**СОДЕРЖАНИЕ**

Введение…………………………………………………………………….6

1.Описание предметной области……………………………………. …....7

1.1 Назначение программного продукта…………………………………..7

1.2. Задачи программного продукта ..……………………………………...7

1.3. Источники входной информации……………………………………...8

1.4. Выходная информация………………………………………………...8

2. Описание маппинга…………………………………………………….10

2.1 Сущности и атрибуты…………………………………………………10

2.2 Таблицы базы данных…………………………………………………11

2.3 Связи между сущностями …………………………………………….12

3. Описание точек доступа………………………………………………..15

4. Разработка REST-сервиса……………………………………………...19

4.1 Структура проекта…………………………………………………….19

4.2 Запуск проекта…………………………………………………………20

* 1. Модель данных………………………………………………………...20
  2. Маршрутизация и контроллеры………………………………………22

Заключение………………………………………………………………...24

Список использованных источников…..………………………………...25

**ВВЕДЕНИЕ**

В наше время информационные технологии проникают во все сферы бизнеса, оказывая значительное влияние на их функционирование. Сфера туризма не исключение, и в ней также становится очевидной необходимость в использовании современных технологических решений для оптимизации бизнес-процессов и улучшения обслуживания клиентов. В этой связи разработка REST-приложения для турагентства, базирующегося на Node.js и использующего PostgreSQL, представляет собой актуальную и важную задачу.

Цель данного курсового проекта заключается в создании программного продукта, который поможет турагентству эффективно управлять своими услугами и обеспечить высокий уровень обслуживания для своих клиентов. Применение технологии REST обеспечит гибкость и масштабируемость приложения, что позволит быстро адаптироваться к изменяющимся потребностям рынка и объему запросов. Выбор Node.js в качестве основы гарантирует высокую производительность и возможность асинхронного программирования, что важно для быстроты реакции на запросы клиентов и обеспечения плавной работы системы. Интеграция с PostgreSQL обеспечит надежное хранение данных о клиентах, туристических пакетах, заказах и других аспектах работы турагентства, что позволит не только эффективно управлять информацией, но и анализировать её для принятия стратегических решений.

**1. Описание предметной области**

1.1 Назначение программного продукта

Конечная цель данного программного продукта заключается в создании удобной и эффективной системы управления туристическими пакетами и заказами для турагентства. Этот продукт будет служить основным инструментом для турагентов в процессе создания, редактирования и удаления туристических пакетов, а также управления заказами от клиентов.

1.2 Задачи программного продукта

Основные задачи и функциональные возможности программного продукта включают:

* Создание новых туристических пакетов с указанием всех необходимых параметров, таких как место назначения, даты путешествия, длительность и стоимость.
* Редактирование существующих туристических пакетов для внесения изменений в условия предложения в соответствии с требованиями клиентов или изменениями на рынке туризма.
* Удаление устаревших или непопулярных туристических пакетов для оптимизации ассортимента и повышения конкурентоспособности турагентства.
* Просмотр и управление списком доступных туристических пакетов.

Таким образом, данный программный продукт направлен на обеспечение удобного и эффективного способа создания и управления туристическими пакетами, что позволит турагентству эффективно удовлетворять потребности своих клиентов и успешно конкурировать на рынке туризма.

1.3. Источники входной информации

Для разработки информационной системы турагентства были использованы следующие источники входной информации:

* Справочная литература: В процессе подготовки курсовой работы были изучены учебники и учебные пособия по управлению турагентства, а также литература по технологиям разработки программного обеспечения и базам данных. Эти источники позволили получить представление о требованиях к функциональности и управлению турагентством.
* Анализ конкурентов и существующих решений. Был проведен анализ конкурентов и изучены существующие информационные системы. Это позволило выявить особенности и лучшие практики в турагентствах.

Эти источники информации стали основой для определения функциональных требований к разрабатываемой информационной системе и обеспечили соответствие сервиса потребностям пользователей и бизнес-целям проекта.

1.4. Выходная информация

Информационная система турагентства предоставляет широкий спектр выходной информации, необходимой для эффективного управления и принятия стратегических решений.

* Информация о заказах и туристических пакетах: детали заказов, расписание и состав заказов клиентов, включая выбранные туристические пакеты, даты поездки, количество путешественников и дополнительные услуги, полная информация о туристических пакетах: описание маршрутов, включенные услуги, стоимость и доступность.
* Управление турагентством: управление туристическими пакетами, возможность создания, редактирования и удаления туристических предложений, включая изменение параметров и цен.
* Управление клиентской базой: информация о клиентах, их предпочтениях, истории заказов и контактных данных для оказания персонализированных услуг.

Эта выходная информация доступна администрации турагентства, менеджерам и сотрудникам, что позволяет им принимать обоснованные решения по оптимизации работы агентства, улучшению обслуживания клиентов и адаптации к изменяющимся потребностям рынка туризма. Кроме того, часть информации может быть предоставлена клиентам через специальные приложения или веб-интерфейсы для повышения уровня сервиса и удобства путешественников.

**2. Описание маппинга**

Приложение для турагентства разрабатывается с использованием технологии Node.js для серверной части и PostgreSQL в качестве системы управления базами данных. Целью приложения является автоматизация управления бронированиями отелей, экскурсиями и клиентской базой, обеспечивая эффективное функционирование турагентства и повышая удовлетворенность клиентов.

2.1 Сущности и атрибуты

При проектировании базы данных для информационной системы турагентства необходимо определить основные сущности и их атрибуты, которые будут храниться в базе данных. Ниже приведено описание каждой сущности и их атрибутов:

* Клиенты:

customer\_id (тип: SERIAL, ограничение: PRIMARY KEY) - уникальный идентификатор клиента.

first\_name (тип: VARCHAR(50), ограничение: NOT NULL) - имя клиента.

last\_name (тип: VARCHAR(50), ограничение: NOT NULL) - фамилия клиента.

contact\_info (тип: TEXT) - контактная информация клиента.

* Бронирования:

booking\_id (тип: SERIAL, ограничение: PRIMARY KEY) - уникальный идентификатор бронирования.

customer\_id (тип: INT, ограничение: FOREIGN KEY REFERENCES Customers(customer\_id)) - идентификатор клиента, совершившего бронирование.

booking\_date (тип: DATE, ограничение: NOT NULL) - дата бронирования.

check\_in\_date (тип: DATE, ограничение: NOT NULL) - дата заезда.

check\_out\_date (тип: DATE, ограничение: NOT NULL) - дата выезда.

adults (тип: INT, ограничение: NOT NULL) - количество взрослых.

children (тип: INT, ограничение: NOT NULL) - количество детей.

* Отели:

hotel\_id (тип: SERIAL, ограничение: PRIMARY KEY) - уникальный идентификатор отеля.

hotel\_name (тип: VARCHAR(100), ограничение: NOT NULL) - название отеля.

address (тип: TEXT) - адрес отеля.

description (тип: TEXT) - описание отеля.

star\_rating (тип: INT) - рейтинг отеля.

amenities (тип: TEXT[]) - перечень удобств отеля.

* Экскурсии:

excursion\_id (тип: SERIAL, ограничение: PRIMARY KEY) - уникальный идентификатор экскурсии.

excursion\_name (тип: VARCHAR(100), ограничение: NOT NULL) - название экскурсии.

location (тип: TEXT) - место проведения экскурсии.

start\_datetime (тип: TIMESTAMP) - дата и время начала экскурсии.

duration (тип: INTERVAL) - продолжительность экскурсии.

price (тип: NUMERIC(10, 2)) - стоимость экскурсии.

max\_participants (тип: INT) - максимальное количество участников экскурсии.

* Содержимое бронирования:

booking\_id (тип: INT, ограничение: FOREIGN KEY REFERENCES Bookings(booking\_id)) - идентификатор бронирования.

hotel\_id (тип: INT, ограничение: FOREIGN KEY REFERENCES Hotels(hotel\_id)) - идентификатор отеля.

excursion\_id (тип: INT, ограничение: FOREIGN KEY REFERENCES Excursions(excursion\_id)) - идентификатор экскурсии.

2.2 Таблицы базы данных

При разработке информационной системы турагентства были созданы следующие таблицы в базе данных для хранения данных о клиентах, бронированиях, отелях, экскурсиях и содержимом бронирований:

Таблица "Клиенты":

CREATE TABLE Customers ( customer\_id SERIAL PRIMARY KEY, first\_name VARCHAR(50) NOT NULL, last\_name VARCHAR(50) NOT NULL, contact\_info TEXT );

Таблица "Бронирования":

CREATE TABLE Bookings ( booking\_id SERIAL PRIMARY KEY, customer\_id INT REFERENCES Customers(customer\_id), booking\_date DATE NOT NULL, check\_in\_date DATE NOT NULL, check\_out\_date DATE NOT NULL, adults INT NOT NULL, children INT NOT NULL );

Таблица "Отели":

TABLE Hotels ( hotel\_id SERIAL PRIMARY KEY, hotel\_name VARCHAR(100) NOT NULL, address TEXT, description TEXT, star\_rating INT, amenities TEXT[] );

Таблица "Экскурсии":

CREATE TABLE Excursions ( excursion\_id SERIAL PRIMARY KEY, excursion\_name VARCHAR(100) NOT NULL, location TEXT, start\_datetime TIMESTAMP, duration INTERVAL, price NUMERIC(10, 2), max\_participants INT );

Таблица "Содержимое бронирования":

CREATE TABLE Booking\_Items ( booking\_id INT REFERENCES Bookings(booking\_id), hotel\_id INT REFERENCES Hotels(hotel\_id), excursion\_id INT REFERENCES Excursions(excursion\_id), PRIMARY KEY (booking\_id, hotel\_id, excursion\_id) );

2.3 Связи между сущностями

При проектировании базы данных для информационной системы турагентства необходимо определить связи между различными сущностями, чтобы эффективно организовать хранение и обработку данных. В данном разделе описываются связи между основными сущностями: клиентами, бронированиями, отелями, экскурсиями и содержимым бронирований.

Связь "Один ко многим" между таблицей "Клиенты" и "Бронирования": Поле customer\_id в таблице "Бронирования" является внешним ключом, ссылается на поле customer\_id в таблице "Клиенты". Эта связь позволяет одному клиенту совершать несколько бронирований.

Связь "Многие ко многим" между таблицей "Бронирования" и "Содержимое бронирования": Поля booking\_id, hotel\_id и excursion\_id в таблице "Содержимое бронирования" являются внешними ключами, ссылается на соответствующие поля в таблицах "Бронирования", "Отели" и "Экскурсии" соответственно. Эта связь позволяет одному бронированию включать несколько отелей и/или экскурсий, а также один отель или экскурсию могут быть включены в несколько бронирований.

Эти связи определяют взаимосвязь между различными сущностями в базе данных и обеспечивают целостность данных при их хранении и обработке. Каждая связь имеет свою роль в организации данных и функционировании информационной системы турагентства.

2.4 Описание моделей Sequelize

В ходе разработки программного средства для сервиса турагентства выявилась проблема сопоставления полей данных между классами, отвечающими за различные функциональные задачи. Для решения этой проблемы была использована библиотека Sequelize в Node.js.

Sequelize предоставляет удобные инструменты для работы с базой данных PostgreSQL и автоматический маппинг объектов JavaScript на таблицы базы данных.

Рассмотрим одну из моделей:

module.exports = (sequelize, DataTypes) => {

const Bookings = sequelize.define('Bookings', {

booking\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

customer\_id: {

type: DataTypes.INTEGER,

allowNull: false,

},

booking\_date: {

type: DataTypes.DATE,

allowNull: false,

},

check\_in\_date: {

type: DataTypes.DATE,

allowNull: false,

},

check\_out\_date: {

type: DataTypes.DATE,

allowNull: false,

},

adults: {

type: DataTypes.INTEGER,

allowNull: false,

},

children: {

type: DataTypes.INTEGER,

allowNull: false,

},

}, {

tableName: 'bookings',

timestamps: false,

});

Bookings.associate = (models) => {

Bookings.belongsTo(models.Customers, { foreignKey: 'customer\_id' });

Bookings.hasMany(models.Booking\_Items, { foreignKey: 'booking\_id' });

};

return Bookings;

};

Модель "Booking" представляет собой сущность бронирования номеров отелей. Ниже представлены описание полей модели:

booking\_id: Целое число, является первичным ключом, автоматически увеличивается.

customer\_id: Целое число, не может быть NULL, внешний ключ, связывающий запись бронирования с конкретным клиентом.

booking\_date: Дата, не может быть NULL, дата, когда было сделано бронирование.

check\_in\_date: Дата, не может быть NULL, дата заселения.

check\_out\_date: Дата, не может быть NULL, дата выселения.

adults: Целое число, не может быть NULL, количество взрослых в бронировании.

children: Целое число, не может быть NULL, количество детей в бронировании.

С помощью этой модели мы можем генерировать новые записи в базе данных, изменять, удалять, а также выводить выборки из бд.

**3. Описание точек доступа**

В данном разделе описаны точки доступа (endpoints) для взаимодействия с информационной системой турагентства. Каждая точка доступа имеет уникальное название, метод запроса (GET, POST, PUT, DELETE), URL-адрес и, при необходимости, параметры запроса и тело запроса.

1. Создание клиента

Метод: POST

URL: localhost:4444/customers

Тело запроса: JSON с полями first\_name, last\_name и contact\_info

1. Получение всех клиентов

Метод: GET

URL: localhost:4444/customers

1. Получение клиента по идентификатору

Метод: GET

URL: localhost:4444/customers-id

Параметры запроса: id (идентификатор клиента)

1. Изменение информации о клиенте

Метод: PUT

URL: localhost:4444/customers

Параметры запроса: id (идентификатор клиента)

Тело запроса: JSON с полями first\_name, last\_name и contact\_info

1. Удаление клиента

Метод: DELETE

URL: localhost:4444/customers

Параметры запроса: id (идентификатор клиента)

1. Создание бронирования

Метод: POST

URL: localhost:4444/booking

Тело запроса: JSON с полями customer\_id, booking\_date, check\_in\_date, check\_out\_date, adults, children

1. Просмотр бронирований

Метод: GET

URL: localhost:4444/booking

1. Просмотр бронирования по идентификатору

Метод: GET

URL: localhost:4444/booking-id

Параметры запроса: id (идентификатор бронирования)

1. Изменение бронирования

Метод: PUT

URL: localhost:4444/booking

Параметры запроса: id (идентификатор бронирования)

Тело запроса: JSON с полями customer\_id, booking\_date, check\_in\_date, check\_out\_date, adults, children

1. Удаление бронирования

Метод: DELETE

URL: localhost:4444/booking

Параметры запроса: id (идентификатор бронирования)

1. Создание экскурсии

Метод: POST

URL: localhost:4444/excursions

Тело запроса: JSON с полями excursion\_name, location, start\_datetime, duration, price, max\_participants

1. Просмотр всех экскурсий

Метод: GET

URL: localhost:4444/excursions

1. Просмотр экскурсии по идентификатору

Метод: GET

URL: localhost:4444/excursions-id

Параметры запроса: id (идентификатор экскурсии)

1. Обновление экскурсии

Метод: PUT

URL: localhost:4444/excursions

Параметры запроса: id (идентификатор экскурсии)

Тело запроса: JSON с полями excursion\_name, location, start\_datetime, duration, price, max\_participants

1. Удаление экскурсии

Метод: DELETE

URL: localhost:4444/excursions

Параметры запроса: id (идентификатор экскурсии)

1. Добавление отеля

Метод: POST

URL: localhost:4444/hotels

Тело запроса: JSON с полями hotel\_name, address, description, star\_rating, amenities

1. Просмотр всех отелей

Метод: GET

URL: localhost:4444/hotels

1. Просмотр отеля по идентификатору

Метод: GET

URL: localhost:4444/hotels-id

Параметры запроса: id (идентификатор отеля)

1. Обновление информации об отеле

Метод: PUT

URL: localhost:4444/hotels

1. Параметры запроса: id (идентификатор отеля)

Тело запроса: JSON с полями hotel\_name, address, description, star\_rating, amenities

1. Удаление отеля

Метод: DELETE

URL: localhost:4444/hotels

Параметры запроса: id (идентификатор отеля)

1. Добавление экскурсии и отеля в заказ

Метод: POST

URL: localhost:4444/booking-item

Тело запроса: JSON с полями booking\_id, hotel\_id, excursion\_id

1. Получение экскурсии и отеля по идентификатору бронирования

Метод: GET

URL: localhost:4444/booking-item

Параметры запроса: booking\_id (идентификатор бронирования)

1. Удаление связи

Метод: DELETE

URL: localhost:4444/booking-item

Параметры запроса: bookingId, hotelId, excursionId (идентификаторы связи)

1. Получение всей информации о бронировании

Метод: GET

URL: localhost:4444/booking-details

Параметры запроса: bookingId (идентификатор бронирования)

1. Количество заказов в месяц

Метод: GET

URL: localhost:4444/count-bookings-by-month

Параметры запроса: year, month (год и месяц)

**4. Разработка REST-сервиса**

4.1 Структура проекта

Проект состоит из нескольких основных компонентов, организованных в соответствии с модульной архитектурой:

* Конфигурация (config/):

Директория содержит файлы конфигурации проекта, такие как настройки порта, базы данных и маршрутов API.

Файлы конфигурации структурированы для удобства управления параметрами приложения в различных окружениях (development, production).

* Сервисы (services/):

Директория содержит сервисы, отвечающие за взаимодействие с внешними ресурсами, такими как база данных.

Каждый сервис организован в отдельный модуль для обеспечения четкой структуры и разделения ответственности.

* Контроллеры (api/controllers/):

Директория содержит контроллеры, обрабатывающие HTTP-запросы и возвращающие соответствующие HTTP-ответы.

Контроллеры используются для взаимодействия с бизнес-логикой и обработки данных, полученных от клиентов.

* Маршрутизация (api/routes/):

Директория содержит файлы маршрутов, определяющие маршруты API и их соответствующие обработчики контроллеров.

Каждый маршрут определяет метод запроса, путь и соответствующий контроллер для обработки запросов.

* Статические ресурсы (public/):

Директория содержит статические ресурсы, такие как изображения, стили CSS, клиентский JavaScript.

* Файлы приложения (app.js, server.js):

Файл app.js является точкой входа приложения, где создается экземпляр Express и настраиваются его middleware.

Файл server.js запускает HTTP-сервер и привязывает Express-приложение к определенному порту.

Также, в проекте используются сторонние библиотеки для облегчения разработки и расширения функциональности, такие как body-parser, helmet, cors.

4.2 Запуск проекта

Запуск проекта обычно осуществляется через команду запуска, указанную в скриптах файла package.json. В проекте скрипт для запуска называется start.

Чтобы запустить проект, вам нужно перейти в корневую директорию проекта в командной строке и выполнить следующую команду: npm start

Это запустит скрипт start, который в свою очередь запустит файл api.js, который является точкой входа в приложение Express. После этого приложение будет доступно по указанному порту.

После успешного запуска проекта вы увидите сообщение в консоли, которое указывает на то, что ваш сервер Express запущен и слушает указанный порт.

Если в процессе запуска возникают ошибки, убедитесь, что все зависимости установлены правильно с помощью команды npm install, и проверьте конфигурацию вашего проекта и код в файлах api.js и package.json.

* 1. Модель данных

Модель данных — это абстрактное представление о данных в информационной системе, которое определяет их структуру, типы, связи и правила хранения. Она служит основой для организации данных в базе данных и обеспечивает их удобное и эффективное использование.

Модель данных описывает, каким образом данные будут представлены и организованы в базе данных, что позволяет разработчикам и администраторам системы лучше понимать структуру данных и обеспечивать их корректное использование.

Рассмотрим следующую модель данных для сущности "Покупатели":

module.exports = (sequelize, DataTypes) => {

const Customers = sequelize.define('Customers', {

customer\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

first\_name: {

type: DataTypes.STRING,

allowNull: false,

},

last\_name: {

type: DataTypes.STRING,

allowNull: false,

},

contact\_info: {

type: DataTypes.TEXT,

allowNull: true,

},

}, {

tableName: 'customers',

timestamps: false,

});

Customers.associate = (models) => {

Customers.hasMany(models.Bookings, { foreignKey: 'customer\_id' });

};

return Customers;

};

Описание модели:

* customer\_id: Тип: INTEGER Определение: Уникальный идентификатор клиента Свойства: Первичный ключ (primaryKey), автоинкремент (autoIncrement)
* first\_name: Тип: STRING, Определение: Имя клиента Свойства: Обязательное поле (allowNull: false)
* last\_name: Тип: STRING, Определение: Фамилия клиента Свойства: Обязательное поле (allowNull: false)
* contact\_info: Тип: TEXT Определение: Контактная информация клиента Свойства: Поле может быть пустым (allowNull: true)

Дополнительные свойства модели:

* tableName: Название таблицы в базе данных, в которой хранятся данные о клиентах.
* timestamps: Указывает, нужно ли добавлять поля createdAt и updatedAt в записи о клиентах. Значение false означает, что эти поля не будут созданы.

Ассоциация с другими моделями:

Модель Customers ассоциирована с моделью Bookings по отношению "один ко многим", где один клиент может иметь несколько заказов. Это отображено в методе Customers.associate, где указано, что у клиента может быть много заказов, связь осуществляется через внешний ключ customers\_id в таблице Bookings.

Эта модель данных используется для хранения информации о клиентах кофейни в базе данных. Каждая запись о клиенте содержит его уникальный идентификатор, имя и контактную информацию. Также эта модель связана с моделью заказов для отслеживания заказов, сделанных каждым клиентом.

4.4 Маршрутизация и контроллеры

Рассмотрим метод createBooking из контроллера BookingsController, который отвечает за создание тура. Рассмотрим этот метод от получения запроса до отправления ответа:

async createBooking(req, res) {

try {

const {

customer\_id, booking\_date, check\_in\_date, check\_out\_date, adults, children,

} = req.body;

const newBooking = await Booking.create({

customer\_id, booking\_date, check\_in\_date, check\_out\_date, adults, children,

});

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

} catch (error) {

console.error('Ошибка при создании бронирования:', error);

res.status(500).json({ error: 'Ошибка при создании бронирования' });

}

}

Этот метод выполняет следующие действия:

* Получение запроса. Метод createBooking ожидает получить HTTP-запрос от клиента с данными о бронировании в теле запроса.
* Обработка запроса. В блоке try метод извлекает данные о бронировании из тела запроса и использует модель Booking для создания новой записи бронирования в базе данных.
* Отправка ответа. Если бронирование успешно создано, метод отправляет ответ обратно клиенту с HTTP-статусом 201 (Created) и JSON-представлением созданного бронирования.
* Обработка ошибок. Если при создании бронирования возникает ошибка, контроллер перехватывает исключение в блоке catch. Ошибка обрабатывается путем логирования сообщения об ошибке и отправки соответствующего HTTP-статуса и сообщения об ошибке в ответе.

**ЗАКЛЮЧЕНИЕ**

Разработка приложения для турагентства с использованием современных технологий является значимым шагом в сфере туризма и гостеприимства. Турагентства играют ключевую роль в организации путешествий, обеспечивая клиентам доступ к разнообразным туристическим услугам и помогая им в выборе оптимальных вариантов отдыха.

Применение технологий Node.js, Express и PostgreSQL в разработке программного продукта для управления турагентством обеспечивает эффективное управление всеми аспектами бизнеса. Node.js обеспечивает высокую производительность и гибкость приложения, позволяя создать мощное и масштабируемое веб-приложение. Фреймворк Express предоставляет удобные средства для разработки веб-приложений, а PostgreSQL, как надежная реляционная база данных, обеспечивает сохранность и целостность данных.

В результате выполнения данной работы было создано полнофункциональное приложение для управления турагентством, которое позволяет эффективно управлять туристическими предложениями, бронированием, клиентами и аналитикой. Этот проект предоставляет практический опыт работы с современными технологиями и позволяет создать инновационное веб-приложение для управления туристическим бизнесом.

В заключении, разработка приложения для турагентства с применением передовых технологий играет важную роль в современной индустрии туризма, обеспечивая туристическим компаниям инструменты для успешного развития и удовлетворения потребностей клиентов в организации путешествий.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация Node.js: <https://nodejs.org/en/docs/>
2. Документация Express.js: <https://expressjs.com/>
3. Документация Sequelize): <https://sequelize.org/>
4. Документация PostgreSQL: <https://www.postgresql.org/docs/>
5. "PostgreSQL: Введение и руководство по администрированию" (Simon Riggs, Hannu Krosing)