Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Системный анализ и проектирование информационных систем (с разделом Разработка веб-приложений)»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Н. Салапура |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Организация отельного бизнеса и его программная поддержка в сети интернет»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 172303  Прохоренко Татьяна Владимировна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

СОДЕРЖАНИЕ

[1 АНАЛИЗ И МОЛЕДИРОВАНИЕ ОРГАНИЗАЦИИ ОТЕЛЬНОГО БИЗНЕСА И ЕГО ПОДДЕРЖКИ В СЕТИ ИНТЕРНЕТ 4](#_Toc135597572)

[1.1 Описание деятельности отельного бизнеса и его поддержки в сети интернет. 4](#_Toc135597573)

[1.2 Разработка функциональной предметной области 5](#_Toc135597574)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 8](#_Toc135597575)

[1.4 Разработка информационной модели предметной области 9](#_Toc135597576)

[1.5 UML-моделирование представления программного средства и их описание 10](#_Toc135597577)

[2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАМНОГО СРЕДСТВА 13](#_Toc135597578)

[2.1 Постановка задачи 13](#_Toc135597579)

[2.2 Обоснование компонентов и технологий для реализации программного средства 13](#_Toc135597580)

[2.3 Архитектурные решения 14](#_Toc135597581)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 15](#_Toc135597582)

[2.5 Проектирование пользовательского интерфейса 21](#_Toc135597583)

[2.6 Методы и средства, используемые для обеспечения безопасности данных 27](#_Toc135597584)

[3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММЫ 29](#_Toc135597585)

[4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ 32](#_Toc135597586)

[4.1 Авторизация 32](#_Toc135597587)

[4.2 Модуль администратора 33](#_Toc135597588)

[4.3 Модуль пользователя 33](#_Toc135597589)

[ЗАКЛЮЧЕНИЕ 36](#_Toc135597590)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 37](#_Toc135597591)

[ПРИЛОЖЕНИЕ А Листинг кода алгоритмов, реализующих основную бизнес-логику. 38](#_Toc135597592)

[ПРИЛОЖЕНИЕ Б Отчет о проверке на заимствование в системе «Антиплагиат» 48](#_Toc135597593)

**ВВЕДЕНИЕ**

Отельный бизнес – это один из самых важных секторов экономики, который обеспечивает жилье и различные услуги для туристов и бизнесменов. Гостиничный бизнес охватывает широкий спектр услуг, от простых номеров с минимальным набором удобств до роскошных люксов, ресторанов и спа-салонов. В настоящее время этот бизнес является одним из наиболее динамично развивающихся секторов экономики, в связи с увеличением количества туристов и бизнесменов, которые путешествуют по всему миру.

С развитием технологий и доступности интернета, большинство отелей и гостиниц используют онлайн-бронирование и продажу своих услуг, что позволяет упростить процесс заказа и обеспечивает более высокую загрузку номеров. Благодаря этому, отельный бизнес стал еще более доступным и удобным для путешественников, которые могут забронировать номера в любой точке мира, не выходя из дома.

Однако, несмотря на все преимущества, которые предоставляет онлайн-бронирование, современные отели и гостиницы сталкиваются со многими проблемами в сфере бизнеса, такими как повышенная конкуренция, изменение потребительских предпочтений и высокие затраты на маркетинг и рекламу. В связи с этим, эффективное использование информационных технологий и программной поддержки в отельном бизнесе становится все более важным фактором, который позволяет организациям улучшить свою деятельность, повысить конкурентоспособность и удовлетворить потребности своих клиентов.

Целью курсовой работы является повышение конкурентно способности отеля при помощи проектирования и реализации понятного для пользователей интерфейса и эстетически приятного дизайна.

В процессе разработки программы необходимо определить ключевой перечень задач, которые предстоит решить при курсовом проектировании, к ним относятся:

* анализ изучаемых процессов;
* разработка uml диаграмм;
* Разработка дизайна веб-сайта;
* выполнение программной реализации проектных решений;
* протестировать и проверить работоспособность программного обеспечения.

Объектом исследования будет выступать отельный бизнес, предметом учет бронирование номеров в отеле

Для достижения поставленных задач и целей необходимо провести анализ предметной области, смоделировать основные процессы и по результатам создать программу.

# АНАЛИЗ И МОЛЕДИРОВАНИЕ ОРГАНИЗАЦИИ ОТЕЛЬНОГО БИЗНЕСА И ЕГО ПОДДЕРЖКИ В СЕТИ ИНТЕРНЕТ

## Описание деятельности отельного бизнеса и его поддержки в сети интернет.

Отельный бизнес — развивающийся в области индустрии гостеприимства и направленный на предоставление туристам и путешественникам временного жилья, питания, а также решением вопросов бытового обслуживания и организацией проведения досуга. Отели могут предоставлять различные типы номеров, включая одноместные, двухместные, номера люкс и т.д.

В настоящее время, большинство отелей предоставляют возможность бронирования номеров через свои официальные веб-сайты или через онлайн-платформы для бронирования отелей. Бронирование номера в отеле через интернет является одним из наиболее популярных способов бронирования номера, особенно среди молодежи и иностранных туристов, которые предпочитают более удобный и быстрый способ бронирования.

Отели также используют интернет технологии для размещения информации о доступных дополнительных услуг помимо номеров, такие как экскурсии, бассейн и т.д. Так же на веб-странице отеля обычно предоставляют контактные данные отеля для связи напрямую, информацию о выдающихся сотрудниках, награды.

Создание такого веб-сайта имеет важное значение для отельного бизнеса, так как это позволяет привлекать новых клиентов, увеличивать доходы и улучшать общее качество обслуживания. Веб-сайт отеля может быть оптимизирован для разных устройств и операционных систем, что обеспечивает удобство использования и улучшает пользовательский опыт.

Для обеспечения эффективной поддержки отельного бизнеса в сети интернет может быть использовано клиент-серверное веб-приложение, которое позволит пользователям бронировать номера в отеле. Для этого используется база данных, которая хранит информацию о номерах, ценах и наличии свободных номеров.

Целью поддержки отельного бизнеса в сети интернет является создание привлекательного интерфейса и информативного веб-сайта отеля, адаптированного под различные устройства и операционные системы. Размещение контактных данных отеля для удобной связи с потенциальными клиентами и реализация системы онлайн-бронирования, обеспечивающую удобный и эффективный процесс бронирования номеров через сайт. Также следует реализовать управление занятостью номеров в режиме реального времени в целях повышения удовлетворенности клиентов путем улучшения пользовательского интерфейса и предоставления дополнительной информации о услугах. Для привлечения клиентов необходимо использовать цифровой маркетинг, анализ данных и отзывы клиентов для определения трендов и улучшения маркетинговых стратегий.

Создание веб-сайта для отельного бизнеса также позволяет упростить процессы управления и контроля занятости номеров в отеле.

## Разработка функциональной предметной области

Продолжим рассмотрение процедуры бронирования номера в отеле. После того, как потенциальный гость нашел подходящий отель, он обращается к менеджеру отеля или использует телефон или электронную почту для запроса на бронирование номера. Затем менеджер отеля проверяет наличие свободных номеров в желаемые даты, а также согласовывает с гостем все детали бронирования, такие как количество гостей, тип номера, продолжительность пребывания и стоимость.

После того, как все детали согласованы, менеджер отеля делает запись о бронировании в журнале бронирования. Далее он должен убедиться, что гость предоставил необходимые документы, такие как паспорт или иной документ, удостоверяющий личность. Это важно для того, чтобы в дальнейшем избежать возможных проблем при заселении гостей.

После проверки документов менеджер отеля подтверждает бронирование гостю и высылает подтверждение на указанный им адрес электронной почты или по почтовому адресу. Помимо этого, менеджер отеля может выставить счет на предоплату за бронь номера.

Такой процесс бронирования номера может занять много времени, что может приводить к недовольству клиентов и убыточности бизнеса. Поэтому внедрение автоматизированной системы бронирования номеров поможет ускорить процесс и повысить удобство для гостей.

Процесс описанный выше в сокращенном виде изображен на рисунках 1.1, 1.2, 1.3.

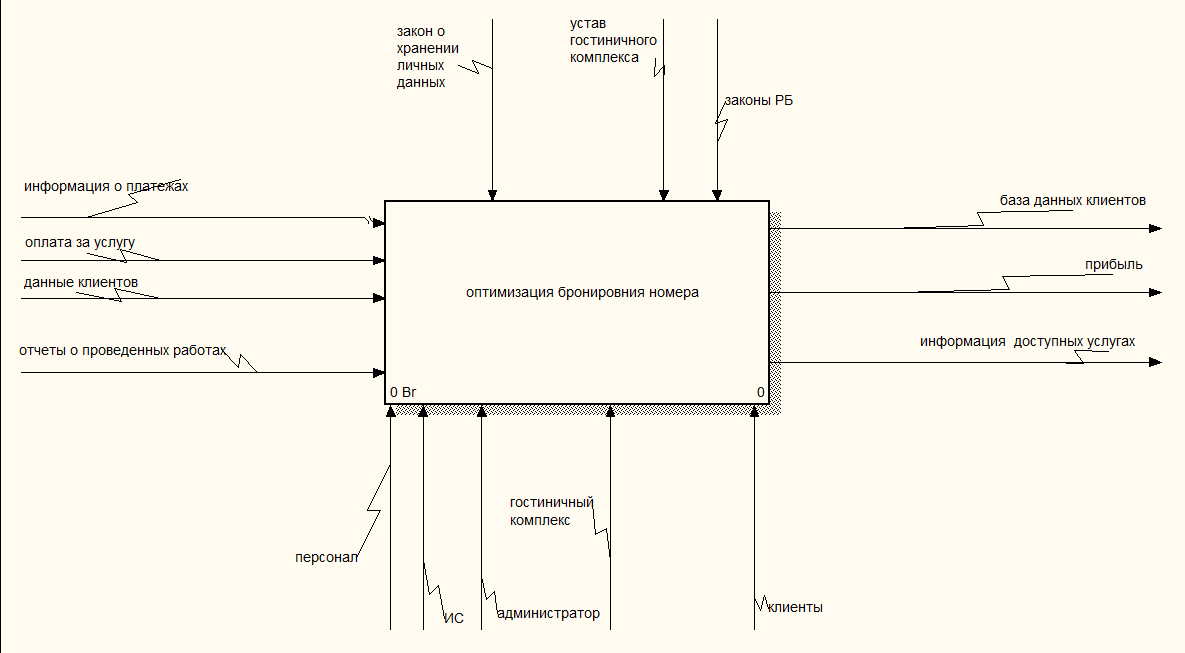


Рисунок 1.1 – Контекстная диаграмма в нотации IDEF0

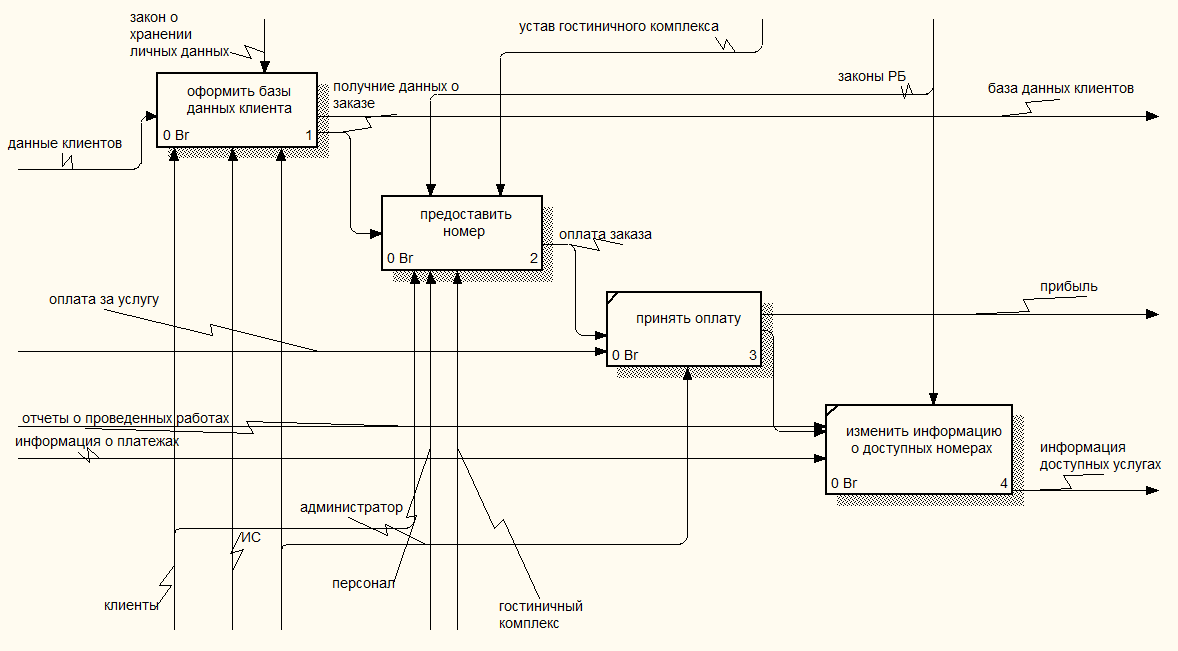


Рисунок 1.2 – Декомпозиция контекстной диаграммы в нотации IDEF0

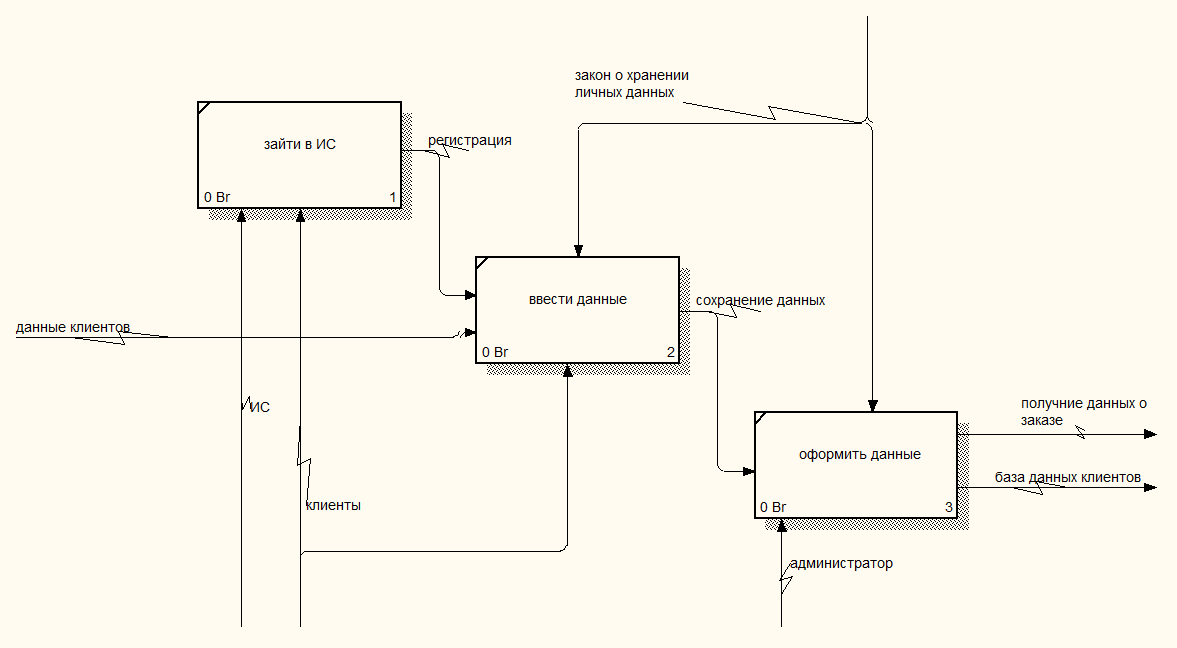


Рисунок 1.3 – Декомпозиция блока “ Оформить базы данных ” в нотации IDEF0

Так же оптимизированное веб-сайтом бронирование требует большего количества персонала который мог бы выполнять данную работу. Поэтому отсюда можно сделать вывод что новых работников необходимо было бы специально обучать, задействовав для обучения квалифицированного сотрудника. Данный процесс несет затрат для отельного бизнеса больше, чем бронирование с помощью оптимизированного для этой работы технического средства в виде веб-сайта. Сам процесс обучения персонала можно увидеть на рисунке 1.4 представленном ниже.



Рисунок 1.4 – BPMN-модель наглядно показывающая затраты на обучение персонала.

## Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

После проведения анализа потребностей персонала отелей и пользователей можно выделить основные требования к программному средству :

− Регистрация и авторизация пользователей: приложение должно предоставлять возможность регистрации новых пользователей с помощью логина и пароля, а также авторизации уже зарегистрированных пользователей;

− Бронирование номеров: пользователи должны иметь возможность бронирования номеров в отеле через приложение. Пользователи должны иметь возможность выбирать доступные номера в отеле, указывать даты пребывания, количество гостей;

* Управление номерами: система должна предоставлять возможность управления номерами, включая добавление новых номеров, изменение цены за ночь, добавление фотографий и описания номеров.
* Управление бронированием: приложение должно предоставлять возможность пользователям отменять свои бронирования, а также просматривать информацию о своих бронирования.

− Обработка исключительных ситуаций;

− Запрос на выполнение необходимых действий.

## Разработка информационной модели предметной области

В данной работе используется база данных MongoDB, имеющая 2 коллекции «Room» и «Users».

Коллекция "Room" служит для хранения информации о номерах отеля. Структура данных в таблице 1.1 предоставляет описание полей, которые присутствуют в коллекции "Room".

Таблица 1.1 – Описание структуры данных Room

|  |  |  |
| --- | --- | --- |
| \_id | ObjectId | Идентификатор |
| Name\_room | String | Название |
| type |  | Тип комнаты |
| guest | String | Количество гостей |
| photoUrl | String | Данные о фото |
| cost | String | Стоимость |
| Internet | Boolean | Наличие интернета |
| Breackfast | Boolean | Наличие завтрака |
| dryer | Boolean | Наличие фена |
| kettle | Boolean | Наличие чайника |
| date | String | Дата добавления номера |
| boking | Array | Массив объектов содержащих информацию о бронировании |

Вторая коллекция «Users» хранит данные об аккаунтах, описание представлено в таблице 1.2.

Таблица 1.2 - Описание структуры данных User

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| email | String | Логин |
| password | String | Пароль |
| isAdmin | Boolean | Является ли аккаунт, аккаунтом администратора |

## UML-моделирование представления программного средства и их описание

Диаграмма вариантов использования, дополняет диаграмму активности UML, предоставляя более общую информацию о функциональности системы и возможных вариантах использования.

На диаграмме вариантов использования отображаются актеры - пользователи или внешние системы, взаимодействующие с отельной системой через веб-интерфейс. Каждый актер представляет собой роль или тип пользователя, которому доступны определенные функциональные возможности системы. Каждый из актеров может пользоваться системой по-своему. Кроме того, у системы есть функционал, который доступен всем актерам.

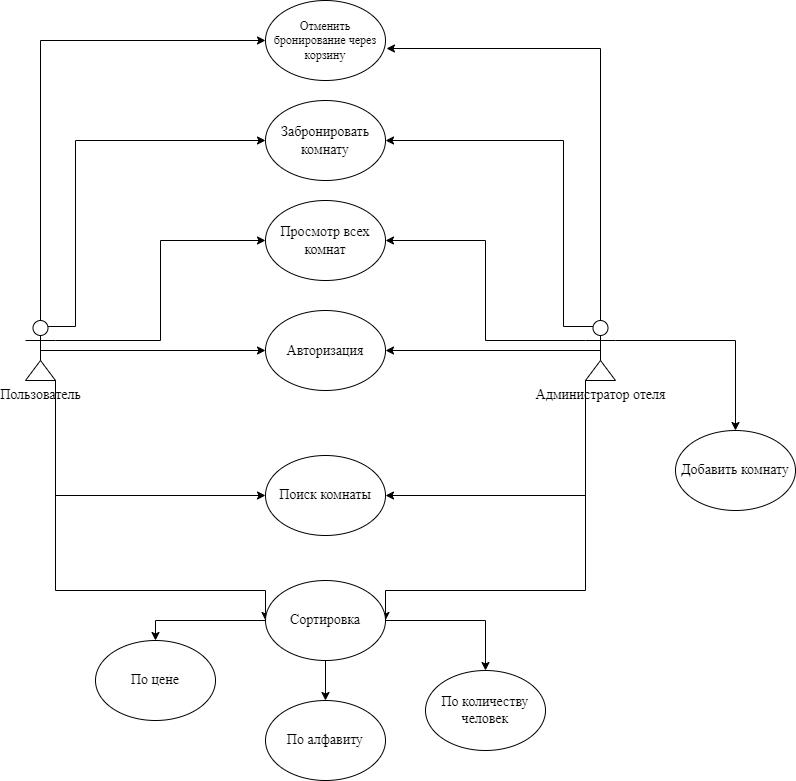


Рисунок 1.5 – Диаграмма вариантов использования

Для представления последовательного взаимодействия между объектами, упорядоченного во времени, применяются последовательностные диаграммы. С использованием таких диаграмм мы получаем набор шагов или операций для каждого отдельного объекта, которые являются неотъемлемыми для достижения целевого результата.

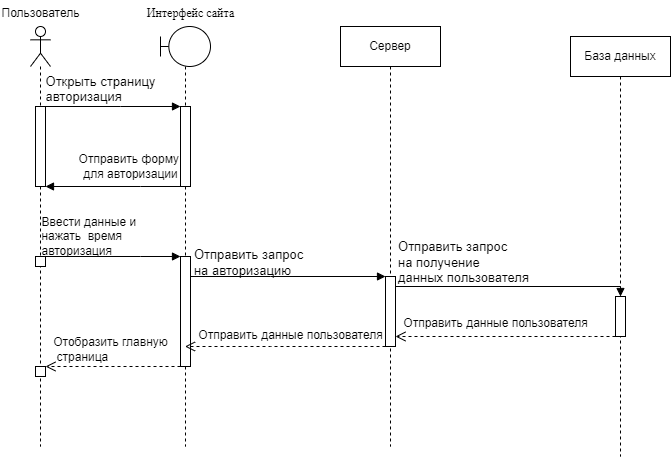


Рисунок 1.6 – Диаграмма последовательности процесса авторизации

Диаграмма активности UML для бронирования номеров в отеле через веб-интерфейс отображает основные этапы процесса и взаимодействие между клиентом и системой. Она помогает лучше понять последовательность действий, необходимых для успешного завершения бронирования, а также подчеркивает важность программного обеспечения для эффективного управления этим процессом.

Пользователь начинает с выбора даты заезда, количества дней пребывания и количества гостей, после чего система осуществляет поиск доступных номеров. Если есть свободные номера, клиент выбирает подходящий вариант. После ввода данных клиент подтверждает бронирование. Таким образом, диаграмма активности UML демонстрирует основные шаги, которые клиенты выполняют для успешного бронирования номеров в отеле через веб-интерфейс, а также подчеркивает важность программной поддержки и автоматизации процесса для повышения удобства и эффективности для клиентов и отельного бизнеса в целом.

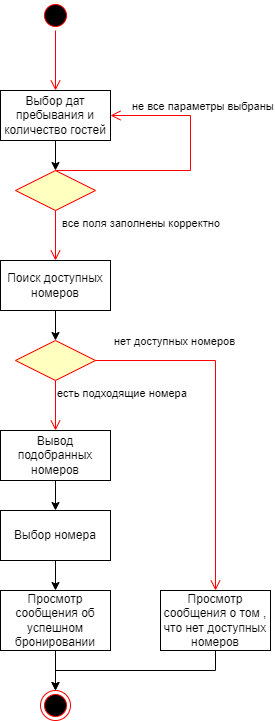


Рисунок 1.7 – Диаграмма событий процесса добавления товара

# ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАМНОГО СРЕДСТВА

## Постановка задачи

Требуется разработать программное средство для организации отельного бизнеса и его программной поддержки в сети интернет. В рамках данного проекта необходимо реализовать информационную систему, которая будет обеспечивать учет и управление номерами, бронирование, а также функции авторизации:

– Учетная запись администратора;

– Учетная запись пользователя.

Необходимо создать программное средство для управления данными отельного бизнеса с использованием авторизации. Данные о номерах, бронировании и других сущностях хранятся в отдельном файле. Разработка программного средства должна включать два функциональных модуля, обеспечивающих работу с данными после успешной авторизации.:

* модуль администратора;
* модуль пользователя.

Модуль администратора включает следующие возможности:

* просмотр всех номеров отеля;
* добавление нового номера;
* удаление записи номера;
* редактирование записи;
* поиск данных;
* сортировка.

Модуль пользователя имеет следующими функциональные возможности:

* просмотр всех номеров отеля;
* бронирование номера;
* отмена брони номера;
* поиск данных;
* сортировка.

## Обоснование компонентов и технологий для реализации программного средства

В данной работе использованы основные фреймворки и технологии, такие как jQuery, DOM, Node.js, Express, JavaScript, HTML и CSS, для разработки программного средства. Для хранения данных была выбрана база данных MongoDB, выбор данных технологий обусловленный следующими причинами:

* Гибкость схемы: MongoDB позволяет хранить документы различной структуры в одной коллекции, что облегчает работу с комплексными иерархическими отношениями. В отличие от SQL, где требуется заранее определить фиксированную схему таблицы перед вставкой данных.Компоненты и технологии, такие как jQuery, DOM, Node.js и Express, обладают широкими возможностями расширения и гибкости в разработке. Они предоставляют различные модули, плагины и библиотеки, которые позволяют удовлетворить разнообразные требования функциональности в программном средстве.
* Масштабируемость: MongoDB поддерживает архитектуру шардинга, позволяющую легко распределять данные по нескольким серверам. Это делает базу данных более подходящей для высоконагруженных приложений, требующих горизонтального масштабирования. В то время как SQL-базы данных обычно используют вертикальное масштабирование, которое может быть дорогостоящим и ограничивать доступные ресурсы.
* Скорость разработки и прототипирование: Использование JavaScript, HTML и CSS позволяет быстро создавать прототипы и демонстрировать функциональность программного средства. Они также обеспечивают удобную среду разработки, богатые инструменты отладки и возможности визуализации пользовательского интерфейса.
* Скорость разработки и прототипирование: Использование JavaScript, HTML и CSS позволяет быстро создавать прототипы и демонстрировать функциональность программного средства. Они также обеспечивают удобную среду разработки, богатые инструменты отладки и возможности визуализации пользовательского интерфейса.
* Эффективность и производительность: MongoDB, выбранная в качестве базы данных, обеспечивает высокую производительность и эффективность при работе с большими объемами данных. Она предлагает механизмы индексирования и оптимизации запросов, что способствует быстрому доступу к информации и выполнению операций записи и чтения.
* Удобство для разработчика: Формат документов MongoDB, основанный на JSON, является интуитивным для разработчиков, работающих с современными веб- и мобильными приложениями. В отличие от SQL, работа с MongoDB может быть более простой и понятной при выполнении сложных запросов.

## Архитектурные решения

Приложение организовано по клиент-серверной модели, которая является широко применяемым подходом в разработке программных средств. Клиент-серверная архитектура предполагает разделение приложения на две основные компоненты: клиентскую часть, которая работает на стороне пользователя, и серверную часть, которая обрабатывает запросы клиента и управляет доступом к ресурсам, таким как база данных.

В данном приложении серверная часть реализована с использованием Node.js - среды выполнения JavaScript, которая позволяет запускать серверный код на стороне сервера. Node.js обладает высокой производительностью и масштабируемостью, что делает его популярным выбором для разработки серверных приложений. Сервер обеспечивает взаимодействие с клиентской частью приложения через API (Application Programming Interface) и управляет доступом к базе данных MongoDB для хранения данных. HTTP-запросы используются для передачи данных и команд между клиентом и сервером.

Клиентская часть приложения отвечает за отображение данных, взаимодействие с пользователем и отправку HTTP-запросов на сервер. Она использует HTML для определения структуры страницы, CSS для оформления и стилей, а JavaScript и jQuery для обработки событий и выполнения динамических операций на стороне клиента.

Использование архитектурного шаблона MVC позволяет нам упростить разработку и поддержку приложения, разделяя его функциональности на логические компоненты и обеспечивая независимость между ними. Это также способствует расширяемости и масштабируемости приложения, так как каждый компонент может быть изменен или заменен отдельно от остальных.

Таким образом, использование Node.js для серверной части, MongoDB для хранения данных, а также архитектурного шаблона MVC обеспечивает надежную и эффективную программную поддержку организации отельного бизнеса в сети интернет.

## Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства

В данном разделе будет представлено описание алгоритмов, которые реализуют ключевую бизнес-логику разрабатываемого программного средства. Ключевая бизнес-логика относится к основным функциональным возможностям приложения, которые позволяют пользователю осуществлять важные операции и достигать своих целей.

Алгоритм добавления комнаты отеля является одним из таких ключевых алгоритмов. Его целью является добавление новой комнаты в систему отеля, чтобы она была доступна для бронирования клиентами. Алгоритм добавления комнаты отеля (рисунок 2.1).



Рисунок 2.1 – Алгоритм добавления комнаты

function add(){

let name=document.getElementById("name\_room").value.toString()

let type= document.getElementById("roomType").value.toString()

let guest=document.getElementById("guest").value.toString()

let photo=photoUrl.toString()

let cost=document.getElementById("roomPrice").value

let internet=document.getElementById("roomAmenitiesInternet").checked

let breakfast=document.getElementById("roomAmenitiesBreakfast").checked

let dryer=document.getElementById("roomAmenitiesHairDryer").checked

let kettle= document.getElementById("roomAmenitiesKettle").checked

console.log(name, dryer,type,guest,internet,photo)

if(name==="" || type =="" || guest=="" || photo=="" ||cost=="" )

{

alert("Заполните все поля!")

return

}

$.ajax("http://127.0.0.1:3000/add\_room.html/add", {

method: "POST",

data: JSON.stringify({

"name\_room":name ,

"type":type,

"guest": guest,

"photoUrl":photo,

"cost":cost,

"internet":internet ,

"breakfast":breakfast ,

"dryer":dryer ,

"kettle":kettle,

"boking":[],

"date": new Date().toDateString(),

}),

contentType: "application/json",

success: function (data) {

alert(data.message);

getData();

},

error: function (err) {

console.log(err);

},

});

}

app.post("/add\_room.html/add", (req, res) => {

console.log(req.body);

db.collection("room")

.insertOne(req.body)

.then(() => {

res.status(201).json({ message: "Успешно добавлено" });

})

.catch((error) => {

res.status(500).json({ message: "Ошибка при удалении объекта" }); });

});

Алгоритм сортировки комнат по стоимости представляет собой процесс упорядочивания списка комнат в информационной системе отеля на основе их стоимости. Данный алгоритм позволяет пользователям быстро и удобно находить комнаты, соответствующие их предпочтениям по цене.

Основная идея алгоритма заключается в использовании метода сортировки данных в базе данных. В данном случае используется MongoDB, которая предоставляет функционал сортировки результатов запросов. (рисунок 2.2)

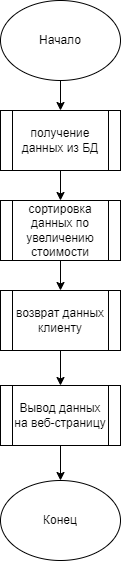


Рисунок 2.2 – Алгоритм добавления комнаты

function sortCost() {

$.ajax("http://127.0.0.1:3000/sortCost", {

method: "post",

success: function (data) {

console.log("Data " + data);

showCard(data);

},

});

}

app.post("/sortCost", (req, res) => {

console.log("sort call");

const rooms = [];

db.collection("room")

.find().sort({ cost: 1 })

.toArray()

.then((rooms) => {

console.log("Rooms: ", rooms);

res.status(201).json(rooms);

console.log("send");

})

.catch(() => {

res.status(500).json({ message: "Ошибка при удалении объекта" });

});

});

Алгоритм удаления номера из базы данных является одним из ключевых алгоритмов приложения. Он администратору позволяет удалить указанный номер из коллекции "room" в базе данных . Алгоритм удаления номера из базы данных обеспечивает эффективное и безопасное удаление данных, сохраняя целостность и согласованность базы данных. Он играет важную роль в функционировании приложения, позволяя пользователям удалить ненужные номера и поддерживать актуальность данных. (рисунок 2.3)



Рисунок 2.3 – Алгоритм удаления комнаты

function deleteRoom(){

const selectElement = document.getElementById("roomTypet");

const selectedValue = selectElement.value;

// console.log(selectedValue); // Вывод выбранного значения в консоль

$.ajax("http://127.0.0.1:3000/del", {

method: "POST",

data: JSON.stringify({

name:selectedValue ,

}),

contentType: "application/json",

success: function (data) {

alert(data.message);

},

error: function (err) {

console.log(err);

},

});

}

app.post("/del", (req, res) => {

console.log(req.body.name);

db.collection("room")

.deleteOne({ name\_room: req.body.name })

.then((result) => {

// Обработка успешного удаления объекта

console.log("Object deleted");

res.status(201).json({ message: "Объект удален" });

})

.catch((error) => {

// Обработка ошибки удаления

console.error("Error deleting object:", error);

res.status(500).json({ message: "Ошибка при удалении объекта" });

});

});

## Проектирование пользовательского интерфейса

Программное средство имеет пользовательский интерфейс, который был разработан с использованием языка разметки HTML и каскадной таблицы стилей CSS. Каждая страница программного средства содержит шапку, которая содержит навигационное меню для удобства перемещения по сайту, а также корзину клиента, которая отображает список забронированных им номеров, за исключением страницы авторизации. Кроме того, каждая страница содержит подвал, который содержит краткую информацию о гостинице.

Этот интерфейс является одним из ключевых элементов программного средства, так как он обеспечивает пользовательскую интеракцию с функциональностью, предоставляемой программным средством. Пользовательский интерфейс должен быть удобным и интуитивно понятным для обеспечения легкой навигации и использования программного средства. Он также должен быть эстетически приятным, чтобы привлекать потенциальных пользователей и создавать положительное впечатление об отеле, который представлен в программном средстве. Пример кода на рисунке 2.4, 2.5,2.6 и 2.7.

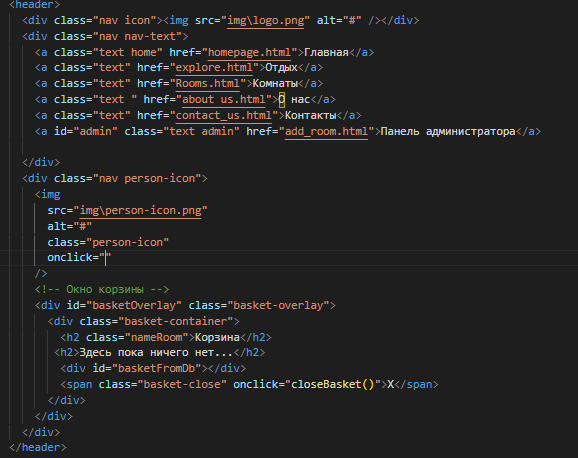


Рисунок 2.4 – Шапка для навигации и просмотра корзины

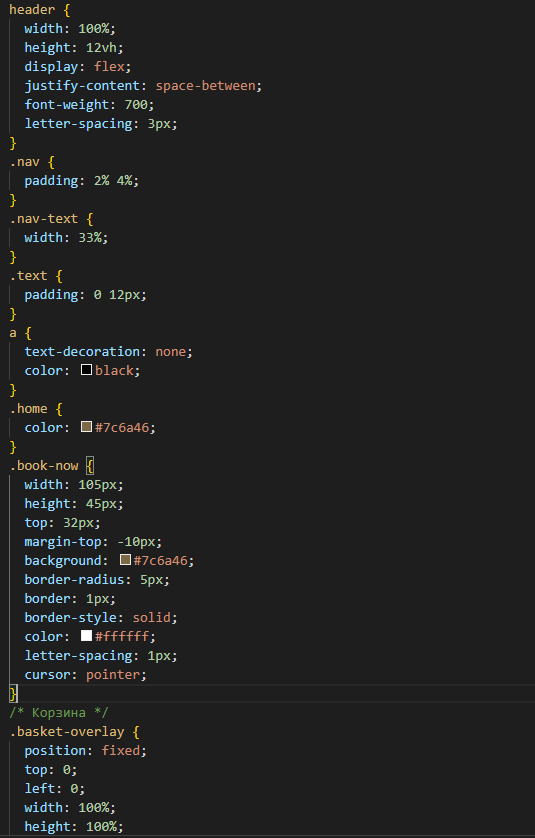


Рисунок 2.4 – Таблица стилей для шапки

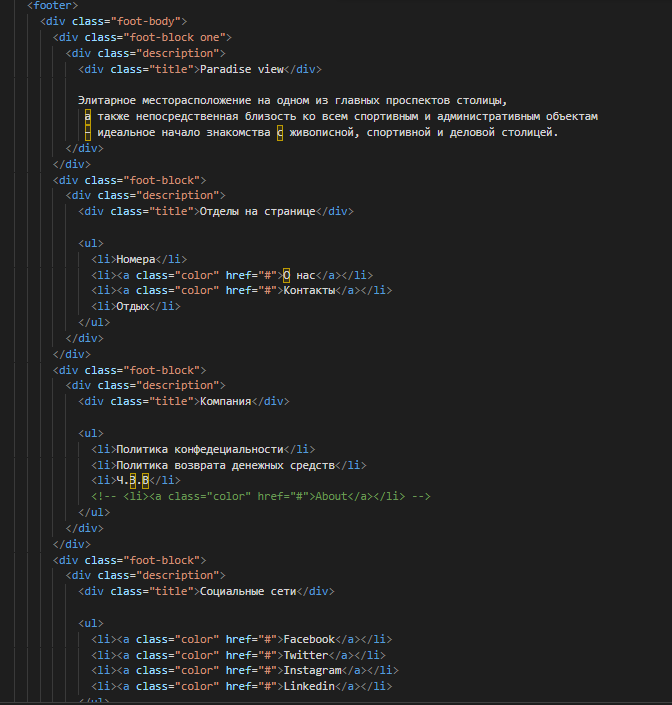


Рисунок 2.6 – Подвал с краткой информацией о гостинице



Рисунок 2.7 – Таблица стилей для подвала страницы

Так же на в корзину динамически выводятся забронированные номера и даты на которые они забронированы. Код добавления на рисунках 2.8 и 2.9.



Рисунок 2.8 – Динамический вывод информации о актуальных бронированиях пользователя



Рисунок 2.10 – Разметка для вывода бронирований

## Методы и средства, используемые для обеспечения безопасности данных

В данном программном обеспечении используется шифрование пароля. Библиотека для шифрования «bcrypt». Пароль хранится в БД в виде хеш-значения (рисунок 2.11). При авторизации, пароль отправленный на сервер сравнивается с хеш-значением из БД с помощью метода «compare» библиотеки «bcrypt» (рисунок 2.12), в лямбда-функцию приходит «result» являющийся истинной или ложью. В зависимости от его значения определяем есть ли совпадение.

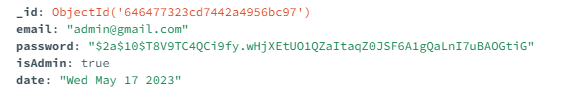


Рисунок 2.11 – Пример хеш-значения пароля

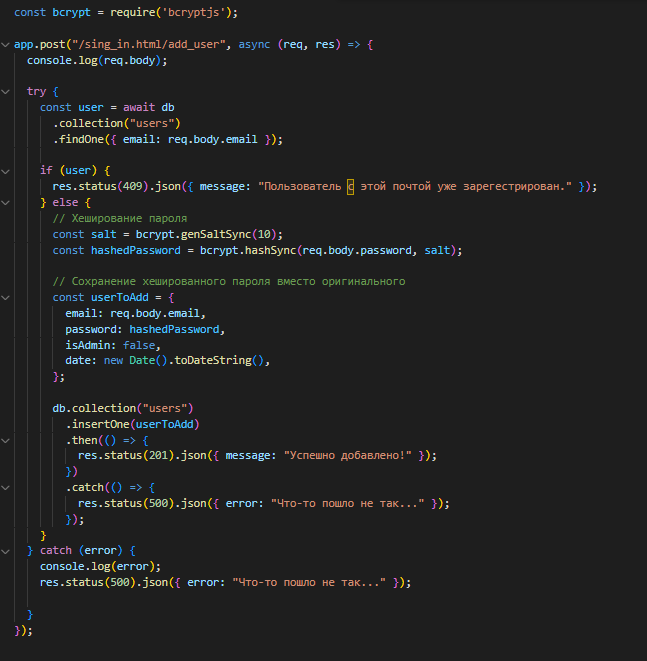


Рисунок 2.12 – Метод «compare» библиотеки «bcrypt»

# ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММЫ

Если данные при авторизации введены не верно, то выводится следующее сообщение (рисунок 3.1).

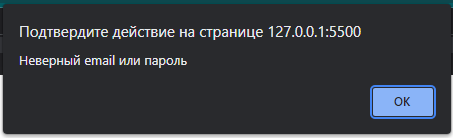


Рисунок 3.1 – Обработка ситуации с некорректным вводом данных для авторизации

При попытке зарегистрировать пользователя на уже существующий в базе емеил выводится сообщение(рисунок 3.2)

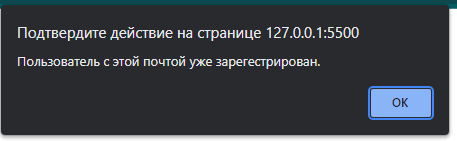


Рисунок 3.2 – Обработка ситуации с некорректной регистрацией

Присутствует проверка на заполнение всех дат при бронировании номера (рисунок 3.3)

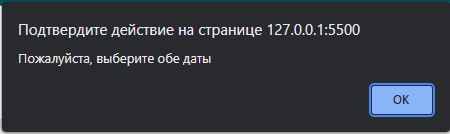


Рисунок 3.3 – Обработка ситуации с некорректными данными

Присутствует сообщение об успешном бронировании номера(рисунок 3.4)

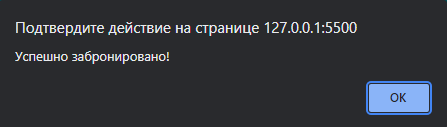


Рисунок 3.2 – Обратная связь при успешном бронировании

При успешной отмене бронирования выводится сообщение как на рисунке 3.5

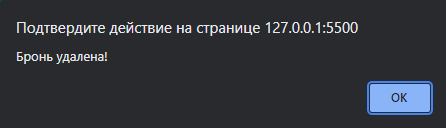


Рисунок 3.5 – Обратная связь при отмене бронирования

Обработка ситуации, когда при добавлении комнаты не все данные введены корректно или не введены вовсе(рисунок 3.6)

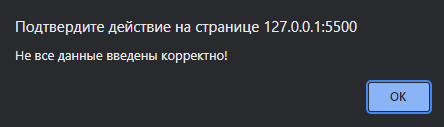


Рисунок 3.6 – Обработка ситуации, когда не все данные введены

# ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

## Авторизация

При запуске программы открывается страница с авторизацией (Рисунок 4.1).

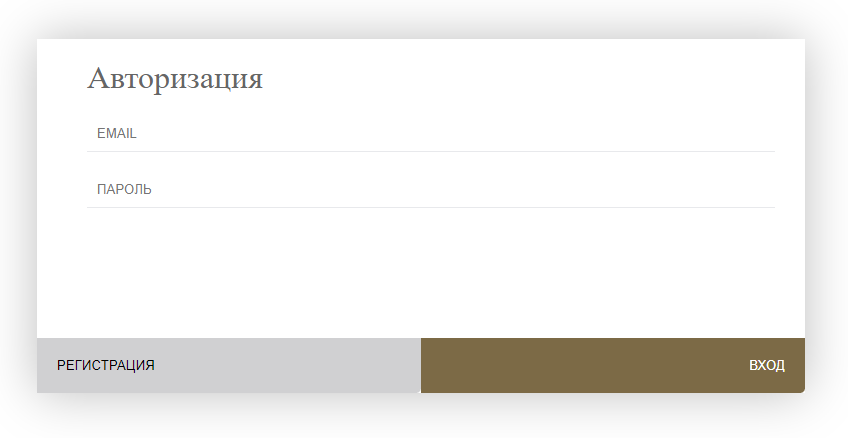


Рисунок 4.1 – Меню авторизации

Далее вводится логин и пароль (Рисунок 4.2).

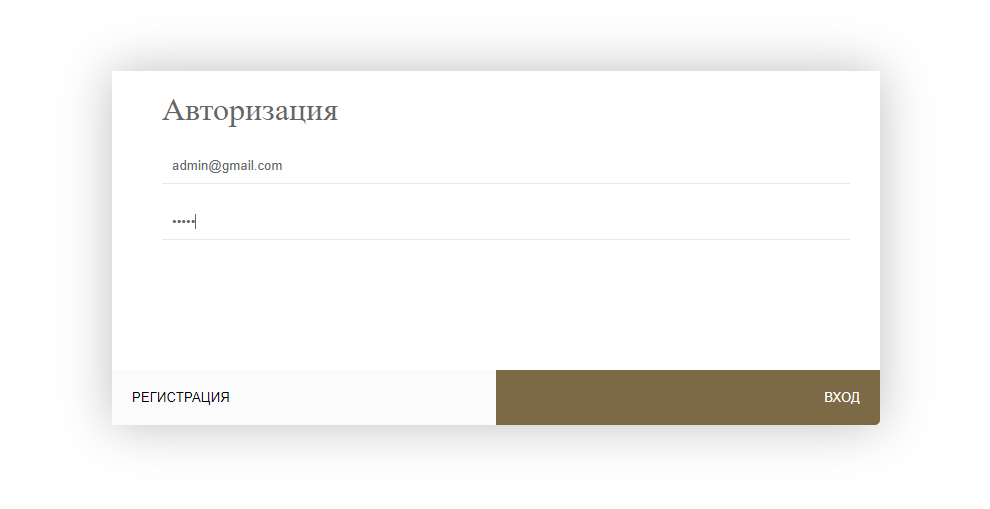


Рисунок 4.2 – Процесс авторизации

## Модуль администратора

При авторизации как администратор в шапке выводится меню как на рисунке 4.3.

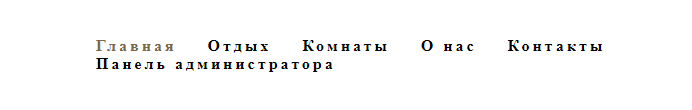


Рисунок 4.5 – Навигационное меню администратора

При выборе пункта «Панель администратора» открывается страница доступная только администратору, на которой есть возможности добавления, удаления и редактирования данных комнаты(рисунок 4.4).

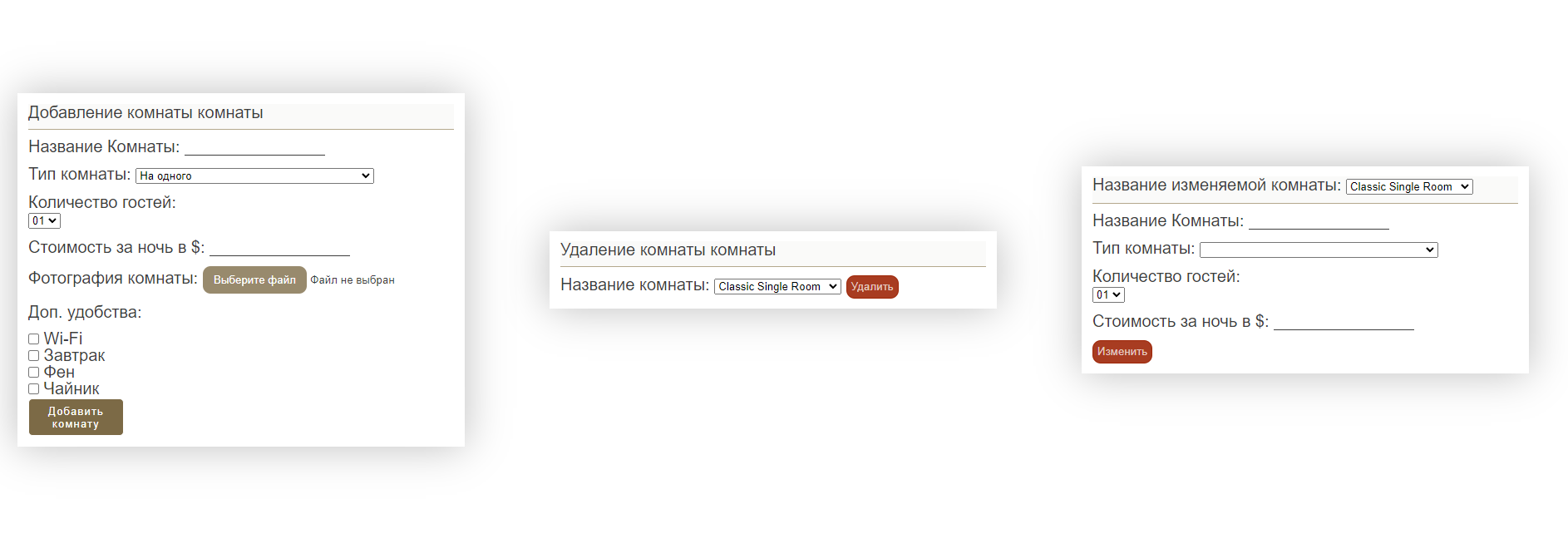


Рисунок 4.4 – Страница возможностей администратора

## Модуль пользователя

При выборе модуля "Пользователь" на главной странице запускается пользовательский интерфейс, который отличается от интерфейса модуля администратора. В пользовательском интерфейсе предоставляется навигационное меню, специально адаптированное для потребностей обычных пользователей. Так же на главной странице расположена форма после выполнения которой перейдет на страницу с номерами подходящими под его желаемые даты пребывания (Рисунок 4.5).

Это позволяет пользователям легко находить и выбирать подходящие комнаты для своего пребывания в отеле. Кроме того, на главной странице доступны и другие функциональности, например, просмотр уже сделанных бронирований пользователя и возможность их отмены, а также другие опции и сервисы, которые облегчают использование системы для пользователя.

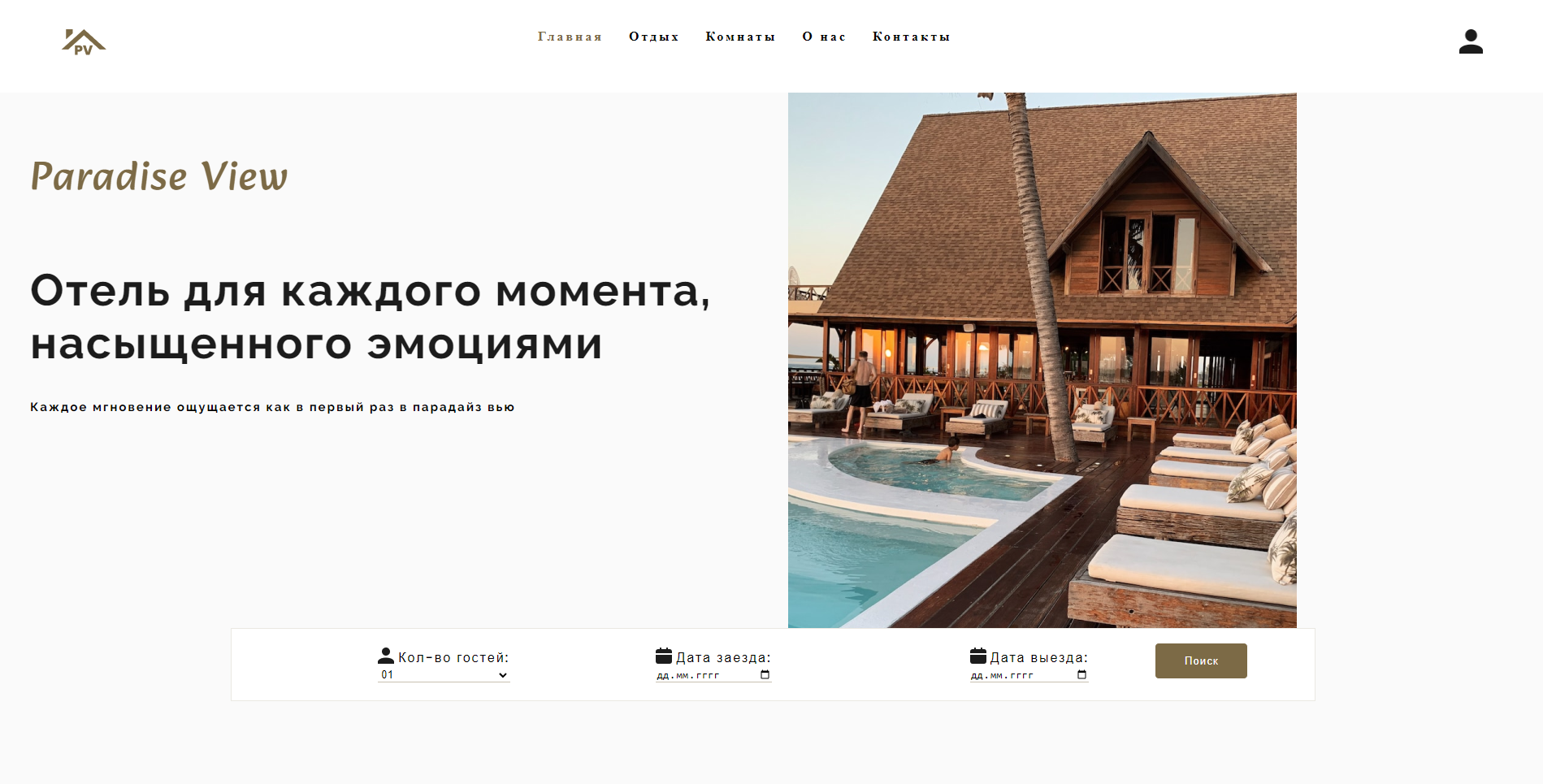


Рисунок 4.5 – Главная страница модуль пользователя

При выборе пункта "Комнаты" открывается страница, на которой отображаются доступные номера отеля. Информация о номерах хранится в базе данных. Пользователь имеет возможность забронировать выбранный номер, просмотреть уже сделанные брони в корзине и при необходимости отменить их..(рисунок 4.6 и 4.7)

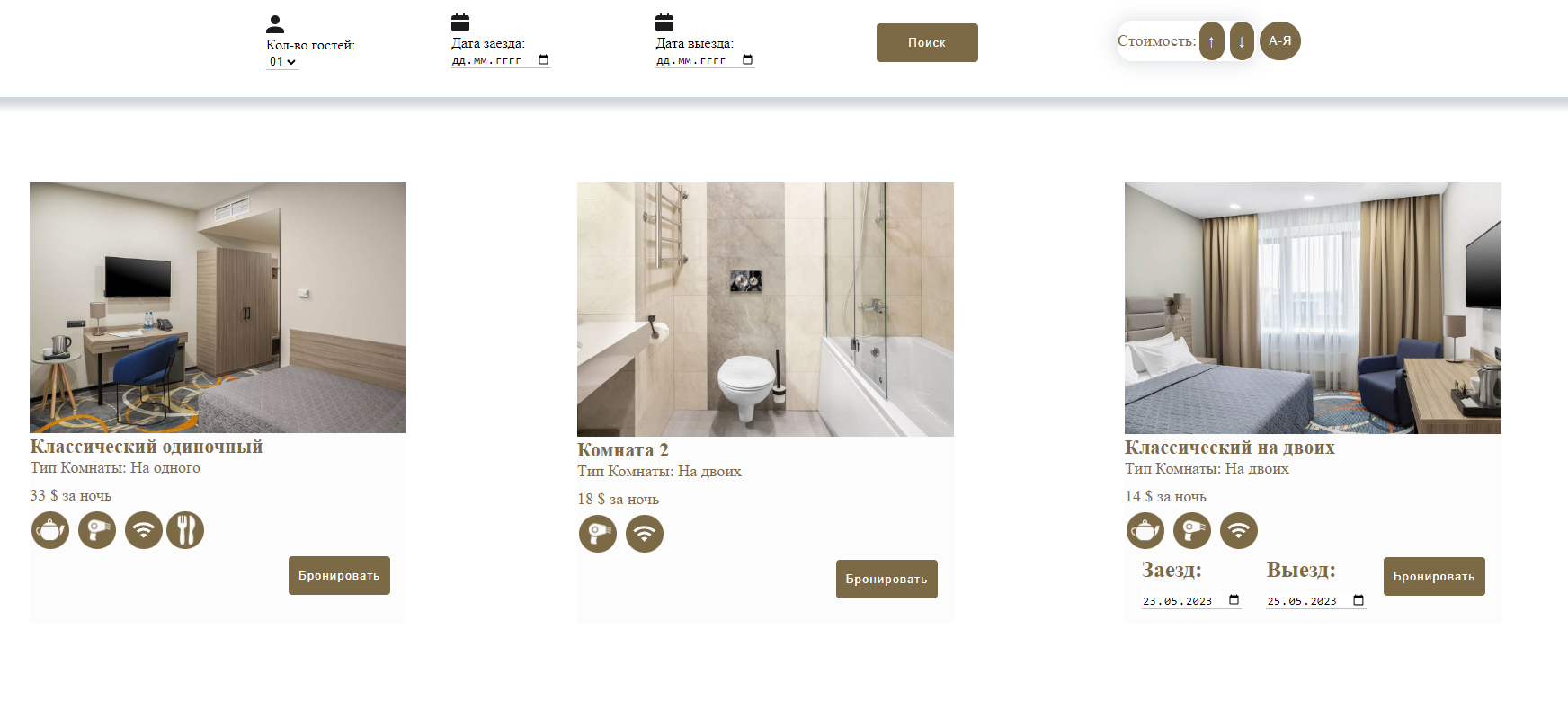


Рисунок 4.5 – Страница с номерами отеля

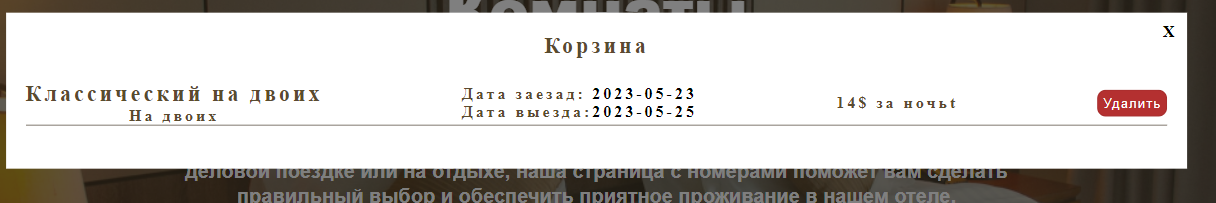


Рисунок 4.5 – Корзина пользователя

## ЗАКЛЮЧЕНИЕ

Организация отельного бизнеса и его программная поддержка в сети интернет играют ключевую роль в современной индустрии гостеприимства. Внедрение эффективных программных решений и использование онлайн-платформ позволяют отельному бизнесу значительно улучшить свою эффективность, повысить уровень обслуживания гостей и расширить свою клиентскую базу.

С помощью программной поддержки отельные предприятия могут автоматизировать рутинные операции, такие как бронирование номеров, учет гостей, управление инвентарем и финансовым учетом. Это позволяет сократить время и ресурсы, улучшить точность данных и минимизировать возможность ошибок. Кроме того, программные решения также обеспечивают централизованное управление, что упрощает контроль и координацию всех аспектов отельного бизнеса.

Использование интернет-платформ и онлайн-сервисов позволяет отелям эффективно привлекать и обслуживать своих клиентов. Онлайн-бронирование номеров, взаимодействие с клиентами через электронные каналы связи, предоставление информации о номерах, услугах и акциях – все это делает процесс поиска и бронирования комфортного проживания более удобным и доступным для гостей. Более того, наличие онлайн-рейтингов и отзывов позволяет клиентам принимать осознанные решения при выборе отеля и повышает уровень доверия.

Программная поддержка и наличие присутствия в сети интернет являются неотъемлемой частью современного отельного бизнеса. Они обеспечивают эффективность, удобство и конкурентоспособность отелей, а также улучшают общий опыт гостей. Эти технологические инструменты позволяют отельному бизнесу успешно адаптироваться к изменяющимся потребностям рынка и создать прочную основу для долгосрочного успеха.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] MDN Web Docs. "JavaScript." Доступно 28 апреля 2023 г. https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide

[2] Черниговский, Дмитрий. Базы данных. Концептуальное проектирование и моделирование. Издательство "БХВ-Петербург", 2019. 416 с.

[3] Хэррон Д. «Node.js. Разработка серверных веб-приложений в JavaScript»: Пер. с англ. Слинкина А.А. - М.: ДМК Пресс, 2012. - 144с.

[4] Документация Node.js. "Начало работы." Доступно 28 апреля 2023 г. https://nodejs.org/en/docs/guides/getting-started-guide/

[5] Кузнецов, Максим и Елена Гусева. Управление онлайн-репутацией гостиницы. Издательство "Лань", 2016. 224 с.

[6] Фримен, Адам, и Тайлер Хенкель. Node.js в действии. Manning Publications, 2017. 416 с.

[7] Пауэлл, Эшли. Изучение Node: Переход на сторону сервера. O'Reilly Media, 2016. 308 с.

[8] Короткова, Юлия. Веб-дизайн и разработка сайта для гостиницы. Издательство "БХВ-Петербург", 2020. 352 с.

[9] Кириченко А. В. Динамические сайты на HTML, CSS, Javascript и Bootstrap. Практика, практика и только практика [Текст] / А. В. Кириченко, Е. В. Дубовик. — 2-е изд. — Санкт-Петербург: Наука и Техника, 2018. — 272 c.

[10] Дакетт Д. JavaScript и jQuery. Интерактивная веб-разработка [Текст] / Д. Дакетт. — Москва: Издательство «Э», 2017 — 640 с.

[11] Хоффман, Эрик и Дэвид Лам. Интернет-маркетинг для гостиничного бизнеса. Издательство "Манн, Иванов и Фербер", 2015. 368 с.

[12] ГОСТ СТП 01-2017 Дипломные проекты(работы). Общие требования. – Введ. 01.01.2014. – Минск: БГУИР, 2014. – 173 с.

[13] Дакетт, Джон. HTML и CSS: Дизайн и создание веб-сайтов. John Wiley & Sons, 2011. 490 с.

[14 Пауэлл, Эшли. Изучение Node: Переход на сторону сервера. O'Reilly Media, 2016. 308 с.

# ПРИЛОЖЕНИЕ А Листинг кода алгоритмов, реализующих основную бизнес-логику.

let photoUrl;

function download() {

var file = document.getElementById("roomImageUpload").files[0];

var reader = new FileReader();

try {

reader.readAsDataURL(file);

reader.onloadend = function () {

photoUrl = reader.result;

console.log(photoUrl);

add();

};

} catch (error) {

alert("Не все данные введены корректно!");

console.log("Ошибка чтения файла:", error);

}

}

document.addEventListener("DOMContentLoaded", function () {

// Код JavaScript

$.ajax("http://127.0.0.1:3000/Rooms", {

method: "post",

success: function (data) {

console.log("Data " + data);

showCard(data);

showCardEdit(data)

},

});

})

function showCard(data) {

console.log(data[0]);

let rooms = document.getElementById("roomTypet");

rooms.innerHTML = "";

for (var i = 0; i < data.length; i++) {

let cardHTmL = `

<option value="${data[i].name\_room}">${data[i].name\_room}</option>`;

rooms.insertAdjacentHTML("beforeend", cardHTmL);

}

console.log(rooms);

}

function showCardEdit(data) {

console.log(data[0].\_id);

// let rooms = document.getElementById("roomTypet");

let roomsEdit = document.getElementById("roomedit");

for (var i = 0; i < data.length; i++) {

let cardHTmL = `

<option value="${data[i].\_id}">${data[i].name\_room}</option>`;

// rooms.insertAdjacentHTML("beforeend", cardHTmL);

roomsEdit.insertAdjacentHTML("beforeend", cardHTmL);

}

}

function deleteRoom(){

const selectElement = document.getElementById("roomTypet");

const selectedValue = selectElement.value;

// console.log(selectedValue); // Вывод выбранного значения в консоль

$.ajax("http://127.0.0.1:3000/del", {

method: "POST",

data: JSON.stringify({

name:selectedValue ,

}),

contentType: "application/json",

success: function (data) {

alert(data.message);

},

error: function (err) {

console.log(err);

},

});

}

function add(){

let name=document.getElementById("name\_room").value.toString()

let type= document.getElementById("roomType").value.toString()

let guest=document.getElementById("guest").value.toString()

// let photo=photoUrl.toString()

let cost=parseInt(document.getElementById("roomPrice").value)

let internet=document.getElementById("roomAmenitiesInternet").checked

let breakfast=document.getElementById("roomAmenitiesBreakfast").checked

let dryer=document.getElementById("roomAmenitiesHairDryer").checked

let kettle= document.getElementById("roomAmenitiesKettle").checked

console.log(name, dryer,type,guest,internet,photoUrl.toString())

if(name==="" || type =="" || guest=="" || photoUrl.toString()=="" ||cost=="" )

{

alert("Заполните все поля!")

return

}

$.ajax("http://127.0.0.1:3000/add\_room.html/add", {

method: "POST",

data: JSON.stringify({

"name\_room":name ,

"type":type,

"guest": guest,

"photoUrl":photoUrl.toString(),

"cost":cost,

"internet":internet ,

"breakfast":breakfast ,

"dryer":dryer ,

"kettle":kettle,

"boking":[],

"date": new Date().toDateString(),

}),

contentType: "application/json",

success: function (data) {

alert(data.message);

// getData();

},

error: function (err) {

console.log(err);

},

});

}

function Edit(){

let name=document.getElementById("name\_room1").value.toString()

let type= document.getElementById("roomType1").value.toString()

let guest=document.getElementById("guest1").value.toString()

// let photo=photoUrl

let cost=parseInt(document.getElementById("roomPrice1").value)

const selectElement = document.getElementById("roomedit");

const selectedValue = selectElement.value;

console.log(cost)

if(name===""){

name="0";

}

if(type ==="" ){

type="0"

}

if(guest===""){

guest="0"

}

// if(!photoUrl){

// photoUrl="0"

// }

if(cost===null ){

cost=0

}

$.ajax("http://127.0.0.1:3000/edit", {

method: "POST",

data: JSON.stringify({

"id": selectedValue ,

"name\_room":name ,

"type":type,

"guest": guest,

// "date": new Date().toDateString(),

}),

contentType: "application/json",

success: function (data) {

alert(data.message);

// getData();

},

error: function (err) {

console.log(err);

},

});

}

console.log("ajax call");

function getData() {

$.ajax("http://127.0.0.1:3000/Rooms", {

method: "post",

success: function (data) {

console.log("Data " + data);

showCard(data);

},

});

}

function sortByName(){

$.ajax("http://127.0.0.1:3000/sortByName", {

method: "post",

success: function (data) {

showCard(data);

},

});

}

function sortCost(orientation) {

console.log(orientation);

$.ajax("http://127.0.0.1:3000/sortCost", {

method: "post",

data: JSON.stringify({

"sort":orientation,

}),

contentType: "application/json",

success: function (data) {

// console.log("Data " + data);

showCard(data);

},

});

}

getData();

function showCard(data) {

console.log(data[0]);

let rooms = document.getElementsByClassName("rooms")[0];

rooms.innerHTML = "";

for (var i = 0; i < data.length; i++) {

let cards = document.createElement("div");

cards.className = "cards";

let img = document.createElement("img");

// let svg = document.createElement("img");

img.src = data[i].photoUrl;

img.className = "foto-room";

// document.getElementsByClassName("rooms")[0]+=data[i]

cards.appendChild(img);

let breakfast = document.createElement("img");

breakfast.src = "icon/breakfast.svg";

let dryer = document.createElement("img");

dryer.src = "icon/dryer.svg";

let wiFi = document.createElement("img");

wiFi.src = "icon/wiFi.svg";

let kettle = document.createElement("img");

kettle.src = "icon/kettle.svg";

if (data[i].kettle) {

kettle.className = "icons";

} else {

kettle.className = "empty-icon";

}

if (data[i].dryer) {

dryer.className = "icons";

} else {

dryer.className = "empty-icon";

}

if (data[i].breakfast) {

breakfast.className = "icons";

} else {

breakfast.className = "empty-icon";

}

if (data[i].internet) {

wiFi.className = "icons";

} else {

wiFi.className = "empty-icon";

}

let cardHTmL = `

<div class="card">

${img.outerHTML}

<div class="cost">

<div class="">

<p class="name-room">${data[i].name\_room} </p>

<p class="text-room">Тип Комнаты: ${data[i].type} </p>

</div>

<div class="text-room">${data[i].cost} $ за ночь</div>

${kettle.outerHTML} ${dryer.outerHTML}

${wiFi.outerHTML}${breakfast.outerHTML}

</div>

<div class="nav button booking">

<button class="book-now" onclick="booking('${data[i].\_id}')">Бронировать</button>

<div class="date-room" id="${data[i].\_id}" >

<div class="chek-date">

<section>

<label for="check-date">

<p class="text-sort">Заезд:</p>

</label>

<input id="check-in-date" class="inp-sort ${data[i].\_id}" type="date" name="check-date" required />

</section>

</div>

<div class="chek-date">

<section>

<label for="check-date">

<p class="text-sort">Выезд:</p>

</label>

<input id="check-out-date" class="inp-sort ${data[i].\_id}" type="date" name="check-date" />

</section>

</div>

</div>

</div>

</div>

`;

rooms.innerHTML += cardHTmL;

}

}

const arrowBlock = document.querySelector(".arrow");

arrowBlock.addEventListener("click", () => {

window.scrollTo({

top: window.pageYOffset + 900,

behavior: "smooth",

});

});

function checkDates(dateIn, dateOut) {

if (dateIn || dateOut)

return false;

else return true;

}

function booking(roomId) {

console.log(roomId);

checkInDate = localStorage.getItem("check-in");

checkOutDate = localStorage.getItem("check-out");

email = localStorage.getItem("email");

if (!checkInDate || !checkOutDate) {

var dateRoomElement = document.getElementById(roomId);

// Устанавливаем свойство "display" элемента на "flex"

dateRoomElement.style.display = "flex";

var checkDate = document.getElementsByClassName(roomId);

checkInDate = checkDate[0].value;

checkOutDate = checkDate[1].value;

if (!checkInDate || !checkOutDate) {

// Одна или обе даты не выбраны

console.log("Выберите обе даты");

alert("Пожалуйста, выберите обе даты");

} else {

console.log(checkInDate, checkOutDate);

// Дополнительная логика или действия

$.ajax("http://127.0.0.1:3000/booking", {

method: "POST",

data: JSON.stringify({

booking: {

email: email,

"check-in": checkInDate,

"check-out": checkOutDate,

},

roomId: roomId,

}),

contentType: "application/json",

success: function (data) {

alert(data);

localStorage.removeItem("check-in");

localStorage.removeItem("check-out");

},

error: function (err) {

console.log(err);

},

});

}

} else {

$.ajax("http://127.0.0.1:3000/booking", {

method: "POST",

data: JSON.stringify({

booking: {

email: email,

"check-in": checkInDate,

"check-out": checkOutDate,

},

roomId: roomId,

}),

contentType: "application/json",

success: function (data) {

alert(data);

localStorage.removeItem("check-in");

localStorage.removeItem("check-out");

},

error: function (err) {

console.log(err);

},

});

}

}

if(localStorage.getItem("newWindow")){

search()

localStorage.removeItem("newWindow");

}

function search() {

var selectElement = document.getElementById("choice");

var selectedValue = selectElement.value;

var datetElements = document.getElementsByClassName("inp-sort");

var outDate = datetElements[2].value;

var inDate = datetElements[1].value;

if(selectedValue === undefined||inDate.toString() ===""||outDate.toString() ===""){

selectedValue=localStorage.getItem("guest")

inDate=localStorage.getItem("check-in")

outDate=localStorage.getItem("check-out")

if(selectedValue === undefined||inDate.toString() ===""||outDate.toString() ===""){

alert("Заполните все поля!");

return;

}

}

localStorage.setItem("guest", selectedValue);

localStorage.setItem("check-in", inDate);

localStorage.setItem("check-out", outDate);

console.log(inDate);

$.ajax("http://127.0.0.1:3000/search", {

method: "POST",

data: JSON.stringify({

checkIn: inDate,

checkOut: outDate,

guest: selectedValue,

}),

contentType: "application/json",

success: function (response) {

// Обработка ответа от сервера

console.log(response);

showCard(response);

},

error: function (error) {

// Обработка ошибки

console.log(error);

},

});

}

# ПРИЛОЖЕНИЕ Б Отчет о проверке на заимствование в системе «Антиплагиат»

