МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

дисциплина Основы кроссплатформенного программирования

Отчет по лабораторной работе №1

Исследование

основных возможностей Git и GitHub.

Выполнил: студент группы ИТС-б-о-21-1

Пушкин Максим Алексеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил: кандидат технических наук, доцент кафедры инфокоммуникаций,

Роман  Александрович  Воронкин

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Ставрополь, 2022

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

2. **Теоретическое сведения**

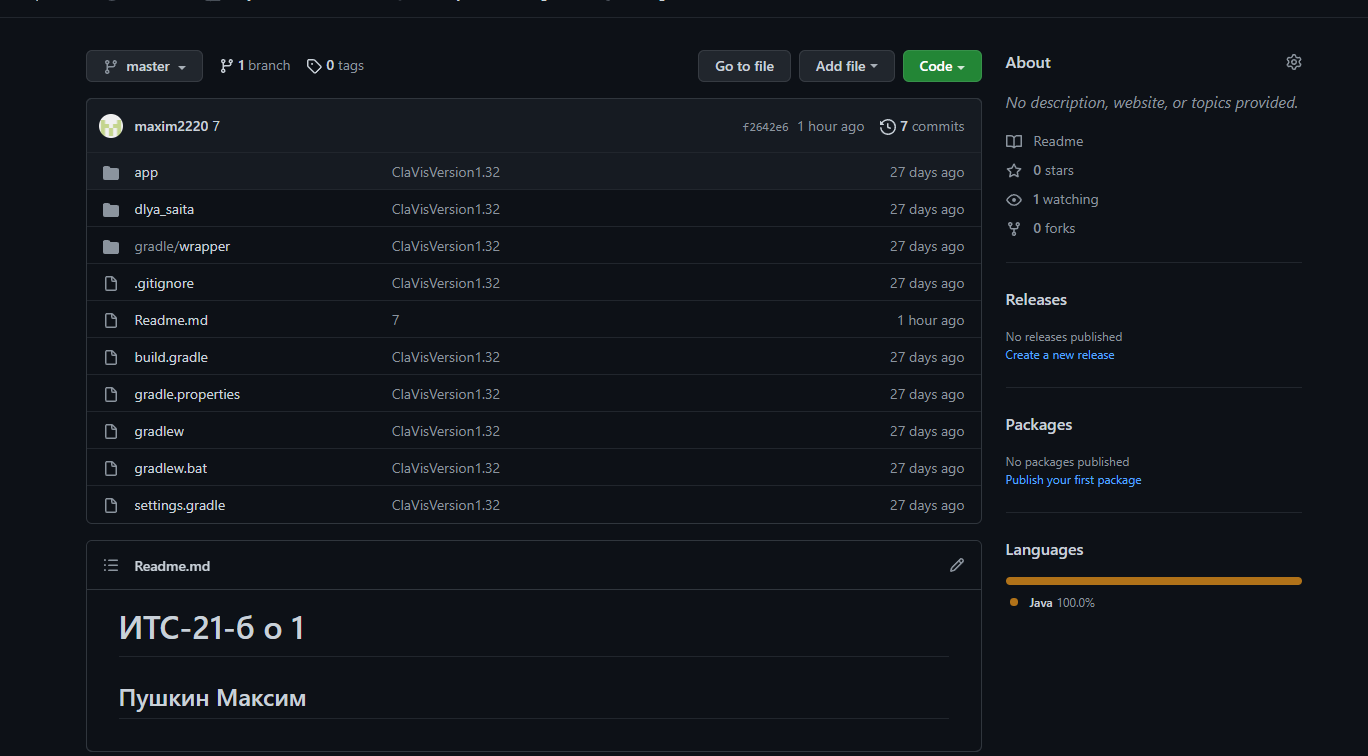
Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях)

Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков (рис. 1.5).

Это очень важное отличие между Git и почти любой другой СКВ. Git переосмысливает практически все аспекты контроля версий, которые были скопированы из предыдущего поколения большинством других систем. Это делает Git больше похожим на миниатюрную файловую систему с удивительно мощными утилитами, надстроенными над ней, нежели просто на СКВ.

**Метод и порядок выполнения работы**

Был создан репозиторий и добавлен файл с проектом и файлом readme.md.



<https://github.com/manulmax21/ClaVis>

**Вопросы для защиты работы**

1. 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. Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.
2. Такой подход имеет множество преимуществ, особенно перед локальными СКВ. Например, все разработчики проекта в определённой степени знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте.
3. Централизованный.
4. 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 (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях)
5. В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. 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. У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged).
7. Учетная запись.
8. Public и private.
9. Установка Git, создание репозитория, внесение изменений, коллективная работа.
10. Для выполнения лабораторной работы необходимо установить Git локально. Для этого необходимо перейти на официальный сайт Git https://git-scm.com/ и скачать версию для Вашей операционной системы. После чего необходимо выполнить установку на свой компьютер. Некоторые дистрибутивы Linux, а также Mac OS поставляются с предустановленным Git. Чтобы убедиться, что Git был успешно установлен, введите команду ниже в терминале, чтобы отобразить текущую версию вашего Git:
11. Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториев, которые вы создавали. Описание (Description). Можно оставить пустым. Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” ( В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать). .gitignore и LICENSE можно сейчас не выбирать
12. GNU GPL Version 3 GNU GPL Version 2 GNU LGPL GNU AGPL MPL Version 2.0 EPL Version 1.0 Ms-PL MIT Apache 2.0 BSD WTFPL Beerware Общественное достояние CC0 Unlicense
13. Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите git clone и введите адрес: Теперь у вас есть локальное хранилище проекта. Обратите внимание, что вы можете переключиться на использование протокола SSL вместо HTTPS для клонирования репозитория. Мы не будем рассматривать это здесь, но вы можете узнать, как это сделать, если вам интересно (например, разработчики часто предпочитают SSL на практике).
14. Перейдите в каталог хранилища и посмотрите на содержимое. Локальный репозиторий включает в себя все те же файлы, ветки и историю коммитов, как и удаленный репозиторий. Введите эту команду, чтобы проверить состояние вашего репозитория: Теперь ваша рабочая копия находится в главной ветке вашего локального репозитория. И она идентична оригинальной главной ветке.
15. Изменится сам репозиторий и обновится комит.

**Вывод**: были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.