РАБОТА №1 ЛАБОРАТОРНАЯ. СОЗДАНИЕ И ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ РЕПОЗИТОРИЯ ПРОЕКТА

1. ЦЕЛЬ РАБОТЫ

Целью выполнения лабораторных работ является:

- получить опыт практической работы с системой контроля версий на примере Tortoise SVN;
- получить навыки создания репозитория.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Под системой контроля версий понимается механизм сохранения промежуточных состояний кода разрабатываемого программного обеспечения. То есть с помощью этой системы программист может управлять своими файлами во времени: смотреть историю изменений файлов и каталогов, возвращаться к более ранним версиям кода, объединять несколько версий файла. Основной областью применения контроля версий является коллективная разработка чего-либо (чаще всего это программы, но область применения программированием не ограничивается). Однако и для разработчика-одиночки контроль версий может быть полезен.

Для решения этих проблем удобно использовать системы контроля версий, например Subversion (SVN).

Subversion (SVN) – одна из наиболее известных систем версионного управления исходным кодом. Применяется во многих Open Source проектах. Подобные системы позволяют узнать, кто, когда и как именно вносил изменения в файлы проекта. Все это значительно облегчает групповую разработку ПО.

Обычно система контроля версий состоит из двух частей:

Сервер, или репозиторий – где хранятся все исходные коды программы, а также история их изменения.

Клиент. Каждый клиент имеет свою локальную копию (working copy) исходных кодов, с которой работает разработчик.

В связи с тем, что разработчики работают только с локальными копиями, могут возникать трудности, когда два и более человек изменяют один и тот же файл.

В системе контроля версий есть две модели, которые позволяют избегать этой проблемы:

Блокировка – изменение – разблокировка. Согласно этой модели, когда кто-либо начинает работу с файлом, этот файл блокируется, и все остальные пользователи теряют возможность его редактирования. Очевидным недостатком такой модели является то, что файлы могут оказаться надолго заблокированными, что приводит к простоям в разработке проекта.

Копирование – изменение – слияние. В данной модели каждый разработчик свободно редактирует свою локальную копию файлов, после чего выполняется слияние изменений. Недостаток этой модели в том, что может возникать необходимость разрешения конфликтов между изменениями файла.

Получение информации о статусе

При работе с рабочей копией, вам часто надо понять, какие файлы вы заменили,/добавили,/удалили или переименовали, или же какие из файлов были изменены и зафиксированы другими.

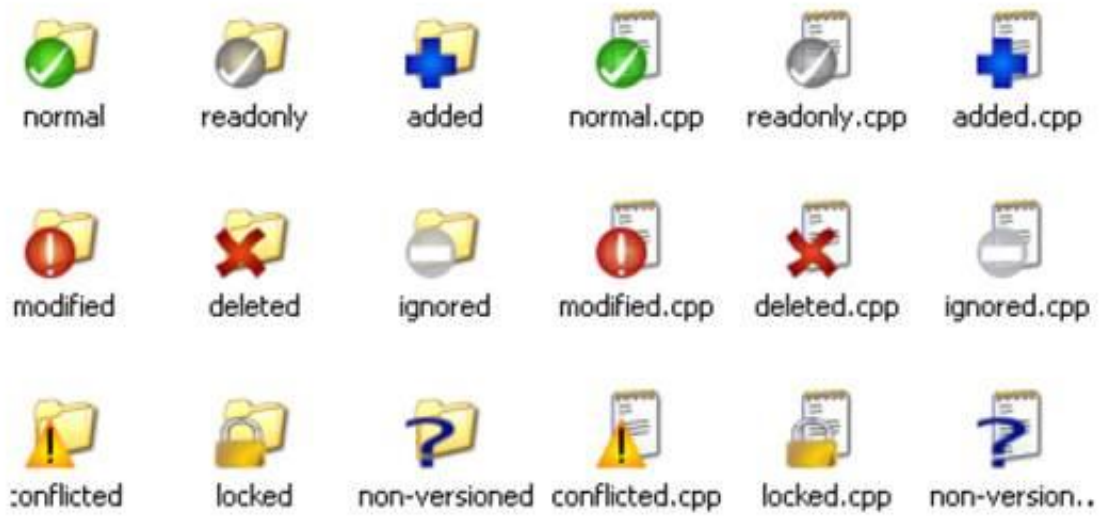


Рис.1.1 Пометки на значках в системе SVN

Теперь, после извлечения рабочей копии из хранилища Subversion, вы можете заметить, что значки в Проводнике Windows немного изменились, и это одна из причин такой популярности TortoiseSVN. TortoiseSVN добавляет так называемую пометку на значке для каждого файла, которая накладывается на исходный зна-

чок файла. Пометки различаются и зависят от статуса файла в Subversion.

– В свеже-извлечённой рабочей копии все пометки выглядят как зеленая галочка. Это означает, что статус Subversion – нормальный.

– Файлы были изменены с момента последнего обновления \ рабочей копии и нуждаются в фиксации.

– В процессе обновления возник конфликт (пометка-желтый восклицательный знак).

– Для файла установлено свойство svn:needs-lock, Subversion (файл как доступный только для чтения). Пометка на файлах означает, что вы должны заблокировать файл перед тем, как начнёте его редактировать.

– Вы владеете блокировкой на файл и его статус в Subversion нормальный. Вы должны разблокировать файл, если вы его не используете, чтобы и другие могли зафиксировать свои изменения в этом файле.

– Пометка папки – некоторые файлы или папки внутри текущей папки запланированы

– Символ плюс – файл или папка запланированы для добавления под управление версиями.

– Символ минус – файл или папка игнорируется системой управления версиями. Это необязательная пометка.

– файлы, не находятся под управлением версиями, но в то же время не являются игнорируемыми (необязательная пометка).

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ

Задание №1

1. ***Создание хранилища (репозитория) SVN***

1.1. На одном из локальных дисков создайте папку, в которой в дальнейшем будет находится репозиторий SVN для проекта. Лучше, чтобы эта папка находилась на достаточно надежном носителе. Пусть это будет папка E:\MyRepository\MyProject.

1.2. Нажмите правой кнопкой мыши на созданной папке и выберите команду TortoiseSVN\Create repository here.

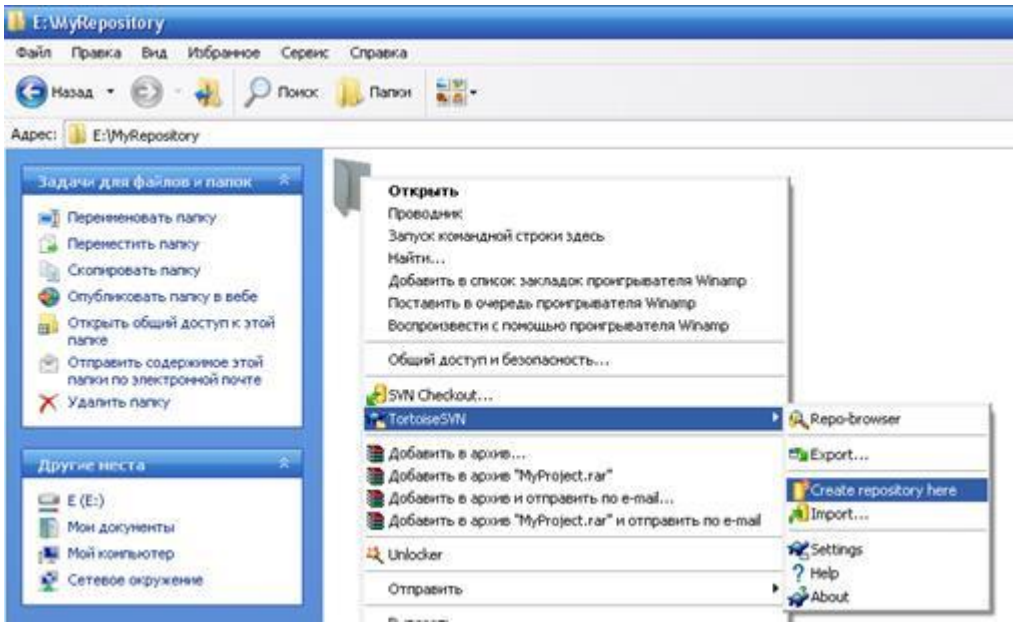


Рис.1.2 Создание репозитория

1.3. Появится окно с сообщением, что репозиторий успешно создан. После чего нажмите кнопку ОК.

1.4.

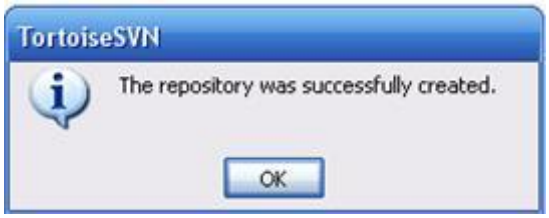


Рис.1.3 Окно сообщения об успешном создании хранилище

1.5. Далее появятся файлы как показано ниже на рисунке/

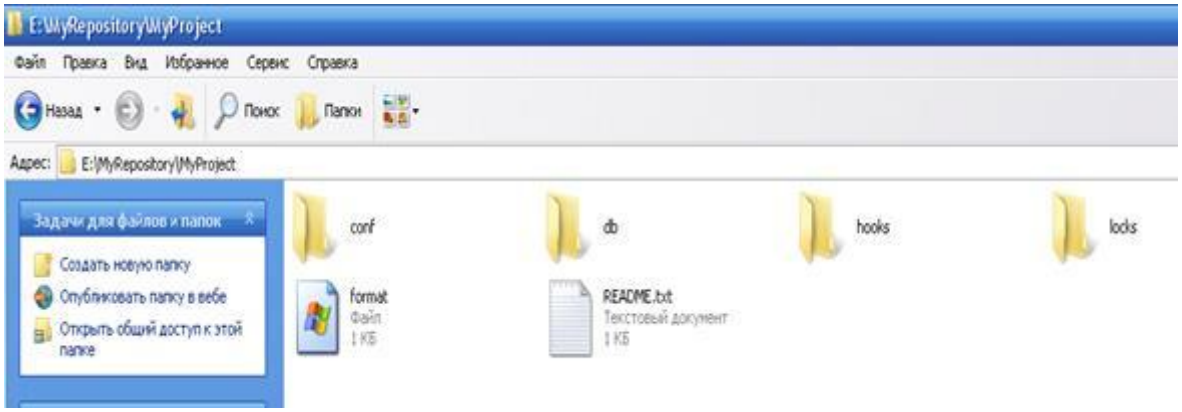


Рис.1.4 Структура репозитория

Примечание: Хранилище будет создано внутри новой папки. Не редактируйте эти файлы самостоятельно!!! В случае возникновения ошибок убедитесь, что папка пуста и доступна для записи.

2. Импорт данных в хранилище

Предполагая, что хранилище уже у вас есть, и вы желаете добавить в него новую папку со всей её структурой, просто выполните следующие шаги:

2.1. При помощи обозревателя хранилища (TortoiseSVN → Repo-browser) создайте новую папку проекта под именем **123** непосредственно в хранилище.

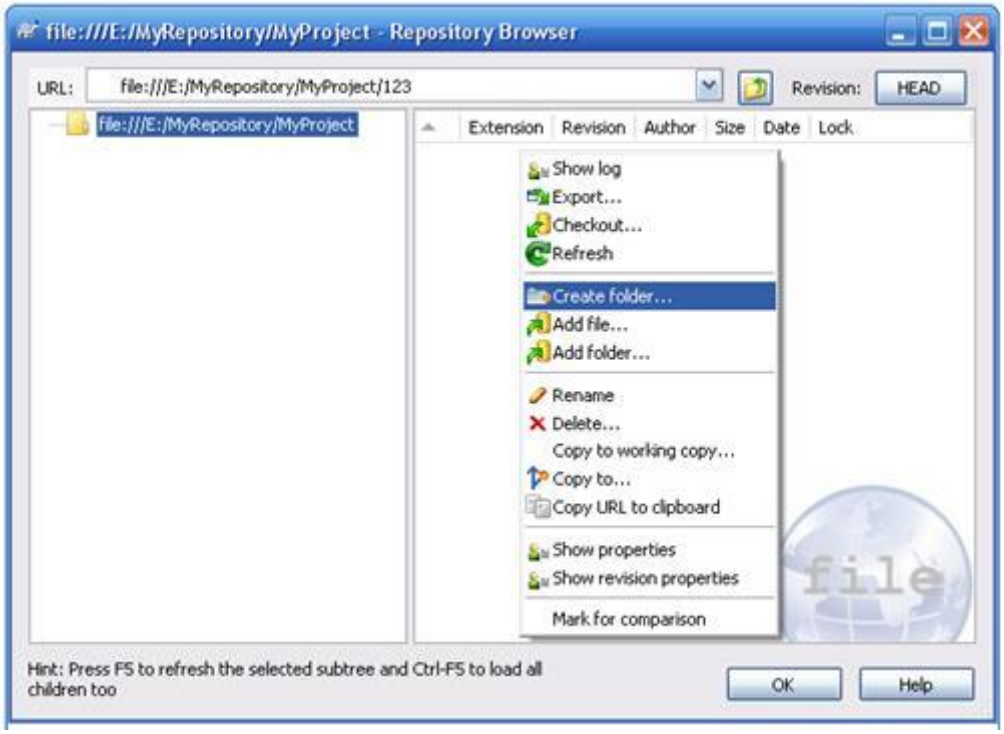


Рис.1.5 Обозреватель хранилища

2.2. Откройте общий доступ хранилище вкладе Доступ пометив флажком в Открыть общий доступ к этой папке и Разрешить изменение файлов в сети.

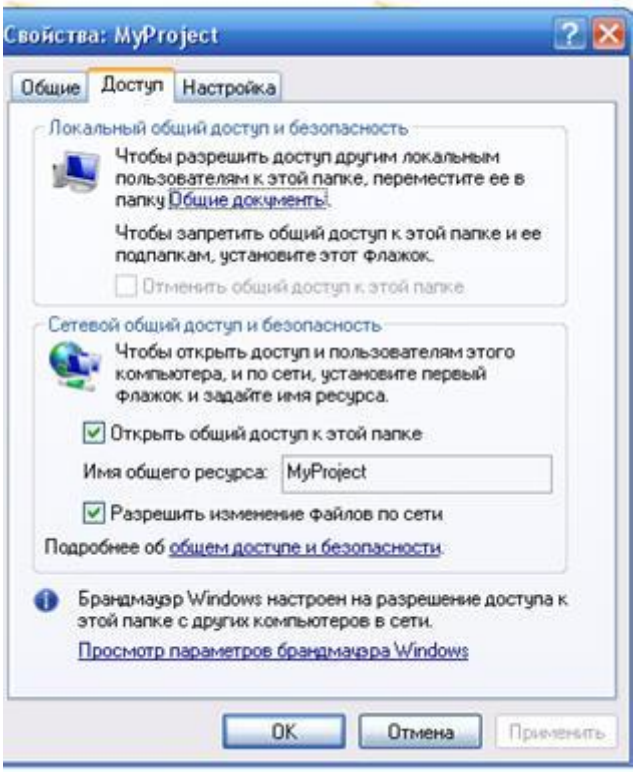


Рис.1.6 Открытие общего доступа к репозиторий.

3. Извлечение рабочей копии

3.1. Для получения рабочей копии вам надо произвести извлечение из хранилища. Выберите в Проводнике Windows папку, в которой хотите поместить вашу рабочую копию. Сделайте правый щелчок для вызова контекстного меню и выберите команду TortoiseSVN -> Извлечь ...,(TortoiseSVN → Checkout...,)

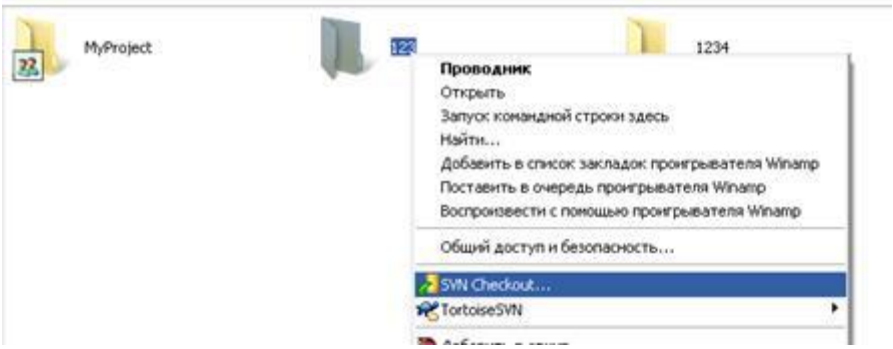


Рис.1.7 Меню для извлечения рабочей копии

3.2. Появившемся диалоговом окне Checkout в поле URL of Repository вставьте путь к репозиторий и нажимаем OK, далее еще раз нажимаете OK. Для того что вставить выделите репозиторий – папку и с помощью правой кнопки мыши откройте обозревателя

хранилища (TortoiseSVN → Repo-browser) в поле URL скопируйте путь. В поле Checkout directory автоматически появляется путь к рабочей копии, т. е. к папке.

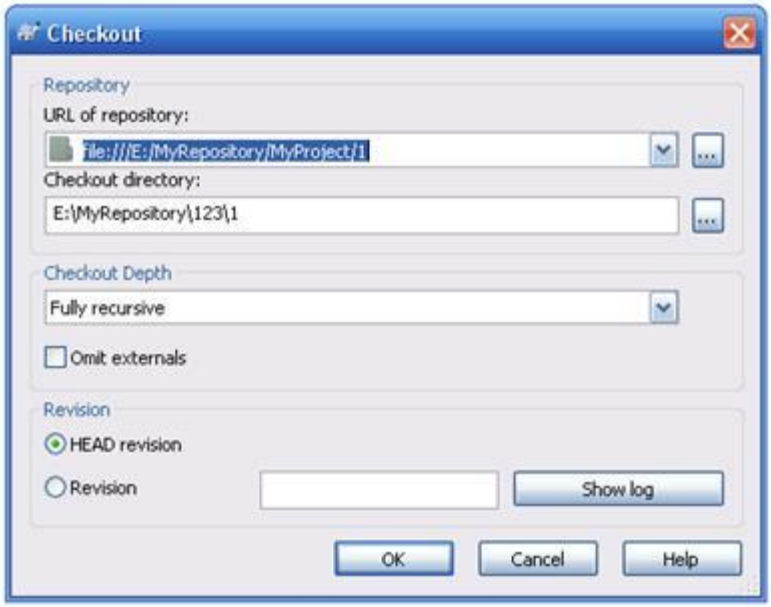


Рис.1.8 Путь к репозиторий

3.3. Выделите рабочую копию, т. е. папку под именем **123** и нажмите F5, либо правой кнопкой мыши нажмите обновить.

Рис.1.9 Свеже-извлечённая рабочая копия

3.4. Повторите действия 3.1–3.3 для рабочей копии **1234**.

Задание №2

1. Аналогично проделать те же действие, но проделав эти же действие на двух разных компьютерах.

2. В одном компьютере например, комп1 создаете репозиторию и одну рабочую копию, т. е. папку под именем **123.**

3. Во втором компьютере например, комп2 создаете только рабочую копию, т. е. папку под именем **1234.**

4. Во втором компьютере соедините рабочую копию с репозиторием указывая путь file:///ip-компьютера/название_репозитория.

5. В комп1 создайте любой файл и отправьте его на комп2. Таким образом проделав 5 ревизий.

6. Чтобы узнать IP- компьютера нажмите пуск →выполнить→ cmd, либо
C:\Windows\System32\cmd.exe /k %windir%\system32\ipconfig.exe

7. Откроется окно командой строки. Введите ipconfig, где вы увидите примерное IP- компьютера: 192.168.8.xxx

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Хранилище (Repository)?
2. Что такое Рабочая копия (Working Copy)?
3. Что такое версия?
4. Понятия управления контроля версиями, применение контроля версий.
5. Чем помогают системы контроля версий типа SVN?

РАБОТА №2 ЛАБОРАТОРНАЯ. ЭКСПОРТ НАСТРОЕК В КОМАНДНОЙ СРЕДЕ РАЗРАБОТКИ

1. ЦЕЛИ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков по выполнению экспорта настроек командной среды разработки.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Современные интегрированные среды позволяют вести командную разработку проектов. Такие возможности предоставляет среда Visual Studio/

Точно настроив интегрированную среду разработки по своему вкусу, вы можете сохранить эти настройки на будущее. Для этого можно экспортировать настройки интегрированной среды в файл или даже передать ряду инсталляций системы Visual Studio, чтобы во всех установленных системах были одинаковые настройки.

Для того чтобы экспортировать выбранную конфигурацию, необходимо выбрать команду *Tools --> Import and Export Settings*, чтобы запустить мастер Import and Export Settings Wizard, как показано на рисунке ниже. На первом этапе работы этого мастера следует выбрать настройку Export, а также настройки, которые следует сохранить при выполнении процедуры экспорта.

Рис.2.1 Мастер Import and Export Settings Wizard.
Порядок выполнения работы

Как показано на рисунке, можно экспортировать множество сгруппированных настроек. На этом снимке экрана видно, как раскрывается раздел Options, демонстрируя, что следует сохранить настройки Debugging и Projects, а также конфигурации Text Editor и Windows Forms Designer. Маленькие пиктограммы с восклицательным знаком свидетельствуют о том, что некоторые настройки не были предназначены для экспорта по умолчанию, поскольку они содержат информацию, которая может нарушить конфиденциальность информации. В этом разделе вы должны сделать свой выбор вручную, если действительно хотите, чтобы эта информация была сохранена в резервном файле. Выбрав настройки, которые хотите экспортировать, переходите к следующему этапу работы мастера, который может занять несколько минут, в зависимости от того, сколько настроек вы экспортируете.

Импортировать файл настроек очень легко. Для этого используется тот же самый мастер, но теперь на первом этапе вам следует выбрать команду Import. Вместо простой перезаписи текущей кон-

фигурации мастер позволяет вам сначала сохранить резервную копию текущих настроек.

Затем можете выбрать существующий файл конфигурации из списка. Это тот же самый список файлов, из которого вы выбираете настройки при первом запуске системы Visual Studio 2013. Кроме того, можете просмотреть файлы настроек, созданные вами заранее. Выбрав файл настроек, можете импортировать только разделы конфигурации или всю ее целиком.

Мастер по умолчанию исключает несколько разделов, таких как External Tools или Command Aliases, чтобы вы не могли непреднамеренно уничтожить пользовательские настройки. Убедитесь, что вы выбрали эти разделы, если хотите восстановить все настройки.

Если вы просто хотите восстановить одну из конфигураций системы Visual Studio 2013, заданных по умолчанию, выполните команду Reset All Settings на первом этапе работы мастера и не выполняйте весь процесс импорта.

Visual Studio предоставляет возможность поделиться настройками с членами команды, с которой вы работаете. Это полезно в тех случаях, например, когда происходит редактирование одних и тех же файлов (используя Team Foundation Server). Каждый программист по своему оформляет код, использует символы табуляции и т. д. Когда разные члены команды работают с одним файлом, простое редактирование файла может привести к незначительным изменениям, не влияющим на работу приложения (лишние пробелы, переносы строк и т. д.) Однако, когда эти файлы добавляются в репозиторий исходного кода (TFS, Git, ...), эти изменения могут вызвать проблемы.

Если вы работаете с командой разработчиков, создание единого файла настроек является хорошей идеей. В разделе настроек Environment --> Import and Export Settings вы можете установить флажок Use Team Settings File.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создайте проект в среде Visual Studio.
2. Задайте параметры работы среды Visual Studio.
3. Выполните экспорт настроек в файл обмена, как было рассмотрено в теоретической части.

4. Импортировать настройки из файла, в который выгружены настройки с другими вариантом задания.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие настройки могут быть установлены для среды Visual Studio, влияющие на особенности командной работы?
2. Как выполняется экспорт настроек?
3. Как выполняется импорт настроек?

РАБОТА №3 ПРАКТИЧЕСКАЯ. СРАВНИТЕЛЬНРЫЙ АНАЛИЗ ОФИССНЫХ ПАКЕТОВ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является:

- получение офисных пакетов Microsoft и OpenOffice;
- изучение принципов и получение практических навыков;
- изучение содержания офисных пакетов;
- получение практических навыков выполнения основных операций в среде офисных пакетов MS и OO;
- получение навыков выполнения анализа.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Пакет Microsoft Office является, пожалуй, самым распространённым программным обеспечением, устанавливаемым на домашние и офисные компьютеры. Однако данные пакет является платным.

Именно этот факт заставляет многих пользователей искать бесплатную альтернативу. Одним из самых развитых альтернативных пакетов является OpenOffice.

Существует мнение, что если речь идет о замене платного ПО бесплатными аналогами, то приходится чем-то жертвовать. В случае с OpenOffice.org это далеко не так. Мощный офисный пакет вполне может потягаться с MS Office.

OpenOffice, по аналогии с Microsoft Office, состоит из нескольких программ, которые и составляют пакет. Тем не менее, в то время как продукт от Microsoft поставляется во множестве модификаций (Standart, Professional, Enterprise и так далее), OpenOffice доступен только в одной версии. Первый является платным решением, и далеко не всем пользователям нужны все те функции, которые он предлагает. По этой причине и сделано такое разделение, чтобы пользователь сам мог выбрать, за какие функции он согласен платить.

Самыми известными приложениями Microsoft Office являются следующие:

Word (текстовый редактор);

Excel (электронные таблицы);

Access (база данных);

PowerPoint (электронные презентации);

Outlook (почтовый клиент, органайзер).

Именно они и продублированы в OpenOffice. Естественно, их названия изменены:

Writer (текстовый процессор);

Calc (электронные таблицы);

Base (база данных);

Impress (электронные презентации).

Аналога в OpenOffice нет только у Outlook. Вместе с рассматриваемым пакетом программ поставляются приложения под названием Draw и Math. Предназначение первого – это создание изображений, а второго – различных формул. По большому счёту, Draw содержит в себе все те функции, которые равномерно распределены в других компонентах OpenOffice. Например, здесь можно нарисовать какие-либо объекты, используя инструменты векторной графики, а также сделать диаграммы.

Что касается Math, то аналогичный инструмент из Microsoft Office носит название Microsoft Equation. Вкратце его функционал мы рассмотрим позже во второй статье вместе с обзором Writer (Math будет наиболее полезен как раз в качестве приложения к текстовому редактору).

Оба пакета максимально комфортно работают на операционной системе Windows 2000 года и выше. Установятся и на менее мощную систему, однако скорость работы и обработки данных будет занимать длительное время, да и от сбоев никто не застрахован. А установка стандартная. Фактически однотипные окна с последовательными нажатиями «Далее» и «Готово», с выборами установки приложений и вызовом программы двойным щелчком. В требованиях и установке отличия OpenOffice от Microsoft office минимальны.

Главные особенности пакетов для пользователя начинаются с их скорости работы. Здесь МО нет равных. Документ МО открывается в 3 раза быстрее, чем документ ОО. Причем при сравнении документов незначительного объема содержащих текст. В пределах своего пакета открытие файлов примерно одинаково. А связано это с тем, что ОС Windows не является родной для ОО, а потому ей каждый раз при вызове необходимо подгружаться в систему и биб-

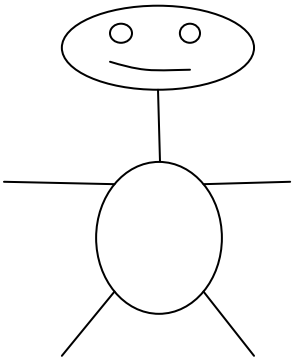
лиотеки, а это занимает время. Это еще одно сравнение OpenOffice и Microsoft Office, которое влияет на оперативность работы.

К одной из особенностей OpenOffice относится наличие глобальных настроек, которые действительны и одинаковы для всех приложений пакета. Они могут быть заданы в любой из шести программ. Все настройки сгруппированы в виде структуры «дерево».

Теперь несколько слов о совместимости OpenOffice и Microsoft Office, ведь при работе с офисными пакетами важно, чтобы документы, созданные в том или ином пакете, могли без сбоев открываться в разных. То, что файл doc корректно прочитается любой версией МО это ясно, аналогично и с ОО. Но если создавать файл в ОО, а открывать его в МО, возникнут недоразумения. С таблицами или редактированием текста в ОО придется потрудиться, многие функции несовершенны, требуют дополнительного пользовательского вмешательства. И затем то, что создано в ОО в МО открывается со сбоями. Тогда как документ МО в ОО открывается без изменений. Дело в том, что МО обладает более мощным функционалом и возможностями, в этом пакете множество замысловатых и простейших функций, которые позволяют максимально упрощать работу с пакетом. В ОО большинство из них тоже есть, однако их необходимо все время настраивать.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать документ отображающий содержание практической работы в среде MS Word.
2. Создать таблицу с данными о составе лабораторных работ, их продолжительности.
3. Создать рисунок аналогичный приведённому ниже.
4. Сохранить документ в формате «.doc».
5. Открыть документ в среде OpenOffice.
6. Сделать вывод о полученном результате. Вывод должен передавать сохранность вида текста, таблицы, рисунка.
7. Выполнить аналогичную работу в среде OpenOffice.
8. Сделать вывод о полученном результате.



Это я))

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Особенности пакета MS Offic.
2. Особенности пакета OpenOffice.
3. Какие составляющие пакета MS Office?
4. Какие составляющие пакета OpenOffice?

РАБОТА №4 ПРАКТИЧЕСКАЯ. СРАВНИТЕЛЬНЫЙ АНАЛИЗ БРАУЗЕРОВ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Получение практических навыков анализа возможностей программных систем на примере трех известных браузеров Google Chrome, Opera, Яндекс Браузер, Mozilla Firefox.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Рис.4.1 Логотипы наиболее распространённых браузеров

Браузер начинают оценивать с удобства его интерфейса. Среднестатистический пользователь интернета проводит в нем достаточно много времени, поэтому удобство перемещения по страницам и использования прочих функций браузера имеет первостепенное значение.

• ***Google Chrome***, как и большинство продуктов одноименной компании, следует принципам минимализма. В данном браузере вы не найдете лишних кнопок и меню, в которых очень просто запутаться. Все те функции, которые большинству пользователей не нужны, удобно спрятаны. Если в них возникнет необходимость, их можно вытащить, поставив галочку в настройках.

• ***Яндекс Браузер*** весьма схож с предыдущим участником сравнения (еще бы, ведь используется аналогичный движок), но имеет и некоторые улучшения, которые пока еще не были введены у конкурентов. Отдельного внимания заслуживает интерфейс мобильной версии браузера, который был начисто избавлен от перегрузки вкладками и меню – на виду было оставлено только самое необходимое.

• *Опера* при первом знакомстве с ней может показаться запутанной, ведь разработчики включили в стандартную панель все мыслимые и немыслимые функции. Некоторые из них, разумеется, найдут свое применение. Например, встроенная экспресс-панель, позволяющая быстро переходить на любимые страницы (аналог закладок, но более практичный). Но остальную часть неиспользуемых функций придется убирать вручную. Помимо облегчения навигации, это может оказать положительное влияние на производительность.

• ***Mozilla Firefox*** в своей базовой комплектации довольно простой и незамысловатый. Присутствует ряд классических кнопок навигации, расположенных в привычных местах. Дополнительные функции и темы оформления устанавливаются пользователями самостоятельно.

Производительность.

Производительность браузера напрямую влияет на скорость загрузки страниц и использование ресурсов компьютера. Вряд ли пользователям может понравиться тот факт, что им приходится ждать дополнительные несколько секунд каждый раз, как они перелистывают страницу, или терпеть систематические зависания системы.

• Обработка страниц в браузере Google Chrome производится высокотехнологичным движком Javascript 8, позволяющем загружать анимации и прочие сложные элементы практически мгновенно. С другой стороны, браузер достаточно требователен к оперативной памяти, быстро заполняя ее. Если хотите пользоваться данным браузером без затруднений, открывая по 5–7 и больше вкладок, в вашей системе должно быть от 1 гигабайта оперативной памяти и выше.

• Создателями Яндекс Браузера с самого начала было выбрано правильное направление развития, заключающееся в отказе от ненужных функций в пользу увеличения производительности. Скорость загрузки страниц тут идет наравне с Chrome, а иногда даже показывает лучший результат. Отчасти это заслуга технологии «Turbo».

• Фанаты Оперы начнут спорить, но факт остается фактом – браузер испытывает проблемы с производительностью. Некоторые скрипты обрабатываются достаточно медленно, аппаратное ускоре-

ние «ускоряет» недостаточно, а расширениями лучше не злоупотреблять, так как они довольно активно забирают ресурсы компьютера. Придется пару часов покопаться в настройках Оперы, дабы исправить все эти неприятные мелочи. В интернете полно руководств о том, как правильно оптимизировать данный браузер.

• Mozilla Firefox располагается на движке собственной разработки, именуемом Gecko, и написана на языке разметки XUL (тоже собственного производства). Эта причина повлекла за собой заметное замедление работы и загрузки страниц на маломощных системах, а также портативных устройствах, использующих версию браузера для настольных ПК. Оперативная память тоже потребляется в большом количестве. Хотя, если у вас компьютер современного образца, беспокоиться будет не о чем.

Дополнительный функционал

• Браузером Google Chrome останется доволен даже самый требовательный пользователь. Среди функций можно выделить тесную интеграцию со всеми сервисами Google, облачное хранение данных (закладки, расширения и прочее хранится на серверах компании, вам не имеет смысла заморачиваться с резервным копированием) и чтение документов в распространенных форматах. Для всего остального существуют бесплатные модули, которых в официальном хранилище несколько тысяч. Особого внимания заслуживает расширение AdBlock Plus, позволяющее избавиться от назойливых рекламных баннеров и насильственного перенаправления на неизвестные страницы.

• Yandex Browser имеет практически идентичный Google набор сервисов, но и уникальные функции у него тоже присутствуют. Например, функция автоматического перевода страниц с английского на русский язык. Или подключаемый через опции «быстрый звонок», позволяющий отыскивать размещенные на страницах номера мобильных телефонов и добавлять их в контакты за пару нажатий мыши. Особо удачно реализовано взаимодействие со смартфонами и прочими мобильными устройствами.

• Опера прежде всего примечательна своей встроенной экспресс-панелью, о которой мы уже говорили ранее. Чтобы получить аналогичную функцию, пользователи других браузеров должны будут установить специальное расширение или плагин. Людям с ограниченными возможностями или просто ленивым индивидам, не же-

лающим барабанить по клавишам, понравится подключаемое голосовое управление. Разумеется, оно пока еще далеко от идеала, но управлять браузером через него вполне реально.

• Функции Mozilla Firefox тоже заслуживают внимания. У браузера в базовом функционале имеется блокировка всплывающих окон, автоматический механизм живого поиска по словарям и многое другое. Если вы не можете найти какую-либо функцию, поищите ее в хранилище расширений. По количеству плагинов Mozilla обгоняет каждого из участников обзора.

Безопасность использования

Через браузеры мы получаем доступ к социальным сетям, электронным платежным системам и прочим ресурсам, где хранится наша личная информация и прочие данные, которые не должны попасть в чужие руки.

К сожалению, мошенники успешно преодолевают защиту, используя обнаруженные ими уязвимости, и проникают туда, где им быть запрещено.

Давайте посмотрим, какие меры в плане обеспечения безопасности предприняли создатели обозреваемых браузеров.

• Google серьезно подошла к обеспечению безопасности своего браузера. Обновления выходят регулярно и автоматически применяются, поэтому беспокоиться об уязвимостях не стоит. Также имеется своеобразная защита – браузер не даст вам просто так попасть на сайты, которые находятся в черном списке, или прочие подозрительные страницы. Даже загрузку файлов из интернета нужно будет дополнительно подтверждать нажатием на кнопку «ДА» (на тот случай, если закачка стартовала в результате срабатывания автоматического скрипта).

• Яндекс Браузер всегда оповестит пользователя, если посещаемый сайт будет иметь отношение к мошенничеству, хакерству или прочим противозаконным действиям. Встроенная утилита, разработанная «Kaspersky Lab», просканирует скачанные файлы на предмет наличия вредоносного кода и исполняемых скриптов. Безопасность и сохранность данных ложится на плечи технологии «Safe Browsing», которая положительно себя зарекомендовала еще на этапе тестирования.

• Опера будет идеальным вариантов для людей, которые хотят скрыть свою активность в сети интернет. Браузер отлично работает

в связке с Tor, что делает весьма затрудненным определение вашего текущего местоположения. Протоколы шифрования SSL и TLS не дадут перехватить передаваемую вами информацию. Что касается появляющихся из-за ошибок в коде уязвимостей, то они очень быстро устраняются стараниями пользователей (любой человек, обнаруживший какую-либо брешь, может отправить пользовательский отчет).

• Mozilla, как и Яндекс Браузер, пользуется технологией «Safe Browsing», позволяя обеспечить частичную защиту пользователей от опасных страниц, содержащих вредоносный код, а также мошеннических сайтов. Даже несмотря на то, что проект Mozilla является свободным программным обеспечением с открытым исходным кодом, защита находится на приемлемом уровне.

По результатам оценивания был выбран несомненный фаворит – браузер Google Chrome. Он гарантирует быструю, удобную и безопасную работу большинству пользователей. Остальные браузеры использовать рекомендуется лишь в конкретных случаях.

Например, Яндекс Браузер без расширений идеально подойдет для владельцев маломощных ПК и мобильных систем. Опера поможет обеспечить частичную анонимность и не будет никуда передавать ваши данные (тогда как Google Chrome и другие проекты на базе этой разработки неоднократно таким грешили). А Mozilla понравится тем, кому нужен действительно широкий функционал – по числу дополнений и расширений ему нет равных.

В принципе, любой из этих браузеров с основными задачами справляется. Можете смело выбирать понравившийся, только не вздумайте пользоваться стандартным решением от Windows – Internet Explorer. Браузер оказался настолько неудачным проектом, что его поддержку Microsoft решила прекратить. Медленный, с ограниченной функциональностью и с кучей багов.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Загрузите страницу https://docs.microsoft.com/ru-ru/aspnet/mvc/pluralsight с помощью браузера Google Chrome.
2. Выполните функции:

a. чтения теста в слух;
b. перевода;

c. просмотра видео.

3. Выполните данные функции, если возможно, с помощью браузеров Mozilla, MS Edge.

4. Отобразите на диаграмме вариантов использования функции, выполняемые браузерами.

5. Сделайте вывод по особенностям выполнения функций.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначены браузеры, какова их основная функция?

2. Какие дополнительные функции могут выполнять браузеры?

3. В чём особенность различных браузеров?

РАБОТА №5 ПРАКТИЧЕСКАЯ. СРАВНИТЕЛЬНЫЙ АНАЛИЗ СРЕДСТВ ПРОСМОТРА ВИДЕО

1. ЦЕЛЬ РАБОТЫ

Целью работы является получение практических навыков работы со средствами просмотра видео и их сравнительного анализа.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Видеоплееры отличаются функционалом, внешним видом (для многих можно менять так называемые «скины»), предназначением, они подходят для разных операционных систем: здесь есть плееры для Windows 10, 8, 7 и более старых.

Даже если вы ранее не задумывались, какой видео проигрыватель используется в вашей системе и вас устраивал старый добрый Windows Media, установленный по-умолчанию, то рано или поздно приходит тот самый момент, когда вопрос о замене плеера станет актуальным.

И этот момент настанет, как только вы попытаетесь изменить пропорции видео, приблизить изображение в проигрываемом фильме, сделать субтитры более читабельными на экране, либо добавить им прозрачности, либо вообще при первом глюке топорного детища от компании Microsoft.

Ведь, как показывает практика, Windows Media годится для просмотра простеньких видео либо самых обычных avi файлов, но при этом его возможностей становится явно недостаточно даже для домашнего пользователя, просматривающего хотя бы несколько современных качественных фильмов пару раз в месяц.

	ComboPlayer	KMPlayer	Media Player Classic Home Cinema
Лицензия	Бесплатная	Бесплатная	Бесплатная
Стоимость	Бесплатная	Бесплатная	Бесплатная
Русский язык	Да	Да	Да
Поддержка macos	Да	Да	Да
Рейтинг	10	10	10
Потоковое воспроизведение	Да	Да	Да
Темы оформления	Да	Да	Да
Визуализация	Да	Да	
Удаленное управление	Да	Да	Да
Поддержка видео	Да	Да	Да
Встроенный менеджер загрузок	Да		
Проигрывание во время скачивания torrent	Да		
Рейтинг			

Все плейеры поддерживаются аудиформаты MP3, WMA, RealAudio, AAC, AC-3, FLAC, ALAC

ComboPlayer – программа для тех, кому надоело устанавливать кучу софта вместо одного легкого многофункционального инструмента. Это плеер аудио, видеофайлов, онлайн ТВ, радио, потоковый проигрыватель, менеджер загрузок и утилита просмотра изображения с ip-камеры в одном флаконе. Что приятно, «комбайн» полностью бесплатный, быстро работает даже на самых слабых устройствах.

Главные особенности ComboPlayer:

воспроизведение аудио- и видеоформатов MPEG-2, MPEG-4, H.264, MKV, FLV, WMV, MP3, 3GP, WEBM и прочих;

проигрывание .torrent файлов во время скачивания;

выбор трансляции телевизионных программ по категориям;

функция прослушивания радио, большое количество доступных радиостанций;

воспроизведение потокового видео с веб-камеры, ТВ и видеонаблюдения;

синхронизация файлов, списков воспроизведения между разными Windows ПК через учетную запись;

Тонкая настройка параметров программы и пользовательской медиатеки;

требуется минимум настроек под себя для комфортного использования;

способность считывать метаданные и субтитры;

интегрированные медиа-кодеки для повышения производительности на фоне аналогов.

На рынке нет другого бесплатного приложения с подобными функциональными возможностями. У ComboPlayer только два небольших недостатка: отсутствие поддержки ОС Windows XP и требование регистрации для синхронизации медиатеки, просмотра онлайн ТВ. При этом, плата за пользование услугой и базовый пакет каналов не взимается.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ ДЛЯ РАБОТЫ

1. Найдите ресурсы для загрузки установочных компонентов для рассматриваемых видеоплейеров. Опишите ресурс и особенность предоставления компонентов, их размер, тип файлов
2. Установите видеоплейеры.
3. Выполните просмотр одного и того же файла в различных плейерах. Сделайте вывод о качестве отображения, звука, наличия «зависания», дефектов изображения, звука.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Содержание модели отображающей требования к системе.
2. Каким образом определяются функциональные требования к системе?
3. Что отображает диаграмма «вариантов использования», каковы её элементы, какие между ними могут быть связи?
4. Как описывается содержание сценария выполнения функционального требования?
5. Как определяется состав и содержание экранных форм, необходимых для реализации сценария?

РАБОТА №6 ПРАКТИЧЕСКАЯ. ОБРАТНОЕ ПРОЕКТИРОВАНИЕ АЛГОРИТМА

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических выполнения обратного проектирования.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Прямым проектированием (Forward engineering) называется процесс преобразования модели в код путем отображения на некоторый язык реализации. Процесс прямого проектирования приводит к потере информации, поскольку написанные на языке UML модели семантически богаче любого из существующих объектно-ориентированных языков. Фактически именно это различие и является основной причиной, по которой мы, помимо кода, нуждаемся и в моделях. Некоторые структурные свойства системы, такие как кооперации, или ее поведенческие особенности, например взаимодействия, могут быть легко визуализированы в UML, но в чистом коде наглядность теряется.

Обратным проектированием (Reverse engineering) называется процесс преобразования в модель кода, записанного на каком-либо языке программирования.

В результате этого процесса вы получаете огромный объем информации, часть которой находится на более низком уровне детализации, чем необходимо для построения полезных моделей. В то же время обратное проектирование никогда не бывает полным. Как уже упоминалось, прямое проектирование ведет к потере информации, так что полностью восстановить модель на основе кода не удастся, если только инструментальные средства не включали в комментариях к исходному тексту информацию, выходящую за пределы семантики языка реализации. Пример, представленный на рис. 3.3, был создан с помощью обратного проектирования библиотеки классов языка Java.

Процесс обратного проектирования делится на два этапа: анализ и генерацию модели.

На первом этапе производятся все подготовительные операции по анализу текста программы на отсутствие синтаксических ошибок. Второй этап – преобразование кода в модель.

Все операции выполняются независимо, что дает больший маневр для разработчика, который, например, хочет провести только синтаксический разбор теста, без генерации модели.

Соответственно при отсутствии ошибок в файле можно приступить к генерации модели. В целях оптимизации времени генерации в предусмотрено три способа проведения обратного проектирования, каждый из которых может охватить и превосходно выполнить определенный сегмент работ:

- FirstLook – приближенная пробежка по телу программы.
- Detailed Analysis – детальный анализ проекта.
- RoundTrip – комбинация двух вышеперечисленных способов. Позволяет безболезненно строить и перестраивать разрабатываемые приложения по принципу круговой разработки.

Нашей целью будет получение графической модели из класса на языке программирования. Обратите внимание на комментарии. Каждая строка снабжена комментарием. Смысл обратного проектирования состоит не только в том, чтобы корректно нарисовать модель, но и для правильного описания спецификации каждой составляющей класса. За основу программы возьмем следующий класс:

```
//It's main class class string public:
char *string; //Structure's pointer
int buffer[100]; //Temporary buffer
char name[10]={«Massiv»}; //Name of data
int a; //Integer
int b; //Integer void string(void); //constructor
void ~string(void); //destructor
char StringCopy(char *, //Buffer char *, //source1 char *); //source2 private:
int tmp_a;
int tmp_b;
```

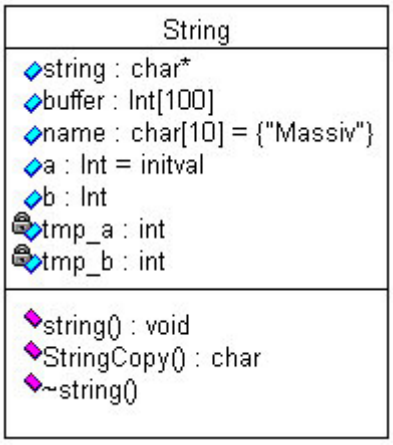


Рис.6.1 Класс, полученный в результате обратного проектирования

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ ДЛЯ РАБОТЫ

Заданием для работы является исходный код, содержащий описание классов. Классы должны содержать операции и их реализацию.

Выполнение должно включать следующие этапы.

1. Анализ кода и выделение состав классов.
2. Отображение классов в среде моделирования. Для каждого класса в отчёте по работе должно быть сделано пояснение – на основе какого программного объекта он выявлен.
3. Выявление переменных классов и отображение их в среде моделирования. Для каждой переменной класса в отчёте по лабораторной работе должно быть сделано пояснение.
4. Выявление операций классов. Для каждого класса необходимо выявить операции и сделано пояснение о свойствах операции, её параметрах и уровне в иерархии наследования.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что из себя представляет обратное проектирование?
2. Для чего выполняется обратное проектирование?
3. Что из себя представляет результат обратного проектирования?
4. Какие модели можно построить в результате обратного проектирования?
5. Существуют ли инструментальные средства для обратного проектирования?

РАБОТА №7 ЛАБОРАТОРНАЯ. НАСТРОЙКИ ДОСТУПА К РЕПОЗИТОРИЮ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков работы с депозитарием проекта Github.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Git – распределённая система управления версиями. Проект был создан для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

GitHub – крупнейший[7] веб-сервис для хостинга IT-проектов и их совместной разработки.

Среди проектов, использующих Git – ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

Программа является свободной и выпущена под лицензией GNU GPL версии 2. По умолчанию используется TCP порт 9418.

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).

В первую очередь надо установить клиент git: обязательно потребуется консольный клиент, доступный по ссылке http://git-scm.com/downloads (поддерживаются основные ОС), графический клиент можно установить по желанию, исходя из своих предпочтений.

Далее работа с git будет объясняться на примере работы с консольным клиентом по следующим причинам:

- Чтобы складывалось понимание происходящего и при возникновении проблем вы могли четко объяснить, что вы делали, и было видно, что пошло не так.
- Все нажатия кнопок в графических клиентах в итоге сводят к выполнению определённых команд консольного клиента, в то же время возможности графических клиентов ограничены по сравнению с консольным
- У тех, кто будет работать в классе на стоящих там компьютерах, не будет другого выбора, кроме как пользоваться консольным клиентом (на сколько мне известно, никаких графических клиентов для git там не установлено)

Аккаунт и репозитории на github.com. Для того что бы использовать web-сервис репозитория необходимо зарегистрироваться на https://github.com/. После чего можно будет создавать свои репозитории или присоединиться к работе над проектами коллег, сделав копию (fork) другого репозитория. Вам предлагается начать с создания fork-а к заведенному репозиторию
https://github.com/andreiled/mipt-cs-4sem.

Создание локального репозитория, связанного с удаленным репозиторием. Следующим шагом после создания репозитория на github, называемого далее удаленным репозиторием, является создание локальной копии этого репозитория на своем компьютере. Особенностью git является наличие на локальном компьютере полной копии репозитория со всей информацией об истории изменений.

1. Открываем консольный клиент.

- На Windows после установки клиента появляется пункт Git Bash в контекстом меню папки. Достаточно перейти в желаемую папку и воспользоваться этим пунктом меню.
- На Unix системах достаточно открыть терминал и перейти в нужную директорию. При стандартной установке консольного клиента будет доступна команда git без дополнительных усилий.

2. Выполняем команду git clone https://github.com/%user_login%/%repo_name%.git. Полную https ссылку на репозиторий для его выкачивания можно также найти на странице самого репозитория на github. После этого в текущей папке появится новая папка с именем %repo_name%, содержащая копию удаленного (remote) репозитория.

3. Переходим в созданную папку репозитория и настраиваем его:

4. git config user.name ivan.ivanov
5. git config user.email ivanov@example.com

Внесение и оформление изменений в локальном репозитории. Воспользовавшись командой git status можно узнать, на какой ветке (branch) репозитория вы сейчас находитесь, какие изменения присутствуют в вашей рабочей копии и другую информацию.

Рабочей копией называется совокупность файлов в локальной папке репозитория за исключением служебных файлов.

После внесения каких-либо изменений в рабочую копию их можно «закоммитить» в локальный репозиторий:

- сначала нужная часть изменений подготавливается к коммиту с использованием команды git add %file_path%
- после чего производится коммит командой git commit Использование команды без аргументов откроет текстовый редактор, где надо будет написать комментарий для коммита, коммит обязательно должен иметь комментарий. Другим вариантом задания

комментария к коммиту является использование команды git commit -m «%commit_message%»

Историю изменений можно посмотреть командой git log или git log --name-only. Если вся история изменений не умещается на экране, то можно пользоваться клавишами прокрутки на клавиатуре («стрелочки», PgUp, PgDown), выход из режима просмотра изменений осуществляется нажатием клавиши «q».

Загрузка локальных изменений в удаленный репозиторий. После того, как были выполнены нужные локальные коммиты, изменения можно загрузить в удаленный репозиторий с помощью команды git push origin master. GIT клиент при этом запросит имя пользователя и пароль для доступа к github.

Выполнение этой команды может закончиться с ошибкой, если в локально репозитории отсутствуют последние изменения, имеющиеся в удаленном репозитории. Для решения этой проблемы надо выполнить команду git pull, которая скачает последние изменения из удаленного репозитория и смержит их с вашими локальными правками, после чего можно повторить команду git push.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ ДЛЯ РАБОТЫ

1. Зарегистрируйтесь на сервисе https://github.com/.
2. Создайте локальный репозитарий, связанный с заданным глобальным репозитарием.
3. Настройте параметры доступа к локальному репозитарию.
4. Выполните загрузку изменений выполненных в локальном репозитарии в глобальный репозитарий.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие функции выполняет система контроля версиями?
2. Что такое локальный депозитарий?
3. Какие функции выполняет сервис **GitHub**?
4. Какие действия нужно выполнить для получения доступа к функционалу **GitHub?**
5. Каким образом изменения в локальном репозитарии GIT можно передать в глобальный репозитарий **GitHub?**

РАБОТА №8 ПРАКТИЧЕСКАЯ. ПЛАНИРОВАНИЕ CODE-REVIEW

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью выполнения работы является получение практических навыков подготовки среды совместной разработки для проведения инспекции программного года.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Просмотр кода (англ. code review) или инспекция кода (англ. code inspection) – систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. Целью просмотра является улучшение качества программного продукта и совершенствование навыков разработчика.

В процессе инспекции кода могут быть найдены и устранены такие проблемы, как ошибки в форматировании строк, состояние гонки, утечка памяти и переполнение буфера, что улучшает безопасность программного продукта. Системы контроля версий дают возможность проведения совместной инспекции кода. Кроме того, существуют специальные инструментальные средства для совместной инспекции кода.

Программное обеспечение для автоматизированной инспекции кода упрощает задачу просмотра больших кусков кода, систематически сканируя его на предмет обнаружения наиболее известных уязвимостей.

В среде Github при работе с командами разработчиков программного обеспечения обычно при подготовке необходимо выполнить следующие действия:

- Добавить среду необходимых членов команды (организация и соавторы).
- Обеспечить возможность отправки версий и их слияния (Pull Requests).
- Обеспечить отслеживание ошибок (issues Github).
- Реализовать возможность комментариев к строкам и URL-запросов.

Как правило, существует два способа настройки в Github для совместной работы необходимых членов команды:

1. Создание организаций. Владелец организации может создавать множество организаций с разными уровнями доступа для различных репозиториев.

2. Добавление сотрудников. Владелец репозитория может добавлять коллабораторов с доступом Read + Write для одного репозитория

Создание организаций разработки. Если вы контролируете несколько команд и хотите установить разные уровни доступа для каждой команды с различными членами и добавить каждого участника в разные репозитории, то организация будет наилучшим вариантом. Любая учетная запись пользователя Github уже может создавать бесплатные организации для репозиториев с открытым исходным кодом. Чтобы создать организацию, необходимо перейти на страницу настроек своей организации:

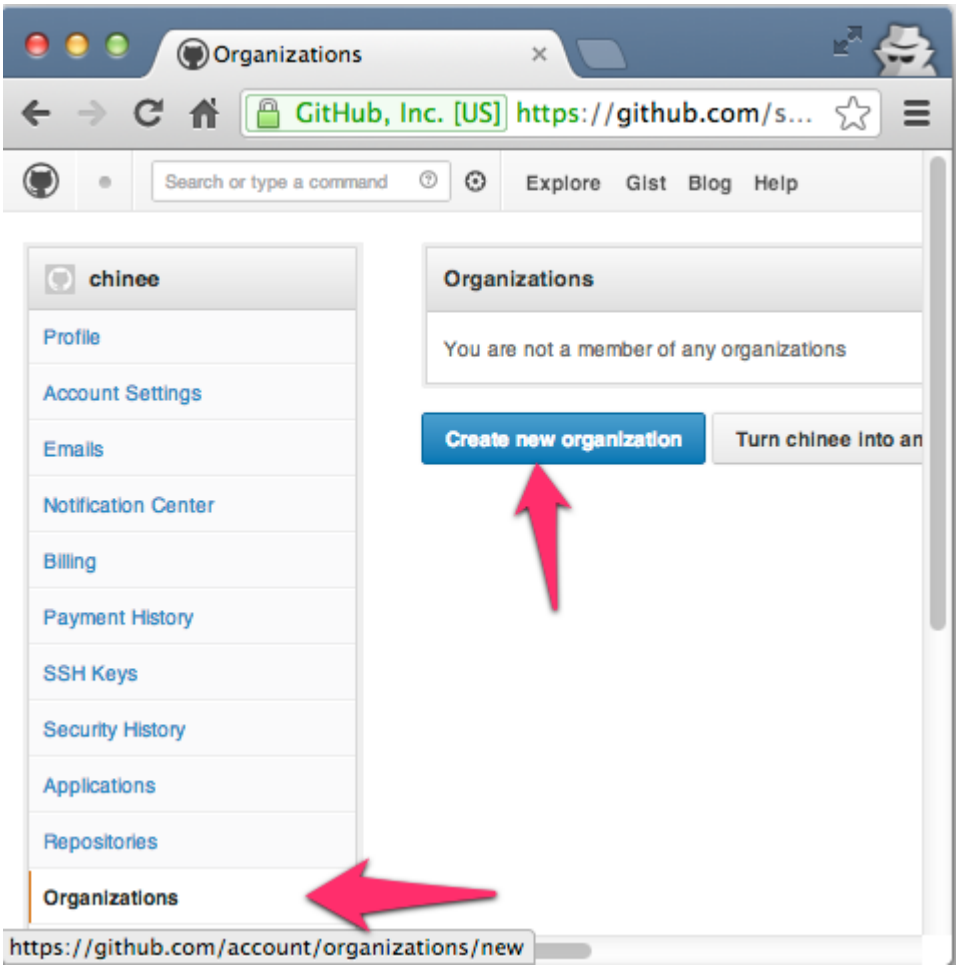


Рис.8.1 Добавление организации

Чтобы получить доступ к странице команд для вашей Организации, необходимо перейти на http://github.com/organizations/[organization-name]/teams, или для создания новых организацй – https://github.com/organizations/[organization-name]/teams/new.

Для создаваемых команд возможны три уровня доступа:

1. Pull Only: выборка и слияние с другим репозиторием или локальной копией. Доступ только для чтения.
2. Push and Pull: – доступно обновлением удаленного репозитория. Читайте + Запись.
3. Push, Pull & Administrative: (1), (2) добавляются права созданием команд, а также удаление аккаунтов организации. Чтение + запись + доступ администратора
4.

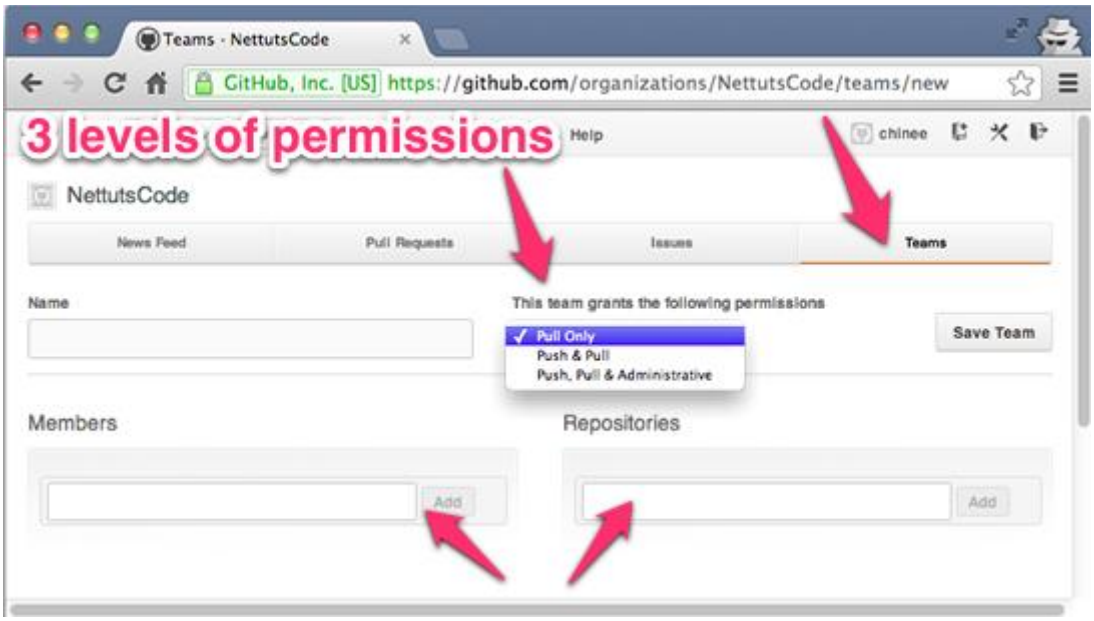


Рис.8.2 Создание команд с различными уровнями доступа

Настройка отправки и слияния (Pull Requests)

Pull Requests – позволяет внести свой вклад в репозиторий, сделав его копию. Копия репозитария может быть отправлена (pull request) владельцу репозитория, чтобы объединить изменения кода. Сам pull request может включать обсуждение качества кода, функций или даже общей стратегии.

В Github есть две модели для создания копии репозитария.

Модель Fork & Pull – используется в общедоступном репозитории, на который у вас нет push-доступа

Share Repository Model – используется в частном репозитории, на который у нас есть push-доступ. В этом случае форк не требуется.

Здесь мы видим рабочий процесс между двумя пользователями (repo-owner и forked-repo-owner) для модели Fork and Pull:

Определите репозиторий Github, в который вы хотите внести свой вклад, и нажмите кнопку «Fork», чтобы создать клон репозитория в вашей собственной учетной записи Github.

Рис.8.3 Вызов команды создания копии репозитария

Это создаст точную копию репозитория в вашем собственном аккаунте.

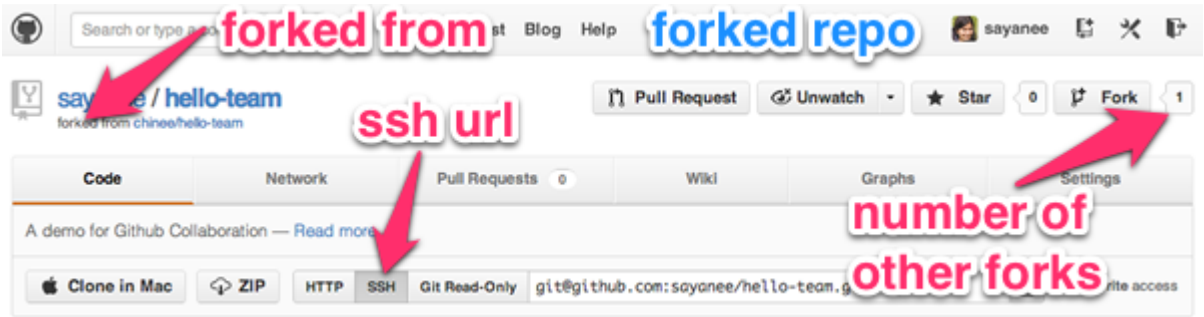


Рис.8.4 Содержание действий в создании копии репозитария

Выберите URL-адрес SSH, чтобы он запрашивал вашу парольную кодовую фразу SSH вместо имени пользователя и пароля каждый раз, когда вы делаете git push или git pull. Затем мы будем клонировать этот репозиторий на наш локальный компьютер:

```
1$ git clone [ssh-url] [folder-name]
2$ cd [folder-name]
```

Создадим новую ветку, чтобы внести очень простое изменение в файл `readme.md`:

```
1$ git checkout -b [new-feature]
```

После внесения соответствующих дополнений для создания новых функций мы просто передадим новые изменения и проверку в ветку git master:

```
1$ git add .
2$ git commit -m «information added in readme»
3$ git checkout master
```

Перейдем на ветку с новой задачей, а так же на псевдоним для удаленного репозитория. Затем мы будем пушить изменения с помощью `git push [git-remote-alias] [branch-name]`:

```
$ git branch
* master
1readme
2$ git remote -v
3origin git@github.com:[forked-repo-owner-username]/[repo-name].git
4(fetch)
5origin git@github.com:[forked-repo-owner-username]/[repo-name].git
6(push)
7$ git push origin readme
```

На исходной «развязанной» странице Github репозитория перейдем к ветке с новой функцией, а затем нажмите кнопку «Pull Request».

После обсуждения возможно, что владелец «форкнутого» репозитория может захотеть добавить изменения в новую функцию. В этом случае мы выберем одну и ту же ветку на нашей локальной машине, зафиксируем ее и запушим ее обратно на Github. Когда мы заходим на страницу запроса в оригинальном репозитории, он будет автоматически обновляться.

Обзор кода

С каждой фиксацией изменений Github позволяет использовать чистый интерфейс для общих комментариев или даже конкретных комментариев к отдельной строчке кода. Возможность делать комментарии или задавать вопросы по каждой отдельной строке кода очень полезна при проведении обзоров строка за строкой. Чтобы просмотреть встроенные комментарии, установите флажок в верхней части каждой фиксации.

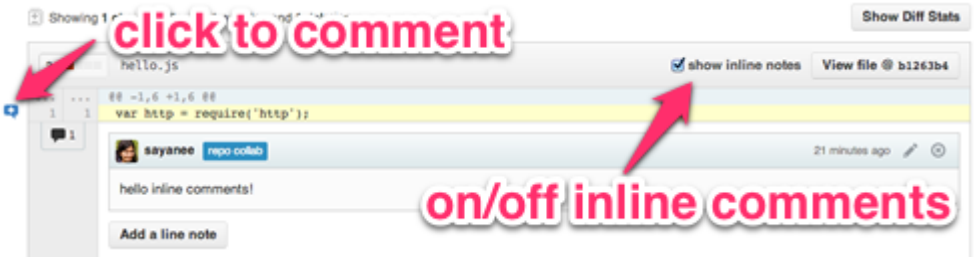


Рис.8.5 Просмотр встроенных комментариев

При обзоре кода могут быть использованы шаблоны URL-адресов, они предоставляют возможность отслеживать различия между фиксациями.

Шаблон сравнения веток – Compare branches / tags / SHA1

https://github.com/[username]/[repo-name]/compare/[starting-SHA1]...[ending-SHA1] .

Рис.8.6 Шаблон сравнения веток

Сравнение без пробелов: добавьте ?w=1 для сравнения URL-адресов.

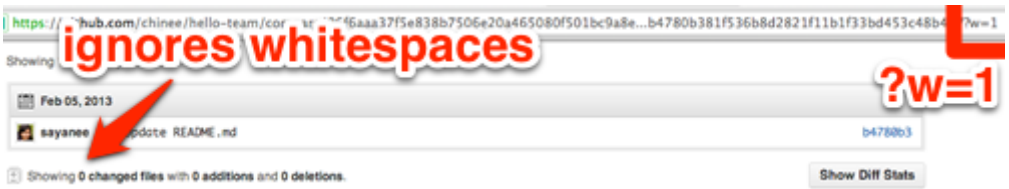


Рис.8.7 Шаблон для сравнения адресов

Diff: добавьте .diff к URL-адресам сравнения, чтобы получить информацию о git diff в текстовом формате. Это может быть полезно для сценариев.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ ДЛЯ РАБОТЫ

1. Создайте репозиторий на сервисе Github.
2. Задайте несколько типов организаций с различными параметрами доступа.
3. Задайте несколько учётных записей для разработчиков.
4. Задайте необходимые параметры для отправки и слияния копии репозитория.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие действия необходимо выполнить что бы подготовить среду разработки для инспекции кода?

Какие методы существуют для настройки Github для совместной работы нескольких разработчиков?

2. Что такое организация разработки в сервисе Github?
3. Какие уровни доступа могут быть заданы для организации в Github?
4. Какие параметры должны быть заданы при настройки отправки и слияния копии репозитория?
5. Какие модели применяются при создания копии репозитария?
6. Что такое «обзор кода»?

РАБОТА №9 ЛАБОРАТОРНАЯ. ПРОВЕРКИ НА СТОРОНЕ КЛИЕНТА

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью выполнения работы является получение практических навыков организации проверки на стороне клиента.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Даже если источником истины является модель предметной области и в конечном итоге необходимо проводить проверки на этом уровне, проверку можно выполнить как на уровне модели предметной области (на сервере), так и на уровне интерфейса пользователя (на клиенте).

Проверка на стороне клиента очень удобна для пользователей. Она экономит время, которое в противном случае тратилось бы на круговой путь к серверу, в результате которого выдавались бы ошибки проверки. С точки зрения бизнеса даже доли секунды, умножаемые в сотни раз каждый день, позволяют значительно сократить расходуемое время, деньги и усилия. Простая и немедленная проверка позволяет пользователям работать эффективнее и повышает точность входных и выходных данных.

Так же как модель представления отличается от модели предметной области, проверка модели представления и проверка модели предметной области имеют сходные черты, но выполняют разные функции. Если вы беспокоитесь о принципе DRY («Не повторяйся»), то в этом случае повторное использование кода может означать взаимозависимость, а в корпоративном приложении взаимозависимость стороны сервера и стороны клиента гораздо хуже, чем нарушение принципа «Не повторяйся».

Даже при проведении проверки на стороне клиента следует всегда проверять команды или входящие объекты переноса данных в серверном коде, ведь API сервера являются возможным вектором атаки. Как правило, рекомендуется выполнять обе проверки, поскольку, с точки зрения взаимодействия с пользователем, в клиентском приложении лучше действовать на опережение и не допускать ввод пользователем недопустимых данных.

Поэтому в коде на стороне клиенты вы обычно проверяете модели представления. Можно также проверять исходящие объекты переноса данных или команды клиента, прежде чем отправлять их службам.

Выполнение проверки на стороне клиента зависит от типа создаваемого клиентского приложения. Существуют отличия при проверке данных в веб-приложениях MVC, в которых программирование по большей части осуществляется в .NET, веб-приложениях SPA, в которых проверка написана на JavaScript или TypeScript, и в мобильных приложениях с кодом на Xamarin и C#.

Принцип DRY (от английского «Don't Repeat Yourself» – не повторяйся) является одним из основополагающих принципов разработки в модели MVC. В модели ASP.NET Core MVC рекомендуется задавать функциональные возможности или поведение только один раз, а затем отражать их в других местах в приложении. Это позволяет свести к минимуму объем кода, а также снизить риск возникновения в нем ошибок и упростить его тестирование и поддержку.

Ярким примером применения принципа «Не повторяйся» является поддержка проверки, реализуемая в модели MVC и на платформе Entity Framework Core Code First. Правила проверки декларативно определяются в одном месте (в классе модели) и затем применяются в рамках всего приложения.

Добавление правил проверки к модели фильма. Откройте файл *Movie.cs*. Класс *DataAnnotations* предоставляет набор встроенных атрибутов проверки, которые декларативно применяются к любому классу или свойству. Кроме того, этот класс содержит атрибуты форматирования, такие как *DataType*, которые обеспечивают форматирование и не предназначены для проверки.

Обновите класс, чтобы использовать преимущества встроенных атрибутов проверки *Required*, *StringLength*, *RegularExpression* и *Range*.

```
public class Movie
{
  public int Id { get; set; }

  [StringLength(60, MinimumLength = 3)]
  [Required]
```

```
  public string Title { get; set; }

  [Display(Name = «Release Date»)]
  [DataType(DataType.Date)]
  public DateTime ReleaseDate { get; set; }

  [Range(1, 100)]
  [DataType(DataType.Currency)]
  [Column(TypeName = «decimal(18, 2)»)]
  public decimal Price { get; set; }

  [RegularExpression(@»^[A-Z]+[a-zA-Z»«'\s-]*$»)]
  [Required]
  [StringLength(30)]
  public string Genre { get; set; }

  [RegularExpression(@»^[A-Z]+[a-zA-Z0-9»«'\s-]*$»)]
  [StringLength(5)]
  [Required]
  public string Rating { get; set; }
}
```

Атрибуты проверки определяют поведение для свойств модели, к которым они применяются:

Атрибуты *Required* и *MinimumLength* указывают, что свойство должно иметь значение. Тем не менее, чтобы удовлетворить требованиям проверки, пользователю достаточно ввести пробел. Атрибут *RegularExpression* ограничивает набор допустимых для ввода символов. В приведенном выше коде в полях *Genre* и *Rating* можно использовать только буквы (заглавная первая буква, пробелы, числа и специальные символы не допускаются).GenreRating

Атрибут *Range* ограничивает диапазон значений.Range

Атрибут *StringLength* позволяет задать максимальную и при необходимости минимальную длину строкового свойства.The StringLength attribute lets you set the maximum length of a string property, and optionally its minimum length.Value types (such as decimal, int, float, DateTime) are inherently required and don't need the [Required] attribute.Наличие правил проверки, которые автоматически применяются ASP.NET Core, помогает повысить степень надежности приложения. Это также гарантирует, что в любом случае будут вы-

полнены все проверки и в базе данных не будут случайно оставлены поврежденные данные.

Пользовательский интерфейс проверки ошибок в модели MVC

Запустите приложение и перейдите к контроллеру фильмов.Run the app and navigate to the Movies controller.

Коснитесь ссылки *Create New* (Создать), чтобы добавить новый фильм. Введите в форму какие-либо недопустимые значения. Если функция проверки jQuery на стороне клиента обнаруживает ошибку, сведения о ней отображаются в соответствующем сообщении.

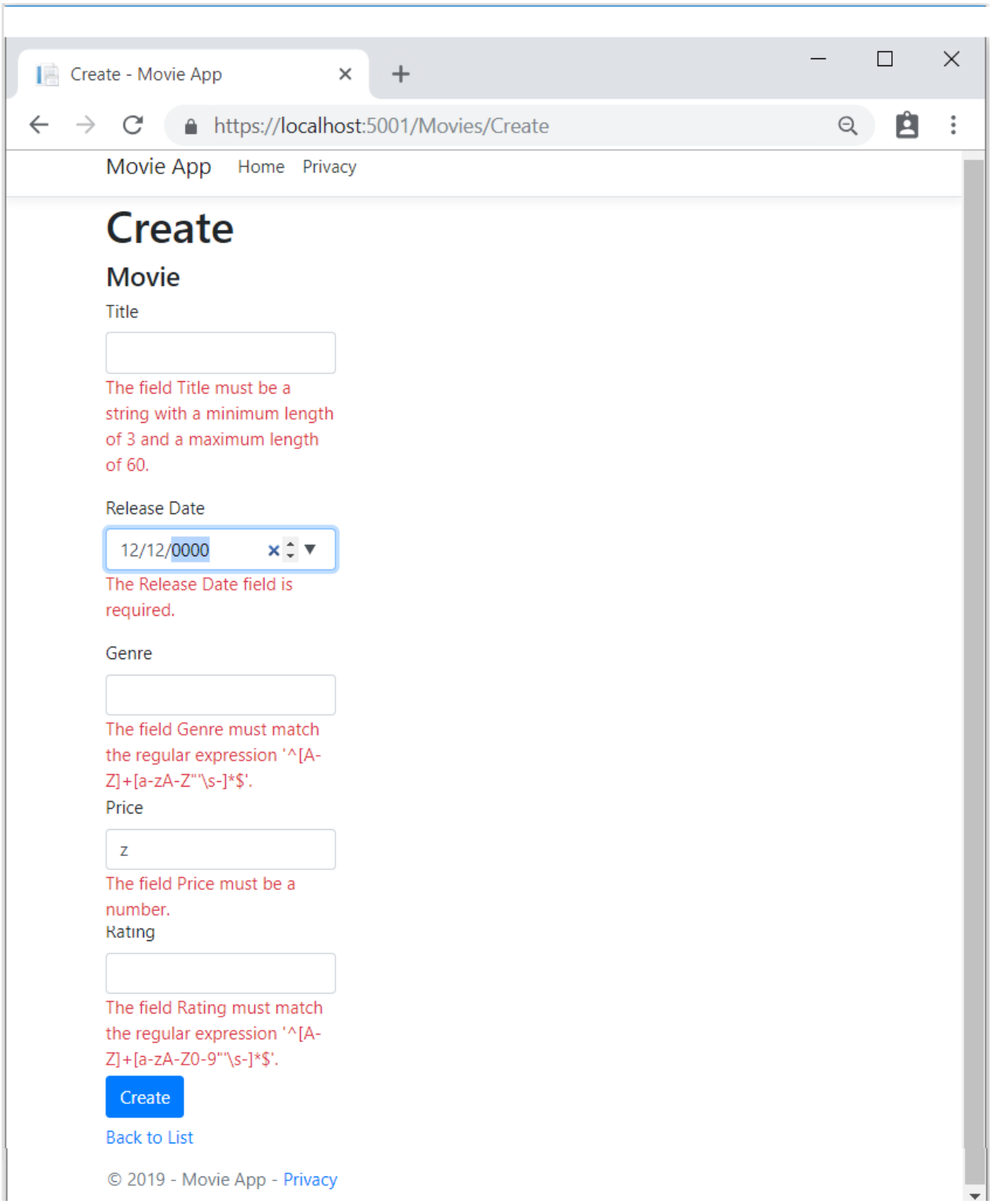


Рис.9.1 Пользовательский интерфейс тестового приложения с реализацией проверки на стороне клиента

Обратите внимание, что для каждого поля, содержащего недопустимое значение, в форме автоматически отображается соответствующее сообщение об ошибке проверки. Эти ошибки применяют-

ся как на стороне клиента (с помощью JavaScript и jQuery), так и на стороне сервера (если пользователь отключает JavaScript).

Серьезное преимущество заключается в том, что для реализации этого пользовательского интерфейса проверки не требуется изменять код в классе *MoviesController* или представлении *Create*. В контроллере и представлениях, создаваемых в рамках этого руководства, автоматически применяются правила проверки, для определения которых к свойствам класса модели *Movie* были применены атрибуты. При проверке с использованием метода действия *Edit* применяются те же правила.

Данные формы передаются на сервер только после того, как будут устранены все ошибки проверки на стороне клиента.

Принципы работы проверки. Вам может быть интересно, как пользовательский интерфейс проверки создается без обновления кода контроллера или представлений. В следующем примере кода показаны два метода *Create*.Create

```
// GET: Movies/Create
public IActionResult Create()
{
  return View();
}

// POST: Movies/Create
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(
  [Bind(«ID,Title,ReleaseDate,Genre,Price, Rating»)]
Movie movie)
{
  if (ModelState.IsValid)
  {
    _context.Add(movie);
    await _context.SaveChangesAsync();
    return RedirectToAction(«Index»);
  }
  return View(movie);
}
```

Первый метод действия *Create* (HTTP GET) отображает исходную форму создания. Вторая версия ([HttpPost]) обрабатывает передачу формы. Второй метод Create (версия [HttpPost]) вызывает

ModelState.IsValid, который определяет наличие ошибок проверки в фильме. При вызове этого метода оцениваются все атрибуты проверки, которые были применены к объекту.Calling this method evaluates any validation attributes that have been applied to the object. При наличии ошибок проверки в объекте метод Create повторно отображает форму. Если ошибок нет, метод сохраняет новый фильм в базе данных. В этом примере форма передается на сервер только после того, как будут устранены все ошибки проверки, обнаруженные на стороне клиента. Второй метод Create не вызывается до тех пор, пока на стороне клиента присутствуют ошибки проверки. При отключении JavaScript в браузере также отключается проверка на стороне клиента.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ ДЛЯРАБОТЫ

1. Создайте приложение MVC в среде Visual Studio, обеспечивающее ввод данных о некотором объекте.
2. Создайте проверку вводимых данных на стороне клиента.
3. Проверьте правильность ввода данных с помощью отрицательного и положительного теста.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что из себя представляет архитектура MVC?
2. Как реализуются проверка на стороне клиента?
3. Что включается в описание операции?
4. Почему при отключении JavaScript отключается проверка на стороне клиента?

РАБОТА №10 ЛАБОРАТОРНАЯ. ПРОВЕРКИ НА СТОРОНЕ СЕРВЕРА

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков по организации проверки на стороне сервера.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Проверка на стороне сервера – это проверка, которая возникает на сервере после отправки данных. Серверный код используется для проверки данных перед их сохранением в базе данных. Если данные не проходят проверку валидности, ответ отправляется обратно клиенту, чтобы сообщить пользователю, какие исправления должны быть сделаны. Проверка на стороне сервера не такая удобная, как проверка на стороне клиента, поскольку она не выдает ошибок до тех пор, пока не будет отправлена вся форма. Тем не менее, проверка на стороне сервера – это последняя линия защиты вашего приложения от неправильных или даже вредоносных данных. Все популярные серверные фреймворки имеют функции для проверки и очистки данных (что делает их безопасными).

В проверке на стороне сервера есть больше возможностей чем при проверке на клиенте. На сервере есть информация, которую клиентская сторона не может знать, эта информация определяется данными, хранимыми на стороне сервера.

C# довольно простой и гибкий язык. Вместе с .NET поставляется довольно много уже готовых классов, что делает его еще проще. Настолько, что вполне можно написать простой многопоточный HTTP-сервер для отдачи статического содержимого всего за 15 минут. Можно было бы использовать уже готовый класс HttpListener и управиться еще быстрее, но цель этой статьи – показать, как вообще можно сделать нечто подобное в C#.

Для начала создадим новый консольный проект:

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace HTTPServer
{class Server
  {static void Main(string[] args)
    {
    }
  }
}
```

В .NET можно очень легко создать TCP-сервер при помощи класса TcpListener, чем мы и воспользуемся:

```
class Server
{             TcpListener            Listener;
// Объект, принимающий TCP-клиентов
  // Запуск сервера
  public Server(int Port)
  {
 // Создаем »слушателя» для указанного порта
    Listener = new TcpListener(IPAddress.Any,
Port);
    Listener.Start(); // Запускаем его
    // В бесконечном цикле
    while (true)
    {  // Принимаем новых клиентов
      Listener.AcceptTcpClient();
    }
  }

  // Остановка сервера
  ~Server()
  {  // Если »слушатель» был создан
    if (Listener != null)
    {  // Остановим его
      Listener.Stop();
    }
  }

  static void Main(string[] args)
  { // Создадим новый сервер на порту 80
    new Server(80);
```

```
    }
  }
```

Если сейчас запустить приложение, то уже можно будет подключиться к порту 80 и… все. Соединение будет лишь простаивать впустую, так как отсутствует его обработчик и оно не закрывается со стороны сервера.

Напишем самый простой обработчик:

```
// Класс-обработчик клиента
class Client
{
// Конструктор класса. Ему нужно передавать приня
того клиента от TcpListener
    public Client(TcpClient Client)
    { // Код простой HTML-странички
      string                Html                =
«<html><body><h1>It works!</h1></body></html>«;

// Необходимые заголовки: ответ сервера, тип и дл
ина содержимого. После двух пустых строк  -  са-
мо содержимое
      string  Str  =  «HTTP/1.1 200 OK\nContent-
type: text/html\nContent-Length:»              +
Html.Length.ToString() + «\n\n» + Html;
      // Приведем строку к виду массива байт
      byte[]       Buffer       =       Encod-
ing.ASCII.GetBytes(Str);
      // Отправим его клиенту
      Client.GetStream().Write(Buffer,  0,  Buf-
fer.Length);
      // Закроем соединение
      Client.Close();
    }
  }
```

Чтобы передать ему клиента, нужно изменить одну строчку в классе Server:

```
  // Принимаем новых клиентов и передаем их на
обработку новому экземпляру класса Client
```

```
new Client(Listener.AcceptTcpClient());
```

Теперь можно запустить программу, открыть в браузере адрес *127.0.0.1* и увидеть большими буквами «It works!» Перед тем, как приступить к написанию парсера HTTP-запроса, сделаем наш сервер многопоточным. Для этого есть два способа: создавать вручную новый поток для каждого клиента или воспользоваться пулом потоков. У обоих способов есть свои преимущества и недостатки. Если создавать по потоку на каждого клиента, то сервер может не выдержать высокой нагрузки, но можно работать с практически неограниченным количеством клиентов одновременно. Если использовать пул потоков, то количество одновременно работающих потоков будет ограничено, но нельзя будет создать новый поток, пока не завершатся старые. Ниже приведены примеры обоих способов. Напишем простую процедуру потока, которая будет лишь создавать новый экземпляр класса Client:

```
static void ClientThread(Object StateInfo)
{
  new Client((TcpClient)StateInfo);
}
```

Для использования первого способа нужно заменить только содержимое нашего бесконечного цикла приема клиентов:

```
// Принимаем нового клиента
TcpClient Client = Listen-
er.AcceptTcpClient();
// Создаем поток
Thread Thread = new Thread(new Parameteri-
zedThreadStart(ClientThread));
// И запускаем этот поток, передавая ему прин
ятого клиента
Thread.Start(Client);
```

Для второго способа нужно проделать то же самое:

```
// Принимаем новых клиентов. После того, как
кли-
```

```
ент был принят, он передается в новый поток (Clie
ntThread)
    // с использованием пула потоков.
    ThreadPool.QueueUserWorkItem(new
WaitCallback(ClientThread),
Listener.AcceptTcpClient());
```

Плюс надо установить максимальное и минимальное количество одновременно работающих потоков. Сделаем это в процедуре Main:

```
    // Определим нужное максимальное количество п
    отоков
    // Пусть будет по 4 на каждый процессор
    int              MaxThreadsCount              =
    Environment.ProcessorCount * 4;
    // Установим максимальное количество рабочих
    потоков
    ThreadPool.SetMaxThreads(MaxThreadsCount,
    MaxThreadsCount);
    // Установим минимальное количество рабочих п
    отоков
    ThreadPool.SetMinThreads(2, 2);
```

Максимальное количество потоков должно быть не меньше двух, так как в это число входит основной поток. Если установить единицу, то обработка клиента будет возможна лишь тогда, когда основной поток приостановил работу (например, ожидает нового клиента или была вызвана процедура Sleep). Итак, теперь переключимся целиком на класс Client начнем обрабатывать HTTP-запрос. Получим текст запроса от клиента:

```
    // Объявим строку, в которой будет хранится з
    апрос клиента
    string Request = ««;
    // Буфер для хранения принятых от клиента дан
    ных
    byte[] Buffer = new byte[1024];
    // Переменная для хранения количества байт, п
    ринятых от клиента
```

```
int Count;
// Читаем из потока клиента до тех пор, пока
от него поступают данные
while                ((Count                =
Client.GetStream().Read(Buffer,    0,    Buf-
fer.Length)) > 0)
{

// Преобразуем эти данные в строку и добавим
ее к переменной Request
  Request += Encoding.ASCII.GetString(Buffer,
0, Count);

// Запрос должен обрываться последовательност
ью \r\n\r\n

// Либо обрываем прием данных сами, если длин
а строки Request превышает 4 килобайта
  // Нам не нужно получать данные из POST-
запроса (и т. п.), а обычный запрос
  // по идее не должен быть больше 4 килобайт
  if (Request.IndexOf(«\r\n\r\n») >= 0 || Re-
quest.Length > 4096)
  {
    break;
  }
}
```

Далее осуществляем парсинг полученных данных:

```
// Парсим строку запроса с использованием рег
улярных выражений
// При этом отсекаем все переменные GET-
запроса
Match    ReqMatch    =    Regex.Match(Request,
@»^\w+\s+([^\s\?]+)[^\s]*\s+HTTP/.*|»);

// Если запрос не удался
if (ReqMatch == Match.Empty)
{
```

```
  // Передаем клиенту ошибку 400   -   невер-
ный запрос
  SendError(Client, 400);
  return;
}

// Получаем строку запроса
string RequestUri = ReqMatch.Groups[1].Value;

// Приводим ее к изначальному виду, преобразу
я экранированные символы
// Например, »%20» -> » »
RequestUri                                    =
Uri.UnescapeDataString(RequestUri);

// Если в строке содержится двоеточие, переда
дим ошибку 400
// Это нужно для защиты от URL типа http://ex
ample.com/../../file.txt
if (RequestUri.IndexOf(«..») >= 0)
{
  SendError(Client, 400);
  return;
}

// Если строка запроса оканчивается на »/», т
о добавим к ней index.html
if (RequestUri.EndsWith(«/»))
{
  RequestUri += «index.html»;
}
```

Ну и наконец осуществим работу с файлами: проверим, есть ли нужный файл, определим его тип содержимого и передадим его клиенту.

```
string FilePath = «www/» + RequestUri;

// Если в папке www не существует данного фай
ла, посылаем ошибку 404
```

```
if (!File.Exists(FilePath))
{  SendError(Client, 404);
  return;
}

// Получаем расширение файла из строки запрос
а
string      Extension      =      RequestU-
ri.Substring(RequestUri.LastIndexOf('.'));

// Тип содержимого
string ContentType = ««;

// Пытаемся определить тип содержимого по рас
ширению файла
switch (Extension)
{
  case «.htm»:
  case «.html»:
    ContentType = «text/html»;
    break;
  case «.css»:
    ContentType = «text/stylesheet»;
    break;
  case «.js»:
    ContentType = «text/javascript»;
    break;
  case «.jpg»:
    ContentType = «image/jpeg»;
    break;
  case «.jpeg»:
  case «.png»:
  case «.gif»:
    ContentType   =   «image/»   +   Exten-
sion.Substring(1);
    break;
  default:
    if (Extension.Length > 1)
    {
```

```
      ContentType = «application/» + Exten-
sion.Substring(1);
    }
    else
    {
      ContentType = «application/unknown»;
    }
    break;
}

// Открываем файл, страхуясь на случай ошибки
FileStream FS;
try
{
  FS = new FileStream(FilePath, File-
Mode.Open, FileAccess.Read, FileShare.Read);
}
catch (Exception)
{

// Если случилась ошибка, посылаем клиенту ош
ибку 500
  SendError(Client, 500);
  return;
}

// Посылаем заголовки
string Headers = «HTTP/1.1 200 OK\nContent-
Type: » + ContentType + «\nContent-Length: »
+ FS.Length + «\n\n»;
byte[] HeadersBuffer = Encod-
ing.ASCII.GetBytes(Headers);
Client.GetStream().Write(HeadersBuffer, 0,
HeadersBuffer.Length);

// Пока не достигнут конец файла
while (FS.Position < FS.Length)
{
  // Читаем данные из файла
```

```
    Count = FS.Read(Buffer, 0, Buffer.Length);
    // И передаем их клиенту
    Client.GetStream().Write(Buffer, 0, Count);
  }

  // Закроем файл и соединение
  FS.Close();
  Client.Close();
```

Также в коде упоминалась пока не описанная процедура SendError. Напишем и ее:

```
  // Отправка страницы с ошибкой
  private void SendError(TcpClient Client, int
  Code)
  {  // Получаем строку вида »200 OK»
    // HttpStatusCode хранит в себе все статус-
  коды HTTP/1.1
    string CodeStr = Code.ToString() + « » +
  ((HttpStatusCode)Code).ToString();
    // Код простой HTML-странички
    string Html = «<html><body><h1>« + CodeStr
  + «</h1></body></html>«;

  // Необходимые заголовки: ответ сервера, тип
  и длина содержимого. После двух пустых строк
  – само содержимое
    string Str = «HTTP/1.1 » + CodeStr +
  «\nContent-type: text/html\nContent-Length:»
  + Html.Length.ToString() + «\n\n» + Html;
    // Приведем строку к виду массива байт
    byte[] Buffer = Encod-
  ing.ASCII.GetBytes(Str);
    // Отправим его клиенту
    Client.GetStream().Write(Buffer, 0, Buf-
  fer.Length);
    // Закроем соединение
    Client.Close();
  }
```

На этом написание простого HTTP-сервера окончено. Оно работает в несколько потоков, отдает статику, имеет простую защиту от плохих запросов и ругается на отсутствующие файлы.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. На основе приведённого примера создать сервер.
2. Реализовать на серверу проверку данных.
3. Создать пример документации.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чём преимущество проверки на сервере?
2. Каким образом проверка на сервере осуществляется в вашем случае?

СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Цель самостоятельной работы обучающихся – получить новые знания по дисциплине «Моделирование и анализ программного обеспечения».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в табл. 1.

Таблица 1

Виды самостоятельной работы

№ п/п	Вид СРС
1	Сравнительный анализ офисных пакетов
2	Сравнительный анализ браузеров
3	Анализ структуры заданного программного модуля

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности. – Москва : НИЦ ИНФРА-М, 2019. – 336 с. – Режим доступа: http://znanium.com/go.php?id=989682. – Загл. с экрана.

Дополнительная литература

1. Рудаков, А. В. Технология разработки программных продуктов [Электронный ресурс] : учебник для студентов учреждений среднего профессионального образования, обучающихся по специальности «Программное обеспечение вычислительной техники и автоматизированных систем» : [профессиональный модуль ПМ.03 «Участие в интеграции программных модулей» (МДК.03.01)] / А. В. Рудаков. – Москва : Академия, 2017. – 208 с. – Режим доступа: http://www.academia-moscow.ru/catalogue/4831/362819/. – Загл. с экрана.

2. Казанский, А. А. Программирование на visual c# 2013 [электронный ресурс]. – Москва : Юрайт, 2018. – 191 с. – Режим доступа: https://biblio-online.ru/book/programmirovanie-na-visual-c-2013-414752. – Загл. с экрана.

Программное обеспечение и интернет-ресурсы

1. http://svnbook.red-bean.com/ – Управление версиями вSubversion. Онлайн версии книги.
2. https://git-scm.com/-distributed-even-if-your-workflow-isnt
3. https://code.tutsplus.com/ru/articles/team-collaboration-with-github--net-29876 Совместная разработка в команде на GitHub

СОДЕРЖАНИЕ