

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАТИКИ ТА ФІЗИКО-МАТЕМАТИЧНИХ ДИСЦИПЛІН

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт

з дисципліни

УПРАВЛІННЯ ІТ ПРОЄКТАМИ

для студентів спеціальності

F6 Інформаційні системи і технології

Ужгород
2025

Управління ІТ проєктами: методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальності F6 Інформаційні системи і технології факультету інформаційних технологій ДВНЗ «Ужгородський національний університет» / Укладачі: В.С. Морохович, І.М. Лях, П.П. Федорка. Ужгород: ДВНЗ «УжНУ», 2025. 79 с.

У методичних вказівках до виконання лабораторних робіт з курсу «Управління ІТ проєктами» сформульовано теми та мети завдань до виконання лабораторних робіт, вимоги до порядку виконання та змісту звіту по проробленій роботі. Також наведено теоретичні відомості, завдання до виконання лабораторних робіт та перелік контрольних запитань на підсумковий контроль.

Укладачі:

Морохович Василь Степанович – доцент, кандидат фізико-математичних наук, доцент кафедри інформатики та фізико-математичних дисциплін факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;

Лях Ігор Михайлович – доцент, доктор технічних наук, професор кафедри інформатики та фізико-математичних дисциплін факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;

Федорка Павло Павлович – доктор філософії, доцент кафедри програмного забезпечення систем факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;

Рецензенти:

Поліщук Володимир Володимирович – професор, доктор технічних наук, професор кафедри програмного забезпечення систем факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;

Маляр Микола Миколайович – професор, доктор технічних наук, декан факультету математики та цифрових технологій ДВНЗ «Ужгородський національний університет»

*Методичні рекомендації розглянуто та затверджено
на засіданні кафедри програмного забезпечення систем
факультету інформаційних технологій
(протокол №13 від «10» червня 2025 р.)*

*Схвалено та рекомендовано до друку
науково-методичною комісією
факультету інформаційних технологій
(протокол №7 від «12» червня 2025 р.)*

ЗМІСТ

ВСТУП.....	4
ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ	6
САМОСТІЙНА РОБОТА	7
ЛАБОРАТОРНА РОБОТА № 1 «РОБОТА З TASK-МЕНЕДЖЕРОМ TRELLO».....	8
ЛАБОРАТОРНА РОБОТА № 2 «АВТОМАТИЗАЦІЯ ПРОЦЕСІВ У TASK-МЕНЕДЖЕРІ TRELLO»	20
ЛАБОРАТОРНА РОБОТА № 3 «РОБОТА З СИСТЕМОЮ JIRA»	30
ЛАБОРАТОРНА РОБОТА № 4 «РОБОТА ІЗ МЕТОДОЛОГІЄЮ SCRUM».....	46
ЛАБОРАТОРНА РОБОТА № 5 «ОСНОВИ РОБОТИ З GIT ТА GITHUB»	56
ЛАБОРАТОРНА РОБОТА № 6 «ОСНОВИ РОБОТИ З SSH КЛЮЧАМИ НА GITHUB»	67
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДЖЕРЕЛ.....	79

ВСТУП

Вивчення дисципліни «Управління ІТ проєктами» є ключовим етапом для фахівців, які прагнуть ефективно керувати складними процесами розробки та впровадження інформаційних технологій. У сучасному світі, де цифровізація охоплює всі сфери життя, успішність будь-якої організації значною мірою залежить від її здатності вчасно та якісно реалізовувати ІТ ініціативи. Методичні рекомендації покликані стати надійним дороговказом у світі проєктного менеджменту, сфокусованого на унікальних особливостях ІТ галузі.

Специфіка ІТ проєктів полягає у їхній високій динамічності, необхідності постійно адаптуватися до мінливих вимог, швидкій зміні технологій та великій ролі людського фактора. На відміну від традиційних проєктів, ІТ проєкти часто стикаються з невизначеністю на початкових етапах, потребують гнучких підходів до планування та реалізації, а також вимагають глибокого розуміння як технічних аспектів, так і бізнес-процесів. Саме тому компетентний управління ІТ проєктами вимагає не лише знання загальних методологій, але й глибокого розуміння індустріальних особливостей, найкращих практик та інструментів, що застосовуються саме в ІТ середовищі.

Методичні рекомендації структуровано таким чином, щоб забезпечити здобувачів освіти комплексними знаннями та практичними навичками з управління ІТ-проєктами в сучасному динамічному середовищі розробки. В межах лабораторних робіт розглянуто ключові аспекти організації командної взаємодії та контролю за виконанням завдань із використанням популярних інструментів управління, таких як Trello і Jira, включаючи автоматизацію робочих процесів.

Особлива увага приділяється практичному засвоєнню методологій гнучкого управління проєктами, зокрема Scrum, що дозволяє ефективно планувати, реалізовувати та контролювати ІТ-продукти у командній розробці. Важливою складовою є оволодіння базовими навичками роботи з системами контролю версій Git та платформою GitHub, а також налаштуванням SSH-ключів для безпечної та ефективної взаємодії з репозиторіями.

Усі теми лабораторних робіт спрямовані на розвиток у студентів практичних умінь, необхідних для роботи в реальних ІТ-командах, із врахуванням сучасних стандартів, методів та інструментів управління проєктами.

Відповідно до освітньої програми, вивчення дисципліни сприяє формуванню у здобувачів вищої освіти таких компетентностей:

ІНТ. Здатність розв’язувати складні спеціалізовані задачі та практичні проблеми в області інформаційних систем та технологій, або в процесі навчання, що характеризуються комплексністю та невизначеністю умов, які потребують застосування теорій та методів інформаційних технологій.

ЗК1. Здатність до абстрактного мислення, аналізу та синтезу.

ЗК3. Здатність спілкуватися з представниками інших професійних груп різного рівня (з експертами з інших галузей знань/видів економічної діяльності).

ЗК4. Здатність розробляти проєкти та управляти ними.

ЗК5. Здатність оцінювати та забезпечувати якість виконуваних робіт.

ФК8. Здатність застосовувати інструменти управління проєктами, у тому числі з використанням гнучких методів управління проєктами.

ФК9. Здатність ефективно здійснювати планування, виконання проєктних дій та прийняття проєктних рішень на основі нормативно-методичних положень, стандартів і норм певної прикладної області для управління ІТ проєктом, формувати вимоги відповідності інформаційної системи технічному завданню.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Змістовий модуль 1.

Тема 1. Загальна характеристика проєкту та стратегічне управління ним.

Тема 2. Методології управління ІТ проєктами.

Тема 3. Форми організаційної структури управління проєктом.

Тема 4. Управління процесом ініціювання проєкту.

Тема 5. Планування проєкту. Ієрархічна структура робіт.

Тема 6. Управління часом (тривалістю) проєкту.

Змістовий модуль 2

Тема 7. Планування ресурсів та витрат проєкту.

Тема 8. Управління командою проєкту.

Тема 9. Контроль виконання проєкту.

Тема 10. Управління ризиками в проєктах.

Тема 11. Управління якістю проєктів.

Тема 12. Управління комунікаціями проєкту.

САМОСТІЙНА РОБОТА

№ п/п	Назва теми
1.	Загальна характеристика проєкту та стратегічне управління ним.
2.	Методології управління ІТ проєктами.
3.	Форми організаційної структури управління проєктом.
4.	Управління процесом ініціювання проєкту.
5.	Планування проєкту. Ієрархічна структура робіт.
6.	Управління часом (тривалістю) проєкту.
7.	Планування ресурсів та витрат проєкту.
8.	Управління командою проєкту.
9.	Контроль виконання проєкту.
10.	Управління ризиками в проєктах.
11.	Управління якістю проєктів.
12.	Управління комунікаціями проєкту.

ЛАБОРАТОРНА РОБОТА № 1 (4 години)

«РОБОТА З TASK-МЕНЕДЖЕРОМ TRELLO»

Тема роботи: робота з task-менеджером Trello.

Мета роботи: здобути базові навички у роботі з task-менеджерами, та навчитись використовувати основні функції.

1.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно сформулювати основні вимоги до обраного програмного забезпечення, визначити його функціональні можливості та узагальнити призначення. Звіт оформлюється у вигляді документа з назвою «Постановка завдання».

Постановка завдання: описати технічне завдання на прикладі програмного продукту для управління задачами та організації командної взаємодії.

1.2 Теоретичні відомості



Рис. 1.1 Ілюстрація системи «Trello»

Trello – це вебплатформа, яка призначена для управління проєктами за гнучкою методологією Kanban методом візуалізації дошок із списками задач.

Trello є класичним інструментом управління невеликими проєктами в найрізноманітніших сферах, таких як ІТ, комерційна, творча та інші. Слід відмітити одну з особливостей даної системи – структурованість. За допомогою даного сервісу можна повноцінно організовувати роботу команди, при цьому виявляти максимальну ефективність командної роботи. Як правило, чим простіша і зрозуміліша система, тим більш корисною вона є у використанні. Вона є максимально легкою під час впровадження на проєкті, інтерфейс є інтуїтивно зрозумілим для користувача, а отже не потребує багато часу для адаптації з боку персоналу.

Інтерфейс даної системи представляє собою канбан-дошки (Boards), які можуть вважатися окремими проєктами. Тому, перш за все, розберемося з поняттям Kanban.

Kanban – це одна з гнучких методологій розробки програмного забезпечення, яка реалізована за принципом «зробити вчасно» та візуалізації процесів. Це сприяє розподілу навантаження між виконавцями. Дана концепція була розроблена менеджерами Toyota, які використовували її для виробничої сфери. Суть полягала у тому, щоб до деталей кріпилися картки з основною інформацією, наприклад номер деталі, номер складу, куди надалі прямує партія деталей та який відділ їх відправляє. Після передачі в інший відділ відбувалася заміна карток з відповідною зміною інформації на них. Таким чином здійснювалося спрощення комунікацій та організації процесів.

На сьогоднішній день ця методологія широко використовується в різних сферах, зокрема в ІТ. Слідування принципу «зробити вчасно» дозволяє дотримуватися балансу між безперервним потоком задач та усуненням затрат на очікування виконання цих задач. Метод Kanban дає можливість ефективно організувати взаємодію учасників процесу та забезпечує високий рівень прозорості в ході виконання роботи.

При першому вході в Trello користувач побачить найбільш популярні шаблони дошок, при цьому будь-який з них можна обрати для створення своєї дошки. Або ж скористатися кнопкою «Create» у верхньому правому кутку та додати нову дошку власноруч.

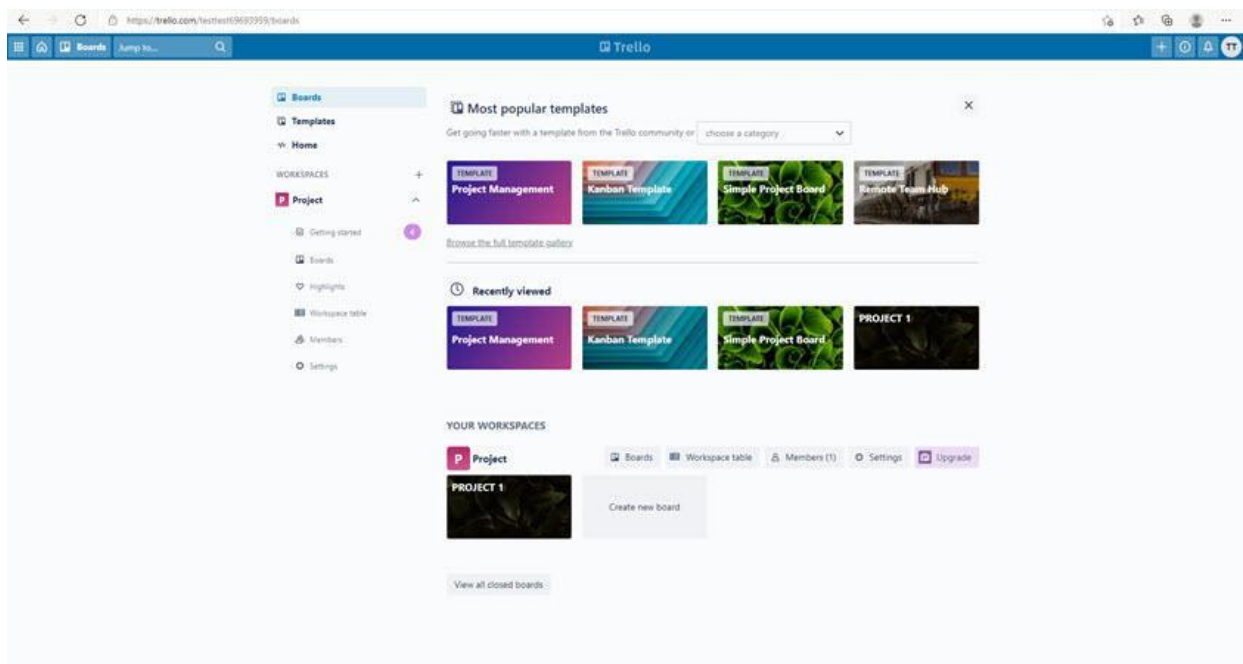


Рис. 1.2 Шаблони для створення проекту

або

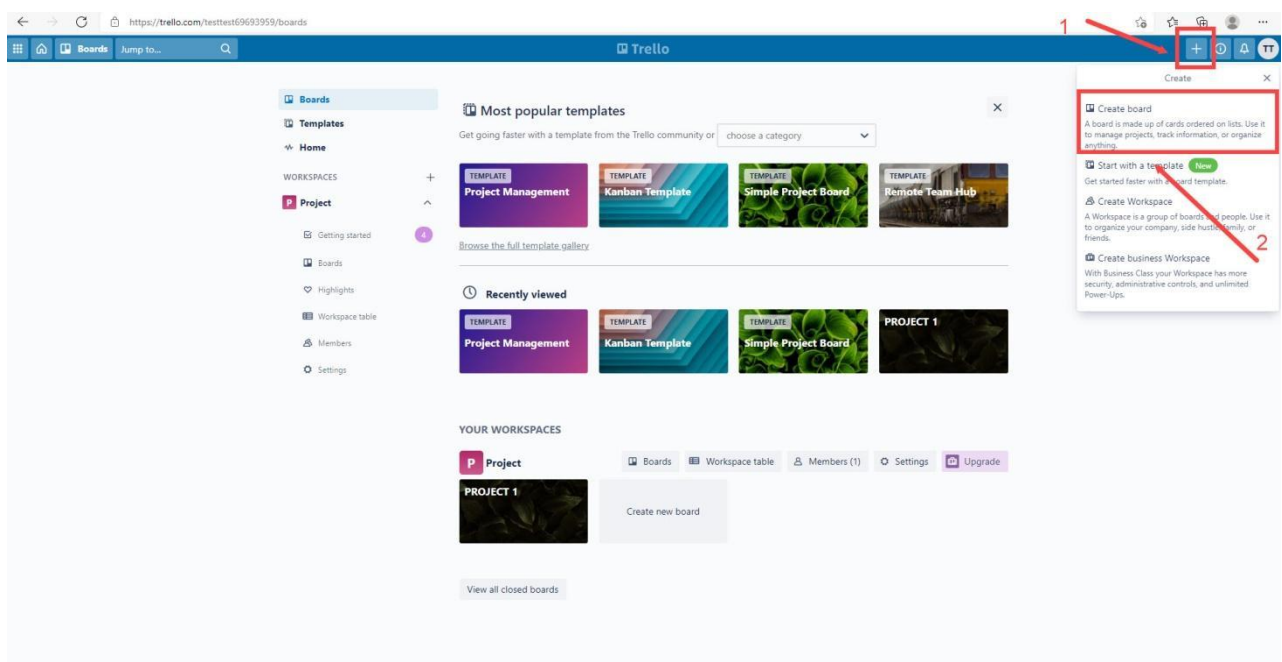


Рис. 1.3 Покрокова інструкція створення проекту

Однією з переваг Kanban-дошки в Trello є можливість налаштування прав доступу. При створенні дошки за замовчуванням статус доступу буде приватний. Але можна також обрати видимість дошки – «Workspace». В такому випадку всі учасники робочого простору матимуть можливість бачити та редагувати дану дошку.

Якщо ж, наприклад, потрібно обмежити доступ до дошки лише кільком користувачам, тоді потрібно використовувати принципово різні дошки.

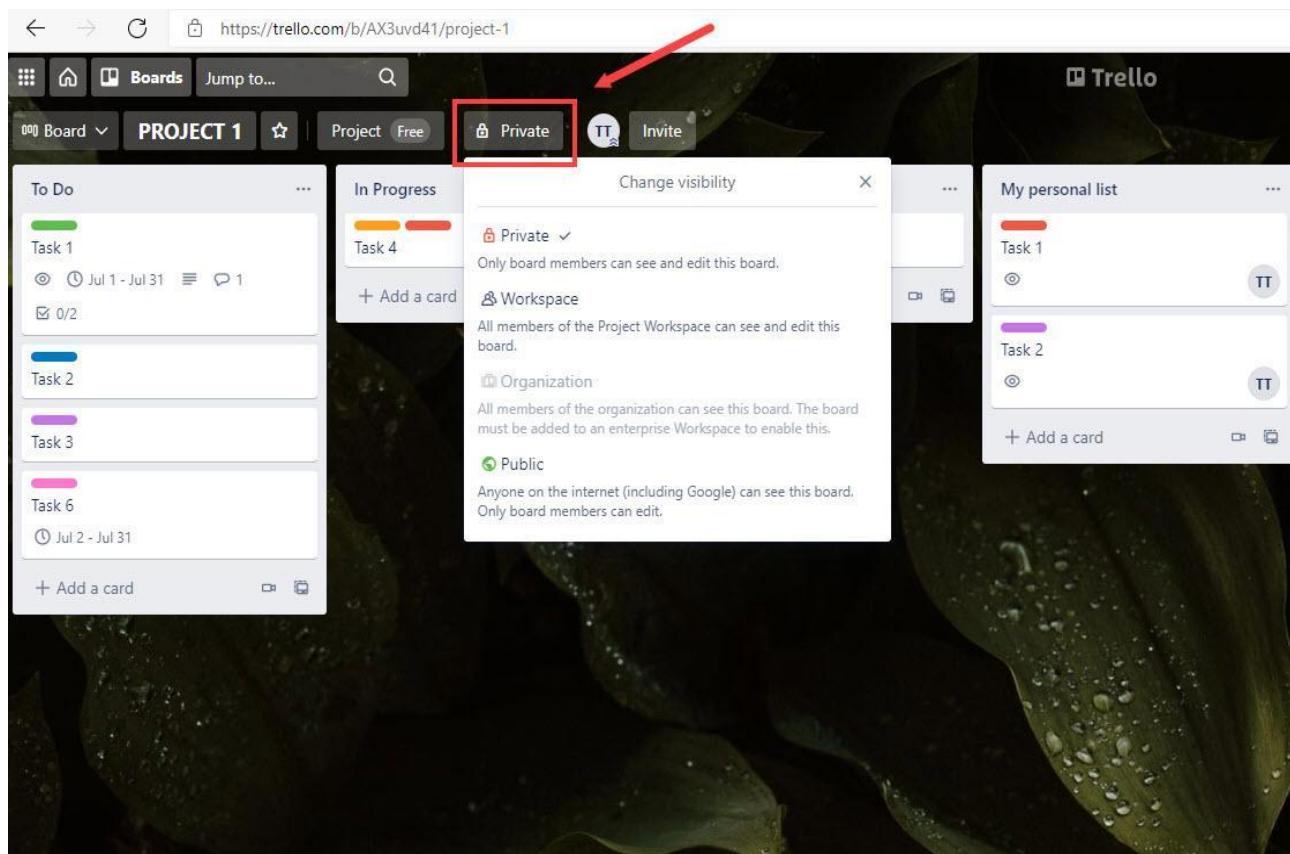


Рис. 1.4 Налаштування обмеження доступу до проєкту

В Trello в межах конкретної дошки створюються так звані списки (List). В залежності від специфіки проєкту дані списки можуть бути представлені у вигляді імпровізованих колонок в залежності від потреб. Класичним варіантом використання у відповідності до методології Kanban є списки To Do, In Progress, Done. Саме так зазвичай візуалізується план виконуваних робіт командою в системі Trello. Створити список можна за допомогою вбудованої опції «Add another list». В кожному списку відображається перелік карток (card) з назвами, під якими маються на увазі задачі, що поставлені до виконання. Називатися вони можуть як завгодно, в залежності від поставлених задач і цілей перед командою. У користувача також може бути свій персональний список, в якому він сам або його менеджер по проєкту створює картки із задачами. З прикладом дошки зі списками можна ознайомитися на наступному зображенні.

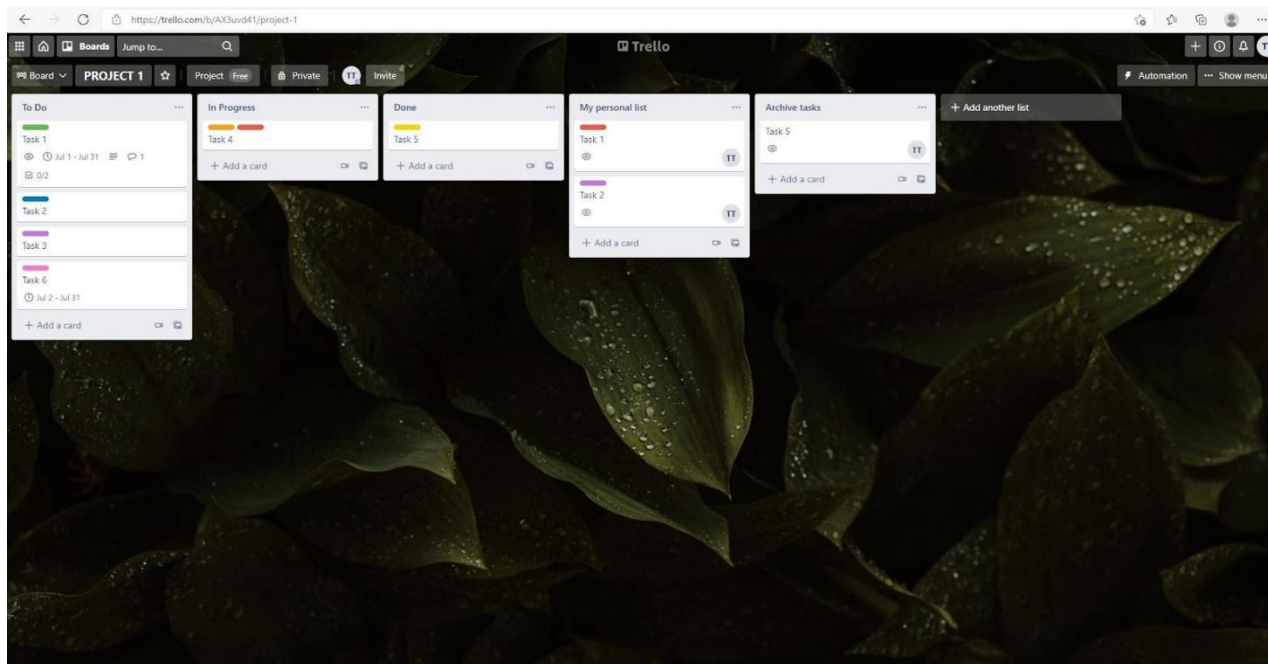


Рис. 1.5 Приклад дошки проекту

Дані картки переміщуються користувачами між списками по мірі готовності виконаних робіт. Здійснюється це шляхом звичайного перетягування картки за допомогою миші або за допомогою опції «Перемістити» в самій картці.

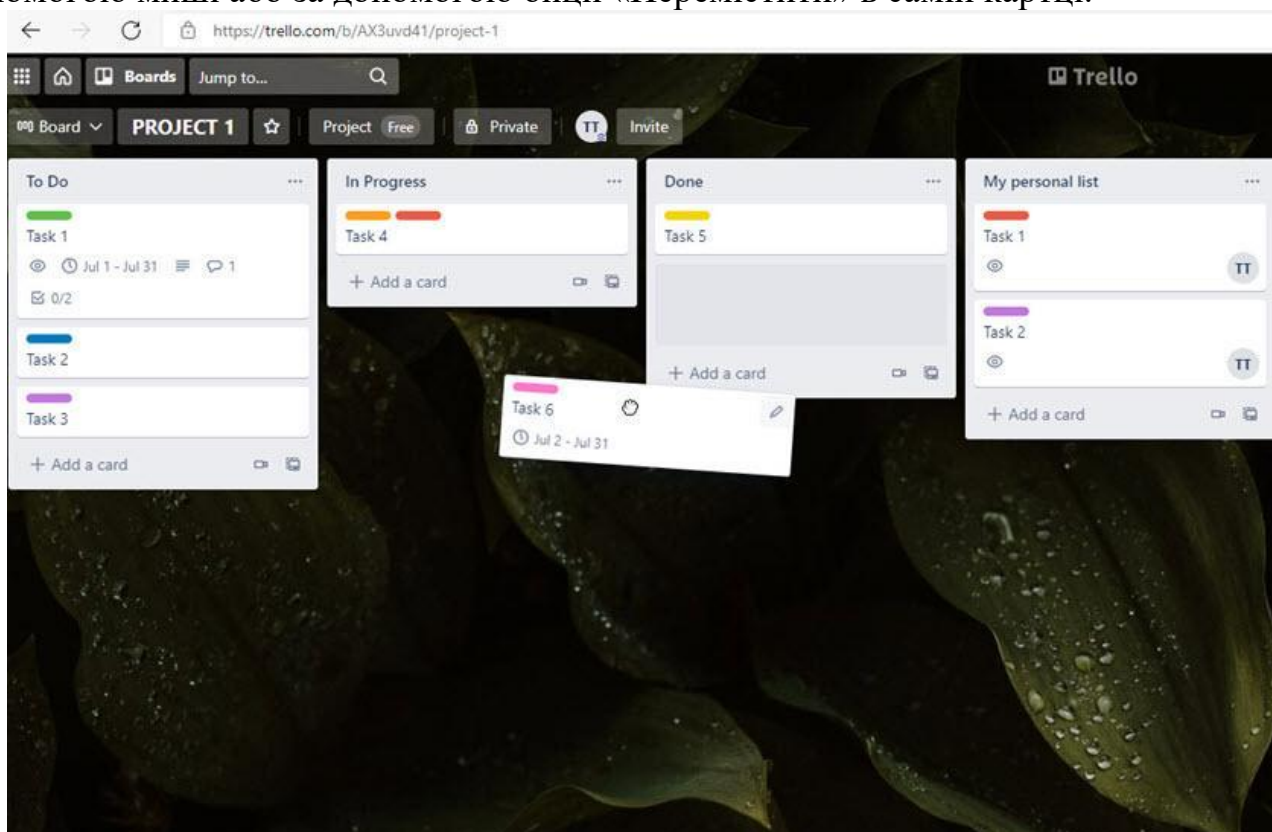


Рис. 1.6 Переміщення картки завдання

Розглянемо більш детально архітектуру карток в Trello. Кожна картка має ряд атрибутів для інформативності під час розподілу та виконання задач між учасниками процесу. До основних атрибутів картки відносяться: назва картки, назва колонки в якій знаходиться картка, учасники, останні оновлення картки, опис та дії (коментарі).

В коментарях можна робити відмітку користувачів, для яких призначається даний коментар, наприклад керівник проєкту/інший співробітник. Робиться це через символ @ – далі логін користувача. Якщо потрібно проінформувати про написання коментаря всіх учасників картки потрібно написати @card. Повідомлення надходять в окреме меню в Trello, а також дублюються на пошту.

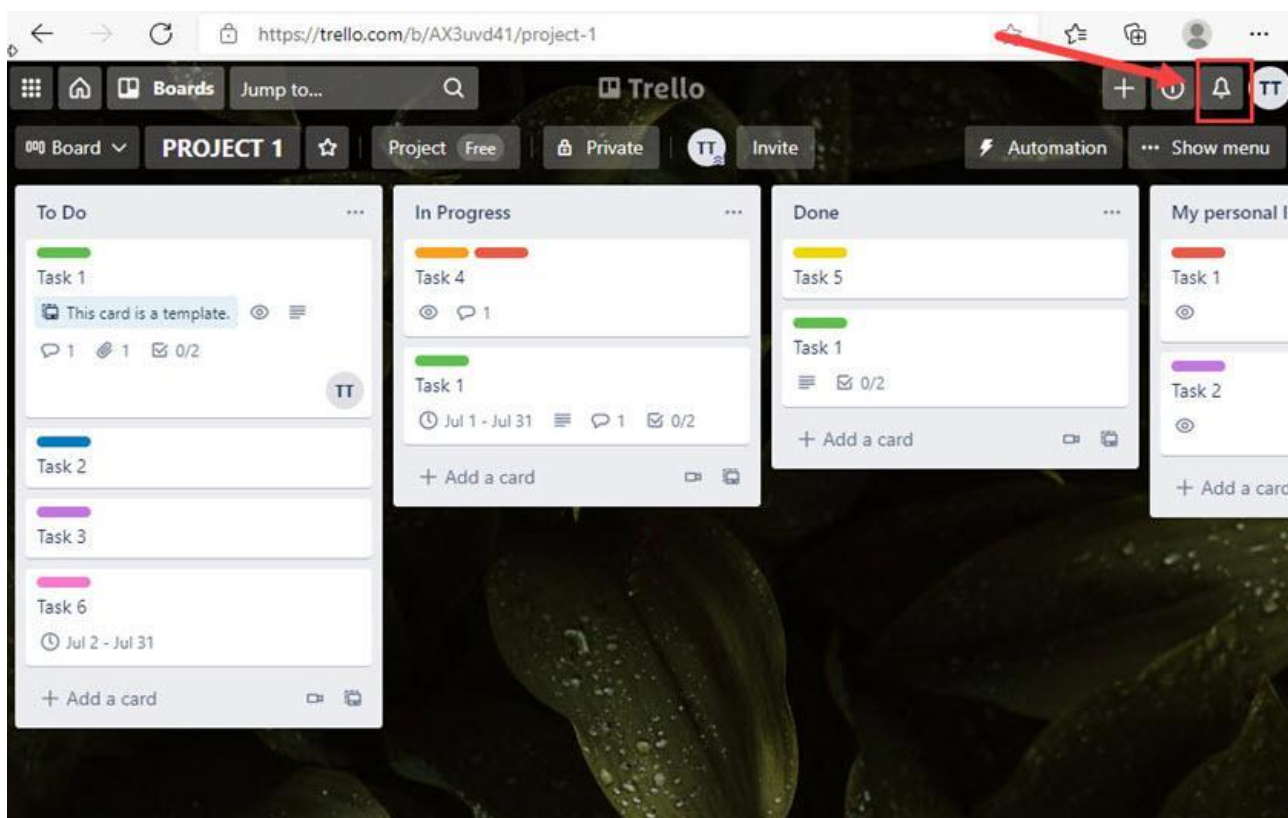


Рис. 1.7 Повідомлення в проєкті

Також є перелік дій, які можна здійснювати над картками, які розбиті на такі категорії:

1. Add to card.
2. Power-ups.
3. Automation.
4. Actions.

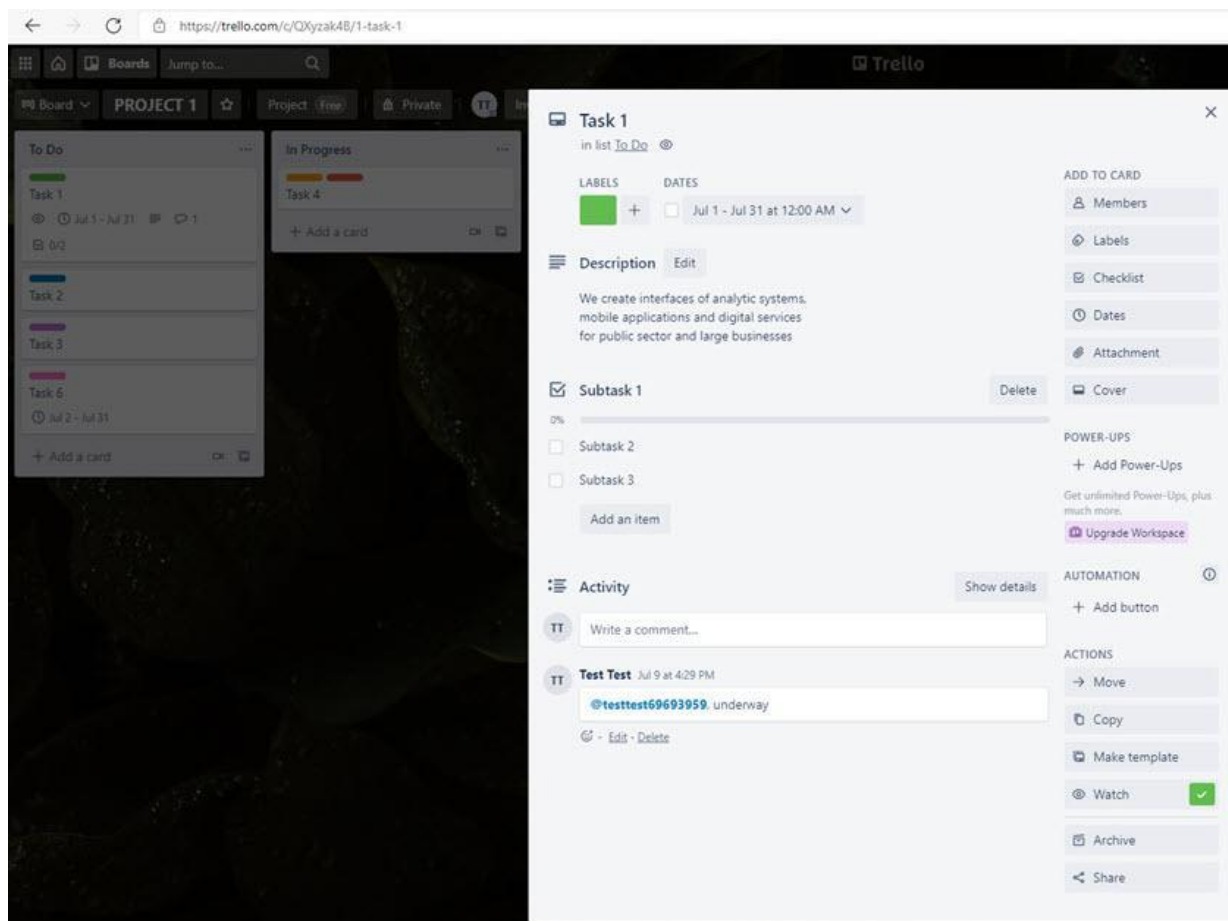


Рис. 1.8 Налаштування картки завдання

До категорії «Add to card» відносяться такі функції:

1. Кнопка «Приєднатися» – потрібна, коли користувач хоче приєднатися та взяти в роботу дану задачу з картки.
2. Кнопка «Учасники» (Members) – дозволяє переглянути всіх учасників картки з можливістю пошуку через пошуковий рядок.
3. Кнопка «Мітки» (Labels) – використовується для класифікації карток в межах одного списку, а також мітками можна задавати певний пріоритет задачі, вказавши назву в мітці та відповідний колір.
4. Кнопка «Чекліст» (Checklist) – дозволяє додати список підзадач до виконання у вигляді чекліста. Такий спосіб є доволі зручним, якщо існує одна задача, яка містить в собі велику кількість підзадач.
5. Кнопка «Дати» (Dates) – дає можливість встановити терміни виконання задачі на картці.

6. Кнопка «Вкладення» (Attachment) – використовується у тому випадку, коли потрібно прикріпити файл або посилання на нього, наприклад інструкцію або файл з виконаним завданням.

7. Кнопка «Обкладинка» (Cover) – призначена для зміни дизайну картки як у вигляді однотонної заливки кольором, так і конкретного зображення. Взагалі налаштування дизайну в Trello є дуже зручними та доброзичливими.

До категорії «Power-ups» відносяться функція кнопка «Додавання покращень» (Add Power-Ups) – дана опція дає змогу додати розширення до картки з каталогу, наприклад інтеграція з Google Drive, Gmail, Slack та ін.

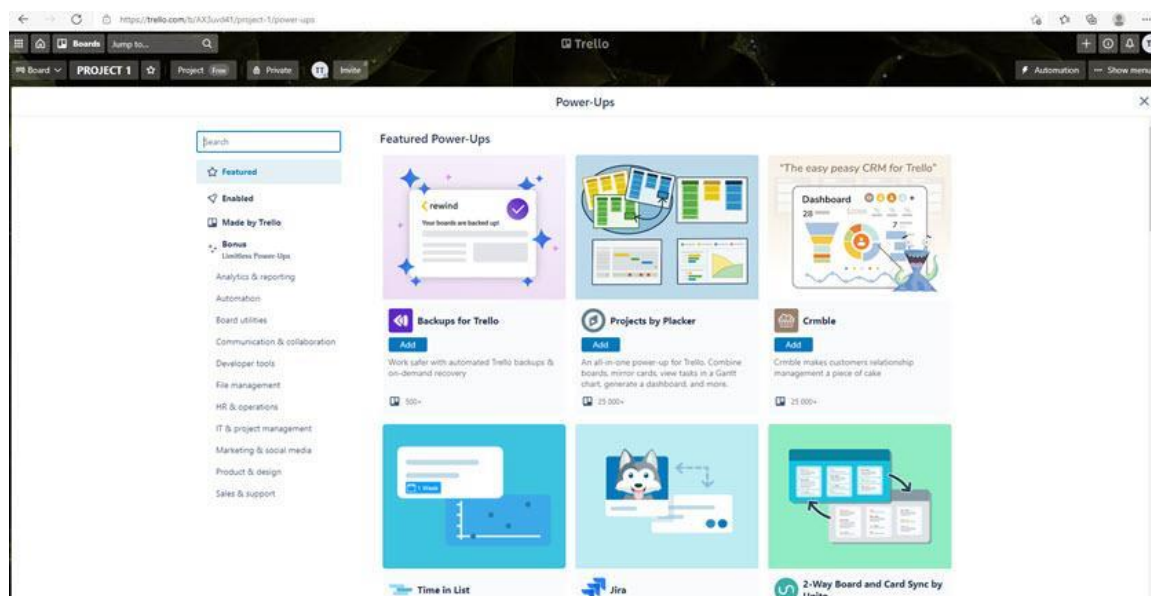


Рис. 1.9 Налаштування «Power-ups»

Остання категорія з діями в картці Trello – категорія «Actions». До неї відносяться:

1. Кнопка «Переміщення» (Move) – дозволяє перемістити картку в інший список.
2. Кнопка «Копіювання» – за її допомогою можна копіювати і створювати подібні картки. Це значно полегшує процес створення карток, особливо коли картки та їх наповнення незначно відрізняються одна від одної.
3. Кнопка «Створити шаблон» – дає можливість зробити картку своєрідним шаблоном-заготівкою для подальшого створення карток в інших списках, але не копіює з батьківської картки такі атрибути як коментарі та терміни роботи над картою.
4. Кнопка «Підписатися» – після натискання на неї, користувач буде отримувати повідомлення про всі зміни у даній картці.

5. Кнопка «Архівація» – необхідна для того, щоб у списках не було перевантаження картками, які вже не використовуються. Для того щоб переглянути заархівовані картки потрібно пройти наступні кроки справа натиснути «Меню» -> Більше -> Архів.

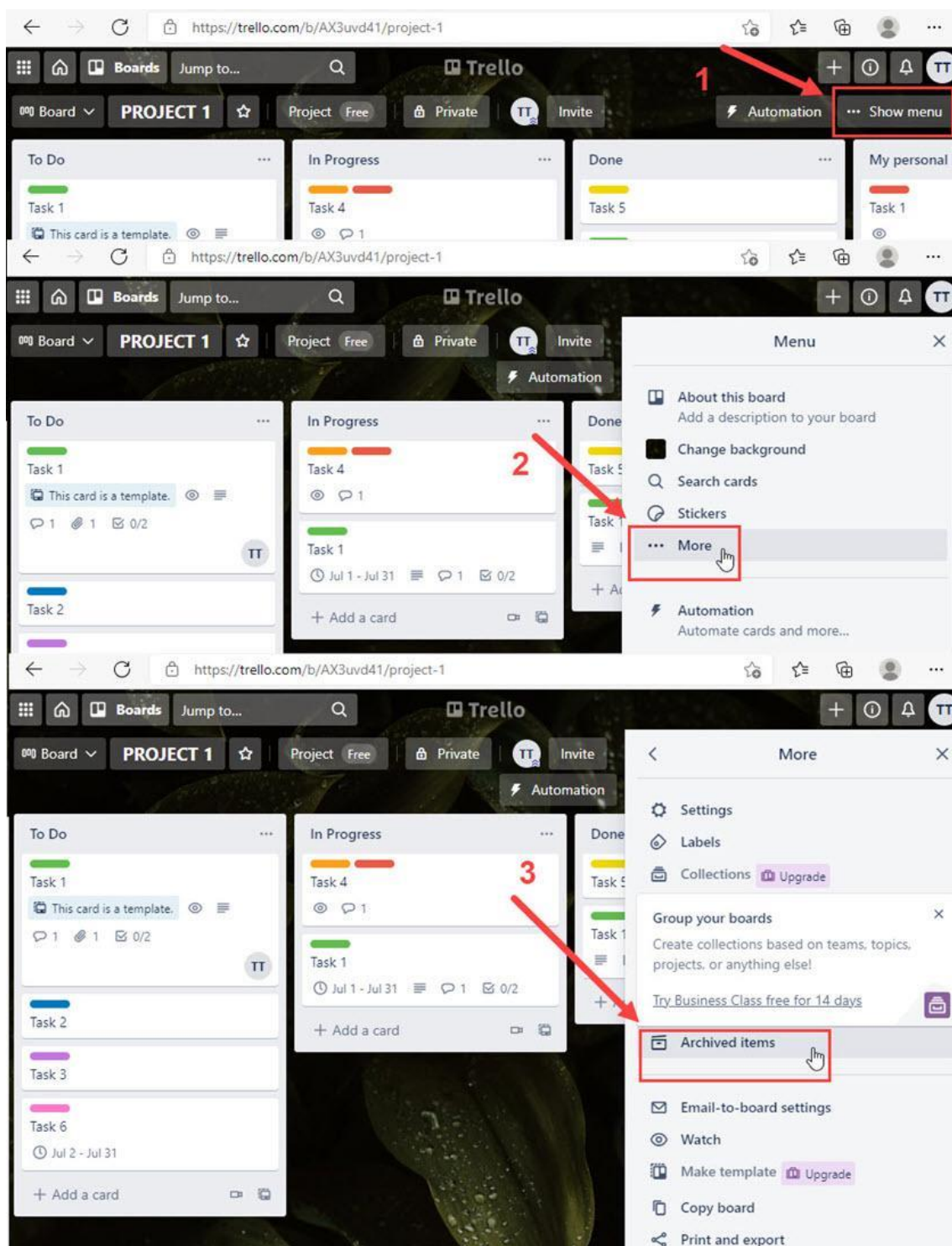


Рис. 1.10 Інструкція архівування картки завдання

Якщо потрібно повернути картку з Архіву в список, потрібно відкрити відповідну картку в Архіві та натиснути кнопку «Повернути».

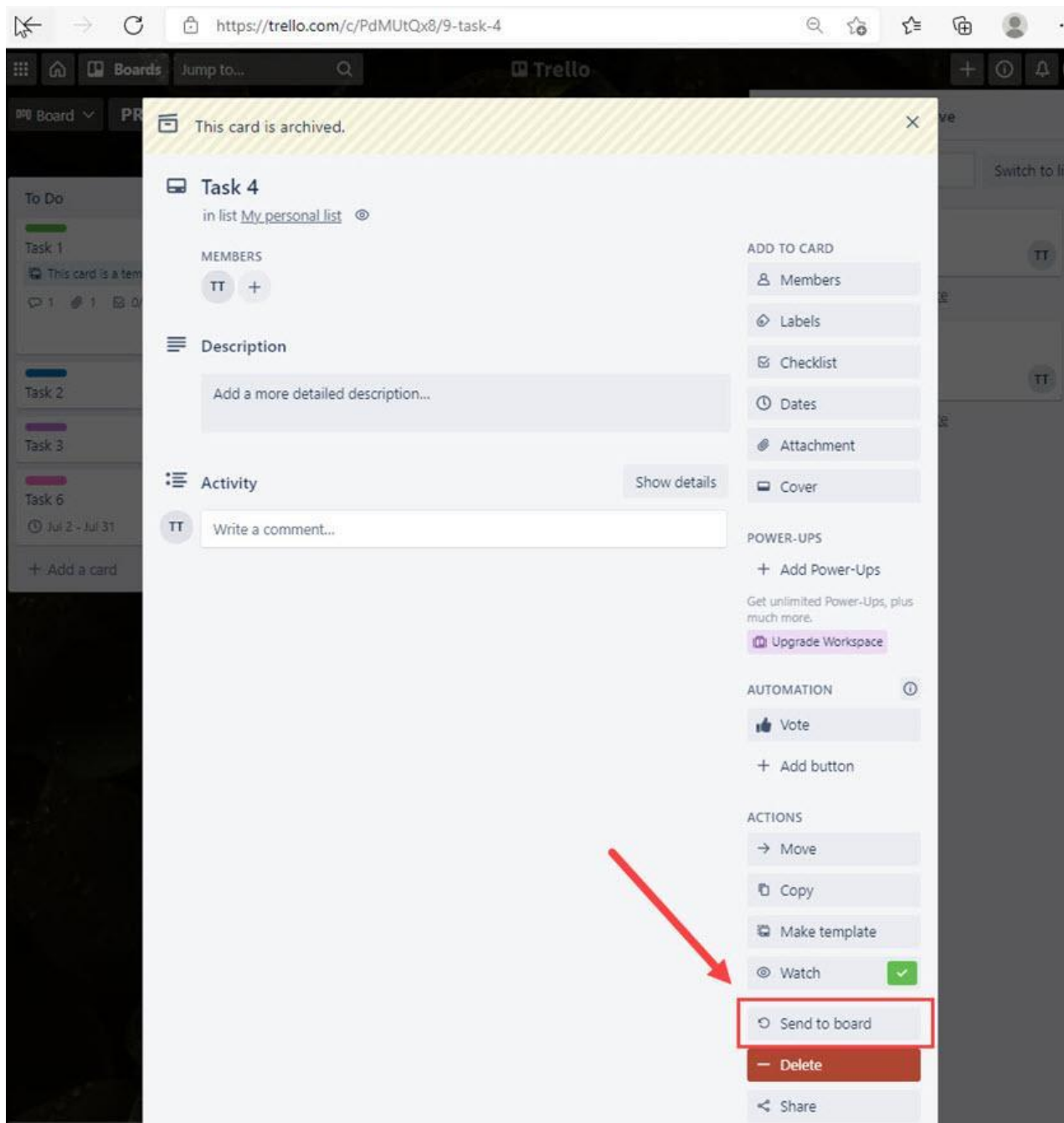


Рис. 1.11 Кнопка повернення картки до проєкту

Кнопка «Поділитися» – за її допомогою формується посилання на картку, яким можна поділитися в соціальній мережі.

Для зручної навігації по картках та швидкого пошуку необхідної картки в Trello реалізована функція пошуку за допомогою фільтрів. Картки можна фільтрувати за мітками, термінами виконання, учасниками.

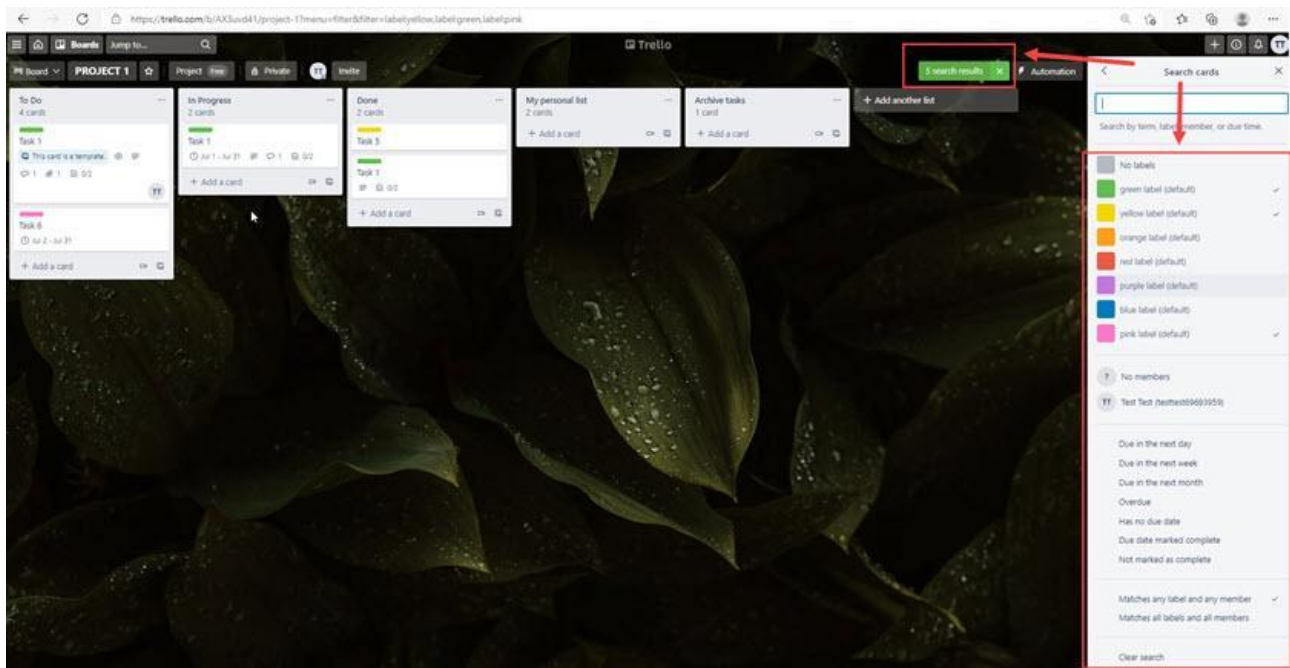


Рис. 1.12 Зразок фільтрування карток в проєкті

1.3 Завдання

1. Необхідно зареєструватися на сайті Trello, для цього потрібно перейти за посиланням: <https://trello.com/uk>
2. Далі потрібно створити свою «робочу область».
3. Створити дошку із тематичною назвою свого проєкту.
4. Зробити необхідні базові налаштування вигляду дошки.
5. Створити не менше 5 списків (колонок, стадій) виконання задач.
6. Створити не менше 30 задач для заповнення дошки і відображення задач за прогресом.
7. Додати не менше ніж 10 описів до обраних за власним бажанням задач.
8. Задати час початку та закінчення задачі не менше ніж до 10-ти задач, а також по 2-3 коментарі до кожної задачі.
9. Налаштувати різні права доступу для учасників (адмін, учасник, гість...)
10. Додати кілька інших користувачів до дошки та заасенити (делегувати) їм задачі.
11. Використати пункти «Фільтри», задати різні умови фільтрації до 5-ти варіантів фільтрів.
12. Зробити звіт про виконану роботу, оформити згідно стандарту та зробити висновок.

1.4 Контрольні запитання

1. Що таке таск-менеджер і яку роль він відіграє в управлінні ІТ-проєктами?
2. У чому полягає принцип роботи платформи Trello?
3. Які особливості методології Kanban реалізовані у Trello?
4. Які основні структурні елементи має інтерфейс Trello?
5. Що таке дошка в Trello і як вона використовується для організації проєктів?
6. Яке призначення мають списки (List) та картки (Card) у контексті таск-менеджменту?
7. Як відбувається взаємодія учасників команди в межах спільної дошки Trello?
8. Які типи атрибутів доступні для картки у Trello і як вони впливають на ефективність управління задачами?
9. Яким чином у Trello реалізовано контроль термінів виконання задач?
10. Як можна класифікувати задачі за допомогою міток (Labels)?
11. У чому полягає функціональність Power-ups і як їх можна використовувати в Trello?
12. Як відбувається обмеження доступу до проєктів у Trello?
13. Яким чином відбувається фільтрація та пошук інформації в межах таск-менеджера Trello?
14. Які дії можна виконати над картками у Trello в межах категорії «Actions»?
15. Які переваги та обмеження має Trello як інструмент для управління ІТ-проєктами?

ЛАБОРАТОРНА РОБОТА № 2 (4 години)

«АВТОМАТИЗАЦІЯ ПРОЦЕСІВ У TASK-МЕНЕДЖЕРІ TRELLO»

Тема роботи: автоматизація процесів у task-менеджері Trello

Мета роботи: навчитись створювати правила автоматизації процесів у Trello.

2.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно ознайомитися з інструментами автоматизації в системі Trello, вивчити основні принципи створення правил, кнопок і запланованих дій, а також реалізувати приклади автоматизації, що охоплюють різні аспекти управління задачами. Результатом роботи має стати практична реалізація автоматизованих сценаріїв, які забезпечують підвищення ефективності командної взаємодії в Trello.

Постановка завдання: описати та реалізувати автоматизацію процесів в середовищі Trello із застосуванням правил, кнопок, календарних команд та інтеграцій.

2.2 Теоретичні відомості

Додаткова інформація, та повна документація з роботи у Trello: <https://trello.com/guide/trello-101> (мова пояснення у guide, буде залежати від вашої авторизації у акаунті Trello, для пояснення роботи українською мовою, зайдіть у свій обліковий запис на Trello)

Інструмент автоматизації Trello Butler Automation налаштовується за лічені хвилини і може заощадити не одну годину роботи вашої команди. Дізнайтеся, як автоматизувати виконання рутинних завдань, що знижують продуктивність.

У цій лабораторній роботі розглядаються наступні поняття:

- Основна інформація про автоматизацію.
- Початок роботи з автоматизацією.
- Автоматизація на основі правил.
- Кнопки карток і дощок.
- Команди календаря та дати завершення.
- Автоматизація для електронної пошти й інтеграції.
- Основна інформація про автоматизацію.

Основна інформація про автоматизацію

Нерідко в моменти найвищої продуктивності наша робота сповільнюється виснажливими адміністративними завданнями, які необхідно виконувати, щоб впорядкувати інформацію досягнути прогресу. На щастя, у Trello вбудована система Automation, що надає змогу автоматизувати будь-яку дію в Trello.

За допомогою автоматизації можна створювати:

- Правила, що у відповідь на певну дію миттєво запускають набір інших дій.
- Кнопки, що після єдиного натискання запускають дію на картці або всій дошці.
- Пов'язані з датами завершення команди, що запускаються у зв'язку з наближенням або закінченням відповідних строків.
- Пов'язані з календарем команди, що запускаються в певні періоди.

«Automation» у Trello використовує просту мову, яка не потребує написання коду – автоматизацію дій можна налаштувати за лічені секунди. Пропонуємо ознайомитися з можливостями «Automation». Початок роботи з автоматизацією виглядає наступним чином.



Рис. 2.1 Кнопка для створення автоматизації

Натисніть «Automation» у меню дошки, щоб відкрити командне вікно, де можна налаштувати автоматизацію для дошки й керувати нею. Ось що можна знайти в «Automation».

- Поради щодо автоматизації. Автоматично виявляйте повторювані дії, які виконуються на дошці, і перетворюйте їх на автоматичні завдання, що можна додати на дошку в один клік. Це чудовий спосіб швидко освоїти автоматизацію.
- Правила, кнопки карток, кнопки дощок, календар і дата завершення. Потрібно натиснути на будь-який із цих елементів, щоб створити, відредагувати, увімкнути або видалити автоматичні завдання на дошці.
- Довідкові відомості. Автоматизація надає величезні можливості, тому пропонується докладна документація з корисними порадами.
- Підключені додатки. Автоматизація дає змогу не лише створювати автоматичні завдання в Trello, а й автоматизувати деякі дії в інших додатках, якими користується командою, як-от у Slack і Jira.
- Акаунт. Дозволяє переглядати кількість виконаних команд і оцінювати використання квоти доступу.

Правила набувають чинності, коли виконується конкретна умова, що призводить до ряду дій. Вони сприяють оптимізації робочих процесів, тому що за умови запуску правила тим чи іншим користувачем кожен учасник буде виконувати встановлені дії. Це означає, що під час роботи жодна важлива деталь не залишиться поза увагою.

Create a Rule

Save Cancel

Trigger

when a card is added to list "Done" by me

Actions

mark the due date as complete

check all the items in all the checklists on the card

remove all the members from the card

Рис. 2.2 Вікно створення правила автоматизації

Щоб створити правила можна використати наступні кроки:

- Натисніть «Створити правило» в меню правил.
- Додайте тригер. Це призведе до виконання попередньо визначених дій.
- Додайте дії. Можна додати стільки дій, скільки потрібно.
- Збережіть нове правило.

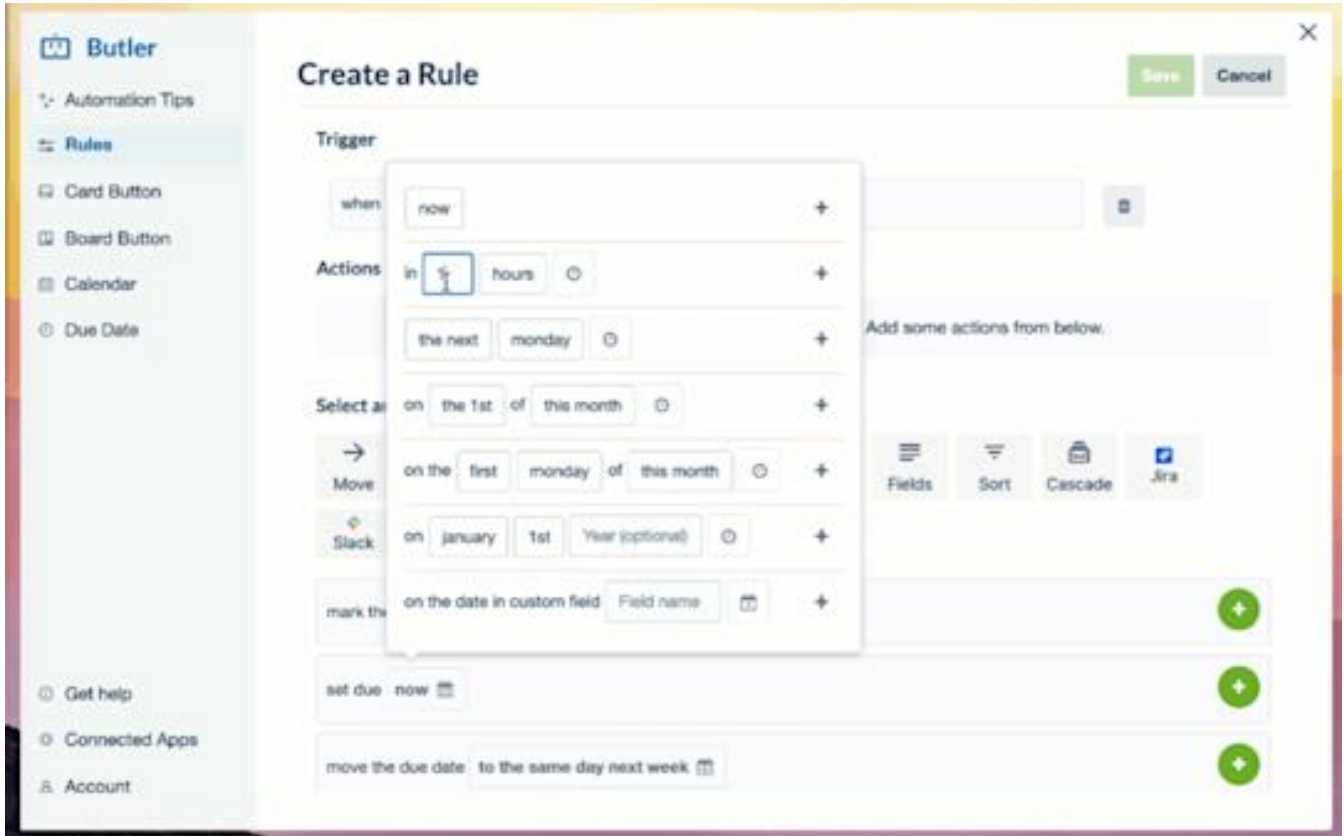


Рис. 2.3 Приклад заповненого правила автоматизації

Деякі приклади корисних правил:

- Відображення термінових завдань
 - Тригер: картка має позначку «Терміново».
 - Дії: перемістіть картку на початок списку, додайте запитання «Чи може хтось взяти це завдання?» у форматі згадування дошки (@) і дату завершення – три дні із цього моменту.
- Керування вхідними запитами
 - Тригер: нову картку додано до списку «Вхідні»
 - Дії: призначте учасника для картки, додайте контрольний список із необхідними підзадачами й дату завершення.
- Завершення завдання

- Тригер: усі пункти контрольного списку виконано
- Дії: позначте дату завершення, перемістіть картку до списку «Готово» і сповістіть про це менеджера у форматі згадування (@)!

Щоб додати правило безпосередньо зі списку, потрібно натиснути значок меню в правому верхньому куті будь-якого списку. Також можна додати декілька корисних шаблонів для сортування списку правил, щоб було легше розпочати роботу.

За певних обставин може стати в нагоді автоматичне виконання завдань. Можлива допомога корисних кнопок карток і дощок. Одним натисканням кнопки можна активувати ряд дій за запитом на відповідному рівні.

Рис. 2.4 Вікно створення кнопки для картки

Ось як додати кнопки до карток і дощок:

- Натисніть «Створити кнопку» в меню кнопок картки або дошки.
- Дайте кнопці назву й виберіть значок для полегшення візуального розпізнання.
- Додайте дії, які мають виконуватися в результаті натискання кнопки.

Приклади корисних кнопок:

- Створіть кнопку картки для миттєвого виконання наступних дій:
 - перенесення картки до іншого списку;
 - додавання учасників, відповідальних за наступний етап;

- додавання дати завершення;
- створення контрольного списку підзадач для відстеження всіх процесів.
- Створіть кнопку дошки для аналізу нескінченних вимог до продукту шляхом:
 - сортування карток за часовими відрізками, датами завершення та кількістю голосів у користувацьких полях тощо, щоб розставити пріоритети для наступного спринту.

Додайте кнопки картки безпосередньо зі зворотного боку картки, натиснувши «Додати кнопку» з розділу автоматизації на картці. Також додаємо декілька корисних шаблонів для кнопок картки, щоб було легше розпочати роботу.

В «Automation» можна встановити зручний розклад, використовуючи заплановані команди на основі календарних дат чи дати завершення для картки.



Рис. 2.5 Створення автоматизації

Команди календаря чудово підходять для автоматизації повторюваних завдань і регулярного обслуговування дошки, як-от переміщення та архівування карток, сортування списків і додавання нових списків до дошки. Можна зробити так, щоб ці команди запускалися в певний час щодня, щотижня, щомісяця або щороку.

Ось способи налаштування команди календаря для виконання планового технічного обслуговування дошки раз на тиждень:

- Щопонеділка о 9:00 ранку...
 - додайте до архіву всі картки в списку «Готово»;
 - перемістіть картки зі списку завдань для наступного спринту до списку «Потрібно зробити»;

- відсортуйте всі картки зі списку поточних завдань за датою завершення.

У командах дати завершення дата завершення картки використовується як тригер. За допомогою цих команд можна запрограмувати ряд дій, коли дата завершення наближається, настає або коли після дати завершення картки минає певний час.



Рис. 2.6 Налаштування дати виконання команди

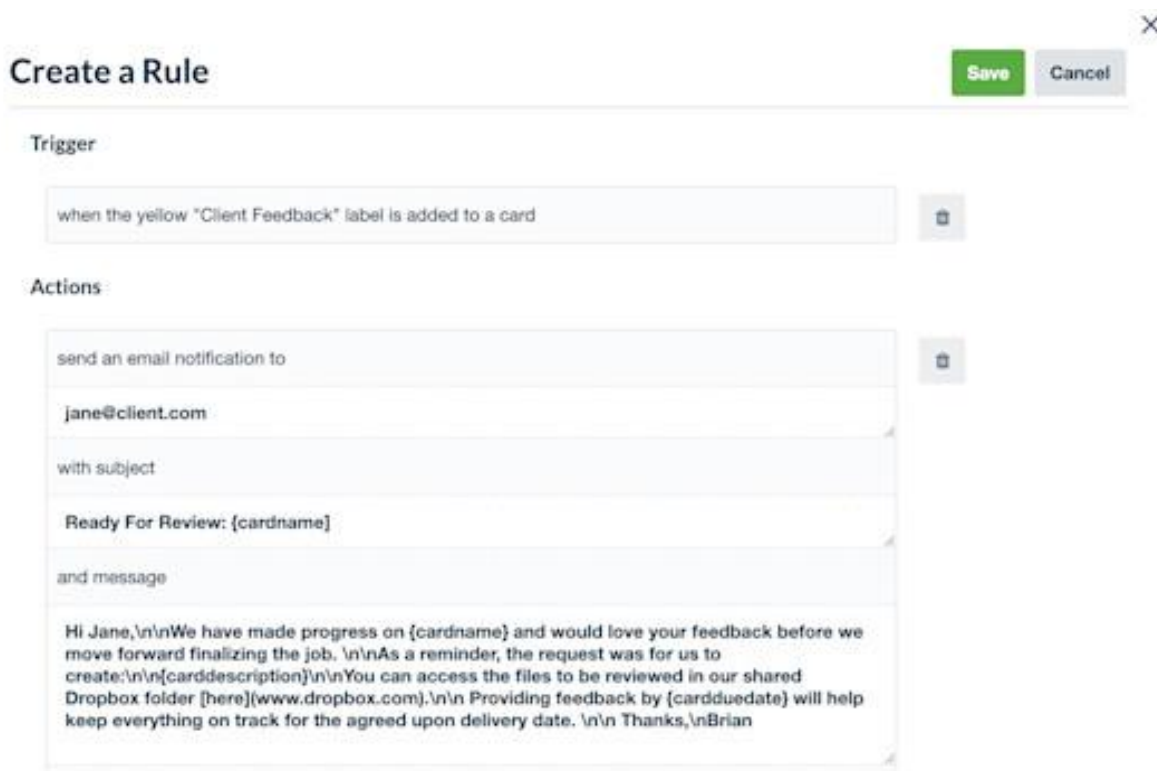
Налаштування команд дати завершення – чудовий спосіб заручитися тим, що жодна деталь не залишиться поза увагою. Невиконану прострочену картку можна позначити так:

- Коли мине термін завершення картки:
 - перемістити її на початок списку;
 - додати до неї червону мітку;
 - опублікувати коментар «@card: який статус?».

Розширюйте можливості дощок через автоматизацію з використанням електронної пошти й інтеграцій зі Slack і Jira. Завдяки цьому можна автоматично ділитися інформацією та оновленнями зі сторонніми учасниками, клієнтами й колегами, які працюють у додатках, відмінних від тих, що використовує команда.

Наприклад, спробуйте створити щотижневі звіти про стан карток на дошці, які надсилаються за допомогою запланованих команд на електронну пошту керівників команд. Якщо ж працюєте із зовнішніми учасниками, установіть правило, яке надсилатиме клієнтам електронного листа, коли потрібно отримати відгук. Тригер для

надсилання листа спрацьовуватиме щоразу, коли картка перетягується до списку «Потрібен відгук».



Create a Rule [Save] [Cancel]

Trigger

when the yellow "Client Feedback" label is added to a card

Actions

send an email notification to

jane@client.com

with subject

Ready For Review: {cardname}

and message

Hi Jane,\n\nWe have made progress on {cardname} and would love your feedback before we move forward finalizing the job. \n\nAs a reminder, the request was for us to create:\n\n{carddescription}\n\nYou can access the files to be reviewed in our shared Dropbox folder [here](www.dropbox.com).\n\nProviding feedback by {cardduedate} will help keep everything on track for the agreed upon delivery date. \n\nThanks,\nBrian

Рис. 2.7 Приклад створення правила автоматизації

Налаштування електронної пошти за допомогою «Automation» не відрізняється від решти параметрів налаштувань.

- Створюючи правило, кнопку картки чи дату завершення, виберіть пункт «Вміст». Створюючи кнопку дошки чи заплановану команду, виберіть у меню дій пункт «Інше».
- Створіть тему й тіло листа для надсилання.
- Спробуйте використати змінні, щоб заповнювати автоматизовані електронні листи зі спеціальним вмістом із ваших карток і дошки.

Інтеграція із Jira стане в пригоді, якщо одна команда відстежує роботу в Trello, а інша – в Jira, адже можна створювати проблеми Jira чи публікувати коментарі до наявних заявок. Завдяки Slack можна тримати всіх у курсі подій, публікуючи коментарі в каналах Slack із важливими оновленнями з Trello.

Create a Card Button Save Cancel

Icon Title Options

☐ ☐ Enabled by default
Close card when action is performed

Actions

move the card to the top of list

Read For Development

Add Another Action

→ + - ⌚ ✓ 👤 💬 📋 ⚙️ 📁 Jira 📢 Slack

Move Add/Remove Dates Checklists Members Content Fields Sort Cascade Jira Slack

create a JIRA task with title {cardtitle}

and description

The following has been approved and ready for development:
{carddescription}
For more information visit: {cardlink}

in project DEV in site teaminspace

post a comment to JIRA issue issue key or link in site Site name or link with message

Comment text

Рис. 2.8 Приклад створення кнопки для картки

Ось як інтегрувати «Automation» у Trello зі Slack і Jira.

- Створюючи автоматизацію, у меню дій натисніть вкладки Jira або Slack.
- Авторизуйте додаток для використання автоматизації в Trello.
- Виберіть дії, які потрібно автоматизувати в зовнішньому додатку.

2.3 Завдання

1. Ознайомитися з принципами початкового налаштування автоматизації в Trello (інструмент Butler).
2. Створити просте правило автоматизації для однієї з дошок (на основі певної події, наприклад: переміщення картки, зміна мітки, встановлення терміну тощо).
3. Налаштувати кнопки карток і дошки для виконання часто повторюваних дій (наприклад, створення чеклісту, зміна статусу, призначення учасника).
4. Додати автоматичну команду для взаємодії з календарем або встановлення терміну завершення завдання.
5. Реалізувати приклад автоматизації з інтеграцією (наприклад, автоматичне надсилання листа або використання Webhook, якщо підтримується).

6. Зробити звіт про виконану роботу, оформити згідно стандарту та зробити висновок.

2.4 Контрольні питання

1. Що таке автоматизація в контексті таск-менеджера Trello та яку роль вона відіграє в управлінні проєктами?

2. Яке призначення інструменту Butler у Trello та які основні компоненти він містить?

3. У чому полягає різниця між правилами, кнопками карток, кнопками дошки та командами календаря в автоматизації Trello?

4. Як створити правило автоматизації, що активується під час переміщення картки між списками?

5. Які дії можна автоматизувати за допомогою кнопок карток? Наведіть приклади.

6. Як реалізується інтеграція Trello з іншими сервісами в рамках автоматизації?

7. Які можливості надає Trello для роботи з датами та календарем в автоматизованих командах?

8. Які переваги автоматизації завдань у Trello з точки зору командної роботи?

9. Які обмеження існують для використання Butler у безкоштовній та платній версіях Trello?

10. Яким чином можна використовувати чеклісти в автоматизованих діях?

11. Як автоматизація може допомогти в дотриманні термінів виконання задач?

12. Які типові помилки допускаються під час створення правил автоматизації, і як їх уникнути?

ЛАБОРАТОРНА РОБОТА № 3 (4 години)

«РОБОТА З СИСТЕМОЮ JIRA»

Тема роботи: робота з системою Jira.

Мета роботи: здобути основні навички у роботі з інтерфейсом системи Jira.

3.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно ознайомитися з основними можливостями системи Jira для управління IT проєктами. Здобувач повинен створити новий проєкт, налаштувати дошку Scrum або Kanban, опрацювати роботу з тікетами (задачами), мітками, компонентами та життєвим циклом задач. Особливу увагу слід приділити створенню Workflow для багів та розумінню статусів тікета на всіх етапах його проходження.

Постановка завдання: ознайомитися з функціональністю Jira та реалізувати управління завданнями в межах створеного проєкту з використанням Scrum або Kanban дошки.

3.2 Теоретичні відомості

Jira – система відстеження помилок (баг-трекер), призначена для організації спілкування з учасниками команди, а також для керування проєктами. Ця система була розроблена компанією Atlassian в 2002 році.

Jira має дуже об'ємний функціонал, в якому не так просто розібратися недосвідченому користувачу. Нижче ми розглянемо найпростіші елементи і функції, які використовуються на проєктах найчастіше.

Jira дозволяє створити власний проєкт, налаштувати його та відстежувати весь процес виконання завдань до самого завершення. За допомогою налаштування адміністратор може визначати доступні дії для різних користувачів.



Рис. 3.1 Ілюстрація системи «Jira»

Для створення проєкту необхідно в лівому кутку натиснути на значок головної сторінки та натиснути Projects (Проекти) - Create project (Створити проєкт).

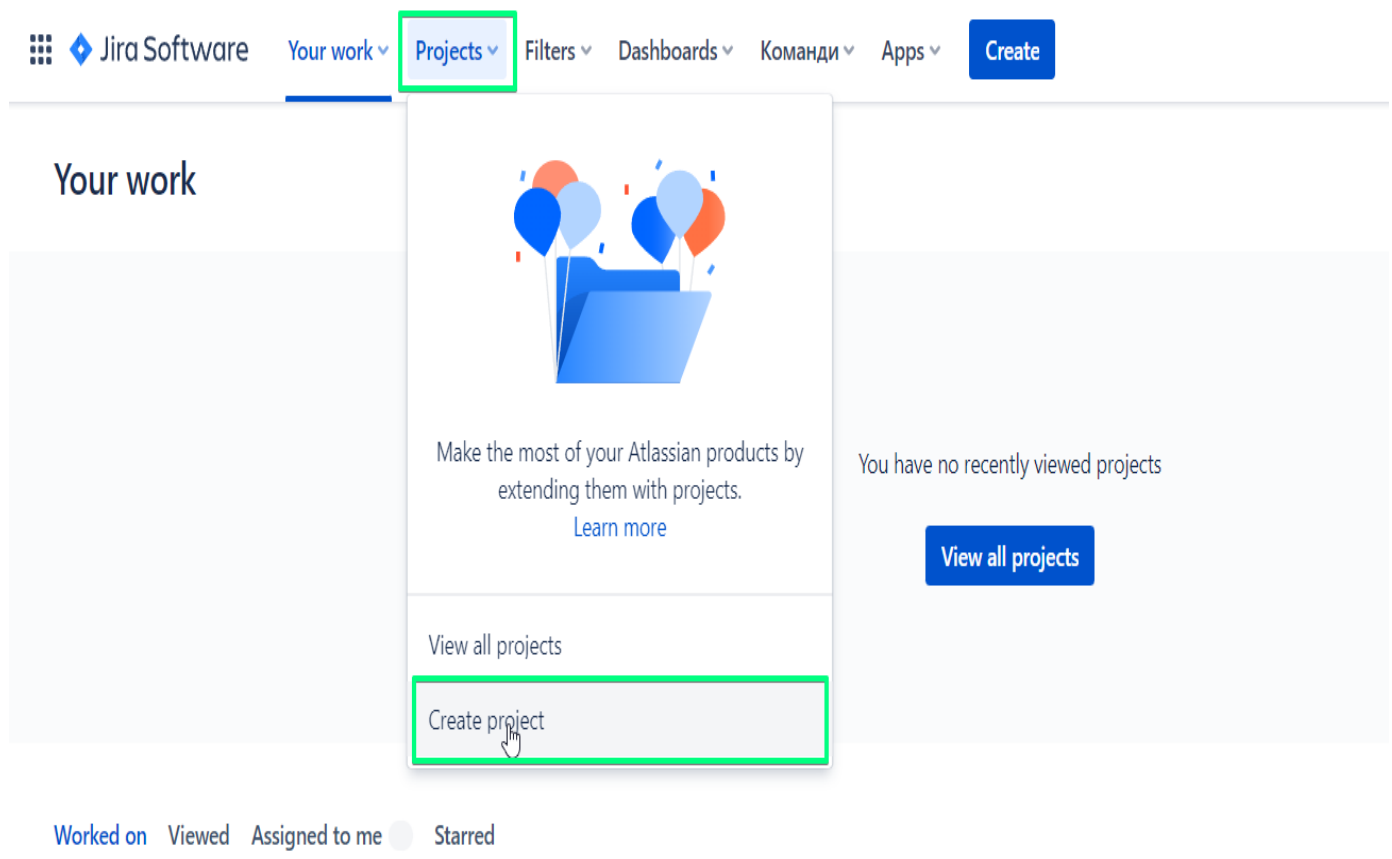


Рис. 3.2 Створення проєкту в Jira

Далі потрібно обрати шаблон проєкту (Project template).

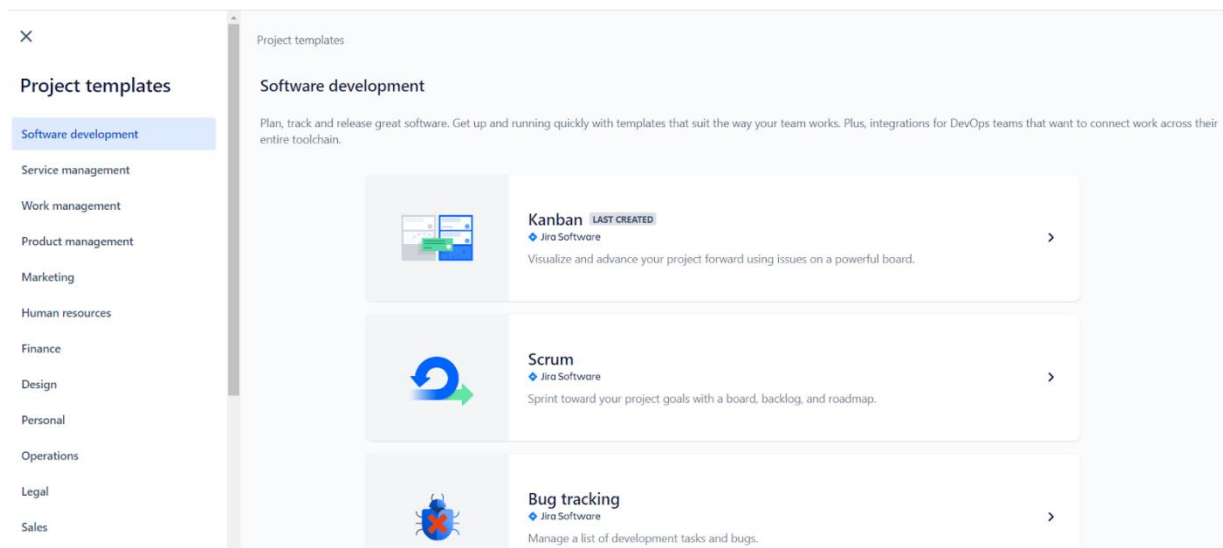


Рис. 3.3 Вибір шаблону для проєкту

В даному випадку обрано Scrum та тип проєкту «Company-managed».

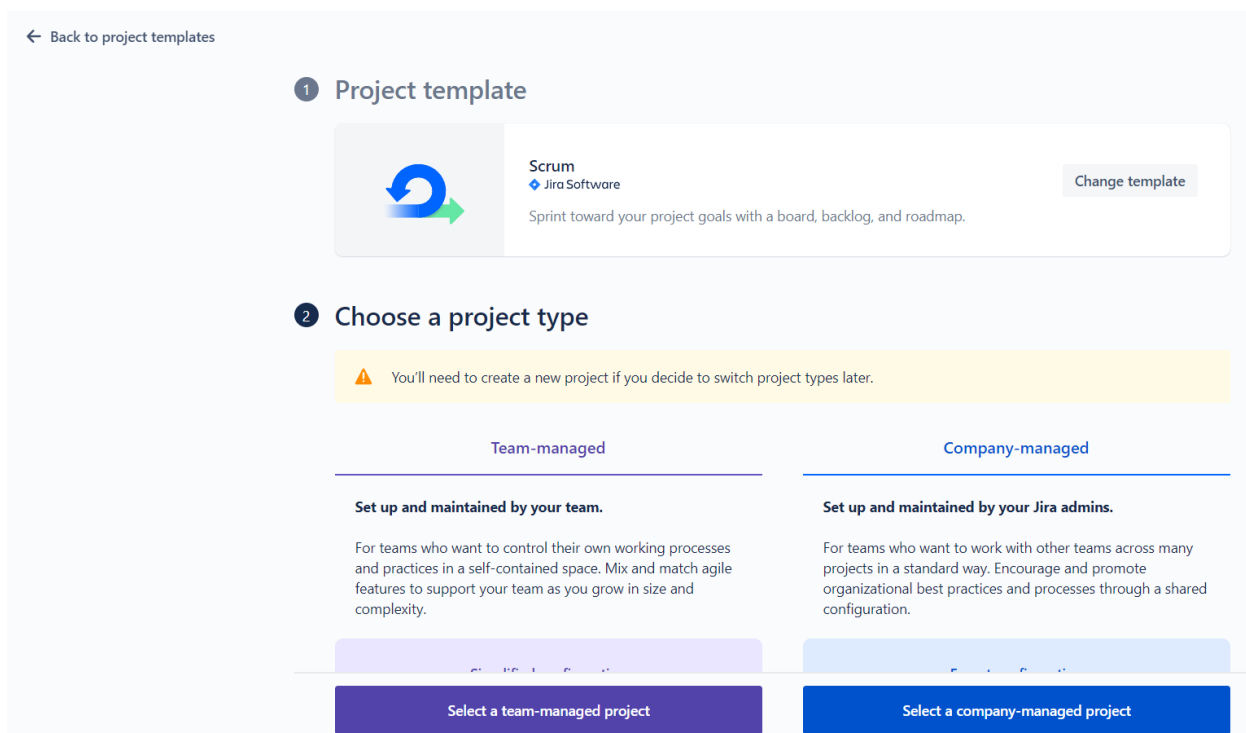


Рис. 3.4 Вибір Scrum-проєкту

Після створення проєкту необхідно створити дошку на якій буде відображатись весь процес роботи з задачами. На дошці можуть бути представлені завдання одного або декількох проєктів. Вони розташовані в стовпцях, які відображають процес роботи команди. За допомогою дошки команда отримує загальне уявлення про всю розпочату роботу, яка наразі виконується, та про завершену роботу.

Для створення Scrum або Kanban дошки необхідно у блоці «Planning» відкрити пошук всіх дошок та натиснути «Create board».

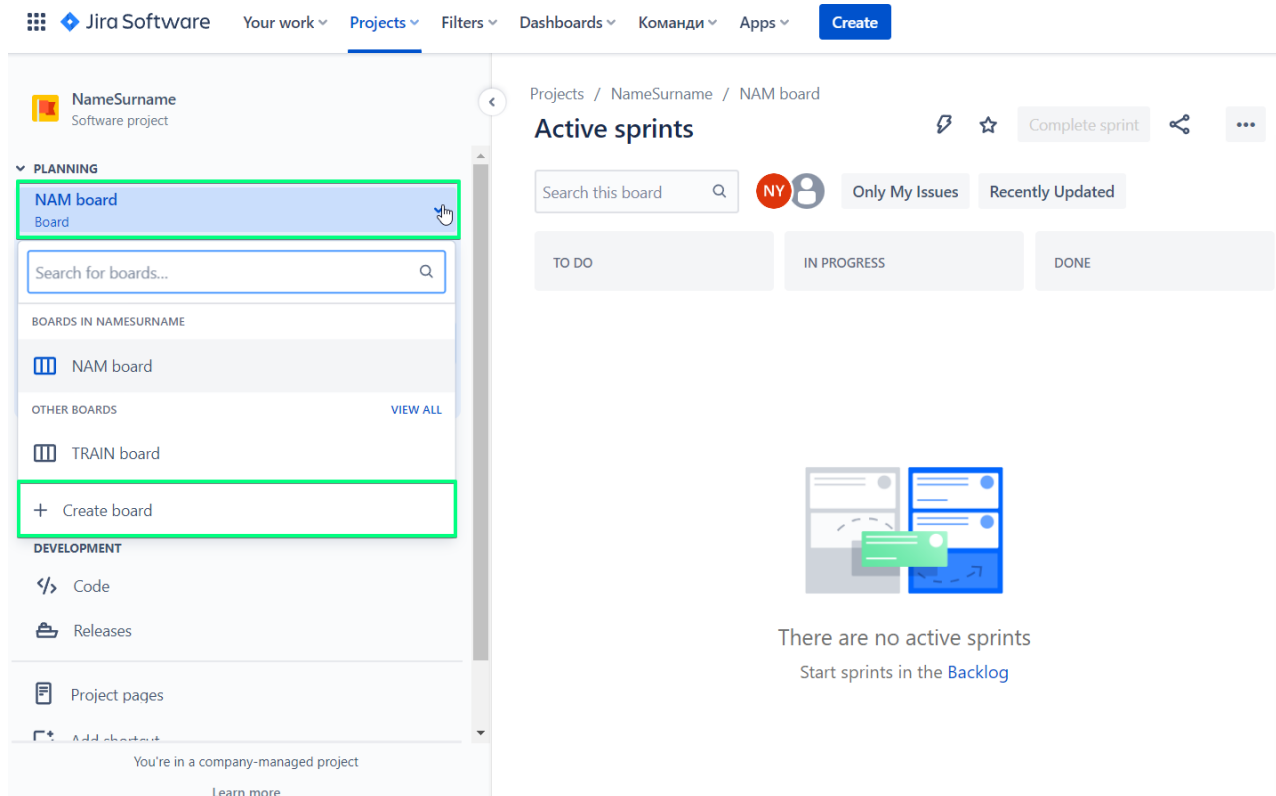


Рис. 3.5 Створення дошки

Обрати тип дошки Kanban / Scrum.

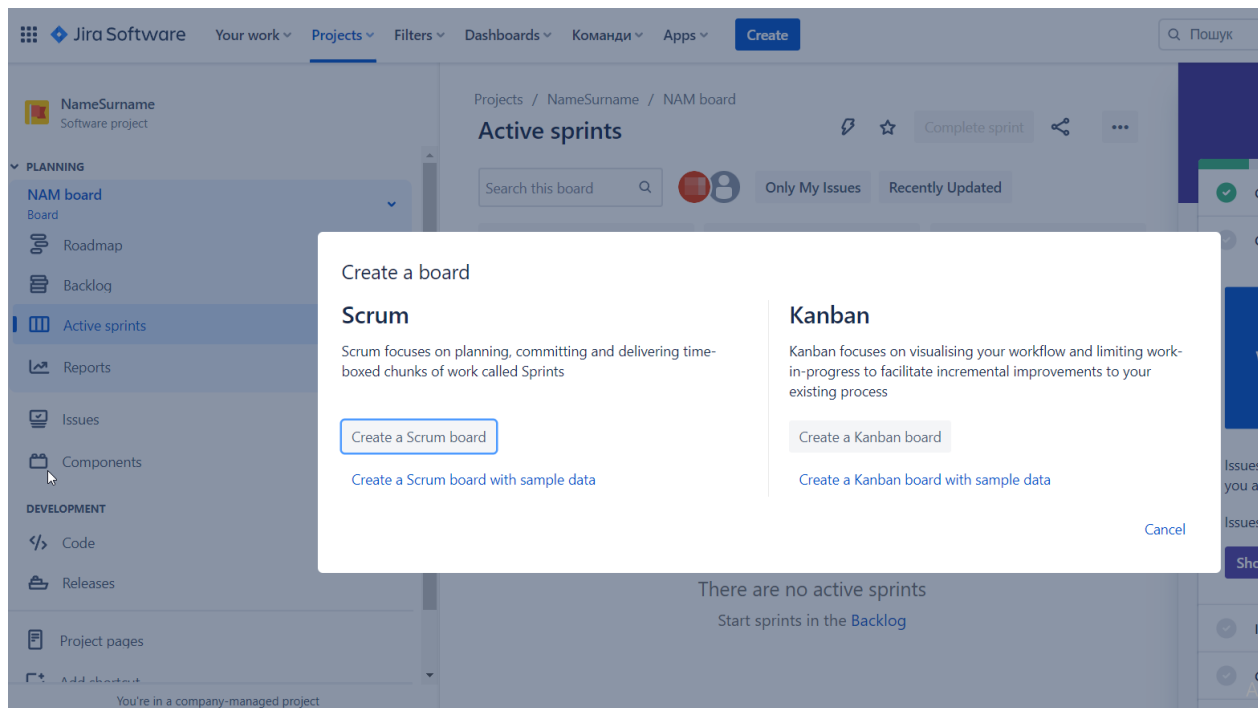


Рис. 3.6 Вибір типу дошки

Хоча візуально дошки схожі, різниця між дошками полягає у відмінностях стратегій для управління проєктами за методикою Agile. Методики Kanban передбачають циклічну та велику рухливість, а Scrum ґрунтується на коротких структурованих спринтах роботи.

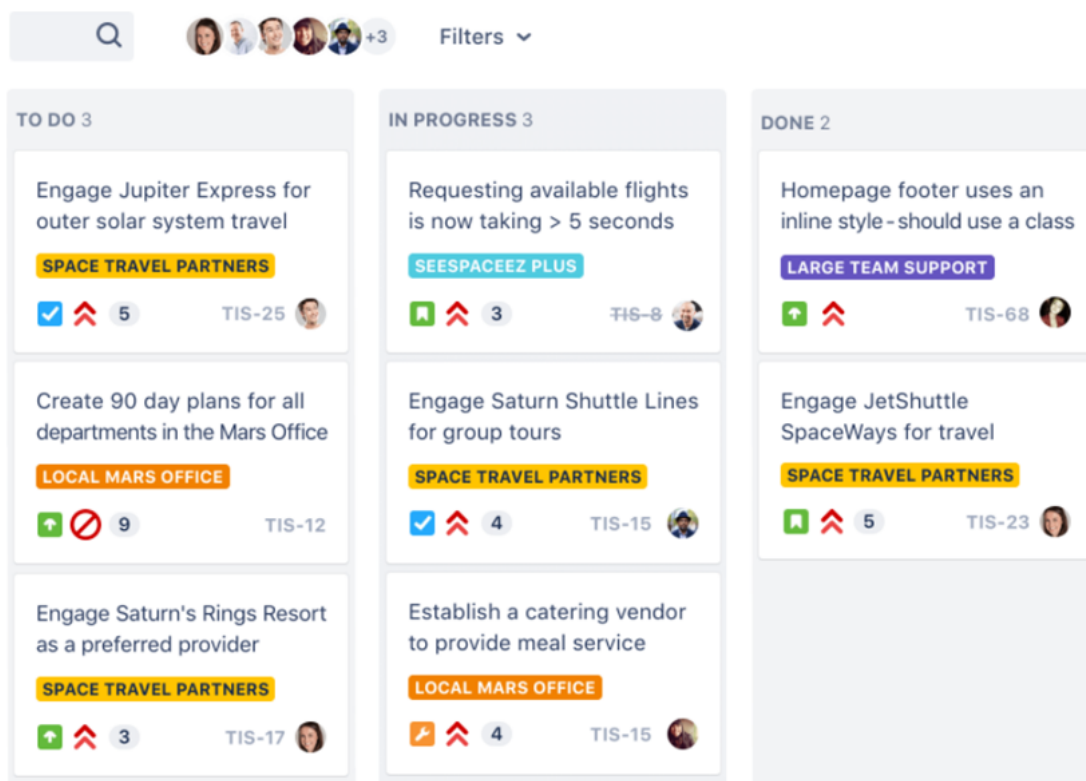


Рис. 3.7 Приклад дошки Scrum

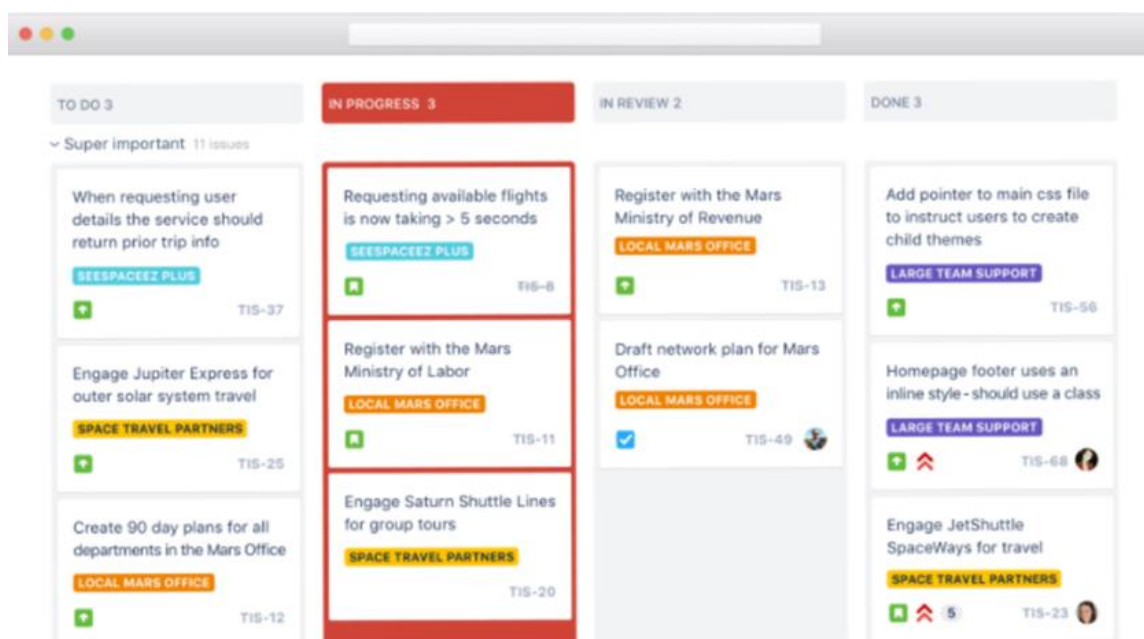


Рис. 3.8 Приклад дошки Kanban

Задачі, що створені в Jira, називаються issue (завдання). Завданням може бути що завгодно: баг, задача щодо розробки, відгуки з форм, що потребують відповіді. Кожне завдання відповідає окремій частині роботи, яку потрібно виконати. Завданню присвоюється унікальний ID, що спрощує його пошук.

Для оптимізації управління в Jira завдання поділяються на 5 типів:

- Epic.
- Story.
- Task.
- Sub-task.
- Bug.

Epic (епік) – велике завдання, яке вирішується за декілька спринтів. Наприклад, епік «Розробити форму А для сайту N», до задач якого можна віднести створення дизайну, опрацювання структури форми, розробка форми, аналітика тощо. Ця задача дуже об'ємна, відповідно кількість учасників буде великою. Епік буде розбитий на частини, в яких будуть описані детальні кроки для вирішення проблеми. Ці частини мають назву Story.

Story (історія) – частина епіка, команда може вирішити таке завдання за 1 спринт. Історія описує реалізований функціонал (роботу) зі сторони кінцевого користувача. Наприклад, в продовження розглядання прикладу епіка «Розробити форму А для сайту N» можна виділити наступні історії:

- Як користувач, можу заповнити та відправити форму для замовлення на сайті.
- Як UX дизайнер, можу розробити зручну форму для найвибагливішого користувача та ін.

Це зроблено для того, щоб частини роботи стали згрупованими та зрозумілими.

Task (завдання) – технічна задача для одного члена команди. Зазвичай, технічні задачі не пов'язані з командною роботою, проте вони необхідні для успішного завершення епіків. Наприклад, завдання створити репозиторій для проєкту, налаштувати тестове оточення тощо. Task може оцінюватися в годинах або ж створюються підзавдання, тому що не завжди є можливість визначити необхідну кількість часу для певної задачі.

Sub-task (підзавдання) – частина історії або завдання, в якій описується мінімальний обсяг роботи одного члена команди; Підзавдання спрощують процес контролю виконання роботи та дозволяють більш точно визначити трудовитрати. Статуси відображають що зроблено, що знаходиться в роботі та ще не розпочаті роботи. Підзавдання створюються членом команди особисто під час планування роботи на наступний спринт.

Bug (баг) – задача, в якій описується помилка в системі. Завданням є фіксування виявлених помилок, що будуть проаналізовані та згодом виправлені.

Для групування завдань за тематикою в Jira використовуються компоненти та мітки. Щоб розпочати роботу з компонентами, необхідно їх створити (адміністратором Jira або керівником проєкту).

Компоненти – це аналог категорій. Дозволяє розділити об'ємну роботу на частини. Наприклад, робота над додатком може бути поділена на такі компоненти, як технічне завдання, документація, сервер тощо.

The screenshot shows the 'Create Issue' form in Jira. The form is titled 'Create Issue' and has a 'Configure Fields' button. The 'Project' field is set to 'Training Center (QATLTC)'. The 'Issue Type' field is set to 'Task'. The 'Summary' and 'Reporter' fields are empty. The 'Component/s' field is highlighted with a red box, and a red arrow points to it. The dropdown menu for 'Component/s' is open, showing three options: 'Attestations', 'Internal Task', and 'Management'. The 'Description' field is empty. The 'Priority' field is set to 'P2'. The 'Labels' field is empty. At the bottom of the form, there are buttons for 'Create another', 'Create', and 'Cancel'.

Рис. 3.10 Вікно створення завдання

Мітки можна додати під час створення задачі,

Вибрати мітки або компоненти можна так:

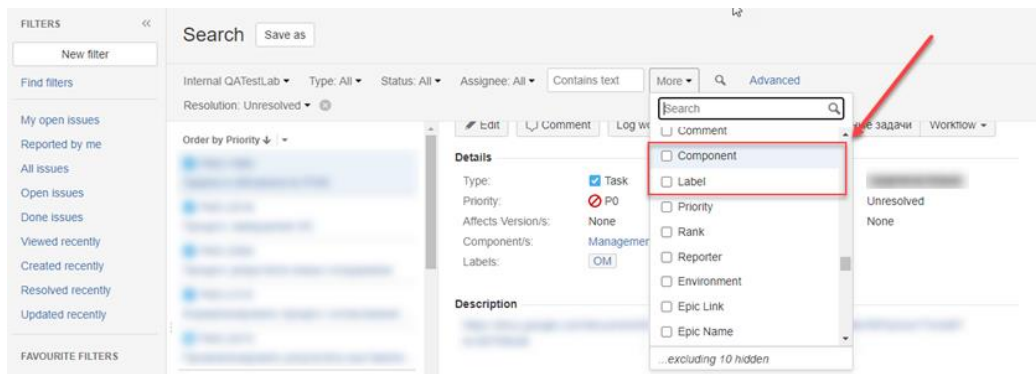


Рис. 3.13 Список додавання мітки (компоненти) до завдання

Вибрати конкретну мітку (або компонент), можна здійснити наступним чином:

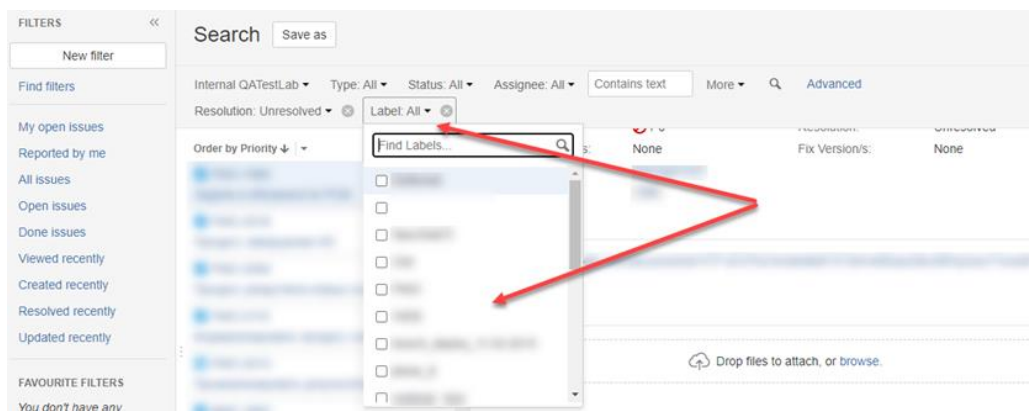


Рис. 3.14 Вибір конкретної мітки (компоненти) для завдання

Для створення нового завдання необхідно натиснути кнопку «Create» у верхній панелі управління.

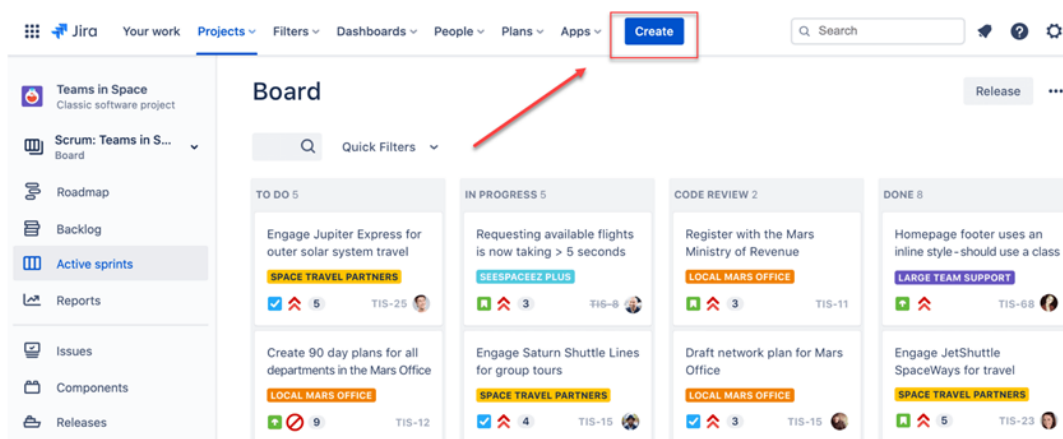


Рис. 3.15 Кнопка створення завдання

Відкриється модальне вікно створення завдання:

Create Issue

Configure Fields

Project*

Issue Type*

Task

Summary*

Reporter*

Start typing to get a list of possible matches.

Component/s

None

Description

Style

B

I

U

A

Visual

Text

Fix Version/s

Start typing to get a list of possible matches or press down to select.

Priority

P2

Labels

Begin typing to find and create labels or press down to select a suggested label.

Labels

Attachment

Drop files to attach, or browse.

Linked Issues

blocking

Issue

Begin typing to search for issues to link. If you leave it blank, no link will be made.

Assignee

Automatic

Assign to me

Epic Link

Choose an epic to assign this issue to.

Create another

Create

Cancel

Рис. 3.16 Модальне вікно створення завдання

Обов'язково потрібно вказати при створенні завдання:

- Project (проект) до якого належить завдання. За замовчуванням завдання створюється в тому проекті, який був відкритий або в якому була створена попередня задача.
- Issue Type (тип завдання) – слід вибрати відповідний з 5 типів.
- Summary (тема) – необхідно створювати з короткою назвою, з якого зрозуміла основна суть завдання.

- Reporter (автор) – той, хто створив завдання.

До додаткових полів відносяться:

- Priority (пріоритет) – вказується терміновість, з якою необхідно виконати завдання. За замовчуванням всі завдання створюються із середнім пріоритетом Medium, для невідкладної задачі вказується пріоритет High або Highest. Найвищий називається Blocker, його слід виставляти коли зламалося щось важливе для ПЗ (не здійснюється оплата, не додаються товари до кошика). Якщо завдання не термінове, то вказується пріоритет Low або Lowest.

- Components (компоненти) – дозволяє прикріплювати завдання до відповідного компоненту (див. вище).

- Assignee (виконавець) – співробітник, який повинен виконати завдання.

- Description (опис) – докладний опис завдання.

- Labels (мітки) – вибір наявної або створення нової мітки (див. вище).

- Attachment (вкладення) – дозволяє додати файли до завдання.

- Linked issues (зв'язані задачі) – дозволяє вибрати завдання, з якими пов'язана дана задача. Для звичайного посилання одного завдання на інше обирається тип relates to.

- Epic Link – вказується посилання на Epic (див. вище).

Робочий процес в Jira являє собою набір статусів і переходів, через які проходить завдання під час свого життєвого циклу (Workflow). Він може містити п'ять основних стадій:

- Open Issue (задача відкрита).
- Resolved Issue (задача вирішена).
- InProgress Issue (задача в процесі вирішення).
- ReOpened Issue (задача перевідкрита).
- Close Issue (задача завершена).

Можна створити окремий життєвий цикл для кожного проекту в Jira, але для створення або розширення такого флоу необхідно залогінитися в систему як Адміністратор з глобальними правами на створення і редагування Workflow. Оскільки процес розробки ПЗ дуже гнучкий і варіативний, то для різних команд і продуктів можуть знадобитися різні статуси й переходи між ними. В баг трекінгових системах найчастіше є можливість ручного додавання нових статусів тікетів.

Можна вибрати не тільки ім'я статусу, але також колір для позначення його «категорії» (статус для тікетів які готові до роботи, в процесі роботи та завершені). І для зручності завантажити іконку, щоб виділити його на тлі інших статусів.

Для додавання нового статусу необхідно:

- Увійти як користувач з дозволом адміністраторів Jira.
- Перейти на сторінку Statuses, вибравши значок Cog Icon> Issues > знайти Statuses ліворуч в категорії Issue Attributes.
- Натиснути «Add Status», вказати ім'я, опис та категорію статусу, яку потрібно додати.

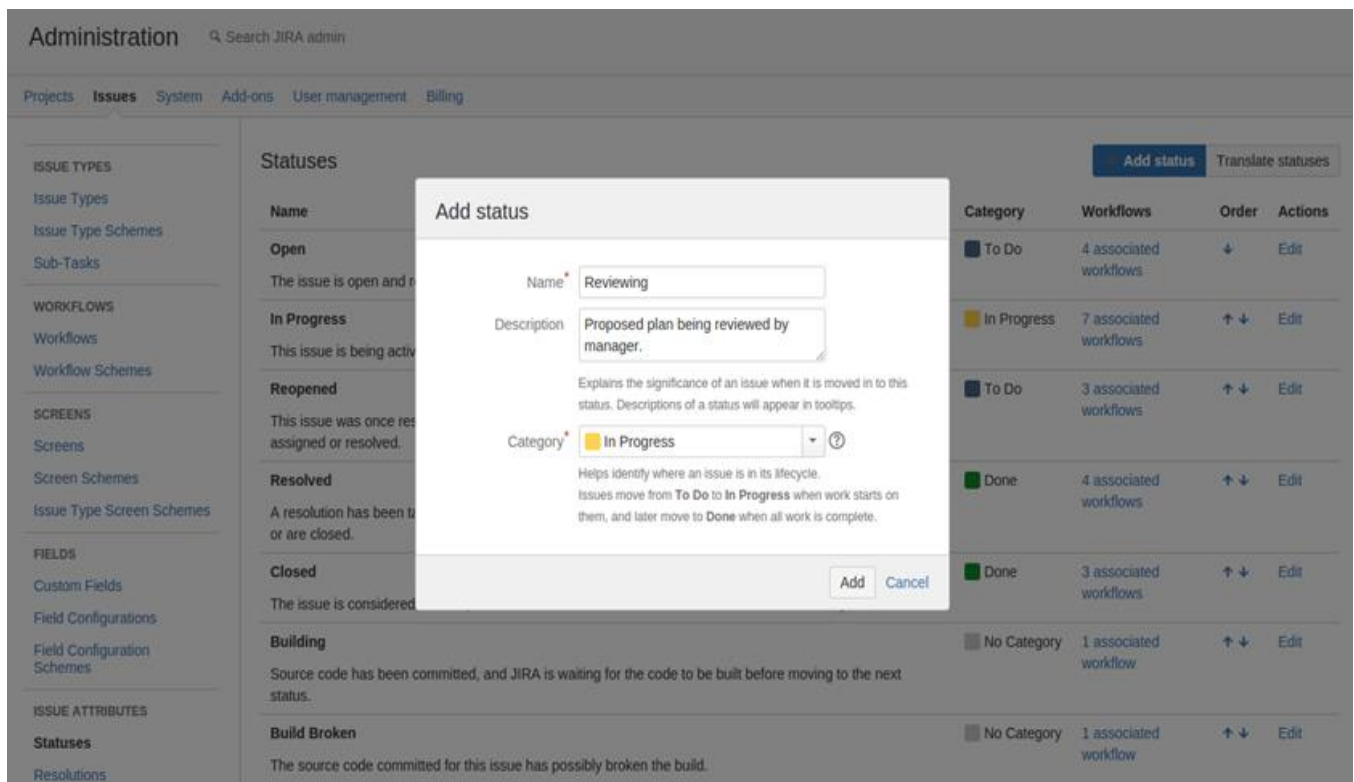


Рис. 3.17 Вікно додавання статусу для завдання

Далі слід додати новий статус до Workflow:

- Натиснути «Cog Icon»>«Issues»> знайти Workflow в категорії Workflows ліворуч.
- Натиснути посилання «Edit y Workflow», до якого потрібно додати статус.
- Натиснути «Add Status», щоб визначити попередні статуси. Вибрати «Revising and Add».

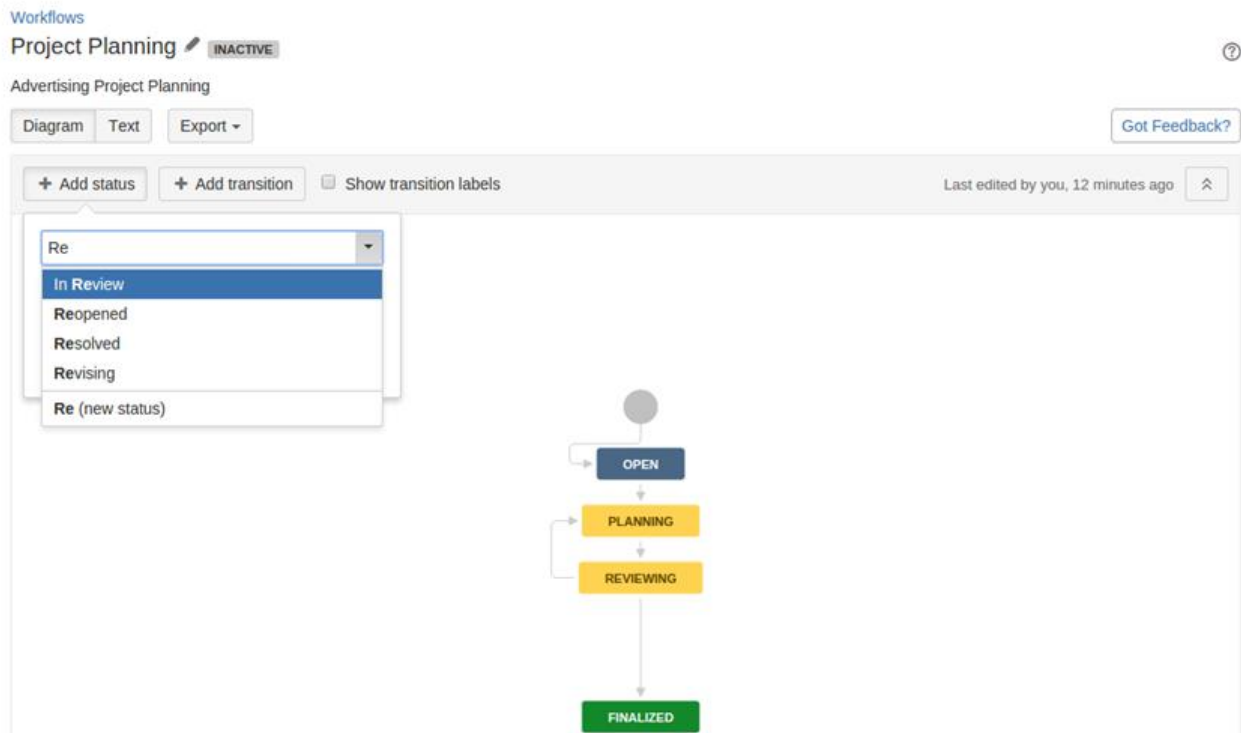


Рис. 3.18 Вікно створення статусу

Натиснути на будь-якому вузлі «Reviewing status» та перемістити його до вузла «Revising Status», щоб створити перехід. З'явиться вікно «Add Transition»;

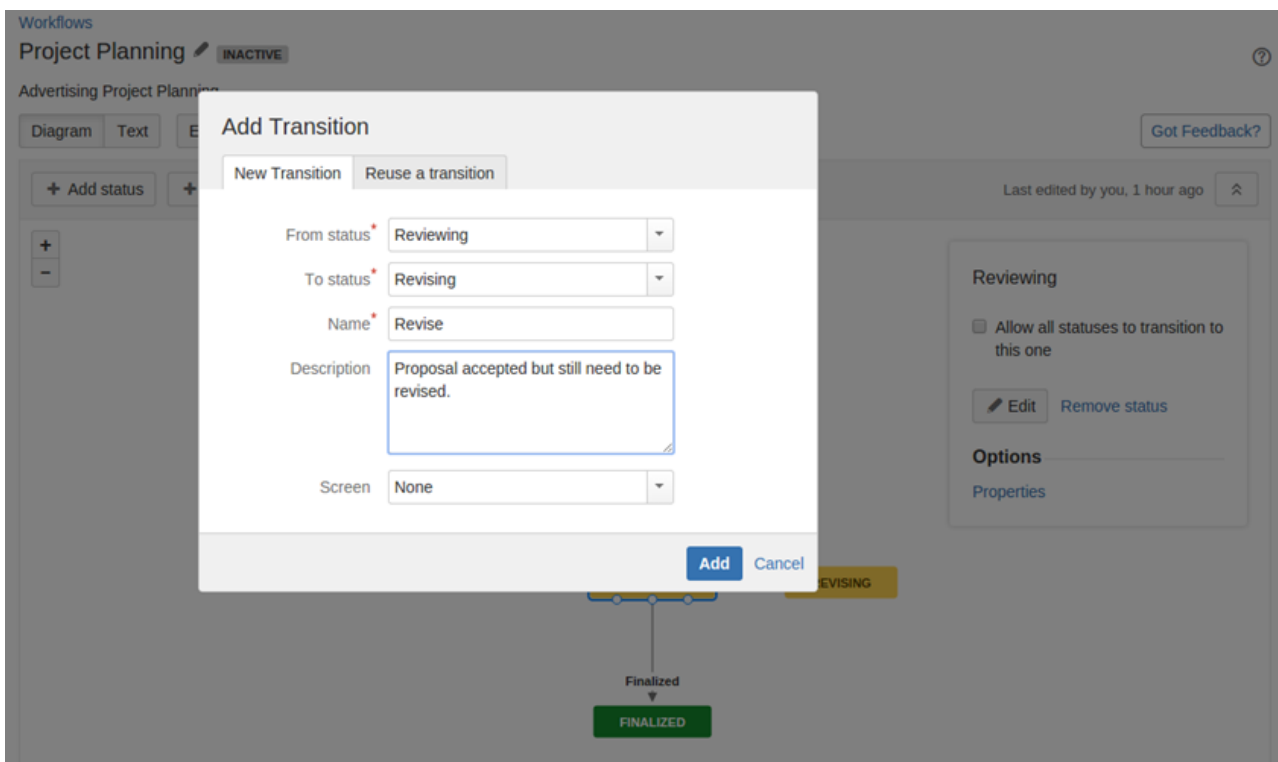


Рис. 3.19 Додавання транзакції

Закінчивши редагування Workflow, необхідно опублікувати чернетку або активувати робочий процес. При створенні власного Workflow необхідно брати до уваги такі параметри, як:

- величину проєкту і команди;
- тривалість проєкту;
- досвід роботи з даною командою і досвід роботи в цілому.

Після створення задачі та активації Workflow розпочинається проходження життєвого циклу. Розглянемо приклад життєвого циклу показаного на рисунку.



Рис. 3.20 Приклад життєвого циклу

Всі нові задачі мають статус «Backlog». Задачі які знаходяться в цьому статусі потребують уточнення щодо пріоритету, термінів, виконавця тощо. Після призначення задачі на виконавця та додавання повної інформації щодо терміну виконання назначається статус «Selected for work» або «In progress», якщо над задачею була розпочата робота. Виконавши зазначену задачу їй присвоюється статус для перевірки «Ready for review».

Після етапу перевірки можливо два сценарії, при першому – задачі присвоюється статус «Done», означає що всі роботи виконані, або задачу перевідкривають зі статусом «Reopened» та повертають її в роботу «In progress». Це можливо якщо по задачі необхідні додаткові уточнення або ж вона не реалізована в достатньому об’ємі, містить

помилки чи розбіжності. Виконана задача зі статусом «Done» також може бути переглянута та повернена у роботу зі статусом «In progress», у випадку якщо були додані нові деталі для її проходження, тощо.

Задачі, які були реалізовані за вимогами, вирішені в процесі, втратили актуальність або були скасовані замовником мають кінцевий статус «Archived». На будь-якому проєкті навігація вирішуваних задач є однією з важливих складових успіху. Зі зростанням обсягу проєкту розібратися в потоці пріоритетних задач стає все складніше, тому важливо знайти методику за якою можна спростити процес управління то допомогти учасникам зорієнтуватись у виконанні задач.

3.3 Завдання

1. Створити новий проєкт у системі Jira.
2. Налаштувати Scrum або Kanban дошку відповідно до обраної методології.
3. Ознайомитися з типами тікетів (Epic, Task, Bug, Story тощо) та описати їх призначення.
4. Створити мітки та компоненти; пояснити їхнє призначення у структуризації задач.
5. Створити задачі різних типів та призначити їх конкретним виконавцям.
6. Відфільтрувати задачі за мітками, компонентами та іншими параметрами.
7. Розробити індивідуальний Workflow для типу тікета Bug.
8. Продемонструвати проходження життєвого циклу тікета – від створення до завершення.
9. Підготувати звіт про виконану роботу, оформити згідно з вимогами, сформулювати висновки.

3.4 Контрольні запитання

1. Що таке Jira і для чого вона використовується в управлінні IT-проєктами?
2. Яка різниця між Scrum та Kanban дошками в Jira?
3. Які основні типи тікетів існують у Jira та які завдання вони описують?
4. Що таке Workflow у Jira, і як його можна змінювати відповідно до типу задачі?

5. Як створити новий проєкт у Jira? Які налаштування доступні на цьому етапі?
6. Що таке компоненти в Jira, і як вони впливають на організацію задач?
7. Яка роль міток у фільтрації та класифікації тікетів?
8. Як відбувається призначення задач конкретним виконавцям у межах проєкту?
9. Яким чином можна фільтрувати задачі за різними параметрами (статус, виконавець, термін тощо)?
10. Опиши життєвий цикл тікета в Jira: від створення до закриття.
11. Як реалізується сповіщення про зміни в задачах у Jira?
12. Які є переваги використання Jira для командної роботи над проєктами?
13. Як інтегрувати Jira з іншими сервісами (наприклад, GitHub, Slack)?
14. Що таке backlog у Scrum-дошці та як з ним працювати?
15. Яким чином Jira дозволяє відстежувати прогрес виконання задач і спринтів?

ЛАБОРАТОРНА РОБОТА № 4 (4 години)

«РОБОТА ІЗ МЕТОДОЛОГІЄЮ SCRUM»

Тема роботи: робота із методологією Scrum.

Мета роботи: поглиблення знань у роботі із платформою Jira, формування статистичних діаграм відносно спринта.

4.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи здобувач повинен ознайомитися з методологією гнучкого управління проєктами Scrum, розглянути її основні елементи та реалізувати проєкт із використанням Scrum-підходу. Необхідно створити Scrum-дошку, заповнити Backlog задачами різного типу (Task, Bug тощо), провести планування спринту, делегувати задачі виконавцям, контролювати хід виконання завдань та проаналізувати підсумки після завершення спринту. Окрему увагу слід приділити оцінці задач, візуалізації статусів та статистиці продуктивності команди.

Постановка завдання: ознайомитися з методологією Scrum та реалізувати її елементи в управлінні проєктом: створити спринт, розподілити задачі та здійснити повний цикл роботи над проєктом у Scrum-середовищі.

4.2 Теоретичні відомості

Користувацьчі історії (User Story) – ефективний спосіб викласти вимоги до програмного забезпечення, що знаходиться в розробці. Такі історії містять короткі поради від імені користувача ПЗ. Оскільки у методології Scrum постановка завдань – це зазвичай прерогатива замовника чи власника ПЗ, вони вважаються головним способом впливу на процес розробки. Кожна User Story має обмеження за обсягом тексту та складністю викладу. Історію найчастіше пишуть на невеликому аркуші, що саме по собі обмежує обсяг. Завдяки користувацьким історіям можна документувати побажання клієнта і оперативно реагувати на запити ринку.

User Story слід вважати спрощеним показником вимог, оскільки вона не має процедури приймального тестування. Складання історії користувача має відповідати приймальній процедурі. Це дасть гарантію, що User Story досягла своєї мети.

Структура історії має приблизно такий вигляд: “Якщо користувач є <тип користувача>, я хочу виконати <дія> для отримання <результат>” (As a product owner I want ...). Подібна структура не лише проста, а й зрозуміла кожному. Переваги застосування User Stories:

- Історії невеликі за обсягом та їх легко створювати.
- Допомагають усім зацікавленим особам обговорювати роботу над проєктом та його підтримку.
- Не потребують постійного обслуговування.
- Актуальні лише при використанні.
- Поліпшують взаємодію з клієнтом.
- Завдяки їм можна поділити проєкт на дрібні етапи.
- Полегшують роботу над проєктами із погано зрозумілими вимогами.
- Спрощують оцінку завдань.

Недоліки User Stories:

- Без попереднього узгодження процедури можуть ускладнити використання як бази для угоди.
- Їх використання потребує тісного контакту з клієнтом протягом усієї роботи над проєктом, що іноді ускладнює робочий процес.
- Має недоліки при масштабуванні на великих проєктах.
- Безпосередньо пов'язані з професійним рівнем розробників.
- Використовуються для початку обговорення, але можуть не завершувати обговорення і не застосовуються для документації системи.

Беклог продукту – це поточні завдання у вигляді списку, складеного за пріоритетністю. Список формують на базі дорожньої карти (roadmap) проєкту та викладених у ній пунктів. Найважливіші завдання зазвичай знаходяться на початку списку. Це необхідно для розуміння, яка робота має бути зроблена першою.

Команда розробників обирає швидкість виконання завдань беклога незалежно від побажань замовника, зважаючи на власну кваліфікацію та досвід минулих спринтів. “Підганяти” програмістів вкрай небажано. Команда сама обирає завдання з беклогу з власних міркувань та можливостей. Виконання відбувається без перерви (Kanban) або кількома ітераціями (Scrum).

Основа беклогу продукту складається з дорожньої карти, пропозицій та умов виконання. У епіках містяться умови та User Story. Давай розглянемо приклад типової дорожньої карти.



Рис. 4.1 Приклад типової дорожньої карти

Створення сайту “Команди в космосі” – перша пропозиція з дорожньої карти. Її потрібно розділити на епіки (на рисунку вони показані зеленим, синім та бірюзовим кольорами) та User Story для кожного епіка.

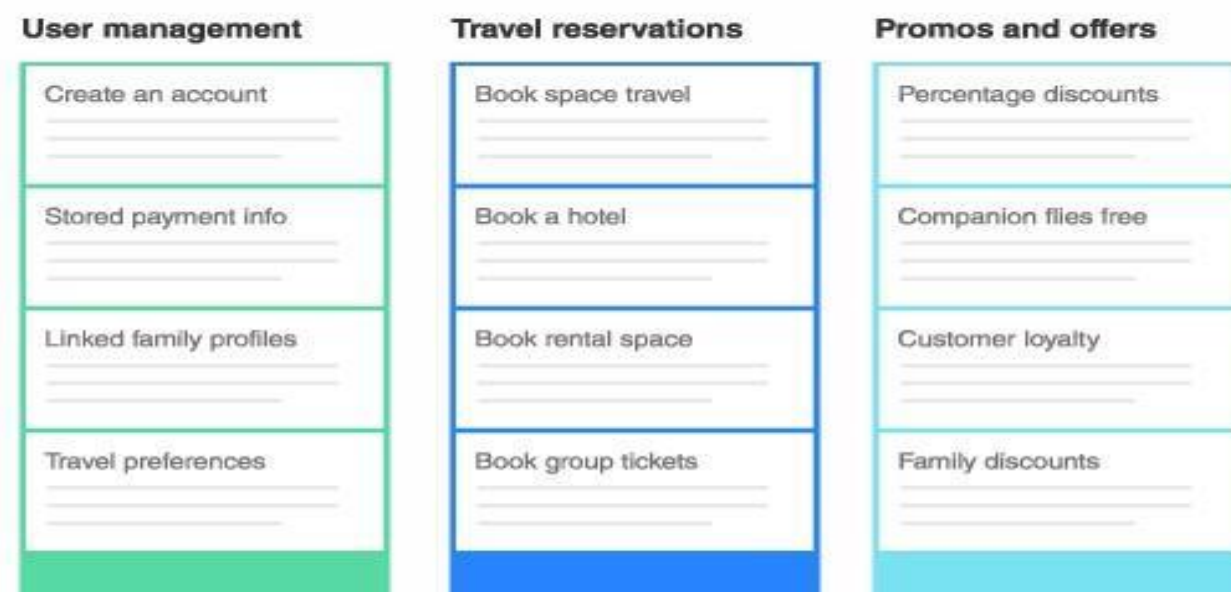


Рис. 4.2 Приклад епіка

Замовник ПЗ формує з кількох User Story один список. У разі необхідності він може змінити порядок виконання історій, щоб розробники спочатку зайнялися одним із найважливіших епіків (ліворуч) або перевірили, як працює пільгове бронювання квитків. Щоб це виконати, потрібно реалізувати історії з епіків (праворуч). Обидва варіанти можна побачити нижче.

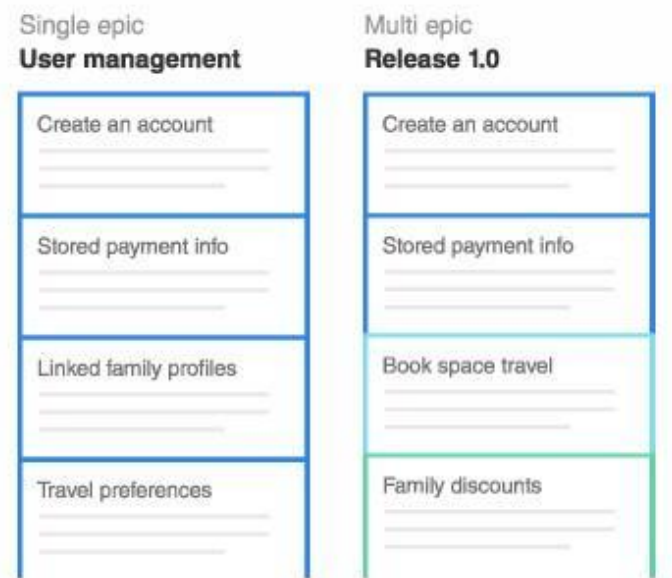


Рис. 4.3 Приклади двох епіків

На основі яких факторів замовник має розставляти пріоритети?

- Актуальність користувачів.
- Наявність зворотного зв'язку.
- Складність розробки.
- Взаємозв'язок між завданнями (щоб виконати “Б”, спочатку потрібно зробити “А”).

Пріоритети у роботі визначає замовник, але свою думку про це можуть висловити й інші сторони. Успіх беклогу залежить, у тому числі, від думки клієнтів та програмістів. Колективними зусиллями вони можуть досягти поліпшення результатів і забезпечити своєчасне постачання готового продукту.

Якщо беклог вже створено, потрібно періодично змінювати його під час подальшої роботи. Замовнику ПЗ варто впевнитись у правильному складанні беклогу перед кожним новим плануванням ітерації. Це допоможе уточнити пріоритети або змінити щось після аналізу останньої ітерації. Коригування беклогу в Agile іноді називають «грумінгом» або «уточненням» або «веденням беклогу».

Якщо беклог вже відносно великий, то замовнику потрібно згрупувати завдання щодо короткостроковості та довгостроковості виконання. Короткострокові завдання потрібно ретельно вивчити, перш ніж надати їм цей статус. Доведеться скласти User Story, з'ясувати всі нюанси всередині команди. Щодо довгострокових завдань, вкрай бажано, щоб розробники дали їм свою оцінку. Це спростить встановлення пріоритетів. Можливо, щось зміниться, але команда покращить розуміння завдань та швидше продовжить роботу.

Беклог – це важлива складова у роботі між замовником та командою програмістів. Замовник завжди може змінити пріоритети, вивчивши відгуки клієнтів, прогнози або після отримання нових вимог. Рекомендується уникати змін безпосередньо під час роботи. Це погано впливає на робочий процес та емоційний стан програмістів.

Спринт – це невеликий проміжок, під час якого потрібно виконати раніше обумовлений обсяг роботи. Спринти ґрунтуються на методологіях Scrum та Agile. Правильний вибір спринтів допомагає agile-команді розробляти якісне програмне забезпечення. “Застосовуючи Scrum, можна розробити продукт за кілька ітерацій із чіткою тривалістю – спринтами. Це допомагає розбивати великі проєкти на дрібні завдання”, – стверджує Меган Кук (англ. Megan Cook), керівник напряму Jira у компанії Atlassian.



Рис. 4.4 Схематичне зображення спринта

На думку авторів методології Scrum, для планування майбутнього спринту всім потрібно зустрітись на окремих зборах. На цьому заході учасники команди повинні з'ясувати відповіді на два головні питання: що потрібно зробити в цьому спринті і яким чином буде зробити це найкраще?

У визначенні списку робочих завдань беруть участь замовник, Scrum-майстер і програмісти. Замовник пояснює мету спринту та завдання з беклогу.

Потім команда розробляє план, за яким відбуватиметься виконання завдань спринті. Цей план разом із обраними робочими завданнями називається беклогом спринту. Після наради щодо планування команда стає до роботи. Розробники обирають завдання з беклога, у міру завершення роботи статус кожного завдання змінюється з «У роботі» на «Готово». Під час

спринту команда проводить щоденні Scrum-наради (стендапи), на яких обговорюються поточні проблеми та перебіг роботи. Такі зустрічі необхідні для виявлення труднощів, здатних вплинути на завершення спринту.

Якщо спринт завершено, команда показує результати своєї роботи на огляді підсумків (demo). З результатами може ознайомитись кожен учасник проєкту. Ознайомлення слід проводити до того, як готовий код буде мерджитися до робочого середовища. Завершує цикл спринтів ретроспектива. На ній команда визначає області, які потрібно покращити у майбутньому спринті.



Рис. 4.5 Схема циклу спринт

Більшість молодих команд стикаються з труднощами, вперше впроваджуючи спринти у свій робочий процес. Щоб уникнути проблем, рекомендуємо вивчити список дій, які потребують першочергової уваги. Що потрібно робити:

- Перевірте, що команда усвідомлює мету спринту та спосіб досягнення успіху. Це необхідно, щоб усі разом рухалися до успішного результату.

- Має бути чіткий та зрозумілий беклог. Якщо беклог вівся невірно, це може стати проблемою, здатною зашкодити робочому процесу.
- Переконайтеся, що ваша оцінка темпів роботи є правильною, з урахуванням літніх відпусток та інших факторів.
- Активно беріть участь у плануванні спринту. Заохочуйте членів команди розширювати план для історій, багів та завдань.
- Відмовляйтесь від завдань, під час яких розробникам не вдасться вирішити питання із залежностями.
- Після затвердження плану призначте співробітника, на якого буде покладено внесення даних до програми управління проєктами (картки Jira або ін.).

Чого варто уникати:

- Не зловживайте великою кількістю історій, тверезо оцінюйте темпи роботи та не призначайте завдання, які буде важко виконати на спринті.
- Пам'ятайте про якість роботи. Перевірте, чи є у вас достатньо часу для контролю якості та виправлення помилок у коді.
- Подбайте, щоб усі члени команди чітко розуміли вміст спринту. Не женіться за швидкістю. Уся команда має рухатися разом.
- Не навантажуйте на розробників зайвий обсяг роботи. Незабаром буде ще один спринт.
- Якщо команда висловлює хвилювання щодо навантаження чи дедлайну, варто врахувати ці думки. Розберіться з проблемами та відкоригуйте їх у разі потреби.

Scrum-дошка – це інструмент, який демонструє, як виконується робота Scrum-команди. Відображати інформацію на такій дошці можна на папері, стіні або в електронному вигляді (JIRA, Trello). Scrum-дошка складається не менше ніж з трьох стовпців: “зробити”, “в роботі” та “готово”. Ось приклад дошки:

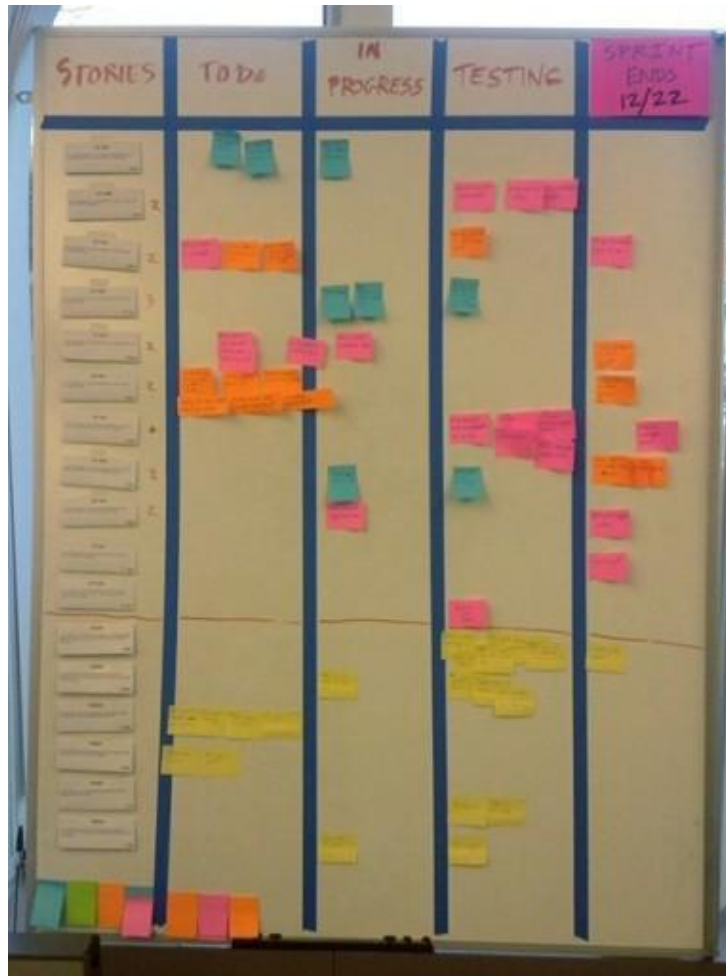


Рис. 4.6 Реальний приклад дошки із завданнями

На Scrum-дошці розміщена вся інформація з беклогу, який раніше затвердили на плануванні. Зазвичай картки бізнес-завдань закріплено на дошці за пріоритетом зверху донизу. Можна поділити їх на конкретні типи робіт (робота над кодом, дизайн та інші).

Після того, як частину роботи завершено, картку пересувають до сусідньої колонки. Показати видимість прогресу роботи команди допомагає “робота, яка залишилася” за днями на Burndown Chart.

Можна також використовувати дошку із фліпчартами. На ній назви робіт пишуть на паперових наклейках та прикріплюють їх на дошку. Щойно робота закінчена, стікери пересувають до іншої колонки.

Діаграма згоряння завдань (Burndown chart) показує кількість роботи, яку виконано, і роботи, що залишилася. Вона оновлюється щодня та доступна всім зацікавленим особам. Графік необхідний відображення прогресу у роботі над спринтом.

Є два види діаграм:

- Burndown chart із відображенням прогресу роботи у спринті.

- Burndown chart із відображенням прогресу роботи до випуску продукту (дані сумуються з кількох спринтів).

Приклад діаграми:

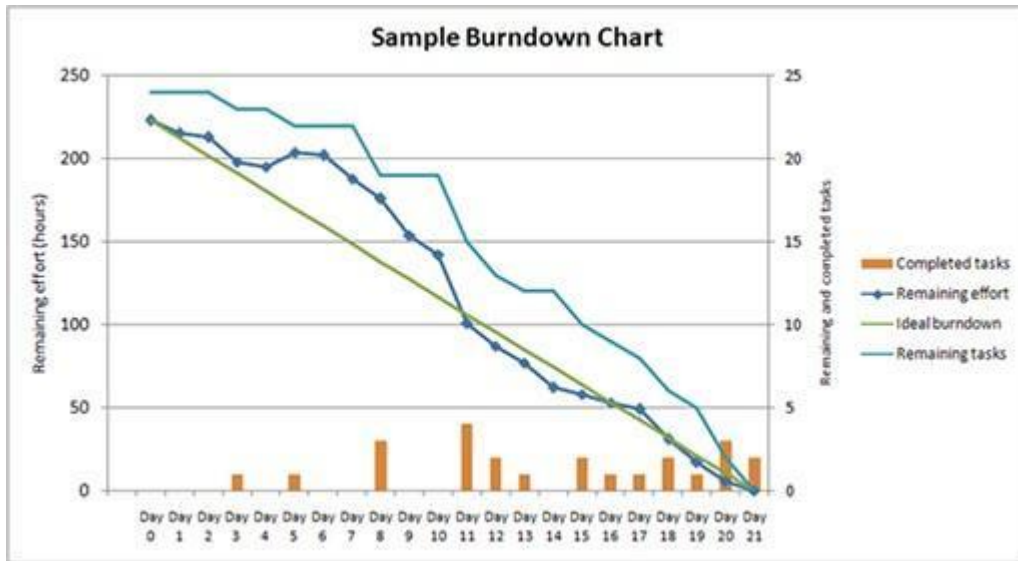


Рис. 4.7 Приклад діаграми просування завдань

У цьому прикладі використовується психологія: діаграма показує не кількість зроблених завдань, а кількість не зроблених (залишок).

Тобто, якщо команда зробила 90 завдань зі 100, може виникнути хибне відчуття, що вже все готове. Адже прогрес із 90 до 100 завдань особливо нічого не змінює.

Якщо ж відображати кількість завдань, що залишилися, не можна не помітити, як із кожним разом їх ставати все менше. Це підсвідомо підганяє учасників проєкту швидше досягти мети – на дошці не повинно залишитися недороблених завдань.

4.3 Завдання

1. Створити новий проєкт, у якому реалізовуватиметься Scrum-підхід.
2. Виконати базове налаштування Scrum-дошки.
3. Заповнити Product Backlog задачами різного типу (Task, Bug, Story, Epic).
4. Провести оцінку задач (наприклад, у годинах або за принципом story points).
5. Призначити задачі конкретним виконавцям відповідно до їхніх ролей.
6. Створити Sprint, налаштувати його параметри (назва, тривалість, ціль, опис).
7. Перенести відповідні задачі до Sprint Backlog.
8. Запустити спринт на виконання.

9. Контролювати прогрес виконання задач через статуси (To Do / In Progress / Done).
10. Завершити спринт, зафіксувати його результати.
11. Провести аналіз ефективності виконання спринту, використовуючи діаграми (burndown chart, velocity тощо).
12. Підготувати структурований звіт про виконану роботу із висновками.

4.4 Контрольні запитання

1. Що таке Scrum і які його основні принципи?
2. Які ролі передбачені в Scrum-команді та які їхні функції?
3. Що таке Product Backlog та хто відповідає за його наповнення?
4. Які типи задач найчастіше використовуються у Scrum?
5. Що таке Sprint і які етапи він охоплює?
6. Як відбувається планування спринту в Scrum?
7. Що таке Sprint Backlog і чим він відрізняється від Product Backlog?
8. Які інструменти Scrum використовуються для моніторингу прогресу виконання задач?
9. Що таке Daily Scrum і для чого він проводиться?
10. Як оцінюється складність або обсяг задач у Scrum?
11. Які графіки та діаграми дозволяють візуалізувати ефективність виконання спринту?
12. Що відбувається під час завершення спринту (Sprint Review та Sprint Retrospective)?
13. Які переваги надає використання Scrum у порівнянні з традиційними моделями управління проєктами?
14. Як можна адаптувати Scrum до невеликих команд або короткострокових проєктів?
15. У яких програмних інструментах найзручніше реалізувати Scrum-дошку та вести спринти?

ЛАБОРАТОРНА РОБОТА № 5 (4 години)

«ОСНОВИ РОБОТИ З GIT ТА GITHUB»

Тема роботи: основи роботи з Git та GitHub.

Мета роботи: познайомитися з Git та GitHub. Почати використовувати для роботи з проєктами Node-RED.

5.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи ознайомитися з основними командами системи керування версіями Git, принципами створення локального репозиторію, індексування, фіксації змін (commit), перегляду історії та використання графічного інтерфейсу. Також необхідно встановити Node-RED, налаштувати роботу з проєктами та дослідити, як ця система інтегрується з Git для відстеження змін у локальному проєкті. Особливу увагу слід приділити практичному застосуванню Git як з командного рядка, так і в GUI-інтерфейсах.

Постановка завдання: у звіті мають бути наведені коди команд, скриншоти виконаних дій, коментарі до процесу та загальні висновки щодо ефективності Git як інструменту для розробки ПЗ.

5.2 Теоретичні відомості

Завантажте <https://notepad-plus-plus.org/downloads/> та встановіть редактор Notepad++, якщо він ще не встановлений у вас в системі.

Завантажте та інсталюйте Git <https://git-scm.com/downloads>. При інсталяції залишайте усі опції за замовченням, окрім редактору, виберіть Notepad++.

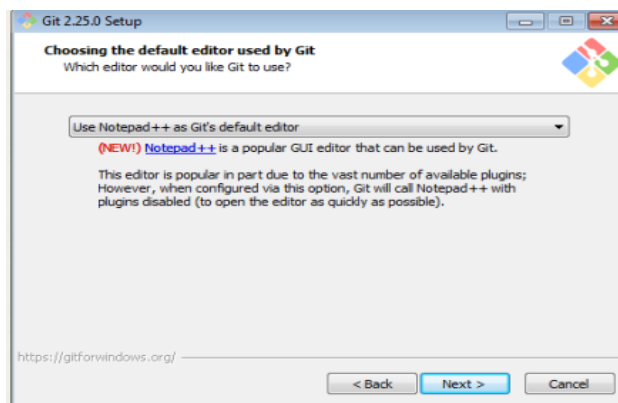


Рис. 5.1 Встановлення Git

У одній із директорій створіть папку «Project». Перейдіть до папки і через контекстне меню виберіть Git Bash Here

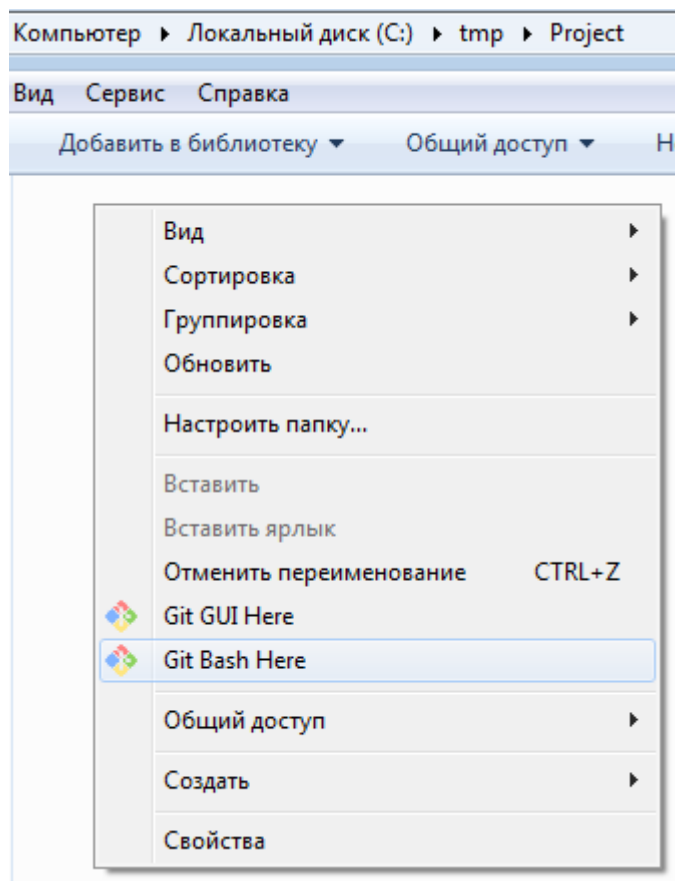


Рис. 5.2 Запуск Git Bash

В консолі введіть команду перегляду конфігурації

`git config --list`

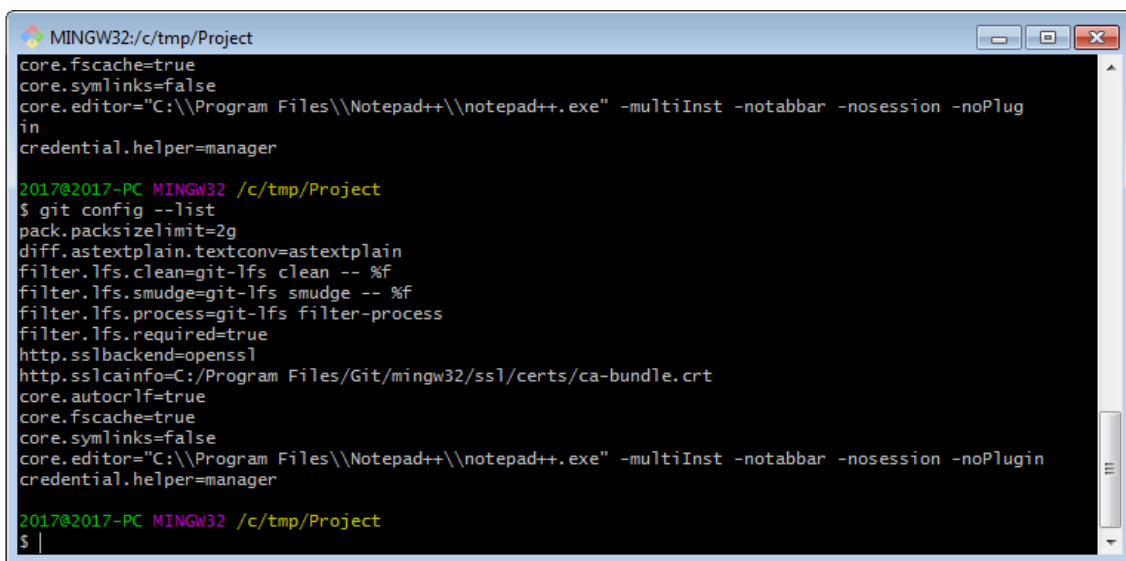


Рис. 5.3 Результат виконання команди для перегляду конфігурацій

У переліку конфігурацій немає зареєстрованого користувача. Використовуючи команди додайте свого користувача та пошту.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Після цього перевірте, що вони добавлені через команду `git config --list`. Використовуючи команду `git init` ініціалізуйте репозиторій.

Зробіть налаштування провідника, щоб відобразилися приховані файли та директорії. Перевірте, що в робочій папці створилася папка з назвою `.git`

Використовуючи Notepad++ у папці створіть текстовий документ «file1X» (де X – номер варіанту) з трьома рядками і збережіть.

Використовуючи `git status`, перевірте стан репозиторію. Зробіть копію екрану.

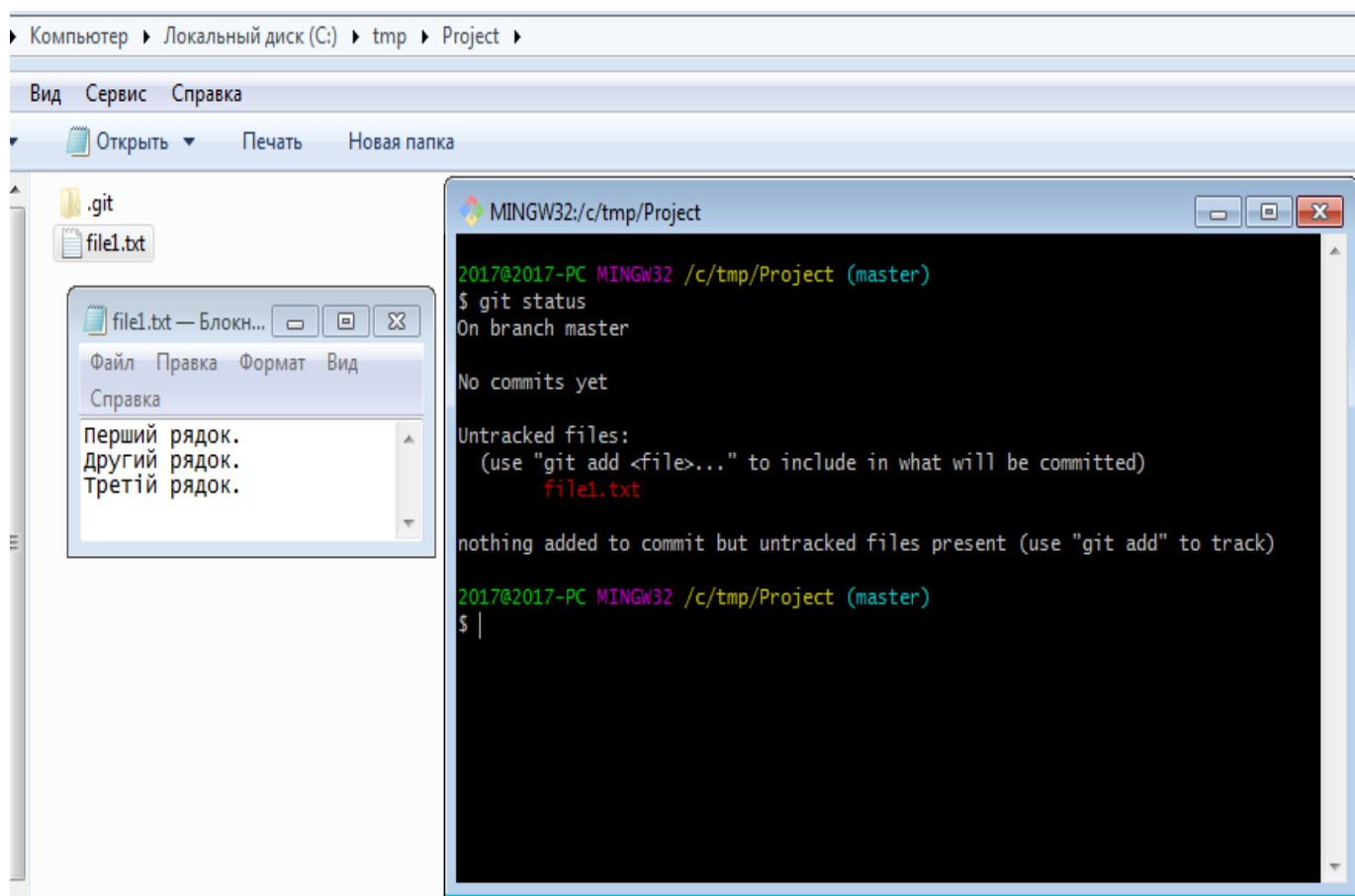


Рис. 5.4 Створення файлу та його коміт через консольну утиліту

Використовуючи команду «Get Gui Here», контекстного меню в папці проекту викличте графічний інтерфейс. Проконтролюйте щоб в налаштуваннях проекту стояв шрифт `utf-8`.

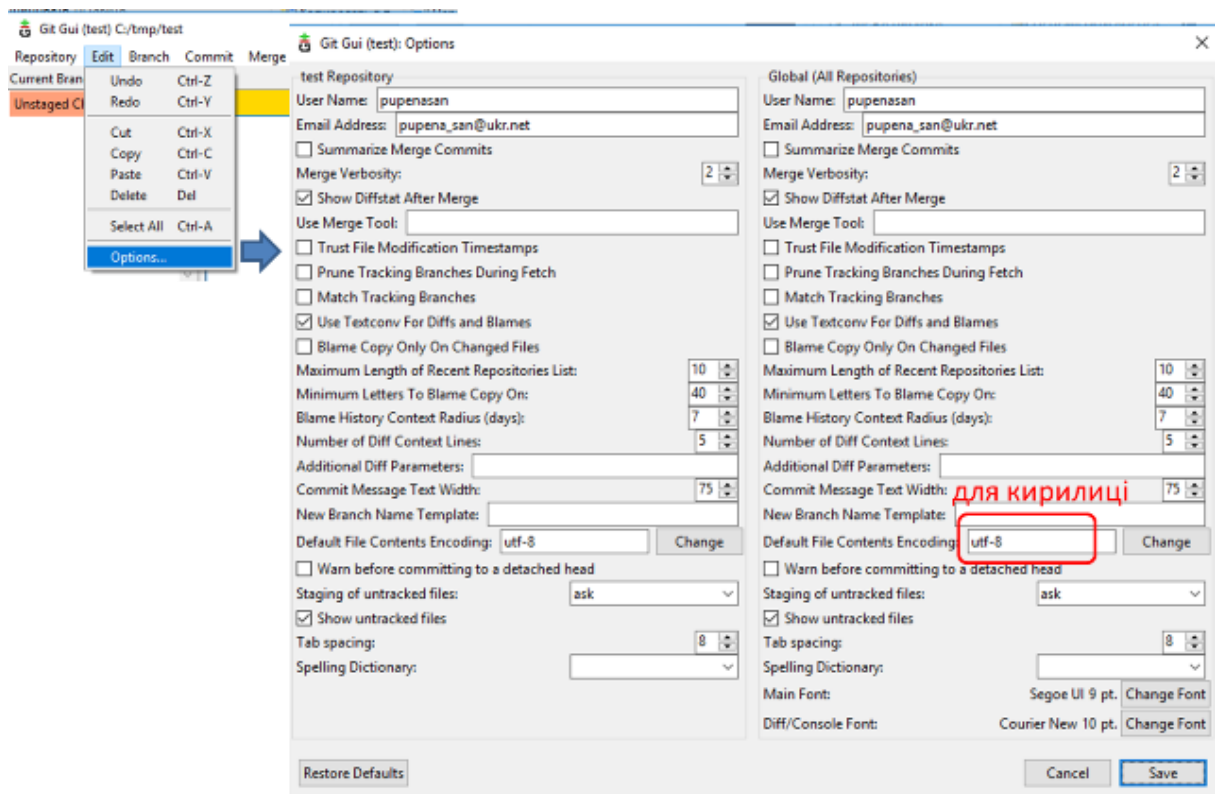


Рис. 5.5 Налаштування роботи з кодуванням шрифтів в Git

Проаналізуйте зміст. Зробіть копію екрану.

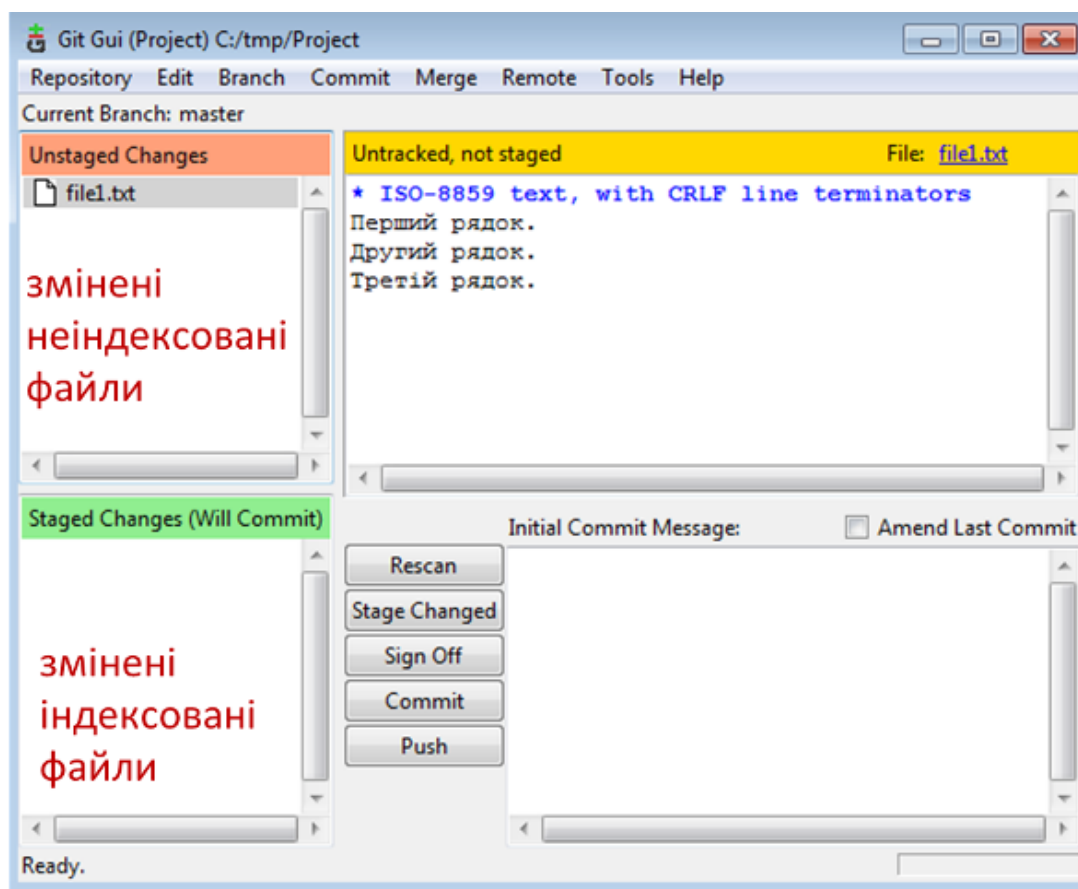


Рис. 5.6 Перегляд змін у файлі

Запустіть команду для додавання файлу на індексування.

```
git add file1X.txt
```

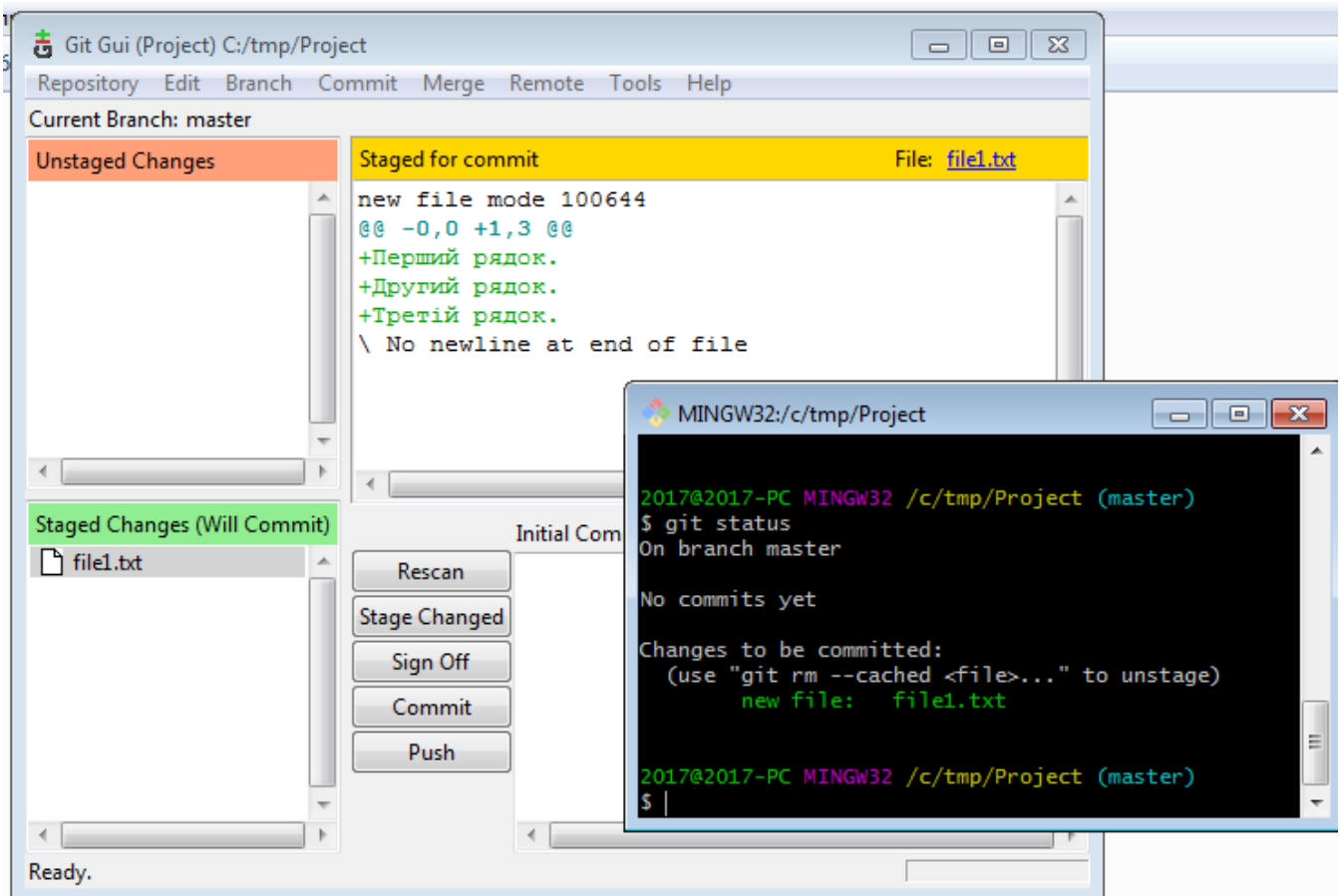


Рис. 5.7 Додавання файлів у індекс

Запустіть команду коміту з повідомленням (опція -m) «Перша версія проєкту», повторіть пункт 5.

```
git commit -m 'Перша версія проєкту'
```

Порівняйте збережені копії екранів, зробіть висновок, щодо їх змісту.

Створіть новий файл в робочій директорії з назвою file2X.txt. Запишіть туди три довільні рядки. У першому файлі видаліть другий рядок, та додайте в кінець рядок з написом «четвертий рядок». Додайте обидва файли до індексу та зробіть коміт.

```
git add *.txt
```

```
git commit -m 'Друга версія проєкту'
```

Виконайте команду для перегляду історії проєкту.

```
git log
```

Зробіть копію екрану. Відкрийте графічний інтерфейс, викличте меню «Repository->Visualize master's History», передивіться історію комітів.

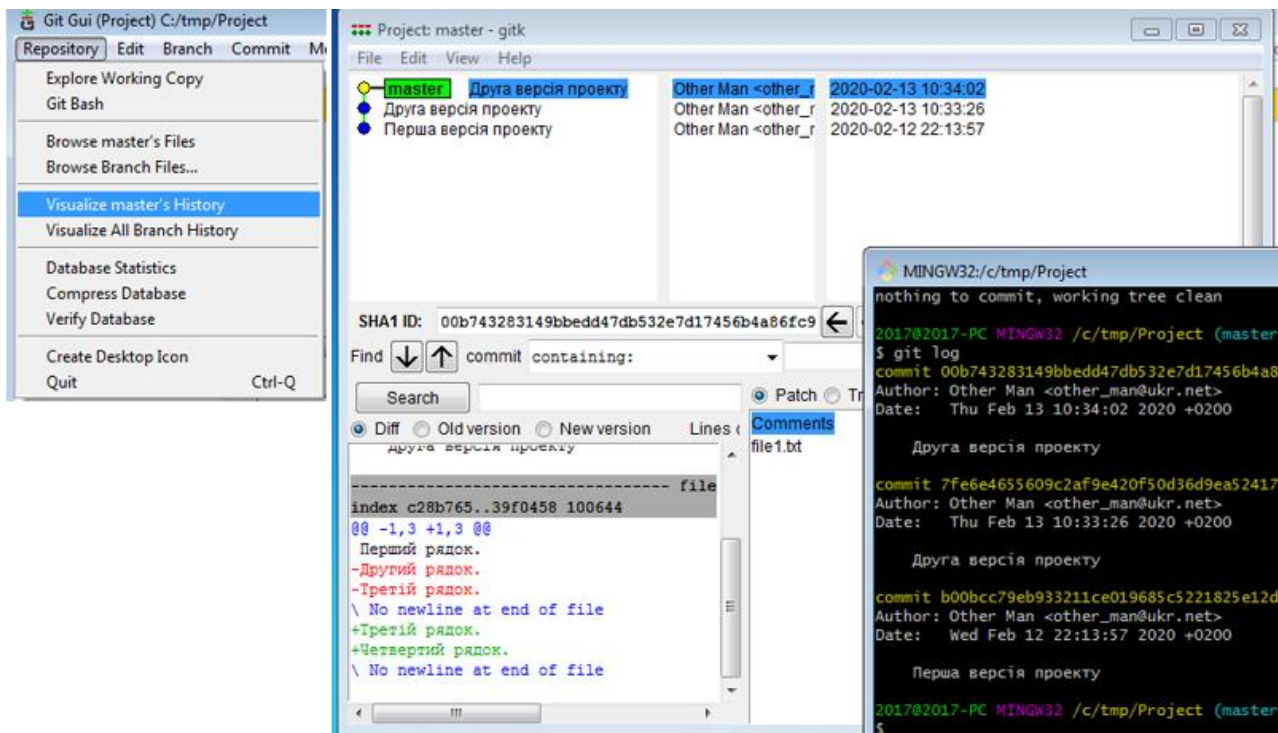


Рис. 5.8 Перегляд історії комітів

Зробіть копію екрану.

Інсталяція Node-RED під Windows можна здійснити за посиланням: <https://nodered.org/docs/platforms/windows> (підтримувані платформи: Windows 10, Windows 8, Windows 7).

Завантажити msi-файл Node.JS LTS версії <https://nodejs.org/uk/>. Запустити на виконання msi-файл від імені адміністратора і встановити Node.JS, при виклику діалоговий вікон все залишати за замовчуванням

Після інсталяції запустіть командний рядок (CMD), у якому введіть:

```
node --version && npm -version
```

npm (Node Package Manager) - це менеджер пакунків для мови програмування JavaScript. Для середовища виконання Node.js є менеджером пакунків за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через веб-сайт npm. Менеджер пакунків та реєстр керуються npm, Inc. Інсталювання проводиться з використанням npm з командою install. Наберіть в командному рядку:

```
npm install -g --unsafe-perm node-red
```

Після цього почнеться процедура інсталяції.

node-red

Можуть виникнути повідомлення про розблокування брандмауером, з якими треба погодитись. Відкрийте браузер, перейдіть до редактору Node-Red, за посиланням <http://127.0.0.1:1880/>. Для того, щоб Node-RED виконувався, вікно з командним рядком не можна закривати.

Node-RED має можливість працювати в режимі проєктів, де на одному робочому місці можна створювати кілька проєктів і керувати ними. За допомогою Notepad++ відкрийте конфігураційний файл settings.js що знаходиться в папці «.node-red» за місцем розташування файлів користувача. Наприклад, якщо зареєстрований користувач в системі «User1», то розміщення буде:

C:\Users\User1\.node-red

Змініть налаштування, активувавши проєкти, як показано на рисунку. Збережіть файл.

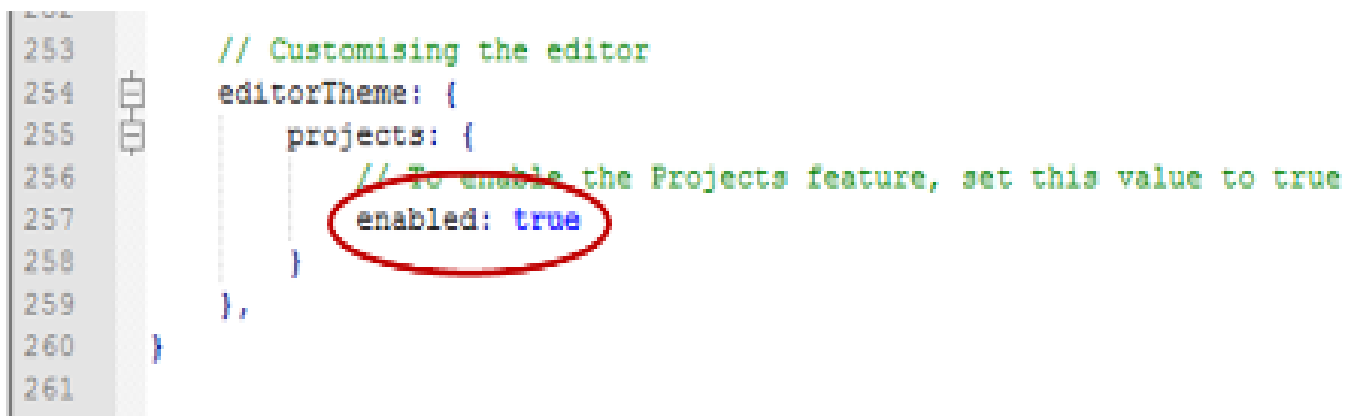


Рис. 5.9 Активація опції роботи з проєктами в Node-RED

Запустіть node-red. Перший раз, після активації опції проєктів, node-red запропонує створити новий проєкт:

У першому вікні необхідно вибрати опцію «Create project». У другому вікні треба ввести користувача. У третьому вікні вказати ім'я нового проєкту, після чого натиснути «Next». У четвертому вікні система запропонує усі існуючі потоки програми node-red перемістити в указаний файл. Це дасть можливість зробити імпорт цих файлів за необхідності. Натисніть Next. У п'ятому вікні зробіть відмову від шифрування «Disable encryption» після чого натисніть «Create Project». На останній сторінці натисніть «Done».

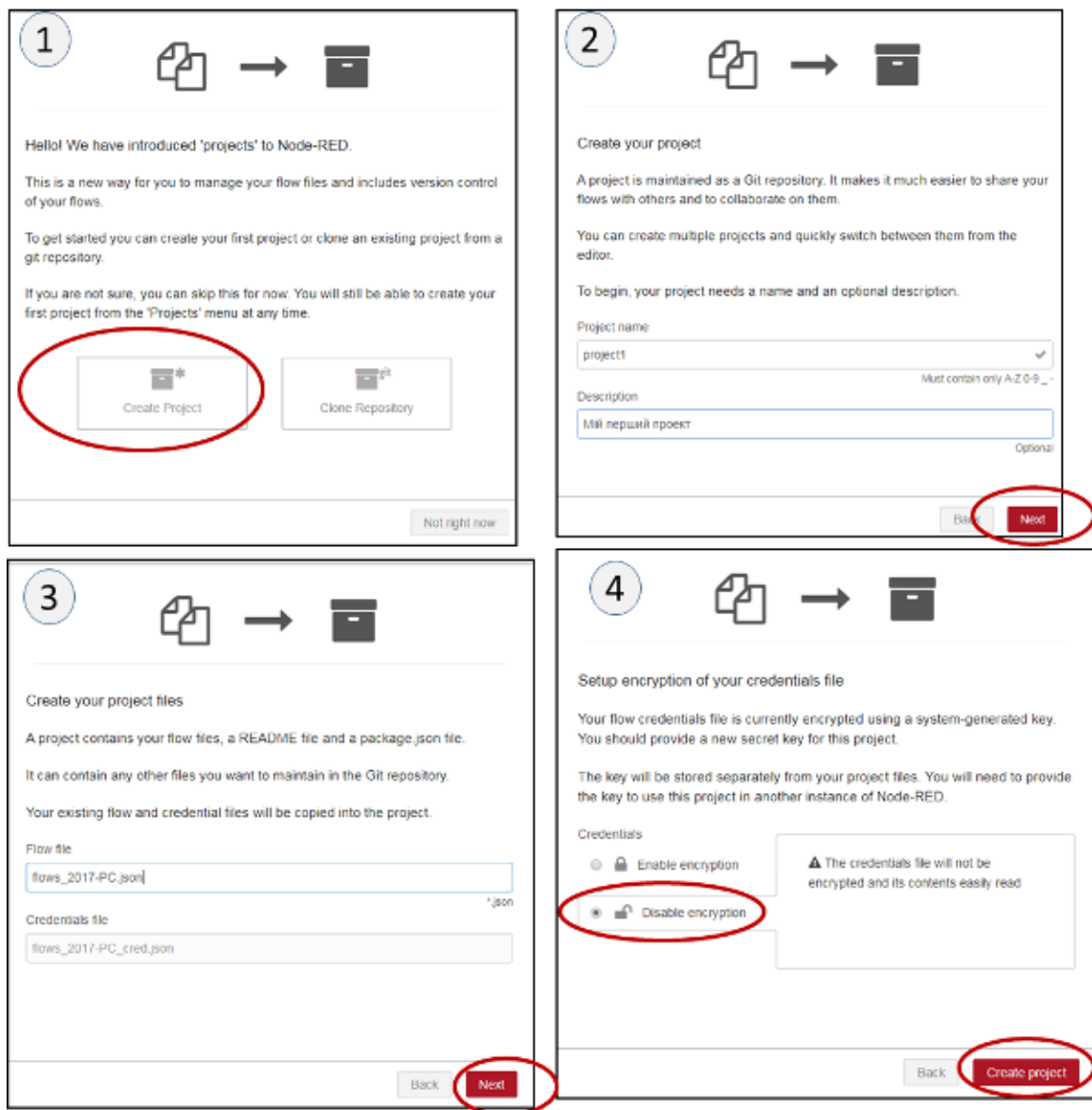


Рис. 5.10 Створення проєкту в Node-RED

Новий проєкт включить в себе усі існуючі до цього потоки. Після цього зробіть розгортання проєкту. Перейдіть на папку node-red

C:\Users\<ім'я користувача>\.node-red

Там можна побачити папку «projects», де зберігатимуться усі локальні проєкти. У цій директорії знайдіть папку з назвою вашого проєкту і зайдіть в неї. Там буде кілька файлів і папка «.git». Node-RED використовує для ведення проєкту систему Git. Тому папка проєкту є робочою папкою Git з репозиторієм. Використовуючи «Git Gui» або «Git Bash» проаналізуйте стан проєкту. Перейдіть в Node-RED на закладку «Project History» і передивіться зроблені зміни в локальному репозиторію. Як видно, Node-RED надає інтерфейс для деяких основних команд керування Git.

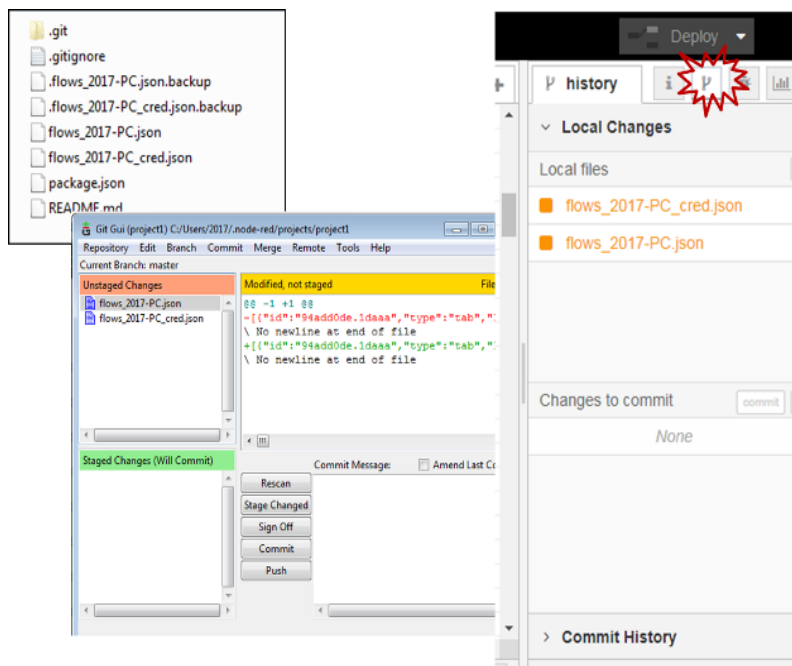


Рис. 5.11 Project History в Node-RED

У Node-RED на закладці «Project History - Local Changes» через кнопку «+All» зробить індексування усіх змінених файлів. Після цього вони з'являться в Changes to commit. Натисніть «Commit» і в полі повідомлення введіть «Мій перший коміт». Перейдіть на вкладку «Project History – Commit History» і подивіться історію комітів. Виберіть останній коміт і у вікні що з'явиться подивіться деталі змін, які були зроблені. Використовуючи з «Git Gui» утиліту «Visualize Master History» порівняйте зміни.

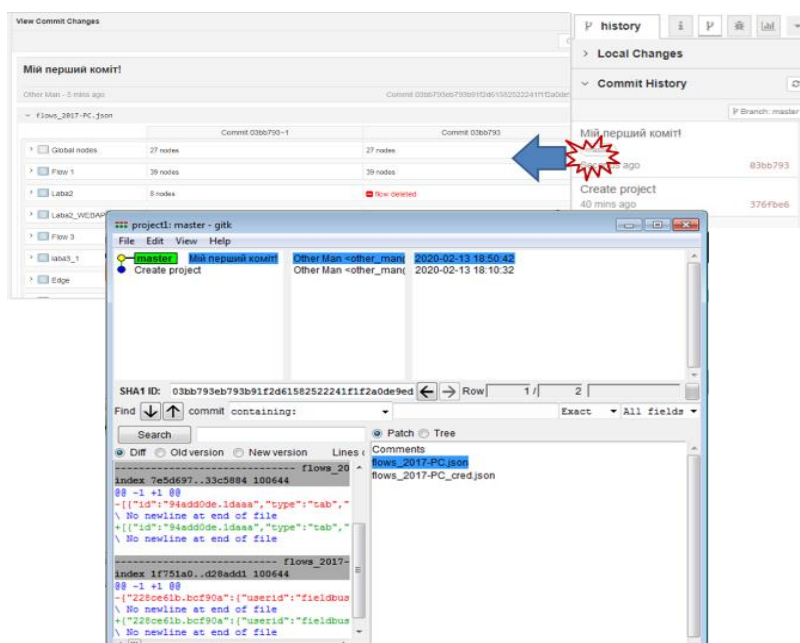


Рис. 5.12 Візуалізація змін в Node-RED

5.3 Завдання

1. Встановити Notepad++ та Git із офіційних джерел. Під час інсталяції Git обрати Notepad++ як редактор.
2. Створити робочу папку Project, відкрити Git Bash у цій папці.
3. Перевірити конфігурацію Git (`git config --list`) та додати своє ім'я й email.
4. Ініціалізувати локальний репозиторій командою `git init`.
5. Увімкнути відображення прихованих файлів, переконатися в наявності папки `.git`.
6. Створити текстовий файл `file1X.txt` з трьома рядками. Перевірити стан репозиторію командою `git status`.
7. Використовуючи Git GUI, переконатися в коректному кодуванні файлу (UTF-8).
8. Додати файл до індексу `git add file1X.txt`, виконати перший коміт `git commit -m "Перша версія проєкту"`.
9. Створити `file2X.txt`, модифікувати `file1X.txt`, індексувати обидва файли, зробити другий коміт.
10. Переглянути історію комітів за допомогою `git log` та Git GUI (Visualize master's history).
11. Встановити Node.js та Node-RED через командний рядок та npm.
12. Активувати режим роботи з проєктами у Node-RED через редагування файлу `settings.js`.
13. Запустити Node-RED, створити новий проєкт, додати потоки, виконати деплой.
14. Дослідити наявність `.git`-папки у директорії проєкту Node-RED.
15. Переглянути зміни у проєкті через вкладку Project History, виконати коміт змін та переглянути їх через Commit History.
16. Порівняти результати з візуалізацією в Git GUI (Visualize Master History).
17. Підготувати звіт із детальними описами кожного кроку, скріншотами та поясненнями.

5.4 Контрольні запитання

1. Що таке система керування версіями і для чого використовується Git?
2. Яка різниця між локальним та віддаленим репозиторієм?
3. Що відбувається при виконанні команди `git init`?
4. Для чого використовуються команди `git add` та `git commit`?
5. Яким чином можна переглянути поточний статус файлів у репозиторії?
6. Що таке індексація файлів у Git?
7. Які параметри можна налаштувати через `git config`?
8. Як переглянути історію змін у репозиторії?
9. Що таке `.git`-папка та яку інформацію вона зберігає?
10. Які переваги дає використання Git GUI?
11. Яким чином Node-RED інтегрує Git для управління проектами?
12. Що таке Project History у Node-RED і які функції воно виконує?
13. Як додати файли до коміту через Node-RED?
14. У чому відмінність між CLI-командами Git та діями в Node-RED?
15. Як Node-RED зберігає проекти й використовує Git як бекенд?

ЛАБОРАТОРНА РОБОТА № 6 (4 години)

«ОСНОВИ РОБОТИ З SSH КЛЮЧАМИ НА GITHUB»

Тема роботи: основи роботи з SSH ключами на GitHub.

Мета роботи: почати використовувати для роботи з проєктами Node-RED та навчитись створювати SSH ключ для роботи на GitHub.

6.1 Порядок виконання роботи і звітність

У ході виконання лабораторної роботи потрібно створити SSH-ключі на локальному комп'ютері та додати їх у власний акаунт GitHub для безпечного з'єднання. Далі необхідно створити приватний репозиторій, зв'язати його з локальним, а також завантажити (push) зміни через Git Bash. Потім слід перевірити результат на GitHub, здійснити push безпосередньо з Node-RED та переконатися у синхронізації змін. Наприкінці потрібно додати інших користувачів до репозиторію для колективної роботи та перевірити доступи.

Постановка завдання: опанувати навички створення, реєстрації та використання SSH-ключів для захищеного підключення до GitHub.

6.2 Теоретичні відомості

Для безпечного з'єднання з GitHub, який буде налаштовуватися в наступному пункті, можна використовувати кілька варіантів. Один з них SSH. Для цього необхідно створити SSH-ключ і прописати його в GitHub. У даному пункті необхідно створити SSH ключ, який використовуватиметься для доступу до віддалених GIT серверів. Детальне пояснення щодо створення ключа можете прочитати за посиланням. На локальному ПК запустіть «Git Bash» з папки користувача.

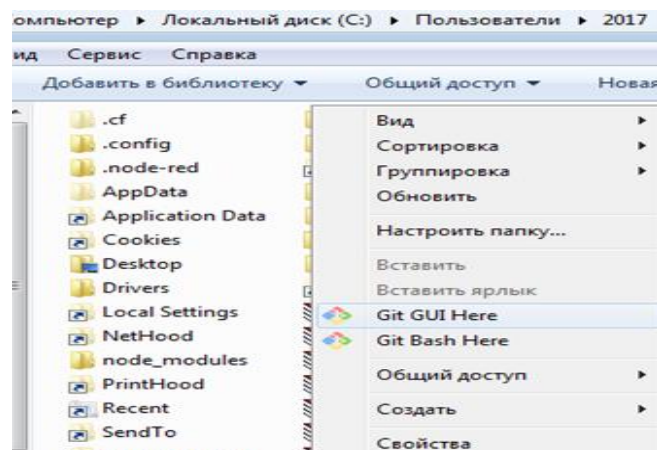
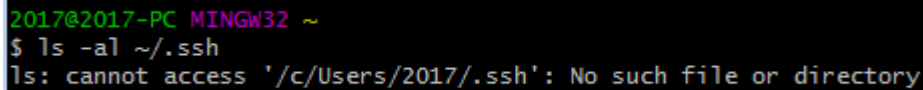


Рис. 6.1 Запуск Git Bash

Запустіть команду перевірки наявності ключів ssh

```
ls -al ~/.ssh
```

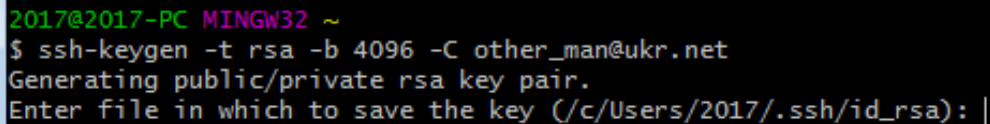


```
2017@2017-PC MINGW32 ~  
$ ls -al ~/.ssh  
ls: cannot access '/c/Users/2017/.ssh': No such file or directory
```

Рис. 6.2 Команда перевірки наявності ключів ssh

Для створення нового SSH ключа необхідно викликати команду в якій вказати свою поштову адресу. Увага, важливо вказувати свою поштову адресу, яка буде використовуватися в подальшому при реєстрації в GitHub.

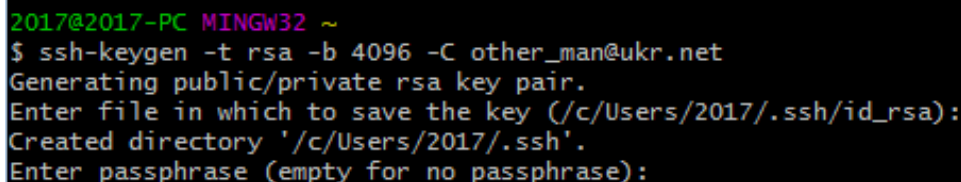
```
ssh-keygen -t rsa -b 4096 -C other_man@ukr.net
```



```
2017@2017-PC MINGW32 ~  
$ ssh-keygen -t rsa -b 4096 -C other_man@ukr.net  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/2017/.ssh/id_rsa): |
```

Рис. 6.3 Запуск створення нового SSH ключа

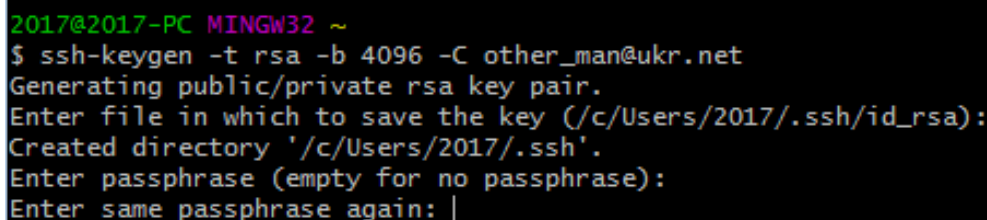
На прохання ввести назву файлу, натисніть клавішу «ENTER».



```
2017@2017-PC MINGW32 ~  
$ ssh-keygen -t rsa -b 4096 -C other_man@ukr.net  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/2017/.ssh/id_rsa):  
Created directory '/c/Users/2017/.ssh'.  
Enter passphrase (empty for no passphrase):
```

Рис. 6.4 Введення секретної фрази

Вкажіть пароль-фразу «passphrase» (необхідно запам'ятати пароль, потім буде вказуватися при з'єднання Git в Node-red). Пароль не буде відображатися при вводі.



```
2017@2017-PC MINGW32 ~  
$ ssh-keygen -t rsa -b 4096 -C other_man@ukr.net  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/2017/.ssh/id_rsa):  
Created directory '/c/Users/2017/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again: |
```

Рис. 6.5 Повторне введення секретної фрази

Повторно вкажіть пароль. Після цього виведеться повідомлення, в якому буде вказано файл з ключем. Цей файл потрібно буде відкрити текстовим редактором, наприклад Notepad++.

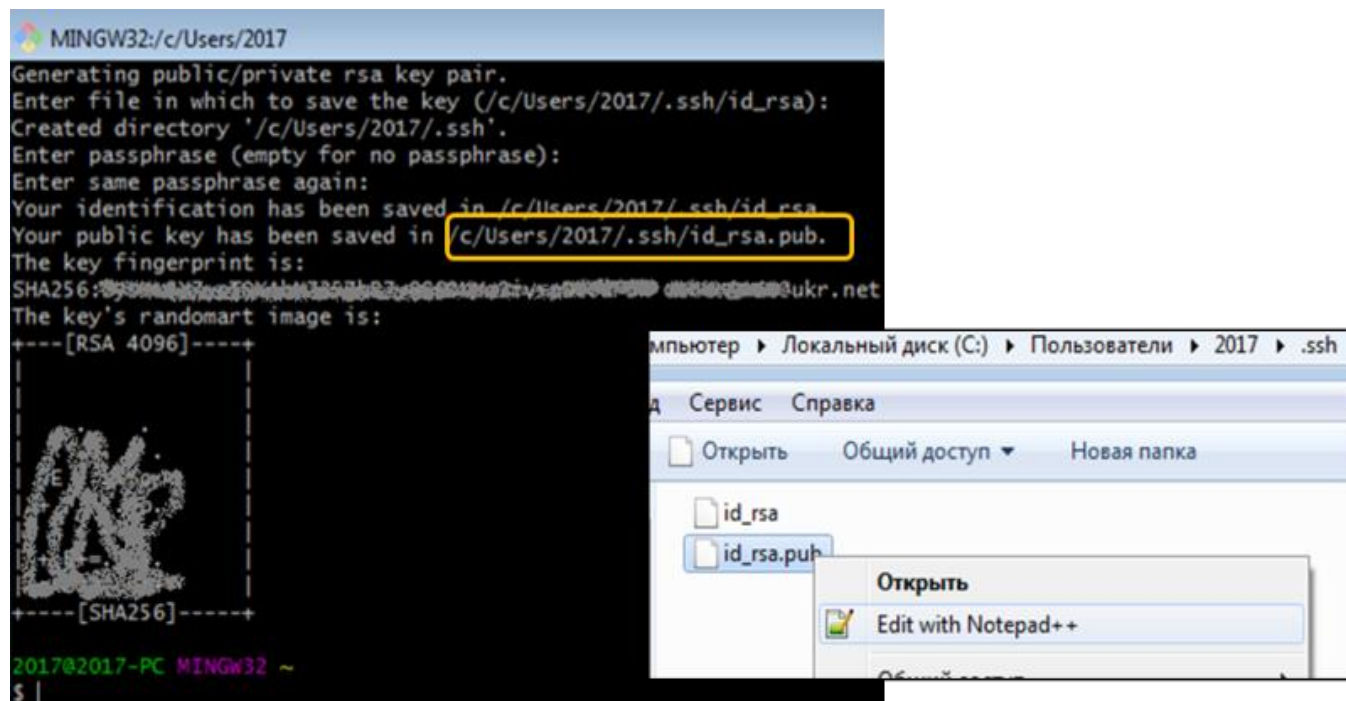


Рис. 6.6 Перегляд шифрованого ключа

Щоб здійснити реєстрацію на GitHub, зайдіть на сайт <https://github.com/>. Зареєструйтеся в системі. Опис процесу реєстрації наведений за цим посиланням.

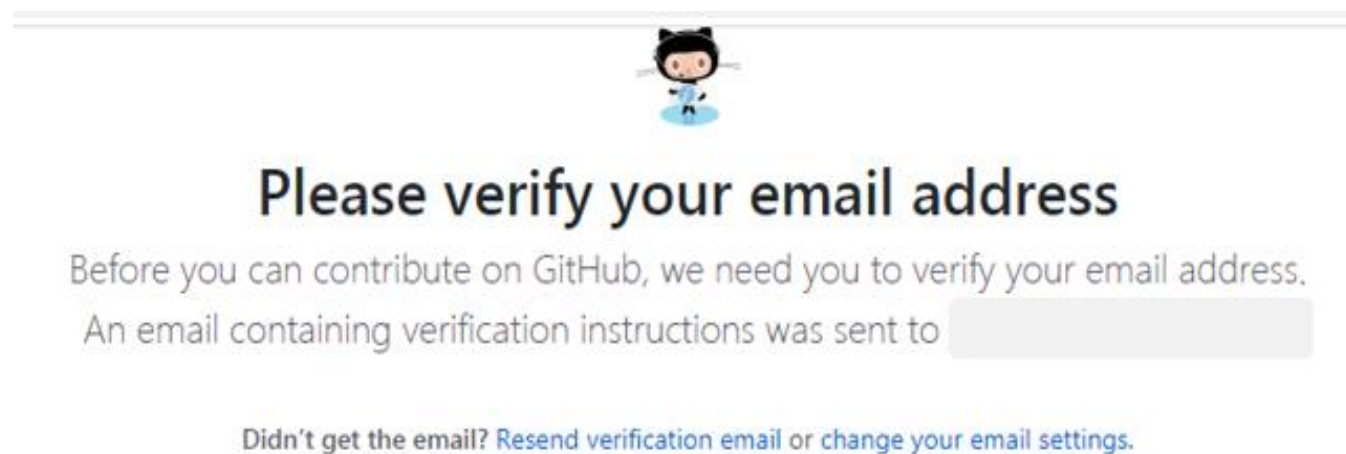


Рис. 6.7 Реєстрація на сайті GitHub

Дочекайтеся, коли прийде лист для підтвердження вашої поштової скриньки, підтвердіть кнопкою «Verify email address».

Almost done, [REDACTED]! To complete your GitHub sign up, we just need to verify your email address: [REDACTED]@ukr.net.

Verify email address

Once verified, you can start using all of GitHub's features to explore, build, and share projects.

Button not working? Paste the following link into your browser: [https://github.com/verify-email](#)

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

Рис. 6.8 Лист для підтвердження

Увійдіть в систему GitHub. Зайдіть в налаштування «Settings». Виберіть пункт «SSH and GPG keys». Натисніть «NewSSH key».

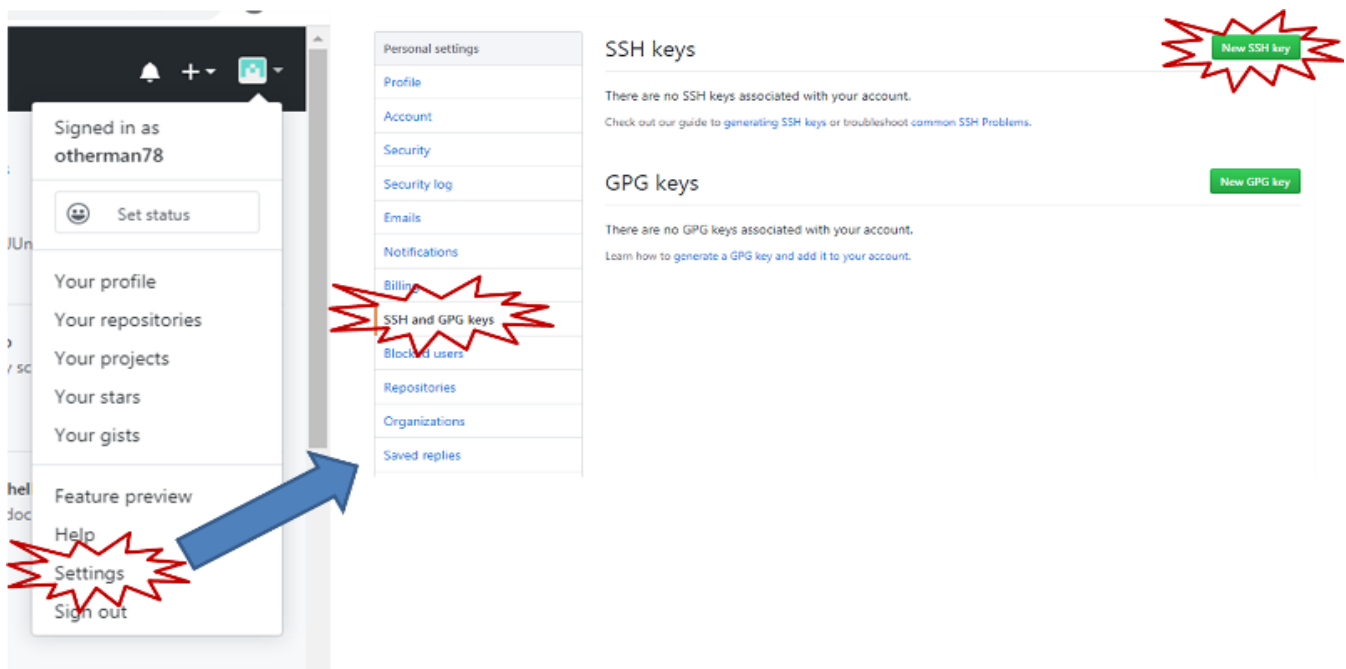


Рис. 6.9 Формування SSH на GitHub

У вікні що з'явиться необхідно ввести найменування та ключ SSH, який було створено в попередньому пункті. Для цього відкрийте файл з ключем (id_rsa.pub) скопіюйте весь зміст у вікно «key». Після цього натисніть «Add SSH key».

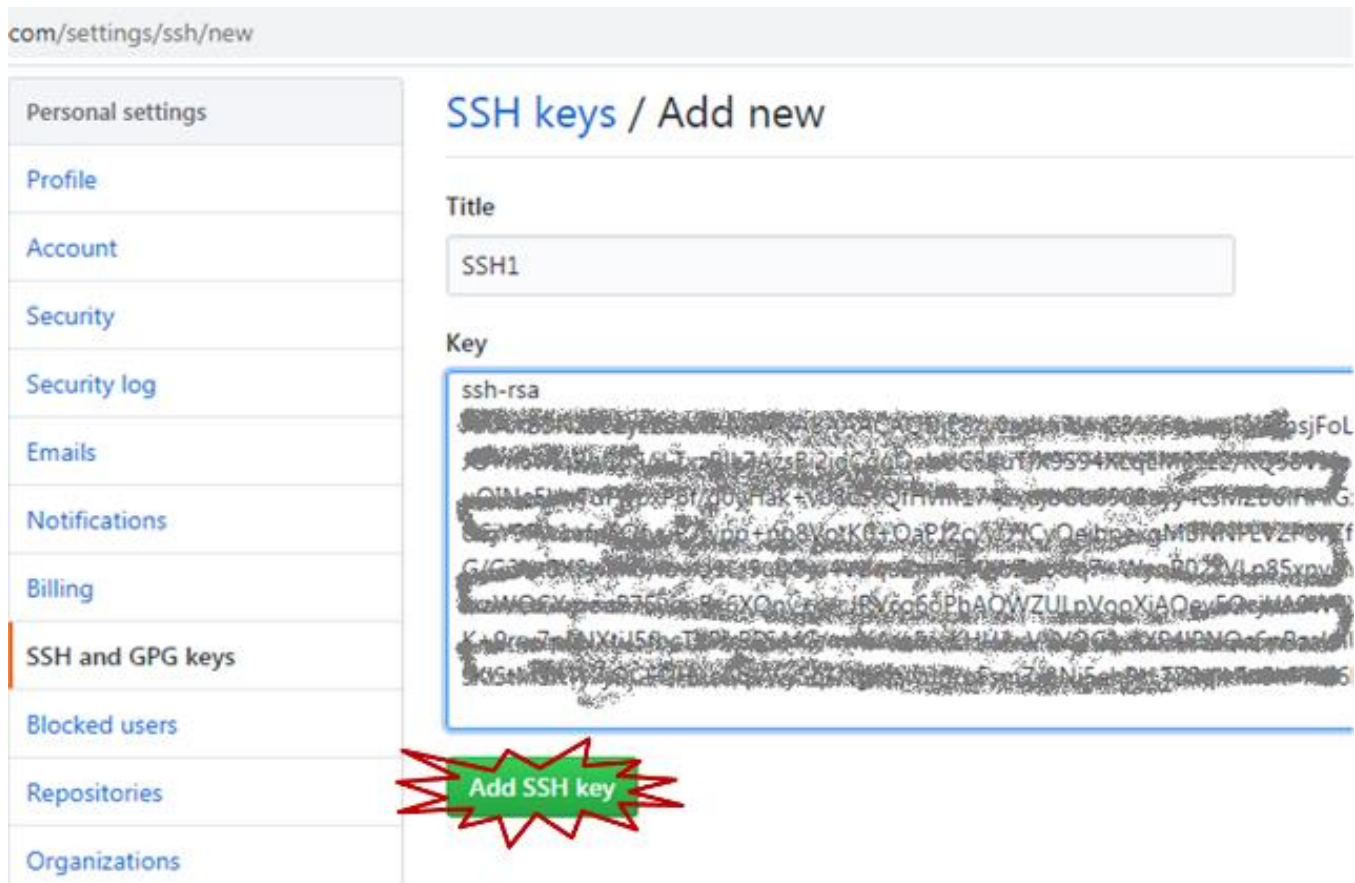


Рис. 6.10 Вікно для копіювання SSH

Після цього необхідно буде ввести користувача і пароль для GitHub. На пошту прийде повідомлення про додавання SSH ключа. Він також буде відображатися у списку ключів GitHub.

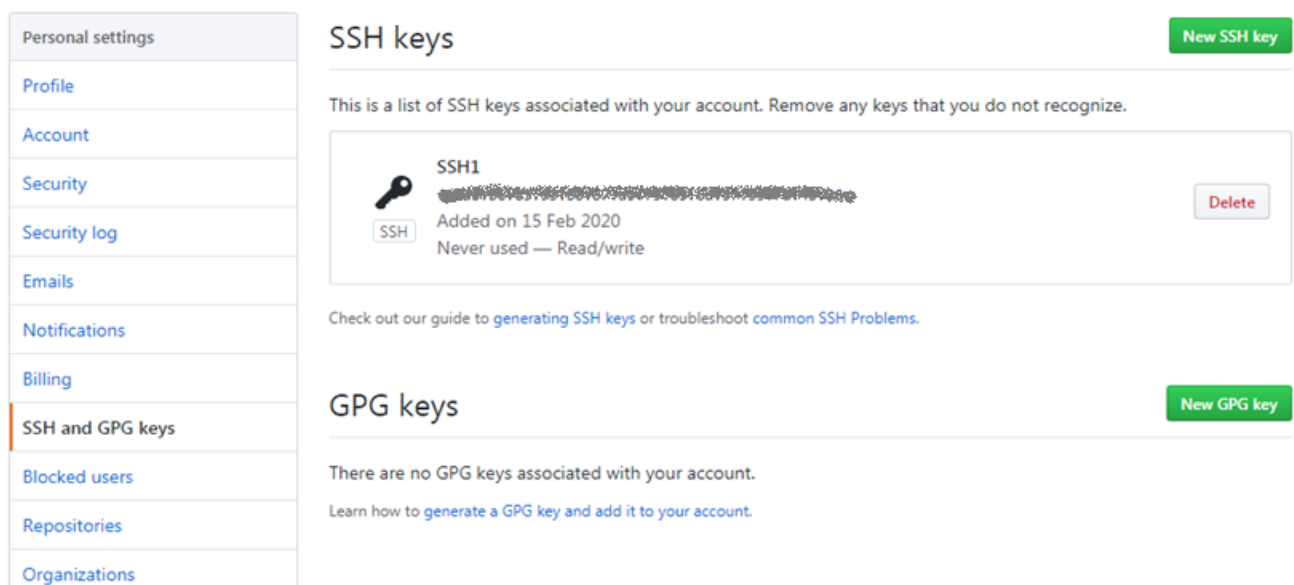


Рис. 6.11 Відображення списку ключів GitHub

На даному кроці в GitHub необхідно створити новий репозиторій, який буде використовуватися для збереження лабораторних робіт. Зайдіть в перелік репозиторіїв. Створіть новий репозиторій з назвою «LabsNodeRED», зробіть його приватним.

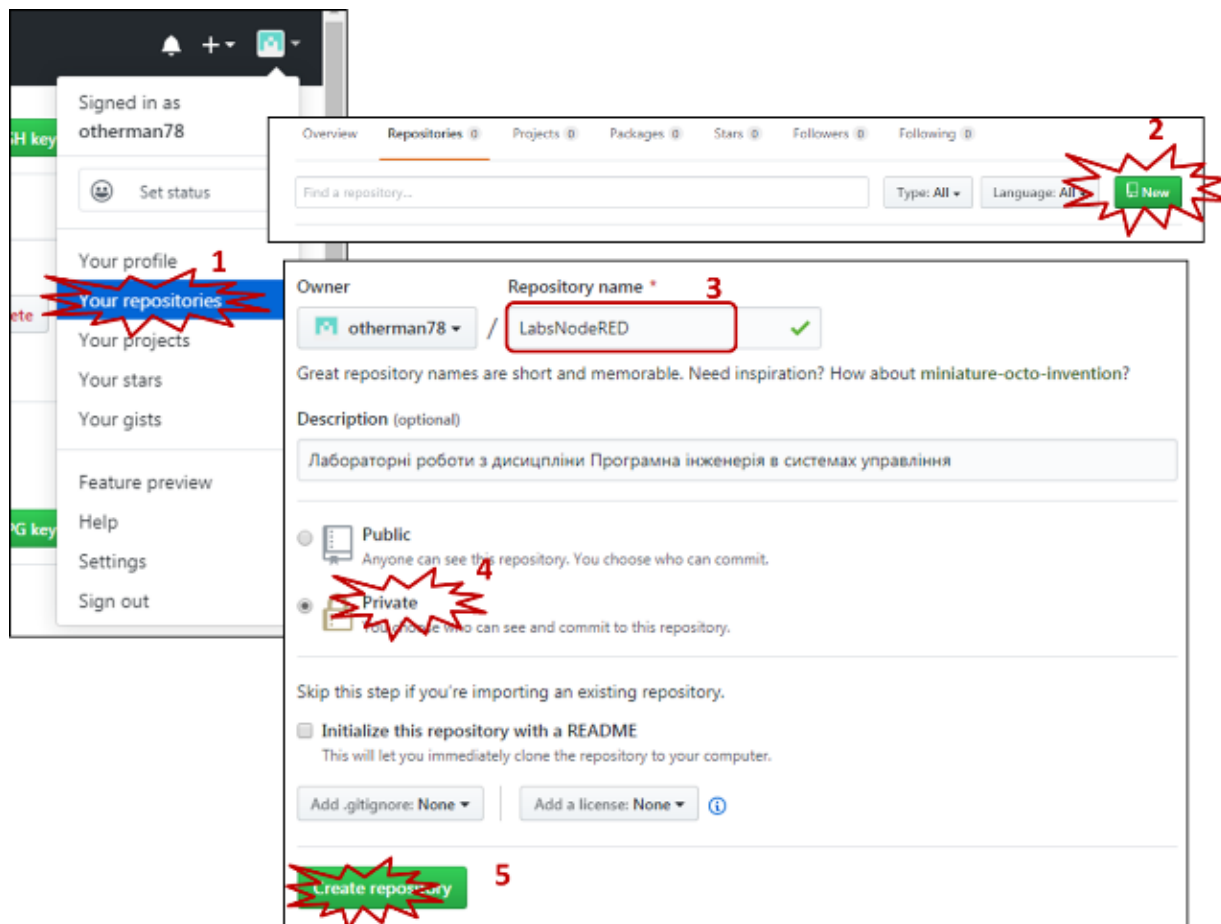


Рис. 6.12 Створення нового репозиторію в GitHub

Після створення відкриється сторінка налаштування репозиторію. Не закривайте сторінку, вона знадобиться в наступному пункті.

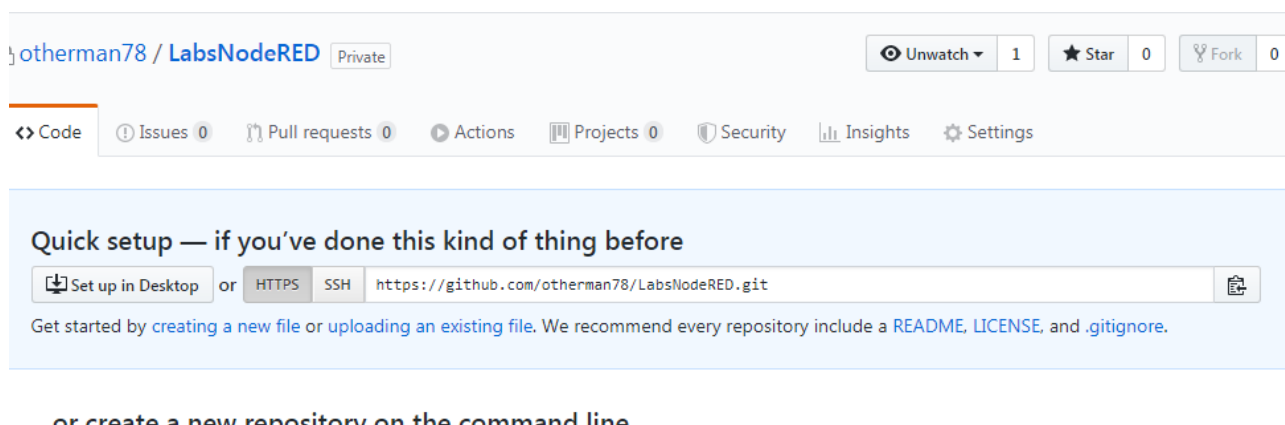


Рис. 6.13 Сторінка налаштування репозиторію

Налаштування з'єднання локального та віддаленого репозиторію. Завантаження файлів локального репозиторію на віддалений Зайдіть в локальну директорію проєкту Nod-RED. Запустіть Git Bash.

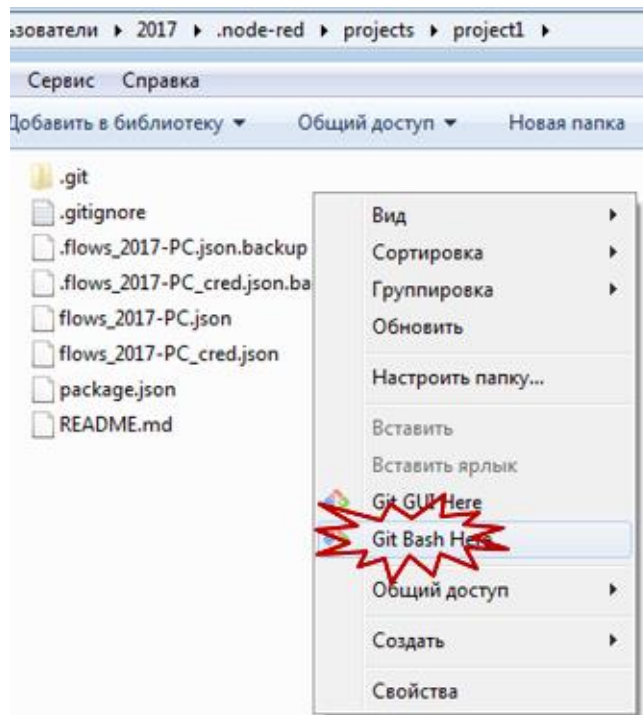


Рис. 6.14 Запуск Git Bash

У вікні <Code> репозиторію GitHub активуйте кнопку SSH і скопіюйте в буфер обміну команди, як показано на рисунку. Введіть пароль, який був вказаний при генеруванні SSH ключа.

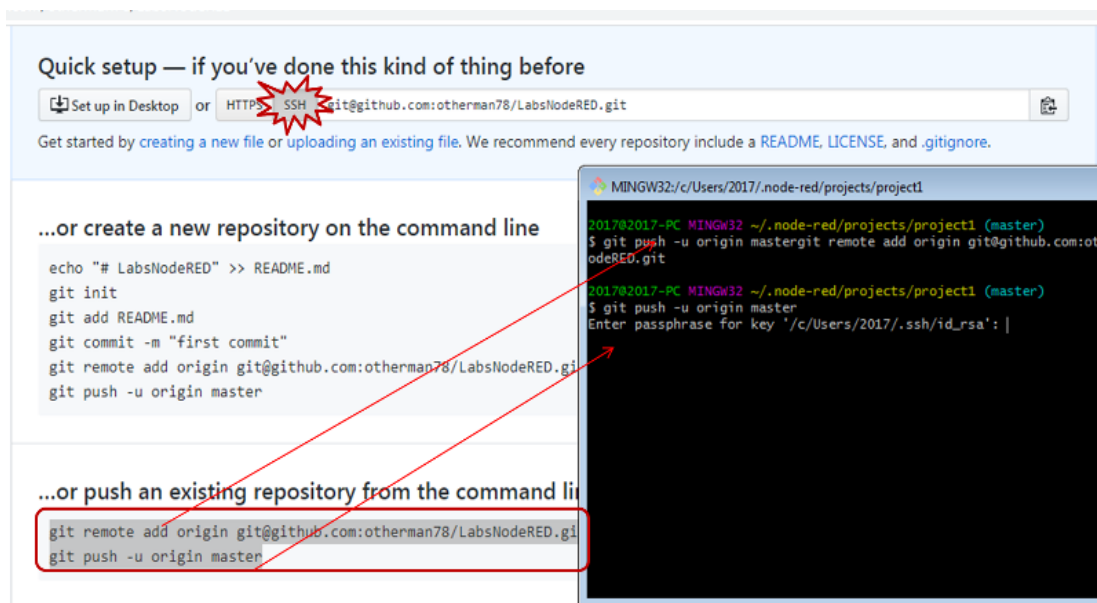


Рис. 6.15 З'єднання локального та віддаленого репозиторію та відправка коміту

Має з'явитися повідомлення, що файли локального репозиторію завантажені на віддалений репозиторій.

```
2017@2017-PC MINGW32 ~/.node-red/projects/project1 (master)
$ git push -u origin master
Enter passphrase for key '/c/Users/2017/.ssh/id_rsa':
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 2 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 14.76 KiB | 629.00 KiB/s, done.
Total 15 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), done.
To github.com:otherman78/LabsNodeRED.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Рис. 6.16 Повідомлення про результат завантаження

Відновіть сторінку <Code> репозиторію GitHub. Вона тепер повинна містити клон локального репозиторію «Node-RED». Передивіться вкладку «Branch» та «Commits».

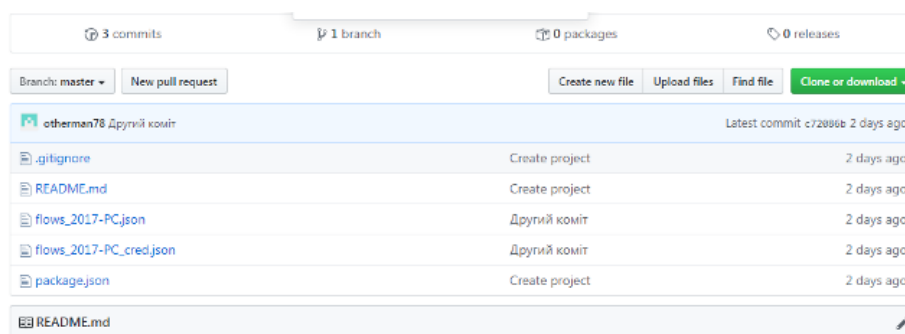


Рис. 6.17 Перегляд змісту репозиторію на GitHub

Команди push можна також робити через Node-RED. Запустіть Node-RED на локальній машині, якщо він не запущений. Відкрийте налаштування і перевірте, що ключ SSH видимий для Node-RED.

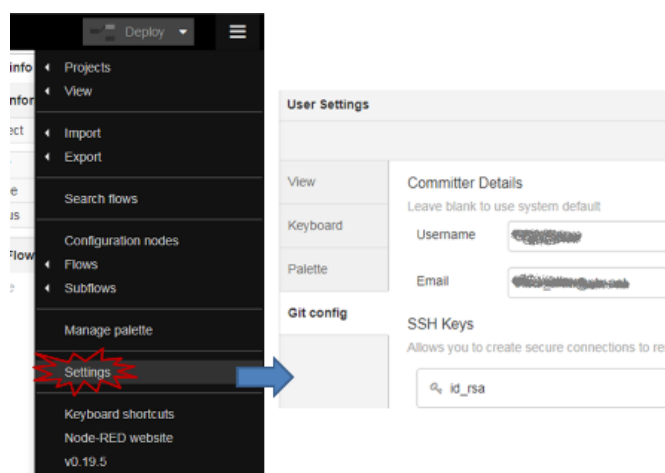


Рис. 6.18 Завантаження даних на GitHub з Node-RED

Завжди можна передивитися і скопіювати значення ключа, натиснувши по його назві. Зайдіть в налаштування проєкту, впевніться що підключення дійсно є для даного репозиторію.

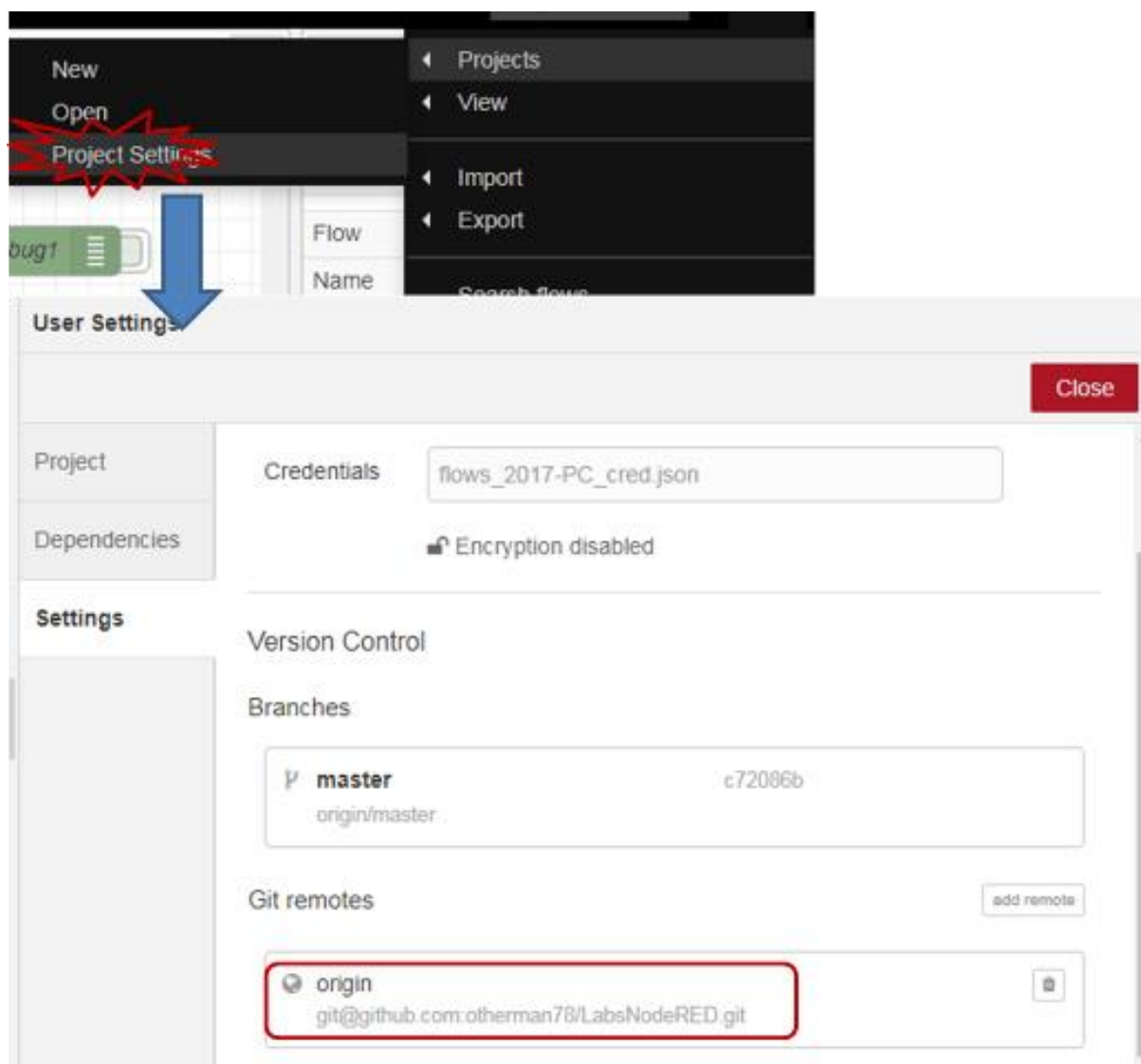


Рис. 6.19 Перевірка підключення локального до віддаленого репозиторію

Якщо його немає, то перезавантажте Node-RED і сторінку браузера розробки і спробуйте знову.

Змініть потік в Node-RED, наприклад перемістіть якийсь вузол. Зробіть розгортання, в Project History зробіть індексування, коміт, після чого перейдіть в «Commit History». Можна побачити в кутку, що є один коміт, який не запущений в віддалений репозиторій. Натисніть кнопку «Push».

Перший раз з'явиться екран для вводу паролю-фрази від SSH ключа, введіть її і натисніть кнопку «Retry».

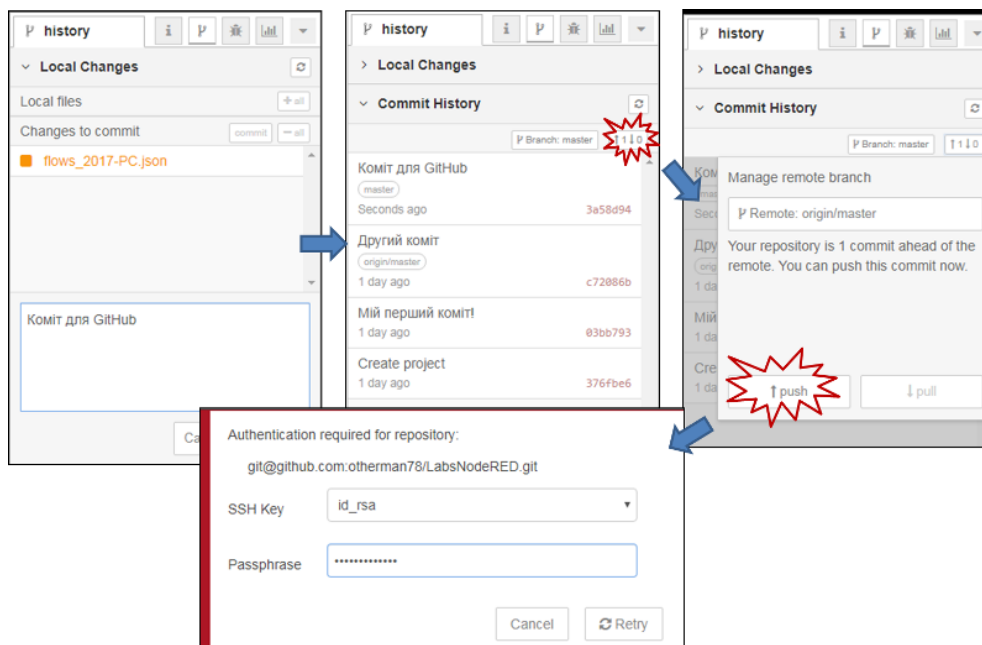


Рис. 6.20 Завантаження на віддалений репозиторій з Node-RED

Перейдіть на сторінку репозиторію GitHub. Обновіть сторінку. Можна побачити, що кількість комітів збільшилася на 1.

Натиснувши на кнопку перегляду останнього коміту, можна побачити усі файли, які в ньому є. А якщо натиснути кнопку по самому коміту, то можна побачити які саме зроблені зміни.

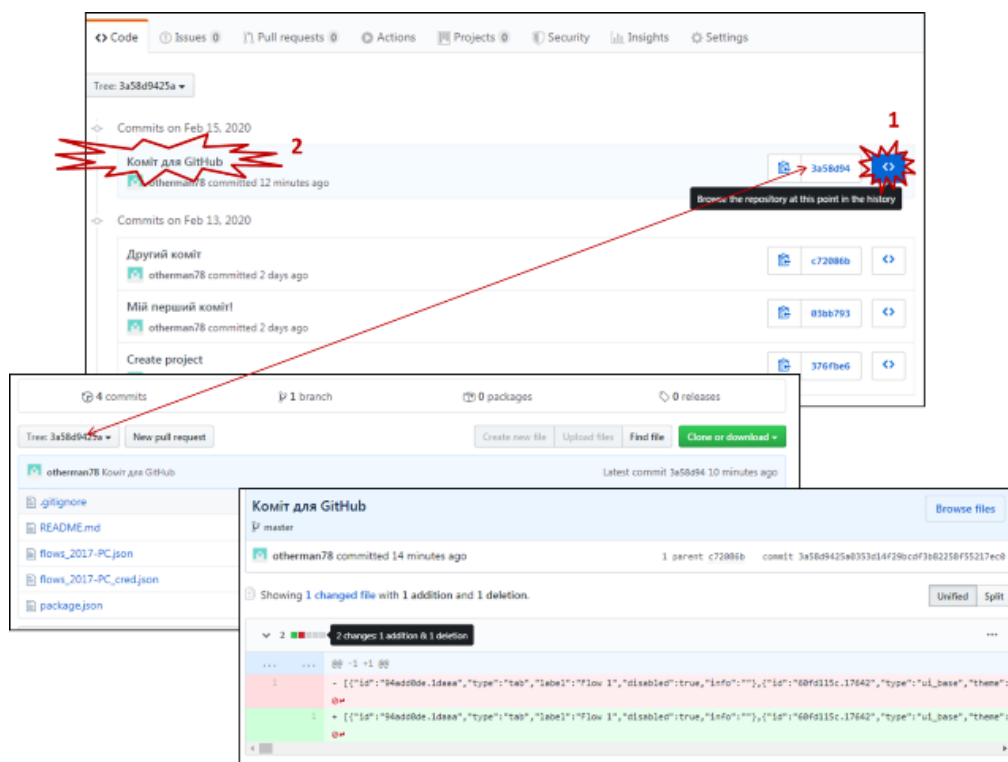


Рис. 6.21 Перевірка коміту в GitHub

Підключення до репозиторію GitHub інших користувачів для колаборативної роботи. Щоб це зробити, потрібно зайти в Setting->Collaborations. Додайте в команду проєкту викладача «pupenasan».

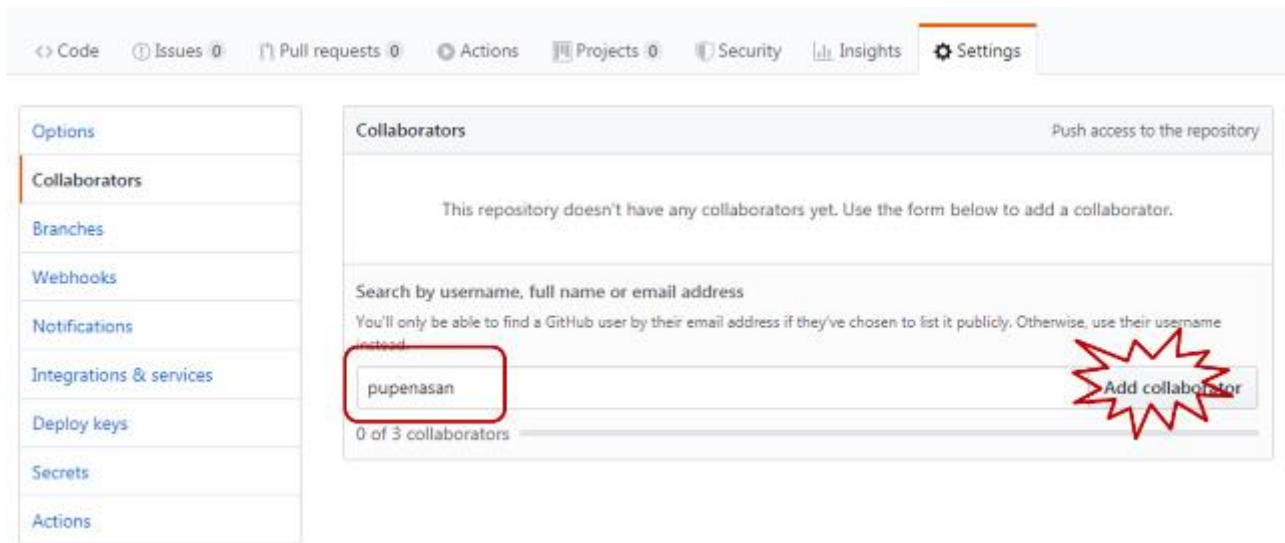


Рис. 6.22 Підключення до репозиторію викладача

Підключіть одного колегу до репозиторію.

6.3 Завдання

1. Встановити Git, якщо не встановлено, з вибором редактора Notepad++.
2. Запустити Git Bash у домашній директорії користувача.
3. Перевірити наявність SSH-ключів за допомогою `ls -al ~/.ssh`.
4. Створити новий SSH-ключ командою:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com".
```

5. Вказати passphrase (пароль-фразу) та підтвердити її.
6. Відкрити файл `id_rsa.pub` у текстовому редакторі, скопіювати публічний ключ.
7. Зареєструвати акаунт на GitHub або увійти в існуючий.
8. У GitHub перейти до Settings → SSH and GPG keys → New SSH key.
9. Ввести назву ключа, вставити значення і натиснути Add SSH key.
10. Створити новий приватний репозиторій на GitHub з назвою LabsNodeRED.
11. Скопіювати SSH-адресу репозиторію для подальшої роботи.
12. У директорії Node-RED запустити Git Bash.
13. Ініціалізувати локальний репозиторій (якщо не було зроблено).
14. Пов'язати локальний репозиторій із віддаленим за SSH:

```
git remote add origin git@github.com:your_username/LabsNodeRED.git
```

15. Виконати команди:

```
git add .
```

```
git commit -m "Initial commit"
```

```
git push -u origin master
```

16. У браузері оновити GitHub-репозиторій і переконатися, що файли з'явилися.
17. В Node-RED запустити проєкт і перевірити доступність SSH підключення.
18. Внести зміни у потік Node-RED, зробити commit та push через Project History.
19. Перевірити на GitHub появу нового коміту та його зміст.
20. Додати іншого користувача до репозиторію через Settings → Collaborators.
21. Додати користувача ripenasan до репозиторію як колаборатора.
22. Підготувати повний звіт з коментарями, командами та скріншотами.

6.4 Контрольні запитання

1. Що таке SSH-ключ і для чого він використовується у Git?
2. Яка різниця між приватним і публічним SSH-ключем?
3. Якою командою можна згенерувати SSH-ключ?
4. Що таке passphrase у контексті SSH-ключа?
5. Яким чином додати SSH-ключ до свого GitHub-акаунту?
6. Для чого використовувати SSH-ключі замість HTTPS?
7. Як перевірити, чи правильно доданий ключ у GitHub?
8. Як зв'язати локальний репозиторій із віддаленим через SSH?
9. Якими командами виконується завантаження (push) у віддалений репозиторій?
10. Які переваги інтеграції Git з Node-RED?
11. Як Node-RED працює з SSH-ключами?
12. Який спосіб перевірки того, що репозиторій успішно підключено через SSH?
13. Що означає commit та push у Git?
14. Як подивитися зміни у репозиторії через GitHub?
15. Як запросити іншого користувача до приватного репозиторію на GitHub?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДЖЕРЕЛ

1. Блага Н. В. Управління проєктами: навч. посібник. Львів: Львівський державний університет внутрішніх справ, 2021. 152 с.
2. Демиденко М. А. Сучасні методи управління проєктами інформатизації: навч. посібник. Дніпро: НТУ «Дніпровська політехніка», 2020. 163 с.
3. Довгань Л. Є, Мохонько Г. А., Малик І. П. Управління проєктами: навч. посібник. КПІ ім. Ігоря Сікорського, 2017. 420 с.
4. Катренко А.В. Управління ІТ-проєктами. Кн. 1. Стандарти, моделі та методи управління проєктами: підручник. Львів: «Новий Світ – 2000», 2011. 550 с.
5. Микитюк П.П. Управління проєктами: навч. посібник. Тернопіль, 2014. 270 с.
6. Кузьмініх В. О., Тараненко Р. А. Основи управління ІТ проєктами : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2019. 75 с.
7. Буріменко Ю. І., Галан Л. В., Лебедєва І. Ю., Щуровська А. Ю. Управління проєктами : навч. посібник Одеса : ОНАЗ ім. О. С. Попова, 2017. 235 с.
8. Березін О. В., Безпарточний М. Г. Управління проєктами : навч. посібник. Суми: Університетська книга, 2014. 272 с.
9. Ноздріна Л. В., Ящук В. І., Полотай О. І. Управління проєктами: підручник. за заг. ред. Л. В. Ноздріної. Київ : Центр учбової літератури, 2010. 432 с.
10. Филипенко О. М., Колеснік Т. С. Управління проєктами: навч. посібник. Харків: ХДУХТ, 2016. 161 с.
11. Офіційна сторінка GitHub. URL: <https://github.com/>