**4.** **Проектно-технологическая часть**

# 4.1 Технологические решения, поддерживающие эксплуатационный цикл программы

При проектировании системы с использованием библиотеки React.js и платформы Node.js был использован свободный фреймворк для веб-API Express. С их помошью можно реализовать архитектурный паттерн Model-View-Controller.

Архитектура Model-View-Controller – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

В данной работе паттерн MVC включает ряд компонентов:

* Модель пре доступ к данным, обрабатывается слоем работы с базой данных. Он осуществляет запросы в базу данных и проверку на корректность. Модель не зависит от представления, а следовательно, не имеет возможности визуализировать данные и не имеет точек взаимодействия с пользователем. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом уведомляя «наблюдателей».
* Представление (View) отвечает за получение необходимых данных из модели и отправляет их пользователю.
* Контроллер обеспечивает общение между пользователем и системой. Он контролирует и направляет данные от пользователя к системе и наоборот.
* Система маршрутизация как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

В общем случае, когда к приложению приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращать к данным через модели и для рендеринга ответа использовать представления. Результат обработки контроллера отправляет в ответ клиенту. В данной работе ответом от веб-API будут являться JSON объекты при запросе данных или код ответа HTTP при их загрузке на сервер.

Модель данного паттерна изображена на рисунке 7.

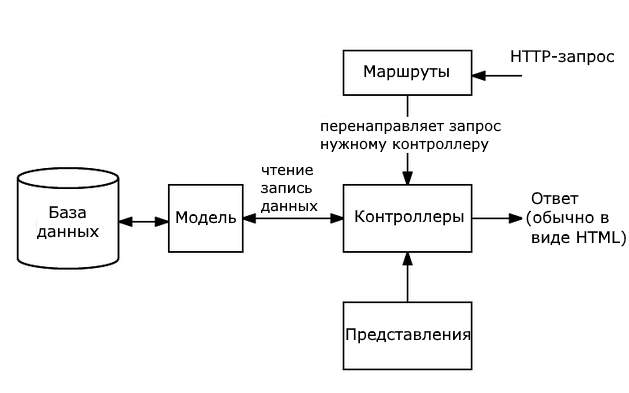


Рисунок 11 – Паттерн Model-View-Controller для веб-приложений Node.js

При проектировании клиентской части необходимо учитывать жизненный цикл React-компонентов. React позволяет определять компоненты как классы или функции. Единственный обязательный метод в подклассе React.Component — render(). Все остальные методы, являются необязательными. Каждый компонент имеет несколько «методов жизненного цикла» (рис. 12). Переопределение такого метода позволяет выполнять код на конкретном этапе этого процесса.

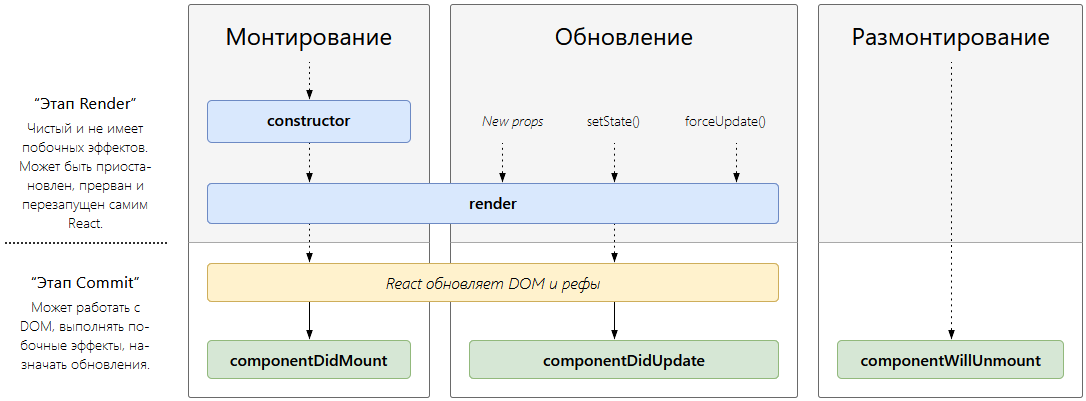


Рисунок 12 – Жизненный цикл компонентов React

* componentDidMount() вызывается сразу после монтирования (то есть, вставки компонента в DOM). В этом методе должны происходить действия, которые требуют наличия DOM-узлов. Это хорошее место для создания сетевых запросов.
* componentDidUpdate() вызывается сразу после обновления. Не вызывается при первом рендере.
* componentWillUnmount() вызывается непосредственно перед размонтированием и удалением компонента. В этом методе выполняется необходимый сброс: отмена таймеров, сетевых запросов и подписок, созданных в componentDidMount().

**Выводы**

В ходе выполнения данной курсовой работы была проанализированная выбранная предметная область, выбран инструментарий для реализации функционала веб-приложения, а также выведены UML-диаграммы сценариев работы пользователя.

Была реализована базовая часть веб-приложения, в которой была заложена функциональность для ее дальнейшего расширения. В ходе реализации веб-приложения были получены практические навыки работы с приложением для создания макетов Figma и библиотекой для веб-разработки React.js.

Таким образом цель курсовой работы достигнута, а все поставленные задачи выполнены.

**Список источников**

1. Абрамов Г.В. Проектирование информационных систем : учебное пособие / Г.В. Абрамов, И.Е. Медведкова, Л.А. Коробова. - Воронеж : Воронежский государственный университет инженерных технологий, 2012,172 с. URL://biblioclub.ru/index.php?page=book&id=141626
2. Диков, А.В. Веб-технологии HTML и CSS / А.В. Диков. – 2-е изд. – Москва: Директ-Медиа, 2012. – 78 с.: ил.,табл., схем. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=96968, по подписке
3. Золотов С.Ю. Проектирование информационных систем: учебное пособие/ ЗолотовС.Ю.— Томск: Томский государственный уни-верситет систем управления и радиоэлектроники, Эль Контент, 2013.— 88 c.— URL: <http://www.iprbookshop.ru/13965>
4. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления. — М. : Горячая линия-Телеком, 2011. — 304 с. — URL: <http://e.lanbook.com/book/5191>
5. Сидоров В.Н., Сломинская Е.Н., Полникова Т.В., Макарова О.Ю. Оформление графической части выпускной квалификацион-ной работы. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2016.
6. Сырецкий Г.А. Проектирование автоматизированных систем. Часть 1: учебное пособие/Сырецкий Г.А.— Новосибирск: Новоси-бирский государственный технический университет, 2014.— 156 c.— URL: <http://www.iprbookshop.ru/47714>
7. Форд Н., Хайгард М., де Ора Б. - 97 этюдов для архитекторов программных систем. - Пер. с англ. - СПб. Символ-Плюс, 2010. - 224 с., ил. ISBN 978-5-93286-176-9
8. Документация React.js [Электронный ресурс] / reactjs.org – Электрон. текстовые дан. – Режим доступа: https://ru.reactjs.org/docs/getting-started.html, свободный.
9. Стефанов СтоянС79 React.js. Быстрый старт. — СПб.: Питер, 2017. — 304 с.: ил. — (Серия «Бестселлеры O’Reilly»).
10. Документация приложения Figma [Электронный ресурс] / figma.com – Электрон. текстовые дан. – Режим доступа: https://help.figma.com/, свободный.