**УДК 004.415**

# РАЗРАБОТКА МЕНЕДЖЕРА ЗАКЛАДОК С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК REACT И EXPRESS

**Артемов А.В.[1], Гришунов С.С. [1], Черепков Е.А. [1], Белов Ю.С.[1]**

*[1]Калужский филиал ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Калуга, e-mail: fn1-kf@mail.ru*

**Данная статья посвящена использованию библиотек React и Express в рамках разработки веб приложения менеджер закладок. Данная задача состоит из двух основных этапов. Разработка серверной и клиентской части. Node.js является хорошим выбором для разработки серверной части приложения, поскольку JavaScript код, выполняемый в среде Node.js, может быть в два раза быстрее, чем код, написанный на компилируемых языках, а так же в браузере и на сервере используются одинаковые концепции языка. Также данная платформа очень привлекательна благодаря простоте и удобству работы с менеджером пакетов для Node.js, который называется npm. React для клиентской части позволяет использовать JSX – язык программирования с близким к HTML синтаксисом, который компилируется в JavaScript, а также обладает высокой производительностью за счет использования Virtual DOM, и высокопроизводительного движка определения изменений в DOM – Fiber. А использование Redux позволяет упростить управление состоянием на клиенте. Таким образом, в данной статье будет подробно описано, каким образом можно использовать все вышеперечисленные инструменты для эффективной разработки веб приложения.**

Ключевые слова: разработка веб-приложения, javascript, react, redux, nodejs, express

# DEVELOPMENT OF BOOKMARK MANAGER USING REACT AND EXPRESS LIBRARIES

**Artemov A.V.[1], Grishunov S.S. [1], Cherepkov E.A. [1], Belov Yu.S.[1]**

*[1]Bauman Moscow State Technical University (Kaluga Branch), Калуга, e-mail: fn1-kf@mail.ru*

**This article focuses on the use of React and Express libraries in the development of a web application bookmark manager. This task consists of two main stages. Development of server and client side. Node.js is a good choice for developing the server side of the application, since the JavaScript code executed in the Node.js environment can be two times faster than the code written in compiled languages, and the same language concepts are used in the browser and on the server. In addition, this platform is very attractive due to the simplicity and convenience of working with the package manager for Node.js, which is called npm. React for the client side allows you to use JSX - a programming language with a close to HTML syntax that is compiled into JavaScript, and has high performance using Virtual DOM, and a high-performance change detection engine in DOM - Fiber. Moreover, the use of Redux allows you to simplify state management on the client. Thus, this article will describe in detail how all of the above tools can be used to effectively develop a web application.**

Keywords: web application development, javascript, react, redux, nodejs, express

**Введение.** В настоящее время в интернете люди сталкиваются с огромным количеством информации, которую необходимо сохранять и систематизировать.

Создание веб-сайта для сохранения ссылок на веб-страницы позволит пользователям сохранять, управлять, синхронизировать, а также получать быстрый доступ к большому количеству полюбившихся вкладок в любое время и в любом месте, поскольку веб-приложение является кроссплатформенным и может использоваться даже на мобильных телефонах.

Для организации максимально удобной работы следует придерживаться следующих принципов:

- Классификация закладок. Данный принцип означает, что все закладки должны быть в папках. Если часть закладок находится вне категории, она должна находиться в специальной папке типа «Без категории».
- Папки должны иметь несколько уровней.
- Работа из нескольких мест. Должна быть возможность использовать сервис на различных устройствах (браузер, десктопное приложение, мобильное приложение) а также взаимодействие через расширение браузера.
- Поиск закладок. Должна быть возможность поиска закладок по различным критериям с использованием сортировок. Среди возможных вариантов: поиск по названию, тегам, сортировка по категориям, дате добавления, названию.
- Бэкапы закладок (возможность импорта/экспорта). Бэкапить можно 2 путями — или в виде локального файла, или в облаке.

**Средства разработки.** В настоящее время одно из наиболее популярных решений для разработки клиентской части веб-приложения – библиотека React, разработанная Facebook. В отличие от всех остальных решений, это самое гибкое, так как оно отвечает только за отображения, а все остальные инструменты позволено выбрать разработчику.

Одной из ее отличительных особенностей является возможность использовать JSX – HTML подобный язык программирования, трансформируемый в чистый Javascript на этапе сборки. Также важным преимуществом является высокая производительность за счет использования Virtual DOM, и высокопроизводительного движка определения изменений в DOM – Fiber, что снижает вероятность возникновения возможных неудобств и улучшает пользовательский опыт. Исходя из всего выше перечисленного будет использоваться именно React.

Для разработки серверной части приложения использовался Node.js [1]. Node.js — это кроссплатформенная среда выполнения JavaScript кода за пределами браузера, с открытым кодом [2,3]. Node.js позволяет писать серверную часть на JavaScript. Это означает что можно писать, как клиентский, так и серверный код на одном и том же языке программирования, что открывает новые возможности и значительно упрощает начало разработки серверной части для фронтенд разработчиков.

Одним из наиболее популярных фреймворков для разработки серверной части на Node.js является Express. Он предоставляет ряд готовых абстракций, которые упрощают создание сервера и серверной логики. Очень удобным является подход с использованием промежуточных обработчиков, как общих для всех запросов на сервер, так и частных для запросов на определенные адреса. Express позволяет очень просто и удобно настроить роутинг, добавлять необходимые заголовки, настроить кросдоменные запросы и так далее.

**Разработка приложения.** Для начала работы необходимо иметь установленный Node.js, его можно найти на официальном сайте nodejs.org, а также пакетный менеджер yarn (yarnpkg.com) а также PostgreSQL.

Первый пакет, который необходимо установить – express, сам фреймворк для создания сервера. Для работы с post-запросами необходимо использовать пакет body-parser, который обрабатывает тело запроса и добавляет поле body в объект запроса. Для возможности совершать с запросы с других доменов используется пакет cors, используемый как промежуточный обработчик.

Для возможности при написании кода использовать стандарт es6, с последующей транспиляцией в код, поддерживаемый большинством окружений, используется компилятор babel. Babel позволяет использовать различные пресеты (набор трансформаций, применяемых к коду при транспиляции). Стандартный набор трансформаций для поддержки самыми распространенными окружениями содержится в babel-preset-env, а @babel/polyfill добавляет методы `Array.from`, `String.prototype.repeat` и другие. Один из вариантов настройки babel – создание файла .babelrc в директории проекта.

*Подключение базы данных.* Для работы с базой данных будет использоваться ORM Sequelize, что позволяет абстрагироваться от способа хранения данных и отказаться от написания sql кода для взаимодействия с базой данных [4].

Создадим конфигурационной файл с указанием директорий, используемых sequelize .sequelizerc и поместим его в корневую директорию.

В конфигурационном файле необходимо указать: config.json – файл, содержащий данные, необходимые для подключения к базе данных, models – директория, содержащая определения моделей, migrations – содержит миграции для базы данных, seeders – содержит данные для инициализации базы данных.

После чего необходимо использовать команду sequelize init, которая сгенерирует все необходимые файлы и папки. Все, что остается сделать – создать базу данных и изменить config.json.

Создать базу можно при помощи команды cretedb, доступной после установки PostgreSQL.

В режиме разработки будет использоваться указанная в блоке development база данных, в то время как в production данные для подключения к базе будут браться из переменной среды.

*Проектирование базы данных.* База данных будет содержать следующие таблицы:

- Закладки
- Коллекции

– Теги
– Пользователи

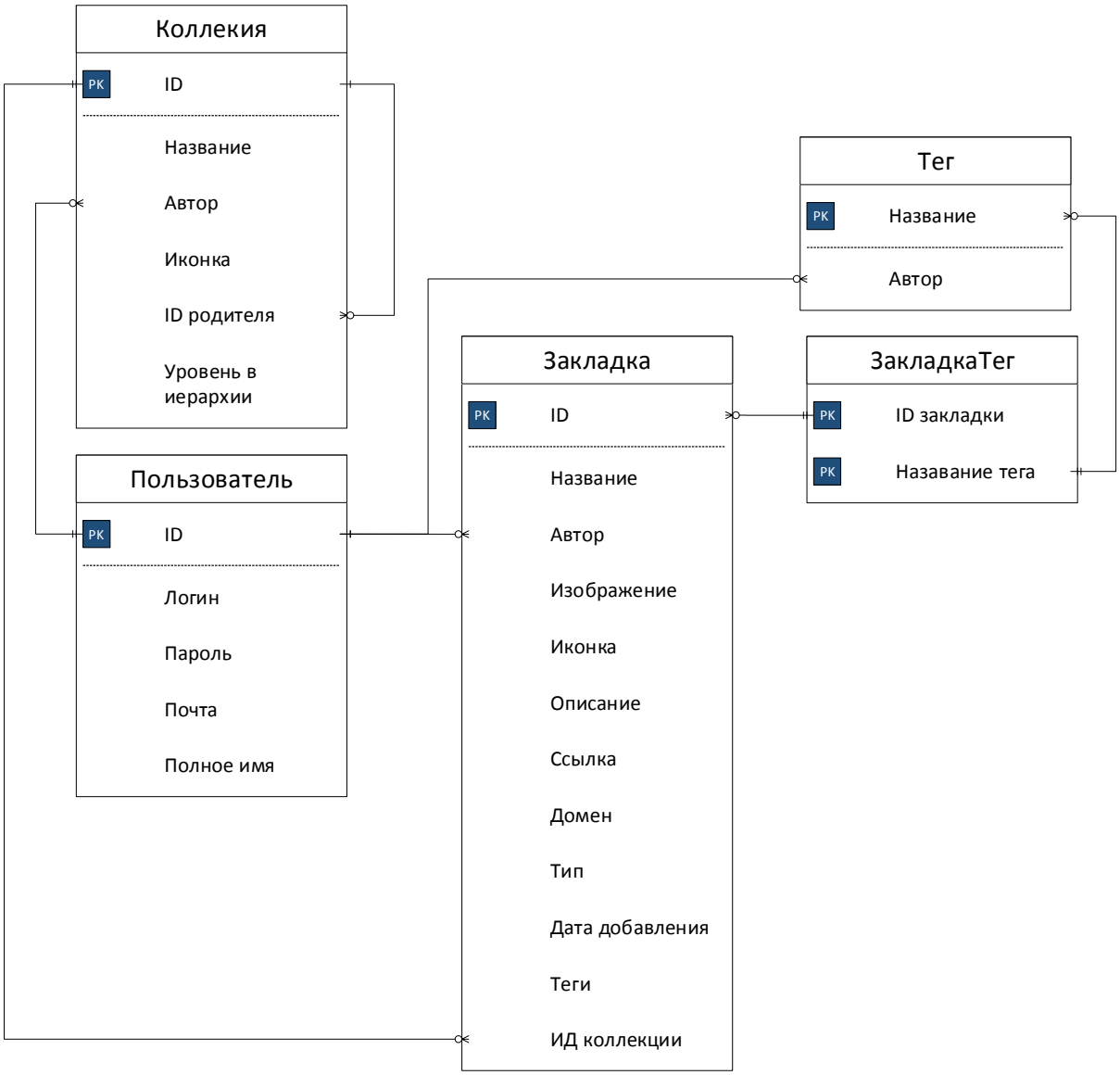


Рис. 1 – Структура базы данных

*Создание моделей.* Для генерации моделей используется команда `sequelize model:create --name Collection -attributes name:string, icon:string, author:string`, в которой указывается название таблицы и перечисляются поля.

Отношения между таблицами задаются следующим образом:

```
Collection.associate = models => {
   Collection.hasMany(models.Bookmark, {
     foreignKey: 'collectionId',
     as: 'bookmarks',
   });
}
```

Для создания иерархической структуры можно использовать пакет sequelize-hierarchy, который добавляет поле, указывающее на родительский элемент, а также уровень в иерархии.

Также для автоматического создания миграций базы данных используется пакет sequelize-auto-migrations, который создает миграции на основе имеющихся моделей и предыдущего состояния базы.

Для применения к базе сделанных изменений необходимо выполнить миграции sequelize db:migrate, после чего в базе появятся таблицы для созданных моделей.

*Реализация логики взаимодействия с сервером.*

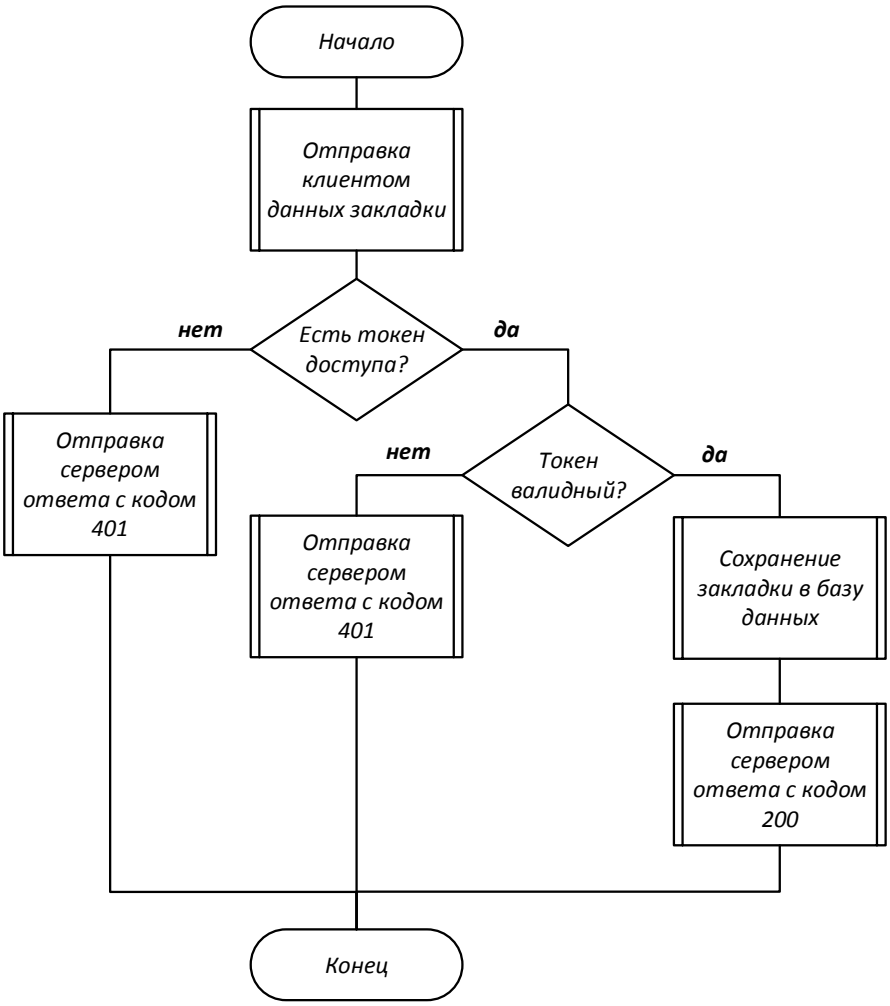


Рис. 2 - Алгоритм добавления закладки:

*Клиентская часть.* Пользовательский интерфейс приложения формируется библиотекой React. Благодаря ей создаются все базовые компоненты системы и происходит разбиение представления на части [5]. Для ускорения разработки используется библиотека компонентов material-ui, содержащая, большую часть готовых компонентов с базовой стилизацией и реализующих удобное взаимодействие с ними не только с помощью мыши, но и с помощью клавиатуры. Основные используемые компоненты material-ui.

На клиенте для управления состоянием используется библиотека Redux. По мере того как требования к одностраничным приложениям JavaScript становятся все более высокими, в состоянии хранится все больше данных. Это состояние может включать ответы сервера и кэшированные данные, а также локально созданные данные, которые еще не были сохранены на сервере. Состояние пользовательского интерфейса также усложняется. Управлять постоянно изменяющимися состояниями сложно. По мере роста приложения, добавление нового функционала становится сложнее, а взаимодействие с существующим функционалом менее предсказуемым. Когда система становится непрозрачной и недетерминированной, трудно выявлять ошибки или добавлять новый функционал.

Redux используется для упрощения управления состоянием на клиенте. Фундаментальные принципы Redux:

- единственный источник правды. Принцип требует хранить все состояние предложения в дереве объектов внутри одного хранилища. Благодаря этому сильно упрощается отладка приложения и ускоряется разработка;
- состояние только для чтения. Единственный способ изменить состояние — это применить действие — объект, который описывает, что случится. Это гарантирует, что представления или функции, реагирующие на события сети не смогут изменить состояние приложения.
- изменения описаны как чистые функции. Для определения того, как дерево состояния будет трансформировано действиями, нужно вызвать чистую функцию. Это гарантирует прозрачность и простоту получения нового состояния.

Исходя из принципов Redux архитектура клиентского приложения выглядит следующим образом:

- actions. Структурный элемент, в котором описано, какие действия могут происходить в приложении. Они могут содержать логику работы с сервером или другие асинхронные действия. Благодаря им очень легко определить, что в приложении может вызывать изменение состояния.
- reducers. В редьюсерах содержится логика изменения состояния. Они представляют собой функции, вызывающиеся при срабатывании действий. Все обработки происходят без сайд эффектов.
- middlewares. Структурный элемент – функция, которая вызывается при срабатывании любого действия, необходима для дополнительной обработки действия и добавления ему дополнительных данных, например, состояния запроса.
- services. Сервисы представляют собой независимые от архитектуры приложения блоки, выполняющие определенную работу. Основное предназначение -

абстрагирование взаимодействия с сетью, локальным хранилищем, определенной работы с DOM.

- utils. Утилиты – набор вспомогательных функций, использующихся во всех других структурных компонентах с целью избегания дублирования кода и отделения сложной логики обработки данных [6].

**Заключение.** Таким образом, в данной статье было рассмотрено использование открытых библиотек для разработки приложения менеджер закладок. React для клиентской части и Express для серверной части. Данные библиотеки отлично подходят для данной задачи и позволяют быстро и качественно разрабатывать веб-приложения.

**Список использованной литературы:**

1. Mark A. Holliday ; Andrew S. Scott A software development course based on server-side Javascript // IEEE Frontiers in Education Conference (FIE). 2016. P.1-5.

2. Долгов А.Н., Нуруллин Р.Ю. Программная платформа NODE.JS // Достижения науки и образования. 2016. № 12 (13). С. 17-18.

3. Stefan Tilkov ; Steve Vinoski Node.js: Using JavaScript to Build High-Performance Network Programs // 2010, Vol. 14 , №6. P.80-83.

4. Токарчук А.М. Применение средств ORM для разработки безопасных веб-приложений // Безопасность информационных технологий. 2010. Т. 17. № 1. С. 113-115.

5. Рыбакова А.С. Разработка списка дел на Javascript с помощью библиотеки React // Новые информационные технологии в научных исследованиях материалы XXI Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. Рязанский государственный радиотехнический университет. 2016. С. 160.

6. Попков И.В., Курзаева Л.В. Использование библиотеки redux для разработки веб-приложений // Аллея науки. 2018. Т. 1. № 7 (23). С. 928-930.