МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ
по дисциплине
«Технологии разработки приложений»
для студентов специальности 10.05.03 Информационная безопасность автоматизированных систем, специализация Анализ безопасности информационных систем

Ставрополь
2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
Семестр 7 4
Лабораторная работа 1. Основы архитектурного проектирования программных приложений 4
Лабораторная работа 2. Подходы к созданию веб- и мобильных приложений 4
Лабораторная работа 3. Использование фреймворков (React, Angular, Vue, Flutter) 5
Лабораторная работа 4. Клиент-серверное взаимодействие, создание и потребление RESTful API 6
Лабораторная работа 5. Базы данных, ORM, интеграция с PostgreSQL, MongoDB, MySQL 6
Лабораторная работа 6. Контейнеризация приложений с помощью Docker, настройка среды исполнения 7
Лабораторная работа 7. Интеграция внешних API и сторонних сервисов в приложения 8
Лабораторная работа 8. Асинхронное программирование и работа с промисами, async/await 8
Лабораторная работа 9. Управление состоянием в веб- и мобильных приложениях 9
Семестр 8 10
Лабораторная работа 1. Работа с системами контроля версий Git, GitHub/GitLab 10
Лабораторная работа 2. Реализация CI/CD пайплайнов с Jenkins, GitHub Actions, GitLab CI 10
Лабораторная работа 3. Паттерны проектирования (MVC, MVVM, Repository, Singleton, Observer). 11
Лабораторная работа 4. Основы тестирования (юнит-тесты, мок-объекты, интеграционные сценарии) 11
Лабораторная работа 5. Обеспечение безопасности приложений, защита от XSS, CSRF, SQL-инъекций 12
Лабораторная работа 6. Мониторинг, логирование, управление исключениями 12
Лабораторная работа 7. Проектирование пользовательских интерфейсов, основы UI/UX-дизайна, прототипирование и тестирование интерфейсов 13
Лабораторная работа 8. Управление техническим долгом, рефакторинг кода, поддерживаемость и масштабируемость приложений 14

ВВЕДЕНИЕ

Цель и задачи освоения дисциплины

Целью освоения дисциплины «Технологии разработки приложений» является формирование у будущего специалиста по специальности 10.05.03 Информационная безопасность автоматизированных систем специализация «Анализ безопасности информационных систем» понимания основных методов, средств и технологий эффективного взаимодействия и реализации успешной командной работы.
Задачи дисциплины:

– изучение теоретических основ формирования и развития навыков командной работы;

– формирование умений управления групповыми проектами;

– овладение навыками эффективного социального взаимодействия, создания благоприятной и конструктивной атмосферы в команде средствами доступных информационных технологий.

Семестр 7

Лабораторная работа 1. Основы архитектурного проектирования программных приложений

Теоретическая часть:

Архитектура программного приложения определяет общую структуру системы, включая способы взаимодействия компонентов, принципы разделения обязанностей и правила взаимодействия между уровнями. Архитектурное проектирование обеспечивает масштабируемость, модульность, отказоустойчивость и сопровождаемость программного обеспечения. Основные архитектурные стили включают одноуровневую архитектуру, клиент-серверную модель, трехуровневую (presentation, logic, data), многослойную архитектуру, микросервисную и событийно-ориентированную архитектуры. Выбор архитектурного стиля зависит от характеристик проекта, требований по нагрузке и возможных сценариев масштабирования. Архитектурные решения документируются с использованием шаблонов, UML-диаграмм и описания взаимодействий компонентов. Большое значение имеет применение паттернов уровня архитектуры — например, слоистой архитектуры, брокера, MVC и пр., а также соблюдение принципов SOLID и подходов DDD (Domain-Driven Design).

Практические задания:

1. Проанализируйте архитектуру типичного интернет-магазина,
2. Разработайте архитектурную схему для новостного портала,
3. Сравните микросервисную и монолитную архитектуру для CRM-системы.

Лабораторная работа 2. Подходы к созданию веб- и мобильных приложений

Теоретическая часть:

Разработка веб- и мобильных приложений требует понимания различий в

платформах, пользовательских интерфейсах, способах развертывания и взаимодействия с внешними сервисами. Веб-приложения запускаются в браузере и обычно состоят из фронтенда (HTML/CSS/JS) и бэкенда (Node.js, Django и др.), мобильные приложения могут быть нативными (Java/Kotlin для Android, Swift для iOS) или кроссплатформенными (Flutter, React Native). Основной задачей разработчика становится выбор подходящей платформы с учетом потребностей бизнеса, целевой аудитории и времени на разработку. Современные подходы включают использование SPA (Single Page Application), PWA (Progressive Web App), гибридных решений, а также активное применение API для интеграции с внешними сервисами и микросервисами.

Практические задания:

1. Сравните особенности PWA и нативных мобильных приложений,
2. Разработайте структуру frontend/backend для приложения заметок,
3. Выберите стек технологий для кроссплатформенного чат-приложения.

Лабораторная работа 3. Использование фреймворков (React, Angular, Vue, Flutter)

Теоретическая часть:

Фреймворки ускоряют разработку за счёт готовой структуры, компонентов и инструментария. Веб-фреймворки типа React, Angular и Vue позволяют строить динамичные интерфейсы, эффективно управлять состоянием и обеспечивать высокую производительность. React — библиотека с гибкой экосистемой и виртуальным DOM, Angular — мощный фреймворк с встроенными инструментами, Vue — легкий и легко осваиваемый вариант для быстрого старта. Для мобильной разработки популярен Flutter, предоставляющий единую кодовую базу и высокую производительность за счёт собственного движка отрисовки. Выбор фреймворка определяется

сложностью приложения, требованиями к быстродействию и возможностями команды.

Практические задания:

1. Создайте простое To-Do приложение на React или Vue,
2. Сравните подходы к работе с состоянием в React и Angular,
3. Разработайте экран авторизации в Flutter с валидацией.

Лабораторная работа 4. Клиент-серверное взаимодействие, создание и потребление RESTful API

Теоретическая часть:

Клиент-серверная архитектура разделяет приложение на две части: клиент (отправляет запросы) и сервер (обрабатывает их). REST (Representational State Transfer) — один из самых популярных подходов к построению API, основанный на HTTP-методах (GET, POST, PUT, DELETE). REST API должен быть stateless, поддерживать идентификаторы ресурсов и использовать формат JSON или XML. Ключевым моментом является правильное проектирование маршрутов, статусов ответов и использование JWT или OAuth для аутентификации. Swagger и Postman применяются для тестирования и документирования API.

Практические задания:

1. Разработайте REST API для управления задачами (CRUD),
2. Реализуйте клиентское приложение для взаимодействия с API,
3. Протестируйте REST API с помощью Postman.

Лабораторная работа 5. Базы данных, ORM, интеграция с PostgreSQL, MongoDB, MySQL

Теоретическая часть:

Базы данных являются основой большинства приложений. Реляционные БД (PostgreSQL, MySQL) используют SQL и таблицы, обеспечивают целостность данных и транзакции. Нереляционные БД (MongoDB)

используют документы и являются более гибкими для хранения неструктурированных данных. ORM (Object-Relational Mapping) упрощает взаимодействие между объектами в коде и записями в базе, повышая читаемость и поддержку кода. SQLAlchemy, Sequelize, Django ORM — популярные инструменты ORM. Важно правильно проектировать схемы данных, индексы и связи между сущностями.

Практические задания:

1. Создайте схему базы данных для системы управления студентами,
2. Реализуйте ORM-модель и CRUD-операции с использованием PostgreSQL,
3. Сравните работу с MongoDB и MySQL на простом проекте.

Лабораторная работа 6. Контейнеризация приложений с помощью Docker, настройка среды исполнения

Теоретическая часть:

Контейнеризация позволяет упаковать приложение вместе со всеми его зависимостями в единый изолированный контейнер, что обеспечивает воспроизводимость, масштабируемость и независимость от окружения. Docker является наиболее популярным инструментом контейнеризации, предоставляющим возможности создания, развертывания и управления контейнерами. Dockerfile описывает, как собрать контейнер, а docker-compose позволяет конфигурировать и запускать многоконтейнерные приложения. Контейнеры могут включать в себя не только само приложение, но и базы данных, кэш-системы и прокси. Контейнеризация играет ключевую роль в DevOps-практиках, упрощает CI/CD процессы и облегчает переносимость приложений между средами.

Практические задания:

1. Создайте Dockerfile для веб-приложения,
2. Настройте docker-compose с приложением и базой данных,

3. Проанализируйте различия между виртуализацией и контейнеризацией.

Лабораторная работа 7. Интеграция внешних API и сторонних сервисов в приложения

Теоретическая часть:

Современные приложения редко существуют изолированно и часто используют сторонние API для получения данных или предоставления функциональности. Интеграция может осуществляться через REST, GraphQL, WebSocket и другие протоколы. При подключении стороннего API важно учитывать аспекты аутентификации (например, OAuth 2.0, API-ключи), обработки ошибок, ограничения по количеству запросов (rate limit), а также формат данных, обычно JSON или XML. Также важно тестировать устойчивость к нестабильному соединению и использовать библиотеки для упрощения запросов, такие как Axios, Retrofit или fetch.

Практические задания:

1. Получить и отобразить данные с открытого API (например, OpenWeather, NASA или YouTube).
2. Реализовать авторизацию через OAuth 2.0.
3. Обработать ошибки соединения и корректно вывести их в UI.

Лабораторная работа 8. Асинхронное программирование и работа с промисами, async/await

Теоретическая часть:

Асинхронность необходима для того, чтобы приложения оставались отзывчивыми, особенно при работе с сетевыми запросами, файловыми операциями или таймерами. В языках JavaScript, Dart и Python распространены конструкции промисов и async/await, которые позволяют писать асинхронный код, похожий на синхронный. Асинхронность тесно связана с концепцией событийного цикла, очередей задач и обработки

ошибок с помощью try/catch. Важно избегать "адов колбэков" (callback hell), структурировать цепочки запросов и обеспечивать читаемость.

Практические задания:

1. Написать асинхронную функцию с использованием async/await.
2. Обработать несколько последовательных запросов к серверу.
3. Реализовать задержку выполнения (например, через setTimeout, sleep) в асинхронном стиле.

Лабораторная работа 9. Управление состоянием в веб- и мобильных приложениях

Теоретическая часть:

Управление состоянием — важный аспект разработки динамичных приложений. Состояние отражает данные, которые влияют на интерфейс пользователя и логику приложения. В разных технологиях используются разные подходы: в React — useState, useReducer, Context API, Redux, MobX; во Flutter — Provider, Bloc, Riverpod. Неправильное управление состоянием может привести к утечкам памяти, лишним перерисовкам и логическим ошибкам. Правильная архитектура состояния повышает производительность, масштабируемость и удобство поддержки кода.

Практические задания:

1. Реализовать управление состоянием в React с помощью useState и useEffect.
2. Построить глобальное состояние через Context API или Redux.
3. Во Flutter — использовать Provider для передачи данных между экранами.

Семестр 8

Лабораторная работа 1. Работа с системами контроля версий Git, GitHub/GitLab

Теоретическая часть:

Контроль версий является неотъемлемой частью процесса разработки, обеспечивая отслеживание изменений, совместную работу над кодом и управление релизами. Git — распределённая система контроля версий, предоставляющая мощные средства работы с ветками, историями коммитов и слияниями. GitHub и GitLab — платформы, добавляющие к Git удобный веб-интерфейс, инструменты управления задачами, ревью кода, автоматизации и CI/CD. Знание базовых операций (clone, commit, push, pull, merge, rebase), разрешения конфликтов и создания pull/merge request'ов — обязательный навык разработчика.

Практические задания:

1. Создайте репозиторий и выполните базовые операции с Git,
2. Настройте ветвление и выполните слияние с разрешением конфликтов,
3. Оформите pull request в командном проекте на GitHub.

Лабораторная работа 2. Реализация CI/CD пайплайнов с Jenkins, GitHub Actions, GitLab CI

Теоретическая часть:

CI/CD (непрерывная интеграция и доставка) — подход, при котором изменения в коде автоматически проходят сборку, тестирование и развертывание. Jenkins, GitHub Actions, GitLab CI позволяют описывать пайплайны как код (pipeline-as-code) и автоматизировать все стадии жизненного цикла приложения. Интеграция с системами контроля версий, триггеры на коммиты и PR, запуск тестов, сборка артефактов, деплой — всё это становится частью автоматизированного процесса. Такой подход повышает надёжность, ускоряет выпуск новых версий и снижает количество

ошибок при релизе.

Практические задания:

1. Настройте простой CI-пайплайн для Node.js-приложения на GitHub Actions,
2. Реализуйте автоматический деплой на удалённый сервер,
3. Сравните синтаксис описания пайплайнов в Jenkins и GitLab CI.

Лабораторная работа 3. Паттерны проектирования (MVC, MVVM, Repository, Singleton, Observer)

Теоретическая часть:

Паттерны проектирования — это проверенные временем шаблоны решений распространённых проблем в архитектуре приложений. MVC (Model-View-Controller) и MVVM (Model-View-ViewModel) разделяют бизнес-логику от пользовательского интерфейса. Repository обеспечивает единый доступ к данным, Singleton гарантирует существование одного экземпляра объекта, Observer реализует реакцию на события. Применение паттернов делает код понятнее, легче расширяемым и сопровождаемым, особенно в масштабных проектах. Знание паттернов критично при построении модульной и гибкой архитектуры.

Практические задания:

1. Реализуйте структуру MVC в простом веб-приложении,
2. Примените паттерн Repository для слоя доступа к данным,
3. Продемонстрируйте работу паттерна Observer в интерфейсе.

Лабораторная работа 4. Основы тестирования (юнит-тесты, мок-объекты, интеграционные сценарии)

Теоретическая часть:

Тестирование — ключевой этап разработки, направленный на выявление и предотвращение ошибок. Юнит-тесты проверяют отдельные функции или классы в изоляции, часто с использованием мок-объектов для

подмены зависимостей. Интеграционные тесты проверяют взаимодействие между компонентами, а системные охватывают работу приложения в целом. Использование фреймворков (Jest, Pytest, JUnit) облегчает написание и выполнение тестов. Хорошее покрытие тестами способствует уверенности в стабильности кода при внесении изменений.

Практические задания:

1. Создайте юнит-тесты для модуля аутентификации,
2. Протестируйте взаимодействие компонентов при авторизации,
3. Примените мок-объекты для изоляции сторонних сервисов.

Лабораторная работа 5. Обеспечение безопасности приложений, защита от XSS, CSRF, SQL-инъекций

Теоретическая часть:

Безопасность приложений требует учета множества угроз: межсайтовые скриптовые атаки (XSS), подделка межсайтовых запросов (CSRF), SQL-инъекции, уязвимости авторизации и передачи данных. Защита строится на правильной валидации ввода, использовании токенов, параметризованных запросах, HTTPS и хэшировании паролей. OWASP предоставляет перечень самых распространенных угроз и рекомендации по защите. Регулярные проверки, аудит кода и обновление зависимостей также являются важными мерами предотвращения атак.

Практические задания:

1. Продемонстрируйте уязвимость XSS и реализуйте защиту от неё,
2. Настройте CSRF-токены в форме логина,
3. Реализуйте защиту от SQL-инъекций с использованием ORM.

Лабораторная работа 6. Мониторинг, логирование, управление исключениями

Теоретическая часть:

Мониторинг и логирование являются неотъемлемыми компонентами

эксплуатационной фазы приложения, позволяющими отслеживать его поведение, производительность и реагировать на возникающие проблемы. Логирование обеспечивает запись значимых событий, ошибок и информации о работе системы, при этом лог-сообщения можно классифицировать по уровням — debug, info, warning, error, critical. Мониторинг позволяет отслеживать метрики состояния приложения, такие как загрузка CPU, использование памяти, время отклика и количество запросов. Современные системы мониторинга, такие как Prometheus, Grafana и ELK-стек (Elasticsearch, Logstash, Kibana), обеспечивают визуализацию, оповещения и аналитику. Управление исключениями предполагает предсказуемую обработку ошибок, генерацию информативных сообщений и недопущение сбоев, влияющих на пользователей. Правильная организация логики try/catch и продуманная архитектура обработки ошибок значительно повышают устойчивость системы.

Практические задания:

1. Настройте логирование событий в веб-приложении,
2. Соберите метрики с помощью Prometheus и отобразите их в Grafana,
3. Реализуйте глобальный обработчик ошибок и исключений.

Лабораторная работа 7. Проектирование пользовательских интерфейсов, основы UI/UX-дизайна, прототипирование и тестирование интерфейсов

Теоретическая часть:

Проектирование пользовательских интерфейсов требует баланса между функциональностью, эстетикой и удобством. UI (User Interface) отвечает за визуальное оформление элементов, таких как кнопки, формы и меню, в то время как UX (User Experience) ориентирован на общее впечатление пользователя и интуитивность взаимодействия. Принципы UI/UX включают простоту, предсказуемость, доступность, логическую иерархию,

согласованность элементов. Прототипирование позволяет на ранних этапах продемонстрировать и протестировать будущий интерфейс, выявить проблемы навигации и восприятия. Для этого используют инструменты Figma, Adobe XD, Balsamiq. Тестирование интерфейсов с участием пользователей даёт возможность оценить удобство и выявить непредвиденные сценарии использования. Прототипы ускоряют согласование с заказчиком и минимизируют риск затрат на переработку уже реализованных элементов.

Практические задания:

1. Создайте прототип пользовательского интерфейса в Figma,
2. Разработайте макет экрана авторизации с учетом UX-принципов,
3. Проведите тестирование прототипа с участием 2–3 пользователей.

Лабораторная работа 8. Управление техническим долгом, рефакторинг кода, поддерживаемость и масштабируемость приложений

Теоретическая часть:

Технический долг возникает, когда разработчики в целях быстроты реализации жертвуют архитектурной чистотой, читаемостью и гибкостью кода, что в будущем приводит к увеличению затрат на сопровождение и развитие системы. Управление техническим долгом включает его выявление, приоритизацию и планомерное устранение. Рефакторинг представляет собой процесс улучшения внутренней структуры кода без изменения его внешнего поведения. Это может быть выделение функций, устранение дублирования, упрощение логики, внедрение паттернов и переименование сущностей. Поддерживаемость и масштабируемость напрямую зависят от архитектурных решений, читаемости, автоматизации и тестов. Внедрение анализа качества кода, использование статического анализа и покрытие тестами помогают отслеживать техническое состояние проекта. Плановое выделение времени на устранение долга становится залогом устойчивого развития программного

продукта.

Практические задания:

1. Определите технический долг в проекте и составьте план его устранения,
2. Проведите рефакторинг устаревшего модуля с добавлением комментариев,
3. Оцените влияние технического долга на производительность команды.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению курсового проекта по дисциплине

«**Технологии разработки приложений**»

для студентов специальности

10.05.03 Информационная безопасность автоматизированных систем

Введение

Настоящие рекомендации предназначены для студентов всех форм обучения при выполнении курсовых работ, для руководителей курсовых работ с целью формирования единых требований при разработке и защите работ. Рекомендации содержат предложения по структуре курсовой работы (далее - работа), объему, содержанию, оформлению, процедуре допуска к защите и порядку защиты работ.

Целями выполнения курсовой работы являются:

- обогащение знаний студентов,
- обучение методам теоретического анализа явлений и закономерностей науки,
- выработка навыков применения теоретических знаний к комплексному решению профессиональных задач, использования справочной литературы, методов математической обработки экспериментальных данных, компьютерных технологий.

Системой курсовых работ (проектов) студент подготавливается к выполнению выпускной квалификационной работы.

Перед прочтением Методических указаний ознакомьтесь с основными ошибками, которые допускают студенты, приступая к написанию курсовой работы. Возможно, это мотивирует на более внимательное отношение к требованиям по написанию курсовой работы.

1. Выбрав тему, студент выходит в сеть Интернет и через поисковую систему (Yandex, Google и т.д. – зависит от предпочтений) находит курсовую работу по данной теме, меняет реквизиты, Ф.И.О. автора и сдаёт преподавателю. В результате получает рецензию: «Плагиат. На переработку».
2. Аналогичные действия п.1, но студент подходит к решению проблемы творчески и находит несколько курсовых работ по данной теме. Компонует материал по своему усмотрению и сдает преподавателю под своим авторством. В результате получает рецензию: «Плагиат. На переработку». Почему дана такая рецензия? По трем основным причинам: во-первых, написание курсовых работ – распространённый бизнес в Интернете и качественный материал бесплатно не выкладывают в сети, таким образом, компоновать приходится из «третьесортного сырья»; во-вторых, курсовая уже написана и защищена ранее, то есть она уже устарела и не отражает современных реалий, в-третьих, такие работы редко соответствуют требованиям, предъявляемым кафедрой к курсовым работам.
3. Студент принимает постороннюю помощь в написании работы. Здесь есть несколько аспектов, но следует обратить внимание на два основных. Во-первых, помощь исходит, как правило, от одного физического или юридического лица для группы студентов. В результате преподаватель получает на проверку несколько работ на разные темы, написанные «одной рукой». Формальных причин для отрицательной рецензии у преподавателя нет, особенно если работа выполнена в строгом соответствии с требованиями, но, как показывает практика, защита таких работ проходит безуспешно. Во-вторых, студент, не приложивший усилий к данной работе, как правило, очень плохо в ней ориентируется, особенно, когда преподаватель уточняет аспекты рассматриваемой темы. Но так как Вы уже вышли на защиту, то получаете заслуженную оценку «неудовлетворительно».
4. Вы самостоятельно осуществляете работу, но так же самостоятельно выбираете требования к курсовой работе. Если Вам они нравятся и понятны, то исполняете, а если не нравятся, пропускаете. Это очень распространённая ошибка. Выполнять надо все без исключения требования к курсовой работе. Если в процессе работы возникли вопросы, следует по ним получить консультацию у научного руководителя.

Это краткий перечень основных ошибок, которые недопустимо совершать после прочтения данных Методических указаний.

1. ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ И ЗАЩИТЫ КУРСОВОЙ РАБОТЫ

Виды работ по этапам выполнения курсовой работы:

а) Предварительный этап:

- выбор темы курсовой работы и оценка возможности раскрытия данной темы;
- подача заявления на закрепление темы;

б) Основной этап:

- сбор материала;
- оформление курсовой работы;

в) Заключительный этап:

- рецензирование работы руководителем и допуск к защите;
- защита работы.

1.1 Предварительный этап

Темы курсовых работ определяются выпускающей кафедрой. Студенту предоставлено право выбора темы курсовой работы. Задание к выбранной теме выдается руководителем работы (проекта) и регистрируется в кафедральном журнале.

1.2 Основной этап

Составление плана курсовой работы

План курсовой работы представляет собой составленный в определенном порядке перечень глав и развернутый перечень вопросов, которые должны быть освещены. Правильно составленный план работы помогает систематизировать материал, обеспечивает последовательность его изложения. План работы составляется самостоятельно, с учетом замысла и индивидуального подхода. План работы рекомендуется **согласовать с руководителем курсовой работы**. В процессе работы план работы может уточняться.

Подбор литературы

Курсовая работа выполняется на основе глубокого изучения литературных источников. Литература по теме работы может быть подобрана студентом при помощи предметных и алфавитных каталогов библиотек. Для этих целей могут быть использованы каталоги книг, указатели журнальных статей, специальные библиографические справочники, тематические сборники литературы, периодически выпускаемые отдельными издательствами, интернет-сайты официальных организаций.

Проработка источников сопровождается выписками, конспектированием. Выписки из текста делают обычно в виде цитаты. После каждой цитаты приводится ссылка на автора и источник. Поэтому при выписке цитат и конспектировании следует делать ссылки: автор, название издания, место издания, издательство, год издания, номер страницы. Конспект может содержать в себе факты, доказательства, примеры, основные положения и выводы автора, а также личное отношение студента к изученному материалу.

Рациональной и эффективной организации исследовательской деятельности способствует составление студентом собственной картотеки проработанных по теме и смежным вопросам литературных источников. Картотека не только помогает получить представление о степени изученности данной проблемы, но и позволяет работать с литературой более целеустремленно.

Оформление курсовой работы

Оформление проекта или работы осуществляется в соответствии с требованиями положения о выполнении и защите курсовых о выполнении и защите курсовых работ (проектов).

Содержание и структура курсовых работ

Курсовая работа должна содержать элементы новизны, наряду с фундаментальным аспектом должен быть проведен анализ современного состояния изучаемой проблемы, а также отражены региональные особенности. Задание по курсовой работе необходимо индивидуализировать с учетом интересов и способностей студентов.

Курсовая работа должна состоять из введения, теоретической части, эмпирической (практической, расчетно-графической) части, заключения, списка литературы и приложения. В отдельных случаях, в соответствии с тематикой работы, эмпирическая часть может отсутствовать.

Во введении обосновывается актуальность выбранной темы исследования, задачи, цели, новизна, теоретическая и практическая значимость исследования.

Теоретическая часть должна содержать анализ состояния изучаемой проблемы на основе обзора научной, научно-информационной, справочной литературы. Представленный материал должен быть логически связан с целью исследования. В параграфах теоретической части необходимо отражать отдельные компоненты проблемы и завершать их выводами.

Эмпирическая (практическая, расчетно-графическая) часть (при наличии) включает описание системы экспериментального исследования, обоснование методов исследования, анализ результатов экспериментального исследования, схемы, графические и математические способы интерпретации полученных данных, выводы.

Заключение содержит выводы, подтверждающие или опровергающие первоначальные предположения (гипотезы), перспективы дальнейшего изучения проблемы, связь с практикой, анализ реализации целей и задач исследования.

Список литературы должен быть составлен в соответствии с требованиями к оформлению библиографий.

Приложение содержит весь фактический материал экспериментальных исследований (анкеты, опросники, схемы, чертежи, расчетные материалы, карты, рисунки, ответы респондентов и т.д.).

Содержание курсовой работы (проекта) рекомендуется представлять в объеме 20-30 страниц печатного текста. Текст работы должен быть напечатан через 1,5 интервала на одной стороне стандартного листа белой бумаги (А - 4). Текст и другие отпечатанные элементы работы должны быть черными, контуры букв и знаков – четкими, без ореола и затенения. Шрифт TimesNewRoman, кегель 14. Курсив и подчеркивание в работе не допускаются. Названия глав и параграфов выделяются полужирным шрифтом. Лист с текстом должен иметь поля: слева – 30 мм, справа – 10 мм, сверху – 20 мм, снизу – 20 мм. Нумерация страниц текста делается в правом нижнем углу листа. Проставлять номер страницы необходимо со страницы, где печатается "Введение", на которой ставится цифра "3". После этого нумеруются все страницы, включая приложения.

Между названием главы и названием параграфа этой главы ставится пробел равный двум интервалам, а название параграфа не должно отделяться от текста этого параграфа пробелом. Названия параграфов отделяются от текста предыдущего параграфа пробелом, равным двум интервалам. Каждая глава, а также введение, выводы, предложения и список использованной литературы начинаются с новой страницы. Слово "Глава" не пишется. Главы имеют порядковые номера в пределах всей работы, обозначаемые арабскими цифрами (например: 1,2,3), после которых ставится точка. Слово "параграф" или значок параграфа в названии не ставятся. Параграфы имеют порядковые номера в пределах глав, обозначаемые арабскими цифрами (например: 1.1. и 1.2.). Заголовки глав и параграфов в тексте работы должны располагаться по центру, точку в конце названия главы и параграфа не ставят. Не допускается переносить часть слова в заголовке.

Нумерация таблиц и рисунков может быть сквозной или соотноситься с номером главы и параграфа. Например, если таблица или рисунок включены в текст первого параграфа второй главы, нумерация следующая: Таблица 2.1.1., рис. 2.1.1. Последняя цифра означает порядковый номер таблицы (или рисунка) в данном параграфе. Таблица помещается в качестве следующей страницы после первого упоминания о ней в тексте.

В рамках отведенного для руководства курсовыми работами времени регулярно проводятся индивидуальные консультации. Руководитель работы осуществляет предварительный просмотр выполненных заданий. После проверки выполнения студентом одного этапа работы, руководитель работы разрешает студенту перейти к следующему.

1.3 Заключительный этап

Работа сдается руководителю. Если работа удовлетворяет требованиям, предъявляемым к курсовым работам, она допускается к защите.

Защита курсовой работы является обязательной формой проверки выполнения работы. Защита производится на заседании кафедры, научно-методического семинара кафедры, научной проблемной группы при непосредственном участии руководителя, в присутствии студентов. Результаты наиболее интересных курсовых работ могут быть доложены на научных конференциях.

Защита состоит в коротком докладе студента по выполненной работе и в ответах на вопросы присутствующих на защите.

Результаты защиты курсовой работы, согласно действующему Положению о текущем контроле и промежуточной аттестации, оцениваются дифференцированной отметкой по пятибалльной системе.

Студент, не представивший в установленный срок работу, получивший неудовлетворительную оценку на защите или не явившийся на защиту по неуважительной причине, считается имеющим академическую задолженность.

Для ликвидации академической задолженности студент обязан в установленные сроки представить курсовую работу руководителю и защитить её перед комиссией.

Допуск к защите курсовой работы

Допуск к защите осуществляет научный руководитель курсовой работы.

Работа не допускается к защите, если она не носит самостоятельного характера, списана из литературных источников или у других авторов, если основные вопросы не раскрыты, изложены

схематично, фрагментарно, в тексте содержатся ошибки, научный аппарат оформлен неправильно, текст написан небрежно.

Неудовлетворительно выполненная работа подлежит переработке в соответствии с замечаниями преподавателя, содержащимися в отзыве.

При получении допуска к защите студент вправе получить дополнительную консультацию по предстоящей защите.

Курсовая работа, студенту не возвращается и хранится на кафедре в течение 2-х лет.

После получения допуска к защите, студент самостоятельно готовится к защите: составляет текст доклада, реагирует на замечания руководителя.

Подготовка к защите курсовой работы включает устранение ошибок и недостатков, изучение дополнительных источников, готовность объяснить любые приведенные в работе положения.

В ходе защиты курсовой работы задача студента - показать углубленное понимание вопросов конкретной темы, хорошее владение материалом по теме.

Защита работы производится в соответствии с планом приёма защит курсовых работ. С ним можно ознакомиться на стенде кафедры или непосредственно уточнить у научного руководителя время и место защиты.

По результатам защиты курсовой работы в зачётную ведомость (для защиты курсовой работы) выставляется оценка.

Критерии оценки курсовой работы:

- «**отлично**» выставляется студенту, показавшему глубокие знания, которые применяются при самостоятельном исследовании избранной темы, умение обобщать практический материал и делать на основе анализа выводы. Курсовая работа с оценкой «отлично» может быть рекомендована на конкурс студенческих работ, использована в качестве доклада на научно-студенческой конференции.
- «**хорошо**» выставляется студенту, показавшему в работе и при её защите полное знание материала, всесторонне осветившему вопросы темы, но не полной мере проявившему самостоятельность в исследовании.
- **«удовлетворительно»** выставляется студенту, раскрывшему в работе основные вопросы избранной темы, но не проявившему самостоятельности в анализе или допустившему отдельные неточности содержания работы.
- «**неудовлетворительно»** выставляется студенту, не раскрывшему основные положения избранной темы и допустившему грубые ошибки в содержании работы.

2. СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

2.1 Структурные элементы курсовой работы

Структурными элементами курсовой работы являются:

1) титульный лист,
2) оглавление,
5) введение;
6) основная часть;
7) заключение;
8) список использованных источников;
9) приложения.

Титульный лист является первой страницей курсовой работы.

Оглавление включает введение, наименование всех глав и параграфов, заключение, список использованных источников и приложения с указанием номеров страниц, с которых начинаются эти элементы работы. Следует обратить внимание, что через оглавление прослеживается структура всей работы. Желательно основную часть работы уместить в 2 главы, каждая из которых разбивается на 2-3 параграфа. Глава не может содержать один параграф. Наличие в главе более трех параграфов говорит о поверхностном представлении данных вопросов, так как к работе предъявляются жесткие требования по объему работы.

Введение. Во введении обосновывается **актуальность темы**, **формулируются цель** курсовой работы **и комплекс взаимосвязанных задач**, подлежащих решению в процессе работы, а также представляется **теоретическая и методологическая база.**

Актуальность темы отражает степень важности её раскрытия на данный момент, то есть,

почему читателю будет интересно, важно или просто необходимо познакомиться с Вашей работой. Это 2-3 предложения, которые анонсируют Вашу работу. Важно во введении**,** раскрывая актуальность темы, не уйти в раскрытие заявленной проблемы. Не следует писать фразы типа: «Мне эта тема очень интересна, поэтому она актуальна». Здесь необходимо заинтересовать читателя предметно и содержательно, а не своим примером.

Цель работы – это результат, к которому Вы желаете прийти и путь к нему. Цель работы одна. Формулируем её с использованием неопределённой формы глагола: изучить, исследовать, проанализировать, рассмотреть и т.д. Цель вашей курсовой работы скрыта в названии работы.

Задачи работы отражают шаги, делая которые, Вы достигаете цели. Вы должны поставить 6 - 7 задач. Они фактически связаны с параграфами Вашей работы. Задачи формулируются аналогично цели с использованием неопределенной формы глагола: изучить, исследовать, проанализировать, рассмотреть, оценить, выявить и т.д.

Теоретическая и методологическая база подразумевает краткое описание использованных источников с указанием основных авторов. Это «поклон» в адрес людей, чьими трудами Вы воспользовались, а также возможность для специалиста с первого взгляда понять, о чем написана работа и какого подхода к данной проблеме придерживаетесь Вы.

Объем этой части работы 1 лист.

Так как введение – это та часть работы, которая подразумевает 100% самостоятельное изложение, важно не перейти на «ты» в общении с подразумеваемым читателем, то есть **всё изложение материала должно осуществляется в неопределенной форме глагола**. Нельзя употреблять местоимение «я».

Основная часть. Требования к содержанию основной части работы даны ранее.

Заключение. В заключении формулируются краткие выводы, полученные в ходе выполнения поставленных задач и цели. Вы кратко излагаете то, что написали в работе, но это самое главное, что вы хотите донести до читателя, самый «сок» Вашего труда. В заключение можно вынести числовые данные в целях иллюстрации вывода. Они должны отражать цель и задачи исследования, и не быть «вырванными» из работы числами.

Список использованных источников должен содержать сведения об источниках, использованных при выполнении работы.

Приложения. В приложении рекомендуется включать материалы, связанные с выполненной работой, которые по каким-либо признакам не могут быть включены в основную часть.

2.2 Содержание основной части курсовой работы

Теоретический раздел

Раздел должен иметь наименование, отражающее теоретические аспекты темы курсовой работы. В общем случае в этом разделе должны быть отражены:

1) характеристика предмета исследования в соответствии темой курсовой работы, в частности, должен присутствовать анализ понятийной базы по заявленной теме.

2) анализ состояния вопроса, являющегося темой работы, при необходимости – методы анализа.

3) основные подходы к решению проблемы, являющейся темой курсовой работы.

Теоретический раздел должен давать ответы на вопросы, что представляет собой проблема, являющаяся темой курсовой работы.

Объем раздела – 10 - 12 с. В разделе должно быть **не менее двух и не более трех подразделов.**

Аналитический раздел

Наименование данного раздела должно быть связано с анализом состояния предмета исследования, являющегося темой курсовой работы.

Исследование проблемы должно быть целевым и глубоким.

В аналитическом разделе используются различные приемы анализа, указываются источники информации, приводятся аналитические таблицы и расчеты, графические способы отражения результатов анализа, делаются выводы.

В данном разделе обязательно использование не только статистического материала, но и результатов анализа, проведенного специалистами по данному вопросу, которые изложены в источниках периодической печати.

Объем аналитического раздела курсовой работы не должен превышать 10-12 страниц.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Перечень основной литературы:

1. Галатенко В.А. Основы информационной безопасности. – М.: Интернет-Университет Информационных Технологий, 2020.
2. Петраков А.В. Защита информации в компьютерных системах и сетях. – М.: Радио и связь, 2021.
3. Шаньгин В.Ф. Комплексная защита информации в корпоративных системах. – М.: ДМК Пресс, 2019.
4. Лопатин В.Н. Информационная безопасность России. – СПб.: Юридический центр Пресс, 2018.
5. Белов Е.Б., Лось В.П. Основы информационной безопасности. – М.: Горячая линия – Телеком, 2020.

Перечень дополнительной литературы:

6. Федеральный закон от 27.07.2006 № 149-ФЗ "Об информации, информационных технологиях и о защите информации".
7. Федеральный закон от 27.07.2006 № 152-ФЗ "О персональных данных".
8. Федеральный закон от 26.12.1995 № 208-ФЗ "О безопасности".
9. Приказы ФСТЭК России
10. ГОСТ Р 50922-2006 "Защита информации. Основные термины и определения".
11. ГОСТ Р 57580.1-2017 "Безопасность финансовых организаций".