МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ) ВОЛЖСКИЙ ФИЛИАЛ

УТВЕРЖДАЮ
Зав. кафедрой ИиТТП
Петрова А.В.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЫ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ РАЗРАБОТКИ ИНТЕРНЕТ-ПРИЛОЖЕНИЙ»

Направление подготовки
23.03.01 «Информатика и вычислительная техника»

Направленность (профиль)
Автоматизированные системы обработки информации и управления в отраслях транспортно-дорожного комплекса

Квалификация
Бакалавр

Кафедра: ИиТТП

Чебоксары

ОГЛАВЛЕНИЕ

РГР №1. Основы HTML и CSS. Создание статического веб-сайта

Цель работы: Освоить основы языка гипертекстовой разметки HTML и каскадных таблиц стилей CSS, научиться создавать статические веб-страницы с адаптивным дизайном.

Теоретическая часть:

1. Структура HTML-документа. Основные теги и атрибуты.
2. Блочная и строчная верстка. Семантические элементы HTML5.
3. Каскадные таблицы стилей CSS. Селекторы, свойства, приоритеты.
4. Flexbox и Grid для создания адаптивных макетов.
5. Медиа-запросы. Адаптивный дизайн.
6. Формы и элементы ввода.

Типовые задания:

1. **Разработка структуры сайта.** Выбрать тематику сайта согласно варианту. Разработать набросок структуры страниц (эскиз), включающий:
   - Шапку сайта (header) с логотипом и названием.
   - Навигационное меню.
   - Основную область контента.
   - Боковую панель (сайдбар).
   - Подвал сайта (footer).
2. **Верстка главной страницы.** Создать HTML-документ, реализующий разработанную структуру с использованием семантических тегов HTML5. Главная страница должна содержать:
   - Приветственный текст и информацию о сайте.
   - Изображения, соответствующие тематике.
   - Список (нумерованный или маркированный).
   - Таблицу с данными.
3. **Оформление с помощью CSS.** Разработать CSS-стили для:
   - Оформления текста (шрифты, цвета, интервалы).
   - Позиционирования блоков с использованием Flexbox или Grid.

– Оформления навигационного меню (горизонтальное или вертикальное).

– Стилизации таблицы и списков

4. **Создание страницы с формой.** Разработать отдельну.ю страницу, содержащую форму со следующими полями (согласно тематике):

– Поле ввода текста (имя, название).

– Поле ввода email.

– Выпадающий список (select).

– Радиокнопки или флажки (checkbox).

– Текстовая область (textarea).

– Кнопка отправки.

5. **Адаптивный дизайн.** Реализовать адаптивность сайта с помощью медиа-запросов:

– Для мобильных устройств (ширина экрана до 768px) — вертикальное расположение блоков.

– Для планшетов (768px — 1024px) — двухколоночная верстка.

– Для десктопов (более 1024px) — трехколоночная верстка.

6. **Валидация и кроссбраузерность.** Проверить созданные страницы с помощью валидатора W3C, исправить ошибки. Убедиться в корректном отображении в различных браузерах.

Таблица 1.1 - 10 вариантов тематик сайтов:

Вар	**Тематика сайта**	**Цветовая схема**	**Дополнительные требования**
1	Портфолио веб-разработчика	Синий + серый	Галерея проектов (минимум 4 работы)
2	Интернет-магазин электроники	Черный + оранжевый	Карточки товаров (минимум 6 товаров)
3	Туристическое агентство	Бирюзовый + белый	Слайдер с популярными направлениями
4	Спортивный клуб	Красный + белый	Расписание тренировок в виде таблицы
5	Ресторан итальянской кухни	Зеленый + бежевый	Меню с ценами и описанием блюд
6	Фотографа-портретиста	Черно-белая	Портфолио с миниатюрами и лайтбоксом

7	Курсов иностранных языков	Желтый + синий	Таблица цен и расписания занятий
8	Архитектурное бюро	Серый + коричневый	Портфолио проектов с описанием
9	Книжный интернет-магазин	Бордовый + бежевый	Каталог книг с фильтрацией по жанрам
10	Салон красоты	Розовый + фиолетовый	Прейскурант услуг и прайс-лист

РГР №2. Клиентское программирование на JavaScript

Цель работы: Освоить основы языка JavaScript для создания интерактивных веб-страниц, научиться обрабатывать события и манипулировать DOM-деревом.

Теоретическая часть:

1. Основы синтаксиса JavaScript. Типы данных, операторы, функции.
2. Объектная модель документа (DOM). Доступ к элементам страницы.
3. Обработка событий. Основные типы событий.
4. Валидация форм на стороне клиента.
5. Асинхронное программирование. Promise, async/await.
6. Fetch API для HTTP-запросов.

Типовые задания:

1. **Интерактивная форма с валидацией.** На основе HTML-формы из РГР №1 реализовать:

– Валидацию всех полей формы с выводом сообщений об ошибках.

– Проверку корректности email (наличие @ и точки).

– Проверку длины введенных значений.

– Блокировку кнопки отправки до полного заполнения всех обязательных полей.

2. **Калькулятор.** Разработать калькулятор для выполнения арифметических операций согласно варианту:

– Поля ввода для двух чисел.

– Кнопки выбора операции (+, -, *, /).

– Кнопка "Рассчитать".

– Вывод результата на страницу без перезагрузки.

– Обработка ошибок (деление на ноль, некорректный ввод).

3. **Игра "Угадай число".** Реализовать игру со следующим поведением:

– При загрузке страницы генерируется случайное число от 1 до 100.

– Пользователь вводит свое предположение в поле ввода.

– При нажатии кнопки "Проверить" выводится подсказка: "больше", "меньше" или "угадал!".

– Счетчик попыток увеличивается.

– При угадывании выводится поздравление и количество попыток.

– Кнопка "Новая игра" для перезапуска.

4. **Динамическое изменение DOM.** Создать интерфейс для управления списком задач (Todo List):

– Поле ввода новой задачи.

– Кнопка "Добавить" для добавления задачи в список.

– Возможность отметить задачу как выполненную (перечеркивание).

– Кнопка "Удалить" для каждой задачи.

– Сохранение списка в localStorage с загрузкой при обновлении страницы.

5. **Асинхронный запрос к API.** Используя Fetch API, выполнить запрос к публичному API (согласно варианту):

– Получить данные в формате JSON.

– Отобразить полученные данные в виде таблицы или карточек.

– Добавить индикатор загрузки (спиннер) на время ожидания ответа.

– Обработать возможные ошибки сети.

Таблица 2.1 - 10 вариантов исходных данных:

Вар	**Тематика формы**	**API для запроса**	**Дополнительное задание**
1	Регистрация пользователя	JSONPlaceholder (users)	Отобразить список пользователей
2	Обратная связь	JSONPlaceholder (posts)	Отобразить последние 10 постов
3	Заказ товара	OpenWeatherMap (погода)	Показать погоду в указанном городе
4	Бронирование столика	The Cat API	Отобразить случайные изображения котов
5	Поиск авиабилетов	JSONPlaceholder (comments)	Отобразить комментарии к посту
6	Регистрация на мероприятие	JSONPlaceholder (todos)	Список задач с чекбоксами
7	Подписка на новости	JSONPlaceholder (photos)	Отобразить миниатюры фотографий

8	Калькулятор кредита	JSONPlaceholder (albums)	Отобразить список альбомов
9	Конвертер валют	JSONPlaceholder (todos)	Список задач с фильтрацией
10	Расчет калорий	JSONPlaceholder (users)	Отобразить карточки пользователей

РГР №3. Серверное программирование. Обработка форм и управление сессиями

Цель работы: Освоить основы серверного программирования, научиться обрабатывать данные форм, управлять пользовательскими сессиями и файлами cookie.

Теоретическая часть:

1. Протокол HTTP. Методы GET и POST.
2. Жизненный цикл сервлета/JSP-страницы.
3. Обработка параметров запроса.
4. Управление сессиями. Cookie и HttpSession.
5. Фильтры и обработка кодировки.
6. JSP и шаблонизаторы.

Типовые задания:

1. **Сервлет для обработки формы.** Создать веб-приложение, состоящее из:

– HTML-страницы с формой (из РГР №1 или №2).

– Сервлета (или JSP-страницы), обрабатывающего данные формы.

– Страницы с результатами обработки.

2. **Валидация на сервере.** Реализовать серверную валидацию данных:

– Проверка обязательных полей.

– Проверка формата email.

– Проверка длины строк.

– При ошибках валидации — возврат на форму с сохранением введенных данных.

3. **Управление сессией.** Реализовать функциональность в соответствии с вариантом:

– Вариант 1-5: Счетчик посещений страницы (сохранять в сессии количество просмотров и выводить его).

– Вариант 6-10: Запоминание данных пользователя (имя, предпочтения) с возможностью сброса.

4. **Голосование/опрос.** Разработать систему голосования со следующими функциями:

– Страница с вариантами для голосования.

– Сохранение голоса в сессии (запрет повторного голосования).

– Отображение результатов голосования (накопленных данных).

– Хранение результатов в ServletContext или файле.

5. **Игра "Орел и решка".** Реализовать игру с виртуальным кредитом:

– При первом посещении устанавливается начальный кредит (100 рублей).

– Пользователь выбирает ставку и вариант (орел/решка).

– Сервер генерирует случайное число и определяет выигрыш/проигрыш.

– Кредит обновляется и сохраняется в сессии.

– При отрицательном кредите игра прекращается.

6. **Cookie-менеджер.** Разработать страницу для управления файлами cookie:

– Форма для установки произвольной cookie (имя, значение, срок).

– Таблица со списком всех установленных cookie.

– Возможность удалить выбранную cookie.

Таблица 3.1 - 10 вариантов исходных данных:

Вар	**Тип формы**	**Начальный кредит**	**Тема голосования**	**Дополнительно**
1	Регистрация	100	Любимый язык программирования	Счетчик посещений
2	Авторизация	200	Лучший браузер	Сохранение темы оформления
3	Обратная связь	150	Оценка качества образования	История последних действий
4	Заказ товара	100	Выбор операционной системы	Корзина покупок
5	Комментарий	200	Любимый жанр кино	Недавно просмотренные товары

6	Поиск	100	Лучший мессенджер	Персонализированные рекомендации
7	Подписка	150	Выбор социальной сети	Настройки уведомлений
8	Отзыв	200	Любимый музыкальный жанр	Выбор языка интерфейса
9	Вопрос	100	Предпочтительный способ оплаты	Сохранение параметров поиска
10	Заявка	150	Оценка работы сервиса	Последние введенные данные

РГР №4. Взаимодействие с базами данных

Цель работы: Научиться интегрировать базы данных в веб-приложения, выполнять CRUD-операции и отображать данные на страницах.

Теоретическая часть:

1. Основы SQL. CRUD-операции (Create, Read, Update, Delete).
2. Подключение к базе данных из приложения (JDBC, PDO).
3. Выполнение запросов и обработка результатов.
4. Предотвращение SQL-инъекций (подготовленные запросы).
5. ORM-технологии (Hibernate, JPA).

Типовые задания:

1. **Проектирование базы данных.** Создать базу данных, состоящую минимум из двух связанных таблиц (согласно варианту):

– Определить структуру таблиц (поля, типы данных).

– Определить связи между таблицами (первичные и внешние ключи).

– Создать SQL-скрипт для создания таблиц и заполнения тестовыми данными.

2. **Подключение к БД.** Настроить подключение к базе данных из веб-приложения:

– Использовать пул соединений (connection pool).

– Реализовать класс-синглтон для управления подключениями.

– Обрабатывать исключения, связанные с БД.

3. **Отображение данных.** Создать страницу для просмотра данных:

– Вывести данные из основной таблицы в виде HTML-таблицы.

– Реализовать постраничный вывод (пагинацию).

– Добавить возможность сортировки по различным полям.

– Для каждой записи добавить ссылки на редактирование и удаление.

4. **Добавление записей.** Создать форму для добавления новых записей:

– Форма с полями, соответствующими структуре таблицы.

– Валидация данных на сервере.

– Сохранение данных в БД.

– Сообщение об успешном добавлении.

5. **Редактирование и удаление.** Реализовать функциональность:

– Страница редактирования записи с предзаполненными полями.

– Обновление данных в БД

– Подтверждение удаления записи (JavaScript confirm).

– Каскадное удаление связанных записей.

6. **Поиск и фильтрация.** Добавить возможность поиска и фильтрации данных:

– Поиск по текстовым полям.

– Фильтрация по категориям/диапазонам дат.

– Отображение количества найденных записей.

Таблица 4.1 - 10 вариантов структур баз данных:

Вар	**Тематика**	**Основная таблица**	**Связанная таблица**	**Дополнительные требования**
1	Студенты	students (id, name, group, year)	grades (id, student_id, subject, grade)	Вычисление среднего балла
2	Товары	products (id, name, price, category_id)	categories (id, name, description)	Фильтр по категориям
3	Заказы	orders (id, customer, date, total)	order_items (id, order_id, product, quantity)	Подсчет суммы заказа
4	Книги	books (id, title, author, year, genre_id)	genres (id, name)	Поиск по автору и году
5	Сотрудники	employees (id, name, position, department_id)	departments (id, name, floor)	Фильтр по отделам
6	Фильмы	movies (id, title, year, rating, director_id)	directors (id, name, birth_date)	Сортировка по рейтингу
7	Музыка	tracks (id, title, duration, album_id)	albums (id, title, year, artist)	Вычисление общей длительности
8	Пользователи	users (id, login, email, role_id)	roles (id, name, permissions)	Проверка уникальности email
9	События	events (id, title, date, location, organizer_id)	organizers (id, name, contact)	Фильтр по датам
10	Блог	posts (id, title, content, date, author_id)	authors (id, name, email, bio)	Поиск по содержимому

РГР №5. Разработка REST API

Цель работы: Научиться проектировать и реализовывать RESTful веб-сервисы, работать с форматами JSON/XML и документировать API.

Теоретическая часть:

1. Архитектурный стиль REST. Принципы и ограничения.
2. HTTP-методы (GET, POST, PUT, DELETE) и коды ответов.
3. Форматы обмена данными (JSON, XML).
4. JAX-RS / Spring MVC для создания REST API.
5. Документирование API (Swagger/OpenAPI).
6. Тестирование API с помощью Postman или curl.

Типовые задания:

1. **Проектирование API.** Разработать спецификацию REST API для заданной предметной области:

– Определить ресурсы и их URI.

– Для каждого ресурса определить поддерживаемые HTTP-методы.

– Описать ожидаемые форматы запросов и ответов.

– Определить возможные коды ответов для каждого метода.

– Составить OpenAPI-спецификацию (YAML или JSON).

2. **Реализация серверной части.** Создать REST-сервис, реализующий спроектированное API:

– GET-методы для получения списка ресурсов и конкретного ресурса.

– POST-метод для создания нового ресурса.

– PUT-метод для обновления существующего ресурса.

– DELETE-метод для удаления ресурса.

– Интеграция с базой данных (из РГР №4).

3. **Обработка ошибок.** Реализовать корректную обработку ошибок:

– Возврат соответствующих HTTP-кодов (404, 400, 500 и т.д.).

– Формирование информативных сообщений об ошибках в JSON.

– Валидация входящих данных.

4. **Документирование.** Подключить Swagger/OpenAPI для автоматической генерации документации:

– Аннотировать контроллеры и методы.

– Создать интерактивную документацию (Swagger UI).

– Обеспечить возможность тестирования API через интерфейс.

5. **Клиентское приложение.** Разработать простое клиентское приложение (HTML+JavaScript), которое:

– Отображает список ресурсов, полученный через API.

– Предоставляет форму для добавления нового ресурса.

– Позволяет удалять ресурсы.

– Использует Fetch API для взаимодействия с сервером.

6. **Тестирование.** Написать скрипты для тестирования API:

– Использовать Postman или написать скрипты на Python.

– Проверить все эндпоинты.

– Проверить обработку некорректных запросов.

Таблица 5.1 - 10 вариантов предметных областей для API:

Вар	**Ресурсы**	**Особые требования**	**Дополнительные эндпоинты**
1	Пользователи, Задачи	Задачи привязаны к пользователям	Фильтр задач по статусу
2	Товары, Категории, Заказы	Связь многие-ко-многим	Получение товаров категории
3	Статьи, Авторы, Комментарии	Комментарии к статьям	Получение комментариев статьи
4	Студенты, Курсы, Оценки	Оценки студентов по курсам	Средний балл студента
5	Фильмы, Актеры, Режиссеры	Связь многие-ко-многим	Поиск по году выпуска
6	Отели, Номера, Бронирования	Бронирование номеров	Проверка доступности дат
7	Врачи, Пациенты, Записи	Записи на прием	Расписание врача на дату
8	Книги, Авторы, Издательства	Книги с несколькими авторами	Поиск по ISBN
9	События, Участники, Билеты	Продажа билетов	Количество свободных мест

10	Проекты, Сотрудники, Задачи	Распределение задач	Задачи сотрудника за период

РГР №6. Одностраничные приложения (SPA) на современном фреймворке

Цель работы: Освоить разработку одностраничных приложений с использованием современных JavaScript-фреймворков (React, Angular или Vue).

Теоретическая часть:

1. Архитектура SPA. Преимущества и недостатки.
2. Компонентный подход. State и Props.
3. Жизненный цикл компонентов.
4. Маршрутизация на клиенте (React Router, Vue Router).
5. Управление состоянием (Redux, Pinia, NgRx).
6. HTTP-клиенты (axios, fetch).

Типовые задания:

1. **Настройка проекта.** Создать проект с использованием выбранного фреймворка (React, Angular или Vue):

– Настроить структуру проекта.

– Подключить необходимые зависимости.

– Настроить маршрутизацию.

2. **Создание компонентов.** Разработать компоненты для интерфейса:

– Компонент навигационного меню.

– Компонент списка элементов (из предметной области).

– Компонент карточки/детального просмотра.

– Компонент формы для создания/редактирования.

3. **Интеграция с API.** Подключить приложение к разработанному в РГР №5 REST API:

– Реализовать сервис для выполнения HTTP-запросов.

– Отображать состояние загрузки (лоадеры).

– Обрабатывать ошибки при запросах.

– Обновлять интерфейс после успешных операций.

4. **Маршрутизация.** Настроить клиентскую маршрутизацию:

– Страница со списком элементов (главная).

– Страница детального просмотра элемента.

– Страница создания нового элемента.

– Страница редактирования элемента.

– Страница "404" для несуществующих маршрутов.

5. **Управление состоянием.** Реализовать глобальное хранилище состояния:

– Хранить список элементов.

– Хранить информацию о текущем пользователе (если есть).

– Хранить состояние загрузки и ошибок.

– Реализовать действия (actions) для работы с API.

6. **Стилизация.** Оформить приложение с использованием CSS-фреймворка:

– Подключить Bootstrap, Material-UI или аналогичный.

– Обеспечить адаптивность интерфейса.

– Добавить анимации переходов.

Таблица 6.1 - 10 вариантов предметных областей:

Вар	**Тип приложения**	**Основные сущности**	**Дополнительные функции**
1	Менеджер задач	Задачи, Проекты, Приоритеты	Drag-and-drop задач
2	Заметки	Заметки, Теги, Категории	Поиск по тегам
3	Контакты	Контакты, Группы	Импорт контактов из vCard
4	Финансовый трекер	Транзакции, Категории, Счета	График расходов по категориям
5	Планировщик событий	События, Участники	Календарь событий
6	Чат-приложение	Сообщения, Контакты	Отправка изображений
7	Магазин	Товары, Корзина	Корзина с подсчетом суммы
8	Блог	Посты, Комментарии	Лайки постов
9	Погодное приложение	Города, Прогноз	Сохранение избранных городов
10	Музыкальный плеер	Треки, Плейлисты	Продолжительность плейлиста

РГР №7. Аутентификация и авторизация в веб-приложениях

Цель работы: Изучить методы аутентификации и авторизации пользователей, научиться обеспечивать безопасность веб-приложений.

Теоретическая часть:

1. Базовая аутентификация. Формы входа.
2. Хеширование паролей (bcrypt, PBKDF2).
3. JWT (JSON Web Tokens) — структура и применение.
4. OAuth 2.0 и OpenID Connect.
5. Защита от основных уязвимостей (XSS, CSRF, SQL-инъекции).
6. Роли и разрешения. Авторизация доступа.

Типовые задания:

1. **Регистрация и вход.** Реализовать систему регистрации и аутентификации:

– Форма регистрации с валидацией (имя, email, пароль).

– Хеширование паролей перед сохранением в БД.

– Форма входа с проверкой учетных данных.

– Сохранение состояния аутентификации (сессия или JWT).

– Защищенные страницы, доступные только авторизованным пользователям.

2. **Управление профилем.** Создать страницу профиля пользователя:

– Отображение информации о пользователе.

– Возможность редактирования профиля (имя, email).

– Смена пароля с подтверждением.

– Загрузка аватара (опционально).

3. **JWT-аутентификация (для REST API).** Реализовать аутентификацию через JWT:

– Эндпоинт для получения токена (login).

– Валидация токена при запросах к защищенным ресурсам.

– Обновление токена (refresh token).

– Выход из системы (logout) на клиенте.

4. **Роли и разрешения.** Внедрить систему ролей:

– Определить роли (администратор, модератор, пользователь).

– Ограничить доступ к определенным страницам/функциям на основе роли.

– Административная панель для управления пользователями.

– Возможность блокировки пользователей.

5. **Защита от атак.** Реализовать механизмы безопасности:

– Защита от CSRF-атак (токены в формах).

– Ограничение количества попыток входа (rate limiting).

– Безопасное хранение чувствительных данных.

– HTTPS (настройка в production-окружении).

6. **Социальная аутентификация (опционально).** Добавить вход через социальные сети:

– Интеграция с Google/Facebook/GitHub OAuth.

– Связывание аккаунтов социальных сетей с локальным аккаунтом.

Таблица 7.1 - 10 вариантов исходных данных:

Вар	**Тип приложения**	**Роли**	**Дополнительные требования**
1	Система управления задачами	admin, user	Админ видит все задачи
2	Блог	admin, author, reader	Авторы создают посты
3	Интернет-магазин	admin, manager, customer	Менеджеры управляют заказами
4	Образовательный портал	admin, teacher, student	Учителя создают курсы
5	Форум	admin, moderator, user	Модераторы удаляют сообщения
6	CRM-система	admin, sales, support	Разные интерфейсы для ролей
7	Библиотека	admin, librarian, reader	Библиотекари выдают книги
8	Медицинская система	admin, doctor, patient	Врачи видят истории болезней
9	Финансовое приложение	admin, analyst, client	Аналитики видят отчеты
10	Проектный менеджер	admin, project_manager, developer	PM создает задачи

РГР №8. Развертывание и оптимизация веб-приложений

Цель работы: Освоить процессы сборки, развертывания и оптимизации веб-приложений, познакомиться с инструментами CI/CD и контейнеризации.

Теоретическая часть:

1. Сборка фронтенда (Webpack, Vite, Parcel).
2. Контейнеризация с Docker. Dockerfile и docker-compose.
3. CI/CD pipelines (GitHub Actions, GitLab CI).
4. Облачные платформы (Heroku, AWS, Netlify, Vercel).
5. Оптимизация производительности (ленивая загрузка, кэширование).
6. Мониторинг и логирование.

Типовые задания:

1. **Настройка сборки проекта.** Настроить инструмент сборки для фронтенд-приложения:
   - Минификация и обфускация JavaScript.
   - Оптимизация изображений.
   - Транспиляция (Babel для поддержки старых браузеров).
   - Сборка CSS (PostCSS, Autoprefixer).
   - Разделение кода (code splitting).
2. **Контейнеризация приложения.** Создать Docker-образ для приложения:
   - Написать Dockerfile для фронтенда.
   - Написать Dockerfile для бэкенда (если приложение полностековое).
   - Создать docker-compose.yml для запуска всего стека (приложение + БД).
   - Оптимизировать размер образа (многоступенчатая сборка).
3. **CI/CD пайплайн.** Настроить автоматическую сборку и развертывание:
   - Создать конфигурацию GitHub Actions или GitLab CI.
   - Автоматическая сборка при пуше в репозиторий.

– Запуск тестов (если есть).

– Автоматическое развертывание на тестовый сервер.

4. **Оптимизация производительности.** Провести аудит производительности:

– Использовать Lighthouse для анализа.

– Оптимизировать критический путь рендеринга.

– Внедрить ленивую загрузку изображений и компонентов.

– Настроить кэширование статических ресурсов.

– Сравнить показатели до и после оптимизации.

5. **Развертывание на облачной платформе.** Опубликовать приложение:

– Выбрать платформу (Netlify/Vercel для фронтенда, Heroku/Render для бэкенда).

– Настроить переменные окружения.

– Подключить пользовательский домен (опционально).

– Настроить HTTPS.

6. **Мониторинг и логирование.** Добавить базовый мониторинг:

– Сбор и анализ логов приложения.

– Интеграция с Sentry для отслеживания ошибок.

– Добавление метрик производительности.

Таблица 8.1 - 10 вариантов конфигураций:

Вар	**Тип приложения**	**Стек технологий**	**Платформа развертывания**
1	React SPA	React + Vite	Netlify
2	Vue SPA	Vue + Webpack	Vercel
3	Angular SPA	Angular CLI	Firebase Hosting
4	Fullstack (React + Node)	MERN (MongoDB, Express, React, Node)	Heroku + Atlas
5	Fullstack (Vue + Django)	Vue + Django + PostgreSQL	Render
6	Fullstack (Angular + Spring)	Angular + Spring Boot + MySQL	AWS (EC2 + RDS)
7	Статический сайт	HTML + CSS + JS	GitHub Pages
8	Fullstack (Next.js)	Next.js + MongoDB	Vercel + Atlas

9	Fullstack (Nuxt.js)	Nuxt.js + PostgreSQL	Heroku
10	Flask-приложение	Python + Flask + SQLite	PythonAnywhere

Требования к оформлению РГР

1. Каждую расчетно-графическую работу следует выполнять в отдельной папке с четкой структурой проекта.

2. **Структура каждой работы:**

– Титульный лист.

– Задание (с указанием варианта).

– Техническое описание разработанного решения.

– Код программы (ключевые фрагменты с комментариями).

– Скриншоты работающего приложения.

– Инструкция по установке и запуску.

– Выводы по работе.

3. **Требования к коду:**

– Код должен быть читаемым, с комментариями.

– Соблюдение стандартов оформления (для каждого языка свои).

– Отсутствие закомментированного кода.

– Корректная обработка ошибок.

4. **Требования к графической части:**

– Скриншоты всех основных страниц приложения.

– Скриншоты результатов тестирования.

– Диаграммы (ER-диаграммы, архитектурные схемы) при необходимости.

5. **Электронная часть:**

– Полный исходный код проекта (в архиве).

– Файлы с зависимостями (package.json, requirements.txt и т.д.).

– Инструкция по установке (README.md).

– Ссылка на работающее приложение (если опубликовано).

Таблица 9.1 - Критерии оценивания

Критерий	**5 баллов**	**4 балла**	**3 балла**	**2 балла**
Функциональность	Полная реализация всех требований	80-90% требований	60-80% требований	Менее 60%
Качество кода	Отличная структура, комментарии, обработка ошибок	Хорошая структура, есть комментарии	Код работает, но плохо структурирован	Код нечитаемый, отсутствует обработка ошибок
Интерфейс	Современный, адаптивный, удобный	Аккуратный, но есть недочеты	Простой, функциональный	Неудобный или отсутствует
Документация	Полная, с инструкциями и скриншотами	Есть, но неполная	Минимальная	Отсутствует
Защита работы	Глубокое понимание всех аспектов	Хорошее понимание	Поверхностное понимание	Отсутствие понимания