

Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)

Красноярский филиал Финуниверситета

УТВЕРЖДАЮ

Заместитель директора
по учебно – методической работе
Красноярского филиала
Финуниверситета

В.С. О.С. Вергейчик
« 02 » апреля 2026 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по профессиональному модулю

ПМ. 03 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ НА СТОРОНЕ СЕРВЕРА

(код, наименование)

09.02.09 Веб-разработка

(код, наименование специальности)

г. Красноярск – 2026

Фонд оценочных средств по профессиональному модулю ПМ. 03 Разработка веб-приложения на стороне сервера разработан на основании федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.09 Веб-разработка

Составители:

Цирулькевич Алена Викторовна, преподаватель

Фонд оценочных средств по дисциплине рассмотрен и рекомендован к утверждению на заседании предметной (цикловой) комиссии общепрофессиональных дисциплин.

Протокол от «02» апреля 2026 г. № 2

Председатель предметной (цикловой)
комиссии


(подпись)

О.А. Полтавец
(инициалы, фамилия)

1. Паспорт фонда оценочных средств по профессиональному модулю ПМ. 03 Разработка веб-приложения на стороне сервера

Результаты обучения (знания, умения)	Общие и профессиональные компетенции	Наименование элементов профессионального модуля, раздела, темы	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
<p>Иметь практический опыт:</p> <ul style="list-style-type: none"> - администрирования сред и платформ разработки. <p>Уметь:</p> <ul style="list-style-type: none"> - использовать серверную инфраструктуру и средства виртуализации; - применять выбранные языки программирования; - использовать выбранную среду программирования и СУБД. <p>Знать:</p> <ul style="list-style-type: none"> - отраслевую нормативную техническую документацию; - особенности выбранной среды программирования; - сетевые протоколы и основы web-технологий. 	ОК 01 – 09 ПК 3.1	<p>МДК 03.01</p> <p>Администрирование сред и платформ разработки информационных ресурсов</p> <p>Раздел 1.</p> <p>Администрирование сред и платформ разработки</p> <p>- Тема 1.1</p> <p>Администрирование сред и платформ разработки информационных ресурсов</p>	<p>- Защита практических работ №1-4 по теме 1.1.</p> <p>- Выполнение и защита индивидуальных заданий (СР).</p> <p>- Экспертное наблюдение за выполнением заданий.</p>	Дифференцированный зачет по МДК 03.01
<p>Иметь практический опыт:</p> <ul style="list-style-type: none"> - разработки серверной части веб-приложений. <p>Уметь:</p> <ul style="list-style-type: none"> - применять выбранные языки программирования для написания программного кода; - использовать выбранную среду программирования и СУБД; - применять инструменты для тестирования 	ОК 01 – 09 ПК 3.2	<p>МДК 03.02</p> <p>Разработка кода информационных ресурсов</p> <p>Раздел 2. Разработка кода информационных ресурсов</p> <p>- Тема 2.1 Виды и назначения серверных языков программирования, среды разработки</p> <p>- Тема 2.2.</p> <p>Программирование на стороне сервера</p>	<p>- Защита практических работ №1-5 по темам 2.1, 2.2.</p> <p>- Выполнение и защита индивидуальных заданий (СР).</p> <p>- Экспертное наблюдение за выполнением заданий.</p> <p>- Защита курсового проекта.</p>	Экзамен по МДК 03.02

<p>программных модулей.</p> <p>Знать:</p> <ul style="list-style-type: none"> - синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки; - методы повышения читаемости программного кода; - шаблоны проектирования. 				
<p>Иметь практический опыт:</p> <ul style="list-style-type: none"> - разработки веб-приложений с использованием программных платформ. <p>Уметь:</p> <ul style="list-style-type: none"> - использовать возможности имеющейся программной архитектуры информационного ресурса; - использовать выбранную среду программирования и СУБД; - применять инструменты для тестирования программных модулей. <p>Знать:</p> <ul style="list-style-type: none"> - назначение и структуру программных платформ; - методы обеспечения безопасности; - особенности выбранной среды программирования. 	<p>ОК 01 – 09</p> <p>ПК 3.3</p>	<p>МДК 03.03</p> <p>Разработка информационных ресурсов с использованием программных платформ</p> <p>Раздел 3. Разработка информационных ресурсов с использованием программных платформ</p> <p>- Тема 3.1. Разработка информационных ресурсов с использованием фреймворков и библиотек</p>	<ul style="list-style-type: none"> - Защита практических работ №1-5 по теме 3.1. - Выполнение и защита индивидуальных заданий (СР). - Экспертное наблюдение за выполнением заданий. 	<p>Экзамен по МДК 03.03</p>
<p>Иметь практический опыт:</p> <ul style="list-style-type: none"> - настройки инфраструктуры; - разработки серверной части информационных ресурсов; 	<p>ОК 01 – 09</p> <p>ПК 3.1 – ПК 3.3</p>	<p>Учебная практика</p> <p>Производственная практика (по профилю специальности)</p>	<ul style="list-style-type: none"> - Защита отчетов по практике. - Экспертное наблюдение за выполнением работ. 	<p>Дифференцированный зачет</p> <p>Дифференцированный зачет</p>

- разработки серверной части веб-приложения.				
--	--	--	--	--

2. Формы промежуточной аттестации по профессиональному модулю

Элементы профессионального модуля	Формы промежуточной аттестации
МДК 03.01 Администрирование сред и платформ разработки информационных ресурсов	Дифференцированный зачет
МДК 03.02 Разработка кода информационных ресурсов	Экзамен
МДК 03.03 Разработка информационных ресурсов с использованием программных платформ	Экзамен
Учебная практика	Дифференцированный зачет
Производственная практика (по профилю специальности)	Дифференцированный зачет
ПМ	Экзамен по модулю

3. Комплект оценочных средств

1. Для текущего контроля успеваемости

МДК.03.01 Администрирование сред и платформ разработки информационных ресурсов

Тема 1.1 Администрирование сред и платформ разработки информационных ресурсов

Практическая работа №1 «Настройка веб-сервера с помощью готового набора программного обеспечения»

– Цель: Получить навыки установки и настройки готовых сборок веб-сервера (OpenServer, XAMPP, WAMP) для локальной разработки.

– Задание:

1) Скачайте и установите один из готовых наборов ПО (например, OpenServer).

2) Изучите структуру и конфигурационные файлы.

3) Настройте несколько версий PHP для переключения между ними.

4) Создайте виртуальный хост для тестового сайта.

5) Подключите и настройте модуль Apache (например, mod_rewrite).

6) Составьте отчет с описанием выполненных шагов и скриншотами.

Практическая работа №2 «Настройка виртуальной машины для использования в качестве веб-сервера» (8 часов)

– Цель: Освоить процесс создания и настройки виртуальной машины под веб-сервер.

– Задание:

1) Установите гипервизор (VirtualBox или VMware).

2) Создайте новую виртуальную машину с ОС Ubuntu Server.

3) Настройте сетевые интерфейсы для доступа к виртуальной машине из локальной сети.

4) Установите и настройте веб-сервер (nginx или apache2).

5) Настройте статический IP-адрес для виртуальной машины.

6) Проверьте доступность веб-сервера с хостовой машины.

Практическая работа №3 «Установка и настройка программного обеспечения для управления сервером» (8 часов)

– Цель: Получить навыки установки и настройки панелей управления сервером (например, ISPConfig, VestaCP, aaPanel).

– Задание:

1) Установите на виртуальную машину панель управления сервером (например, aaPanel).

2) С помощью панели управления создайте новый сайт, базу данных, пользователя FTP.

3) Настройте SSL-сертификат для сайта через панель управления (Let's Encrypt).

4) Загрузите тестовый сайт и проверьте его работу.

5) Опишите функционал установленной панели в отчете.

Практическая работа №4 «Установка и настройка программного обеспечения для разработки, доставки и запуска контейнерных приложений» (8 часов)

– Цель: Ознакомиться с основами контейнеризации приложений с помощью Docker.

– Задание:

1) Установите Docker и Docker Compose на виртуальную машину.

2) Создайте файл docker-compose.yml для запуска связки: веб-сервер (nginx) + PHP + MySQL.

3) Запустите контейнеры и проверьте работоспособность.

4) Выполните базовые команды Docker: ps, stop, rm, exec.

5) Составьте инструкцию по развертыванию проекта с помощью Docker.

МДК.03.02 Разработка кода информационных ресурсов

Тема 2.1 Виды и назначения серверных языков программирования, среды разработки

Практическая работа №1 «Установка и настройка языка программирования» (6 часов)

– Цель: Установить и настроить среду для разработки на выбранном серверном языке (например, PHP, Python, Node.js).

– Задание:

1) Установите интерпретатор выбранного языка программирования.

2) Настройте переменные окружения (PATH).

3) Установите и настройте IDE (VS Code, PhpStorm, PyCharm) с плагинами для выбранного языка.

4) Настройте отладчик (Xdebug для PHP, debugger для Python/Node.js).

5) Напишите простейший скрипт "Hello World" и проверьте его выполнение.

Практическая работа №2 «Выполнение типовой практической задачи на основе процедурного подхода программирования» (6 часов)

– Цель: Реализовать алгоритм обработки данных с использованием процедурного программирования.

– Задание: Разработайте набор функций (процедур) для решения задачи:

a) Функция для вычисления факториала числа.

b) Функция для проверки, является ли число простым.

c) Функция для сортировки массива (пузырьковая сортировка).

d) Функция для обработки данных формы (валидация email, длины строки).

e) Протестируйте работу функций на тестовых данных.

Практическая работа №3 «Выполнение типовой практической задачи на основе объектно-ориентированного программирования» (6 часов)

– Цель: Реализовать задачу с использованием классов и объектов.

– Задание: Создайте класс User со свойствами: id, name, email, password (хэшированный). Реализуйте методы:

a) __construct() для инициализации.

b) save() для сохранения пользователя в базу данных (имитация).

c) find(\$id) для поиска пользователя по ID.

d) delete() для удаления пользователя.

e) Создайте несколько объектов и продемонстрируйте работу методов.

Практическая работа №4 «Создание программных модулей используя шаблоны проектирования» (8 часов)

– Цель: Применить шаблоны проектирования (Design Patterns) при разработке программных модулей.

– Задание: Реализуйте модуль для работы с базой данных, используя шаблон Singleton для подключения. Реализуйте класс Logger, используя шаблон Singleton для записи логов в файл. Реализуйте простой пример использования шаблона Factory Method для создания объектов разных типов (например, разные типы уведомлений: Email, SMS).

Тема 2.2. Программирование на стороне сервера

Практическая работа №1 «Программирование задач с использованием переменных» (6 часов)

– Цель: Закрепить навыки работы с переменными, типами данных, операциями ввода-вывода.

– Задание: Разработайте скрипт, который:

1) Запрашивает у пользователя имя и возраст.

2) Выводит приветствие: "Привет, [имя]! Тебе [возраст] лет."

3) Вычисляет, сколько лет исполнится пользователю через 5 и 10 лет.

4) Проверяет, является ли возраст совершеннолетним.

5) Использует разные типы данных (строки, целые, логические).

Практическая работа №2 «Программирование задач с использованием массивов» (6 часов)

– Цель: Освоить работу с индексированными и ассоциативными массивами.

– Задание: Создайте массив товаров (минимум 5), каждый товар — ассоциативный массив с полями: id, name, price, quantity. Напишите функции для:

- a) Вывода списка всех товаров в виде HTML-таблицы.
- b) Подсчета общей стоимости всех товаров на складе.
- c) Поиска товара по имени.
- d) Добавления нового товара в массив.

Практическая работа №3 «Программирование задач с использованием условных операторов» (6 часов)

– Цель: Закрепить навыки использования конструкций if-else, switch для управления ходом программы.

– Задание: Разработайте скрипт-калькулятор, который принимает два числа и операцию (+, -, *, /) через параметры запроса (GET) и выводит результат. Обработайте ошибки: деление на ноль, ввод нечисловых значений, недопустимая операция. Используйте switch для выбора операции.

Практическая работа №4 «Создание пользовательских функций» (4 часа)

– Цель: Научиться создавать и использовать пользовательские функции для структурирования кода.

– Задание: Создайте библиотеку функций для работы с текстом:

- a) reverseString(\$str) — переворот строки.
- b) wordCount(\$str) — подсчет слов.
- c) camelCaseToSnakeCase(\$str) — преобразование camelCase в snake_case.
- d) truncateString(\$str, \$length) — обрезка строки до заданной длины с добавлением "...".
- e) Продемонстрируйте работу функций на примерах.

Практическая работа №5 «Запись и получение информации из базы данных» (4 часа)

– Цель: Освоить подключение к базе данных и выполнение запросов.

– Задание: Создайте базу данных testdb с таблицей users (id, name, email).

Напишите скрипт, который:

- 1) Подключается к базе данных с использованием PDO или mysqli.
- 2) Выполняет запрос SELECT и выводит всех пользователей в HTML-таблице.
- 3) Реализует форму для добавления нового пользователя (INSERT).
- 4) Реализует удаление пользователя по ID (DELETE).

МДК.03.03 Разработка информационных ресурсов с использованием программных платформ

Тема 3.1. Разработка информационных ресурсов с использованием фреймворков и библиотек

Практическая работа №1 «Работа с формами» (16 часов)

– Цель: Научиться обрабатывать формы с использованием выбранного фреймворка (например, Laravel, Django, Express.js).

– Задание:

- 1) Создайте новый проект в выбранном фреймворке.
- 2) Создайте маршруты для отображения формы и обработки POST-запроса.
- 3) Создайте форму обратной связи (поля: имя, email, сообщение).
- 4) Реализуйте обработку данных формы на сервере: получение данных, базовая валидация.

5) Отправьте пользователю подтверждение о получении сообщения.

Практическая работа №2 «Реализация регистрации и авторизации» (14 часов)

– Цель: Реализовать систему аутентификации пользователей с использованием встроенных механизмов фреймворка.

– Задание:

- 1) Создайте модель пользователя (User) и миграции для таблицы users.
- 2) Реализуйте страницу регистрации: хэширование пароля, сохранение в БД.
- 3) Реализуйте страницу входа (логин): проверка email и пароля, создание сессии.
- 4) Создайте защищенную страницу (личный кабинет), доступную только авторизованным пользователям.

5) Реализуйте выход из системы (logout).

Практическая работа №3 «Создание базовых функций работы с данными. Создание, чтение, модификация, удаление» (12 часов)

– Цель: Реализовать CRUD-операции (Create, Read, Update, Delete) для одной сущности.

– Задание: Для сущности "Товар" (поля: название, описание, цена, количество) реализуйте:

- 1) Страницу со списком всех товаров (Read).
- 2) Страницу добавления нового товара с формой (Create).
- 3) Страницу редактирования товара с предзаполненной формой (Update).
- 4) Возможность удаления товара (Delete) с подтверждением.
- 5) Используйте шаблонизатор для отображения страниц.

Практическая работа №4 «Валидация данных» (12 часов)

– Цель: Научиться применять правила валидации для данных, поступающих от пользователя.

– Задание:

- 1) К форме добавления/редактирования товара добавьте валидацию:
 - Название — обязательно, минимум 3 символа.
 - Цена — обязательно, числовое значение, больше 0.
 - Количество — целое число, неотрицательное.
- 2) При ошибках валидации форма должна отображаться снова с подсветкой полей, содержащих ошибки, и выводом сообщений об ошибках.
- 3) Сохраняйте введенные пользователем данные в полях формы при повторном отображении (old input).

Практическая работа №5 «Разработка API» (12 часов)

– Цель: Создать простое REST API для управления сущностями.

– Задание:

1) Создайте контроллер для API.

2) Реализуйте эндпоинты (endpoints):

– GET /api/items — получение списка всех товаров в формате JSON.

– GET /api/items/{id} — получение одного товара по ID.

– POST /api/items — добавление нового товара (данные в body

запроса в формате JSON).

– PUT /api/items/{id} — обновление товара.

– DELETE /api/items/{id} — удаление товара.

3) Протестируйте API с помощью инструментов (Postman, curl).

2. Вопросы и для промежуточной аттестации

Вопросы к дифференцированному зачету по МДК 03.01

1. Что такое виртуализация? Какие типы виртуализации существуют?

2. Охарактеризуйте гипервизоры 1-го и 2-го типа. Приведите примеры.

3. Какие серверные операционные системы используются для веб-серверов?

Сравните Windows Server и Linux.

4. Что такое веб-сервер? Опишите принцип работы. Сравните Apache и Nginx.

5. Какие протоколы передачи данных используются в веб-разработке?

Охарактеризуйте HTTP/HTTPS, FTP, SSH.

6. Что такое DNS? Как происходит разрешение доменных имен?

7. Какие существуют виды хостинга (виртуальный, VPS, выделенный сервер, облачный)? Их преимущества и недостатки.

8. Что такое контейнеризация? В чем отличие контейнеров от виртуальных машин?

9. Что такое Docker? Основные понятия: образ, контейнер, Dockerfile, Docker Compose.

10. Какие инструменты используются для управления серверами (панели управления, системы мониторинга)?

11. Как настроить виртуальный хост в Apache/Nginx?

12. Что такое SSL-сертификат? Как его установить на веб-сервер?

13. Какие существуют способы обеспечения безопасности веб-сервера?

14. Что такое нагрузочное тестирование сервера? Какие инструменты используются?

15. Опишите процесс развертывания веб-приложения на удаленном сервере.

16. Что такое CI/CD? Основные принципы.

17. Какие существуют стратегии резервного копирования серверных данных?

18. Что такое файловая система Linux? Основные каталоги и их назначение.

19. Основные команды для администрирования Linux (работа с файлами, процессами, пользователями).

20. Как настроить сетевые параметры в Linux (интерфейсы, маршрутизация)?
Вопросы к дифференцированному зачету по МДК 03.02

1. В чем разница между интерпретируемыми и компилируемыми языками программирования?

2. Какие серверные языки программирования вы знаете? Охарактеризуйте их (PHP, Python, Node.js, Java, C#).

3. Что такое процедурное программирование? Основные принципы.

4. Что такое объектно-ориентированное программирование (ООП)? Назовите основные принципы ООП (инкапсуляция, наследование, полиморфизм).

5. Что такое класс и объект? Приведите пример.

6. Что такое конструктор и деструктор класса?

7. Какие модификаторы доступа существуют в ООП (public, private, protected)?

8. Что такое абстрактные классы и интерфейсы? В чем их отличие?

9. Что такое рефакторинг кода? Цели и методы рефакторинга.

10. Что такое шаблоны проектирования (Design Patterns)? Какие группы шаблонов существуют?

11. Опишите шаблон Singleton. Для чего он используется?

12. Опишите шаблон Factory Method. Приведите пример использования.

13. Какие типы данных существуют в выбранном языке программирования?

14. Как работать с массивами (индексированными и ассоциативными)?

15. Какие условные операторы используются для ветвления?

16. Какие виды циклов существуют? (for, while, foreach)

17. Что такое функции? Как объявить и использовать пользовательскую функцию?

18. Что такое область видимости переменных?

19. Как работать с файловой системой на серверном языке (чтение, запись)?

20. Как выполнить подключение к базе данных из кода?

Вопросы к дифференцированному зачету по МДК 03.03

1. Что такое веб-фреймворк? Для чего он нужен?

2. Какие существуют типы веб-фреймворков (микрофреймворки, полноценные)? Приведите примеры.

3. Что такое архитектура MVC (Model-View-Controller)? Опишите назначение каждого компонента.

4. Что такое маршрутизация (routing) в веб-фреймворках?

5. Как работают middleware (посредники) в веб-фреймворках?

6. Что такое ORM (Object-Relational Mapping)? Преимущества использования.

7. Что такое миграции базы данных? Для чего они нужны?

8. Как работать с формами во фреймворке? Защита от CSRF.

9. Как реализуется валидация данных во фреймворке?

10. Что такое аутентификация и авторизация? Как они реализуются во фреймворке?

11. Что такое шаблонизаторы? Какие шаблонизаторы вы знаете (Blade, Twig, Jinja)?

12. Что такое REST API? Принципы построения.
13. Какие HTTP-методы используются в REST API?
14. Что такое сессии и куки? Как с ними работать во фреймворке?
15. Какие методы обеспечения безопасности веб-приложений реализованы во фреймворках (защита от XSS, SQL-инъекций, CSRF)?
16. Как выполняется отладка приложений во фреймворке?
17. Что такое автозагрузка классов (autoloading)?
18. Как организована работа с зависимостями (Composer, npm, pip)?
19. Что такое тестирование веб-приложений? Какие виды тестов можно писать во фреймворке?
20. Как выполняется развертывание (деплой) приложения, созданного на фреймворке?

3. К экзамену по модулю

Теоретические вопросы:

1. Что такое виртуализация? Какие типы виртуализации существуют (аппаратная, программная, контейнерная)?
2. Охарактеризуйте гипервизоры 1-го и 2-го типа. Приведите примеры.
3. Какие серверные операционные системы используются для веб-серверов? Сравните Windows Server и Linux.
4. Что такое веб-сервер? Опишите принцип работы. Сравните Apache и Nginx.
5. Какие протоколы передачи данных используются в веб-разработке? Охарактеризуйте HTTP/HTTPS, FTP, SSH.
6. Что такое DNS? Как происходит разрешение доменных имен?
7. Какие существуют виды хостинга (виртуальный, VPS, выделенный сервер, облачный)? Их преимущества и недостатки.
8. Что такое контейнеризация? В чем отличие контейнеров от виртуальных машин?
9. Что такое Docker? Основные понятия: образ, контейнер, Dockerfile, Docker Compose.
10. Какие инструменты используются для управления серверами (панели управления, системы мониторинга)?
11. Как настроить виртуальный хост в Apache/Nginx?
12. Что такое SSL-сертификат? Как его установить на веб-сервер?
13. Какие существуют способы обеспечения безопасности веб-сервера?
14. Что такое нагрузочное тестирование сервера? Какие инструменты используются?
15. Опишите процесс развертывания веб-приложения на удаленном сервере.
16. Что такое CI/CD? Основные принципы.
17. Какие существуют стратегии резервного копирования серверных данных?
18. Что такое файловая система Linux? Основные каталоги и их назначение.

19. Основные команды для администрирования Linux (работа с файлами, процессами, пользователями).
20. Как настроить сетевые параметры в Linux (интерфейсы, маршрутизация)?
21. В чем разница между интерпретируемыми и компилируемыми языками программирования?
22. Какие серверные языки программирования вы знаете? Охарактеризуйте их (PHP, Python, Node.js, Java, C#).
23. Что такое процедурное программирование? Основные принципы.
24. Что такое объектно-ориентированное программирование (ООП)? Назовите основные принципы ООП (инкапсуляция, наследование, полиморфизм).
25. Что такое класс и объект? Приведите пример.
26. Что такое конструктор и деструктор класса?
27. Какие модификаторы доступа существуют в ООП (public, private, protected)?
28. Что такое абстрактные классы и интерфейсы? В чем их отличие?
29. Что такое статические свойства и методы класса? Для чего они используются?
30. Что такое рефакторинг кода? Цели и методы рефакторинга.
31. Что такое шаблоны проектирования (Design Patterns)? Какие группы шаблонов существуют?
32. Опишите шаблон Singleton. Для чего он используется?
33. Опишите шаблон Factory Method. Приведите пример использования.
34. Опишите шаблон MVC (Model-View-Controller). Назначение каждого компонента.
35. Какие типы данных существуют в выбранном языке программирования?
36. Как работать с массивами (индексированными и ассоциативными)?
37. Какие условные операторы используются для ветвления?
38. Какие виды циклов существуют? (for, while, foreach)
39. Что такое функции? Как объявить и использовать пользовательскую функцию?
40. Что такое область видимости переменных?
41. Что такое рекурсия? Приведите пример.
42. Как работать с файловой системой на серверном языке (чтение, запись)?
43. Как выполнить подключение к базе данных из кода? (PDO, mysqli)
44. Что такое SQL-инъекции? Как защититься от них?
45. Что такое prepared statements? Для чего они нужны?
1. Что такое веб-фреймворк? Для чего он нужен? Какие существуют типы веб-фреймворков (микрофреймворки, полноценные)? Приведите примеры.
2. Что такое архитектура MVC (Model-View-Controller)? Опишите назначение каждого компонента.
3. Что такое маршрутизация (routing) в веб-фреймворках? Как она работает?
4. Как работают middleware (посредники) в веб-фреймворках? Приведите примеры использования.
5. Что такое ORM (Object-Relational Mapping)? Преимущества использования. Примеры ORM для разных языков.

6. Что такое миграции базы данных? Для чего они нужны?
7. Как работать с формами во фреймворке? Защита от CSRF.
8. Как реализуется валидация данных во фреймворке?
9. Что такое аутентификация и авторизация? Как они реализуются во фреймворке?
10. Что такое сессии и куки? Как с ними работать во фреймворке?
11. Что такое шаблонизаторы? Какие шаблонизаторы вы знаете (Blade, Twig, Jinja)? Их преимущества.
12. Что такое REST API? Принципы построения. Какие HTTP-методы используются в REST API?
13. Что такое CRUD? Как он соотносится с HTTP-методами?
14. Что такое JWT (JSON Web Token)? Для чего используется?
15. Какие методы обеспечения безопасности веб-приложений реализованы во фреймворках (защита от XSS, SQL-инъекций, CSRF)?
16. Что такое валидация на стороне сервера? Чем она отличается от клиентской валидации?
17. Как выполняется отладка приложений во фреймворке? Какие инструменты используются?
18. Что такое автозагрузка классов (autoloading)? Как она работает в современных фреймворках?
19. Что такое Composer? Для чего он используется? Файл composer.json.
20. Что такое тестирование веб-приложений? Какие виды тестов можно писать во фреймворке (unit, feature)?

Практико-ориентированные задания:

1. Установите на виртуальную машину (VirtualBox/VMware) операционную систему Ubuntu Server. Настройте статический IP-адрес, установите веб-сервер Nginx. Проверьте доступность сервера с хостовой машины, создав тестовую HTML-страницу.
2. Настройте веб-сервер Apache для работы двух сайтов (виртуальных хостов) на одном сервере: site1.local и site2.local. Для каждого сайта создайте свою HTML-страницу-заглушку. Настройте файл hosts на клиентской машине для доступа к этим сайтам.
3. Установите Docker на виртуальную машину. Создайте Dockerfile для простого PHP-приложения. Запустите контейнер и пробросьте порт 8080 на хост-машину.
4. Установите панель управления сервером (например, aaPanel или VestaCP). С помощью панели создайте новый сайт, базу данных MySQL и пользователя FTP. Загрузите тестовый сайт через FTP и проверьте его работу.
5. Настройте автоматическое получение и установку SSL-сертификата Let's Encrypt для одного из сайтов с помощью Certbot. Проверьте работу сайта по HTTPS и настройте автоматический редирект с HTTP на HTTPS.
6. Напишите скрипт на выбранном языке программирования (PHP/Python/Node.js), который:

- Принимает из GET-параметра строку текста.
- Подсчитывает количество символов, количество слов и количество гласных букв в этой строке.
- Возвращает результат в формате JSON.

7. Создайте класс Product со свойствами: id, name, price, quantity. Реализуйте методы:

__construct(\$name, \$price, \$quantity)

- getTotalCost() — возвращает общую стоимость (price * quantity).
 - __toString() — возвращает строковое представление товара.
- Создайте массив из 3-5 объектов и выведите информацию о каждом товаре.

8. Напишите скрипт, который подключается к базе данных MySQL, создает таблицу users (id, name, email, created_at), и заполняет её тестовыми данными (5 записей). Затем выведите всех пользователей в виде HTML-таблицы, отсортированных по дате регистрации.

9. Реализуйте простую форму обратной связи (поля: имя, email, сообщение). При отправке формы:

- Выполните валидацию: имя не пустое, email корректный, сообщение не менее 10 символов.
- Если есть ошибки, выведите их под соответствующими полями и сохраните введенные данные.
- Если ошибок нет, сохраните сообщение в файл messages.txt и выведите пользователю сообщение об успехе.

10. Напишите функцию для работы с файловой системой, которая:

- Принимает путь к директории.
- Рекурсивно обходит все файлы и поддиректории.
- Возвращает массив, где ключ — путь к файлу, значение — размер файла в байтах.
- Выведите список всех файлов размером более 1 МБ.

11. Создайте новый проект в выбранном фреймворке (Laravel, Django, Express.js). Создайте маршруты:

- / — главная страница с приветствием.
- /about — страница "О нас".
- /contacts — страница с контактной информацией.

Используйте шаблонизатор для отображения страниц.

12. Создайте модель Category (категория товаров) с полями: name, slug, description. Создайте миграцию для этой таблицы. Создайте несколько категорий через сиды (seeders) или вручную в базе данных.

13. Создайте модель Product со связью с категорией (один ко многим). Реализуйте:

- Контроллер для вывода списка всех товаров.
- Контроллер для вывода товаров конкретной категории (по slug).
- В представлении выведите товары в виде карточек с названием, ценой и названием категории.

14. Реализуйте систему аутентификации пользователей (регистрация, вход, выход) с использованием встроенных средств фреймворка. Создайте страницу "Личный кабинет", доступную только авторизованным пользователям.
15. Добавьте в проект возможность создания заказов:
16. Форма заказа (выбор товара из списка, указание количества, контактные данные).
17. Валидация формы на сервере.
18. Сохранение заказа в базу данных.
19. Отправка уведомления администратору на email (имитация или реальная отправка).
20. Создайте REST API для управления товарами:
 - GET /api/products — получение списка товаров в JSON.
 - GET /api/products/{id} — получение одного товара.
 - POST /api/products — добавление нового товара (аутентификация обязательна).
 - PUT /api/products/{id} — обновление товара.
 - DELETE /api/products/{id} — удаление товара.
21. Протестируйте API с помощью Postman.
22. Реализуйте поиск товаров по названию или описанию. Добавьте форму поиска на страницу каталога. Результаты поиска должны выводиться с подсветкой найденных совпадений.
23. Добавьте в проект пагинацию (постраничный вывод) для списка товаров (по 10 товаров на странице). Выведите ссылки на страницы.
24. Создайте middleware, который логирует (записывает в файл) все входящие запросы к API: метод, URL, IP-адрес, время запроса. Добавьте этот middleware ко всем маршрутам API.
25. Настройте загрузку изображений для товаров. Добавьте поле image в модель Product. Реализуйте возможность загрузки изображения при создании/редактировании товара. Изображения должны сохраняться на сервер, а в базе храниться путь к файлу.