Министерство образования Российской Федерации

Тверской государственный технический университет

Кафедра Программного обеспечения

Программа для работы с личными делами студентов

Использование системы контроля версий в процессе разработки ПО

Выполнил: Бакеев А.В.

Проверил: Мальков А.А

Тверь, 2024

Оглавление

[Введение 3](#_Toc178678338)

[Постановка задачи 3](#_Toc178678339)

[Построенная диаграмма 4](#_Toc178678340)

[Код для Mermaid 4](#_Toc178678341)

# Введение

В рамках данной лабораторной работы уделим внимание построению UML-диаграмм, которые отражают ключевые аспекты модели проекта, позволяя простым и наглядным образом представить взаимодействия компонентов системы, их поведение и организацию. Основными целями работы являются визуализация описанного в предыдущей работе функционала, с помощью UML-диаграмм, таких как диаграммы классов, диаграммы взаимодействия и диаграммы состояний, а также применение полученных знаний на практике для создания собственной модели проекта.

# Постановка задачи

**Вариант 2.** Разработать программный модуль «Личные дела студентов». Программный модуль предназначен для получения сведений о студентах сотрудниками деканата, профкома и отдела кадров. Сведения должны храниться в течение всего срока обучения студентов и использоваться при составлении справок и отчетов.

1. Если система контроля версий Git не установлена, то установить ее (параметры оставить по умолчанию).
2. Запустить Git GUI (или консоль). Создать новый репозиторий. Добавить в папку репозитория файлы. Зафиксировать состояние репозитория (выполнить commit).
3. Внести изменения в файлы. Зафиксировать новое состояние репозитория.
4. Создать новую ветку 1. Внести в нее изменения и зафиксировать их.
5. Переключиться на ветку мастера. Внести в нее изменения и зафиксировать их.
6. Продемонстрировать слияние веток.
7. Просмотреть дерево изменений веток (историю).
8. Создать удаленный репозиторий.
9. Загрузить на него свой проект.
10. Обеспечить доступ к нему все членам команды.
11. Скачать каждым членов команды файлы с удаленного репозитория, произвести изменения и совершить слияние.

# Установка git

Открываем терминал и пишем brew install git

macbook@MacBook-Pro-macbook ~ % brew install git

==> **Fetching git**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/4315a95**

######################################################################### 100.0%

==> **Downloading https://mirrors.edge.kernel.org/pub/software/scm/git/git-htmldoc**

######################################################################### 100.0%

==> **Downloading https://mirrors.edge.kernel.org/pub/software/scm/git/git-manpage**

######################################################################### 100.0%

==> **Downloading https://cpan.metacpan.org/authors/id/R/RJ/RJBS/Net-SMTP-SSL-1.04**

######################################################################### 100.0%

==> **Downloading https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.47.0.**

######################################################################### 100.0%

…

==> **Running `brew cleanup git`...**

Disable this behaviour by setting HOMEBREW\_NO\_INSTALL\_CLEANUP.

Hide these hints with HOMEBREW\_NO\_ENV\_HINTS (see `man brew`).

Removing: /Users/macbook/Library/Caches/Homebrew/git--Net::SMTP::SSL--1.04.tar.gz... (2.4KB)

==> **Caveats**

==> **git**

The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.

Subversion interoperability (git-svn) is now in the `git-svn` formula.

# Создание локального репозитория и коммит

Настройка репозитория:

С помощью cd путь перемещаемся в папку, где хотим сделать локальный репозиторий и пишем git init

macbook@MacBook-Pro-macbook Мальков % cd "Лабораторные работы"

macbook@MacBook-Pro-macbook Лабораторные работы % git init

Initialized empty Git repository in /Users/macbook/Desktop/4 курс/Мальков/Лабораторные работы/.git/

macbook@MacBook-Pro-macbook Лабораторные работы %

Первый коммит:

Пишем git add. Для фиксации текущий изменений, а затем git commit -m для непосредственного коммита

macbook@MacBook-Pro-macbook Лабораторные работы % git add .

macbook@MacBook-Pro-macbook Лабораторные работы % git commit -m "Initial commit"

[main (root-commit) 07d8e6f] Initial commit

2 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 "\320\222\320\260\321\200\320\270\320\260\320\275\321\202 2 \320\233\320\260\320\261 1.docx"

create mode 100644 "\320\222\320\260\321\200\320\270\320\260\320\275\321\202 2 \320\233\320\260\320\261 2.docx"

macbook@MacBook-Pro-macbook Лабораторные работы %

# Новые ветки и их слияние

Создаем новую ветку командой macbook@MacBook-Pro-macbook Лабораторные работы % git checkout -b branch\_2

Switched to a new branch 'branch\_2'

Добавляем в нее изменения и коммитим стандартной конструкцией git add . а затем git commit -m «Дублируем отчет 2»

Переключаемся на основную ветку macbook@MacBook-Pro-macbook Лабораторные работы % git checkout main

Switched to branch 'main'

Добавляем в нее изменения и коммитим стандартной конструкцией git add . а затем git commit -m «Дублируем отчет 1»

При этом, после переключения на main ветку, дублированный отчет 2 не будет видно, т.к он находится в другой