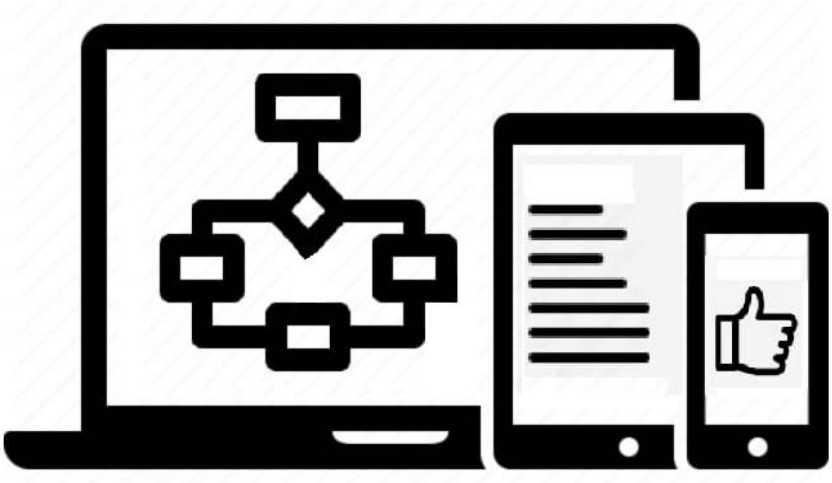
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА ПРИЛОЖЕНИЙ

Методические рекомендации к лабораторным работам для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» очной формы обучения



Могилев 2026

УДК 004
ББК 32.81
Р17

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «29» октября 2025 г., протокол № 3

Составитель канд. техн. наук Ю. В. Вайнилович

Рецензент канд. техн. наук В. М. Ковальчук

Методические рекомендации к лабораторным работам по дисциплине «Разработка web-интерфейса приложений» предназначены для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» очной формы обучения.

Учебное издание

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА ПРИЛОЖЕНИЙ

Ответственный за выпуск	В. В. Кутузов
Корректор	И. В. Голубцова
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Цель учебной дисциплины состоит в формировании у студентов глубоких теоретических знаний и практических навыков в области веб-программирования, глубоком представлении об основных технологиях и инструментах, используемых при разработке веб-сайтов и приложений как на стороне клиента, так и на стороне сервера.

Методические рекомендации по дисциплине «Разработка web-интерфейса приложений» к лабораторным работам предназначены для оказания помощи студентам при выполнении лабораторных работ по данной дисциплине. Они содержат задания для самостоятельного выполнения, инструкции, пояснения и рекомендации, которые помогают студентам освоить конкретные технологии и принципы программирования, связанные с разработкой интернет-приложений.

По результатам выполнения каждой лабораторной работы студент предоставляет отчет, который содержит:

- титульный лист;
- цель работы;
- постановку задачи;
- описание результатов выполненной работы.

1 Лабораторная работа № 1. Фиксированная верстка готового дизайн-макета

Цель работы: освоить технологии HTML, CSS.

Теоретический материал

Изучить основы работы в Figma с использованием [1–4].
Ознакомиться со следующим материалом.
1 Основы CSS:
- https://webref.ru/course/css-basics.

2 Основы позиционирования элементов:
- https://webref.ru/course/position;
- https://webref.ru/course/block-model;
- https://webref.ru/course/block-inline.

3 CSS grid layout:
- основные понятия Grid Layout;
- полное руководство по CSS Grid;
- верстка на Grid в CSS;
- https://css-tricks.com/snippets/css/complete-guide-grid;
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/.

4 Flexbox:
- https://webref.ru/layout/flexbox-tutorial;
- https://habr.com/ru/post/467049/;
- CSS: Flexbox;
- шпаргалка по Flexbox.

Задание для самостоятельного выполнения

Разработать дизайн-макет браузерной версии лендинга. Сверстать спроектированный лендинг. Варианты тем лендингов выбираются из приложения А.

Требования к верстке

Верстка валидная. Для проверки валидности верстки используйте сервис https://validator.w3.org/. Валидной верстке соответствует надпись «Document checking completed. No errors or warnings to show».

Верстка семантическая. Запрещается использование CSS-фреймворков (bootstrap, foundation и т. д.), JS-фреймворков (Angular, React, Vue и т. д.), устаревших библиотек (jQuery и т. д.).

Для создания многоколоночных структур или элементов, имеющих относительное горизонтальное расположение, должно быть использовано одно из свойств:
- display: flex;

– display: grid;
– display: inline-block.

Основные блоки должны быть точно расположены на заданной ширине экрана так, как в макете Figma.

Изображения, логотипы (если они есть) должны быть расположены в рамках логического контейнера с правильным подходом по центрированию и расположению. Допускается незначительное отклонение от макета в угоду сеточной или колоночной структуре.

Иконки, картинки должны сохранять идеальное расстояние до начала соответствующего им текста.

Иконки, картинки должны сохранять свои пропорции.

Если использован правильный шрифт, следует проверить высоту текста – он должен соответствовать исходнику. Ширина может варьироваться. Но общепринятой практикой является добавление свойства межбуквенного интервала (letter-spacing) тексту заголовков, девиза (motto) или цитат.

Если в строке несколько объектов визуально одинаковой ширины, то ширина содержащих их блоков должна быть одинаковой. Разница размеров изображений не имеет значения, важно совпадение размеров блоков. Если в макете ширина блоков разная, то делать ее все равно нужно одинаковой.

Некоторые элементы должны быть интерактивными. «Интерактивный» означает, что у кнопки или элемента появляется визуальный эффект или анимация (на ваше усмотрение и исходя из макета: анимация курсора, изменение цвета заднего фона, затемнение, нижнее подчеркивание, изменение шрифта) при каких-либо действиях пользователя, например при наведении курсора. Использовать JavaScript для обработки пользовательских событий в данном задании не обязательно. Обычно такой эффект реализуют при помощи псевдокласса hover и следующих свойств:

– cursor: pointer;
– background;
– text-decoration: underline;
– color.

2 Лабораторная работа № 2. Адаптивная верстка готового дизайн-макета. Технологии CSS Grid и Flexbox

Цель работы: овладеть техниками и методологиями адаптивного дизайна; выработать умение применять их для создания гибких и отзывчивых веб-интерфейсов.

Теоретический материал

Ознакомиться со следующим материалом.

CSS3-медиазапросы (https://html5book.ru/css3-mediazaprosy/).

CSS3 Адаптивный веб-дизайн. Медиазапросы (https: // www.schoolsw3.com /css/ css_rwd_mediaqueries.php).

Задание для самостоятельного выполнения

Разработать дизайн-макеты для планшетной и мобильной версий лендинга. Сверстать спроектированные страницы.

Требования к верстке

Должны быть выполнены все требования лабораторной работы № 2 до порогового значения (меньше 320px). Это означает, что отступы, размеры блоков и прочее не должны уходить за правый край экрана и не должен появляться горизонтальный скролл.

Задание оценивается путем изменения размеров окна браузера Google Chrome или подключением эмуляции устройств через панель разработчика (DevTools -> Toogle Device Toolbar), выбрав значение ширины экрана.

При проверке должна отсутствовать вертикальная полоса прокрутки, т. к. она «съедает» часть пространства отзывчивой верстки своей шириной. Чтобы ее отключить, необходимо выбрать режим эмуляции Responsive, а также установить тип устройства Mobile (рисунок 1). Если тип устройства не отображается, в верхней панели device toolbar нажмите на три точки справа и выберите Add device type.

Рисунок 1 – Режим эмуляции

«Responsive» – это размеры, заданные в относительных величинах от ширины окна или родительского блока, которые плавно меняют свои значения при уменьшении или увеличении окна браузера. Главное, чтобы при наложении картинки, например, в 768px на макет шириной 768px размеры или отступы совпадали.

3 Лабораторная работа № 3. Основы работы в Figma. Разработка дизайн-макета web-страницы

Цель работы: разработать дизайн экранов; выбрать подходящие элементы управления; определить навигацию и создать прототипы интерфейса для браузерного приложения.

Теоретический материал

Перед выполнением практического задания ознакомиться с [1–6].

Задание для самостоятельного выполнения

Разработать дизайн-макеты для 5–7 с. декстопного, планшетного и мобильного вариантов веб-приложения.

4 Лабораторная работа № 4. DOM Api

Цель работы: приобрести практические навыки навигации и управления DOM, понимания и применения CSS через JavaScript, а также обработки сложного поведения веб-документов.

Теоретический материал

Ознакомиться со следующим материалом.

Что такое Объектная Модель Документа (DOM) (https://developer.mozilla.org/ru/docs/Web/API/Document_Object_Model/Introduction).

Управление документами (https: // developer.mozilla.org / ru / docs / Learn / JavaScript/ Client-side_web_APIs/ Manipulating_documents).

Урок 7. JavaScript в веб-разработке (https://smartiqa.ru/courses/web/lesson-7-js).

Задание для самостоятельного выполнения

Разработать виртуальную клавиатуру. Пример представлен на рисунке 2.

Рисунок 2 – Пример виртуальной клавиатуры

Требования к приложению

Дизайн на ваше усмотрение.

Index.html должен быть пустым. Все необходимые элементы генерируются с использованием JS.

Если нажато несколько кнопок, то на виртуальной клавиатуре подсвечиваются все нажатые кнопки (также нет исключений для Ctrl, Alt и Shift).

Виртуальная клавиатура способна переключаться между двумя языковыми раскладками (английский + любой другой язык).

Назначение сочетания клавиш для переключения раскладки клавиатуры зависит от вас.

Кнопки на виртуальной клавиатуре отображают символы выбранного языка.

Приложение сохраняет выбранный язык после перезагрузки страницы и отображает клавиатуру на этом языке.

На странице должна быть указана комбинация клавиш для смены языка, чтобы пользователю было понятно, как переключать раскладку клавиатуры.

Нажатия клавиш анимированы.

Щелчки мышью по кнопкам на виртуальной клавиатуре и нажатия клавиш на физической клавиатуре должны вводить символы в текстовую область, расположенную на странице над виртуальной клавиатурой.

Нажатие клавиши со стрелкой вверх, вниз, влево или вправо вводит символ стрелки в поле ввода или реализует навигацию по текстовой области.

Нажатие Enter должно переместить текстовый курсор на следующую строку.

Клавиша Tab создает горизонтальный отступ.

Нажатие остальных функциональных клавиш на клавиатуре не приводит к вводу символов.

Клавиша Backspace удаляет символ перед текстовым курсором.

Клавиша Del удаляет символ после текстового курсора.

Клавиши Shift, Alt, Ctrl, Caps Lock и Space должны работать, как на настоящей клавиатуре.

Технические требования

Работа приложения проверяется в браузере Google Chrome последней версии.

Запрещается использовать Angular/React/Vue и другие фреймворки.

Не разрешается использовать jQuery, другие js-библиотеки.

JS-код приложения должен быть читаемым.

5 Лабораторная работа № 5. Асинхронный JavaScript

Цель работы: научиться получать данные от API и отображать их на веб-странице.

Теоретический материал

Ознакомиться со следующим материалом.

Using the Fetch API (https: // developer.mozilla.org/en-US/docs/ Web/API/ Fetch_API/Using_Fetch).

Практическое ES6 руководство о том, как сделать HTTP запрос с помощью Fetch API (https://jem-space.ru/praktichieskoie-es6-rukovodstvo-o-tom-kak-sdielat-http-zapros-s-pomoshchiu-fetch-api/).

Working with JSON (https: // developer.mozilla.org / en-US/docs / Learn / JavaScript/ Objects/JSON).

Что такое JSON (https://habr.com/ru/post/554274/).

Задание для самостоятельного выполнения

Разработать приложение, отображающее полученные от API фото. Добавить приложению поиск. При вводе поискового запроса изменяются фото, которые отображаются в приложении.

Варианты API.

1 Unsplash API:

– сайт: https://unsplash.com/developers;

– документация: https://unsplash.com/documentation.

2 Flickr API:

– сайт: https://www.flickr.com/services/;

– документация: https://www.flickr.com/services/api/flickr.photos.search.html.

Требования к верстке

Верстка адаптивная. Приложение хорошо выглядит при ширине страницы от 1920px до 768px.

Интерактивность элементов, с которыми пользователи могут взаимодействовать, изменение внешнего вида самого элемента и состояния курсора при наведении, использование разных стилей для активного и неактивного состояния элемента, плавные анимации.

В футере приложения есть ссылка на GitHub автора приложения, год создания приложения.

Представленные варианты страниц не для копирования, а как пример того, что можно сделать.

Технические требования

Работа приложения проверяется в браузере Google Chrome последней версии.

Можно использовать bootstrap, material design, css-фреймворки, html и css препроцессоры.

Не разрешается использовать jQuery, другие js-библиотеки и фреймворки.

JS-код приложения должен быть читаемым.

6 Лабораторная работа № 6. Разработка SPA-приложения с использованием библиотеки React

Цель работы: научиться разрабатывать web-приложения с использованием библиотеки React.

Теоретический материал

Перед выполнением практического задания ознакомиться с источниками [6, 7].

Задание для самостоятельного выполнения

У вашей бабушки скопилось большое количество ёлочных игрушек. Она попросила вас помочь их разобрать, чтобы выбрать те, которыми будет наряжать ёлку в этом году.

Зная, как бережно бабушка относится к этим игрушкам, вы отнеслись к её просьбе очень внимательно.

Вы составили опись всех имеющихся игрушек, указав для каждой её название, количество экземпляров, год покупки, форму (шар, фигурка, снежинка и т. д.), цвет, размер, отдельно отметили любимые бабушкины игрушки.

Теперь вам предстоит создать приложение, которое позволит отсортировать игрушки по названию и количеству экземпляров, найти игрушку по названию, сгруппировать игрушки по видам, добавлять игрушки в избранное и удалять из него, а также с интерактивной страницей, на которой выбранными игрушками можно украсить новогоднюю ёлку.

Функционал приложения

1 Страница с игрушками содержит карточки всех игрушек, а также фильтры, строку поиска, поле для сортировки.

2 Карточка игрушки содержит её изображение, название, текстом или условным значком обозначены количество экземпляров, год покупки, форма, цвет, размер, является ли игрушка любимой. Карточки игрушек добавляются динамически средствами JavaScript.

3 Добавление игрушек в избранное:

– кликая по карточке с игрушкой или по кнопке на ней, игрушку можно добавлять в избранное или удалять из избранного. Карточки добавленных в избранное игрушек внешне отличаются от остальных;

– на странице отображается количество добавленных в избранное игрушек. При попытке добавить в избранное больше 20 игрушек выводится всплывающее уведомление с текстом «Извините, все слоты заполнены».

4 Сортировка.

Сортируются только те игрушки, которые в данный момент отображаются на странице:

– сортировка игрушек по названию в возрастающем и спадающем порядке;

– сортировка игрушек по году их приобретения в возрастающем и спадающем порядке.

5 Фильтры в указанном диапазоне «от» и «до»:

– фильтры по количеству экземпляров;

– фильтры по году покупки;

– для фильтрации в указанном диапазоне используется range slider с двумя ползунками. При перемещении ползунков отображается их текущее значение, разный цвет слайдера до и после ползунка. Range slider можно создать на основе input[type=range] или использовать для этой цели библиотеку noUiSlider, или другую на ваш выбор.

6 Фильтры по значению.

Выбранные фильтры выделяются стилем:

– фильтры по форме;

– фильтры по цвету;

– фильтры по размеру;

– можно отобразить только любимые игрушки;

– можно отфильтровать игрушки по нескольким фильтрам одного типа;

– для нескольких фильтров одного типа отображаются игрушки, которые соответствуют хоть одному выбранному фильтру. Например, можно отобразить снежинки и колокольчики; или белые, синие и красные игрушки; или большие и средние.

7 Можно отфильтровать игрушки по нескольким фильтрам разного типа. Для нескольких фильтров разного типа отображаются только те игрушки, которые соответствуют всем выбранным фильтрам. Например, можно отобразить только синие шары или любимые белые и красные игрушки, купленные в 1940–1960 гг.

Если игрушек, соответствующих всем выбранным фильтрам, нет, на странице выводится уведомление в человекочитаемом формате, например «Извините, совпадений не обнаружено».

8 Сброс фильтров:

– есть кнопка reset для сброса фильтров;

– кнопка reset сбрасывает только фильтры, не влияя на порядок сортировки или игрушки, добавленные в избранное. После использования кнопки reset фильтры остаются работоспособными;

– при сбросе фильтров кнопкой reset ползунки range slider сдвигаются к краям, значения ползунков возвращаются к первоначальным, range slider закрашивается одним цветом.

9 Сохранение настроек в local storage. Выбранные пользователем фильтры, порядок сортировки, добавленные в избранное игрушки сохраняются при перезагрузке страницы. Есть кнопка сброса настроек, которая очищает local storage.

10 Поиск:

– при открытии приложения курсор находится в поле поиска;

– автозаполнение поля поиска отключено (нет выпадающего списка с предыдущими запросами);

– есть placeholder;

– в поле поиска есть крестик, позволяющий очистить поле поиска;

– если нет совпадения последовательности букв в поисковом запросе с названием игрушки, выводится уведомление в человекочитаемом формате, например «Извините, совпадений не обнаружено»;

– при вводе поискового запроса на странице остаются только те игрушки, в которых есть указанные в поиске буквы в указанном порядке. При этом не обязательно, чтобы буквы были в начале слова. Регистр символов при поиске не учитывается;

– поиск ведётся только среди игрушек, которые в данный момент отображаются на странице;

– если очистить поле поиска, на странице отображаются игрушки, соответствующие всем выбранным фильтрам и настройкам сортировки.

11 Дополнительный функционал на выбор:

– в процессе сортировки, фильтрации, поиска карточки с изображениями игрушек плавно меняют своё положение. Код для плавного перемещения карточек можно написать самостоятельно или использовать для этой цели какую-либо библиотеку. Библиотека может использоваться только для перемещения карточек, фильтрацию, сортировку и поиск необходимо написать полностью самостоятельно;

– очень высокое качество оформления приложения и дополнительный, не указанный в п. 1–10, сложный в реализации функционал, улучшающий качество приложения, удобство пользования им.

7 Лабораторная работа № 7. Библиотека компонентов MaterialUI, CSS-фреймворк Bootstrap

Цель работы: изучить и научиться применять компоненты и утилиты библиотеки готовых компонентов MaterialUI и фреймворка Bootstrap для разработки веб-приложений.

Теоретический материал

Ознакомиться со следующим материалом.

Официальная документация по фреймворку Bootstrap (https: // getbootstrap. com/).

Официальная документация по MaterialUI (https: // mui.com/material-ui / getting-started/).

Задание для самостоятельного выполнения

Для стилизации страниц веб-приложения использовать не менее 15 готовых компонентов, одну из библиотек компонентов на ваш выбор: Bootstrap или MaterialUI.

8 Лабораторная работа № 8. Пакеты для React-приложений сторонних разработчиков

Цель работы: ознакомиться с экосистемой React-пакетов и применить их для расширения возможностей React-приложения.

Теоретический материал

Ознакомиться со следующим материалом.

PDF, Excel, Docx generate on React and Node js (https://dev.to/golam_mostafa/pdf-excel-docx-generate-on-react-and-node-js-2keh).

Агрегатор библиотек (https://www.npmjs.com/).

Задание для самостоятельного выполнения

Реализовать функциональность экспорта данных в минимум двух разных форматах из следующих:

- word (docx);
- excel (xlsx**,** exceljs);
- pdf (jsPDF, pdfmake, react-pdf);
- google docs (google-docs-node).

Данные должны быть на русском языке. Отчет должен иметь шапку и/или одну или несколько табличных частей. Данные должны быть из реального состояния приложения.

Примеры отчетов: список товаров с ценами и характеристиками, отчет по заказам за период, сводка по категориям, избранное пользователя.

Создать дашборд/страницу аналитики с минимум двумя графиками разных типов (линейные, круговые, диаграммы и т. д.). Рекомендуемые библиотеки: Chart.js, react-chartjs-2, Victory, Nivo, Plotly.js.

Данные берутся из реального состояния приложения. Должны присутствовать подписи осей, легенда, заголовки на русском.

Примеры графиков: динамика продаж по месяцам (линейный график), распределение товаров по категориям (круговая диаграмма), топ-10 популярных товаров (столбчатая диаграмма), сравнение цен (график area).

Реализовать минимум две формы с использованием библиотек управления формами. Рекомендуемые библиотеки: formik, react-hook-form.

Каждая форма должна содержать минимум три критерия из списка:

- не менее семи полей разных типов;
- валидация (schema-based через «yup» или «zod»);

- зависимые поля (одно поле влияет на другое);
- динамические поля (добавление/удаление);
- загрузка файлов;
- сложные типы: datepicker, мультиселект, autocomplete;
- асинхронная валидация (проверка на сервере);
- мультишаговая форма (wizard).

Примеры форм: форма добавления товара (название, описание, цена, категория, фото, характеристики), форма регистрации (email, пароль, подтверждение, телефон, адрес, аватар), форма создания заказа (товары, доставка, оплата), форма фильтрации с множественными критериями.

Реализовать на одной из страниц функциональность перетаскивания элементов.

Рекомендуемые библиотеки: dnd-kit/core + dnd-kit/sortable, hello-pangea/dnd, react-dnd.

Примеры реализации: перетаскивание товаров в корзине, приоритизация списка избранного, сортировка категорий на админ-панели.

Реализовать таблицу с расширенным функционалом.

Рекомендуемые библиотеки: tanstack/react-table, ag-grid-react, react-data-grid.

Обязательный функционал: сортировка, фильтрация и пагинация.

Дополнительный функционал (один на выбор):

- выбор строк (чекбоксы);
- группировка данных;
- изменяемые колонки (редактирование in-place);
- закрепление колонок (sticky columns).

Примеры реализации: прайс-лист, история заказов, список пользователей.

9 Лабораторная работа № 9. Архитектурный паттерн Redux

Цель работы: изучить и научиться применять архитектурный паттерн Redux в разработке приложений; освоить принципы однонаправленного потока данных, централизации состояния приложения и управления изменениями стейта с помощью Redux.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.
Введение в Redux & React-redux (https://habr.com/ru/articles/498860/).
Redux Toolkit (https://rajdee.gitbook.io/redux-toolkit-in-russian).
Справочник React. Redux Toolkit (https://reactdev.ru/libs/redux-toolkit/).

Задание для самостоятельного выполнения

Задание выполняется на основе приложения, разработанного в лабораторной работе № 9. Реализовать управление состоянием приложения с помощью Redux Toolkit.

10 Лабораторная работа № 10. Архитектура веб-проектов на Node.js

Цель работы: изучить современные подходы к проектированию и организации архитектуры веб-приложений на платформе Node.js; освоить принципы разделения ответственности между компонентами системы; реализовать масштабируемую и поддерживаемую структуру проекта.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.

Node.js: история создания, архитектура и лучшие практики разработки для создания эффективных и устойчивых приложений (https:// devrating.org / ru/ content/nodejs).

Реализация чистой архитектуры в приложениях Node.js (https://appmaster.io/ ru/blog/chistaia-arkhitektura-nodejs-prilozhenii).

Задание для самостоятельного выполнения

Разработать микросервис управления каким-либо ресурсом (например, книгами, пользователями, продуктами, заказами и т. д.), который предоставляет RESTful API для работы с этим ресурсом. Приложение должно быть строго разделено на три архитектурных слоя: слой представления, сервисный слой, слой данных.

11 Лабораторная работа № 11. Работа с SQL базой данных. ORM

Цель работы: практически освоить разработку веб-приложений с использованием проектирования базы данных PostgreSQL, ORM Sequelize, REST API на Express.js, клиентского приложения на React, тестирования API в Postman.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.

REST API: для чего нужен и как работает (https://cloud.yandex.ru/ docs/ glossary/rest-api).

REST API (https://blog.skillfactory.ru/glossary/rest-api/).

СУБД PostgreSQL: почему её стоит выбрать для работы с данными и как установить (https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/).

Графический клиент pgAdmin (https://metanit.com/sql/postgresql/1.2.php).

Гибкая ORM для Node.js – Sequelize (https://proglib.io/p/gibkaya-orm-dlya-node-js-sequelize-2022-10-12).

Руководство по Sequelize (https://my-js.netlify.app/docs/guide/sequelize/).

Задание для самостоятельного выполнения

Создать базу данных в PostgreSQL, состоящую не менее чем из трех таблиц.

Создать модель БД с использованием ORM sequelize.

Разработать REST API, реализующее CRUD-операции для каждой сущности базы данных, включая:

- создание новой записи;
- получение списка записей с поддержкой пагинации;
- получение списка записей с поддержкой сортировки;
- получение списка записей с поддержкой фильтрации, в том числе по нескольким полям одновременно
- получение списка записей с поддержкой поиска, в том числе по нескольким полям одновременно;
- получение детальной информации по ID;
- обработка случая отсутствия записи;
- обновление записи;
- удаление записи;
- проверка существования записи.

Реализовать валидацию данных на уровне моделей с использованием встроенных и пользовательских валидаторов ORM Sequelize.

Провести тестирование всех эндпоинтов с использованием сервиса Postman.

12 Лабораторная работа № 12. Работа с NoSQL базой данных. ORM

Цель работы: практически освоить разработку веб-приложений с использованием нереляционной базы данных MongoDB, ODM Mongoose, REST API на Express.js, клиентского приложения на React, тестирования API в Postman.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.

Инсталляция NongoDB-сервера (https://www.mongodb.com/try/download).

Руководство по MongoDB (https://metanit.com/nosql/mongodb/).

Учебник по MongoDB Compass (https://www.alldevstack.com/ru/mongodb-tutorial/mongodb-compass.html).

Пакет mongoose (https://www.npmjs.com/package/mongoose).

Документация по mongoose (https://mongoosejs.com/docs/).

Задание для самостоятельного выполнения

Создать базу данных в MongoDB, состоящую не менее чем из трех таблиц.

Создать модель БД с использованием ODM Mongoose.

Разработать REST API, реализующее CRUD-операции для каждой сущности базы данных, включая:

- создание новой записи;
- получение списка записей с поддержкой пагинации;
- получение списка записей с поддержкой сортировки;
- получение списка записей с поддержкой фильтрации, в том числе по нескольким полям одновременно
- получение списка записей с поддержкой поиска, в том числе по нескольким полям одновременно;
- получение детальной информации по ID;
- обработка случая отсутствия записи;
- обновление записи;
- удаление записи;
- проверка существования записи.

Реализовать валидацию данных на уровне моделей с использованием встроенных и пользовательских валидаторов ORM Sequelize.

Провести тестирование всех эндпоинтов с использованием сервиса Postman.

13 Лабораторная работа № 13. Разработка web-сервера

Цель работы: изучить принципы работы фреймворка Express, принципы обработки HTTP-запросов и ответов.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.

Начало работы с Express (https://metanit.com/web/nodejs/4.1.php).

Express 4.x – API Reference (https://expressjs.com/en/api.html#express).

Задание для самостоятельного выполнения

Создать на сервере:

- GET-сервис, который возвращает веб-страницу с фронтенд-кодом;
- GET-сервис, который возвращает какие-то данные в формате JSON;

– POST-сервис, который возвращает какие-то данные в формате JSON;
– POST-сервис, который принимает какие-то данные;
– DELETE-сервис для отмены или удаления каких-то данных;
– какой-либо сервис для получения данных в формате XML/HTML/JSON в зависимости от заголовка запроса Accept;
– исходные данные хранятся в JSON-файлах;
– обработать исключительные ситуации;
– при изменении JSON-файлов приложение должно продолжать работу.

Создать на клиенте:

– получить с бэкенда данные по GET- и POST-сервисам;
– отобразить данные в любом виде (кнопки, списки, радиокнопки и т. д.);
– при каком-либо событии (нажатии на кнопку, выборе из списка и т. д.), в результате которого некоторые данные должны измениться, отправить ответ, запросить и отобразить обновлённые данные;
– добавить кнопки, скачивающие файл с данными в форматах XML/HTML/JSON.

14 Лабораторная работа № 14. Авторизация/аутентификация пользователей для доступа к API

Цель работы: разработать систему авторизации/аутентификации пользователей для доступа к API.

Теоретический материал

Ознакомиться со следующим материалом.

Требования аутентификации и авторизации API (https://starkovden.github.io/authentication-and-authorization.html).

Учимся работать с аутентификацией в Node используя Passport.js (https://medium.com/nuances-of-programming/учимся-работать-с-аутентификацией-в-node-используя-passport-js-58c14b9fe823).

Руководство по аутентификации в Node.js без passport.js и сторонних сервисов (https://habr.com/en/companies/ruvds/articles/457700/).

Простая авторизация на NODE JS. Роли пользователя. Express и MongoDB. JWT Access Token, bcrypt (https://www.youtube.com/watch?v=d_aJdcDq6AY).

Продвинутая JWT авторизация на React и Node js. Access, refresh, активация по почте (https://www.youtube.com/watch?v=fN25fMQZ2v0&t=2517s).

React + Node.js – JWT токен, авторизация – Облачное хранилище (https://www.youtube.com/watch?v=ITBPoR3p7YA).

JWT авторизация. Основы JWT-механизма (https://www.youtube.com/watch?v=IB9gsnlRt-4).

Задание для самостоятельного выполнения

Разработать страницу регистрации пользователя, включающую в себя:

– ввод номера телефона пользователя. Зарегистрироваться можно только на номер Беларуси;

– ввод email-пользователя;

– ввод даты рождения пользователя. Предусмотреть, что зарегистрироваться может пользователь, которому уже исполнилось 16 лет;

– возможность выбора способа задания пароля (самостоятельно или автоматически). Если пользователь вводит пароль самостоятельно, он его должен повторить (не используя вставку пароля в поле проверки). Пароль должен содержать минимум 8 символов, но не более 20. Должен включать в себя хотя бы одну заглавную букву, одну строчную букву, одну цифру и один специальный символ. Не должен входить в TOP-100 самых распространенных паролей 2025 г.;

– ввод ФИО пользователя. При этом отчество не является обязательным параметром;

– автогенерацию никнейма пользователя. Если пользователю автосгенерированный никнейм не нравится, предусмотреть возможность новой его генерации (по истечении пяти попыток необходимо предоставить возможность самостоятельного ввода никнейма);

– если никнейм пользователя уже существует, сообщить об этом;

– необходимость обязательного прочтения «Соглашения пользователя».

15 Лабораторная работа № 15. Тестирование React-приложений

Цель работы: ознакомиться с инструментами и методиками тестирования React-приложений.

Теоретический материал

Перед выполнением практического задания ознакомиться с материалом.

Модульные тесты JS: Автоматическое тестирование (https://ru.hexlet.io/courses/js-testing/lessons/unit-tests/theory_unit).

How to Start Unit Testing Your JavaScript Code (англ.) (https://www.freecodecamp.org/news/how-to-start-unit-testing-javascript/).

Jest – JS: Автоматическое тестирование (https://ru.hexlet.io/courses/js-testing/lessons/jest/theory_unit).

Тестирование React приложений (https://jestjs.io/ru/docs/tutorial-react).

Тестирование React-Redux-приложения (https://habr.com/en/ articles/340514/).

Тестирование React-компонентов с Jest и Enzyme (перевод) (https://medium.com/@karafizi/тестирование-react-компонентов-с-jest-и-enzyme-перевод-985dcab18b7e).

Задание для самостоятельного выполнения

Разработать не менее 10 unit-тестов для react-приложения из лабораторной работы № 6:

– тесты должны проверять различные сценарии использования компонентов, включая граничные случаи и некорректные входные данные;

– должно быть тестирование асинхронных операций (запросы к API);

– тесты должны быть независимыми друг от друга и не иметь побочных эффектов.

Разработать не менее 10 snapshot-тестов:

– snapshot-тесты должны покрывать основные состояния компонентов и их различные вариации;

– осмысленное именование снимков для лучшей организации и понимания;

– снимки должны обновляться осознанно, только после внесения намеренных изменений в UI;

– использовать возможности Jest для мокирования, ассертов и работы со снимками.

Покрыть тестами основные компоненты приложения:

– важно тестировать наиболее важные и сложные компоненты;

– следует тестировать как презентационные компоненты (отображение данных), так и контейнерные компоненты (логика и управление состоянием);

– тестировать взаимодействия между компонентами.

Проверить вывод, события, props и состояния:

– следует проверять корректность отображения данных, полученных из props и состояния;

– протестировать обработчики событий и их влияния на состояние компонентов;

– протестировать изменение состояния компонентов при различных действиях пользователя.

Использовать моки для изоляции внешних зависимостей:

– следует мокировать все внешние зависимости, такие как запросы к API, сторонние библиотеки и модули;

– проверять, что моки вызываются с правильными аргументами.

Автоматизировать запуск тестов через npm scripts.

Настроить скрипт для генерации отчета о покрытии кода тестами.

Протестировать стили, например, с использованием jest-styled-components или аналогичных библиотек.

Покрыть кода тестами:

– целевой уровень покрытия кода тестами составляет не менее 80 %;

– сгенерировать отчет о покрытии.

Список литературы

1 **Джейсон, Х.** Работа с Git. Полное руководство для начинающих и продвинутых пользователей / Х. Джейсон. – М. : ДМК Пресс, 2018. – 544 с.

2 Figma – где скачать и как установить программу на компьютер. – URL: https://web-design.center/ustanovka-figma (дата обращения: 21.02.2026).

3 **Окунев, А.** Руководство по Figma / А. Окунев. – URL: https://medium.com/slashdesigner/figma-guide-5235b8a8ab4f (дата обращения: 21.02.2026).

4 **Нагаева, И. А.** Основы web-дизайна. Методика проектирования: учеб. пособие / И. А. Нагаева, А. Б. Фролов, И. А. Кузнецов. – М. ; Берлин : Директ-Медиа, 2021. – 236 с.

5 Figma с нуля – основы работы с Фигмой для веб-разработчика, верстальщика и дизайнера. Полный обзор. – URL: https://www.youtube.com/watch?v=z6mlqOGmjQQ (дата обращения: 21.02.2026).

6 Лекция 3: Основы разработки интерфейсов мобильных приложений интерфейсов. – URL: https://intuit.ru/studies/courses/12643/1191/ lecture/21986?page=1 (дата обращения: 21.02.2026).

7 Типы адаптивных макетов. – URL: https://habr.com/ru/post/158703/ (дата обращения: 21.02.2026).

Приложение А
(справочное)

Варианты заданий для выполнения лабораторных работ

1 Автострахование.
2 Агентство по сдаче автомобилей в аренду.
3 Аренда коньков, роликов, велосипедов, лыж.
4 Аэропорт: пассажирское расписание и перевозки.
5 Банковская система вкладов (физических и юридических лиц).
6 Банковская система кредитования (физических и юридических лиц).
7 Биллинг сотовой компании.
8 Ветеринарная лечебница.
9 Клуб обучения танцам.
10 Магазин косметики.
11 Машиностроительное предприятие: система по разработке и модификации изделий (ведение архива, стандартов и пр.).
12 Нефтеперерабатывающая компания.
13 Парикмахерская.
14 Поставка вин.
15 Приемная комиссия вуза.
16 Производство мебели (прием индивидуальных и типовых заказов и изготовление).
17 Рекламное агентство.
18 Риелторская компания: аренда, продажа первичного и вторичного жилья.
19 Санаторий.
20 Система управления проектом для IT-компании.
21 Складская логистика.
22 Спа-салон (услуги, обслуживающий персонал и пр.).
23 Страховая компания.
24 Такси.
25 Транспортная логистика.
26 Туристическое агентство (путешествия за рубеж).
27 Туристическое агентство (путешествия по Беларуси).
28 Учет оборудования на крупном промышленном предприятии.
29 Филармония.
30 Электронный проездной.