Изображение выглядит как Шрифт, Графика, логотип, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ОТЧЕТ О ПРОХОЖДЕНИИ ПРАКТИКИ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место прохождения практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по 14 июля 2025 г.

Выполнил:

Студент гр. ФИ201 14 июля 2025 г. В.В. Иванов

Руководитель практики

от профильной организации

Заведующий кафедрой ИТ 14 июля 2025 г. А.И. Пестунов

Руководитель практики

от университета

Кандидат технических наук, доцент 14 июля 2025 г. А.Л. Осипов

Новосибирск 2025

Изображение выглядит как Шрифт, Графика, логотип, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ЗАДАНИЕ НА ПРАКТИКУ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по 14 июля 2025 г.

Выдано студенту 3 курса ФИ201 группы

Иванову Валерию Викторовичу

Индивидуальное задание на практику, содержание, планируемые результаты: целью практики является освоение и применение современных технологий разработки веб-приложений на примере реализации программного приложения «База данных отелей». Приложение должно предоставлять пользователю возможность просмотра, поиска отелей.

Рабочий график (план) проведения практики

|  |  |
| --- | --- |
| Этапы практики | Период |
| Ознакомление с темой. | 17.06.25-18.06.25 |
| Составление плана работы, цели, задачи. | 18.06.25-19.06.25 |
| Реализация информационной системы. | 19.06.25-30.06.25 |
| Анализ полученных результатов, оформление и подготовка к защите отчета по практике | 1.07.25-14.07.25 |

Руководитель практики

от университета

Кандидат технических наук, доцент 17 июня 2025 г. А. Л. Осипов

Задание согласовано

Руководитель практики от профильной организации

Заведующий кафедрой информационных технологий

А.И. Пестунов 17 июня 2025 г.

Задание получено

Иванов Валерий Викторович 17.06.2025

Изображение выглядит как Шрифт, Графика, логотип, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ**

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Вид практики: производственная практика

Тип практики: Производственно-технологическая

Место практики: НГУЭУ, кафедра информационных технологий, 630099, г. Новосибирск, ул. Каменская, 52/1

Сроки прохождения практики: с 17 июня 2025 г. по 14 июля 2025 г.

Студент(ка) 3 курса, ФИ201 группы

Иванов Валерий Викторович

Работал: НГУЭУ

|  |  |  |
| --- | --- | --- |
| Дата | Краткое описание видов работ | Отметка о выполнении |
| 17.06.2025 | Ознакомление с инструкциями организации по правилам противопожарной безопасности, правилам охраны труда, техники безопасности; требованиями по соблюдению санитарно-эпидемиологических правил и гигиенических нормативов; планами эвакуации при возникновении пожара; правилами внутреннего трудового распорядка | ✓ |
| 18.06.2025 | Подборка и анализ научных источников для реализации информационной системы | ✓ |
| 19.06.2025 | Проектирование архитектуры приложения и структуры базы данных | ✓ |
| 20.06.2025 | Разработка серверной части (REST API) | ✓ |
| 27.06.2025 | Разработка клиентского интерфейса | ✓ |
| 9.07.2025 | Тестирование, демонстрация работы приложения | ✓ |
| 11.07.2024 | Составление и оформление отчета о прохождении практики | ✓ |
| 14.07.2024 | Защита отчета о прохождении практики | ✓ |

Руководитель практики от профильной организации

А. И. Пестунов 14.07.2025 г.

Работы выполнены в установленные сроки**,** содержание практики соответствует индивидуальному заданию.

Руководитель практики

от университета

Кандидат технических наук, доцент 14 июля 2025 г. А. Л. Осипов

**Содержание**

[1. Введение 2](#_Toc202818299)

[2. Постановка задачи 4](#_Toc202818300)

[3. Описание проекта 5](#_Toc202818301)

[4. Реализация проекта 11](#_Toc202818302)

[5. Тестирование и результаты 13](#_Toc202818303)

[6. Заключение 16](#_Toc202818304)

[7. Список литературы 17](#_Toc202818305)

# Введение

В рамках учебной практики я разработал веб-приложение «База данных отеля», предназначенное для поиска информации об отелях по четырём критериям: название отеля, город, вместимость (количество мест) и цена номера категории «стандарт». Основная цель практики заключалась в приобретении и закреплении практических навыков разработки полнофункциональных веб-приложений, включающих серверную и клиентскую части, а также работу с реляционными базами данных. Проект позволил мне углубить понимание современных технологий веб-разработки, изучить новые инструменты и применить теоретические знания в реальных задачах.

Задачи практики были направлены на комплексное освоение процесса разработки:

* Настройка и использование PostgreSQL для создания и управления базой данных с информацией об отелях и городах.
* Разработка серверной части (бэкенда) на языке программирования Go с использованием фреймворка Gin для создания REST API.
* Создание клиентской части (фронтенда) на React с удобным и интуитивно понятным интерфейсом, включающим форму поиска и таблицу результатов.
* Интеграция всех компонентов приложения для обеспечения корректного взаимодействия между базой данных, бэкендом и фронтендом.
* Реализация обработки ошибок, таких как пустой ввод или некорректные числовые значения, для повышения надёжности приложения.
* Тестирование функциональности и отладка возникающих проблем, включая настройку кросс-доменных запросов (CORS) и устранение ошибок на фронтенде.

Актуальность проекта обусловлена высоким спросом на веб-приложения в туристической индустрии, где системы управления данными об отелях, гостиницах и бронировании становятся всё более востребованными. Такие приложения упрощают доступ к информации для пользователей, будь то туристы, ищущие подходящий отель, или администраторы, управляющие базами данных. Разработка подобных систем требует от программиста навыков работы с базами данных, серверными технологиями и современными интерфейсами, что делает проект значимым для профессионального роста.

В процессе работы я изучил и применил ряд технологий и инструментов, включая PostgreSQL для управления базой данных, язык Go и фреймворк Gin для создания API, библиотеку React для построения интерфейса и Axios для выполнения HTTP-запросов. Также я освоил настройку окружения на Windows, включая установку PostgreSQL, Go и Node.js, а также работу с инструментами, такими как pgAdmin и Visual Studio Code. Для стилизации интерфейса я использовал CSS, обеспечив современный дизайн с адаптивностью для мобильных устройств и анимациями для улучшения пользовательского опыта.

В ходе разработки я столкнулся с рядом технических проблем, которые потребовали глубокого анализа и поиска решений. Среди них — настройка кодировки UTF-8 в PostgreSQL для корректной работы с русскими символами, устранение ошибки CORS при взаимодействии фронтенда и бэкенда, а также исправление ошибки "hotels.map is not a function" на фронтенде, связанной с некорректной обработкой ответов API. Эти вызовы позволили мне углубить понимание отладки и взаимодействия компонентов веб-приложений. Для решения задач я изучил официальную документацию PostgreSQL [1], Go [2], Gin [3], React [4], Axios [5] и материалы по CSS [6], что обеспечило успешную реализацию проекта.

Учебная практика не только закрепила мои знания, но и развила навыки самостоятельного изучения технологий, анализа проблем и их систематического решения. Проект стал важным шагом в освоении полного цикла разработки веб-приложений, от проектирования базы данных до создания интерактивного интерфейса, и подготовил меня к дальнейшему развитию в области программирования.

# Постановка задачи

Задача проекта — создание веб-приложения для поиска информации об отелях. Пользователь должен выбирать критерий поиска (название отеля, город, вместимость, цена номера), вводить значение и получать результаты в виде таблицы. Мой вариант задания (№8) включает следующие свойства:

* Название отеля (name, список, отдельная таблица).
* Город (city, список, отдельная таблица).
* Вместимость (capacity, целое число).
* Цена номера категории «стандарт» (standard\_room\_price, число).

Требования к проекту:

1. База данных на PostgreSQL с таблицами cities, hotels, hotel\_cities, каждая с 15 записями.
2. Бэкенд на Go с Gin, предоставляющий REST API с эндпоинтом /hotels.
3. Фронтенд на React с формой поиска (select, input, кнопка) и таблицей результатов.
4. Современный дизайн с адаптивностью и анимациями (используется CSS).
5. Обработка ошибок: пустой ввод, некорректные числовые значения.
6. Полнофункциональное приложение с корректным поиском по всем полям.

# Описание проекта

Проект представляет собой веб-приложение с клиент-серверной архитектурой, разработанное для предоставления пользователям удобного инструмента для поиска информации об отелях. Приложение состоит из трёх ключевых компонентов: базы данных, серверной части (бэкенда) и клиентской части (фронтенда). Эти компоненты взаимодействуют между собой, обеспечивая выполнение поисковых запросов и отображение результатов в удобном формате. Ниже представлено детальное описание каждого компонента, их взаимодействия, структуры данных и особенностей работы приложения.

Архитектура приложения Архитектура приложения построена на модели клиент-сервер, где:

* База данных (PostgreSQL) хранит информацию об отелях и городах, обеспечивая надёжное и структурированное хранение данных.
* Бэкенд, реализованный на языке Go с использованием фреймворка Gin, предоставляет REST API для обработки запросов от фронтенда и взаимодействия с базой данных.
* Фронтенд, разработанный на React, обеспечивает интерактивный пользовательский интерфейс, включающий форму поиска и таблицу для отображения результатов.

Взаимодействие компонентов происходит следующим образом:

1. Пользователь выбирает критерий поиска (например, "название отеля" или "город") через выпадающий список () и вводит значение в текстовое поле ().
2. При нажатии кнопки «Поиск» фронтенд формирует GET-запрос к бэкенду через библиотеку Axios. Пример запроса: http://localhost:8080/hotels?field=name&value=Гранд.
3. Бэкенд обрабатывает запрос, формирует SQL-запрос к базе данных PostgreSQL, извлекает соответствующие данные и возвращает их в формате JSON.
4. Фронтенд принимает JSON-ответ, проверяет его на корректность (например, является ли ответ массивом данных) и отображает результаты в таблице.
5. В случае ошибок (например, пустой ввод или некорректное значение) бэкенд возвращает JSON с сообщением об ошибке, а фронтенд отображает это сообщение пользователю.

Структура базы данных База данных реализована на PostgreSQL и включает три таблицы: cities, hotels и hotel\_cities. Структура таблиц разработана для поддержки связи многие-ко-многим между отелями и городами, что позволяет одному отелю быть связанным с несколькими городами (например, сетевые отели) и одному городу — с несколькими отелями.

* Таблица cities:
  + id (SERIAL, первичный ключ): Уникальный идентификатор города.
  + name (VARCHAR(100), NOT NULL): Название города, например, «Москва», «Санкт-Петербург».
* Таблица hotels:
  + id (SERIAL, первичный ключ): Уникальный идентификатор отеля.
  + name (VARCHAR(100), NOT NULL): Название отеля, например, «Гранд Отель Европа».
  + capacity (INTEGER, NOT NULL): Вместимость отеля (количество мест), например, 100 или 200.
  + standard\_room\_price (DECIMAL(10,2), NOT NULL): Цена номера категории «стандарт», например, 6500.00.
* Таблица hotel\_cities:
  + hotel\_id (INTEGER, внешний ключ): Ссылка на id из таблицы hotels.
  + city\_id (INTEGER, внешний ключ): Ссылка на id из таблицы cities.
  + Первичный ключ: (hotel\_id, city\_id), обеспечивающий уникальность пары отель-город.

Связь многие-ко-многим реализована через таблицу hotel\_cities, которая позволяет гибко связывать отели с городами. Например, отель «Москва Марриотт» может быть связан с городом «Москва» (city\_id=1). Схема базы данных представлена на Рис. 1.

Изображение выглядит как текст, снимок экрана, диаграмма, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1: Схема базы данных (таблицы cities, hotels, hotel\_cities).

Бэкенд реализован на языке Go с использованием фреймворка Gin, который обеспечивает лёгкую маршрутизацию и обработку HTTP-запросов. Основной эндпоинт /hotels принимает GET-запросы с двумя параметрами:

* field: Поле для поиска (name, city, capacity, standard\_room\_price).
* value: Значение для поиска (например, «Гранд», «Москва», «100», «6500»).

Бэкенд выполняет следующие действия:

* Проверяет, что параметр value не пустой. Если пустой, возвращается ошибка: {"error": "Search value cannot be empty"}.
* Формирует SQL-запрос в зависимости от выбранного поля:
  + Для name и city используется оператор ILIKE для поиска с игнорированием регистра и поддержкой частичного совпадения (например, «Моск» найдёт «Москва»).
  + Для capacity и standard\_room\_price используется точное сравнение (=), с проверкой, что введённое значение является числом (целым для capacity, дробным для standard\_room\_price).
* Выполняет запрос к базе данных с использованием драйвера github.com/lib/pq.
* Возвращает результаты в формате JSON, содержащем массив объектов с полями id, name, city, capacity, standard\_room\_price.

Пример ответа для запроса http://localhost:8080/hotels?field=name&value=Гранд:

[

{

"id": 1,

"name": "Гранд Отель Европа",

"city": "Санкт-Петербург",

"capacity": 200,

"standard\_room\_price": 12000.00

}

]

Бэкенд также обрабатывает ошибки, такие как некорректные числовые значения, возвращая сообщения вроде {"error": "Invalid capacity value: must be an integer"}.

Фронтенд Фронтенд разработан на React и включает компонент App.jsx, который содержит:

* Форму поиска: выпадающий список () для выбора поля (название отеля, город, вместимость, цена номера), текстовое поле () для ввода значения и кнопку «Поиск».
* Таблицу для отображения результатов с колонками: Название, Город, Вместимость, Цена номера.
* Обработку ошибок: сообщения, такие как «Введите запрос» или «Некорректное значение для вместимости», отображаются в интерфейсе.

Для отправки запросов используется библиотека Axios, которая упрощает выполнение HTTP-запросов и обработку ответов. Пример запроса:

axios.get('http://localhost:8080/hotels', { params: { field, value } })

Фронтенд обрабатывает ответы API, проверяя, является ли результат массивом с помощью Array.isArray, чтобы избежать ошибки "hotels.map is not a function". Если ответ содержит ошибку, она отображается пользователю.

Стилизация Интерфейс стилизован с использованием CSS, обеспечивая современный и удобный дизайн:

* Тёмная тема с тёмным фоном (#1e1e1e) и светлым текстом (#e6e6e6) для комфортного восприятия.
* Адаптивный дизайн с использованием flexbox и медиа-запросов, чтобы форма и таблица корректно отображались на мобильных устройствах (например, при ширине экрана менее 600px форма становится вертикальной).
* Анимации (например, @keyframes fadeIn) для плавного появления заголовка и эффектов наведения на кнопке.
* Таблица имеет чёткие границы, заголовки выделены цветом (#f2f2f2), а строки чередуются для лучшей читаемости.

Примеры использования Приложение поддерживает следующие сценарии использования:

* Поиск по названию: Пользователь выбирает «Название отеля», вводит «Гранд» и получает список отелей, содержащих «Гранд» в названии (например, «Гранд Отель Европа»).
* Поиск по городу: Ввод «Москва» возвращает отели, связанные с Москвой, такие как «Москва Марриотт» и «Динамо Москва».
* Поиск по вместимости: Ввод «100» возвращает отели с точной вместимостью 100 мест (например, «Самарская Панорама»).
* Поиск по цене: Ввод «6500» возвращает отели с ценой номера 6500.00 (например, «Самарская Панорама»).
* Обработка ошибок: Пустой ввод показывает сообщение «Введите запрос», а некорректное значение для вместимости (например, «abc») — «Некорректное значение для вместимости».

Эти сценарии обеспечивают гибкость и удобство для пользователей, ищущих отели по различным критериям.

Использованные технологии

* PostgreSQL 16: Реляционная база данных для хранения информации [1].
* Go (Gin): Язык программирования и фреймворк для создания REST API [2, 3].
* React: Библиотека JavaScript для построения интерфейса [4].
* Axios: Библиотека для HTTP-запросов [5].
* CSS: Для стилизации интерфейса с тёмной темой, адаптивностью и анимациями [6].

# Реализация проекта

Реализация включала следующие этапы:

Настройка окружения Установил PostgreSQL 16 и pgAdmin [1]. Добавил C:\Program Files\PostgreSQL\16\bin в PATH для использования psql. Установил Go, создал проект (go mod init hotelapp), добавил зависимости:

go get github.com/gin-gonic/gin

go get github.com/lib/pq

Для фронтенда установил Node.js, создал React-приложение (npx create-react-app hotel-frontend), добавил Axios и CSS:

npm install axios

Создание базы данных Создал SQL-скрипт init\_db.sql для базы hoteldb с кодировкой UTF-8:

CREATE DATABASE hoteldb;

\c hoteldb

CREATE TABLE cities (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE hotels (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

capacity INT NOT NULL,

standard\_room\_price DECIMAL(10,2) NOT NULL

);

CREATE TABLE hotel\_cities (

hotel\_id INT REFERENCES hotels(id),

city\_id INT REFERENCES cities(id),

PRIMARY KEY (hotel\_id, city\_id)

);

Добавил 15 записей в каждую таблицу (например, отели «Гранд Отель Европа», «Жемчужина Сочи»). Выполнил скрипт через pgAdmin, проверил данные запросом SELECT \* FROM hotels LIMIT 5; (Рис. 2).

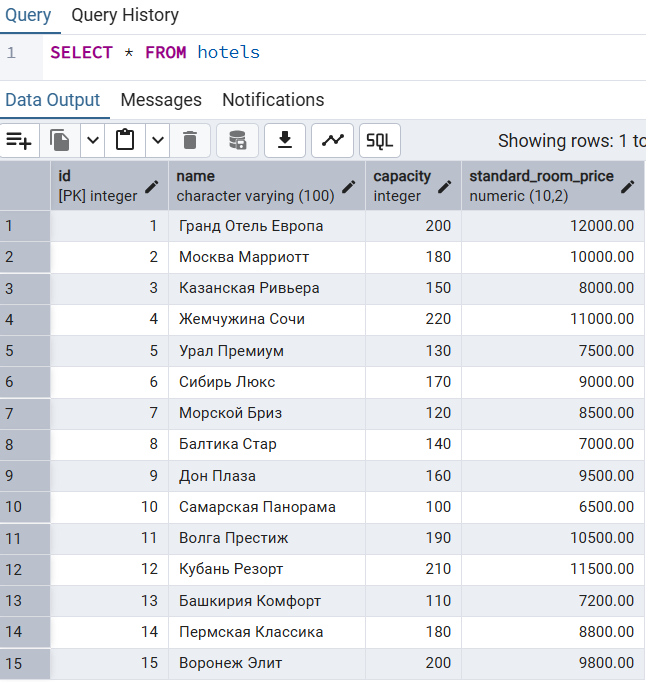


Рис. 2: Скриншот pgAdmin с таблицей hotels.

Бэкенд Создал main.go с подключением к PostgreSQL:

connStr := "user=hoteluser password=hotelpassword dbname=hoteldb sslmode=disable"

db, err := sql.Open("postgres", connStr)

Эндпоинт /hotels обрабатывает GET-запросы с параметрами field и value. Для name и city используется ILIKE, для capacity и standard\_room\_price — точное сравнение (=). Добавлена обработка ошибок:

* Пустой ввод: {"error": "Search value cannot be empty"}.
* Некорректные числа: {"error": "Invalid capacity value: must be an integer"} или {"error": "Invalid price value: must be a number"}. Решил проблему CORS, добавив middleware:

c.Writer.Header().Set("Access-Control-Allow-Origin", "\*")

Фронтенд Создал компонент App.jsx с формой поиска (select, input, кнопка) и таблицей. Использовал Axios для запросов:

axios.get('http://localhost:8080/hotels', { params: { field, value } })

Исправил ошибку "hotels.map is not a function", инициализировав hotels как пустой массив и проверяя Array.isArray:

const [hotels, setHotels] = useState([]);

setHotels(Array.isArray(data) ? data : []);

Добавил обработку ошибок для отображения сообщений, таких как «Введите запрос».

4.5. Стилизация Использовал CSS для современного дизайна:

* Интерфейс показан на Рис. 3.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3: Интерфейс приложения с формой поиска.

# Тестирование и результаты

естировал приложение на всех этапах:

* База данных: Проверил создание таблиц и данные через pgAdmin (SELECT \* FROM hotels;, Рис. 2).
* Бэкенд: Использовал curl:

curl "http://localhost:8080/hotels?field=name&value=Гранд" (Рис. 4).

Проверены все поля:

* name: «Гранд» вернул «Гранд Отель Европа».
* city: «Москва» вернул отели в Москве.
* capacity: «100» вернул отели с вместимостью 100.
* price: «6500» вернул отели с ценой 6500.00.
* Ошибки: пустой ввод и некорректные числа возвращали JSON-ошибки
* Фронтенд: Тестировал на http://localhost:3000:
  + Поиск по всем полям.
  + Обработка ошибок: пустой ввод показывал «Search value cannot be emty» (Рис. 5).
  + Адаптивность на мобильных устройствах (Рис. 6).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 4: Скриншот проверки бэкенда через curl.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 5: Сообщение об ошибке «Search value cannot be emty».

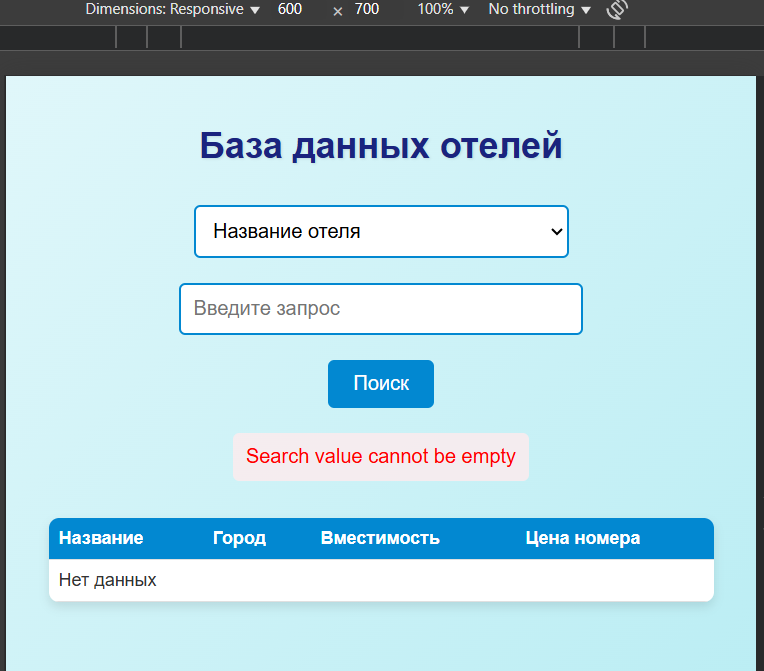


Рис. 6: Интерфейс на мобильном устройстве (DevTools, 600px).

# **Заключение**

Я успешно разработал веб-приложение «База данных отеля», соответствующее требованиям задания. Приложение включает PostgreSQL-базу данных, REST API на Go (Gin) и React-интерфейс с CSS. Основные достижения:

* Создание базы данных с 15 записями в таблицах cities, hotels, hotel\_cities.
* Реализация API с точным поиском по capacity и price.
* Фронтенд с обработкой ошибок и адаптивным дизайном.
* Решение проблем: CORS, "hotels.map is not a function", кодировка UTF-8.

Приобретённые навыки:

* Работа с PostgreSQL: создание таблиц, SQL-запросы.
* Разработка API на Go: обработка запросов, JSON, логирование.
* Создание интерфейсов на React: управление состоянием, Axios, CSS.
* Отладка: curl, DevTools, логи.

В будущем можно добавить:

* Пагинацию для больших данных.
* Фильтрацию по нескольким полям.
* Анимацию загрузки запросов.

# Список литературы

1. Официальная документация PostgreSQL. — URL: https://www.postgresql.org/docs/17/ [1]

2. Документация Go. — URL: https://go.dev/doc/ [2]

3. Документация Gin. — URL: https://gingonic.com/docs/ [3]

4. Документация React. — URL: https://reactjs.org/docs/ [4]

5. Документация Axios. — URL: https://axioshttp.com/docs/ [5]

6. MDN Web Docs по CSS (Flexbox, анимации). — URL: https://developer.mozilla.org/ [6]

7. Статья о настройке CORS. — URL: https://developer.mozilla.org/enUS/docs/Web/HTTP/CORS [7]