Государственное бюджетное профессиональное образовательное учреждение
Республики Хакасия
«Техникум коммунального хозяйства и сервиса»

Комплект контрольно-оценочных средств

по ПМ.03 Разработка веб приложения на стороне сервера (по выбору)

для подготовки специалистов среднего звена/квалифицированных рабочих, служащих по специальности/профессии

09.02.09 Веб-разработка

Абакан, 2025

Комплект контрольно-оценочных средств разработан на основе Федерального государственного образовательного стандарта среднего профессионального образования, по профессии/специальности *09.02.09 Веб-разработка* и программы **ПМ.03 Разработка веб приложения на стороне сервера (по выбору)**

СОДЕРЖАНИЕ

1. ПАСПОРТ КОМПЛЕКТА КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ

1.1. Область применения контрольно-оценочных средств (далее – КОС)

Контрольно-оценочные средства предназначены для оценки освоения основного вида деятельности и уровня сформированности соответствующих ему общих и профессиональных компетенций в процессе текущего и рубежного контроля, промежуточной аттестации.

1.1.1. Перечень общих компетенций

Код	Наименование 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;
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;
ОК 09	Пользоваться профессиональной документацией на государственном и иностранном языках

1.1.2. Перечень профессиональных компетенций

Код	Наименование видов деятельности и профессиональных компетенций
ВД.3	Разработка веб приложения на стороне сервера
ПК 3.1.	Администрировать среды и платформы разработки информационных ресурсов
ПК 3.2.	Создавать программный код на стороне сервера в соответствии с техническим заданием (спецификацией) с использованием языков программирования, библиотек и фреймворков
ПК 3.3.	Осуществлять отладку программного кода на стороне сервера на уровне программных модулей, межмодульных взаимодействий и взаимодействий с окружением

1.1.3. В результате освоения профессионального модуля обучающийся должен:

Владеть навыками	В разработке веб приложения на стороне сервера
Уметь	• применять выбранные языки программирования для написания программного кода; • использовать выбранную среду программирования и средства системы управления базами данных; • применять инструменты для тестирования программных модулей; • использовать возможности имеющейся программной архитектуры информационного ресурса. • использовать серверную инфраструктуру и средства виртуализации • владеть методами поиска информации по специальности, уметь выбирать необходимые технические средства и системы при решении конкретных задач и проблем
Знать	• отраслевую нормативную техническую документацию; • особенности выбранной среды программирования; • сетевые протоколы и основ web-технологий; • основы информационной безопасности web-ресурсов. • методы повышения читаемости программного кода; • синтаксис выбранного языка программирования, особенностей программирования на этом языке, стандартных библиотек языка программирования;

Распределение оценивания результатов обучения по видам контроля

Код и наименование элемента умений или знаний	Вид аттестации	
	Вид аттестации	Промежуточный контроль
У 1. Умение решать задачи математического анализа, линейной алгебры и аналитической геометрии	*расчетное задание*	*расчетное задание*
З 1. Знание основных методов математического анализа, аналитической геометрии, линейной алгебры,	*расчетное задание, устный ответ*	*устный ответ*

элементарной теории вероятностей		

2. ФОРМЫ КОНТРОЛЯ И ОЦЕНКИ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ ПО ТЕМАМ (РАЗДЕЛАМ)

Предметом оценки служат умения и знания, предусмотренные ФГОС по дисциплине (*название дисциплины*), направленные на формирование общих и профессиональных компетенций.

(технология оценки З и У по дисциплине прописывается в соответствии со спецификой дисциплины. Если экзамен проводится поэтапно или предусмотрена рейтинговая система оценки, то это подробно описывается)

Контроль и оценка освоения учебной дисциплины по темам (разделам)

Элемент учебной дичциплины	Формы и методы контроля					
	Текущий контроль		Рубежный контроль (контроль по разделу)		Промежуточная аттестация	
	Форма контроля	Проверяемые ОК; У, З	Форма контроля	Проверяемые ОК; У, З	Форма контроля	Проверяемые ОК; У, З
Раздел 1 (наименование)						
Тема 1.1 (название)	*Устный опрос №1 (или Практическая работа 1,…)*	*У1, З2, ОК3, ОК6.,ПК1.1, ПК1.2*	–	–	–	–
Тема 1.2	*Устный опрос №2 (или Тест №1, Лабораторная работа 1…)*	*У2, У3, З1, З3, ОК2, ОК3., ПК1.1, ПК1.2*	–	–	–	–

Тема 1.n	*Устный опрос №3 (или Самостоятельная работа 1, Практическая работа 2….)*	*У.., З.., ОК…*	–	–	–	–
Контроль по разделу 1	–	–	*Контрольная работа №1*	*У1.., У2…. З.1.. З2… ОК…, ОК…, ОК… ПК1.1, ПК1.2*	–	–
Раздел 2 (наименование)						
Тема 2.1	*Устный опрос (или Самостоятельная работа 2 Практическая работа 3…)*	*У1, У2, З 1, З2, З3, ОК 3, ОК 7 ПК1.1-ПК1.3*	–	–	–	–
Тема 2.2	*Самостоятельная работа 2 (или…)*	*У.., З.., З.., ОК.., ОК…, ОК.. ПК1.1-ПК1.3*	–	–	–	–
Тема 2.n	*Форма контроля*	*У.., З.., З.., ОК.., ОК…, ПК...*	–	–	–	–
Контроль по разделу 2	–	–	*Контрольная работа № 2*	*ОК…, ОК…, ОК… У.., У2…. З... З…ПК*	–	–
Раздел N(наименование)						
Тема N.1	*Форма контроля*	*У.., З.., ОК.,.ПК*	–	–	–	–
Тема N.2	*Форма контроля*	*У.., З.., ОК..,ПК*	–	–	–	–
Контроль по разделу N	–	–	*Форма контроля*	*З,У, ОК, ПК*	–	–
Промежуточная я аттестация	–	–	–	–	*Форма контроля*	*У, З, ОК, ПК*

3 ПРАКТИЧЕСКИЕ ЗАДАНИЯ

3.1 Практические работы по дисциплине МДК 03.01 Администрирование сред и платформ разработки информационных ресурсов

Практическое занятие №1: Создание полноценной инфраструктуры для разработки и тестирования веб-приложений с использованием готовых комплексных решений, таких как XAMPP, WAMP или MAMP

Представьте себя архитектором, задача которого – создать идеальное место для строительства веб-приложений. Это место должно быть удобным, надежным и обеспечивать всем необходимым для быстрого прототипирования и тестирования. Ваша миссия – создать такую инфраструктуру, используя готовые решения, что позволит быстро начать работу, не тратя время на рутинные настройки.

Начать стоит с тщательного исследования рынка готовых комплексных решений, а именно XAMPP, WAMP, MAMP и, возможно, Open Server. Создайте сравнительную таблицу, где детально опишите особенности каждого пакета. Обратите внимание на то, какие операционные системы они поддерживают, насколько гибко можно настраивать каждый компонент (Apache, PHP, MySQL или MariaDB), есть ли встроенная поддержка таких полезных инструментов, как phpMyAdmin или Xdebug, насколько легко обновлять компоненты и как реализована поддержка виртуальных хостов и SSL. Важно понимать, какое решение лучше соответствует вашим личным предпочтениям и задачам. В отчете подробно опишите, почему вы выбрали именно этот пакет.

Если вы чувствуете в себе силы, попробуйте автоматизировать процесс установки выбранного пакета. Напишите скрипт (например, на PowerShell для WAMP или shell-скрипт для XAMPP под Linux), который установит и настроит всё необходимое с минимальным участием пользователя. Это не только сэкономит время в будущем, но и позволит глубже понять процесс установки и настройки.

Теперь пришло время заняться безопасностью. Представьте, что ваш веб-сервер – это крепость, которую нужно защитить от злоумышленников. Отключите ненужные модули Apache, ограничьте доступ к директориям с помощью .htaccess, настройте mod_security, чтобы отражать распространенные веб-атаки, и не забудьте включить логирование всех запросов и ошибок. Позаботьтесь о безопасности MySQL или MariaDB: удалите анонимных пользователей и тестовые базы данных, смените пароль пользователя root, ограничьте доступ к базе данных только с локального хоста и включите ведение журнала запросов. Кроме того, настройте фаервол для ограничения доступа к портам сервера и обязательно включите HTTPS, получив бесплатный SSL-

сертификат от Let's Encrypt и перенаправляя все HTTP-запросы на HTTPS.

В качестве дополнительного задания, изучите возможность использования нескольких версий PHP на одном сервере. Это может быть полезно, если вам нужно поддерживать несколько проектов, требующих разных версий PHP. Настройте несколько виртуальных хостов для работы с разными версиями PHP, используя, например, mod_fcgid или php-fpm.

Для комфортной разработки настройте интеграцию выбранного пакета с инструментами разработки. Установите Xdebug для отладки PHP-кода в IDE, установите Composer для управления зависимостями PHP-проектов и настройте Git для контроля версий кода.

Чтобы не потерять ценные данные, автоматизируйте процесс резервного копирования. Напишите скрипт, который автоматически создает резервные копии файлов веб-сайтов и баз данных, и настройте его автоматический запуск по расписанию с помощью cron.

И наконец, настройте мониторинг сервера, чтобы всегда быть в курсе его состояния. Установите инструменты мониторинга, такие как phpSysInfo или Netdata, настройте мониторинг основных параметров сервера (загрузка процессора, использование памяти, трафик) и настройте уведомления, чтобы получать оповещения при достижении пороговых значений.

В завершение работы создайте подробную документацию, описывающую все шаги установки и настройки, а также обоснование каждого принятого решения. Не забудьте включить схемы сети и описание архитектуры, скриншоты всех этапов настройки, примеры конфигурационных файлов и инструкции по использованию созданной инфраструктуры для разработки и тестирования веб-приложений.

Практическое занятие №2: Создание масштабируемого и отказоустойчивого веб-сервера на виртуальных машинах с балансировкой нагрузки

Представьте себя главным инженером, отвечающим за надежность и производительность веб-приложения, которое должно выдерживать большие нагрузки и оставаться доступным в любое время. Ваша задача – создать масштабируемый и отказоустойчивый веб-сервер, используя несколько виртуальных машин и балансировщик нагрузки.

Первым шагом будет выбор подходящих технологий. Обоснуйте свой выбор программного обеспечения для виртуализации (VirtualBox, VMware или Hyper-V), операционной системы (Ubuntu Server, CentOS или Debian), веб-сервера (Apache или Nginx) и базы данных (MySQL/MariaDB или PostgreSQL). Важно тщательно проанализировать все доступные варианты и выбрать наиболее подходящие для ваших нужд.

Затем создайте необходимое количество виртуальных машин. Вам потребуется минимум две виртуальные машины для веб-серверов и одна виртуальная машина для балансировщика нагрузки. Установите и настройте выбранную операционную систему на каждой виртуальной машине и настройте сетевые параметры, такие как IP-адреса, маски подсети, шлюзы и DNS-серверы.

Установите и настройте выбранный веб-сервер (Apache или Nginx) на каждой виртуальной машине. Настройте синхронизацию файлов веб-сайтов между серверами, чтобы все веб-серверы отображали одинаковый контент. Для этого можно использовать, например, rsync или GlusterFS. Настройте базу данных на одном из серверов или используйте отдельный сервер базы данных. Также, настройте кэширование данных на веб-серверах, чтобы уменьшить нагрузку на базу данных. Memcached или Redis могут помочь в этом.

Теперь пришло время настроить балансировщик нагрузки. Установите и настройте выбранный балансировщик нагрузки (HAProxy, Nginx или Apache) и настройте балансировку нагрузки между веб-серверами, используя один из доступных алгоритмов (например, round-robin или least connections). Настройте мониторинг работоспособности веб-серверов, чтобы балансировщик нагрузки мог автоматически перенаправлять трафик на работающие серверы в случае отказа одного из них. Не забудьте настроить SSL-сертификаты на балансировщике нагрузки, чтобы обеспечить безопасное соединение с веб-сайтом.

После завершения всех настроек тщательно протестируйте систему. Проверьте доступность веб-сайта через балансировщик нагрузки, сымитируйте отказ одного из веб-серверов, чтобы убедиться, что трафик автоматически перенаправляется на другой сервер, проведите нагрузочное тестирование для оценки производительности системы и проанализируйте логи веб-серверов и балансировщика нагрузки для выявления проблем.

В качестве дополнительного задания изучите инструменты автоматизации развертывания, такие как Ansible, Chef или Puppet, и напишите сценарии для автоматической настройки веб-серверов и балансировщика нагрузки. Это позволит автоматизировать процесс развертывания и упростить масштабирование системы в будущем.

В завершение, создайте подробную документацию, в которой опишите все шаги установки и настройки, а также обоснование каждого принятого решения. Не забудьте включить схемы сети и описание архитектуры, скриншоты всех этапов настройки, примеры конфигурационных файлов и инструкции по автоматическому развертыванию системы.

Практическое занятие №3: Мастер автоматизации: Управление сервером с помощью Ansible

Вообразите, что вы – дирижер оркестра серверов, и ваша задача – добиться слаженной и эффективной работы всей системы. Вместо того, чтобы настраивать каждый сервер вручную, вы используете мощный инструмент автоматизации –

Ansible – чтобы описать желаемое состояние системы и позволить Ansible самостоятельно привести все серверы к этому состоянию.

Ваш путь начнется с настройки Ansible на управляющем сервере (control node). Установите Ansible, настройте SSH-ключи для безопасного доступа к управляемым серверам (managed nodes) и создайте inventory-файл, содержащий список управляемых серверов.

Теперь пришло время создать Playbooks – сценарии, описывающие последовательность задач, которые необходимо выполнить на управляемых серверах. Создайте Playbook для обновления списка пакетов и установки обновлений безопасности, для установки и настройки веб-сервера (Apache или Nginx), для установки и настройки базы данных (MySQL/MariaDB или PostgreSQL), для создания пользователей и групп, для управления файлами и каталогами, для настройки фаервола (iptables или ufw) и для развертывания веб-приложений (с использованием Git).

Чтобы сделать код более читаемым и повторно используемым, преобразуйте Playbooks в Roles. Создайте Roles для установки и настройки веб-сервера, для установки и настройки базы данных, для настройки безопасности и для развертывания веб-приложений.

Для гибкой настройки используйте переменные для задания значений параметров, таких как имя веб-сайта, версия PHP или пароль пользователя. Используйте шаблоны Jinja2 для создания конфигурационных файлов на основе этих переменных.

Не забудьте о мониторинге. Настройте автоматический мониторинг сервера с помощью Ansible, установив инструменты мониторинга (например, Netdata или Prometheus) и настроив отправку уведомлений по email или в Slack при возникновении проблем.Чтобы автоматизировать выполнение нескольких задач, создайте Workflow, который автоматически выполняет последовательность действий, таких как развертывание веб-приложения, обновление базы данных и перезапуск веб-сервера.

Перед развертыванием на production-серверах тщательно протестируйте все Playbooks и Roles на тестовом сервере. Убедитесь, что все задачи выполняются корректно и idempotent (то есть, повторный запуск Playbook не приводит к изменению состояния системы).

В завершение, создайте подробную документацию, описывающую все Playbooks и Roles, а также инструкции по их использованию. Не забудьте включить схемы и диаграммы, отображающие логику работы автоматизированных процессов.

3.2 Практические работы по дисциплине МДК 03.02 Разработка кода информационных ресурсов

Практическое занятие №1: «Установка и настройка языка программирования и инструментов разработки» (Python)

Цель: Глубокое погружение в процесс установки и настройки Python и необходимых инструментов, освоение лучших практик организации проектов.

Установка Python:

Установите последнюю стабильную версию Python с официального сайта (python.org).

Установите Python для нескольких пользователей (если это возможно в вашей ОС).

Убедитесь, что Python добавлен в системную переменную PATH.

Проверьте установку, открыв командную строку/терминал и введя python --version.

Выбор и настройка IDE:

Сравните несколько IDE (PyCharm, VS Code, Sublime Text, Atom) и выберите наиболее подходящую для вас, основываясь на их функциональности, удобстве использования и наличии необходимых расширений.

Установите выбранную IDE.

Настройте IDE: установите тему оформления, настройте автодополнение кода, проверку синтаксиса и другие полезные параметры.

Создание виртуального окружения:

Создайте виртуальное окружение для проекта с помощью venv (или virtualenv, если используете старую версию Python).

Активируйте виртуальное окружение.

Укажите виртуальное окружение в настройках IDE для проекта.

Установка пакетов и управление зависимостями:

Изучите возможности pip для управления пакетами.

Создайте файл requirements.txt, в котором перечислите все зависимости проекта и их версии (используйте pip freeze > requirements.txt).

Установите необходимые пакеты, используя файл requirements.txt (используйте pip install -r requirements.txt).

Ознакомьтесь с инструментами для управления зависимостями, такими как poetry или pipenv, и попробуйте использовать их в проекте.

Простой скрипт с использованием библиотек:

Напишите скрипт Python, который выполняет следующие действия:

Получает список статей с веб-сайта (например, с помощью requests и BeautifulSoup).

Анализирует текст статей и вычисляет частоту встречаемости определенных слов (например, с помощью nltk).

Сохраняет результаты анализа в файл (например, CSV или JSON).

Документация:

Создайте файл README.md в корне проекта и подробно опишите:

Цель проекта.

Инструкции по установке и настройке окружения.

Описание используемых библиотек и их назначения.

Инструкции по запуску скрипта.

Включите скриншоты настроек IDE и терминала.

Практическое занятие №2: «Разработка консольного приложения на основе процедурного подхода к программированию» (Python)

Цель: Развить навыки проектирования и разработки приложений, используя процедурный стиль программирования, а также научиться тестировать код.

Задание:

ToDo List с расширенными функциями:

Расширьте функциональность приложения ToDo List, включив в него следующие возможности:

Приоритет задачи (высокий, средний, низкий).

Срок выполнения задачи (дата и время).

Категории задач (например, "Работа", "Личное", "Учеба").

Подробное описание задачи.

Организуйте хранение задач в структурированном формате (например, с использованием словарей или списков словарей).

Интерактивный консольный интерфейс:

Создайте удобный и интуитивно понятный консольный интерфейс для взаимодействия с пользователем.

Используйте меню для навигации по функциям приложения.

Обеспечьте возможность фильтрации задач по приоритету, сроку выполнения и категории.

Добавьте возможность сортировки задач по разным критериям.

Сохранение и загрузка данных:

Используйте формат JSON для сохранения и загрузки списка задач из файла.

Обеспечьте автоматическое сохранение списка задач при каждом изменении.

Добавьте возможность резервного копирования списка задач.

Обработка ошибок:

Реализуйте обработку возможных ошибок, таких как:

Некорректный ввод данных пользователем.

Отсутствие файла с данными.

Ошибки при чтении или записи в файл.

Предоставляйте пользователю понятные сообщения об ошибках.

Тестирование:

Напишите модульные тесты для основных функций приложения, используя библиотеку unittest или pytest.

Обеспечьте покрытие тестами не менее 80% кода.

Документация:

Создайте подробную документацию для приложения, описывающую:

Архитектуру приложения.

Описание функций и их параметров.

Инструкции по использованию приложения.

Примеры использования.

Практическое занятие №3: «Проектирование и разработка системы управления библиотекой с использованием ООП» (C#)

Цель: Углубленное изучение и применение принципов ООП для создания сложной системы.

Задание:

Расширение классов:

Расширьте классы Book, Library и Reader новыми атрибутами и методами, отражающими более сложные аспекты управления библиотекой.

Book: добавьте поля для хранения информации о жанре, издательстве, годе издания, количестве страниц, рейтинге и т.д.

Library: добавьте методы для поиска книг по различным критериям (например, по жанру, автору, году издания), для подсчета общего количества книг, для определения самых популярных книг и т.д.

Reader: добавьте поля для хранения истории взятых книг, текущего списка взятых книг, информацию о штрафах и т.д.

Реализация дополнительных классов:

Создайте дополнительные классы для:

Genre: для представления жанра книги.

Publisher: для представления издательства.

Loan: для представления записи о выдаче книги читателю.

Fine: для представления записи о штрафе читателя.

Использование принципов ООП:

Примените принципы инкапсуляции, наследования и полиморфизма для создания гибкой и расширяемой архитектуры системы.

Используйте интерфейсы для определения общего поведения различных классов (например, интерфейс ILendable для книг, которые можно выдавать читателям).

Примените абстрактные классы для создания базовых классов с общим поведением.

Взаимодействие с пользователем:

Разработайте консольный интерфейс, который позволит пользователю:

Добавлять, удалять, редактировать и просматривать информацию о книгах, читателях, жанрах, издательствах и т.д.

Выдавать и возвращать книги.

Просматривать историю взятых книг читателем.

Оплачивать штрафы.

Формировать отчеты о работе библиотеки.

Сохранение и загрузка данных:

Используйте XML или JSON для сохранения и загрузки данных о библиотеке.

Обеспечьте возможность импорта и экспорта данных в различных форматах.

Обработка исключений:

Реализуйте обработку исключений для предотвращения сбоев в работе программы.

Предоставляйте пользователю информативные сообщения об ошибках.

Практическое занятие №4: «Применение шаблонов проектирования для создания расширяемой системы управления библиотекой» (C#)

Цель: Освоение и практическое применение продвинутых шаблонов проектирования для создания гибкой, расширяемой и поддерживаемой системы.

Задание:

Реализация шаблонов проектирования:

В дополнение к шаблонам, реализованным в предыдущем задании (Singleton, Factory Method, Strategy, Observer), реализуйте следующие шаблоны проектирования:

Repository (Репозиторий): Создайте репозиторий для доступа к данным о книгах, читателях, жанрах и т.д. Репозиторий должен предоставлять методы для получения, добавления, обновления и удаления данных.

Unit of Work (Единица работы): Обеспечьте согласованное выполнение нескольких операций с данными в рамках одной транзакции.

Dependency Injection (Внедрение зависимостей): Используйте IoC-контейнер (например, Autofac или Ninject) для управления зависимостями между классами.

Command (Команда): Реализуйте систему команд для выполнения различных действий в библиотеке (например, команда для добавления книги, команда для выдачи книги и т.д.).

Decorator (Декоратор): Добавьте возможность применять различные декораторы к книгам (например, декоратор для добавления автографа автора, декоратор для нанесения защитного покрытия и т.д.).

Гибкая архитектура:

Обеспечьте возможность добавления новых шаблонов проектирования и новых функциональных возможностей без изменения существующего кода.

Используйте принципы SOLID для создания слабосвязанных и легко тестируемых классов.

Документация:

Создайте подробную документацию, описывающую архитектуру системы, используемые шаблоны проектирования, принципы их работы и способы добавления новых шаблонов.

Включите диаграммы классов и последовательностей для визуализации структуры системы.

ТЕМА 2.2 Программирование на стороне сервера

Практическое занятие №1: "Программирование задач с использованием переменных"

Целью данного занятия является формирование у обучающихся устойчивых навыков разработки алгоритмов и их программной реализации на языке Python, в основе которых лежит использование переменных различных типов. Важность освоения данной компетенции обусловлена тем, что переменные являются фундаментальным элементом любого языка программирования, позволяющим хранить и манипулировать данными в процессе выполнения программы.

Практическая часть заключается в разработке консольного приложения, которое будет выполнять ряд вычислений и операций с данными, хранящимися в переменных. Приложение должно обеспечивать интерактивное взаимодействие с пользователем, позволяя вводить данные и просматривать результаты.

Реализация приложения начинается с определения необходимых переменных различных типов: целочисленных, вещественных, строковых и логических. Необходимо продемонстрировать умение выбора подходящего типа данных в зависимости от характера хранимой информации. Например, для хранения количества товаров подойдет целочисленный тип, для хранения цены товара – вещественный тип, для хранения наименования товара – строковый тип, а для хранения статуса товара (активный/неактивный) – логический тип.

Далее, необходимо разработать алгоритмы для выполнения различных операций с переменными. В частности, требуется реализовать следующие функции: вычисление суммы, разности, произведения и частного двух чисел, вычисление площади и периметра геометрических фигур (например, прямоугольника и круга), объединение строк, преобразование типов данных (например, из строки в число и наоборот) и выполнение логических операций (например, сравнение двух чисел).

Приложение должно обеспечивать корректную обработку вводимых пользователем данных. Необходимо реализовать проверку на корректность вводимых данных, например, проверку на то, что вводимое значение является числом, а также обработку исключений, которые могут возникнуть при выполнении операций (например, деление на ноль).

На завершающем этапе разрабатывается консольный интерфейс для взаимодействия с пользователем. Интерфейс должен быть интуитивно понятным и обеспечивать удобный ввод данных и просмотр результатов. Необходимо предусмотреть возможность выбора операций, которые будут выполняться с данными, а также возможность просмотра истории выполненных операций.

Результатом выполнения практического занятия является работающее консольное приложение, демонстрирующее умение использовать переменные различных типов, выполнять операции с данными и обеспечивать интерактивное взаимодействие с пользователем.

Задача: Разработайте консольное приложение "Калькулятор", которое выполняет основные арифметические операции.

Описание:

Приложение должно запрашивать у пользователя два числа и операцию, которую необходимо выполнить (+, -, *, /). В зависимости от введенной операции, приложение должно выполнить соответствующее действие и вывести результат на экран.

Требования:

1. Использовать переменные для хранения чисел и операции.
2. Реализовать проверку на корректность вводимых данных (например, что введено число, а не текст).
3. Обрабатывать исключение деления на ноль.
4. Выводить результат в формате: "Число 1 [операция] Число 2 = Результат".
5. Предусмотреть возможность повторного выполнения вычислений до тех пор, пока пользователь не решит завершить работу.
6. Обеспечить понятный и информативный интерфейс для пользователя.
7. Все вычисления свести к одной функции

Практическое занятие №2: "Программирование задач с использованием массивов"

Целью данного занятия является формирование практических навыков работы с массивами (списками) в языке Python. Массивы являются мощным инструментом для хранения и обработки коллекций данных одного типа, что позволяет эффективно решать широкий спектр задач.

В рамках данного занятия студентам предстоит разработать программный модуль, реализующий основные операции над массивами. Важность данного модуля заключается в его универсальности, так как он может быть использован в различных приложениях, требующих обработки больших объемов данных.

Первым этапом является проектирование структуры данных для хранения массива. В Python для этих целей используются списки, которые обладают гибкостью и позволяют хранить элементы различных типов. Необходимо продемонстрировать умение создавать списки, добавлять и удалять элементы из списка, а также получать доступ к элементам по индексу.

Далее, необходимо разработать алгоритмы для выполнения основных операций над массивами: сортировка (по возрастанию и убыванию), поиск элемента (линейный и бинарный поиск), вычисление суммы элементов, вычисление среднего арифметического значения, поиск минимального и максимального элемента.

Особое внимание следует уделить эффективности алгоритмов. Например, для сортировки массива рекомендуется использовать алгоритмы сортировки, такие как быстрая сортировка (quicksort) или сортировка слиянием (mergesort), которые имеют логарифмическую сложность. Для поиска элемента в отсортированном массиве рекомендуется использовать бинарный поиск, который также имеет логарифмическую сложность.

Практическая значимость занятия заключается в реализации алгоритмов обработки массивов в виде отдельных функций, которые могут быть переиспользованы в других проектах.

На завершающем этапе разрабатывается модуль для визуализации массивов с помощью средств matplotlib.

Задача: Разработайте консольное приложение "Анализатор текста".

Описание:

Приложение должно запрашивать у пользователя текст, а затем выполнять различные операции анализа текста:

1. Подсчет количества слов в тексте.
2. Подсчет количества символов в тексте (с учетом и без учета пробелов).
3. Поиск наиболее часто встречающегося слова в тексте.
4. Определение доли каждого слова в тексте.
5. Вывод слов в порядке убывания частоты встречаемости.
6. Предусмотреть функцию для вывода статистики

Требования:

1. Использовать массивы (списки) для хранения слов и их количества.
2. Реализовать алгоритмы для выполнения указанных операций анализа текста.
3. Обрабатывать текст, приводя его к нижнему регистру и удаляя знаки препинания.
4. Выводить результаты анализа в понятном и структурированном формате.
5. Все вычисления должны быть реализованы в отдельных функциях

Практическое занятие №3: "Программирование задач с использованием условных операторов"

Данное практическое занятие ориентировано на формирование у обучающихся компетенций, связанных с использованием условных операторов в языке Python. Условные операторы позволяют реализовывать ветвление алгоритмов, то есть изменять ход выполнения программы в зависимости от выполнения определенных условий.

Практическим заданием является разработка программы, которая будет анализировать введенные пользователем данные и принимать решения на основе этих данных. Важно, чтобы разработанная программа корректно обрабатывала различные сценарии и обеспечивала пользователю понятную обратную связь.

Первым шагом является проектирование логики программы. Необходимо определить, какие условия будут проверяться, и какие действия будут выполняться в зависимости от результата проверки. Например, можно разработать программу, которая будет определять, является ли введенное число положительным, отрицательным или нулем.

Далее, необходимо реализовать проверку различных условий с использованием условных операторов if, elif и else. Необходимо продемонстрировать умение использовать сложные логические выражения, включающие операторы and, or и not.

Также необходимо научиться использовать вложенные условные операторы для реализации более сложных алгоритмов. Например, можно разработать программу, которая будет определять, является ли год високосным, используя вложенные условные операторы для проверки делимости на 4, 100 и 400.

Программа должна обеспечивать понятную обратную связь для пользователя. В зависимости от результатов проверки условий, программа должна выводить сообщения, поясняющие принятое решение. Необходимо предусмотреть обработку ситуаций, когда введенные пользователем данные не соответствуют ожидаемому формату.

На завершающем этапе необходимо протестировать программу на различных наборах входных данных, чтобы убедиться в ее корректной работе во всех предусмотренных сценариях.

Результатом выполнения практического занятия является работающая программа, демонстрирующая умение использовать условные операторы для реализации ветвящихся алгоритмов и обеспечивающая интерактивное взаимодействие с пользователем.

Задача: Разработайте консольное приложение "Игра: Угадай число".

Описание:

Приложение должно генерировать случайное число в заданном диапазоне (например, от 1 до 100). Пользователь должен угадать это число, вводя свои варианты. После каждой попытки приложение должно сообщать, больше или меньше введенное число, чем загаданное. Игра продолжается до тех пор, пока пользователь не угадает число.

Требования:

1. Использовать условные операторы для проверки вариантов пользователя.
2. Генерировать случайное число с использованием модуля random.
3. Ограничить количество попыток пользователя.
4. Выводить сообщения о количестве оставшихся попыток.
5. Сообщать пользователю о том, что он выиграл или проиграл.
6. После победы пользователя выводить статистику
7. Все должны быть разделено на отдельные функции.

3.3 Практические работы по дисциплине МДК 03.03 Разработка информационных ресурсов с использованием программных платформ

Практическое занятие №1: «Работа с формами»

Цель: освоить создание и обработку форм во Flask, включая валидацию данных с использованием WTForms.

Задание:

Проектирование формы: *(остается без изменений)*

Создайте форму обратной связи, включающую следующие поля:

Имя (текстовое поле, обязательное).

Email (текстовое поле, обязательное, с валидацией формата).

Тема сообщения (выпадающий список с вариантами).

Сообщение (текстовое поле, многострочное, обязательное).

Приложение файла (поле для загрузки файла, необязательное, с ограничением по типу и размеру).

Создание формы во Flask с использованием WTForms:

Используйте библиотеку WTForms для определения формы.

Определите поля формы с необходимыми типами и валидаторами.

```python
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, SelectField, FileField
from wtforms.validators import DataRequired, Email, Optional

class ContactForm(FlaskForm):
    name = StringField('Имя', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
    subject = SelectField('Тема', choices=[('question', 'Вопрос'), ('feedback', 'Отзыв'), ('other', 'Другое')])
    message = TextAreaField('Сообщение', validators=[DataRequired()])
    attachment = FileField('Приложение', validators=[Optional()])
```

Валидация данных:

Реализуйте валидацию данных на стороне сервера в представлении Flask.

```python
from flask import Flask, render_template, request, redirect, url_for
# ... previous imports
app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key' # Must set for CSRF protection

@app.route('/contact', methods=['GET', 'POST'])
def contact():
    form = ContactForm()
    if form.validate_on_submit():
        # Process the validated form data
        name = form.name.data
        email = form.email.data
        subject = form.subject.data
        message = form.message.data
        # Process attachment if any
        if form.attachment.data:
            # Save attachment or something similar
            pass
        # Redirect to a success page or similar
        return redirect(url_for('success'))
    return render_template('contact.html', form=form)
```

Выведите сообщения об ошибках валидации пользователю в понятном формате, передавая их в шаблон.

Обработка данных формы:

После успешной валидации данных сохраните их (например, в файл или базу данных).

Отправьте уведомление администратору сайта о новом сообщении (например, по email, используя библиотеку smtplib).

Перенаправьте пользователя на страницу подтверждения отправки сообщения.

Интеграция с шаблоном Jinja2:

Отобразите форму в HTML-шаблоне Jinja2.

Используйте form.field_name для доступа к полям формы в шаблоне.

Используйте form.field_name.errors для отображения ошибок валидации.

Используйте CSS для стилизации формы.

```
<form method="POST" action="{{ url_for('contact') }}" enctype="multipart/form-data">
    {{ form.csrf_token }}
    {{ form.name.label }} {{ form.name }}
    {% if form.name.errors %}
        <ul>
            {% for error in form.name.errors %}
                <li>{{ error }}</li>
            {% endfor %}
        </ul>
    {% endif %}
    <!-- Render other fields similarly -->
    <button type="submit">Отправить</button>
</form>
```

Дополнительные требования:

Реализуйте защиту от CSRF-атак, используя Flask-WTF.

Используйте CAPTCHA (например, Flask-ReCaptcha) для защиты от спама.

Оптимизируйте загрузку больших файлов, используя потоковую передачу.

Практическое занятие №2: «Реализация регистрации и авторизации»

Цель: Освоить процесс регистрации и авторизации пользователей на сайте с использованием Flask и Flask-Login.

Задание:

Проектирование форм: *(остается без изменений)*

Создайте форму регистрации, включающую следующие поля:

Имя пользователя (текстовое поле, обязательное, уникальное).

Email (текстовое поле, обязательное, с валидацией формата).

Пароль (текстовое поле, скрытое, обязательное, с минимальной длиной и требованиями к сложности).

Подтверждение пароля (текстовое поле, скрытое, обязательное, с проверкой совпадения).

Создайте форму авторизации, включающую поля:

Имя пользователя (текстовое поле).

Email (текстовое поле).

Пароль (текстовое поле, скрытое).

Поля для email и имя пользователя должна быть взаимозаменяемыми (то есть можно входить по чему-то одному).

Настройка Flask-Login:

Установите Flask-Login: pip install Flask-Login.

Настройте Flask-Login в вашем приложении:

```python
from flask import Flask
from flask_login import LoginManager

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key' # Replace with a strong secret key

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login' # Define login view

@login_manager.user_loader
def load_user(user_id):
    # Load user from database
    return User.get(user_id) # Assuming you have a User model
```

Создание моделей:

Создайте модель пользователя в базе данных, используя Flask-SQLAlchemy (или другую ORM).

Реализуйте методы get_id(), is_active(), is_authenticated(), is_anonymous() для модели пользователя, как требует Flask-Login.

```python
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
# from werkzeug.security import generate_password_hash, check_password_hash

db = SQLAlchemy()

class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128))

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
```

Реализация регистрации:

Обработайте данные формы регистрации:

Проверьте уникальность имени пользователя и email.

Захешируйте пароль с использованием Werkzeug (вместо hashlib)

Сохраните данные пользователя в базе данных.

Отправьте письмо подтверждения регистрации на email пользователя (опционально).

Автоматически авторизуйте пользователя после успешной регистрации, используя login_user из Flask-Login.

Реализация авторизации:

Обработайте данные формы авторизации:

Найдите пользователя в базе данных по имени пользователя или email.

Проверьте соответствие введенного пароля хешу пароля в базе данных.

Авторизуйте пользователя, используя login_user из Flask-Login.

Перенаправьте пользователя на личный кабинет или другую защищенную страницу.

```
from flask import Flask, render_template, request, redirect, url_for
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

# User loader callback
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Login route
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        # Form submission logic
        username = request.form["username"]
        password = request.form["password"]

        user = User.query.filter_by(username=username).first()

        if user and user.check_password(password):
            login_user(user)
            return redirect(url_for("profile"))
        else:
            return "Invalid credentials"

    return render_template("login.html")

# Profile route, require login
@app.route("/profile")
@login_required
def profile():
    return f"Hello, {current_user.username}"
```

```
# Logout route
@app.route("/logout")
@login_required
def logout():
    logout_user()
    return redirect(url_for("login"))
```

Безопасность:

Используйте CSRF-защиту для форм, предоставляемую Flask-WTF.

Реализуйте защиту от перебора паролей (например, используя Flask-Limiter).

Обеспечьте безопасное хранение паролей (хеширование с солью, используя Werkzeug).

Дополнительные требования:

Реализуйте восстановление пароля (с отправкой ссылки на email).

Добавьте возможность авторизации через социальные сети (например, Google, Facebook) с использованием Flask-OAuthlib (или аналогичной библиотеки).

Используйте @login_required decorator для защиты определенных маршрутов.

Практическое занятие №3: «Создание базовых функций работы с данными: Создание, чтение, модификация, удаление»

Цель: Научиться выполнять основные операции CRUD (Create, Read, Update, Delete) с данными в базе данных с использованием Flask и Flask-SQLAlchemy.

Задание:

Определение моделей:

Разработайте модели для представления следующих сущностей, используя Flask-SQLAlchemy:

Категория (название, описание).

Товар (название, описание, цена, категория, изображение).

```python
from flask import Flask
    from flask_sqlalchemy import SQLAlchemy
    from sqlalchemy import Column, Integer, String, Float, ForeignKey
    from sqlalchemy.orm import relationship

    app = Flask(__name__)
    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///example.db'
    db = SQLAlchemy(app)

    class Category(db.Model):
        id = Column(Integer, primary_key=True)
        name = Column(String(50), nullable=False)
        description = Column(String(200))
        products = relationship("Product", backref="category") # Relationship to Product

    class Product(db.Model):
        id = Column(Integer, primary_key=True)
        name = Column(String(50), nullable=False)
        description = Column(String(200))
        price = Column(Float, nullable=False)
        image = Column(String(100))
        category_id = Column(Integer, ForeignKey('category.id'))
```

Определите связи между моделями (например, связь "один ко многим" между категорией и товарами).

Создание представлений:

Создайте представления (view functions) для выполнения следующих операций:

Создание новой категории.

Просмотр списка категорий.

Редактирование существующей категории.

Удаление категории.

Создание нового товара.

Просмотр списка товаров (с возможностью фильтрации по категории).

Просмотр информации о конкретном товаре.

Редактирование существующего товара.

Удаление товара.

```python
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import Column, Integer, String, Float, ForeignKey
from sqlalchemy.orm import relationship
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///example.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db = SQLAlchemy(app)

# Model definitions here
from forms import CategoryForm, ProductForm

@app.route('/categories')
def list_categories():
    categories = Category.query.all()
    return render_template('categories.html', categories=categories)
```

Использование Flask-SQLAlchemy:

Используйте возможности Flask-SQLAlchemy для взаимодействия с базой данных.

Выполняйте операции CRUD с использованием методов SQLAlchemy (например, db.session.add(), db.session.query(), db.session.delete(), db.session.commit()).

Валидация данных:

Реализуйте валидацию данных при создании и редактировании категорий и товаров.

Используйте WTForms для создания форм валидации.

Проверьте обязательность заполнения полей, корректность форматов данных и уникальность значений.

Выведите сообщения об ошибках валидации пользователю, используя form.errors.

Интеграция с шаблонами Jinja2:

Отобразите данные в HTML-шаблонах Jinja2, используя render_template.

Создайте удобный интерфейс для навигации между категориями и товарами.

Используйте CSS для стилизации страниц.

Дополнительные требования:

Реализуйте пагинацию для списков категорий и товаров, используя Flask-Paginator или аналогичную библиотеку.

Добавьте возможность загрузки изображений товаров, используя Flask-Uploads или аналогичную библиотеку.

Используйте транзакции (db.session.commit()) для обеспечения целостности данных.

Практическое занятие №4: «Валидация данных»

Цель: Углубленное изучение методов валидации данных в Flask с использованием WTForms, разработка кастомных валидаторов и обработка сложных сценариев валидации.

Задание:

Анализ существующих форм:

Возьмите формы, созданные в практических занятиях №1 и №2 (форма обратной связи, формы регистрации и авторизации).

Реализация дополнительных валидаторов:

Добавьте следующие валидаторы для полей форм:

Валидатор для проверки надежности пароля (проверка наличия символов верхнего и нижнего регистра, цифр и специальных символов, минимальная длина).

Валидатор для проверки уникальности имени пользователя и email (проверка наличия таких записей в базе данных).

Валидатор для проверки соответствия введенного значения заданному диапазону (например, для поля "возраст").

Валидатор для проверки размера загружаемого файла.

Кастомные валидаторы:

Напишите собственные валидаторы для:

Проверки наличия стоп-слов в тексте сообщения (стоп-слова должны храниться в отдельном файле).

Проверки принадлежности загруженного изображения к определенному типу (например, JPG, PNG, GIF).

Обработка сложных сценариев валидации:

Реализуйте зависимую валидацию (когда значение одного поля зависит от значения другого поля). Например:

Если пользователь выбирает определенную тему сообщения, он должен заполнить дополнительное поле с подробностями.

Если пользователь загружает файл, то он должен указать его название и описание.

Реализуйте валидацию на основе регулярных выражений (например, для проверки номера телефона или почтового индекса).

Вывод сообщений об ошибках:

Настройте вывод сообщений об ошибках валидации на странице в понятном и информативном формате.

Используйте CSS для стилизации сообщений об ошибках.

Тестирование:

Напишите тесты для всех реализованных валидаторов, чтобы убедиться в их корректной работе.

Используйте моки для изоляции тестируемого кода от зависимостей.

Документация:

Опишите все реализованные валидаторы и примеры их использования в документации к проекту.

Практическое занятие №5: «Разработка API»

Цель: Освоить создание RESTful API во Flask с использованием Flask-RESTful или Flask-API, а также научиться работать с форматами данных JSON и XML.

Задание:

Проектирование API:

Разработайте API для управления списком книг (как в практическом задании №3).

API должен поддерживать следующие операции:

Получение списка всех книг (GET /books).

Получение информации о конкретной книге (GET /books/<book_id>).

Создание новой книги (POST /books).

Редактирование существующей книги (PUT /books/<book_id>).

Удаление книги (DELETE /books/<book_id>).

API должен поддерживать форматы данных JSON и XML.

Реализация API во Flask:

Используйте Flask-RESTful или Flask-API для создания API.

Определите ресурсы для каждой операции (например, BookListResource для получения списка книг и BookResource для управления конкретной книгой).

Используйте HTTP-методы (GET, POST, PUT, DELETE) для определения типа операции.

```python
from flask import Flask, request
from flask_restful import Api, Resource

app = Flask(__name__)
api = Api(app)

books = []

class BookListResource(Resource):
    def get(self):
        return books

    def post(self):
        data = request.get_json()
        book = {'id': len(books) + 1, 'title': data['title']}
        books.append(book)
        return book, 201
```

Сериализация и десериализация данных:

Используйте библиотеки marshmallow или Flask-REST-Marshmallow для сериализации и десериализации данных в форматы JSON и XML.

Определите схемы для каждой модели, чтобы указать, какие поля должны быть включены в API.

Валидация данных:

Реализуйте валидацию данных на стороне API при создании и редактировании книг.

Используйте возможности библиотек marshmallow или Flask-REST-Marshmallow для определения правил валидации.

Возвращайте сообщения об ошибках валидации в формате JSON или XML.

Аутентификация и авторизация:

Реализуйте аутентификацию и авторизацию для защиты API.

Используйте Basic Authentication, API Key Authentication или OAuth 2.0.

Ограничьте доступ к определенным операциям только для авторизованных пользователей.

Тестирование:

Напишите юнит-тесты для API, используя библиотеку pytest или unittest.

Проверьте корректность работы всех операций (GET, POST, PUT, DELETE), а также валидацию данных и аутентификацию.

Документация:

Создайте документацию для API, используя инструменты Swagger (OpenAPI) или ReDoc.

Опишите все ресурсы, операции, параметры и форматы данных.

Включите примеры запросов и ответов.

4 ПРОМЕЖУТОЧНЫЙ КОНТРОЛЬ

МДК 03.01. Тесты для промежуточного контроля

Тема 2.1: Виды и назначения серверных языков программирования, среды разработки

1. Какой язык программирования наиболее часто используется для разработки бэкенда веб-приложений?
   - a) JavaScript
   - b) HTML
   - c) Python

- d) CSS

2. Какой тип языков программирования требует предварительной компиляции перед выполнением?
   - a) Интерпретируемые
   - b) Компилируемые
   - c) Скриптовые
   - d) Функциональные
3. Что такое IDE (Integrated Development Environment)?
   - a) Набор библиотек для работы с базой данных
   - b) Комплекс инструментов для разработки программного обеспечения
   - c) Язык разметки веб-страниц
   - d) Система контроля версий
4. Какой подход к программированию подразумевает разделение программы на отдельные процедуры или функции?
   - a) Объектно-ориентированный
   - b) Процедурный
   - c) Функциональный
   - d) Декларативный
5. Какой принцип ООП позволяет создавать новые классы на основе существующих, наследуя их свойства и методы?
   - a) Инкапсуляция
   - b) Полиморфизм
   - c) Наследование
   - d) Абстракция
6. Что такое рефакторинг кода?
   - a) Процесс добавления нового функционала в программу
   - b) Процесс оптимизации кода без изменения его функциональности
   - c) Процесс изменения интерфейса программы
   - d) Процесс исправления ошибок в коде
7. Какой шаблон проектирования обеспечивает создание экземпляра класса через фабричный метод?
   - a) Singleton
   - b) Factory Method
   - c) Observer
   - d) Strategy
8. Что из перечисленного является серверным языком программирования?
   - a) JavaScript
   - b) HTML
   - c) PHP

- d) CSS

9. Что означает термин "масштабируемость" применительно к веб-приложениям?
   - a) Способность приложения работать на разных операционных системах
   - b) Способность приложения обрабатывать увеличивающуюся нагрузку
   - c) Простота использования приложения для конечного пользователя
   - d) Высокая скорость работы приложения
10. Что такое система контроля версий, например, Git?
   - a) Инструмент для создания графического интерфейса
   - b) Система для отслеживания изменений в коде
   - c) Тип базы данных
   - d) Протокол передачи данных
11. Что такое Dependency Injection?
   - a) Шаблон проектирования, позволяющий уменьшить зависимость между классами
   - b) Метод для создания веб-страниц
   - c) Метод хранения информации
   - d) Протокол передачи данных
12. Какой шаблон позволяет обеспечить существование только одного экземпляра класса?
   - a) Factory
   - b) Prototype
   - c) Adapter
   - d) Singleton
13. Что такое SOLID?
   - a) Принципы объектно-ориентированного проектирования
   - b) Язык программирования
   - c) Тип базы данных
   - d) Фреймворк для тестирования
14. Какая цель инкапсуляции?
   - a) Объединение данных и методов, работающих с ними, в классе
   - b) Наследование
   - c) Отправка уведомлений
   - d) Объединение нескольких классов в один
15. В чем суть полиморфизма?
   - a) Возможность создавать объекты разных классов с одинаковыми именами
   - b) Возможность объекту принимать разные формы

- c) Возможность создавать классы на основе других
- d) Возможность переиспользовать код

Тема 3.1: Разработка информационных ресурсов с использованием фреймворков и библиотек

1. Что такое веб-фреймворк?
   - a) Набор инструментов для создания графического интерфейса
   - b) Готовая структура для разработки веб-приложений
   - c) Язык программирования для создания веб-сайтов
   - d) Система управления базами данных
2. Что такое ORM (Object-Relational Mapping)?
   - a) Технология для преобразования кода на JavaScript
   - b) Способ взаимодействия с базой данных через объекты
   - c) Метод шифрования данных
   - d) Библиотека для работы с графикой
3. Что такое маршрутизация в веб-фреймворке?
   - a) Процесс стилизации веб-страниц
   - b) Метод преобразования данных в формат JSON
   - c) Определение соответствия между URL и обработчиками запросов
   - d) Система управления пользователями
4. Для чего используются шаблонизаторы в веб-фреймворках?
   - a) Для создания динамических веб-страниц
   - b) Для валидации данных
   - c) Для работы с базой данных
   - d) Для управления сессиями
5. Что такое CSRF (Cross-Site Request Forgery) и как от этого защититься?
   - a) Вид SQL-инъекции
   - b) Тип атаки, при которой злоумышленник выполняет действия от имени пользователя
   - c) Метод взлома паролей
   - d) Спам
6. Что такое MVC (Model-View-Controller) в контексте веб-фреймворков?
   - a) Язык программирования
   - b) Архитектурный паттерн, разделяющий приложение на три компонента
   - c) Тип базы данных
   - d) Стандарт протокола http
7. Что такое REST API?
   - a) Протокол передачи данных
   - b) Архитектурный стиль для создания веб-сервисов

- c) Способ валидации данных
- d) Метод для хранения данных

8. Какие типы валидации данных вы знаете?
   - a) На стороне клиента
   - b) На стороне сервера
   - c) Оба варианта
   - d) Отсутствует валидация
9. Какие методы http вы знаете?
   - a) GET
   - b) POST
   - c) DELETE
   - d) Все перечисленные
10. Для чего нужен Middleware?
    - a) Для расширения функциональности
    - b) Для фильтрации
    - c) Для выполнения задач до или после запроса
    - d) Для всего перечисленного
11. Что такое JWT?
    - a) Способ создания html
    - b) Способ аутентификации пользователей
    - c) Способ запуска программы
    - d) Тип базы данных
12. В чем разница между flask и django?
    - a) Разные языки
    - b) Разные способы создания веб-сайта
    - c) Все верно
    - d) Неверно
13. Что такое blueprint во flask?
    - a) Способ работы с css
    - b) Способ для организации приложение
    - c) Способ получения данных
    - d) Нет такого понятия
14. Что такое cookie?
    - a) Небольшой текстовый файл, который веб-сервер сохраняет на компьютере пользователя
    - b) Элемент для создания веб-страницы
    - c) Текстовый файл для базы данных
    - d) Способ шифрования кода
15. Что такое сессии?
    - a) Механизм для хранения данных

- b) Механизм для входа в систему
- c) Серия http запросов между клиентом и сервером
- d) Серия запросов в базу данных

Тема 1.1: Администрирование сред и платформ разработки информационных ресурсов

1. Что такое виртуализация?
   - a) Процесс создания веб-сайтов
   - b) Технология для запуска одной или нескольких операционных систем на одном физическом сервере
   - c) Система управления базами данных
   - d) Язык программирования для веб-разработки
2. Что такое гипервизор?
   - a) Программа для управления файлами
   - b) Программное обеспечение для создания и управления виртуальными машинами
   - c) Метод шифрования данных
   - d) Инструмент для мониторинга сети
3. Какие протоколы используются для передачи данных по сети?
   - a) HTML, CSS, JavaScript
   - b) TCP, UDP, IP
   - c) HTTP, FTP, SSH
   - d) SQL, NoSQL
4. Что такое веб-сервер?
   - a) Компьютер, на котором хранятся файлы веб-сайта
   - b) Программа, которая обрабатывает HTTP-запросы и отправляет ответы
   - c) Язык программирования для веб-разработки
   - d) Устройство для хранения данных
5. Что такое бэкап (резервное копирование) данных?
   - a) Функция для перемещения файлов
   - b) Процесс создания копии данных для восстановления в случае потери
   - c) Метод для организации данных
   - d) Защита файлов паролем
6. Что такое контейнеризация?
   - a) Вид шифрования данных
   - b) Способ упаковки приложения и его зависимостей в один образ для запуска
   - c) Способ создания веб-страниц
   - d) Способ хранения данных
7. Что такое балансировка нагрузки?

- a) Технология для оптимизации трафика в сети
- b) Функция для распределения нагрузки между несколькими серверами
- c) Способ экономии ресурсов
- d) Контроль трафика на устройстве

8. Что такое docker?
   - a) Виртуализация
   - b) Контейнеризация
   - c) Сквозное шифрование
   - d) Система управления базами данных
9. Какие способы изоляции предоставляет контейнеризация?
   - a) Процессы и файлы
   - b) Сеть
   - c) Хост
   - d) Все перечисленные
10. Что такое Dockerfile?
    - a) файл, содержащий инструкции для сборки Docker-образа
    - b) способ шифрования данных
    - c) виртуальная машина
    - d) сеть
11. Для чего нужен Docker Compose?
    - a) Для оркестрации
    - b) Для хранения данных
    - c) Для упрощения определения и запуска многоконтейнерных приложений
    - d) Для создания веб-страницы
12. Что позволяет делать система оркестрации Kubernetes?
    - a) Автоматизировать развёртывание, масштабирование и управление контейнерными приложениями
    - b) Получать данные
    - c) Увеличить мощность
    - d) Изменять файлы
13. Для чего нужен CI/CD (Continuous Integration/Continuous Delivery)?
    - a) Интеграция сайта
    - b) Автоматизация этапов разработки, тестирования и развёртывания программного обеспечения
    - c) Просто добавление стилей к сайту
    - d) Составление веб-сервера
14. Какие есть альтернативы CI/CD?
    - a) Jenkins

- b) GitLab CI
- c) CircleCI
- d) Все перечисленные

15. В чем преимущество контейнеров перед виртуальной машиной?
    * a) Быстрая загрузка
    * b) Меньшее потребление ресурсов
    * c) Портативность
    * d) Все перечисленное

Ключи ответов:

Тема 2.1:

1. c
2. b
3. b
4. b
5. c
6. b
7. b
8. c
9. b
10. b
11. a
12. d
13. a
14. a
15. b

Тема 3.1:

1. b
2. b
3. c
4. a
5. b
6. b
7. b
8. c
9. d
10. d
11. b
12. b
13. b
14. a

15.c

Тема 1.1:

1. b
2. b
3. c
4. b
5. b
6. b
7. b
8. b
9. d
10. a
11. c
12. a
13. b
14. d
15. d

Тесты для промежуточного контроля МДК 03.02.

Тема 2.1: Виды и назначения серверных языков программирования, среды разработки

(Вопросы основаны на практическом занятии №1: Установка и настройка языка программирования)

1. Что необходимо установить для запуска Python-программ?
    - a) Веб-браузер
    - b) Интерпретатор Python
    - c) Текстовый редактор
    - d) Антивирусное ПО
2. Какой инструмент используется для установки дополнительных библиотек в Python?
    - a) git
    - b) pip
    - c) npm
    - d) apt
3. Что такое виртуальное окружение в Python?
    - a) Тип базы данных
    - b) Изолированная среда для запуска Python-проектов
    - c) Веб-сервер
    - d) Система контроля версий
4. Какой командой создается виртуальное окружение в Python?

- a) python -m virtualenv venv
- b) python -m venv
- c) create venv
- d) virtualenv create

5. Что такое requirements.txt?
   - a) Файл с настройками IDE
   - b) Файл с описанием зависимостей Python-проекта
   - c) Файл с HTML-кодом
   - d) Файл с CSS-стилями
6. Какой программой открывают файл py?
   - a) Текстовый редактор
   - b) Браузер
   - c) Интерпретатор
   - d) Microsoft Word
7. Что такое PyCharm?
   - a) Библиотека для js
   - b) Система контроля
   - c) Фреймворк для тестов
   - d) IDE
8. Какие данные могут находится в readme файле?
   - a) Описание проекта
   - b) Инструкция по использованию
   - c) Обо всем
   - d) Об авторе
9. Что делают с виртуальным окружением?
   - a) Архивируют
   - b) Устанавливают зависимости
   - c) Удаляют
   - d) Сжимают
10. Какое расширение файла для скрипта?
    - a) tx
    - b) js
    - c) sql
    - d) py
11. Что необходимо прописать в PATH для использования python в командной строке?
    - a) Название директории
    - b) Путь до python.exe
    - c) ничего
    - d) все файлы python

12. Где находится документация для модулей?
    - a) В интернете
    - b) Должна быть документация
    - c) В коде
    - d) На диске
13. Какой менеджер для библиотек используется?
    - a) pip
    - b) js
    - c) html
    - d) text
14. Как активировать venv?
    - a) venv.activate
    - b) source venv/bin/activate
    - c) cd venw/source
    - d) sudo activate
15. Что такое IDE (Integrated Development Environment)?
    - a) Набор библиотек для работы с базой данных
    - b) Комплекс инструментов для разработки программного обеспечения
    - c) Язык разметки веб-страниц
    - d) Система контроля версий

(Вопросы основаны на практическом занятии №2: Выполнение типовой практической задачи на основе процедурного подхода к программированию)

1. Что такое процедурное программирование?
    - a) Подход, основанный на объектах
    - b) Подход, основанный на функциях
    - c) Подход, основанный на классах
    - d) Подход, основанный на событиях
2. Как создать функцию в Python?
    - a) class MyFunction:
    - b) function MyFunction():
    - c) def MyFunction():
    - d) void MyFunction():
3. Какой тип данных лучше всего подходит для хранения списка задач в ToDo List?
    - a) Строка
    - b) Число
    - c) Список
    - d) Словарь

4. Как открыть файл для записи в Python?
    - a) open("file.txt", "r")
    - b) open("file.txt", "w")
    - c) open("file.txt", "a")
    - d) open("file.txt", "x")
5. Как закрыть файл в Python?
    - a) file.close()
    - b) close(file)
    - c) file.end()
    - d) end(file)
6. Как вернуть значения из функций?
    - a) с помощью output
    - b) с помощью return
    - c) с помощью output_data
    - d) ничего не нужно, значения и так вернутся
7. Какой цикл лучше использовать, если нужно перебрать все значения списка?
    - a) for
    - b) while
    - c) do-while
    - d) нужно использовать range
8. Для чего нужна функция в программировании?
    - a) Чтобы упростить код
    - b) Чтобы не повторять код
    - c) Чтобы можно было читать код
    - d) для всего перечисленного
9. Чем процедурное программирование отличается от объектно-ориентированного?
    - a) Только названием
    - b) Разными подходами к организации кода
    - c) Только переменными
    - d) Циклами
10. Что значит "обработать исключения"?
    - a) Сообщить об ошибках
    - b) Вернуть код на шаг назад
    - c) Попробовать решить проблему в коде
    - d) Поймать ошибку и перенаправить код
11. Зачем нужно сохранять todo list в файл?
    - a) Указать на наличие задач
    - b) Чтобы задачи не потерялись после закрытия программы

- c) Все равно никто не пользуется
- d) Чтобы пользователь понимал что он использует

12. Какие файлы используются для записи, чтения и добавления?
    - a) w, r, rw
    - b) w, a, r
    - c) w, r, a
    - d) a, r, rw
13. Зачем нужен json?
    - a) Шифрования
    - b) Валидации
    - c) Читаемости и хранения данных
    - d) Оптимизации
14. Почему важно обрабатывать ошибки?
    - a) Для работы
    - b) Для пользователя
    - c) Чтобы ошибки не влияли на остальную работу программы
    - d) Их невозможно обработать
15. Что такое ""цикл""?
    - a) Повторяющийся участок кода
    - b) Что-то на фоне
    - c) Изображение
    - d) Оператор

(Вопросы основаны на практическом занятии №3: Выполнение типовой практической задачи на основе объектно-ориентированного программирования)

1. Что такое класс в ООП?
    - a) Функция
    - b) Шаблон для создания объектов
    - c) Переменная
    - d) Тип данных
2. Что такое объект в ООП?
    - a) Класс
    - b) Экземпляр класса
    - c) Функция
    - d) Переменная
3. Какой принцип ООП подразумевает сокрытие внутренней реализации объекта от внешнего мира?
    - a) Наследование
    - b) Полиморфизм
    - c) Инкапсуляция

- d) Абстракция

4. Что такое наследование в ООП?
    - a) Создание нового объекта на основе существующего
    - b) Сокрытие внутренней реализации
    - c) Возможность объекту принимать разные формы
    - d) Передача данных между объектами
5. Что такое полиморфизм в ООП?
    - a) Создание нового класса на основе существующего
    - b) Сокрытие внутренней реализации
    - c) Возможность объекту принимать разные формы
    - d) Передача данных между объектами
6. Что такое инкапсуляция?
    - a) Наследование
    - b) Сокрытие реализации
    - c) Имитация
    - d) Создание нового объекта
7. Для чего нужно ""наследование""?
    - a) Для исключения кода
    - b) Для переиспользования кода
    - c) Для вызова функций
    - d) Для безопасности
8. Как вызвать метод класса?
    - a) obj = MyClass()
    - b) obj: MyClass
    - c) call MyClass
    - d) new MyClass
9. Что такое родительский и дочерний класс?
    - a) Классы, которые наследуют друг друга
    - b) Классы, которые расположены на разных страницах
    - c) Синонимы
    - d) Классы, которые взаимодействуют друг с другом
10. Какую задачу выполняет конструктор?
    - a) Обрабатывает данные
    - b) Вставляет информацию
    - c) Создает объект
    - d) Вызывает другие функции
11. Что такое интерфейс?
    - a) набор абстрактных методов
    - b) Новый стиль программирования
    - c) Набор html

- d) Функция

12. Как использовать приватные поля?
    - a) С помощью get set
    - b) С помощью this
    - c) Нельзя
    - d) С помощью private
13. Что позволяет использовать абстрактный класс?
    * a) Не реализовывать методы
    * b) Получать данные с сервера
    * c) Настраивать
    * d) Запускать
14. Зачем нужен полиморфизм?
    - a) Для отправки данных
    - b) Для расширения функциональности
    - c) Для оптимизации
    - d) Чтобы было красиво
15. Что такое экземпляры класса?
    - a) Связь между классами
    - b) Функция
    - c) Все объекты этого класса
    - d) Не знаю

(Вопросы основаны на практическом занятии №4: Создание программных модулей используя шаблоны проектирования)

1. Что такое шаблон проектирования?
    - a) Готовый код для решения определенной задачи
    - b) Описание повторяющейся проблемы и ее решения
    - c) Стандартный интерфейс для работы с базами данных
    - d) Метод тестирования программного обеспечения
2. Какой шаблон проектирования ограничивает создание только одного экземпляра класса?
    - a) Factory
    - b) Singleton
    - c) Observer
    - d) Strategy
3. Какой шаблон проектирования позволяет создавать объекты без указания конкретного класса?
    - a) Singleton
    - b) Factory Method
    - c) Observer
    - d) Strategy
4. Какой шаблон проектирования определяет семейство алгоритмов,

инкапсулирует их и делает их взаимозаменяемыми?
    - a) Singleton
    - b) Factory Method
    - c) Observer
    - d) Strategy
5. Какой шаблон проектирования определяет зависимость типа "один ко многим" между объектами?
    - a) Singleton
    - b) Factory Method
    - c) Observer
    - d) Strategy
6. Для чего нужен шаблон проектирования?
    - a) Для ограничения
    - b) Для удобства
    - c) Для унификации
    - d) Для всего перечисленного
7. Какой шаблон позволяет создавать объекты без указания класса?
    - a) Singleton
    - b) Factory
    - c) Prototype
    - d) Adapter
8. Как работает шаблон проектирования Singleton?
    - a) Создание одного объекта
    - b) Создание нескольких
    - c) Копирование
    - d) Подсчет
9. Какие есть типы шаблонов проектирования?
    - a) Порождающие
    - b) Структурные
    - c) Поведенческие
    - d) Все перечисленные
10. Что такое порождающие шаблоны?
    - a) Общие стратегии решения задач создания объектов
    - b) Отношения между объектами
    - c) Взаимодействие между объектами
    - d) Поведение объекта
11. Для чего нужен шаблон проектирования Builder?
    - a) Для создания сложных объектов
    - b) Для упрощения кода
    - c) Для контроля

- d) Для обработки данных

12. Что позволяет сделать Facade?
    - a) Наследование
    - b) Изменение кода
    - c) упрощение сложной системы
    - d) Перезаписать код
13. Что позволяет Iterator?
    * a) Последовательный перебор
    * b) Массивы
    * c) Запись данных
    * d) Получать данные с сервера
14. Что позволяет Observer?
    - a) Контролировать
    - b) При создании объекта
    - c) Уведомлять
    - d) Отслеживать данные
15. Для чего нужен Strategy?
    - a) Для отладки
    - b) Для определения разных стратегий поведения объектов
    - c) Создания нескольких объектов
    - d) Для красоты

Ключи ответов:

(Вопросы основаны на практическом занятии №1: Установка и настройка языка программирования)

1. b
2. b
3. b
4. b
5. b
6. c
7. d
8. c
9. b
10. d
11. b
12. a
13. a
14. b
15. b

(Вопросы основаны на практическом занятии №2: Выполнение типовой практической задачи на основе процедурного подхода к

программированию)

1. b
2. c
3. c
4. b
5. a
6. b
7. a
8. d
9. b
10. d
11. b
12. c
13. c
14. c
15. a

(Вопросы основаны на практическом занятии №3: Выполнение типовой практической задачи на основе объектно-ориентированного программирования)

1. b
2. b
3. c
4. a
5. c
6. b
7. b
8. a
9. a
10. c
11. a
12. a
13. a
14. b
15. c

(Вопросы основаны на практическом занятии №4: Создание программных модулей используя шаблоны проектирования)

1. b
2. b
3. b
4. d

5. c
6. d
7. b
8. a
9. d
10. a
11. a
12. c
13. a
14. c
15. b

Тема 2.2: Программирование на стороне сервера (по выбору)
Вариант 1

1. Какой тип данных служит для хранения символов в Python?
    - a) int
    - b) float
    - c) str
    - d) bool
2. Как создать пустой список в Python?
    - a) list = {}
    - b) list = ()
    - c) list = []
    - d) list = list()
3. Какой оператор используется для проверки равенства в Python?
    - a) =
    - b) ==
    - c) :=
    - d) is
4. Какой цикл выполнится хотя бы один раз, даже если условие ложно?
    - a) for
    - b) while
    - c) do...while (в Python отсутствует)
    - d) отсутствует цикл
5. Как объявить функцию в Python?
    - a) function myFunction():
    - b) def myFunction():
    - c) void myFunction():
    - d) int myFunction():
6. Какая функция используется для открытия файла в Python?

- a) readfile()
- b) open()
- c) fileopen()
- d) fopen()

7. Что такое SQL-запрос?
   - a) Инструкция на языке SQL для взаимодействия с базой данных
   - b) Текстовый файл для базы данных
   - c) Структура данных для веб-страницы
   - d) Тип данных
8. Каким способом нужно сделать запрос в базу данных?
   - a) С помощью SQL
   - b) С помощью GET
   - c) С помощью POST
   - d) С помощью запроса на сайт
9. Что такое переменные?
   - a) данные
   - b) функции
   - c) структура данных
   - d) именованное хранилище
10. Что позволяет реализовать ветвление кода?
    - a) переменные
    - b) циклы
    - c) ветвление
    - d) функции
11. Как добавить элемент в массив?
    - a) добавление = []
    - b) добавление.push
    - c) array.append
    - d) array <<
12. Для чего нужны try, catch блоки?
    - a) Чтобы отловить ошибки
    - b) Чтобы ничего не сломалось
    - c) Чтобы пользователь не видел ошибок
    - d) Для всего перечисленного
13. Как вызвать функцию?
    * a) call()
    * b) run()
    * c) с помощью имени функции
    * d) open()
14. Какой формат для открытия файла (только чтение)?

- a) a+
- b) w+
- c) w
- d) r

15. Какие типы данных используются в python?
    - a) int, string, boolean, float
    - b) char, text, double
    - c) number, text
    - d) нет типов
16. Что означает цикл for?
    - a) выполнение кода пока условие истинно
    - b) выполнить код поочереди для всех элементов
    - c) для выбора
    - d) повтор кода
17. Что делает оператор ""=="
    - a) Передает данные
    - b) Изменяет данные
    - c) Сравнивает данные
    - d) Передает данные
18. Что такое глобальная и локальная переменные?
    - a) Показывают где находятся данные
    - b) Показывает какой код относится к определенному блоку
    - c) Разделяют область видимости
    - d) Задают тип
19. Сколько раз выполнится цикл "for"?
    - a) Зависит от кода
    - b) 0
    - c) 1
    - d) Зависит от длинны массива
20. Что будет если в цикле не будет ""break""?
    * a) Ошибка
    * b) Зависит от условия
    * c) Будет выполнятся бесконечно
    * d) Ничего

Вариант 2

1. Какой тип данных служит для хранения чисел с плавающей запятой в Python?
    - a) int
    - b) float
    - c) str

- d) bool

2. Как получить длину списка в Python?
   - a) list.size()
   - b) list.length()
   - c) len(list)
   - d) list.count()
3. Какой оператор используется для проверки неравенства в Python?
   - a) =
   - b) ==
   - c) !=
   - d) is not
4. Какой цикл используется для повторения блока кода, пока условие истинно?
   - a) for
   - b) while
   - c) do...while (в Python отсутствует)
   - d) Отсутствует
5. Как вернуть значение из функции в Python?
   - a) return()
   - b) return value
   - c) send()
   - d) print value
6. Какая функция используется для чтения данных из файла в Python?
   - a) write()
   - b) read()
   - c) open()
   - d) readFile()
7. Какой оператор SQL используется для выбора данных из таблицы?
   - a) INSERT
   - b) UPDATE
   - c) SELECT
   - d) DELETE
8. Что делает команда DELETE из SQL?
   - a) Добавление данных
   - b) Получение
   - c) Изменение
   - d) Удаление
9. Что содержит заголовок скрипта?
   - a) комментарий, указывающий интерпретатор
   - b) основные константы

- c) библиотеку
- d) ничего

10. Какие существуют логические операторы?
    - a) ==, !=, >=, <=
    - b) and, or, not
    - c) def, int, float
    - d) только ==
11. Где хранятся данные в массиве?
    - a) В элементах
    - b) В массиве
    - c) В индексе
    - d) Не хранятся
12. Для чего нужно логирование?
    - a) Для отображения данных
    - b) Для записи информации
    - c) Для тестов
    - d) В качестве аргументов
13. Как остановить цикл?
    - a) Stop
    - b) Остановить
    - c) Return
    - d) Break
14. Как называется переменная, с помощью которой открывается файл?
    - a) Stream
    - b) Folder
    - c) Text
    - d) File
15. Что позволяет использовать "with"?
    * a) Открыть файл
    * b) Гарантирует закрытие файла
    * c) Автоматизирует открытие
    * d) Поменять размер
16. Что такое рекурсия?
    - a) Повторный вызов функции
    - b) Выполнение кода поочередно
    - c) Что-то на уровне системы
    - d) Новый код
17. С помощью какой команды можно угадать тип переменной?
    - a) Guess_type
    - b) typeof

- c) show
- d) type

18. Что означает "область видимости"?
* a) Область, которая видит
* b) На что имеет право объект
* c) Где видна переменная
* d) Что будет выполнять программа

19. Как называются массивы?
- a) Tuple
- b) Map
- c) List
- d) Set

20. Что нужно, чтобы при делении не было остатка?
* a) int()
* b) int() + if == 0
* c) float
* d) ничег

МДК 03.03

Коллоквиум: Термины по предмету МДК 03.03 "Разработка информационных ресурсов с использованием программных платформ"

1. **Веб-фреймворк:** Готовая программная платформа, упрощающая разработку веб-приложений, предоставляя инструменты для маршрутизации, работы с шаблонами, управления базами данных и обеспечения безопасности.
2. **Маршрутизация (Routing):** Механизм в веб-фреймворках, определяющий соответствие между URL-адресами и функциями-обработчиками, отвечающими за генерацию ответа на запрос пользователя.
3. **Шаблонизатор:** Программный компонент, предназначенный для динамической генерации HTML-страниц на основе шаблонов и данных, полученных от сервера.
4. **База данных (Database):** Организованная структура для хранения и управления данными, обеспечивающая эффективный поиск, доступ и обновление информации.
5. **ORM (Object-Relational Mapping):** Технология, позволяющая взаимодействовать с базами данных через объекты, представляющие записи в таблицах, упрощая разработку и повышая переносимость кода.
6. **Модель (Model):** Представление данных в веб-приложении, часто реализуемое в виде классов, которые отражают структуру таблиц в базе данных и предоставляют методы для работы с данными.

7. **Валидация данных:** Процесс проверки данных, поступающих от пользователя, на соответствие заданным требованиям и ограничениям, направленный на обеспечение корректности и безопасности данных.
8. **REST API:** Архитектурный стиль для создания веб-сервисов, основанный на использовании HTTP-методов (GET, POST, PUT, DELETE) для выполнения операций над ресурсами.
9. **Аутентификация:** Процесс проверки подлинности пользователя, направленный на подтверждение его права доступа к определенным ресурсам или функциям веб-приложения.
10. **Авторизация:** Процесс определения прав доступа пользователя к конкретным ресурсам или функциям веб-приложения, основанный на его аутентифицированных данных.
11. **Сессия (Session):** Механизм для хранения информации о состоянии пользователя между HTTP-запросами, позволяющий поддерживать контекст взаимодействия с веб-приложением.
12. **CSRF (Cross-Site Request Forgery):** Тип веб-атак, при которой злоумышленник выполняет действия от имени пользователя, не подозревающего о происходящем.
13. **Шаблон проектирования:** Типовое решение часто встречающейся проблемы проектирования программного обеспечения, позволяющее создавать более гибкий, масштабируемый и поддерживаемый код.
14. **Middleware:** Программный компонент, который перехватывает HTTP-запросы и ответы между веб-сервером и приложением, позволяя выполнять различные операции, такие как аутентификация, логирование или преобразование данных.
15. **Контейнеризация:** Технология упаковки приложения и всех его зависимостей (библиотек, исполняемой среды) в один образ, который может быть легко перенесен и запущен на любой платформе.

МДК 03.01. Разработка информационных ресурсов с использованием программных платформ

Инструкция: Выберите один правильный ответ из предложенных вариантов.

Блок 1: Введение в веб-фреймворки

1. Что такое веб-фреймворк?
   - a) Набор готовых HTML-шаблонов
   - b) Программная платформа, упрощающая разработку веб-приложений
   - c) Инструмент для графического дизайна веб-сайтов
   - d) Система управления базами данных
2. Какую задачу решает маршрутизация во фреймворке?
   - a) Стилизацию веб-страниц
   - b) Сопоставление URL-адресов с функциями-обработчиками
   - c) Валидацию введенных пользователем данных
   - d) Шифрование данных

3. Что такое шаблонизатор?
    - a) Инструмент для создания CSS-стилей
    - b) Компонент для динамической генерации HTML-страниц
    - c) Библиотека для работы с JavaScript
    - d) Механизм для управления сессиями пользователей
4. Какое из перечисленных преимуществ *не* относится к использованию веб-фреймворков?
    - a) Ускорение разработки
    - b) Повышение безопасности
    - c) Улучшение структуры кода
    - d) Увеличение размера итогового приложения
5. Какой архитектурный паттерн часто используется в веб-фреймворках?
    - a) Publish-Subscribe
    - b) Singleton
    - c) MVC (Model-View-Controller)
    - d) Observer
6. Какой уровень MVC отвечает за взаимодействие с базой данных?
    - a) View
    - b) Controller
    - c) Model
    - d) Template
7. Что не относится к задачам Controller?
   * a) Обработка запросов
   * b) Задание логики обработки запросов
   * c) Генерация интерфейса
   * d) Получение данных
8. Какой файл содержит структуру?
    - a) Control
    - b) Module
    - c) View
    - d) Model
9. Что означает "" DRY""
    - a) Не сушите код
    - b) Повторите код
    - c) Не повторяйте код
    - d) не мочить код
10. Какие задачи выполняет View?
    - a) обработка и логика
    - b) Создание интерфейса
    - c) Управление
    - d) Логирование

Блок 2: Базы данных и модели

1. Что такое база данных?
    - a) Набор HTML-файлов

- b) Организованная структура для хранения и управления данными
- c) Библиотека JavaScript
- d) Инструмент для редактирования изображений

2. Что такое ORM (Object-Relational Mapping)?
   - a) Язык запросов к базе данных
   - b) Библиотека для работы с графикой
   - c) Технология для преобразования кода на JavaScript
   - d) Способ взаимодействия с базой данных через объекты
3. Какая из перечисленных баз данных является реляционной?
   - a) MongoDB
   - b) Redis
   - c) MySQL
   - d) Cassandra
4. Какая из перечисленных баз данных является NoSQL базой данных?
   - a) MySQL
   - b) PostgreSQL
   - c) Oracle
   - d) MongoDB
5. Что такое SQL-инъекция?
   - a) Тип атаки, при которой вредоносный SQL-код внедряется в запросы к базе данных
   - b) Метод шифрования данных
   - c) Технология для ускорения работы базы данных
   - d) Способ создания резервных копий базы данных
6. Что такое миграция в контексте баз данных?
   - a) Процесс переноса базы данных на другой сервер
   - b) Изменения структуры базы данных
   - c) Оптимизация запросов к базе данных
   - d) Резервное копирование данных
7. Какой SQL-запрос используется для добавления данных?
   - a) DELETE
   - b) CREATE
   - c) INSERT
   - d) UPDATE
8. Что такое транзакция?
   - a) Неделимая последовательность операций над базой данных
   - b) Почтовый перевод
   - c) Финансовая операция
   - d) Оплата хостинга
9. Что такое первичный ключ?
   - a) Значение для быстрого выбора
   - b) Уникальный индентификатор строки
   - c) Значение для индексации
   - d) Столбец для отображения данных

10. В чем разница между left join и right join?
    - a) Связывают все данные только слева или только справа
    - b) Связывают только часть данных
    - c) Не существует такой команды
    - d) Легко запомнить

Блок 3: Безопасность веб-приложений

1. Что такое XSS (Cross-Site Scripting)?
    - a) Тип SQL-инъекции
    - b) Тип атаки, при которой злоумышленник внедряет вредоносный JavaScript-код на веб-страницу
    - c) Метод шифрования данных
    - d) Способ защиты от спама
2. Для чего используется HTTPS?
    - a) Для ускорения загрузки веб-страниц
    - b) Для шифрования данных, передаваемых между браузером и сервером
    - c) Для валидации введенных пользователем данных
    - d) Для уменьшения размера веб-страниц
3. Что такое хеширование паролей?
    - a) Шифрование данных в базе данных
    - b) Преобразование пароля в необратимый вид для безопасного хранения
    - c) Замена букв в пароле на цифры
    - d) Скрытие пароля от пользователя
4. Что такое "соль" при хешировании паролей?
    - a) Случайная строка, добавляемая к паролю перед хешированием
    - b) Специальный символ в пароле
    - c) Метод шифрования данных
    - d) Название алгоритма хеширования
5. Для чего нужны CSRF-токены?
    - a) Защиты от спама
    - b) Защиты базы данных
    - c) Защиты cookies
    - d) Защиты от CSRF атак
6. Что такое защита от SQL-инъекций?
    - a) защита запросов
    - b) защита данных
    - c) фильтрация данных, поступающих в SQL-запрос
    - d) не требуется
7. Какие угрозы безопасности?
    - a) XSS
    - b) Инъекции
    - c) Сбор данных
    - d) Все перечисленные

8. Что можно использовать в качестве авторизации?
    - a) JSON
    - b) JWT
    - c) httpOnly
    - d) cookie
9. Какой самый надёжный способ хранения пароля?
    - a) hash
    - b) password
    - c) text
    - d) ничего
10. Что такое fail2ban?
    - a) Свободное программное обеспечение для защиты компьютерных систем от брутфорс атак
    - b) Улучшение html
    - c) новый js
    - d) улучшение загрузки сайта

Ключи ответов:

Блок 1:

1. b
2. b
3. b
4. d
5. c
6. c
7. c
8. d
9. c
10. b

Блок 2:

1. b
2. d
3. c
4. d
5. a
6. b
7. c
8. a
9. b
10. a

Блок 3:

1. b
2. b
3. b
4. a

5. d
6. c
7. d
8. b
9. a
10. a

5 ЭКЗАМЕНАЦИОННЫЕ БИЛЕТЫ

5.1 Экзаменационные билеты по дисциплине МДК 03.01

Билет №1

Теоретический вопрос 1: Объясните разницу между языками программирования с динамической и статической типизацией. Приведите примеры.

Теоретический вопрос 2: Что такое рефакторинг кода и зачем он нужен?

Практическое задание: Разработайте функцию на Python, принимающую список чисел и возвращающую их сумму.

Билет №2

Теоретический вопрос 1: Опишите основные принципы объектно-ориентированного программирования (ООП).

Теоретический вопрос 2: Что означает термин "масштабируемость" применительно к веб-приложениям?

Практическое задание: Создайте класс Rectangle на Python с атрибутами width и height. Добавьте методы для вычисления площади и периметра.

Билет №3

Теоретический вопрос 1: Что такое система контроля версий (VCS) и зачем она нужна?

Теоретический вопрос 2: Что такое Dependency Injection?

Практическое задание: Инициализируйте Git-репозиторий в новой папке, добавьте файл README.md, закоммитьте его и создайте новую ветку с именем feature.

Билет №4

Теоретический вопрос 1: Объясните разницу между GET и POST запросами в HTTP протоколе.

Теоретический вопрос 2: Что такое веб-фреймворк?

Практическое задание: Используя библиотеку requests в Python, сделайте GET-запрос к сайту https://www.example.com и выведите код ответа.

Билет №5

Теоретический вопрос 1: Что такое JSON и как он используется для обмена данными между сервером и клиентом?

Теоретический вопрос 2: Что такое ORM (Object-Relational Mapping)?

Практическое задание: Создайте словарь в Python, представляющий информацию о книге (название, автор, год издания), и преобразуйте его в JSON-строку.

Билет №6

Теоретический вопрос 1: Что такое виртуализация?

Теоретический вопрос 2: Что такое гипервизор?

Практическое задание: Установите VirtualBox или VMware Player и создайте виртуальную машину с операционной системой Linux.

Билет №7

Теоретический вопрос 1: Объясните, что такое контейнеризация и какие преимущества она дает по сравнению с виртуализацией.

Теоретический вопрос 2: Что такое docker?

Практическое задание: Установите Docker и запустите контейнер с образом hello-world.

Билет №8

Теоретический вопрос 1: Что такое веб-сервер и как он обрабатывает HTTP-запросы?

Теоретический вопрос 2: Что такое REST API?

Практическое задание: Установите веб-сервер Apache или Nginx и настройте его для обслуживания статического HTML-файла.

Билет №9

Теоретический вопрос 1: Что такое база данных и какие типы баз данных существуют?

Теоретический вопрос 2: Что такое blueprint во flask?

Практическое задание: Установите СУБД MySQL/MariaDB или PostgreSQL и создайте новую базу данных с таблицей users (id, username, email).

Билет №10

Теоретический вопрос 1: Что такое SQL и какие основные операторы SQL вы знаете?

Теоретический вопрос 2: Для чего используются шаблонизаторы в веб-фреймворках?

Практическое задание: Напишите SQL-запрос для добавления нового пользователя в таблицу users (см. билет №9).

Билет №11

Теоретический вопрос 1: Что такое CI/CD (Continuous Integration/Continuous Delivery)?

Теоретический вопрос 2: Какие есть альтернативы CI/CD?

Практическое задание: Опишите, какие шаги нужно предпринять для настройки CI/CD для веб-приложения с использованием Jenkins.

Билет №12

Теоретический вопрос 1: Объясните, что такое тестирование программного обеспечения и какие уровни тестирования существуют.

Теоретический вопрос 2: Что такое SOLID?

Практическое задание: Используя библиотеку unittest или pytest в Python, напишите тест для функции, которая проверяет, является ли число четным.

Билет №13

Теоретический вопрос 1: Что такое облачные вычисления и какие модели облачных сервисов существуют (IaaS, PaaS, SaaS)?

Теоретический вопрос 2: Что позволяет использовать контейнеры?

Практическое задание: Создайте аккаунт на платформе AWS, Google Cloud Platform или Azure и запустите виртуальную машину.

Билет №14

Теоретический вопрос 1: Что такое микросервисная архитектура и каковы ее преимущества и недостатки?

Теоретический вопрос 2: Что такое cookie?

Практическое задание: Опишите, как можно разделить монолитное веб-приложение на микросервисы.

Билет №15

Теоретический вопрос 1: Что такое API (Application Programming Interface) и какие типы API вы знаете?

Теоретический вопрос 2: Что такое инкапсуляция?

Практическое задание: Разработайте простой REST API с использованием Flask или Django для управления списком книг (добавление, просмотр, редактирование, удаление).

Билет №16

Теоретический вопрос 1: Что такое Docker Compose и как его использовать

для запуска нескольких контейнеров?

Теоретический вопрос 2: Для чего нужно Middleware?

Практическое задание: Создайте файл docker-compose.yml для запуска веб-приложения с базой данных MySQL.

Билет №17

Теоретический вопрос 1: Что такое Kubernetes и как он используется для оркестрации контейнеров?

Теоретический вопрос 2: Какие есть способы изоляции предоставляет контейнеризация?

Практическое задание: Опишите процесс развертывания веб-приложения в Kubernetes с использованием Deployment и Service.

Билет №18

Теоретический вопрос 1: Что такое автоматизация конфигурации и какие инструменты для этого используются (Ansible, Chef, Puppet)?

Теоретический вопрос 2: В чем суть полиморфизма?

Практическое задание: Напишите Ansible playbook для установки Apache и PHP на сервере Ubuntu.

Билет №19

Теоретический вопрос 1: Что такое мониторинг и для чего он нужен в веб-разработке?

Теоретический вопрос 2: Что позволяет делать система оркестрации Kubernetes?

Практическое задание: Установите инструмент мониторинга (например, Prometheus или Grafana) и настройте его для отслеживания основных метрик веб-сервера.

Билет №20

Теоретический вопрос 1: Объясните, что такое базы данных NoSQL и в каких случаях их целесообразно использовать.

Теоретический вопрос 2: В чем преимущество контейнеров перед виртуальной машиной?

Практическое задание: Установите MongoDB и создайте новую базу данных с коллекцией products (name, price, description).

Билет №21

Теоретический вопрос 1: Что такое шаблоны проектирования? Приведите несколько примеров.

Теоретический вопрос 2: Что такое область видимости (scope) переменной в

JavaScript?
Практическое задание: Покажите на примере Python-кода реализацию шаблона "Фабрика".

Билет №22
Теоретический вопрос 1: Что такое деплой?
Теоретический вопрос 2: Что такое cookie?
Практическое задание: Опишите процесс деплоя своего пет-проекта.

Билет №23
Теоретический вопрос 1: Какие метрики самые важные для мониторинга?
Теоретический вопрос 2: Что такое сессии?
Практическое задание: Реализуйте мониторинг метрик с помощью Prometheus

---

Билет №24
Теоретический вопрос 1: Что такое нагрузочное тестирование?
Теоретический вопрос 2: Что такое автоматизация?
Практическое задание: Реализуйте нагрузочный тест своего сайта с помощью Locust

---

Билет №25
Теоретический вопрос 1: Что такое безопасность?
Теоретический вопрос 2: Для чего нужен Docker Compose?
Практическое задание: Выполните тест безопасности на своем веб-сайте.

5.2 Экзаменационные билеты по МДК 03.02 "Разработка кода информационных ресурсов"

Инструкция: Каждый билет содержит два теоретических вопроса и одно практическое задание. Ответьте на вопросы и выполните практическое задание.

Билет № 1
Вопрос 1: Что такое интерпретируемые и компилируемые языки программирования? Приведите примеры для каждого типа.
Вопрос 2: Какие основные типы данных есть в Python? Приведите примеры.
Практическое задание: Напишите Python-скрипт, который запрашивает у пользователя имя и возраст, а затем выводит приветствие, содержащее эти данные.

Билет № 2

Вопрос 1: Что такое серверные языки программирования и для чего они используются?

Вопрос 2: Что такое переменные и как их использовать в Python?

Практическое задание: Напишите Python-скрипт, который вычисляет площадь прямоугольника на основе введенных пользователем значений длины и ширины.

Билет № 3

Вопрос 1: Что такое процедурный подход в программировании?

Вопрос 2: Что такое условные операторы в Python и как они работают?

Практическое задание: Напишите Python-скрипт, который проверяет, является ли введенное пользователем число четным или нечетным.

Билет № 4

Вопрос 1: Что такое объектно-ориентированный подход в программировании (ООП)?

Вопрос 2: Что такое циклы в Python и как их использовать?

Практическое задание: Напишите Python-скрипт, который выводит таблицу умножения для введенного пользователем числа (от 1 до 10).

Билет № 5

Вопрос 1: Что такое классы и объекты в Python? Приведите пример.

Вопрос 2: Что такое функции в Python и как их определять и вызывать?

Практическое задание: Напишите Python-функцию, которая принимает строку в качестве аргумента и возвращает ее в обратном порядке.

Билет № 6

Вопрос 1: Что такое наследование, полиморфизм и инкапсуляция (основные принципы ООП)?

Вопрос 2: Что такое списки и кортежи в Python и чем они отличаются?

Практическое задание: Напишите Python-скрипт, который создает список чисел и вычисляет их сумму и среднее значение.

Билет № 7

Вопрос 1: Что такое модули и пакеты в Python?

Вопрос 2: Что такое словари в Python и как их использовать?

Практическое задание: Напишите Python-скрипт, который создает словарь с данными о студентах (имя, возраст, специальность) и выводит информацию о конкретном студенте по его имени.

Билет № 8

Вопрос 1: Что такое исключения в Python и как их обрабатывать?

Вопрос 2: Как работать с файлами в Python (чтение и запись)?

Практическое задание: Напишите Python-скрипт, который читает данные из текстового файла и записывает их в другой файл в обратном порядке.

Билет № 9

Вопрос 1: Что такое рефакторинг кода и каковы его цели?

Вопрос 2: Что такое SQLite и для чего он используется?

Практическое задание: Создайте базу данных SQLite с одной таблицей (например, "users" с полями "id", "name", "email").

Билет № 10

Вопрос 1: Какие методы рефакторинга кода вы знаете?

Вопрос 2: Как подключиться к базе данных SQLite в Python?

Практическое задание: Напишите Python-скрипт, который подключается к базе данных SQLite и выводит список всех пользователей из таблицы "users".

Билет № 11

Вопрос 1: Что такое шаблоны проектирования программного обеспечения?

Вопрос 2: Как добавить данные в таблицу SQLite в Python?

Практическое задание: Напишите Python-скрипт, который добавляет нового пользователя в таблицу "users" в базе данных SQLite.

Билет № 12

Вопрос 1: Какие шаблоны проектирования вы знаете?

Вопрос 2: Как выполнить запрос SELECT в SQLite с использованием Python?

Практическое задание: Напишите Python-скрипт, который выбирает из таблицы "users" всех пользователей с определенным именем или email.

Билет № 13

Вопрос 1: Что такое MVC?

Вопрос 2: Как выполнить запрос UPDATE в SQLite с использованием Python?

Практическое задание: Напишите Python-скрипт, который обновляет email пользователя в таблице "users" в базе данных SQLite.

Билет № 14

Вопрос 1: Что такое шаблон проектирования Factory?

Вопрос 2: Как выполнить запрос DELETE в SQLite с использованием Python?

Практическое задание: Напишите Python-скрипт, который удаляет пользователя из таблицы "users" в базе данных SQLite по его ID.

Билет № 15

Вопрос 1: Что такое шаблон проектирования Singleton?

Вопрос 2: Как использовать параметры в SQL-запросах для защиты от SQL-инъекций?

Практическое задание: Напишите безопасный Python-скрипт, который выбирает пользователя из таблицы "users" по имени, используя параметризованный запрос.

Билет № 16

Вопрос 1: Что такое шаблон проектирования Observer?

Вопрос 2: Как использовать fetchone(), fetchall() и execute()?

Практическое задание: Создайте python-скрипт для добавления, удаления, получения и изменения информации в SQLite таблице.

Билет № 17

Вопрос 1: Для чего нужно использовать try except блоки?

Вопрос 2: Какие есть способы задания строки запроса в SQL и что лучше использовать?

Практическое задание: Создайте python-скрипт для подключения к БД и вывода информации, обработайте возможные ошибки.

Билет № 18

Вопрос 1: Как подключить модуль?

Вопрос 2: Какие есть способы использования for?

Практическое задание: Разработайте систему для вывода расписания, с подключение к SQLite БД и чтением оттуда.

Билет № 19

Вопрос 1: Какие есть способы создания БД, кроме SQLite?

Вопрос 2: Какие есть типы запросов?

Практическое задание: Напишите скрипт для подсчета общей суммы чего-либо из БД.

Билет № 20

Вопрос 1: Что такое ORM?

Вопрос 2: Какие есть способы подключения ORM к Python?

Практическое задание: Создайте две модели, свяжите их и выведите информацию из них.

Билет № 21

Вопрос 1: Что такое flask?

Вопрос 2: Как создать минимальный проект на flask?

Практическое задание: Создайте эндпоинт который будет получать список пользователей из БД и возвращать json.

Билет № 22

Вопрос 1: Как деплоить приложение?

Вопрос 2: Что такое uwsgi?

Практическое задание: Разверните flask приложение, используйте nginx.

Билет № 23

Вопрос 1: Что такое docker?

Вопрос 2: Для чего нужен Dockerfile?

Практическое задание: Запакуйте flask приложение в docker контейнер.

Билет № 24

Вопрос 1: Как настроить логирование?

Вопрос 2: Что такое переменные окружения?

Практическое задание: Реализуйте систему логирования в существующем приложении.

Билет № 25

Вопрос 1: Что такое тестирование кода?

Вопрос 2: Какие виды тестирования кода существуют?

Практическое задание: Создайте unit test для разработанного flask приложения.

5.3 Экзаменационные билеты по дисциплине МДК 03.03 Разработка информационных ресурсов с использованием программных платформ

Билеты 1-10 (Основы фреймворков и платформ)

Билет 1

1. Что такое веб-фреймворк? В чем его преимущества?
2. Назначение и структура программной платформы. **Практика:** Установите и настройте среду для работы с веб-фреймворком (Flask/Django/Laravel).

Билет 2

1. Сравнение серверных фреймворков Flask, Django и Express.js.
2. Основные компоненты программной платформы (сервер, клиент, база данных).

   Практика: Создайте минимальное веб-приложение на выбранном фреймворке.

Билет 3

1. Что такое MVC (Model-View-Controller) и как он применяется во фреймворках?
2. Какие бывают шаблонизаторы? Примеры (Jinja2, Pug, Twig).
   Практика: Подключите шаблонизатор в веб-приложении и отобразите простую HTML-страницу.

Билет 4

1. Разница между библиотекой и фреймворком. Примеры.
2. Основные параметры маршрутизации в веб-фреймворках.
   Практика: Реализуйте обработку двух разных маршрутов в веб-приложении.

Билет 5

1. Назначение шаблонизаторов и их возможности.
2. Как работает маршрутизация в Flask/Django?
   Практика: Создайте маршрут для отображения страницы с динамическими параметрами в URL.

Билет 6

1. Что такое middleware и зачем оно используется?
2. Основные HTTP-методы и их применение.

   Практика: Настройте middleware для логирования запросов в вашем приложении.

Билет 7

1. Отличия серверного и клиентского рендеринга.
2. Как работает ORM? Примеры популярных ORM.
   Практика: Настройте подключение к базе данных с использованием ORM.

Билет 8

1. Основные задачи backend-разработчика при работе с веб-фреймворками.
2. Какие бывают схемы организации базы данных (однотабличная, реляционная, NoSQL)?
   Практика: Создайте таблицу пользователей в базе данных и выполните запись данных.

Билет 9

1. Различия между реляционными и NoSQL базами данных.
2. Как связаны модели данных и ORM?
   Практика: Реализуйте создание и чтение данных через ORM.

Билет 10

1. Что такое миграции базы данных и зачем они нужны?
2. Основные типы связей между таблицами.

   Практика: Создайте миграцию для добавления новой колонки в таблицу пользователей.

Билеты 11-20 (Работа с данными, API, маршрутизация)

Билет 11

1. Как организуется работа с формами во фреймворках?
2. Как осуществляется обработка GET и POST-запросов? **Практика:** Создайте простую HTML-форму и обработайте её данные на сервере.

Билет 12

1. Как работает процесс валидации данных в веб-приложении?
2. Что такое CSRF и как защитить веб-приложение от этой атаки? **Практика:** Реализуйте серверную валидацию данных при регистрации пользователя.

Билет 13

1. Основные методы работы с базой данных (CRUD).
2. Что такое REST API и его основные принципы? **Практика:** Реализуйте маршрут API для получения списка пользователей в формате JSON.

Билет 14

1. Разница между REST API и GraphQL.
2. Какие существуют методы аутентификации пользователей? **Практика:** Реализуйте API-метод для создания новой записи в базе данных.

Билет 15

1. Что такое статические и динамические маршруты?
2. Как передавать параметры в URL? **Практика:** Реализуйте маршрут, принимающий параметр из URL и выводящий его.

Билет 16

1. Основные уровни безопасности веб-приложения.
2. Что такое SQL-инъекция и как от нее защититься? **Практика:** Реализуйте безопасный SQL-запрос с параметризацией.

Билет 17

1. Что такое токен-авторизация (JWT)?
2. Различия между сессиями и cookies. **Практика:** Настройте механизм авторизации с JWT.

Билет 18

1. Основные механизмы аутентификации пользователей.
2. Как работает шифрование паролей?
   Практика: Реализуйте хеширование паролей при регистрации пользователя.

Билет 19

1. Как защитить API от несанкционированного доступа?
2. Разница между HTTP и HTTPS.
   Практика: Реализуйте middleware для проверки авторизации пользователей.

Билет 20

1. Основные угрозы веб-приложениям (XSS, CSRF, MITM).
2. Как использовать CORS в веб-приложениях?
   Практика: Настройте CORS для разрешения запросов с другого домена.

Билеты 21-30 (Практика и безопасность)

Билет 21

1. Основные методы хранения пользовательских данных.
2. Как реализовать сброс пароля в веб-приложении?
   Практика: Реализуйте форму восстановления пароля.

Билет 22

1. Как защитить API-ключи в веб-приложении?
2. Что такое капча и зачем она нужна?
   Практика: Добавьте Google reCAPTCHA в форму регистрации.

Билет 23

1. Как работает двухфакторная аутентификация?
2. Различие между HTTP и WebSockets.
   Практика: Реализуйте проверку кода из SMS/почты при входе в систему.

Билет 24

1. Как работают асинхронные запросы?
2. Какие инструменты мониторинга ошибок существуют?
   Практика: Реализуйте обработку ошибок API и логирование запросов.

Билет 25

1. Что такое DevOps и зачем он нужен веб-разработчикам?
2. Как использовать контейнеризацию в веб-разработке?
   Практика: Запустите веб-приложение во временном контейнере Docker.

Билет 26

1. Принципы безопасного хранения данных пользователей.
2. Что такое OAuth и как он работает?
   Практика: Реализуйте вход через Google OAuth.

Билет 27

1. Основные стратегии кеширования в веб-приложениях.

2. Что такое API rate limiting?
   Практика: Реализуйте ограничение количества запросов к API.

Билет 28

1. Что такое CDN и зачем он нужен?
2. Как настроить защиту от ботов?
   Практика: Настройте защиту API с использованием Cloudflare.

Билет 29

1. Основные этапы тестирования веб-приложений.
2. Различия между функциональным и нагрузочным тестированием.
   Практика: Проведите тестирование API с помощью Postman.

Билет 30

1. Основные тренды в веб-разработке.
2. Как автоматизировать CI/CD для веб-приложений?
   Практика: Настройте автоматический деплой веб-приложения.

Приложение 5

Макет программы государственной итоговой аттестации

Государственное бюджетное профессиональное образовательное учреждение Республики Хакасия

Техникум коммунального хозяйства и сервиса

УТВЕРЖДЕНА

Приказ № _____

От «____»_____20 _____г.

Рассмотрена:

на заседании педагогического

совета техникума

Протокол № ____от _____2019 г.

Программа согласована:

председатель государственной экзаменационной комиссии:_____

_____

_____

_____

(ФИО, подпись)

ПРОГРАММА
ГОСУДАРСТВЕННОЙ ИТОГОВОЙ АТТЕСТАЦИИ

для подготовки _____по профессии/специальности (выбрать)

_____ _____

Код наименование

Абакан, 20____

Содержание

1. Паспорт программы

1.1. Наименование специальности/профессии:_____
«_____».

Квалификация (и): _____;

Срок получения образования по образовательной программе - _____.

1.2. Форма ГИА _____

1.3. Уровень демонстрационного экзамена _____

1.4. Программа ГИА разработана на основе:

– Федерального государственного образовательного стандарта среднего профессионального образования _____;

– Порядка проведения государственной итоговой аттестации по образовательным программам среднего профессионального образования, утв. Приказом Министерства просвещения РФ №800 от 8.11.2021г;

– Оценочных материалов для демонстрационного экзамена в 20____году;

1.3. Цель государственной итоговой аттестации.

Государственная итоговая аттестация проводится с целью определения соответствия результатов освоения студентами образовательной программы требованиям федерального государственного образовательного стандарта среднего профессионального образования по специальности/профессии _____ «_____».

Задачи: (выбрать)

демонстрационного экзамена – определение уровня освоения выпускниками материала, предусмотренного образовательной программой, и степени сформированности профессиональных умений и навыков путем проведения независимой экспертной оценки выполнения выпускником практических заданий в условиях реальных или смоделированных производственных процессов.

Дипломного проекта (работы) – систематизация и закрепление знаний выпускника по специальности, а также определения уровня готовности выпускника к самостоятельной деятельности.

1.4. Требования к результатам освоения основной профессиональной образовательной программы

Выпускник, получивший квалификации: _____

_____

должен быть подготовлен к выполнению следующих основных видов деятельности:

1. ВД 01. _____
2.
3. ВД 0…. _____

Требования ФГОС по профессии в соответствии с квалификациями	**Формы проверки освоения результатов**			
	Промежуточная аттестация			ГИА
	Дифференциро в анный зачет по учебным практикам	Дифференциро в анный зачет по производственн	Экзамен по модулю (квалификацио	

		ом практикам	н ный)	

ПК 1.1.				
ПК 1.2.				
ПК 1.3.				
ПК 1.4.				
….				

2. Требования к дипломным проектам (работам)

3. Методика оценивания дипломного проекта (работы)

4. Задание на демонстрационный экзамен

описать

Приложение 1 комплект оценочной документации с критериями оценки

5. Условия проведения

5.1. План проведения ДЭ:

- дата и время начала проведения ДЭ,
- расписание экзаменов в составе экзаменационных групп
- планируемая продолжительность времени
- технические перерывы