Правила работы с репозиторием проекта "BookHub"

Этот документ описывает основные правила и процессы, принятые в команде "Fourth energoblock"для работы с данным репозиторием. Соблюдение этих правил обязательно для всех участников проекта.

1 Общий процесс внесения изменений

Любые изменения — будь то добавление документации, исправление ошибок или любая другая задача — должны вноситься через **Pull Request (PR)**. Прямые коммиты в ветку `main` **запрещены** настройками репозитория.

Порядок действий для каждого участника:

1. **Синхронизация:** Перед началом любой работы убедитесь, что ваша локальная ветка `main` содержит самые последние изменения с GitHub.

```
git checkout main
git pull origin main
```

2. **Создание новой ветки:** Для каждой новой задачи создавайте отдельную ветку от `main`. Название ветки должно соответствовать правилам именования (см. раздел 2).

```
git checkout -b <nazvanie-vashej-vetki>
```

3. **Работа над задачей:** Внесите необходимые изменения в файлы проекта.

4. **Коммиты:** Сделайте один или несколько коммитов, чтобы сохранить ваши изменения. Сообщения коммитов должны соответствовать принятому стилю (см. раздел 3).

```
git add .
git commit -m "tip: kratkoe opisanie vashih izmenenij"
```

5. **Отправка на GitHub:** Отправьте вашу ветку с коммитами в удаленный репозиторий.

```
git push origin <nazvanie-vashej-vetki>
```

6. **Создание Pull Request:** На сайте GitHub появится предложение создать Pull Request из вашей ветки. Нажмите на него.
   - **Целевая ветка:** `main`.
   - **Название PR:** Кратко и понятно опишите суть всех изменений.
   - **Reviewers:** В правой панели назначьте всех остальных участников команды для проверки вашего кода.

2 Правила именования веток

Название ветки должно давать понять, над какой задачей ведется работа. Мы используем следующие префиксы:

- `feature/` — для создания новых артефактов или добавления крупных блоков информации (например, `feature/create-questionnaire`).
- `docs/` — для обновления документации (например, `docs/update-contribution-guide`).
- `fix/` — для исправления ошибок или опечаток (например, `fix/correct-typos-in-readme`).

Пример правильного названия: `feature/add-communication-plan`

3 Стиль именования коммитов

Мы придерживаемся стандарта **Conventional Commits**. Это помогает нам поддерживать историю изменений чистой и понятной, а также в будущем позволит автоматизировать создание отчетов об изменениях.

Формат сообщения: `<тип>: <краткое описание>`

Основные типы:

- **feat**: Добавление нового файла-артефакта или раздела в документ.
- **docs**: Изменения, касающиеся только документации (например, обновление `README.md` или этого файла).
- **fix**: Исправление опечаток, ошибок в форматировании.
- **style**: Изменения, не влияющие на содержание (пробелы, форматирование и т.д.).

Примеры сообщений:

- `feat: Add initial draft of meeting protocol`
- `docs: Describe branch naming rules`
- `fix: Correct contact info in communication plan`

4 Система одобрений (Approve'ов)

Для обеспечения высокого качества нашей работы и общего понимания процесса, каждое изменение должно быть проверено командой.

- **Правило:** Для слияния Pull Request в ветку `main` необходимо получить **одобрение (Approve)** от всех остальных участников команды.
- **Расчет:** Так как в нашей команде 5 человек, для слияния PR необходимо получить **4 одобрения**.
- **Ограничение:** Автор Pull Request'а не может одобрить собственный PR.

Процесс ревью:

1. После создания PR автор запрашивает ревью у всех членов команды.
2. Ревьюеры просматривают изменения, оставляют комментарии, если есть вопросы или замечания.
3. Если изменения соответствуют требованиям, ревьюер ставит "Approve".
4. После получения 4-х одобрений PR может быть успешно слит в `main`.