Лабораторная работа 5.

Распределение ролей

Цель работы: Приобрести практические навыки организации командной работы над проектом, распределения ролей в команде, настройки и использования системы контроля версий (на примере Git) для совместной разработки.

Задачи:

Сформировать команду и распределить роли в соответствии с методологией разработки.

Настроить локальное и удаленное рабочее окружение (Git, удаленный репозиторий на GitHub/GitLab).

Отработать основные workflow-процессы командной работы: создание веток, код-ревью, мерж-реквесты (pull requests), разрешение конфликтов.

Составить документ, регламентирующий процесс работы команды.

Ход работы:

Часть 1: Подготовительный этап (Теоретическая часть и планирование)

Формирование команды. Студенты объединяются в команды по 3-4 человека.

Выбор и описание проекта. Каждая команда выбирает небольшой учебный проект (например, простой веб-сайт-визитка, консольный калькулятор с расширенными функциями, простой API на Python/Java). Кратко описывается функционал проекта (3-5 пунктов требований).

Распределение ролей. Команда распределяет между собой следующие роли:

Тимлид (Team Lead): Координатор работы, принимает финальные решения по мержу кода, проводит код-ревью.

Разработчик #1: Отвечает за реализацию определенного модуля/функционала.

Разработчик #2: Отвечает за реализацию другого модуля/функционала.

Тестировщик/Технический писатель: Составляет тест-кейсы, проверяет работоспособность собранного приложения, ведет CHANGELOG (файл с историей изменений).

Определение workflow. Команда выбирает модель ветвления (например, GitFlow или упрощенную модель с ветками main, develop и feature). Решает, кто и при каких условиях создает мерж-реквесты.

Часть 2: Практический этап (Настройка и работа с Git)

Инициализация репозитория.

Тимлид создает новый пустой публичный репозиторий на GitHub/GitLab.

Добавляет в репозиторий файлы README.md (описание проекта) и .gitignore (шаблон для выбранного языка программирования).

Все участники команды делают git clone этого репозитория на свои локальные машины.

Настройка окружения и первое коммит.

Каждый разработчик настраивает глобальные параметры Git (имя, email) на своей машине.

Тимлид создает ветку develop от main и пушит ее на сервер.

Все участники переключаются на ветку develop (git checkout develop).

Реализация функционала в feature-ветках.

Каждый разработчик (включая тимлида, если он тоже пишет код) создает свою feature-ветку от develop для своей задачи (например, git checkout -b feature/add-auth-system).

Разработчики реализуют свой функционал, делая коммиты в своих ветках. Требование: коммиты должны быть небольшими и логически завершенными, с понятными сообщениями (например, "Add user login method", "Fix validation bug in email field").

Разработчики регулярно пушат свои ветки на удаленный сервер (git push origin feature/...).

Код-ревью и мерж.

Когда функционал в feature-ветке готов, разработчик создает Pull Request (MR) из своей ветки в ветку develop.

Тимлид и/или другой разработчик проводят код-ревью: проверяют код, оставляют комментарии.

Разработчик вносит правки по результатам ревью, делает дополнительные коммиты в свою feature-ветку.

После одобрения тимлида, код мержится в develop.

Задача тестировщика: после мержа сделать git pull origin develop, проверить, что собранное приложение работает корректно. Зафиксировать изменения в CHANGELOG.md.

Моделирование конфликта.

Преподаватель инициирует ситуацию, когда двое разработчиков изменяют один и тот же файл в разных feature-ветках.

Один из них успешно мержит свою ветку в develop.

Второй разработчик должен перед мержем сделать git rebase develop (или git merge develop) в своей ветке, разрешить конфликт вручную, и только затем создать Pull Request.

Часть 3: Анализ и отчетность

Составление отчета. Команда совместно готовит краткий отчет в формате Markdown в своем репозитории в файле LAB\_REPORT.md.