**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

На рисунке 2.1 можно увидеть, каким образом будет осуществляться взаимодействие между рабочей программой и системами контроля версий.

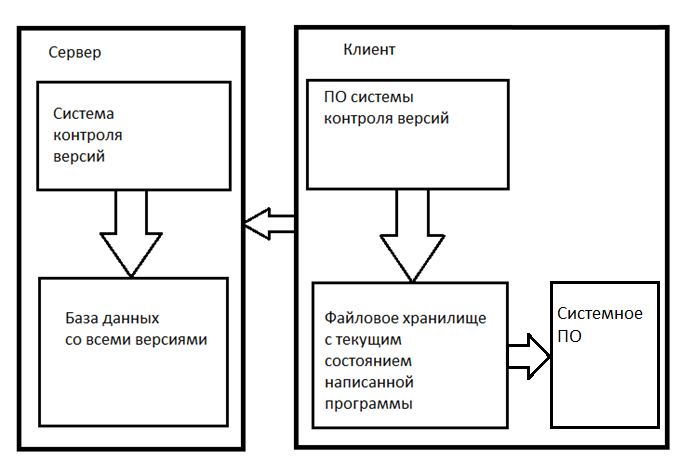


Рисунок 2.1 – Структурная схема лабораторного практикума

Таким образом, взаимодействие будет осуществляться между клиентом и сервером. На сервере будет установлена система контроля версий, в случае с Mercurial и Subversion дополнительно будет установлен веб-сайт (для просмотра результатов работы). На клиенте же, помимо клиентской части, из серверного репозитория с бинарными данными будет воссоздана файловая система в какой-то ревизии. В файлах и папках, которые будут получены в результате этого, будет находиться разрабатываемое ПО; причём для всех учащихся первоначальная версия репозитория будет выглядеть одинаково.

**2.1 Выбор децентрализованной системы контроля версий**

При выборе системы контроля версий для написания реального проекта учитывается популярность того или иного инструмента. С целью определить наиболее используемую систему, в 2014 году был проведён опрос среди разработчиков сайтов под систему WordPress. Результаты представлены на рисунке 2.2.

Рисунок 2.2 – Популярность систем контроля версий для WordPress-проектов

Анализируя результаты этого опроса, можно увидеть, что наиболее популярными децентрализованными системами контроля версий являются Git и Mercurial. Именно поэтому они были взяты за основу в лабораторных работах 2, 3 и 4.

Для лабораторной работы, идущей после изучения ЦСКВ (первая), был взят Mercurial. На самом деле, именно Mercurial используют многие программисты, переходящие с ЦСКВ на ДСКВ, так как эта система является заменой для более ранних систем, например Subversion.

Система Mercurial – это кроссплатформенная распределенная система контроля версий с открытым кодом, разработанная для эффективной работы с очень большими репозиториями. У нее два основных назначения:

− она хранит все предыдущие версии каждого файла;

− она может объединить разные версии вашего кода, то есть сотрудники могут независимо работать над кодом и затем объединять свои изменения.

Система Mercurial разделяет момент внесения кода в репозиторий и момент получения этого кода всеми остальными. И это означает, что вы можете создавать наборы изменений (командами hg com или hg commit), но все остальные не получат ваши изменения. Когда у вас накопятся изменения, которые вас устраивают, вы записываете (hg push) их в главный репозиторий, находящийся на центральном сервере.

В системе Subversion мыслят ревизиями. Ревизия — это то, как выглядит вся файловая система в определенный момент времени. В системе Mercurial вы мыслите наборами изменений (changesets). Набор изменений — это четкий список изменений между двумя соседними ревизиями. Система Subversion, по сути, система контроля изменений для файлов, а в Mercurial контроль изменений применяется ко всему каталогу, включая все подкаталоги.

Большинство людей работают с Mercurial через интерфейс командной строки. Так можно работать в Windows, Unix, и Mac. Команда для Mercurial — это hg.

В качестве языка программирования для второй лабораторной работы был выбран tcl и библиотека tk. Язык tcl (от англ. Tool Command Language — «командный язык инструментов») — скриптовый язык высокого уровня. Библиотека tk (от англ. Toolkit — «набор инструментов», «инструментарий») — кроссплатформенная библиотека базовых элементов графического интерфейса, распространяемая с открытыми исходными текстами.

При помощи tcl и tk, в отличие от zenity, можно создавать довольно сложные программы (как на рисунке 2.3).

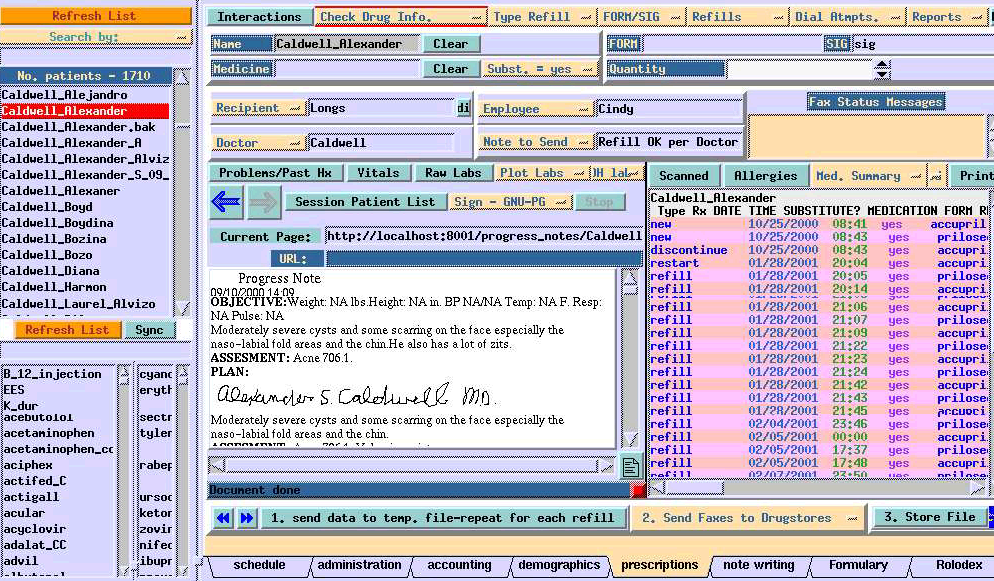


Рисунок 2.3 – Пример программы, созданной при помощи tcl/tk

В лабораторной работе по системе Mercurial будет предложено создать простейшую программу-калькулятор (так же, как и в первой лабораторной). Смысл заключается в том, чтобы показать различия между централизованными и децентрализованными системами контроля версий. Различные языки программирования при этом не дадут возможности учащимся брать за основу готовые результаты работы из первой лабораторной – их придётся сделать заново, как это было бы на настоящем проекте.

Для 3 и 4-й лабораторных работ был выбран Git, как самая популярная система контроля версий. Система Git – распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

Для хранения бинарных файлов, например, электронной библиотеки, Git подходит лучше. По сравнению с Mercurial он не ориентирован на расчет дельты файлов, что для бинарного содержимого не очень эффективно. Сами файлы меняются редко, а основные операции с ними — это перемещение и добавление.

Для третьей лабораторной работы было решено использовать что-то проще, чем язык программирования, так как Git традиционно считается самой сложной системой контроля версий для освоения. Таким образом, было принято решение использовать язык разметки; самый популярный из таких – HTML. HTML (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language — «язык [гипертекстовой](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) разметки») — стандартный [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) содержат описание разметки на языке HTML (или [XHTML](https://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык [XHTML](https://ru.wikipedia.org/wiki/XHTML) является более строгим вариантом HTML, он следует всем ограничениям [XML](https://ru.wikipedia.org/wiki/XML) и, фактически, XHTML можно воспринимать как приложение языка XML к области разметки гипертекста.

В третьей лабораторной работе учащимся будет предложено создать HTML-сайт с множеством страниц (по количеству вариантов). Преимущества использования HTML в данном случае заключается в том, что учащийся сможет наглядно увидеть результаты своей работы и не будет тратить время на отладку кода. Данная работа является самой сложной в практикуме, так как Git имеет достаточно высокий порог вхождения; поэтому в данной работе было принято решение сосредоточиться не на написании кода, а на собственно Git-репозитории.

Пример простого документа на HTML показан на рисунке 2.4.

Для четвёртой лабораторной работы, в которой будут рассматриваться дополнительная функциональность Git (поверх базового уровня), было решено создать репозиторий, в котором учащиеся, путём перехода по различным веткам (название которых являются результатами выполнения некоторых команд в Git-репозитории) будут получать дальнейшие подсказки для достижения выбранной версии ветки. Таким образом, учащимся не придётся учить что-либо новое для того, чтобы узнать больше о возможностях Git. Данная работа является завершающей в цикле, поэтому она подразумевает использование всех навыков работы с системами контроля версий, которые учащиеся приобрели до этого.

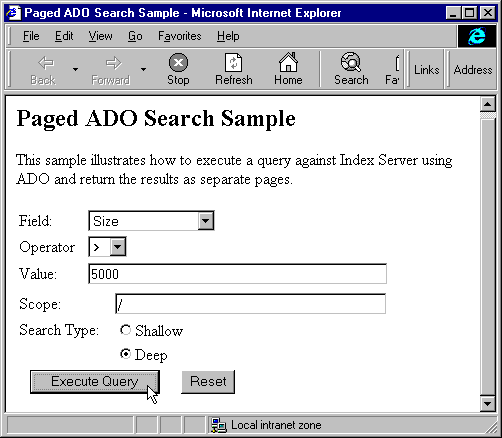


Рисунок 2.4 – Пример браузера, использующего разметку HTML

**2.2 Технологии, использованные при разработке**

При разработке лабораторного практикума была использована программа VirtualBox. Программа VirtualBox (Oracle VM VirtualBox) — программный продукт виртуализации для операционных систем Microsoft Windows, Linux, FreeBSD, Mac OS X, Solaris/OpenSolaris, ReactOS, DOS и других. При написании программ использовалась операционная система версии 14.04 Linux Ubuntu. Кроме этого, все лабораторные работы рекомендуется запускать именно в этой среде, так как именно в ней нужно затратить минимальное количество времени на установку zenity и tcl/tk.

Также в процессе разработки программного обеспечения с целью проверки правильности рекомендаций, которые были даны учащимся, использовался аккаунт Microsoft Azure. Microsoft (Windows) Azure — название облачной платформы от Microsoft. Предоставляет возможность разработки и выполнения приложений и хранения данных на серверах, расположенных в распределенных дата-центрах.

В основе работы Microsoft Azure лежит запуск виртуальной машины для каждого экземпляра приложения. Разработчик определяет необходимый объем для хранения данных и требуемые вычислительные мощности (количество виртуальных машин), после чего платформа предоставляет соответствующие ресурсы. Когда первоначальные потребности в ресурсах изменяются, в соответствии с новым запросом заказчика платформа выделяет под приложение дополнительные или сокращает неиспользуемые ресурсы дата-центра.

Аккаунт Microsoft Azure обеспечит не только все базовые функции операционной системы, но и дополнительные: выделение ресурсов по требованию для неограниченного масштабирования, автоматическую синхронную репликацию данных для повышения отказоустойчивости, обработку отказов инфраструктуры для обеспечения постоянной доступности и многое другое.

В аккаунт Azure в процессе разработки была добавлена операционная система Linux Ubuntu без рабочего стола. Поэтому для взаимодействия с данным удалённым сервером использовалась программа PuTTY. PuTTY – это свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin. Также имеется возможность работы через последовательный порт.

Утилита PuTTY позволяет подключиться и управлять удаленным узлом (например, сервером). В PuTTY реализована только клиентская сторона соединения — сторона отображения, в то время как сама работа выполняется на другой стороне. В рамках разработки данного лабораторного практикума использовались исключительно консольные возможности данной утилиты.

Для взаимодействия же с системой контроля версий при разработке лабораторного практикума, был использован проект SourceTree. Утилита SourceTree ­– бесплатный Git GUI клиент для настольных компьютеров под управлением Windows. Благодаря ему, не нужно постоянно работать с командной строкой, а большинство стандартных операций (создание пакета изменений, отправка изменений на сервер, скачивание данных с сервера, объединение веток) можно сделать в несколько кликов. Тем ни менее, SourceTree имеет в себе и встроенную поддержку терминала, которую можно использовать для разрешения сложных конфликтов или при использовании специфических операций (таких, как выборка одного набора изменений и его добавление в другую ветку). Таким образом, для опытного разработчика, программа SourceTree может значительно облегчить работу, заменив ввод множества команд несколькими кликами мышью.

Программа SourceTree основана на утилите mSysGit, требует его установки на машину и работает через исполнение его командных строк. Тем не менее, утилита SourceTree содержит проект mSysGit внутри себя (а не использует его через внешние зависимости), и устанавливает его в поддиректорий на Windows-машине, что затрудняет добавление нужных SSL-сертификатов.