Приложение 1.2.1 к ОПОП по специальности
09.02.11 Разработка и управление программным обеспечением

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ МЕЖДИСЦИПЛИНАРНОГО КУРСА

МДК 02.02 Осуществление интеграции программных модулей

для обучающихся специальности
09.02.11 Разработка и управление программным обеспечением

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют лабораторные занятия.

Состав и содержание лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой профессионального модуля «Разработка и интеграция модулей программного обеспечения» предусмотрено проведение лабораторных занятий.

В результате их выполнения, обучающийся должен:

уметь:

- У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие;
- У 2.3.2 работать с API и устанавливать соединения между компонентами;
- У 2.3.3 отслеживать и устранять конфликты и ошибки интеграции;
- У 2.3.4 анализировать и определять зависимости между модулями и компонентами;
- У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Содержание практических и лабораторных занятий ориентировано на освоение вида деятельности программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями:***

ПК 2.3 Выполнять интеграцию модулей и компонентов программного обеспечения.

А также формированию общих компетенций:

ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности.

Лабораторные занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 2.1. Основы интеграции программных модулей
Лабораторное занятие №37
Создание клиентского приложения для работы с публичным API

Цель: научиться разрабатывать клиентские приложения для взаимодействия с внешними веб-сервисами через REST API

Выполнив работу, вы будете уметь:
У 2.3.2 работать с API и устанавливать соединения между компонентами.

Материальное обеспечение:
WebStorm, Postman

Задание:
1. Выбрать публичное API и изучить его документацию;
2. Настроить проект и установить необходимые зависимости для работы с HTTP-запросами;
3. Реализовать отправку запросов и парсинг полученных данных (JSON/XML);
4. Вывести структурированную информацию в интерфейс приложения и добавить обработку ошибок.

Порядок выполнения работы:
1. Ознакомиться с документацией выбранного публичного API и протестировать endpoints через Postman или браузер;
2. Создать проект в выбранной среде разработки и подключить библиотеки для работы с сетью;
3. Написать код для формирования GET/POST-запросов с необходимыми параметрами и заголовками;
4. Реализовать обработку ответов сервера (статус-коды, таймауты, невалидные данные) и вывод результатов;
5. Протестировать приложение, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №38
Создание REST API приложения с реализацией: добавления, удаления, изменения и создания данных

Цель: научиться разрабатывать серверные REST API приложения с полной реализацией операций создания, чтения, обновления и удаления данных (CRUD)

Выполнив работу, вы будете уметь:
У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие.

Материальное обеспечение:
WebStorm, Postman, Node.js

Задание:
1. Разработать структуру хранилища данных для тестовой сущности;
2. Настроить серверное приложение и определить маршруты (endpoints) для CRUD-операций;
3. Реализовать обработчики запросов с валидацией данных и корректными HTTP-статусами;
4. Протестировать все методы API и убедиться в корректности создания, чтения, обновления и удаления данных.

Порядок выполнения работы:
1. Выбрать технологический стек и создать проект серверного приложения;
2. Настроить подключение к базе данных или файловому хранилищу;
3. Реализовать маршруты для методов GET, POST, PUT/PATCH, DELETE с соответствующей бизнес-логикой;
4. Протестировать API с помощью Postman, проверить статус-коды и валидацию;
5. Оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №39
Расширение функционала REST API приложения: работа с удаленным источником данных

Цель: научиться интегрировать внешние и удаленные источники данных в REST API-приложение для обеспечения масштабируемости, отказоустойчивости и работы с распределенными данными

Выполнив работу, вы будете уметь:
У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Материальное обеспечение:
WebStorm, Postman, HTTP-клиенты, MySQL

Задание:
1. Изучить документацию и схемы данных выбранного удаленного источника (внешняя СУБД, облачный сервис или стороннее API);
2. Настроить безопасное подключение, аутентификацию и хранение параметров соединения (переменные окружения, секреты);
3. Интегрировать удаленный источник в REST API и реализовать маршруты для получения, сохранения и обновления данных через него;
4. Добавить механизмы обработки сетевых ошибок, таймаутов, повторных попыток подключения и/или кэширования ответов.

Порядок выполнения работы:
1. Проверить доступность удаленного источника данных с помощью тестовых утилит (curl, Postman, DB-клиент);
2. Настроить переменные окружения и подключить клиентскую библиотеку удаленного источника к проекту API;
3. Реализовать обработчики маршрутов, выполняющие запросы к удаленному источнику и возвращающие стандартизированные ответы;
4. Внедрить обработку исключений подключения, настроить логирование и добавить кэширование или механизм повторных запросов;
5. Протестировать расширенный функционал, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №40
Расширение функционала REST API приложения: работа со статическими изображениями (ресурсами) - загрузка, передача, удаление

Цель: научиться расширять функционал REST API-приложений для безопасной обработки, хранения и передачи статических файлов (изображений) с реализацией операций загрузки, скачивания и удаления

Выполнив работу, вы будете уметь:
У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Материальное обеспечение:
WebStorm, Postman, Node.js

Задание:
1. Настроить серверное приложение для приема multipart/form-data запросов и валидации загружаемых файлов;
2. Реализовать маршрут загрузки изображений с сохранением в локальную папку или внешнее хранилище;
3. Реализовать маршруты для передачи (отдачи) файлов клиенту и удаления файлов по идентификатору;
4. Протестировать полный цикл работы с ресурсами и обеспечить корректные HTTP-статусы и заголовки.

Порядок выполнения работы:
1. Подготовить проект и создать директорию для хранения загружаемых файлов (или настроить подключение к облачному хранилищу);
2. Настроить обработчик POST-запроса для приема файлов с проверкой типа, размера и безопасного именования;
3. Реализовать GET-запрос для отдачи статических ресурсов и DELETE-запрос для удаления файлов с проверкой существования и корректными ответами сервера;
4. Настроить отдачу правильных MIME-типов, базовое кэширование и обработку исключений (файл не найден, неверный формат, превышение лимита);
5. Протестировать API через Postman, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №41
Расширение функционала REST API приложения: обработка path и query параметров

Цель: научиться проектировать и реализовывать маршруты REST API с использованием path и query параметров для идентификации, фильтрации, сортировки и пагинации данных

Выполнив работу, вы будете уметь:
У 2.3.4 анализировать и определять зависимости между модулями и компонентами.

Материальное обеспечение:
WebStorm, Postman, Node.js

Задание:
1. Спроектировать маршруты API с использованием path-параметров для идентификации конкретных ресурсов;
2. Реализовать обработку query-параметров для фильтрации, сортировки и пагинации данных;
3. Настроить валидацию входных параметров и обработку ошибок при некорректных или отсутствующих значениях;
4. Протестировать комбинации path и query параметров, убедиться в корректности ответов сервера и HTTP-статусов.

Порядок выполнения работы:
1. Изучить документацию фреймворка по работе с маршрутами и параметрами запроса;
2. Создать базовые эндпоинты с path-параметрами (например, /items/{id}, /categories/{name}/products);
3. Добавить поддержку query-параметров для поиска, фильтрации по полям, сортировки и разбивки на страницы (page, limit, sort, order);
4. Реализовать валидацию типов, диапазонов и обязательности параметров, настроить возврат ошибок 400 Bad Request или 404 Not Found;
5. Протестировать API с помощью Postman/curl, проверить граничные случаи, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №42
Расширение функционала REST API приложения: обработка ошибок, передача сообщений об ошибке пользователю

Цель: освоить принципы централизованной обработки исключений в REST API, научиться стандартизировать форматы ответов об ошибках и обеспечивать безопасную передачу понятных сообщений клиенту без раскрытия внутренней архитектуры приложения

Выполнив работу, вы будете уметь:
У 2.3.3 отслеживать и устранять конфликты и ошибки интеграции.

Материальное обеспечение:
WebStorm, Postman, Node.js

Задание:
1. Реализовать централизованный обработчик ошибок (middleware/exception handler) для перехвата исключений на всех уровнях маршрутизации и бизнес-логики;
2. Стандартизировать формат ответов об ошибках, протестировать обработку некорректных запросов, отсутствующих ресурсов и внутренних сбоев, зафиксировать корректность HTTP-статусов и безопасность возвращаемых сообщений.

Порядок выполнения работы:
1. Изучить теоретические основы обработки ошибок в REST API: классификация кодов состояний HTTP, стандарты структурированных ошибок, принципы глобального перехвата исключений, правила безопасного логирования и сокрытия stack trace/путей к БД.
2. Ответить на вопросы: В чём разница между ошибками 400, 401, 403, 404 и 500? Почему клиент должен получать только контекстные сообщения об ошибке, а технические детали оставаться на сервере? Как централизованный обработчик упрощает поддержку, мониторинг и тестирование API?
3. Изучить структуру проекта: выделить слой валидации входных данных, слой бизнес-логики, глобальный обработчик исключений, форматировщик JSON-ответов и конфигурацию логирования.
4. Ознакомиться с типовым алгоритмом реализации: перехват исключения → определение типа/причины → маппинг на соответствующий HTTP-статус → формирование стандартизированного JSON-ответа (код, сообщение, поля с деталями) → запись технических деталей в серверный лог → возврат ответа клиенту.
5. Реализовать обработчик, выполнить серию тестовых запросов: неверный формат тела запроса, отсутствие ресурса по ID, попытка доступа без авторизации, имитация падения БД, убедиться в стабильности работы и отсутствии «сырых» исключений в ответе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №43
Расширение функционала REST API приложения: валидация полученных данных

Цель: освоить принципы и механизмы серверной валидации входящих данных в REST API, научиться реализовывать проверку форматов, типов и бизнес-правил запросов для обеспечения целостности данных, безопасности и отказоустойчивости модуля

Выполнив работу, вы будете уметь:
У 2.3.4 анализировать и определять зависимости между модулями и компонентами.

Материальное обеспечение:
WebStorm, Postman, Node.js

Задание:
1. Реализовать middleware/декораторы для валидации тела запроса (body), параметров пути (path) и строки запроса (query) на ключевых эндпоинтах API с использованием схем или пользовательских правил;
2. Протестировать обработку валидных и невалидных данных, убедиться в корректности HTTP-статусов, детализации сообщений об ошибках и отсутствии выполнения бизнес-логики при сбоях валидации.

Порядок выполнения работы:
1. Изучить теоретические основы валидации на сервере: типы проверок (синтаксическая, семантическая, бизнес-логика), принципы санитизации, риски отсутствия валидации (SQL/NoSQL-инъекции, поломка БД, уязвимости XSS, перегрузка ресурсов), популярные подходы и библиотеки.
2. Ответить на вопросы: Почему клиентской валидации недостаточно для защиты API? Как правильно комбинировать строгую проверку типов с гибкими бизнес-правилами? В чём разница между валидацией и санитизацией, и почему обе необходимы?
3. Изучить структуру проекта: выделить слой маршрутизации, слой валидации (схемы/правила), слой бизнес-логики и глобальный обработчик ошибок.
4. Ознакомиться с типовым алгоритмом реализации: перехват запроса → парсинг тела/параметров → применение схем валидации → при успехе передача данных в контроллер; при ошибке формирование ответа 400 Bad Request с деталями нарушений → возврат клиенту без выполнения бизнес-логики.
5. Реализовать валидацию для CRUD-эндпоинтов, выполнить серию тестовых запросов (отсутствующие обязательные поля, неверные типы данных, выход за границы диапазонов, недопустимые символы/паттерны), убедиться в стабильности API и корректности ответов.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №44
Расширение функционала REST API приложения: добавление фоновых задач

Цель: научиться интегрировать механизмы асинхронной обработки и фоновых задач в REST API-приложение для выполнения длительных или ресурсоемких операций без блокировки HTTP-запросов

Выполнив работу, вы будете уметь:
У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие.

Материальное обеспечение:
WebStorm, Postman, Redis

Задание:
1. Настроить брокер сообщений и подключить его к серверному приложению;
2. Реализовать маршруты API для постановки фоновых задач в очередь и проверки их статуса;
3. Создать воркер (worker) с логикой выполнения длительной или ресурсоемкой операции;
4. Протестировать асинхронный цикл: запрос → постановка в очередь → выполнение → получение результата/статуса.

Порядок выполнения работы:
1. Развернуть брокер сообщений (Redis/RabbitMQ) локально или в Docker-контейнере, проверить доступность порта и соединение;
2. Интегрировать клиентскую библиотеку очереди задач в проект API и настроить конфигурацию подключения;
3. Реализовать POST-endpoint для приема параметров и постановки задачи в очередь, а также GET-endpoint для проверки статуса и получения результата;
4. Написать код воркера, имитирующий длительную операцию (обработка данных, генерация отчета, отправка уведомления), добавить логирование и обработку ошибок;
5. Протестировать взаимодействие API и воркера через Postman/curl, убедиться в неблокирующем характере HTTP-запросов, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №45
Расширение функционала REST API приложения: добавление аутентификации и авторизации, создание ролевой системы

Цель: научиться интегрировать механизмы аутентификации и авторизации в REST API-приложение, реализовывать безопасную работу с учетными данными и проектировать ролевую модель доступа (RBAC) для защиты маршрутов

Выполнив работу, вы будете уметь:
У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Материальное обеспечение:
WebStorm, Postman, Node.js, MySQL

Задание:
1. Разработать схему базы данных для хранения пользователей, ролей и их связей;
2. Реализовать endpoints регистрации и аутентификации с безопасным хешированием паролей и выдачей токенов доступа;
3. Настроить middleware/гарды для проверки валидности токена и применения ролевых ограничений к защищенным маршрутам;
4. Протестировать сценарии доступа для разных ролей и убедиться в корректной работе механизмов защиты.

Порядок выполнения работы:
1. Создать таблицы/коллекции для пользователей и ролей, настроить подключение СУБД к проекту;
2. Реализовать маршруты регистрации и входа, добавить хеширование паролей и генерацию JWT-токенов (или сессионных куки);
3. Разработать middleware для проверки токена, извлечения данных пользователя и привязки ролей к конкретным endpoint'ам;
4. Настроить ограничения доступа (например, администратор, модератор, пользователь), протестировать API с разными учетными записями через Postman;
5. Проверить обработку ошибок доступа, логирование попыток несанкционированного входа, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №46
Создание клиентского приложения для работы с публичным WebSocket

Цель: научиться разрабатывать клиентские приложения для организации двусторонней связи в реальном времени с использованием протокола WebSocket

Выполнив работу, вы будете уметь:
У 2.3.2 работать с API и устанавливать соединения между компонентами.

Материальное обеспечение:
WebStorm, Postman, WebSocket

Задание:
1. Выбрать публичный WebSocket-сервер и изучить документацию по формату и структуре передаваемых сообщений;
2. Настроить клиентский проект и реализовать установку WebSocket-соединения;
3. Реализовать отправку и прием сообщений, обработку событий соединения и парсинг входящих данных;
4. Добавить механизмы обработки обрывов связи, повторного подключения и вывода данных в интерфейс приложения.

Порядок выполнения работы:
1. Проверить доступность выбранного публичного WebSocket-сервера через браузер или тестовые утилиты;
2. Создать клиентский проект, подключить необходимую библиотеку для работы с WebSocket и настроить параметры подключения;
3. Реализовать обработчики событий: открытие соединения, получение сообщений, закрытие соединения и возникновение ошибок;
4. Написать логику отправки тестовых сообщений/подписки на поток данных, реализовать парсинг JSON-ответов и механизм реконнекта при обрыве связи;
5. Протестировать стабильность соединения и корректность обработки данных в реальном времени, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №47
Создание серверного приложения для работы по WebSocket протоколу

Цель: научиться разрабатывать серверные приложения для поддержки двусторонней связи в реальном времени по протоколу WebSocket, управлять пулом подключений и маршрутизировать сообщения между клиентами

Выполнив работу, вы будете уметь:
У 2.3.2 работать с API и устанавливать соединения между компонентами.

Материальное обеспечение:
WebStorm, Postman, WebSocket

Задание:
1. Настроить серверную среду и выбрать фреймворк/библиотеку для реализации WebSocket-сервера;
2. Реализовать базовый сервер с обработкой событий подключения, получения сообщений и отключения клиентов;
3. Добавить функционал маршрутизации сообщений (широковещательная рассылка или отправка конкретному клиенту) и управление списком активных сессий;
4. Протестировать серверное приложение с несколькими параллельными клиентами, проверить стабильность соединения и корректность обработки данных.

Порядок выполнения работы:
1. Создать проект сервера, установить необходимые зависимости и настроить базовую конфигурацию (порт, хост);
2. Реализовать инициализацию WebSocket-сервера и обработчик события connection (сохранение идентификатора/объекта сессии в пул активных клиентов);
3. Написать логику обработки входящих сообщений: парсинг, валидация, отправка ответов выбранному клиенту или всем подключенным участникам;
4. Реализовать механизмы поддержания соединения (heartbeat ping/pong), обработку корректного закрытия сессии и перехват сетевых исключений;
5. Протестировать работу сервера через несколько клиентских соединений, убедиться в корректной маршрутизации и отсутствии утечек памяти/зависаний, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №48
Создание микросервисного приложения с взаимодействием по REST

Цель: научиться проектировать и разрабатывать распределённые микросервисные приложения с организацией межсервисного взаимодействия по протоколу REST, обеспечивая независимое развертывание и согласованную передачу данных между компонентами

Выполнив работу, вы будете уметь:
У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие;
У 2.3.4 анализировать и определять зависимости между модулями и компонентами.

Материальное обеспечение:
WebStorm, Postman, WebSocket

Задание:
1. Разработать архитектуру из двух и более независимых микросервисов с чётко определёнными зонами ответственности;
2. Реализовать REST API для каждого сервиса и настроить межсервисные HTTP-запросы для обмена данными;
3. Настроить сетевое взаимодействие между сервисами, обеспечить валидацию и обработку ошибок при межсервисных вызовах;
4. Протестировать сквозной сценарий взаимодействия, убедиться в корректности передачи данных и независимости развёртывания компонентов.

Порядок выполнения работы:
1. Спроектировать схему взаимодействия микросервисов, определить endpoints, форматы запросов/ответов и последовательность вызовов;
2. Создать отдельные проекты/модули для каждого микросервиса, реализовать базовые CRUD-операции и внутренние бизнес-процессы;
3. Интегрировать HTTP-клиент в вызывающий сервис, настроить маршрутизацию, обработку статус-кодов, таймаутов и механизмы повторных попыток (retry);
4. Упаковать сервисы в Docker-контейнеры (опционально), настроить Docker Compose для совместного запуска и сетевого взаимодействия, проверить доступность endpoints;
5. Протестировать сквозной сценарий через Postman/curl, зафиксировать результаты взаимодействия, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Тема 2.2. Управление и мониторинг интегрированной системы
Лабораторное занятие №49
Настройка конфигурации REST API приложения

Цель: научиться проектировать и внедрять системы конфигурирования REST API-приложений для гибкого управления параметрами среды, безопасного хранения секретов и поддержки различных профилей развёртывания (dev, test, prod)

Выполнив работу, вы будете уметь:
У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие.

Материальное обеспечение:
WebStorm, Postman, WebSocket

Задание:
1. Изучить принципы внешнего конфигурирования и выбрать подходящий формат/инструмент для управления настройками приложения;
2. Разделить конфигурацию на профили окружений (разработка, тестирование, продакшн) и вынести конфиденциальные данные в переменные окружения;
3. Подключить систему конфигурирования к REST API-приложению и настроить чтение параметров для БД, логирования, сетевых параметров и политик безопасности;
4. Реализовать валидацию конфигурации при запуске и протестировать переключение между профилями окружения.

Порядок выполнения работы:
1. Создать структуру проекта, подготовить шаблоны файлов конфигурации (.env, config.yaml, appsettings.json и др.) с чётким разделением по окружениям;
2. Настроить загрузчик конфигурации, реализовать чтение переменных окружения и маппинг параметров в структуры/объекты приложения;
3. Интегрировать конфигурацию в основные модули API (подключение к БД, настройка логгера, CORS, rate-limiting, порты запуска);
4. Добавить валидацию обязательных параметров, настроить fallback-значения, типы данных и обработку ошибок при некорректных настройках;
5. Протестировать запуск приложения с разными профилями окружения, проверить корректность применения настроек, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №50
Упаковка REST API приложения в контейнер и доставка на другое устройство

Цель: научиться контейнеризировать REST API-приложение с использованием Docker, оптимизировать сборочные образы, экспортировать их и обеспечивать корректный запуск сервиса на разнородных устройствах

Выполнив работу, вы будете уметь:
У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие.

Материальное обеспечение:
WebStorm, Postman, Docker

Задание:
1. Подготовить проект REST API и проанализировать его зависимости для корректной контейнеризации;
2. Написать Dockerfile с настройкой рабочего окружения, установкой зависимостей и оптимизацией размера итогового образа;
3. Собрать локальный образ, протестировать его запуск и убедиться в доступности всех endpoints API;
4. Экспортировать образ в переносимый файл, доставить на другое устройство, загрузить и запустить контейнер с проверкой работоспособности.

Порядок выполнения работы:
1. Изучить структуру проекта, определить язык, фреймворк, порт прослушивания и необходимые системные зависимости для написания Dockerfile;
2. Создать Dockerfile, настроить базовый образ, копирование исходного кода, установку пакетов, команду запуска приложения и применить многоэтапную сборку (при необходимости);
3. Собрать образ командой docker build, запустить контейнер локально, проверить доступность API через curl/Postman и убедиться в корректной работе;
4. Экспортировать готовый образ в .tar-архив с помощью docker save, перенести файл на целевое устройство, загрузить через docker load и запустить контейнер с нужными параметрами (порты, переменные окружения);
5. Проверить работоспособность сервиса на новом устройстве, зафиксировать шаги миграции и оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:

«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Тема 2.3. Безопасность при интеграции
Лабораторное занятие №51
Добавление SSL сертификата в приложение

Цель: научиться обеспечивать безопасную передачу данных в веб- и API-приложениях путём настройки HTTPS-соединений, работы с SSL/TLS сертификатами и конфигурирования защищённых сетевых интерфейсов

Выполнив работу, вы будете уметь:
У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Материальное обеспечение:
WebStorm, Postman, OpenSSL, Apache

Задание:
1. Изучить типы SSL-сертификатов и выбрать метод их получения (самоподписанный для разработки или через локальный УЦ);
2. Сгенерировать приватный ключ и сертификат с использованием утилит командной строки;
3. Настроить серверное приложение или обратный прокси для работы по протоколу HTTPS с использованием созданных сертификатов;
4. Реализовать принудительный редирект с HTTP на HTTPS, протестировать безопасное соединение и валидировать параметры TLS-сессии.

Порядок выполнения работы:
1. Ознакомиться с документацией фреймворка/веб-сервера по настройке TLS и подготовить рабочее окружение;
2. Сгенерировать пару «приватный ключ – сертификат» через OpenSSL или mkcert, проверить их корректность, формат и срок действия;
3. Интегрировать сертификаты в конфигурацию приложения (или настроить Nginx/Apache как reverse proxy), указать пути к файлам, параметры TLS и порты прослушивания;
4. Настроить редирект HTTP→HTTPS, отключить устаревшие версии протоколов и небезопасные шифры, проверить запуск приложения на портах 80/443;
5. Протестировать доступ по HTTPS через браузер и curl, проанализировать цепочку доверия и статус соединения, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:

«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №52
Настройка конфигурации безопасности приложения

Цель: научиться применять комплексные меры защиты на уровне конфигурации веб-приложения, обеспечивая устойчивость к распространенным векторам атак, безопасную обработку запросов и корректное управление политиками доступа

Выполнив работу, вы будете уметь:
У 2.3.3 отслеживать и устранять конфликты и ошибки интеграции.

Материальное обеспечение:
WebStorm, Postman, Redis

Задание:
1. Проанализировать текущую конфигурацию приложения и выявить уязвимости на уровне HTTP-заголовков, обработки входных данных и управления зависимостями;
2. Настроить базовые механизмы защиты: политики CORS, безопасные HTTP-заголовки (HSTS, X-Content-Type-Options, X-Frame-Options, CSP) и отключение информационных заголовков сервера;
3. Реализовать ограничение частоты запросов (Rate Limiting) и строгую валидацию/санитизацию входных данных на уровне маршрутов;
4. Настроить безопасное управление секретами, логирование подозрительных запросов и протестировать устойчивость конфигурации к базовым атакам.

Порядок выполнения работы:
1. Изучить документацию фреймворка по настройке безопасности, установить необходимые пакеты/middleware и подготовить тестовое окружение;
2. Настроить политики CORS, добавить библиотеку для управления HTTP-заголовками безопасности, убедиться в скрытии версий сервера и фреймворка;
3. Реализовать Rate Limiting для критических endpoints, настроить схемы валидации входных данных (типы, длина, форматы), проверить возврат корректных 4xx-ошибок при невалидных запросах;
4. Вынести конфиденциальные параметры (ключи, строки подключения) в переменные окружения, настроить логирование событий безопасности (превышение лимитов, заблокированные запросы), протестировать конфигурацию через Postman/curl;
5. Провести базовое сканирование заголовков и лимитов, зафиксировать результаты до/после настройки, оформить отчёт о проделанной работе.

Форма представления результата:

Отчет о проделанной работе

Критерии оценки:

«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Тема 2.4. Оптимизация и масштабируемость интегрированных решений

Лабораторное занятие №53

Реализация кэширования данных в REST API

Цель: научиться повышать производительность и снижать нагрузку на серверные компоненты REST API-приложений за счёт внедрения механизмов кэширования, управления жизненным циклом кэша и настройки стратегий инвалидации данных

Выполнив работу, вы будете уметь:

У 2.3.5 работать с различными форматами данных и протоколами передачи данных.

Материальное обеспечение:

WebStorm, Postman, Redis

Задание:

1. Проанализировать существующие endpoints API и определить данные/запросы, наиболее подходящие для кэширования;
2. Развернуть кэш-хранилище и подключить его к проекту приложения, настроить клиентское подключение и параметры соединения;
3. Реализовать логику работы с кэшем: проверка наличия данных, сохранение ответов БД, автоматическая и ручная инвалидация кэша при изменениях;
4. Настроить HTTP-заголовки кэширования (Cache-Control, ETag) и протестировать поведение API до и после внедрения кэша.

Порядок выполнения работы:

1. Изучить документацию по выбранному механизму кэширования, определить стратегию (TTL, cache-aside, lazy loading) и подготовить рабочее окружение;
2. Развернуть кэш-сервер (локально или в Docker-контейнере), проверить доступность порта, установить клиентскую библиотеку и настроить конфигурацию подключения;
3. Модифицировать сервисный слой API: добавить проверку кэша перед обращением к базе данных, реализовать сохранение результатов запросов и механизм инвалидации при POST/PUT/DELETE-операциях;
4. Добавить HTTP-заголовки кэширования для клиентской стороны, настроить логирование обращений к кэшу (hit/miss), обработать сценарии недоступности кэш-сервиса (fallback к БД);
5. Провести функциональное и/или нагрузочное тестирование, сравнить время отклика и количество запросов к БД, зафиксировать результаты, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.

Лабораторное занятие №54
Реализация кэширования данных в REST API

Цель: научиться анализировать производительность REST API-приложений с использованием инструментов профилирования, выявлять узкие места (CPU, память, запросы к БД, I/O-операции) и применять программные и архитектурные оптимизации для повышения скорости отклика и пропускной способности

Выполнив работу, вы будете уметь:
У 2.3.3 отслеживать и устранять конфликты и ошибки интеграции.

Материальное обеспечение:
WebStorm, Postman, Redis

Задание:
1. Подготовить тестовое окружение, настроить инструменты профилирования и нагрузочного тестирования для целевого REST API;
2. Провести базовый нагрузочный тест, собрать метрики производительности (время отклика, RPS, потребление CPU/памяти) и запустить профилировщик параллельно с тестированием;
3. Проанализировать отчёты профилирования, выявить узкие места (медленные запросы к БД, блокирующие операции, избыточные вычисления, проблемы с памятью) и реализовать оптимизации;
4. Провести повторное нагрузочное тестирование, сравнить метрики «до» и «после», убедиться в повышении производительности и корректности работы приложения.

Порядок выполнения работы:
1. Изучить документацию по выбранным инструментам, подготовить стабильную тестовую среду, наполнить БД тестовыми данными и зафиксировать начальные параметры сервера;
2. Настроить сценарий нагрузочного теста (виртуальные пользователи, длительность, целевые endpoints), запустить профилировщик параллельно с тестом и собрать исходные отчёты по времени выполнения функций и потреблению ресурсов;
3. Проанализировать результаты: выявить топ «тяжёлых» вызовов, длительные SQL-запросы, проблемы с блокировками или утечками памяти, составить план конкретных оптимизаций;

4. Внести изменения в код (оптимизация алгоритмов, добавление индексов/кэша, переход на асинхронные вызовы, настройка пулов соединений), повторно запустить профилирование и нагрузочный тест;
5. Сравнить результаты, задокументировать внесённые изменения и их влияние на метрики, оформить отчёт о проделанной работе.

Форма представления результата:
Отчет о проделанной работе

Критерии оценки:
«5» (отлично): выполнены все задания лабораторной работы, студент четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания лабораторной работы; студент ответил на все контрольные вопросы с замечаниями.

«3» (удовлетворительно): выполнены все задания лабораторной работы с замечаниями; студент ответил на все контрольные вопросы с замечаниями.

«2» (неудовлетворительно): студент не выполнил или выполнил неправильно задания лабораторной работы; студент ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы.