МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

СРЕДНЕПРОФЕССИОНАЛЬНЫЙ КОЛЛЕДЖ

Д.Ф. Мамхягов

МДК 03.01 МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРАКТИКУМ
для студентов II курса специальности
09.02.07 – Информационные системы техники и оборудования

Черкесск
2025

ББК 32.973.4
УДК 004.41
М 22

Рассмотрено на заседании ЦК «Информационные технологии»
Протокол № 1 от «02» 09. 2024 г.
Рекомендовано к изданию редакционно-издательским советом СКГА.
Протокол № 27 от «07» 09. 2024 г.

Рецензенты: Моисеенко Л.А. – преподаватель СПК «СКГА»

М 22 **Мамхягов, Д. Ф.** МДК 03.01 Моделирование анализ программного обеспечение: практикум для студентов II курса специальности 09.02.07-Информационные системы и программирование / Д.Ф. Мамхягов. – Черкесск: БИЦ СКГА, 2025. –32 с.

Практикум содержит типовые задания по изучению технологии моделирования и анализа программного обеспечения. Выполнение практических заданий позволит развить навыки студентов в области компьютерных информационных технологий.

УДК 004.41
ББК 32.973.4

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Практикум подготовлен в соответствии с Федеральным государственным образовательным стандартом среднего профессионального образования по учебной дисциплине «Моделирование и анализ программного обеспечения». Практикум содержит типовые задания, предполагающие изучение основ программирования. Выполнение практических заданий позволит развить навыки студентов в области компьютерных информационных технологий.

Практикум адресован студентам специальности 09.02.07 Информационные системы и программирование, рекомендуются для использования в среднеспециальных учебных заведениях соответствующего профиля.

ПРАКТИЧЕСКАЯ РАБОТА № 1
Тема: "Создание и изучение возможностей репозитория проекта".

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

Часть 5: Отчет

1. **Подготовьте отчет** о выполненной работе, в котором отразите:
   - 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).

ПРАКТИЧЕСКАЯ РАБОТА № 2
Тема: "Экспорт настроек в командной среде разработки".

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.

Часть 4: Отчет

1. **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).

ПРАКТИЧЕСКАЯ РАБОТА № 3
Тема: "Сравнительный анализ офисных пакетов".

Цели работы:

1. Ознакомиться с различными офисными пакетами и их функциональными возможностями.
2. Провести сравнительный анализ популярных офисных пакетов по различным критериям.
3. Научиться делать выводы на основе собранной информации и анализа.

Задание:

Часть 1: Выбор офисных пакетов для анализа

1. **Выберите три или более офисных пакетов** для сравнения. Например, вы можете рассмотреть:
   - Microsoft Office (Word, Excel, PowerPoint)
   - Google Workspace (Docs, Sheets, Slides)
   - LibreOffice (Writer, Calc, Impress)
   - Apple iWork (Pages, Numbers, Keynote)
   - WPS Office
   - Другие офисные пакеты по вашему выбору.

Часть 2: Определение критериев сравнения

1. **Определите критерии, по которым вы будете сравнивать офисные пакеты**. Например:
   - Функциональность (наличие необходимых инструментов и возможностей)
   - Удобство использования (интерфейс, доступность функций)
   - Совместимость (поддержка различных форматов файлов)
   - Цена (бесплатные и платные версии)
   - Доступность (онлайн/офлайн доступ, кроссплатформенность)
   - Поддержка и обновления (частота обновлений, наличие технической поддержки)

Часть 3: Сбор информации

1. **Проведите исследование** по каждому выбранному офисному пакету. Используйте официальные сайты, обзоры, форумы и другие источники информации.
2. **Заполните таблицу** с собранной информацией по каждому из критериев для всех офисных пакетов. Пример таблицы:

Критерий	Microsoft Office	Google Workspace	LibreOffice	Apple iWork
Функциональность				
Удобство				
Совместимость				
Цена				
Доступность				
Поддержка				

Часть 4: Анализ и выводы

1. **На основе собранной информации** проанализируйте результаты сравнения. Выделите сильные и слабые стороны каждого офисного пакета.
2. **Сделайте выводы** о том, какой офисный пакет лучше всего подходит для различных сценариев использования (например, для студентов, бизнеса, удаленной работы и т.д.).

Часть 5: Подготовка отчета

1. **Подготовьте отчет** о выполненной работе, в котором отразите:
   - Цели и задачи работы.
   - Краткое описание каждого офисного пакета.
   - Заполненную таблицу с критериями сравнения.
   - Анализ и выводы, основанные на собранной информации.
   - Рекомендации по выбору офисного пакета для различных пользователей.

Критерии оценки:

- Полнота и точность собранной информации.
- Корректность заполненной таблицы.
- Глубина анализа и обоснованность выводов.
- Качество отчета (структура, ясность, полнота описания).

ПРАКТИЧЕСКАЯ РАБОТА № 4
Тема: "Сравнительный анализ веб-браузеров".

Цели работы:

1. Ознакомиться с основными веб-браузерами и их функциональными возможностями.
2. Провести сравнительный анализ популярных браузеров по различным критериям.

3. Научиться делать выводы на основе собранной информации и анализа.

Задание:

Часть 1: Выбор веб-браузеров для анализа

1. **Выберите три или более веб-браузеров** для сравнения. Например, вы можете рассмотреть:
   - Google Chrome
   - Mozilla Firefox
   - Microsoft Edge
   - Safari
   - Opera
   - Brave
   - Другие браузеры по вашему выбору.

Часть 2: Определение критериев сравнения

1. **Определите критерии, по которым вы будете сравнивать веб-браузеры**. Например:
   - Скорость загрузки страниц
   - Удобство интерфейса (дизайн, доступность функций)
   - Безопасность (защита от вредоносных сайтов, функции безопасности)
   - Поддержка расширений (наличие и разнообразие доступных расширений)
   - Потребление ресурсов (использование оперативной памяти и процессора)
   - Совместимость с веб-стандартами (поддержка HTML5, CSS3 и других технологий)
   - Платформенная доступность (доступность на различных операционных системах)

Часть 3: Сбор информации

1. **Проведите исследование** по каждому выбранному веб-браузеру. Используйте официальные сайты, обзоры, тесты производительности и другие источники информации.
2. **Заполните таблицу** с собранной информацией по каждому из критериев для всех браузеров. Пример таблицы:

Критерий	Google Chrome	Mozilla Firefox	Microsoft Edge	Safari	Opera	Brave
Скорость загрузки						
Удобство интерфейса						
Безопасность						
Поддержка расширений						
Потребление ресурсов						
Совместимость с веб-стандартами						
Платформенная доступность						

Часть 4: Анализ и выводы

1. **На основе собранной информации** проанализируйте результаты сравнения. Выделите сильные и слабые стороны каждого браузера.
2. **Сделайте выводы** о том, какой веб-браузер лучше всего подходит для различных сценариев использования (например, для обычных пользователей, разработчиков, любителей безопасности и т.д.).

Часть 5: Подготовка отчета

1. **Подготовьте отчет** о выполненной работе, в котором отразите:
   - Цели и задачи работы.
   - Краткое описание каждого веб-браузера.
   - Заполненную таблицу с критериями сравнения.
   - Анализ и выводы, основанные на собранной информации.
   - Рекомендации по выбору веб-браузера для различных пользователей.

Критерии оценки:

- Полнота и точность собранной информации.
- Корректность заполненной таблицы.
- Глубина анализа и обоснованность выводов.
- Качество отчета (структура, ясность, полнота описания).

ПРАКТИЧЕСКАЯ РАБОТА № 5
Тема: "Сравнительный анализ средств просмотра видео".

Цели работы:

1. Ознакомиться с различными средствами просмотра видео и их функциональными возможностями.
2. Провести сравнительный анализ популярных видео-плееров по различным критериям.
3. Научиться делать выводы на основе собранной информации и анализа.

Задание:

Часть 1: Выбор средств просмотра видео для анализа

1. **Выберите три или более средства просмотра видео** для сравнения. Например, вы можете рассмотреть:
   - VLC Media Player
   - Windows Media Player
   - KMPlayer
   - PotPlayer
   - QuickTime Player
   - Media Player Classic
   - Другие плееры по вашему выбору, включая онлайн-сервисы (например, YouTube, Vimeo).

Часть 2: Определение критериев сравнения

1. **Определите критерии, по которым вы будете сравнивать средства просмотра видео**. Например:
   - Поддерживаемые форматы видео и аудио
   - Удобство интерфейса (дизайн, доступность функций)
   - Качество воспроизведения (поддержка HD, 4K и других разрешений)
   - Наличие дополнительных функций (субтитры, эквалайзер, плейлисты)
   - Потребление ресурсов (использование оперативной памяти и процессора)
   - Поддержка потокового видео (возможность воспроизведения онлайн-контента)
   - Платформенная доступность (доступность на различных операционных системах)

Часть 3: Сбор информации

1. **Проведите исследование** по каждому выбранному средству просмотра видео. Используйте официальные сайты, обзоры, тесты производительности и другие источники информации.
2. **Заполните таблицу** с собранной информацией по каждому из критериев для всех плееров. Пример таблицы:

Критерий	VLC Media Player	Windows Media Player	KMPlayer	PotPlayer	QuickTime Player
Поддерживаемые форматы					
Удобство интерфейса					
Качество воспроизведения					
Наличие дополнительных функций					
Потребление ресурсов					
Поддержка потокового видео					
Платформенная доступность					

Часть 4: Анализ и выводы

1. **На основе собранной информации** проанализируйте результаты сравнения. Выделите сильные и слабые стороны каждого средства просмотра видео.

2. **Сделайте выводы** о том, какое средство просмотра видео лучше всего подходит для различных сценариев использования (например, для обычных пользователей, профессионалов, любителей потокового видео и т.д.).

Часть 5: Подготовка отчета

1. **Подготовьте отчет** о выполненной работе, в котором отразите:
   - Цели и задачи работы.
   - Краткое описание каждого средства просмотра видео.
   - Заполненную таблицу с критериями сравнения.
   - Анализ и выводы, основанные на собранной информации.
   - Рекомендации по выбору средства просмотра видео для различных пользователей.

Критерии оценки:

- Полнота и точность собранной информации.
- Корректность заполненной таблицы.
- Глубина анализа и обоснованность выводов.
- Качество отчета (структура, ясность, полнота описания).

ПРАКТИЧЕСКАЯ РАБОТА № 6
Тема: "Обратное проектирование алгоритма".

Цели работы:

1. Ознакомиться с концепцией обратного проектирования и его применением в разработке программного обеспечения.
2. Научиться анализировать существующий алгоритм и восстанавливать его описание.
3. Развить навыки документирования и представления алгоритмов в виде блок-схем и псевдокода.

Задание:

Часть 1: Выбор алгоритма для обратного проектирования

1. **Выберите алгоритм**, который вы хотите исследовать. Это может быть алгоритм сортировки (например, сортировка пузырьком, быстрая сортировка), алгоритм поиска (например, бинарный поиск) или любой другой алгоритм, который вам интересен.
2. **Изучите его реализацию** на одном из языков программирования (например, Python, Java, C++ и т.д.). Если у вас нет конкретного алгоритма, вы можете воспользоваться примерами из учебников или онлайн-ресурсов.

Часть 2: Анализ алгоритма

1. **Проанализируйте выбранный алгоритм**:
   - Определите его входные данные и выходные данные.
   - Опишите основные шаги алгоритма, включая циклы и условия.
   - Обратите внимание на сложность алгоритма (временную и пространственную).

Часть 3: Восстановление алгоритма

1. **Составьте описание алгоритма**:
   - Напишите текстовое описание алгоритма, в котором подробно опишите его работу и логику.
   - Используйте псевдокод для представления алгоритма. Псевдокод должен быть понятным и четким, чтобы его можно было легко преобразовать в код на любом языке программирования.
2. **Создайте блок-схему**:
   - На основе описания алгоритма нарисуйте блок-схему, которая визуально представляет его логику. Используйте стандартные символы блок-

схем (прямоугольники, ромбы и т.д.) для обозначения операций, условий и циклов.

Часть 4: Тестирование алгоритма

1. **Реализуйте алгоритм** на выбранном вами языке программирования.
2. **Напишите тесты**, чтобы проверить корректность работы алгоритма. Используйте различные наборы входных данных для проверки его надежности и устойчивости.

Часть 5: Подготовка отчета

1. **Подготовьте отчет**, в котором отразите:
   - Цели и задачи работы.
   - Описание выбранного алгоритма, его входные и выходные данные.
   - Текстовое описание алгоритма.
   - Псевдокод.
   - Блок-схему.
   - Результаты тестирования и выводы о корректности работы алгоритма.

Критерии оценки:

- Полнота и точность анализа алгоритма.
- Ясность и корректность псевдокода.
- Качество блок-схемы.
- Корректность реализации алгоритма и результаты тестирования.
- Структура и ясность отчета.

ПРАКТИЧЕСКАЯ РАБОТА № 7
Тема: "Планирование Code Review".

Цели работы:

1. Ознакомиться с концепцией code review и его важностью в процессе разработки программного обеспечения.
2. Научиться планировать и организовывать процесс проверки кода.
3. Развить навыки критического анализа кода и предоставления конструктивной обратной связи.

Задание:

Часть 1: Введение в Code Review

1. **Изучите теоретические аспекты** code review:
   - Что такое code review и его основные цели.
   - Преимущества и недостатки проведения проверки кода.
   - Различные подходы к организации code review (например, парное программирование, ревью в командах, использование инструментов для автоматизации).

Часть 2: Планирование процесса Code Review

1. **Определите основные этапы процесса code review**:
   - Подготовка к ревью (выбор кода, который будет проверяться).
   - Проведение ревью (анализ кода, общение с автором).
   - Завершение ревью (подведение итогов, внесение правок).
2. **Составьте план проведения code review**:
   - Определите участников ревью (кто будет проверять код, кто является автором).
   - Установите временные рамки для каждого этапа процесса.
   - Определите критерии проверки кода (стилистические, функциональные, производительность и т.д.).
   - Выберите инструменты, которые будут использоваться для проведения ревью (например, GitHub, GitLab, Bitbucket, Gerrit).

Часть 3: Проведение Code Review

1. **Выберите проект или часть кода**, которую вы будете проверять. Это может быть ваш собственный проект или открытый проект на платформе, такой как GitHub.
2. **Проведите code review** по разработанному плану:
   - Изучите код, обратив внимание на его структуру, читаемость, соответствие стандартам и наличие ошибок.
   - Подготовьте список комментариев и предложений по улучшению кода. Обратите внимание на положительные моменты, а также на области, требующие доработки.

Часть 4: Подготовка отчета

1. **Подготовьте отчет о проведенном code review**, в котором отразите:
   - Цели и задачи работы.
   - Описание проекта или кода, который вы проверяли.
   - Пошаговый план проведения code review.
   - Основные выводы и рекомендации по улучшению кода.
   - Примеры конструктивных комментариев, которые вы предоставили автору кода.

Критерии оценки:

- Полнота и точность анализа кода.
- Ясность и структурированность плана проведения code review.
- Качество комментариев и предложений по улучшению кода.
- Структура и ясность отчета.

ПРАКТИЧЕСКАЯ РАБОТА № 8
Тема: "Проверки на стороне клиента с использованием C# и ASP.NET".

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.

Часть 7: Подготовка отчета

1. **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.

ПРАКТИЧЕСКАЯ РАБОТА № 9
Тема: "Проверки на стороне сервера".

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

1. **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.

Часть 7: Подготовка отчета

1. **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.

ПРАКТИЧЕСКАЯ РАБОТА № 10
Тема: "Настройки доступа к репозиторию".

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

1. **Добавьте пользователей в репозиторий**:
   - Пригласите участников (например, коллег или однокурсников) в ваш репозиторий.
   - Используйте функционал платформы для управления доступом (например, раздел "Settings" на GitHub).
2. **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.

Часть 6: Подготовка отчета

1. **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.

Список литературы

1. Рудаков, А.В. Технология разработки программных продуктов [Текст]: учебник для студ. учреждений сред. проф. образования / А.В.Рудаков.- М.: Академия, 2022. — 208 с.

2. Федорова, Г.Н. Участие в интеграции программных модулей [Текст]: учеб. пособие для студ. учреждений сред. проф. образования / Г.Н.Федорова.- М.: Академия, 2021. — 304 с.

3. Боронина Л.Н. Основы управления проектами [Текст]: учебное пособие / Л.Н. Боронина, З.В. Сенук. —Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2023. — 136 с.

МАМХЯГОВ Давлет Фралевич

МДК 03.01 МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРАКТИКУМ
для студентов II курса специальности
09.02.07 – Информационные системы техники и оборудования

Печатается в редакции автора

Корректор Чагова О.Х.
Редактор Чагова О.Х.

Сдано в набор 20.12.2024 г.
Формат 60х84/16
Бумага офсетная
Печать офсетная
Усл. печ. л. 1,62
Заказ № 5023
Тираж 100 экз.

Оригинал-макет подготовлен
в Библиотечно-издательском центре СКГА
369000, г. Черкесск, ул. Ставропольская, 36