**Підготовка до проєкту**

**Тімлід**

1. Створює **пустий публічний** репозиторій для проєкту **на своєму** GitHub-акаунті.
2. Клонує репозиторій <https://github.com/goitacademy/parcel-project-template> собі на машину із GitHub Desktop.
3. У GitHub Desktop переходить в налаштування клонованого репозиторію та змінює значення **remote** на url раніше створеного репозиторію свого проєкту вибравши HTTPS.
4. Відкриває проєкт у редакторі коду та змінює значення у файлі **package.json** згідно з інструкцією. Робить коміт та пушить зміни на **GitHub.**
5. Перевіряє роботу авто деплою.
6. Заходить до **«Settings»** репозиторію та вносить наступні зміни.

* У вкладці **Options** у секції **Merge button** ставить чек на **Automatically delete head branches.**
* У вкладці **Options** у секції **GitHub Pages** налаштовує хостинг проекту з гілки **gh-pages.**
* У вкладці **«Branches»** у розділі **«Branch protection rules»** встановлює правила для гілки **main.**
* На вкладці **Manage access** додає членів команди в репозиторій. Кожному буде надіслано інвайт на пошту.

**Розробники**

1. Встановлюють **LTS версію Node.js.**
2. Клонують репозиторій, створений тимлідом через **GitHub Desktop.**
3. У терміналі виконують команду **npm .**

Розробка фічі або багфікс

1. Кожна фіча або багфікс виконується окремою гілкою.
2. Перед початком роботи над новою фічею або багфіксом оновлюємо локальний **main**, щоб він був ідентичний **main на GitHub**. Для цього в **GitHub** **Desktop** переходимо на гілку **main** і робимо **fetch+pull** щоб синхронізувати локальну гілку з тією, що на **GitHub.**
3. Робимо нову гілку з **main.** Рекомендується давати назву гілки за шаблоном **feature/назва-фічі** та **bugfix/назва-бага**.
4. Працюємо у цій гілці та вносимо необхідні зміни. Комітимо зміни за потребою.
5. Після того, як фіча готова, відправляємо історію (комміти) локальної гілки на GitHub і відкриваємо **Pull request.**
6. Призначаємо мінімум одного **Reviewer,** щоб зробив ревью коду.
7. Якщо необхідні редагування, вносимо їх у тій самій гілці і відправляємо зміни на **GitHub.** Поточний пул-реквест оновиться автоматично, відкривати новий не треба.
8. Після того як пул-реквест пройшов ревю коду, **тімлід** приймає його (мерже у гілку **main**).
9. Після того, як пул-реквест був прийнятий, розробник, у **GitHub Desktop,** переходить на гілку **main** і робить **fetch+pull** щоб синхронізувати локальну гілку з тієї що на **GitHub.**
10. Після синхронізації гілки **main** розробник видаляє свою локальну гілку фічі/бага в GitHub Desktop.

Вирішення git-конфліктів

Відбуваються коли два розробники одночасно змінюють один і той же рядок в якомусь файлі. Виникають під час пул-реквесту, через що він не може бути успішно смержений у базову гілку. Краще дозволяти локально, а не GitHub.

1. У **GitHub Desktop** переходимо в гілку **main** і робимо **fetch + pull,** щоб синхронізувати її з гілкою на GitHub, в якій є всі зміни.
2. Переходимо у гілку фічі/багфікса і зливаємо (merge) до неї гілку **main.**
3. Вирішуємо конфлікт і робимо коміт.
4. Відправляємо зміни на GitHub, щоб оновити пул-реквест.