Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

РАЗРАБОТКА ИНТЕРНЕТ-ПРИЛОЖЕНИЙ

Методические указания по выполнению лабораторных работ
для студентов направлений
«Программная инженерия» и «Бизнес-информатика»
(уровень бакалавриата)

2022

Владимиров Михаил Владимирович
Разработка интернет-приложений: Методические указания к лабораторным работам для студентов направления «Программная инженерия» и «Бизнес-информатика» (уровень бакалавриата) / М.В. Владимиров. – Томск, 2022. – 24 с.

Оглавление

1 Введение

Целью лабораторных работ по дисциплине «Разработка интернет-приложений» является закрепление теоретических основ по разработке современных интернет-приложений, а также формирование у студентов способности к самостоятельному или при помощи преподавателя анализу теоретического материала.

В результате проведения лабораторных работ студенты должны: получить знания об основных технологиях разработки, методах взаимодействия между элементами системы, подходах к проектированию ПО и разработке сложных систем; овладеть навыками разработки современных веб-приложений на основе паттерна SPA, овладеть основными навыками разработки серверной части на основе современных фреймворках.

При изучении данной дисциплины необходимо знание студентами информатики в объеме первого и второго семестров.

Пособие предназначено для студентов, обучающихся по направлению «Программная инженерия» и «Бизнес-информатика» всех форм обучения.

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Разработка REST API»

Цель занятия

Изучить основные понятия, касающиеся разработки интерфейса взаимодействия API; получить навыки реализации API разными методами (GET, POST, PUT, DELETE); овладеть программным обеспечением для разработки и тестирования API, овладеть навыками работы с современным фреймворком.

Порядок выполнения лабораторной работы

В качестве технологий для разработки выбраны фреймворк Symfony на основе языка PHP, база данных MySQL. Для начала выполнения лабораторной работы данные технологии должны быть установлены на ПК и готовы к работе.

Актуальная документация по Symfony находится по ссылке: https://symfony.com

В первую очередь для разработки REST API на основе фреймворка Symfony необходимо инициализировать новый проект, выполнив в консоли следующую команду:

symfony new <project_name>

После создания проекта в текущей директории появится папка с таким же названием. Для дальнейшей работы необходимо зайти в папку проекта в консоли, выполнив команду:

cd <project_name>

Следующий этап – настройка проекта. Чтобы подготовить проект к разработке, необходимо установить зависимости для его корректной работы. Среди необходимых зависимостей пакет улучшений в безопасности, в сериализации, ORM и MakerBundle. Для установки зависимостей требуется выполнить команды:

composer require serialize security orm

composer require maker-bundle --dev

Установку зависимостей необходимо завершить настройкой .env файла, в котором нужно настроить данные для работы с базой данных. Файл находится в корне папки и скрыт. Найти его можно через редактор VIM или IDE от JetBrains.

Корректная настройка файла .env позволяет без ошибок создать базу данных, выполнив команду:

php bin/console doctrine:database:create --if-not-exists

После настройки проекта можно приступать к разработке. В первую очередь надо проектировать систему. Понять, какие потребуются сущности и связи между ними. Далее нужно создать сущность и все необходимые поля для ее корректной работы. Для создания сущности необходимо использовать MakerBundle, выполняя команду:

php bin/console make:entity <entity_name>

Далее после запуска команды необходимо ответить на вопросы о создаваемой сущности, что позволит сгенерировать нужную структуру класса. По завершении работы команды Вам будут последовательно предложены 2 команды для генерации миграции и ее применения к текущей базе данных. Эти команды необходимо выполнить. После создания сущности создается миграция, которая изменяет состав базы данных, а после создается контроллер также при помощи MakerBundle, выполнив команду:

php bin/console make:controller <controller_name>

В контроллере настраивается префикс, который будет относиться ко всему контроллеру, а также методы CRUD для обработки основной информации о сущности:

- получение списка объектов сущности, хранящихся в базе данных, методом GET;
- получение определенного объекта по его идентификатору методом GET;
- создание объекта сущности методом POST;
- редактирование объекта сущности методом PUT;
- удаление объекта сущности методом DELETE.

Метод получения списка объектов должен быть реализован с помощью функции findAll из соответствующего репозитория.

Метод получения определенного элемента осуществляется с использованием метода find($id) из соответствующего репозитория.

Создание объекта осуществляется путем передачи данных на серверную часть в формате FormData и сохранения их в соответствующие поля нового объекта сущности. Необходимо также предусмотреть валидацию данных для обязательных полей. Сохранение данных в базу данных обязательно.

Редактирование объекта должно осуществляться аналогично созданию, но данные необходимо передавать в формате JSON. Выходные данные также должны быть в формате JSON.

Удаление объекта необходимо реализовать стандартной командой из репозитория. Поиск объекта также реализуется по идентификатору объ-

екта. Для несуществующих объектов должно быть предусмотрено сообщение об ошибке.

Все ответы должны возвращаться в формате JSON.

Чтобы начать тестирование разработанных API, необходимо запустить сервер для разработки, выполнив команду:

symfony *serve*

После запуска сервера для разработки все API будут доступны по адресу http://localhost:8000/ . Данные запросы можно протестировать через CURL в консоли или через ПО Postman, позволяющее управлять запросами.

Сдача лабораторной работы проходит путем демонстрации работоспособности разработанного API и написания отчета.

2.2 Практическое занятие «Разработка SPA»

Цель занятия

Изучить основы разработки одностраничных веб-приложений (SPA), получить работы в компонентном подходе.

Порядок выполнения работы

В качестве технологий для разработки SPA (одностраничных веб-приложений) используется NodeJS на основе языка программирования JavaScript.

Актуальная документация по ReactJS находится по ссылке: https://ru.legacy.reactjs.org/docs/getting-started.html

Перед началом выполнения лабораторной работы необходимо обеспечить работоспособность окружения, а именно актуальных версий nodejs и npm. Далее в рамках лабораторной работы будет применяться термин npx, который является частью окружения и устанавливается вместе с npm.

Для инициализации нового проекта необходимо выполнить команду:

npx create-react-app <project_name>

После создания проекта появится папка с таким же названием. Чтобы начать работу с проектом, необходимо в консоли зайти в папку проекта, выполнив команду:

cd <project_name>

Далее необходимо создать в проекте следующую структуру по выбранной теме: форма добавления и список добавленных элементов. Каждый элемент должен содержать данные из созданного объекта, а также 2 кнопки: редактировать и удалить.

Кнопка «Удалить» должна удалять элемент из списка.

Кнопка «Редактировать» должна переводить объект в режим редактирования. Объект в режиме редактирования должен отображать форму, идентичную форме создания объекта, но с предзаполненными данными в полях ввода данных, а также дополнительную кнопку «Отмена», которая будет выводить объект из режима редактирования без изменения данных.

Таким образом, в проекте должны быть реализованы 3 основных компонента: компонент со списком элементов, компонент с формой для создания/изменения объекта, компонент с отображением объекта.

В первую очередь создается компонент с отображением списка элементов. В данном компоненте должно находиться состояние, хранящее в себе массив элементов списка. Моковые данные лучше сохранять, используя хук useEffect, который будет запускаться при инициализации компонента.

Чтобы запустить рендер всех элементов списка, требуется использовать функцию map, чтобы отобразить все элементы массива.

Все функции, касающиеся изменений списка объектов, должны быть реализованы в данном компоненте и переданы в качестве входных параметров в дочерние компоненты.

Компонент отображения элемента в качестве состояния должен хранить данные о текущем элементе и флаг для управления режимом отображения.

Состояние, содержащее данные об элементе, требуется для осуществления возможности изменить данные объекта не меняя весь список. Все функции, касающиеся изменения определенного объекта, должны быть реализованы в данном компоненте.

Состояние режима должно управлять отображением компонента. В режиме отображения компонент должен рендерить информацию о переданном объекте. В режиме редактирования вместо отображения компонент должен рендерить компонент формы добавления объекта с предзаполненными данными из текущего объекта.

Компонент, реализующий логику работы формы создания/редактирования должен работать следующим образом: в качестве входных параметров должна передаваться функция обработки отправки формы. Необязательным входным параметром должен быть редактируемый объект. Если объект был передан, то форма должна работать в режиме редактирования (подставлять данные в соответствующие поля). Если объект не передан, то форма должна работать в режиме создания объекта.

По итогу выполнения лабораторной работы должно получиться одностраничное веб-приложение, которое позволяет просмотреть список

созданных объектов, отредактировать или удалить любой из объектов, а также добавить новый.

Для тестирования требуется запустить сервер для разработки, выполнив команду:

npm start

После запуска сервера для разработки проект будет доступен в браузере по адресу http://localhost:3000/.

Сдача лабораторной работы проходит путем демонстрации работоспособности разработанного функционала и написания отчета.

2.3 Практическое занятие «Интеграция API»

Цель занятия

Овладение навыками интеграции API в клиентское приложение, изучение инструментов JavaScript, направленных на организацию взаимодействия с серверной частью.

Порядок выполнения работы

Для выполнения данной лабораторной работы понадобится результат выполнения первой и второй лабораторных работ. Основной процесс разработки будет выполняться в проекте лабораторной работы №2. Но прежде, чем начать внедрение разработанных в Лабораторной работе №1 API в клиентскую часть приложения (Лабораторная работа №2), необходимо выполнить некоторые настройки.

Т.к. клиентская и серверная части приложения находятся на разных доменах (localhost:3000 и localhost:8000 соответственно), то все запросы будут считаться кросс-доменными (т.е. между разными доменами), что приведет к ошибке CORS. Данная ошибка возникает при попытке отправки кроссдоменных запросов и блокирует их в целях безопасности, если серверная часть не настроена соответственно.

Чтобы избежать возникновения данной ошибки, необходимо установить дополнительный пакет в проект, разработанный во время выполнения лабораторной работы №1. Для установки данного пакета нужно войти в папку проекта, как описано в лабораторной работе №1 и выполнить команду:

composer require cors

После выполнения этой команды ошибка возникнуть не должна. Теперь необходимо запустить сервер для разработки, чтобы клиент мог об-

ратиться к серверной части для работы с API. Запуск сервера выполняется путем запуска в консоли следующей команды:

symfony serve

После настройки серверной части необходимо запустить сервер для разаботки для клиентского приложения (проект, разработанный на лабораторной работе №2), перейдя в консоли в папку проекта и запустив команду:

npm start

После запуска веб-приложения в браузере можно приступать к внедрению. Для работы с запросами необходимо использовать функцию fetch() языка JavaScript.

Внедрить в клиентское приложение необходимо минимум 4 запроса: на получение данных, на создание объекта, на редактирование объекта и на удаление объекта. Количество запросов зависит от выбранной темы и функциональной необходимости.

В каждом запросе после получения ответа требуется полученные данные конвертировать из строки в объект или массив, выполнив функцию: response.json(). Далее полученные данные нужно обработать в зависимости от выполняемой функции.

Для запроса получения данных нужно сохранить весь массив объектов в состояние соответствующего компонента.

При создании объекта требуется добавить созданный объект в конец массива (списка элементов)

При редактировании объекта требуется обновить соответствующее состояние соответствующего объекта.

При удалении объекта требуется убрать удаляемый элемент из списка всех элементов.

Каждый запрос должен находиться в правильном месте и указывать определенные параметры.

Для запроса на получения списка элементов необходимо указать только url: fetch(<url>). Прочие параметры указывать не нужно.

Для запроса на создание элемента необходимо указать метод запроса и данные, которые были введены пользователем: fetch(<url>, {method: ‘POST’, body: data}). В данном случае переменная data должна содержать

объект стандартного в JavaScript класса FormData, в котором должны находиться все введенные данные.

Для запроса на редактирование элемента необходимо указать метод запроса PUT или PATCH, а также передать в виде JSON данные, введенные пользователем: fetch(<url>, {method: 'PUT', body: JSON.stringify(data),}).

Для удаления объекта требуется передать в параметрах только метод: fetch(<url_with_id>, {method: 'DELETE',}).

Все внедренные методы должны корректно работать при этом не перезагружая страницу.

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

2.4 Практическое занятие «Создание консольной команды»

Цель занятия

Овладение навыками создания консольных команд, изучение инструментов для работы с консолью в рамках фреймворка.

Порядок выполнения работы

Для решения определенных задач часто возникает необходимость для создания консольных команд. Данные команды используются при работе по таймеру (cron), для реализации функций, которые требуют ограничение на доступ из вне и т.д..

В рамках данной лабораторной работы будет использоваться проект, разработанный на лабораторной работе №1.

Для выполнения данной лабораторной работы необходимо создать консольную команду, которая позволит посмотреть, создать, редактировать или удалить объект созданного ранее класса и сохранить изменения в базу данных.

Для создания консольной команды в рамках серверной части приложения можно использовать MakerBundle. Чтобы создать новую консольную команду, необходимо в консоли зайти в папку проекта (как описано в лабораторной работе №1) и выполнить команду:

php bin/console make:command

Далее необходимо ввести название команды. После создания команда будет доступна к выполнению в консоли в таком виде:

php bin/console <command_name>

Теперь необходимо настроить команду. В первую очередь нужно настроить контейнер зависимостей. Для этого необходимо в классе, описывающем команду, создать параметр $repository. Далее требуется создать конструктор, входным параметром в который будет объект класса с репозиторием созданной ранее сущности. В самом конструкторе нужно объект репозитория сохранить в параметр класса.

Теперь, благодаря контейнеру зависимостей фреймворка, в рамках класса команды можно работать с базой данных. После этого можно приступать к разработке алгоритма выполнения команды.

Для организации интерактивности в консоли требуется использовать объекты Question Helper, которые уже установлены в проекте.

В первую очередь после запуска команды должен появиться выбор в виде списка значений и возможности выбора действия путем передвижения по списку стрелками на клавиатуре.

Далее команда должна запросить идентификатор изменяемого объекта, если выбрано не создание объекта. Для прочих команд необходимо выгузить элемент для дальнейшего изменения из базы данных.

Далее алгоритм должен по очереди запрашивать значения для всех полей объекта, если того требует команда. После введения всех данных необходимо сохранить изменения в базе данных и написать пользователю сообщение об успешно выполненном сохранении.

Все вопросы должны быть сформулированы понятно и показывать значение «по-умолчанию» для возможности быстрого пропуска заполнения значения любого поля.

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

2.5 Практическое занятие «Создание чата на серверной стороне проекта»

Цель занятия

Овладение навыками разработки функционала на основе веб-сокетов, изучение инструментов для тестирования веб-сокетов, овладение навыками работы с веб-сокет-сервером.

Порядок проведения занятия

В веб-разработке существует ряд задач, которые требуют организации двустороннего канала для обмена данными между клиентом и сервером. Самый эффективный способ – это настроить веб-сокет, который позволит серверной части уведомлять об обновлениях всех подключенных клиентских приложений. Частным случаем использования такого функционала является чат.

Задачей в рамках данной лабораторной работы является разработка единой комнаты чата, к которой подключаются все клиенты. При отправке сообщения одним клиентом его должны получать все.

Для разработки подобного функционала требуется установить дополнительные зависимости в проект. Для языка PHP и фреймворка Symfony в частности подходит инструмент Ratchet. Установить пакет, который автоматизирует часть процессов, можно командой:

Composer require cboden/ratchet

После установки данного пакета можно приступать к разработке. Для начала разработки необходимо создать класс Chat, который будет реализовывать всю логику работы с данными. В этом классе должны обязательно присутствовать следующие функции: onOpen, onMessage, onError, onClose.

onOpen – это функция, которая отвечает за обработку подключения нового клиента. Основной задачей данной функции является сохранение данных подключенного клиента в хранилище, чтобы далее у других функций была возможность отправлять сообщения именно этому клиенту.

onClose – это функция, которая обрабатывает событие закрытия соединения с клиентом. Основной процесс данной функции – это удаление клиента из хранилища, чтобы он более не считался актуальным участником процесса взаимодействия серверной и клиентской частей.

onError – это функция, которая обрабатывает ошибочное закрытие соединения. В случае возникновения ошибки также необходимо удалить клиента из хранилища и логировать сообщение об ошибке. При выполне-

нии данной лабораторной работы процесс логирования реализовывать не обязательно.

onMessage – это функция, которая реализует основную логику взаимодействия. В данной функции обрабатываются все входящие сообщения, а также рассылаются все необходимые данные для всех участников сети. В рамках текущей лабораторной работы необходимо реализовать получение сообщения и пересылку его всем прочим подключенным клиентам за исключением автора сообщения.

Документация по данному инструменту находится по ссылке: http://socketo.me/docs/hello-world

После разработки основного класса чата требуется настроить веб-сокет-сервер. Веб-сокет-сервер играет роль точки входа в проект и принимает все запросы на подключение, которые далее передает в основной класс чата.

Для создания веб-сокет-сервера необходимо создать консольную команду, которая будет запускать сервер, встроенный в установленный пакет Ratchet. Процесс создания команды идентичен процессу, описанному в лабораторной работе №4.

Как только новая команда была создана, необходимо вычистить из новой команды неиспользуемые элементы, а также прописать в теле команды код, запускающий сервер и перенаправляющий запросы в чат: $server = IoServer::factory(new Chat(), 8080);

Данная строка кода позволяет инициализировать сервер. Запуск сервера выполняет следующая строка кода: $server->run();

После написания данного кода в теле команды необходимо запустить данную команду в консоли. После запуска веб-сокет-сервер будет доступен по адресу ws://localhost:8080/.

Чтобы протестировать подключение, необходимо использовать ПО Postman или самую простую html-страницу.

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

2.6 Практическое занятие «Создание чата на клиентской стороне»

Цель занятия

Овладение навыками внедрения сокетов в клиентскую часть веб-приложений, изучение особенностей жизненного цикла при работе с веб-сокетами.

Порядок выполнения работ

Основной задачей данной лабораторной работы является внедрение веб-сокетов, разработанных на лабораторной работе №5, в проект клиентского приложения, разработанного на лабораторной работе №2.

Данная лабораторная работа делится на две основные части: внедрение роутинга и интеграция веб-сокетов.

Внедрение роутинга требуется для разграничения функций, реализованный в лабораторной работе №2 и чата, который будет разработан в рамках текущей лабораторной работы.

Для внедрения роутинга в проект требуется установка дополнительного пакета React Router. Чтобы установить пакет, требуется в консоли выполнить команду:

npm install --save react-router-dom localforage match-sorter sort-by

Актуальная информация о пакете находится по ссылке: https://reactrouter.com/en/main/start/tutorial.

В файле App.js необходимо предусмотреть инициализацию роутера до компонента таким образом: let router = createBrowserRouter([{path: “/”, element: <Component />},]);

Количество объектов в массиве определяет количество страниц в роутере.

Для тестирования описанного функционала необходимо предусмотреть наличие меню, которое будет перенаправлять пользователя на разные страницы.

Необходимо функционал, разработанный при выполнении лабораторной работы №2, разместить на отдельной странице. Также необходимо создать новую страницу «Чат», на которой будет реализован чат.

Для реализации чата на новой странице в первую очередь требуется создать соответствующую верстку.

Следующим этапом будет создание глобальной в рамках компонента переменной, которая будет хранить информацию о сокет-соединении.

Используя хук useEffect, при инициализации компонента нужно создать соединение и сохранить его в глобальную переменную, созданную ранее. Создание соединения происходит следующим блоком кода: let conn = new WebSocket('ws://localhost:8080');

Далее требуется описать обработчики событий: onopen, onclose, onerror, onmessage.

Как правило, onopen используется для проверки создания соединения в тестовых проектах. Выполняется это следующим кодом: conn.open = e => {console.log('Connection established!')};

Обработчики событий onclose и onerror требуют переподключения к серверу. Дополнительно в обработчике события возникновения ошибки нужно выводить текст об ошибке в консоль, чтобы можно было осуществлять отладку.

Обработчик события получения сообщения onmessage должен реализовывать логику обработки сообщений. Для этого требуется создать состояние, хранящее в себе весь список сообщений. Далее при получении сообщения в событии onmessage нужно полученные данные добавить в массив, хранящийся в состоянии.

Для тестирования описанного функционала необходимо открыть одну и ту же страницу в разных вкладках браузера. Написав и отправив сообщение в одной вкладке, оно автоматически должно появиться в другой вкладке – это показатель работоспособности системы.

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

2.7 Практическое занятие «Докеризация приложения»

Цель занятия

Овладение навыками работы с ПО Docker, овладение навыками работы с yaml-файлами, овладение навыков контейнеризации серверного приложения.

Порядок выполнения работ

Контейнеризация приложения необходима для стандартизации окружения для разработки, для автоматизации ряда процессов развертывания, а также для ряда процессов, которые проходят на этапе публикации на сервер.

Актуальная документация по ПО Docker находится по ссылке: https://www.docker.com

Для выполнения лабораторной работы потребуется установить актуальную версию ПО Docker и плагина Docker Compose.

Docker помогает упаковывать приложение в контейнеры. Docker Compose необходим для объединения ряда контейнеров в рамках одного проекта.

В проекте должно быть такое количество контейнеров, какое количество различных задач стоит перед проектом. В рамках лабораторной работы необходимо обернуть в Docker серверную часть проекта, разработанную при выполнении лабораторных работ 1 и 5.

В рамках данного проекта задачи и, соответственно, контейнеры должны быть следующие:

- контейнер с базой данных для хранения информации;

- контейнер для обработки запросов на основе PHP;

- контейнер для работы с веб-сокетами на основе PHP;

- контейнер для проксирования запросов в нужный контейнер на основе NGINX.

В корне проекта нужно создать файл docker-compose.yml. В этом файле должно находиться минимум 2 раздела: version и services. Блок vesion должен содержать текущую версию файла. Блок services описывает все сервисы и взаимосвязь между ними.

Также в проекте должна появиться папка docker, которая будет содержать все необходимые данные для работы docker. В данной папке должны находиться временные файлы, такие как база данных и логи. Также данная папка должна содержать дополнительные файлы, такие как Dockerfile (файл для настройки определенных пакетов в рамках контейнеров) и default.conf (настройки для веб-сервера nginx с описанием правил проксирования запросов в нужные контейнеры).

Документация по настройке NGINX находится по ссылке: https://nginx.org/en/docs/

После настройки контейнеров и сопутствующих файлов необходимо собрать и запустить контейнеры. Для этого требуется в консоли запустить команду:

docker-compose up -d --build

Если лог запуска не показал ошибок, то все контейнеры запущены успешно. Для проверки работоспособности контейнеров можно выполнить команду:

docker-compose ps

Также нужно выполнить все команды по созданию базы данных и применению миграций. Для этого можно выполнить команду по следующему шаблону:

docker-compose exec <php_code_container_name> <console_command>

После успешного запуска контейнеров необходимо запустить клиентское приложение, разработанное при выполнении лабораторных работ 2, 3 и 6 и проверить работоспособность серверной части и интеграции ее с клиентской частью.

Если все функции работают так же, как и при отсутствии контейнеров, то лабораторная работа выполнена успешно. Чтобы остановить работу контейнеров, требуется в консоли выполнить команду:

docker-compose down

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

2.8 Практическое занятие «Авторизация в приложении»

Цель занятия

Получение навыков по разработке процесса авторизации, инструменты авторизации на серверной части проекта, внедрение элементов авторизации в клиентскую часть приложения.

Порядок выполнения работ

Процесс авторизации является одним из важнейших и базовых процессов в современных веб-сервисах. Процесс авторизации направлен на ограничение доступа пользователей к контенту, который требует определенную роль или принадлежность пользователю.

В рамках данной лабораторной работы требуется создать сущность пользователя, API для регистрации пользователя, API для авторизации

пользователя, а также необходимо ограничить доступ к API, разработанные в ходе выполнения лабораторной работы №1, по критерию принадлежности пользователю. Также важно внедрить данные ограничения и функционал авторизации и регистрации в клиентскую часть приложения.

Для создания сущности пользователя в консоли, находясь в папке проекта, необходимо выполнить команду:

php bin/console make:user

Далее в процессе выполнения команды будут заданы уточняющие вопросы, ответы на которые позволят сформировать необходимые класс сущности, класс репозитория и настройки проекта.

После создания сущности пользователя необходимо создать контроллер, который будет реализовывать функции для работы с пользователем. Создается контроллер аналогично инструкции, описанной в лабораторной работе №1. В данном контроллере должно быть 2 метода: метод регистрации и метод авторизации.

Метод регистрации должен получать на вход адрес электронной почты, пароль и повтор пароля нового пользователя. Должна быть предусмотрена валидация входных данных: пароли должны совпадать, адрес электронной почты должен отсутствовать в базе данных.

По итогам работы метода новый пользователь должен быть сохранен в базе данных, а в ответе должны присутствовать данные о пользователе и хэш-токен для авторизации пользователя.

Метод авторизации должен получать в качестве входных данных адрес электронной почты и пароль. Пароль должен совпадать с хранящимся в зашифрованном виде в базе данных для пользователя с таким же адресом электронной почты. Ответ данного метода должен соответствовать ответу метода регистрации.

Следующим этапом необходимо создать custom authentificator, который будет проверять наличие токена в заголовках запросов и находить пользователя, который связан с данным токеном. Актуальное описание создания собственного аутентификатора находится в документации по Symfony, ссылка на которую приведена в п.2.1. данных методических указаний.

После настройки процесса аутентификации необходимо доработать API, созданные при выполнении лабораторной работы №1. Но прежде чем дорабатывать API, нужно обновить сущность. Обновление сущности

происходит так же, как и ее создание, описанное в лабораторной работе №1. Необходимо добавить новое поле user с типом данных ManyToOne к сущности User. Теперь все элементы данной сущности будут связаны с определенным пользователем.

Доработка API заключается в том, чтобы при создании элемента автоматически в объект добавлялась связь с пользователем. В контроллере при передаче токена в заголовке объект пользователя можно получить, используя функцию $this->getUser();.

Следующий этап выполнения лабораторной работы – создание форм авторизации и регистрации на клиентской части приложения. Доступ к странице с функционалом из лабораторной работы №1 должен быть ограничен для пользователей, которые не авторизованы. При успешной авторизации полученный токен должен сохраняться в localStorage, которое предоставляется всеми современными браузерами. Наличие токена в локальном хранилище браузера и должно являться критерием допуска на страницу с функционалом лабораторной работы №1.

Также нужно доработать все запросы к серверу на данной странице. В каждый из них необходимо поместить заголовок Authorization со значением токена.

Если все функции работают для авторизованного пользователя, но не работают для неавторизованного, то лабораторная работа считается выполненной.

Сдача лабораторной работы проходит путем демонстрации работоспособности заданного функционала и написания отчета.

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Целями самостоятельной работы являются систематизация, расширение и закрепление теоретических и практических знаний, приобретение навыков разработки интернет-приложений.

Самостоятельная работа студента по дисциплине «Разработка интернет-приложений» включает следующие виды его активности:

1. проработка лекционного материала;

2. подготовка к лабораторным работам;

3. подготовка к экзамену.

3.2 Проработка лекционного материала

Данный вид самостоятельной работы направлен на получение навыков работы с конспектом, структурирования материала, а также умения выделить основные пункты и положения, изложенные на лекции. Кроме того, проработка лекционного материала способствует более глубокому пониманию и прочному запоминанию теоретической части дисциплины.

При проработке лекционного материала необходимо:

1. отработать прослушанную лекцию, то есть прочитать конспект, прочитать документацию и сопоставить его материал с конспектом; восполнить пробелы, если они остались после лекции в силу того, что студент что-то не понял или не успел записать;

2. перед каждой последующей лекцией прочитать предыдущую, чтобы не тратилось много времени для восстановления контекста изучения дисциплины при продолжающейся теме, а также чтобы максимально правильно ответить на вопросы теста, который проводится на каждой лекции.

Для наиболее эффективной работы с конспектом рекомендуется сначала просмотреть его целиком, чтобы выделить структуру лекции. Эту структуру полезно выписать в виде плана. Затем по каждому пункту нужно выделить основные положения, определения и формулы, если они есть. Формулы тоже полезно записывать, чтобы кроме зрительной, включалась еще и моторная память.

3.3 Подготовка к лабораторным работам

Для подготовки к практическим занятиям необходимо изучить теоретические вопросы по теме работы, проработать основные понятия, необходимые для решения практических задач и выполнения индивидуального задания по практической работе.

Лабораторная работа «Разработка REST API»

При подготовке к данной лабораторной работе необходимо повторить изученный материал по организации баз данных, а также по объектно-ориентированному программированию. Данная информация понадобится в процессе выполнения лабораторной работы.

Для студентов, выполняющих лабораторную работу на личном ПК требуется развернуть окружение, а именно PHP версии не меньше 8.2, MySQL, Composer и Symfony-cli.

Также необходимо установить на компьютер следующее ПО: Postman и одну из доступных IDE.

Лабораторная работа «Разработка SPA»

При подготовке к выполнению данной лабораторной работы требуется повторить изученный материал по предмету «Основы гипертекстового представления контента», а именно основы HTML, CSS и JavaScript.

Для студентов, выполняющих лабораторную работу на личном ПК требуется развернуть окружение, а именно NodeJS актуальной версии и NPM.

Лабораторная работа «Интеграция API»

При подготовке к данной лабораторной работе необходимо изучить документацию по JavaScript в области теории и доступных функций по организации взаимодействия с сервером. Необходимо изучить процесс взаимодействия через встроенную функцию fetch и дополнительные инструменты и их особенности.

Лабораторная работа «Создание консольной команды»

Для успешной подготовки к лабораторной работе по теме «создание консольной команды» необходимо необходимо повторить изученный материал о работе с консолью или терминалом. Основные команды, ко-

торые используются в стандартном shell-терминале, команды, которые предоставляет composer, symfony-cli и сопутствующих бандлов.

Необходимо также учитывать, что разработка будет вестись на языке PHP и фреймворке symfony, в связи с чем стоит повторить пройденный материал и перечитать документацию по теме.

Лабораторная работа «Создание чата на серверной стороне проекта»

В данном разделе рассматриваются только задачи по созданию канала между двумя модулями системы. Разработка чата включает в себя основы объектно-ориентированного программирования, основы работы с консолью или терминалом, основа организации серверов для обработки запросов.

Также в рамках подготовки к выполнению работы требуется изучить документацию пакета для организации сокет-соединений о основ работы сокетов.

Лабораторная работа «Создание чата на клиентской стороне»

Для подготовки к данной работе необходимо учитывать рекомендации, приведенные в предыдущем пункте.

Кроме этого, для качественного выполнения лабораторной работы необходимо повторить пройденный материал по организации жизненного цикла в рамках компонентов React-приложения.

Лабораторная работа «Докеризация приложения»

При подготовке к данной лабораторной работе нужно изучить актуальную документацию по использованию ПО Docker, а также лекционный материал. Особенное внимание необходимо уделить принципу разбиения проекта на контейнеры.

Также требуется изучить возможности cli, которое предоставляет ПО Docker, а именно команды для запуска, остановки, проверки и проксирования команд.

Лабораторная работа «Авторизация в приложении»

При подготовке к лабораторной работе по теме «Авторизация в приложении», следует обратить внимание на теоретические основы процесса авторизации, а также на виды токенов, используемых в веб-приложениях.

4. Рекомендуемая литература

1. Организация баз данных: Методические указания к лабораторным работам, курсовой работе и организации самостоятельной работы / П. В. Сенченко - 2018. 85 с.

2. Информатика и программирование: Учебное пособие / Н. В. Пермякова - 2016. 188 с.

3. Основы гипертекстового представления интернет-контента: учебное пособие / Ю. П. Ехлаков, Э. К. Ахтямов - 2017. 181 с.