МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО–КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМ. ПРОФЕССОРА Н.И.ЧЕРВЯКОВА

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**КУРСОВАЯ РАБОТА**

по дисциплине

«ИНТЕРНЕТ-ТЕХНОЛОГИИ»

на тему:

«Разработка сервиса бронирования отелей»

**Выполнил:**

Куделя Максим

Владимирович

Студент 4 курса

группы ПИН–б–о–17–1

Направления подготовки

09.03.03

Прикладная информатика

очной формы обучения

(подпись)

**Руководитель работы:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО, должность, кафедра)

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя) (дата)

Работа выполнена и

защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (И.О. Фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2021 г.

УТВЕРЖДАЮ

Зав. кафедрой ПИ

Азаров И.В.

Институт математики и информационных технологий им. профессора Н.И.Червякова

Кафедра прикладной информатики

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Прикладная информатика в экономике

**ЗАДАНИЕ**

**на курсовую работу**

студента Куделя Максима Владимировича

по дисциплине интернет технологии

1. Тема работы: Разработка сервиса бронирования отелей

Цель: формирование у студента навыков научно-исследовательской работы, повышение уровня его профессиональной (теоретической и практической) подготовки, углубление знаний поучебной дисциплине, развитие интереса и навыков самостоятельной работы с научной и справочной литературой.

1. Задачи:

Рассмотреть виды авторизации и аутентификации в веб;

Разработать SPA сайт;

1. Перечень подлежащих разработке вопросов:
   1. Рассмотреть виды авторизации и аутентификации в веб
   2. Разработать SPA сайт
2. Исходные данные:

Методические указания к курсовой работе по дисциплине «Интернет-технологии» для студентов направления 09.03.03 «Прикладная информатика» (профиль «Прикладная информатика в экономике»)

1. Список рекомендуемой литературы:

Методические указания к курсовой работе по дисциплине «Интернет-технологии» для студентов направления 09.03.03 «Прикладная информатика»

1. Списки рекомендуемой литературы:
2. Контрольные сроки представления отдельных разделов курсовой работы:

25 % – « » 2020 г.

50 % – « » 2020г.

75 % – « » 2020г.

100 % – « » 2020г.

1. Срок защиты студентом курсовой работы «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020г.

Дата выдачи задания «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2021г.

Руководитель курсовой работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(ученая степень, звание) (личная подпись) (инициалы, фамилия)*

Задание принял к исполнению студент очной формы обучения 4 курса группы ПИН–б–о–17–1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(личная подпись) (инициалы, фамилия)*

**Отзыв**

на курсовой проект студента 4 курса

Куделя Максима Владимировича

Актуальность: Курсовая работа посвящена разработке информационной системы. У любой современной компании существует [сайт](https://cetera.ru/webdevelopment/). Это один из элементов престижа, ведь именно в Интернете потенциальные клиенты будут в первую очередь искать информацию о фирме. И если у нее нет хотя бы одной страничка с прайсом, это покажется подозрительным – насколько же это неуспешная фирма, если не может даже небольшой веб-ресурс создать? Компания может донести информацию максимально быстро до огромного количества людей с помощью собственного сайта сделать. Веб-ресурс позволяет представить информацию о компании и ее товарах или услугах сжато и одновременно полноценно.

В первой главе рассмотрены виды авторизации и аутентификации в веб приложениях.

Во второй главе был спроектирован и разработан сервис бронирования отелей.

Проанализирована литература в объеме 7 источников.

За время работы студент проявил себя как квалифицированный специалист в области программирования, способный к самостоятельному поиску литературных источников, анализу информации и синтезу новых решений.

Таким образом, работа выполнена на высоком уровне, соответствует требованиям, предъявляемым к курсовым проектам, и заслуживает оценки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Научный руководитель:

" " 2021г.

**Содержание**

[**Содержание** 4](#_Toc69762037)

[**Введение** 5](#_Toc69762038)

[**Теоретическая часть** 6](#_Toc69762039)

[**Введение** 6](#_Toc69762040)

[**Основные понятия, используемые при изучении объекта** 7](#_Toc69762041)

[**Классификация элементов** 8](#_Toc69762042)

[**Подробная характеристика элементов** 10](#_Toc69762043)

[Аутентификация по паролю: 10](#_Toc69762044)

[Аутентификация по сертификатам: 12](#_Toc69762045)

[Аутентификация по одноразовым паролям: 13](#_Toc69762046)

[Аутентификация по ключам доступа: 14](#_Toc69762047)

[Аутентификация по токенам: 15](#_Toc69762048)

[**Заключение** 18](#_Toc69762049)

[**Практическая часть** 19](#_Toc69762050)

[**Разработка базы данных** 20](#_Toc69762051)

[**Разработка серверной части приложения** 23](#_Toc69762052)

[**Разработка SPA модуля** 26](#_Toc69762053)

[**Заключение** 36](#_Toc69762054)

[**Список литературы** 36](#_Toc69762055)

# **Введение**

С каждым годом персональные компьютеры становятся всё более производительными. В связи с этим появилась возможность переложить вычисления с сервера на клиент. Это позволит быстрее выполнять некоторые задачи и разгрузить сервер. Такие веб-приложения стали называться SPA (Single-page application).

SPA основан на том, что используется всего одна основная страница, а весь остальной контент подгружается асинхронно с сервера.

Помимо этого, требуется рассмотреть виды авторизации и аутентификации в веб приложениях для более четкого осознания принципов разработки.

Основной задачей работы является разработка SPA-приложения. Данная задача будет решена с помощью пакета прикладных программ на ПК и описана в практической части курсовой работы.

Объектом курсовой работы является – разработка SPA.

Предметом курсовой работы является – основные виды авторизации и аутентификации в веб приложениях.

Практическая значимость курсовой работы заключается в формировании у студента навыков научно-исследовательской работы, повышение уровня его профессиональной (теоретической и практической) подготовки, углубление знаний по учебной дисциплине, развитие интереса и навыков самостоятельной работы с научной и справочной литературой.

# **Теоретическая часть**

## **Введение**

Single Page Application – сокращенно SPA, в переводе на русский язык означает “Приложение одной страницы”. Другими словами SPA – это web-приложение, размещенное на одной web-странице, которая для обеспечения работы загружает весь необходимый код вместе с загрузкой самой страницы. Приложение такого типа появились сравнительно недавно, с началом эры HTML5 и SPA является типичным представителем приложений на HTML5.

Отдельно следует обратить внимание на такую важную часть разработки web-приложения как авторизацию и аутентификацию. Предоставление каждому посетителю сайта уникальной учетной записи позволяет однозначно идентифицировать любого из них. Идентификация позволяет веб-сайту менять оформление и контент в зависимости от предпочтений и интересов посетителей.Многие современные сайты содержат различные разделы, предназначенные для разных групп пользователей: от обычных посетителей до администраторов. Также возможно использование авторизации пользователей для предоставления им доступа к различным ресурсам на персональной основе.

Корпоративные сайты, расположенные в защищенных внутренних сетях предприятий, могут содержать страницы с отчетами, предназначенными только для руководителей или сотрудников определённых отделов. Рядовые сотрудники не должны иметь доступа к этим ресурсам.

Также современные веб-сайты могут иметь различные настройки, одни из которых предназначены для удобства широкого круга пользователей, в то время как другие рассчитаны только на использование веб-мастерами. Авторизация позволяет определить, какие функции сайта необходимы и достаточны для каждого конкретного пользователя.

## **Основные понятия, используемые при изучении объекта**

**Аутентификация** — предоставление доказательств, что пользователь тот кем он представился.

**Авторизация** — проверка, что пользователю разрешен доступ к запрашиваемому ресурсу.

При помощи механизма аутентификации можно подстроить сайт под нужды и полномочия каждого пользователя, начиная с банальной персонализации приветствия, выдаваемого пользователю при входе на сайт.

Большинство сайтов, относящихся к сегменту электронной коммерции, могут хранить адреса, номера кредитных карт и другую информацию о своих клиентах. Это избавляет покупателей от необходимости вводить эти данные заново при каждом заказе.

Эти сайты также могут накапливать информацию о заказах, просмотрах товаров и привычках своих клиентов, чтобы предлагать им сопутствующие товары или получать полезную маркетинговую статистику. Можно давать посетителям возможность настроить под себя оформление электронного магазина, выбрать любимые категории товаров и так далее.

Аутентификация сама по себе позволяет разработчику использовать многие полезные функции и настроить сайт согласно предпочтениям каждого конкретного пользователя.

Авторизация же использует уникальный идентификатор пользователя для того, чтобы определить, какие действия пользователь имеет право совершить на сайте: какие страницы просмотреть, какие данные изменить и т. д.

## **Классификация элементов**

**Существуют следующие виды аутентификации:**

Аутентификация по паролю:   
Этот метод основывается на том, что пользователь должен предоставить username и password для успешной идентификации и аутентификации в системе. Пара username/password задается пользователем при его регистрации в системе, при этом в качестве username может выступать адрес электронной почты пользователя.

Аутентификация по сертификатам:   
Сертификат представляет собой набор атрибутов, идентифицирующих владельца, подписанный certificate authority (CA). CA выступает в роли посредника, который гарантирует подлинность сертификатов (по аналогии с ФМС, выпускающей паспорта). Также сертификат криптографически связан с закрытым ключом, который хранится у владельца сертификата и позволяет однозначно подтвердить факт владения сертификатом.

Аутентификация по одноразовым паролям:   
Аутентификация по одноразовым паролям обычно применяется дополнительно к аутентификации по паролям для реализацииtwo-factor authentication (2FA). В этой концепции пользователю необходимо предоставить данные двух типов для входа в систему: что-то, что он знает (например, пароль), и что-то, чем он владеет (например, устройство для генерации одноразовых паролей). Наличие двух факторов позволяет в значительной степени увеличить уровень безопасности, что м. б. востребовано для определенных видов веб-приложений.

Аутентификация по ключам доступа:   
Этот способ чаще всего используется для аутентификации устройств, сервисов или других приложений при обращении к веб-сервисам. Здесь в качестве секрета применяются ключи доступа (access key, API key) — длинные уникальные строки, содержащие произвольный набор символов, по сути заменяющие собой комбинацию username/password.

Аутентификация по токенам:   
Такой способ аутентификации чаще всего применяется при построении распределенных систем Single Sign-On (SSO), где одно приложение (service provider или relying party) делегирует функцию аутентификации пользователей другому приложению (identity provider или authentication service). Типичный пример этого способа — вход в приложение через учетную запись в социальных сетях. Здесь социальные сети являются сервисами аутентификации, а приложение доверяет функцию аутентификации пользователей социальным сетям.

## **Подробная характеристика элементов**

### Аутентификация по паролю:

Применительно к веб-приложениям, существует несколько стандартных протоколов для аутентификации по паролю:

HTTP authentication

Этот протокол, описанный в стандартах HTTP 1.0/1.1, существует очень давно и до сих пор активно применяется в корпоративной среде. Применительно к веб-сайтам работает следующим образом:

Сервер, при обращении неавторизованного клиента к защищенному ресурсу, отсылает HTTP статус “401 Unauthorized” и добавляет заголовок “WWW-Authenticate” с указанием схемы и параметров аутентификации.

Браузер, при получении такого ответа, автоматически показывает диалог ввода username и password. Пользователь вводит детали своей учетной записи.

Во всех последующих запросах к этому веб-сайту браузер автоматически добавляет HTTP заголовок “Authorization”, в котором передаются данные пользователя для аутентификации сервером.

Сервер аутентифицирует пользователя по данным из этого заголовка. Решение о предоставлении доступа (авторизация) производится отдельно на основании роли пользователя, ACL или других данных учетной записи.

Стоит отметить, что при использовании HTTP-аутентификации у пользователя нет стандартной возможности выйти из веб-приложения, кроме как закрыть все окна браузера.

Forms authentication   
Для этого протокола нет определенного стандарта, поэтому все его реализации специфичны для конкретных систем, а точнее, для модулей аутентификации фреймворков разработки.

Работает это по следующему принципу: в веб-приложение включается HTML-форма, в которую пользователь должен ввести свои username/password и отправить их на сервер через HTTP POST для аутентификации. В случае успеха веб-приложение создает session token, который обычно помещается в browser cookies. При последующих веб-запросах session token автоматически передается на сервер и позволяет приложению получить информацию о текущем пользователе для авторизации запроса.

Приложение может создать session token двумя способами:

Как идентификатор аутентифицированной сессии пользователя, которая хранится в памяти сервера или в базе данных. Сессия должна содержать всю необходимую информацию о пользователе для возможности авторизации его запросов.

Как зашифрованный и/или подписанный объект, содержащий данные о пользователе, а также период действия. Этот подход позволяет реализовать stateless-архитектуру сервера, однако требует механизма обновления сессионного токена по истечении срока действия. Несколько стандартных форматов таких токенов рассматриваются в секции «Аутентификация по токенам».

Необходимо понимать, что перехват session token зачастую дает аналогичный уровень доступа, что и знание username/password. Поэтому все коммуникации между клиентом и сервером в случае forms authentication должны производиться только по защищенному соединению HTTPS.

Два протокола, описанных выше, успешно используются для аутентификации пользователей на веб-сайтах. Но при разработке клиент-серверных приложений с использованием веб-сервисов (например, iOS или Android), наряду с HTTP аутентификацией, часто применяются нестандартные протоколы, в которых данные для аутентификации передаются в других частях запроса.

Существует всего несколько мест, где можно передать username и password в HTTP запросах:

URL query — считается небезопасным вариантом, т. к. строки URL могут запоминаться браузерами, прокси и веб-серверами.

Request body — безопасный вариант, но он применим только для запросов, содержащих тело сообщения (такие как POST, PUT, PATCH).

HTTP header —оптимальный вариант, при этом могут использоваться и стандартный заголовок Authorization (например, с Basic-схемой), и другие произвольные заголовки.

### Аутентификация по сертификатам:

На стороне клиента сертификат вместе с закрытым ключом могут храниться в операционной системе, в браузере, в файле, на отдельном физическом устройстве (smart card, USB token). Обычно закрытый ключ дополнительно защищен паролем или PIN-кодом.

В веб-приложениях традиционно используют сертификаты стандарта X.509. Аутентификация с помощью X.509-сертификата происходит в момент соединения с сервером и является частью протокола SSL/TLS. Этот механизм также хорошо поддерживается браузерами, которые позволяют пользователю выбрать и применить сертификат, если веб-сайт допускает такой способ аутентификации.

Во время аутентификации сервер выполняет проверку сертификата на основании следующих правил:

Сертификат должен быть подписан доверенным certification authority (проверка цепочки сертификатов).

Сертификат должен быть действительным на текущую дату (проверка срока действия).

Сертификат не должен быть отозван соответствующим CA (проверка списков исключения).

После успешной аутентификации веб-приложение может выполнить авторизацию запроса на основании таких данных сертификата, как имя владельца, эмитент, серийный номер сертификата или отпечаток открытого ключа сертификата.

Использование сертификатов для аутентификации — куда более надежный способ, чем аутентификация посредством паролей. Это достигается созданием в процессе аутентификации цифровой подписи, наличие которой доказывает факт применения закрытого ключа в конкретной ситуации. Однако трудности с распространением и поддержкой сертификатов делает такой способ аутентификации малодоступным в широких кругах.

### Аутентификация по одноразовым паролям:

Популярный сценарий использования одноразовых паролей — дополнительная аутентификация пользователя во время выполнения важных действий: перевод денег, изменение настроек и т. п.

Существуют разные источники для создания одноразовых паролей. Наиболее популярные:

Аппаратные или программные токены, которые могут генерировать одноразовые пароли на основании секретного ключа, введенного в них, и текущего времени. Секретные ключи пользователей, являющиеся фактором владения, также хранятся на сервере, что позволяет выполнить проверку введенных одноразовых паролей. Пример аппаратной реализаций токенов — [RSA SecurID](http://www.emc.com/security/rsa-securid/index.htm); программной — приложение [Google Authenticator](https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2).

Случайно генерируемые коды, передаваемые пользователю через SMS или другой канал связи. В этой ситуации фактор владения — телефон пользователя (точнее — SIM-карта, привязанная к определенному номеру).

Распечатка или scratch card со списком заранее сформированных одноразовых паролей. Для каждого нового входа в систему требуется ввести новый одноразовый пароль с указанным номером.

В веб-приложениях такой механизм аутентификации часто реализуется посредством расширения forms authentication: после первичной аутентификации по паролю, создается сессия пользователя, однако в контексте этой сессии пользователь не имеет доступа к приложению до тех пор, пока он не выполнит дополнительную аутентификацию по одноразовому паролю.

### Аутентификация по ключам доступа:

В большинстве случаев, сервер генерирует ключи доступа по запросу пользователей, которые далее сохраняют эти ключи в клиентских приложениях. При создании ключа также возможно ограничить срок действия и уровень доступа, который получит клиентское приложение при аутентификации с помощью этого ключа.

Хороший пример применения аутентификации по ключу — облако Amazon Web Services. Предположим, у пользователя есть веб-приложение, позволяющее загружать и просматривать фотографии, и он хочет использовать сервис Amazon S3 для хранения файлов. В таком случае, пользователь через консоль AWS может создать ключ, имеющий ограниченный доступ к облаку: только чтение/запись его файлов в Amazon S3. Этот ключ в результате можно применить для аутентификации веб-приложения в облаке AWS.

Использование ключей позволяет избежать передачи пароля пользователя сторонним приложениям (в примере выше пользователь сохранил в веб-приложении не свой пароль, а ключ доступа). Ключи обладают значительно большей энтропией по сравнению с паролями, поэтому их практически невозможно подобрать. Кроме того, если ключ был раскрыт, это не приводит к компрометации основной учетной записи пользователя — достаточно лишь аннулировать этот ключ и создать новый.

С технической точки зрения, здесь не существует единого протокола: ключи могут передаваться в разных частях HTTP-запроса: URL query, request body или HTTP header. Как и в случае аутентификации по паролю, наиболее оптимальный вариант — использование HTTP header. В некоторых случаях используют HTTP-схему Bearer для передачи токена в заголовке (Authorization: Bearer [token]). Чтобы избежать перехвата ключей, соединение с сервером должно быть обязательно защищено протоколом SSL/TLS.

Кроме того, существуют более сложные схемы аутентификации по ключам для незащищенных соединений. В этом случае, ключ обычно состоит их двух частей: публичной и секретной. Публичная часть используется для идентификации клиента, а секретная часть позволяет сгенерировать подпись. Например, по аналогии с digest authentication схемой, сервер может послать клиенту уникальное значение nonce или timestamp, а клиент — возвратить хэш или HMAC этого значения, вычисленный с использованием секретной части ключа. Это позволяет избежать передачи всего ключа в оригинальном виде и защищает от replay attacks.

### Аутентификация по токенам:

Реализация этого способа заключается в том, что identity provider (IP) предоставляет достоверные сведения о пользователе в виде токена, а service provider (SP) приложение использует этот токен для идентификации, аутентификации и авторизации пользователя.  
На общем уровне, весь процесс выглядит следующим образом:

Клиент аутентифицируется в identity provider одним из способов, специфичным для него (пароль, ключ доступа, сертификат, Kerberos, итд.).

Клиент просит identity provider предоставить ему токен для конкретного SP-приложения. Identity provider генерирует токен и отправляет его клиенту.

Клиент аутентифицируется в SP-приложении при помощи этого токена.

Процесс, описанный выше, отражает механизм аутентификации активного клиента, т. е. такого, который может выполнять запрограммированную последовательность действий (например, iOS/Android приложения). Браузер же — пассивный клиент в том смысле, что он только может отображать страницы, запрошенные пользователем. В этом случае аутентификация достигается посредством автоматического перенаправления браузера между веб-приложениями identity provider и service provider.

Существует несколько стандартов, в точности определяющих протокол взаимодействия между клиентами (активными и пассивными) и IP/SP-приложениями и формат поддерживаемых токенов. Среди наиболее популярных стандартов — OAuth, OpenID Connect, SAML, и WS-Federation. Некоторая информация об этих протоколах — ниже в статье.

Сам токен обычно представляет собой структуру данных, которая содержит информацию, кто сгенерировал токен, кто может быть получателем токена, срок действия, набор сведений о самом пользователе (claims). Кроме того, токен дополнительно подписывается для предотвращения несанкционированных изменений и гарантий подлинности.

При аутентификации с помощью токена SP-приложение должно выполнить следующие проверки:

Токен был выдан доверенным identity provider приложением (проверка поля issuer).

Токен предназначается текущему SP-приложению (проверка поля audience).

Срок действия токена еще не истек (проверка поля expiration date).

Токен подлинный и не был изменен (проверка подписи).

В случае успешной проверки SP-приложение выполняет авторизацию запроса на основании данных о пользователе, содержащихся в токене.

Существует несколько распространенных форматов токенов для веб-приложений:

**Simple Web Token** (SWT) — наиболее простой формат, представляющий собой набор произвольных пар имя/значение в формате кодирования HTML form. Стандарт определяет несколько зарезервированных имен: Issuer, Audience, ExpiresOn и HMACSHA256. Токен подписывается с помощью симметричного ключа, таким образом оба IP- и SP-приложения должны иметь этот ключ для возможности создания/проверки токена.

**JSON Web Token (JWT)** — содержит три блока, разделенных точками: заголовок, набор полей (claims) и подпись. Первые два блока представлены в JSON-формате и дополнительно закодированы в формат base64. Набор полей содержит произвольные пары имя/значения, притом стандарт JWT определяет несколько зарезервированных имен (iss, aud, exp и другие). Подпись может генерироваться при помощи и симметричных алгоритмов шифрования, и асимметричных. Кроме того, существует отдельный стандарт, описывающий формат зашифрованного JWT-токена.

**Security Assertion Markup Language (SAML)** — определяет токены (SAML assertions) в XML-формате, включающем информацию об эмитенте, о субъекте, необходимые условия для проверки токена, набор дополнительных утверждений (statements) о пользователе. Подпись SAML-токенов осуществляется при помощи ассиметричной криптографии. Кроме того, в отличие от предыдущих форматов, SAML-токены содержат механизм для подтверждения владения токеном, что позволяет предотвратить перехват токенов через man-in-the-middle-атаки при использовании незащищенных соединений.

## **Заключение**

Таким образом, можно сделать вывод о том, что каждый способ аутентификации подходит для своей конкретной задачи:

По паролю – Аутентификация пользователей;

По сертификатам – Аутентификация пользователей в безопасных приложениях; аутентификация сервисов

По одноразовым паролям – Дополнительная аутентификация пользователей (для достижения two-factor authentication)

По ключам доступа – Аутентификация сервисов и приложений

По токенам – Делегированная аутентификация пользователей; делегированная авторизация приложений.

# **Практическая часть**

В практической части работы было разработано SPA приложение, выполняющее роль сервиса бронирования отелей. Приложение было разработано на основе фреймворка laravel и vue.js. Причинами выбора данных фреймворков послужили: масштабируемость, простота освоения и внедрения. Основной особенностью SPA является его архитектура: это буквально одна страница, которая постоянно взаимодействует с пользователем, динамически переписывая текущую страницу, а не загружая целые новые страницы с сервера. Мы видим примеры одностраничных приложений каждый день: Trello, Facebook, Gmail, Twitter - вот несколько примеров SPA.

Особенность архитектуры SPA заключается в том, что все элементы, необходимые для работы софта находятся на одной странице. Они загружаются при инициализации. Также данный вид приложений загружает дополнительные модули после запроса от пользователя. Любая пользовательская активность фиксируется для удобства навигации. Это позволяет скопировать ссылку и открыть софт на том же этапе взаимодействия на другой вкладке, браузере или устройстве.

При загрузке новых модулей в SPA контент на них обновляется только частично, так как нет необходимости повторно загружать неизменные элементы. Это увеличивает скорость ответа и сокращает передаваемый объем данных между браузером и сервером.

Данный вид софта по способу взаимодействия с пользователем больше всего похож на работу десктопных приложений, но на сервере.

За основы был взят фреймворк laravel, именно на нем будут формироваться главные запросы, и взаимодействие с базой данных.

## **Разработка базы данных**

Главный этап проектирования приложения – создание базы данных, именно на ее основе будут сформированы основные методы работы с данными. Так как мы используем laravel она уже имеет встроенный механизм взаимодействия с базой данных.

Первым делом необходимо создать базу, для этого используем open Server и его встроенный сервер баз данных postgresSQL.

После этого нам необходимо связать нашу базу с сервером, для этого необходимо прописать в .env файле данный по нашей базе. Следующим этапом необходимо создать миграции с моделями данных. Миграции отвечают за данные, которые будут созданы в базе.

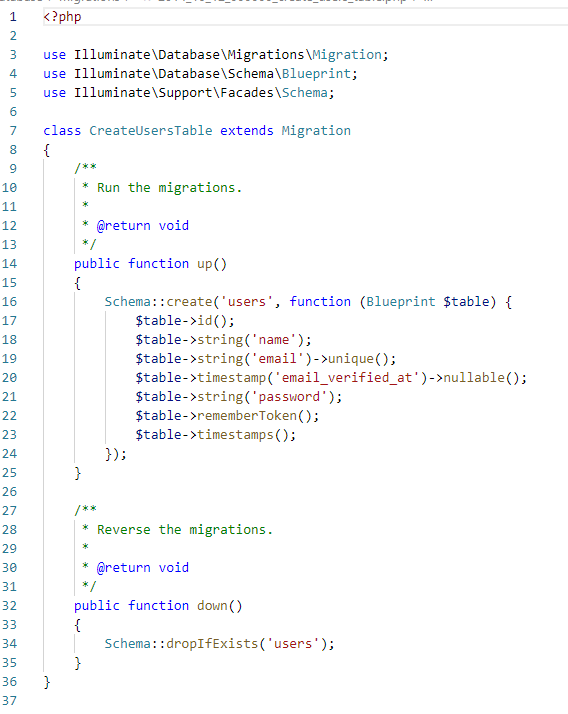


Рисунок 1 – файл миграции таблицы “пользователи”

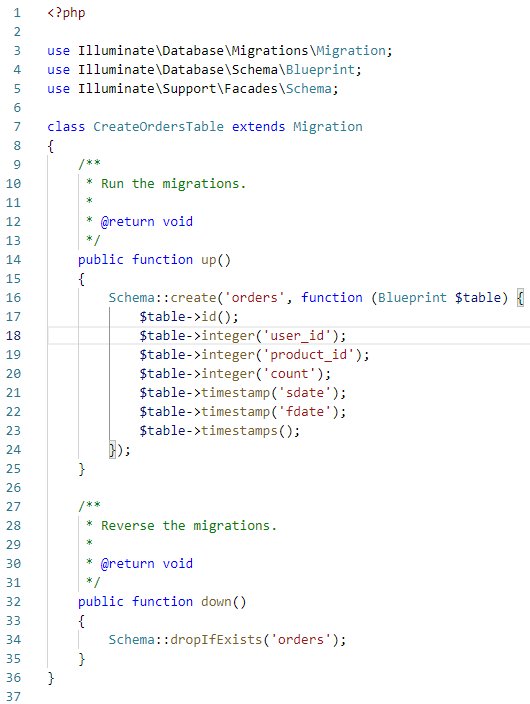


Рисунок 2 – файл миграции заказов

После этого необходимо связать наши таблицы, для этого в моделях необходимо прописать соответствующую команду для внешнего (BelongTo) и первичного ключа(hasMany).



Рисунок 3 - Пример модели “Заявка”

Далее необходимо провести миграцию, это можно сделать, выполнив следующую команду: php artisan migratte. После этого наша база примет следующий вид:

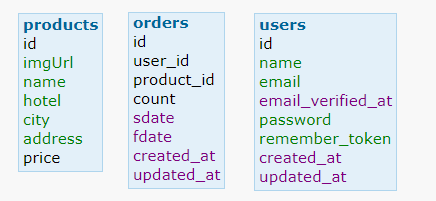


Рисунок 4 – схема базы данных

## **Разработка серверной части приложения**

За основу серверной части берем фреймворк laravel, она уже включает в себя шаблоны по формированию запросов. Пропишем главный роуты по которым будем получать основную страницу с кодом js, так как с определенными роутами будем работать на уровне SPA.

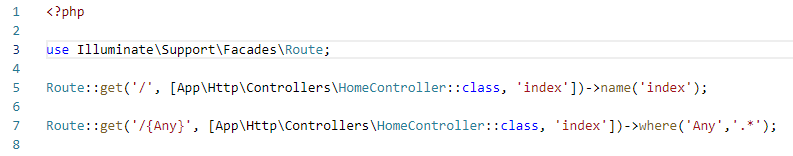


Рисунок 5 – главный роут приложения

Далее определим api роуты, они будет отвечать за взаимодействие с базой данных(POST запросы).

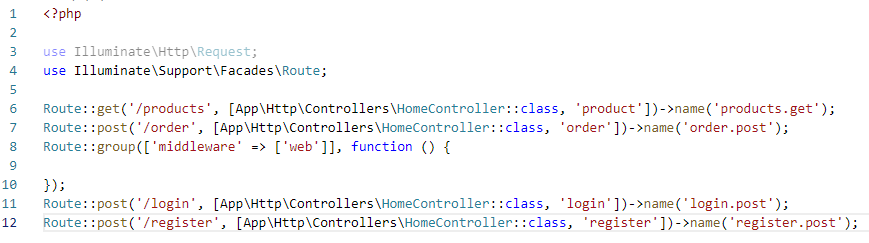


Рисунок 6 – api приложения

Далее нам необходимо определить работу api методов. Опишем их реализацию в HomeController:



Рисунок 7 – методы api в HomeController.



Рисунок 8 – метод login в HomeController



Рисунок 9 – метод регистрации в HomeController

## **Разработка SPA модуля**

Нам необходимо создать главную страницу сайта “index”, который и будет выступать в качестве одностраничного приложения:

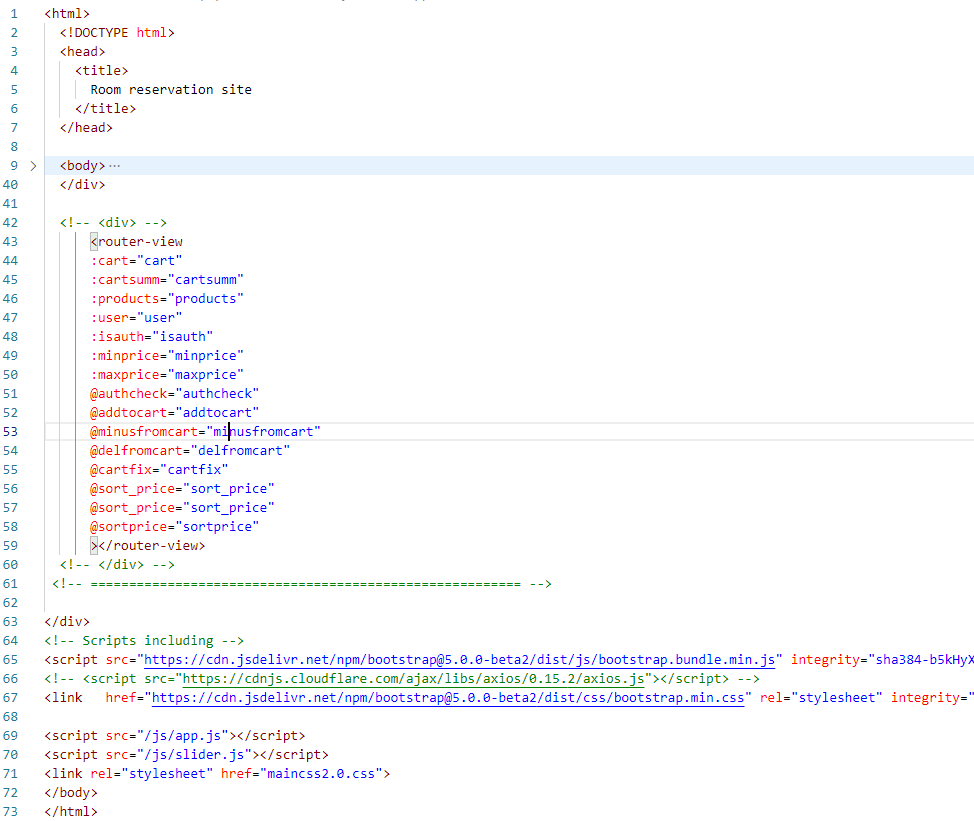


Рисунок 10 – файл index

Далее подключаем главный js файл и определяем объект vue.

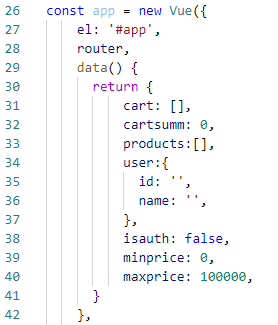


Рисунок 11 – Vue объект

Далее создаем файл router, который будет отвечать за routing внутри SPA приложения.

В нем мы подключаем компоненты vue и определяем роуты к ним. В выходной функции зададим mode:history который будет отвечать за сохранение страниц в истории(Пользователь даже не заметит что находится на одной странице).

Первым этапом разработки модуля будет создание vue:



Рисунок 12 – файл router

Далее подключим, созданный роутер к главному js файлу с VUE объектом



Рисунок 13 – подключение router’a

Следующим этапом будет создание vue компонента, который и будет связан с js. Для примера приведем vue component главной страницы:

В главном js фале создадим переменные и методы для:

Получение товаров из базы данных

Работы с пользователем

Добавление, удаление товаров в корзине.

Методы сортировки и фильтрации

Опишем структуру компонента index.vue (компонент главной страницы)



Рисунок 14 – раздел с версткой в главном компоненте

Так как в index.blade.php мы определили какие данные и функции будем передавать, нам необходимо определить эти данные и функции в index.vue

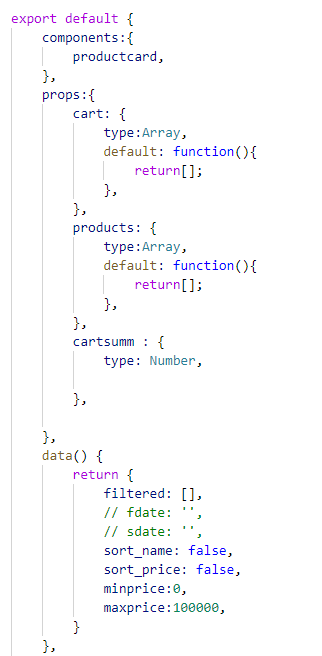


Рисунок 15 – раздел с js кодом в главном компоненте

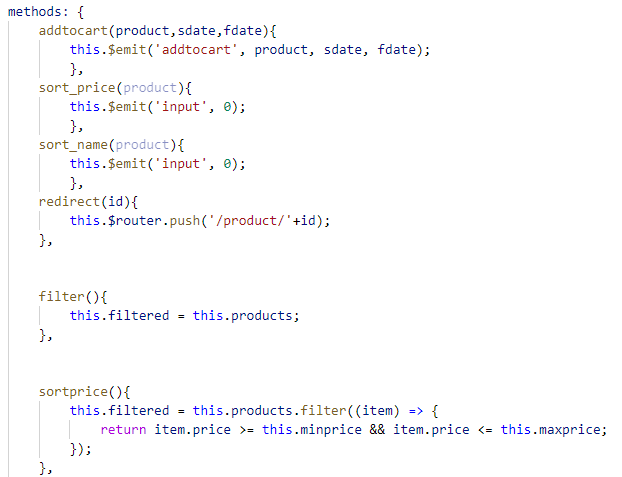
****

Рисунок 16 – основные методы в index.vue

Перечисленные методы отвечают за добавление товара в корзину, сортировку и фильтрацию товаров.

Все товары выводим через свойство-цикл v-for, который выводит на страницу массив объектов products, ранее запрошенных из базы данных в главном js файле (Здесь и в дальнейшем для отправления запроса была применена js библиотека axios).

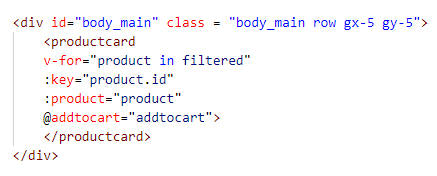


Рисунок 17 – вывод объектов на страницу

Далее добавим компонент Cart, который будет содержать интерфейс взаимодействия с товарами, добавленными в корзину. Реализация корзины представляет из себя реализацию выводов товаров на главную страницу, отличие заключается лишь в массиве. Для корзины используется массив cart, а для всех продуктов – productes. В нем так же опишем функции, передаваемые из родительского компонента, а также добавим функцию оформления заказа (Отправление записи в базу данных).



Рисунок 18 – метод отправки заказов на сервер

Теперь добавим компоненты, отвечающие за логин и регистрацию пользователя. Они будут содержать формы для ввода данных и функции отправки данных на сервер (Вся валидация данных реализована на стороне сервера).



Рисунок 19 – пример метода авторизации в компоненте login.

Для того чтобы пользователь видел под каким аккаунтом он авторизован, был реализован механизм вывода имени профиля в хедер. Данный механизм объявлен как отдельный компонент vue.js и будет выводиться на всех страницах вне зависимости от текущего uri параметров vue-router.

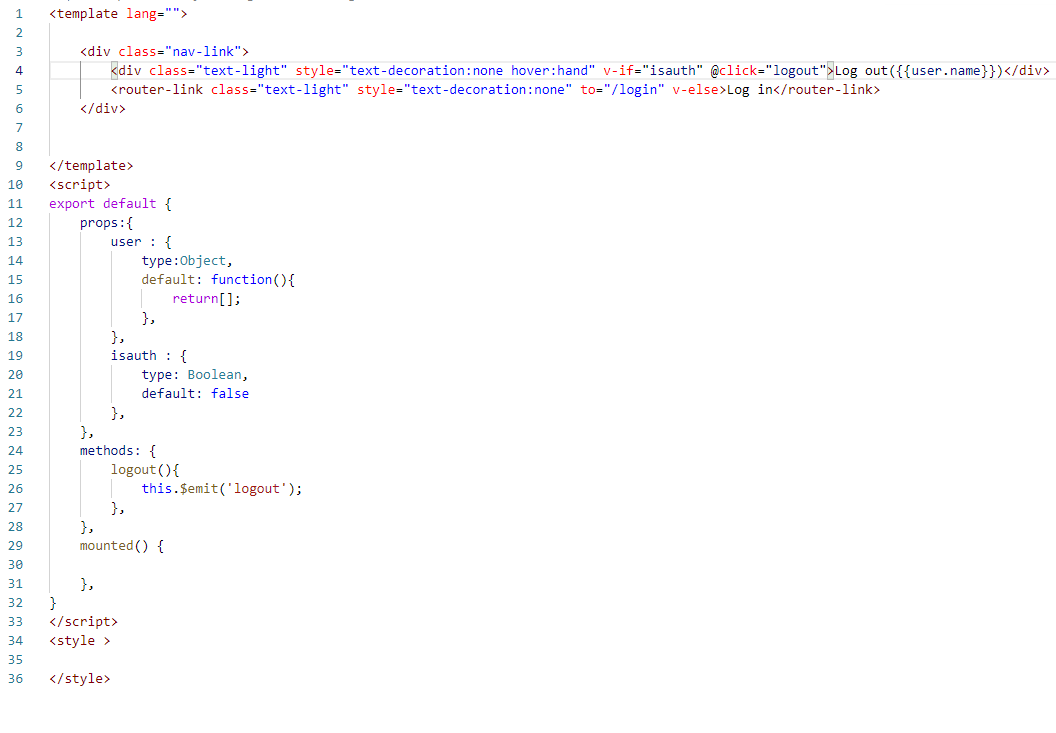


Рисунок 20 – компонент Logout, ответственный за вывод имени профиля пользователя

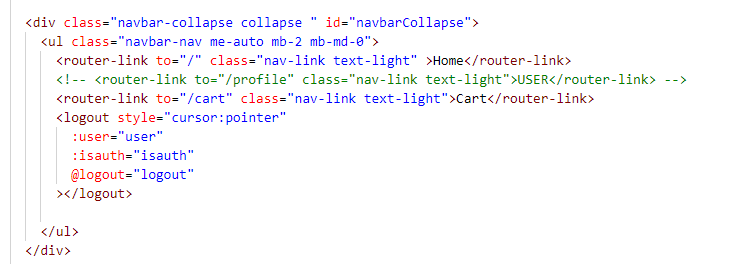


Рисунок 21 – подключение компонента logout к index.blade.php

После выполнения всех вышеперечисленных операций, получим следующий результат:

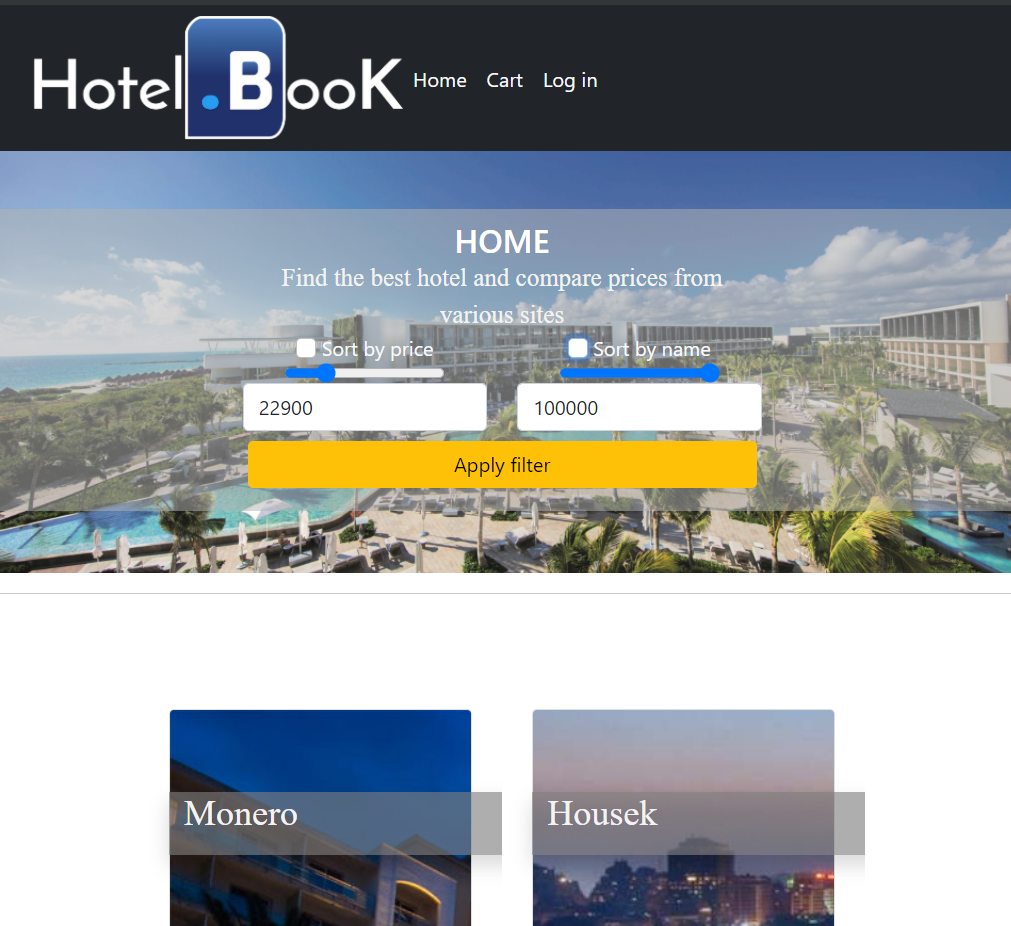


Рисунок 22 – страница Home

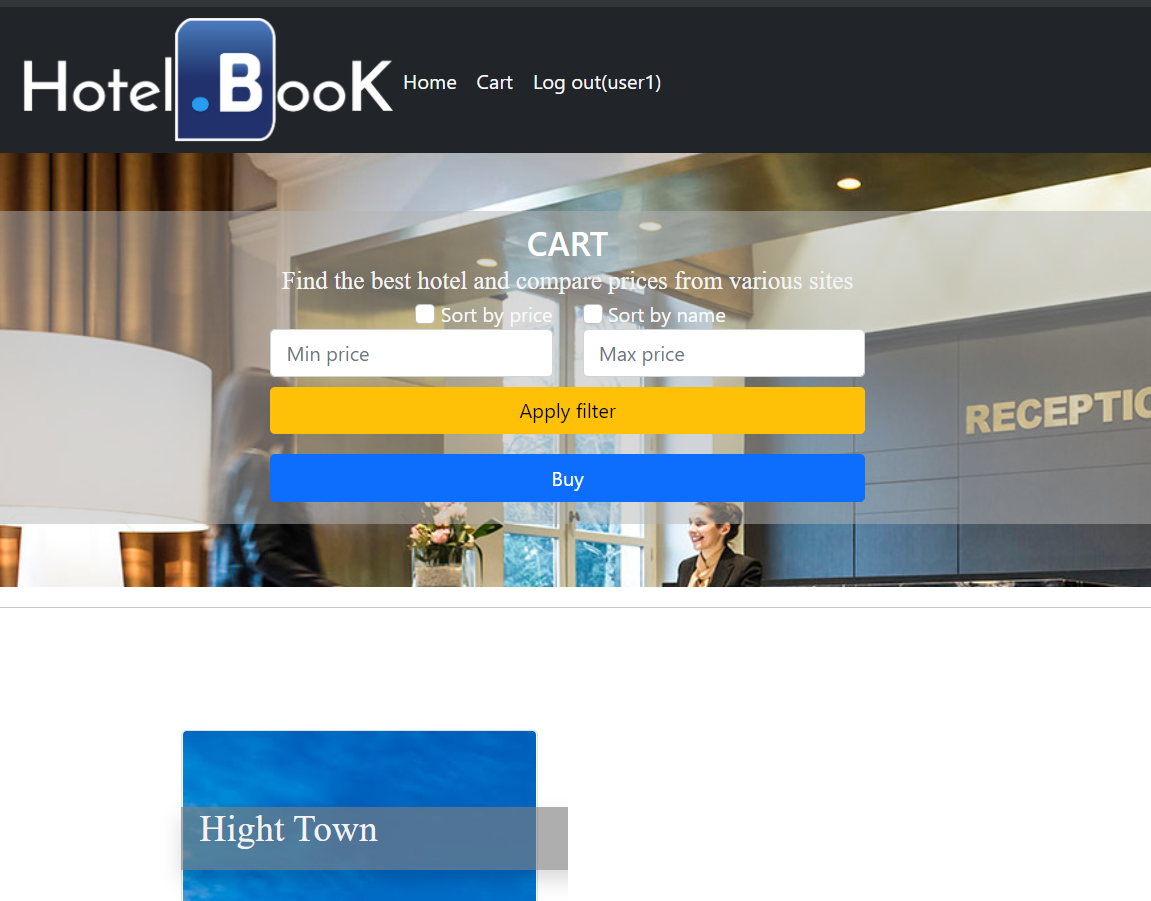


Рисунок 23 - страница Cart

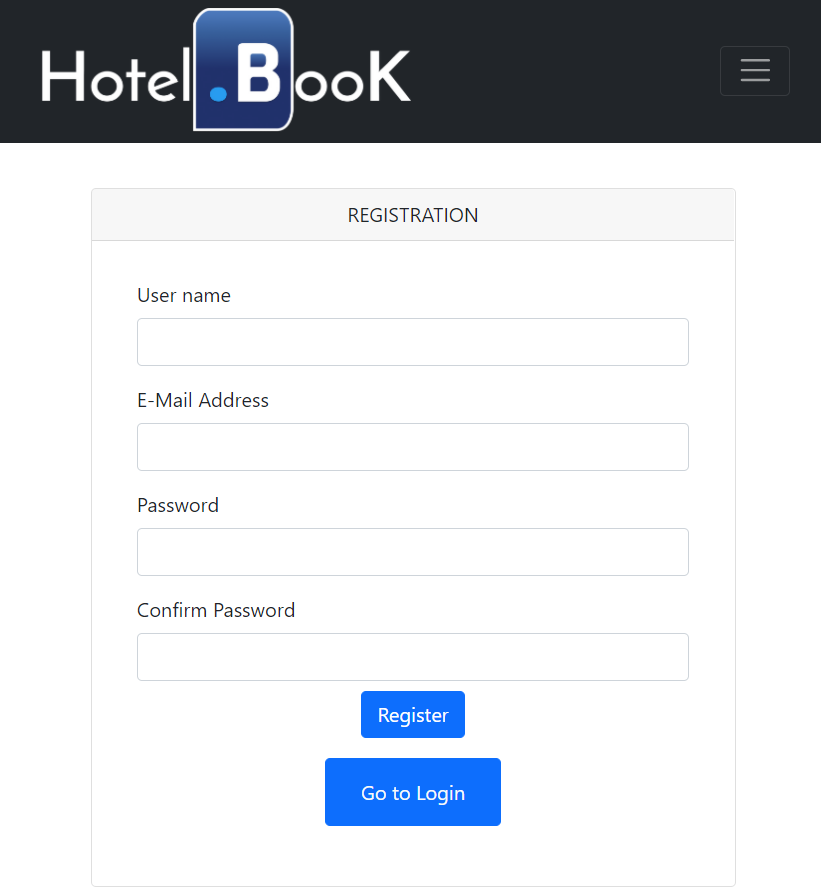


Рисунок 24 – страница регистрации

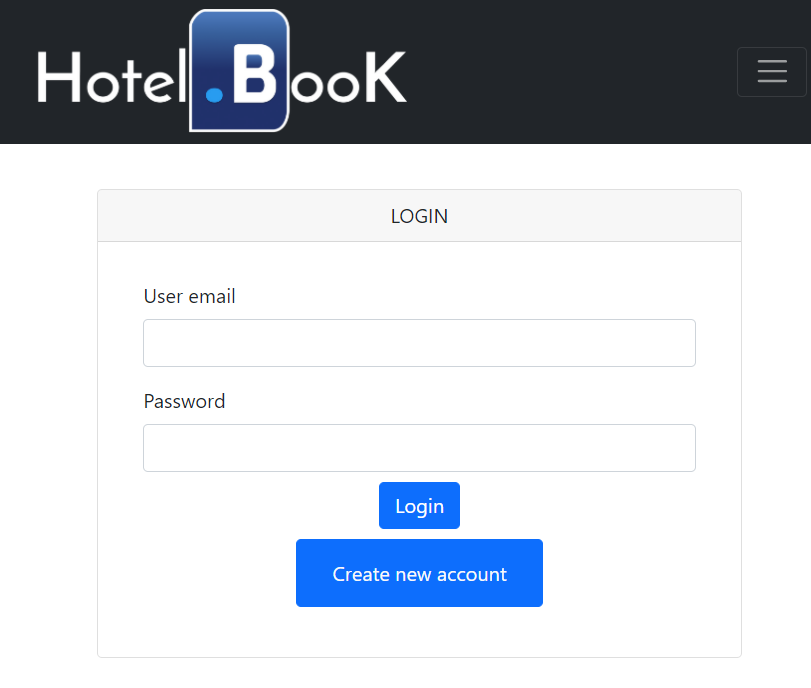


Рисунок 25 – страница авторизации

## **Заключение**

Таким образом, в ходе проделанной работы по созданию SPA-приложения, мы получили рабочий, функциональный прототип одностраничного сайта, который удовлетворяет всем требованиям задания. В ходе работы были получены практические и теоретические навыки и знания работы с фреймворками laravel(бэкэнд) и vue.js(фронтэнд).

В конечном итоге мы имеем прототип приложения с 4 полнофункциональными страницами:

Home;

Cart;

Login;

Register;

Которые предоставляют минимальную базовую функциональность для сервиса бронирования отелей.

# 

# **Список литературы**

1. Министерство науки и высшего образования российской федерации федеральное государственное автономное образовательное учреждение высшего образования «северо-кавказский федеральный университет» - Методические указания к курсовой работе по дисциплине «Интернет-технологии» для студентов направления 09.03.03 «Прикладная информатика» (профиль «Прикладная информатика в экономике») Ставрополь 2020
2. К. Дж. Дейт. Введение в системы баз данных. 8е издание.: Перевод с английского и редакция К. А. Птицина. Издательский дом «Вильяме», 2005. – 1328 c.
3. VUE.js – The Progressive JavaScript Framework: сайт - http://vuejs.org (дата обращения: 25.03.2021). Текст: - электронный.
4. Laravel – The PHP Framework For Web Artisans – https://laravel.com (дата обращения: 15.03.2021). Текст: - электронный.
5. Сайтостроение от А до Я. Авторизация и защита веб ресурсов в ASP.NET: сайт – https://www.internet-technologies.ru/articles/avtorizaciya-i-zaschita-veb-resursov-v-asp-net.html (дата обращения: 18.03.2021). Текст: - электронный.
6. HABR. Обзор способов и протоколов аутентификации в веб-приложениях: сайт – https://habr.com/ru/company/dataart/blog/262817/ (дата обращения: 02.0.4.2021). Текст: - электронный.
7. Database Answers. сайт – <http://www.databaseanswers.org/> (дата обращения: 05.03.2021). Текст: - электронный.