

Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

Тема роботи:

«Автоматизація процесу формування звітів у Jira з розпізнаванням табличних
даних»

Виконав студент гр. КН-20

Костін Є. С.

Керівник

Тронь В. В.

Нормоконтроль

Маринич І. А.

Завідувач кафедри

Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: інформаційних технологій

Кафедра: автоматизації, комп'ютерних наук і технологій

Ступінь вищої освіти: Бакалавр

Спеціальність: 122 – Комп'ютерні науки

ЗАТВЕРДЖУЮ

Зав. кафедрою: к.т.н. Рубан С.А.

« 27 » березня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-20 Костіну Євгену Сергійовичу

1. Тема кваліфікаційної роботи: «Автоматизація процесу формування звітів у Jira з розпізнаванням табличних даних»

затверджено наказом по університету № 235с від 27.03.2024 р.

2. Термін здачі кваліфікаційної роботи: 05.06.2024 р.

3. Склад кваліфікаційної роботи: Пояснювальна записка обсягом 59с., додатки, презентація у Microsoft PowerPoint (20 слайдів) в електронному та друкованому вигляді

4. Консультанти кваліфікаційної роботи:

Розділ 1-3

доц. Тронь В. В.

Нормоконтроль

доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	<i>Вступ</i>	<i>01.04.24</i>
2	<i>Розділ 1</i>	<i>05.04.24</i>
3	<i>Розділ 2</i>	<i>01.05.24</i>
4	<i>Висновки</i>	<i>25.05.24</i>
5	<i>Оформлення кваліфікаційної роботи</i>	<i>28.05.24</i>
6	<i>Підготовка презентації та графічного матеріалу</i>	<i>20.05.24</i>
7	<i>Підготовка доповіді до захисту</i>	<i>05.06.24</i>

6. Дата видачі завдання: 29.01.2024р.

Керівник _____ /Тронь В. В./

7. Запевнення: Я, Костін Євген Сергійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Костін Є. С./

АНОТАЦІЯ

Костін Є. С. Автоматизація процесу формування звітів у Jira з розпізнаванням табличних даних.

Кваліфікаційна робота на здобуття ступеня вищої освіти – бакалавр, за спеціальністю 122 Комп'ютерні науки. Криворізький національний університет, Кривий Ріг, 2024.

Робота складається зі вступу, двох розділів, висновків, переліку використаної літератури з 22 позицій та 0 додатків. Загальний обсяг роботи становить 59 сторінок, з яких основний зміст роботи викладено на 50 сторінках, включає 1 таблицю і 57 рисунків.

Ця робота присвячена розробці додатку для експорту даних із системи Jira. У рамках дослідження було проведено аналіз існуючих інструментів експорту, що виявив низку недоліків. На основі цього аналізу було розроблено концепцію та архітектуру нового додатку, що враховує виявлені проблеми.

Основною метою розробки є створення інструменту, який забезпечить користувачам більш гнучкий та зручний процес експорту даних, дозволить автоматизувати цей процес та зберігати всі необхідні метадані.

Результатом роботи є готовий до використання додаток для експорту даних із системи Jira, який надає користувачам нові можливості для автоматизації та налаштування експорту даних. Додаток буде доступний для завантаження та інтеграції через Atlassian Marketplace, що забезпечить його доступність для широкого кола користувачів.

JIRA, ЗАСТОСУНОК ДО JIRA, ВЕБ-ТЕХНОЛОГІЇ, REACT, JAVASCRIPT, HTML, ATLASSIANFORGE, NODEJS,

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ I. АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ЩО РОЗШИРЮЮТЬ МОЖЛИВОСТІ ЕКСПОРТУ	9
1.1 Аналіз підходів до підвищення ефективності процесу формування звітів у Jira	9
1.2 Аналіз засобів автоматизації процесу формування звітів у Jira	11
1.2.1 Exporter - Export Issues to Excel CSV PDF, Deiser.....	11
1.2.2 Time in status, SaaSJet.....	13
1.2.3 BigTemplate, AppFire.....	13
1.2.4. Better Excel Exporter for Jira, Midori.	15
1.2.5 Xporter Cloud для Jira, XBlend.	16
1.2.6. Advanced Export, Kanbanalytics.....	18
1.2.7 Requirements Management for Jira, Ease Solutions Pte Ltd.....	20
1.2.8 Інші додатки та порівняльна таблиця	20
Висновки до розділу:	22
РОЗДІЛ II. Проектування та реалізація застосунку для автоматизації процесу формування звітів у Jira з розпізнаванням табличних даних	23
2.1 Варіанти використання застосунку	23
2.2 Архітектура застосунку	25
2.3 Програмна реалізація застосунку	27
2.3.1. Структура додатку.	27
2.3.2 Розгортання додатку Atlassian Forge.....	28
2.3.3 Налаштування маніфесту.	30
2.3.5 Jira Rest API.....	37
2.4 Засоби тестування та середовища.	43
2.5 Використання застосунку.....	47

2.5.1. Головне вікно.....	47
2.5.2. Вікно редагування шаблону.....	48
2.5.3. Вікно планувальника.	52
Висновки до розділу:	54
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	57

ВСТУП

Для досягнення високих показників ефективності при організації робочих процесів у компаніях різного масштабу та напрямку необхідним є використання сучасних інструментів управління проектами. Система Jira, що розроблена компанією Atlassian, є однією з найпопулярніших систем, яка дозволяє вести проекти, контролювати завдання та аналізувати результати. Завдяки своїй гнучкості та потужному функціоналу, Jira використовується в різних галузях. Вона забезпечує ефективну співпрацю між членами команди, дозволяє відстежувати прогрес виконання завдань і сприяє прийняттю обґрунтованих рішень на основі аналітичних даних.

Хоча Jira задовольняє широкий спектр потреб користувачів, на певних етапах виникає потреба у використанні більш спеціалізованого програмного забезпечення. Це зумовлює необхідність зручних шляхів експорту та імпорту даних.

Однак інструменти експорту, що вже передбачені у Jira, мають певні обмеження. Вони недостатньо гнучкі у налаштуванні, що ускладнює їх використання для спеціалізованих потреб, вони обмежені у автоматизації, через що доводиться витратити додаткові ресурси. Усе це не лише збільшує час, необхідний для виконання завдань, але й робить систему ненадійною, через залежність від людського фактору. Крім того, стандартні інструменти експорту можуть втрачати деталі і метадані, що є критичним для повного та точного аналізу даних. Похибки під час обробки великого обсягу даних - ще одна причина уникати вбудованих інструментів експортування.

Ця робота присвячена розробці та реалізації додатку для експорту даних із системи Jira. Метою є створення інструменту, який забезпечить користувачам більшу гнучкість і зручність у процесі експорту даних, дозволить автоматизувати цей процес та зберігати всі необхідні метадані. Розроблений додаток повинен підтримувати роботу з великими обсягами даних без втрати продуктивності та забезпечити інтеграцію з іншими системами.

Для досягнення поставленої мети необхідно: Провести аналіз існуючих інструментів експорту даних у Jira та визначити їхні основні недоліки; Розробити концепцію та архітектуру додатку, яка забезпечить вирішення виявлених проблем; Реалізувати додаток, використовуючи сучасні технології та методології розробки програмного забезпечення. Забезпечити можливість автоматизації процесу експорту даних, включаючи налаштування розкладу виконання задач.

Результатом виконання роботи стане готовий до використання додаток для експорту даних із системи Jira, який буде надавати користувачам нові можливості для автоматизації та налаштування експорту даних. Це дозволить підвищити ефективність робочих процесів, зменшити час на виконання рутинних завдань та знизити ризик помилок при експорті даних. Розроблений додаток буде доступний для завантаження та інтеграції через Atlassian Marketplace, що забезпечить його доступність для широкого кола користувачів.

РОЗДІЛ І.

АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ЩО РОЗШИРЮЮТЬ МОЖЛИВОСТІ ЕКСПОРТУ

Так як задача створення програмного продукту який би задовільнив всі потреби користувачів є неможливою. Компанія Atlassian пропонує можливість розширяти функціонал стандартної Jira за допомогою користувацьких додатків. Для їх створення пропонується використовувати Atlassian development kit, а платформа Atlassian Marketplace використовується для їх розповсюдження. Для вирішення проблем експортування, на ринку наявні користувацькі додатки різного масштабу. Від програм що виконують лише просте переформатування даних, до систем керування інтеграцією із базами даних, та систем для ґрунтового бізнес аналізу.

1.1 Аналіз підходів до підвищення ефективності процесу формування звітів у Jira

Jira – потужна система управління проектами розроблена компанією Atlassian[1]. Вона дозволяє побудувати ефективну систему взаємодій у команді, контролювати виконання завдань та гнучко налаштовувати робочий процес під мінливі умови сучасного світу. Цей інструмент використовується різноманітними бізнес-організаціями від компаній-розробників програмного забезпечення до керівництва готелів, або фабрик.

Для керування проектом Jira пропонує наступну схему:

Jira Instance (сайт) – верхній рівень системи, репрезентує організацію, для одного акаунту можна створити лише один сайт;

Projects (Проекти) – Другий рівень системи, один сайт може займатись одразу декількома проектами, кожний проект можна налаштовувати індивідуально;

Issues (задачі або тікети) – Третій рівень системи, основна одиниця над якою можуть здійснюватися операції, репрезентує задачу яку необхідно виконати, має велику кількість налаштувань та операцій. Може поділятися на типи та підпорядковуватись іншим задачам, так наприклад задача типу Epic може містити у собі декілька задач типу Story. Для задачі можна назначати працівника.

Field configuration (Схема полів задачі) – налаштування для типу задачі (можна налаштувати різні схеми полів для різних типів задач). Визначає список і роль робочих полів у задачі – які будуть відображатись у задачі, які будуть зберігати інформацію, які обов'язкові для заповнення і тому інше.

Workflow (Робочий процес) – налаштування для типу задачі. Визначає статуси у яких може перебувати задача, та порядок їх зміни. Зміна статусу задачі може слугувати тригером для автоматичної дії.

Зображення структури проекту наведено на рисунку 1.1

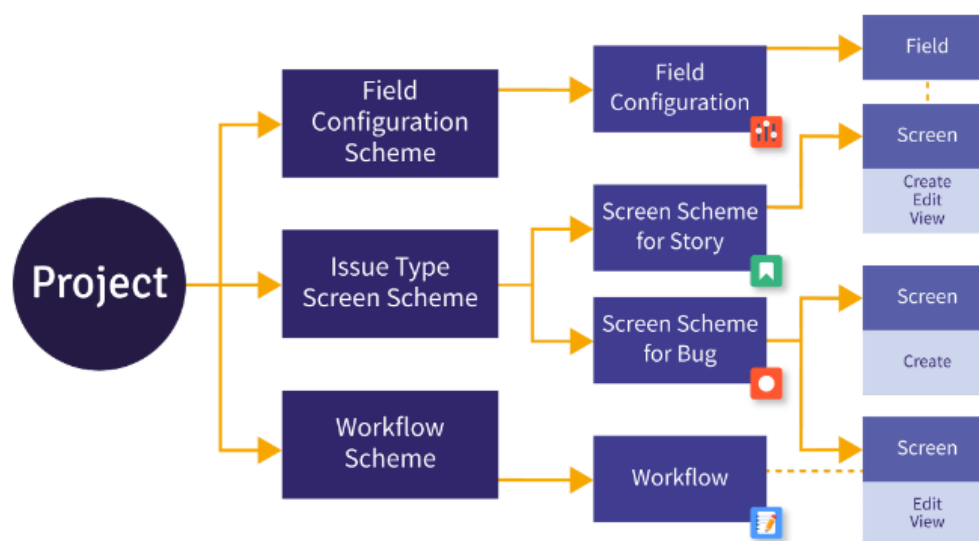


Рисунок 1.1 – Приблизна схема організації проекту у Jira

Щоб ефективно керувати проектом необхідна можливість аналізувати продуктивність робочих процесів, для чого можна використати запропоновані за замовчуванням звіти або користувацькі застосунки до Jira,

Запропоновані звіти поділяються на такі групи: Звіти про прогрес проекту (Project Progress Reports): Відображають загальний стан проекту, включаючи

кількість завершених задач, задач в процесі та залишених задач. Звіти про робоче навантаження (Workload Reports): Показують розподіл завдань між членами команди, дозволяючи керівникам ефективно керувати ресурсами. Звіти про продуктивність (Performance Reports): Аналізують продуктивність команди або її окремих членів за визначений період часу. Звіти про цикли випусків (Release Reports): Відображають стан виконання завдань у рамках певного випуску продукту. Звіти про дефекти (Defect Reports): Аналізують кількість знайдених і виправлених дефектів під час робочого процесу. Основним недоліком використання готових звітів є недостатня гнучкість їх налаштування, та неможливість автоматизації процесу звітування.

Для аналізу також можна використати користувацькі додатки. Основним плюсом цього підходу є можливість вибору серед майже необмеженої кількості варіації звітів. Мінусом же є необхідність витратити час на пошук додатку, що буде виконувати аналіз за необхідними параметрами, відповідати стандартам якості та матиме прийнятну ціну використання.

З вищесказаного виходить, що робити аналіз даних запропонованими методами не є оптимальною стратегією, тож пропонується винести аналіз за рамки Jira. Необхідно знайти спосіб автоматичного формування і експортування звітів, які можуть бути використані у сучасних засобах бізнес-аналізу. При цьому для забезпечення необхідного рівня гнучкості інструмент повинен мати можливість налаштування полів, що входять до звіту.

1.2 Аналіз засобів автоматизації процесу формування звітів у Jira

В якості інструменту що б виконував наведені вимоги можна розглянути наступні користувацькі застосунки:

1.2.1 Exporter - Export Issues to Excel CSV PDF, Deiser.

Exporter – додаток що вдосконалює можливості експорту запропоновані Jira, основний інтерфейс викликається кнопкою, при виборі групи задач у вікні фільтрів

Jira. Дозволяє експортувати дані у форматі CSV(Comma-Separated Values) з можливістю вибору роздільника для імпорту до програмного забезпечення за вашим вибором, експортувати дані у форматі Excel для побудови звітів поза системами Jira, та за допомогою спеціальної функції дозволяє масово пов'язувати до 300 задач з іншими (новими або існуючими) [2]. Додатково можна експортувати коментарі до задач та схеми робочого процесу проекту, що є корисним для аудиту і створення звітів, також у додатку наявна можливість автоматизованого експорту задач.

Хоча додаток задовольняє більшу частину потреб до подібного забезпечення, він не є достатньо гнучким для постійного використання. Наприклад якщо при формуванні звіту була допущена помилка у JQL(Jira Query Language) виразі, для її виправлення потрібно буде вводити всі параметри знову. Відсутність інтерфейсу для перегляду вже створених звітів робить неможливим роботу у команді та роботу одночасно над декількома проектами.

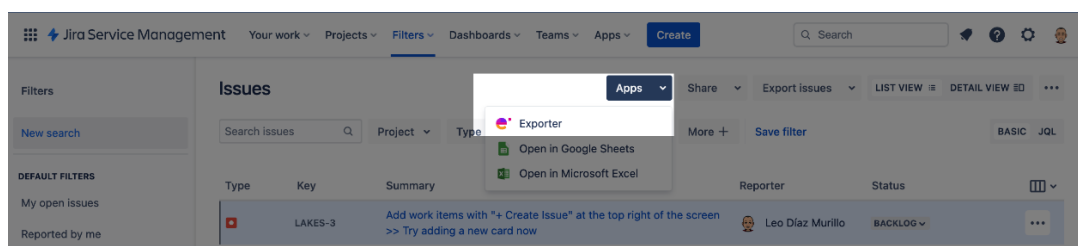


Рисунок 1.2 – Кнопка запуску додатку

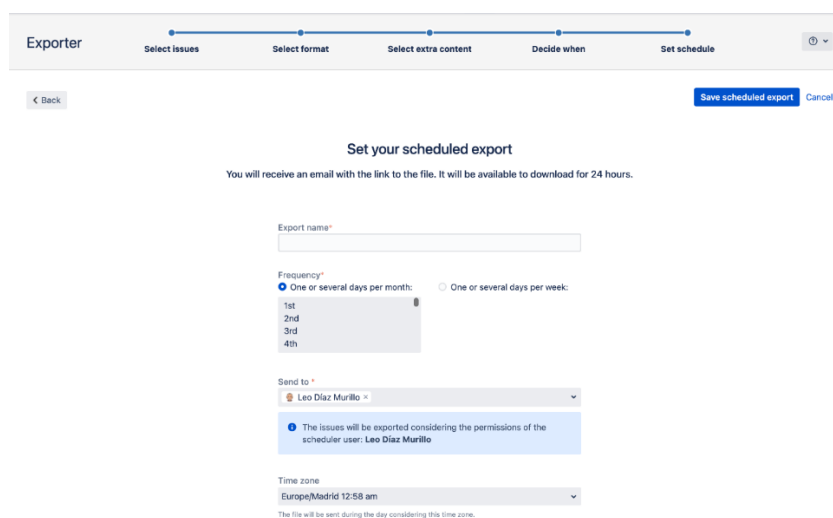


Рисунок 1.3 – Вікно запланованого експорту

1.2.2 Time in status, SaaSJet.

Time in status – додаток що дозволяє створити звіт з декількох вибраних шаблонів. Інтерфейс виконаний одним вікном, як таблиця з налаштуваннями. Основне призначення додатку - швидкий аналіз поточних задач, задля знаходження тих які уповільнюють проект. Додаток дозволяє відстежувати зміни у задачах, відстежувати і систематизувати їх за швидкістю виконання та відобразити звіт у форматі діаграм або таблиці, це дозволяє швидко знаходити “вузькі місця” у графіку проекту[3]. Програма інтегрується з більшістю популярних інструментів для бізнес аналізу.

Це потужний додаток для аналізу інформації, який гарно працює в рамках наданих за замовчуванням шаблонів. Проте він не дозволяє створювати повністю користувацькі шаблони, та не має функцій планування звітів, що робить неможливим автоматизацію документування.

Reporter	Assigned Time	Unassigned	Serge K.	Ihor Hutslaliuk	Jane Fox	Serhii Kibitkin	Danylo Ostapov	Yulia Borivets	Total
Jane Fox	16d 20h 5m	0m	0m	0m	0m	0m	0m	0m	16d 20h 5m
Jane Fox	27m	32d 16h 30m	0m	0m	0m	0m	0m	0m	32d 16h 58m
Jane Fox	30m	0m	32d 16h 30m	0m	0m	0m	0m	0m	32d 17h
Jane Fox	29m	32d 13h 30m	0m	3h	0m	0m	0m	0m	32d 17h

Рисунок 1.4 – Вікно інтерфейсу Time in status

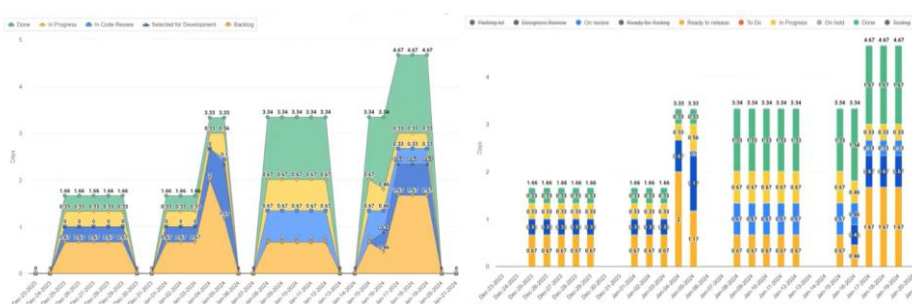


Рисунок 1.5 – Приклади діаграм створених у Time in status

1.2.3 BigTemplate, AppFire.

BigTemplate - це інструмент, розроблений для імпорту та експорту даних з Jira та BigPicture, основною метою розробки було розширення функціоналу

екосистеми додатків AppFire, тож як окремо взятий додаток він програє по кількості функцій. Він підтримує найпопулярніші формати файлів, включаючи PDF(Portable Document Format), DOC(формат для роботи з Microsoft Word), XLS(формат для роботи з Microsoft Excel), CSV та XML(eXtensible Markup Language), що значно розширює сферу використання експортованих даних[4]. Тобто користувачі зможуть швидко передавати інформацію проекту в інструменти, такі як Microsoft Excel, Word або MS Project. Особливістю цього додатка є можливість створити шаблон результуючого документа.

Templates License Widget

Export templates ⓘ

Upload multiple document templates to make exporting easily adaptable to your company's internal needs. [Design a template](#) or use the pre-installed one, make it visible on the Jira Cloud issue page, or one of the available modules, and go beyond the boring default.

MODULE	NAME	DESCRIPTION	FILE
WIDGET	Order	A beautiful purchase order document for all your expenses. You can easily generate ...	order_token.docx
WIDGET	Lend	A fully-fledged agreement of tool granting with space for photos, terms and conditio...	lend_style.docx
WIDGET	Lend	A fully-fledged agreement of tool granting with space for photos, terms and conditio...	lend_token.docx
WIDGET	Default	A simple table document that contains all styles used by BigTemplate plugin for refe...	default_style.docx
WIDGET	Leave	Exquisite non-working day application template to ease the process in your compan...	leave_style.docx

Рисунок 1.6 - Вікно керування шаблонами

Не дивлячись на унікальні функції, окремо взятий додаток значно програє конкурентам, через відсутність можливостей для автоматизації, та надмірно складний інтерфейс налаштування звітів. Наприклад для прив'язки значення поля до комірки таблиці необхідно отримати айді поля, використовуючи інструменти розробника у браузері.

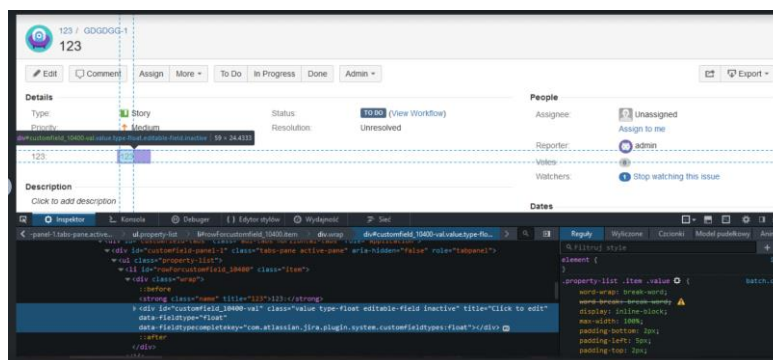


Рисунок 1.7 – Скріншот із інструкції по використанню додатку

1.2.4. Better Excel Exporter for Jira, Midori.

Better Excel Exporter - додаток розроблений для оптимізації процесів імпорту та експорту даних у Excel, можливості якого здебільшого зосереджені на поглиблених маніпуляціях з Excel форматом, навіть частково підтримкою VBA(Visual Basic for Applications). Цей додаток дозволяє експортувати різні типи даних з Jira безпосередньо в Excel з дотриманням правильної типізації даних, серед можливостей наявні: експорт задач з обраними або усіма полями (включаючи системні та користувацькі поля), батьківських і дочірніх задач, коментарів, вкладень, пов'язаних задач, версій, компонентів, історії змін, вбудованих робочих журналів та журналів Tempo Timesheets, даних Jira Software та Jira Service Management[5]. Підтримка більшості функцій Excel: форматування даних (чисел, дат), атрибути клітинок (кольори, рамки, вирівнювання, шрифти), використання формул Excel, побудова діаграм, сводних таблиць, підтримка Visual Basic (VBA) та інших інструментів Excel. Вибір даних для експорту здійснюється за допомогою стандартних фільтрів Jira, а основний інтерфейс розташований у модальному вікні.

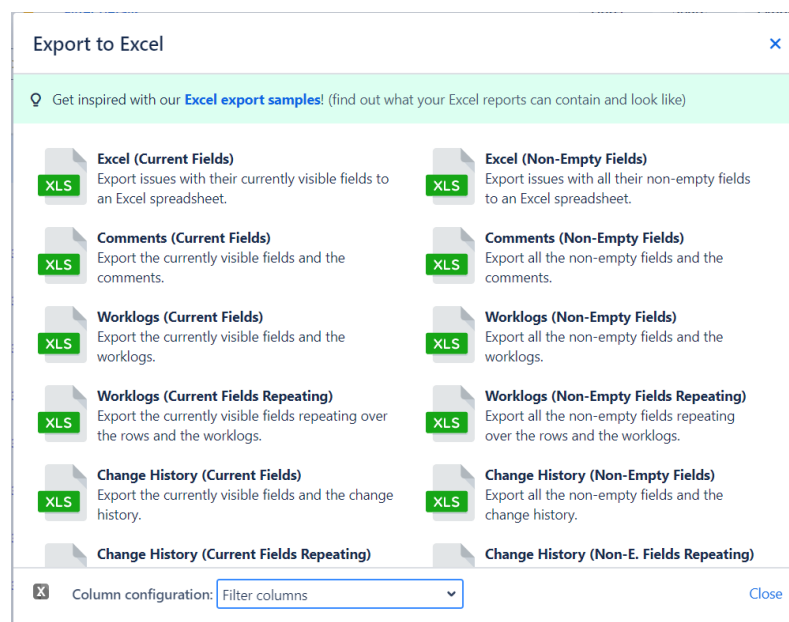


Рисунок 1.8 – Вікно вибору шаблону звіту

Основна незручність полягає у тому, що навіть для мінімальних змін у конфігурації звіту, необхідно користуватись інтерфейсом налаштувань що розташований окремо.



Рисунок 1.9 – Вікно налаштувань Better Excel Exporter.

1.2.5 Xporter Cloud для Jira, XBlend.

Xporter Cloud – це додаток, що забезпечує створення документів з використанням даних із завдань Jira. Додаток підтримує експорт таких елементів завдань, як поля (включаючи системні та користувацькі), посилання (зв'язані завдання), коментарі, робочі журнали, підзавдання, компоненти, зміни статусу завдання, прикріплені зображення, вкладення тощо [6].

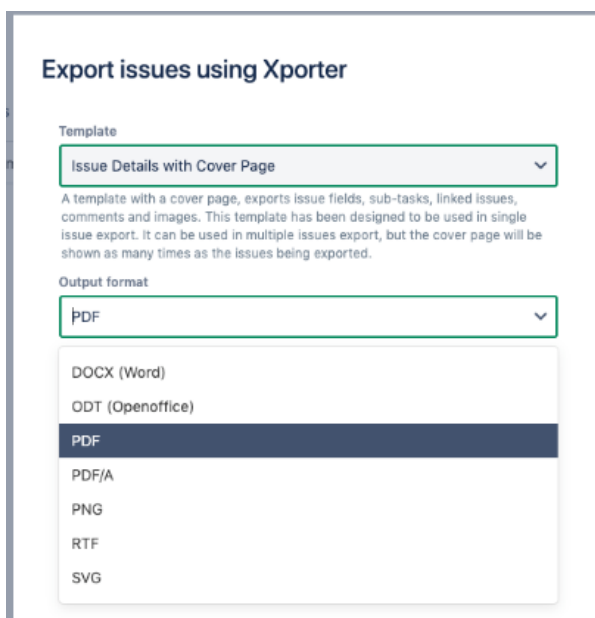


Рисунок 1.10 – Вікно експортування однієї задачі

Xporter Cloud дозволяє створювати різноманітні документи, зокрема індивідуальні журнали змін проєкту, документи про покращення продукту, виписки зі змін про випуск у власному форматі, вимоги до документів для зустрічей Scrum, листи із завдання на основі шаблону, документацію до програмного забезпечення. Документи можна генерувати на основі однієї або кількох задач.

Крім того, додаток дозволяє налаштовувати автоматичну генерацію звітів при зміні статусу завдань. Згенеровані документи можна отримувати електронною поштою або надсилати на файлові сервери, такі як FTP(File Transfer Protocol), FTPS(File Transfer Protocol Secure), SFTP(Secure File Transfer Protocol) або Confluence. Xporter доступний для версій Jira Server та Cloud і підтримує англійську, французьку, німецьку та іспанську мови. Додаток інтегрується з такими інструментами, як Smart Checklist, Issue Checklist для Jira, Gliffy Diagrams для Jira, Table Grid Editor, Tempo Timesheets та Xray Test Management для Jira. Одним із найбільших недоліків додатку є складність використання – налаштування шаблонів здійснюється за допомогою завантаження файлу потрібного формату, в якому на потрібних місцях будуть вказані айді для передачі інформації.

```

#{if (%${SubtasksCount}>0)}}
This issue has subtasks. Let's list them below:
#{for subtasks}


|                                |                                         |
|--------------------------------|-----------------------------------------|
| <b>Key</b>                     | \${Subtasks[n].Key}                     |
| <b>Summary</b>                 | \${Subtasks[n].Summary}                 |
| <b>AssigneeUserDisplayName</b> | \${Subtasks[n].AssigneeUserDisplayName} |


#{end}
#{end}

```

Рисунок 1.11 – Шаблон для експорту у .docx

Основним інтерфейсом виступає модальне вікно при налаштуванні експорту, також додаток має об'ємну сторінку конфігурацій.

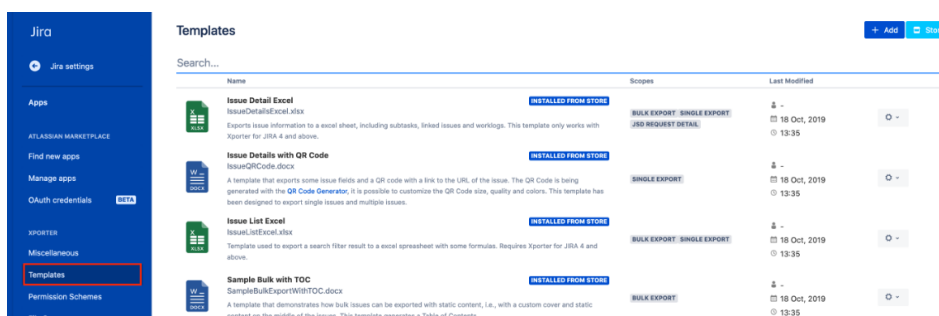
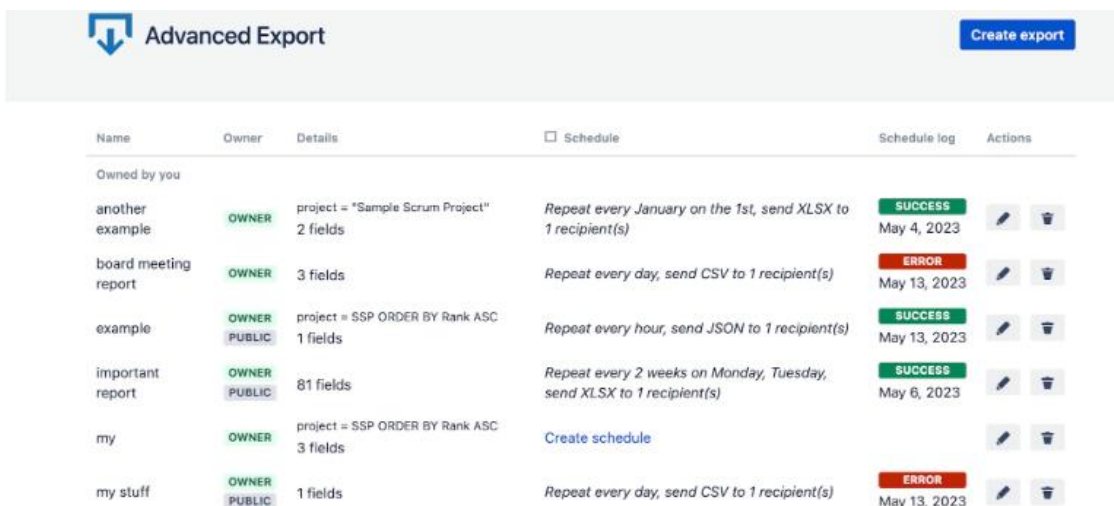


Рисунок 1.12 – Сторінка конфігурацій додатку

1.2.6. Advanced Export, Kanbanalytics

Advanced Export - додаток розроблений для полегшення та автоматизації експорту даних із Jira. Має два основні інтерфейси: панель створення звіту, панель керування шаблонами.



Name	Owner	Details	<input type="checkbox"/> Schedule	Schedule log	Actions
Owned by you					
another example	OWNER	project = "Sample Scrum Project" 2 fields	Repeat every January on the 1st, send XLSX to 1 recipient(s)	SUCCESS May 4, 2023	
board meeting report	OWNER	3 fields	Repeat every day, send CSV to 1 recipient(s)	ERROR May 13, 2023	
example	OWNER PUBLIC	project = SSP ORDER BY Rank ASC 1 fields	Repeat every hour, send JSON to 1 recipient(s)	SUCCESS May 13, 2023	
important report	OWNER PUBLIC	81 fields	Repeat every 2 weeks on Monday, Tuesday, send XLSX to 1 recipient(s)	SUCCESS May 6, 2023	
my	OWNER	project = SSP ORDER BY Rank ASC 3 fields	Create schedule		
my stuff	OWNER PUBLIC	1 fields	Repeat every day, send CSV to 1 recipient(s)	ERROR May 13, 2023	

Рисунок 1.13 - Менеджер звітів Advanced Export

Для вибору задач використовується JQL вираз - згенерований звіт буде містити не лише вибрані задачі, а ще й історію їх зміни, також можливо налаштувати які саме поля будуть відображатись у звіті. Додаток підтримує формати .csv, .xlsx, .json, та відображає попередній вигляд звіту під час налаштування[7]. Сформований звіт відправляється на: пошту, API(application programming interface), SFTP сервер або може бути завантажений за посиланням. Налаштувавши розклад для вказаного шаблону, можна автоматизувати звітування. У додатку присутня інформаційна панель для керування розкладами звітування, та реалізована функція обмеження доступу, для створення приватних звітів.

- Export issues: JQL для фільтрації задач. Ви можете використовувати будь-який фільтр задач Jira, який ви визначили раніше, натиснувши «From filter».
- Fields to export: набір полів для додавання як стовпців звіту. Деякі користувацькі поля можуть бути невідомі програмі, у такому випадку ви побачите попередження. Ви можете додати їх, але вони будуть представлені як звичайний текст.

- History export: якщо позначено, програма експортуватиме всі зміни, внесені до проблем, в окремі рядки. Він також додасть 3 додаткові стовпці:
 - As Of Date - дата зміни або поточна дата для рядка, що містить поточні значення
 - Changed Fields - список полів, які були змінені на цю дату
 - Change Author - користувач, який вніс зміни
- Date/Time Format: ви можете вибрати один із попередньо визначених форматів
- Time Difference: формат різниці в часі між 2 датами (використовується, наприклад, для стовпця «Час у цьому стані»)
 - CSV Separator: використовується лише для експорту CSV, ігнорується для інших форматів

The screenshot shows the 'Advanced Export' interface in JIRA. At the top, it says 'Advanced Export Dashboard / Edit Support tickets'. Below that, there's a search bar with the query 'project = "Sample Scrum Project" AND resolution = Unresolved ORDER BY priority DESC'. The 'Fields to export' section shows 'Issue Type', 'Priority', 'Project', and 'Status' selected. The 'History export' section is checked, indicating that historical changes will be included. The 'Date/Time Format' is set to '13/05/2023' and the 'Time Difference Format' is '2d 10h 30m'. The 'CSV Separator' is set to 'Comma'. A warning message states: 'Your browser timezone (CEST) differs from your JIRA settings (America/Detroit - EDT). All times exported in EDT timezone.' Below this, a preview table shows 10 out of 16 issues. The table has columns: Key, As Of Date, Time In This State, Changed Fields, Change Author, Created, Issue Type, Priority, Project, and Status. Two rows are visible, showing changes for issue SSP-14.

Key	As Of Date	Time In This State	Changed Fields	Change Author	Created	Issue Type	Priority	Project	Status
SSP-14	26/12/2019	1233d 22h 55m	Priority	Tomasz Kustrzynski	06/08/2017	Story	High	Sample Scrum Project	To Do
SSP-14	06/08/2017	871d 16h 8m			06/08/2017	Story	Medium	Sample Scrum Project	To Do

At the bottom, there are options to export to CSV, Excel, or JSON, along with 'Download' and 'Create schedule' buttons.

Рисунок 1.14 - Конструктор шаблону Advanced Export

1.2.7 Requirements Management for Jira, Ease Solutions Pte Ltd

Requirements Management – великий додаток, що має на меті розширення можливостей керування вимогами проекту у Jira. Requirements Management for Jira контролює процес від стадії початку розробки до випуску продукту. В рамках контролю вимог до проекту у додаток також входить покращена система експортування - Microsoft Excel Add-In[8]. На меті вона має замінити функціонал імпорту/експорту стандартної Jira для кращої взаємодії з основним додатком. На відміну від інших додатків продукт Ease Solutions працює із їх додатком для Excel, такий підхід дозволяє реалізувати окрім базових можливостей, функцію оновлення звіту прямо у Excel.

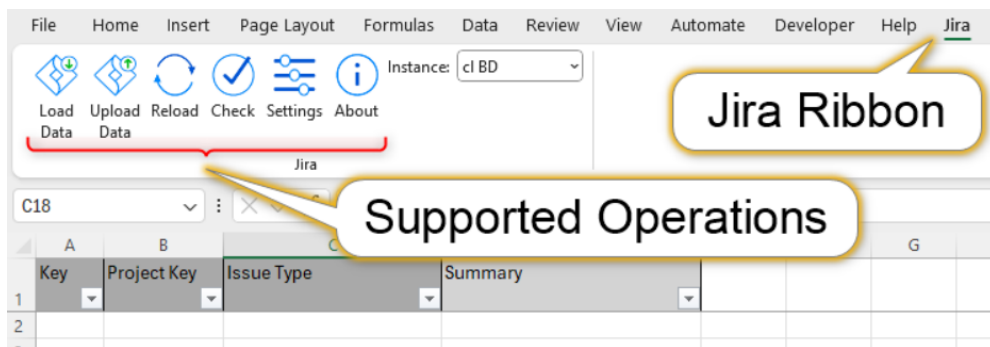


Рисунок 1.15 – Секція додатку у Excel

Microsoft Excel Add-In є більш примітивним порівняно із іншими додатками на ринку, і реалізує лише базові функції: експорт/імпорт з Excel, підтримка 6 основних полів задачі, можливість експортувати зв'язані задачі.

1.2.8 Інші додатки та порівняльна таблиця

Окремо можна виділити додатки які інтегрують Ексель із Jira, а не просто надають можливості експортування, серед них: Excel Online Integration for Microsoft Office 365, Mobility Stream - додаток що дозволяє експортувати задачі із Jira, налаштовуючі необхідні поля, у автоматично оновлювані таблиці, забезпечує створення таблиць із спільним доступом.

Таблиця 1.1 - Порівняльна таблиця можливостей додатків

Можливість	Advanced Export	Time in Status	Big Template	Exporter	Better Excel Exporter	Xporter Cloud	Requirements Management
Створення шаблонів звітів, і браузер для їх перегляду	Так	Так	Ні	Ні	Ні	Ні	Ні
Вибір полів для експорту	Так	Так	Так	Так	Ні	Так	Так
Формати .csv .json .xlsl	Так	Так	Так	Так	Ні	Так	Ні
Експорт історії полів	Так	Так	Так	Так	Так	Ні	Ні
Попередній перегляд звіту	Так	Так	Ні	Так	Ні	Ні	Ні
Кіль-кість змін у задачі для вибраних полів	Ні	Ні	Так	Ні	Ні	Ні	Ні
Звітування за графіком	Так	Ні	Ні	Так	Ні	Так	Ні
Експорт на зовнішні ресурси	Так	Так	Ні	Ні	Ні	Так	Так

Так як не один із додатків повною мірою не задовольняє усі наведені вимоги, доцільно розробити свій додаток. При розробці основні зусилля будуть зосереджені на функціях що дозволять включити до звіту додаткові дані для аналізу, при цьому зберігаючи необхідну гнучкість налаштувань та зручність використання.

Висновки до розділу:

У даному розділі було досліджено систему Atlassian Jira як популярний інструмент для оптимізації робочих процесів та керування проектами, було оглянуто його структуру, сильні та слабкі сторони.

Додатково було проведено детальний порівняльний аналіз інструментів, доступних користувачам цієї системи. Його результати виявили недосконалість наявних рішень, з чого було зроблено висновок про необхідність створення власного застосунку.

РОЗДІЛ II.

Проектування та реалізація застосунку для автоматизації процесу формування звітів у Jira з розпізнаванням табличних даних

2.1 Варіанти використання застосунку

Функціональні вимоги додатку:

Конструктор шаблону звіту - при створенні шаблону користувач зможе задати: JQL запит для вибору задач що увійдуть до звіту; стовпці та їх порядок у звіті; графік звітування; параметри автоматичного експорту за графіком; формат звіту на вибір .csv .json .xlsl. В якості стовпців у звіті можна буде обрати: поле задачі; поле задачі з історією зміни; лічильник, що буде рахувати зміни у полі або полях вибраних користувачем для кожної задачі. У конструкторі має бути передбачена можливість попереднього перегляду створюваного звіту.

Менеджер звітів - головне вікно інтерфейсу, в якому буде відображатись таблиця з уже створеними шаблонами, і кнопками для виклику основних можливостей: створювати, переглядати, редагувати, видаляти шаблони звітів (з можливістю групових операцій), змінювати розклад звітування та налаштування автоматичного експорту, переглядати журнал експортованих звітів.

Додаток буде працювати на платформі Jira Cloud останньої версії, з підтримкою браузерів Opera, Google Chrome. Додаток буде написаний на англійській мові, використовувати формат дат ISO(International Organization for Standardization). Додаток повинен працювати з максимальним часом відповіді 10 секунд використовуючи LTE(Long-Term Evolution) з'єднання через браузер Google Chrome. Час відповіді не повинен перевищувати 10 секунд: якщо у додатку збережено 100 шаблонів, якщо у звіті вказано 1000 задач. Обмеження щодо кількості і регулярності звітування встановлюються квотами сервісів JIRA, доступність і надійність додатку залежить від сервісів JIRA. Безпека буде забезпечена стандартними механізмами Jira, персональні дані будуть зберігатись і

оброблюватись на серверах Jira. На рисунку 2.1 наведена діаграма використання додатку.

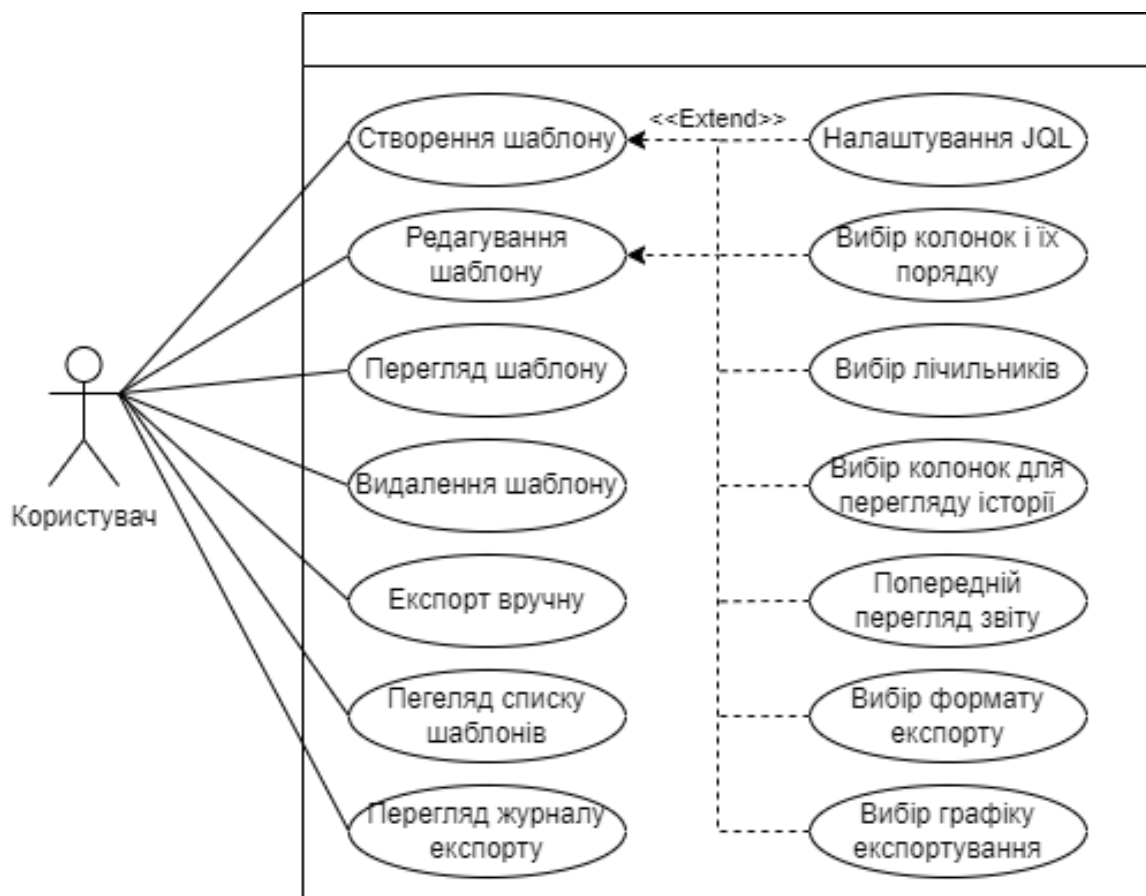


Рисунок 2.1 Діаграма використання

Користувач може використати додаток наступним чином: Він може створити шаблон для ознайомлення з інформацією попереднього перегляду, навіть не генеруючи повноцінний звіт; Користувач матиме можливість актуалізувати застарілий шаблон; Користувач може ознайомитись з інформацією, що міститься в шаблоні, наприклад після проходження певного терміну, або якщо інший користувач створив шаблон; Якщо шаблон більше не потрібний його можна видалити; Користувач може обрати зручний для нього спосіб експортування: кожного разу переглядаючи шаблон, експортувати вручну, або скористатися можливістю планування.

2.2 Архітектура застосунку

Конструктор звітів буде відповідати за налаштування об'єкту Template, і його дочірніх об'єктів. Також буде розроблено невеличкий функціональний компонент для редагування об'єкту Column.

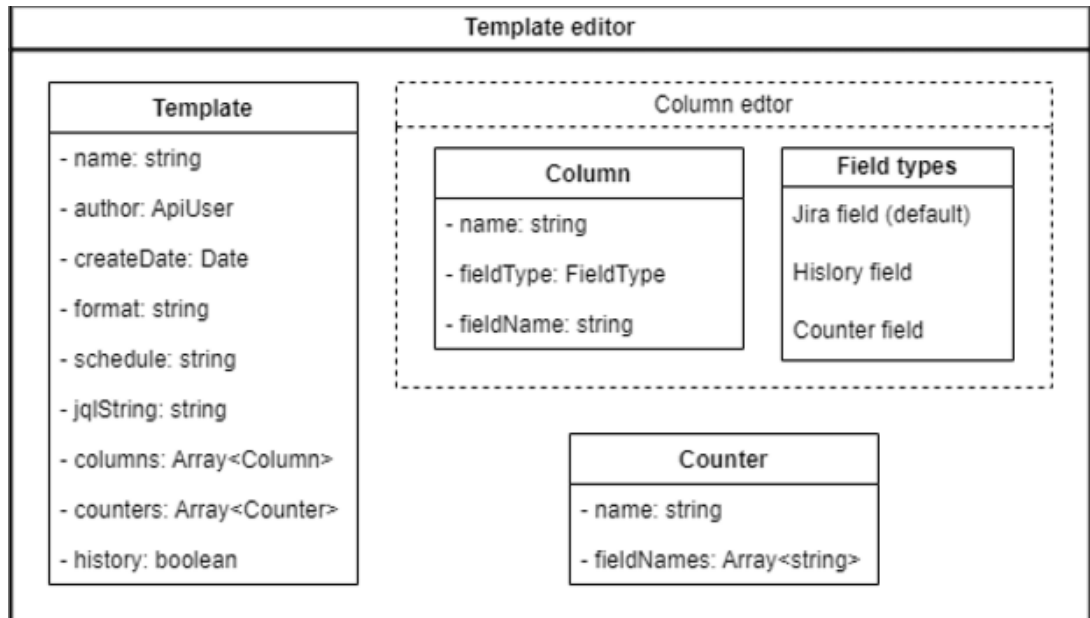


Рисунок 2.2 - Структура конструктора звітів

Об'єкт Counter буде редагуватись за допомогою звичайного Select елементу. Для забезпечення швидкодії, та належного рівня безпеки дані звітів не будуть зберігатись у додатку, а за їх обробку будуть відповідати наступні структури: при створенні або редагуванні шаблону додаток буде запитувати задачі у JIRA REST API. Відповідь міститиме об'єкт ApiSearchIssueResponse, який у свою чергу буде містити масив об'єктів ApiIssue, кожен такий об'єкт відповідає задачі Jira і містить всю її інформацію (за замовчуванням поле changelog у відповіді присутнім не буде). Опціонально, при налаштуванні шаблону можна додати історію змін задачі, тоді у об'єкті ApiIssue буде наявне поле changelog яке буде містити об'єкт IssueChangelog. У ньому поле histories містить масив з об'єктами ChangelogHistory кожен з яких відповідає події зміни задачі, містить як дату і автора зміни так і змінені значення полів - масив об'єктів HistoryItem. У

результуючому звіті будуть відображені лише значення обраних користувачем полів, та зміни у них, при необхідності.

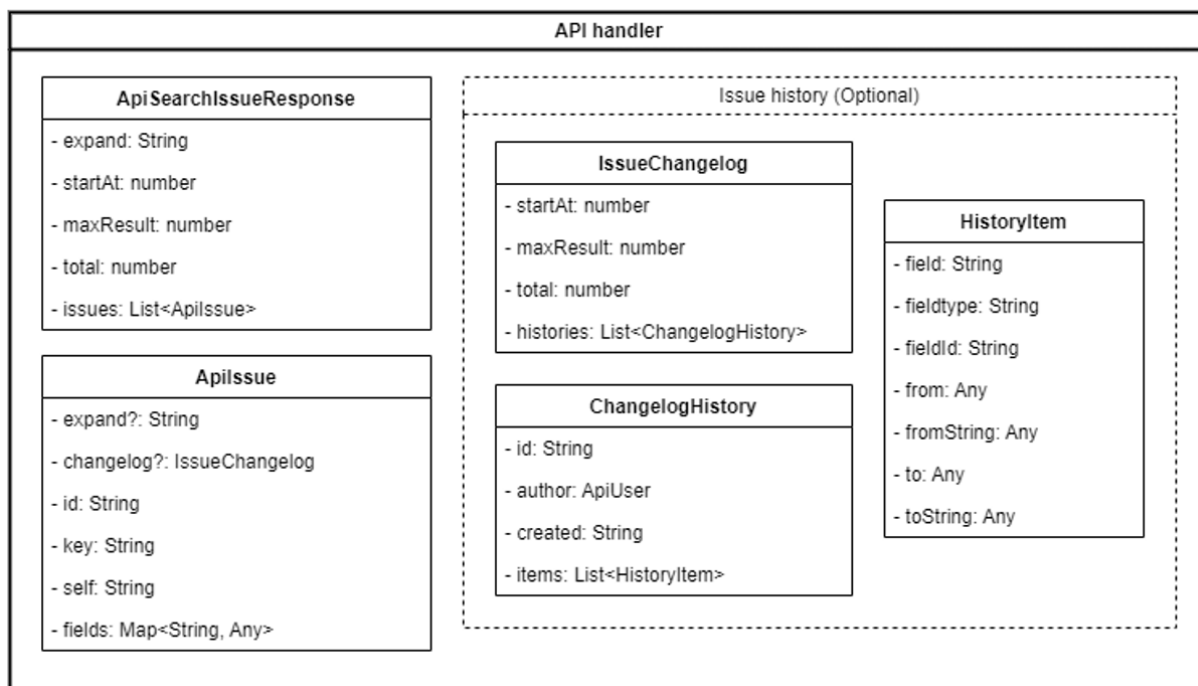


Рисунок 2.3 - Структура даних звіту

Для збереження шаблонів між сеансами роботи, та виклику автоматичних функцій шаблони будуть зберігатись на серверах Jira в форматі ключ - значення. Шаблони будуть зберігатись кожен під своїм ключем, а для пришвидшення завантаження сховище міститиме єдиний запис під фіксованим ключем в якому буде зберігатись список усіх шаблонів та їх ключів.

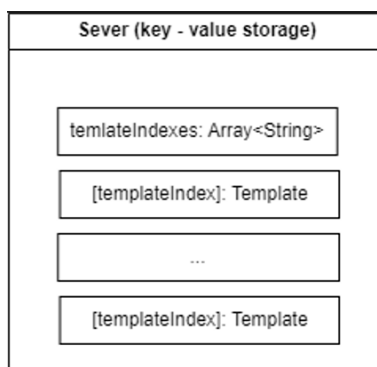


Рисунок 2.4 - Структура сховища

Деякі поля мають достатньо комплексну структуру (наприклад ApiUser), тому для їх коректної обробки були створені допоміжні структури.

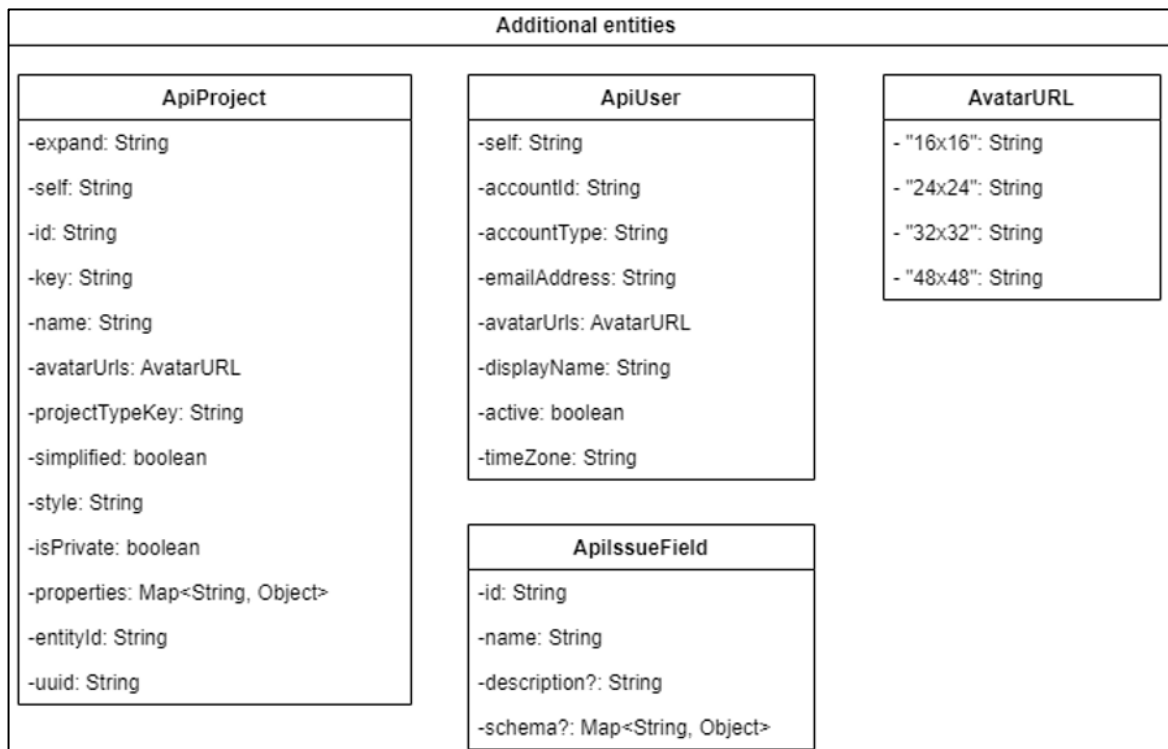


Рисунок 2.5 - Допоміжні структури

Менеджер звітів не має окремих структур для опису, так як не містить функцій редагування даних і відповідає лише за навігацію у додатку.

2.3 Програмна реалізація застосунку

Додаток буде написаний на мові TypeScript (v4.5) з використанням бібліотеки React (v18.2), в якості середовища виконання буде виступати модифікація Node.js (v20.1) – Atlassian Forge (v9.0). Код додатку буде організовано за стандартним шаблоном React. Розробка буде вестись у середовищі Visual Studio Code.

2.3.1. Структура додатку.

Структура збереження файлів додатку буде відповідати стандартній структурі побудови React додатків, та зображена на рисунку 2.6

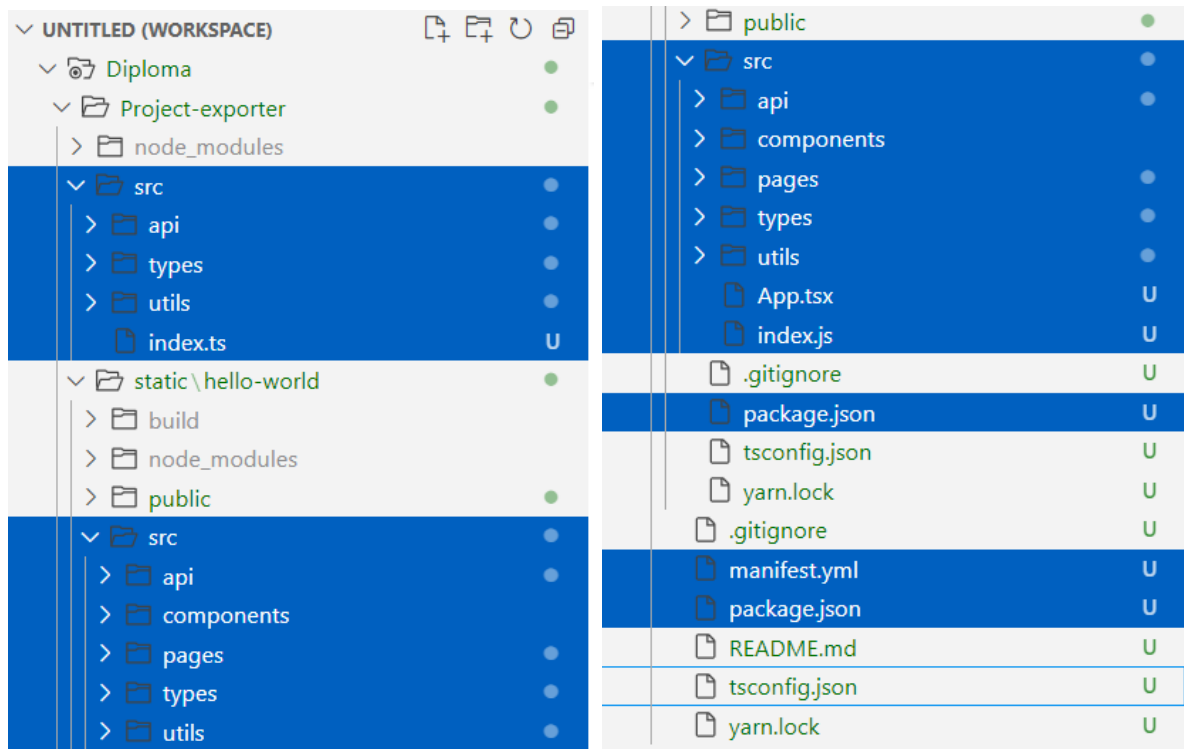


Рисунок 2.6 – Файлова структура додатку

У папці «./src» – буде розташована серверна частина додатку. «./src/api» – буде зберігати методи звернення до АПІ, «./src/types» – має у собі інтерфейси необхідні для роботи, «./src/index.ts» – буде містити реєстр усіх можливих звернень із клієнтської частини до серверної. У папці «./static/hello-world/src» розташована клієнтська частина додатку, де «./static/hello-world/src/components» - містить стандартні модулі, що будуть використані у декількох вікнах інтерфейсу, «./static/hello-world/src/pages» – міститиме код вікон інтерфейсу, «./static/hello-world/src/types» - інтерфейси необхідні для роботи, «./static/hello-world/src/utils» - допоміжні функції, «./static/hello-world/src/app.tsx» - головне вікно додатку.

2.3.2 Розгортання додатку Atlassian Forge.

Для розгортання додатку необхідно встановити Node.js, станом на 2024 рік платформа Atlassian підтримує версії 18.x або 20.x [9]. За допомогою Node Packet Manger необхідно встановити Forge CLI(Command Line Interface) командою «npm install -g @forge/cli». Далі необхідно створити аккаунт Atlassian, і на сторінці керування аккаунтом, створити «API token», його необхідно використати для

авторизації у Forge CLI за допомогою команди «forge login». Після авторизації необхідно скерувати термінал у директорію де буде створений додаток, і ввести команду «forge create». Консоль запропонує вам ввести назву додатку, обрати тип додатку: «Custom UI або UI Kit»(у роботі використано Custom UI(User Interface), так як Atlassian концентрує зусилля на його підтримці та позиціонує цей тип як найбільш сучасний). Далі необхідно обрати продукт для якого буде створений додаток, у нашому випадку це Jira Software (Jira у списку).

```
Creating an app in your current directory:

D:\Learn\Diploma

Press Ctrl+C to cancel.

Name your app. The app name can include dashes, spaces, and underscores.

? Enter a name for your app: 123

Start with a template. Each template contains the required files to be a valid app.

? Select a category: Custom UI
? Select a product:
  Show All
  Compass
  Confluence
> Jira
  Jira Service Management
```

Рисунок 2.7 – Результат виконання команди forge create

Далі необхідно перейти у директорію проекту, та налаштувати файл manifest.yaml. Архітектура створеного додатку майже повністю збігається зі стандартною схемою React, для розгортання за замовчуванням встановлено react-scripts, то ж для оновлення коду клієнтської частини необхідно перед зверненням до Forge CLI прописувати команду «npm run build». Також необхідно зазначити що будь-які команди до Forge CLI, необхідно виконувати у директорії верхнього рівня, і в ній же завжди має бути файл manifest.yaml, а команда «npm run build» повинна виконуватись у директорії де розташована лише клієнтська частина додатку. Для першого розгортання команди необхідно вводити у такій послідовності «npm run

build», «forge deploy», «forge install», при введені останньої команди консоль запитає назву проекту на який потрібно встановити додаток, перед встановленням необхідно впевнитись, що аккаунт в якому ви створили «API token» має дозвіл встановлювати додатки у проект [10].

Для подальшого оновлення коду у процесі розробки платформою Atlassian передбачено два механізми, ручний та автоматичний. Для оновлення вручну необхідно встановити у маніфесті флаг “runtime:name: nodejs18”, як зображено на рисунку 2.8, Та виконати послідовність команд «npm run build», «forge deploy».

```
1  app:
2  id: <your app id>
3  runtime:
4    name: nodejs18.x
```

Рисунок 2.8 – Розташування флагоу runtime у маніфесті.

При перевизначені деяких змінних у маніфесті може виникнути необхідність додатково ввести команду «forge install --upgrade», повідомлення про це може з’явитись після виконання «forge deploy».

Автоматичний варіант розгортання передбачає роботу із Docker, в такому режимі додаток працює із локальної машини, та автоматично оновлюється, що дозволяє зберегти час, проте робота в такому режимі потребує значних системних ресурсів.

2.3.3 Налаштування маніфесту.

Для побудови додатку на платформі Atlassian Forge, потрібно заповнити спеціальний файл manifest, він включає у себе три обов’язкові властивості найвищого рівня – modules, app, permission[11]. Структура маніфесту що використана у додатку наведена на рисунку 2.9.

```

modules:
  jira:globalPage:
    - key: project-exporter-hello-world-page
      resource: main
      layout: basic
      resolver:
        function: resolver
        title: Project exporter
      function:
        - key: resolver
          handler: index.handler
        - key: scheduler
          handler: index.schedulerCheck
      scheduledTrigger:
        - key: my-scheduled-trigger
          function: scheduler
          interval: hour
  resources:
    - key: main
      path: static/hello-world/build
  app:
    id: ari:cloud:ecosystem::app/16d36d11-17aa-40a1-8
    runtime:
      name: nodejs18.x
    permissions:
      content:
        styles:
          - unsafe-inline
        scripts:
          - unsafe-inline
          - unsafe-hashes
    scopes:
      - storage:app
      - read:jira-work
      - write:jira-work

```

Рисунок 2.9 – Структура маніфесту

Властивість `Modules` – характеризує основні інтерфейси програми, індивідуальні для кожного продукту Atlassian, тобто при розробці додатку для Jira Software, будуть використані модулі які не можна буде використати для Jira Confluence, або інших[12]. Додаток для продукту Jira Software, основні підтримувані модулі: Jira admin page – цей модуль розміщує сторінку у боковій стрічці вкладки додатків, гарно пасує для розташування налаштувань додатку, розміщення керівництва користувача, може містити підрозділи; група модулів Jira custom field – ці модулі слугують для створення користувацьких полів для задач, що стане у нагоді при розробці додатку який має додавати особливі властивості для групи задач; група модулів Jira dashboard gadget – дозволяє створити модуль що буде працювати на титульній сторінці Jira, зручний для виведення коротких звітів, діаграм або зображень, що стосуються всіх проектів. Jira global page – розташовується у вкладці App як окрема сторінка, це зручно у випадку коли додаток реалізує функціонал відсторонений від стандартних можливостей платформи, та не прив’язаний до проекту, або задачі; група модулів Jira issue – реалізують можливість модифікації у режимі перегляду задачі, дозволяють додати користувацькі опції у різні меню цього режиму перегляду, з чого випливає що використання цих модулів доцільне якщо функціонал додатку стосується обробки окремих задач, або тісно зв’язаний із їх структурою; група модулів Jira project page – використовується додатками, функціонал яких пов’язаний з окремим проектом,

модулі цієї групи дозволяють додавати користувацькі опції та сторінки до бокової стрічки режиму огляду проекту, як приклад використання можна навести додаток що буде автоматично збирати статистику проекту та відображати панель гаджетів що можна налаштувати.

```

1  modules:
2    | jira:globalPage:
3    |   - key: project-exporter-hello-world-page
4    |     resource: main
5    |     layout: basic
6    |     resolver:
7    |       | function: resolver
8    |       | title: Project exporter

```

Рисунок 2.10 – Налаштування модулів для додатку

Виходячи із цієї інформації для додатку доцільно використати модуль Jira global page, так як функціонал не пов'язаний з окремими проектами, не має гаджетів для візуалізації, та не додає функціоналу при редагуванні окремих задач. У роботі модуль використано з налаштуваннями, що зображені на рисунку 2.10, де поле key – ідентифікатор модулю для звернення, title – назва, яка відображається у меню вибору додатків, resource – посилання на глобальну властивість Resources, що містить адресу розташування коду модуля, layout – що може набути значень native, blank, basic, і відповідає за макет глобальної сторінки, визначає чи відображається сторінка з елементами керування за замовчуванням (native), розміщує всю область перегляду з полем ліворуч і елементами навігації (basic для UI Kit), чи залишиться порожньою, що дозволяє повністю налаштувати (blank для Custom UI), resolver – вказує посилання на функцію що буде відповідати за виклики серверних функцій з клієнтської частини додатку.

Також у модулі маніфесту входить ряд загальних модулів, вони працюють у додатках для всіх продуктів Atlassian. Модуль Consumer – дозволяє прив'язати resolver до черги, та визначити спосіб реагування на події. Приклад оформлення цього модулю зображено на ілюстрації 2.11.


```

1 import Resolver from "@forge/resolver";
2 const resolver = new Resolver();
3
4 resolver.define("event-listener", async ({ payload, context }) => {
5     // process the event
6 });
7
8 export const handler = resolver.getDefinitions();

```

```

1 modules:
2   consumer:
3     - key: queue-consumer
4       # Name of the queue for which this consumer will be invoked
5       queue: queue-name
6     resolver:
7       function: consumer-function
8       # resolver function to be called with payload
9       method: event-listener

```

Рисунок 2.11 – приклад визначення модулю Consumer

Модуль Function – модуль який містить ключ функції, і адресу коду. Слугує для індексації, та збереження посилань серверної сторони програми, тобто resolver спочатку записується тут, а потім при необхідності звернутись до нього використовується ключ. В роботі цей модуль заповнено згідно з рисунком 2.12.

```

9   | function:
10  |   - key: resolver
11  |     handler: index.handler
12  |   - key: scheduler
13  |     handler: index.schedulerCheck

```

Рисунок 2.12 – Приклад оформлення модулю Function

Модуль Scheduled trigger – дозволяє встановити функцію що буде викликатись циклічно, із заданим інтервалом. Він починає працювати приблизно через 5 хвилин після розгортання програми, і далі виконується згідно вказаного інтервалу, запланувати виконання можна на кожну годину, кожен день, або раз на тиждень. В роботі використано тригер з погодинним інтервалом, для реалізації автоматичного експорту. Спосіб створення тригера зображено на малюнку 2.13.

```

14     |   |   |   |   | scheduledTrigger:
15     |   |   |   |   |     - key: my-scheduled-trigger
16     |   |   |   |   |     function: scheduler
17     |   |   |   |   |     interval: hour

```

Рисунок 2.13 – Оформлення модулю Scheduled trigger

Модуль Trigger – дозволяє викликати функцію після вказаної події у екосистемі Jira. Інсталювання додатку, події у проєкті, спрацювання Scheduled trigger, та інші події можуть бути обрані як ціль Trigger. Приклад використання зображено на малюнку 2.14.

```

modules:
  trigger:
    - key: ignore-self-trigger
      function: main
      events:
        - avi:jira:updated:issue
      filter:
        ignoreSelf: true

```

Рисунок 2.14 - Приклад використання Trigger

Модуль Web trigger – ще один вид тригерів, як і минулі слугує для «непрямого» виклику функції. Цей модуль спрацює при отриманні HTTP(HyperText Transfer Protocol) запиту, щоб викликати його, спочатку необхідно згенерувати URL(Uniform Resource Locator) для чого використовується команда forge webtrigger. Згенероване посилання не буде містити засобів аутентифікації, тож може бути викликане будь-ким. При необхідності до заголовку запиту можна додати перевірку токена. Цей модуль був використаний у роботі для тестування функції запланованих, що дозволило зберегти час, адже мінімальний інтервал «запланованого тригера» дорівнює годині. Для створення модулю необхідно описати структуру подібну до Scheduled trigger, опустивши задання інтервалу.

Resources - наступний атрибут, що оголошується на верхньому рівні маніфесту, подібно до модулю Function, його роль індексувати і зберігати

посилання. Тут необхідно вказати шлях до файлів які визначають користувацький інтерфейс (HTML, Cascading Style Sheets, JavaScript файли, картинки, інше), і вказати для кожної адреси ключ, на який потім можна буде посилатись у інших атрибутах маніфесту. Приклад оформлення і використання цього атрибуту наведено на рисунку 2.15

```

1  modules:
2    jira:issuePanel:
3      - key: hello-world-panel
4        title: Custom UI App
5        description: A Forge app with resources
6        resource: my-resource-1 # link to the resources listed below
7  app:
8    id: "<your app id>"
9  resources: # list below the resource entries for your app
10 - key: my-resource-1
11   path: relative/path/to/resource/one/directory

```

Рис. 2.15 – Приклад оформлення Resources

App – обов’язковий атрибут верхнього рівня, містить основні параметри конфігурації додатку forge. Більшість опцій в ньому заповнюються автоматично при виконанні команди `forge create`, з них: `id` – визначає унікальний ідентифікатор прив’язаний до додатку, генерується автоматично; `runtime` – визначає середовище виконання серверного коду, може набувати значень `nodejs18.x` або `sandbox`, необхідно зазначити що `sandbox` визнаний як `legacy version`, і не рекомендується для використання, у нових версіях Forge CLI встановлюється автоматично.

Permissions – останній обов’язковий атрибут що задається у маніфесті, властивості вказані у цьому атрибуті визначають рівень доступу додатку до сторонніх ресурсів. Властивість `Scopes` – визначає список необхідних для додатку можливостей протоколу OAuth 2.0. У додатку вони оголошені для використання Jira Rest API. `Scopes` може бути оголошено як вказано на малюнку 2.16.

```

1  permissions:
2    scopes:
3      - 'read:confluence-content.summary'
4      - 'write:jira-work'

```

```

1  permissions:
2    scopes: []

```

Рисунок 2.16 – Приклади оформлення Scopes

Властивість Content – визначає опції Content Security Policy, необхідні для роботи додатка. Вона розбита на дві секції scripts та styles, приклад оформлення зображено на малюнку 2.17.

```

25  permissions:
26    content:
27      styles:
28        - unsafe-inline
29      scripts:
30        - unsafe-inline
31        - unsafe-hashes

```

Рисунок 2.17 – Оформлення Content

Властивість External – визначає список зовнішніх ресурсів, з якими додаток зможе працювати, при чому цей список визначається як для клієнтської та серверної частин додатку, так і для функцій Forge, таких як resolver. Властивість поділена на наступні секції: fetch – дозволяє вказати ресурси з якими будуть спілкуватись функції клієнтської або серверної сторони, для чого є окремі опції client, backend; Fonts – зовнішні шрифти; Styles – зовнішні стилі, frames – зовнішні iframe, images – зовнішні ілюстрації, media – зовнішні відео, scripts – зовнішні скрипти. Ресурси необхідно вказувати згідно до стандарту CSP: script-src, приклад наведено на рисунку 2.18.

```

1  permissions:
2    external:
3      styles:
4        - 'https://www.example-dev.com/styleSheet.css'

```

```

1  permissions:
2    external:
3      fetch:
4        backend:
5          - '*.example-dev.com'

```

```

1  permissions:
2    external:
3      frames:
4        - 'https://www.example-dev.com/embed/page'

```

Рисунок 2.18 – Приклади оформлення External

2.3.5 Jira Rest API.

Для отримання інформації із сайту Jira користувача додаток буде звертатись до спеціалізованого API. Jira REST API - це набір ресурсів запропонованих компанією Atlassian для створення додатків, сценаріїв взаємодії з Jira або розробки будь-якого іншого типу інтеграцій. Для виклику доступні версії 2 та 3, вони пропонують однаковий перелік операцій, але третя версія знаходиться у стані розробки і може змінюватись[13]. Основна перевага третьої версії API – підтримка Atlassian Document Format у різних полях, та більш детальна документація.

Під час виклику API, з додатку Forge автентифікація проводиться автоматично, використовуючи scopes з маніфест файлу для отримання списку дозволів. В документації для кожного API вказано список необхідних для його роботи scopes. Для оптимізації використання інтернет трафіку, та покращення швидкодії - операції API які оброблюють велику кількість інформації, організовані так, що за замовчуванням повертають лише частину даних. Для розширення

відповіді використовується параметр запити «expand» - після формули запити необхідно написати «?expand={об'єкт для розширення}», при необхідності розширити одразу декілька об'єктів, їх можна писати через кому. Зразком використання може слугувати запит наведений на малюнку 2.19.

```
GET issue/JRACLOUD-34423?expand=names,renderedFields
```

Рис. 2.19 – Зразок використання expand

Ще одним механізмом оптимізації слугує пагінація, всі операції що повертають велику кількість об'єктів обернені у JSON структуру яка містить дані про систему сторінок: з якого елемента починається відповідь, максимальна кількість елементів у запиті, всього можливих елементів, чи входить останній елемент до цієї сторінки. Зображення цієї структури наведено на малюнку 2.20 . Для окремих операцій також можна вказати порядок результатів, спосіб виклику залежить від операції.

```

1  {
2      "startAt" : 0,
3      "maxResults" : 10,
4      "total": 200,
5      "isLast": false,
6      "values": [
7          { /* result 0 */ },
8          { /* result 1 */ },
9          { /* result 2 */ }
10     ]
11 }
```

Рисунок 2.20 – Зразок структури сторінок

У інструментарії Atlassian існує ряд спеціальних заголовків, що визначають важливу метадату для запитів та відповідей: X-Atlassian-Token: no-check – має бути включений при виконанні запити для обробки багато-частинних форм, інакше

запит буде заблоковано; X-Force-Accept-Language – можна використати під час запиту, щоб отримати відповідь на конкретній мові, за замовчуванням мова відповіді обирається в залежності від налаштувань аккаунту, або сайту Jira; X-AccountId – може повернутись із відповіддю, і буде містити айді аккаунта питаючого користувача.

Для операцій з дуже тривалим часом виконання передбачена можливість асинхронного запиту. Вдалий виклик такої операції поверне код 303, а також адресу запланованої задачі в заголовку Location, подальша обробка результатів запиту проходить як обробка результатів асинхронної події. Необхідно зауважити що при обробці деяких запитів одночасно послідовність виконання не гарантується, то при необхідності строго впорядкування слід починати наступну операцію лише після завершення попередньої.

Більшість операцій потребують наявності у того хто викликає необхідних повноважень, однак існують ті для яких можна забезпечити анонімний доступ. Потрібно зауважити, що операції можна викликати від імені користувача, або від імені додатку, дає можливість більш гнучкого налаштування, наприклад якщо для виклику операції необхідні повноваження, наявність яких у інших користувачів небажана з причин безпеки, можна надати повноваження лише додатку, та організувати виклик від його імені.

Загалом значення статусів операцій співвідноситься із стандартом статусів відповідей HTTP, але в деяких випадках разом із помилкою може повертатись тіло запиту, із деталізацією помилки. API можна викликати і з клієнтської частини і з серверної частин додатку, але із клієнтської частини виклик завжди буде проходити від імені користувача, а з серверної метод виклику можливо обрати. Різницю способів виклику тієї самої операції можна побачити на малюнку 2.21, де перший фрагмент коду виконується на серверній частині, і пропонує аргумент «.asUser()», а другий - виконується на клієнтській частині, і не має подібного аргументу.

```

const res : APIResponse = await api
  .asUser()
  .requestJira(url: route`/rest/api/3/jql/parse?validation=strict`, init: {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(value: bodyData),
  });

return await res.json();

```

```

const res : Response = await requestJira(restPath: `/rest/api/3/jql/parse?validation=strict`,
  method: "POST",
  headers: {
    Accept: "application/json",
    "Content-Type": "application/json",
  },
  body: JSON.stringify(value: bodyData),
});

return await res.json();

```

Рисунок 2.21 – Різниця у виклику API з різних частин додатку.

Першою операцією, що була використана в роботі була операція розпізнавання JQL виразу. У додатку - виконує роль перевірки введеного JQL виразу, і повертає розгорнутий опис помилки. Коротку документацію для цієї операції зображено на малюнку 2.22[14], а реалізацію цього запиту у додатку – на малюнку 2.23.

POST Parse JQL query EXPERIMENTAL

Parses and validates JQL queries.

Validation is performed in context of the current user.

This operation can be accessed anonymously.

Permissions required: None.

Data Security Policy: Not exempt from app access rules

Scopes

Connect app scope required: READ

OAuth 2.0 scopes required:

Classic **RECOMMENDED:** read:jira-work

Granular: read:field:jira, validate:jql:jira, read:jql:jira

Request

Request parameters

validation string **REQUIRED**

Request body application/json

queries array<string> **REQUIRED**

```

POST /rest/api/3/jql/parse
Forge curl Node.js Java Python PHP
1 // This sample uses Atlassian Forge
2 // https://developer.atlassian.com/platform/forge/
3 import api, { route } from "@forge/api";
4
5 var bodyData = `{
6   "queries": [
7     "summary ~ test AND (labels in (urgent, blocker) OR la
8     "issue.property[\"spaces here\"].value in (\"Service r
9     "invalid query",
10    "summary = test",
11    "summary in test",
12    "project = INVALID",
13    "universe = 42"
14  ]
15 `};
16
17 const response = await api.asUser().requestJira(route`/res
18   method: 'POST',
19   headers: {
20     'Accept': 'application/json',
21     'Content-Type': 'application/json'
22   },
23   body: bodyData
24 `});
25
26 console.log(`Response: ${response.status} ${response.statu
27 console.log(await response.json());

```

Рисунок 2.22 – Початок документації Parse JQL


```

1 import api, { route, storage } from "@forge/api";
2
3 export async function validateJQL(request: string) {
4   const bodyData = {
5     queries: [request],
6   };
7
8   const res = await api
9     .asUser()
10    .requestJira(route`/rest/api/3/jql/parse?validation=strict`, {
11      method: "POST",
12      headers: {
13        Accept: "application/json",
14        "Content-Type": "application/json",
15      },
16      body: JSON.stringify(bodyData),
17    });
18
19 return await res.json();

```

Рисунок 2.23 –Реалізація функції обробки Parse JQL

Другою операцією використаною у роботі став запит на прикріплення файлу до задачі. Використовується як метод для запису звітів при автоматичному експортуванні. Зображення документації та реалізації наведено на рисунках 2.24[15], та 2.25 відповідно.

POST Add attachment

Tip: Use a client library. Many client libraries have classes for handling multipart POST operations. For example, in Java, the Apache HTTP Components library provides a [MultiPartEntity](#) class for multipart POST operations.

This operation can be accessed anonymously.

Permissions required:

- [Browse Projects and Create attachments project permission](#) for the project that the issue is in.
- If [issue-level security](#) is configured, issue-level security permission to view the issue.

Data Security Policy: Not exempt from app access rules

Scopes

Connect app scope required: WRITE

OAuth 2.0 scopes required:

Classic RECOMMENDED: write:jira-work

Granular: read:user:jira, write:attachment:jira, read:attachment:jira, read:avatar:jira

Request

Path parameters

issueIdOrKey string **REQUIRED**

Request body multipart/form-data Expand all

POST /rest/api/3/issue/{issueIdOrKey}/attachments

Forge curl Node.js Java Python PHP

```

1 // This sample uses Atlassian Forge
2 // https://developer.atlassian.com/platform/forge/
3 import api, { route } from "@forge/api";
4
5 const response = await api.asUser().requestJira(route`/res
6   method: 'POST',
7   headers: {
8     'Accept': 'application/json'
9   }
10  });
11
12 console.log(`Response: ${response.status} ${response.statu
13 console.log(await response.json());

```

200 Response

```

1 [
2   {
3     "author": {
4       "accountId": "5b10a2844c20165700ede21g",
5       "active": true,
6       "avatarUrls": {
7         "16x16": "https://avatar-management--avatars.serv
8         "24x24": "https://avatar-management--avatars.serv
9         "32x32": "https://avatar-management--avatars.serv
10        "48x48": "https://avatar-management--avatars.serv
11      },
12      "displayName": "Mia Krystof",
13      "emailAddress": "mia@example.com",
14      "self": "https://your-domain.atlassian.net/rest/api

```

Рисунок 2.24 – Фрагмент документації Add Attachment

```

83   try {
84     const response : APIResponse = await api
85       .asApp()
86       .requestJira(url: route`/rest/api/3/issue/${issueKey}/attachments`, init: {
87         method: "POST",
88         body: formData,
89         headers: {
90           Accept: "application/json",
91           "X-Atlassian-Token": "no-check",
92         },
93       });
94
95     console.log(message: `Response: ${response.status} ${response.statusText}`);
96     const res : any = await response.json();
97     console.log(message: "Attachment Res", ...optionalParams: res);
98     return res;
99   } catch (error) {
100    console.log(message: "Attachment error", ...optionalParams: error);
101
102    return error;
103  }
104 };

```

Рисунок 2.25 – Реалізація функції Add Attachment

Третьою операцією використаною в роботі став запит на задачі що відповідають фільтру. У роботі операція виконується при генерації попереднього перегляду, та при експорті звітів вручну і автоматично. В залежності від налаштувань звіту операція може бути викликана з флагом `expand=history`. Документація рисунок 2.26[16], реалізація рисунок 2.27.

POST Search for issues using JQL (POST)

Searches for issues using [JQL](#).

There is a [GET](#) version of this resource that can be used for smaller JQL query expressions.

This operation can be accessed anonymously.

Permissions required: Issues are included in the response where the user has:

- [Browse projects project permission](#) for the project containing the issue.
- If [issue-level security](#) is configured, issue-level security permission to view the issue.

Data Security Policy: Not exempt from app access rules

Scopes

Connect app scope required: READ

OAuth 2.0 scopes required:

Classic **RECOMMENDED:** `read:jira-work`

Granular: `read:issue-details:jira`, `read:field.default-value:jira`, `read:field.option:jira`, `read:field:jira`, `read:group:jira`

Request

Request body application/json Expand all

A JSON object containing the search request.

expand array<string>

POST /rest/api/3/search

Forge curl Node.js Java Python PHP

```

1 // This sample uses Atlassian Forge
2 // https://developer.atlassian.com/platform/forge/
3 import api, { route } from "@forge/api";
4
5 var bodyData = `{
6   "expand": [
7     "names",
8     "schema",
9     "operations"
10  ],
11  "fields": [
12    "summary",
13    "status",
14    "assignee"
15  ],
16  "fieldsByKeys": false,
17  "jql": "project = HSP",
18  "maxResults": 15,
19  "startAt": 0
20 `};
21
22 const response = await api.asUser().requestJira(route`/r
23   method: 'POST',
24   headers: {
25     'Accept': 'application/json',
26     'Content-Type': 'application/json'
27   },
28   body: bodyData
29 });
30

```

Рисунок 2.26 – Документація Search issues by JQL

```

23     const bodyData: BodyData = {
24       jql: searchJQL,
25       maxResults: maxResultsInit,
26       startAt,
27       expand: ["changelog"],
28     };
29
30     console.log(message: "bodyData", ...optionalParams: bodyData);
31     const res: APIResponse = await api.asApp().requestJira(url: route`/rest/api/3/search`, init: {
32       method: "POST",
33       headers: {
34         Accept: "application/json",
35         "Content-Type": "application/json",
36       },
37       body: JSON.stringify(value: bodyData),
38     });
39
40     const result: ApiSearchIssueResponse = await res.json();

```

Рисунок 2.27 –Реалізація Search issues by JQL

2.4 Засоби тестування та середовища.

Після розгляду тонкощів створення додатку, необхідно розглянути середовища розгортання можливості тестування, та подальшого супроводження додатку. При виконанні команди `forge create`, Forge CLI одразу створює три середовища `development` – середовище розробки, `staging` – середовище постановки, `production` – середовище продукту.

Середовища визначають місце де розгорнутий додаток, із середовища його можна встановити на сайт Atlassian командою `forge install`. За замовчуванням `forge CLI` завжди буде розгортатись у середовищі розробки, використовуючи флаг «`–environment`» можна вказати інше середовище. У середовищах розробки, та постановки назва додатку завжди буде мати відповідний маркер, на відміну від середовища продукту. Середовище розробки – як можна здогадатись із назви є основним середовищем в якому додаток буде перебувати під час розробки, для нього працюють команди `forge tunnel`, `forge logs`, але нема можливості переглянути `scopes` розгорнутого додатка. Середовище постановки – підходить для збереження стабільних версій програми, не може виконувати команду `forge tunnel`, та додаток у цьому середовищі не відображає `scopes`. Середовище продукту – підходить для продукту готового до використання, воно не дозволяє виконувати команди `forge`

tunnel, forge logs, проте відображає scopes, що дозволяє користувачу зрозуміти до яких даних він надає доступ, встановлюючи програму на свій сайт Jira. Схему використання середовищ для розробки і підтримки додатку рекомендовану Atlassian зображена на рисунку 2.28.

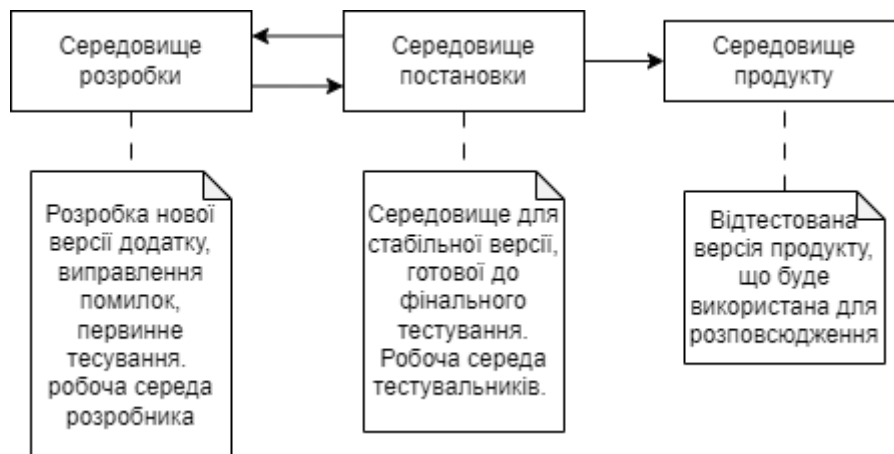


Рисунок 2.28 –Схема використання середовищ

Також Forge CLI підтримує використання змінних прив'язаних до середовища. Переглянути їх можна командою «forge variables list», а оголосити - командою «forge variables set MY_KEY my-value» де MY_KEY – ключ значення, а my-value - значення. Додатково можна захистити змінну від перегляду, додавши при оголошенні флаг --encrypt , очистити змінну можна викликавши «forge variables unset MY_KEY», а для зчитування у додатку використовується команда «process.env.MY_KEY». Для кожного середовища змінні будуть унікальні тож для оголошення змінної для середовищ постановки та продукту потрібно використати флаг –environment.

Кожного разу розгортаючи додаток ви створюєте нову версію у відповідному середовищі, номер версії генерується автоматично, і його неможливо змінити вручну. Forge CLI показує цифровий варіант версії лише адміністраторам сайтів у спеціальному меню (Рисунок 2.29), для всіх останніх вона буде відображатись як «остання» або «застаріла». В залежності від значущості змін, можливі дві поведінки: якщо маніфест був змінений, оновлення вважається суттєвим, а для його встановлення необхідний окремий дозвіл адміністратора; у випадку якщо зміни не

торкались маніфесту, оновлення вважається незначним і встановлюється автоматично. Приклад зображення версій з точки зору розробника для різних середовищ зображено на рисунку 2.30

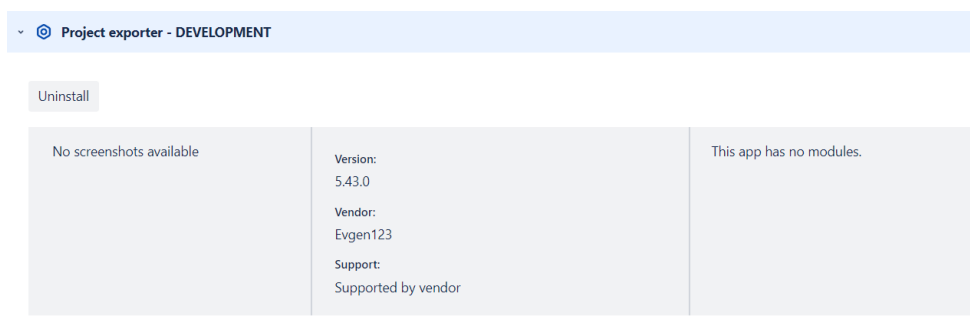


Рисунок 2.29 – Зображення спеціального меню для перегляду версії

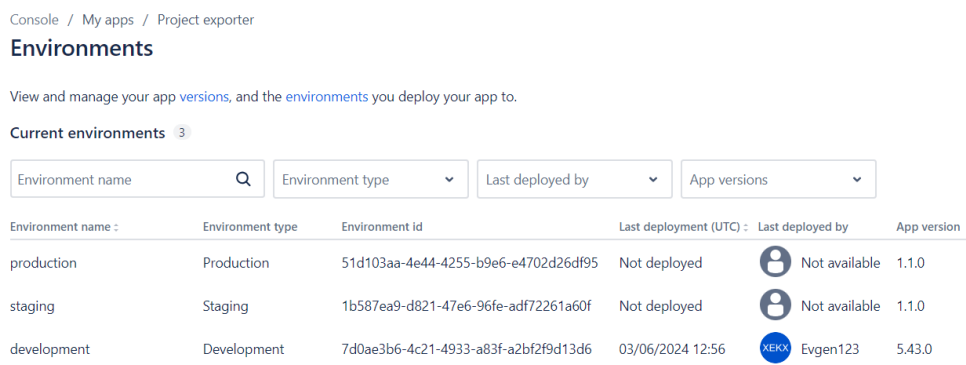


Рисунок 2.30 – Приклад роботи версій для різних середовищ

Тему тестування додатку доцільно поділити за частинами. Для клієнтської сторони використовуються здебільшого стандартні інструменти веб розробника, з одним уточненням, Forge CLI відображає сайт за допомогою iframe, через що ряд інструментів не працюють. Для серверної частини використовується спеціальна консоль розробника від Atlassian, вона доступна за адресою: «[developer.atlassian.com /console/](https://developer.atlassian.com/console/)». Консоль дозволяє переглянути загальні відомості про додаток такі як id, кількість середовищ розробки, в яких середовищах розгорнуто, кількість сайтів де встановлено, стан розповсюдження, дозволи додатку, та аккаунти яким дозволено керувати додатком[17]. На малюнку 2.31 наведено інтерфейс програми.

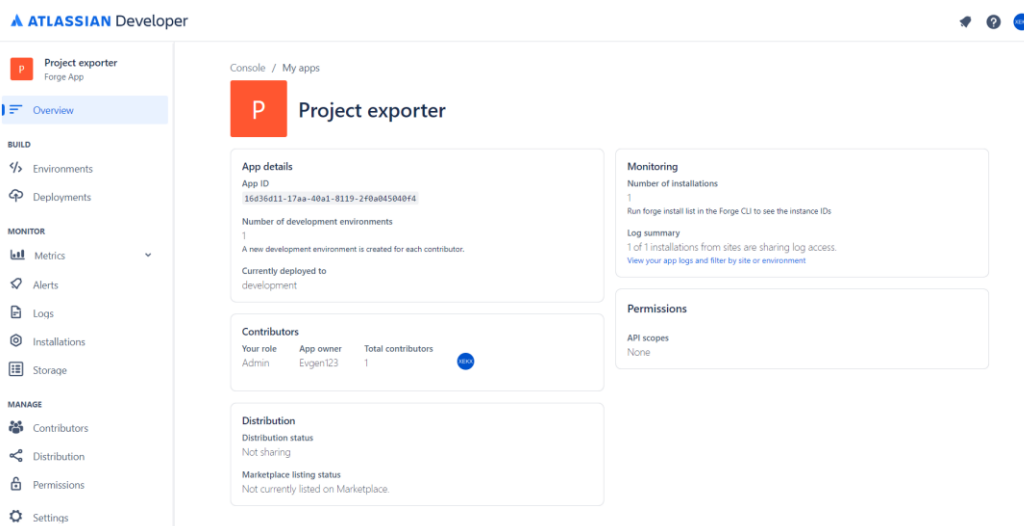


Рисунок 2.31 – Вигляд інтерфейсу консолі Atlassian Developer

Більш детально можна переглянути інформацію про середовища і версії додатку, журнал розгортання додатку (у тому числі розгортання іншими розробниками), показники і статистику викликів серверної сторони додатку та АПІ, журнал серверної частини додатку(його можна переглянути лише за допомогою цієї консолі та команди `forge logs`), список сайтів де встановлено додаток, та сховище додатку(можна відфільтрувати за сайтом, чи за середовищем

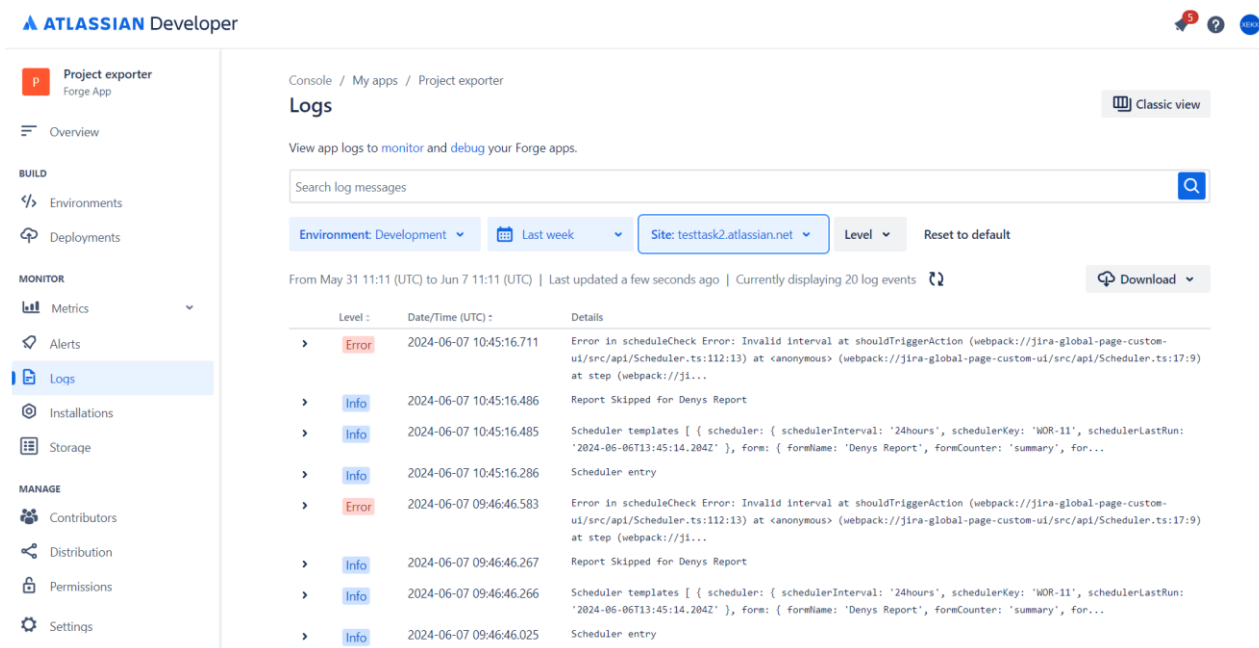


Рисунок 2.32 – Вигляд журналу Forge CLI

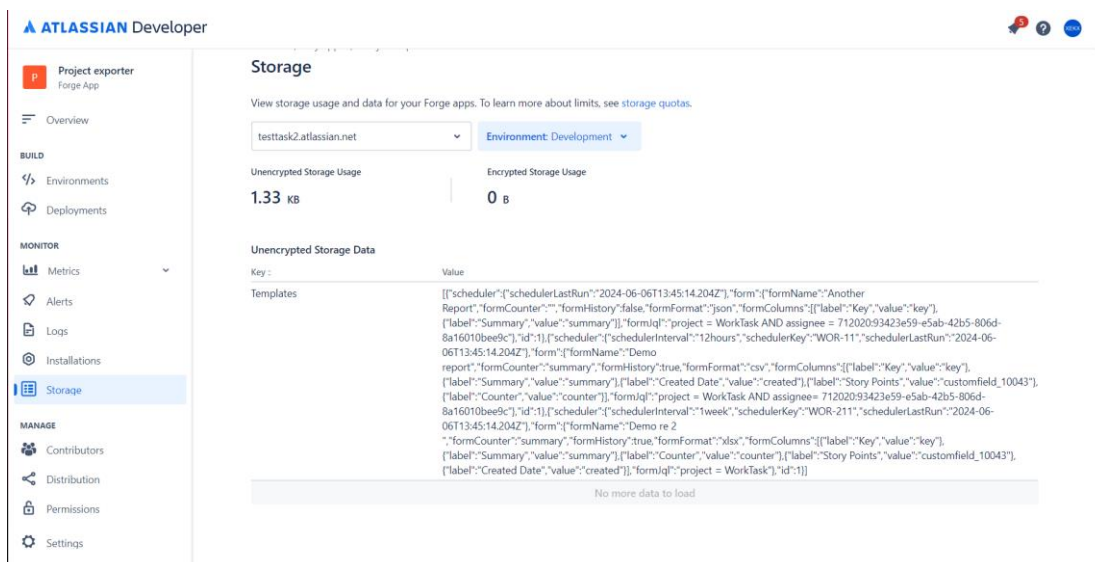


Рисунок 2.33 – Вигляд сховища Forge CLI

2.5 Використання застосунку

2.5.1. Головне вікно.

В якості головного вікна програми виступає динамічна таблиця в якій перелічено всі створені шаблони звітів. Для кожного шаблону передбачені функціональні кнопки: «Редагування», «Видалення», та «Планування», окремо додано функціональну кнопку «Створення». Кнопки «Створення» та «Редагування» відповідають за відкриття форми редагування звіту, а кнопка «Планування» - за відкриття форми планувальника експорту. Інтерфейс головного вікна додатку наведено на малюнку 2.34.

REPORT BROWSER			Create Report		
Key	History	Action Buttons			
Denys Report	true	Edit	Schedule	Remove	
Report 1	false	Edit	Schedule	Remove	
Another Report	false	Edit	Schedule	Remove	
Demo report	true	Edit	Schedule	Remove	
Demo re 2	true	Edit	Schedule	Remove	

Рисунок 2.34 - Вигляд головного вікна програми з шаблонами та без.

2.5.2. Вікно редагування шаблону.

Після натискання на кнопку створення або редагування шаблону, користувач переходить до відповідної форми. Вона включає в себе ряд полів для характеристики шаблону, таблицю попереднього перегляду, текстовий рядок для перевірки поля-фільтру, двох основних та трьох додаткових кнопок. Зображення інтерфейсу наведено на малюнку 2.35.

Template Name *

JQL filter

Check

All ok!

Counter filter

Input comma-separated values. You also should add Counter to the columns

Include history

Columns

Key x
Summary x
x v

Export format *

.json

.csv

.xlsx

Key	Summary
-----	---------

The table is empty and this is the
empty view

Cancel
Export
Get data
Save Report

Рисунок 2.35 – Форма редагування шаблону

Поле назви шаблону – визначає назву згенерованого файлу та ім'я шаблону у таблиці головного вікна програми, поле приймає текстові значення, різним шаблонам дозволяється мати однакові назви.

Поле JQL – виконує роль фільтру задач, вираз що описано у полі розпізнається як JQL-запит, і використовується під час запити даних через REST API. Для покращення користувацького досвіду у формі передбачена додаткова кнопка «Check», що дозволить користувачу швидко перевірити введений вираз на правильність.

Поле-лічильник - поле створене для відображення статистики змін у звіті. Воно працює наступним чином: користувач повинен через кому ввести ключі полів зміну яких хоче відстежити. При генерації звіту програма виведе число, що буде відповідати загальній кількості змін у запитуваних полях, для кожної задачі, для цього поля підтримується валідація для покращення користувацького досвіду. Для відображення лічильника у звіті необхідно обрати колонку Counter. Можливі ключі полів задачі Jira наведені на рисунку 2.36.

```
{
  "key": "statuscategorychangedate",
  "name": "Status Category Changed"
},
{
  "key": "issuetype",
  "name": "Issue Type"
},
{
  "key": "parent",
  "name": "Parent"
},
{
  "key": "timespent",
  "name": "Time Spent"
},
{
  "key": "customfield_10030",
  "name": "Total forms"
},
{
  "key": "customfield_10031",
  "name": "Work category"
},
{
  "key": "project",
  "name": "Project"
},
{
  "key": "customfield_10032",
  "name": "Approvals"
},
{
  "key": "customfield_10033",
  "name": "Request participants"
},
{
  "key": "fixVersions",
  "name": "Fix versions"
}
```

Рисунок 2.36 – Приклад можливих полів задачі

Поле Колонки – відповідає за наявність і порядок колонок у результуючому звіті, у коді передбачена можливість збільшення кількості підтримуваних колонок. Колонки займають місця за порядком визначення, тобто якщо необхідно встановити першою колонкою Summary, а другою Counter, потрібно обирати їх у такому ж порядку. Зовнішній вигляд компоненту зображено на малюнку 2.37.

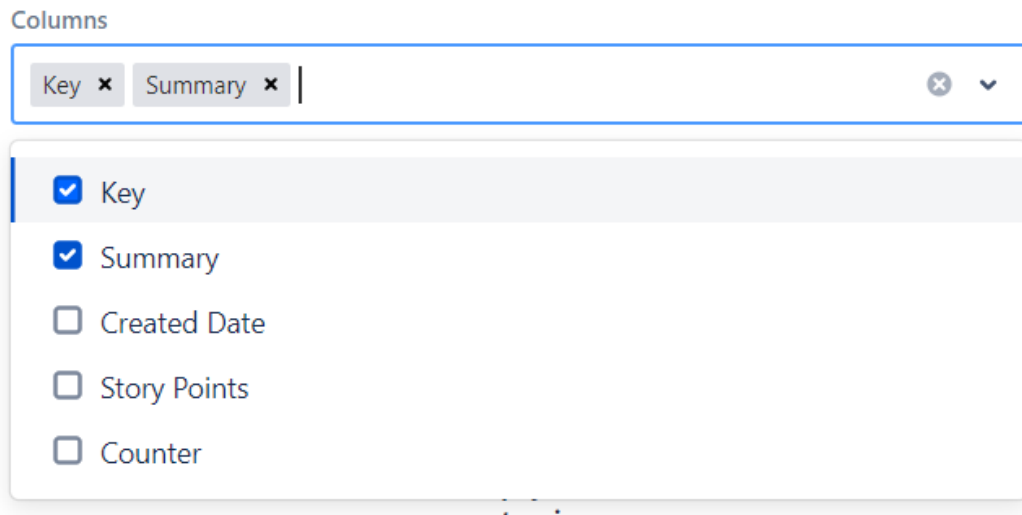


Рисунок 2.37 – Поле Колонки у розгорнутому вигляді

Флаг формату – визначає формат згенерованого звіту, може бути в одному з трьох значень: json, csv, xlsx. Формат звіту визначається кожному окремому шаблону, і на нього посилаються усі функції експортування наявні у програмі. Зображення елемента інтерфейса наведено на малюнку 2.38

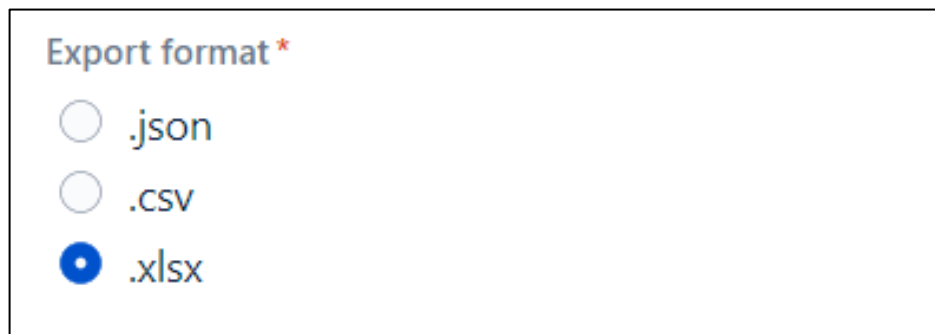


Рисунок 2.38 – Елемент «Флаг формату»

Таблиця попереднього перегляду – допоміжне поле, що дозволяє користувачу ознайомитись із змістом майбутнього звіту, до його експортування.

Для генерації або оновлення таблиці використовується кнопка «Get Data», при зміні обраних колонок таблиця оновлюється самостійно. На одній сторінці таблиці може бути до десяти задач. Зображення порожньої і заповненої таблиці наведено на малюнку 2.39.

Key	Summary	Counter	Created Date	Story Points
The table is empty and this is the empty view				
WOR-228	name	4	2024-04-27T11:57:28.788+0100	
WOR-227	name	0	2024-04-27T11:57:27.656+0100	
WOR-226	Team + Var	0	2024-03-29T10:01:00.756+0000	
WOR-225	Team+CustomF	0	2024-03-29T10:01:00.753+0000	2
WOR-224	Team+Standart	0	2024-03-29T10:01:00.710+0000	
WOR-223	Bug Report Template	0	2024-03-26T09:46:47.062+0000	
WOR-222	Node 1	0	2024-03-26T08:10:55.070+0000	
WOR-221	Node 1	0	2024-03-26T08:05:04.682+0000	
WOR-220	Test4	0	2024-03-23T18:16:04.503+0000	2
WOR-219	Test4	0	2024-03-23T18:16:03.001+0000	2

< **1** 2 3 4 5 ... 8 >

Рисунок 2.39 – Таблиця попереднього перегляду

Флаг історія – визначає чи буде у звіті розписана історія зміни задачі. У вимкненому стані кожна задача буде згадуватись у таблиці лише один раз. У увімкненому стані задача дублюється, кількість дублікатів відповідає кількості змін у задачі. При чому для кожного дублікату змінене поле набуває старого значення, відобразатись дублікат буде лише в тому випадку якщо змінене поле вказано у списку колонок, та увімкнено флаг історії. Наглядно принцип відображення історії продемонстровано на малюнку 2.40. На першій частині малюнку зображено звіт з вимкненим флагом, на другому з увімкненим, звернувши увагу на задачу WOR-228 можна побачити чотири строки із цим номером задачі,

але з різними значеннями у полі summary, це означає що поле summary було змінено чотири рази.

Key	Summary	Counter	Created Date	Story Points
WOR-228	name	4	2024-04-27T11:57:28.788+0100	
WOR-227	name	0	2024-04-27T11:57:27.656+0100	
WOR-226	Team + Var	0	2024-03-29T10:01:00.756+0000	
WOR-225	Team+CustomF	0	2024-03-29T10:01:00.753+0000	2
WOR-224	Team+Standart	0	2024-03-29T10:01:00.710+0000	
WOR-223	Bug Report Template	0	2024-03-26T09:46:47.062+0000	

Key	Summary	Counter	Created Date	Story Points
WOR-228	name1	4	2024-04-27T11:57:28.788+0100	
WOR-228	name12	4	2024-04-27T11:57:28.788+0100	
WOR-228	name1	4	2024-04-27T11:57:28.788+0100	
WOR-228	name	4	2024-04-27T11:57:28.788+0100	
WOR-227	name	0	2024-04-27T11:57:27.656+0100	
WOR-226	Team + Var	0	2024-03-29T10:01:00.756+0000	

Рисунок 2.40 – Зображення роботи флагу історія

Додаткова кнопка «Export» – реалізує можливість експорту звіту вручну, на відміну від автоматичного експорту ця функція пропонує одразу завантажити створений звіт.

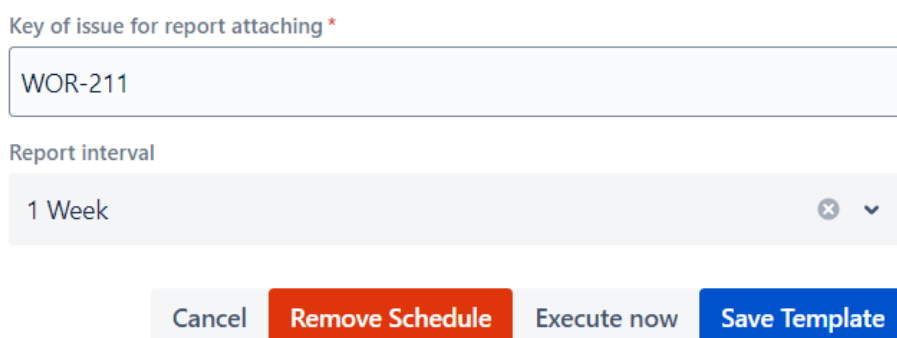
Для виходу із форми редагування шаблону передбачено кнопки «Save Report» та «Cancel», що реалізують функції відповідно виходу із збереженням шаблону та виходу без збереження шаблону.

2.5.3. Вікно планувальника.

Для переходу у вікно планувальника, потрібно з головного вінка додатку натиснути кнопку «Schedule», на потрібному шаблоні звіту. Для налаштування запланованого експорту користувач повинен заповнити наступні поля. Поле ключа задачі - приймає текстове значення, що повинно відповідати ключу існуючої задачі на сайті Jira. Вказуючи ключ, користувач обирає задачу з якою буде працювати

планувальник. Далі користувач обирає один із перелічених інтервалів, через який буде створюватись новий звіт. Для перевірки налаштувань користувач може натиснути «Execute now», як наслідок до задачі, ключ якої було наведено у відповідному полі, буде прикріплено файл звіту – ім'я файлу буде відповідати назві шаблону, а формат – вибраному формату у вікні редагування шаблонів. Інтерфейс форми планувальника зображено на малюнку 2.41.

Користувач може налаштувати подальшу взаємодію інструментами Jira. Так, наприклад, щоб надсилати повідомлення електронною поштою кожного разу, коли до задачі буде прикріплено новий файл, можна налаштувати систему Jira Automation, або додати свій акаунт як наглядача за задачею. Доступ до Jira Automation, можна отримати у налаштуваннях проекту. Приблизне правило, що б демонструвало описаний приклад наведено на рисунку 2.42.



The screenshot shows a form for scheduling a report. It has a text input field labeled "Key of issue for report attaching *" containing "WOR-211". Below it is a dropdown menu labeled "Report interval" with "1 Week" selected. At the bottom, there are four buttons: "Cancel", "Remove Schedule" (in red), "Execute now", and "Save Template" (in blue).

Рисунок 2.41 – Інтерфейс форми планувальника

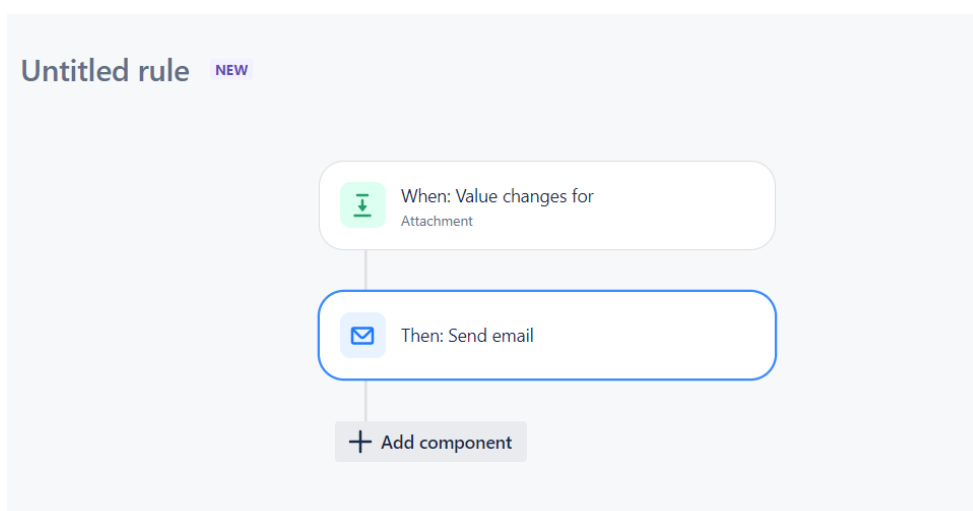


Рисунок 2.42 – Приклад правила Jira Automation

Висновки до розділу:

Цей розділ було присвячено розробці додатку, що допомагає автоматизувати процес формування звітів. Було сформульовано функціональні та нефункціональні вимоги, структуру класів.

У розділі було описано механізм розгортання додатків на Atlassian Forge, розглянуто тонкощі процесу, основні можливі конфігурації та вимоги до системи. Додатково було розглянуто і описано механізми тестування, розповсюдження та супроводження застосунку, розгорнутого на даній платформі.

За результатами цього розділу було написано програму, що дозволяє створювати і детально налаштовувати шаблони звітів, планувати автоматичний експорт, або завантажувати вручну.

ВИСНОВКИ

У результаті виконання роботи був розроблений застосунок, що дозволяє експортувати дані із системи Jira, забезпечуючи необхідний рівень гнучкості та зручності у використанні. Створений інструмент відповідає потребам сучасних компаній, що використовують Jira для управління проектами, і значно розширює можливості стандартних засобів експорту.

Були досліджені існуючі інструменти аналізу та експорту даних у Jira, виявлені їхні основні обмеження, такі як недостатня гнучкість, обмежені можливості автоматизації, а також проблеми з обробкою великих обсягів даних. Виявлені недоліки стали основою для визначення вимог до нового додатку, який повинен усунути вказані проблеми. Створена концепція додатку враховує всі виявлені проблеми та потреби користувачів. Архітектура додатку була розроблена таким чином, щоб забезпечити максимальну гнучкість, можливість автоматизації та надійність під час обробки великих обсягів даних. Особлива увага була приділена збереженню всіх необхідних метаданих та деталей під час експорту даних.

Додаток був реалізований з використанням сучасних технологій та методологій розробки програмного забезпечення. Платформа Atlassian Forge, на якій був створений додаток, забезпечила зручне середовище для розробки та інтеграції з Jira. Використання Jira Query Language (JQL) дозволило створювати складні запити до бази даних Jira та отримувати необхідну інформацію для експорту.

Додаток надає можливість автоматизації процесу експорту даних, включаючи налаштування розкладу виконання задач. Це значно зменшує час, необхідний для виконання рутинних завдань, та знижує ризик помилок, пов'язаних з людським фактором. Основні переваги додатку включають: Можливість автоматизації експорту даних, що дозволяє командам зосередитися на більш важливих завданнях, зменшуючи час, витрачений на рутинні операції, а гнучкі налаштування додатку дозволяють адаптувати процес експорту під специфічні

потреби користувачів, що забезпечує більш точний та релевантний аналіз даних; Зниження ризику помилок, так як автоматизовані процеси знижують залежність від людського фактору, що зменшує кількість помилок під час експорту даних. Підтримка різних форматів файлів (CSV, JSON, XLSX), що забезпечує гнучкість у використанні даних в інших інструментах та системах.

Розробка даного додатку є важливим кроком у напрямку покращення процесу управління проектами за допомогою Jira. Для подальшого розвитку можна розглянути: можливість включення більш складних сценаріїв автоматизації, які враховують різні умови та події у системі Jira, що дозволить ще більше знизити ручну працю та підвищити ефективність процесів; покращення інтерфейсу додатку, що забезпечить ліпший користувацький досвід.

У підсумку, результати виконаної роботи підтверджують ефективність обраного підходу до розробки додатку для експорту даних із системи Jira. Використання сучасних технологій та методів дозволило створити інструмент, який відповідає вимогам сучасного бізнесу та забезпечує підвищення ефективності робочих процесів з аналізу даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Welcome to Jira // Atlassian: [Веб-сайт]. URL: <https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software> (дата звернення: 06.01.2024).

2. Exporter - Export Issues to Excel CSV PDF // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1212073/exporter-export-issues-to-excel-csv-pdf> (дата звернення: 06.01.2024).

3. Time in Status // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1219732/time-in-status> (дата звернення: 06.01.2024).

4. BigTemplate // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1215229/bigtemplate-export-to-pdf-word-excel> (дата звернення: 06.01.2024).

5. Better Excel Exporter for Jira // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1212652/better-excel-exporter-for-jira> (дата звернення: 06.01.2024).

6. Xporter - Export issues from Jira // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/891368/xporter-export-issues-from-jira> (дата звернення: 06.01.2024).

7. Advanced Export // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1217474/advanced-export> (дата звернення: 06.01.2024).

8. R4J - Requirements Management for Jira // Atlassian Marketplace: [Веб-сайт]. URL: <https://marketplace.atlassian.com/apps/1213064/r4j-requirements-management-for-jira> (дата звернення: 06.01.2024).

9. Prepare to build your first Forge app // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/platform/forge/getting-started/> (дата звернення: 06.01.2024).

10. Part 1: Build a Jira hello world app // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/platform/forge/build-a-hello-world-app-in-jira/> (дата звернення: 06.01.2024).
11. Manifest // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/platform/forge/manifest/> (дата звернення: 06.01.2024).
12. Modules // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/platform/forge/modules/> (дата звернення: 06.01.2024).
13. About // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/#about> (дата звернення: 06.01.2024).
14. JQL // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-jql/#api-rest-api-3-jql-parse-post> (дата звернення: 06.01.2024).
15. Issue search // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-issue-search/#api-rest-api-3-search-post> (дата звернення: 06.01.2024).
16. Issue attachments // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-issue-attachments/#api-rest-api-3-issue-issueidorkey-attachments-post> (дата звернення: 06.01.2024).
17. Manage your apps // Atlassian Developer: [Веб-сайт]. URL: <https://developer.atlassian.com/platform/forge/manage-your-apps/> (дата звернення: 06.01.2024).
18. Моркун Н. В., Маринич І. А. Методичні вказівки до виконання кваліфікаційної роботи бакалавру для студентів спеціальності 151 “Автоматизація та комп’ютерно-інтегровані технології”. Кривий Ріг : Видавничий центр КНУ, 2019. 50 с.
19. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

20. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

21. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

22. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).