

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Индивидуальное домашнее задание

по БАЗАМ ДАННЫХ

Разработка прототипа прикладного приложения для БД

Студент

Глубоков Г.В.

Группа АИ-20

Руководитель

Доцент

Алексеев В.А.

Липецк 2022 г.

Цель работы

Получение первичных навыков разработки прикладных приложений для БД, освоение фреймворков для работы с БД.

Задание кафедры

Реализовать прикладное приложение, обеспечивающее просмотр и редактирование содержимого спроектированной в ходе лабораторного практикума БД, выполнение и просмотр результатов запросов, вызов хранимых процедур.

Характеристика предметной области

1. Описание предметной области

Туристическое агентство «SeaBreeze» предоставляет возможность клиентам просматривать/покупать туры в разные страны мира.

Клиент приходит в офис компании либо посещает сайт агентства в Интернете. При необходимости сотрудник агентства проконсультирует клиента по вопросам турпродукта. При покупке/оформления тура заключается договор.

Назначение АИС:

Упрощение получения информации, покупки путёвок и ускорение работы.

Решаемые задачи:

- предоставление информации о доступных турах.
- продажа путёвок.
- улучшение работы внутри агентства.

2. Основные бизнес-процессы

1. Продажа турпродукта клиентам:

- а) Менеджер продаёт посетителям путёвки и каталога.

2. Формирование расписания туров.

- а) Администратор из доступных туров формирует расписание
- б) Администратор указывает цену тура, его описание, дату старта и окончания (администратор)

3. Добавление и удаление туров

- а) Администратор добавляет или удаляет туры из общего списка туров.

3. Основные категории пользователей разрабатываемой АИС:

1. Клиент

Клиент должен иметь возможность просматривать/покупать турпродукт.

2. Менеджер

Менеджер должен иметь возможность предоставлять услуги продажи заказа клиенту

3. Главный менеджер

Главный менеджер должен иметь возможность добавлять и/или удалять туры, устанавливать их цену и дату, просматривать список купленных заказов.

4. Основные виды информации:

Тур (id тура, название, дата старта, дата окончания, цена тура, описание тура)

Отель (id отеля, название отеля, класс отеля)

Город (код города, название города)

Страна (код страны, название страны)

Заказ (id заказа, дата оформления заказа)

Клиент (id клиента, ФИО клиента, логин, пароль, пол клиента, номер паспорта клиента, номер телефона клиента)

Менеджер (id менеджера, ФИО менеджера, номер телефона менеджера, пол менеджера, должность)

Транспорт (id транспорта, вид транспорта)

5. Формируемые отчеты:

1) Количество проданных путёвок за месяц.

ФИО клиента	id заказа	ФИО менеджера	Дата продажи
Иванов Иван Иванович	34	Николаева Марина Анатольевна	03.05.2022
Куликов Алексей Дмитриевич	27	Нефёдов Сергей Николаевич	12.06.2022
Итого:			2

2) Самые популярные туры.

Название тура	количество продаж	лучший менеджер
«Мадрид»	206	Николаева Марина Анатольевна
«Рим»	180	Нефёдов Сергей Николаевич

3) Постоянные клиенты

id клиента	количество купленных туров	Общая стоимость
357	11	470000
13	13	360000

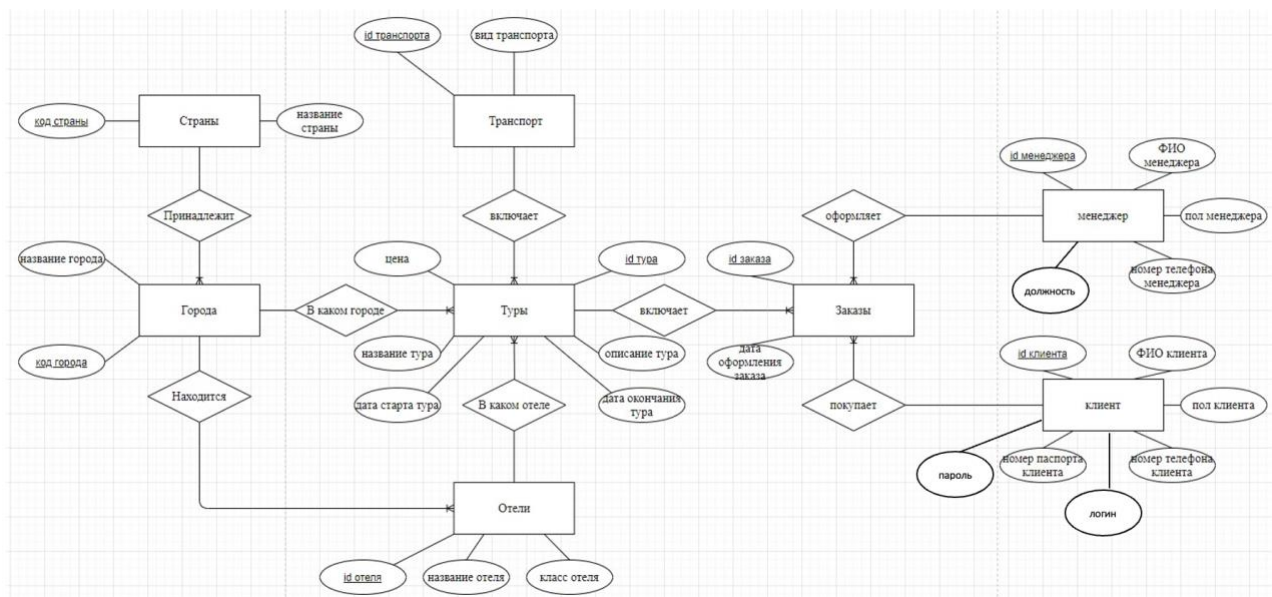


Рисунок 1 – Общая концептуальная модель.

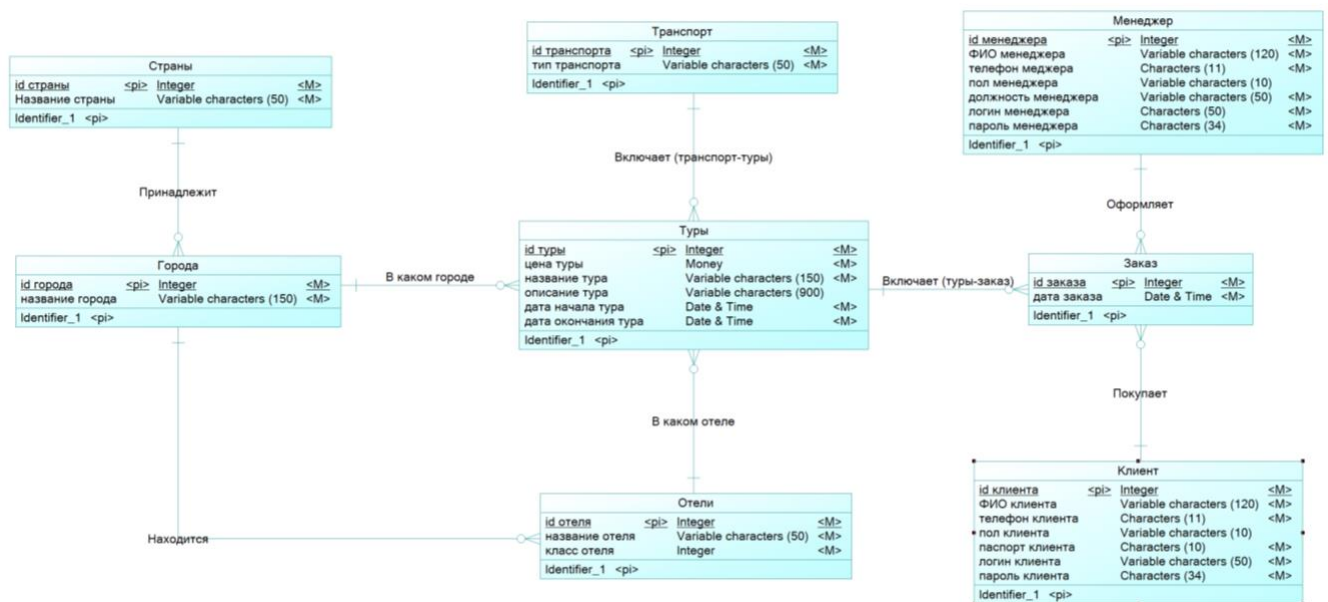


Рисунок 2 – Физическая модель.

Разработка прикладного приложения

Платформа разработки

Приложение разработано в JetBrains WebStorm на языке программирования Node.js, Express, React, HTML, CSS.

ОС: MacOS Monterey.

При разработке БД использовались: Sybase PowerDesigner 16.6, PostgreSQL, pgAdmin 4 v6

Связь приложения с БД

Server

app.js

```
require('dotenv').config()
const express = require('express')
const bodyParser = require('body-parser')
const ride = require('./Controllers/RideController.js')
const user = require('./Controllers/UserController.js')
const entity = require('./Controllers/EntityController')
```

```
const app = express()
const port = process.env.PORT
```

```
const cors = require('cors');
app.use(cors());
app.use(bodyParser.json())
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
)
new ride.RideController(app);
new user.UserController(app);
new entity.EntityController(app)
app.listen(port, () => {
  console.log(`App running on port ${port}.`)
})
```

BaseModel.js

```
const Pool = require('pg').Pool
```

```
class BaseModel {
```

```
  connection;
  table;
```

```
  constructor() {
```



```

        this.connection = new Pool({
            user: 'german',
            host: 'localhost',
            database: 'tour',
            password: "",
            port: 5432,
        })
    }

}

module.exports = {
    BaseModel
}

```

BaseController.js

```

class BaseController {

    routes = [];
    app;

    constructor(app) {
        this.app = app
    }

    attachRoutes() {
        this.routes.forEach((route) => {
            this.app[route.type](route.path, route.handler.bind(this))
            console.log(` ${route.type.toUpperCase()}: ${route.path}`)
        })
    }
}

```

```

module.exports = {
    BaseController
}

```

EntityController.js

```

const base = require('./BaseController')
const model = require('./models/Entity')

class EntityController extends base.BaseController {

```

```
model;
routes = [

  {
    path: '/collections',
    type: 'post',
    handler: this.getCollections,
  },
  {
    path: '/tour',
    type: 'post',
    handler: this.getTour,
  },
  {
    path: '/addRide',
    type: 'post',
    handler: this.addRide,
  },
  {
    path: '/giveUnfRide',
    type: 'post',
    handler: this.takeUnfRide,
  },
  {
    path: '/takeRide',
    type: 'post',
    handler: this.takeRide,
  },
  {
    path: '/sucRide',
    type: 'post',
    handler: this.sucRide,
  },
  {
    path: '/addTour',
    type: 'post',
    handler: this.addTour,
  },
  {
    path: '/delTour',
    type: 'post',
  },
]
```

```

        handler: this.delTour,
    },
    {
        path: '/reportType1',
        type: 'post',
        handler: this.reportType1,
    },
    {
        path: '/reportType2',
        type: 'post',
        handler: this.reportType2,
    },
    {
        path: '/reportType3',
        type: 'post',
        handler: this.reportType3,
    },
];

constructor(app) {
    super(app);
    this.model = new model.Entity()
    this.attachRoutes()
}

async reportType1(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.reportType1()
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async reportType2(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.reportType2()
        response
            .status(200)

```

```

        .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async reportType3(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.reportType3()
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async delTour(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.delTour(request.body)
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async getCollections(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.getCollections()
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async sucRide(request, response) {

```

```

    try {
      console.log(request.body);
      let res = await this.model.sucRide(request.body)
      response
        .status(200)
        .json(res)
    } catch (e) {
      response.status(404).json(e);
    }
  }

  async getTour(request, response) {
    try {
      console.log(request.body);
      let res = await this.model.getTour(request.body)
      response
        .status(200)
        .json(res)
    } catch (e) {
      response.status(404).json(e);
    }
  }

  async addTour(request, response) {
    try {
      let res = await this.model.addTour(request.body)
      response
        .status(200)
        .json(res)
    } catch (e) {
      response.status(404).json(e);
    }
  }

  async takeUnfRide(request, response) {
    try {
      console.log(request.body);
      let res = await this.model.takeUnfRide(request.body)
      response
        .status(200)
        .json(res)
    } catch (e) {

```

```

        response.status(404).json(e);
    }
}

async takeRide(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.takeRide(request.body)
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}

async addRide(request, response) {
    try {
        console.log(request.body);
        let res = await this.model.addRide(request.body)
        response
            .status(200)
            .json(res)
    } catch (e) {
        response.status(404).json(e);
    }
}
}

module.exports = {
    EntityController
}

```

RideController.js

```

const base = require('../BaseController')
const model = require('../models/Ride')

class RideController extends base.BaseController {

    model;
    routes = [
        {
            path: '/ride',

```

```

        type: 'get',
        handler: this.getAll
    },
    {
        path: '/room/:id',
        type: 'get',
        handler: this.getOne
    },
    {
        path: '/room',
        type: 'post',
        handler: this.create
    },
];

constructor(app) {
    super(app);
    this.model = new model.Ride()
    this.attachRoutes()
}

async getAll(request, response) {
    response
        .status(200)
        .json(await this.model.getAll())
}

async getOne(request, response) {
    const result = await this.model.getById(request.params.id)
    if (result) {
        response
            .status(200)
            .json(result)
    } else {
        response
            .status(404)
            .json({
                error: 'Not found'
            })
    }
}
}

```

```

async create(request, response) {
  const result = await this.model.create(request.body)
  if (result.rowCount) {
    response
      .status(201)
      .json({
        status: "Success"
      })
  } else {
    response
      .status(400)
      .json({
        error: "Bad request"
      })
  }
}
}

```

```

module.exports = {
  RideController
}

```

UserController.js

```

const base = require('../BaseController')
const model = require('../models/User')

```

```

class UserController extends base.BaseController {

```

```

  model;
  routes = [

    {
      path: '/auth',
      type: 'post',
      handler: this.checkUser,
    },
    {
      path: '/registration',
      type: 'post',
      handler: this.createUser,
    },

```



```

    {
      path: '/client',
      type: 'post',
      handler: this.getUser,
    },
    {
      path: '/timeRegClient',
      type: 'post',
      handler: this.TRU,
    },
    {
      path: '/addManager',
      type: 'post',
      handler: this.addManager,
    },
  ],

  constructor(app) {
    super(app);
    this.model = new model.User()
    this.attachRoutes()
  }

  async addManager(request, response) {
    try {
      console.log(request.body);
      let res=await this.model.addManager(request.body)
      response
        .status(200)
        .json(res)
    }
    catch (e){
      response.status(404).json(e);
    }
  }

  async checkUser(request, response) {
    try {
      console.log(request.body);
      let hash=this.model.hash(request.body.password)
      let res=await this.model.checkUser(request.body.login, hash)

```

```

        response
            .status(200)
            .json(res)
    }
    catch (e){
        response.status(404).json(e);
    }
}

async createUser(request, response) {
    try {
        console.log(request.body);
        let hash=this.model.hash(request.body.password)
        let res=await this.model.createUser(request.body, hash)
        response
            .status(200)
            .json(res)
    }
    catch (e){
        response.status(404).json(e);
    }
}

async TRU(request, response) {
    try {
        console.log(request.body);
        let res=await this.model.TRU(request.body)
        response
            .status(200)
            .json(res)
    }
    catch (e){
        response.status(404).json(e);
    }
}

async getUser(request, response) {
    try {
        console.log(request.body.user);
        let res=await this.model.getUser(request.body)
        response
            .status(200)
            .json(res)
    }
    catch (e){

```

```

        response.status(404).json(e);
    }
}

async getOne(request, response) {
    const result = await this.model.getByld(request.params.id)
    if (result) {
        response
            .status(200)
            .json(result)
    } else {
        response
            .status(404)
            .json({
                error: 'Not found'
            })
    }
}

async create(request, response) {
    const result = await this.model.create(request.body)
    if (result.rowCount) {
        response
            .status(201)
            .json({
                status: "Success"
            })
    } else {
        response
            .status(400)
            .json({
                error: "Bad request"
            })
    }
}

}

module.exports = {
    UserController
}

```

Entity.js

```
const base = require('./BaseModel')
```

```
class Entity extends base.BaseModel {
```

```
  constructor() {  
    super();  
    this.table = 'tour'  
  }
```

```
  async addTour(data) {  
    try {  
      if(data.name_country=== "" || data.name_city=== "" || data.name_hotel=== "" ||  
data.class_hotel=== "" || data.type_transport || data.name_tour=== "" || data.date_start_tour=== "" || data.date_end  
_tour=== "" || data.price_tour=== "" || data.description_tour=== ""){  
        console.log("Не все данные введены")  
        return "Не все данные введены";  
      }  
      console.log(data)  
      let queryTour = {  
        id_hotel: 0,  
        id_city: 0,  
        id_transport: 0,  
        price_tour: data.price_tour,  
        name_tour: data.name_tour,  
        description_tour: data.description_tour,  
        date_start_tour: data.date_start_tour,  
        date_end_tour: data.date_end_tour,  
      }  
  
      let country;  
      let checkCountry = await this.connection.query(  
        `SELECT id_country  
        FROM country  
        WHERE name_country='${data.name_country}'`  
      )  
      let checkCity = await this.connection.query(  
        `SELECT id_city  
        FROM city  
        WHERE name_city='${data.name_city}'`  
      )
```

```

let checkTransport=await this.connection.query(
  `SELECT id_transport
  FROM transport
  WHERE type_transport='${data.type_transport}'`)
let checkHotel=await this.connection.query(
  `SELECT id_hotel
  FROM hotels
  WHERE name_hotel='${data.name_hotel}'`)
if (checkCountry.rowCount=== 0) {
  checkCountry = await this.connection.query(
    `INSERT INTO country(
      name_country)
    VALUES ('${data.name_country}')
    RETURNING id_country`
  )
}
country=checkCountry.rows[0].id_country;
console.log(country);
if(checkCity.rowCount===0){
  checkCity = await this.connection.query(
    `INSERT INTO city(
      name_city,id_country)
    VALUES ('${data.name_city}',${country})
    RETURNING id_city`
  )
}
queryTour.id_city=checkCity.rows[0].id_city;
if(checkTransport.rowCount===0){
  checkTransport = await this.connection.query(
    `INSERT INTO transport(
      type_transport)
    VALUES ('${data.type_transport}')
    RETURNING id_transport`
  )
}
queryTour.id_transport=checkTransport.rows[0].id_transport;
if(checkHotel.rowCount===0){
  checkTransport = await this.connection.query(
    `INSERT INTO hotels(
      name_hotel,class_hotel,id_city)
    VALUES ('${data.name_hotel}',${data.class_hotel},${queryTour.id_city})
    RETURNING id_hotel`
  )
}

```

```

    )
  }
  queryTour.id_hotel=checkHotel.rows[0].id_hotel;

  console.log(queryTour);
  const create =await this.connection.query(
    ,
    INSERT INTO public.tour(
      id_hotel, id_city, id_transport, price_tour, name_tour, description_tour, date_start_tour,
date_end_tour)
      VALUES (${queryTour.id_hotel}, ${queryTour.id_city}, ${queryTour.id_transport},
${queryTour.price_tour}, '${queryTour.name_tour}', '${queryTour.description_tour}',
'${queryTour.date_start_tour}', '${queryTour.date_end_tour}');
    ,
  )
  if (create.rowCount !== 0) {
    return (
      "Запрос выполнен успешно"
    );
  } else return ("Пользователь не найден");

} catch (e) {
  console.log(e)
  return ("Ошибка");
}

}

async reportType1() {
  try {
    let res = await this.connection.query(
      `SELECT cl.fio_client, r.id_ride, m.fio_manager, r.date_order from ride r
      JOIN client cl ON r.id_client=cl.id_client
      JOIN manager m ON r.id_manager=m.id_manager
      where date_order > CURRENT_DATE - INTERVAL '1 months' `
    )
    console.log(res.rows)
    res.rows.forEach(r => {
      let [month, day, year] = [
        r.date_order.getMonth() + 1,

```

```

        r.date_order.getDate(),
        r.date_order.getFullYear(),
    ];
    r.date_order = day + '.' + month + '.' + year;
    })
    return res;
} catch (e) {
    return "Error"
}
}
}
async reportType2() {
    try {
        let res = await this.connection.query(
            `SELECT Count(m.fio_manager),m.fio_manager,t.name_tour from ride r
            JOIN tour t ON r.id_tour=t.id_tour
            JOIN manager m ON r.id_manager=m.id_manager
            GROUP BY t.name_tour, m.fio_manager
            Order by count desc `
        )
        console.log(res.rows)
        return res;
    } catch (e) {
        return "Error"
    }
}
}
async reportType3() {
    try {
        let res = await this.connection.query(
            `SELECT cl.fio_client, Count(r.id_client),SUM(t.price_tour) from ride r
            JOIN tour t ON r.id_tour=t.id_tour
            JOIN client cl ON r.id_client=cl.id_client
            Group by cl.fio_client
            Order by count,sum desc`
        )
        console.log(res.rows)
        return res;
    } catch (e) {
        return "Error"
    }
}
}
async addRide(data) {
    try {

```

```

    let now = new Date()
    console.log(now.toISOString());
    let [month, day, year] = [
        now.getMonth() + 1,
        now.getDate(),
        now.getFullYear(),
    ];
    const res = await this.connection.query(
        `INSERT INTO ride(
            id_client, id_tour, date_order)
            VALUES (${+data.id_client}, ${+data.id_tour}, '${year+'-'+month+'-'+day}');`
    )
    return "OK";
} catch (e) {

}
}
async delTour(data) {
    try {
        const res = await this.connection.query(
            `DELETE FROM tour
                WHERE id_tour=${data.id_tour}`
        )
        return "OK";
    } catch (e) {
        return "Ошибка";
    }
}

async getTour(data) {
    try {
        const res = await this.connection.query(
            `SELECT t.id_tour, t.price_tour, t.name_tour, c.name_city,
tr.type_transport, h.name_hotel, h.class_hotel, t.description_tour, t.date_start_tour, t.date_end_tour
            FROM ${this.table} t
            JOIN city c ON t.id_city=c.id_city
            JOIN transport tr ON t.id_transport=tr.id_transport
            JOIN hotels h ON t.id_hotel=h.id_hotel
            WHERE t.id_tour=${+data.id_tour}`
        )
        return res.rows[0];
    } catch (e) {

```



```

    }
  }

  async sucRide(data) {
    try {
      console.log(data);
      const res = await this.connection.query(
        `UPDATE ride
        SET id_manager=${data.id_manager}
        WHERE id_ride=${data.id_ride};`
      )
      return res.rows[0];
    } catch (e) {

    }
  }

  async takeRide(id_manager) {
    try {
      console.log(id_manager);
      let res = await this.connection.query(
        `SELECT
t.date_start_tour,t.date_end_tour,t.name_tour,cy.name_city,h.name_hotel,c.fio_client,c.number_client,c.p
asport_client,t.price_tour from ride r
      JOIN client c ON r.id_client=c.id_client
      JOIN tour t ON r.id_tour=t.id_tour
      JOIN hotels h ON t.id_hotel=h.id_hotel
      JOIN city cy ON t.id_city=cy.id_city
      where r.id_manager=` + id_manager.id_manager
      )
      console.log(res);
      res.rows.forEach(r => {
        let [month, day, year] = [
          r.date_start_tour.getMonth() + 1,
          r.date_start_tour.getDate(),
          r.date_start_tour.getFullYear(),
        ];
        r.date_start_tour = day + '.' + month + '.' + year;
        [month, day, year] = [
          r.date_end_tour.getMonth() + 1,
          r.date_end_tour.getDate(),

```

```

        r.date_end_tour.getFullYear(),
    ];
    r.date_end_tour = day + '.' + month + '.' + year;
    })
    return res.rows;
} catch (e) {
    console.log(e)
    return e
}
}

async takeUnfRide() {
    try {
        let res = await this.connection.query(
            `SELECT r.id_ride,
t.date_start_tour,t.date_end_tour,t.name_tour,cy.name_city,h.name_hotel,c.fio_client,c.number_client,c.p
asport_client,t.price_tour from ride r
        JOIN client c ON r.id_client=c.id_client
        JOIN tour t ON r.id_tour=t.id_tour
        JOIN hotels h ON t.id_hotel=h.id_hotel
        JOIN city cy ON t.id_city=cy.id_city
        where r.id_manager is NULL;`
        )
        res.rows.forEach(r => {
            let [month, day, year] = [
                r.date_start_tour.getMonth() + 1,
                r.date_start_tour.getDate(),
                r.date_start_tour.getFullYear(),
            ];
            r.date_start_tour = day + '.' + month + '.' + year;
            [month, day, year] = [
                r.date_end_tour.getMonth() + 1,
                r.date_end_tour.getDate(),
                r.date_end_tour.getFullYear(),
            ];
            r.date_end_tour = day + '.' + month + '.' + year;
        })
        return res.rows;
    } catch (e) {
        return e
    }
}
}

```

```

async getCollections() {
  try {
    const res = await this.connection.query(
      `SELECT t.id_tour, t.price_tour, t.name_tour,c.name_city,
tr.type_transport,h.name_hotel,h.class_hotel, t.description_tour, t.date_start_tour, t.date_end_tour
FROM ${this.table} t
JOIN city c ON t.id_city=c.id_city
JOIN transport tr ON t.id_transport=tr.id_transport
JOIN hotels h ON t.id_hotel=h.id_hotel`
    )

    if (res.rowCount !== 0) {
      res.rows.forEach(r => {
        let [month, day, year] = [
          r.date_start_tour.getMonth() + 1,
          r.date_start_tour.getDate(),
          r.date_start_tour.getFullYear(),
        ];
        r.date_start_tour = day + '/' + month + '/' + year;
        [month, day, year] = [
          r.date_end_tour.getMonth() + 1,
          r.date_end_tour.getDate(),
          r.date_end_tour.getFullYear(),
        ];
        r.date_end_tour = day + '/' + month + '/' + year;
      })
      console.log(res.rows);
      return res.rows
    } else {
      return ("No data");
    }
  } catch (e) {
    console.log("Error")
    throw ("Error")
  }
}
}

```

```
module.exports = {  
  Entity  
}
```

Ride.js

```
const base = require('./BaseModel')
```

```
class Ride extends base.BaseModel {  
  
  constructor() {  
    super();  
    this.table = 'ride'  
  }  
  async getAll() {  
    const res = await this.connection.query(  
      `SELECT *  
      FROM ${this.table}`  
    )  
    return res.rows  
  }  
}
```

```
module.exports = {  
  Ride  
}
```

User.js

```
const base = require('./BaseModel')
```

```
const crypto = require('crypto');
```

```
class User extends base.BaseModel {  
  
  constructor() {  
    super();  
    this.table = 'client'  
  }  
  
  hash(pass) {  
    let name = pass;  
    let hash = crypto.createHash('md5').update(name).digest('hex');
```

```

    console.log(hash);
    return hash;
}

async checkUser(login_client, pass_client) {
    try {
        const res = await this.connection.query(
            `SELECT id_client
            FROM ${this.table}
            WHERE (login_client='${login_client}' AND pass_client='${pass_client}')`
        )
        let check_m = await this.connection.query(
            `SELECT *
            FROM manager
            WHERE (login_manager='${login_client}' AND pass_manager='${pass_client}')`
        )
        if(check_m.rowCount !== 0) {
            if(check_m.rows[0].post_manager==="младший") {
                res.msg = "manager"
                res.id_manager = check_m.rows[0].id_manager;
                console.log(res)
                return res;
            }
            else{
                res.msg = "mainManager"
                res.id_manager = check_m.rows[0].id_manager;
                console.log(res)
                return res;
            }
        }
        console.log(res.rows[0]);
        if (res.rowCount !== 0) {
            return res.rows[0]
        }
        else{
            return "No user";
        }
    } catch (e) {
        console.log(e)
        throw ("Неверный логин или пароль")
    }
}

```

```

    }

    async createUser(data, pass_client) {
      try {
        const check = await this.connection.query(
          `SELECT id_client
          FROM ${this.table}
          WHERE (login_client='${data.login}' OR passport_client='${data.passport}')`
        )
        if (check.rowCount === 0) {
          const create = await this.connection.query(
            `INSERT INTO client(
              fio_client, number_client, gender_client, passport_client, login_client, pass_client)
              VALUES ('${data.name}', '${data.number_phone}', '${data.gender}', '${data.passport}',
                '${data.login}', '${pass_client}');`
          )
          console.log(create);
          if (create.rowCount) {
            const res = await this.connection.query(
              `SELECT id_client
              FROM ${this.table}
              WHERE (login_client='${data.login}' OR passport_client='${data.passport}')`
            )
            return ({
              id: res.rows[0],
              msg: "Запрос выполнен успешно",
            });
          }
          return ("Ошибка");
        } else {
          return ("Такой пользователь существует");
        }
      }

      } catch (e) {
        console.log(e)
        return ("Ошибка");
      }
    }
  }
}

```

```

async addManager(data) {
  try {
    const check = await this.connection.query(
      `SELECT id_manager
      FROM manager
      WHERE login_manager='${data.login_manager}'`
    )
    console.log(check)
    if (check.rowCount === 0) {
      const create = await this.connection.query(
        `INSERT INTO manager(
        fio_manager, number_manager, gender_manager, login_manager, pass_manager)
        SELECT fio_client, number_client, gender_client, login_client, pass_client FROM client
WHERE login_client='${data.login_manager}';
        UPDATE public.manager
        SET post_manager='младший'
        WHERE login_manager='${data.login_manager}';`
      )
      console.log(create);
      if (create.rowCount !== 0) {
        return (
          "Запрос выполнен успешно"
        );
      }
      else return ("Пользователь не найден");
    } else {
      return ("Такой менеджер существует");
    }
  }

  } catch (e) {
    console.log(e)
    return ("Ошибка");
  }

}

async TRU(data) {
  try {
    const check = await this.connection.query(
      `SELECT id_client

```

```

FROM ${this.table}
WHERE (fio_client='${data.name}' AND passport_client='${data.passport}')`
)
if (check.rowCount === 0) {
  const create = await this.connection.query(
    `INSERT INTO client(
      fio_client,passport_client)
      VALUES ('${data.name}','${data.passport}')`;
  )
  console.log(create);
  if (create.rowCount) {
    const res = await this.connection.query(
      `SELECT id_client
      FROM ${this.table}
      WHERE (fio_client='${data.name}' AND passport_client='${data.passport}')`
    )
    return ({
      id: res.rows[0],
      fio_client: data.name,
      passport_client: data.passport,
      number_client: "",
      msg: "Запрос выполнен успешно",
    });
  }
  return ("Ошибка");
} else {
  return check.rows[0];
}

} catch (e) {
  console.log(e)
  return ("Ошибка");
}

}

async getUser(data) {
  try {
    console.log(data)
    const get = await this.connection.query(

```



```

        `SELECT *
        FROM ${this.table}
        WHERE id_client=${+data.user}`
    )
    console.log(get.rows[0]);
    return get.rows[0];
} catch (e) {
    console.log(e)
    return ("Ошибка");
}

}

}
}

module.exports = {
    User
}

```

Client

Auth.jsx

```

import React, {useEffect, useState} from 'react';
import './Auth.css';
import '../Axios/request';
import '../Axios/request';
import {Navigate} from "react-router-dom";
import axios from "axios";

const Auth = () => {
    const [login,setLogin]=useState("");
    const [password,setPassword]=useState("");
    const [user,setUser]=useState(localStorage.user);
    if(user) {
        return (
            <Navigate to={{pathname: '/'}}/>
        )
    }
    else{

```

```

return (
  <div className={"auth"}>
    <div className={"auth-window"}>
      <h1>Авторизация</h1>
      <input value={login} onChange={event => setLogin(event.target.value)} type="text"/>
      <input value={password} onChange={event => setPassword(event.target.value)}
type="password"/>
      <button onClick={() => {
        axios.post('http://localhost:3001/auth',{login, password}).then((resp)=>{
          if(resp.data!=="Неверный логин или пароль"){
            if(resp.data.msg==="manager"){
              console.log(resp.data);
              localStorage.setItem('user',resp.data.rows[0].id_client);
              localStorage.setItem('manager',resp.data.id_manager);
              setUser(localStorage.user)
              // console.log(localStorage.manager)
            }
            else if(resp.data.msg==="mainManager"){
              console.log(resp.data);
              localStorage.setItem('user',resp.data.rows[0].id_client);
              localStorage.setItem('mainManager',resp.data.id_manager);
              setUser(localStorage.user)
              console.log(localStorage.mainManager)
            }
            else {
              localStorage.setItem('user',resp.data.id_client);
              setUser(localStorage.user)
            }
          }
        })
      }}>Войти
      </button>
      <a href="http://localhost:3000/registration">Нет аккаунта?</a>
    </div>
  </div>
);
}
};

```

```

export default Auth;

```

Client.jsx

```
import React, {useEffect, useState} from 'react';
import axios from "axios";

const Client = () => {
  const [client,setClient]=useState({});
  useEffect(()=>{
    let ignore = false;

    if (!ignore) {
      axios.post('http://localhost:3001/client',localStorage).then((resp)=>{
        setClient(resp.data)
      })
    }
    return () => { ignore = true; }
  },[])
  return (
    <div>
      <h1>
        {client.fio_client}</h1>
      </div>
    );
};
```

export default Client;

Card.jsx

```
import React, {useState} from 'react';
import './Card.css'
import axios from "axios";

const Card = (props) => {
  return (
    <div className={"Card"}>
      <h1>{props.result.name_tour}</h1>
      <p>Город {props.result.name_city}</p>
      <p>Отель {props.result.name_hotel} Оценка {props.result.class_hotel}</p>
      <p>Дата начала {props.result.date_start_tour.replaceAll('/','.')} Дата окончания
{props.result.date_end_tour.replaceAll('/','.')}</p>
      <p>Цена <strong>{props.result.price_tour}</strong></p>
      <button onClick={()=>{
        window.location.assign('http://localhost:3000/select?id_card='+props.result.id_tour)
```

```

    }}>Оформить</button>
    {
      (localStorage.mainManager&& <button onClick={()=>{
        axios.post('http://localhost:3001/delTour',{id_tour:props.result.id_tour}
        ).then((resp)=>{
          alert(resp.data)
        })
      }}>Удалить</button>)
    }
  </div>
);
};

```

export default Card;

Collections.jsx

```

import React, {useEffect, useState} from 'react';
import axios from "axios";
import Card from "./Card";
import './Collections.css'

const Collections = () => {
  const [res, setRes] = useState();
  const [user,setUser]=useState()

  useEffect(() => {
    getCollections();
    setUser(localStorage,user)
  }, [])

  function getCollections() {
    console.log(res);
    axios.post('http://localhost:3001/collections', null).then((resp) => {

      if (resp.data) {
        setRes(resp.data);
      }
    })
  }

  return (
    <div className="Content">

```

```

    <div className={"Collections"}>
      {(res) && res.map(result =>
        <Card result={result} key={result.id_tour}/>
      )}
    </div>
  </div>
);
};

export default Collections;

```

Header.jsx

```

import React, {useEffect, useState} from 'react';
import './Header.css'

const Header = () => {
  const [user, setUser]=useState(localStorage.user)
  useEffect(() => {
    setUser(localStorage.user)
  }, [])
  if (user!==undefined) {
    console.log(localStorage.mainManager)
    console.log(localStorage.manager)
    if (localStorage.manager) {
      return (
        <div className={"Header"}>
          <button className={"btn-header"} onClick={() => {
            localStorage.removeItem('manager');
            setUser(localStorage.manager);
            localStorage.removeItem('user');
            setUser(localStorage.user);
          }}>Выйти
        </button>
          <button className={"btn-header ride"} onClick={() => {
            window.location.assign('http://localhost:3000/ride')
          }}>Заказы
        </button>
        </div>
      );
    }
  }
};

```

```

    } else if (localStorage.mainManager) {
      return (
        <div className={"Header"}>
          <button className={"btn-header"} onClick={() => {
            localStorage.removeItem('mainManager');
            setUser(localStorage.mainManager);
            localStorage.removeItem('user');
            setUser(localStorage.user);
          }}>Выйти
        </button>
          <button className={"btn-header ride"} onClick={() => {
            window.location.assign('http://localhost:3000/editing')
          }}>Редактирование
        </button>
          <button className={"btn-header ride"} onClick={() => {
            window.location.assign('http://localhost:3000/report')
          }}>Отчёты
        </button>
        </div>
      );
    } else {
      return (
        <div className={"Header"}>
          <button className={"btn-header"} onClick={() => {
            localStorage.removeItem('user');
            setUser(localStorage.user);
          }}>Выйти
        </button>
        </div>
      );
    }
  } else {
    return (
      <div className={"Header"}>
        <button className={"btn-header"} onClick={() => {
          window.location.assign('http://localhost:3000/auth');
        }}>Авторизация
      </button>
      </div>
    )
  }
};

```

```
export default Header;
```

Editing.jsx

```
import React, {useState} from 'react';
import axios from "axios";
import {Navigate} from "react-router-dom";

const Editing = () => {
  const [loginManager, setLoginManager] = useState("")
  const [Country, setCountry] = useState("")
  const [City, setCity] = useState("")
  const [Hotel, setHotel] = useState("")
  const [Rate, setRate] = useState("")
  const [Transport, setTransport] = useState("")
  const [NameTour, setNameTour] = useState("")
  const [DateStart, setDateStart] = useState("")
  const [DateEnd, setDateEnd] = useState("")
  const [Price, setPrice] = useState("")
  const [Description, setDescription] = useState("")
  return (
    <div className={"Editing"}>
      <div className="Content">
        <div className="addManager">
          <h1>Введите логин пользователя для добавления его в менеджеры</h1>
          <input value={loginManager} onChange={(event) => setLoginManager(event.target.value)}
type="text"/>
          <button onClick={() => {
            axios.post('http://localhost:3001/addManager', {login_manager: loginManager}).then((resp)
=> {
              if (resp.data) {
                alert(resp.data)
              }
            })
          }}>Добавить
        </button>
      </div>
      <div className={"addTour"}>
        <h1>Введите данные для добавления тура</h1>
        <h3>Введите страну</h3>
        <input value={Country} onChange={(event)=>setCountry(event.target.value)} type="text"/>
        <h3>Введите город</h3>
```

```

<input value={City} onChange={(event)=>setCity(event.target.value)} type="text"/>
    <h3>Введите отель</h3>
<input value={Hotel} onChange={(event)=>setHotel(event.target.value)} type="text"/>
    <h3>Введите оценку отеля</h3>
<input value={Rate} onChange={(event)=>setRate(event.target.value)} type="text"/>
    <h3>Введите транспорт</h3>
<input value={Transport} onChange={(event)=>setTransport(event.target.value)}
type="text"/>
    <h3>Введите название</h3>
<input value={NameTour} onChange={(event)=>setNameTour(event.target.value)}
type="text"/>
    <h3>Введите описание</h3>
    <textarea value={Description} onChange={(event => setDescription(event.target.value))}
name="" id="" cols="30" rows="10"></textarea>
    <h3>Введите дату начала</h3>
<input value={DateStart} onChange={(event)=>setDateStart(event.target.value)} type="text"/>
    <h3>Введите дату окончания</h3>
<input value={DateEnd} onChange={(event)=>setDateEnd(event.target.value)} type="text"/>
    <h3>Введите стоимость</h3>
<input value={Price} onChange={(event)=>setPrice(event.target.value)} type="text"/><br/>
<button onClick={() => {
    axios.post('http://localhost:3001/addTour', {
        name_country:Country,
        name_city: City,
        name_hotel: Hotel,
        class_hotel:Rate,
        type_transport:Transport,
        name_tour:NameTour,
        date_start_tour: DateStart,
        date_end_tour:DateEnd,
        price_tour:Price,
        description_tour:Description,
    }).then((resp)=>{
        alert(resp.data)
    })
}}>Добавить</button><br/>
</div>
</div>
</div>
);
};

```



```
export default Editing;
```

Report.jsx

```
import React from 'react';
```

```
const Report = () => {  
  return (  
    <div className={"Report"}>  
      <div className="Content">  
        <h1>Отчёты</h1>  
        <button onClick={()=>{  
          window.location.assign('http://localhost:3000/reportTable?type='+1)  
        }}>Отчёт о количестве проданных путевок за месяц</button><br/>  
        <button onClick={()=>{  
          window.location.assign('http://localhost:3000/reportTable?type='+2)  
        }}>Отчёт о самых популярных турах</button><br/>  
        <button onClick={()=>{  
          window.location.assign('http://localhost:3000/reportTable?type='+3)  
        }}>Отчёт о постоянных клиентах</button><br/>  
      </div>  
    </div>  
  );  
};
```

```
export default Report;
```

ReportTable.jsx

```
import React, {useEffect, useState} from 'react';  
import {useSearchParams} from "react-router-dom";  
import axios from "axios";  
import './ReportTable.css'
```

```
const ReportTable = () => {  
  const [searchParams, setSearchParams] = useSearchParams();  
  const [res, setRes] = useState([]);  
  useEffect(() => {  
    if (searchParams.get("type") === "1") {  
      axios.post('http://localhost:3001/reportType1', null).then(resp => {  
        setRes(resp.data.rows);  
        console.log(res);  
      });  
    }  
  });  
};
```

```

    })
  }
  if (searchParams.get("type") === "2") {
    axios.post('http://localhost:3001/reportType2', null).then(resp => {
      setRes(resp.data.rows);
      console.log(res);
    })
  }
  if (searchParams.get("type") === "3") {
    axios.post('http://localhost:3001/reportType3', null).then(resp => {
      setRes(resp.data.rows);
      console.log(res);
    })
  }
}, [])
return (
  <div className={"Table"}>
    {(searchParams.get("type") === "1" && <div className="Content">
      <h1>Отчёт о количестве проданных путёвок за месяц</h1>
      <table>

        <tr>

          <th>ФИО Клиента</th>
          <th>id заказа</th>
          <th>ФИО менеджера</th>
          <th>Дата Заказа</th>
        </tr>
        {res.map((r) => {
          return (
            <tr>
              <td>{r.fio_client}</td>
              <td>{r.id_ride}</td>
              <td>{r.fio_manager}</td>
              <td>{r.date_order}</td>
            </tr>
          )
        })}
      </table>
    </div>
  )}
  {(searchParams.get("type") === "2" && <div className="Content">

```

```

<h1>Самые продаваемые туры</h1>
<table>

  <tr>

    <th>ФИО менеджера</th>
    <th>Количество продаж</th>
    <th>Название тура</th>
  </tr>
  {res.map((r) => {
    return (
      <tr>
        <td>{r.fio_manager}</td>
        <td>{r.count}</td>
        <td>{r.name_tour}</td>
      </tr>
    )
  })}
</table>
</div>
)}
{{(searchParams.get("type") === "3" && <div className="Content">
  <h1>Отчёт о постоянных клиентах</h1>
  <table>

    <tr>

      <th>ФИО клиента</th>
      <th>Количество покупок</th>
      <th>Сумма покупок</th>
    </tr>
    {res.map((r) => {
      return (
        <tr>
          <td>{r.fio_client}</td>
          <td>{r.count}</td>
          <td>{r.sum}</td>
        </tr>
      )
    })}
  </table>
</div>

```

```

    })
  </div>
)
};

```

```
export default ReportTable;
```

Ride.jsx

```

import React, {useEffect, useState} from 'react';
import axios from "axios";
import './Ride.css'

```

```

const Ride = () => {
  const [res, setRes] = useState([]);
  const [nres, setNRes] = useState([]);
  useEffect(() => {
    axios.post('http://localhost:3001/giveUnfRide', null).then((resp) => {
      console.log(resp)
      if (resp.data) {
        setRes(resp.data);
      }
    })
    console.log(localStorage.manager)
    axios.post('http://localhost:3001/takeRide', {id_manager: +localStorage.manager}).then((resp) => {
      if (resp.data) {
        setNRes(resp.data);
      }
    })
  }, [])
  return (
    <div className={"Ride"}>
      <div className={"Content"}>
        <h1>Неподтверждённые заказы</h1>
        <div className="unRide">
          {res.map((result) => {
            return (
              <div className={"Card"}>
                <p>ФИО {result.fio_client}</p>
                <p>Паспорт {result.pasport_client}</p>
                <p>Номер телефона {result.number_client}</p>
                <p>Название тура {result.name_tour}</p>

```

```

        <p>Город {result.name_city}</p>
        <p>ОТЕЛЬ {result.name_hotel}</p>
        <p>Дата старта тура {result.date_start_tour}</p>
        <p>Дата окончания тура {result.date_end_tour}</p>
        <p>Цена тура {result.price_tour}</p>
        <button onClick={()=>{

            console.log(result)
            axios.post('http://localhost:3001/sucRide',{id_manager:
+localStorage.manager,id_ride:result.id_ride} ).then((resp) => {
                if (resp.data) {
                    setNRes(resp.data);
                    console.log(resp.data);
                }
            })
        }}>Подтвердить</button>
    </div>
)
}}
</div>
</div>

```

```

<div className="Content">
    <h1>Подтверждённые заказы</h1>
    <div className="sucRide">
        {nres.map((result) => {
            return (
                <div className="Card">
                    <p>ФИО {result.fio_client}</p>
                    <p>Паспорт {result.pasport_client}</p>
                    <p>Номер телефона {result.number_client}</p>
                    <p>Название тура {result.name_tour}</p>
                    <p>Город {result.name_city}</p>
                    <p>ОТЕЛЬ {result.name_hotel}</p>
                    <p>Дата старта тура {result.date_start_tour}</p>
                    <p>Дата окончания тура {result.date_end_tour}</p>
                    <p>Цена тура {result.price_tour}</p>
                </div>
            )
        })}
    </div>
</div>

```

```

    </div>
  );
};

export default Ride;

```

Reg.jsx

```

import React, {useEffect, useState} from 'react';
import {Navigate} from "react-router-dom";
import axios from "axios";
import './Reg.css'

const Reg = () => {
  const [gender,setGender]=useState("")
  const [name,setName]=useState("");
  const [login,setLogin]=useState("");
  const [number_phone,setNumberPhone]=useState("");
  const [passport,setPassport]=useState("");
  const [password,setPassword]=useState("");
  const [conf_password,setConfPassword]=useState("");
  const [user,setUser]=useState(localStorage.user);
  useEffect(()=>{
    setUser(localStorage.user)
  },[])

  // console.log(localStorage.user);
  if(user===undefined) {

    return (
      <div className={"reg"}>
        <div className={"reg-window"}>
          <h1>Регистрация</h1>
          <p>Введите имя</p>
          <input value={name} onChange={event => setName(event.target.value)} type="text"/>
          <p>Введите логин</p>
          <input value={login} onChange={event => setLogin(event.target.value)} type="text"/>
          <p>Введите пол</p>
          <input type="radio" checked={gender === 'мужской'} value="мужской"
onChange={(event)=>setGender(event.target.value)} /> Мужской
          <input type="radio" checked={gender === 'женский'} value="женский"
onChange={(event)=>setGender(event.target.value)} /> Женский

```

```

        <p>Введите номер телефона</p>
        <input value={number_phone} onChange={event => setNumberPhone(event.target.value)}
type="text"/>
        <p>Введите номер паспорта</p>
        <input value={passport} onChange={event => setPassport(event.target.value)} type="text"/>
        <p>Введите пароль</p>

        <input value={password} onChange={event => setPassword(event.target.value)}
type="password"/>
        <p>Повторите пароль</p>
        <input value={conf_password} onChange={event => setConfPassword(event.target.value)}
type="password"/> <br/>

        <button onClick={() => {
            if(conf_password===password) {
                axios.post('http://localhost:3001/registration',{name, number_phone, passport,gender,
login, password}).then((resp)=>{
                    // console.log(resp.data)
                    if(resp.data.msg==="Запрос выполнен успешно"){
                        localStorage.setItem('user',resp.data.is.id_client)
                        setUser(localStorage.user)

                    }
                })
            }
            else{
                alert("Проверьте введённые данные")
            }
        }
        }>Зарегистрироваться
        </button>
    </div>
</div>
);
}
else{
    return (
        window.location.assign('http://localhost:3000/')
    )
}
};

```

```
export default Reg;
```

App.js

```
import './App.css';
import {BrowserRouter, Link, Route, Routes} from "react-router-dom";
import Main from './Main/Main';
import Auth from './Auth/Auth';
import React from "react";
import Reg from './Registration/Reg';
import Client from './Client/Client';
import Collections from './Collections/Collections';
import Header from './Header/Header';
import Select from './Client/Select';
import Ride from './Manager/Ride';
import Editing from './Manager/Editing';
import Report from './Manager/Report';
import ReportTable from './Manager/ReportTable';

function App() {

  return (
    <BrowserRouter>
      <div className="App">
        <Header/>
        <Routes>
          <Route path="/" element={<Collections/>}/>
          <Route path="/auth" element={<Auth/>}/>
          <Route path="/registration" element={<Reg/>}/>
          <Route path="/client" element={<Client/>}/>
          <Route path="/select" element={<Select/>}/>
          <Route path="/ride" element={<Ride/>}/>
          <Route path="/editing" element={<Editing/>}/>
          <Route path="/report" element={<Report/>}/>
          <Route path="/reportTable" element={<ReportTable/>}/>
        </Routes>

      </div>
    </BrowserRouter>
  );
}

export default App;
```




Регистрация

Введите имя

Введите логин

Введите пол

☐ Мужской ☐ Женский

Введите номер телефона

Введите номер паспорта

Введите пароль

Повторите пароль

Рисунок 3 – Регистрация в систему

Авторизация

Войти

[Нет аккаунта?](#)

Рисунок 4 – Авторизация

Выйти Редактирование Отчеты			
Сакура Город Токио Отель Lord Elgin Hotel Оценка 4 Дата начала 1.2.2021 Дата окончания 15.2.2021 Цена 180000.00 Оформить Удалить	Нил Город Каир Отель Hyatt House Оценка 3 Дата начала 1.5.2021 Дата окончания 15.5.2021 Цена 80000.00 Оформить Удалить	Sea Breeze Город Токио Отель Sunword Hotel Оценка 4 Дата начала 1.8.2021 Дата окончания 11.8.2021 Цена 50000.00 Оформить Удалить	Сосны и реки Город Москва Отель Измайлово Оценка 3 Дата начала 1.7.2021 Дата окончания 8.7.2021 Цена 18000.00 Оформить Удалить
Райский круиз Город Оттава Отель Fairmont Nile City Оценка 5 Дата начала 10.6.2021 Дата окончания 24.6.2021 Цена 18000.00 Оформить Удалить			

Рисунок 5 – Главное меню для Главного менеджера

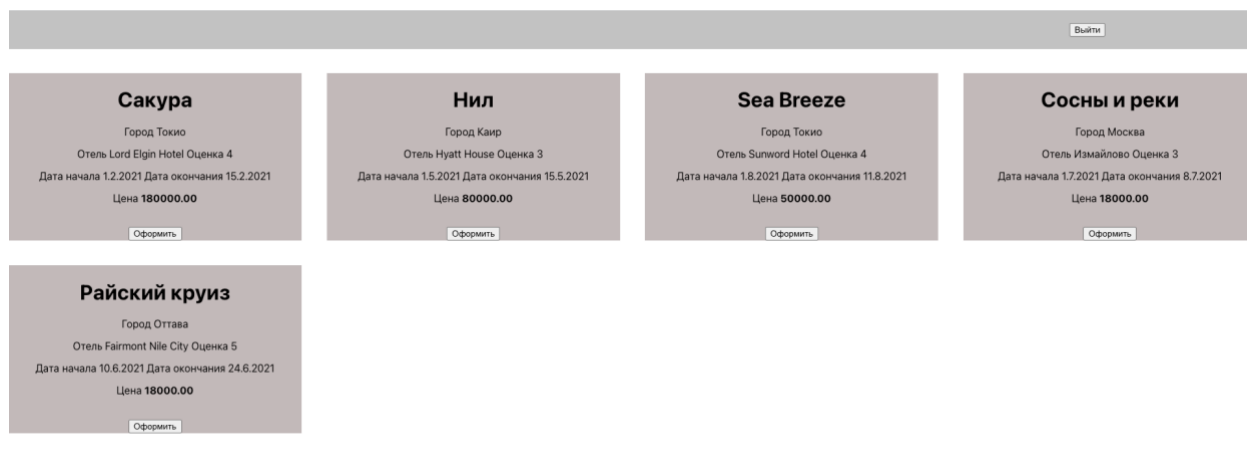


Рисунок 6 – Главное меню для пользователя

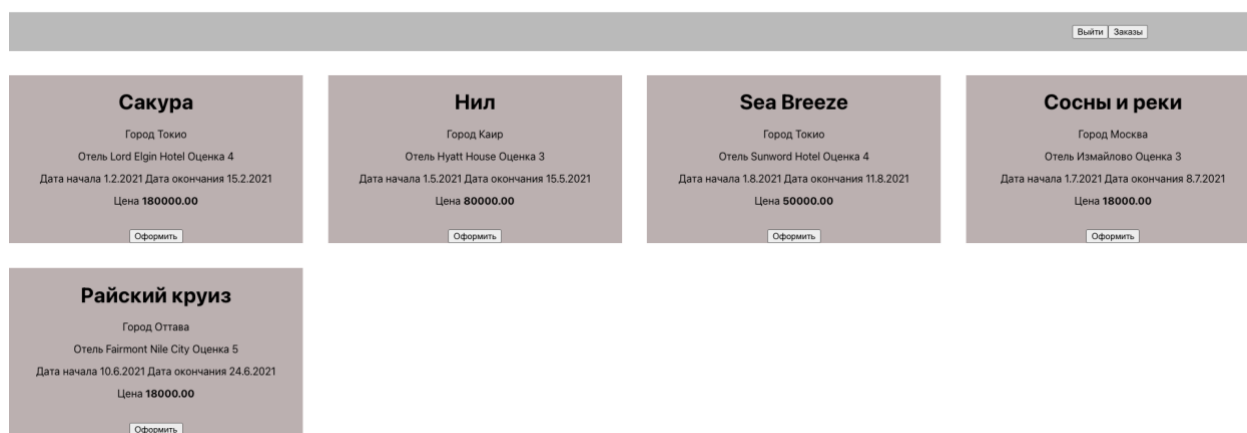


Рисунок 7 – Главное меню для младшего менеджера

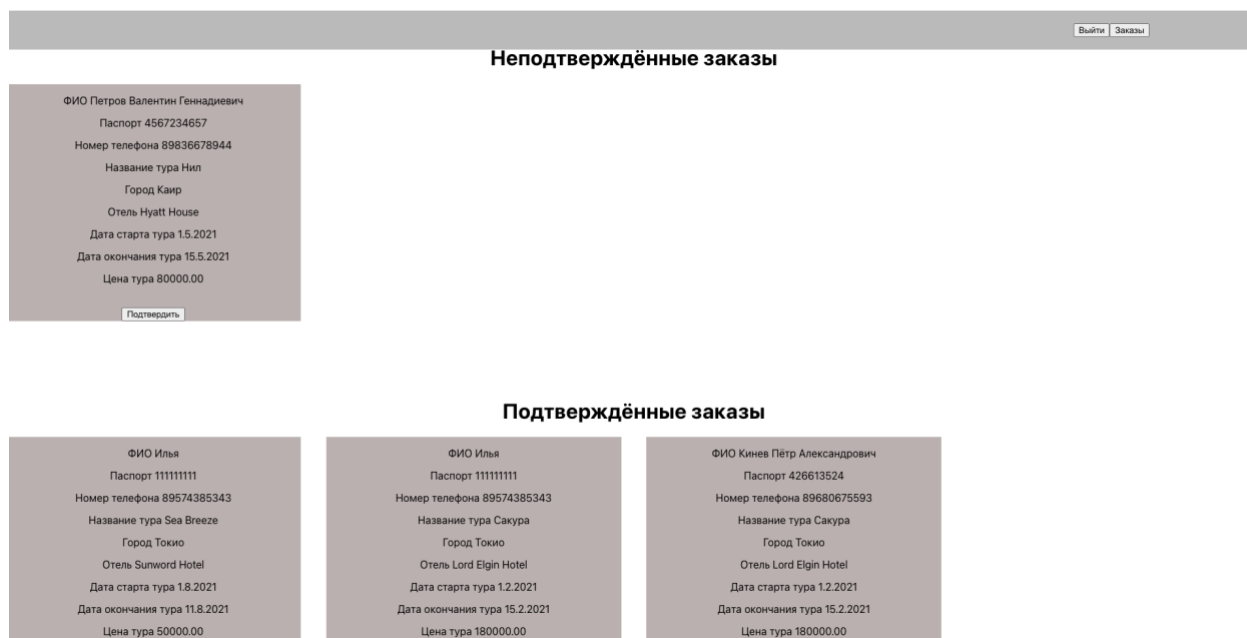


Рисунок 8 – Страница подтверждения заказов для младшего менеджера

Имя Илья

Телефон 89574385343

Паспорт 111111111

Тур Сакура

Цена 180000.00

Подтвердить

Рисунок 9 – Страница покупки туров

Введите логин пользователя для добавления его в менеджеры

Добавить

Введите данные для добавления тура

Введите страну

Введите город

Введите отель

Введите оценку отеля

Введите транспорт

Введите название

Введите описание

Введите дату начала

Введите дату окончания

Введите стоимость

Добавить

Рисунок 10– Страница редактирования для главного менеджера

Отчёты

Отчёт о количестве проданных путёвок за месяц

Отчёт о самых популярных турах

Отчёт о постоянных клиентах

Рисунок 11– Страница выбора отчёта для главного менеджера

Отчёт о количестве проданных путёвок за месяц

ФИО Клиента	id заказа	ФИО менеджера	Дата Заказа
Илья	9	Илья	25.12.2022
Илья	10	Илья	26.12.2022

Рисунок 11– пример отчёта для главного менеджера

Вывод:

При выполнении индивидуального домашнего задания были получены навыки разработки прикладного приложения для БД.