Лабораторная работа №20

Тема: Изучение систем контроля версий GIT, GITHUB, GitLAB и др.

Вопросы для самоконтроля

**Назначение систем CVS**: Системы контроля версий (CVS – Concurrent Versions System) предназначены для управления изменениями в коде и документации, а также для координации работы нескольких разработчиков над проектами. Они позволяют отслеживать изменения, возвращаться к предыдущим версиям, сравнивать различные версии и управлять конфликтами при слиянии правок. 2. **Виды CVS**: - **Centralized Version Control Systems (CVCS)**: Это традиционные системы, такие как CVS, Subversion (SVN), которые используют центральный репозиторий. Все пользователи работают с одной историей изменений. Одна из основных особенностей — наличие единой, централизованной базы данных, что может привести к проблемам при одновременной работе нескольких пользователей. - **Distributed Version Control Systems (DVCS)**: К таким системам относится Git, Mercurial и другие. В них каждый разработчик имеет полную копию репозитория со всей историей изменений. Это позволяет работать в оффлайн-режиме и синхронизировать изменения по мере необходимости. 3. **Особенности подхода к работе с данными в Git**: - **Локальные репозитории**: Каждый разработчик имеет полный локальный репозиторий, включая всю историю изменений, что позволяет работать без постоянного подключения к сети. - **Снимки (snapshots)**: Git хранит данные как снимки, а не просто как набор изменений. Это делает операции, такие как переключение версий, быстрыми и легкими. - **Гибкость ветвления**: Git позволяет легко создавать и переключаться между ветками, что способствует экспериментированию и параллельной работе над различными функциями. - **Обширные механизмы слияния**: Git предоставляет мощные инструменты для слияния веток и разрешения конфликтов. **Преимущества Git**: - Высокая скорость работы. - Эффективная обработка большого количества данных. - Возможность работы в оффлайн-режиме. - Широкие возможности для совместной работы. 4. **Процесс установки и настройки Git**: - **Установка Git**: - **Windows**: 1. Скачайте установочный файл с [официального сайта Git](https://git-scm.com/). 2. Запустите установочный файл и следуйте инструкциям.

1. **Назначение сервиса GitHub**: GitHub — это веб-сервис для хостинга репозиториев Git, который предоставляет инструменты для управления проектами, совместной работы и отслеживания версий. Он улучшает возможности распределенного контроля версий за счет поддержки функций, таких как pull-реквесты, управление задачами, код-ревью и другие инструменты для сотрудничества разработчиков. 2. **Назначение сервиса GitHub Desktop**: GitHub Desktop — это десктопное приложение, которое упрощает использование Git и GitHub. Оно предоставляет графический интерфейс для выполнения операций Git, таких как коммиты, пуши, пуллы и создание веток, позволяя разработчикам работать с репозиториями без необходимости использовать командную строку. 3. **Термины git и github**: 1. **Репозиторий**: Хранилище для вашего проекта, которое содержит все его файлы и историю изменений. 2. **Рабочая область и хранилище**: Рабочая область (working directory) — это место на вашем компьютере, где вы работаете над файлами проекта. Хранилище (repository) — это .git каталог, где Git хранит все необходимые данные для управления версиями. 3. **Форк**: Копия репозитория, созданная для внесения изменений без затрагивания оригинального проекта. Обычный способ работы с открытыми проектами — это создание форка, внесение изменений и отправка их через pull-реквест. 4. **Ветка**: Логическая линия разработки, которая отделена от основной линии (обычно `main` или `master`). Используется для работы над новыми функциями или исправлениями ошибок без риска нарушить стабильную версию. 5. **Мастер**: Имя основной ветки в репозитории Git, хотя в новых версиях Git по умолчанию используется имя `main`. 6. **Коммит**: Снимок состояния файлов в определенный момент времени, который содержит информацию о внесенных изменениях. 7. **Пул**: Команда, использующаяся для получения изменений из удаленного репозитория и слияния их с локальной веткой. 8. **Пуш**: Команда для отправки локальных изменений в удаленный репозиторий, обновляя его содержимое. 9. **Пулреквест**: Запрос на вливание изменений из одной ветки в другую. Типично используется для обсуждения и проверки изменений перед их присоединением к основной ветке. 10. **Мердж**: Операция, объединяющая изменения из одной ветки в другую. 11. **Кодревью**: Процесс проверки и обсуждения кода другими разработчиками перед его слиянием в основной проект для обеспечения качества и соответствия стандартам.