МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Т. Ф. ГОРБАЧЕВА»
Филиал КузГТУ в г. Белово

Кафедра инженерно-экономическая

МДК. 03.01 Моделирование и анализ программного обеспечения

Методические рекомендации
по выполнению практических и лабораторных работ
для специальности
09.02.07 «Информационные системы и программирование»

Составитель: Витвицкий М.Н.
Рассмотрены и утверждены на заседании кафедры
Протокол № 4 от 06.12.2025 г.
Рекомендовано учебно-методической комиссией специальностей СПО в качестве электронного издания для использования в учебном процессе
Протокол № 4 от 11.12.2025 г.

Белово 2025

СОДЕРЖАНИЕ

ОРГАНИЗАЦИЯ ПРАКТИЧЕСКОЙ (ЛАБОРАТОРНОЙ) РАБОТЫ

Практическая (лабораторная) работа обучающихся может рассматриваться как организационная форма обучения, обеспечивающих управление учебной деятельностью или деятельность обучающихся по освоению общих и профессиональных компетенций, знаний и умений учебной и научной деятельности на примере выполнения тематических практических заданий.

Практическая (лабораторная) работа обучающихся проводится с целью: систематизации и закрепления полученных теоретических знании и закрепления практических навыков студентов. В рамках дисциплины “Моделирование и анализ программного обеспечения” студент должен закрепить на примере, разработанных практических заданий следующие навыки и умения:

- работать с проектной документацией, разработанной с использованием графических языков спецификаций;

- применять приемы работы с инструментальными средами проектирования программных продуктов.

- применять стандартные метрики по прогнозированию затрат, сроков и качества.

- определять этапы решения задачи; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы; составлять план действия; определять необходимые ресурсы; владеть актуальными методами работы в профессиональной и смежных сферах; реализовывать составленный план;

- оценивать результат и последствия своих действий (самостоятельно или с помощью наставника)

- определять задачи для поиска информации; определять необходимые источники информации; планировать процесс поиска; структурировать получаемую информацию; выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска, применять средства информационных технологий для решения профессиональных задач; использовать современное программное обеспечение; использовать различные цифровые средства для решения профессиональных задач.

- применять современную научную профессиональную терминологию; определять и выстраивать траектории профессионального развития и самообразования;

- выполнять процесс измерения характеристик компонент программного продукта для определения соответствия заданным критериям

- проводить сравнительный анализ программных продуктов и средств разработки, с целью выявления наилучшего решения согласно критериям, определенным техническим заданием

- выбирать способы решения задач профессиональной деятельности применительно к различным контекстам

- использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

- планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях

- эффективно взаимодействовать и работать в коллективе и команде

- пользоваться профессиональной документацией на государственном и иностранном языках

Иметь практический опыт:

- измерения характеристик программного проекта

- использования основные методологии процессов разработки программного обеспечения.

- оптимизации программного кода с использованием специализированных программных средств

- выполнять процесс измерения характеристик компонент программного продукта для определения соответствия заданным критериям

- проводить сравнительный анализ программных продуктов и средств разработки, с целью выявления наилучшего решения согласно критериям, определенным техническим заданием

ПЛАНИРОВАНИЕ ПРАКТИЧЕСКОЙ (ЛАБОРАТОРНОЙ) РАБОТЫ

При разработке рабочей программы по учебной дисциплине или профессиональному модулю при планировании содержания практической работы преподавателей устанавливается содержание и объем учебной информации или практических заданий, которые выносятся на практическую (лабораторную) работу, определяются формы и методы контроля результатов. Содержание практической (лабораторной) работы определяется в соответствии с рекомендуемыми видами заданий согласно программе учебной дисциплины модуля.

В соответствии с ведущей дидактической целью содержанием практических занятий являются решение разного рода задач, в том числе профессиональных (анализ производственных ситуаций, решение ситуационных производственных задач, выполнение профессиональных функций в деловых играх и т.п.), выполнение вычислений, расчетов, чертежей, работа с измерительными приборами, оборудованием, аппаратурой, работа с нормативными документами, инструктивными материалами, справочниками, составление проектной, плановой и другой технической и специальной документации и др.

Виды заданий для практической (лабораторной) работы, их содержание и характер имеют вариативный и дифференцированный характер, учитывают специфику данной дисциплины и индивидуальные особенности студента.

Перед выполнением студентами практической (лабораторной) работы преподаватель проводит инструктаж по выполнению задания, который включает цель задания, его содержание, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. В процессе инструктажа преподаватель предупреждает обучающихся о возможных типичных ошибках, встречающихся при выполнении задания.

Инструктаж проводится преподавателем за счет объема времени, отведенного на изучение дисциплины.

Практическая (лабораторная) работа может осуществляться индивидуально или группами обучающихся в зависимости от цели, объема, конкретной тематики работы, уровня сложности уровня умений обучающихся.

Отчет по практической (лабораторной) работе обучающихся предоставляется в электронном виде.

КОНТРОЛЬ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ (ЛАБОРАТОРНОЙ) РАБОТЫ

Контроль результатов практической (лабораторной) работы студентов осуществляться в пределах времени, отведенного на обязательные учебные занятия по дисциплине и практическую (лабораторную) работу обучающихся по дисциплине, может проходить в письменной, устной или смешанной форме, с представлением продукта деятельности учащегося.

В качестве форм и методов контроля практической (лабораторной) работы обучающихся, могут быть использованы, зачеты, тестирование, самоотчеты, контрольные работы, защита творческих работ и др., которые могут осуществляться на учебном занятии или вне его (например, оценки за реферат).

Критериями оценки результатов практической (лабораторной) работы обучающегося являются:

- уровень освоения учащимся учебного материала;
- умение обучающегося использовать теоретические знания при выполнении практических задач;
- сформированность общих и профессиональных компетенций;
- обоснованность и четкость изложения ответа;
- оформление материала в соответствии с требованиями.

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

Практическая (лабораторная) работа - выполнение проверочных упражнений, нацеленные на выявление степени освоенности материала, уровня подготовки студентов, выявление «проблемных сегментов» в рамках дисциплины или темы с целью корректировки дальнейшего плана обучения (дополнительное выделение времени на повторение или тотальное изучение темы и пр.). Данный вид может реализовываться как в групповом порядке (по вариантам), так и в индивидуальном (для каждого студента готовится отдельный пакет заданий: теория и практика).

Критерии оценки :

соответствие теме;

глубина проработки материала:

правильность и полнота использования источников;

оформление отчета.

Практическая (лабораторная) работа состоит из 2 частей:

1. Теоретическое часть (теоретическая часть работы);
2. Практическое задание (выдается преподавателем индивидуально согласно перечню).

Темы практических занятий

№ раздела (темы)	Вопросы, выносимые на изучение	Количество часов
Тема 3.1.1. Задачи и методы моделирования и анализа программных продуктов.	Практическое занятие №1 «Сравнительный анализ офисных пакетов»	2
	Практическое занятие №2 «Сравнительный анализ браузеров»	4
	Практическое занятие №3 «Сравнительный анализ средств просмотра видео»	4
Тема 3.1.2 Организация ревьюирования. Инструментальные средства ревьюирования	Практическое занятие №4 «Планирование code- review»	2

Темы лабораторных работ

№ раздела (темы)	Вопросы, выносимые на изучение	Количество часов
Тема 3.1.1. Задачи и методы моделирования и анализа программных продуктов.	Лабораторная работа №1 «Создание и изучение возможностей репозитория проекта»	2
	Лабораторная работа №2 «Экспорт настроек в командной среде разработки»	2
	Лабораторная работа №3 «Обратное проектирование алгоритма»	4
Тема 3.1.2 Организация ревьюирования. Инструментальные средства ревьюирования	Лабораторная работа №4 «Настройки доступа к репозиторию»	4
	Лабораторная работа №5 «Проверки на стороне клиента»	2
	Лабораторная работа №6 «Проверки на стороне сервера»	4

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 1.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ОФИСНЫХ ПАКЕТОВ

Целью работы является:

- изучение офисных пакетов;
- изучение принципов и получение практических навыков в сравнении программных продуктов;
- изучение содержания офисных пакетов;
- получение практических навыков выполнения основных операций в среде офисных пакетов;
- получение навыков выполнения анализа.

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 счету, 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 библиотеки, 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.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1.Microsoft Office, LibreOffice, OnlyOffice проведите сравнительный анализ для данных офисных пакетов по примеру представленному ранее.

2. Критерии для сравнения:

1) Состав и функциональность модулей — Текстовый редактор: стили, оглавления, сноски, редактор формул, совместное редактирование. — Таблицы: сводные таблицы, диаграммы, продвинутые функции, Power Query/аналог, надстройки. — Презентации: анимации, переходы, запись/трансляция, редактор видео/аудио. — Дополнительные модули: формы, диаграммы Ганта, базы данных (аналог Access), заметки, публикуемые шаблоны.

2) Совместимость форматов — Родные форматы: OOXML (DOCX/XLSX/PPTX), ODF (ODT/ODS/ODP), собственные расширения. — Точность импорта/экспорта: макеты, стили, шрифты, сложные таблицы, SmartArt. — Макросы и VBA/JS-скрипты: поддержка, совместимость, безопасность. — Работа с PDF: экспорт, редактирование, аннотации, OCR.

3) Совместная работа и коллаборация — Одновременное редактирование: синхронное/асинхронное, блокировка фрагментов. — Комментарии, упоминания, журнал изменений, сравнение версий. — Рецензирование: режим правок, правила согласования, рабочие процессы утверждения.

4) Облачные и серверные возможности — Веб-редакторы: функциональность, отзывчивость, офлайн-режим. — Собственная облачная платформа: хранилище, управление пользователями, SSO. — Самостоятельный хостинг: серверные редакторы, требования, масштабирование, контейнеры. — Интеграции с DMS/ECM: SharePoint, Nextcloud, OwnCloud, Alfresco, Google Drive, OneDrive.

5) Платформенная поддержка — ОС: Windows, macOS, Linux, мобильные (iOS/Android), веб. — Аппаратные требования: производительность на слабых ПК. — Поддержка ARM/Apple Silicon, оптимизация под сенсорные экраны.

6) Надстройки и расширяемость — Плагины/надстройки: каталог, SDK, API, скрипты. — Автоматизация: VBA, JavaScript, Python, .NET/COM, REST. — Интеграция с корпоративными системами: ERP/CRM, BI, электронный документооборот.

7) Безопасность и соответствие — Шифрование и защита: пароли, DRM/IRM, метки конфиденциальности. — Политики: DLP, контроль макросов, надежные каталоги. — Соответствие стандартам и сертификации: ISO, GDPR, HIPAA (при необходимости).

8) Управление и администрирование — Централизованное развёртывание и обновления. — Политики групп (GPO/MDM), конфигурирование. — Телеметрия, логи, мониторинг, аудит действий пользователей.

9) UX и удобство — Интерфейс: ленточный/классический, настраиваемость. — Подсказки, обучение, шаблоны, мастер-операции. — Доступность: экранные дикторы, горячие клавиши, контрастность.

10) Производительность и стабильность — Скорость открытия больших файлов, устойчивость к сбоям. — Качество рендеринга шрифтов и объектов. — Эффективность работы с сетевыми хранилищами.

11) Лицензирование и стоимость владения — Модель лицензирования: подписка, бессрочные, открытая лицензия. — Стоимость: на пользователя/устройство/сервер. — Условия для образования/НКО/госструктур. — Совокупная стоимость владения (TCO): поддержка, инфраструктура, миграция.

12) Локализация и поддержка — Языки интерфейса, проверка орфографии/грамматики. — Техническая поддержка: SLA, каналы, база знаний. — Сообщество: форумы, расширения, регулярность релизов.

13) Совместимость с шрифтами и верстkой — Поддержка OpenType-функций, внедрение шрифтов. — Замены отсутствующих шрифтов, метрики, переносы.

14) Специальные возможности для бизнеса — Шаблоны договоров, серийные письма, слияние данных. — Электронная подпись: X.509, ГОСТ/КриптоПро, встроенные провайдеры. — Формы и опросы, заполнение и сбор данных.

15) Миграция и обратная совместимость — Инструменты миграции, конвертация пакетами. — Совместимость с устаревшими форматами (DOC/XLS/PPT). — Поддержка долгосрочного архива: PDF/A, ODF.

3. Составить таблицу сравнения по данным критериям.

Описание формы отчета

Отчет по практической работе следует оформлять в текстовом файле с

расширением .doc, docx (или .pdf).

Файл отчета должен содержать:

заполненную таблицу;

иллюстрации при необходимости;

выводы по теме практической работы.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 2.
СРАВНИТЕЛЬНЫЙ АНАЛИЗ БРАУЗЕРОВ
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Рисунок 1. Логотипы наиболее распространенных 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 не-используемых 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 техно-логии «Turbo».

Фанаты Оперы начнут спорить, но факт остается фактом – браузер испытывает проблемы с производительностью. Некоторые скрипты обрабатываются достаточно медленно, аппаратное ускорение «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 экс- пресс-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 желающим 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.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Загрузите страницу https://docs.microsoft.com/ru-ru/aspnet/mvc/pluralsight с помощью браузеров OPERA, Mozilla, Chrom и оцените загрузку сайта по выбранным вами критериям.

2. Выполните функции:

чтения теста в слух;

перевода;

просмотра видео.

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

4. Сделайте вывод по особенностям выполнения функций.

Описание формы отчета

Отчет по практической работе следует оформлять в текстовом файле с расширением .doc, docx (или .pdf).

Файл отчета должен содержать:

заполненную таблицу;

иллюстрации при необходимости;

выводы по теме практической работы.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 3.
СРАВНИТЕЛЬНЫЙ АНАЛИЗ СРЕДСТВ ПРОСМОТРА ВИДЕО

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

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.

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.

Таблица 1.

Параметры	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		
Проигрываниево время скачивания torrent	Да		

Все плейеры поддерживаются аудио форматы 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.

ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ ДЛЯ РАБОТЫ

1. Найдите ресурсы для загрузки установочных компонентов видеоплейеров, используемых в вашей операционной системе. 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.

Описание формы отчета

Отчет по практической работе следует оформлять в текстовом файле с расширением .doc, docx (или .pdf).

Файл отчета должен содержать:

заполненную таблицу;

иллюстрации при необходимости;

выводы по теме практической работы.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 4.

ПЛАНИРОВАНИЕ CODE – REVIEW.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

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

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

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

GitHub – крупнейший 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 дает 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 создает в хранилище новые файлы, соответствующие новым состояниям измененных 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 создается 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:

1. Чтобы складывалось понимание происходящего и при возникновении проблем вы могли четко объяснить, что вы делали, и было видно, что пошло не так.

2. Все нажатия кнопок в графических клиентах в итоге сводят к

выполнению определенных команд консольного клиента, в то же время возможности графических клиентов ограничены по сравнению с консольным.

3. 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 предлага ется начать с создания forka к заведенному репозиторию https://github.com/MiPO

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.

В ОС Windows после установки клиента, появляется пункт Git Bash в контекстом меню папки. Достаточно перейти в желаемую папку и воспользоваться этим пунктом меню. В 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 <u>ваш репозиторий</u>

3. Переходим в созданную папку репозитория и настраиваем.

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 фай-лов).

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 этого 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 соединит их с вашими локальными правками, после чего можно повторить команду git push.

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.

Описание формы отчета

Отчет по практической работе следует оформлять в текстовом файле с расширением .doc, docx (или .pdf).

Файл отчета должен содержать:

описание этапов создания локального репозитория связанного с глобальным по заданным критериям

иллюстрации при необходимости;

выводы по теме практической работы.

ЛАБОРАТОРНАЯ РАБОТА №1

«СОЗДАНИЕ И ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ РЕПОЗИТОРИЯ ПРОЕКТА»

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цели работы:

1. Ознакомиться с основными концепциями систем контроля версий (СКВ).
2. Научиться создавать и настраивать репозиторий проекта.
3. Изучить основные команды и возможности, предоставляемые системой контроля версий.
4. Понять, как работать с удаленными репозиториями.

Задание:

Часть 1: Подготовка к работе

1. **Установите Git** на ваш компьютер. Если он уже установлен, убедитесь, что у вас последняя версия.

Для Windows: используйте Git for Windows.

Для macOS: установите через Homebrew, используя команду brew install git.

Для Linux: используйте пакетный менеджер вашего дистрибутива (например, sudo apt install git для Ubuntu).

2. **Создайте учетную запись** на GitHub (или другой платформе для хостинга репозиториев, например, GitLab или Bitbucket).

Часть 2: Создание локального репозитория

1. **Создайте новую папку** на вашем компьютере, которая будет использоваться для проекта. Например, создайте папку my_project.

2. **Инициализируйте новый репозиторий** в этой папке с помощью команды:

```
3. git init
```

4. **Создайте файл README.md** и добавьте в него краткое описание вашего проекта.

5. **Добавьте файл в репозиторий** и зафиксируйте изменения:

```
6. git add README.md
7. git commit -m "Добавлен файл README.md"
```

Часть 3: Работа с удаленным репозиторием

1. **Создайте новый репозиторий** на GitHub (или другой платформе). Не добавляйте README.md, так как он уже создан локально.

2. **Свяжите локальный репозиторий с удаленным** с помощью команды:

```
3. git remote add origin https://github.com/ваш_логин/my_project.git
```

4. **Отправьте (push) ваш локальный репозиторий** на удаленный:

```
5. git push -u origin master
```

Часть 4: Изучение возможностей Git

1. **Создайте новую ветку** и переключитесь на нее:

```
2. git checkout -b feature-branch
```

3. **Внесите изменения** в файл README.md, добавив новую информацию о проекте.

4. **Сохраните изменения** в новой ветке:

```
5. git add README.md
6. git commit -m "Обновлено описание проекта"
```

7. **Переключитесь обратно на основную ветку (master)**:

```
8. git checkout master
```

9. **Слияние (merge)** изменений из ветки feature-branch в master:

```
10. git merge feature-branch
```

11. **Отправьте изменения** на удаленный репозиторий:

```
12. git push origin master
```

Подготовьте отчет о выполненной работе, в котором отразите:

Отчет по практической работе следует оформлять в текстовом файле с расширением .doc, docx (или .pdf).

Файл отчета должен содержать:

Процесс создания локального и удаленного репозитория.

Использованные команды и их назначение.

Описание внесенных изменений и их слияния.

Личный опыт и возможные трудности, с которыми вы столкнулись.

Критерии оценки:

- Полнота выполнения задания.
- Правильность и корректность использования команд Git.
- Качество отчета (структура, ясность, полнота описания).

ЛАБОРАТОРНАЯ РАБОТА №2
«ЭКСПОРТ НАСТРОЕК В КОМАНДНОЙ СРЕДЕ РАЗРАБОТКИ»
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цели работы:

1. Ознакомиться с основами работы с настройками в командной среде разработки.

2. Научиться экспортировать и импортировать настройки для обеспечения консистентности рабочего окружения.

3. Изучить инструменты и команды, используемые для управления настройками.

Задание:

Часть 1: Подготовка к работе

1. **Выберите командную среду разработки** (IDE или текстовый редактор), с

которой вы будете работать. Это может быть:

Visual Studio Code

IntelliJ IDEA

PyCharm

Eclipse

Другие инструменты по вашему выбору.

2. **Убедитесь, что у вас установлены все необходимые плагины и расширения**, которые вы планируете использовать в своей среде разработки.

Часть 2: Экспорт настроек

1. **Откройте настройки вашей среды разработки**. Обычно это можно сделать через меню "Preferences" или "Settings".

2. **Найдите раздел, связанный с экспортом настроек**. Это может быть опция "Export Settings", "Export Configuration" или аналогичная.

3. **Экспортируйте настройки** в файл. Убедитесь, что вы сохранили файл в удобном для вас месте. Если ваша среда разработки поддерживает экспорт в различные форматы (например, JSON, XML), выберите подходящий для вас формат.

4. **Запишите, какие настройки были экспортированы** (например, настройки интерфейса, плагины, конфигурации проекта и т.д.).

Часть 3: Импорт настроек

1. **Создайте новую установку вашей среды разработки** на другом компьютере или в виртуальной машине, если это возможно. Это поможет вам лучше понять процесс импорта.

2. Откройте настройки вашей новой среды разработки.

3. **Найдите раздел, связанный с импортом настроек**. Это может быть опция "Import Settings", "Import Configuration" или аналогичная.

4. **Выберите файл, который вы экспортировали ранее**, и импортируйте его. Убедитесь, что все настройки были успешно применены.

5. **Запишите, какие настройки были успешно импортированы** и были ли какие-либо ошибки или предупреждения во время процесса.

Подготовьте отчет о выполненной работе, в котором отразите:

Процесс экспорта и импорта настроек.

Использованные команды и меню в вашей среде разработки.

Описание настроек, которые вы экспортировали и импортировали.

Личный опыт, трудности, с которыми вы столкнулись, и полезные советы для других пользователей.

Критерии оценки:

- Полнота выполнения задания.
- Правильность выполнения процессов экспорта и импорта.
- Качество отчета (структура, ясность, полнота описания).

ЛАБОРАТОРНАЯ РАБОТА №3
«ОБРАТНОЕ ПРОЕКТИРОВАНИЕ АЛГОРИТМА»
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических навыков выполнения обратного проектирования.

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 ниже, 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;
//Structu
re's
pointer
int  buff-
er[100];
```

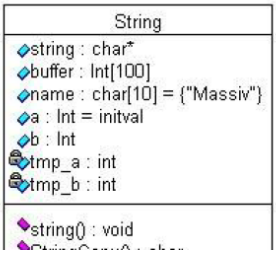

Рисунок 1.

ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ ДЛЯ РАБОТЫ

Заданием для работы является исходный код (выдается преподавателем), 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 отчете по работе должно быть сделано пояснение, указывающее, на основе какого программного объекта он выявлен.

3. Выявление переменных классов и отображение их в среде моделирования. Для каждой переменной класса в отчете по лабораторной работе должно быть сделано пояснение.

4. Выявление операций классов. Для каждого класса необходимо выявить операции и сделать пояснения о свойствах операции, ее параметрах и уровне в иерархии наследования.

Подготовьте отчет о выполненной работе, в котором отразите:

результат проведения проектирования кода;

иллюстрации при необходимости;

выводы по теме практической работы.

ЛАБОРАТОРНАЯ РАБОТА №4
«НАСТРОЙКИ ДОСТУПА К РЕПОЗИТОРИЮ»

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цели работы:

1. Изучить основные принципы управления доступом к репозиториям в системах контроля версий (например, Git).

2. Научиться настраивать права доступа для различных пользователей и групп.

3. Понять важность управления доступом для обеспечения безопасности и целостности кода.

Задание:

Часть 1: Введение в управление доступом

1. **Изучите теоретические аспекты управления доступом к репозиториям**:

Понимание ролей и прав доступа (чтение, запись, администрирование).

Различия между публичными и приватными репозиториями.

Принципы работы с ветками и их влияние на доступ.

Основные команды Git для управления доступом (например, git config, git remote).

Часть 2: Создание репозитория

1. **Создайте новый репозиторий** на платформе Git (например, GitHub, GitLab, Bitbucket):

Выберите тип репозитория: публичный или приватный.

Настройте описание и README файл.

Часть 3: Настройка пользователей и прав доступа

2. **Добавьте пользователей в репозиторий**:

Пригласите участников (например, коллег или однокурсников) в ваш репозиторий.

Используйте функционал платформы для управления доступом (например, раздел "Settings" на GitHub).

3. **Настройте права доступа** для каждого пользователя:

Определите роли (например, администратор, разработчик, читатель).

Убедитесь, что каждый пользователь имеет соответствующие права для выполнения своих задач.

Часть 4: Управление ветками

1. **Создайте несколько веток** в репозитории:

Основная ветка (например, main или master).

Ветки для разработки (например, feature-branch, bugfix-branch).

2. **Настройте правила доступа к веткам**:

Установите защиту для основной ветки, чтобы предотвратить прямые коммиты.

Настройте обязательные проверки (например, код-ревью, тесты) перед слиянием изменений.

Часть 5: Тестирование настроек доступа

1. **Проверьте настройки доступа**:

Убедитесь, что пользователи с различными ролями могут выполнять свои задачи (например, коммитить, пушить, создавать pull requests).

Проверьте, что пользователи, не имеющие прав, не могут выполнять запрещенные действия.

Подготовьте отчет о выполненной работе, в котором отразите:

Цели и задачи работы.

Описание созданного репозитория и его настроек.

Пошаговое описание добавления пользователей и настройки прав доступа.

Примеры управления ветками и установленных правил.

Выводы о важности управления доступом и его влиянии на безопасность проекта.

Критерии оценки:

- Полнота и точность настройки доступа.
- Корректность управления ветками и соблюдение правил.
- Ясность и структурированность отчета.
- Качество документации и соблюдение стандартов.

ЛАБОРАТОРНАЯ РАБОТА №5
«ПРОВЕРКИ НА СТОРОНЕ КЛИЕНТА»

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цели работы:

1. Ознакомиться с концепцией проверок на стороне клиента в веб-приложениях на основе ASP.NET.

2. Научиться реализовывать валидацию форм с использованием C# и JavaScript.

3. Развить навыки создания пользовательского интерфейса с учетом обратной связи от системы валидации.

Задание:

Часть 1: Введение в проверки на стороне клиента

1. **Изучите теоретические аспекты проверок на стороне клиента**:

Что такое проверки на стороне клиента и их преимущества по сравнению с

проверками на стороне сервера.

Различные типы валидации (первичная, вторичная, асинхронная).

Основные атрибуты HTML5 для валидации форм и как их использовать в ASP.NET.

Часть 2: Создание проекта

1. **Создайте новый проект в Visual Studio**:

Выберите тип проекта "ASP.NET Web Application".

Выберите шаблон "Web Application (Model-View-Controller)" или "Web Application (Web Forms)", в зависимости от ваших предпочтений.

Часть 3: Разработка формы

1. **Создайте HTML-форму** в соответствующем представлении (например, в Create.cshtml для MVC или в Default.aspx для Web Forms):

Поля:

- Имя (текстовое поле)
- Email (поле для ввода email)
- Пароль (поле для ввода пароля)
- Подтверждение пароля (поле для ввода подтверждения пароля)
- Кнопка отправки формы

2. **Добавьте атрибуты HTML5** для первичной валидации:

Укажите обязательные поля с помощью атрибута required.

Задайте шаблон для email (например, с помощью type="email").

Часть 4: Реализация валидации на стороне клиента

1. **Напишите JavaScript-код**, который будет выполнять следующие проверки:

Проверка, что все обязательные поля заполнены.

Проверка формата email (например, с использованием регулярных выражений).

Проверка, что пароль и подтверждение пароля совпадают.

Отображение сообщений об ошибках рядом с соответствующими полями, если проверки не пройдены.

Пример JavaScript-кода для валидации:

```
document.getElementById("form").onsubmit = function(event) {
var isValid = true;
var errorMessage = "";

var name = document.getElementById("name").value;
var email = document.getElementById("email").value;
var password = document.getElementById("password").value;
var confirmPassword = document.getElementById("confirmPassword").value;
if (!name) {
isValid = false;
errorMessage += "Имя обязательно.\n";
}
var emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
if (!email.match(emailPattern)) {
isValid = false;
errorMessage += "Введите корректный email.\n";
}
if (password !== confirmPassword) {
isValid = false;
errorMessage += "Пароли не совпадают.\n";
}

if (!isValid) {
event.preventDefault();
alert(errorMessage);
}
```

2. **Обработайте событие отправки формы**:

При успешной валидации, форма должна отправляться.

При наличии ошибок, предотвратите отправку формы и отобразите сообщения.

Часть 5: Реализация серверной валидации

1. **Добавьте валидацию на стороне сервера** в контроллере (для MVC) или в обработчике события (для Web Forms):

Убедитесь, что данные также проверяются на сервере для повышения безопасности.

Пример серверной валидации в контроллере MVC:

```
[HttpPost]
public ActionResult Create(UserModel model)
{
if (ModelState.IsValid)
{
// Логика сохранения данных в базу данных return RedirectToAction("Index");
}
return View(model);
```

Часть 6: Тестирование

1. **Проведите тестирование разработанной формы**:

Проверьте, что все проверки работают корректно как на стороне клиента, так и на стороне сервера.

Убедитесь, что сообщения об ошибках отображаются в правильных местах.

Проверьте, что форма отправляется только при успешной валидации.

Подготовьте отчет о выполненной работе, в котором отразите:

Цели и задачи работы.

Описание реализованной формы и используемых полей.

Пошаговое описание валидации, которую вы реализовали.

Примеры сообщений об ошибках и их отображение.

Выводы о важности проверок на стороне клиента и их влиянии на пользовательский опыт.

Критерии оценки:

- Полнота и точность реализации валидации.
- Корректность отображения сообщений об ошибках.
- Ясность и структурированность отчета.
- Качество кода и его соответствие стандартам.

ЛАБОРАТОРНАЯ РАБОТА №6

«ПРОВЕРКИ НА СТОРОНЕ СЕРВЕРА»

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цели работы:

1. Изучить концепцию валидации данных на стороне сервера в веб-приложениях.

2. Научиться реализовывать серверные проверки с использованием C# и ASP.NET.

3. Понять важность валидации данных для обеспечения безопасности и целостности веб-приложений.

Задание:

Часть 1: Введение в проверки на стороне сервера

1. **Изучите теоретические аспекты проверок на стороне сервера**:

Понимание валидации данных и ее роли в веб-приложениях.

Различия между валидацией на стороне клиента и на стороне сервера.

Основные типы валидации: обязательные поля, формат данных, диапазоны значений и т.д.

Причины, по которым серверная валидация необходима (безопасность, предотвращение атак и т.д.).

Часть 2: Создание проекта

2. **Создайте новый проект в Visual Studio**:

Выберите тип проекта "ASP.NET Web Application".

Выберите шаблон "Web Application (Model-View-Controller)" или "Web API", в зависимости от ваших предпочтений.

Часть 3: Разработка модели данных

1. **Создайте модель данных** (например, User Model), которая будет содержать поля:

Имя (string)

Email (string)

Пароль (string)

Дата рождения (DateTime)

2. **Добавьте атрибуты валидации** к полям модели с использованием встроенных атрибутов валидации:

[Required] для обязательных полей.

[EmailAddress] для проверки формата email.

[StringLength] для ограничения длины пароля.

[DataType(DataType.Date)] для поля даты рождения.

Пример модели:

```
public class UserModel
{
[Required(ErrorMessage = "Имя обязательно.")]
public string Name { get; set; }
[Required(ErrorMessage = "Email обязателен.")]
[EmailAddress(ErrorMessage = "Некорректный формат email.")]
public string Email { get; set; }
[Required(ErrorMessage = "Пароль обязателен.")]
[StringLength(100, MinimumLength = 6, ErrorMessage = "Пароль должен
содержать от 6 до 100 символов.")]
public string Password { get; set; }
[DataType(DataType.Date)]
public DateTime DateOfBirth { get; set; }
}
```

Часть 4: Реализация контроллера

1. **Создайте контроллер** (например, User Controller), который будет обрабатывать запросы на регистрацию пользователя.

2. **Добавьте метод действия** для обработки POST-запроса, который будет принимать модель данных и выполнять валидацию.

Пример метода контроллера:

```
[HttpPost]
public ActionResult Register(UserModel model)
{
if (ModelState.IsValid)
{
// Логика сохранения данных в базу данных
return RedirectToAction("Index");
```

```
}
// Если модель недействительна, верните представление с моделью
return View(model);
}
```

Часть 5: Обработка ошибок валидации

1. **Обработайте ошибки валидации**:

Убедитесь, что ошибки валидации возвращаются обратно в представление.

Отобразите сообщения об ошибках рядом с соответствующими полями формы.

Пример представления с отображением ошибок:

```
@model YourNamespace.Models.UserModel
<form asp-action="Register" method="post">
<div>
<label asp-for="Name"></label>
<input asp-for="Name" />
<span asp-validation-for="Name" class="text-danger"></span>
</div>
<div>
<label asp-for="Email"></label>
<input asp-for="Email" />
<span asp-validation-for="Email" class="text-danger"></span>
</div>
<div>
<label asp-for="Password"></label>
<input asp-for="Password" type="password" />
<span asp-validation-for="Password" class="text-danger"></span>
</div>
<div>
  <label asp-for="DateOfBirth"></label>
<input asp-for="DateOfBirth" type="date" />
<span asp-validation-for="DateOfBirth" class="text-danger"></span>
</div>
```

```
<button type="submit">Зарегистрироваться</button>
</form>
```

Часть 6: Тестирование

1. **Проведите тестирование разработанной функциональности**:

Проверьте, что все проверки работают корректно и сообщения об ошибках отображаются в нужных местах.

Убедитесь, что данные сохраняются только при успешной валидации.

Подготовьте отчет о выполненной работе, в котором отразите:

Цели и задачи работы.

Описание модели данных и используемых атрибутов валидации.

Пошаговое описание реализации контроллера и его методов.

Примеры отображения ошибок в представлении.

Выводы о важности серверной валидации и ее влиянии на безопасность приложения.

Критерии оценки:

- Полнота и точность реализации валидации.
- Корректность обработки ошибок и их отображение.
- Ясность и структурированность отчета.

Качество кода и его соответствие стандартам

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Основная литература

1. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем : учебник для среднего профессионального образования / Е. А. Черткова. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 146 с. — (Профессиональное образование). — ISBN 978-5-534-18094-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/563828.

Дополнительная литература

2. Казанский, А. А. Программирование на C# : учебное пособие для среднего профессионального образования / А. А. Казанский. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 181 с. — (Профессиональное образование). — ISBN 978-5-534-21380-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/569863.

3. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : Учебное пособие / Г. Н. Федорова. – Москва : КУРС : ИНФРА-М, 2022. – 336 с. (Среднее профессиональное образование). – ISBN 9785906818416. – Текст : непосредственный.

4. Гниденко, И. Г. Технология разработки программного обеспечения : учебник для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 248 с. — (Профессиональное образование). — ISBN 978-5-534-18131-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/563151.

5. Ерошевич К.В., Моделирование и анализ программного обеспечения систем: методические материалы к практическим занятиям, лабораторным и самостоятельным работам для студентов специальности СПО 09.02.07 Информационные системы и программирование очной формы обучения / сост. К. В. Ерошевич; Кузбасский государственный технический университет имени Т.Ф. Горбачева». – Кемерово, 2024. – Текст: электронный.

Составитель
Витвицкий Максим Николаевич

Методические указания по выполнению самостоятельной работы
для студентов очной формы обучения
по направлению специальности
09.02.07 «Информационные системы и программирование»

Публикуется в авторской редакции