Dokumentacja miniprojektu bazy danych

Temat: Hotel z rezerwacją pokoi

Autor: Dariusz Cebula dcebula@student.agh.edu.pl

Proponowany schemat bazy danych:

- Kolekcja `Hotel`:
  - `name` - nazwa hotelu
  - `description` - opis hotelu
  - `address` - adres hotelu
  - `key-features` - kluczowe cechy hotelu np. darmowe WiFi
  - `facilities` - udogodnienia oferowane przez hotel np. spa, basen
  - `hotel-area-info` - informację o pobliskich miejscach i atrakcjach wraz z odległością w metrach
  - `check-in` - godziny zameldowania
  - `check-out` - godziny wymeldowania

Kolekcja `Hotel` głównie służy do przetrzymywania informacji o hotelu, którę będą wyświetlane na stronie dla klientów


```
{
  "_id": {
    "$oid": "661eb850859031cbe592c7ca"
  },
  "name": "Tranquil Haven Hotel",
  "description": "Welcome to Tranquil Haven Hotel, where every moment is an escape into serenity. Nestled amidst
lush greenery and tranquil surroundings, our hotel offers a sanctuary for those seeking respite from the hustle
and bustle of everyday life. With elegantly appointed rooms and suites, each designed to provide the utmost
comfort and relaxation, guests are invited to unwind in style. Indulge in our world-class amenities, including a
rejuvenating spa offering a range of wellness treatments, a picturesque swimming pool surrounded by verdant
gardens, and exquisite dining experiences that tantalize the taste buds with culinary delights from around the
globe. Whether you're here for a romantic getaway, a family vacation, or a corporate retreat, our attentive staff
is dedicated to ensuring your stay is nothing short of extraordinary. Discover the perfect blend of luxury and
tranquility at Tranquil Haven Hotel, where every moment is a peaceful escape.",
  "address": "123 Serenity Lane Peaceful Valley, CA 98765 United States",
  "key-features": [
    "pet friendly",
    "free wifi",
    "air conditioning",
    "elevator",
    "safe",
    "spa",
    "luxurious accommodations",
    "scenic location",
    "fitness center",
    "event spaces",
    "gourmet dining"
  ],
  "facilities": [
    "Spa and Wellness Center",
    "Fine Dining Restaurants",
    "Outdoor Pool",
    "Fitness Center",
    "Conference and Event Spaces",
    "Business Center",
    "Lounge Areas",
    "Children's Play Area",
    "24-Hour Reception and Security"
  ],
  "hotel-area-info": [
    {
      "Nearest Airport": 20000
    },
    {
      "City Center": 5000
```

```
    },
    {
      "Tranquil Park": 1000
    },
    {
      "Artisan Museum": 3000
    },
    {
      "Serenity Mall": 2000
    },
    {
      "Tranquil Market": 500
    },
    {
      "Nearest Bus Stop": 100
    }
  ],
  "check-in": "15:00",
  "check-out": "12:00"
}
```

- Kolekcja `Guest`:
  - `firstname` - imię gościa
  - `lastname` - nazwisko gościa
  - `birthday` - data urodzin gościa
  - `address` - adres gościa
  - `email` - adres e-mail gościa
  - `phone` - numer telefonu gościa

Kolekcja `Guest` przetrzymuję informację o gościach co złożyli rezerwację pokoi

```
{
  "_id": {
    "$oid": "661ec025859031cbe592c7d4"
  },
  "first_name": "John",
  "last_name": "Doe",
  "birthday": "1980-05-15",
  "address": "123 Main Street, Cityville, USA",
  "email": "johndoe@example.com",
  "phone": "+1234567890"
}
```

- Kolekcja `Room`:
  - `room_number` - numer pokoju
  - `price` - cena za jeden nocleg
  - `capacity` - pojemność pokoju (liczba gości)
  - `include` - elementy uwzględnione w pokoju np. łóżko typu queen, sofa
  - `area` - powierzchnia pokoju w metrach kwadratowych
  - `features` - dodatkowe funkcje i udogodnienia dostępne w pokoju np. darmowe WiFi, klimatyzacja, minibar

Kolekcja `Room` przetrzymuję informacje o pokojach dostępnych w hotelu do rezerwacji

```
{
  "_id": {
    "$oid": "661ec440859031cbe592c7e4"
  },
  "room_number": 101,
  "price": 150,
  "capacity": 2,
  "include": [
    {
      "queen bed": 1
    }
  ],
  "area": 25,
```

```
  "features": [
    "free wifi",
    "air conditioning",
    "minibar",
    "flat-screen TV"
  ]
},
{
  "_id": {
    "$oid": "661ec460859031cbe592c7e5"
  },
  "room_number": 102,
  "price": 200,
  "capacity": 4,
  "include": [
    {
      "king bed": 1
    },
    {
      "sofa bed": 1
    }
  ],
  "area": 35,
  "features": [
    "free wifi",
    "air conditioning",
    "mini fridge",
    "work desk",
    "balcony"
  ]
}
```


- Kolekcja `Food`:
  - `name` - nazwa typu wyżywienia
  - `description` - opis wyżywienia
  - `type` - typ wyżywienia
  - `price` - cena wyżywienia za jeden dzień

Kolekcja `Food` przetrzymuje informację o dostępnych typach wyżywienia jakie możemy wybrać przy rezerwacji pokoju


```
{
  "_id": {
    "$oid": "661ed5de859031cbe592c816"
  },
  "name": "All-Inclusive Plan",
  "description": "Includes breakfast, lunch, and dinner and 24-hours minibar with beverages and snacks.",
  "type": "all_inclusive",
  "price": 50
},
{
  "_id": {
    "$oid": "661ed63a859031cbe592c818"
  },
  "name": "Breakfast Only",
  "description": "Includes breakfast only.",
  "type": "breakfast_only",
  "price": 10
},
{
  "_id": {
    "$oid": "661ed683859031cbe592c819"
  },
  "name": "Breakfast and Dinner",
  "description": "Includes breakfast and dinner.",
  "type": "breakfast_dinner",
  "price": 20
},
{
  "_id": {
```

```
      "$oid": "661ed690859031cbe592c81a"
    },
    "name": "Breakfast, Lunch, and Dinner",
    "description": "Includes breakfast, lunch, and dinner.",
    "type": "full_board",
    "price": 30
}
```

- Kolekcja `Reservation`:
  - `guests` - lista identyfikatorów gości zerezerwowanych na pobyt
  - `rooms` - lista numerów pokoi zarezerwowanych na pobyt
  - `check_in` - data zameldowania
  - `check_out` - data wymeldowania
  - `total_guests` - całkowita liczba gości objętych rezerwacją
  - `status` - status rezerwacji (`"confirmed"`, `"pending"`, `"canceled"`)
  - `notes` - dodatkowe uwagi lub życzenia związane z rezerwacją
  - `created_at` - data i czas utworzenia rezerwacji
  - `updated_at` - data i czas ostatniej aktualizacji rezerwacji
  - `food_type` - typ wyżywienia np. `all_inclusive`, `full_board`, `breakfast_only`

Kolekcja `Reservation` przetrzymuję informację o wszystkich rezerwacjach. Rezerwacja jest dokonywaa na konkretną liczbą gości i wszyscy są potem dopisywani do kolekcji `Guest`, chyba że już kiedyś złożyli rezerwację w tym hotelu. Dodatkowo rezerwacja może obejmować większą liczbę pokoi.

```
{
    "_id": {
      "$oid": "661ec9bc859031cbe592c80b"
    },
    "guests": [
      {
        "$oid": "661ec025859031cbe592c7d4"
      }
    ],
    "rooms": [
      103
    ],
    "check_in": "2024-07-15",
    "check_out": "2024-07-20",
    "total_guests": 1,
    "total_price": 750,
    "status": "confirmed",
    "notes": "Guest prefers a room with a city view.",
    "created_at": "2024-06-01T09:00:00Z",
    "updated_at": "2024-06-02T14:30:00Z",
    "food_type": "all_inclusive"
},
{
    "_id": {
      "$oid": "661ecad4859031cbe592c80d"
    },
    "guests": [
      {
        "$oid": "661ec025859031cbe592c7d4"
      },
      {
        "$oid": "661ec048859031cbe592c7d5"
      },
      {
        "$oid": "661ec066859031cbe592c7d6"
      }
    ],
    "rooms": [
      101,
      103
    ],
    "check_in": "2024-09-05",
    "check_out": "2024-09-10",
    "total_guests": 3,
    "total_price": 1620,
```

```
  "status": "confirmed",
  "notes": "Group reservation for a family vacation.",
  "created_at": "2024-08-01T10:00:00Z",
  "updated_at": "2024-08-02T12:00:00Z",
  "food_type": "full_board"
}
```

- Kolekcja `Review`:
  - `guest_id` - identyfikator gościa, który wystawił recenzję
  - `categories` - oceny poszczególnych kategorii np. `facilities`, `comfort`, `location`
  - `topics` - tematy omawiane w recencji np. `room`, `breakfast`
  - `date` - data wystawienia recenzji
  - `text` - treść omawianej recenzji
  - `score` - ogólna ocena recenzji
  - `reservation_id` - identyfikator rezerwacji na którą została wystyawiona recenzja

Kolekcja `Review` przechowuję wszystkie recenzje dodane przez gości, któzy odwiedzili nasz hotel. Recenzja obejmuję ocenę w różnych kategoriach i posiada ocenę końcową na podstawie średniej ze wszystkich kategorii. Recenzja musi być wystawiona na konkretną rezerwację i przez gościa, który jest powiązany z tą rezerwacją.

```
{
  "_id": {
    "$oid": "661ebc53859031cbe592c7ce"
  },
  "guest_id": {
    "$oid": "661ec025859031cbe592c7d4"
  },
  "categories": {
    "staff": 9,
    "facilities": 8,
    "cleanliness": 9,
    "comfort": 9,
    "value_for_money": 8,
    "location": 10,
    "free_wifi": 10
  },
  "topics": [
    "breakfast",
    "room",
    "clean"
  ],
  "date": "2024-04-16T08:00:00Z",
  "text": "I had a fantastic stay at the Tranquil Haven Hotel. The staff were incredibly helpful and friendly,
and the facilities were top-notch. My room was spotlessly clean and very comfortable. The breakfast options were
delicious, and the hotel's location was perfect for exploring the city. The free WiFi was also a great bonus.
Highly recommended!",
  "score": 9,
  "reservation_id": {
    "$oid": "661ec9bc859031cbe592c80b"
  }
}
```

Aplikacja Hotelu została napisana w frameworku Express. Korzystam z modułu `mongodb` do komunikacji z serwerem bazy danych MongoDB i modułu `node-crone` do stworzenia funkcji, które będą wywoływane regularnie o określonej godzinie. Do wyświetlania zawartości na stronie używam schematów `pug` do któych przekazuje odpowiednie dane.

Przykładowe zapytania do bazy danych

Dodanie nowej rezerwacji

- na początku użytkownik wybiera zakres dat i ilość gości

- następnie na nowej stronie jeśli są dostępne pokoje z wystarczającą ilością miejsc, wyświetlany jest formularz dla kazdego gościa do wypełnienia oraz dostępne pokoje w podanym terminie i dostępne typy wyżywienia
- jeśli użytkownik już kiedyś rezerwował pokój to wystarczy, że poda swój adres email, w przeciwnym wypadku wymagane są pełne informacje
- do pobrania dostępnych pokoi wykonuję agregację na kolekcji `room` i przy wyborze pokoi sprawdzam tylko rezerwację które mają status `confirmed` lub `pending`

```
import express from "express"
import {MongoClient} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.post('/', async (req,res) => {
    const {dateFrom, dateTo, guestCount} = req.body
    console.log(req.body)
    let counts = []
    for(let i=1; i<= parseInt(guestCount); i++){
        counts.push(i)
    }

    await client.connect()
    const db = client.db("Hotel")
    const foodCollection = db.collection('food')
    const roomCollection = db.collection('room')

    const availableRooms = await roomCollection.aggregate([
        {
            $lookup: {
                from: 'reservation',
                let: {room_number: '$room_number'},
                pipeline: [
                    {
                        $match: {
                            $expr: {
                                $and: [
                                    {
                                        $in: ['$$room_number', '$rooms']
                                    },
                                    {
                                        $or: [
                                            {
                                                $and: [
                                                    { $gte: ['$check_in', dateFrom] },
                                                    { $lte: ['$check_in', dateTo] },
                                                    { $or: [
                                                        { $eq: ["$status", "confirmed"] },
                                                        { $eq: ["$status", "pending"] }
                                                    ] }
                                                ]
                                            },
                                            {
                                                $and: [
                                                    { $gte: ['$check_out', dateFrom] },
                                                    { $lte: ['$check_out', dateTo] },
                                                    { $or: [
                                                        { $eq: ["$status", "confirmed"] },
                                                        { $eq: ["$status", "pending"] }
                                                    ] }
                                                ]
                                            }
                                        ]
                                    }
                                ]
                            }
                        }
                    }
                ],
                as: "reservations"
            },
```

```
        },
        {
            $match: { reservations: { $eq: [] } },
        },
        {
            $project: {
                _id: 1,
                room_number: 1,
                price: 1,
                capacity: 1,
                include: 1,
                area: 1,
                features: 1,
            },
        },
    ]).toArray();


    console.log(availableRooms)
    const food = await foodCollection.find().toArray()

    let maxCapacity = 0;
    availableRooms.forEach(room => {
        maxCapacity += room.capacity
    })

    if(maxCapacity < guestCount){
        res.send("Not enough room available for your reservation try different dates")
    } else {
        res.render('reservationGuestData', {data: {dateFrom: dateFrom, dateTo: dateTo, guestCount: guestCount,
count: counts, room: availableRooms, food: food}})
    }
})

export default router
```

- po wypełnieniu formularzy i wyborze pokoi i wyrzywienia, jeśli wszystkie dane są poprawnie wprowadzone, tworzona jest rezerwacja o statusie `pending`
- jeśli użytkownik o podanym email jest już bazie danych to pobieramy jego dane z bazy danych, w przeciwnym wypadku nowy użytkowniki jest dodawany do kolekcji `guest`
- na stronie wyświetlane są szczegóły stworzonej rezerwacji
- następnie użytkownik ma opcję, albo anulować, albo zapłacić za rezerwację

```
import express from "express"
import {MongoClient} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.post('/', async (req,res) => {
    let {note, rooms, food, dateFrom, dateTo, count} = req.body

    const roomNumbers = Array.isArray(rooms) ? rooms.map(room => parseInt(room)) : [parseInt(rooms)];

    console.log(roomNumbers)

    try{
        await client.connect()
        const db = client.db("Hotel")
        const guestCollection = db.collection('guest')
        const roomCollection = db.collection('room')
        const reservationCollection = db.collection('reservation')
        const foodCollection = db.collection('food')

        const roomInfo = await roomCollection.aggregate([
            {
                $match: {
                    room_number: {$in: roomNumbers}
```

```
            }
        },
        {
            $group: {
                _id: null,
                capacity: {
                    $sum: "$capacity"
                },
                price_per_day: {
                    $sum: "$price"
                }
            }
        }
    ]).toArray()

    if(roomInfo[0].capacity < count){
        res.send("Not enough rooms for guests in chosen rooms")
        return
    }

    let guestsIds = []
    let guestsInfo = []
    for(let i=1; i<=parseInt(count); i++){
        let email = req.body[`email_${i}`]
        let firstname = req.body[`first_name_${i}`]
        let lastname = req.body[`last_name_${i}`]
        let birthday = req.body[`birthday_${i}`]
        let address = req.body[`address_${i}`]
        let phone = req.body[`phone_${i}`]

        let guest = await guestCollection.findOne({email: email})

        if(!guest){
            if(!validateGuestData(email, firstname, lastname, birthday, address, phone)){
                res.send("Invalid guest data")
                return
            }

            let newGuest = {
                email: email,
                first_name: firstname,
                last_name: lastname,
                birthday: birthday,
                address: address,
                phone: phone
            }

            const result = await guestCollection.insertOne(newGuest)
            guestsIds.push(result.insertedId)
            guestsInfo.push(newGuest)
        } else{
            guestsIds.push(guest._id)
            guestsInfo.push(guest)
        }

    }
    const getFoodType = await foodCollection.findOne({type: food})

    const days = getDays(dateFrom,dateTo)

    const total_price = roomInfo[0].price_per_day * days + getFoodType.price * days

    const currDate = new Date().toISOString()

    const newReservation = {
        guests: guestsIds,
        rooms: roomNumbers,
        check_in: dateFrom,
        check_out: dateTo,
        total_guests: parseInt(count),
        total_price: total_price,
        status: "pending",
```

```
            notes: note,
            create_at: currDate,
            updated_at: currDate,
            food_type: food
        }
        console.log(newReservation,guestsInfo)
        const result = await reservationCollection.insertOne(newReservation)

        console.log(result)
        console.log(result.insertedId)

        res.render('reservationResult', {data: {newReservation: newReservation, guests: guestsInfo,
reservationId: result.insertedId.toString()}})
    } catch(error){
        console.error("Error adding reservation:", error);
        res.status(500).send("Internal Server Error");
    }
})

function validateGuestData(email, firstName, lastName, birthday, address, phone) {
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(email)) {
        return false;
    }

    const phoneRegex = /^\+\d{1,3}\d{3,}$/;
    if (!phoneRegex.test(phone)) {
        return false;
    }

    const birthdayRegex = /^\d{4}-\d{2}-\d{2}$/;
    if (!birthdayRegex.test(birthday)) {
        return false;
    }

    if(firstName == "" || lastName == "" || address == ""){
        return false
    }

    return true;
}


function getDays(dateFrom, dateTo){
    const from = new Date(dateFrom);
    const to = new Date(dateTo);
    const difference = to - from;
    const days = difference / (1000 * 3600 * 24);
    return Math.ceil(days) + 1;
}

export default router
```

Wyświetlanie recenzji wraz z filtrowanie oraz dodanie nowej recezji

- na stronie z recenzjami początkowo wyświetlane są średnie statystyki z wszystkich recezji oraz 10 najwyżej ocenionych recezji
- użytkonik może przefiltrować recenzję po dacie lub ogólnej ocenie lub wybrać temat recezji do wyświetlenia oraz wybrać typ sortowania `asc` lub `desc`

```
router.get('/', async (req,res) => {
    await client.connect()
    const db = client.db("Hotel")
    const reviewCollection = db.collection("review")

    const summary = await reviewCollection.aggregate([
        {
            $group: {
                _id: null,
                avgScore: {$avg: "$score"},
```

```
                avgStaff: {$avg: "$categories.staff"},
                avgFacilities: {$avg: "$categories.facilities"},
                avgCleanliness: {$avg: "$categories.cleanliness"},
                avgComfort: {$avg: "$categories.comfort"},
                avgValueForMoney: {$avg: "$categories.value_for_money"},
                avgLocation: {$avg: "$categories.location"},
                avgFreeWifi: {$avg: "$categories.free_wifi"},
                count: {$sum: 1}
            }
        }
    ]).toArray()

    const reviews = await reviewCollection.aggregate([
        {
            $lookup: {
                from: 'guest',
                localField: 'guest_id',
                foreignField: '_id',
                as: 'guest_info'
            }
        },
        {
            $unwind: '$guest_info'
        },
        {
            $sort: {score: -1}
        },
        {
            $limit: 10
        }
    ]).toArray()

    console.log(summary)
    console.log(reviews)

    res.render('review', {data: {summary: summary[0], reviews: reviews}})
})
```

- pobranie recenzji w zależności od wybranego filtra `date` lub `score` i opcjonalnie wybranie recenzji zawierających okrelony `topic`

```
router.post('/', async (req,res) => {
    const {topic, sort, asc_desc} = req.body

    const ascDesc = parseInt(asc_desc)

    await client.connect()
    const db = client.db("Hotel")
    const reviewCollection = db.collection('review')

    const matchStage = topic ? {topics: topic} : {}
    const sortStage = sort === 'date' ? {date: ascDesc} : {score: ascDesc}

    const summary = await reviewCollection.aggregate([
        {
            $group: {
                _id: null,
                avgScore: {$avg: "$score"},
                avgStaff: {$avg: "$categories.staff"},
                avgFacilities: {$avg: "$categories.facilities"},
                avgCleanliness: {$avg: "$categories.cleanliness"},
                avgComfort: {$avg: "$categories.comfort"},
                avgValueForMoney: {$avg: "$categories.value_for_money"},
                avgLocation: {$avg: "$categories.location"},
                avgFreeWifi: {$avg: "$categories.free_wifi"},
                count: {$sum: 1}
            }
        }
    ]).toArray()

    const reviews = await reviewCollection.aggregate([
```

```
        {
            $match: matchStage
        },
        {
            $lookup: {
                from: 'guest',
                localField: 'guest_id',
                foreignField: '_id',
                as: 'guest_info'
            }
        },
        {
            $unwind: '$guest_info'
        },
        {
            $sort: sortStage
        },
        {
            $limit: 10
        }
    ]).toArray();

    res.render('review', {data: {summary: summary[0], reviews: reviews}})
})
```

- dodanie nowej recenzji
- może być dodany tylko przez użytkownika, który złożył rezerwację
- aby dodać nową recezję użytkownik musi podać unikalny token identyfikujący rezerwację, którą otrzyma np. na adres email, i podać adres email powiązany z tą rezerwacją
- poniżej w formularzu użytkownik może sprecyzować temat recezji, treść recenzji, oraz odpowiednie oceny dla każdej kategorii
- jeśli token i email się zgadza to recezja zostanie dodana do bazy danych

```
import express from "express"
import {MongoClient, ObjectId} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.get('/', async (req,res) => {
    res.render('addReview')
})

router.post('/submit', async (req,res) => {
    const {email, token, topics, text, staff, facilities, cleanliness, comfort, value_for_money, location,
free_wifi} = req.body

    console.log(req.body)

    try{
        await client.connect()
        const db = client.db("Hotel")
        const guestCollection = db.collection("guest")
        const reservationCollection = db.collection("reservation")
        const reviewCollection = db.collection("review")

        const guest = await guestCollection.findOne({email})

        console.log(guest)

        if(!guest){
            return res.status(400).send("Guest with provided email not found")
        }

        const reservation = await reservationCollection.findOne({_id: new ObjectId(token)})

        console.log(reservation)

        if(!reservation){
```

```
            return res.status(400).send("Reservation with provided token not found")
        }

        const reservationGuests = reservation.guests.map(id => id.toString())

        if(!reservationGuests.includes(guest._id.toString())){
            return res.status(400).send("User is not assigned to this reservation")
        }

        const score = parseFloat(((parseInt(staff) + parseInt(facilities) + parseInt(cleanliness) +
parseInt(comfort) + parseInt(value_for_money) + parseInt(location) + parseInt(free_wifi)) / 7).toFixed(2))

        const currentDate = new Date().toISOString()

        const newReview = {
            guest_id: guest._id,
            categories: {
                staff: parseInt(staff),
                facilities: parseInt(facilities),
                cleanliness: parseInt(cleanliness),
                comfort: parseInt(comfort),
                value_for_money: parseInt(value_for_money),
                location: parseInt(location),
                free_wifi: parseInt(free_wifi)
            },
            topics: topics.split(',').map(topic => topic.trim()),
            date: currentDate,
            text: text,
            score: score,
            reservation_id: reservation._id
        };

        console.log(newReview)

        const result = await reviewCollection.insertOne(newReview)
        console.log(result)

        res.redirect('/reviews')
    } catch(error){
        console.error("Error adding review:", error);
        res.status(500).send("Internal Server Error");
    }
})
export default router
```

Wyświetlenie informacji o rezerwacji

- użytkownik który złożył rezerwację może wyświetlić szczegółowe informacje na jej temat
- aby to zrobić musi podać unikalny token rezerwacji i adres email powiązany z tą rezerwacją
- jeśli token i email się zadza to zostana wyświetlone szczegółowe informacje
- jeśli stan rezerwacji jest `confirmed` to użytkownik ma opcję anulowania rezerwacji
- jeśli stan rezerwacji jest `pending` to użytkowniki ma opcję zapłacenia lub anulowania
- jeśli stan rezerwacji jest `canceled` to nie ma dodatkowych opcji

```
import express from "express"
import {MongoClient, ObjectId} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.post('/', async (req,res) => {
    const {token, email} = req.body

    console.log(token)

    try{
        await client.connect()
        const db = client.db("Hotel")
```

```
        const reservationCollection = db.collection('reservation')
        const guestCollection = db.collection('guest')

        const guest = await guestCollection.findOne({email})

        console.log(guest)

        if(!guest){
            return res.status(400).send("Guest with provided email not found")
        }

        const reservation = await reservationCollection.findOne({_id: new ObjectId(token)})

        console.log(reservation)

        if(!reservation){
            return res.status(400).send("Reservation with provided token not found")
        }

        const reservationGuests = reservation.guests.map(id => id.toString())

        if(!reservationGuests.includes(guest._id.toString())){
            return res.status(400).send("User is not assigned to this reservation")
        }

        const reservationFullInfo = await reservationCollection.aggregate([
            {
                $match:{
                    _id: new ObjectId(token)
                }
            },
            {
              '$lookup': {
                'from': 'guest',
                'localField': 'guests',
                'foreignField': '_id',
                'as': 'guests_info'
              }
            }, {
              '$lookup': {
                'from': 'room',
                'localField': 'rooms',
                'foreignField': 'room_number',
                'as': 'rooms_info'
              }
            }, {
              '$lookup': {
                'from': 'food',
                'localField': 'food_type',
                'foreignField': 'type',
                'as': 'food_info'
              }
            }, {
                '$lookup': {
                'from': 'review',
                'localField': '_id',
                'foreignField': 'reservation_id',
                'as': 'reviews'
              }
            }
          ]).toArray()


        console.log(reservationFullInfo[0])
        res.render('reservationDetails', {data: {reservationId: token, reservationFullInfo:
reservationFullInfo[0]}})
    } catch(error){
        console.error("Error showing reservation details:", error);
        res.status(500).send("Internal Server Error");
    }
})
```

```
export default router
```

- kiedy użytkownik zdecyduje się anulować rezerwację to sprawdzana jest data, można jeszcze anulować rezerwację
- podobnie jest przy próbie zapłacenia za rezerwację
- jeśli operacja się powiedzie to też zminiane jest pole `updated_at` oznaczające date i godzinę zmiany rezerwacji

```
import express from "express"
import {MongoClient, ObjectId} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.post('/', async (req,res) => {
    const {reservation, dateFrom, dateTo} = req.body

    console.log(req.body)

    const currDate = new Date().toISOString()

    if(currDate >= dateTo){
        res.send("It's no longer available to cancel this reservation")
    } else {
        try{
            await client.connect()
            const db = client.db("Hotel")
            const reservationCollection = db.collection('reservation')

            const result = await reservationCollection.updateOne(
                { _id: new ObjectId(reservation) },
                { $set: {status: 'canceled', updated_at: currDate}}
            )

            res.render("reservationUpdate", {data: {message: "Reservation canceled successfully"}})
        } catch(error){
            console.error("Error adding reservation:", error);
            res.status(500).send("Internal Server Error");
        }
    }
})

export default router
```

```
import express from "express"
import {MongoClient, ObjectId} from "mongodb"

const router = express.Router()

const client = new MongoClient('mongodb://127.0.0.1:27017')

router.post('/', async (req,res) => {
    const {reservation, dateFrom, dateTo} = req.body

    console.log(req.body)

    const currDate = new Date().toISOString()

    if(currDate >= dateTo){
        res.send("It's no longer available to pay for this reservation")
    } else {
        try{
            await client.connect()
            const db = client.db("Hotel")
            const reservationCollection = db.collection('reservation')

            const result = await reservationCollection.updateOne(
```

```
                { _id: new ObjectId(reservation) },
                { $set: {status: 'confirmed', updated_at: currDate}}
            )

            res.render("reservationUpdate", {data: {message: "Reservation payed successfully"}})
        } catch(error){
            console.error("Error adding reservation:", error);
            res.status(500).send("Internal Server Error");
        }
    }
})

export default router
```

Panel admina

- na pierwszej stronie panelu admina wyświetlane są informację o popularności: pokoi, typów wyżywienia i rezerwacji gości
- wyświetlane są również informację o ocenach pokoi, typów wyżywienia i gości na podstawie wszystkich recenzji
- wyświetlany jest również aktualny stan hotelu: zajęte pokoje dzisiejszego dnia oraz rezerwację, które mają zameldowanie lub wymeldowanie dzisiejszego dnia
- admin ma możliwość wyświetlenia wykazu dla podanego miesiąca, które wyświetli jakie rezerwację były w podanym miesiącu i sumaryczną kwotę z tych recenzji
- admin może modyfikować pokoje: wyświetlają się wszystkie, może poszczególny usunąć albo dodać nowy
- podobnie takie same opcje ma dla typów wyżywienia

```
router.get('/', async (req,res) => {
    const curr = new Date().toISOString()
    const today = curr.split('T')[0]
    console.log(today)
    try{
        await client.connect()
        const db = client.db('Hotel')
        const reservationCollection = db.collection('reservation')
        const reviewCollection = db.collection('review')

        const popularityAggregation = await reservationCollection.aggregate([
            { $match: { status: "confirmed" } },
            {
                $facet: {
                    roomAggregation: [
                        { $unwind: "$rooms" },
                        {
                            $group: {
                                _id: "$rooms",
                                count: { $sum: 1 }
                            }
                        },
                        { $sort: { count: -1 } },
                        { $limit: 5 },
                        {
                            $project: {
                                room_number: "$_id",
                                count: 1,
                                _id: 0
                            }
                        }
                    ],
                    foodAggregation: [
                        {
                            $group: {
                                _id: "$food_type",
                                count: { $sum: 1 }
                            }
                        },
                        { $sort: { count: -1 } },
                        { $limit: 5 },
                        {
                            $project: {
```

```
                        food_type: "$_id",
                        count: 1,
                        _id: 0
                    }
                }
            ],
            guestAggregation: [
                { $unwind: "$guests" },
                {
                    $group: {
                        _id: "$guests",
                        count: { $sum: 1 }
                    }
                },
                { $sort: { count: -1 } },
                { $limit: 5 },
                {
                    $lookup: {
                        from: "guest",
                        localField: "_id",
                        foreignField: "_id",
                        as: "guest_info"
                    }
                },
                {
                    $unwind: "$guest_info"
                },
                {
                    $project: {
                        guest_info: 1,
                        count: 1,
                        _id: 0
                    }
                }
            ]
        }
    }
]).toArray()

const roomRatings = await reviewCollection.aggregate([
    {
        $lookup: {
            from: 'reservation',
            localField: 'reservation_id',
            foreignField: '_id',
            as: 'reservation'
        }
    },
    { $unwind: "$reservation" },
    { $unwind: "$reservation.rooms" },
    {
        $group: {
            _id: "$reservation.rooms",
            average_score: { $avg: "$score" },
            review_count: { $sum: 1 }
        }
    },
    {
        $lookup: {
            from: 'room',
            localField: '_id',
            foreignField: 'room_number',
            as: 'room_details'
        }
    },
    { $unwind: "$room_details" },
    {
        $project: {
            room_number: "$_id",
            average_score: 1,
            review_count: 1,
            price: "$room_details.price",
```

```
            capacity: "$room_details.capacity",
            features: "$room_details.features"
        }
    }
]).toArray();

const topGuests = await reviewCollection.aggregate([
    {
        $group: {
            _id: "$guest_id",
            average_score: { $avg: "$score" },
            review_count: { $sum: 1 }
        }
    },
    { $sort: { average_score: -1, review_count: -1 } },
    { $limit: 5 },
    {
        $lookup: {
            from: 'guest',
            localField: '_id',
            foreignField: '_id',
            as: 'guest_details'
        }
    },
    { $unwind: "$guest_details" },
    {
        $project: {
            guest_id: "$_id",
            average_score: 1,
            review_count: 1,
            first_name: "$guest_details.first_name",
            last_name: "$guest_details.last_name",
            email: "$guest_details.email"
        }
    }
]).toArray();

const topFoodTypes = await reviewCollection.aggregate([
    {
      '$lookup': {
        'from': 'reservation',
        'localField': 'reservation_id',
        'foreignField': '_id',
        'as': 'reservation'
      }
    }, {
      '$unwind': '$reservation'
    }, {
      '$group': {
        '_id': '$reservation.food_type',
        'average_score': {
          '$avg': '$score'
        },
        'review_count': {
          '$sum': 1
        }
      }
    }, {
      '$project': {
        '_id': 0,
        'food_type': '$_id',
        'average_score': 1,
        'review_count': 1
      }
    }
  ]).toArray()


  const todayStatus = await reservationCollection.aggregate([
    {
        $match: {
            status: "confirmed",
```

```
                    $or: [
                        { check_in: { $lte: today }, check_out: { $gt: today } },
                        { check_in: today },
                        { check_out: today }
                    ]
                }
            },
            {
                $facet: {
                    occupiedRooms: [
                        { $match: { check_in: { $lte: today }, check_out: { $gt: today } } },
                        { $unwind: "$rooms" },
                        {
                            $group: {
                                _id: "$rooms",
                                guests: { $addToSet: "$guests" }
                            }
                        },
                        { $unwind: "$guests" },
                        { $unwind: "$guests" },
                        {
                            $lookup: {
                                from: "guest",
                                localField: "guests",
                                foreignField: "_id",
                                as: "guest_details"
                            }
                        },
                        { $unwind: "$guest_details" },
                        {
                            $group: {
                                _id: "$_id",
                                guests: { $addToSet: "$guest_details" }
                            }
                        }
                    ],
                    todayReservations: [
                        { $match: { $or: [{ check_in: today }, { check_out: today }] } },
                        {
                            $lookup: {
                                from: "guest",
                                localField: "guests",
                                foreignField: "_id",
                                as: "guest_details"
                            }
                        },
                        { $project: { _id: 1, check_in: 1, check_out: 1, guests: 1, rooms: 1, guest_details: 1 }
}
                    ],
                }
            }
        ]).toArray();


        const [popular] = popularityAggregation

        console.log(todayStatus[0].occupiedRooms[0])

        res.render('admin', {data: {popularRoom: popular.roomAggregation, popularFood: popular.foodAggregation,
popularGuest: popular.guestAggregation, roomRatings: roomRatings, topGuests: topGuests, topFoodTypes:
topFoodTypes, todayStatus: todayStatus[0]}})
    } catch(error){
        console.error("Error adding review:", error);
        res.status(500).send("Internal Server Error");
    }
})
```

-dodanie i usunięcie pokoju

```
router.post('/', async (req,res) => {
    const {room_number, price, capacity, area, include, features} = req.body

    console.log(req.body)

    const formattedInclude = include.split(',').map(item => {
        const [key, value] = item.split(' ');
        return { [key]: parseInt(value) };
    });
    const formattedFeatures = features.split(',')

    console.log(formattedFeatures,formattedInclude)

    try{
        await client.connect()
        const db = client.db("Hotel")
        const roomCollection = db.collection("room")

        const room = await roomCollection.findOne({room_number: parseInt(room_number)})

        if(!room){
            const newRoom = {
                room_number: parseInt(room_number),
                price: parseFloat(price),
                capacity: parseInt(capacity),
                area: parseInt(area),
                include: formattedInclude,
                features: formattedFeatures
            }

            await roomCollection.insertOne(newRoom)

            res.send('Room added successfully')
        } else {
            res.send('Room number already exists')
        }
    } catch(error){
        console.error("Error adding room:", error)
        res.status(500).send("Internal Server Error")
    }
})
```

```
router.post('/', async (req,res) => {
    const {room_number} = req.body

    console.log(room_number)

    await client.connect()
    const db = client.db("Hotel")
    const roomCollection = db.collection("room")

    const result = await roomCollection.deleteOne({room_number: parseInt(room_number)})

    if (result.deletedCount === 1) {
        res.send('Room deleted successfully')
    } else {
        res.send('Room not found')
    }
})
```

-dodanie i usunięcie typu wyżywienia

```
router.post('/', async (req,res) => {
    const {type, name, description, price} = req.body

    console.log(req.body)
```

```
    try {
        await client.connect()
        const db = client.db("Hotel")
        const foodCollection = db.collection("food")

        const food = await foodCollection.findOne({ type: type })

        if (!food) {
            const newFood = {
                name: name,
                description: description,
                type: type,
                price: parseFloat(price)
            }

            await foodCollection.insertOne(newFood)

            res.send('Food added successfully')
        } else {
            res.send('Food type already exists')
        }
    } catch (error) {
        console.error("Error adding food:", error)
        res.status(500).send("Internal Server Error")
    }
})
```

```
router.post('/', async (req,res) => {
    const {type} = req.body

    console.log(type)

    await client.connect()
    const db = client.db("Hotel")
    const foodCollection = db.collection("food")

    const result = await foodCollection.deleteOne({type: type})

    if (result.deletedCount === 1) {
        res.send('Food deleted successfully')
    } else {
        res.send('Food not found')
    }
})
```

Zadania zaplanowane

- przy użyciu modułu `node-cron` ustawiłem zadanie zaplanowane, które codzinne o północy sprawdzi wszystkie rezerwacje o statusie `pending` i jeżeli są takie których data zakwaterowania jest wcześniejsza niż dzisiejsza data to zamieni ich status na `canceled`. Dzięki temu mam automatyzację zwalniania zarezerwowanych pokoi, z rezerwacji które nie zostały opłacone

```
cron.schedule("0 0 * * *", () => {
    console.log('Running cron job to cancel pending reservations');
    cancelPendingReservations();
});
```

```
import { MongoClient } from "mongodb";

const client = new MongoClient('mongodb://127.0.0.1:27017')

async function cancelPendingReservations() {
```

```
    try {
        await client.connect()
        const db = client.db("Hotel")
        const reservationCollection = db.collection('reservation')

        const currentDate = new Date().toISOString();

        const result = await reservationCollection.updateMany(
            {
                status: "pending",
                check_in: { $lt: currentDate }
            },
            { $set: { status: "canceled" } }
        )

        console.log(`Updated ${result.modifiedCount} pending reservations to canceled.`);

    } catch (error) {
        console.error("Error canceling pending reservations:", error);
    }
}

export default cancelPendingReservations
```

Inne

- na stronie głównej wyświetlane są informacje pobrane z kolekcji `hotel`
- na podstronie o pokojach są wyświetlane pokoje z kolekcji `room`

```
router.get('/', async (req,res) => {
    await client.connect()
    const db = client.db("Hotel")
    const hotelCollection = db.collection("hotel")
    const info = await hotelCollection.findOne()

    console.log(info)

    res.render('home', {data: info})
})
```

```
router.get('/', async (req,res) => {
    await client.connect()
    const db = client.db("Hotel")
    const roomCollection = db.collection("room")
    const rooms = await roomCollection.find().toArray()

    console.log(rooms)

    res.render('rooms', {data: {rooms: rooms}})
})
```

Frontend

Do wyświetlania informacji na stronie skorzystałem ze schematów `pug` do których przekazuję niezbędne informację, które generowane są na stronie

- przykład strony panelu admina

```
doctype html
html(lang='pl')
    head
        meta(charset='utf-8')
        meta(name='viewport' content='width=device-width, initial-scale=1')
        title Tranquil Haven Hotel - Admin Panel
        link(href='https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css' rel='stylesheet'
integrity='sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH' crossorigin='anonymous')
```

```
        link(rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css')
        link(rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css')

        style.
            .roomCard{
                border: 1px solid black;
                padding: 0.2em;
                margin: 0.2em;
                max-width: 40em;
            }
    body
        nav(class='navbar navbar-expand-md bg-body-secondary')
            div(class='container-fluid')
                i(class='bi bi-buildings')
                a(class='navbar-brand me-auto fs-3' href='/') Tranquil Haven Hotel
                button(class='navbar-toggler' type='button' data-bs-toggle="collapse" data-bs-target="#navbarNav"
aria-controls="navbarNav" aria-expanded='false' aria-label="Toggle navigation")
                    span(class='navbar-toggler-icon')
                div(class='collapse navbar-collapse' id='navbarNav')
                    ul(class='navbar-nav ms-auto')
                        li(class='nav-item dropdown')
                            a(class='nav-link dropdown-toggle' href='#' role='button' data-bs-toggle='dropdown'
aria-expanded='false') Offer
                            ul(class='dropdown-menu')
                                li
                                    a(class='dropdown-item' href='/rooms') Rooms
                                    a(class='dropdown-item' href='/reviews') Reviews
                                    hr(class='dropdown-divider')
                                    a(class='dropdown-item' href='/reservationInfo') Your reservation
                                    hr(class='dropdown-divider')
                                    a(class='dropdown-item' href='/reservation') Reservation
                        li(class='nav-item')
                            a(class='dropdown-item' href='/admin') Admin Panel

        main
            br
            h1 Admin Panel:
            hr
            h2 Today hotel status:
            h3 Occupied rooms:
            ul
                each room in data.todayStatus.occupiedRooms
                    li Room number: #{room._id}
            hr
            h2 Today reservations that starts or ends:
            ul
                each reservation in data.todayStatus.todayReservations
                    li
                        h4 Check-in: #{reservation.check_in} Check-out: #{reservation.check_out}
                        h4 Rooms:
                        ul
                            each room in reservation.rooms
                                li #{room}
                        h4 Guests:
                        ol
                            each guest in reservation.guest_details
                                li #{guest.first_name} #{guest.last_name}, email: #{guest.email}, tel: #
{guest.phone}
                hr
            hr
            h2 Modify Rooms:
            form(method='get' action='/admin/room')
                input(type='submit' value='Show rooms')
            hr
            h2 Modify Foods:
            form(method='get' action='/admin/food')
                input(type='submit' value='Show foods')
            hr
            h2 Get Monthly reservation list:
            form(method='get' action='/admin/reservation')
                input(type='text' name='date' placeholder='YYYY-MM')
```

```
                input(type='submit' value='Get reservation list')
            hr
            h2 Popularity List:
            hr
            h3 Rooms:
            ol
                each room in data.popularRoom
                    li Room number: #{room.room_number}, count: #{room.count}
            h3 Food types:
            ol
                each food in data.popularFood
                    li Type: #{food.food_type}, count: #{food.count}
            h3 Guests:
            ol
                each guest in data.popularGuest
                    li #{guest.guest_info.first_name} #{guest.guest_info.last_name}, email: #
{guest.guest_info.email} | count: #{guest.count}
            hr
            h2 Score list:
            hr
            h3 Rooms:
            ol
                each room in data.roomRatings
                    li Room number: #{room.room_number}, review count: #{room.review_count}, average score: #
{room.average_score}

            h3 Food types:
            ol
                each food in data.topFoodTypes
                    li Type: #{food.food_type}, review count: #{food.review_count}, average score: #
{food.average_score}
            h3 Guests:
            ol
                each guest in data.topGuests
                    li #{guest.first_name} #{guest.last_name}, email: #{guest.email} | review count: #
{guest.review_count}, average score: #{guest.average_score}

        footer
            div(class='bg-body-secondary')
                div(class='row')
                    div(class='col-md-4')
                        p(class='text-center') dcebula@student.agh.edu.pl
                    div(class='col-md-4')
                    div(class='col-md-4')
                        p(class='text-center') Dariusz Cebula
                            i(class='bi bi-c-circle')


        script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity='sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz' crossorigin='anonymous')
```

- wygląd tej strony prezentuje się następująco

Tranquil Haven Hotel

Offer Admin Panel

- Rooms
- Reviews
- Your reservation
- Reservation

Admin Panel:

Today hotel status:

Occupied rooms:

- Room number: 104

Today reservations that starts or ends:

- Check-in: 2024-06-19 Check-out: 2024-06-22

  Rooms:

  - 104

  Guests:

  1. Dariusz Cebula, email: dcebula@student.agh.edu.pl, tel: +48111222333

Modify Rooms:

Show rooms

Modify Foods:

Show foods

Get Monthly reservation list:

YYYY-MM Get reservation list

Popularity List:

Rooms:

1. Room number: 103, count: 4
2. Room number: 101, count: 2
3. Room number: 104, count: 1

Food types:

1. Type: breakfast_only, count: 2
2. Type: all_inclusive, count: 2
3. Type: breakfast_dinner, count: 1
4. Type: full_board, count: 1

Guests:

1. John Doe, email: johndoe@example.com | count: 3
2. Alice Smith, email: alice.smith@example.com | count: 2

Reszta podstron została wykonana analogicznie