**3 РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ**

Для проведения лабораторного практикума были выбраны Subversion, Mercurial и Git в качестве систем контроля версий и tcl/tk, zenity и HTML в качестве примеров для написания программ или разметки.

**3.1 Использованные протоколы для работы с системами контроля версий**

В данном дипломном проекте при первоначальной разработке использовался аккаунт Microsoft Azure. В связи с тем, что качество интернет-соединения было достаточно невысокое (по сравнению с требованиями Azure), и оно на настоящий момент не может предоставить скорости, которой будет достаточно для качественного отображения удалённого рабочего стола, было принято решение тестировать и проводить отладку программного комплекса лабораторного практикума с использованием программы PuTTY, которая взаимодействует с удалённым сервером через SSH (без использования удалённого рабочего стола).

SSH (англ. Secure Shell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео. Также SSH может использовать сжатие передаваемых данных для последующего их шифрования.

Кроме этого, при разработке использовался SSH-туннель — туннель, создаваемый посредством SSH-соединения и используемый для шифрования туннелированных данных. Он используется для того, чтобы обезопасить передачу данных в Интернете (аналогичное назначение имеет IPsec). При пересылке через SSH-туннель незашифрованный трафик любого протокола шифруется на одном конце SSH-соединения и расшифровывается на другом.

Все эти меры предосторожности были нужны потому, что аккаунт в Azure является платным, а в случае перехвата злоумышленниками пары логин-пароль и последующего несанкционированного доступа к аккаунту разработчика, поддержка Azure-аккаунта закрывается (однако, злоумышленник может потратить достаточно большое количество средств, которые всё ещё будут на счету разработчика).

Кроме непосредственно разработки и взаимодействия с удалённой ОС необходимо было использовать сетевой протокол для общения с системами контроля версий. Все используемые в лабораторном практикуме системы контроля версий умеют работать с четырьмя сетевыми протоколами для передачи данных: локальный, Secure Shell (SSH), HTTP и собственный (соответственно, SVN, Git или Mercurial). Важно понимать, что за исключением протокола HTTP, все эти протоколы требуют, чтобы система контроля версий была установлена и работала на сервере.

Базовым является локальный протокол, при использовании которого удалённый репозиторий — другой каталог на диске. Наиболее часто он используется, если все члены команды имеют доступ к общей файловой системе, например, к NFS, или, что менее вероятно, когда все работают на одном компьютере. Последний вариант не столь хорош, поскольку все копии вашего репозитория находятся на одном компьютере, делая возможность потерять всё более вероятной.

Наиболее часто используемый транспортный протокол — это SSH. Причина этого в том, что доступ по SSH уже есть на многих серверах, а если его нет, то его очень легко настроить. Кроме того, SSH — единственный из сетевых протоколов, предоставляющий доступ и на чтение, и на запись. Два других сетевых протокола (HTTP и Git) в большинстве случаев дают доступ только на чтение, поэтому даже если они вам доступны, вам всё равно понадобится SSH для записи. К тому же SSH это протокол с аутентификацией, и благодаря его распространённости обычно его легко настроить и использовать.

Огромный плюс протоколов HTTP и HTTPS в простоте их настройки. По сути, всё, что необходимо сделать — поместить пустой репозиторий внутрь каталога с HTTP документами, установить перехватчик post-update. Теперь каждый, имеющий доступ к веб-серверу, на котором был размещён репозиторий, может его склонировать.

Одним из примеров собственных протоколов является Git-протокол. Так, вместе с Git'ом поставляется специальный демон, который слушает порт 9418 и предоставляет сервис, схожий с протоколом SSH, но абсолютно без аутентификации. Чтобы использовать Git-протокол для репозитория, вы должны создать файл git-daemon-export-ok, иначе демон не будет работать с этим репозиторием, но следует помнить, что в протоколе отсутствуют средства безопасности. Соответственно, любой репозиторий в Git'е может быть либо доступен для клонирования всем, либо не доступен никому. Как следствие, обычно вы не можете отправлять изменения по этому протоколу. Вы можете открыть доступ на запись, но из-за отсутствия авторизации в этом случае кто угодно, зная URL вашего проекта, сможет его изменить.

При составлении лабораторного практикума был использован протокол SSH, как наиболее популярный, доступный и действенный способ соединения.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

**3.2 Форматы файлов при использовании систем контроля версий**

В данном цикле лабораторных работ используются такие системы контроля версий, в которых при сохранении новых версий на сервере используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных. Таким образом, на сервере будут лежать уже сжатые файлы; прочитать из них информацию без предварительной переработки не получится.

На клиенте же ситуация другая: в каталоге .git (в случае системы контроля версий Git) находится вся нужная информация про файлы и папки, а в корне самой директории – непосредственно текущая версия репозитория. Подробный пример можно увидеть на рисунке 3.1:

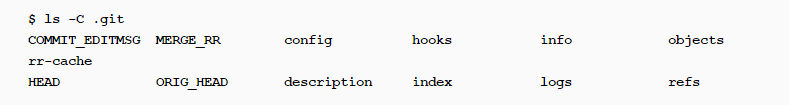


Рисунок 3.1 – Содержание .git репозитория на клиенте

В папке с базой данных объектов хранятся преимущественно каталоги, состоящие из двух символов – первых двух букв хэша sha1 объекта, хранящегося в Git. Внутри лежат сжатые и закодированные данные про объекты, хранящиеся в Git.

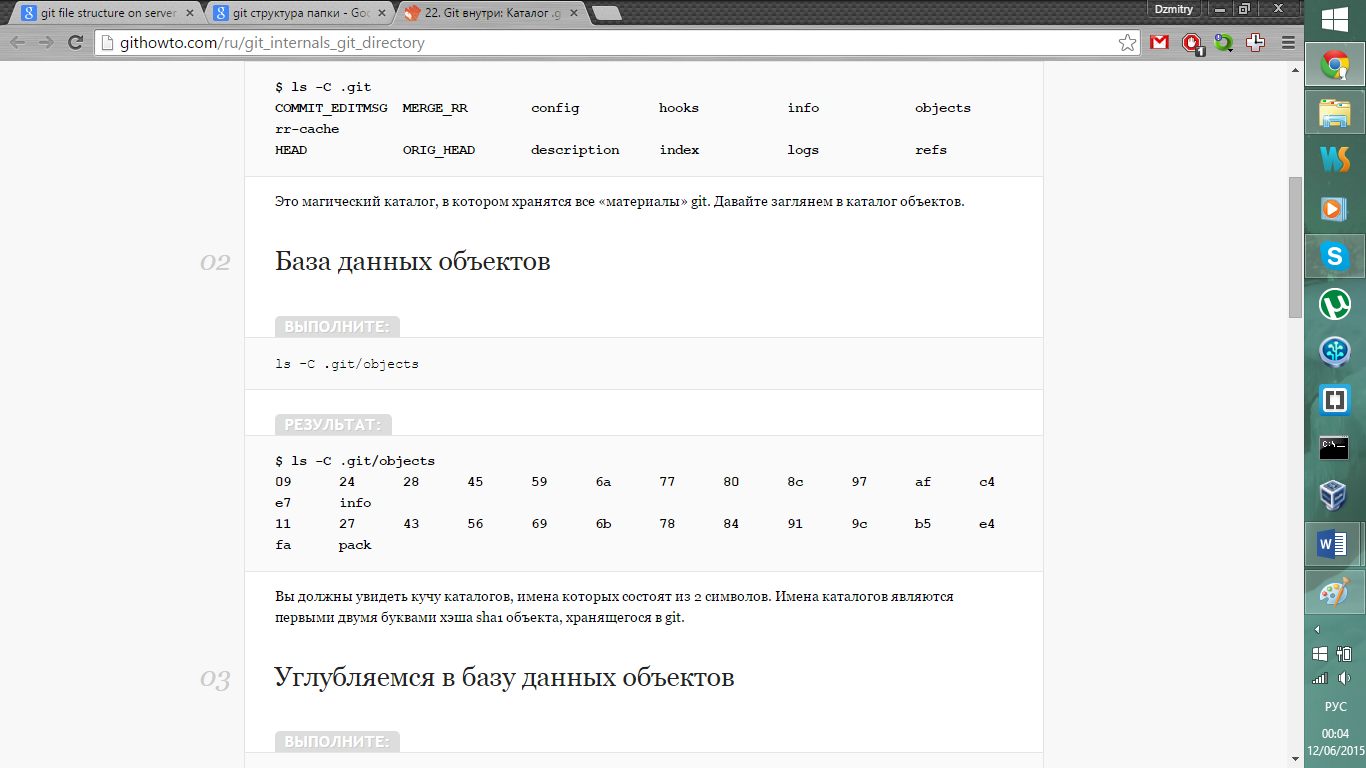


Рисунок 3.2 – Содержание папки object