***Ответы на вопросы по лабораторной работе №3***

Тема: Знакомство с системой GIT.

Выполнил: Рыжков Александр

Группа: БИС23 – 01

1. *Что представляет собой система GIT?*

Система GIT – это система контроля версий, разработанная Линусом Торвальдсом для проекта ядра Linux. Git является распределительной системой контроля версий, то есть позволяет нескольким разработчикам работать параллельно и независимо, отправляя изменения на центральный сервер.

**Основные особенности GIT включают**:

1. Распределенная система контроля версий: каждый разработчик работает со своей локальной копией репозитория, что позволяет эффективно сотрудничать и обеспечивает сохранность данных.
2. Ветвление: GIT поддерживает создание и слияние веток для организации параллельной работы над различными функциями и задачами без риска конфликтов.
3. Версионный контроль: GIT сохраняет историю изменений, позволяя вернуться к предыдущим версиям кода, что упрощает процесс отладки и устранения проблем.
4. Кросспоточное взаимодействие: разработчики могут обмениваться изменениями между ветками и легко объединять их после согласования.
5. Масштабируемость: GIT хорошо масштабируется и может обрабатывать большие проекты с большим количеством разработчиков и множеством изменений.
6. Открытое программное обеспечение: GIT распространяется под лицензией GNU General Public License, что делает его доступным для всех разработчиков.
7. Перечислите возможности систем управления версиями.
8. Контроль версий - системы управления версиями позволяют отслеживать изменения в коде, сохранять предыдущие версии и возвращаться к ним в случае необходимости.
9. Совместная работа - системы управления версиями облегчают совместную работу над проектом, позволяя нескольким разработчикам работать одновременно и не мешая друг другу.
10. Откат изменений - системы управления версиями дают возможность отката к предыдущим версиям проекта, если внесенные изменения оказались неудачными.
11. Управление ветками - системы управления версиями поддерживают создание веток проекта для параллельной разработки различных функций или исправлений без риска нарушить основную ветку.
12. Интеграция с системами сборки - системы управления версиями могут быть интегрированы с системами автоматической сборки, что позволяет автоматизировать процесс сборки проекта и управлять зависимостями.
13. Интеграция с CI/CD - многие системы управления версиями, такие как Git, предлагают интеграцию с CI/CD (Continuous Integration/Continuous Delivery), что позволяет автоматически собирать, тестировать и развертывать проект при каждом изменении кода.
14. Управление доступом - системы управления версиями предоставляют возможность управления доступом к репозиторию, позволяя ограничить доступ только определенным группам разработчиков или конкретным людям.
15. Что такое коммит и когда он выполняется?

***Коммит (Commit)*** — это сохранение изменений в системе контроля версий. Он выполняется тогда, когда разработчик хочет сохранить изменения в своем рабочем каталоге. Коммит сохраняет состояние репозитория на определенный момент времени, что позволяет вернуться к этому состоянию в случае возникновения проблем или внесения новых изменений.

1. Как посмотреть историю коммитов?

Чтобы посмотреть историю коммитов в GIT, нужно выполнить следующие команды:

- Перейти в папку с проектом

- Ввести команду “git log”

- В итоге будет указана история коммитов, автор, сообщение коммита и дата

По умолчанию (без аргументов) “git log” перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядке — последние коммиты находятся вверху.

1. Как создать пустой *GIT* репозиторий?

Это можно сделать с помощью команды “git init”. Данная команда создает репозиторий, то есть хранилище всех изменений файлов в этом каталоге.

По шаговый план создания пустого репозитория с первым коммитом:

- Создать новую папку для репозитория

- Войти в данную папку с помощью команды “cd” в командной строке

- Создать пустой репозиторий с помощью команды:

git init

- Добавить файлы проекта с помощью команды:

git add

- Создайте первый коммит с помощью команды:

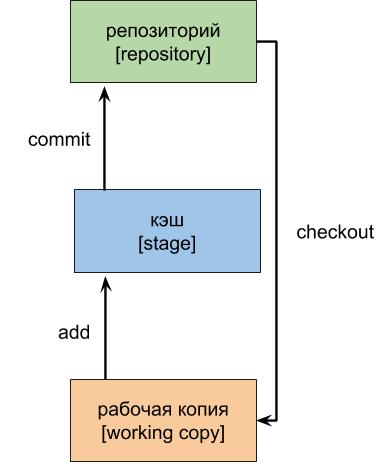
Git commit -m “Первый коммит”

1. Какую архитектуру имеет система контроля версий *GIT*?

Система контроля версий *GIT*имеет архитектуру трех деревьев. Схематично она выглядит так как показано на рисунке ниже.

Основные определения:

1. **Working сору** — это копия данных, с которой вы работаете в данный момент. Она может быть в репозитории или на компьютере.
2. **“Add” в Git** — это операция, при которой файлы добавляются в индекс перед коммитом. Это позволяет вам работать с несколькими файлами одновременно, а затем отправить их все в одном коммите.
3. **Stage (кэш) в GIT** — это временное хранилище файлов, которые были изменены, но еще не были зафиксированы в репозитории. Кэш используется для оптимизации процесса коммита, чтобы не загружать все файлы из репозитория каждый раз, когда вы хотите изменить один файл.
4. **Коммит (Commit)** — это сохранение изменений в системе контроля версий.
5. **Репозиторий в GIT** — это место, где хранятся все изменения, которые были сделаны в проекте. Он состоит из нескольких веток, каждая из которых представляет собой отдельную версию проекта.
6. **Checkout** — это процесс, при котором вы берете копию данных из репозитория.
7. Суть ее заключается в том, что дополнительно добавляется ещё одно место, которое можно назвать кэшем или *stage*в английской терминологии. Рабочая копия и репозиторий идейно не отличается от их аналогов в архитектуре двух деревьев.



1. Как создать новую ветку в *GIT* и соединить ветки?

**Создание новой ветки в GIT:**

1. Необходимо перейти в директорию проекта, который необходимо контролировать с помощью GIT.
2. Создать новую ветку с помощью команды ‘git branch’ в командной строке.
3. Подтвердить создание новой ветки, введя ‘git checkout’ в командную строку.
4. Теперь создана новая ветка и можно начинать работу над новым функционалом или исправлениями.

**Соединение веток в GIT (merge):**

1. Чтобы объединить изменения между двумя ветками, необходимо перейти на ту ветку, которую нужно объединить.
2. Далее нужно использовать команду ‘git merge’ с именем ветки, которую необходимо объединить. Например, ‘git merge origin/master’ для объединения ветки ‘master’ из удаленного репозитория ‘origin’.
3. GIT автоматически проведет merge и покажет конфликты, если таковые возникнут.
4. Нужно разрешить конфликты, если необходимо, и ввести ‘git commit’ для фиксации изменений.
5. Теперь ветка объединена с другой веткой, и можно продолжить работу над проектом.
6. Поясните понятие «*staging area*».

“Staging area” — это область в системе контроля версий, где хранятся изменения, которые еще не были применены к рабочему дереву или не были отправлены на сервер. Эта область позволяет разработчикам работать с изменениями, не затрагивая рабочую копию проекта. После того, как изменения были внесены в staging area, они могут быть отправлены на сервер с помощью команды “git push”.

1. Как сравнить версии файла в *GIT*?

Для сравнения версий файла в Git можно использовать команду "git diff". В терминале это должно выглядеть следующим образом: git diff "Название файла". Git покажет различия между текущей версией файла и последним коммитом в репозитории. Вы увидите добавленные, удаленные и измененные строки кода. Вы также можете указать определенные коммиты или ветки для сравнения, используя соответствующие параметры команды "git diff".