Частное учреждение образования «Минский колледж предпринимательства»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №27

по учебному предмету

«ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Подпись и дата

Инв. № дубл.

Взам. инв. №

Подпись и дата

Инв. № подл.

Выполнили учащиеся 3 курса группы П-2208

_____Р.И. Шидловский

_____А.И. Асиевская

_____А.В. Быков

_____Б.Д. Садовский

Преподаватель

_____ А.П.Кислюк

2024

Тема проекта: Создание и внедрение онлайн-платформы для продажи чая с удобным каталогом, функцией поиска и фильтрации товаров, а также возможностью оформления заказов и доставки

Цель проекта: Разработка сайта-каталога чая с целью повышения продаж. Сайт будет служить удобной платформой для онлайн-покупок чая, обеспечивая пользователям простой доступ к ассортименту, а также возможности для быстрого поиска и фильтрации товаров.

1. **Что такое система управления версиями?**
   Система управления версиями (СУВ) — это инструмент, который помогает отслеживать изменения в коде или документах, а также управлять различными версиями файлов. Она позволяет нескольким разработчикам работать над одним проектом, фиксировать изменения, отслеживать историю изменений и возвращаться к предыдущим версиям. Git — это одна из самых популярных СУВ.
2. **Как создать репозиторий?**
   Чтобы создать локальный репозиторий в Git:
   Открыть командную строку
   Перейдите в каталог вашего проекта
   Ввести команду: git init
   Чтобы создать удаленный репозиторий (например на GitHub):
   Зарегистрироваться на GitHub
   Создайте новый репозиторий через интерфейс сайта
   Скопировать URL репозитория
   Добавить удаленный репозиторий к локальному проекту
   (git remote add origin https://github.com/username/repo.git)
3. **Как создать ветку?**
   Ветка (branch) — это независимая линия разработки в репозитории. Ветки позволяют работать над новыми фичами или исправлениями, не затрагивая основную (главную) ветку. Чтобы создать ветку:
   git branch <название_ветки>
   Для переключения на новую ветку:
   git checkout <название_ветки>
   Или можно объединить создание и переключение на ветку в одну команду:
   git checkout -b <название_ветки>
4. **Как провести слияние? Как разрешить конфликт и что это такое?**

Слияние (merge) — это процесс объединения изменений из одной ветки в другую. Например, вы хотите объединить ветку feature в основную ветку main:

```
git checkout main       # Переключаемся на основную ветку
git merge feature       # Сливаем изменения из ветки feature
```

Конфликт слияния возникает, когда Git не может автоматически объединить изменения в двух файлах. Это происходит, когда одна и та же строка была изменена в обеих ветках. Чтобы разрешить конфликт:

Откройте файл, в котором произошел конфликт

Найдите метки конфликта (например, <<<<<<<, =======, >>>>>>>)

Выберите какие изменения оставить и удалите метки

После разрешения конфликта добавьте файл в индекс:

```
git add <файл>
```

Завершите слияние

```
git merge --continue
```

5. **Как зафиксировать изменения?**

   Чтобы зафиксировать изменения, сначала добавьте файлы в индекс (staging area), а затем выполните команду для коммита:

   Добавьте файлы в индекс:

   ```
   git add <файл>
   ```

   Чтобы добавить все файлы

   ```
   git add .
   ```

   Зафиксировать изменения с сообщением

   ```
   git commit -m "Описание изменений"
   ```

6. **Как провести откат? Различия в reset и revert, мягкий и жесткий reset.**

   Откат изменений — это процесс отмены коммитов.

   reset — команда для отмены изменений в локальном репозитории. Она изменяет историю коммитов и может быть использована для отмены одного или нескольких последних коммитов.

   Мягкий reset (git reset --soft): Откатывает коммиты, но оставляет изменения в рабочем каталоге (индексированные файлы остаются).

   ```
   git reset --soft <commit_id>
   ```

   Жесткий reset (git reset --hard): Откатывает коммиты и удаляет все изменения в рабочем каталоге. Это необратимо!

   ```
   git reset --hard <commit_id>
   ```

   revert — создает новый коммит, который отменяет изменения предыдущего коммита, не изменяя историю:

git revert <commit_id>

7. **Какова последовательность действий при работе с локальным репозиторием?**
   Создание репозитория: git init.
   Добавление файлов: git add <файл> или git add .
   Коммит изменений: git commit -m "Описание изменений".
   Работа с ветками: git branch, git checkout, git merge.
   Просмотр состояния: git status, git log.
   Откат изменений: git reset, git revert.
8. **Какова последовательность действий при работе с удаленным репозиторием?**
   Клонирование репозитория:
   git clone <URL репозитория>
   Добавление удаленного репозитория:
   git remote add origin <URL репозитория>
   Получение изменений с удаленного репозитория
   git fetch
   Отправка изменений в удаленный репозиторий
   git push origin <ветка>
9. **Каковы возможности при работе с удаленным репозиторием? Как его клонировать, получать и отправлять данные?**
   Клонирование репозиториев
   git clone <URL>
   Это создаст локальную копию удаленного репозитория
   Получение данных из удаленного репозитория:
   fetch: Загружает изменения, но не сливает их с текущей веткой.
   git fetch
   pull: Загружает изменения и сливает их с текущей веткой.
   git pull
   Отправка изменений в удаленный репозиторий:
   push: Отправляет локальные коммиты на сервер.
   git push origin <ветка>