10.5281/zenodo.11108185

ШОГЕНОВ Алим Музаринович
эксперт в области разработки программного обеспечения, инженер-разработчик,
Туту.ру, Россия, г. Москва

КОМПЛЕКСНЫЙ АНАЛИЗ ФРЕЙМВОРКА РАЗРАБОТКИ REACT-REDUX

Аннотация. *Создание научной статьи рассматривает применение фреймворка React-Redux для разработки кроссплатформенных веб-приложений с целью обеспечения высокой производительности и удобства использования. Исследование фокусируется на оценке удовлетворенности пользователей, удобстве формирования пользовательского интерфейса и процессе создания простых и сложных приложений с использованием React-Redux. Рассмотрены преимущества данного подхода, такие как централизованное хранение данных, управление состоянием приложения, и возможность создания эффективных и масштабируемых приложений для платформ iOS и Android. Также проведено сравнение React-Redux с альтернативными фреймворками для кроссплатформенной разработки, подчеркивая его преимущества в обработке больших объемов данных, управлении состоянием в сложных приложениях и создании масштабируемых приложений.*

Ключевые слова: *кроссплатформенные веб-приложения, фреймворки, React-Redux, управление состоянием приложения, масштабируемость, производительность, сравнение фреймворков, пользовательский интерфейс, разработка приложений для Android и iOS.*

Введение

Разработка веб-приложений часто представляет собой сложную задачу для разработчиков, особенно когда требуется создание приложений как для платформы Android, так и для IOS [1]. Это приводит к необходимости знания двух разных стеков технологий и созданию отдельных приложений под каждую платформу. Однако некоторые существующие гибридные веб-приложения, несмотря на свою универсальность, не всегда обеспечивают аналогичный пользовательский опыт на разных нативных платформах. В данном исследовании рассматривается подход к решению этой проблемы с использованием фреймворка React-Redux [2], который позволяет создавать кроссплатформенные веб-приложения с высокой скоростью получения данных без кэширования.

React, разработанный компанией Facebook в 2015 году, обеспечивает возможность написания кода на JavaScript ES6 и создания мобильных приложений для iOS и Android с использованием единого кодовой базы [3]. Для обработки сложных приложений React использует дополнительную зависимость под названием Redux, предназначенную для управления состоянием в приложениях React Native [4]. В данном исследовании также рассматриваются другие зависимости, разработанные сообществом React Native, такие как базы данных и пользовательский интерфейс.

Цели научной работы:

- Оценить удовлетворенность пользователями веб-приложением, разработанным с использованием фреймворка React-Redux.
- Проверить удобство формирования пользовательского интерфейса по сравнению с альтернативными платформами.
- Сравнить процесс создания как простых, так и сложных приложений с использованием React-Redux.

Приложение, созданное с применением фреймворка React-Redux, обладает высокой универсальностью и способно работать на операционных системах iOS и Android. Это обеспечивает пользователям широкий функционал и удовлетворяет самые разнообразные потребности. Одним из ключевых преимуществ такого подхода является возможность централизованного хранения данных на сервере, что обеспечивает их сохранность и целостность.

Тем не менее, благодаря использованию Redux, часть данных может также храниться на стороне клиента, что способствует их более быстрой обработке и доступу к ним. Такой гибкий механизм управления данными позволяет оптимизировать процессы взаимодействия с приложением и повышает его производительность. Важно отметить, что наличие данных на клиентской стороне также способствует снижению нагрузки на сервер и улучшению отзывчивости приложения. Таким образом, использование React-Redux открывает широкие возможности для создания эффективных и масштабируемых приложений, способных эффективно удовлетворять потребности пользователей.

Тенденции в использовании React-Redux

С развитием технологий во всем мире многие организации стараются использовать эффективные и быстрые фреймворки пользовательского интерфейса. Redux – это библиотека, которая может быть использована с любым фреймворком пользовательского интерфейса, включая React, Angular, Vue и чистый JavaScript. Вопреки тому, что Redux может работать с различными фреймворками, наиболее часто он используется вместе с React, хотя они взаимно независимы.

При использовании Redux с другими фреймворками пользовательского интерфейса обычно применяются библиотеки привязки пользовательского интерфейса или подключения, чтобы связать Redux с выбранным фреймворком, вместо прямого взаимодействия с хранилищем из кода пользовательского интерфейса. React-Redux представляет собой первоначальную библиотеку привязки пользовательского интерфейса для React, что снижает использование привязочного слоя и обеспечивает прямое взаимодействие.

Необходимость применения React-Redux в приложении

1. React-Redux – официальная привязка React к Redux. Она позволяет компонентам React читать данные из хранилища Redux и отправлять действия в центральное хранилище данных для их обновления. Разработчик может легко управлять состоянием приложения с помощью функции глобального доступа.

2. Когда происходит изменение состояния, это влияет на дочерние компоненты, что в конечном итоге отражается на производительности. Однако библиотека Redux централизует управление состоянием приложения. Это позволяет разработчику использовать значимые функции разработки, такие как отмена/возврат, сохранение состояния и многое другое.

3. В React очень сложно отслеживать состояние приложения в процессе отладки. Но Redux предоставляет отличный опыт разработчика, который позволяет "отладку во времени", а также отправлять полные отчеты об ошибках на сервер.

4. React имеет сложный пользовательский интерфейс, где поток данных может быть сложным, когда мы используем больше компонентов для совместного использования одних и тех же данных. Однако Redux гибок с любыми уровнями пользовательского интерфейса и имеет большую экосистему дополнений, чтобы соответствовать вашим требованиям.

5. В React очень сложно повторно использовать компоненты, потому что он тесно связан с корневым компонентом. Redux упрощает эту сложность и обеспечивает глобальную доступность, что помогает создавать приложения, которые могут часто использоваться для проверки и запуска в различных средах (клиентская, серверная и мобильная).

Формулировка проблемы

В современной разработке пользовательского интерфейса (UI) столкнулись с трудностями при масштабировании приложений, особенно в случае использования библиотек, таких как React и Angular. Несмотря на то, что эти библиотеки обладают внутренним механизмом управления состоянием компонентов, они иногда оказываются недостаточно эффективными при работе с крупными и сложными проектами. В небольших приложениях управление состоянием внутри каждого компонента может быть довольно простым и интуитивно понятным. Однако, по мере роста размеров приложения, возникают сложности в управлении состояниями, доступ к которым требуется из различных компонентов приложения.

Одной из основных проблем становится управление глобальным состоянием приложения. В масштабируемом приложении может быть множество компонентов, которые должны иметь доступ к общим данным или состояниям. Без правильной организации и инструментов для управления этими общими

состояниями разработка и поддержка приложения может стать сложной задачей.

Таким образом, необходимо найти эффективные подходы и инструменты для управления состояниями в масштабируемых приложениях, чтобы обеспечить легкость разработки, поддержки и масштабирования приложений на основе современных библиотек и фреймворков пользовательского интерфейса.

Архитектура React-Redux

Система React-Redux представляет собой структуру, разработанную для обеспечения устойчивости веб-приложений. Она интегрирует данные, хранящиеся в хранилище, с пользовательским интерфейсом [5].

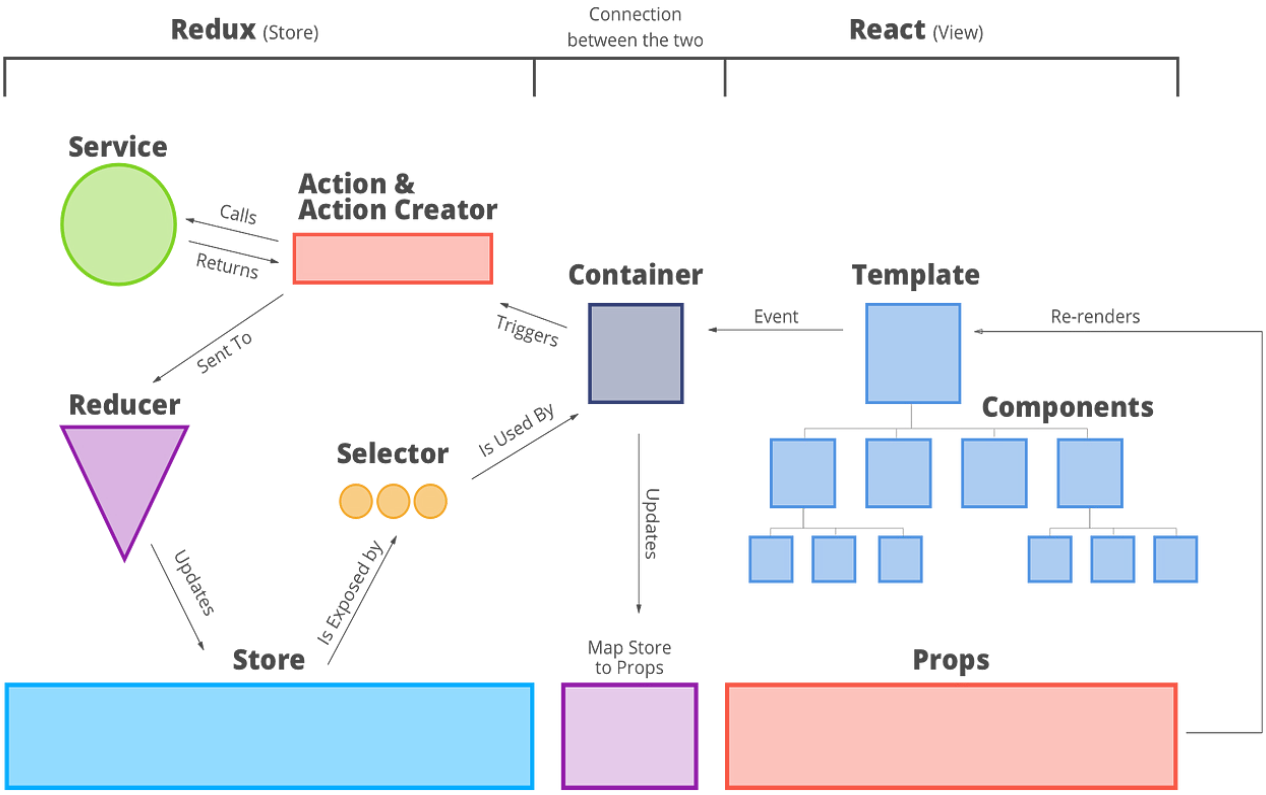

Рис. 1. Архитектура типичного фреймворка React-Redux

Архитектура React-Redux, изображенная на рисунке 1, включает следующие ключевые блоки:

- **Компонент (Component)** ответственен за поддержку метаданных и предоставление доступа к ним другим сервисам. Компонент может быть шаблоном, контейнером и т.д. Это набор кода, который использует свойства и может вызывать другие компоненты, передавая им свойства. Компонент начинается с API.
- **Шаблон (Template)** отвечает за перемещение данных из источников в цели. Он дополняет компоненты определенными свойствами и может предоставить любое название, но передает только одно название для каждого объекта.
- **Контейнер (Container)** является мостом, соединяющим React с Redux. Он подключает их во многих аспектах, используя модуль React-Redux, обычно называемый connect. Контейнер принимает три аргумента: объект, отображающий состояние в свойства (mapStateToProps), объект, отображающий действия на отправку (mapActionsToDispatch) и mergeProperties, который объединяет все свойства и передает их в DOM для рендеринга.
- **Экшены и Генераторы экшенов (Actions & Action Creators)** предоставляют платформу для вычисления и выполнения сервисов. Экшен представляет собой объект, содержащий тип экшена и состояние, измененное в результате действия. Генератор экшена – это код, создающий экшен и отправляющий его в редюсер.
- **Редюсер (Reducer)** создает отображения между источником и целью. Для каждого

действия каждый редюсер вызывается и получает переданное действие, после чего должен обработать его или передать дальше.

- **Хранилище (Store)** отслеживает выполнение рабочих процессов. Оно содержит все состояния приложения в виде дерева и позволяет изменять состояние внутри хранилища через диспетчеризацию действия на него. Хранилище не является классом, а представляет собой объект с несколькими методами.
- **Селектор (Selector)** определяет, как получать данные из хранилища в контейнере. Это функция, принимающая состояние Redux в качестве аргумента и возвращающая данные, полученные из этого состояния. Селекторы обеспечивают оптимизацию производительности и инкапсулируют глобальное дерево состояний.

Соответствующее исследование

Большое количество статей и учебников посвящены темам разработки веб-приложений для нативных платформ. Кроссплатформенная разработка упоминается хотя бы в некоторых из них, хотя не всегда в качестве основной темы. Чтобы пролить свет на смежную литературу, мы рассмотрим работы, сравнивающие несколько фреймворков для кроссплатформенной разработки приложений. Веб-технологии предоставляют один из способов создания приложений, охватывающих разные платформы. Нативные приложения также полезны для сравнения кроссплатформенных подходов, особенно в отношении их внешнего вида и производительности. Очень популярным способом управления состоянием в кроссплатформенном приложении является использование Angular с хранилищем NGRX или React с хранилищем Mobx. Оба фреймворка хорошо установлены и проверены временем.

Чтобы обеспечить максимально объективное сравнение и оценку фреймворков, были выбраны только те, которые используют JavaScript в качестве основного языка программирования. Для промышленного применения обычно какой-то из парадигм исключается из-за множества предварительных условий. С появлением новых фреймворков выбор технологии не так прост. Из-за возможных различий во внутренних механизмах фреймворков, использующих другие языки, были исключены те, что используют что-то, кроме JavaScript. Было выбрано три фреймворка для кроссплатформенной разработки из-за их новизны, актуальности и (предполагаемого) интереса разработчиков. Каждый из этих фреймворков имеет свои преимущества и недостатки, и выбор конкретного фреймворка зависит от потребностей пользователя и конкретной задачи приложения.

Применение React-Redux

С появлением новых технологий происходит быстрое модернизирование, что приводит к увеличению объема данных на пользовательском интерфейсе. Многие компании сейчас внедряют этот фреймворк, чтобы помочь с различными приложениями бизнеса.

1. Обработка больших объемов данных на фронтенде:

Redux широко используется там, где значительное количество данных изменяется со временем. Применение Redux рекомендуется, когда состояния данных изменяются часто и требуется управление состоянием приложения. Это считается лучшей практикой для обработки изменений в интерфейсе пользователя и поддержания его актуальности.

2. Управление состоянием в сложных приложениях:

Некоторые библиотеки или технологии фронтенда, такие как React, имеют собственные средства управления состоянием приложения. Однако, с ростом сложности приложений становится сложно отслеживать и управлять состоянием только средствами React. В таких случаях Redux становится полезным инструментом, позволяющим более эффективно управлять состоянием приложения и упрощать его разработку и отладку [7].

3. Создание масштабируемых приложений:

Приложения могут увеличиваться в размерах со временем, и при использовании Redux мы можем легко масштабировать приложение вверх или вниз. Redux обеспечивает эффективное управление состоянием приложения независимо от его размеров, сохраняя при этом производительность системы даже при увеличении объема данных.

Для наглядного представления применения React-Redux, давайте рассмотрим пример кода с его использованием:

```
import React from 'react';
import { connect } from 'react-redux';
import { incrementCounter, decrementCounter } from './actions';

const Counter = ({ count, increment, decrement }) => {
return (
<div>
<h2>Counter: {count}</h2>
<button onClick={increment}>Increment</button>
<button onClick={decrement}>Decrement</button>
</div>
);
};

const mapStateToProps = (state) => {
return {
count: state.counter
};
};

const mapDispatchToProps = (dispatch) => {
return {
increment: () => dispatch(incrementCounter()),
decrement: () => dispatch(decrementCounter())
};
};

export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

Рис. 2.

Перспективы развития

Улучшение производительности и оптимизация ресурсов: необходимо сосредоточиться на повышении производительности приложений и оптимизации использования ресурсов.

Расширение функциональности: развитие новых возможностей для улучшения гибкости и адаптивности React-Redux к различным требованиям проектов.

Поддержка современных стандартов и технологий: важно оставаться согласованным с последними трендами и стандартами разработки, обеспечивая совместимость с новыми технологиями.

Расширение экосистемы и поддержка сообщества: продолжение развития экосистемы инструментов и активная поддержка сообщества разработчиков для обмена опытом и решения общих проблем.

Адаптация к изменяющимся потребностям пользователей: обеспечение адаптивности к требованиям рынка и запросам пользователей для улучшения пользовательского опыта и удовлетворения клиентских потребностей.

Заключение

На основании представленного исследования можно сделать вывод, что применение фреймворка React-Redux в разработке кросс-платформенных веб-приложений обладает значительными преимуществами. React-Redux обеспечивает эффективное управление состоянием приложения, обработку больших объемов данных и создание масштабируемых приложений для различных платформ. Он также позволяет разработчикам упростить процесс создания и отладки приложений, что способствует повышению производительности и удовлетворенности пользователей. Вместе с тем, следует отметить, что развитие React-Redux требует постоянного внимания к улучшению производительности, расширению функциональности и адаптации к изменяющимся потребностям пользователей. Поддержка современных стандартов и технологий, расширение экосистемы и активная поддержка сообщества

разработчиков играют важную роль в дальнейшем развитии и успешном применении React-Redux в разработке веб-приложений.

Литература

1. The State of Mobile Development: iOS vs Android – https://themanifest.com/app-development/mobile-development-ios-vs-android.
2. Redux: A Predictable State Container for JavaScript Apps – https://redux.js.org/.
3. Facebook. (2015). React: A JavaScript library for building user interfaces – https://reactjs.org/.
4. React Native: A framework for building native apps using React – https://reactnative.dev/.
5. React-redux на React Hooks и React Context – https://vadim-budarin.medium.com/redux-на-react-hooks-react-context-ad673192309b.
6. My Awesome React/Redux Structure – https://betterprogramming.pub/my-awesome-react-redux-structure-6044e5007e22.

SHOGENOV Alim Muzarinovich
Expert in Software Engineering, Software Engineer, Tutu.ru,
Russia, Moscow

COMPREHENSIVE ANALYSIS OF THE REACT-REDUX DEVELOPMENT FRAMEWORK

Abstract. *This scientific article examines the application of the React-Redux framework for developing cross-platform web applications with the aim of ensuring high performance and user convenience. The study focuses on assessing user satisfaction, ease of user interface creation, and the process of developing both simple and complex applications using React-Redux. The advantages of this approach are discussed, such as centralized data storage, application state management, and the ability to create efficient and scalable applications for iOS and Android platforms. A comparison of React-Redux with alternative frameworks for cross-platform development is also conducted, highlighting its advantages in handling large volumes of data, managing state in complex applications, and creating scalable applications.*

Keywords: *cross-platform web applications, frameworks, React-Redux, application state management, scalability, performance, framework comparison, user interface, Android and iOS app development.*