МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П.М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра информатики и управления бизнесом

Допущен к защите

«\_\_»\_\_\_\_\_\_\_20\_\_г.

Заведующий кафедрой

\_\_\_\_\_\_\_\_С.А.Ермоченко

ДИПЛОМНЫЙ ПРОЕКТ

РАЗРАБОТКА WEB -ПРИЛОЖЕНИЯ В СФЕРЕ ЗДРАВОХРАНЕНИЯ

Специальность: 1-31 03 07 Прикладная информатика

Мащёнский Дмитрий Юриевич,

4 курс, 41 группа

Научный руководитель:

Осипов Александр Владимирович,

Старший преподаватель

Витебск, 2022

**РЕФЕРАТ**

Дипломный проект представлен следующим образом. Электронные носители: 1 компакт-диск. Пояснительная записка составляет 86 страниц, 32 рисунка, 3 таблицы, 17 литературных источников, 3 приложения.

Ключевые слова: ReactJS, Redux, Saga, JavaScript, Node JS

Предметная область: автоматизация и учёт пациентов в сфере здравоохранении.

Объект разработки: web-приложение для автоматизации и учета пациентов в сфере здравоохранении.

Целью разработки является web-приложение, которое предоставит возможность автоматизировать процесс докторов. При разработке веб-приложения были использованы следующие среды разработки и сторонние программы: Visual Studio Code, Chrome, Postman. Языки программирования, которые были использованы при разработке: ReacteJS, JavaScript, NodeJS База данных – MySQL.

В результате технико-экономического обоснования было выяснено, что инвестирование в разработку является выгодным вложением, а само приложение является прибыльным, что говорит об экономической обоснованности данной разработки.

Разработанный продукт может использоваться работниками медучреждения для упрощения работы с личными делами пациентов, записи пациентов, записи нового пациента, поиска существующей истории болезни пациента, обновления данных последующего обследования пациента, записи времени посещения пациента

Задачи, поставленные перед реализацией дипломного проекта, были выполнены. В перспективе возможно расширение функционала новыми возможностями.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 6

1 ОБЗОР ЛИТЕРАТУРЫ 8

1.1 Обзор аналогов 8

1.1.1 ABM Retail 8

1.1.2 CloudShop 10

1.1.3 FloraPOS 12

1.2 Обзор средств разработки 13

1.2.1 Языки программирования 14

1.2.2 Философия REST 14

1.2.3 Обзор инструментов разработки серверной части приложения 15

1.2.4 Обзор инструментов разработки клиентской части приложения 17

1.2.5 Обзор систем управления базами данных 21

1.3 Постановка задачи 22

1.4 Обоснование выборки средств разработки 22

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 24

2.1 Модуль работы backend 24

2.2 Модуль работы frontend 24

2.3 Модуль работы с данными 25

2.4 Модуль терминала 25

2.5 Модуль админ-панели 26

2.6 Модуль авторизации для терминала 26

2.7 Модуль авторизации для админ-панели 27

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 28

3.1 Модуль работы backend 28

3.2 Модуль работы с данными 40

3.3 Модуль реляционной базы данных 44

3.4 Модуль работы frontend 49

3.5 Модуль терминала 50

3.6 Модуль админ-панели 53

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 56

4.1 Аутентификация и авторизация для админ–панели с применением JWT-токенов 56

4.2 Аутентификация и авторизация для терминала с применением JWT-токенов 59

4.3 Алгоритм работы флориста в терминале 62

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 66

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 71

6.1 Руководство пользователя для терминала 71

6.2 Руководство пользователя для админ-панели 76

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА ПО РАЗРАБОТКЕ СИСТЕМЫ АВТОМАТИЗАЦИИ И УЧЕТА ДЛЯ ЦВЕТОЧНЫХ МАГАЗИНОВ 80

7.1 Характеристика программного средства 80

7.2 Расчет затрат на разработку ПО 80

7.2.1 Затраты на основную заработную плату команды разработчиков 80

7.2.2 Затраты на дополнительную заработную плату команды разработчиков 81

7.2.3 Отчисления в фонд социальной защиты населения и фонд обязательного страхования 82

7.2.4 Прочие затраты 82

7.3 Оценка экономического эффекта от продажи ПО 83

7.3.1 Экономический эффект от продажи ПО для организации разработчика 83

7.4 Расчет показателей эффективности инвестиций в разработку ПО 84

ЗАКЛЮЧЕНИЕ 86

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 87

ПРИЛОЖЕНИЕ А 89

ПРИЛОЖЕНИЕ Б 108

ПРИЛОЖЕНИЕ В 109

# ВВЕДЕНИЕ

Сегодняшняя эпоха - это эпоха информатизации. С развитием технологий и научной теории традиционная медицина, основой которой является биотехнология, постепенно начала оцифровываться. И появилось современное здравоохранение, включающее в себя новое поколение информационных технологий. "Современное здравоохранение" - это не просто технический прогресс, но и всесторонние, многоуровневые изменения. Это изменение воплощено в следующем: изменения медицинской модели (от ориентированной на болезнь помощи к ориентированной на пациента), изменения в конструкции информатизации (от клинической информатизации к региональной медицинской информатизации), изменения в медицинском управлении (от общего управления к персонализированному управлению) и изменения в концепции профилактики и лечения (от сосредоточение внимания на лечении заболеваний, чтобы сосредоточиться на профилактическом здравоохранении).Эти изменения направлены на удовлетворение индивидуальных потребностей людей при одновременном повышении эффективности медицинской помощи, что значительно повышает качество медицинского обслуживания и медицинского обслуживания и представляет собой будущее направление развития современной медицины. Этот обзор начнется с концепции интеллектуального здравоохранения, затем кратко представит ключевые технологии, поддерживающие интеллектуальное здравоохранение, и объяснит достижения и проблемы ИТ, проанализировав статус применения этих технологий в важных областях медицины, прежде чем, наконец, выдвинуть будущие перспективы интеллектуального здравоохранения.

С точки зрения принятия решений, создание интегрированной платформы управления может реализовать такие функции, как распределение ресурсов, анализ качества и анализ эффективности, и может снизить медицинские расходы, максимально эффективно использовать ресурсы, и помогать больницам принимать решения относительно развития. Что касается удобства работы с пациентами, то пациенты могут получить доступ к множеству функций, таких как системы физического осмотра, онлайн-записи на прием и взаимодействие между врачом и пациентом. Эти автоматизированные системы делают процессы лечения пациентов более лаконичными. Пациенты ждут меньше времени и получают более гуманное обслуживание. В целом, интеграция, усовершенствование и автоматизация - это будущие направления развития "современных больниц ".

Таким образом, целью дипломного проекта является разработка web-приложение, которое предоставит возможность автоматизировать процесс докторов.

# 1 ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Обзор аналогов

Разрабатываемый программный продукт, конечно же имеет аналоги. Существует множество решений, которые предоставляют возможность автоматизировать процесс докторов.

Найти универсальный продукт, который смог бы решить все необходимые задачи, очень сложно. Максимально нецелесообразно переплачивать за продукт, в котором есть избыточный функционал, часть из которого останется непригодным. Поэтому стоит задача разработать продукт, соответствующий всем требованиям и реализующий все необходимые задачи.

В качестве аналогов были рассмотрены следующие программные продукты:

- Online-zapis;

- Medods;

### 1.1.1 Online-zapis

Online-zapis (рисунок 1.1) – это готовое решение для автоматизации любых заведений, которые работают с клиентами.

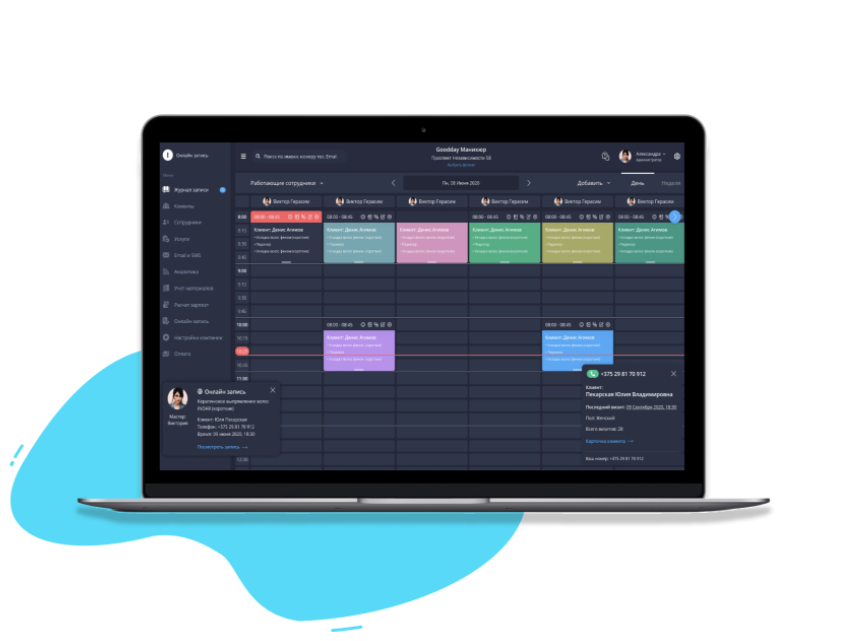


Рисунок 1.1 – Online-zapis

Решения, реализованные в Online-zapis:

* онлайн-запись для медицинского центра;
* онлайн-запись для салона красоты;
* онлайн-запись для шиномонтажа;
* онлайн-запись для юристконсульта;
* онлайн-запись для стоматологии;
* онлайн-запись для барбершопа;
* онлайн-запись для автомойки;
* онлайн-запись для коворкинга;
* онлайн-запись для лаборатории;
* онлайн-запись для спа-салона;
* онлайн-запись для СТО;
* онлайн-запись для фотостудии;
* онлайн-запись для поликлиники;
* онлайн-запись для массажной студии;
* онлайн-запись для техосмотра;
* онлайн-запись для персонального тренера;
* онлайн-запись для косметологии;

Компания Online-zapis позваляет внедрять клиентов через социальные сети, а также производить запись через TikTok, Facebook, Instagram.

### 1.1.2 МЕDODS

МЕDODS (рисунок 1.2) – это готовое решение для автоматизации медзаведения.



Рисунок 1.2 – МЕDODS [2]

Возможности, реализованные в МЕDODS:

* Онлайн-запись;
* ЭМК;
* Кассы по 54-ФЗ;
* СМС;
* Телефония;
* Лаборатории;
* МДЛП;
* Агрегаторы;
* Безопасность;

В основном это приложение для руководителей, маркетологов, администраторов, врачей. Приложение избавляет от заполнения бумаг, так как уже есть: гибкие и легкие шаблоны. Важным плюсом приложения, является быстрый поиск пациента и онлайн-уведомления о приходе пациента. Почему выбирают – это приложение дешевле большинство конкурентов, не нужно разбираться, можно сразу начинать работу, шифрование данных при передаче и хранении.

## 1.2 Обзор средств разработки

Для разработки веб-приложения используются языки программирования и фреймворки, которые облегчают разработку. В процессе разработки для отдельных модулей проекта может использоваться несколько языков программирования.

### 1.2.1 Языки программирования

**1.2.1.1 NODE JS**

*Node JS* –это платформа, основанный на движке JavaScript V8 Chrome.

Преимущества использования Node JS для разработки веб-приложений:

*Мощный стек технологий*:JavaScript, безусловно, является одним из самых мощных языков программирования, используемых во всем мире. Поэтому любой, кто решит использовать Node JS, естественно, получит полный доступ ко всем функциям JavaScript. Это означает отличную скорость и производительность приложений, легко обрабатываемые и повторно используемые коды, а также беспрепятственный доступ к многочисленным бесплатным инструментам. Это, естественно, помогает убедиться, что веб-компании и по всему миру имеют гораздо больше гибкости, когда дело доходит до работы над проектом. Фактическая работа по разработке также становится намного менее трудоемкой. Разработчики, имеющие навыки работы с внешним JavaScript, могут выполнять разработку на стороне сервера с минимальными усилиями.

*Гарантирует быстрые результаты обработки***:** Одна из лучших особенностей использования Node JS заключается в том, что он действительно очень быстрый. Движок, предназначенный для реализации Node JS, изначально создавался для браузера Chrome. Версия Chrome V8 была написана на C++ и предназначена для компиляции в машинный режим функций, написанных на JavaScript. Он может управлять всеми связанными процессами на чрезвычайно высоких скоростях. Кроме того, такие функции, как асинхронная обработка запросов и неблокирующий ввод/вывод, могут помочь Node JS быстро обрабатывать запросы без каких-либо задержек. Он также поставляется с моделью, основанной на событиях, которая обеспечивает быструю синхронизацию, если один и тот же язык используется как для клиентов, так и для серверной части. Это действительно может пригодиться при создании приложений реального времени с функциями, основанными на событиях.

*Богатая экосистема NodeJS* : Node JS имеет очень богатую экосистему и включает в себя более 836000 библиотек. Он также имеет менеджер пакетов по умолчанию, называемый npm, который обеспечивает быстрое улучшение инструментов JavaScript. Более 97% всех современных веб-приложений используют модули npm, и это делает Node JS идеальным решением для обширных требований к разработке веб-приложений.

*Отличная масштабируемость*: приложения, разработанные с помощью Node JS, чрезвычайно масштабируемы, что обеспечивает лучшую производительность и желаемый отклик приложений. Это одна из основных причин, почему разработчики Node Js настоятельно рекомендуют работать с Node JS.

### 1.2.2 Обзор инструментов разработки клиентской части приложения

*React* –это библиотека JavaScript, разработанная Facebook, которая, среди прочего, использовалась для создания Instagram.com. Его цель — позволить разработчикам легко создавать быстрые пользовательские интерфейсы для веб-сайтов и приложений. Основная концепция React.js — виртуальный DOM. Это дерево, основанное на компонентах JavaScript, созданных с помощью React, которое имитирует дерево DOM. Он делает минимально возможное количество манипуляций с DOM, чтобы поддерживать ваши компоненты React в актуальном состоянии. Разработчику достаточно описать, как части приложения выглядят в разных состояниях, а React будет обновлять их, когда они изменятся. Декларативное представление делает код более предсказуемым.

Будучи частью языка JavaScript, использование React дает множество преимуществ. Продукты, созданные с помощью React, легко масштабировать, единый язык, используемый на серверной/клиентской/мобильной стороне, обеспечивает выдающуюся производительность, существуют шаблоны рабочего процесса для удобной командной работы, код пользовательского интерфейса удобочитаем и удобен в сопровождении и многое другое. Ведущие мировые компании использовали React и другие технологии JS в некоторых продуктах, определяющих рынок (наиболее яркими примерами являются Instagram, Reddit и Facebook).

Одной из основных причин использования React.js для веб-разработки является оптимизированный интерфейс библиотеки и язык кодирования. Таким образом, облегченный API React усилен возможностями высокой производительности для обеспечения беспрепятственного и быстрого рабочего процесса разработки. Компоненты и концепции React действительно просты для понимания, поэтому здесь не нужно много времени для обучения.

В отличие от других популярных фреймворков, таких как Vue и Angular, здесь нет множества дополнительных HTML-атрибутов (созданных, когда JavaScript «втиснут» в HTML — стандартная практика для традиционных фреймворков и решений библиотек JS). В конечном счете, помещая JSX в JavaScript (буквально наоборот), React предоставляет гораздо более чистый, лучше читаемый и более полный код.

Использование React.js для веб-разработки может оказаться чрезвычайно удобным, потому что React — это один из тех случаев, когда вы осваиваете одну технологию, чтобы легко использовать ее на разных платформах. И все из-за того, что это библиотека по своей природе, основной целью которой является создание отдельных элементов и компонентов веб-дизайна (все, что угодно, от кнопок и меток до сеток и интерактивных функций).

Вдобавок ко всему, значительный вклад вносит обширное, давно сложившееся сообщество. Текущая экосистема React настолько обширна, что позволяет разработчикам создавать настольные решения и мобильные приложения, создавать статические веб-сайты, обрабатывать серверный рендеринг и интегрировать передовые технологические концепции (такие как виртуальная реальность и 360-градусные обзоры) с веб-решениями — и все это с помощью аналогичной простой веб-разработки. руководства и философии.

Виртуализируя и сохраняя DOM в памяти, React предоставляет невероятно быстрые возможности рендеринга, при этом все изменения представления легко отражаются в виртуальном DOM. Специализированный алгоритм сравнения сопоставляет прежние и существующие состояния виртуального DOM, вычисляя наиболее эффективный способ применения новых изменений, не требуя слишком большого количества обновлений. Затем вводится минимальное количество обновлений для достижения максимального времени чтения/записи, что приводит к общему повышению производительности.

Особый спрос на React для веб-разработки также продиктован возможностями Flux и Redux, которые он предлагает «из коробки». Создатели Facebook впервые представили программную архитектуру на основе Flux, которая дополнила стандартные компоненты React возможностями однонаправленного потока данных и предложила более оптимальную структуру действий.

Таким образом, центральный диспетчер используется для управления созданными действиями и хранилищами обновлений. Затем он обновляет представления в соответствии с изменениями в магазине. При этом все данные остаются в хранилищах — дубликаты не генерируются, что помогает поддерживать синхронизацию всех данных модели во всем приложении, не уходя далеко.

Помимо прочего, наши специалисты лично объясняют, почему мы используем React JS для веб-разработки, рассказывая об образцовом наборе инструментов и технологическом стеке. *Инструменты разработчика React* вместе с *инструментами разработки Redux* предлагают чрезвычайно удобные возможности, в то время как вы можете легко установить и использовать их как обычные расширения Chrome. В частности, с их помощью вы можете эффективно проверять компоненты иерархии на основе React (включая связанные реквизиты и состояния), проверять действия отправки и просматривать изменения состояния сразу в расширении (которые также можно записывать и использовать в качестве резервной копии для отладки в расширении). будущее).

*V8* – движок, который компилирует JavaScript в машинный код, пропуская стадию компиляции в промежуточный байт-код [12].

*ExpressJS* – это популярный, быстрый и гибкий MVC NodeJS фреймворк, который предлагает широкий набор функций для веб-приложений, а также для разработки мобильных приложений [13]. Он имеет API, который позволяет пользователям настраивать маршруты для отправки и получения запросов между frontend и базой данных. Кроме того, Express поставляется с системой просмотра, которая поддерживает более 14 шаблонизаторов и согласование контента.

Redux – это библиотека для управления глобальным состоянием приложения.

**Redux** предлагает решение, гарантируя, что:

* Ваше состояние заключено в хранилище, которое обрабатывает все обновления и уведомляет весь код, который подписывается на хранилище, об обновлениях состояния. Вам больше не нужно передавать состояние через все дерево компонентов, но вы можете подписаться на эти изменения гораздо ближе к тому месту, где необходима информация.
* Все изменения вносятся последовательно, чтобы обеспечить предсказуемый конечный результат без неожиданных эффектов и условий гонки.
* Состояние неизменяемо, а это означает, что каждое изменение состояния приводит к созданию совершенно новой версии состояния, что позволяет нам писать более предсказуемый код или просматривать любую предыдущую версию состояния с помощью **инструментов разработчика Redux**. Это приводит к невероятному опыту отладки.

Библиотека возникла на основе шаблона Flux. Flux – это архитектурный подход, который состоит из 4 компонент:

«Действие» – вспомогательная функция, которая упрощает передачу данных Диспетчеру.

«Диспетчер» – принимает Действия и рассылает данные обработчикам.

«Хранилище» – контейнер, в котором находится состояние приложения.

«Представление» – React-компоненты, которые собирают состояния из хранилищ и передают дочерним компонентам.

Принцип работы Flux представлен на рисунке 1.6.

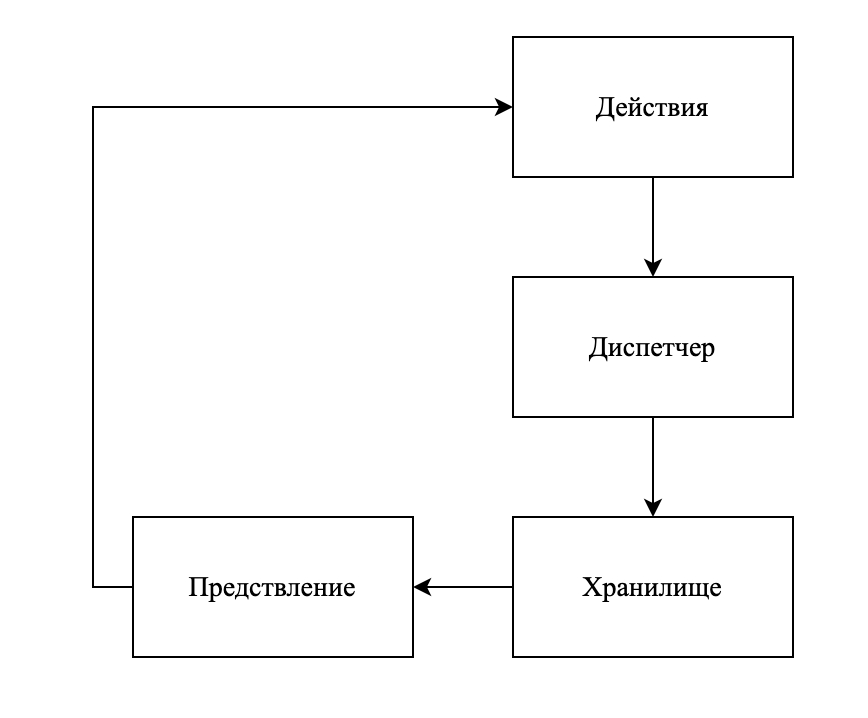


Рисунок 1.6 – Принцип работы Flux

«Действие» – это такая функция, которая вызывается из «Представления», чтобы отправить действие «Диспетчеру». Для различения действий используется константный подход, эта константа посылается вместе с данными действия.

«Диспетчер» – это главный узел процесса. Он получает данные и отправляет их всем зарегистрированным действиям, в свою очередь обработчики вызываются в определенном порядке, а далее действия могут быть обработаны в хранилище и состояние приложения будет изменено. Пример работы «Диспетчера» приведен на рисунке 1.7.

Функции, которые вызываются в «Диспетчерах» должны быть «чистыми».

Чистая функция – это такая функция, которая является детерминированной и не обладает сторонним эффектами.

Детерминированная функция – это такая функция, которая для одного и того же набора входных значений будет возвращен одинаковый результат.

Стоит отметить, что к сторонним эффектам можно отнести такие вещи, как: операции ввода-вывода, реагирование на исключительные ситуации, вызов обработчиков исключительных ситуаций, модификация глобальных переменных.

Существует так же другой тип побочных эффектов – это изменение входных значений в процессе работы функции.

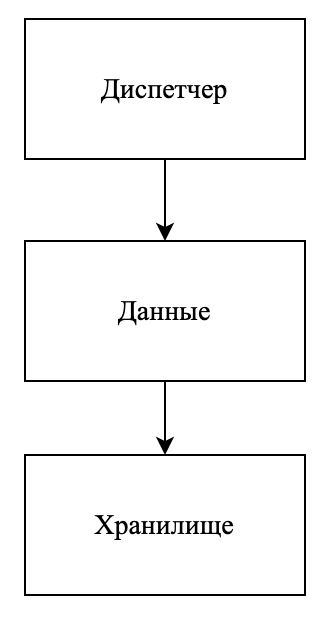


Рисунок 1.7 – Принцип работы «Диспетчера»

Хранилище во Flux занимается управлением состояниями приложения. Хранилище хранит данные и методы для получения данных. Подробная роль Хранилища во Flux представлена на рисунке 1.8.

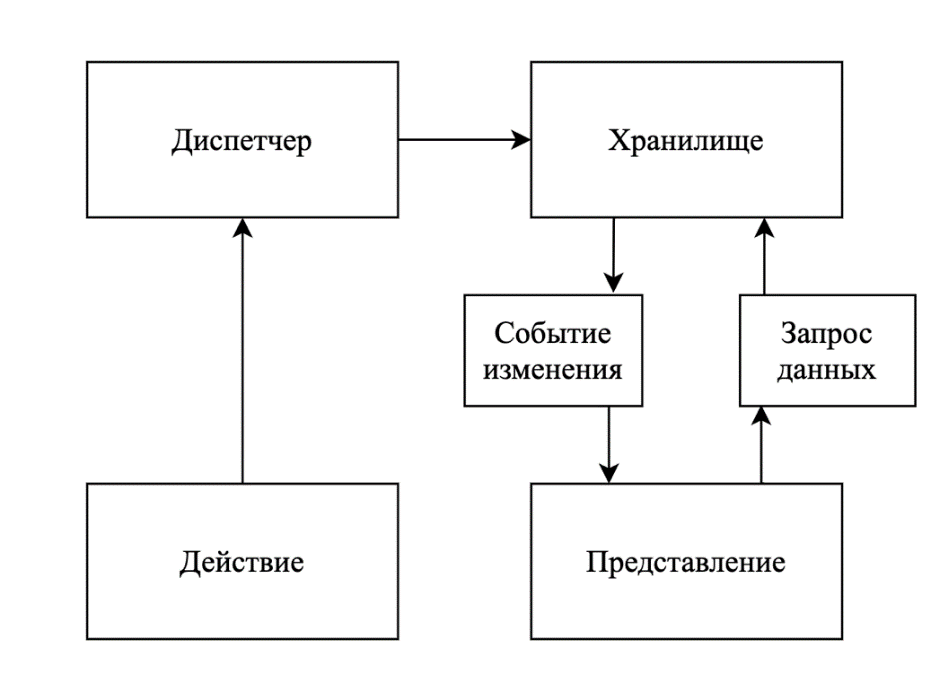


Рисунок 1.8 – Роль Хранилища во Flux

Как видно из рисунка, «Представление» подписано на хранилище. «Действие» вызывает «Диспетчера», в котором происходит обработка данных. Данные изменяются и кладутся в хранилище, как только произошло изменение «Представление» оповещается об этом с помощью события.

«Представление» – это компонента, написанная на React, которые подписываются на события и получают из хранилища состояние приложения, связанные с этим компонентом, каждый раз, когда оно обновляется.

Стоит отметить, что подпоясывание на «Хранилище» происходит в главном представлении, а далее данные передаются по свойствам дочерних «представлений».

Рисунок, с «Представлением» в шаблоне Flux представлен на рисунке 1.9.

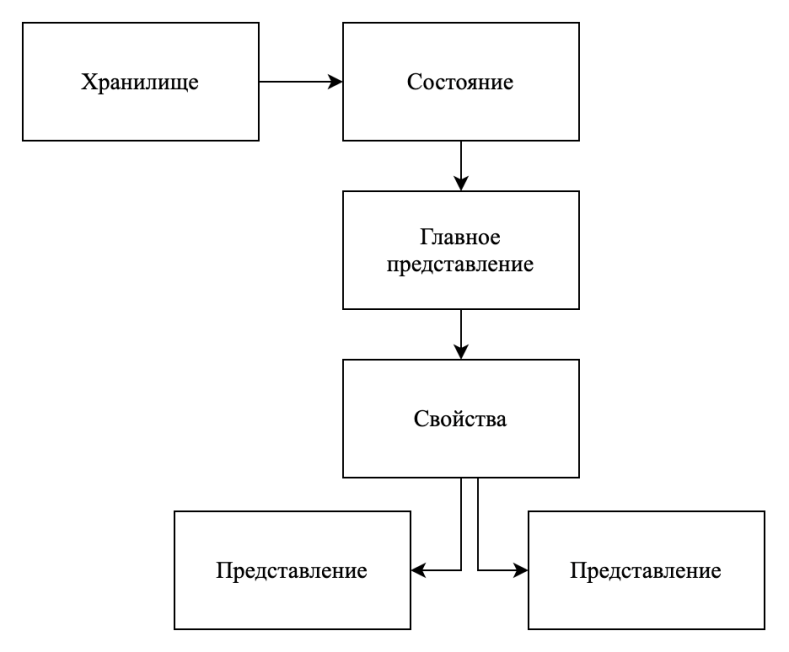


Рисунок 1.9 – Роль «Представления» во Flux

### 1.2.5 Обзор систем управления базами данных

*MySQL* – был запущен еще в 1995 году. С тех пор он претерпел несколько изменений в владельце / руководстве, прежде чем в 2010 году перешел в корпорацию Oracle. Хотя сейчас за это отвечает Oracle, MySQL по-прежнему является программным обеспечением с открытым исходным кодом, что означает, что вы можете свободно использовать и изменять его.

Плюсы реляционной базы данных:

* удобный инструмент для работы приложений в реальном времени;
* отлично подходит для обработки больших объемов данных.

Минусы реляционной базы данных:

* медленный доступ к данным;
* трудоемкость разработки.

Мы используем MySQL для внутренних операций, так как мы являемся разработчиком программного обеспечения. Мы используем этот инструмент в нашем серверном приложении для получения данных. В основном мы используем MySQL для нашей Системы управления реляционными базами данных (СУБД). Мы храним данные наших клиентов определенным образом. Мы должны создать различные уровни схемы. Мы предоставляем данные клиента, такие как имя, номер мобильного телефона, что было куплено и что им нравится или не нравится. Как только мы создадим схему, набор данных, нам нужно сохранить уникальные ограничения. Например, для клиентов мы присвоим каждой записи уникальный идентификатор, который будет первичным ключом. Все эти ограничения SQL мы будем использовать, и не должно быть никаких отклонений или репликации изолированных данных. Когда мы выполняем какие-либо операции обновления, это не должно влиять на какие-либо изолированные данные. Нам нужно позаботиться о многих вещах, таких как согласованность, изоляция и изменчивость при создании любой базы данных. Есть также еще несколько концепций, таких как нормализация и PL / SQL. Например, если я хочу запустить задание в определенное время, продукт будет использовать PL / SQL. Мы как бы внедряем наши навыки кодирования в базы данных. Наши операции осуществляются на месте. Тем не менее, мы находимся в процессе переноса всего в облако.