Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Реалізація системи виявлення помилок з автоматичним контролем задач в Jira на базі Python

Виконав: студент IV курсу, групи СП-42 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Мидзин А. В.
(підпис) (прізвище та ініціали)

Керівник _____ Цебрій О. Р.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Мидзин Арсену Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація системи виявлення помилок з автоматичним контролем задач в Jira на базі Python

Керівник роботи к.ф.-м.н., доц. Цебрій О. Р.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_____» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи Вирішено проблему ручної роботи Manual QA

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступна частина
Аналіз предметної області та теоретичних основ інтеграції Sentry та Asana
Визначення методики реалізації інтеграції
Реалізація інтеграційного рішення
Визначення основних аспектів охорони праці та безпеки життєдіяльності
Висновки роботи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Порівняльні таблиці, ілюстративні зображення, інформативні зображення для доповнення тексту,
скрішоти для візуалізації роботи певних процесів, графіки ефективності.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи		
2	Пошук та підбір наукових і технічних джерел по темі		
3	Аналіз та систематизація інформації з обраних джерел		
4	Проектування архітектури програмного рішення		
5	Вивчення і освоєння необхідних нових технологій		
6	Розробка інтеграції відповідно до технічного завдання		
7	Тестування та налагодження		
8	Підготовка проміжного звіту про виконану роботу		
9	Виконання завдання з розділу «Безпека життєдіяльності»		
9	Виконання завдання до підрозділу «Основи охорони праці»		
10	Оформлення кваліфікаційної роботи		
11	Проведення нормо контролю		
12	Перевірка тексту на плагіат		
13	Попередній захист та внесення коригувань		
14	Захист кваліфікаційної роботи		

Студент _____ Мидзин А. М.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Цебрій О. Р.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтеграційного рішення між сервісами Asana та Sentry з метою оптимізації внутрішніх процесів компанії // Кваліфікаційна робота освітнього рівня “Бакалавр” // Мидзин Арсен Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2025 // Ст. – 78, рис. – 21, табл. – 3, бібліогр. – 20.

Основна мета дослідження полягає у створенні прототипу інноваційної інтеграції, що дозволяє автоматично створювати та закривати завдання в системі керування проєктами Asana на основі подій, зафіксованих у системі виявлення помилок Sentry. Використання такої інтеграції дозволяє підвищити продуктивність команд розробки, зменшити час реагування на помилки та мінімізувати вплив людського фактора у процесі роботи із завданнями.

У розробці інтеграції використано сучасні вебтехнології та REST API обох платформ. Реалізовано механізм обробки вебхуків із Sentry для створення задач в Asana за заданими умовами, а також систему автоматичного закриття задач після виправлення помилок. Прототип підтримує фільтрацію івентів за рівнем важливості та релізами, що дозволяє адаптувати його до потреб конкретного проєкту.

Робота підкреслює значення автоматизації процесів у сучасному розробницькому циклі та демонструє приклад ефективної взаємодії інструментів моніторингу та управління проєктами. Розроблене рішення має потенціал для масштабування та інтеграції з іншими сервісами, роблячи внесок у підвищення продуктивності та якості програмного забезпечення.

Ключові слова роботи: інтеграція, Sentry, Asana, автоматизація задач, моніторинг помилок, REST API, вебхуки, DevOps.

ABSTRACT

The Bachelor's Qualification Thesis, completed by Arsen Viktorovych Mydzyn, a student of group SP-42 at Ternopil Ivan Puluj National Technical University, focuses on designing an integration solution between Asana and Sentry aimed at optimizing internal business processes through the automation of previously manual tasks.

The primary objective of the research is to develop a prototype of an innovative integration capable of automatically creating and closing tasks in the Asana project management system based on events generated by the Sentry error monitoring platform.The use of such an integration significantly enhances the efficiency of development teams, reduces response time to errors, and minimizes human involvement in task management.

The integration was developed using modern web technologies and the REST APIs of both platforms. A mechanism was implemented to process webhooks from Sentry for task creation in Asana under specified conditions, as well as a system for automatically closing tasks once the errors are resolved. The prototype supports filtering of events by severity and release version, allowing it to be tailored to the needs of a specific project. Testing demonstrated stable performance of the solution and confirmed its viability for deployment in real-world development environments.

This thesis highlights the importance of automation in modern development cycles and provides an example of effective interaction between monitoring and project management tools. The developed solution has potential for scaling and integration with other services, contributing to improved productivity and software quality.

Keywords: integration, Sentry, Asana, task automation, error monitoring, REST API, webhooks, DevOps.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

URL – Uniform Resource Locator

UI – User Interface

UX – User Experience

SaaS – Software as a Service

SDK – Software Development Kit

JSON – JavaScript Object Notation

CI/CD – Continuous Integration / Continuous Deployment

JWT – JSON Web Token

IDE – Integrated Development Environment

Sentry – Система моніторингу помилок

Asana – Система управління проєктами

CRUD – Create, Read, Update, Delete

DevOps – Development & Operations

REST – Representational State Transfer

SPA – Single Page Application

CLI – Command Line Interface

OAuth – Open Authorization

VPN – Virtual Private Network

ВСТУП

У сучасних умовах розробки програмного забезпечення, де оперативне виявлення та усунення помилок має вирішальне значення, автоматизація процесів виступає важливим фактором підвищення ефективності. Зростає потреба у впровадженні інтеграційних рішень, які об’єднують різноманітні сервіси в єдину функціональну систему. Одним із прикладів такої інтеграції є зв’язок між системою виявлення та моніторингу помилок Sentry і платформою керування проєктами Asana, що дає змогу автоматично створювати завдання на основі зафіксованих помилок та закривати їх після усунення.

У процесі розробки ПЗ Sentry виконує роль системи, яка виявляє, фіксує та класифікує помилки в реальному часі. У свою чергу, Asana є потужним інструментом для організації задач і координації командної роботи. Поєднання цих двох сервісів дозволяє усунути необхідність ручного переносу помилок із системи моніторингу у трекер завдань, тим самим мінімізуючи людський фактор, пришвидшуючи обробку інцидентів та покращуючи комунікацію між командами.

Застосування сучасних вебтехнологій та REST API відкриває широкі можливості для реалізації подібної інтеграції. Автоматичне створення задач у Asana на основі івентів із Sentry дає змогу команді розробників оперативно реагувати на помилки, не витрачаючи час на ручні дії. Окрім цього, реалізація механізму автоматичного закриття задач після фіксу помилки сприяє підтримці актуальності інформації у системі задач та знижує ризики дублювання чи забутих багів.

Автоматизація процесу обробки помилок також позитивно впливає на продуктивність команд, дозволяючи зосередитись на вирішенні технічних завдань, а не на рутинній адміністративній роботі. Таке рішення особливо цінне для стартапів, невеликих команд або великих компаній із великою кількістю мікросервісів, де обробка інцидентів відбувається щодня.

Метою даної роботи є розробка інтеграційного прототипу, який дозволяє автоматично створювати задачі у системі Asana на основі івентів, отриманих із Sentry, а також закривати ці задачі після того, як помилка була виправлена у коді. Для досягнення цієї мети необхідно проаналізувати принципи роботи обох систем, вивчити можливості їх API, розробити архітектуру інтеграції, реалізувати логіку обробки подій та протестувати рішення у реальних умовах.

Етапи дослідження включають: огляд існуючих підходів до інтеграції подібних систем, вибір оптимальної технології для реалізації зв'язку між Sentry та Asana, розробку алгоритмів обробки подій, реалізацію інтеграції та її тестування. Завершальним етапом стане оцінка ефективності запропонованого рішення та формулювання рекомендацій щодо його практичного використання у щоденній роботі команд розробки.

У світі, де швидкість випуску оновлень та якість продукту йдуть пліч-о-пліч, автоматизована обробка помилок є не просто зручністю, а необхідністю. Розроблена інтеграція дозволяє не лише зменшити навантаження на менеджерів проєктів та розробників, а й створити ефективну систему обміну інформацією між сервісами, що підвищує прозорість і контроль за процесами у команді.

1.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНИХ ОСНОВ ІНТЕГРАЦІЇ SENTRY

1.1 Огляд системи моніторингу помилок Sentry

Sentry — це потужна система моніторингу помилок, яка допомагає розробникам оперативно виявляти, діагностувати та виправляти помилки в реальному часі. Вона підтримує широкий спектр мов програмування, включаючи JavaScript, Python, Java, PHP, .NET, Ruby, а також популярні фреймворки на кшталт React, Django, Laravel тощо.

Головна перевага Sentry — це гнучкість і простота у використанні. Інтеграція в проект зазвичай займає лише кілька хвилин, після чого сервіс починає збирати інформацію про всі винятки, помилки та неочікувану поведінку системи. Sentry автоматично групує схожі помилки, додає стек трейс, контекст виконання, параметри запиту, дані користувача тощо — усе, що потрібно для швидкого відтворення проблеми [1].

Інтерфейс Sentry є зручним та інтуїтивно зрозумілим. Він дозволяє фільтрувати помилки за середовищем (продакшн, дев, staging), статусом (нові, фіксовані), сервісами тощо. Також можна налаштувати оповіщення (email, Slack, Teams), щоб миттєво дізнаватися про критичні помилки.

Sentry допомагає скоротити час реагування на помилки та підвищує якість продукту. Завдяки відкритому коду (open-source) і активній спільноті, його легко кастомізувати під власні потреби.

1.1.1 Детальне знайомство з Sentry

Sentry — це не просто сервіс для збору помилок, а потужна екосистема для моніторингу життєвого циклу додатка в продакшені. Його мета — надати розробникам усі необхідні інструменти для розуміння, аналізу й усунення проблем, які впливають на досвід користувача. Sentry функціонує як SaaS-рішення або може бути розгорнуте локально, що особливо важливо для enterprise-команд з підвищеними вимогами до безпеки [2].

Однією з найбільш потужних функцій є підтримка дистрибутивного трасування (distributed tracing). Це означає, що при виконанні запиту через кілька мікросервісів або частин архітектури, Sentry вміє зібрати повну картину — від фронтенду до бекенду. Кожна подія містить trace ID, що дозволяє візуалізувати послідовність дій, тривалість виконання кожного етапу, вузькі місця і фатальні помилки. Це особливо важливо для команд, які використовують Kubernetes, serverless-функції або мікросервісну архітектуру.

Унікальна особливість Sentry — надання контексту навколо події. Крім стек-трейсу, до кожної помилки автоматично додаються:

1) environment variables
2) зміст HTTP-запиту/відповіді
3) стан користувача в сесії
4) значення локальних змінних
5) брейкпоінти та рівні логування

Це дозволяє розробникам бачити ситуацію очима машини, що суттєво знижує час на локалізацію помилки. У багатьох інших сервісах ці деталі доступні лише через додаткове налаштування.

Sentry побудовано з урахуванням спільної розробки. Події можна призначати відповідальним, залишати внутрішні коментарі, обговорювати стратегії виправлення.

Можна налаштувати правила автоматичної маршрутизації подій, наприклад: всі помилки з продакшну в payments-service йдуть одразу до відповідальної команди.

У Sentry можна створювати складні правила оповіщень з фільтрацією за типом помилки, середовищем, тегами, частотою повторення. Наприклад:

— "Надіслати повідомлення в Slack, якщо помилка повторилася >10 разів за 5 хвилин".

— "Створити завдання в Jira, якщо помилка виникла у версії релізу v2.3.0".

Також підтримуються rate limits на алерти, щоб уникнути спаму. Це значно ефективніше, ніж базові email-повідомлення, що характерні для більшості альтернатив.

Sentry підтримує показники перформансу на фронтенді, зокрема:

— час до появи першого вмісту на екрані

— момент, коли завантажується найбільший елемент сторінки

— період між дією користувача та її візуальним відображенням

— і час, необхідний для повної інтерактивності сторінки

А ще він дозволяє відстежувати JS помилки, які не спричиняють креш, але погіршують UX (наприклад, клацання по недоступній кнопці або невірний контент через неправильний state). Це дає можливість підняти якість UI та мінімізувати “мовчазні” баги [3].

Із Sentry зручно відслідковувати зв’язок між кодом і помилками. Підтримується інтеграція з GitHub, GitLab, Bitbucket — можна:

— автоматично лінкувати помилку до коміту

— бачити, хто останнім редагував код

— створювати issue в трекері одним кліком

— отримувати “suspect commits” — які рядки ймовірно викликали баг

Ця функція корисна в CI/CD-пайплайнах — можна, наприклад, зупинити реліз, якщо в новій версії зафіксовано надмірну кількість помилок.

Sentry має відкриту структуру та API, які дозволяють:

1) створювати власні інтеграції з будь-яким інструментом

2) додавати власні теги, контекст, метрики

3) використовувати Webhooks для з’єднання з внутрішніми сервісами компанії

Також є SDK для створення обгорток під нестандартні технології, що дозволяє кастомізувати Sentry під специфічні продукти.

Одна з найновіших фіч — Session Replay. Вона дозволяє "переглядати" дії користувача до того моменту, як виникла помилка. Це як запис відео з кліками, переходами, формами — але з повним контролем над приватністю. Можна бачити, що бачив користувач і як взаємодіяв із системою. Це особливо цінно при дослідженні UX-проблем, що не викликають явних помилок (табл. 1.1).

Таблиця 1.1 - Порівняння з конкурентами

Функція	Sentry	Bugsnag	Rollbar
Distributed tracing	✓	✗	✗
Frontend performance	✓	частково	✗
Session Replay	✓	(обмежено)	✗
Open-source	✓	✗	(обмежено)

Sentry — це не просто моніторинг помилок. Це комплексний інструмент стабільності, що дозволяє отримати повну картину роботи додатка в реальному часі. Завдяки своїй гнучкості, масштабованості та активній спільноті, Sentry стає основним вибором як для стартапів, так і для enterprise-команд. Якщо ви цінуєте швидке реагування, точну діагностику та безперервне покращення продукту — Sentry стане вашим незамінним союзником [4].

Крім відстеження помилок і виключень, Sentry також збирає статистичні дані

про продуктивність застосунку. Це дає змогу розробникам аналізувати поведінку системи, ідентифікувати «вузькі місця» та покращувати загальну ефективність роботи програмного забезпечення. Зібрана статистика включає час відповіді, частоту викликів певних функцій, кількість помилок на одиницю часу та інші ключові показники продуктивності (рис. 1.1).

Рисунок 1.1- Приклад статистики Sentry

1.1.2 API документація Sentry

API (інтерфейс прикладного програмування) забезпечує можливість взаємодії між програмами шляхом обміну запитами та відповідями. У випадку Sentry, цей інтерфейс відкриває доступ до ключових функцій системи моніторингу помилок, дозволяючи інтегрувати її з іншими сервісами, автоматизувати робочі процеси та використовувати можливості платформи без необхідності роботи через веб інтерфейс.

API Sentry надає розширену функціональність для взаємодії з системою помилок, що дозволяє розробникам автоматично обробляти дані, отримувати важливі

метрики, а також інтегрувати Sentry з іншими інструментами для оптимізації процесу моніторингу і виправлення помилок [5].

Отримання даних. Однією з основних функцій API є можливість отримувати дані, що зберігаються в Sentry. Це включає в себе дані про помилки, проєкти, релізи та інші аспекти системи. API дає змогу виконувати запити на отримання різноманітних даних, які можуть бути використані для аналізу стану проєктів або для подальшої обробки в інших інструментах. Наприклад, можна отримати всі помилки, які сталися протягом певного часу, або всі події, пов'язані з конкретним релізом.

Одним з основних типів запитів є GET запити, які дозволяють отримувати інформацію. Наприклад, ви можете отримати список всіх проєктів вашої організації через API запит, що дозволяє легко інтегрувати цю інформацію в інші бізнес-процеси або автоматизувати завдання.

Детальніше, API підтримує кілька типів запитів для отримання даних, таких як:

1) Запит на отримання детальної інформації про помилку.
2) Отримання загальної статистики по проєктах.
3) Аналіз груп помилок або окремих подій.
4) Створення і оновлення даних

API Sentry дозволяє не лише отримувати інформацію, але й створювати нові ресурси або оновлювати існуючі. Це може бути корисним для автоматизації процесів на етапі інтеграції з іншими системами або для оновлення даних без необхідності використовувати інтерфейс користувача [6].

За допомогою POST запитів можна створювати нові проєкти, релізи або повідомлення про помилки. Наприклад, якщо ви хочете додати новий реліз до проєкту, API дає можливість зробити це безпосередньо через програмний інтерфейс.

Що стосується PUT або PATCH запитів, вони дозволяють змінювати існуючі дані, наприклад, оновлювати статус помилки або змінювати метки на конкретній події.

Такі операції часто використовуються при автоматизації тестування або CI/CD

процесів, коли потрібно постійно оновлювати або взаємодіяти з даними на основі змін у програмному коді або оточенні.

Однією з найбільших переваг використання API є здатність інтегрувати Sentry з іншими інструментами. API дозволяє зв'язати Sentry з іншими системами для покращення моніторингу, реагування на помилки та управління релізами. Наприклад, ви можете інтегрувати Sentry з платформами, такими як Jira, Slack або GitHub, щоб автоматично створювати завдання або сповіщення на основі помилок, які виникають у вашій програмі (рис. 1.2).

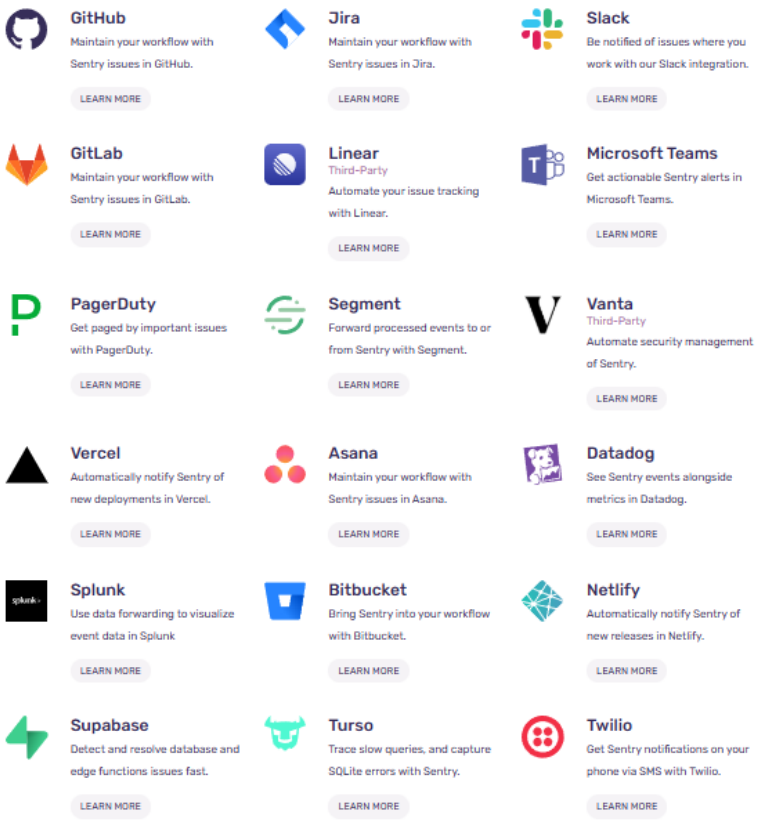

Рисунок 1.2- Частина сервісів з якими може інтегруватись сентрі

Інтеграція з CI/CD пайплайнами дозволяє автоматично оновлювати статуси помилок або релізів після того, як код деплоється в різні середовища. Таке автоматичне оновлення статусів або сповіщень може значно підвищити ефективність команди і скоротити час реагування на критичні помилки [7].

Інтеграція з іншими інструментами дозволяє створювати єдину автоматизовану систему, де кожен інструмент працює на максимальну ефективність, передаючи дані через API, що забезпечує взаємодію між системами без необхідності виконувати зайву роботу вручну.

API Sentry побудоване таким чином, щоб підтримувати ефективну роботу з великими обсягами даних. Враховуючи, що помилки та події можуть генеруватися у великих обсягах, API підтримує пагінацію запитів, щоб зменшити навантаження на систему і полегшити обробку великих наборів даних. Це означає, що ви можете отримувати дані порціями, не перевантажуючи систему.

Пагінація дозволяє налаштувати кількість елементів на сторінку (через параметр per_page), а також здійснювати запити з використанням параметра cursor, щоб отримувати наступні порції даних. Це особливо корисно при роботі з великими обсягами помилок або подій, коли потрібно обробити всі дані по черзі.

Крім того, API дає можливість фільтрувати дані, що дозволяє вам отримувати тільки необхідну інформацію, зменшуючи кількість непотрібних запитів і покращуючи продуктивність системи.

API Sentry використовує RESTful архітектуру, що означає, що всі запити здійснюються через стандартні HTTP методи. Це дозволяє зручно інтегрувати API з іншими системами, а також використовувати стандартні інструменти для взаємодії з платформою. Основні HTTP методи, що використовуються в API Sentry, включають GET, POST, PUT, PATCH та DELETE (рис. 1.3).

За допомогою методу GET можна здійснювати запити на отримання інформації з системи Sentry, наприклад, список подій чи деталей помилки. Наприклад, при запиті на отримання списку всіх проєктів вашої організації, API віддасть список проєктів у форматі JSON, що зручно для подальшого оброблення. Оскільки GET запити призначені для отримання даних, вони не змінюють стан сервера.

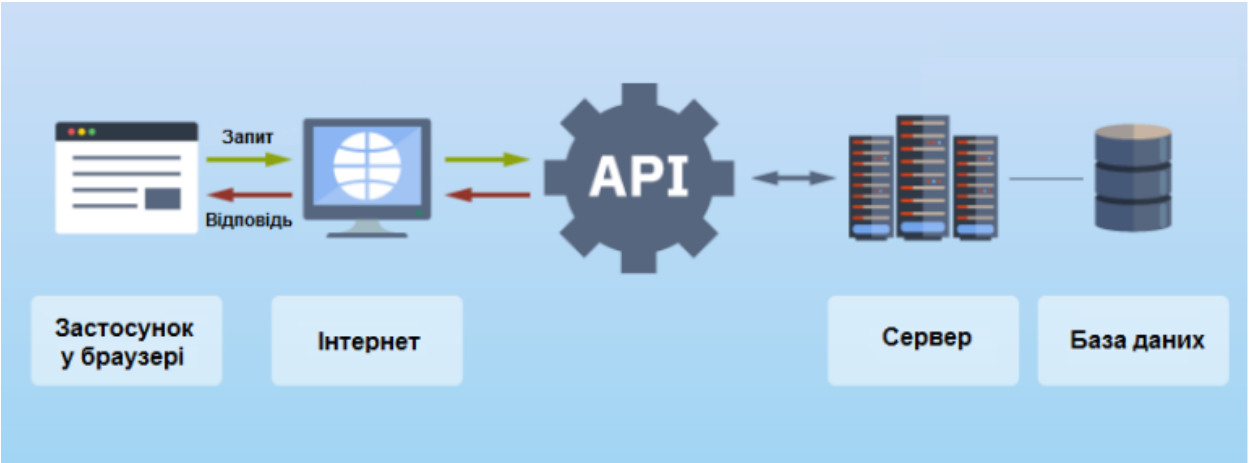

Рисунок 1.3- Принцип роботи API

Метод POST дозволяє додавати нові об'єкти в систему, зокрема створювати проєкти, релізи чи повідомлення про помилки через API-запити. Наприклад, якщо потрібно додати нову групу помилок або створити новий проєкт у Sentry, це можна зробити за допомогою POST запиту.

Для зміни вже наявних ресурсів застосовуються методи PUT і PATCH. Перший зазвичай замінює весь вміст ресурсу, тоді як другий дозволяє оновити лише окремі атрибути. Наприклад, можна оновити інформацію про помилку або змінити метки для певної події.

Метод DELETE призначений для видалення даних, таких як окремі помилки або події з системи.

Використання API Sentry надає чимало переваг, які можуть суттєво покращити роботу команди розробників та ефективність процесів моніторингу помилок.

Завдяки API можна автоматизувати типові завдання: від відстеження помилок до створення задач на їх усунення чи інтеграції з сервісами оповіщення. Такий підхід мінімізує ручну працю і підвищує продуктивність розробницької команди [8].

Інтерфейс API забезпечує гнучку інтеграцію Sentry з зовнішніми системами, що є суттєвим для команд, які працюють з інструментами управління проєктами, CI/CD та внутрішньої комунікації.

Завдяки можливості обробляти значні об’єми інформації, API Sentry ефективно працює в умовах масштабних проєктів і великих команд, що щоденно взаємодіють з великою кількістю помилок та подій.

API Sentry — це потужний інструмент для програмного доступу до даних та ресурсів Sentry. Цей API сприяє автоматизації процесів, спрощує інтеграцію з зовнішніми платформами та оптимізує управління інформацією. Його використання дозволяє покращити контроль над помилками та загальну продуктивність розробницького циклу [9].

1.1.3 Ознайомлення з Sentry Webhook

Webhook — це спосіб налаштування автоматичної передачі даних між системами в момент, коли в одній із них відбувається визначена подія. У такому випадку система ініціює HTTP-запит і надсилає набір даних (payload) іншій стороні. Це відрізняється від традиційних API запитів, оскільки з вебхуками немає необхідності кожен раз запитувати дані вручну. Замість цього, система сама ініціює передачу інформації, коли відбувається подія, що вас цікавить [10].

Вебхуки є потужним інструментом для інтеграції різних програмних продуктів і автоматизації процесів. Вони дозволяють зберігати постійний потік даних між системами, скорочуючи необхідність на постійні запити для отримання нової інформації. У Sentry вебхуки використовуються для оперативного надсилання інформації про події, збої чи зміни до зовнішніх платформ. Це дозволяє налаштувати швидку реакцію на інциденти, автоматизувати частину процесів та інтегрувати Sentry з іншими сервісами, наприклад, інструментами для управління проєктами або обміну повідомленнями.

Вебхуки в Sentry використовуються для того, щоб передавати дані про події,

помилки, оновлення статусу та інші зміни в системах або додатках без необхідності запитувати ці дані вручну. Завдяки вебхукам, Sentry може інтегруватися з іншими інструментами або системами, такими як системи керування завданнями (наприклад, Jira або Trello), комунікаційні платформи (Slack, Microsoft Teams) або інші сервіси, де ці дані можуть бути оброблені або використані для подальших дій.

Основною перевагою використання вебхуків є те, що вони дозволяють миттєво реагувати на події. Якщо, наприклад, виявляється критична помилка у вашому додатку, вебхук дозволяє одразу надіслати сповіщення у Slack або створити задачу в Jira без затримок, що дає можливість вашій команді швидко зреагувати на проблему.

Вебхуки в Sentry часто використовуються для автоматизації створення нових завдань або тикетів, відправки сповіщень до чатів і месенджерів, або для передачі метрик і даних про помилки в інші аналітичні інструменти, з якими працює ваша команда. Це не лише спрощує процеси, але й дозволяє зберігати інформацію актуальною у реальному часі [11].

Коли відбувається подія в Sentry (наприклад, нова помилка або зміна статусу помилки), система може автоматично надіслати запит на вказану URL-адресу через HTTP POST. Цей запит містить дані про подію, в тому числі деталі помилки, метки, інформацію про користувача, що спричинив помилку, і багато іншого. Вебхук працює на основі налаштувань, визначених вами, і він забезпечує автоматичне сповіщення зовнішніх систем без необхідності додаткових запитів.

У випадку, якщо ви налаштовуєте вебхук у Sentry, ви вказуєте URL, на який хочете отримувати дані, та визначаєте тип подій, про які вам потрібно отримувати сповіщення. Потім, кожного разу, коли на платформі відбувається одна з цих подій, вебхук надсилає запит на зазначену вами URL-адресу. Вебхук може передавати деталі помилок, інформацію про релізи, дані про користувачів або навіть метки помилок.

Цей процес автоматизує сповіщення про нові помилки та зміни, що дуже зручно для команд, які повинні оперативно реагувати на інциденти, не чекаючи на традиційні сповіщення або не витрачаючи час на періодичні перевірки.

Управління вебхуками в Sentry дуже зручне й інтуїтивно зрозуміле. Кожен користувач, який має необхідні права доступу, може налаштувати вебхуки через налаштування організації. У розділі налаштувань вебхуків ви можете:

1) Вказати URL-адресу, на яку буде надсилатися запит.

2) Вибрати типи подій, які повинні спричинити надсилання даних (наприклад, створення нової помилки, зміна статусу помилки, новий реліз тощо).

3) Налаштувати фільтри для уточнення, які події вас цікавлять (наприклад, можна вибрати конкретні проєкти або метки).

Крім того, є можливість налаштувати декілька вебхуків для різних сценаріїв. Це означає, що ви можете одночасно відправляти дані в різні інструменти — наприклад, сповіщення у Slack для миттєвого реагування на помилки та повідомлення в Jira для автоматичного створення задач (рис. 1.4).

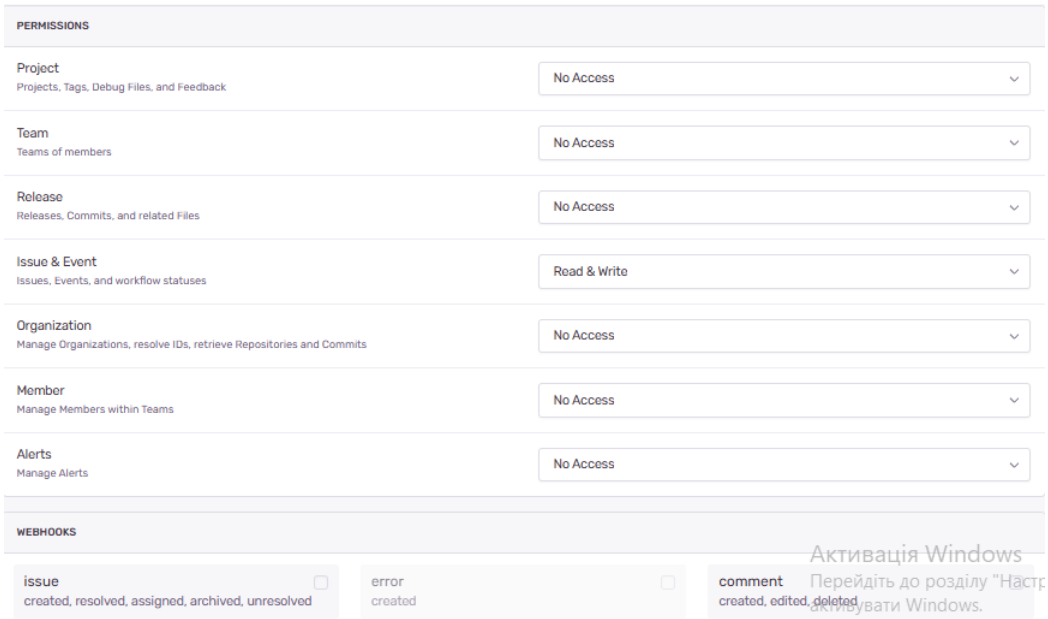

Рисунок 1.4- Налаштування Sentry Webhook

Одним з важливих аспектів є те, що ви можете контролювати, які саме події або зміни у Sentry повинні ініціювати виклик вебхука. Це дозволяє фільтрувати тільки найбільш важливі події і уникати надмірної кількості сповіщень або запитів [12].

Вебхуки в Sentry мають широкі можливості для налаштування і використання. Нижче наведено ключові можливості, завдяки яким вебхуки стають ефективним засобом для автоматизованого моніторингу та обробки помилок.

1) Автоматичне сповіщення про критичні помилки. У випадку критичної помилки вебхук здатен негайно передати повідомлення в зовнішню систему, таку як Slack або Microsoft Teams, забезпечуючи своєчасне інформування команди.

2) Інтеграція з системами управління задачами. Вебхуки можна використовувати для створення нових тикетів в системах, таких як Jira або Trello, що допомагає автоматизувати процес відстеження помилок і завдань. Наприклад, якщо в Sentry з’являється нова помилка, вебхук може одразу створити завдання для її виправлення.

3) Перехід від події до дії. Кожного разу, коли виникає подія в Sentry, вебхук автоматично надсилає відповідні дані, і це може спонукати до негайної дії. Такий підхід особливо корисний для команд, що орієнтуються на оперативну реакцію на інциденти й підтримання надійності програмних рішень.

4) Відправка метрик та статистики в інші аналітичні інструменти. Вебхуки також можуть бути корисні для збору статистики та аналітики. Дані, що надійшли через вебхук, можна передавати в інші системи для аналізу і виведення більш детальних звітів.

1.2 Огляд системи управління завданнями Asana

У сучасному динамічному світі ефективне управління завданнями є ключовим елементом успішного функціонування будь-якої команди або організації. Однією з популярних систем, що допомагає впорядкувати робочі процеси, є Asana. Це онлайн-платформа для управління проектами, яка дозволяє командам ефективно

співпрацювати, контролювати хід виконання завдань та досягати поставлених цілей.

У Asana користувачі можуть створювати задачі, закріплювати їх за виконавцями, додавати дедлайни, коментарі, прикріплювати файли та створювати вкладені підзавдання. Вона підтримує різні формати відображення задач: список, дошку (канбан), календар та часову шкалу (таймлайн), що дозволяє адаптувати платформу до стилю роботи конкретної команди [13].

Asana вирізняється простим у використанні інтерфейсом та підтримкою інтеграцій з такими сервісами, як Slack, Zoom, Google Workspace чи Microsoft Teams. Автоматизація стандартних дій дозволяє економити час і зменшити кількість помилок. Платформа зручна як для малих команд, так і для великих компаній з численними паралельними проєктами.

Таким чином, Asana є універсальним інструментом для керування роботою, який дозволяє зосередитися не на самому контролі, а на досягненні результатів. Якщо ваша команда шукає спосіб покращити організацію праці, зменшити кількість пропущених дедлайнів і підвищити прозорість процесів — варто звернути увагу на Asana.

1.2.1 Огляд системи управління завданнями Asana

Управління завданнями: сучасний виклик для команд

Управління завданнями стало критично важливою складовою будь-якого проекту — незалежно від галузі чи розміру команди. У світі, де дедлайни стискаються, а вимоги клієнтів зростають, компаніям потрібні інструменти, які допомагають зберігати фокус, структуру та прозорість. Саме для цього існують системи управління завданнями — платформи, що дозволяють ставити задачі, відстежувати їх виконання, координувати роботу команди та аналізувати результати.

Існує багато рішень у цій сфері: Trello, Jira, ClickUp, Monday.com тощо. Та коли мова йде про універсальність, простоту використання, багатофункціональність і гнучкість, Asana часто займає одну з лідерських позицій. Але чому саме вона? Розглянемо детальніше (рис. 1.5, 1.6).

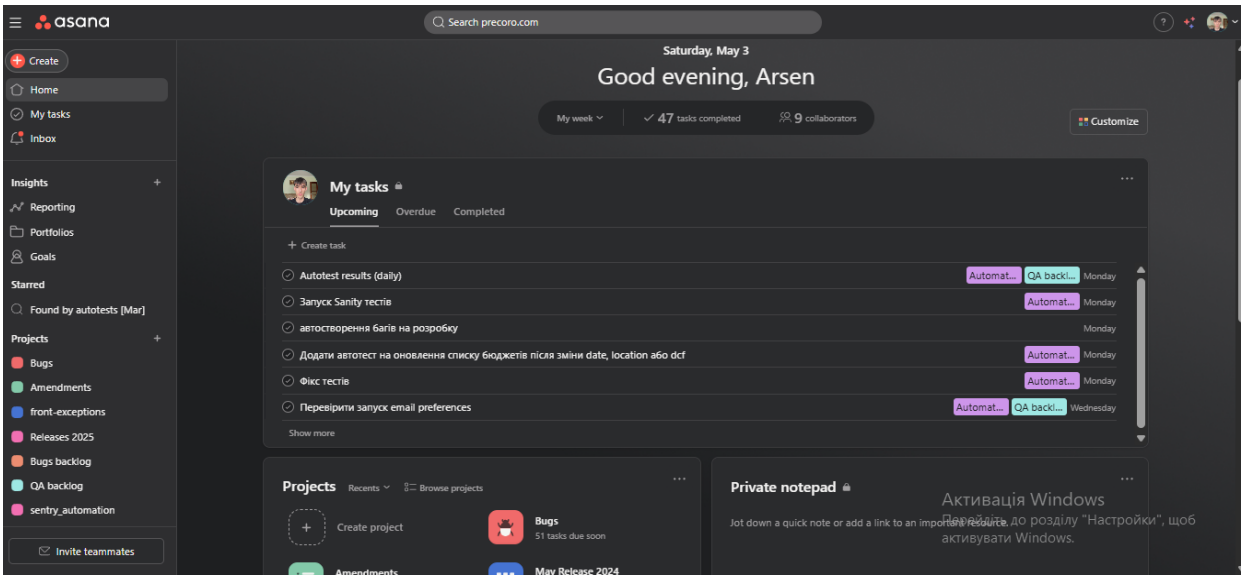

Рисунок 1.5 - Інтерфейс асани

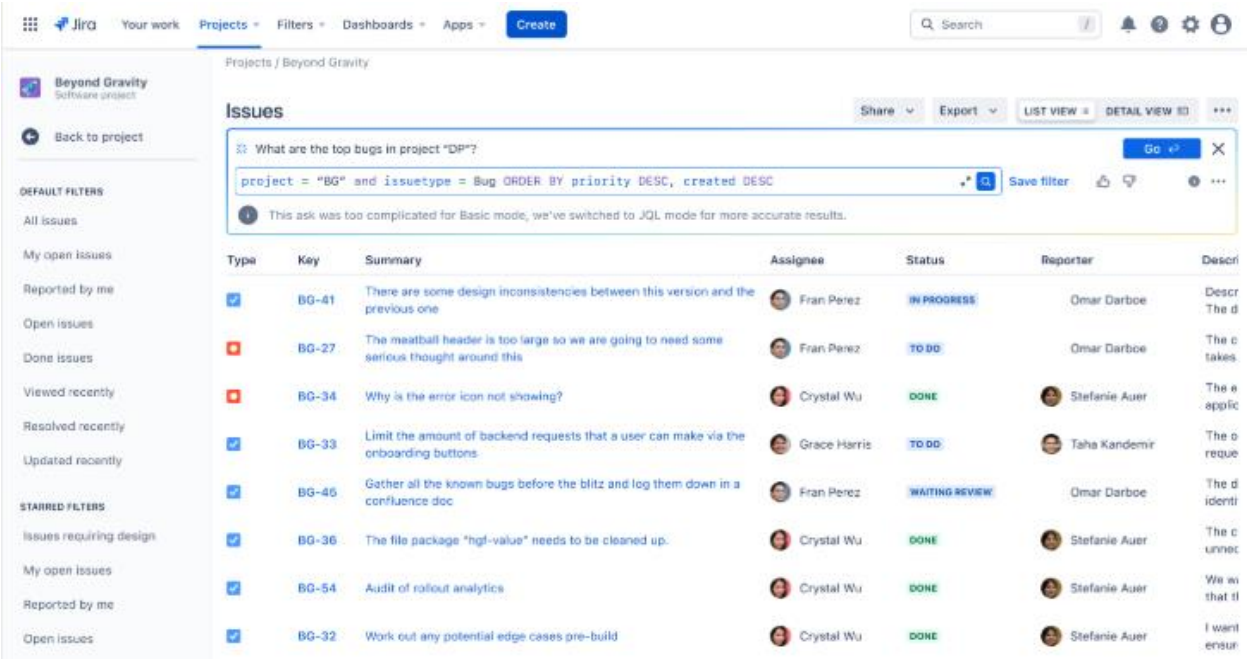

Рисунок 1.6 - Інтерфейс Jira

Asana — це сервіс управління завданнями на основі хмарних технологій, заснований у 2008 році одним із співзасновників Facebook — Дастіном Московіцем. Основна мета Asana — надати користувачам простий спосіб бачити, над чим працює команда, хто за що відповідає і до яких результатів це веде [14].

Платформа побудована за принципом: «Менше хаосу — більше фокусу». Вона дозволяє структурувати як прості завдання (наприклад, написати лист клієнту), так і складні багаторівневі проєкти (впровадження нового продукту на ринок).

У центрі всього — завдання. Кожне з них можна:

— призначити конкретній людині;

— додати термін виконання;

— прикріпити файли та коментарі;

— створити підзавдання;

— налаштувати залежності між завданнями;

— позначити пріоритет.

Завдання об’єднуються в проєкти, які можуть мати кілька варіантів відображення: список, дошка (канбан), календар або таймлайн (діаграма Ганта). Це дозволяє кожному члену команди бачити роботу в зручному для себе вигляді.

Asana підтримує розділення на робочі простори (наприклад, відділи компанії), у межах яких формуються команди, проєкти та завдання. Така ієрархія дозволяє структурувати всю діяльність компанії без плутанини.

Однією з сильних сторін Asana є її здатність інтегруватися з десятками інших сервісів: Slack, Google Calendar, Dropbox, Zoom, GitHub, Salesforce, Microsoft Teams тощо. Таким чином, дані безперервно передаються між платформами, створюючи єдину взаємодіючу цифрову систему [15].

Завдяки функції «Rules» можна створити прості автоматичні дії: наприклад, якщо завдання завершено, воно автоматично змінює статус або відправляє повідомлення в Slack. Це особливо корисно в проектах з повторюваними сценаріями.

Asana надає інструменти аналітики, які дозволяють створювати звіти про

виконання завдань, ефективність членів команди. Це корисно не лише для менеджерів, а й для самих виконавців (рис. 1.7, 1.8).

Рисунок 1.7 - Приклад звіту Asana

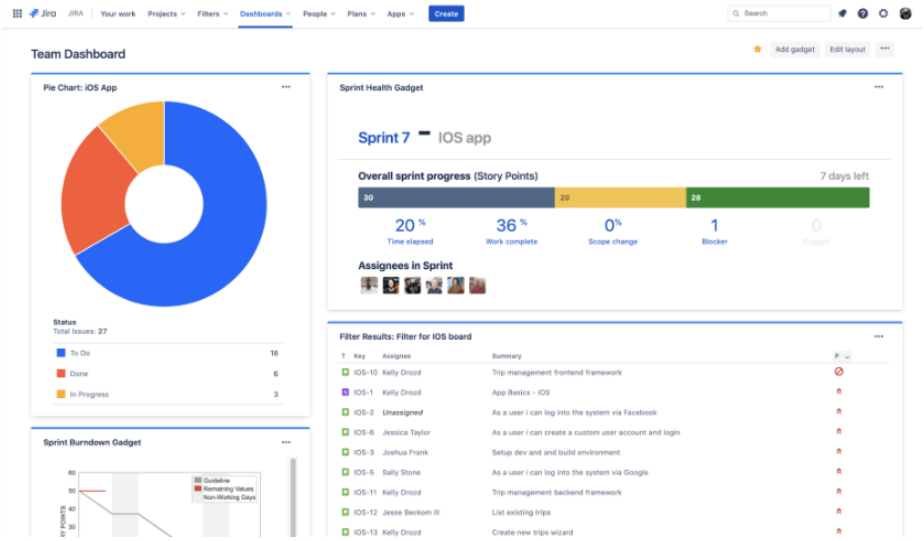

Рисунок 1.8 - Приклад звіту Jira

Asana має інтуїтивно зрозумілий, «чистий» інтерфейс. Навіть новий користувач з мінімальним досвідом у таск-менеджерах швидко розбереться, як усе працює.

Система однаково добре підходить для маркетингової агенції, IT-компанії, HR-відділу чи освітнього проєкту. Її можна налаштувати «під себе» без написання коду чи складних конфігурацій [16].

Asana працює однаково стабільно як для маленької команди з 5 осіб, так і для

міжнародної корпорації з кількома тисячами співробітників.

Платформа не просто дозволяє ставити завдання — вона змінює культуру роботи. Ви завжди знаєте, хто над чим працює, які задачі на черзі, які вже виконані — і де саме може бути вузьке місце [17].

Asana підійде для:

— Стартапи, які хочуть масштабувати проекти без втрати контролю.

— Маркетингові агенції з великою кількістю паралельних кампаній.

— IT-команди для керування беклогом, релізами, тестуванням.

— HR-відділи для онбордингу, підбору персоналу та внутрішніх процесів.

— Освітні проєкти, які хочуть структурувати навчальні програми.

Asana виходить за межі звичайного планувальника завдань, пропонуючи широкий функціонал для командної роботи. Це цифровий помічник, що змінює спосіб роботи команди. Вона дозволяє не лише організувати завдання, а й перетворити хаос у структурований процес, де кожен знає свою роль і бачить загальну мету.

Якщо вам важлива прозорість, командна взаємодія, швидкість виконання та якість проєктного менеджменту — Asana варта вашої уваги. Вона не вимагає технічних знань, не потребує складного навчання і вже з першого дня роботи починає економити вам час (табл. 1.2).

Таблиця 1.2 - Порівняння з конкурентами

Характеристика	Asana	Trello	ClickUp	Jira
Тип інструменту	Управління проєктами	Канбан-дошка	Комплексне управління проєктами	Управління проєктами (особливо для розробки ПЗ)

Продовження таблиці 1.2

Призначення	Командна співпраця, задачі	Візуалізація задач	Універсальний інструмент	Розробка ПЗ, гнучке управління
Інтерфейс	Сучасний, гнучкий, список/таймлайн	Простий, канбан-стиль	Багатофункціональний, але складніший	Орієнтований на технічних користувачів
Можливість інтеграції	Slack, Google Drive, Teams, Zapier	Slack, Google Drive	Slack, Google Drive, GitHub, Zoom	GitHub, Bitbucket, Confluence
Автоматизація	Є (запуски, правила, шаблони)	Обмежена	Потужна (залежності, автоматизація задач)	Потужна (в тому числі кастомні скрипти)
Звіти та аналітика	Є (дашборди, діаграми)	Обмежені	Потужні	Дуже потужні, технічна деталізація
Ціна	Безкоштовно / від $10,99/кор.	Безкоштовно / від $5/кор.	Безкоштовно / від $7/кор.	Безкоштовно / від $7,75/кор.
Платформи	Web, Android, iOS	Web, Android, iOS	Web, Android, iOS	Web, Android, iOS

1.2.2 API документація Asana

Asana — це платформа для керування завданнями, яка здобула визнання завдяки зручному інтерфейсу, широким можливостям налаштування та адаптивному підходу до організації роботи. Проте реальна сила платформи розкривається, коли користувачі інтегрують її в загальний цифровий ландшафт організації за допомогою API (Application Programming Interface). За допомогою API Asana можна істотно розширити функціональність платформи, зокрема через автоматизацію процесів та створення власних інтеграцій.

Asana надає RESTful API, що дозволяє програмно взаємодіяти з усіма основними об’єктами системи, включаючи завдання (tasks), проєкти (projects), користувачів (users), коментарі (stories), команди (teams), робочі простори (workspaces) тощо [18].

Функціонал API охоплює такі операції:

— Створення, редагування та видалення завдань;

— Призначення виконавців і встановлення термінів;

— Автоматичне створення підпроєктів;

— Відстеження змін у завданнях та проєктах;

— Отримання інформації про прогрес і статус виконання;

— Інтеграція з внутрішніми звітами, аналітичними системами, CRM тощо.

Окрім іншого, API Asana підтримує використання вебхуків, що дає змогу оперативно реагувати на зміни в системі — наприклад, при оновленні статусу завдання або завершенні етапу проєкту (рис. 1.9, 1.10).

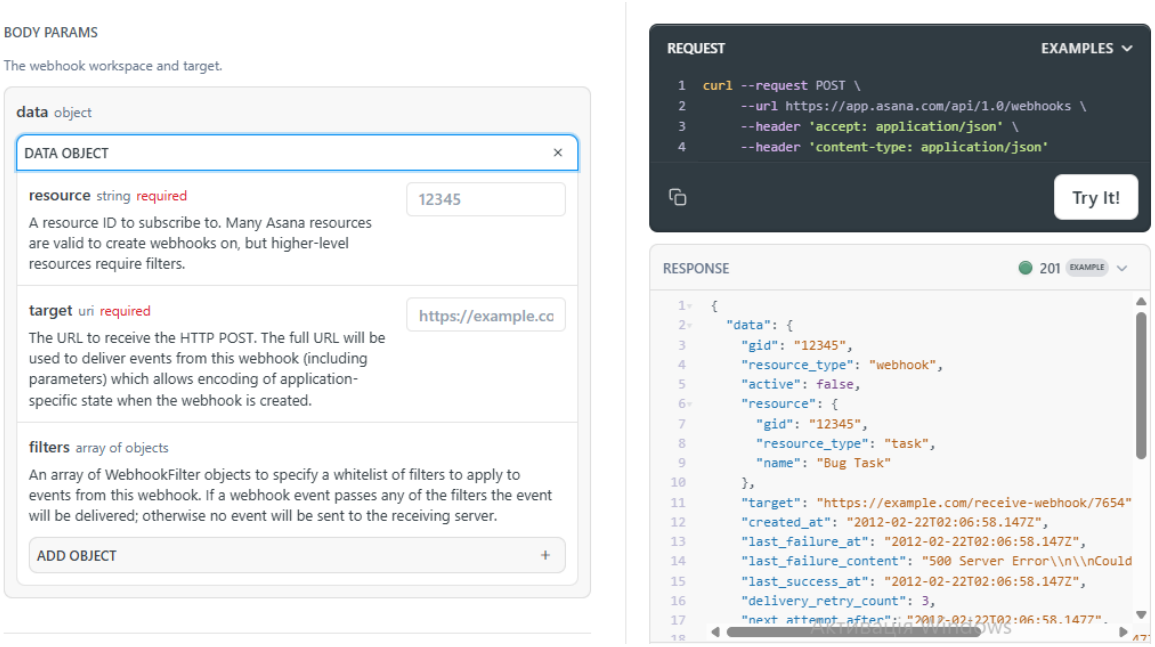

Рисунок 1.9 - Оголошення вебхука через API Asana

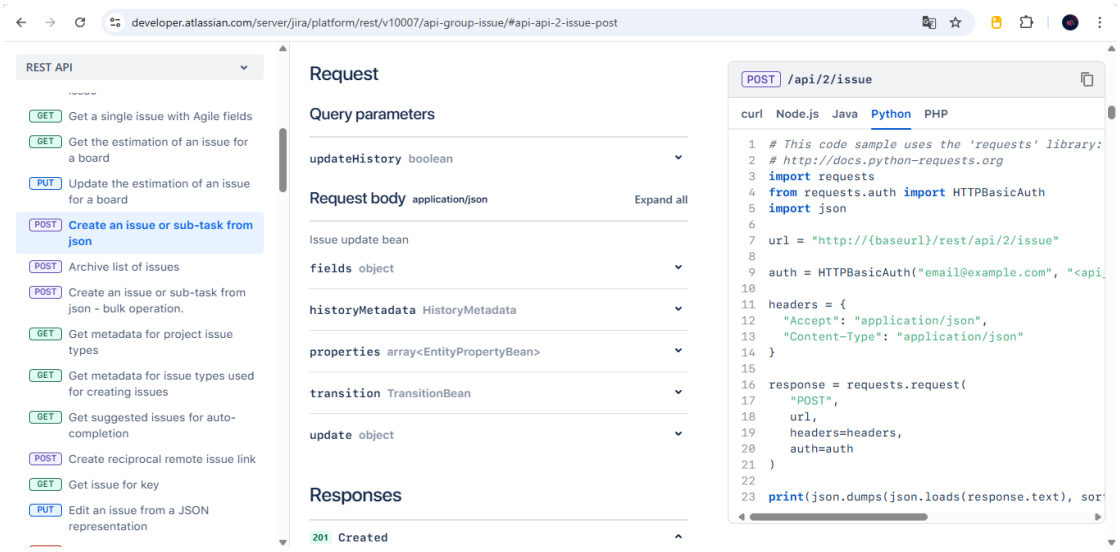

Рисунок 1.10 – Створення issue через API Jira

Документація API Asana є детальною, структурованою та доступною для розуміння як початківцям, так і досвідченим розробникам. Вона доступна у відкритому доступі за адресою: https://developers.asana.com. Документація містить:

— Опис базових концепцій (об'єкти, ресурси, аутентифікація);

— Чіткі приклади HTTP-запитів та відповідей;

— Сторінку з інструментом API Explorer, що дозволяє тестувати запити безпосередньо в браузері;

— Розділи для роботи з помилками, обмеженнями (rate limits) та безпекою;

— Інформацію про оновлення та зміни в API.

Варто відзначити наявність SDK та клієнтських бібліотек для популярних мов програмування — таких як JavaScript (Node.js), Python, Ruby та PHP — що значно полегшує впровадження інтеграцій у практичних проєктах.

API дає змогу значно скоротити кількість ручної роботи в управлінні проєктами. Наприклад, замість ручного створення схожих задач для повторюваних процесів (щотижневі наради, регулярні звіти тощо), можна реалізувати сценарій, де завдання створюються автоматично за заданим шаблоном.

Багато організацій використовують декілька цифрових платформ одночасно: CRM, ERP, HelpDesk, внутрішні бази даних. API дозволяє об’єднати дані з Asana з іншими системами. Це забезпечує централізоване управління інформацією, уникнення дублювання, автоматичне оновлення статусів і кращу синхронізацію дій між відділами.

Через API можна отримати детальні дані щодо виконання завдань, прогресу команди, ефективності окремих учасників проєктів. Ці дані можуть бути використані для побудови кастомізованих аналітичних панелей у Power BI, Tableau, Google Data Studio тощо.

Використання API дозволяє адаптувати Asana до специфіки конкретної організації. Якщо базового функціоналу не достатньо, API дає змогу створити надбудову або окремий модуль з розширеними функціями — наприклад, автоматичне створення завдань з вхідних запитів клієнтів або побудову складних ланцюгів затверджень.

Вебхуки в API Asana дають змогу запускати дії в інших сервісах у відповідь на зміни всередині платформи, наприклад, при оновленні завдання.

API Asana використовує стандарт OAuth 2.0 для авторизації. Це гарантує захист

персональних даних, контроль доступу та можливість інтеграції з сучасними системами авторизації. Додатково впроваджено механізм обмеження запитів (rate limiting), що захищає API від перевантаження і зловживань.

Офіційна документація API Asana — цінне джерело для бізнесів, які прагнуть інтегрувати цю платформу в свої цифрові процеси. Завдяки своїй структурованості, доступності та підтримці сучасних стандартів API Asana є гнучким і надійним інструментом для реалізації як базових сценаріїв автоматизації, так і складних архітектур у корпоративному середовищі.

Рекомендовано до використання технічними спеціалістами, IT-відділами, проектними менеджерами, розробниками кастомних рішень та всіма, хто прагне максимізувати продуктивність роботи з Asana.

1.2.3 Ознайомлення з AsanaWebhook

У сучасних умовах цифрової трансформації важливо забезпечити оперативну обробку подій, що дозволяє будувати гнучкі, масштабовані й адаптивні бізнес-системи. Одним із механізмів, який дозволяє системам миттєво реагувати на внутрішні чи зовнішні зміни, є вебхуки (webhooks). У контексті платформи Asana вебхуки відіграють ключову роль у побудові автоматизованих робочих процесів і забезпеченні оперативного обміну даними з іншими сервісами та внутрішніми системами.

Asana Webhook — це функціональність, яка дозволяє зовнішнім додаткам підписуватися на зміни, що відбуваються у конкретних об’єктах (наприклад, завданнях або проєктах), і миттєво отримувати сповіщення у вигляді HTTP POST-запитів при настанні відповідних подій. Такий підхід дозволяє значно зменшити навантаження на інтерфейси прикладного програмування (API), знижує

затримки під час обробки інформації та забезпечує миттєву синхронізацію між компонентами системи.

Вебхуки в Asana функціонують за класичною моделлю «підписки на подію». Розробник створює запит до API Asana з вказанням ресурсу (наприклад, ID завдання або проєкту), який слід моніторити, та URL-адреси сервера, на який Asana надсилатиме сповіщення. Кожного разу, коли відбувається подія, пов'язана з цим ресурсом (створення, зміна, видалення), Asana відправляє POST-запит з відповідними даними на вказану адресу.

Передача даних відбувається у форматі JSON, що забезпечує зручність інтеграції з більшістю сучасних мов програмування та вебфреймворків. Події можуть бути агрегованими (batch), що дозволяє обробляти кілька змін одночасно в межах одного запиту (рис. 1.11, 1.12).

Для створення вебхука необхідно:

— Бути авторизованим через Asana API з використанням токену доступу (access token).

— Надати валідний endpoint (URL), який відповідає на вхідні HTTP-запити з кодом 200 OK.

— Підтвердити реєстрацію вебхука шляхом обробки handshake-запиту з верифікаційним заголовком X-Hook-Secret.

Рисунок 1.11 - Принцип роботи Asana Webhook

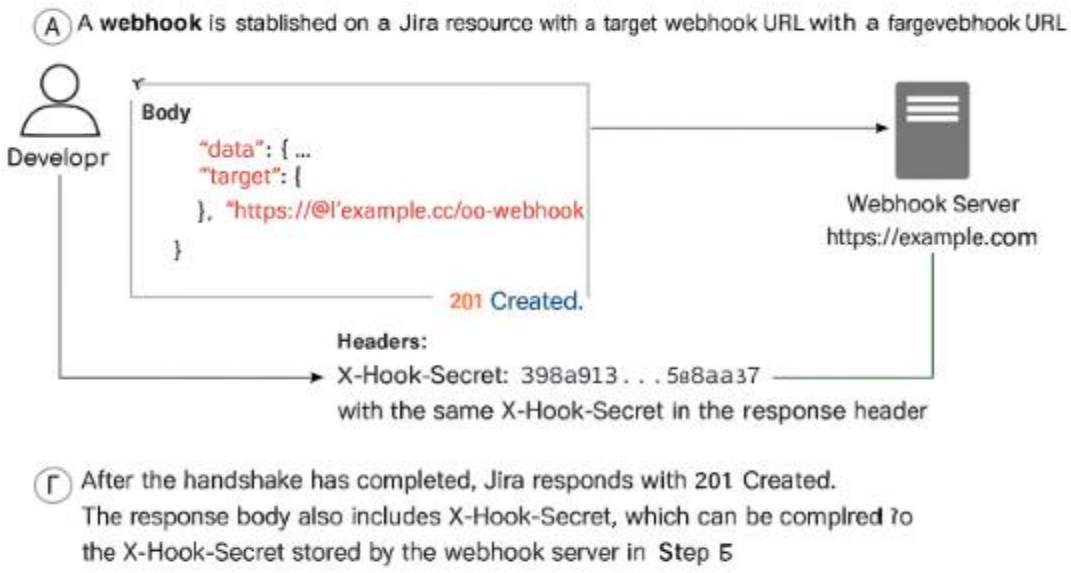

Рисунок 1.12 - Принцип роботи Jira Webhook

Asana також очікує, що сервер-отримувач оброблятиме вхідні події швидко (менше 5 секунд), інакше з'єднання може бути перерване. У разі непрацездатності вебхука протягом певного часу, Asana автоматично його деактивує, що захищає систему від "висячих" підписок.

Один із найпоширеніших сценаріїв використання вебхуків — інтеграція Asana з іншими системами (наприклад, Slack, Jira, Google Sheets, CRM). Наприклад, при завершенні завдання в Asana система може автоматично надсилати повідомлення в Slack або оновлювати статус у внутрішній базі даних.

Інтеграція Asana Webhook з Sentry дозволяє автоматизувати створення завдань у Asana на основі нових критичних помилок, зафіксованих у Sentry. Наприклад, помилки рівня "error" або "fatal" у Sentry — система надсилає HTTP-запит на вказану адресу, де сервер обробляє цю інформацію та за допомогою API Asana створює відповідне завдання для команди розробників. Такий підхід забезпечує безперервну передачу проблем з середовища моніторингу в систему управління завданнями, мінімізуючи час реакції та виключаючи потребу в ручному дублюванні інформації.

Завдяки вебхукам можна побудувати незалежну систему нотифікацій, яка

інформує відповідальних осіб про зміни в задачах, що стосуються їхньої участі. Це особливо ефективно в проєктах з великою кількістю учасників, де важливо не пропустити жодної ключової дії.

Технічні характеристики та обмеження:

— Підтримувані ресурси: На момент написання, Asana підтримує вебхуки для завдань, проєктів і портфелів.

— Обмеження: Є обмеження на кількість активних вебхуків на одного користувача або додаток.

— Безпека: Asana використовує верифікаційний заголовок X-Hook-Secret, який дозволяє впевнитися, що запит надходить саме з Asana. Рекомендується додатково захищати endpoint токенами або авторизацією.

— Формат повідомлень: JSON-повідомлення містить список подій із зазначенням типу змін, ідентифікаторів об’єктів, часу події та додаткових метаданих.

На відміну від регулярного опитування (polling), яке вимагає періодичних запитів до API, вебхуки надсилають повідомлення лише при фактичних змінах. Це дозволяє зменшити навантаження на сервер і оптимізувати використання API-лімітів.

Вебхуки забезпечують реакцію в реальному часі — між зміною в Asana і сповіщенням проходить менше однієї секунди. Такий підхід особливо цінний у сценаріях, де затримка навіть у кілька хвилин є критичною (наприклад, у службах підтримки або розподілі завдань між командами).

Вебхуки дозволяють створювати сценарії, які враховують специфіку бізнесу: автоматичне інформування, запуск зовнішніх обчислень, зміна логіки робочих процесів без втручання користувача.

Незважаючи на очевидні переваги, використання вебхуків вимагає дотримання низки рекомендацій:

— Забезпечити високу доступність та надійність endpoint-сервера;

— Реалізувати обробку помилок та логування подій;

— Перевіряти автентичність запитів з боку Asana;

— Уникати дублювання обробки при агрегації подій.

У разі складних систем із багатьма паралельними вебхуками рекомендовано реалізувати систему черг або буферизації повідомлень.

Вебхуки Asana є ефективним і високопродуктивним механізмом для автоматизації процесів, що дозволяє у режимі реального часу отримувати інформацію про зміни у системі управління завданнями. Завдяки своїй простоті впровадження, широкому охопленню функцій та можливості глибокої інтеграції з внутрішніми та зовнішніми платформами, Asana Webhook є ключовим інструментом для побудови адаптивних, реактивних систем управління інформацією.

У поєднанні з Asana API, вебхуки відкривають шлях до побудови комплексної цифрової екосистеми, що дозволяє організаціям не лише координувати роботу, а й передбачати події, оптимізувати ресурси та автоматизувати взаємодію на всіх рівнях проєктного менеджменту.

1.3 Існуючі рішення

Інтеграція систем моніторингу, таких як Sentry, із системами управління завданнями, зокрема Asana, є важливою складовою сучасних DevOps-практик. Вона дозволяє автоматично передавати інформацію про помилки або інциденти безпосередньо у середовище планування задач, що підвищує швидкість реагування та ефективність командної роботи. На сьогодні існує кілька типів рішень, які забезпечують подібну інтеграцію без потреби у повноцінній розробці.

Sentry має вбудовану інтеграцію з Asana, яка дозволяє прямо з інтерфейсу моніторингу створювати нові завдання. Це рішення швидке у налаштуванні та не вимагає технічних навичок. Проте його функціонал є обмеженим — наприклад, немає можливості реалізувати гнучку логіку або визначати складні правила розподілу задач.

Платформи на зразок Zapier, Make, n8n, Zoho Flow дозволяють створювати складніші сценарії без програмування. Наприклад, за допомогою таких сервісів можна реалізувати автоматичне створення задач у певному проєкті Asana при реєстрації критичної помилки в Sentry, додати відповідального, дедлайн тощо. Однак ці платформи мають обмеження в обсязі доступних подій, а також залежать від платних підписок і сторонніх серверів.

Існують також рішення з відкритим кодом або шаблони на GitHub, які дозволяють використати Sentry Webhooks і Asana API у кастомізованій логіці. Такі рішення часто використовуються як база для побудови власної інтеграції, дозволяючи реалізувати базову логіку швидко, із подальшим розширенням функціоналу за потреби.

Хоча згадані інструменти забезпечують інтеграцію між Sentry та Asana без розробки, їх можливості є обмеженими у довгостроковій перспективі. Вони підходять для малих команд або для тестування концепції, однак при зростанні складності бізнес-процесів постає потреба у повноцінному кастомному рішенні. Тому наступні розділи присвячені глибшому аналізу: з одного боку — переваг і недоліків безкодового підходу, з іншого — потенціалу та гнучкості власної інтеграції.

1.3.1 Варіанти реалізація без розробки власної інтеграції

Існує кілька готових рішень для інтеграції Sentry з Asana, які не вимагають розробки власного коду. Ці рішення включають вбудовані інтеграції, платформи безкодових (no-code) та низькокодових (low-code) автоматизацій, а також інструменти для побудови кастомних робочих процесів. Кожен з цих підходів має свої переваги та обмеження, які слід враховувати при виборі оптимального рішення для конкретних потреб організації.

Найпростішим варіантом інтеграції є використання офіційної інтеграції між Sentry та Asana. Ця інтеграція дозволяє автоматично створювати завдання в Asana при виникненні певних подій у Sentry, таких як нові помилки або інциденти. Налаштування інтеграції зазвичай не вимагає технічних знань і може бути виконано через інтерфейс користувача обох систем (рис. 1.13).

Переваги:

— Простота налаштування та використання.

— Відсутність необхідності в програмуванні.

— Швидкий старт для малих команд або організацій з обмеженими ресурсами.

— Обмеження:

— Обмежена гнучкість у налаштуванні правил обробки подій.

— Відсутність можливості реалізації складної логіки або бізнес-процесів.

— Залежність від оновлень та підтримки з боку розробників обох платформ.

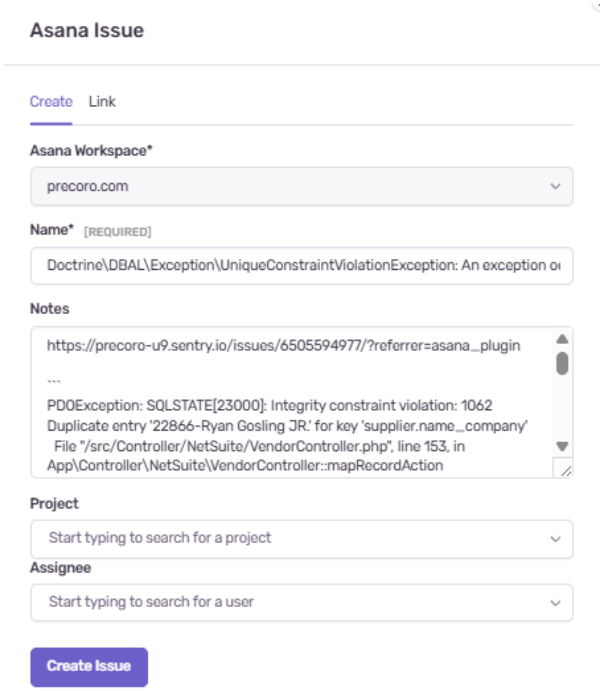

Рисунок 1.13- Форма створення задач з влаштованої інтеграції з Asana

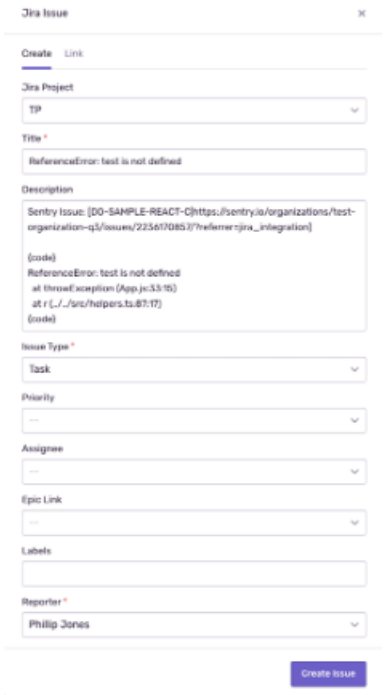

Рисунок 1.14- Форма створення задач з влаштованої інтеграції з Jira

Для більш складних сценаріїв інтеграції можна використовувати платформи безкодових (no-code) та низькокодових (low-code) автоматизацій, такі як Zapier, Make (раніше Integromat), n8n, Pipedream та інші. Такі сервіси надають можливість створювати автоматизовані інтеграції між різними платформами, зокрема між Sentry та Asana, без необхідності писати програмний код.

Zapier є однією з найпопулярніших платформ безкодових автоматизацій. Вона підтримує інтеграцію з понад 3,000 додатків, включаючи Sentry та Asana. Користувачі мають змогу налаштовувати автоматичні ланцюжки дій, так звані "Zaps", які реагують на події в одній системі та ініціюють дії в іншій.

Серед переваг:

— Широкий вибір попередньо налаштованих інтеграцій.

— Програмний інтерфейс сервісу відзначається простотою та зручністю для кінцевого користувача

Обмеження:

— Обмеження на кількість операцій у безкоштовному плані.

— Вартість підписки може бути високою для великих обсягів використання.

— Обмежена гнучкість у налаштуванні складних сценаріїв.

Make — це потужна платформа для автоматизації, яка дозволяє створювати складні робочі процеси з використанням візуального редактора. Вона підтримує інтеграцію з багатьма сервісами, включаючи Sentry та Asana.

Переваги:

— Підтримка складних сценаріїв та логіки.

— Більша гнучкість у налаштуванні процесів порівняно з Zapier.

— Можливість обробки великих обсягів даних.

Обмеження:

— Крута крива навчання для нових користувачів.

— Потребує часу для освоєння всіх можливостей платформи.

— Може бути надмірним для простих автоматизацій.

n8n — це open-source інструмент для побудови автоматизованих сценаріїв, що надає можливість створювати гнучкі й складні інтеграції між численними онлайн-сервісами. Вона підтримує інтеграцію з Sentry та Asana через відповідні вузли (nodes).

Переваги:

— Відкрите програмне забезпечення з можливістю самостійного хостингу.

— Велика гнучкість у налаштуванні процесів.

— Безкоштовне використання при самостійному хостингу.

Обмеження:

— Потребує технічних знань для налаштування та підтримки.

— Може бути складним для користувачів без досвіду в автоматизації.

— Обмежена кількість готових шаблонів порівняно з конкурентами.

Pipedream — це платформа для автоматизації, яка дозволяє створювати

серверлес-функції та інтеграції між різними сервісами. Вона підтримує інтеграцію з Sentry та Asana через відповідні API.

Переваги:

— Підтримка складних сценаріїв та логіки.

— Можливість використання власного коду для налаштування процесів.

— Безкоштовне використання при невеликому обсязі операцій.

Обмеження:

— Потребує знань у програмуванні для створення кастомних функцій.

— Може бути складним для користувачів без досвіду в розробці.

— Обмежена кількість готових інтеграцій порівняно з конкурентами.

Для організацій з унікальними вимогами або специфічними бізнес-процесами, які не можуть бути реалізовані за допомогою стандартних інтеграцій, існують інструменти для побудови кастомних робочих процесів. Це можуть бути серверні рішення, які використовують API Sentry та Asana для створення власних інтеграцій.

1.3.2 Власна інтеграція

Хоча на ринку доступні різноманітні інструменти для автоматичної інтеграції Sentry та Asana, у багатьох випадках саме розробка власної інтеграції є оптимальним рішенням. Особливо це актуально для організацій, які мають складні внутрішні процеси, високі вимоги до автоматизації або необхідність підтримувати специфічну логіку обробки інцидентів. Власна інтеграція відкриває можливість повністю контролювати весь потік задач — від створення і асайну до закриття після виправлення (рис. 1.15).

Рисунок 1.15- Структура власної інтеграції

Основною перевагою власної інтеграції є максимальна гнучкість. Стандартні платформи автоматизації працюють за шаблонними сценаріями та часто мають обмеження у функціоналі. Наприклад, неможливо реалізувати складну логіку розподілу завдань між виконавцями залежно від типу помилки, середовища чи рівня критичності. Власна інтеграція дозволяє:

1) Налаштувати індивідуальні правила асайну: розподіл задач на основі тегів, сервісів, пріоритетів або ключових слів.

2) Визначати умови для створення задач: не всі помилки потребують створення задачі; власна інтеграція дозволяє фільтрувати події на основі складної логіки.

3) Реалізувати автоматичне закриття задач після фіксу: інтеграція з системами контролю версій (наприклад, GitHub або GitLab) дає змогу оновлювати статус задачі при злитті pull request.

4) Створювати зв’язки між задачами або формувати підзадачі на основі одного інциденту.

Завдяки використанню API Asana та Webhooks Sentry, власна інтеграція може повністю автоматизувати цикл задач. Наприклад:

1) У Sentry фіксується нова критична помилка.

2) Webhook передає інформацію на бекенд сервіс інтеграції.

3) Сервіс обробляє дані, визначає тип помилки, авторів коду, середовище, тощо.

4) В Asana створюється задача в потрібному проєкті, автоматично додається виконавець, встановлюється дедлайн і пріоритет.

5) Коли фікс інтегрується в основну гілку репозиторію, інтеграція автоматично закриває задачу в Asana.

Таке налаштування дозволяє не лише скоротити час на ручне дублювання подій, але й мінімізує помилки, що виникають через людський фактор.

Готові рішення працюють через сторонні платформи, що може бути неприйнятним з точки зору безпеки або відповідності політикам компанії (compliance). Власна інтеграція розгортається у внутрішній інфраструктурі компанії, що дозволяє:

— Повністю контролювати всі дані, що передаються між системами.

— Впроваджувати додаткові шари безпеки (аутентифікація, шифрування, журналювання).

— Відповідати внутрішнім стандартам безпеки, включаючи GDPR, ISO 27001, SOC2 та інші.

Контроль над кодом інтеграції також дозволяє здійснювати аудит, тестування та швидко вносити зміни при зміні бізнес-процесів.

Ще однією ключовою перевагою є масштабованість. При збільшенні обсягу подій або рості кількості команд, власна інтеграція може бути легко адаптована до нових вимог. Наприклад:

— Додавання нових правил маршрутизації задач для окремих відділів або ліній підтримки.

— Підтримка багатомовності в описі задач або специфічних форматів для різних проєктів.

— Інтеграція з іншими системами (Jira, Slack, Notion) для створення

мультиканального сповіщення.

Гнучкість архітектури дозволяє розширювати функціонал без обмежень, що характерні для сторонніх сервісів автоматизації.

Хоча початкова розробка власної інтеграції потребує вкладень часу та ресурсів, у довгостроковій перспективі це може бути більш економічно вигідним рішенням. Платформи типу Zapier або Make беруть плату залежно від кількості операцій, а при високій активності ця вартість стрімко зростає.

Власна інтеграція не має обмежень на кількість транзакцій і може бути оптимізована для мінімального використання ресурсів. Крім того, підтримка власного коду уможливлює оперативне усунення помилок або оновлення відповідно до змін API, не чекаючи офіційних оновлень сторонніх провайдерів.

Власна інтеграція між Sentry та Asana — це стратегічно вигідне рішення для команд, що прагнуть максимальної ефективності, автоматизації та контролю. Вона дозволяє реалізувати повний цикл обробки помилок із глибокою кастомізацією, покращити внутрішні процеси та забезпечити відповідність безпековим стандартам. У контексті високих вимог до гнучкості та продуктивності, розробка кастомної інтеграції виправдовує себе як з технічної, так і з бізнесової точки зору.

Таблиця 1.3 - Порівняння усіх рішень:

Рішення	Спосіб підключення	Гнучкість налаштувань	Підтримка логіки/фільтрів	Надійність
Zapier	Через тригери та дії	Обмежена	Обмежена (в основному ключові поля)	Середня
Make (ex-Integromat)	Сценарії з блоками	Висока	Так, підтримує сценарії та фільтри	Висока

Продовження таблиця 1.3

Unito	Двостороння синхронізація	Обмежена	Ні	Середня
Ручне створення задач	Немає інтеграції	Повна (людський контроль)	Так	Низька
Власна інтеграція (через API)	Webhook + сервер + API	Повна	Так	Висока

2. РОЗРОБКА ІНТЕГРАЦІЇ ТА ЇЇ ТЕСТУВАННЯ

2.1 Аналіз вимог

Загальна мета проєкту

Метою проєкту є створення інтеграції між системою моніторингу помилок Sentry та системою керування завданнями Asana, яка дозволяє автоматизувати процес створення, оновлення та закриття задач у Asana на основі нових івентів (помилок), що надходять у Sentry. Основною задачею інтеграції є усунення необхідності ручного створення задач, що значно скоротить час реакції на помилки та зменшить ймовірність їх пропущення.

Автоматичне створення задач у Asana. Кожен новий івент у Sentry, який відповідає встановленим умовам, автоматично створює відповідне завдання в Asana. Завдання має містити ключову інформацію: назву помилки, опис, URL до помилки в Sentry, теги, стейк трейс, інформацію про оточення тощо.

Автоматичне закриття задач після фіксу: після того, як івент у Sentry позначено як вирішений (resolved), відповідна задача в Asana автоматично закривається.

Підтримка кастомних правил асайнменту. Інтеграція повинна дозволяти налаштовувати правила призначення відповідального виконавця у Asana на основі: тегів у Sentry, назви помилки, URL, інших кастомних критеріїв.

Управління дублями задач:

— Якщо одна й та сама помилка повторюється, не створювати нову задачу, а оновлювати існуючу (наприклад, збільшення лічильника чи оновлення коментаря).

Обробка помилок інтеграції:

— У випадку невдач при створенні задач або інших критичних помилках у роботі інтеграції – надсилати повідомлення в Slack у відповідний канал.

— Інформування також дублюється через Sentry (логування self-errors).

— Нефункціональні вимоги

— Надійність:

— Система має бути стійкою до тимчасових збоїв API (наприклад, Asana або Sentry можуть бути тимчасово недоступні).

— Повинна бути реалізована система повторних спроб (retry) або черга обробки подій.

Гнучкість:

— Всі кастомні правила мають бути легко конфігурованими через файл конфігурації або просту адмінку.

— Підтримка мульті-проектного оточення (підтримка кількох Sentry-проектів та кількох Asana workspace/проєктів).

Безпека:

— Зберігання токенів та секретів через безпечні механізми (наприклад, AWS Secrets Manager, Vault або environment variables з обмеженим доступом).

— Всі зовнішні запити мають бути захищеними (HTTPS, валідація даних).

Масштабованість:

— Система має підтримувати обробку великої кількості івентів без втрати продуктивності.

Інтеграції зі сторонніми сервісами

— Slack – для повідомлень про помилки, що виникли в самій інтеграції, або інших важливих подій.

— Sentry (internal) – для логування помилок самої інтеграції.

— Asana – як цільова система для створення та оновлення задач.

— Sentry (external) – як джерело івентів, на основі яких відбувається автоматизація.

— Calamari – перевірка доступності відповідального перед асайном задачі (з урахуванням відпусток чи відсутності).

Очікувані результати

— Зниження часу на обробку нових помилок.

— Прозорість у розподілі відповідальності (через кастомні правила асайнменту).

— Підвищення надійності процесу обробки помилок.

— Мінімізація ризику пропуску критичних помилок.

2.2 Архітектура проекту

Інтеграція між платформами Sentry та Asana покликана автоматизувати процес створення та супроводу задач, пов’язаних із помилками у продакшн-системі. Коли в Sentry з’являється новий івент (помилка), система автоматично створює відповідну задачу в Asana. Після виправлення помилки та підтвердження цього в Sentry, відповідна задача в Asana автоматично закривається. Такий підхід значно полегшує комунікацію між командами розробки, забезпечує кращу відслідковуваність та автоматизацію процесів.

Архітектура реалізована у вигляді окремого застосунку на мові програмування Python. Вона модульна, кожен компонент відповідає за певну частину логіки: обробка вебхуків, взаємодія з API, логування, допоміжні утиліти та конфігурації. Далі наведено детальний опис кожного з основних файлів проєкту (рис. 2.1).

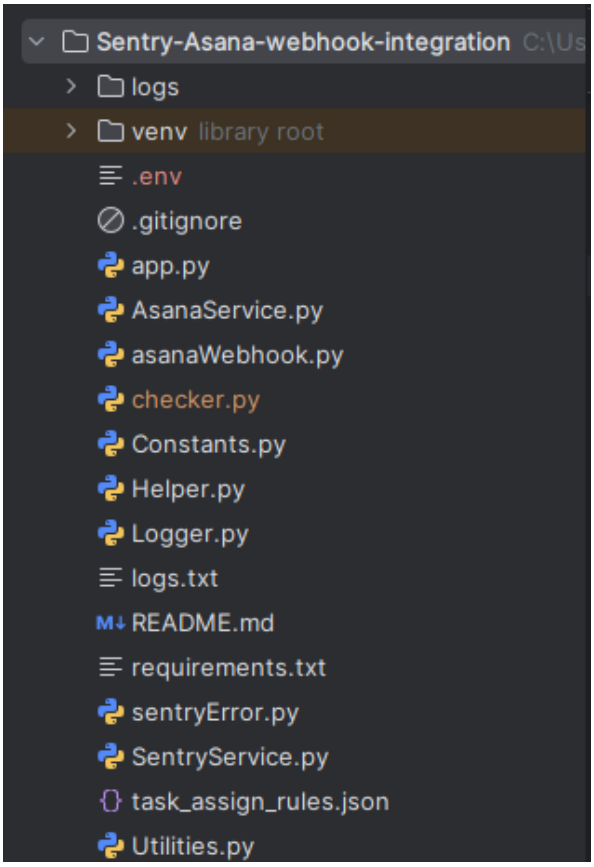

Рисунок 2.1- Архітектра проекту

2.2.1 Основний компонент app.py

Це основний вхідний файл системи, з якого починається виконання інтеграції. Він відповідає за ініціалізацію всіх необхідних сервісів, запуск серверу або обробника вебхуків, а також виклик основної логіки залежно від типу запиту.

У файлі може міститися базова логіка маршрутизації запитів (наприклад, за допомогою Flask), виклики методів обробки запитів з Sentry та Asana, ініціалізація логування, завантаження конфігураційних даних тощо. Це свого роду “точка входу”, яка координує взаємодію між різними модулями.

2.2.2 Модуль Asana API

Модуль, відповідальний за взаємодію з Asana API. Тут зібрані всі методи, що дозволяють створювати, оновлювати, завершувати задачі в Asana, а також призначати відповідальних користувачів.

Ключові функції:

Створення задач з передачею заголовку, опису та додаткових параметрів;

— Закриття задач за відповідними ідентифікаторами;

— Оновлення існуючих задач (наприклад, статус, коментарі);

— Отримання списку користувачів або проєктів;

— Пошук задач за специфічними атрибутами (наприклад, ID помилки із Sentry).

Цей модуль інкапсулює всі HTTP-запити до Asana, працює з токенами, обробляє помилки взаємодії та здійснює логування важливих подій.

2.2.3 Реалізує логіки створення, підтвердження та обробки вебхуків

Цей файл реалізує логіку створення, підтвердження та обробки вебхуків з боку Asana. Вебхуки в Asana дозволяють отримувати повідомлення про події, пов’язані із задачами, наприклад, коли задача оновлена вручну користувачем.

Файл включає такі частини:

— Метод для реєстрації вебхука на потрібний проєкт;

— Обробку вхідних повідомлень;

— Верифікацію підпису запиту;

— Обробку типових подій: зміна статусу задачі, оновлення опису, видалення

тощо.

Цей компонент важливий для синхронізації стану задачі у випадку ручного втручання користувача (рис. 2.2).

{'user': {'gid': '1208100461534378', 'resource_type': 'user'}, 'created_at': '2025-05-03T12:55:35.965Z', 'action': 'changed', 'parent': None, 'resource': {'gid': '1210151521892015', 'resource_type': 'task', 'resource_subtype': 'default_task'}, 'change': {'field': 'custom_fields', 'action': 'changed', 'new_value': {'gid': '1207100760528488', 'resource_type': 'custom_field', 'resource_subtype': 'enum', 'enum_value': {'gid': '1209464008800903', 'resource_type': 'enum_option'}}}}

Рисунок 2.2- Приклад вебхуку з асани

2.2.4 Початкові данні для роботи

Файл, який містить захардкоджену інформацію, що часто використовується в різних частинах проєкту. Це можуть бути:

- Ідентифікатори проєктів Asana;
- Токени доступу до API (у dev-середовищі);
- Значення статусів;
- Повідомлення про помилки;
- Регулярні вирази;
- Типи подій.

Наявність такого модуля дозволяє централізовано управляти конфігурацією і швидко змінювати ключові параметри без потреби редагування всієї логіки проєкту.

2.2.5 Модуль допоміжних методів

Модуль містить допоміжні методи, які використовуються у багатьох частинах проєкту. Основна особливість — всі функції переважно є статичними (@staticmethod), що дозволяє викликати їх без створення об'єктів.

Основні функції:

— Форматування рядків (наприклад, назви задач);

— Перетворення дат та часу;

— Фільтрація даних;

— Створення унікальних ключів;

— Повторне виконання HTTP-запитів з логікою retry;

— Перевірка наявності поля в JSON-структурах.

Цей файл грає роль "утилітного ящика", в якому зберігаються часто використовувані дрібні інструменти.

2.2.6 Організацію системи логування

Файл відповідає за організацію системи логування. Тут реалізовані:

— Ініціалізація логера;

— Формат логів;

— Розділення логів за рівнями: DEBUG, INFO, WARNING, ERROR;

— Обробка винятків;

— Запис логів у файл із зазначенням дати.

Централізоване логування дозволяє відслідковувати помилки та активність

інтеграції в продакшн-режимі, що критично важливо для моніторингу стабільності системи.

2.2.7 Зберігання логів проєкту

Це директорія, в якій зберігаються всі логи, згенеровані застосунком. Структура логів організована за датами.

Кожен файл логів містить інформацію про всі дії, які відбулися за відповідну дату, включаючи запити до Asana і Sentry, обробку помилок, роботу вебхуків тощо.

2.2.8 Файл документації

Файл документації, в якому детально описано:

- Призначення інтеграції;
- Необхідні залежності;
- Покроковий процес установки;
- Приклади конфігурації;
- Приклади API-викликів;
- Часті помилки та способи їх вирішення.

Цей файл слугує основним джерелом інформації для нових розробників або тих, хто впроваджує систему.

2.2.9 Залежності проєкту

Файл, в якому перераховані всі залежності проєкту — бібліотеки, які необхідні для коректної роботи скриптів.

Він дозволяє легко відтворити середовище проєкту за допомогою команди pip install -r requirements.txt.

2.2.10 Обробка подій

Модуль, що відповідає за обробку подій, які надходять з Sentry. Він обробляє JSON-вебхуки, які містять інформацію про нові помилки, їх статус, пріоритет, файл, у якому виникла помилка, та інші деталі (рис. 2.3).

Основна логіка:

— Розпізнавання нових помилок;

— Перевірка, чи вже існує задача по цьому івенту;

— Створення нової задачі або оновлення існуючої;

— Визначення, коли помилка була виправлена, та закриття задачі в Asana.

{'action': 'unresolved', 'installation': {'uuid': '262bc32b-912c-443b-83e5-18b36ffd7fee'}, 'data': {'issue': {'id': '6569512525', 'shareId': None, 'shortId': 'PRECORO-US-49F', 'title': '[HC]Attachments that have been sending to a Google provider and have been in the processing status for more than 4 hours: Count 2, 130719, 130742', 'culprit': '', 'permalink': None, 'logger': 'monolog.app', 'level': 'fatal', 'status': 'unresolved', 'statusDetails': {}, 'substatus': 'regressed', 'isPublic': False, 'platform': 'php', 'project': {'id': '4506162475499520', 'name': 'precoro-us', 'slug': 'precoro-us', 'platform': 'php-symfony'}, 'type': 'default', 'metadata': {'title': '[HC]Attachments that have been sending to a Google provider and have been in the processing status for more than 4 hours: Count 2, 130719, 130742', 'sdk': {'name': 'sentry.php.symfony', 'name_normalized': 'sentry.php'}, 'initial_priority': 75}, 'numComments': 0, 'assignedTo': {'email': 'mariia.kozoriz@precoro.com', 'type': 'user', 'id': '3451421', 'name': 'Mariia Kozoriz'}, 'isBookmarked': False, 'isSubscribed': False, 'subscriptionDetails': None, 'hasSeen': False, 'annotations': [{'url': 'https://app.asana.com/0/0/1210087218882535', 'displayName': 'Asana Issue'}], 'issueType': 'error', 'issueCategory': 'error', 'priority': 'high', 'priorityLockedAt': None, 'isUnhandled': False, 'count': '5', 'userCount': 0, 'firstSeen': '2025-04-26T03:43:00.435717Z', 'lastSeen': '2025-05-03T03:43:41.110363Z'}}, 'actor': {'type': 'application', 'id': 'sentry', 'name': 'Sentry'}}

Рисунок 2.3 - Приклад вебхука з сентрі

2.2.11 Правила призначення задач

JSON-файл, в якому зберігаються правила призначення задач. Наприклад, залежно від типу помилки, модуля, сервісу або відповідального розробника (рис. 2.4).

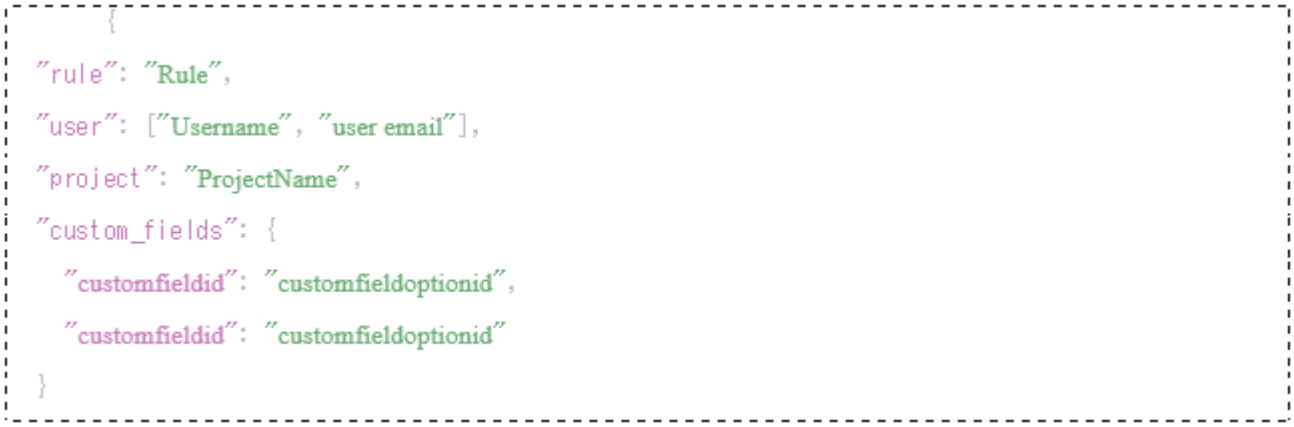

Рисунок 2.4 - JSON-файл, в якому зберігаються правила призначення задач

Це дозволяє налаштувати гнучкий розподіл задач між розробниками або командами без змін у коді.

2.2.12 Уникнення циклічних імпортів

Файл створено з метою уникнення циклічних імпортів. У ньому містяться функції, які використовуються у кількох модулях

2.3 Тестування інтеграції

Тестування програмних продуктів є ключовим етапом у процесі розробки, оскільки воно дозволяє впевнитися у відповідності роботи системи заданим вимогам, відсутності серйозних дефектів та стабільності функціонування в різних умовах. Тестування допомагає виявити помилки на ранніх етапах, зменшити ризики в продакшн-середовищі та підвищити довіру до системи з боку користувачів.

У випадку інтеграції між Sentry та Asana тестування має особливе значення, оскільки система автоматично створює та закриває задачі на основі помилок у продакшн-середовищі. Будь-яка несправність в інтеграції може призвести до:

— пропуску критичних помилок;

— дублювання задач;

— некоректного асайну задач;

— або до спаму в Slack/Asana.

Основні напрямки тестування:

Функціональне тестування:

— Перевірка, що задачі створюються лише за умов, визначених у конфігурації.

— Перевірка, що задачі оновлюються або закриваються відповідно до статусу в Sentry.

Інтеграційне тестування:

— Перевірка взаємодії з API Asana, Sentry, Slack та Calamari.

— Обробка помилок і нестабільних відповідей від сторонніх сервісів.

Тестування асайнменту:

— Перевірка кастомних правил розподілу задач.

— Врахування даних з Calamari (чи є відповідальний у відпустці).

Негативне тестування:

— Моделювання збоїв (наприклад, недоступність API, некоректні токени, помилки формату).

— Перевірка, що система не "ламається", а коректно логує помилки та надсилає повідомлення у Slack.

Тестування безпеки та доступу:

— Перевірка захищеності збереження токенів.

— Обмеження доступу до конфігурації та логів.

Загалом, ретельне тестування дозволяє гарантувати, що інтеграція працює стабільно, автоматизація дій не призводить до небажаних наслідків, а команда може покладатися на систему як на надійний інструмент обробки помилок.

2.3.1 Заходи для відловлювання помилок

У контексті інтеграції між Sentry та Asana, одним із ключових аспектів стабільної роботи є своєчасне виявлення ситуацій, коли система не створила задачу в Asana, хоча мала б це зробити. З цією метою були реалізовані декілька механізмів моніторингу та самоперевірки.

1) Внутрішній моніторинг через Sentry

Для самої інтеграції використовується вбудований Sentry, який дозволяє логувати виключення, що виникають під час обробки подій. Це забезпечує:

— миттєве сповіщення про помилки в роботі самої інтеграції;

— деталізований стек трейс;

— можливість аналізу повторюваних збоїв.

2) Верифікація створення задач – verify_task_created

Ключовим елементом контролю є метод verify_task_created, який виконує перевірку фактичної наявності задачі в Asana після того, як має відбутись її створення. Алгоритм включає:

Фільтрацію подій від Sentry, які мають відповідати наступним умовам:

— подія походить з опрацьовуваних проектів;

— тип події — created або unresolved;

— джерело події — саме Sentry, а не інший актор;

— виключення подій, пов'язаних із міграціями ('migrations' in title).

Отримання останнього очікуваного імені задачі, сформованого на основі ID

issue з Sentry.

Пошук задачі через API Asana в межах останніх 5 хвилин, з урахуванням часового поясу (Kyiv Time).

Якщо очікувана задача не знайдена — надсилається повідомлення у Slack, що сигналізує про потенційну помилку в інтеграції.

3) Slack-нотифікації

Slack використовується як канал оперативного реагування:

— при виявленні невідповідності (очікувана задача не створена);

— при виключеннях в процесі виконання перевірки.

Це дозволяє команді швидко виявляти та вирішувати проблеми, не чекаючи фідбеку від кінцевих користувачів.

Завдяки такому підходу інтеграція має вбудовану перевірку власної роботи та оперативне інформування про будь-які аномалії. Це дозволяє підтримувати високу надійність та вчасно виявляти критичні збої, які могли б призвести до втрати важливих помилок у продуктивному середовищі.

2.3.2 Підготовка середовища для тестування

Для безпечного та ізольованого тестування інтеграції між Sentry та Asana було створено окреме тестове середовище, яке не впливає на основні бізнес-процеси компанії. Це дозволяє вільно перевіряти поведінку системи без ризику створення зайвих задач у продуктивних проєктах.

Основні компоненти середовища тестування:

— Тестовий проєкт в Asana. Використовується спеціально створений проект в Asana, який не має жодного стосунку до робочих просторів компанії. Усі тестові задачі створюються саме в цьому проекті.

— Локальний сервер + Ngrok. Для отримання webhook-ів із Sentry використовується локальний сервер, піднятий розробником. Доступність іззовні забезпечується за допомогою Ngrok, який генерує публічний HTTPS URL, що передається в налаштуваннях Sentry як ціль для надсилання подій.

— Окремий Sentry-проєкт або фільтрована подія. Події для тестування надсилаються або з окремого тестового проєкту в Sentry, або через симуляцію реальних подій за допомогою спеціально сформованих payload-ів.

Таке середовище дозволяє імітувати реальні ситуації, зберігаючи контроль над усім процесом тестування (рис. 2.5).

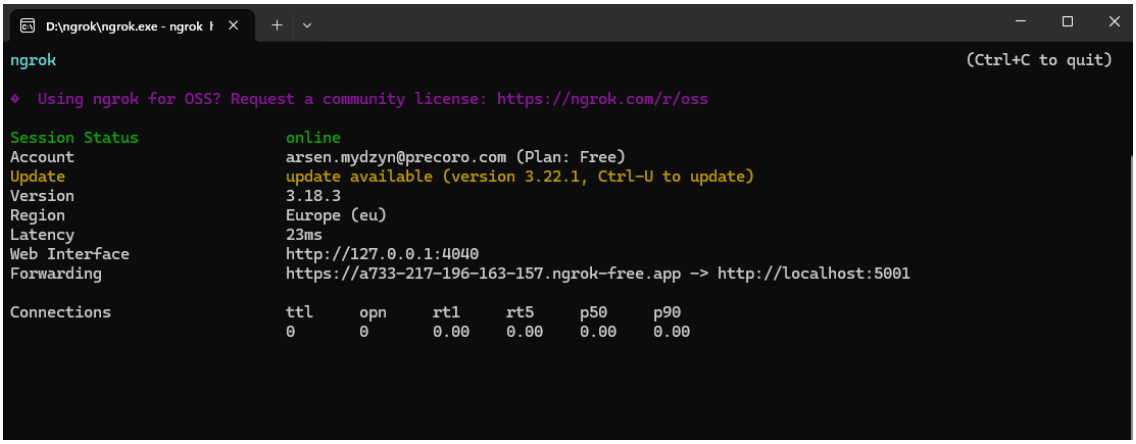

Рисунок 2.5 - Піднятий через ngrok сервер

2.3.3 Реалізація тестування

Після налаштування середовища відбувається поетапне тестування ключових сценаріїв інтеграції. Основні кроки реалізації:

1) Імітація webhook-подій із Sentry

Через Sentry (тестовий проєкт або вручну сформовані події) надсилаються webhook-и на публічний URL, згенерований через Ngrok. Це дозволяє перевірити:

— коректність обробки подій;

— створення задач у відповідному проєкті Asana;

— логіку фільтрації подій (наприклад, виключення "migrations").

2) Візуальна перевірка задач у тестовому проєкті Asana

Перевіряється, чи створена задача має правильну назву, опис, чи присутні потрібні поля та чи відповідає очікуванням.

3) Тестування обробки edge-case-ів

Включає:

— надсилання невалідних даних;

— симуляцію відсутності ключових полів;

— перевірку реакції на зміни у статусі задач (resolved).

4. Перевірка Slack-нотифікацій та обробки помилок

Імітуються ситуації, коли задача не створена або виникає виключення — перевіряється, чи коректно інтеграція інформує через Slack.

5. Логування та self-tracking через Sentry

Усі помилки, які виникають під час тестування, автоматично фіксуються у внутрішньому проєкті Sentry для інтеграції.

Таке тестування дозволяє переконатися, що інтеграція працює як очікується у повноцінних умовах, але без ризику вплинути на продуктивне середовище. Це створює безпечний цикл розробки, де кожна зміна може бути перевірена перед релізом (рис. 2.6).

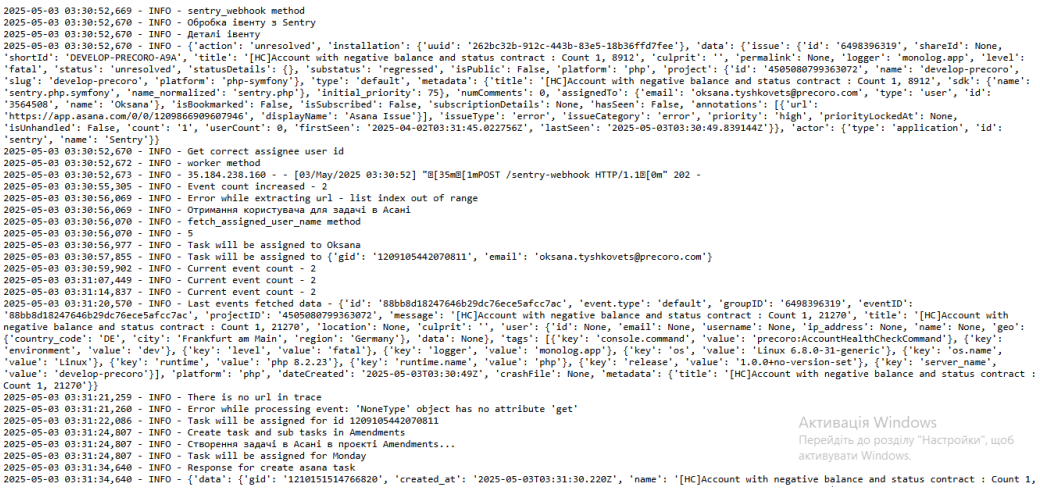

Рисунок 2.6 – Вигляд лінки на яку виводятись логи

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Аварії з викидом радіоактивних речовин

За даними світової статистики, що фіксується в головній службі небезпечних ситуацій, великі аварії на підприємствах та об’єктах різних типів, де лінійні розміри зон ураження досягають декілька сотень або навіть тисяч метрів, є досить рідкісними. Проте тим не менш, у світі в середньому за рік відбувається близько 2 – 3 подібних аварій.

Аварії із загибеллю понад 25 осіб і числом поранених більше 100 реєструється в середньому раз на 2,5 роки.

У цілому, як вважають фахівці, спостерігається неухильне зростання кількості промислових і енергетичних аварій, викликане, з одного боку, збільшенням кількості небезпечних об’єктів, з іншого боку, зростанням питомої щільності населення в зонах розвитку промислових і енергетичних об’єктів [19].

Найбільша в історії людства радіаційна катастрофа на Чорнобильській АЕС сталась 26 квітня у 1986 році. Кілька років після катастрофи всі офіційні джерела СРСР повідомляли, що жертвами Чорнобиля стали тільки 33 людини – в основному пожежники, які брали участь в найперших роботах. Потім почали з’являтись окремі повідомлення про те, що від променевої хвороби загинуло кілька десятків ліквідаторі, а захворіли тисячі. Про жертви серед місцевого населення не говорилось взагалі. Режим секретності з питань аварії ЧАЕС, який існував до 1991 року, не дозволяв відтворити об’єктивну картину масштабів ураження населення [19].

За сучасними уявленнями, аварія на ЧАЕС має серйозні наслідки пролонгованої дії, в тому числі такі, що можуть виявлятись на генетичному рівні в окремих груп персоналу АЕС, ліквідаторів і населення, яке проживає поблизу зони аварії.

У результаті вибуху четвертого реактора Чорнобильської атомної електростанції стався величезний викид радіоактивних речовин в атмосферу. Ці

радіоактивні опади випали в основному в межах євро-азіатського континенту, але особливо у великих кількостях на значних територіях Білорусі, Російської Федерації та України.

За оцінками, протягом 1986–1987 рр. до ліквідації наслідків аварії було залучено понад 350000 чоловік-«ліквідаторів» з числа військовослужбовців, працівників АЕС, місцевої міліції та пожежних служб. Досить високі дози радіації отримали близько 240000 чоловік під час проведення робіт з ліквідації наслідків аварії в межах 30-кілометрової зони, виконання робіт з консервації аварійного 4-го блоку АЕС – будівництва «Саркофагу», очищення дахів АЕС, створення системи захисту водних об'єктів.

У даний час приблизно п'ять мільйонів людей проживають в районах, де рівні радіоактивного забруднення ґрунтів цезієм перевищують 37 кБк/м2. З них приблизно 270000 людей продовжують жити в районах, які класифікувалися повноважними органами як зони посиленого контролю ,де зараження 137Cs перевищує 555 кБк/м2.

Можна припустити збільшення кількості випадків смерті від раку протягом усього життя серед осіб, які зазнали впливу радіації в результаті аварії. У зв'язку з тим, що в даний час неможливо визначити, які конкретні випадки раку були викликані радіацією, кількість таких випадків смерті можна оцінити лише статистично на основі використання інформації та проекцій, отриманих під час досліджень на людях, які вижили після вибухів атомних бомб [19, с. 202, 210].

3.2 Проведення інструктажів з охорони праці

Охорона праці є невід'ємною складовою безпеки та здоров'я працівників у робочому середовищі. Важливим елементом ефективної системи охорони праці є проведення інструктажів, які дозволяють ознайомити працівників з ризиками,

правилами та процедурами, необхідними для забезпечення безпечних умов праці.

У даному розділі розглянемо процес проведення інструктажів з охорони праці та його значення для працівників та організації.

Інструктаж з охорони праці є систематичним інформуванням працівників про правила безпеки та здоров'я на робочому місці. Це процес передачі необхідних знань, навичок та умінь, які дозволяють працівникам свідомо та безпечно виконувати свої обов'язки. Типи інструктажів включають загальні інструктажі, інструктажі для конкретних професій та робіт, а також інструктажі для особливих ситуацій та надзвичайних випадків.

Проведення інструктажів з охорони праці має велике значення як для працівників, так і для організації. Для працівників інструктажі є важливим джерелом інформації про потенційні ризики та способи їх уникнення. Вони допомагають зрозуміти правила та процедури, які забезпечують безпеку та здоров'я на робочому місці. Крім того, інструктажі сприяють формуванню свідомого ставлення до охорони праці, підвищують самосвідомість та відповідальність працівників за власну безпеку та безпеку оточуючих.

Організації проведення інструктажів має також важливе значення. Вони є ефективним інструментом управління ризиками та зменшенням можливості виникнення нещасних випадків на робочому місці. Інструктажі допомагають організації дотримуватися вимог законодавства з охорони праці, покращувати репутацію та відносини зі співробітниками, а також знижувати витрати, пов'язані з нещасними випадками та професійними захворюваннями [20].

Процес проведення інструктажів з охорони праці включає декілька етапів: Першим етапом є підготовка до інструктажу, включаючи визначення мети, вибір методів та форм проведення. Наступним кроком є збір та підготовка необхідної інформації, матеріалів та документації. Далі проводиться сам інструктаж, включаючи пояснення правил, демонстрацію технік та практичні вправи. Після проведення інструктажу слід здійснити контроль засвоєння матеріалу та відповідності вимогам

безпеки. Необхідно також забезпечити постійне оновлення інструктажів з охорони праці в разі змін у виробничому процесі, технологіях чи вимогах законодавства.

Проведення інструктажів з охорони праці є важливою складовою безпеки та здоров'я працівників. Вони дозволяють ознайомити працівників з ризиками та процедурами безпеки, сприяють формуванню свідомого ставлення до охорони праці та підвищують рівень безпеки на робочому місці. Проведення інструктажів також має важливе значення для організації, допомагаючи управляти ризиками, забезпечувати дотримання вимог законодавства та покращувати відносини з працівниками.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно реалізовано поставлену на початковому етапі мету — розроблено інтеграційне рішення, яке забезпечує автоматизоване створення та супровід задач у системі Asana на основі подій, що надходять із платформи моніторингу помилок Sentry. Цей підхід виявився ефективним засобом зменшення ручної роботи з боку розробників, забезпечив підвищення швидкості реагування на критичні помилки та дозволив знизити вплив людського фактора на процес керування інцидентами в інформаційних системах.

На початкових етапах було детально проаналізовано предметну область, визначено можливості та обмеження кожної з платформ — Sentry та Asana, зокрема в частині доступу до функціоналу через REST API та підтримки обробки подій через webhook-механізми. Було виявлено, що хоча існують готові рішення інтеграції на основі no-code платформ, таких як Zapier чи Make, вони не надають належної гнучкості та обмежені у масштабованості. Саме тому було обґрунтовано доцільність розробки власного інтеграційного прототипу, який дозволяє детально контролювати логіку взаємодії та адаптувати її до потреб конкретного проєкту.

У процесі реалізації було спроектовано архітектуру рішення, що включає модуль прийому повідомлень від Sentry, обробку отриманих даних, формування задач із відповідними тегами та описом у системі Asana, а також логіку автоматичного закриття задач після фіксації помилки. Особливу увагу приділено обробці дублювань, захисту від збоїв у роботі інтеграції та зручному логуванню подій, що дозволяє ефективно відслідковувати працездатність рішення в режимі реального часу.

Проведене тестування підтвердило стабільність та функціональну відповідність реалізованої системи очікуванням. Рішення продемонструвало здатність до коректної роботи з великим обсягом подій, дозволило зменшити часові витрати на ручну

обробку помилок і забезпечило прозорість у розподілі задач між учасниками команди. Позитивні результати тестування стали підставою для висновку про готовність рішення до застосування в умовах реального виробничого середовища.

Під час роботи також було досліджено безпекові аспекти функціонування інтеграції. Зокрема, приділено увагу захисту webhook-інтерфейсів, правильній роботі з токенами доступу, а також принципам збереження конфіденційної інформації під час взаємодії з API сторонніх сервісів. Таким чином, реалізоване рішення відповідає базовим вимогам безпеки, що дозволяє його безпечно застосовувати в організаціях, які працюють із персональними чи бізнес-критичними даними.

У практичному вимірі розроблене програмне забезпечення є корисним для команд розробників, які прагнуть автоматизувати рутинні процеси, підвищити ефективність усунення помилок та забезпечити безперервну передачу інформації між сервісами. Особливо воно буде доречним у мікросервісних архітектурах, де частота появи інцидентів є високою, а швидкість їхньої обробки критичною для стабільності функціонування системи.

На завершення варто наголосити, що розроблена інтеграція є відкритою до масштабування та подальшого розвитку. Передбачена структура дозволяє адаптувати її до нових платформ, змін у вимогах бізнесу чи розширенні функціоналу без істотних модифікацій основної логіки. Таким чином, результати даної роботи мають прикладне значення та можуть бути основою для побудови більш комплексних інтеграційних систем у межах сучасної DevOps-практики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Sentry. Official Documentation. [Електронний ресурс] – Режим доступу: https://docs.sentry.io

2) Asana. Developer Guide. [Електронний ресурс] – Режим доступу: https://developers.asana.com

3) Zapier. Automation Platform Overview. [Електронний ресурс] – Режим доступу: https://zapier.com

4) Make.com (ex-Integromat) Documentation. [Електронний ресурс] – Режим доступу: https://www.make.com

5) GitHub – Sentry & Asana Integration Examples. [Електронний ресурс] – Режим доступу: https://github.com

6) REST API Design Guidelines. [Електронний ресурс] – Режим доступу: https://restfulapi.net

7) JSON Web Token Introduction. [Електронний ресурс] – Режим доступу: https://jwt.io

8) OAuth 2.0 Authorization Framework. [Електронний ресурс] – Режим доступу: https://oauth.net/2/

9) DevOps Culture and Automation – Microsoft Docs. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/devops

10) Calamari Leave Management API. [Електронний ресурс] – Режим доступу: https://calamari.io

11) HashiCorp Vault Documentation. [Електронний ресурс] – Режим доступу: https://www.vaultproject.io

12) AWS Secrets Manager. [Електронний ресурс] – Режим доступу: https://aws.amazon.com/secrets-manager

13) Slack API Reference. [Електронний ресурс] – Режим доступу: https://api.slack.com

14) Jenkins Documentation. [Електронний ресурс] – Режим доступу: https://www.jenkins.io/doc

15) Docker Documentation. [Електронний ресурс] – Режим доступу: https://docs.docker.com

16) n8n Workflow Automation Platform. [Електронний ресурс] – Режим доступу: https://n8n.io

17) Microsoft Teams Integration Guide. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com

18) Jira API Guide. [Електронний ресурс] – Режим доступу: https://developer.atlassian.com

19) Плачкова С.Г., Плачков І.В. Енергетика: історія, сучасність і майбутнє. – Київ, 2013. – 301 с. – ISBN 978-966-8163-18-0.

20) Бар’яхтяр В. Катастрофи на АЕС та атомна енергетика. – Київ, 2014. – 36 с. – ISSN 1819-7329.: наказ Державної служби України з надзвичайних ситуацій від 17 травня 2022 року №253.

ДОДАТКИ

Додаток А – Лістинг коду

Лістинг коду В.1 – Оголошення вебхука Sentry

```
@sentry_bp.post('/sentry-webhook')
def sentry_webhook():
    logger.setup_logger()
    """Handle incoming webhook events from Sentry."""
    logging.info("sentry_webhook method")
    event_data = request.get_json()
    event_queue.put(event_data)  # Додаємо івент в чергу
    # Запускаємо потік для обробки івентів
    threading.Thread(target=worker, daemon=True).start()
    return jsonify({'message': 'Event received and will be processed.'}),
202
```

Лістинг коду В.2 – Оголошення вебхука Asana

```
@asana_bp.post('/asana-webhook')
def asana_webhook():
    x_hook_signature = request.headers.get("X-Hook-Signature")
    x_hook_secret = request.headers.get("X-Hook-Secret")
    if x_hook_secret:
        new_secret = x_hook_secret
        update_x_hook_secret(new_secret)
        load_dotenv(override=True)
        logging.info("-------------ASANA                                WEBHOOK
ESTABLISHED-------------")
        return jsonify({}), 200, {"X-Hook-Secret": new_secret}
    elif x_hook_signature:
        stored_secret = helper.get_x_hook_secret()
        computed_signature = hmac.new(
            stored_secret.encode("utf-8"),
```

```
                request.data,
                hashlib.sha256
            ).hexdigest()
            if not hmac.compare_digest(x_hook_signature, computed_signature):
                return jsonify({"error": "Invalid signature"}), 401
            else:
                events = request.json.get("events", [])
                for event in events:
                    asana_event_queue.put(event)
                threading.Thread(target=asana_worker, daemon=True).start()
                return jsonify({'message': 'Custom field change received and
will be processed.'}), 202
        else:
            return jsonify({"error": "Missing necessary headers"}), 400
```

Лістинг коду В.3 – Setup Logger

```
@staticmethod
def setup_logger():
    current_date = datetime.now().strftime('%Y-%m-%d')
    log_filename = os.path.join('logs', f'log_{current_date}.log')
    if not os.path.exists('logs'):
        os.makedirs('logs')
    log_files = [f for f in os.listdir('logs') if f.startswith('log_') and
f.endswith('.log')]
    log_files.sort(reverse=True)
    if len(log_files) > 5:
        files_to_delete = log_files[5:]
        for file in files_to_delete:
            os.remove(os.path.join('logs', file))
            logging.info(f"Deleted old log file: {file}")
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)
    for handler in logger.handlers[:]:
        logger.removeHandler(handler)
    file_handler = logging.FileHandler(log_filename)
```

```
    file_handler.setLevel(logging.INFO)
    formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
    file_handler.setFormatter(formatter)
    logger.addHandler(file_handler)
    return logger
```

Лістинг коду В.4 – Приклад методу створення задачі

```
def create_asana_task_in_bugs(self, user_id, issue_id, project_name, title, event_data, custom_fields=None):
    """Create a new bug task in Asana."""
    logger = logging.getLogger(__name__)
    logger.info("Creating Asana bug task...")
    stage_id, title = self._prepare_stage_and_title(project_name, title)
    followers = self._get_followers(project_name, user_id, event_data)
    custom_fields = self._prepare_custom_fields(custom_fields, stage_id, title, issue_id)
    task_data = self._build_task_data(user_id, issue_id, title, event_data, followers, custom_fields)
    return self._send_task_to_asana(task_data)
```

Лістинг коду В.5 – Приклад методу обробки івенту

```
    def handle_generic_project(self, event_data, issue_id, project_id, annotations, project_name):
        """Обробка івентів для інших проектів."""
        user_name, user_email, project, custom_fields = self.sentry_service.check_assign_conditions(event_data)
        if user_name is None:
            user_email, project = self.sentry_service.fetch_assigned_username_and_email(issue_id, project_id, project_name)
        user_id = self.asana_service.get_assignee_user_id_by_email(user_email, project_name)
```

```
        title,                              event_data                              =
self.sentry_service.fetch_last_event_message_and_title(issue_id)
        if self.should_create_task(issue_id):
            task_id  =  self.asana_service.create_asana_task(project,  user_id,
issue_id, project_name, title, event_data, custom_fields)
            logging.info("Task created successfully.")
            if '[HC]' in title:
   self.asana_service.create_data_fix_sub_tasks_for_bugs_task(user_id,
task_id)
                logging.info("Sub tasks successfully created")
            self.assign_asana_task_to_sentry_issue(task_id=task_id,
issue_id=issue_id, annotations=annotations)
            logging.info("Assigned asana task to issue")
            self.assign_sentry_issue_by_user_email(user_id, issue_id)
            logging.info("Issue successfully assigned")
```

Лістинг коду В.6 – Метод перевірки створення задачі

```
def verify_task_created(self, event_data):
    try:
        if not self._should_verify(event_data):
            return
        issue_id = event_data['data']['issue']['id']
        task_name, _ = self.fetch_last_event_message_and_title(issue_id)
        if not self._is_task_recently_created(task_name):
            self.slack_service.send_message_to_slack(event_data)

    except Exception as e:
        error_message = f"Exception - {e}\nEvent_data - {event_data}"
        self.slack_service.send_message_to_slack(error_message)
```

Лістинг коду В.7 – Приклад допоміжного методу з файлу Helper.py

```
    @staticmethod
```

```
    def request_get(url: str, headers: dict, slack_service=None, retries: int =
3, delay: int = 5):
        """Performs a GET request with retries and optional Slack logging on
failure."""
        for attempt in range(1, retries + 1):
            try:
                response = requests.get(url=url, headers=headers, timeout=10)
                response.raise_for_status()
                return response
            except requests.RequestException as e:
                logging.warning(f"[Attempt {attempt}/{retries}] GET request
failed: {e}")
                if attempt < retries:
                    time.sleep(delay)
        error_msg = f"GET request failed for URL: {url}"
        if slack_service:
            try:
             slack_service.send_message_to_slack(f"{error_msg}\nError:
{str(e)}")
            except Exception as slack_err:
                logging.error(f"Failed to send error to Slack: {slack_err}")
        raise Exception(f"{error_msg}\nDetails: {str(e)}")
```

Додаток Б – Диск із кваліфікаційною роботою бакалавра