

Д.С. Бузин, М.Т. Азизов

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРЫ ДЛЯ СЕРВИСА ПО УПРАВЛЕНИЮ ИНТЕРАКТИВНЫМИ ПОДПИСКАМИ

Аннотация. В исследовании проводится комплексный анализ процесса создания web-приложения для контроля приобретенных подписок. Рассмотрены и изучены существующие решения Billy, Outflow, SubscriptMe, Apple Subscriptions, Google Subscriptions. а также разработана клиентская часть приложения с помощью библиотеки React.

Ключевые слова: клиент-серверная архитектура, интерактивные подписки, сервер.

D.S. Buzin, M.T. Azizov

DEVELOPMENT OF A CLIENT-SERVER ARCHITECTURE FOR A SERVICE FOR MANAGING INTERACTIVE SUBSCRIPTIONS

Abstract. The study provides a comprehensive analysis of the process of creating a web application to control purchased subscriptions. Reviewed and studied existing solutions Billy, Outflow, Subscript Me, Apple Subscriptions, Google Subscriptions. and also developed the client side of the application using the React library.

Keywords: client-server architecture, interactive subscriptions, server.

Введение

В наше передовое время существует множество возможностей для предоставления цифровых услуг. Одни компании предлагают их пользователю на бесплатной основе, другие – на платной. Платные варианты приобретения возможности пользования цифровым продуктом также бывают разными. Сегодня одним из популярных вариантов использования различных сервисов является приобретение их по подписке. Главным достоинством данного способа является его низкая стоимость и разделенная оплата по месяцам. Пользователи по всему миру считают такой способ использования цифровых сервисов наиболее удобным и менее затратным.

Соответственно, при наличии некоторого количества таких подписок пользователи сталкиваются с проблемой их слежения. Чтобы решить данную проблему, удобно иметь сервис, в котором можно легко наблюдать за ежемесячной потраченной суммой, датами оплат и количеством индивидуальных подписок на различные сервисы.

На данный момент существующих решений для слежения за приобретенными подписками крайне мало, и почти у каждого есть свои недостатки в реализации.

Пользователи не могут выбрать, какой сервис использовать, потому что некоторые не обладают определенной платформой, другие не готовы платить дополнительные деньги для расширения функционала данных сервисов и др.

Бузин Дмитрий Сергеевич

магистрант кафедры информационной безопасности. Российский технологический университет «МИРЭА», Москва. Сфера научных интересов: информационная безопасность, информатика и вычислительная техника.

Электронный адрес: Vuzin.97@mail.ru

Азизов Мукум Тимурович

магистрант кафедры информационной безопасности. Российский технологический университет «МИРЭА», Москва. Сфера научных интересов: информационная безопасность, информатика и вычислительная техника.

Электронный адрес: mukimazizov@icloud.com

Цель исследования – разработка полнофункционального сервиса, в котором пользователи получают и полный функционал, и точные данные о подписках, и удобство использования его на разных платформах в сети Интернет.

Виды сервисов и их характеристики

Billy (см. Рисунок 1). Решение предоставил разработчик из Москвы, который выпустил данное приложение в GooglePlay. Использовать его может любой желающий, имеющий девайс на операционной системе Android.



Рисунок 1. Billy [2]

Одним из ключевых недостатков данного сервиса является отсутствие уже заготовленных подписок: пользователю каждый раз приходится создавать вручную подписку и

заполнять поля информации с нуля. В бесплатной версии приложения пользователь получает ограниченный функционал и может добавить только две подписки. Также неоспоримым недостатком является то, что пользователь не может менять валюту в графе суммы платежа, а всегда должен переводить ее самостоятельно в доллары.

Outflow (см. Рисунок 2). Приложение, в первую очередь, основано на том, что получает всю информацию о подписках из базы подписок GoogleAccount, что является небезопасным, так как приложения не использует новые технологии защиты данных пользователей. Факт того, что пользователь должен иметь подписки в GoogleAccount, отбрасывает большинство потенциальных пользователей данного сервиса.

Сервис рассчитан на тех людей, которые обладают устройством на базе операционной системы iOS.

В целом приложение выглядит достаточно интуитивным и полезным, но для получения полного контроля над мониторингом своих подписок нужно платить разработчику.

	Weekly	Monthly	Yearly
	\$0.00	\$42.98	\$0.00
NETFLIX		\$11.99 MONTHLY	6 DAYS
Spotify		\$9.99 MONTHLY	15 DAYS
Dropbox		\$21.00 MONTHLY	15 DAYS

Рисунок 2. Outflow [8]

SubscriptMe (см. Рисунок 3). Данное приложение предназначено только для iOS-пользователей. При первой авторизации оно подключается к почтовому ящику пользователя и сканирует его на предмет наличия электронных писем от компаний, которые предлагают платные подписки.

Данный подход вполне можно назвать нетривиальным, но он обладает множеством недостатков в реализации. Например, приложение является наиболее уязвимым к утечке данных информации и паролей электронных ящиков пользователей. Также приложение не всегда правильно отбирает подписки из почтового ящика, и многие пользователи жалуются на то, что из всего числа подписок им отображается около 60...80 %.

Преимуществом функционала данного сервиса можно назвать то, что он предоставляет статистику самых востребованных подписок среди всех пользователей. Стоит упомянуть, что приложение полностью бесплатное и не нуждается в дополнительной оплате за использование полного функционала.

AppleSubscriptions (см. Рисунок 4). Один из самых популярных сервисов по управлению интерактивными подписками для носителей продукции Apple. В нем можно хранить данные банковской карты, оплачивать напрямую подписки и их отменять.

Разработка клиент-серверной архитектуры для сервиса по управлению ...

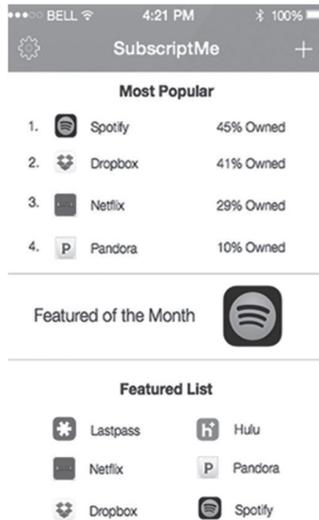


Рисунок 3. SubscriptMe [11]



Рисунок 4. AppleSubscriptions [1]

За проработанным функционалом скрываются главные недостатки – невозможность переноса данных на другую платформу и привязанность к одному аккаунту.

GoogleSubscriptions (см. Рисунок 5). Аналогично AppleSubscriptions данный сервис предоставляет многофункциональное слежение за подписками.

Оно может хранить конфиденциальные данные о пользователе, но имеет ряд недоработок, например, отсутствие уведомлений и примитивный интерфейс, где невозможно отслеживать полную сумму потраченных денег за разные промежутки времени.

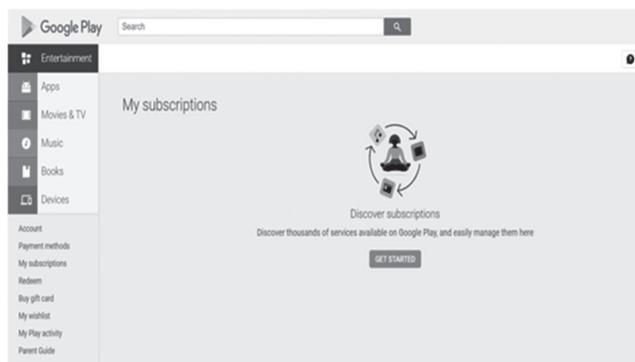


Рисунок 5. GoogleSubscriptions [4]

В каждом вышеуказанном сервисе продуманы правильные решения, но количество недостатков не дает с уверенностью выбрать один единственный для повседневного использования.

Разработка полнофункционального сервиса

В наше время существует три основных паттерна при создании архитектуры любых приложений и сервисов, предназначенных для прямого использования. Паттерны разработаны для того, чтобы упростить коммуникацию разработчикам при создании нового решения заданной проблемы проектирования с помощью заранее проработанной и повторяемой архитектурной конструкции.

После тщательного анализа готовых архитектур было принято решение использовать MVC-архитектуру.

Во-первых, в нашем решении между представлением и другими частями приложения связь невозможна. Всё должно решаться в своих же компонентах для высокоэффективной работы, что является основой модели MVC.

Во-вторых, логику приложения можно расположить там, где это будет нужно, – в контроллере или модели.

Реализация решения начинается с выбора подходящего стека технологий. Выбор готовых инструментов происходит на основе реализуемых частей приложения.

Так как решение будет реализовано на основе ранее поставленных задач, то требуется определить стек технологий для реализации каждой задачи:

- 1) выбор инструментов и реализация клиентской части приложения;
- 2) выбор инструментов и реализация серверной части приложения;
- 3) выбор инструментов и реализация базы данных;
- 4) выбор инструментов и тестирование приложения.

Сейчас используются различные полноценные библиотеки-фреймворки для создания клиентской части веб-приложений. Для решения задачи реализации клиентской части проекта была выбрана библиотека React в первую очередь за то, что польза Virtual DOM безгранична. Наше решение должно быть реактивным, и за счет того, что Virtual DOM имеет возможность увеличить производительность высоконагруженных приложений, можно снизить вероятность возникновения возможных неудобств и улучшить пользовательский опыт [10].

Во-вторых, React часто используется для создания SPA (Single Page Application). Преимущество этой библиотеки для данного вида реализации заключается в том, что все ком-

поненты страницы располагаются в понятной иерархии, и навигация по дереву компонентов не составляет труда.

Использование React и его изоморфного подхода помогает производить рендеринг всех компонентов на странице быстрее, что, в свою очередь, позволяет пользователям чувствовать себя более комфортно.

Также стоит упомянуть, что код, написанный в клиентской части приложения, можно использовать и в серверной части, тем самым не нужно дублировать один и тот же функционал. Из этого следует, что время разработки приложения снижается. Благодаря этой библиотеке можно использовать код, написанный ранее для реализации приложения на мобильных операционных системах.

Для реализации серверной части приложение будем использовать серверную среду NodeJS. Главное преимущество ее использования – код может быть написан на языке JavaScript, что заведомо уменьшает время на реализацию [3].

Выбор фреймворка для реализации веб-сервера в среде NodeJS в первую очередь зависит от конкретных специализаций приложения.

Выбор фреймворка для серверной части реализации происходил из двух кандидатов:

- Express.js;
- NestJS.

Для реализации решения был выбран Express.js за счет его легковесности и удобства использования. Этот фреймворк полноценно справится с поставленными задачами. С помощью этой технологии можно создать полноценный работающий HTTP-сервер. Также с помощью данного фреймворка можно с легкостью написать собственный Rest API.

Конкретные действия, которые выполняет сервер как связующее звено с базой данных:

- добавляет данные о подписке с помощью PUT HTTP-запроса;
- изменяет данные о подписке с помощью POST HTTP-запроса;
- удаляет данные о подписке с помощью DELETE HTTP-запроса;
- забирает данные о подписке, которые нужно визуализировать на странице подписок, с помощью GET HTTP-запроса.

Выбор инструмента для разработки базы данных состоял из двух кандидатов – MongoDB и PostgreSQL. В нашем случае выбор был сделан в пользу MongoDB.

Эту нереляционную базу данных легко можно связать с рассматриваемым приложением. Достаточно вместе с обращениями к серверу асинхронно подключать нужные коллекции из базы данных [6].

Чтобы было проще ориентироваться в базе и правильно управлять меняющимися данными, было принято решение использовать вспомогательную библиотеку Mongoose [7], которая к тому же обеспечивает проверку схемы всей базы данных и отчетливое представление объектов в MongoDB [5].

Для определения объектов в MongoDB реализуются схемы, показанные на Рисунке 6.

Приложение доступно любому пользователю в сети Интернет – любой может зайти на страницу сервиса и в качестве зарегистрированного участника начать добавлять свои подписки.

Алгоритм и функционал работы приложения. В первую очередь приложение переводит пользователя на страницу входа в свой аккаунт (см. Рисунок 7).

```

export const UserSchema = new mongoose.Schema({
  username: String,
  firstName: String,
  lastName: String,
  password: String,
});

export const SubscriptionSchema = new mongoose.Schema({
  subscriptionName: String,
  subscriptionDate: Number,
  subscriptionValue: Number,
  subscriptionAmount: Number,
  subscriptionBillingEvery: Number,
  subscriptionFrequency: String,
});

export const SubscriptionsPageSchema = new mongoose.Schema({
  subscription: [SubscriptionSchema],
  countOfSubscriptions: Number,
  sumOfSubscriptions: Number
});

```

Рисунок 6. Реализующиеся схемы для определения объектов в MongoDB

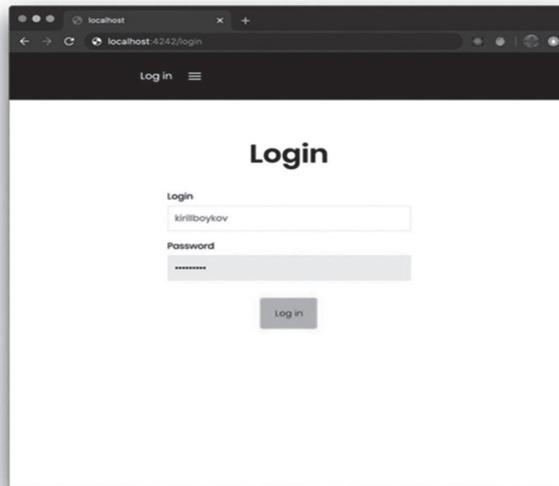


Рисунок 7. Страница входа в аккаунт

При первом входе в собственный аккаунт он может добавить подписки через всплывающее окно добавления. Для этого нужно обратиться к кнопке Addsubscription (см. Рисунок 8).

В данном окне он может начать вводить любое слово и информация о подписках в базе данных, постарается помочь ему найти совпадение. Если совпадения нет, то пользователю ничего не мешает создать собственную подписку (см. Рисунок 9).

После успешного заполнения данной формы пользователь переносится обратно на SubscriptionsPage, где можно увидеть полный список своих подписок (см. Рисунок 10).

На данном рисунке пользователь может наблюдать за изменяющимся счетчиком подписок, суммой, потраченной за этот месяц, и суммой всех денежных средств, которые были потрачены за всё время пользования сервисом.

Разработка клиент-серверной архитектуры для сервиса по управлению ...

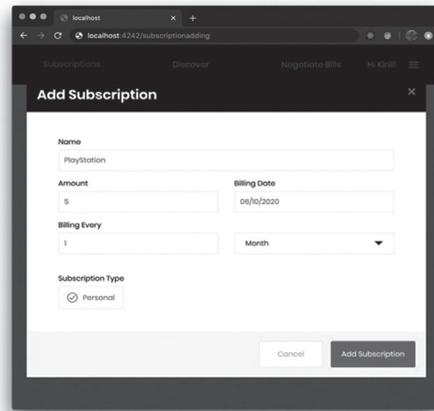


Рисунок 8. Окно добавления подписки

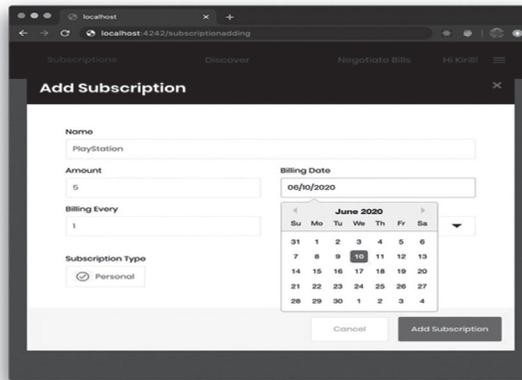


Рисунок 9. Демонстрация функционала окна

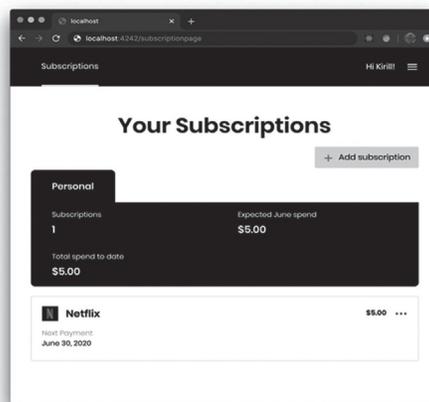


Рисунок 10. Пример отображения подписок

При необходимости пользователь может изменить данные о подписке, если, допустим, цена приобретения подписки изменилась. Для этого нужно навести курсор на «три точки», и появляется вспомогательное меню, где можно выбрать пункт Editsubscription, которое открывает окно изменения информации. Выглядит окно так же, как окно добавления подписки, и пользователю не составит труда изменить данные.

Пользователь также может изменить свои данные, нажав на иконку полки в правом верхнем углу страницы. Оно откроет SideBar, где будут предложены две вариации кнопок – Profile и Logout (см. Рисунок 11), первая из которых переносит пользователя на страницу изменения информации о себе (см. Рисунок 12).

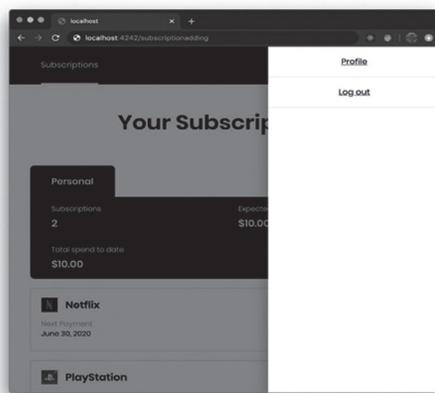


Рисунок 11. SideBar

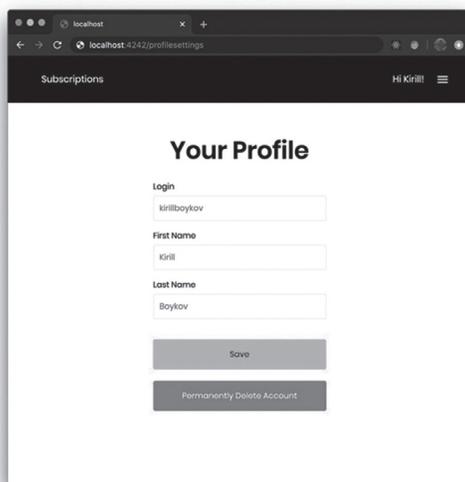


Рисунок 12. Страница изменения данных о пользователе

При нажатии на кнопку Save пользователь сохраняет новые данные, а при нажатии на кнопку Permanently Delete Account – удаляет всю информацию о себе и о подписках из базы данных.

Разработка клиент-серверной архитектуры для сервиса по управлению ...

Тестирование – неотъемлемая часть разработки такого рода приложения. Оно включает в себя следующие понятия:

- Unit-тестирование;
- интеграционное тестирование;
- тестирование end-to-end.

Тестирование будет происходить на клиентской и серверной части приложения (см. Таблицы 1, 2).

Unit-тестирование. Данный вид тестирования также называют модульным. В нашем приложении на unit-тестирование проверяются все компоненты клиентской части с помощью фреймворка Jest.

Jest работает по принципу проверки «снимков». То есть при первом запуске тестирования компонентов создается файл «снимка», а потом можно посмотреть в созданной директории snapshots, соответствует ли отрендеренный компонент ожидаемому результату.

Интеграционное тестирование. Очень важный этап тестирования всего приложения, потому что позволяет произвести тестирование логики взаимосвязей и обнаружить проблемы, возникшие на этапе объединения интегрированных модулей.

Тестирование end-to-end. Необходимо при тестировании всего приложения. Заключается оно в открытии приложения в браузере и выполнении заранее продуманных кейсов пользователя приложения. В нашем приложении это реализуется с помощью `asynс/await` тест-кейсов.

Таблица 1

Результаты тестирования клиентской части

	Unit-тестирование	Интеграционное тестирование	Тестирование end-to-end
Метрика	Количество строк кода	Покрытие actions	Корректная работа пользовательского кейса
Покрытие	87 %	86 %	100 %

Таблица 2

Результаты тестирования серверной части

	Unit-тестирование	Интеграционное тестирование	Тестирование end-to-end
Метрика	Количество строк кода	Покрытие actions	Корректная работа конечного вызова
Покрытие	93 %	83 %	100 %

Заключение

Таким образом, в ходе исследования было создано web-приложение, которое помогает пользователям упростить слежение за приобретенными подписками.

В исследовании были выполнены и полностью описаны следующие задачи:

- рассмотрены и изучены существующие решения Billy, Outflow, SubscriptMe, AppleSubscriptions, GoogleSubscriptions, их возможности в сравнении;
- разработана клиентская часть приложения с помощью библиотеки React, а также серверная часть приложения с помощью платформы Node.js и фреймворка Express.js;
- протестированы все микросервисы с помощью фреймворков Jest и Enzyme.

Литература

1. Apple Subscriptions [Электронный ресурс]. URL: <https://support.apple.com/en-us/HT202039> (дата обращения: 2.03.2022).
2. Billy [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=vmax.billy&hl=en_US (дата обращения: 1.03.2022).
3. *David Herron*. Node.js Web Development. Fourth Edition, 2018, 492 p. [Электронный ресурс]. URL: <https://expressjs.com/ru/>
4. Google Subscriptions [Электронный ресурс]. URL: <https://myaccount.google.com/payments-and-subscriptions> (дата обращения: 2.03.2022).
5. *Kristina Chodorow*. 50 Tips and Tricks for MongoDB Developers, 2011, 135 p.
6. MongoDB [Электронный ресурс]. URL: <https://www.mongodb.com/>
7. Mongoose [Электронный ресурс]. URL: <https://mongoosejs.com/docs/>
8. Outflow [Электронный ресурс]. URL: <https://apps.apple.com/in/app/outflow-subscription-manager/id975011878> (дата обращения: 1.03.2022).
9. React [Электронный ресурс]. URL: <https://ru.reactjs.org/>
10. *Robin Wieruch*. The Road to learn React, 2018, 208 p.
11. SubscriptMe [Электронный ресурс]. URL: <https://apps.apple.com/us/app/subscriptme-subscription-tracker/id963594756> (дата обращения: 1.03.2022).

References

1. Apple Subscriptions. Available at: <https://support.apple.com/en-us/HT202039> (date of the application: 02.03.22).
2. Billy. Available at: https://play.google.com/store/apps/details?id=vmax.billy&hl=en_US (date of access: 01.03.22).
3. David Herron (2018) Node.js Web Development - Fourth Edition, 492 p. Available at: <https://expressjs.com/en/>
4. Google Subscriptions. Available at: <https://myaccount.google.com/payments-and-subscriptions> (date of access: 02.03.22).
5. Kristina Chodorow (2011) 50 Tips and Tricks for MongoDB Developers, 135 p.
6. MongoDB. Available at: <https://www.mongodb.com/>
7. Mongoose. Available at: <https://mongoosejs.com/docs/>
8. Outflow. Available at: <https://apps.apple.com/in/app/outflow-subscription-manager/id975011878> (date of access: 03.01.22).
9. React. Available at: <https://ru.reactjs.org/>
10. Robin Wieruch (2018) The Road to learn React, 208 p.
11. SubscriptMe. Available at: <https://apps.apple.com/us/app/subscriptme-subscription-tracker/id963594756> (date of the application: 03.01.22).