Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Основы кроссплатформенного программирования»
Вариант

Выполнил:
Якушенко Антон Андреевич
2 курс, группа ИТС-б-о-23-1, 11.03.02 «Инфокоммуникационные технологии и системы связи», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: ИССЛЕДОВАНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ GIT И GITHUB

Цель: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Порядок выполнения работы:

Ссылка на репозиторий: https://github.com/Yakush766/LABA1.git

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный мною язык программирования.

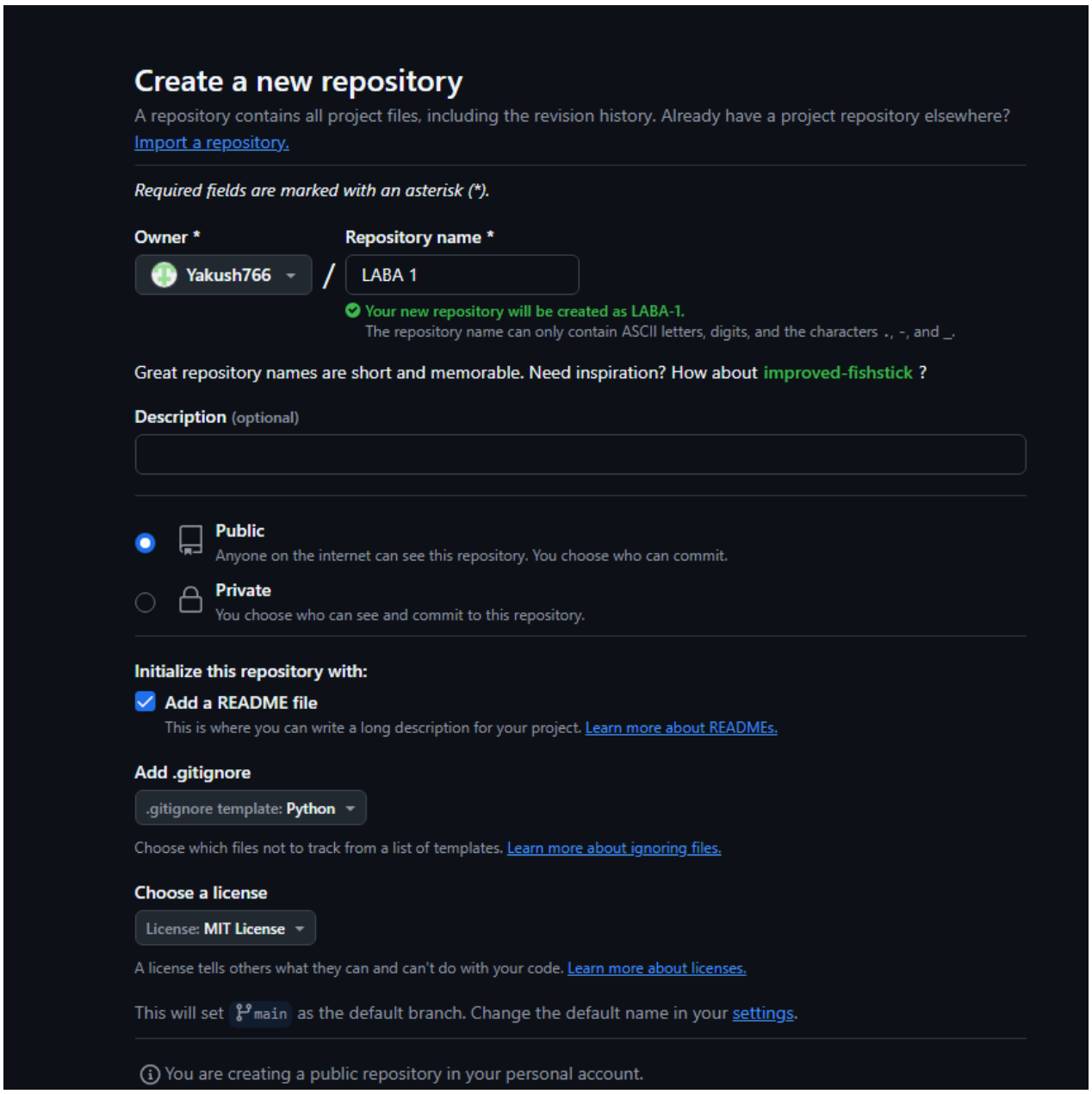

Рисунок 1. Создание репозитория

3. Выполнил клонирование созданного репозитория на рабочий компьютер.

```
C:\Users\anton>git clone https://github.com/Yakush766/LABA1.git
Cloning into 'LABA1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование репозитория

4. Добавил в файл README.md информацию.

```
1  # LABA1
2
3  ФИО: Якушенко Антон Андреевич
4
5  Группа: ИТС-6-о-23-1
```

Рисунок 3. Внесение изменений в файл

6. Написал небольшую программу на выбранном мною языке программирования. Зафиксировал изменения при написании программы в локальном репозитории.

```
1   import random
2
3   number = random.randint(1, 100)  # Загаданное число
4   print("Угадайте число от 1 до 100!")
5   while True:
6   guess = int(input("Ваш вариант: "))
7   if guess == number:
8   print("Поздравляем, вы угадали!")
9   break
10  print("Меньше" if guess > number else "Больше")
```

Рисунок 4. Программа на языке Python

7.Зафиксировал изменения при написании программы в локальном репозитории. Сделал не менее 7 коммитов.

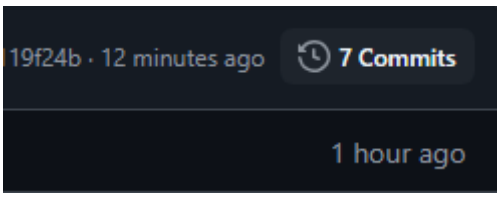

Рисунок 5. Добавление 7 коммитов в репозиторий

8. Добавил отчет по лабораторной работе в формате PDF в папку ДОК репозитория. Зафиксировал изменения. Отправил файл из локального репозитория в удалённый репозиторий GitHub.

Ответы на контрольные вопросы:

1 Что такое СКВ и каково ее назначение?

– Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2 В чем недостатки локальных и централизованных СКВ?

– Локальные СКВ можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. Централизованные СКВ имеет очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

3 К какой СКВ относится Git?

– Git относится к распределенной системе управления версиями.

4 В чем концептуальное отличие Git от других СКВ?

– Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени

5 Как обеспечивается целостность хранимых данных в Git?

– В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хешсумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом.

Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

6 В каких состояниях могут находится файлы в Git? Как связаны эти состояния?

– У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное

(committed), изменённое (modified) и подготовленное (staged). Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной.

Если версия файла изменена и добавлена в индекс, значит, она подготовлена. И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7 Что такое профиль пользователя в GitHub?

– Профиль пользователя в GitHub представляет собой совокупность данных и настроек, связанных с определенным пользователем. Он содержит информацию о пользователях, такую как имя, адрес электронной почты, аватар, вкладки проектов, список слежения, подписки и т.д.

8 Какие бывают репозитории в GitHub?

– В GitHub существует множество различных типов репозиториев, каждый из которых выполняет свою функцию. Одним из основных типов являются локальные и удаленные репозитории Git. Локальный репозиторий создается командой "git init" и служит для хранения истории изменений проекта на компьютере пользователя. Удаленный репозиторий используется для обмена данными между пользователями и совместной работы над проектом.

9 Укажите основные этапы модели работы с GitHub.

– Основные этапы работы с GitHub можно разделить на несколько ключевых шагов:

– Регистрация аккаунта: Создание учетной записи на платформе GitHub. Создание репозитория: Настройка нового репозитория для вашего проекта.

– Клонирование репозитория локально: Использование команды git clone для получения копии репозитория на вашем компьютере.

– Ознакомление с файлами: Анализ структуры репозитория и его содержимого.

– Добавление новых файлов и изменение существующих:

– Редактирование файлов, создание новых и внесение изменений.

– Локальная работа с изменениями: Отслеживание изменений с помощью команд git add, git commit и git status.

– Публикация изменений: Публикация ваших изменений в удаленный репозиторий с использованием команды git push.

– Обновление из удаленного репозитория: Загрузка последних изменений от других участников с помощью команды git pull.

– Проверка и утверждение изменений (Merge Requests): Обсуждение и утверждение предложенных изменений через систему запросов на слияние (merge requests).

– Поддержание актуальности и безопасности: Постоянное обновление своего локального репозитория, чтобы избежать несоответствий с удаленным репозиторием.

10 Как осуществляется первоначальная настройка Git после установки?

– После установки вам нужно настроить некоторые параметры, такие как имя пользователя и адрес электронной почты. Делается это, выполнив команду `git config --global user.name "Ваше имя"` и `git config --global user.email "ваш@email.com"`.Клонирование репозитория. Для этого найдите нужный репозиторий на GitHub или другом сервисе и скопируйте ссылку на

него. Затем запустите команду `git clone URL_REPOSITORY`, где URL_REPOSITORY — это ссылка на ваш репозиторий. Это создаст локальную копию репозитория на вашем компьютере. Работа с файлами: Войдите в папку, которую создали при клонировании, и начните работать с файлами. Используйте команды `git add`, `git commit` и `git push`, чтобы добавлять новые файлы, делать коммиты и публиковать свои изменения.

11 Опишите этапы создания репозитория в GitHub.

– Регистрация на GitHub.

Войти в свой аккаунт: Авторизуйтесь под своей учетной записью. Создание нового репозитория: Перейдите на главную страницу GitHub и нажмите кнопку "Создать новый репозиторий". Заполнение информации о репозитории:

Название репозитория: Введите название вашего проекта.

Выбор организации: Выберите организацию, если она есть, или оставьте пустым для личного проекта.

Тема: Укажите тему проекта, если это применимо.

Платформа/язык: определите платформу и язык программирования, если они известны.

Публичный или частный репозиторий: решите, будет ли репозиторий открытым или закрытым. Открытые репозитории доступны всем пользователям GitHub, тогда как закрытые требуют приглашения или членства для доступа.

Настройки репозитория: Дополнительные настройки, такие как использование линкованного контейнера или конфигурации CI/CD, могут быть сделаны на этом этапе.

12 Какие типы лицензий поддерживаются GitHub при создании репозитория?

– GitHub поддерживает различные типы лицензий при создании репозитория. Среди них можно выделить GNU General Public License (GNU

GPL), которая позволяет свободно распространять и модифицировать программное обеспечение, и GNU Lesser General Public License (LGPL), предназначенную для разработки программного обеспечения с собственнической лицензией. Эти лицензии позволяют пользователям получить доступ к исходному коду и участвовать в разработке проектов, сохраняя при этом авторские права на исходный код. Кроме того, GitHub поддерживает интеграцию с различными системами контроля версий, включая Git, позволяя пользователям выбирать наиболее подходящий инструмент для управления своим кодом.

13 Как осуществляется клонирование репозитория GitHub?

Зачем нужно клонировать репозиторий? – Клонирование репозитория GitHub осуществляется с помощью команды git clone. Эта команда позволяет загрузить всю историю изменений репозитория и сохранить её на вашем компьютере. Клонирование необходимо для:

– Работа с проектом локально: Клонируя репозиторий на свой компьютер, вы получаете возможность работать с ним без подключения к интернету. Это особенно полезно, когда вы находитесь в местах с плохим интернет-соединением или когда вам нужно быстро внести изменения и проверить их перед отправкой в основную ветвь.

– Безопасность и резервные копии: Обладая локальной копией репозитория, вы всегда сможете восстановить проект в случае утраты доступа к серверу или другим проблемам с подключением.

– Совместная работа и разработка: Локальная копия репозитория позволяет нескольким участникам команды одновременно работать над проектом, синхронизируя изменения позже.

14 Как проверить состояние локального репозитория Git?

– Проверить состояние локального репозитория Git можно с помощью команды git status.

15 Как изменяется состояние локального репозитория Git после выполнения следующих операций:

– добавления/изменения файла в локальный репозиторий Git;

– добавления нового/ измененного файла под версионный контроль с помощью команды git add ;

– фиксации(коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push?

1. Добавление/изменение файла в локальный репозиторий Git:

- Состояние репозитория: Файл перейдет в состояние "измененный" (modified).

2. Добавление нового/изменённого файла под версионный контроль с помощью команды `git add`:

- Состояние репозитория: Добавленные файлы переходят в состояние "отслеживаемый" (tracked).

3. Фиксация (коммит) изменений с помощью команды `git commit`:

- Состояние репозитория: Измененные файлы становятся неизменёнными, так как изменения фиксируются и добавляются в историю.

Новые файлы продолжают оставаться в состоянии "отслеживаемые".

4. Отправка изменений на сервер с помощью команды `git push`:

- Состояние репозитория: Все изменения, сделанные на локальном уровне, передаются на удаленный сервер. Состояние файлов остается неизменным.

16 У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в

синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

– Для того чтобы оба локальных репозитория находились в синхронизированном состоянии с репозиторием на GitHub, следует выполнить следующие шаги:

–Клонирование репозитория на первый компьютер

–Клонирование репозитория на второй компьютер

–Работа с изменениями

– Внесите изменения в файлы.

–Добавьте измененные файлы в индекс

–Зафиксируйте изменения

–Публикуйте изменения на сервере

–Получите последние изменения из репозитория

–Выполнить действия для внесения изменений для второго компьютера

17 GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub. – Помимо GitHub, существуют и другие популярные сервисы, работающие с Git: Bitbucket,GitLab, Visual Studio Team Services, Beanstal, CodeBase.

Особенности GitLab:

– Хостинг репозиториев: Поддержка неограниченного количества пользователей и репозиториев.

– CI/CD: Интеграция с CI/CD, что позволяет автоматизировать процесс развертывания и тестирования.

– Мониторинг и аналитика: Встроенные инструменты для мониторинга производительности, анализа кода и выявления ошибок.

– Поддержка открытого исходного кода: полностью открытый исходный код платформы, что делает ее доступной для использования и расширения.

Преимущества:

– Бесплатный хостинг для открытых проектов.

– Гибкость в настройках и интеграциях.

– Встроенная поддержка CI/CD.

– Простота использования и понятный интерфейс.

Недостатки:

– Может потребоваться больше времени на изучение функционала и конфигураций.

– Меньшее количество интеграций с внешними инструментами по сравнению с GitHub.

– Ограниченные возможности для крупных организаций.

18 Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств. – Работа с GitKraken. Этот клиент предоставляет простой и интуитивный интерфейс для выполнения типичных операций Git.

– Откройте GitKraken и нажмите на значок плюса (+) в верхнем левом углу.

– Выберите пункт "New Repository..." (Новый репозиторий...).

– В появившемся окне введите имя нового репозитория и выберите местоположение для хранения файлов.

– Нажмите "Create Repository" (Создать репозиторий). Теперь у вас есть новый пустой репозиторий.

Клонирование существующего репозитория

– Введите URL репозитория, который хотите клонировать, в адресную строку браузера.

– Нажмите на "Clone Repository from Browser" (Клонировать репозиторий из браузера).

– Выберите место для сохранения репозитория на вашем компьютере.

– GitKraken автоматически клонирует репозиторий и открывает его в новом окне.Вы увидите структуру вашего репозитория в правой части окна.

Добавление файлов

– Переместите файлы, которые хотите добавить, в папку с репозиторием.

– В GitKraken перейдите на вкладку "Files" (Файлы) и убедитесь, что новые файлы отображаются в разделе "Untracked Files" (Нераспознанные файлы).

– Щелкните правой кнопкой мыши на любом из файлов и выберите "Stage File for Commit" (Стадия файла для коммита).

– Повторите этот шаг для всех файлов, которые хотите добавить.

Все добавленные файлы переместятся в раздел "Staged Files" (Стадийные файлы).

Коммит изменений:

– Перейдите на вкладку "Commit" (Коммит).

– В области "Commit Message" (Текст коммита) введите сообщение для коммита.

– Нажмите кнопку "Commit" (Коммит).

Вывод: в ходе данной лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга ITпроектов GitHub.