**Процесс** **взаимодействия** **веток**

После того как проект был создан и проинициализирован, идёт следующее:

1. Переходим в ветку **main** если мы ещё это не сделали. Git checkout main
2. Создаём ветку для разработки (**develop**):

git checkout –b develop (создали ветку и перешли на неё)

git push origin develop (отправили данную ветку на удалённый сервер)

1. Создание веток для фронтенда и бэкенда.

Чтобы структура работы между фронтендом и бэкендом не препятствовала друг другу, требуется создать отдельные ветки, которые будут дочерней веткой для develop.

* 1. Ветка для работы над серверной частью:

git checkout –b backend/develop

git push origin develop

* 1. Ветка для работы с клиентской частью:

git checkout –b frontend/develop

git push origin frontend/develop

Теперь у нас есть три основные ветки:

* main – для релизов (stable code).
* develop – общая ветка для интеграции новых фич и работы над проектом.
* backend/develop и frontend/develop – для разделения фронтенд и бэкенд разработки.

1. Работа с ветками

Когда начинается разработка конкретного функционала, например, авторизации на фронтенде или создания API на бэкенде, каждая фича должна разрабатывать в отдельной ветке. Это позволяет изолировать изменения и упрощает управление.

* 1. фронтенд (React) – добавления функциональности:

Для каждой новой фичи, связанной с фронтендом, требуется создавать ветку от frontend/develop

git checkout frontend/develop

git checkout -b frontend/feature/authentication

git push origin frontend/feature/authentication

* 1. бэкенд (Django) – добавление функциональности:

Для каждой новой фичи на бэкенде, создавай ветку от backend/develop

git checkout backend/develop

git checkout -b backend/feature/user-api

git push origin backend/feature/user-api

Когда фича готова и протестирована, ты создаёшь Pull Request(PR) для слияния ветки frontend/feature/{feature\_name} или backend/feature/{feature\_name} в соответствующую ветку frontend/develop или backend/develop.

1. Интеграция изменений
   1. После того как работа над фичами закончена, ветки frontend/develop и backend/develop сливаются в общую ветку develop:

git checkout develop

git merge frontend/develop

git merge backend/develop

git push origin develop

* 1. Когда проект готов к релизу, ты сливаешь develop в main, таким образом все изменения попадают в стабильную ветку:

git checkout main

git merge develop

git push origin main

* 1. Так же после проверки и слияние веток в главную ветку, дочернюю ветку по типу: backend/feature/user-api требуется удалить.

git branch -d backend/feature/user-api

git push origin --delete backend/feature/user-api

1. Работа с багами и релизами
   1. Исправление багов:

Если нужно срочно исправить баг, требуется создать ветку от main:

git checkout main

git checkout -b hotfix/fix-login-issue

git push origin hotfix/fix-login-issue

* 1. Подготовка релизов:

Когда проект готов к выпуску новой версии, требуется создать ветку для релиза:

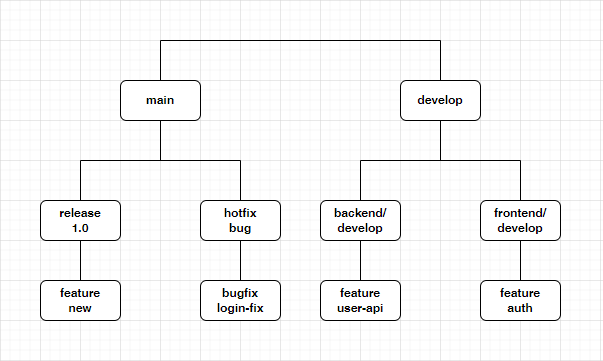
git checkout develop

git checkout -b release/1.0

git push origin release/1.0

После тестирования и всех правок, ветка release/1.0 сливается как в main, так и в develop.

**Схема взаимодействия веток**



* main — стабильная версия для продакшн кода.
* develop — основная ветка для разработки.
* backend/develop и frontend/develop — для разработки каждой части приложения.
* feature/{название} — ветки для разработки новых фич, создаются от веток backend/develop или frontend/develop.
* release/{версия} — ветки для подготовки релиза, создаются от develop.
* hotfix/{название} — ветки для исправления критических ошибок на продакшне, создаются от main.

**Полный гайд от начала до конца**

1. Инициализируйте ветки develop, backend/develop и frontend/develop как описано выше.
2. При разработке фич — создавайте ветки от frontend/develop или backend/develop.
3. После завершения фичи — создавайте PR для слияния изменений в ветки frontend/develop или backend/develop.
4. Объединяйте все изменения в ветке develop для интеграции.
5. Когда готов релиз — сливайте ветку develop в main.
6. Для багфиксов и срочных изменений — используйте ветки hotfix.