

## **Лабораторная работа №4**

### **Разработка и интеграция модулей проекта (командная работа)**

#### **1 Цель работы**

1.1 Получить навыки совместной разработки с использованием системы контроля версий.

1.2 Получить навыки интеграции компонентов ПО при командной разработке.

#### **2 Литература**

2.1 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). — URL: <https://znanium.com/catalog/product/1794453> . — Режим доступа: по подписке. — Текст : электронный. — гл.8.

#### **3 Подготовка к работе**

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

#### **4 Основное оборудование**

4.1 Персональный компьютер.

#### **5 Задание**

Работа выполняется в команде из 2-4 человек. Каждый участник будет отвечать за свою часть проекта, например:

Вариант 1:

- участник 1: создание модели данных и настройка БД,
- участник 2: реализация CRUD-операций в сервисе,
- участник 3: разработка логики фильтрации, сортировки, пагинации,
- участник 4: настройка логирования и исключений.

Вариант 2:

- участник 1: создание, обновление, удаление сущности,
- участник 2: чтение сущности,
- участник 3: настройка GUI.

Критерии выполнения задания:

- код успешно протестирован и собран в ветке main,
- все участники выполнили задачи и освоили работу с Git на практике,
- создана чистая история коммитов, PR корректно объединены и задокументированы,

- финальная версия с пометкой v1.0 или v1.1 готова и зафиксирована в репозитории,
- подготовлен отчет с описанием выполненных задач и примерами кода каждого участника и выводами по командной работе. В отчете должны быть скриншоты/команды всего процесса работы с git.

## 5.1 Разработка функциональности

5.1.1 Перед началом новой задачи синхронизироваться с основной веткой.

В основной ветке получить последние изменения:

5.1.2 Создать ветку для фичи.

Каждый участник выбирает функциональность и создает свою ветку:

```
git checkout -b feature/add-entity
```

Примеры имён для задач:

- feature/add-user — для новой функциональности.
- bugfix/fix-login — для исправления бага.

5.1.3 Разработать код и выполнить коммиты.

Разработать функциональность и зафиксировать изменения с осмысленными сообщениями. Сообщение должно содержать:

- краткое описание изменений (50 символов или меньше).
- тело сообщения для более сложных изменений (необязательно).

5.1.4 Запустить ветку в удалённый репозиторий.

Регулярно отправлять изменения, чтобы отслеживать прогресс:

```
git push origin feature/add-entity
```

## 5.2 Создание Pull Request (PR) и отправка на ревью

5.2.1 Создать Pull Request для своей фичи.

После завершения работы над фичей и коммита всех изменений, отправить свою ветку на удалённый репозиторий:

```
git push origin <название_текущей_ветки>
```

На платформе (например, GitHub, GitLab) выбрать свою ветку и создать новый Pull Request в основную ветку (main).

5.2.2 Оформить PR с описанием изменений.

Добавить подробное описание, включающее:

- краткое резюме — что реализовано и почему.
- подробности — какие изменения в коде, какие файлы затронуты.
- инструкции по тестированию — как проверить работоспособность изменений.

Выбрать одного из участников команды для ревью и назначить его в качестве ревьюера PR.

5.2.3 Убедиться, что PR готов к проверке.

Проверить свой код ещё раз, чтобы избежать очевидных ошибок и соответствовать стандартам проекта.

Использовать функцию Draft PR, если PR ещё не завершён, и переместить его в Ready for Review, когда он будет готов к проверке.

### 5.3 Получение отзыва, исправление замечаний и обновление Pull Request

#### 5.3.1 Получить отзыв от ревьюера.

Ревьюер оставляет свои комментарии, предложения или обнаруженные ошибки в PR. Он может выделить конкретные строки или дать общий фидбэк.

Комментарии могут включать:

- предложения по улучшению логики или читаемости кода.
- замечания по стилю и формату кода (например, форматирование или именование).
- обнаруженные ошибки или случаи, когда код может привести к багам.

#### 5.3.2 Ответить на комментарии и обсудить предложения.

Просмотреть все замечания и предложения. Если что-то неясно, ответить ревьюеру в комментариях к PR, чтобы уточнить его ожидания или обсудить альтернативные подходы.

Подумать над каждой рекомендацией. Даже если замечание кажется необязательным, оно может улучшить качество кода.

#### 5.3.3 Внести необходимые исправления в код.

Переключиться на свою ветку и внести исправления, соответствующие замечаниям. Зафиксировать каждое изменение с осмысленным сообщением, чтобы оно было понятно ревьюеру:

Запустить изменения в ветку, связанную с PR. Платформа автоматически обновит PR, отразив последние изменения:

```
git push origin <название_текущей_ветки>
```

#### 5.3.4 Отметить исправленные комментарии как решённые.

После внесения исправлений вернуться к PR и отметить исправленные комментарии как Resolved (Решённые).

#### 5.3.5 Дождаться повторного ревью.

Ревьюер проверяет изменения, чтобы убедиться, что все замечания учтены. В зависимости от сложности исправлений, ревью может пройти в несколько этапов.

Если не останется нерешённых комментариев и PR будет полностью удовлетворять требованиям, ревьюер должен отметить его как Approved (Одобен).

#### 5.3.6 Сливание Pull Request после одобрения.

После одобрения команда сливает PR в основную ветку. На момент слияния PR не должен содержать конфликтов с основной веткой.

Если конфликты есть, сначала разрешить их (например, вручную), затем завершить ребеиз:

- закоммитить изменения
- завершить слияние, выбрав Merge PR на платформе.

#### 5.3.7 Удаление ветки после слияния.

Удалить feature-ветку локально и на сервере, чтобы репозиторий оставался чистым:

#### 5.4 Подготовка релиза и финальная версия

##### 5.4.1 Создать ветку для подготовки релиза.

После завершения всех задач и тестирования создать ветку release для подготовки релиза:

```
git checkout -b release
```

##### 5.4.2 Обновить версию проекта и README.md.

Обновить версию в файлах (например, AssemblyInfo.cs или appsettings.json) и указать новую версию в README:

##### 5.4.3 Закоммитить изменения и создать тег для релиза.

Закоммитить обновления и добавить тег с номером версии:

##### 5.4.4 Создать Pull Request и финализировать релиз.

Создать PR для ветки release в main и обсудить его с командой. После одобрения провести слияние.

##### 5.4.5 Создать релиз на платформе (GitHub, GitLab).

Зайти в раздел Releases и создать новый релиз, привязав его к тегу v1.0.0, с описанием нововведений.

## 6 Порядок выполнения работы

6.1 Выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

## 7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

## 8 Контрольные вопросы

8.1 Какие основные шаги необходимо выполнить при начальной настройке проекта в Git?

8.2 Какие обязательные разделы должны быть включены в базовый файл README.md проекта?

8.3 Как правильно организовать процесс разработки новой функциональности с использованием ветвления в Git?

8.4 Какая информация должна быть указана в описании Pull Request для эффективного код-ревью?

8.5 Как должен действовать разработчик при получении замечаний в процессе код-ревью?

8.6 Каковы основные этапы подготовки релизной версии проекта?.