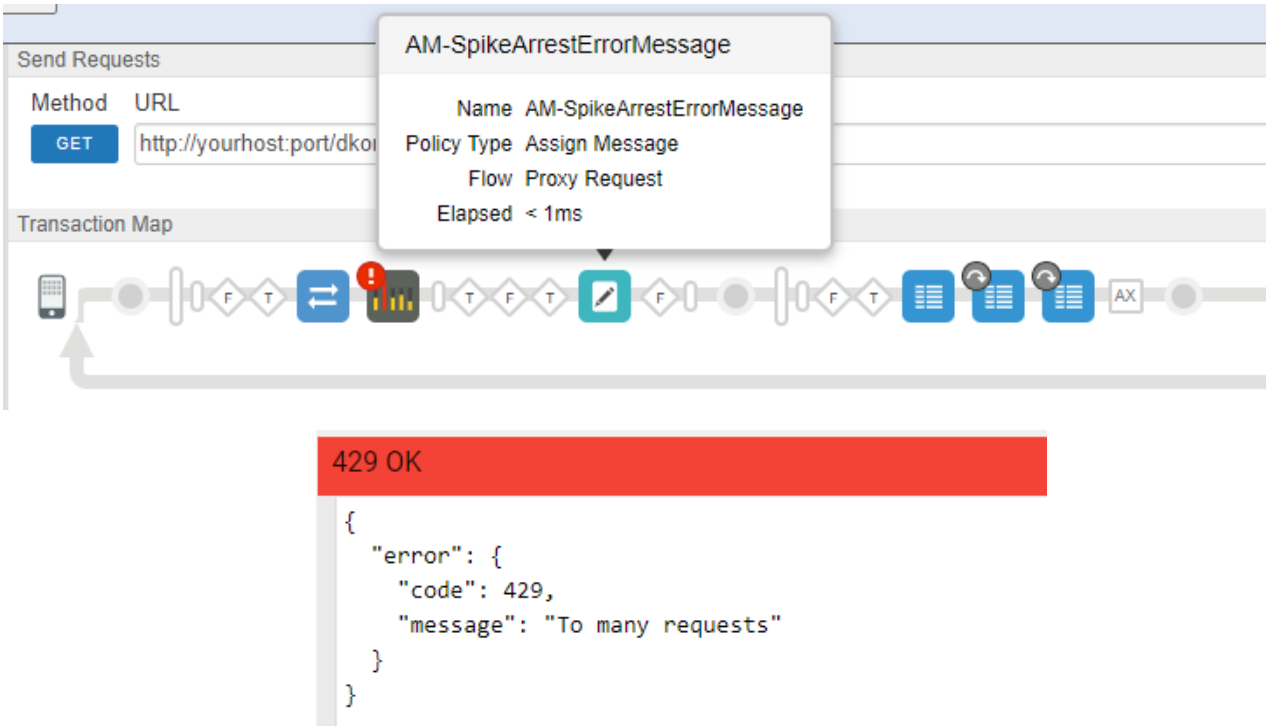

Рисунок 3.16 - Обробка помилки при різкому сплеску трафіку

2) Quota - правило викликається, якщо відбувається перевищення обмеження на кількість запитів. У цьому випадку, викликається крок "AM-QuotaErrorMessage", який відповідає за відображення повідомлення про помилку.

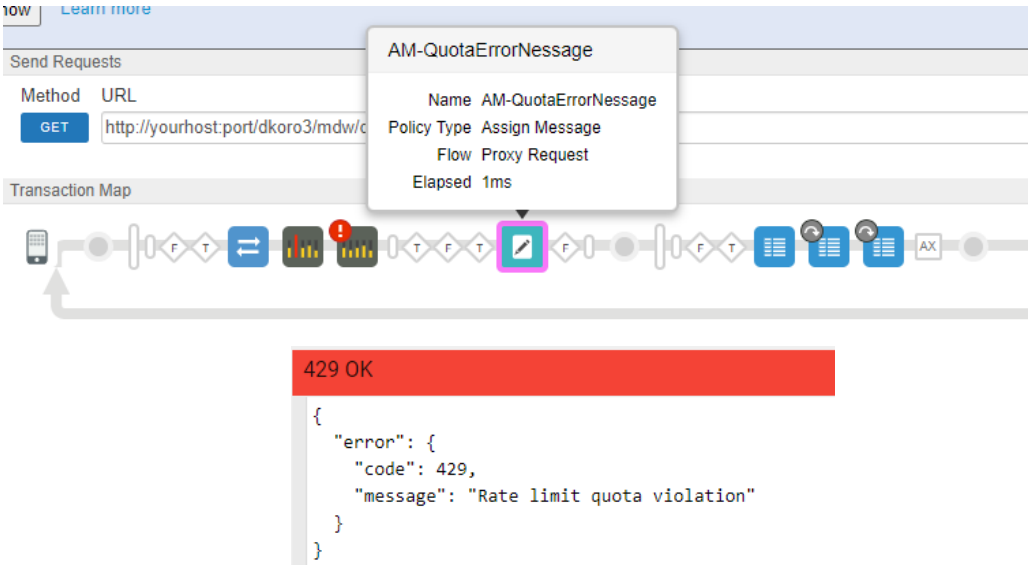

Рисунок 3.17 - Обробка помилки при перевищенню квоти трафіку

3) InvalidToken - це правило викликається, коли виникає помилка "Invalid Access Token", яка вказує на те, що токен доступу не є валідним. У цьому випадку, викликається крок " AM-InvalidAccessToken ", який відповідає за відображення повідомлення про помилку.

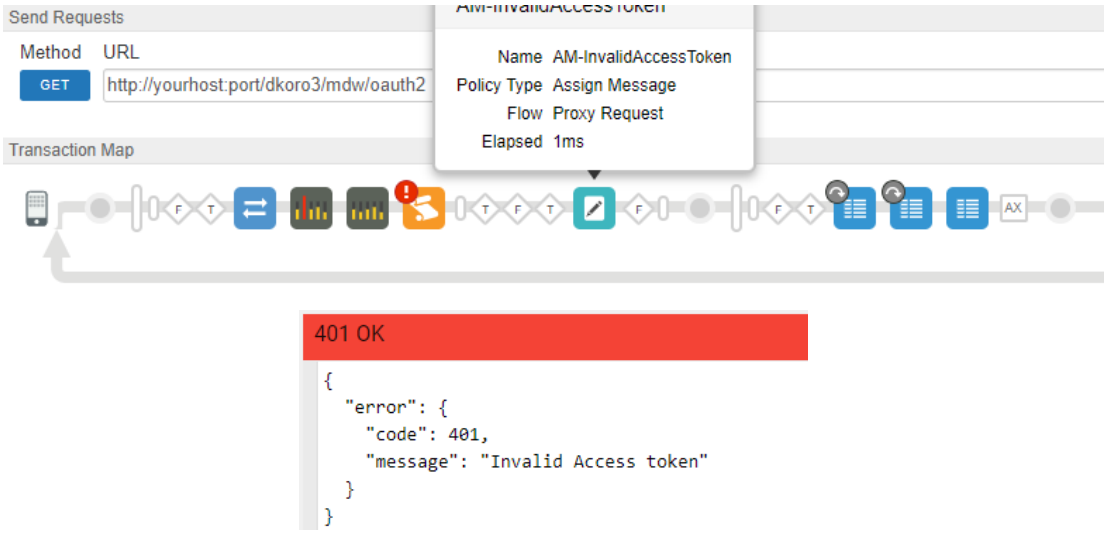

Рисунок 3.18 - Обробка помилки Invalid Access Token

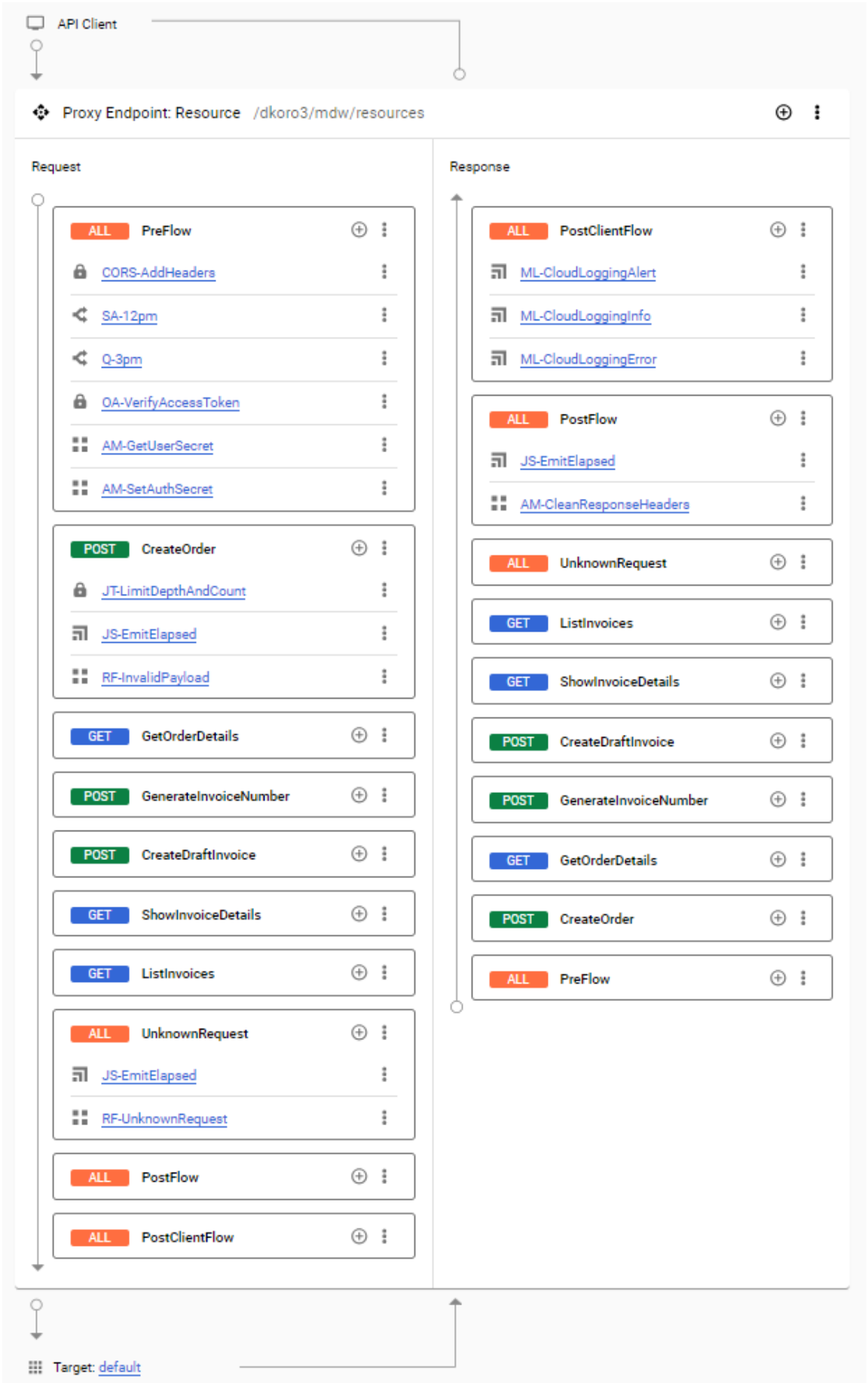

Рисунок 3.19 - Схема реалізації Recourses сервісу

4) TokenExpired - це правило викликається, коли виникає помилка "Access Token Expired", яка вказує на те, що токен доступу закінчився. У цьому випадку, викликається крок "AM-AccessTokenExpired", який відповідає за відображення повідомлення про помилку.

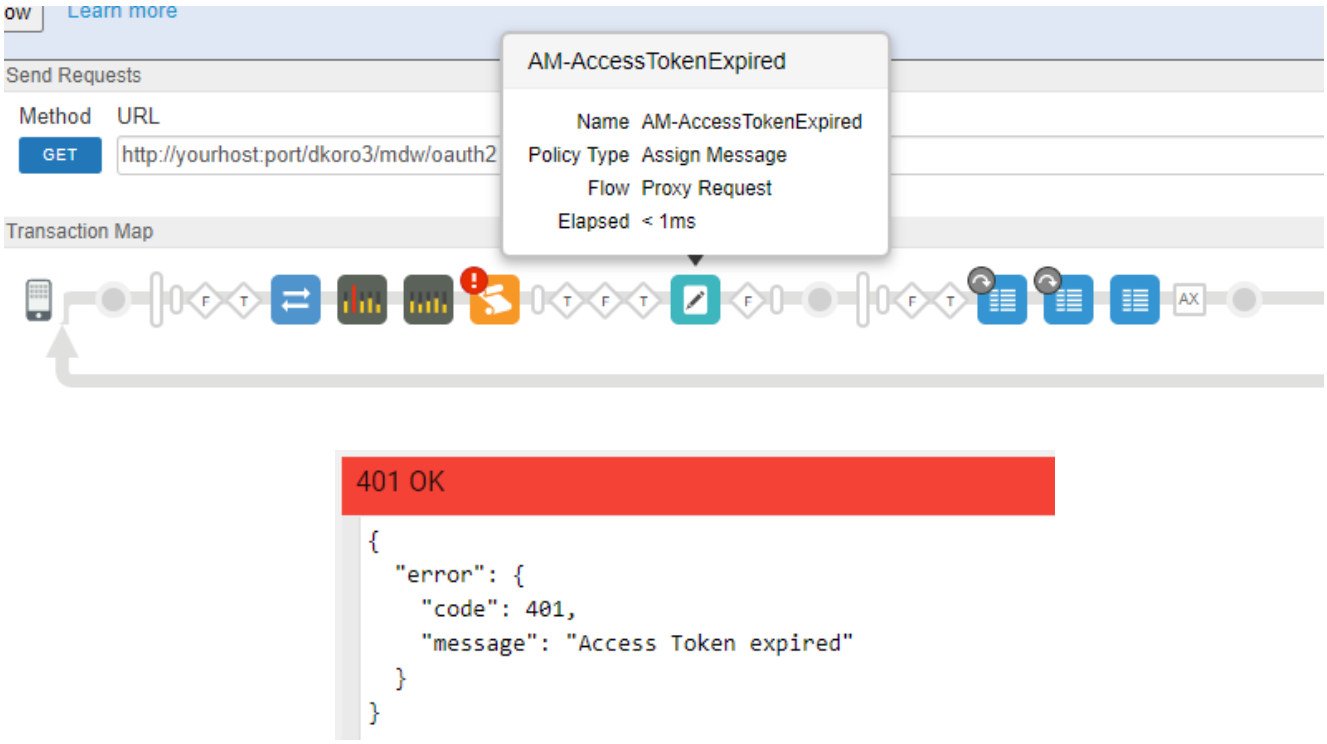

Рисунок 3.20 - Обробка помилки Access Token Expired

Кожне правило містить ім'я помилки, яке викликає це правило, та умову, яка вказує, коли це правило має бути виконане. Умова може включати різні параметри, такі як код статусу відповіді, або інші атрибути помилки.

PreFlow - це набір кроків, які виконуються перед тим, як запит буде відправлений до цільового сервера. У ProxyEndpoint для ресурсів, PreFlow містить наступні кроки:

1) CORS-AddHeaders - цей крок додає заголовки CORS (Cross-Origin Resource Sharing) до запиту. CORS - це механізм, який дозволяє веб-послугам обмінюватися даними між різними доменами.

2) SA-12pm - крок виконується для обмеження кількості запитів, дозволяє надсилати максимум 12 запитів на хвилину зі швидкістю один запит на 5

секундний інтервал. Якщо протягом 5 секундного інтервалу надходить більше одного запиту, такі запити блокуються. Запити допускаються, якщо кількість запитів менше налаштованого обмеження швидкості 12 за хвилину

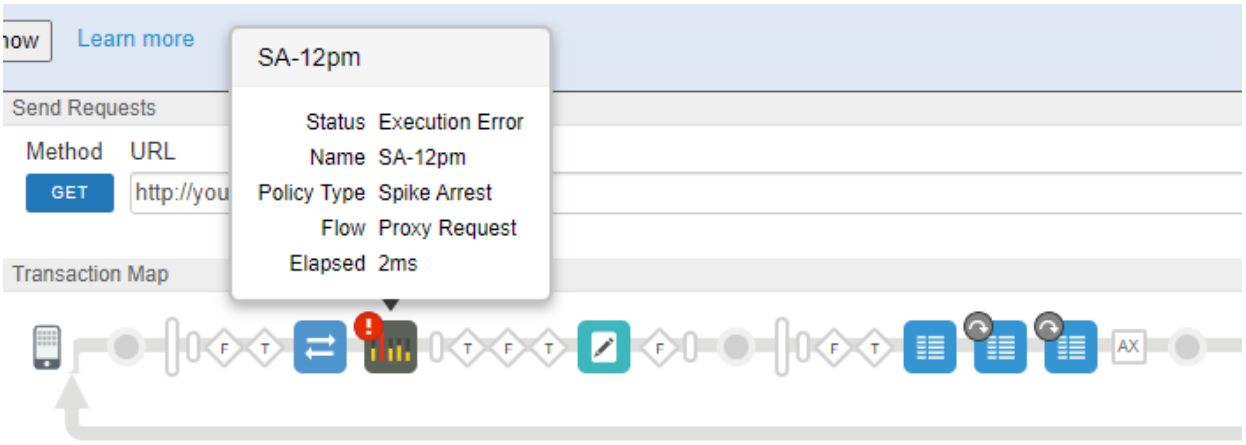

Рисунок 3.21 - Спрацювання захисту API при різкому сплеску трафіку

3) Q-3pm - крок виконується для обмеження кількості запитів, які можуть бути оброблені в певний проміжок часу. У цьому випадку, "3pm" вказує на обмеження до 3 запитів на хвилину.

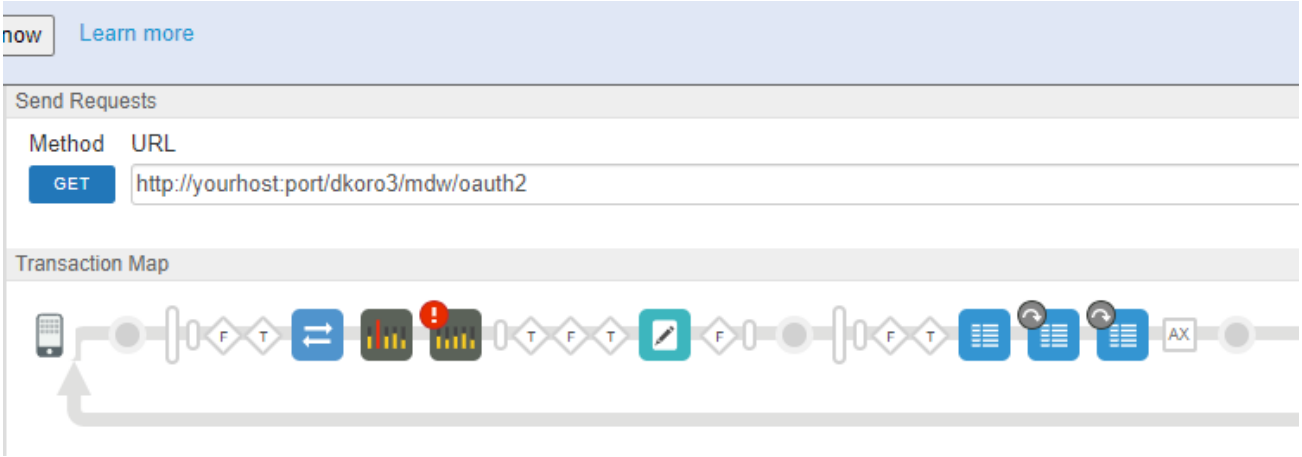

Рисунок 3.22 - Спрацювання захисту API при перевищені квоти трафіку

4) OA-VerifyAccessToken - цей крок перевіряє токен доступу, який був переданий в запиті, щоб перевірити його валідність.

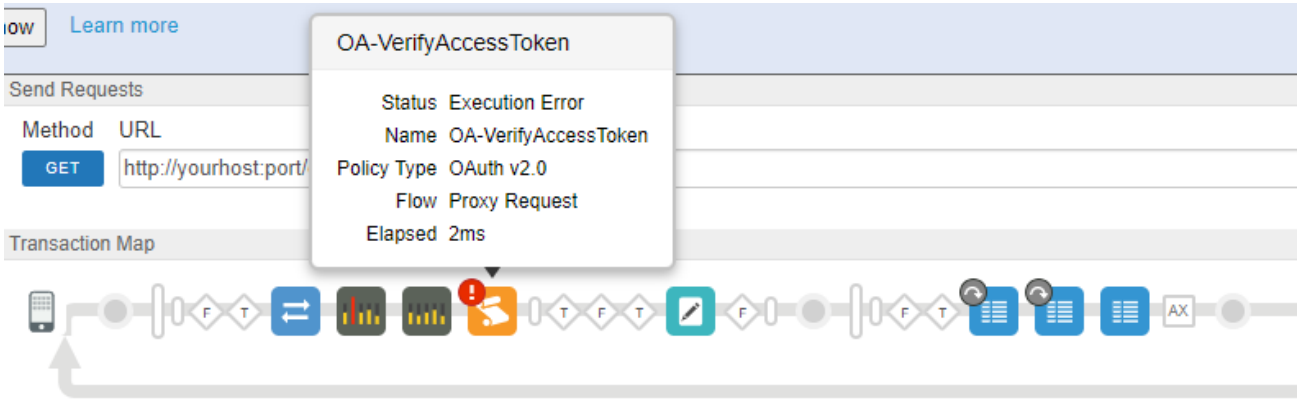

Рисунок 3.23 - Робота полісі при не валідному токену доступу

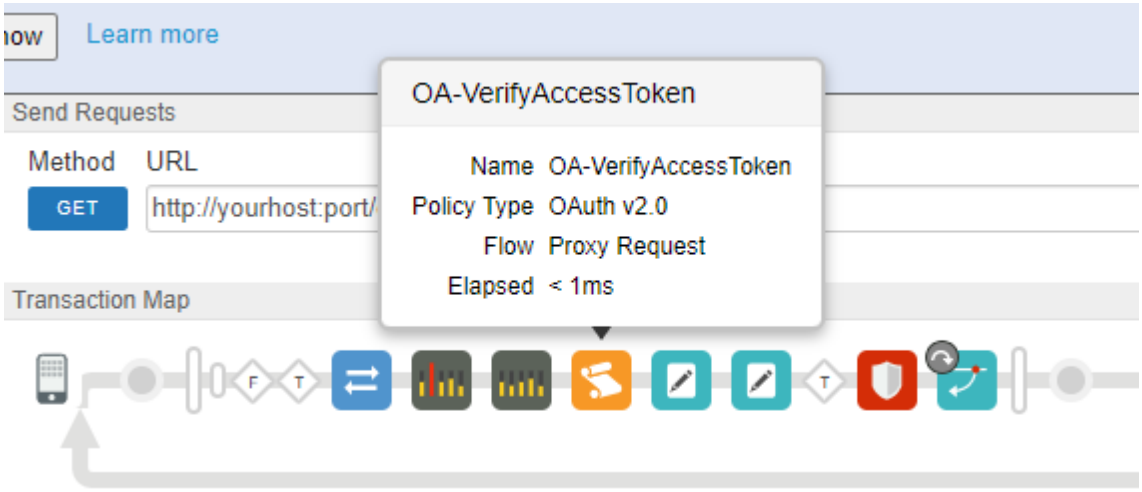

Рисунок 3.24 - Робота полісі з валідним токену доступу

5) AM-GetUserSecret - цей крок отримує секрет користувача з токену доступа, який використовується для аутентифікації на бекенд сервісі.

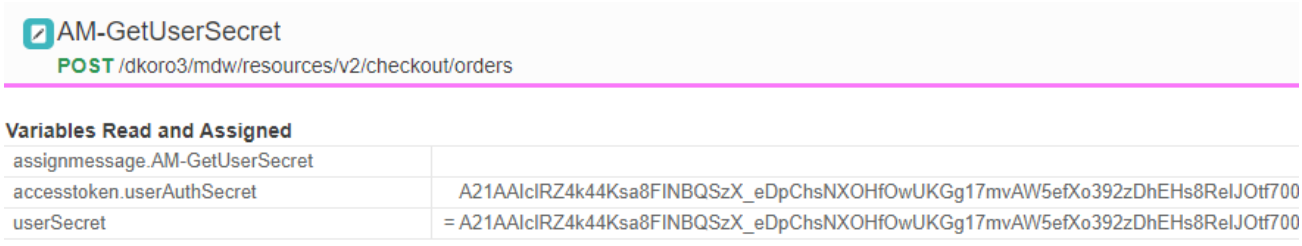

Рисунок 3.25 - Отримання за запис у змінну прихованого секрету користувача

6) AM-SetAuthSecret - цей крок встановлює секрет користувача, який був отриманий в попередньому кроці у хедер авторизації як Bearer токен доступу.

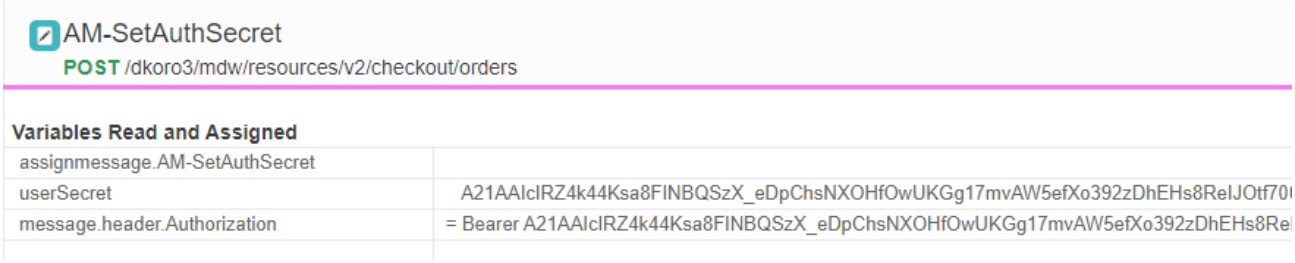

Рисунок 3.26 - Встановлення секрету користувача у хедер авторизації

Ці кроки виконуються в заданому порядку перед тим, як запит буде відправлений до цільового сервера.

PostFlow - це набір кроків, які виконуються після того, як отримано відповідь від цільового сервера. У ProxyEndpoint для ресурсів, PostFlow містить наступні кроки:

1) AM-CleanResponseHeaders - цей крок видаляє непотрібні заголовки відповіді, щоб зменшити об'єм даних, які передаються назад до клієнта.

Цей крок виконується після отримання відповіді від цільового сервера, але перед тим, як відповідь буде повернена клієнту.

Flows - це набір кроків, які виконуються при певних умовах, які визначаються в полі Condition. У ProxyEndpoint для ресурсів, Flows містить наступні кроки:

1) CreateOrder - цей крок виконується, коли клієнт намагається створити нове замовлення. Він перевіряє, чи відповідь містить неправильний JSON payload та відповідно встановлює відповідну помилку, якщо така атака виявлена.

2) GetOrderDetails - цей крок виконується, коли клієнт намагається отримати деталі замовлення. Він не потребує додаткових дій, тому відповідь буде повернена без змін.

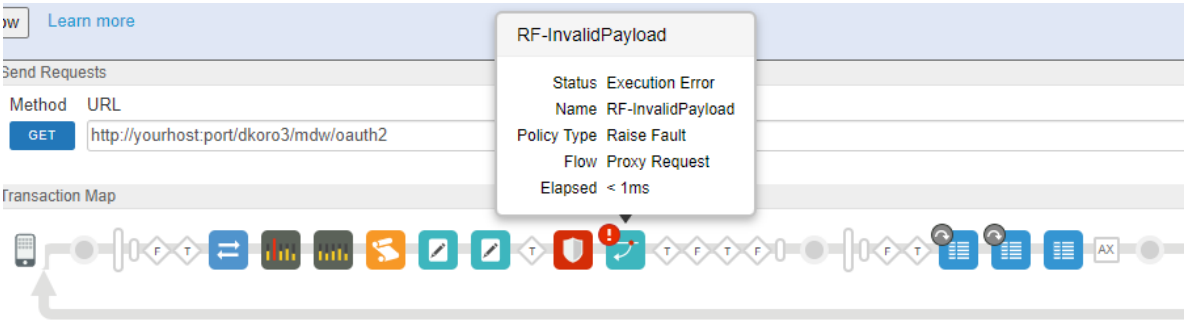

Рисунок 3.27 - Спрацювання захисту від атак неправильними JSON payload

3) GenerateInvoiceNumber - цей крок виконується, коли клієнт намагається згенерувати новий номер рахунку. Він не потребує додаткових дій, тому відповідь буде повернена без змін.

4) CreateDraftInvoice - цей крок виконується, коли клієнт намагається створити чернетку рахунку. Він не потребує додаткових дій, тому відповідь буде повернена без змін.

5) ShowInvoiceDetails - цей крок виконується, коли клієнт намагається отримати деталі рахунку. Він не потребує додаткових дій, тому відповідь буде повернена без змін.

6) ListInvoices - цей крок виконується, коли клієнт намагається отримати список рахунків. Він не потребує додаткових дій, тому відповідь буде повернена без змін.

7) UnknownRequest - цей крок виконується, коли отримано невідомий запит. Він встановлює відповідну помилку, якщо запит невідомий.

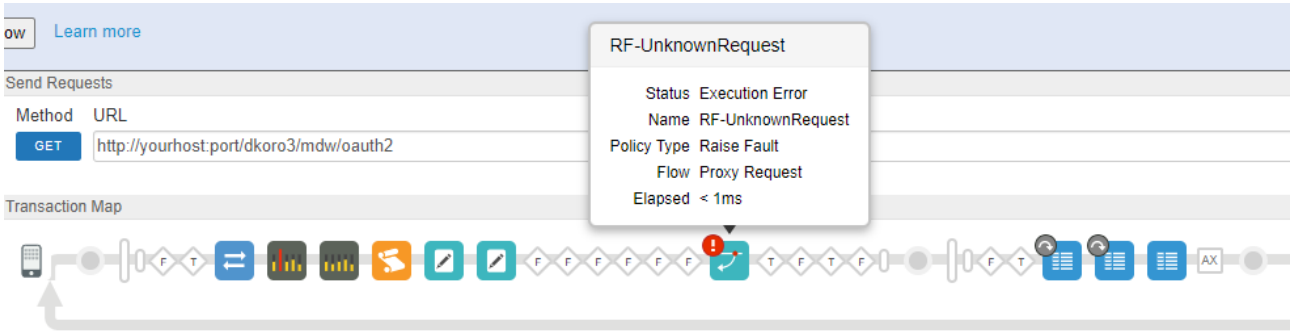

Рисунок 3.28 - Обробка помилки Unknown Request

PostClientFlow - це кроки, які виконуються після того, як відповідь відправлена клієнту. У ProxyEndpoint для ресурсів, PostClientFlow містить наступні кроки:

1) ML-CloudLoggingAlert - цей крок виконується, якщо відповідь містить код статусу 429. Він відправляє повідомлення про помилку до системи Google Cloud Logging.

2) ML-CloudLoggingInfo - цей крок виконується, якщо відповідь містить код статусу 200 (OK). Відправляє інформаційне повідомлення до системи Google Cloud Logging.

3) ML-CloudLoggingError - цей крок виконується, якщо відповідь містить код статусу 500, 400 або 401, 403, 404. Відправляє повідомлення про помилку до системи Google Cloud Logging.

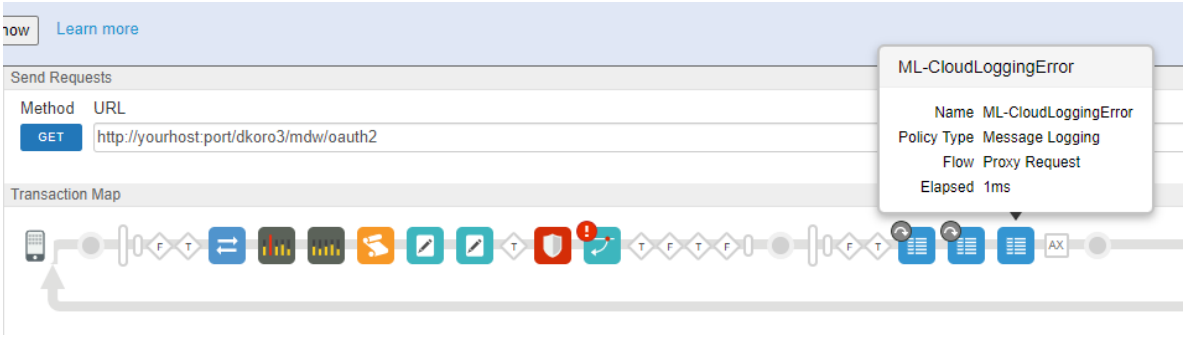

Рисунок 3.29 - Логування Errors у Cloud Logging

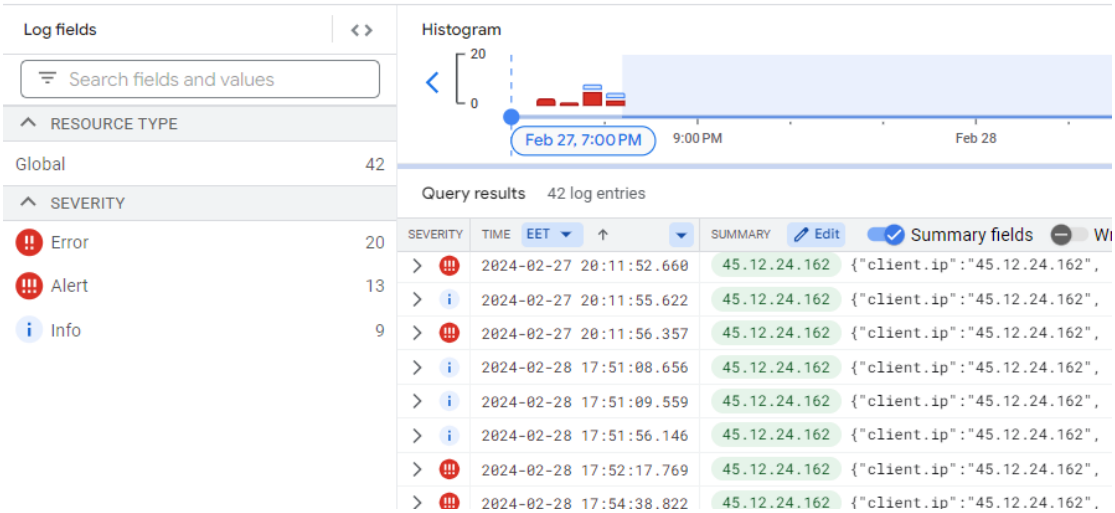

Рисунок 3.30 - Реєстрація Alert, Errors та Info повідомлень у Cloud Logging

RouteRule - це правило маршрутизації, яке визначає, які запити будуть оброблені цим проксі-сервером.

3.4. Реалізація порталу для розробників

Інтеграційний портал для розробників - це веб-портал, який дозволяє розробникам отримувати доступ до API, документації, ключів доступу та інших інструментів, необхідних для роботи з API.

Головна сторінка має посиланні на основні сторінки Get Started та API’s.

Get Started – сторінка з інформацією для розробників як почати працювати. API’s – сторінка з списком API та фільтрацією для швидкого пошуку. Також на головна сторінка містить посилання для реєстрації та авторизації розробників.

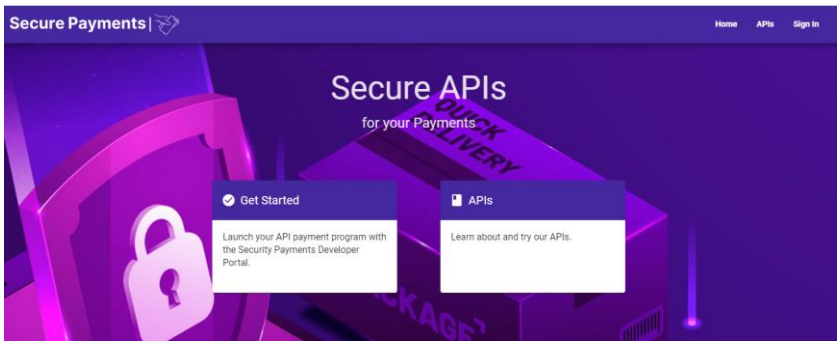

Рисунок 3.31 - Головна сторінка порталу

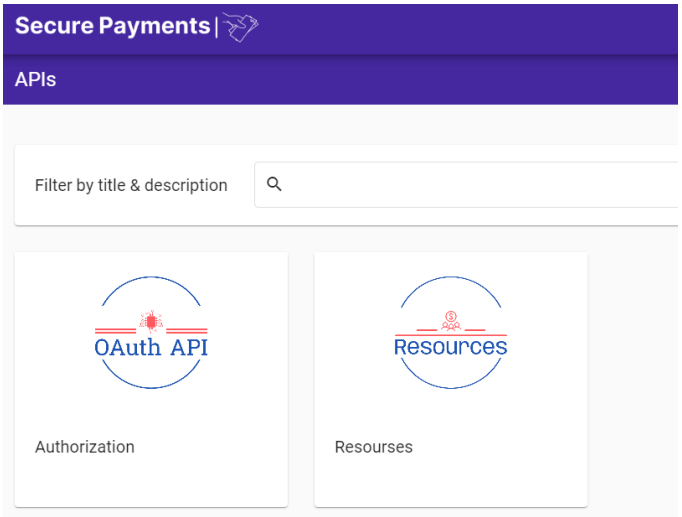

Рисунок 3.32 - Сторінка API’s

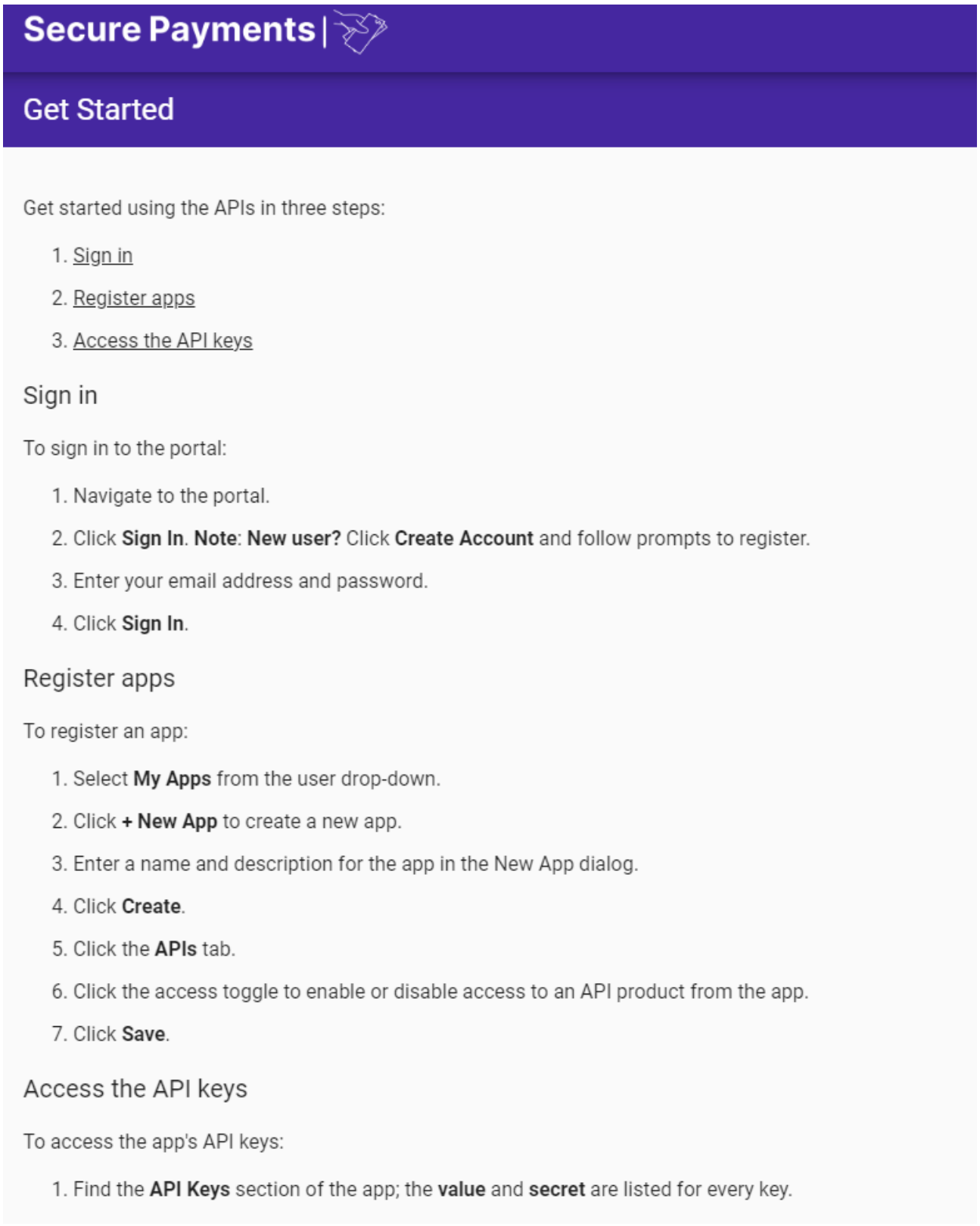

Рисунок 3.33 - Сторінка Get Started

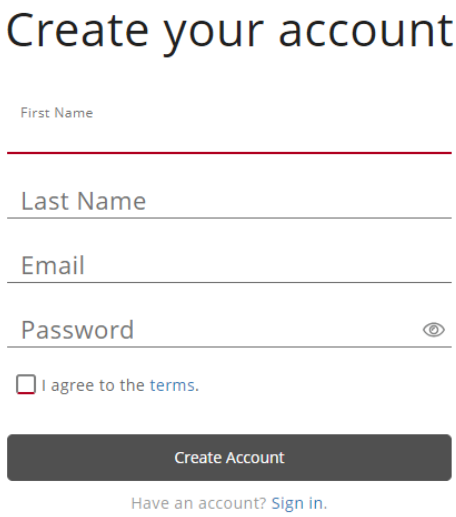

Рисунок 3.34 - Форма реєстрації розробника

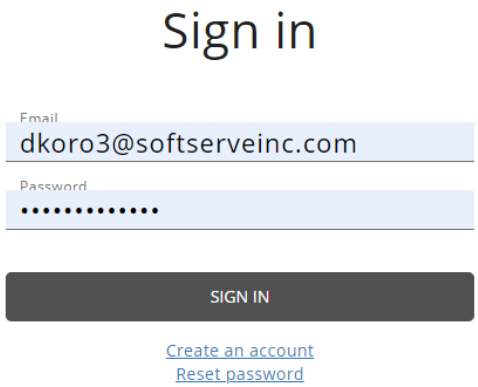

Рисунок 3.35 - Форма авторизації розробника

Після реєстрації розробник може зареєструвати свій додаток (App) отримати клієнтський секрет та пароль та почати працювати з документацією API.

Документація розроблена за стандартом OpenAPI Specs, який є стандартом для опису RESTful API. Це дозволяє описувати різні аспекти API, такі як шляхи, параметри, відповіді та схеми даних, у структурованому форматі, який може бути легко зрозумілий та використаний розробниками.

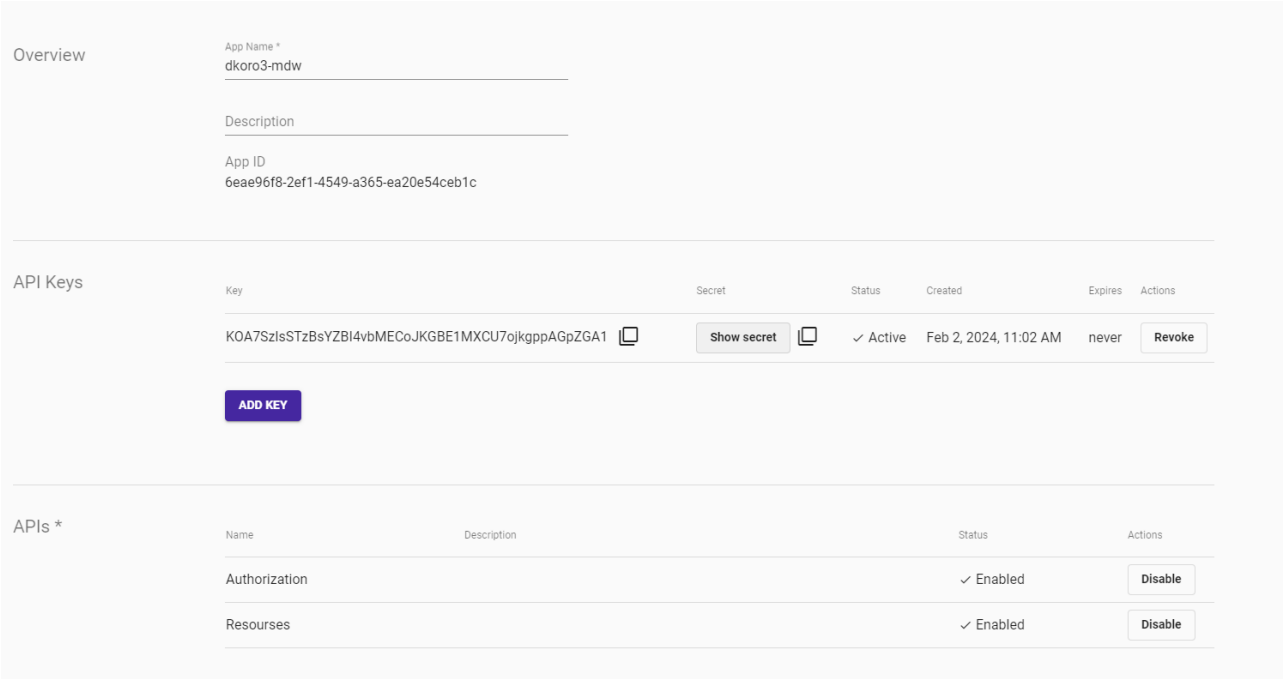

Рисунок 3.36 - Сторінка додатка (App) розробника

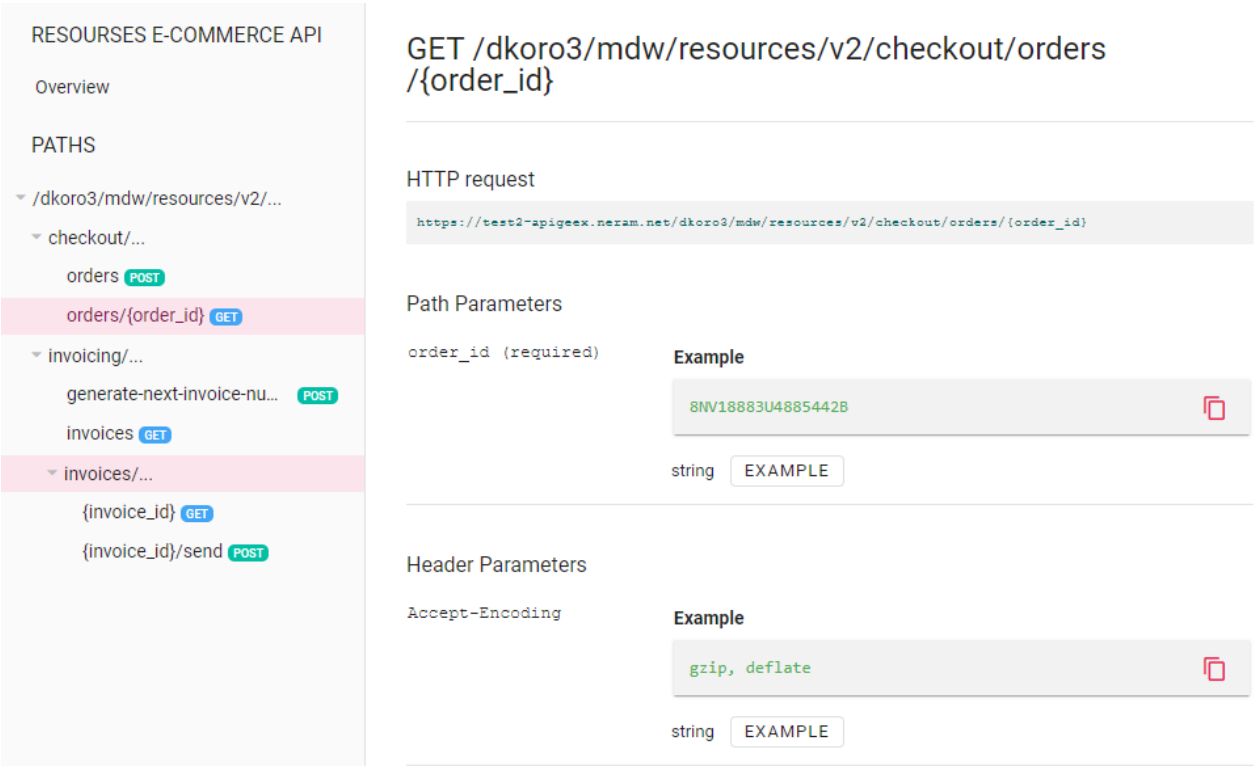

Рисунок 3.37 - OpenAPI документація API Resources

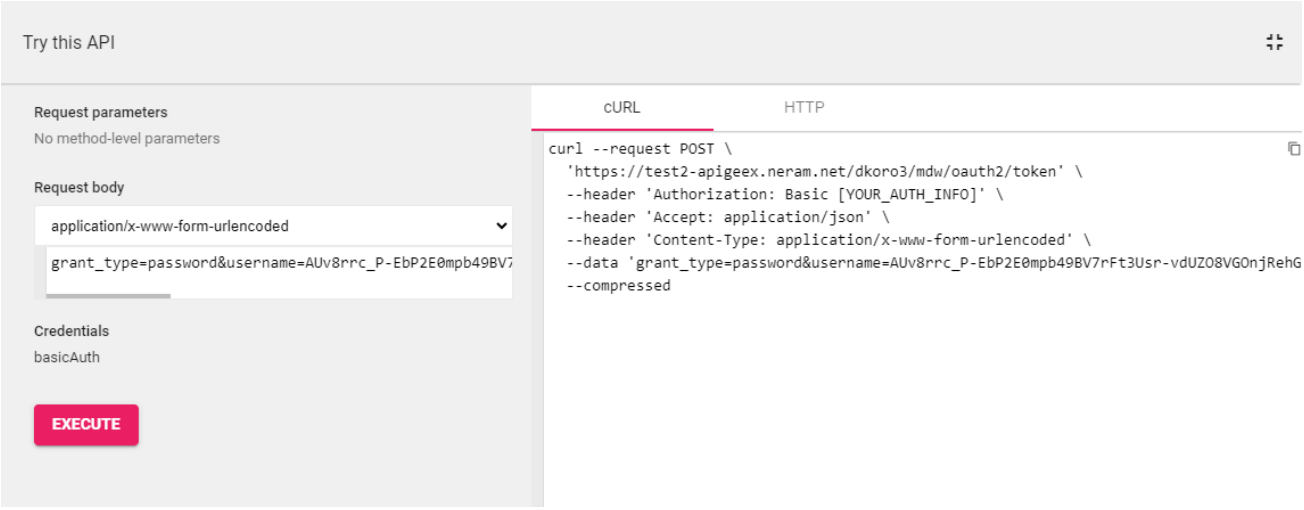

Рисунок 3.38 - Запит з девелоперського порталу до OAuth

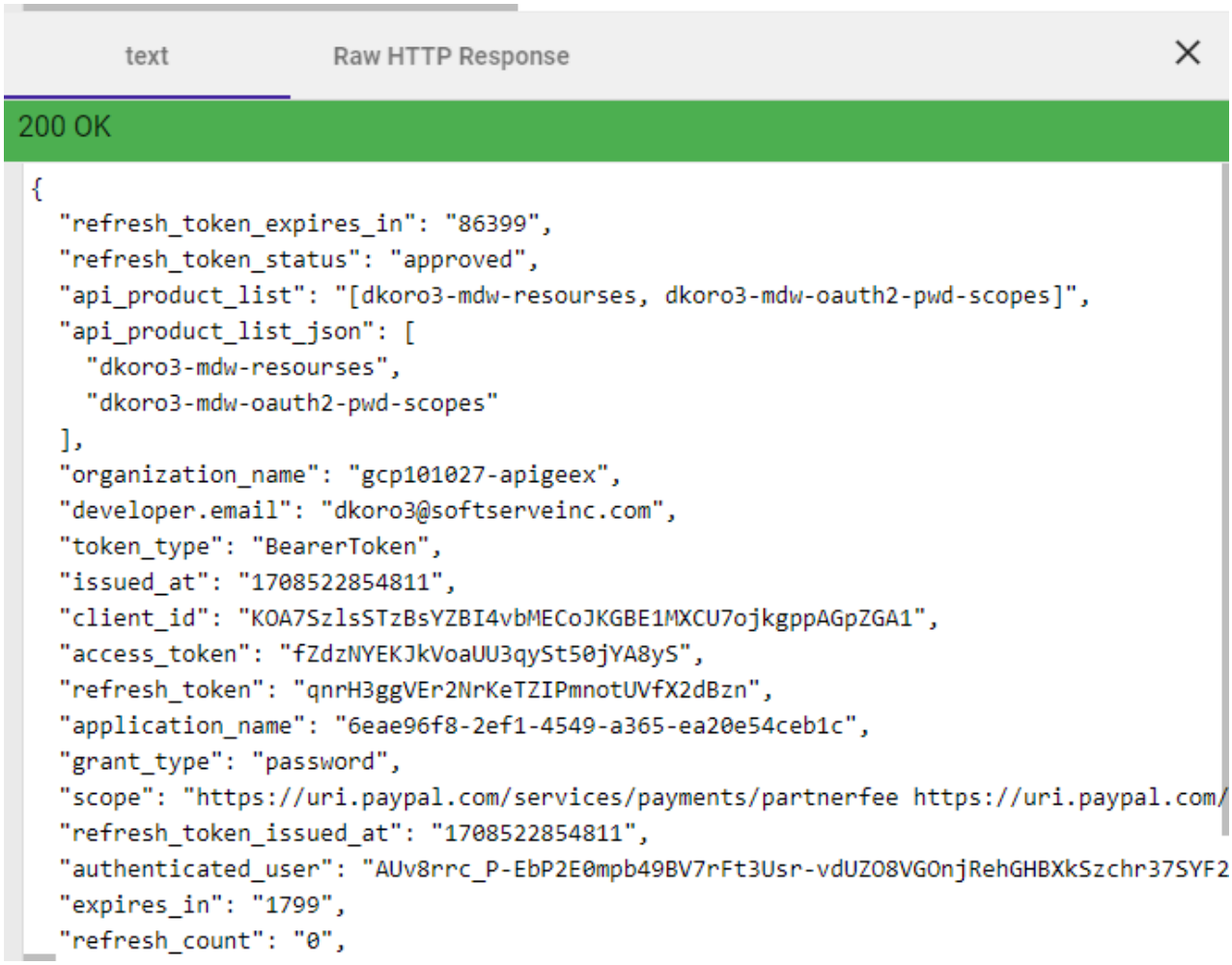

Рисунок 3.39 - Відповідь OAuth сервера

ВИСНОВКИ

У роботі було розглянуто два типи дозволів OAuth: "Client Credentials Grant Type" та "Resource Owner Password Grant Type", та їх використання на платформі Apigee X для електронної комерції. Було проведено порівняльний аналіз цих типів дозволів, виявлено їх переваги та недоліки.

У роботі було розроблено та впроваджено модель управління безпекою API на базі платформи Apigee X для електронної комерції, що використовує обидва типи дозволів OAuth. Ця модель дозволяє краще контролювати доступ до ресурсів, забезпечуючи швидку та надійну безпеку API. Крім того, вона використовує алгоритми управління трафіком для покращення безпеки. Результати роботи можуть бути використані для впровадження моделі управління безпекою API в сфері електронної комерції.

Було розроблено систему протоколювання та аудиту для виявлення та аналізу спроб несанкціонованого доступу або інших безпекових інцидентів, використовуючи Apigee X для збору, аналізу та зберігання логів безпеки за допомогою Google Cloud Logging. Ця система дозволяє забезпечити високий рівень безпеки та забезпечити відповідність з вимогами регуляторів.

У рамках даної роботи було розроблено та впроваджено девелоперський портал для платіжної системи, який забезпечує зручний та безпечний доступ до API для розробників. Девелоперський портал використовує механізми авторизації та аутентифікації, що забезпечують безпеку та конфіденційність даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cloudinary.com. eCommerce APIs: 6 Types of APIs and Their Business Benefits [Electronic resource]. 2024. URL: https://cloudinary.com/guides/e-commerce-platform/ecommerce-apis-6-types-of-apis-and-their-business-benefits (accessed: 26.01.2024).

2. Munsch A., Munsch P. The Future of API (Application Programming Interface) Security: The Adoption of APIs for Digital Communications and the Implications for Cyber Security Vulnerabilities // Journal of International Technology and Information Management. 2021. Vol. 29, № 3.

3. BigCommerce.com. Understanding Ecommerce APIs and the Role They Play in Your Tech Stack [Electronic resource]. 2024. URL: https://www.bigcommerce.com/articles/ecommerce-website-development/api-guide/ (accessed: 26.01.2024).

4. Nambudiri R., Ramnarayan S., Xavier C. Apigee: People Management Practices and the Challenge of Growth // Richard Ivey School of Business Foundation. 2017. Vol. 1, № 1.

5. Google. Apigee documentation [Electronic resource]. 2023. URL: https://cloud.google.com/apigee/docs (accessed: 03.01.2024).

6. salt.security. API Security Trends [Electronic resource]. 2023. URL: https://salt.security/ (accessed: 26.01.2024).

7. OWASP. OWASP API Security Top 10 2023. 2023.

8. datatracker.ietf.org. The OAuth 2.0 Authorization Framework [Electronic resource]. 2013. URL: https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-31 (accessed: 07.02.2024).

9. Idris M., Syarif I., Winarno I. Web Application Security Education Platform Based on OWASP API Security Project // EMITTER International Journal of Engineering Technology. 2022.

10.Siriwardena P. Advanced API Security // Advanced API Security. 2020.

11.Idris M., Syarif I., Winarno I. Development of Vulnerable Web Application Based on OWASP API Security Risks // International Electronics Symposium 2021: Wireless Technologies and Intelligent Systems for Better Human Lives, IES 2021 - Proceedings. 2021.

12.Hussain F. et al. Enterprise API Security and GDPR Compliance: Design and Implementation Perspective // IT Professional. 2020. Vol. 22, № 5.

13.OWASP. OWASP API Security Project // 2019. 2019.

14.Mchergui A., Alharbi S.J., Moulahi T. API Security Testing: The Challenges of Security Testing for Restful APIs // Int J Innov Res Sci Eng Technol. 2023. Vol. 8, № 5.

15. Rindell, K., Ruohonen, J., Holvitie, J., Hyrynsalmi, S., & Leppänen, V. (2021). Security in agile software development: A practitioner survey. Information and Software Technology, 131. https://doi.org/10.1016/j.infsof.2020.106488

16. Saeed, S., Jhanjhi, N. Z., Naqvi, M., & Humayun, M. (2019). Analysis of software development methodologies. International Journal of Computing and Digital Systems, 8(5), 445–460. https://doi.org/10.12785/ijcds/080502

17. Saeedi, K., & Visvizi, A. (2021). education sciences Software Development Methodologies, HEIs, and the Digital Economy. https://doi.org/10.3390/educsci

18. Richard Szeliski. (2022). Computer Vision: Algorithms and Applications, 2nd ed. The University of Washington.

19. Modzelewska-Kapituła, M., & Jun, S. (2022). The application of computer vision systems in meat science and industry – A review. Meat Science, 192, 108904. https://doi.org/10.1016/J.MEATSCI.2022.108904

20. Scott Krig. (2014). Computer Vision Metrics: Survey, Taxonomy, and Analysis (Vol. 442). Apress OPEN.

ДОДАТКИ

Програмний код API Proxy

dkoro3-mdw-oauth2-pwd-scopes.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<APIProxy revision="131" name="dkoro3-mdw-oauth2-pwd-scopes">
  <DisplayName>oauth2-pwd-scopes-demo</DisplayName>
  <Description>Dispense OAuth v2.0 Bearer tokens for password grant_types.
  <CreatedAt>1708356052658</CreatedAt>
  <LastModifiedAt>1708431657132</LastModifiedAt>
  <BasePaths>/dkoro3/mdw/oauth2</BasePaths>
  <BasePaths>/dkoro3/mdw/resources</BasePaths>
  <Policies>
    <Policy>AM-CleanResponseHeaders</Policy>
    <Policy>RF-AuthenticationFailed</Policy>
    <Policy>RF-InvalidGrantType</Policy>
    <Policy>RF-InvalidRequest</Policy>
    <Policy>RF-UnknownRequest</Policy>
    <Policy>SC-GetBackendSecret</Policy>
    <Policy>JS-EncodeAuthHeader</Policy>
    <Policy>EV-GetCalloutValues</Policy>
    <Policy>OA-GenerateAccessToken</Policy>
    <Policy>OA-VerifyAccessToken</Policy>
    <Policy>AM-SetAuthSecret</Policy>
    <Policy>AM-GetUserSecret</Policy>
    <Policy>CORS-AddHeaders</Policy>
    <Policy>SA-12pm</Policy>
    <Policy>Q-3pm</Policy>
    <Policy>AM-SpikeArrestErrorMessage</Policy>
    <Policy>AM-QuotaErrorNessage</Policy>
    <Policy>AM-ClientAuthenticationFailed</Policy>
    <Policy>AM-InvalidAccessToken</Policy>
    <Policy>AM-AccessTokenExpired</Policy>
    <Policy>JT-LimitDepthAndCount</Policy>
    <Policy>RF-InvalidPayload</Policy>
    <Policy>ML-CloudLoggingAlert</Policy>
    <Policy>ML-CloudLoggingInfo</Policy>
    <Policy>ML-CloudLoggingError</Policy>
  </Policies>
  <ProxyEndpoints>
    <ProxyEndpoint>OAuth</ProxyEndpoint>
    <ProxyEndpoint>Resource</ProxyEndpoint>
  </ProxyEndpoints>
  <Resources>
    <Resource>jsc://core-min.js</Resource>
    <Resource>jsc://enc-utf16-min.js</Resource>
    <Resource>jsc://enc-base64-min.js</Resource>
    <Resource>jsc://encodeAuthHeader.js</Resource>
  </Resources>
  <TargetEndpoints>
    <TargetEndpoint>default</TargetEndpoint>
  </TargetEndpoints>
</APIProxy>
```

Програмний код OAuth:

OAuth.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProxyEndpoint name="OAuth">
  <Description>the OAuth2 token dispensing endpoint</Description>
  <HTTPProxyConnection>
    <BasePath>/dkoro3/mdw/oauth2</BasePath>
    <Properties/>
    <VirtualHost>secure</VirtualHost>
  </HTTPProxyConnection>
  <FaultRules>
    <FaultRule name="SpikeArrest-OAuth">
      <Step>
        <Name>AM-SpikeArrestErrorMessage</Name>
      </Step>
      <Condition>(fault.name = "SpikeArrestViolation")</Condition>
    </FaultRule>
    <FaultRule name="Quotat-OAuth">
      <Step>
        <Name>AM-QuotaErrorNessage</Name>
      </Step>
      <Condition>(fault.name = "QuotaViolation")</Condition>
    </FaultRule>
  </FaultRules>
  <PreFlow name="PreFlow">
    <Request>
      <Step>
        <Name>CORS-AddHeaders</Name>
      </Step>
      <Step>
        <Name>SA-12pm</Name>
      </Step>
      <Step>
        <Name>Q-3pm</Name>
      </Step>
    </Request>
    <Response/>
  </PreFlow>
  <PostFlow name="PostFlow">
    <Request/>
    <Response>
      <Step>
        <Name>AM-CleanResponseHeaders</Name>
      </Step>
      <Step>
        <Name>AM-ClientAuthenticationFailed</Name>
        <Condition>oauthV2.failed = true</Condition>
      </Step>
    </Response>
  </PostFlow>
  <Flows>
    <Flow name="GetToken">
      <Description>dispense tokens for OAuth2.0, for password grant_type</Description>
      <Request>
        <Step>
          <Name>RF-InvalidGrantType</Name>
          <Condition>request.formparam.grant_type != "password"</Condition>
        </Step>
```

OAuth.xml

Press Alt+F1 for Accessibility Options.

```
            <Step>
              <Name>RF-InvalidRequest</Name>
              <Condition>request.formparam.username = null OR request.formparam.password = null</Condition>
            </Step>
            <Step>
              <Name>JS-EncodeAuthHeader</Name>
            </Step>
            <Step>
              <Name>SC-GetBackendSecret</Name>
            </Step>
            <Step>
              <Name>RF-AuthenticationFailed</Name>
              <Condition>calloutResponse.status.code = 401</Condition>
            </Step>
            <Step>
              <Name>EV-GetCalloutValues</Name>
            </Step>
            <Step>
              <Name>OA-GenerateAccessToken</Name>
            </Step>
          </Request>
          <Response/>
          <Condition>(proxy.pathsuffix MatchesPath "/token") and (request.verb = "POST")</Condition>
        </Flow>
        <Flow name="UnknownRequest">
          <Request>
            <Step>
              <Name>RF-UnknownRequest</Name>
            </Step>
          </Request>
          <Response/>
        </Flow>
      </Flows>
      <PostClientFlow>
        <Request/>
        <Response>
          <Step>
            <Name>ML-CloudLoggingAlert</Name>
            <Condition>(message.status.code = 429)</Condition>
          </Step>
          <Step>
            <Name>ML-CloudLoggingInfo</Name>
            <Condition>(response.status.code = 200)</Condition>
          </Step>
          <Step>
            <Name>ML-CloudLoggingError</Name>
            <Condition>(message.status.code = 500) or (message.status.code = 400) or (message.status.code = 401)</Condition>
          </Step>
        </Response>
      </PostClientFlow>
      <RouteRule name="NoRouteRule"/>
    </ProxyEndpoint>
```

Програмний код Resource:

Resource.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProxyEndpoint name="Resource">
  <Description>The operational API for resources</Description>
  <HTTPProxyConnection>
    <BasePath>/dkoro3/mdw/resources</BasePath>
    <Properties/>
    <VirtualHost>secure</VirtualHost>
  </HTTPProxyConnection>
  <FaultRules>
    <FaultRule name="SpikeArrest">
      <Step>
        <Name>AM-SpikeArrestErrorMessage</Name>
      </Step>
      <Condition>(fault.name = "SpikeArrestViolation")</Condition>
    </FaultRule>
    <FaultRule name="Quota">
      <Step>
        <Name>AM-QuotaErrorNessage</Name>
      </Step>
      <Condition>(fault.name = "QuotaViolation")</Condition>
    </FaultRule>
    <FaultRule name="InvalidToken">
      <Step>
        <Name>AM-InvalidAccessToken</Name>
      </Step>
      <Condition>(fault.name = "invalid_access_token") or (fault.name = "InvalidAccessToken")</Condition>
    </FaultRule>
    <FaultRule name="TokenExpired">
      <Step>
        <Name>AM-AccessTokenExpired</Name>
      </Step>
      <Condition>(fault.name = "access_token_expired")</Condition>
    </FaultRule>
  </FaultRules>
  <PreFlow name="PreFlow">
    <Request>
      <Step>
        <Name>CORS-AddHeaders</Name>
      </Step>
      <Step>
        <Name>SA-12pm</Name>
      </Step>
      <Step>
        <Name>Q-3pm</Name>
      </Step>
      <Step>
        <Name>OA-VerifyAccessToken</Name>
      </Step>
      <Step>
        <Name>AM-GetUserSecret</Name>
      </Step>
      <Step>
        <Name>AM-SetAuthSecret</Name>
      </Step>
    </Request>
```

Resource.xml

Press Alt+F1 for Accessibility Options.

```
    <Response/>
  </PreFlow>
  <PostFlow name="PostFlow">
    <Request/>
    <Response>
      <Step>
        <Name>AM-CleanResponseHeaders</Name>
      </Step>
    </Response>
  </PostFlow>
  <Flows>
    <Flow name="CreateOrder">
      <Description/>
      <Request>
        <Step>
          <Name>JT-LimitDepthAndCount</Name>
        </Step>
        <Step>
          <Name>RF-InvalidPayload</Name>
          <Condition>jsonattack.failed = "true"</Condition>
        </Step>
      </Request>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/checkout/orders") and (request.verb = "POST")</Condition>
    </Flow>
    <Flow name="GetOrderDetails">
      <Description/>
      <Request/>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/checkout/orders/*") and (request.verb = "GET")</Condition>
    </Flow>
    <Flow name="GenerateInvoiceNumber">
      <Description/>
      <Request/>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/invoicing/generate-next-invoice-number") and (request.verb = "POST")</Condition>
    </Flow>
    <Flow name="CreateDraftInvoice">
      <Description/>
      <Request/>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/invoicing/invoices/*/send") and (request.verb = "POST")</Condition>
    </Flow>
    <Flow name="ShowInvoiceDetails">
      <Description/>
      <Request/>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/invoicing/invoices/*") and (request.verb = "GET")</Condition>
    </Flow>
    <Flow name="ListInvoices">
      <Description/>
      <Request/>
      <Response/>
      <Condition>(proxy.pathsuffix MatchesPath "/v2/invoicing/invoices") and (request.verb = "GET")</Condition>
    </Flow>
```

Resource.xml

```
Press Alt+F1 for Accessibility Options.
111     <Flow name="UnknownRequest">
112       <Request>
113         <Step>
114           <Name>RF-UnknownRequest</Name>
115         </Step>
116       </Request>
117       <Response/>
118     </Flow>
119   </Flows>
120   <PostClientFlow>
121     <Request/>
122     <Response>
123       <Step>
124         <Name>ML-CloudLoggingAlert</Name>
125         <Condition>(message.status.code = 429)</Condition>
126       </Step>
127       <Step>
128         <Name>ML-CloudLoggingInfo</Name>
129         <Condition>(response.status.code = 200)</Condition>
130       </Step>
131       <Step>
132         <Name>ML-CloudLoggingError</Name>
133         <Condition>(message.status.code = 500) or (message.status.code = 400) or (message.status.code = 401)</Condition>
134       </Step>
135     </Response>
136   </PostClientFlow>
137   <RouteRule name="default">
138     <TargetEndpoint>default</TargetEndpoint>
139   </RouteRule>
140 </ProxyEndpoint>
141
```

Програмний код політик:

AM-AccessTokenExpired.xml

```
Press Alt+F1 for Accessibility Options.
1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <AssignMessage continueOnError="false" enabled="true" name="AM-AccessTokenExpired">
3     <DisplayName>AM-AccessTokenExpired</DisplayName>
4     <Properties/>
5     <Set>
6       <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
7         <![CDATA[{
8     "error" : {
9       "code" : 401,
10      "message" : "Access Token expired"
11    }
12  }
13  ]]>
14      </Payload>
15      <StatusCode>401</StatusCode>
16      <ReasonPhrase>Unauthorized</ReasonPhrase>
17    </Set>
18    <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
19    <AssignTo createNew="false" transport="http" type="response"/>
20  </AssignMessage>
21
```

AM-CleanResponseHeaders.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage name="AM-CleanResponseHeaders">
  <Remove>
    <Headers>
      <Header name="Accept"/>
      <Header name="user-agent"/>
      <Header name="Host"/>
      <Header name="x-forwarded-for"/>
      <Header name="X-Forwarded-Proto"/>
      <Header name="X-Forwarded-Port"/>
      <Header name="apikey"/>
      <Header name="date"/>
      <Header name="Authorization"/>
      <Header name="Signature"/>
      <Header name="X-Powered-By"/>
    </Headers>
  </Remove>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
</AssignMessage>
```

AM-ClientAuthenticationFailed.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-ClientAuthenticationFailed">
  <DisplayName>AM-ClientAuthenticationFailed</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
      <![CDATA[{
  "error" : {
    "code" : 401,
    "message" : "Client Authentication failed"
  }
}
]]>
    </Payload>
    <StatusCode>401</StatusCode>
    <ReasonPhrase>Unauthorized</ReasonPhrase>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
</AssignMessage>
```

AM-GetUserSecret.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-GetUserSecret">
  <DisplayName>AM-GetUserSecret</DisplayName>
  <Properties/>
  <AssignVariable>
    <Name>userSecret</Name>
    <Ref>accesstoken.userAuthSecret</Ref>
  </AssignVariable>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

AM-InvalidAccessToken.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-InvalidAccessToken">
  <DisplayName>AM-InvalidAccessToken</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
      <![CDATA[{
  "error" : {
    "code" : 401,
    "message" : "Invalid Access token"
  }
}
]]>
    </Payload>
    <StatusCode>401</StatusCode>
    <ReasonPhrase>Unauthorized</ReasonPhrase>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
</AssignMessage>
```

AM-SpikeArrestErrorMessage.xml

```
Press Alt+F1 for Accessibility Options.
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-SpikeArrestErrorMessage">
  <DisplayName>AM-SpikeArrestErrorMessage</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json">{
  "error" : {
    "code" : 429,
    "message" : "To many requests"
  }
}</Payload>
    <StatusCode>429</StatusCode>
    <ReasonPhrase>Too Many Requests</ReasonPhrase>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
</AssignMessage>
```

AM-QuotaErrorNessage.xml

```
Press Alt+F1 for Accessibility Options.
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-QuotaErrorNessage">
  <DisplayName>AM-QuotaErrorNessage</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json">{
  "error" : {
    "code" : 429,
    "message" : "Rate limit quota violation"
  }
}</Payload>
    <StatusCode>429</StatusCode>
    <ReasonPhrase>Too Many Requests</ReasonPhrase>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
</AssignMessage>
```

AM-SetAuthSecret.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage continueOnError="false" enabled="true" name="AM-SetAuthSecret">
  <DisplayName>AM-SetAuthSecret</DisplayName>
  <Properties/>
  <Set>
    <Headers>
      <Header name="Authorization ">Bearer {userSecret}</Header>
    </Headers>
  </Set>
  <AssignVariable>
    <Name>name</Name>
    <Value/>
    <Ref/>
  </AssignVariable>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</AssignMessage>
```

CORS-AddHeaders.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CORS continueOnError="false" enabled="true" name="CORS-AddHeaders">
  <DisplayName>CORS-AddHeaders</DisplayName>
  <AllowOrigins>{request.header.origin}</AllowOrigins>
  <AllowMethods>GET, PUT, POST, DELETE</AllowMethods>
  <AllowHeaders>origin, x-requested-with, accept, content-type, authorization, grant_type, referer</AllowHeaders>
  <ExposeHeaders>*</ExposeHeaders>
  <MaxAge>3628800</MaxAge>
  <AllowCredentials>false</AllowCredentials>
  <GeneratePreflightResponse>true</GeneratePreflightResponse>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</CORS>
```

EV-GetCalloutValues.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables continueOnError="false" enabled="true" name="EV-GetCalloutValues">
  <DisplayName>EV-GetResponseCalloutValues</DisplayName>
  <JSONPayload>
    <Variable name="scope">
      <JSONPath>$.scope</JSONPath>
    </Variable>
    <Variable name="userAuthSecret">
      <JSONPath>$.access_token</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse</Source>
</ExtractVariables>
```

JS-EncodeAuthHeader.xml

```
Press Alt+F1 for Accessibility Options.
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Javascript continueOnError="false" enabled="true" timeLimit="200" name="JS-EncodeAuthHeader">
  <DisplayName>JS-EncodeAuthHeader</DisplayName>
  <Properties/>
  <IncludeURL>jsc://core-min.js</IncludeURL>
  <IncludeURL>jsc://enc-utf16-min.js</IncludeURL>
  <IncludeURL>jsc://enc-base64-min.js</IncludeURL>
  <ResourceURL>jsc://encodeAuthHeader.js</ResourceURL>
</Javascript>
```

encodeAuthHeader.js

```
Press Alt+F1 for Accessibility Options.
var username = context.getVariable("request.formparam.username");
var password = context.getVariable("request.formparam.password");

var words = CryptoJS.enc.Latin1.parse(username + ":" + password);
var base64 = CryptoJS.enc.Base64.stringify(words);
context.setVariable("encodedAuthHeader","Basic " + base64);
```

JT-LimitDepthAndCount.xml

```
Press Alt+F1 for Accessibility Options.
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JSONThreatProtection continueOnError="true" enabled="true" name="JT-LimitDepthAndCount">
  <DisplayName>JT-LimitDepthAndCount</DisplayName>
  <Properties/>
  <ObjectEntryCount>20</ObjectEntryCount>
  <Source>request</Source>
</JSONThreatProtection>
```

ML-CloudLoggingAlert.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MessageLogging continueOnError="false" enabled="true" name="ML-CloudLoggingAlert">
  <DisplayName>ML-CloudLoggingAlert</DisplayName>
  <CloudLogging>
    <LogName>projects/{organization.name}/logs/dkoro3-mdw</LogName>
    <Message contentType="application/json">{
        "organization": "{organization.name}",
        "environment": "{environment.name}",
        "proxy": "{apiproxy.name}",
        "verb": "{request.verb}",
        "client.ip": "{proxy.client.ip}",
        "proxy.basepath": "{proxy.basepath}",
        "response.code": "{message.status.code}",
        "response.reason": "{message.reason.phrase}",
        "response.content": "{escapeJSON(message.content)}"
    }</Message>
  </CloudLogging>
  <logLevel>ALERT</logLevel>
</MessageLogging>
```

ML-CloudLoggingError.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MessageLogging continueOnError="false" enabled="true" name="ML-CloudLoggingError">
  <DisplayName>ML-CloudLoggingError</DisplayName>
  <CloudLogging>
    <LogName>projects/{organization.name}/logs/dkoro3-mdw</LogName>
    <Message contentType="application/json">{
        "organization": "{organization.name}",
        "environment": "{environment.name}",
        "proxy": "{apiproxy.name}",
        "verb": "{request.verb}",
        "client.ip": "{proxy.client.ip}",
        "proxy.basepath": "{proxy.basepath}",
        "response.code":"{message.status.code}",
        "response.reason":"{message.reason.phrase}",
        "response.content":"{escapeJSON(message.content)}"
    }</Message>
  </CloudLogging>
  <logLevel>ERROR</logLevel>
</MessageLogging>
```

ML-CloudLoggingInfo.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MessageLogging continueOnError="false" enabled="true" name="ML-CloudLoggingInfo">
  <DisplayName>ML-CloudLoggingInfo</DisplayName>
  <CloudLogging>
    <LogName>projects/{organization.name}/logs/dkoro3-mdw</LogName>
    <Message contentType="application/json">{
        "organization": "{organization.name}",
        "environment": "{environment.name}",
        "proxy": "{apiproxy.name}",
        "verb": "{request.verb}",
        "client.ip": "{proxy.client.ip}",
        "proxy.basepath": "{proxy.basepath}",
        "response.code":"{response.status.code}",
        "response.reason":"{message.reason.phrase}",
        "response.content":"{escapeJSON(message.content)}"
    }</Message>
  </CloudLogging>
</MessageLogging>
```

OA-GenerateAccessToken.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 name="OA-GenerateAccessToken">
  <Operation>GenerateAccessToken</Operation>
  <ExpiresIn ref="flow.variable">1800000</ExpiresIn>
  <RefreshTokenExpiresIn>28800000</RefreshTokenExpiresIn>
  <SupportedGrantTypes>
    <GrantType>password</GrantType>
  </SupportedGrantTypes>
  <GrantType>request.formparam.grant_type</GrantType>
  <Scope>scope</Scope>
  <Attributes>
    <Attribute name="authenticated_user" ref="request.formparam.username" display="true">UNDEFINED</Attribute>
    <Attribute name="grant_type" ref="request.formparam.grant_type" display="true">UNDEFINED</Attribute>
    <Attribute name="userAuthSecret" ref="userAuthSecret" display="false">UNDEFINED</Attribute>
    <Attribute name="scope" ref="scope" display="true">UNDEFINED</Attribute>
  </Attributes>
  <GenerateResponse enabled="true"/>
</OAuthV2>
```

OA-VerifyAccessToken.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 continueOnError="false" enabled="true" name="OA-VerifyAccessToken">
  <DisplayName>OA-VerifyAccessToken</DisplayName>
  <Properties/>
  <Attributes/>
  <ExternalAuthorization>false</ExternalAuthorization>
  <Operation>VerifyAccessToken</Operation>
  <SupportedGrantTypes/>
  <GenerateResponse enabled="true"/>
  <Tokens/>
  <RFCCompliantRequestResponse>true</RFCCompliantRequestResponse>
</OAuthV2>
```

Q-3pm.xml

Press Alt+F1 for Accessibility Options.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota continueOnError="false" enabled="true" name="Q-3pm">
  <DisplayName>Q-3pm</DisplayName>
  <Properties/>
  <Identifier ref="request.header.authorization"/>
  <Allow count="4"/>
  <Interval>3</Interval>
  <TimeUnit>minute</TimeUnit>
  <Distributed>true</Distributed>
  <Synchronous>true</Synchronous>
</Quota>
```

RF-AuthenticationFailed.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault name="RF-AuthenticationFailed">
  <DisplayName>RF-AuthenticationFailed</DisplayName>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <FaultResponse>
    <Set>
      <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
        <![CDATA[{
  "error" : {
    "code" : 401,
    "message" : "User Authentication failed"
  }
}
]]>
      </Payload>
      <StatusCode>401</StatusCode>
      <ReasonPhrase>Unauthorized</ReasonPhrase>
    </Set>
  </FaultResponse>
</RaiseFault>
```

RF-InvalidGrantType.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault name="RF-InvalidGrantType">
  <DisplayName>RF-InvalidGrantType</DisplayName>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <FaultResponse>
    <Set>
      <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
        <![CDATA[{
  "error" : {
    "code" : 400,
    "message" : "Invalid grant type"
  }
}
]]>
      </Payload>
      <StatusCode>400</StatusCode>
      <ReasonPhrase>Bad Request</ReasonPhrase>
    </Set>
  </FaultResponse>
</RaiseFault>
```

RF-InvalidPayload.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault continueOnError="false" enabled="true" name="RF-InvalidPayload">
  <DisplayName>RF-InvalidPayload</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <FaultResponse>
    <Set>
      <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
        <![CDATA[{
  "error" : {
    "code" : 400,
    "message" : "The request failed due to a JSON payload."
  }
}
]]>
      </Payload>
      <StatusCode>400</StatusCode>
      <ReasonPhrase>Bad Request</ReasonPhrase>
    </Set>
  </FaultResponse>
</RaiseFault>
```

RF-InvalidRequest.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault name="RF-InvalidRequest">
  <DisplayName>RF-InvalidRequest</DisplayName>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <FaultResponse>
    <Set>
      <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
        <![CDATA[{
  "error" : {
    "code" : 400,
    "message" : "Missing username or password"
  }
}
]]>
      </Payload>
      <StatusCode>400</StatusCode>
      <ReasonPhrase>Bad Request</ReasonPhrase>
    </Set>
  </FaultResponse>
</RaiseFault>
```

RF-UnknownRequest.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault name="RF-UnknownRequest">
  <DisplayName>RF-UnknownRequest</DisplayName>
  <Description>Unknown Request</Description>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <FaultResponse>
    <Set>
      <Payload contentType="application/json" variablePrefix="%" variableSuffix="#">
        <![CDATA[{
  "error" : {
    "code" : 404,
    "message" : "That request was unknown. Try a different request."
  }
}
]]>
      </Payload>
      <StatusCode>404</StatusCode>
      <ReasonPhrase>Not Found</ReasonPhrase>
    </Set>
  </FaultResponse>
</RaiseFault>
```

SA-12pm.xml

Press Alt+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SpikeArrest continueOnError="false" enabled="true" name="SA-12pm">
  <DisplayName>SA-12pm</DisplayName>
  <Properties/>
  <Identifier ref="request.header.some-header-name"/>
  <MessageWeight ref="request.header.weight"/>
  <Rate>12pm</Rate>
</SpikeArrest>
```

SC-GetBackendSecret.xml

Press Alt+F1 for Accessibility Options.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ServiceCallout continueOnError="false" enabled="true" name="SC-GetBackendSecret">
3    <DisplayName>SC-GetBackendSecret</DisplayName>
4    <Properties/>
5    <Request>
6      <Set>
7        <Verb>POST</Verb>
8        <Headers>
9          <Header name="Content-Type">application/x-www-form-urlencoded</Header>
10         <Header name="Authorization">{encodedAuthHeader}</Header>
11       </Headers>
12       <FormParams>
13         <FormParam name="grant_type">client_credentials</FormParam>
14         <FormParam name="ignoreCache">true</FormParam>
15         <FormParam name="return_authn_schemes">true</FormParam>
16         <FormParam name="return_client_metadata">true</FormParam>
17         <FormParam name="return_unconsented_scopes">true</FormParam>
18       </FormParams>
19     </Set>
20   </Request>
21   <Response>calloutResponse</Response>
22   <HTTPTargetConnection>
23     <Properties>
24       <Property name="success.codes">2XX,4XX</Property>
25     </Properties>
26     <URL>https://api-m.sandbox.paypal.com/v1/oauth2/token</URL>
27   </HTTPTargetConnection>
28 </ServiceCallout>
29
```

default.xml

Press Alt+F1 for Accessibility Options.

```
1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <TargetEndpoint name="default">
3     <Description/>
4     <FaultRules/>
5     <PreFlow name="PreFlow">
6       <Request/>
7       <Response/>
8     </PreFlow>
9     <PostFlow name="PostFlow">
10      <Request/>
11      <Response>
12      </Response>
13    </PostFlow>
14    <Flows/>
15    <HTTPTargetConnection>
16      <Properties/>
17      <URL>https://api-m.sandbox.paypal.com</URL>
18    </HTTPTargetConnection>
19  </TargetEndpoint>
20
```

OpenAPI Specification OAuth Server

apispec.yaml

```yaml
openapi: 3.0.0
info:
  title: OAuth e-Commerce API
  version: 1.0.0
servers:
  - url: https://test2-apigeex.neram.net/dkoro3/mdw
components:
  securitySchemes:
    basicAuth:
      type: http
      scheme: basic
paths:
  /oauth2/token:
    post:
      tags:
        - OAuth
      summary: Get Token
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                grant_type:
                  type: string
                  example: password
                username:
                  type: string
                  example: >-
                    AUv8rrc_P-EbP2E0mpb49BV7rFt3Usr-vdUZO8VGOnjRehGHBXkSzchr37SYF2GNdQFYSp72jh5QUhzG
                password:
                  type: string
                  example: >-
                    EMnAWe06ioGtouJs7gLYT9chK9-2jJ--7MKRXpI8FesmY_2Kp-d_7aCqff7M9moEJBvuXoBO4clKtY0v
            encoding:
              grant_type:
                style: form
                explode: false
              username:
                style: form
                explode: false
              password:
                style: form
                explode: false
      security:
        - basicAuth: []
      responses:
        '200':
          description: OK
          headers:
            content-type:
              schema:
                type: string
                example: application/json
            x-request-id:
              schema:
                type: string
                example: b990a0b5-a020-421e-bd1e-54f9325d4904
            Content-Length:
              schema:
                type: integer
                example: '4027'
            date:
              schema:
                type: string
                example: 'Fri, 09 Feb 2024 09:05:25 GMT'
```

! apispec.yaml

```yaml
          via:
            schema:
              type: number
              example: 1.1 google
            Alt-Svc:
              schema:
                type: string
                example: 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000'
        content:
          application/json:
            schema:
              type: object
              example:
                refresh_token_expires_in: '86399'
                refresh_token_status: approved
                api_product_list: '[dkoro3-mdw-resourses, dkoro3-mdw-oauth2-pwd-scopes]'
                api_product_list_json:
                  - dkoro3-mdw-resourses
                  - dkoro3-mdw-oauth2-pwd-scopes
                organization_name: gcp101027-apigeex
                developer.email: dkoro3@softserveinc.com
                token_type: BearerToken
                issued_at: '1707469525819'
                client_id: KOA7SzlsSTzBsYZBI4vbMECoJKGBE1MXCU7ojkgppAGpZGA1
                access_token: C3g6e8X2uXtkflW47H9Nv7puHMPj
                refresh_token: UDCCRI2suI2lPwoAGgOwV6riHLyL43Mz
                application_name: 6eae96f8-2ef1-4549-a365-ea20e54ceb1c
                grant_type: password
                scope: >-
                  https://uri.paypal.com/services/payments/partnerfee
                  https://uri.paypal.com/services/vault/payment-tokens/read
                  https://uri.paypal.com/services/disputes/read-buyer
                refresh_token_issued_at: '1707469525819'
                authenticated_user: >-
                  AUv8rrc_P-EbP2E0mpb49BV7rFt3Usr-vdUZO8VGOnjRehGHBXkSzchr37SYF2GNdQFYSp72jh5QUhzG
                expires_in: '1799'
                refresh_count: '0'
                status: approved
      '400':
        description: Bad Request
        headers:
          content-type:
            schema:
              type: string
              example: application/json
          x-request-id:
            schema:
              type: string
              example: 05d1133c-54e0-4f8f-8726-9c223fcb2bb8
          Content-Length:
            schema:
              type: integer
              example: '77'
          date:
            schema:
              type: string
              example: 'Fri, 09 Feb 2024 09:06:10 GMT'
          via:
            schema:
              type: number
              example: 1.1 google
          Alt-Svc:
            schema:
              type: string
              example: 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000'
```

apispec.yaml

```yaml
        content:
          application/json:
            schema:
              type: object
            example:
              error:
                code: 400
                message: Invalid grant type
      '401':
        description: Unauthorized
        headers:
          content-type:
            schema:
              type: string
              example: application/json
          x-request-id:
            schema:
              type: string
              example: 72207cf6-6c62-483a-91d7-c89e113df96a
          Content-Length:
            schema:
              type: integer
              example: '87'
          date:
            schema:
              type: string
              example: 'Fri, 09 Feb 2024 09:35:47 GMT'
          via:
            schema:
              type: number
              example: 1.1 google
          Alt-Svc:
            schema:
              type: string
              example: 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000'
        content:
          application/json:
            schema:
              type: object
            examples:
              example-0:
                summary: '401'
                value:
                  error:
                    code: 401
                    message: Client Authentication failed
              example-1:
                summary: '401'
                value:
                  error:
                    code: 401
                    message: User Authentication failed
      '429':
        description: Too Many Requests
        headers:
          content-type:
            schema:
              type: string
              example: application/json
          x-request-id:
            schema:
              type: string
              example: 0fb4dcaf-0af4-4819-b373-6ae399b00f6a
          Content-Length:
            schema:
              type: integer
              example: '73'
```

apispec.yaml

```yaml
          example: '73'
        date:
          schema:
            type: string
            example: 'Fri, 09 Feb 2024 09:07:07 GMT'
        via:
          schema:
            type: number
            example: 1.1 google
        Alt-Svc:
          schema:
            type: string
            example: 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000'
      content:
        application/json:
          schema:
            type: object
          examples:
            example-0:
              summary: '429'
              value:
                error:
                  code: 429
                  message: To many request
            example-1:
              summary: '429'
              value:
                error:
                  code: 429
                  message: Rate limit quota violation
```