# Лабораторна робота № 2.
# Узгодження технічного завдання на розробку. Віхи проекту

*Мета лабораторної роботи.* Отримати практичні навички у створенні віх проекту. Сформувати технічне завдання на розробку системи контролю версій (лише функціональні вимоги). Ознайомитися з ролями архітектора проекту, тімліда та бізнес-аналітика. Набути практичних навичок використання функціональності "Projects" (проекти) на GitHub.

## Порядок виконання роботи:

1. Прочитати теоретичний матеріал до лабораторної роботи.
2. Продовжити роботу у команді з 3 чоловік.
3. Виділити в команді ролі (тімліда, програміста, тестувальника). Можна змінити ролі і тоді навести матрицю відповідальних.
4. Виділити базові функціональності системи, що розробляється, і оформити у вигляді технічного завдання (ТЗ) (варіанти узгодити з викладачем).
5. Розбити проект на віхи (етапи, milestones) відповідно до ТЗ і узгодити терміни з семестровими планом (у викладача).
6. Розбити проект на групи завдань і автономні завдання.
7. Для кожної ролі члена команди додати проект в Solution. Кожен проект повинен відповідати завданням, виділеним з ТЗ для кожної ролі учасника команди.

## Завдання за ролями у команді:

*Для ролі тімліда (основна роль на даному етапі):*

a. Створити на веб-хостингу репозиторій, який буде використаний у 2 – 5 лабораторних роботах.

b. Запросити членів команди до проекту, відправивши кожному invite на e-mail.

c. розпланувати завдання у відповідності до *Загального прикладу використання функціональності "Projects" на GitHub (п.п. 1-4).*

d. Створити Solution в VisualStudio (або у іншому обраному середовищі розроблення).

e. Додати в Solution проект Core, який буде містити загальну функціональність та інтерфейси для всього Солюшин.

f. Додати загальні інтерфейси взаємодії компонент.

g. Зафіксувати зміни на сервері Assembla (або будь-якому іншому сервері, що підримує роботу з СКВ).

*Для ролі програміста (допоміжна роль):*

a. Отримати робочу копію сховища на локальний комп'ютер.

b. Створити проект і додати його в репозиторій.

c. Створити прототип програми відповідно до описаної архітектури.

d. Реалізувати базові функції.

e. Зафіксувати зміни на сервері.

f. Змінити статус завдання: «Прийнято», «Отримано», «У процесі», «Завершено», «Помилка», «Необхідне узгодження».

*Для ролі тестувальника (допоміжна роль):*

a. Створити проект (unit-test) і додати його в репозиторій.

b. Зафіксувати зміни на сервері.

c. На основі розробленої специфікації проекту скласти майбутній план тестування.

d. Змінити статус завдання: «Прийнято», «Отримано», «У процесі», «Завершено», «Помилка», «Необхідне узгодження».

8. Оформити звіт, перевірити у викладача і захистити роботу.

9. Зберегти файл, який містить звіт з лабораторної роботи, в системі Mentor (mentor.khai.edu).

**Зміст звіту:**

1. Постановка завдання.
2. ТЗ проекту.
3. План завдань (запланованих робіт) проекту.
3. Віхи проекту (milestones).
4. Структура Солюшин з описом кожного проекту.
5. При необхідності – пояснення до реалізації.
6. Текст програми.
7. Скріншоти створеного сховища та історії змін.
8. Базовий план тестування з тестовими випадками.
9. Висновки.

## Теоретичні відомості

На GitHub є можливість планувати завдання. Це зазвичай відбувається за допомогою інструменту, що називається "Issue Tracker" (відстежувач проблем або завдань). Користувачі можуть створювати нові завдання, призначати їх собі або іншим учасникам проекту, призначати теги, надавати коментарі та використовувати інші інструменти для організації робочого процесу.

Також, для більш розгалужених та складних задач, є можливість використовувати функціональність "Projects" (проекти) на GitHub. Вона дозволяє створювати дошки для керування проектами, розміщувати задачі на цих дошках та встановлювати їх статус, наприклад, "в розробці", "завершено", "очікує перевірки" і так далі.

Загалом, GitHub надає різноманітні інструменти для планування, відстеження та управління завданнями, що допомагає забезпечити ефективну співпрацю в команді та ведення проектів.

**Загальний приклад використання функціональності "Projects" на GitHub для керування проектом:**

*Не плутати з віхами проєкту (milestones).*

*1. Створення проекту.*

У Вашому репозиторії перейдіть до вкладки "Projects" та натисніть кнопку "New project". Дайте проекту назву і, за необхідності, додайте опис.

*2. Створення колонок.*

Після створення проекту ви можете створити колонки для організації завдань. Натисніть кнопку "Add column" і додайте колонки відповідно до Вашого процесу розробки, наприклад: "To Do", "In Progress", "Review", "Done".

*3. Додавання карток.*

У кожній колонці Ви можете додавати картки, які представляють завдання. Картки можуть містити інформацію про завдання, таку як опис, мітки, відповідальних, теги тощо.

*4. Призначення завдань.*

Коли створено картку, Ви можете призначити її собі або іншим учасникам команди. Це дозволяє відстежувати, хто працює над кожним завданням.

*5. Рух карток по колонках.*

Після того, як завдання розпочато, Ви можете перетягувати картки між колонками відповідно до їх стану. Наприклад, коли завдання готово для перевірки, Ви можете перемістити його в колонку "Review".

*6. Використання міток і фільтрів.*

Ви можете додавати мітки до карток для категоризації завдань, наприклад, за пріоритетом або типом завдання. Це допомагає швидко фільтрувати та знаходити потрібні завдання.

Всі учасники команди можуть бачити та взаємодіяти з проектом на GitHub. Вони можуть додавати нові картки, коментувати завдання та переміщати їх по колонках.

Це лише загальний приклад використання функціональності "Projects" на GitHub. Кожен проект може мати власні унікальні потреби та процеси, тому Ви можете налаштувати колонки та карти відповідно до Вашого проекту.

**Опис завдань тімліда, розробника та тестувальника за допомогою функціональності "Projects" на GitHub:**

1. Створення проекту на GitHub.

Тімлід створює новий проект на GitHub для керування розробкою програмного забезпечення.

2. Створення колонок у проекті.

Тімлід створює колонки у проекті для різних етапів розробки, наприклад: "To Do", "In Progress", "Review", "Done".

3. Розподіл завдань.

Тімлід створює картки для кожного завдання та розподіляє їх між розробником та тестувальником з урахуванням їхніх навичок та можливостей.

*Завдання тімліда:*

- створення карток для розробників та тестувальників;
- призначення відповідальних осіб для кожної картки;
- встановлення термінів виконання для кожного завдання;
- відслідковування прогресу роботи розробників та тестувальників.

*Завдання розробника:*

- виконання завдань, призначених тімлідом;
- переміщення карток між колонками відповідно до стану завдань (наприклад, з "To Do" в "In Progress" після початку роботи над завданням);
- додавання коментарів про прогрес та будь-які проблеми, які виникають під час роботи.

*Завдання тестувальника:*

- виконання завдань, призначених тімлідом;
- перевірка розроблених функціональностей на відповідність вимогам;
- додавання коментарів про виявлені помилки або неузгодженості.

4. Реакція на зміни.

- відповідь на зміни вимог або проблеми, виниклі під час розробки або тестування;

- переміщення карток або створення нових для відповідного відображення змін у проекті.