

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему_

АНАЛІЗ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЛАНЦЮГА ЗАСОБІВ ДЛЯ
КЕРУВАННЯ МІЖНАРОДНИМ ПРОЄКТОМ
ANALYSIS AND SOFTWARE IMPLEMENTATION OF THE TOOLCHAIN FOR
THE INTERNATIONAL PROJECT MANAGEMENT

Виконав: студент(ка) 3 курсу, групи КНТ-211м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Безугла А.М.

(прізвище та ініціали)

Керівник Табунщик Г.В.

(прізвище та ініціали)

Рецензент Шитикова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Безуглої Анни Максимівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Аналіз та програмна реалізація ланцюга засобів для керування міжнародним проєктом. Analysis and Software Implementation of the Toolchain for the International Project Management

керівник проєкту (роботи) Табунщик Галина Володимирівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року
 № 363

2. Строк подання студентом проєкту (роботи) 12 грудня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5. Висновки. 6. Текст програми 7. Презентація
 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Табунщик Г.В., к.т.н., професор		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “4” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10–13 тижні	Додатки
8	Нормоконтроль та рецензування.	14–15 тижні	
9	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Безугла А.М.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Табунщик Г.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
61 с., 3 табл., 29 рис., 2 дод., 25 джерел.

КЕРУВАННЯ ПРОЕКТАМИ, МОДЕЛІ КЕРУВАННЯ, TRELLO,
РОЗРОБКА, ATlassian.

Об'єкт дослідження – процес автоматизування управління проектом.

Предмет дослідження – гнучкі методології розробки.

Мета роботи – вивчення особливостей та практик управління проектами та автоматизація управління проектами.

Матеріали, методи та технічні засоби: платформа API Postman, програма для запуску збору даних у командному рядку для Postman - Newman, вебдодаток Trello, мова програмування JavaScript.

Результати. Проведено аналіз ринку діяльності конкурентів та поточної ситуації на ньому. Проведено порівняння конкурентів. Створено інформаційну

технологію автоматизованого управління проектами за рахунок платформи API Postman.

Висновки – автоматизоване управління проектами які використовують гнучкі методології.

Галузь використання – у проектах які використовують agile-методології.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 61 pages, 29 figures, 3 tables, 2 appendixes, 25 sources.

PROJECT MANAGEMENT, MANAGEMENT MODELS, TRELLO, DEVELOPMENT, ATLISSIAN.

Object of research - the process of automation of project management.

Subject of research - agile development methodologies.

Purpose - to study the features and practices of project management and automation of project management.

Materials, methods and techniques: Postman API platform, program to start data collection on the command line for Postman - Newman, Trello web application, JavaScript programming language.

Results. The market analysis of competitors' activities and the current situation on it was carried out. A comparison of competitors was made. Created information technology

technology for automated project management using the Postman API platform.

Conclusions - automated project management that uses flexible methodologies.

Scope - in projects that use agile methodologies.

.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ	9
1.1 Аналіз предметної області.....	9
1.1.1 Гнучке управління.....	9
1.1.2 Види гнучких методологій.....	10
1.1.3 Класичні методології	10
1.1.4 Гнучкі методології	11
1.1.5 Scrum	14
1.2 Постановка задачі.....	18
2 МАТЕРІАЛИ І МЕТОДИ	20
2.1 Програмне забезпечення	20
3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	27
4 РОЗРОБКА ПРОГРАМИ.....	30
4.1 Теоретична частина.....	30
4.1 Програмне забезпечення	33
Висновки	46
Перелік джерел посилання	47
Додаток А Текст програми.....	50
Додаток Б Презентація.....	53

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ДСТУ – Державний стандарт України;
- ПЕОМ – персональний комп'ютер;
- ПЗ – програмне забезпечення;
- Agile – це сукупність підходів і моделей поведінки, орієнтованих на використання ітеративної розробки;
- API – Application Programming Interface;
- CLI – Command-line interface;
- HTTP – Hypertext Transfer Protocol;
- IT – інформаційні технології;
- IPO – Initial public offering;
- JSON – JavaScript Object Notation;
- PGA – Programmable-Gain Amplifier;

ВСТУП

Управління стає все більш важливим видом діяльності в сучасному складному суспільстві. Оскільки організації стають більшими, складнішими та глобальнішими за масштабами, потреба в ефективному процесі управління проєктами ніколи не була такою великою. У кожній з цих організацій управління є важливим видом діяльності. Менеджери постійно потрібні для планування, організації, спрямування та контролю операцій, щоб забезпечити досягнення організаціями своїх цілей. Щоб робити це ефективно, керівники проєктів повинні мати глибоке розуміння інструментів і методів, необхідних для успішного управління проєкт.

Попри це, багато організацій не володіють ресурсами або персоналом, необхідними для ефективного управління своїми проєктами та програмами. Ці організації часто покладаються на третіх осіб, які надають їм рекомендації щодо того, як організувати свої проєкти та програми. Також часто може поставати проблема перед інтернаціональними командами, через різні часові пояси інколи ефективність сумісної роботи може губитися. Інколи може бути важко все проконтролювати коли все змінюється надто швидко. На жаль, невеликі проєкти не можуть дозволити собі витрати на наймання консультанта, який би допоміг їм у цьому процесі, а навчити людей, які не пройшли підготовку, бути менеджерами проєктів - нелегке завдання. Згідно з Серрадор і Пінто [1] як наслідок, багато проєктів не досягають бажаних результатів і стають дорогими невдачами для організації витрачаючи тріліони.

Одним з пріоритетних шляхів підвищення якості роботи є аналіз та впровадження засобів для керування ланцюжка міжнародного проєкту. Ці інструменти можуть бути використані для підвищення ефективності бізнес-процесів, а також для скорочення витрат і забезпечення інших переваг, таких як підвищення задоволеності клієнтів і поліпшення морального духу співробітників.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

З розвитком технологій галузі стикаються з більшою конкуренцією з боку бізнес-середовища, і вони повинні адаптуватися і залишатися на висоті, щоб залишатися конкурентоспроможними. Отже, організації використовують проектну діяльність для роботи з великими, складними проектами. Це економить час, допомагає ефективно управляти ресурсами та допомагає адаптуватися до швидко мінливого бізнес-середовища. Для організацій стає все більш важливим мати модель управління проектами, яка йде в ногу з цими досягненнями [2].

1.1.1 Гнучке управління

Нам необхідно зрозуміти, як великі компанії управляють потоком свого часу, налагоджують процеси в команді та своєчасно відстежують всі зміни. Проекти характеризуються унікальністю, інноваційністю, складністю, невизначеністю, неоднозначністю і піддаються мінливим вимогам. Звідси компанії та проекти повинні постійно розвиватися, щоб відповідати мінливим вимогам. Проекти, пов'язані з такими сферами, як інформаційні технології, фармацевтика, будівництво, мають одне в загальному. Ними потрібно керувати, планувати, укомплектовувати, організовувати, контролювати, контролювати та оцінений ринок.

Впровадження гнучкого управління проектами стало результатом що росте впливу технологій на управління проектами. Гнучкі методи використовуються для того, щоб зробити операції більш ефективними та економічно доцільними [3].

Гнучка методологія та її фреймворки - це те, що дозволяє великому бізнесу оптимізувати своє керування часом та ефективно управляти проектами.

1.1.2 Види гнучких методологій

Різні типи проектів вимагають різних процедурних моделей для успішного виконання. Історично склалося так, що інструменти та методи, розроблені для виконання практики управління проектами, використовувалися лише в обмеженій кількості галузей, таких як аерокосмічна, оборонна та будівельна. Сьогодні використання практик управління проектами поширилося в різних областях, а управління проектами стало предметною дисципліною власний. Це ілюструє, як практика управління проектами стала більш більш укоріненими в сучасному бізнесі. В рамках дисципліни використовуються дві методології, традиційні методології управління проектами та гнучкі методології управління проектами методології.

1.1.3 Класичні методології

Використовуючи класичне управління проектами, менеджери встановлюють графіки виконання проектів з визначеними обсягами, термінами та ресурсами. Традиційно управління проектами базується на ідеї, що краще планування призводить до кращих проектів. Багато початкових зусиль витрачається на планування та визначення обсягу проекту, не змінюючи його, наскільки це можливо, а потім на виконання завдань відповідно до графіка, неухильно дотримуючись плану [4].

Одним із найбільш часто використовуваних традиційних підходів до управління проектами є метод водоспаду описаний Ройсом (1970), представлений нижче на рисунку 1.1 [5]. Модель водоспаду виглядає як

логічна послідовність кроків протягом життєвого циклу розробки ПЗ. У такій моделі кожен етап активується після завершення попереднього етапу.



Рисунок 1.1 – Модель водоспаду [5]

Участь зацікавлених сторін у цьому процесі дуже обмежена, і ці моделі не представляють жодні результати клієнтам до кінця проєкту. Весь товар доставляється замовнику в кінці фази. Отже, якщо продукт не задовольняє вимоги споживачів, це призведе до додаткових витрати [6].

1.1.4 Гнучкі методології

Гнучка методологія організовує методи та інструменти управління проєктами у фази або процеси проєкту у стандартизований спосіб. Гнучку методологію управління проєктами можна умовно розділити на планово-орієнтовані методи, які слідують класичному водоспадному процесу, і гнучкі методи, які слідують ітеративному підходу, заснованому на тестуванні [7]. Деякими прикладами гнучких методів розробки є Scrum, eXtreme Programming, Crystal методології, метод динамічної розробки програмного

забезпечення (DSDM), розробка, керована функціями, та ощадлива розробка програмного забезпечення (Lean software development). Порівняльний аналіз гнучких методологій можна побачити на таблиці 1.1 [9]. Звіт Standish Group Chaos Report [8] за 2015 рік, який узагальнює дані на основі результатів більш ніж 10 000 проєктів у період з 2011 по 2015 роки підкреслює, що, загалом, для проєктів будь-якого розміру проєкти Agile більш успішні, ніж проєкти водоспаду.

Таблиця 1.1 – Порівняння гнучких методологій

Голівка	Ключові моменти	Унікальні особливості	Недоліки
Екстремальне програмування (XP)	Невеликі команди, щодня проводяться зустрічі, неформальний тип спілкування всередині команди, мінімум документації	Постійне вдосконалення проєкту з метою покращення його ефективності та адаптації до змін	Підходить скоріше для індивідуальних практик, ніж для управління в глобальному масштабі, адже в останньому випадку є ризик формування недисциплінованих команд
Scrum	Самостійна, мала, самостійно організована команда розробників, тривалість ітерації - 2-4 тижні, команда самостійно вирішує, скільки часу їй необхідно для завершення роботи над завданням	Високий ступінь комунікації та взаємодії усередині команди, наявність чітко визначеної формальної організації цієї методології	Існує ризик зростання термінів реалізації проєкту внаслідок витрат на проведення Скрам-заходів

Продовження таблиці 1.1

Голівка	Ключові моменти	Унікальні особливості	Недоліки
“Ощадлива розробка” (lean)	Застосування засобів візуалізації, розробка через випробування, невеликі ітерації	Головними функціями програмного забезпечення є ті, які є важливими для замовника, відбувається постійна мотивація колективу	Рішення приймаються тривалий час, придатні лише для малих проектів
“Розробка керована функціональністю” (fdd)	П'ятиетапний процес, об'єктно-орієнтована розробка, дуже короткі ітерації (можуть тривати кілька годин)	Легкість методу, об'єктне моделювання	Ця методологія зосереджена лише на розробці та впровадженні, і лише незначна увага приділяється розробці та впровадженню
Kanban	Самоорганізований колектив, всі зосереджені на виконанні поставленого завдання	Обмеження по терміну виконання немає, але є обмеження за кількістю "роботи зараз", тобто фактично, коли немає інформації про те, що може знадобитися замовнику від ПЗ в майбутньому	Водою методики є її тривалий характер

Всі основні принципи гнучкої методології прописані в маніфесті для розробників. Це своєрідний "Маніфест руху гнучкої розробки програмного забезпечення" [10] з набором правил для роботи. Це стало основним керівництвом для розробників під час написання програмного забезпечення.

1.1.5 Scrum

Прикладом методології гнучкої розробки програмного забезпечення є Scrum. Scrum - це полегшений фреймворк, котрий допомагає людям, командам та організаціям створювати цінність за рахунок адаптивних рішень для вирішення складних проблем [11]. Що стосується концепцій і практик, то вони ґрунтуються на трьох основних категоріях: Ролі, Артефакти та Тимчасові блоки [12]. У Таблиці 1.2 [13] підсумовано три компоненти та основні елементи кожного з них.

Таблиця 2.1 – Фреймворк Scrum

Ролі	Тимчасові блоки	Артефакти
Власник продукту	Щоденна скрам-зустріч	Беклог продукту
Скрам команда	Щоденна скрам-зустріч скрамів	Беклог спринту
	Зустріч з огляду спринту	Діаграма згортання
	Зустріч з огляду спринту	

Scrum найчастіше використовується при створенні складного програмного забезпечення як метод ітеративної та інкрементальної розробки. Scrum значно підвищує продуктивність і скорочує час розробки порівняно з класичними методами водоспаду розробки. За допомогою процесів Scrum організації можуть плавно адаптуватися до швидко мінливих вимог і виробляти продукти, що відповідають висхідним бізнес-цілям. Scrum приносить користь організації, допомагаючи їй [9]:

- поліпшити якість готових продуктів;
- краще справлятися зі змінами (і очікувати змін);
- надавати більш якісні оцінки, витрачаючи менше часу на їх створення;
- мати більший контроль над графіком і статусом проєкту.

Спринт є контейнером для всіх інших подій. Кожна подія в Scrum є формальною можливістю перевірити та адаптувати артефакти Scrum. Ці події

спеціально розроблені для забезпечення необхідної прозорості. Невиконання будь-яких подій за призначенням призводить до втрати можливостей для перевірки та адаптації. Події використовуються в Scrum для створення регулярності та мінімізації потреби в зустрічах, не визначених в Scrum. Оптимально, коли всі події проводяться в один і той же час і в одному і тому ж місці, щоб зменшити складність [11]. Основні етапи представлено на рис. 1.2 [9].

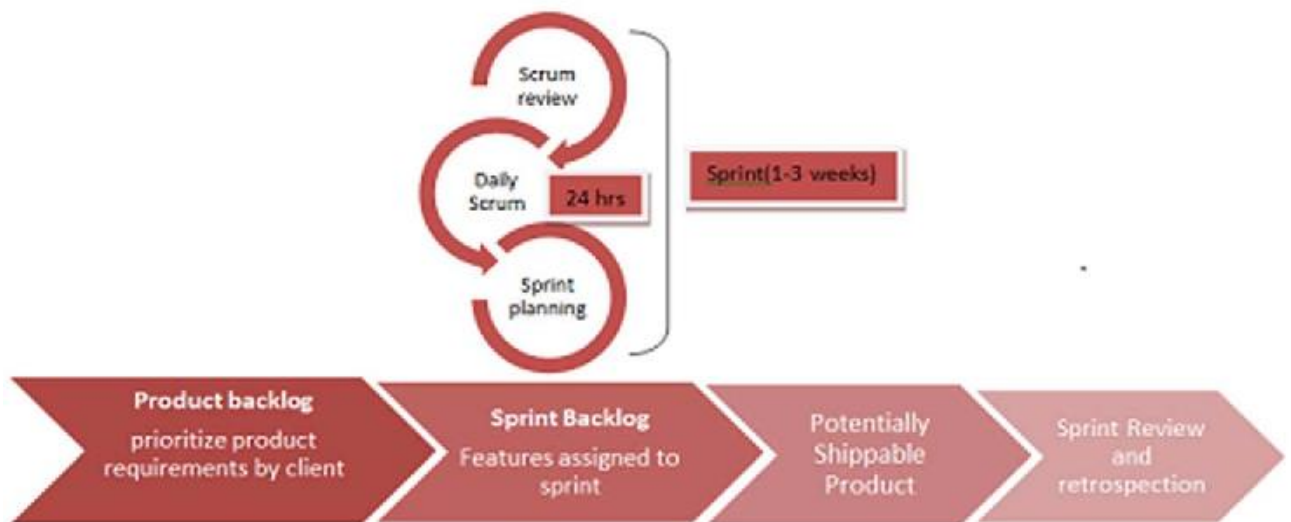


Рисунок 1.2 – Цикли scrum [9]

Процес Scrum виглядає простим, але є досить практичним, щоб глибоко вплинути на робочий досвід і зафіксувати ключові характеристики гнучкого підходу. До ключових практик Scrum можна віднести наступні моменти [14] - [15]:

- самокерована, крос-функціональна, самокерована, самоорганізована, і згуртовані команди з восьми або менше членів команди розробляють програмне забезпечення в спринтах;

- спринти - це ітерації фіксованої 30-денної тривалості, де кожен спринт забезпечує інкрементну, протестовану функціональність, що представляє цінність для користувача;

- робота в рамках спринту є фіксованою. Після того, як обсяг Спринту обрано, жодна зовнішня робота не може бути додана, окрім як командою розробників;
- Scrum майстер наставляє та керує командами, які є відповідіає за досягнення успішних результатів на кожному спринті;
- вся робота, яка повинна бути зроблена виконується у вигляді Product Backlog, який включає вимоги, що підлягають виконанню, дефекти, робоче навантаження, а також інфраструктуру;
- беклог продукту розробляється, управляється та пріоритизується власником продукту, який є невіднятним членом команди і який несе основну відповідальність за взаємодію із зовнішніми клієнтами;
- щоденні наради зі спеціальними питаннями є основним методом комунікації;
- Scrum значною мірою фокусується на тайм-боксингу. Спринти, щоденні наради, наради з перегляду релізів завершуються у встановлений час;
- Scrum допускає, щоб вимоги, архітектура та дизайн з'являлися в ході проєкту;
- Scrum забезпечує демонстрацію для зовнішніх зацікавлених сторін в кінці кожної ітерації, і кожна ітерація є адаптивним плануванням, керованим клієнтом.

Фундаментальною одиницею Scrum є невелика команда людей - Scrum Team. Скрам-команда складається з одного Скрам-майстра, одного Власника продукту та Розробників. Усередині Скрам-команди не існує підкоманд або ієрархії. Це згуртована команда професіоналів, сфокусована на досягненні однієї мети — Цілі Продукту. Скрам-команди є крос-функціональними, що означає, що їх учасники володіють усіма навичками, необхідними для створення цінності кожного Спринту. Вони також є самокерованими, тобто внутрішньо вирішують, хто, що, коли і як робить [12]. Схема скрам-команди наведена на рисунку 1.3 [16].

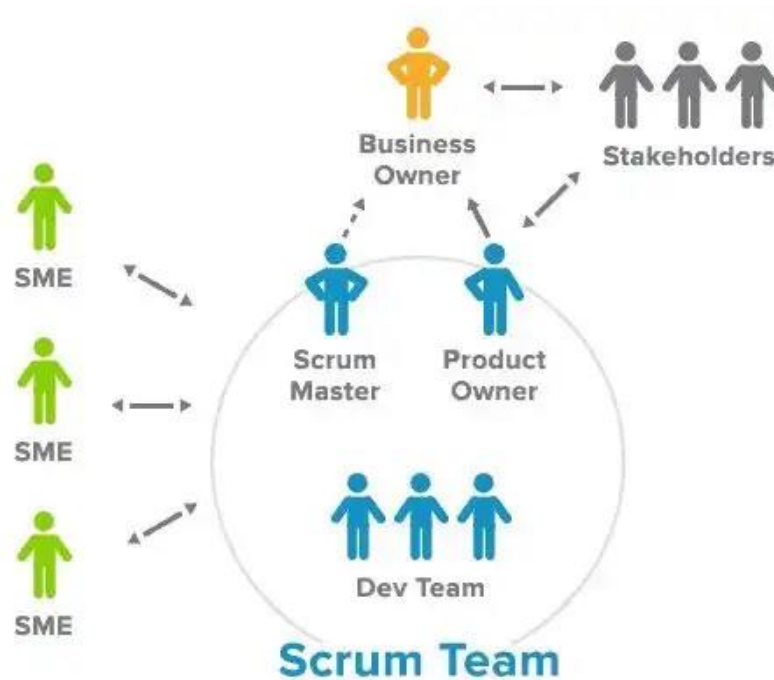


Рисунок 1.3 – Схема взаємодії скрам-команди [16]

Існує декілька гнучких методологій, і хоча Scrum є найпопулярнішою серед них, вона не єдина. Тому важливо мати чітке уявлення про всі ці методології, щоб розуміти їх переваги та недоліки.

Також хотілося б зазначити, що є офіційні опитування щодо популярності використання різних гнучких методологій. Компанія VersionOne оприлюднила результати 10-го щорічного дослідження "Стан гнучких методів розробки" [17]. Повний звіт опитування можна побачити на рисунку 1.4 [17]. Результати, отримані на основі понад 3800 відповідей з усього світу, свідчать про те, що гнучкі методи приносять відчутні переваги; вони неухильно стають підходом за замовчуванням до розробки програмного забезпечення і починають поширюватися на сфери, що не пов'язані з програмним забезпеченням.

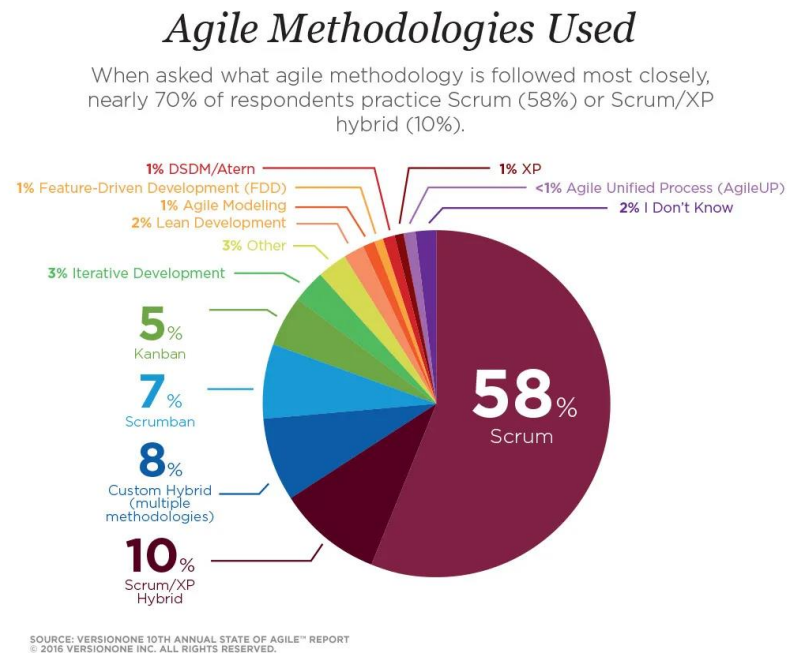


Рисунок 1.4 – Діаграма використання гнучких методологій [17]

1.2 Постановка задачі

У сучасному світі як великі корпорації, так і малий бізнес не можуть функціонувати без систем управління проектами. Це викликано тим, що системи управління проектами є невід'ємною частиною сучасних бізнес-процесів. По суті, це інструменти управління проектами, які можна адаптувати під потреби компаній і продуктів. Плануйте спринти та релізи, відстежуйте проекти та завдання в Trello та Jirra, використовуйте Agile дошки, створювати робочі процеси, зберігати базу знань, працювати зі звітами.

В результаті автоматизації їхні процеси стають швидшими та ефективнішими, що дає змогу заощадити час та підвищити продуктивність праці працівників. Роботодавці також можуть заощадити гроші, оскільки їм не потрібно платити високі зарплати своїм працівникам. Крім того, вони можуть зосередитися на інших важливих завданнях і передати на аутсорсинг ті частини своїх проектів, які їм не потрібно виконувати самостійно.

Тому було запропоновано зробити аналіз та реалізувати ланцюг засобів для керування міжнародним проєктом:

- а) провести аналіз предметної області;
- б) провести порівняльний аналіз чинних інструментів необхідних для управління проєктами;
- в) реалізація управління проєктами;
- г) написання тестів для перевірки відповіді сервера.

2 МАТЕРІАЛИ І МЕТОДИ

Спростити управління робочими процесами для різних команд сьогодні можна за допомогою цілого сімейства продуктів, розроблених з цією метою. У минулому він створювався як інструмент для відстеження помилок і завдань. Однак сьогодні це також потужний інструмент управління роботою, який можна використовувати в найрізноманітніших ситуаціях, від управління вимогами і управління тестовими сценаріями до гнучкої розробки програмного забезпечення.

2.1 Програмне забезпечення

2.1.1 ClickUp

Немає сумнівів, що ClickUp є одним з найпопулярніших інструментів для покращення продуктивності у світі, і його люблять компанії всіх видів у всьому світі. У безкоштовній версії ClickUp доступний ряд функцій, включаючи призначені коментарі, списки справ, контрольні списки завдань, редагування тексту, багата панель інструментів завдань, прості/нестандартні статуси, спринти, цілі та багато іншого. Приклад інтерфейсу наведено на рис. 2.1 [18].

На додаток до відмінного обслуговування, ClickApps дозволяє налаштувати роботу вашої команди в рамках кожного окремого простору, дозволяючи вам включати пріоритети, теги, користувацькі поля, а також синхронізувати ваші простори з вашим робочим процесом. Ця потужна функція дозволяє синхронізувати роботу вашої групи з роботою вашої команди. У двох розділах платформи ті, кому надано звання адміністратора, мають можливість включити або відключити функцію ClickApps [18].

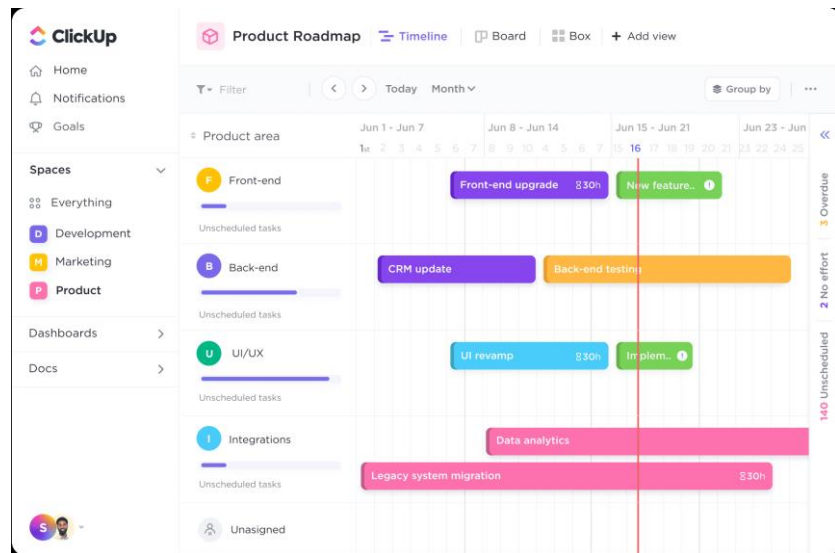


Рисунок 2.1 – Приклад дошки у ClickUp [18]

2.1.2 Asana

Співпрацювати з Asana легко, саме тому вона так популярна. Ви можете додавати виконавців та вкладення до завдань, відстежувати їх, співпрацювати в режимі реального часу та коментувати їх. Серед можливостей Asana - кілька робочих просторів, відстеження завдань, співпраця в режимі реального часу та коментування. Завантажуючи завдання в систему, користувачі можуть навіть бачити завдання та пріоритети своїх колег. Прозорість, якої потребує кожна організація та гнучкий проєкт, важко переоцінити. Існує навіть можливість відстежувати хід виконання проєктів та завдань з різних пристроїв та браузерів. Коли Asana працює, нема потреби покладатися на сторонні додатки або електронну пошту для корпоративної комунікації [19]. На рис. 2.2 [19] наведено приклад роботи з Asana.

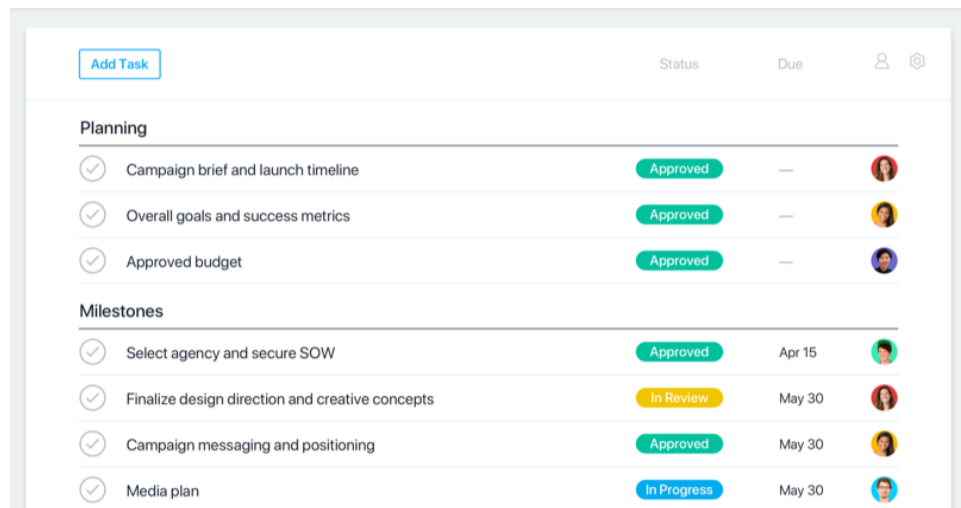


Рисунок 2.2 – Приклад інтерфейсу Asana [19]

2.1.3 Kanbanize

Ще одним популярним інструментом проектного менеджменту є Kanbanize. Команди можуть візуалізувати основні ініціативи та розділити їх на робочі елементи, використовуючи дошки Kanban. Завдяки робочим обмеженням, фільтрам, доступу на основі ролей та користувацьким полям, можна візуалізувати роботу так, як потрібно. Можна створювати різні кастомні робочі процеси, за допомогою використання дошки Kanban або використовуючи функцію часової шкали для модифікації існуючого робочого процесу. Крім того, успішно реалізовано метод створення електронних листів, оскільки він дозволяє відстежувати час, щоб бачити години, витрачені на конкретні завдання або проекти, а також аналітику для моніторингу продуктивності.

2.1.4 Notion

Існує кілька способів використання Notion, таких як ведення нотаток, створення документів, управління проектами, створення завдань, використання методу Канбан тощо, і все це в одному місці. На перший погляд, він здається досить простим і легким у використанні, але деякі

користувачі виявили, що він може бути досить лякаючим через кількість способів, якими він може бути використаний. Приклад дошки Kanban можна побачити на рисунку 2.3 [20].

KANBAN BOARD EXAMPLE

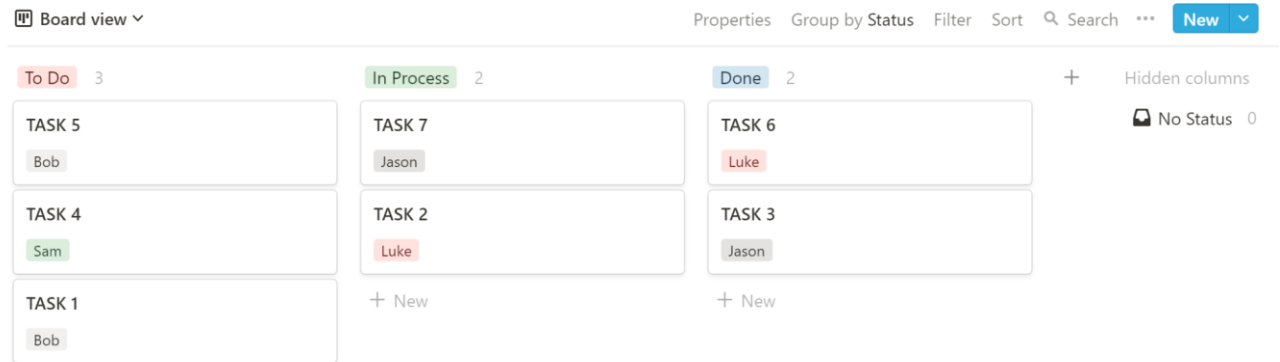


Рисунок 2.3 – Приклад дошки Kanban у Notion [20]

2.1.5 Jira

Спочатку розроблена як платформа для спостереження за проблемами для розробників ПЗ, Jira перетворилася на щось більше, ніж просто інструмент управління проектами. Приклад оформлення сайту наведено на рисунку 2.4 [21]. Сьогодні вона доступна в трьох окремих формах [21]:

- а) Jira Core: основна платформа управління проектами Jira;
- б) Jira Software: пропонує всі можливості Jira Core, але також включає додатковий Agile функціонал;
- в) Jira Service Desk: призначена для ІТ-фахівців або інших форм служба підтримки.

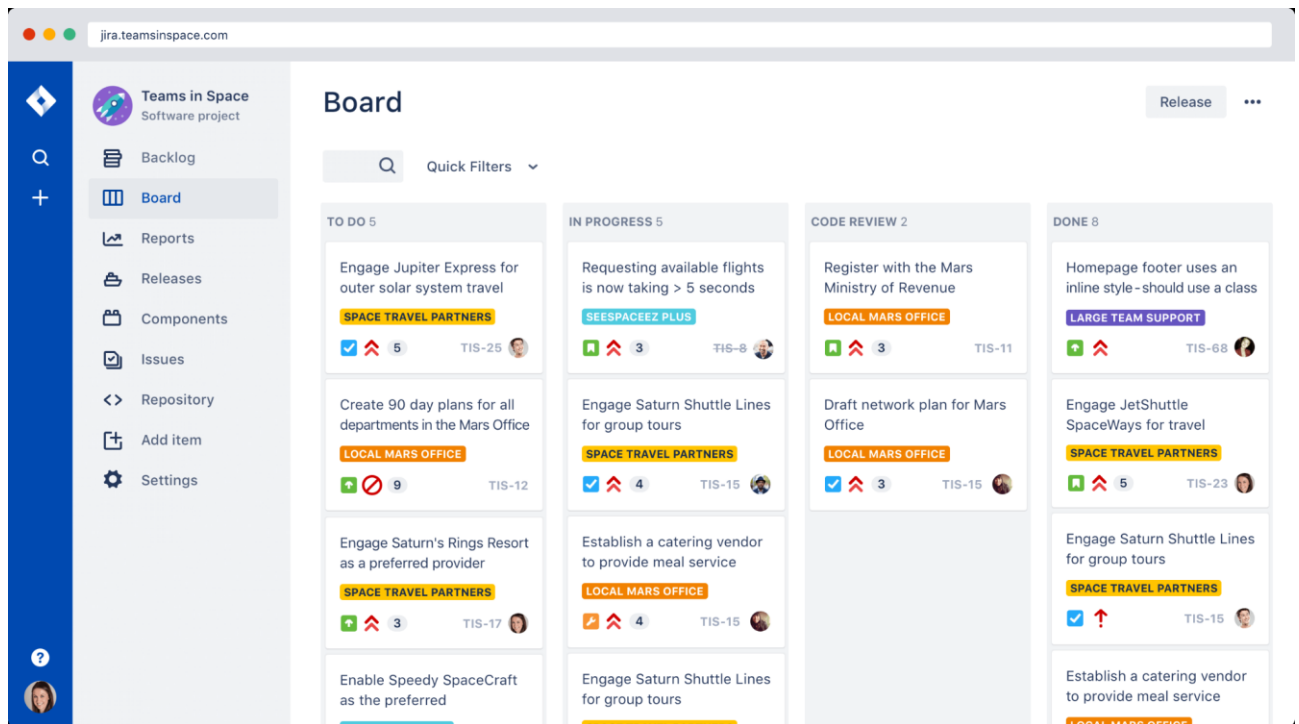


Рисунок 2.4 – Приклад Jirra [21]

2.1.6 Trello

З точки зору управління проектами та контролю, Trello - це хмарний сервіс управління проектами. Він досить простий у використанні та розумінні для пересічної людини, а також відповідає методології Kanban. Крім того, він майже весь безкоштовний, що робить його ідеальним як для індивідуального використання, так і для роботи невеликих команд.

Окрім онлайн-версії, Trello доступний у вигляді мобільного додатку для Android та iOS. Працювати з додатками можна і в автономному режимі, що робить їх досить зручними у використанні. Приклад інтерфейсу можна подивитись на рисунку 2.5 [25].

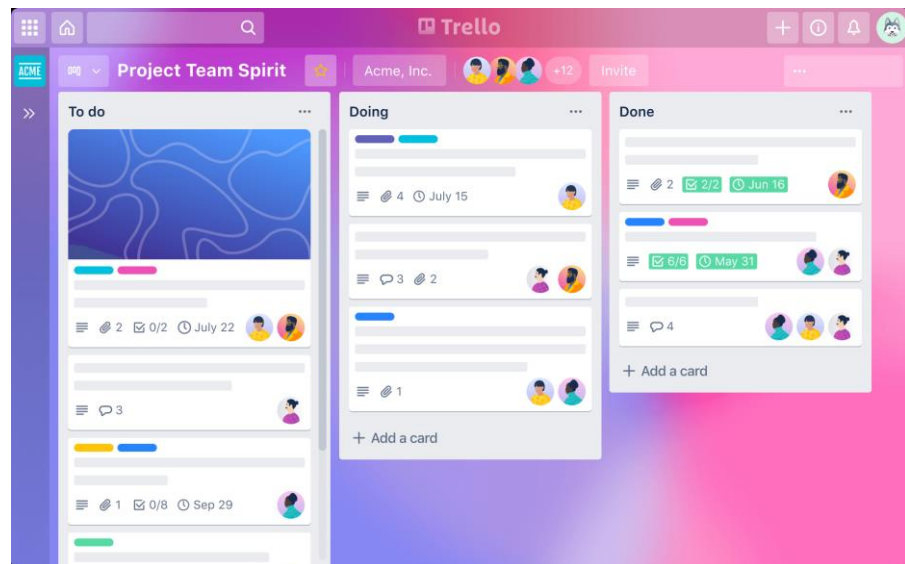


Рисунок 2.5 – Приклад інтерфейсу Trello [25]

Ми спостерігаємо збільшення кількості інструментів для координації завдань між різними командами через віддалену роботу та більше розвинення інтернаціональних проектів. Насправді Gartner очікує [22], що витрати на соціальні платформи та платформи для спільної роботи зростуть на 17% до 4,5 мільярдів доларів цього року, причому найбільшим фактором будуть платформи для управління роботою. Це конкурентне поле. Поряд з Trello, інші постачальники програмного забезпечення для управління роботою задовольняють попит бізнесу. Кілька з них нещодавно завершили IPO, включаючи Asana та Monday, які залучили \$4 млрд та \$7 млрд відповідно. Workfront (придбаний Adobe за 1 мільярд доларів) і Wrike (придбаний Citrix за 2,25 мільярда доларів) - інші конкуренти, які були придбані за останні пару місяців. Пакет Microsoft 365 також поставляється з інструментом управління завданнями під назвою Planner.

Багато з цих засобів створені на основі того ж самого підходу до списків справ у стилі Kanban, який використовує Trello; вони надають схожі дошки, списки та макети карток, та інші "види". Крім того, більшість з них мають функції автоматизації та співпраці. Решта конкуруючих компаній мають більш зосереджені на електронних таблицях погляди на управління проектами; серед них можна виділити до них належать окремі програми, що

добре фінансуються, такі як Smartsheet та Airtable, а так же такі продукти, як Microsoft Lists та Google Tables.

Trello вже тривалий час утримує значну частину ринку, переважно завдяки простоті використання. Відповідно до останньої публічно доступної статистики, у 2018 році було зафіксовано щонайменше 50 мільйонів зареєстрованих користувачів, хоча Atlassian каже, що зараз ця цифра суттєво вища. Це також було досягнуто завдяки інтегруванню в портфоліо додатків Atlassian, зокрема, Jira.

Якщо вам потрібен більш простий дизайн дошки, але вам потрібні картки та оновлення статусів, тоді слід скористатися Trello. Для користувачів не будуть доступні такі функції, які є складними, як історія або відстеження проблем, але ви можете створити різні дошки для бэклога, спринту або наступної ітерації. Використання Trello може бути доцільним для меншої команди, яка хоче керувати кількома проектами за прийнятну ціну. Що добре в Trello - це бонуси, які допомагають додавати діаграми вигорання або діаграми Ганта. Є змога додати потрібні вам функції, не заплативши за них на самому початку. Все це в інтуїтивно зрозумілому та зручному інтерфейсі.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

В роботі було використовувано платформу для побудови та використання API - Postman Version 8.12.1 (8.12.1), яка значно спрощує життєвий цикл API. Всі артефакти API можна зберігати, каталогізувати та співпрацювати в одному місці. Окрім зберігання та управління специфікаціями API та сценаріями робочих процесів, Postman також може керувати тестовими кейсами та робочими процесами, тестовими кейсами та результатами, метриками та всім іншим, що пов'язано з API. На додаток до полегшення проектування, тестування, документування, моделювання, розповсюдження та доступності API, платформа містить набір інструментів, які прискорять весь життєвий цикл API. Можливості наведені на рисунку 3.1.



Рисунок 3.1 – Можливості Postman

Створювати і запускати тестування напряму в Postman або в складі CI/CD конвеєру через Newman (засіб для збору даних, який надає можливість виконувати і тестувати колекції Postman прямо з командного рядка). Postman можна застосовувати для написання функціональних тестів, інтеграційних тестів та тестів використовуватися для написання функціональних тестів, тестів інтеграції, регресійних тестів і т.д. Виконавче середовище Postman, яке базується на Node.js містить підтримку загальних шаблонів і бібліотек, які можуть бути використані для оперативного створення тестів.

Запускає колекції Postman з командного рядка. Дозволяє запускати, тестувати та інтегрувати з серверами безперервної інтеграції та легко будувати системи.

Newman запускає колекції так, як вони виконуються у завантажувачі колекцій Postman. Він має рівні можливості з Postman і дозволяє запускати їх. За допомогою команди `newman run` ви можете вказати, яку саме збірку слід запустити. Якщо ваша колекція Postman доступна у вигляді URL-адреси (наприклад, зі служби хмарного API), Newman може отримати її і також запустити. Експортувати колекцію Postman як JSON-файл і запустити його за допомогою Newman. Модуль Node.js робить Newman простим у використанні в JavaScript проектах. З Newman ви можете використовувати Reporters для перегляду поточної колекції, що виконується. Ви можете налаштувати Reporters за допомогою опції `-r` або `--reporters`. Newman поставляється з `cli`, `json`, `junit`, `progress` і `emojitrain`. У Newman за замовчуванням увімкнено CLI Reporter, тому немає необхідності вказувати його як частину параметра. Однак увімкнення одного або декількох інших звітів не призведе до виводу даних за допомогою CLI. Приклад роботи Newman можна побачити на рисунку 3.2 [23].

```
Downloads $ newman run Postman_Newman_IntegrationCollection.json
newman
```

Postman-Newman Integration

Register User

POST https://reqres.in/api/register [200 OK, 620B, 513ms]
✓ Status code is 200

Get User

GET https://reqres.in/api/users/4 [200 OK, 738B, 60ms]
✓ Status code is 200

Login User

POST https://reqres.in/api/login [200 OK, 465B, 250ms]
✓ Status code is 200

	executed	failed
iterations	1	0
requests	3	0
test-scripts	6	0
prerequisite-scripts	4	0
assertions	3	0
total run duration: 986ms		
total data received: 228B (approx)		
average response time: 274ms [min: 60ms, max: 513ms, s.d.: 185ms]		

summary

Рисунок 3.2 – Newman [23]

4 РОЗРОБКА ПРОГРАМИ

4.1 Теоретична частина

Планується, що окремою колекцією будуть представлені методики взаємодії з Jira та Trello на основі прописаних можливих сценаріїв. Крім того, всі змінні планується винести в окрему колекцію оточення, щоб забезпечити безпеку та масштабованість проекту, а також його управління в майбутньому. Потім вони будуть запускатися з командного рядка Newman або, за необхідності, з будь-якої платформи CI/CD. Протягом всієї колекції оточень в додаток вбудовується секретний Trellokey і токен, щоб сервер Trello розумів, що наш додаток прикріплений до певного облікового запису, на якому будуть створюватися, видалятися та оновлюватися всі необхідні сутності. За допомогою терміналу ми відправимо HTTP-запит на сервер Trello, який визначить, куди нам потрібно його відправити та надішле у відповідь, яку ми проаналізуємо за допомогою письмових тестів. Повну схему можна побачити на рисунку 4.1 [25].

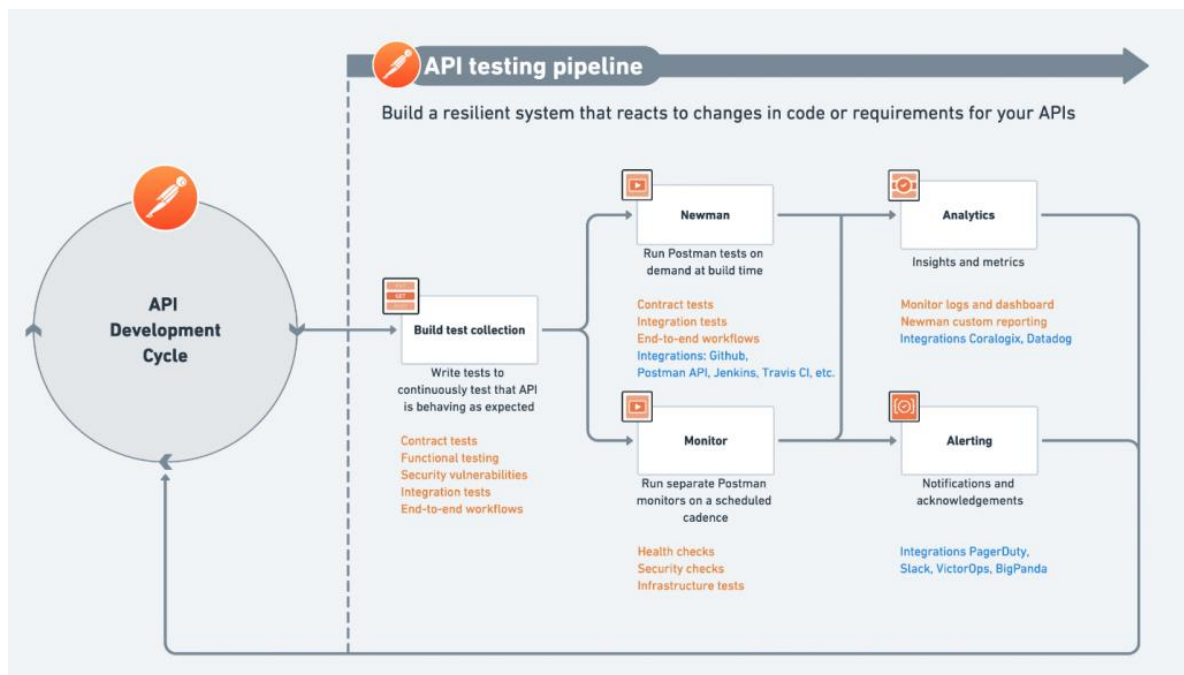


Рисунок 4.1 – Поетапна схема роботи [25]

Делегуючи автентифікацію та авторизацію, Trello не потрібно турбуватися про збереження або управління іменами користувачів та паролями в додатку. В результаті, додаток передає управління Trello (ідентифікуючи себе за допомогою токена API), і як тільки Trello дозволить користувачеві вибрати обліковий запис і увійти в систему, Trello передасть управління назад Trello разом з токеном API, щоб Trello зміг перепризначити користувача та управління назад в додаток. Для того, щоб почати користуватися Trello, потрібно зробити декілька кроків, і перший з них - отримати ключ API. Ключ API можна знайти на сайті Trello за адресою <https://trello.com/app-key>. Приклад того як це виглядає можна побачити на рисунку 4.2. Ключ API буде чітко позначений на сторінці.

Developer API Keys

Key:

b75aac0cba423144bb4fedfef1b9c854

Рисунок 4.2 – API ключ Trello

Необхідно, щоб ключ API являв собою 32-символьний рядок, що складається з випадкових буквено-цифрових символів. Через природу процесу авторизації необхідно, щоб API-ключ був загальнодоступним. Сам по собі API-ключ не дає доступу до даних користувача Trello. Приклад дошки можна побачити на рисунку 4.3 [24]. Однак, API токени не повинні бути загальнодоступними, оскільки вони надають доступ до даних користувача. Під API-ключем на цій же сторінці є гіперпосилання "Токен", натиснувши на яке, ви перейдете на наступну сторінку, яка дозволяє отримати токен.

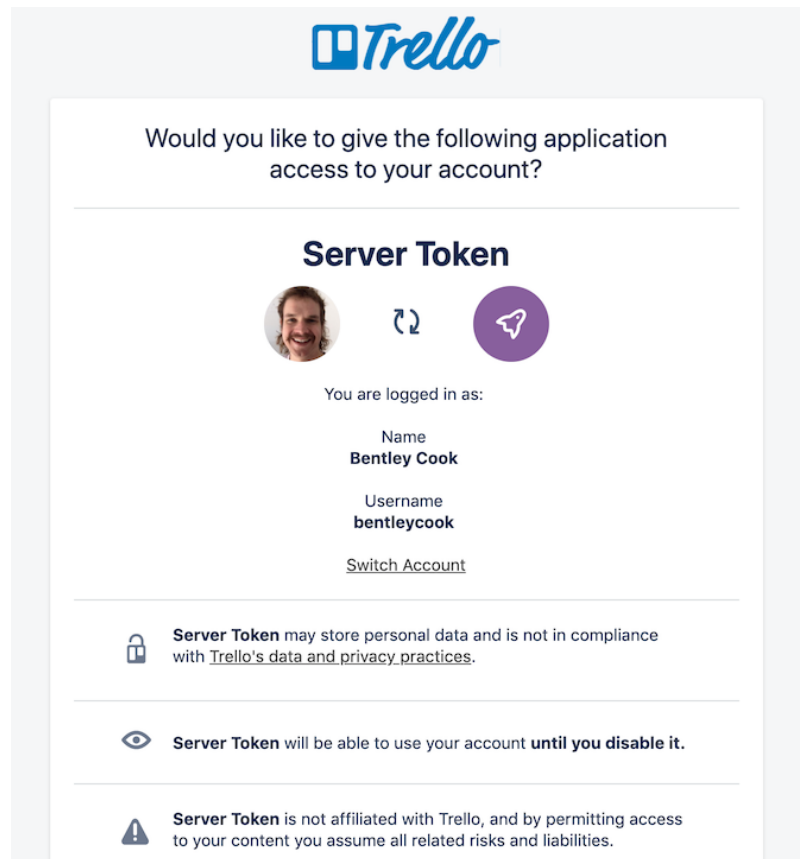


Рисунок 4.3 – Приклад Trello Token [24]

Ви можете читати та записувати дані Trello за допомогою API токенів та ключів Trello.

Далі потрібно перейти до необхідних методів за допомогою офіційного Trello Rest API, щоб написати код для реалізації впровадження, який містить [25]:

- Create a Board;
- Create a new List;
- Create a new Card;
- Update a Card;
- Delete a Board;
- Get a Board.

Та після цього можна побачити опис методів та параметрів які зображені на рисунку 4.4

Create a Board

POST /1/boards/

Create a new board.

Request

QUERY PARAMETERS

name **REQUIRED**

string

The new name for the board. 1 to 16384 characters long.

Min length: 1, Max length: 16384

defaultLabels

boolean

Determines whether to use the default set of labels.

Default: true

defaultLists

boolean

Determines whether to add the default set of lists to a board (To Do, Doing, Done). It is ignored if `idBoardSource` is provided.

Default: true

desc

string

A new description for the board, 0 to 16384 characters long

Min length: 0, Max length: 16384

idOrganization

string

The id or name of the Workspace the board should belong to.

Pattern: `^[0-9a-fA-F]{24}$`

Рисунок 4.4 – Приклад опису методів [25]

4.2 Програмне забезпечення

На початку виконання роботи потрібно сформуванати всі необхідні змінні та додати їх до колекції середовищ з метою автоматизації управління проектом. Змінні середовища повинні бути створені так, як показано на рисунку. Повне значення кожної змінної:

- trelloKey - ключ для роботи з trello;
- trelloToken - токен для роботи з trello;
- boardID - id дошки для взаємодії з другими методами;
- idListTodo - id листа для складання плану;
- IdListDone - ідентифікатор листа для визначення статусу виконано;
- idCard - id картки;
- boardName - назва дошки;
- increment - змінна для формування індивідуальної назви дошки.

✓	trelloKey	48859f892926f3af3c74581d0c89...
✓	trelloToken	15daea39866e0cb2c339a4369dd...
✓	boardId	607de740f2943a26bbcf4a0b
✓	idListTodo	60732a7295818c0a8aa6d532
✓	idListDone	6071ea3a7c919e4b7a65e88d
✓	idCard	6071ea3f2187297db42fe9de
✓	boardName	InternationalProject

Рисунок 4.5 – Змінні середовища Trello

Це буде найважливіший і перший крок у впровадженні системи. Саме створення дошки буде, тому що надалі на ній будуть створюватися всі суб'єкти. Робиться це за методом http POST: <https://api.trello.com/1/boards/?name={{boardName}}&key={{trelloKey}}&token={{trelloToken}}>.

З метою уникнення появи дублікатів та уніфікації даних, перед відправкою методу слід створити унікальне ім'я дошки. Для цього в якості

ідентифікатора дошки можна просто використовувати унікальний ідентифікатор системи.

```
function getUniqueBoardName () {
    if( pm.environment.get("increment") === undefined)
    {
        pm.environment.set("increment", "0");
    }
    let variableForName =
pm.environment.get("increment");
    const boardName = "My board name " +
++variableForName;
    pm.environment.set("increment", variableForName);
    return boardName;
}
```

Ми перевіряємо статус коду після одержання відповіді від сервера. Так само в майбутньому статус коду буде перевірятися на всіх методах.

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Для цього ми перевіряємо, що дошка створена з вказаною назвою, що вона приватна, і записуємо в змінну id дошки, щоб потім можна було створювати картки.

```
pm.test("Board should be created", function () {
  pm.expect(jsonData.name).to.eql(pm.environment.get("boardName"));
  pm.expect(jsonData.closed).to.eql(false);
});
pm.test("Board should be private", function () {
  pm.expect(jsonData.prefs.permissionLevel).to.eql("private");
});
```

Як тільки ця дошка буде вдало створена, необхідно створити два листи зі статусами TODO та DONE. Створивши ці статуси, можна відстежувати стан активних завдань і відповідно оновлювати їх. Спосіб створення - метод http POST:
<https://api.trello.com/1/lists?key={{trelloKey}}&token={{trelloToken}}&name=TODO&idBoard={{boardId}}>.

Як ми вже згадували раніше, в цих методах є перевірка за кодом статусу, а також перевірка за назвою, яка є специфічною для кожного листа.

```
pm.test(" the name of the list is DONE", function () {
  var jsonData = pm.response.json();
  pm.expect(jsonData.name).to.eql("DONE");
});
```

Після цього листи створюються саме під ту дошку, яка має вказаний boardId.

```

pm.test("list was created in the desired board",
function () {
    const Id =
postman.getEnvironmentVariable("boardId");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idBoard).to.eql(Id);
});

```

Далі формується картка з заданням:
<https://api.trello.com/1/cards?key={{trelloKey}}&token={{trelloToken}}&name=Postman&idList={{idListTodo}}>

При отриманні відповіді перевіряється створення під дошку з вказаним boardID та отримується лист зі статусом TODO.

```

pm.test("test that the card was created in the desired
TODO list", function () {
    const Id =
postman.getEnvironmentVariable("idListTodo");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idList).to.eql(Id);
});
pm.test("list was created in the desired board", function
() {
    const Id = postman.getEnvironmentVariable("boardId");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idBoard).to.eql(Id);
});

```

Крім того, в метод внесена перевірка умови на створення картки з вказаним ім'ям у методі.

```
pm.test(" the name of the list is Postman", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.eql("Postman");
});
```

Це робиться за допомогою методу http PUT. Після цього картка завдання переходить зі списку TODO до списку DONE:
<https://api.trello.com/1/cards/:id?key={{trelloKey}}&token={{trelloToken}}&name=Postman&idList={{idListDone}}>.

Коли вона повертається назад, перевіряється, чи вона все ще знаходиться на потрібній дошці, і чи не переміщена вона на іншу літеру.

```
pm.test("test that the card was created in the desired
TODO list", function () {
    const Id =
postman.getEnvironmentVariable("idListTodo");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idList).to.eql(Id);
});

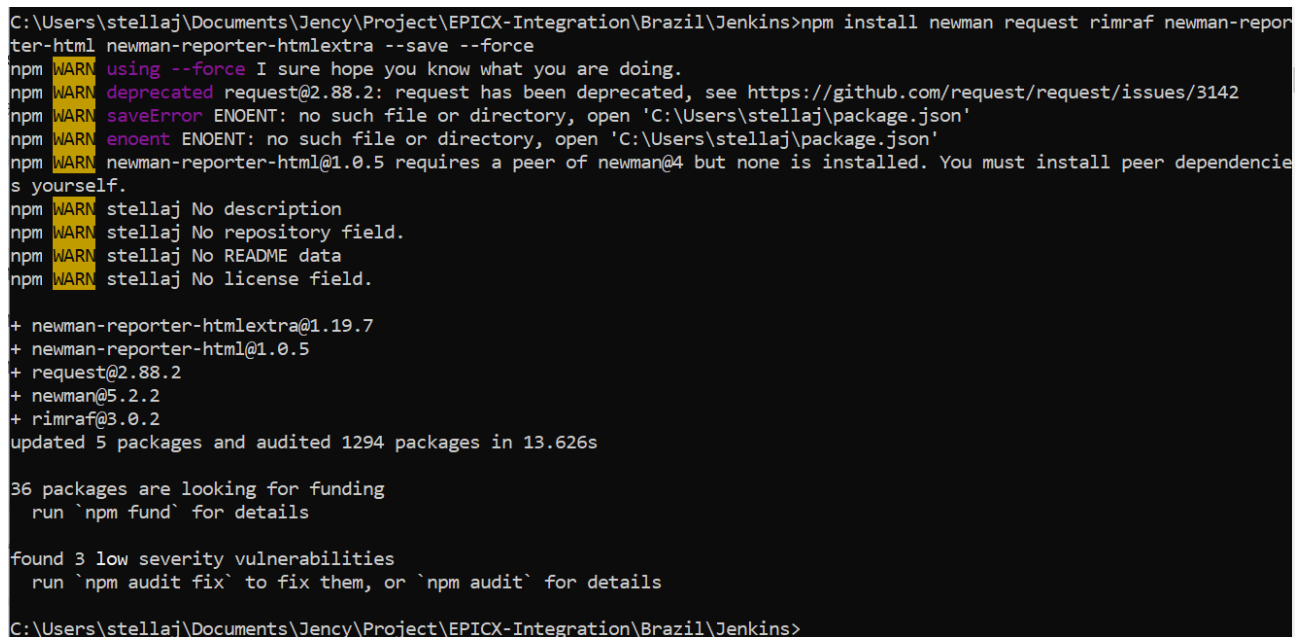
pm.test("list was created in the desired board", function
() {
    const Id = postman.getEnvironmentVariable("boardId");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idBoard).to.eql(Id);
});
```

Вже в останньому порядку буде видалятися дошки за допомогою метода DELETE:

`https://api.trello.com/1/boards/:id?key={{trelloKey}}&token={{trelloToken}}`.

І на останок перевірка, що дошка видалена методом GET:
`https://api.trello.com/1/boards/:id?key={{trelloKey}}&token={{trelloToken}}`.

За рахунок до того встановленого newman використовуємо команду:
`npm install -g newman`. Як встановлюється newman можна побачити на
 рисунку 4.6.



```
C:\Users\stellaj\Documents\Jency\Project\EPIX-Integration\Brazil\Jenkins>npm install newman request rimraf newman-reporter-html newman-reporter-htmlextra --save --force
npm WARN using --force I sure hope you know what you are doing.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\stellaj\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\stellaj\package.json'
npm WARN newman-reporter-html@1.0.5 requires a peer of newman@4 but none is installed. You must install peer dependencies yourself.
npm WARN stellaj No description
npm WARN stellaj No repository field.
npm WARN stellaj No README data
npm WARN stellaj No license field.

+ newman-reporter-htmlextra@1.19.7
+ newman-reporter-html@1.0.5
+ request@2.88.2
+ newman@5.2.2
+ rimraf@3.0.2
updated 5 packages and audited 1294 packages in 13.626s

36 packages are looking for funding
  run `npm fund` for details

found 3 low severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details

C:\Users\stellaj\Documents\Jency\Project\EPIX-Integration\Brazil\Jenkins>
```

Рисунок 4.6 – Встановлення newman

Виконавши експорт колекції та змінних середовищами у форматі json файлу, відтворюємо колекцію з середовищами через термінал і спостерігаємо за результатом роботи та відповідями з логами. Приклад наведено на рисунку 4.7

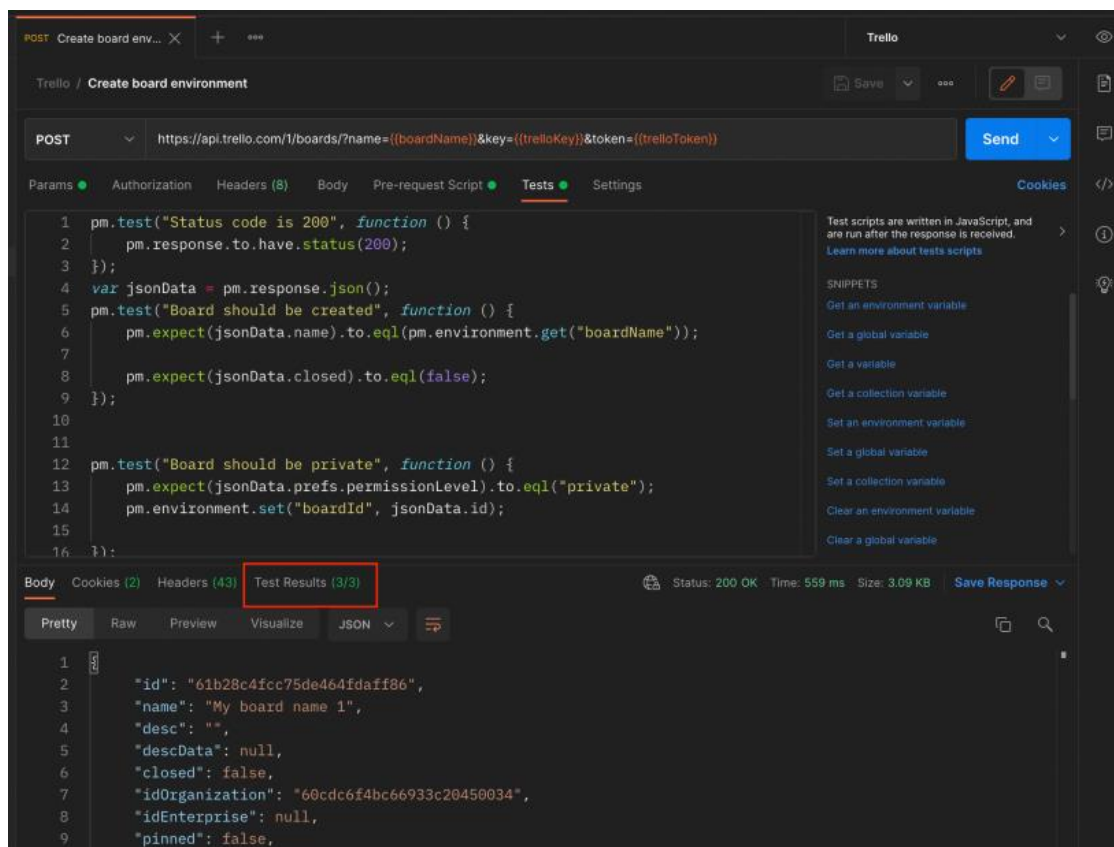


Рисунок 4.7 – Результат обробки

Результат у Trello можна побачити на рисунку 4.8

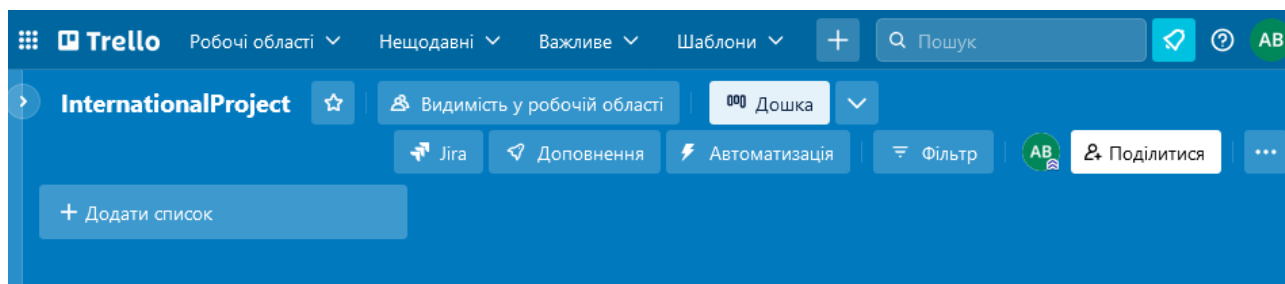


Рисунок 4.8 – Дошка Trello

Методика створення листа була успішно виконана на рисунку 4.9.

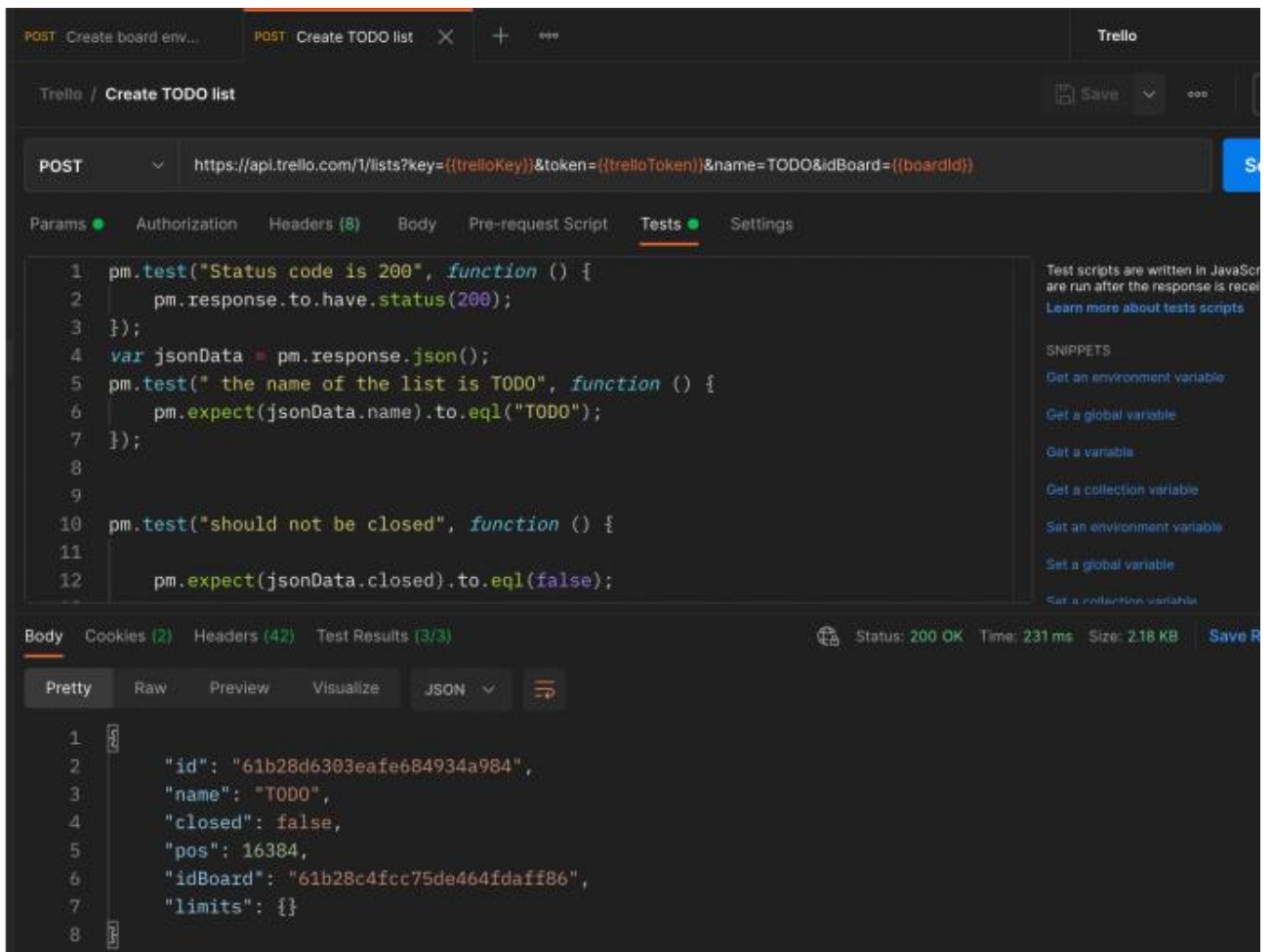


Рисунок 4.9 – Створення листа

Та успішна поява створеного листа на сайті на рисунку 4.10.

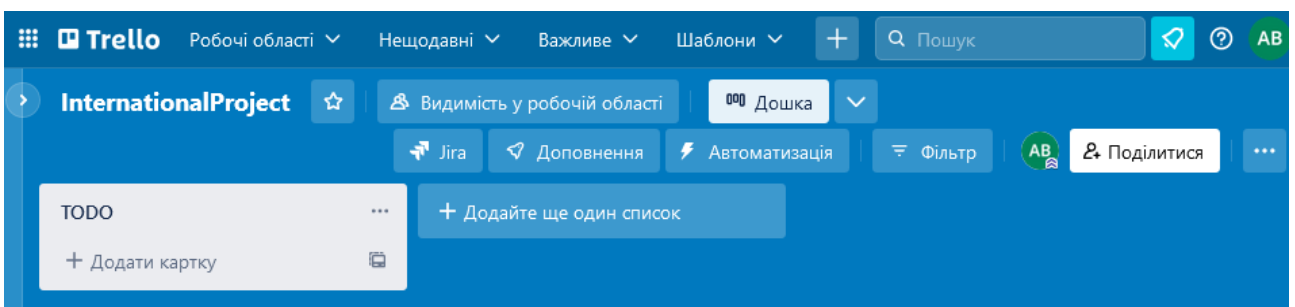


Рисунок 4.10 – Відображення листа на сайті

Здійснення методу створення картки зображено на рисунку 4.11.

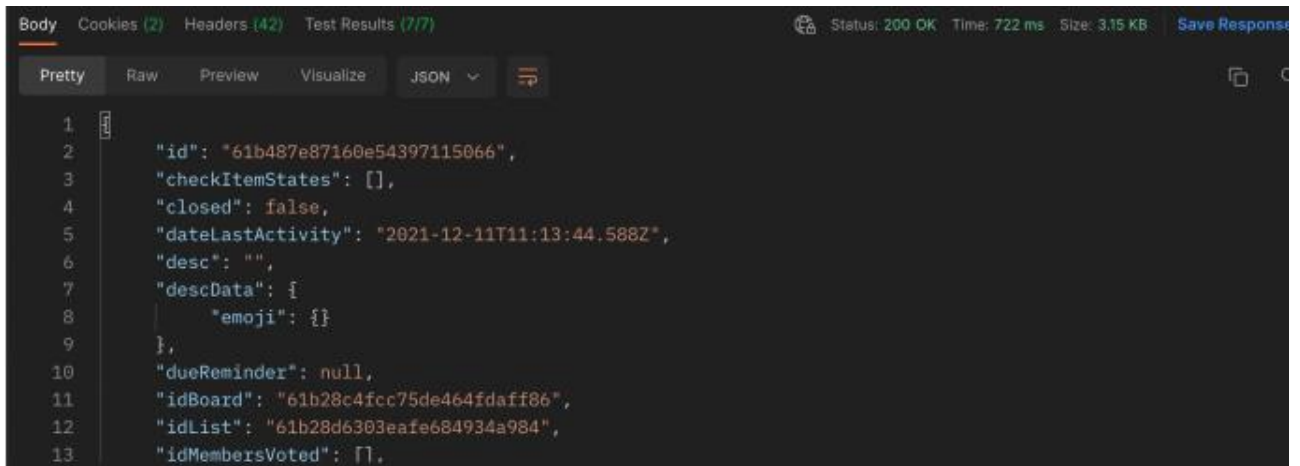


Рисунок 4.11 – Створення картки

Та успішний вивід створеної нової картки на рисунку 4.12.

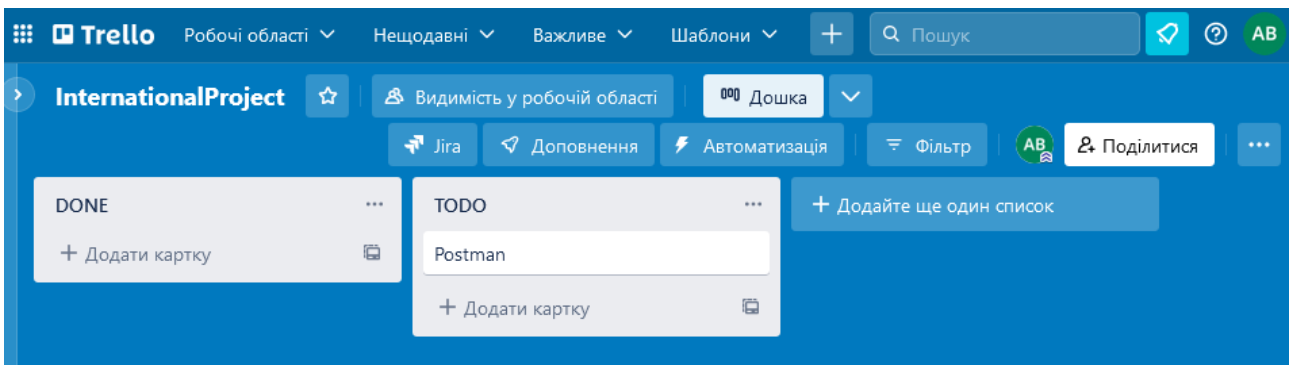


Рисунок 4.12 – Створення картки на дошці Trello

Результат поновлення картки та переходу на іншу дошку на рисунку 4.13.

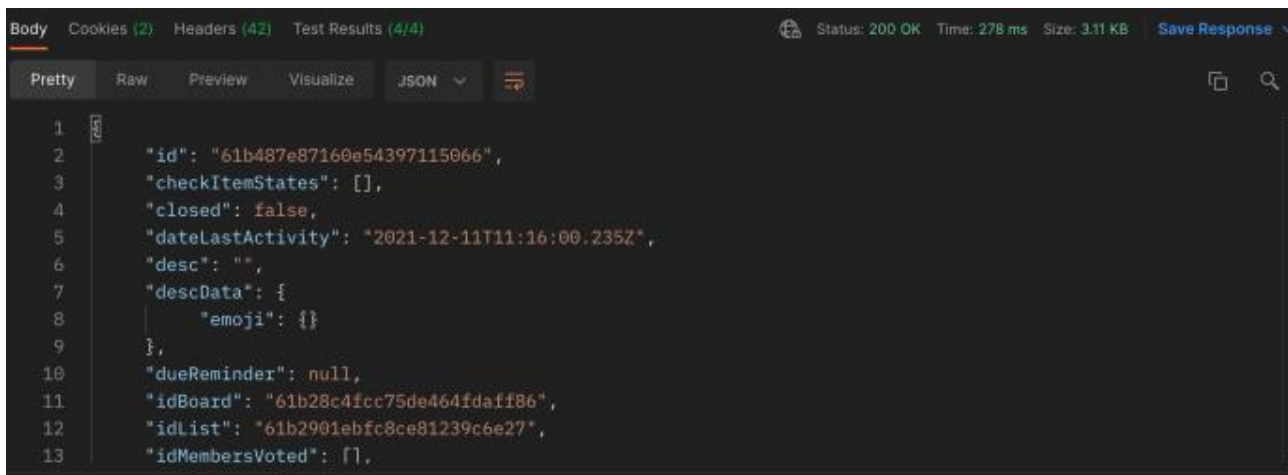


Рисунок 4.13 – Поновлення картки

Остаточний результат поновленої карти на рисунку 4.14

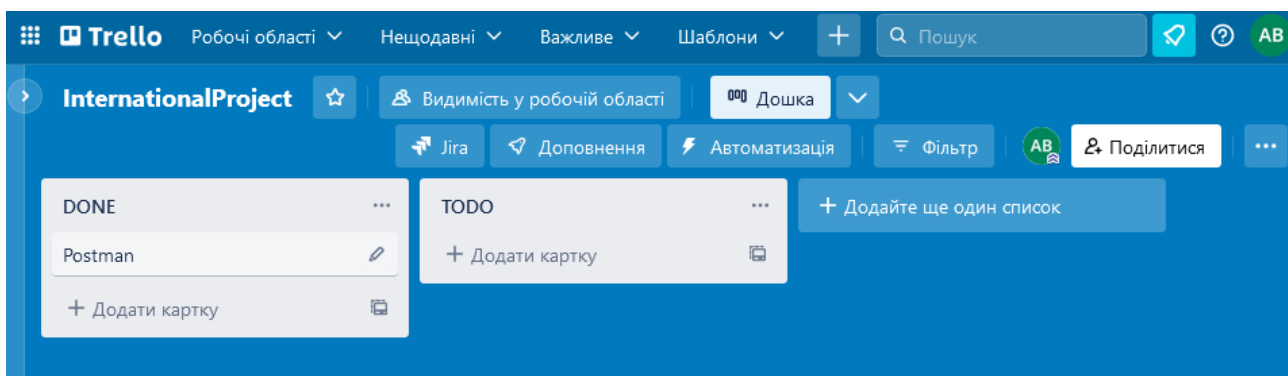


Рисунок 4.14 – Відображення картки на сайті

Після того нам потрібно виконати команду видалення дошки на рисунку 4.15.

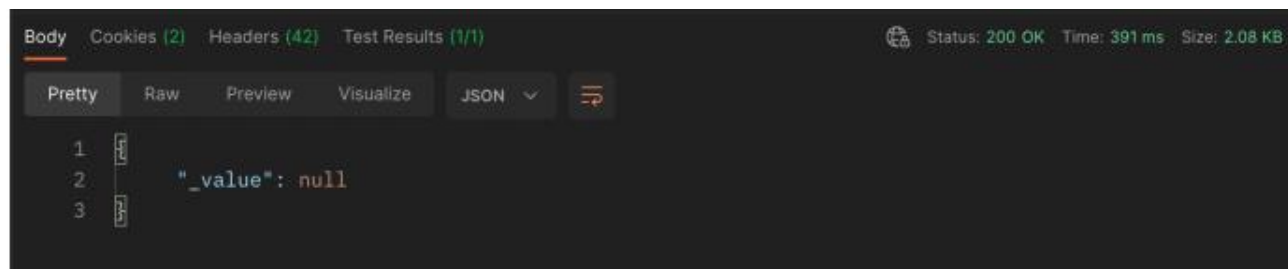


Рисунок 4.15 – Видалення дошки

Треба запустити за рахунок автоматичного запуску як на рисунку 4.16.

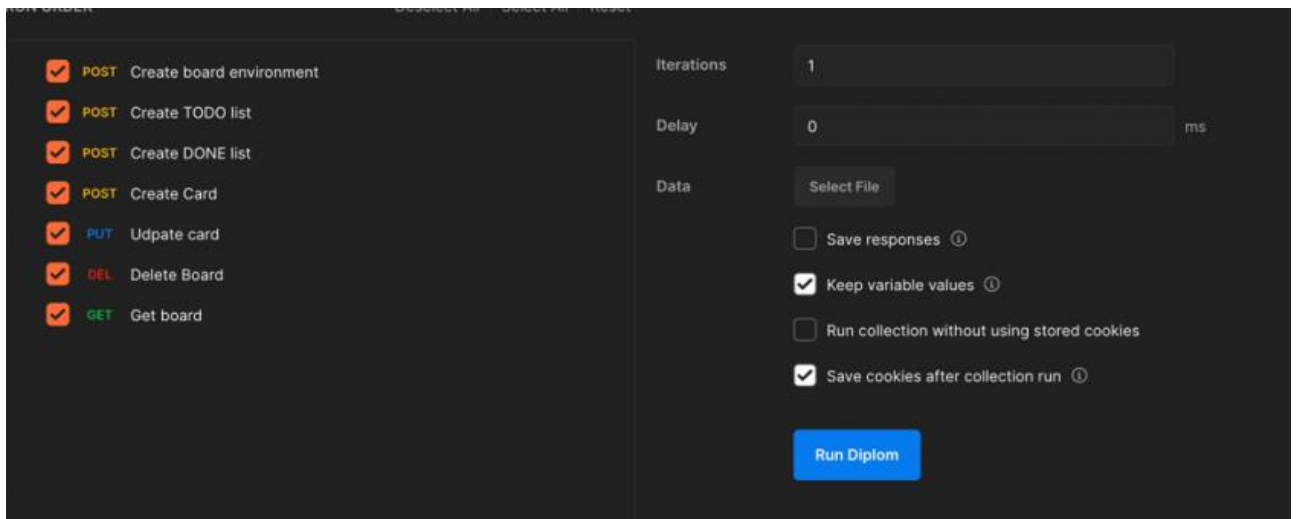


Рисунок 4.16 – Старт колекції

На рисунку 4.17 можна подивитись, що всі тести виконані та правильні. В кінці отримано жаданий результат.

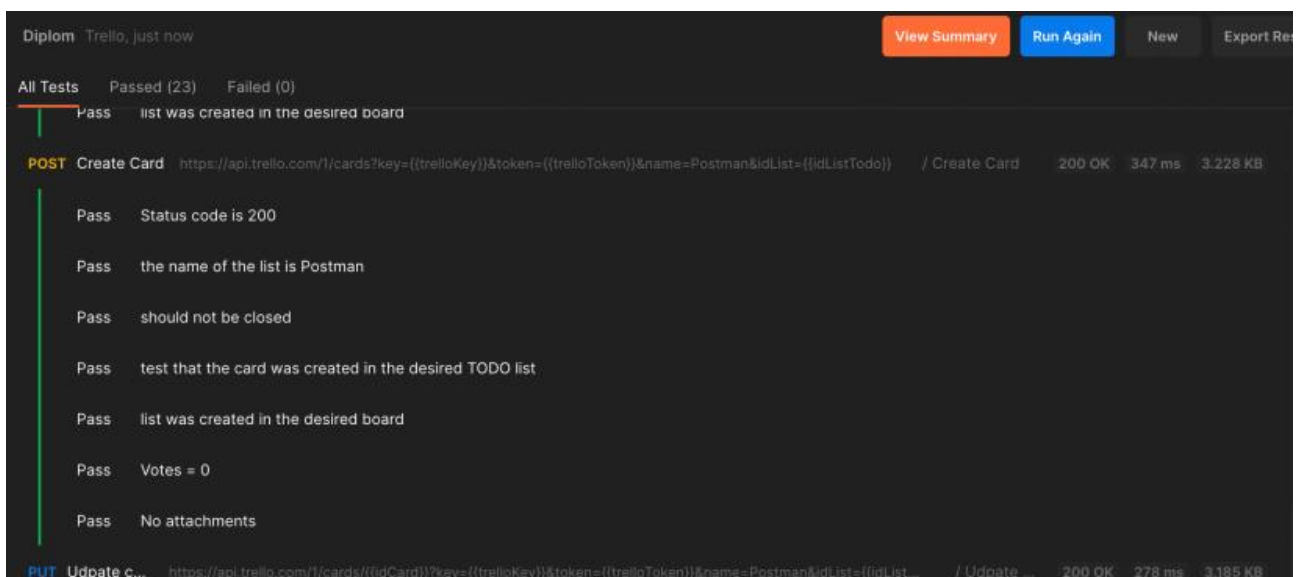


Рисунок 4.17 – Підсумок виконання колекції

Попередньо завантаживши колекцію та оточення, запустимо їх через термінал за допомогою команди: `newman run Diplom.postman_collection.json -e Trello.postman_environment.json`. Після того можемо перевірити результат у терміналі як на рисунку 4.18.

```

Delete Board
DELETE https://api.trello.com/1/boards/61b48b9a16762b815943f021?key=48859f892926f3af3c74581d0c894
✓ Status code is 200

Get board
GET https://api.trello.com/1/boards/61b48b9a16762b815943f021?key=48859f892926f3af3c74581d0c894
✓ Board has been deleted

```

	executed	failed
iterations	1	0
requests	7	0
test-scripts	7	0
prerequisite-scripts	1	0
assertions	23	0

```

total run duration: 2.2s
total data received: 3.5kB (approx)
average response time: 300ms [min: 162ms, max: 566ms, s.d.: 128ms]
serdu4ok@MacBook-Air-Serdu4ok Downloads %

```

Рисунок 4.18 - Підсумок роботи через термінал

ВИСНОВКИ

В рамках проекту було проаналізовано ринок та на основі аналізу обрано найкращу платформу. Для розробки інформаційної технології автоматизованого управління проектами були використані інструменти Postman. Цей інструмент використовується для тестування API запитів шляхом відправки JSON запиту на сервер, відповіді на заголовки і коди статусу, а також відправки фактичних запитуваних даних на консоль. З його допомогою є можливість створювати дошки, листи з певними статусами, картки, і навіть видаляти їх та перевіряти статус.

Система є масштабованою, і є можливість впроваджувати її в режимі безперервної інтеграції та безперервної доставки. Програма була протестована на декількох типах пошти, в тому числі: Автоматична (Postman), Ручна (Postman) та з терміналу (Newman).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Serrador P. Does Agile work? – A quantitative analysis of agile project success [Electronic resource] / Pedro Serrador, Jeffrey K. Pinto // International Journal of Project Management. – 2015. – Vol. 33, no. 5. – P. 1040–1051. – Mode of access: <https://doi.org/10.1016/j.ijproman.2015.01.006>
2. Comella-Dorda S., Lohiya S., Speksnijder G. An operating model for company-wide agile development [Electronic resource] // McKinsey Digital. — 2016. — Mode of access: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/an-operating-model-for-company-wide-agile-development>
3. Ika L. A. Project Success as a Topic in Project Management Journals [Electronic resource] / Lavagnon A. Ika // Project Management Journal. – 2009. – Vol. 40, no. 4. – P. 6–19. – Mode of access: <https://doi.org/10.1002/pmj.20137>
4. Nicholls G. M. Determining When Simplified Agile Project Management Is Right for Small Teams [Electronic resource] / Gillian M. Nicholls, Neal A. Lewis, Ted Eschenbach // Engineering Management Journal. – 2015. – Vol. 27, no. 1. – P. 3–10. – Mode of access: <https://doi.org/10.1080/10429247.2015.11432031>
5. Моделі життєвого циклу ПЗ [Електронний ресурс] // Pidru4niki. – Режим доступу: https://pidru4niki.com/1701120547727/informatika/modeli_zhittyevogo_tsiklu
6. Wysocki R. K. Effective project management: Traditional, agile, extreme / Robert K. Wysocki. – 6th ed. – Indianapolis, Ind : Wiley, 2012. – 774 p.
7. Thesing T. Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project [Electronic resource] / Theo Thesing, Carsten Feldmann, Martin Burchardt // Procedia Computer Science. – 2021. – Vol. 181. – P. 746–756. – Mode of access: <https://doi.org/10.1016/j.procs.2021.01.227>
8. Clancy T. The standish group report [Electronic resource] / Tom Clancy. – [S. l.] : Chaos report, 2014. – 12 p. – Mode of access:

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

9. Srivastava A. SCRUM model for agile methodology [Electronic resource] / Apoorva Srivastava, Bhardwaj Sukriti, Saraswat Shipra // 2017 International Conference on Computing, Communication and Automation (ICCCA). – 2017. – Mode of access: <https://www.grafiati.com/uk/>

10. Beck K. Manifesto for Agile Software Development [Electronic resource] / Kent Beck // Manifesto for Agile Software Development. – Mode of access: <http://agilemanifesto.org/>

11. Schwaber K. The scrum guide [Electronic resource] / Ken Schwaber, Jeff Sutherland // Scrum Alliance. – 2020. – Mode of access: <https://billlewisstraining.com/wp-content/uploads/2017/02/PMP-Agile-Study-Materials.pdf>

12. Schwaber K. Agile Project Management with Scrum / Ken Schwaber // Microsoft. – 2004

13. Cho J. J. An Exploratory Study on Issues and Challenges of Agile Software Development with Scrum [Electronic resource] : Theses. USU. Management Information Systems. Electronic dissertations / Cho Juyun Joey. – [S. l.]. – Mode of access: <http://digitalcommons.usu.edu/etd/599/>

14. Abbas J. Quintessence of Traditional and Agile Requirement Engineering [Electronic resource] / Jalil Abbas // Journal of Software Engineering and Applications. – 2016. – Vol. 09, no. 03. – P. 63–70. – Mode of access: <https://doi.org/10.4236/jsea.2016.93005>

15. Larman C. Agile and Iterative Development: A Manager's Guide / Craig Larman. – [S. l.] : Addison-Wesley Professional, 2003. – 368 p.

16. Tijmen van Graft. The role of the product owner in CD [Electronic resource] / Tijmen van Graft // Medium. – Mode of access: <https://tijmen-van-graft.medium.com/the-role-of-the-product-owner-in-cd-39f17fce18>

17. 10th Annual State of Agile Report // VERSIONONE.COM. – 2016

18. Why Should You Use ClickUp? | ClickUp [Electronic resource] //

ClickUp. – Mode of access: <https://clickup.com/blog/why-clickup/>

19. Asana Work Management - Features, Uses & Product [Electronic resource] // Asana. – Mode of access: <https://asana.com/product>

20. Stanke B. How to Build a Kanban Board in Notion – Helping Companies Deliver More Value Through Better Process Management [Electronic resource] // BOB STANKE. – Mode of access: <https://www.bobstanke.com/blog/how-to-build-a-notion-kanban-board>

21. Morpus N. What is Jira? A Beginner's Guide to Using Jira in 2022 [Electronic resource] / Nicholas Morpus // The Motley Fool. – Mode of access: <https://www.fool.com/the-ascent/small-business/project-management/articles/what-is-jira/>

22. Wiles J. Employees Increasingly Seek Value and Purpose at Work [Electronic resource] / Jackie Wiles // Gartner. – 2022. – Mode of access: <https://www.gartner.com/en/articles/employees-seek-personal-value-and-purpose-at-work-be-prepared-to-deliver>

23. How To Use Command Line Integration With Newman In Postman? [Electronic resource] // Software Testing Help. – Mode of access: <https://www.softwaretestinghelp.com/postman-newman/>

24. API Introduction [Electronic resource] // Atlassian Developer. – Mode of access: <https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/>

25. The Trello REST API [Electronic resource] // Atlassian Developer. – Mode of access: <https://developer.atlassian.com/cloud/trello/rest/api-group-actions/#api-actions-id-delete>

ДОДАТОК А
Текст програми

```

function getUniqueBoardName () {
  if( pm.environment.get("increment") === undefined)
  {
    pm.environment.set("increment", "0");
  }
  let variableForName = pm.environment.get("increment");
  const boardName = "My board name " +
  ++variableForName;
  pm.environment.set("increment", variableForName);
  return boardName;
}

pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});

pm.test("Board should be created", function () {
  pm.expect(jsonData.name).to.eql(pm.environment.get("boardName"));
  pm.expect(jsonData.closed).to.eql(false);
});
pm.test("Board should be private", function () {
  pm.expect(jsonData.prefs.permissionLevel).to.eql("private");
  pm.environment.set("boardId", jsonData.id);
});

pm.test(" the name of the list is DONE", function () {
  var jsonData = pm.response.json();
  pm.expect(jsonData.name).to.eql("DONE");
});

pm.test("list was created in the desired board",
function () {
  const Id =
postman.getEnvironmentVariable("boardId");
  var jsonData = pm.response.json();
  pm.expect(jsonData.idBoard).to.eql(Id);
});

pm.test("test that the card was created in the desired
TODO list", function () {
  const Id =
postman.getEnvironmentVariable("idListTodo");

```

```

        var jsonData = pm.response.json();
        pm.expect(jsonData.idList).to.eql(Id);
    });

    pm.test("list was created in the desired board",
function () {
    const Id = postman.getEnvironmentVariable("boardId");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idBoard).to.eql(Id);
});

    pm.test(" the name of the list is Postman", function ()
{
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.eql("Postman");
});

    pm.test("test that the card was created in the desired
TODO list", function () {
    const Id =
postman.getEnvironmentVariable("idListTodo");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idList).to.eql(Id);
});

    pm.test("list was created in the desired board",
function () {
    const Id =
postman.getEnvironmentVariable("boardId");
    var jsonData = pm.response.json();
    pm.expect(jsonData.idBoard).to.eql(Id);
});

```

ДОДАТОК Б

Презентація



Аналіз та програмна реалізація ланцюга засобів для керування міжнародним проектом

Виконав студент КНТ-211м Безугла А.М.

Керівник к.т.н., професор Табунщик Г.В.

Рисунок Б.1 – Слайд 1

Об'єкт дослідження – процес автоматизування управління проектом.

Предмет дослідження – гнучкі методології розробки.

Мета роботи – вивчення особливостей та практик управління проектами та автоматизація управління проектами.

Результати - проведено аналіз ринку діяльності конкурентів та поточної ситуації на ньому. Проведено порівняння конкурентів. Створено інформаційну технологію автоматизованого управління проектами за рахунок платформи API Postman.

Одним з пріоритетних шляхів підвищення якості роботи є аналіз та впровадження засобів для керування ланцюжка міжнародного проекту. Ці інструменти можуть бути використані для підвищення ефективності бізнес-процесів, а також для скорочення витрат і забезпечення інших переваг, таких як підвищення задоволеності клієнтів і поліпшення морального духу співробітників.

Рисунок Б.2 – Слайд 2

Аналіз предметної області

Види методологій



Класичні методології

У такій моделі кожен етап активується після завершення попереднього етапу.

Гнучкі методології

Організовує методи та інструменти управління проєктами у фази або процеси проєкту у стандартизований спосіб

3

Рисунок Б.3 – Слайд 3

Класичні методології



4

Рисунок Б.4 – Слайд 4

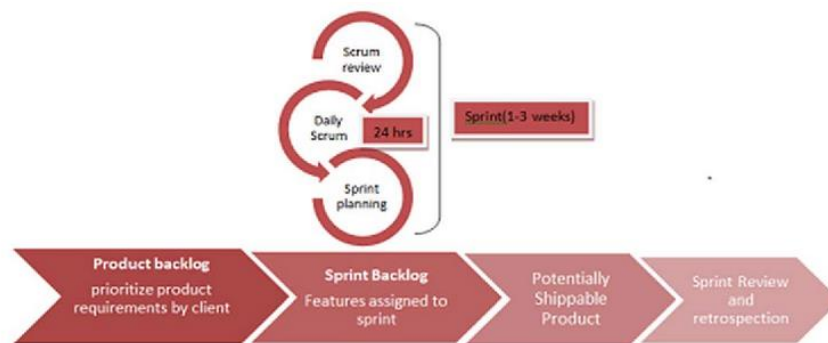
Гнучкі методології

Назва	Ключові моменти	Унікальні особливості	Недоліки
Екстремальне програмування (XP)	Невеликі команди, щодня проводяться зустрічі	Постійне вдосконалення проекту	Підходить для індивідуальних практик
Scrum	Самостійна, ітерації - 2-4 тижні	Високий ступінь комунікації та взаємодії всередині команди	Ризик зростання термінів реалізації проекту
"Розробка керована функціональністю" (fdd)	П'ятиетапний процес, дуже короткі ітерації	Легкість методу, об'єктне моделювання	Зосереджена лише на розробці та впровадженні,
"Ощадлива розробка" (lean)	Застосування засобів візуалізації, розробка через випробування, невеликі ітерації	Головними функціями програмного забезпечення є ті, які є важливими для замовника	Рішення приймаються тривалий час, придатні лише для малих проектів
Kanban	Самоорганізований колектив, всі зосереджені на виконанні поставленого завдання	Обмеження по терміну виконання немає, але є обмеження за кількістю "роботи зараз"	Важко методика є її тривалий характер

5

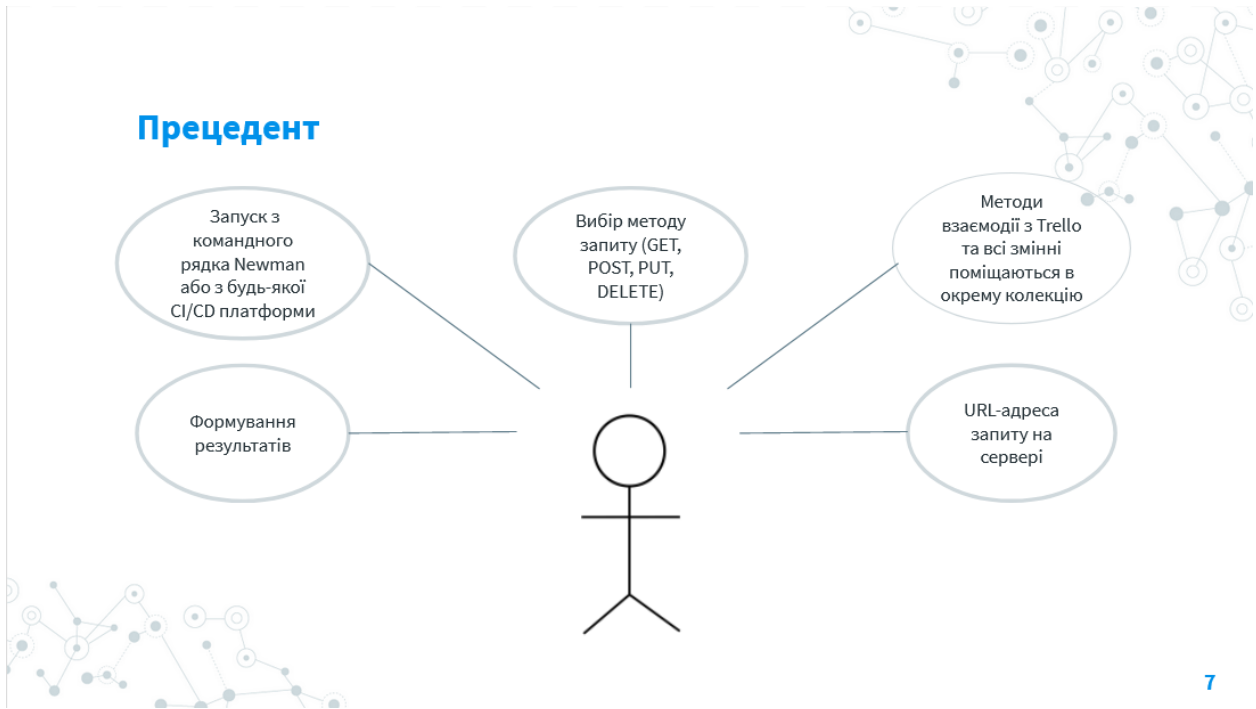
Рисунок Б.5 – Слайд 5

Scrum



6

Рисунок Б.6 – Слайд 6



7

Рисунок Б.7 – Слайд 7

Вибір мови програмування та середовища розробки

Назва	Ключові моменти	Недоліки
SoapUI	Створення комплексних сценаріїв та підтримка асинхронного тестування, створити тест легко та швидко за допомогою технологій перетягування об'єктів «мишкою» (Drag and drop)	Обмеження в безкоштовній версії, платна підписка коштує чималих грошей
Tricentis Tosca	Підтримка багатьох масивів протоколів, тести API можуть використовуватися як на мобільних, так і на браузерних і пакетних програмах	
Rest-Assured	Має цілий набір вбудованих функціоналів, наявність яких означає, що користувачеві не доведеться кодувати заново	Вимагають кваліфікованих умінь і багато зусиль для імплементації правильної платформи
Karate DSL	Здатний запускати тестування та генерувати звіти як будь-який інший стандартний проект Java	
Postman	Багатий інтерфейс робить цей інструмент простим та зручним у використанні; має функції Run, Test, Document і Monitoring	Може мати недоліки для деяких типів проектів, і ці недоліки вимагають пильної оцінки

8

Рисунок Б.8 – Слайд 8

Вибір мови програмування та середовища розробки

- ◎ мова програмування JavaScript
- ◎ платформа API Postman
- ◎ програма для запуску збору даних у командному рядку для Postman - Newman
- ◎ вебдодаток Trello

9

Рисунок Б.9 – Слайд 9

Змінні середовища Trello

Назва	Значення
trelloKey	ключ для роботи з Trello
trelloToken	токен для роботи з trello
boardID	id дошки для взаємодії з другими методами
idListTodo	id листа для складання плану
IdListDone	ідентифікатор листа для визначення статусу виконано
idCard	id картки
boardName	назва дошки
increment	змінна для формування індивідуальної назви дошки

Рисунок Б.10 – Слайд 10

Приклад автоматичного запуску

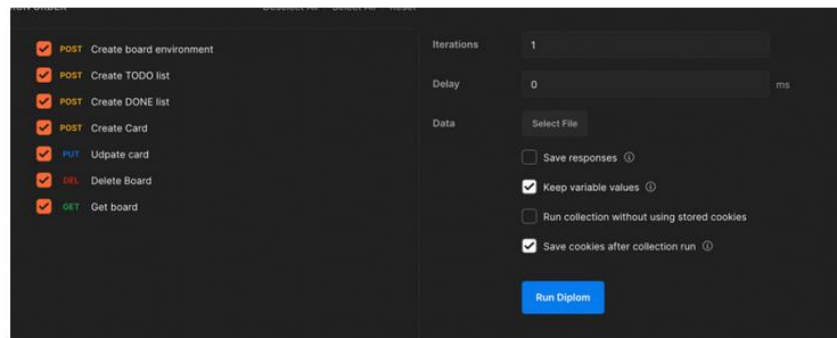


Рисунок Б.11 – Слайд 11

Підсумок виконання колекції

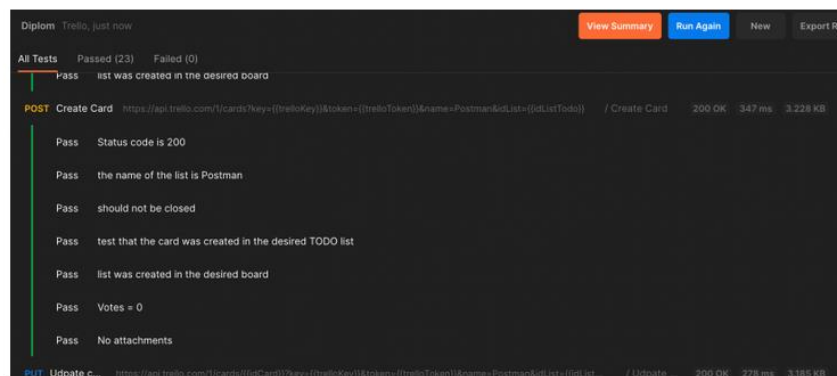


Рисунок Б.12 – Слайд 12

Підсумок роботи через термінал

Delete Board

DELETE https://api.trello.com/1/boards/61b48b9a16762b815943f021?key=48859f892926f3af3c74581d0c

✓ Status code is 200

Get board

GET https://api.trello.com/1/boards/61b48b9a16762b815943f021?key=48859f892926f3af3c74581d0c894

✓ Board has been deleted

	executed	failed
iterations	1	0
requests	7	0
test-scripts	7	0
prerequisite-scripts	1	0
assertions	23	0

total run duration: 2.2s

total data received: 3.5kB (approx)

average response time: 300ms [min: 162ms, max: 566ms, s.d.: 128ms]

Рисунок Б.13 – Слайд 13

Приклад відображення картки на сайті

Trello

Робочі області

Нещодавні

Важливе

Шаблони

Пошук

InternationalProject

Видимість у робочій області

Дошка

Ліга

Доповнення

Автоматизація

Фільтр

AB

Поділитися

DONE

Postman

+ Додати картку

TODO

+ Додати картку

+ Додайте ще один список

Рисунок Б.14 – Слайд 14

Висновки

У ході роботи було проаналізовано ринок та обрано найбільш оптимальну платформу. Було розроблено Інформаційну технологію автоматизованого управління проектами за допомогою інструментарію Postman.

Проведено тестування на декількох видах а саме:

- Ручне (Postman)
- Автоматичне(Postman)
- З терміналу (Newman)

15

Рисунок Б.15 – Слайд 15



Дякую за увагу!

16

Рисунок Б.16 – Слайд 16