Bazy danych - Hotel Management Project

Wiktor Woźny i Remigiusz Kozicki

github link: https://github.com/remekozicki/hotel-management-project

Technologie:

1. Baza danych: mongodb
2. Backend: Node.js + Express.js
3. Frontend: React.js

Spis treści

Struktura bazy danych

Przy tworzeniu struktury bazy danych chcieliśmy jak najbardziej wykorzystać potencjał bazy MongoDB zagnieżdżając dokumenty (podejście embedding). Pozwoliło nam to na łatwiejsze zdefiniowanie relacji między niektórymi dokumentami. W strukturze pojawiają się jednak pojedyncze elementy podejścia tabelarycznego (referencing). W naszej bazie, aby opisać relację między rezerwacją a użytkownikiem, dodaliśmy pola typu reservation_id albo client_id. Pozwoliło nam to również na uporządkowanie dokumentów w bardziej przejrzysty sposób.

Kolekcje

1. rooms – kolekcja przechowująca dokumenty, które zawierają informacje na temat typów pokoi oraz samych pokoi występujących w hotelu:

   _id: ID typu pokoju
   type: słowna nazwa typu pokoju
   image: URL do obrazka przedstawiającego typ pokoju
   size: rozmiar pokoju
   price: cena za pokój za noc
   description: słowny opis typu pokoju
   rooms_array: tablica z dokumentami przedstawiającymi pokoje z danego typu:
   room_number: numer pokoju
   room_reservations: tabela rezerwacji danego pokoju:
   _id: ID rezerwacji
   dates: dokument z datami opisującymi przebieg rezerwacji:
   start_date: data rozpoczęcia rezerwacji
   end_date: data zakończenia rezerwacji
   status: status rezerwacji
   room_reviews: tablica z dokumentami przedstawiającymi opinie danego typu pokoju:
   client_id: ID użytkownika, który napisał opinie
   stars: wystawiona ocena (od 1 do 5)
   body: treść opinii


```
_id: ObjectId('628d0d2ece9567c2f36e01b7')
type: "Medium apartment"
image: "https://pixabay.com/get/g02af120e3a1ba6a58a52c8b83fc4b06cfbaa9cca23274…"
size: 3
price: 119
description: "Medium apartment, 1 bedroom with 3 beds, 1 bathroom and an amusing vie…"
▾ rooms_array: Array
  ▾ 0: Object
      room_number: 104
    ▾ room_reservations: Array
  ▾ 1: Object
      room_number: 105
    ▾ room_reservations: Array
      ▾ 0: Object
          _id: ObjectId('6475c575b6d66400cdac95a9')
        ▾ dates: Object
            start_date: "2023-05-21T05:33:17 -02:00"
            end_date: "2023-05-25T04:19:27 -02:00"
          status: "accepted"
  ▾ 2: Object
      room_number: 106
    ▾ room_reservations: Array
▾ room_reviews: Array
  ▾ 0: Object
      client_id: ObjectId('6475bca0dda1e9eb6a22e693')
      stars: 4
      body: "consequat aute nostrud officia irure deserunt ex anim culpa ipsum sunt…"
  ▾ 1: Object
      client_id: ObjectId('644fd98545d4d5cfb71e6d21')
      stars: 3.2
      body: "irure amet ea aliqua mollit consectetur fugiat duis amet nisi laborum …"
  ▾ 2: Object
      _id: "6475bca079d85f87d98888dd"
      client_id: ObjectId('644fd9856dd4d86f458e9a5a')
      stars: 4
      body: "consequat aute nostrud officia irure deserunt ex anim culpa ipsum sunt…"
```

2. users – kolekcja przechowująca dokumenty, które zawierają informacje na temat użytkowników:

   _id: ID użytkownika
   firstname: imię
   lastname: nazwisko
   email: email
   phone: numer telefonu
   address: obiekt przechowujący dane adresowe użytkownika:
      city: miasto
      street: ulica
      building_number: numer budynku
   registered: data zajerestrowania użytkownika
   reservations_history: tabela przechowująca ID rezerwacji wykonanych przez użytkownika:
      reservation_id: ID rezerwacji
      room_number: numer pokoju zarezerwowanego
      dates: dokument z datami opisującymi przebieg rezerwacji:
         start_date: data rozpoczęcia rezerwacji
         end_date: data zakończenia rezerwacji
      status: status rezerwacji
   role: rola użytkownika (na ten moment client albo employee)


```
_id: ObjectId('644fd98545d4d5cfb71e6d21')
firstname: "Thomas"
lastname: "Wilkerson"
email: "thomaswilkerson@zaphire.com"
phone: "+48 (988) 443-2465"
▾ address: Object
    city: "Leola"
    street: "446 Canal Avenue"
    building_numer: "4461"
registered: "2017-02-26T01:26:49 -01:00"
▾ reservations_hitory: Array
  ▾ 0: Object
      reservation_id: ObjectId('628d101398e88560beb96210')
      room_number: 205
    ▾ dates: Object
        start_date: "2023-09-04T05:33:17 -02:00"
        end_date: "2023-09-06T04:19:27 -02:00"
      status: "pending"
  ▾ 1: Object
      reservation_id: ObjectId('628d101398e88560beb96209')
      room_number: 101
    ▾ dates: Object
        start_date: "2023-05-04T05:33:17 -02:00"
        end_date: "2023-05-06T04:19:27 -02:00"
      status: "pending"
role: "client"
```

Zapytania

Po stworzeniu struktury zaimplementowaliśmy zapytania, których potem używaliśmy przy backendzie (lub nie):

1. **zapytanie zwraca listę z numerami pokojów o danym typie, które są dostępne w danym okresie czasowym (dwie wersje):**

```
db.rooms.aggregate([
    {$unwind: "$rooms_array"},
    {$unwind: {
        path: "$rooms_array.room_reservations",
        preserveNullAndEmptyArrays: true
    }},
    {$project:{
        _id: 0,
        room_type: "$type",
        room_number: "$rooms_array.room_number",
        description: "$description",
        size: "$size",
        price: "$price",
        reservation_status: "$rooms_array.room_reservations.status",
        start_date: {
            $toDate: "$rooms_array.room_reservations.dates.start_date"
        },

        end_date: {
            $toDate: "$rooms_array.room_reservations.dates.end_date"
        }
    }},
    {$match:{
        $and:[
            {$or:[
                {$nor:[
                    {$and:[
                        {"start_date":{$gte: ISODate(input_start_date)}},
                        {"start_date":{$gte: ISODate(input_start_date)}},
                        {"end_date": {$lte: ISODate(input_end_date)}},
                    ]},
                    {$and:[
                        {"start_date":{$lte: ISODate(input_start_date)}},
                        {"end_date": {$gte: ISODate(input_start_date)}},
                    ]},
                    {$and:[
                        {"start_date":{$lte: ISODate(input_end_date)}},
                        {"end_date": {$gte: ISODate(input_end_date)}},
                    ]},
                ]},
                {"reservation_status": {$eq:"cancelled"}}

            ]},
            {"room_type": input_type},
        ]
    }},
    {$project:{
        start_date: 0,
        end_date: 0,

    }},
    {$group:{
        _id: "$room_number"
    }}
])
```

	{} _id
1	102
2	103
3	101

```
db.rooms.aggregate([
  {
    $match: {
      type: input_type
    }
  },
  {
    $unwind: "$rooms_array"
  },
  {
    $addFields: {
      "rooms_array.available": {
        $not: {
          $anyElementTrue: {
            $map: {
              input: "$rooms_array.room_reservations",
              as: "reservation",
              in: {
                $and: [
                  { $lte: [input_start_date, "$$reservation.dates.end_date"] },
                  { $gte: [input_end_date, "$$reservation.dates.start_date"] }
                ]
              }
            }
          }
        }
      }
    }
  },
  {
    $match: {
      "rooms_array.available": true
    }
  },
  {
    $group: {
      _id: null,
      room_numbers: {
        $push: "$rooms_array.room_number"
      }
    }
  },
  {
    $project: {
      _id: 0,
      room_numbers: 1
    }
  }
])
```

	{} room_numbers
1	[new NumberInt("101")]

1) $match - wyszukujemy pokoje danego typu

2) $unwind - rozpakuwywujemy tablice rooms_array

3) $addFields - tworzymy tablice "available"

$not zaprzecza wynik

$anyElementTrue (false - zarezerwowany, true - wolny)

$anyElementTrue ewaluuje tablice z pokojami na true albo false z zależności od warunku (true - zarezerwowany, false - wolny)

$map dla każdego elementu z tablicy rezerwacji sprawdza warunek z $in i zwraca tablice z wynikami

4) wyszukujemy w tablicy wyników wolne pokoje (false - zarezerwowany, true - wolny)

5)$group grupujemy po nr pokoi i dodajemy je do tablicy

2. **zapytanie zwraca wszystkie rezerwacje w podanym przedziale czasowym i o podanym statusie:**

```
db.rooms.aggregate([
    {$unwind: "$rooms_array"},
    {$unwind:"$rooms_array.room_reservations"},
    {$project:{
        _id: 0,

        reswervation_id: "$rooms_array.room_reservations._id",
        reservation_status: "$rooms_array.room_reservations.status",
        start_reservation: "$rooms_array.room_reservations.dates.start_date",
        end_reservation: "$rooms_array.room_reservations.dates.end_date",
        start_date: {
            $toDate: "$rooms_array.room_reservations.dates.start_date"
        },

        end_date: {
            $toDate: "$rooms_array.room_reservations.dates.end_date"
        }

    }},
    {$match:{
        $and:[
            {"start_date":{$gte: ISODate(input_start_date)}},
            {"end_date": {$lte: ISODate(input_end_date)}},
            {"reservation_status": {$eq:input_status}}
        ]
    }}
])
```

	end_date	end_reservation	reservation_status	reswervation_id	start_date	start_reservation
1	2023-05-25T06:19:27.000Z	2023-05-25T04:19:27 -02:00	accepted	6475c575b6d66400cdac95a9	2023-05-21T07:33:17.000Z	2023-05-21T05:33:17 -02:00
2	2023-05-27T06:19:27.000Z	2023-05-27T04:19:27 -02:00	accepted	6475c5753a048dedd8474d92	2023-05-21T07:33:17.000Z	2023-05-21T05:33:17 -02:00

3. **zapytanie zwraca dane wszystkich klientów:**

```
db.users.find(
    {
        role: "client"
    },
    {
        firstname: 1,
        lastname: 1,
        role: 1
    }

)
```

	_id	firstname	lastname	role
1	644fd98545d4d5cfb71e6d21	Thomas	Wilkerson	client
2	644fd985e33fc99144f92d1f	Desiree	Erickson	client
3	644fd985f75613ee8115a2d7	Arnold	Williamson	client
4	644fd9855a807472100a464b	Denise	Villarreal	client
5	644fd985a0d8afb57cf4a7d3	Jackie	Murray	client
6	644fd98513b89455bfa12a5b	Mendez	Hartman	client
7	644fd9856575a2033b354196	Lilly	Strong	client
8	644fd9856dd4d86f458e9a5a	Florine	Spence	client

4. **zapytanie zwraca dane wszystkich pracowników:**

```
db.users.find(
{
        role: "employee"
    },
    {
        firstname: 1,
        lastname: 1,
        role: 1
    }
)
```

	_id	firstname	lastname	role
1	644fd9e06be09ca9ff6c6661	Florence	Roth	employee
2	644fd9e046d1d7fa4f17e8ad	Myra	Baker	employee
3	644fd9e0b98e1df9a38143d1	Frank	Griffith	employee

5. **zapytanie zwraca typy pokojów, które mają średnią ocenę wyższą niż zadana (dla wywołania z przykładu - 3.5):**

```
db.rooms.aggregate([
    {$unwind: "$room_reviews"},
    {$group:{
        _id: "$type",
        avg_stars: {$avg: "$room_reviews.stars"}
    }},
    {$match:{
        avg_stars: {$gte: input_avg_stars}
    }},
    {$project:{
        _id: 0,
        type: "$_id",
        avg_stars: "$avg_stars"
    }}
])
```

	{} avg_stars	{} type
1	3.733333333333333	Medium apartment
2	3.6	Double room with balcony
3	4.666666666666667	Kingsize premium apartment

6. **zapytanie zwraca informacje o rezerwacjach dla konkretnego użytkownika:**

```
let input_clientid = "644fd98545d4d5cfb71e6d21";

db.users.aggregate([
  { $match: { "_id": ObjectId(input_clientid) } },
  { $unwind: "$reservations_hitory" },
  {
    $project: {
      "_id": 0,
      "reservation_id": "$reservations_hitory.reservation_id",
      "room_number": "$reservations_hitory.room_number",
      "start_date": "$reservations_hitory.dates.start_date",
      "end_date": "$reservations_hitory.dates.end_date",
      "status": "$reservations_hitory.status"
    }
  }
])
```

	{} end_date	{} reservation_id	{} room_number	{} start_date	{} status
1	2023-09-06T04:19:27 -02:00	628d101398e88560beb96210	205	2023-09-04T05:33:17 -02:00	pending
2	2023-05-06T04:19:27 -02:00	628d101398e88560beb96209	101	2023-05-04T05:33:17 -02:00	pending

7. **zapytania dodają nową rezerwację do bazy danych (do odpowiedniej tablicy room_reservations w kolekcji rooms oraz do odpowiedniej tablicy reservations_history w kolekcji users):**

```
const roomTypeId = ObjectId("628d0d2e22b44fb4124bb51d");
const reservation = {
  _id: ObjectId(),
  client_id: ObjectId("644fd98545d4d5cfb71e6d21"),
  dates: {
    start_date: "2023-06-01T00:00:00Z",
    end_date: "2023-06-05T00:00:00Z"
  },
  status: "pending"
};

db.rooms.updateOne(
  { "_id": roomTypeId },
  { $push: { "rooms_array.$[room].room_reservations": reservation } },
  { arrayFilters: [{ "room.room_number": 101 }] }
);
```

```
▾ rooms_array: Array
  ▾ 0: Object
      room_number: 101
    ▾ room_reservations: Array
      ▸ 0: Object
      ▸ 1: Object
      ▾ 2: Object
          _id: ObjectId('64870d540000000000d1fef9')
          client_id: ObjectId('644fd98545d4d5cfb71e6d21')
        ▸ dates: Object
          status: "pending"
```

```
const userId = ObjectId("644fd98545d4d5cfb71e6d21");
const reservation = {
  reservation_id: ObjectId("64870d540000000000d1fef9"),
  room_number: roomNumber,
  dates: {
    start_date: "2023-06-01T00:00:00Z",
    end_date: "2023-06-05T00:00:00Z"
  },
  status: "pending"
};

db.users.updateOne(
  { "_id": userId },
  { $push: { "reservations_hitory": reservation } }
);
```

```
▾ reservations_hitory: Array
  ▸ 0: Object
  ▸ 1: Object
  ▾ 2: Object
      reservation_id: ObjectId('64870d540000000000d1fef9')
      room_number: 101
    ▾ dates: Object
        start_date: "2023-06-01T00:00:00Z"
        end_date: "2023-06-05T00:00:00Z"
      status: "pending"
```

8. **zapytanie dodaje nową recenzję do typu pokoju:**

```
const roomTypeId = ObjectId("628d0d2e9b4f386adf3c679c");
const review = {
  client_id: ObjectId("644fd985d84f95c6777a9d53"),
  stars: 4.5,
  body: "osenfoanoc pwpidcmapjdcpa poajdpoa fdpoak dk apdcmapmcaanfcosa"
};

db.rooms.updateOne(
  { "_id": roomId },
  { $push: { "room_reviews": review } }
);
```

```
▸ rooms_array: Array
▾ room_reviews: Array
  ▸ 0: Object
  ▸ 1: Object
  ▸ 2: Object
  ▾ 3: Object
      client_id: ObjectId('644fd985d84f95c6777a9d53')
      stars: 4.5
      body: "osenfoanoc pwpidcmapjdcpa poajdpoa fdpoak dk apdcmapmcaanfcosa"
```

9. **zapytania zmieniają status rezerwacji o określonym id:**

```
db.rooms.updateOne(
  {
    "rooms_array.room_reservations._id": ObjectId("628d101398e88560beb96209")
  },
  {
  $set:
    {
        "rooms_array.$[outer].room_reservations.$[inner].status": "accepted"
    }
  },
  {
  arrayFilters: [
        { "outer.room_reservations._id": ObjectId("628d101398e88560beb96209") },
        { "inner._id": ObjectId("628d101398e88560beb96209") }
    ]
  }
);
```

```
  room_number: 101
▾ room_reservations: Array
  ▾ 0: Object
      _id: ObjectId('628d101398e88560beb96209')
      client_id: ObjectId('644fd98545d4d5cfb71e6d21')
    ▸ dates: Object
      status: "accepted"
  ▸ 1: Object
```

```
db.users.updateOne(
  { "reservations_hitory.reservation_id": ObjectId("628d101398e88560beb96209") },
  { $set: { "reservations_hitory.$.status": "accepted" } }
);
```

```
▾ reservations_hitory: Array
  ▸ 0: Object
  ▾ 1: Object
      reservation_id: ObjectId('628d101398e88560beb96209')
      room_number: 101
    ▸ dates: Object
      status: "accepted"
```

10. zapytanie dodaje nowego użytkownika do bazy:

```
const firstname_input = "John";
const last_name = "Doe";
const email_input = "johndoe@example.com";
const phone_input = "+1 (123) 456-7890";
const address_input = {
    "city": "New York",
    "street": "123 Main Street",
    "building_numer": 1234
}
const role_input = "client";

db.users.insertOne({
  "firstname": firstname_input,
  "lastname": last_name,
  "email": email_input,
  "phone": phone_input,
  "address": address_input,
  "registered": Date(),
  "reservations_hitory": [],
  "role": role_input
})
```

```
_id: ObjectId('648887f1ce75e168e24b450e')
firstname: "John"
lastname: "Doe"
email: "johndoe@example.com"
phone: "+1 (123) 456-7890"
▾ address: Object
    city: "New York"
    street: "123 Main Street"
    building_numer: 1234
registered: "Tue Jun 13 15:14:57 UTC 2023"
▾ reservations_hitory: Array
role: "client"
```

11. zapytanie zwraca najczęściej rezerwowany typ pokoju:

```
db.rooms.aggregate([
  {$unwind: "$rooms_array"},
  {$unwind: "$rooms_array.room_reservations"},
  {$match: {
      "rooms_array.room_reservations.status": "accepted"
  }},
  {$group: {
      _id: "$type",
      count: { $sum: 1 }
  }},
  {$sort:{
    count: -1
  }}
])
```

	{} _id	{} count
2	Medium apartment	1

Back-end – serwer express.js

1. Połączenie z bazą:

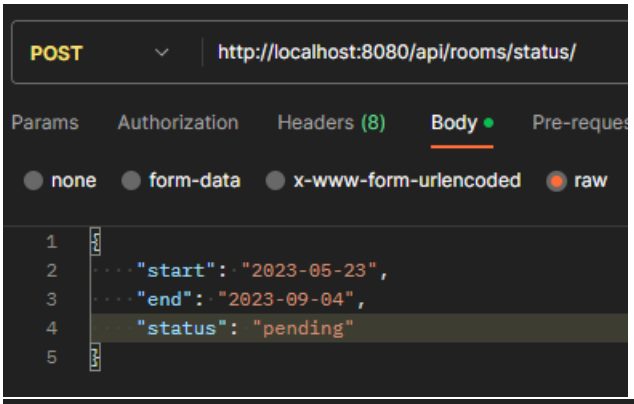
Poniższy kod konfiguruje podstawowe elementy serwer Express oraz nawiązuje połączenie z bazą danych MongoDB:

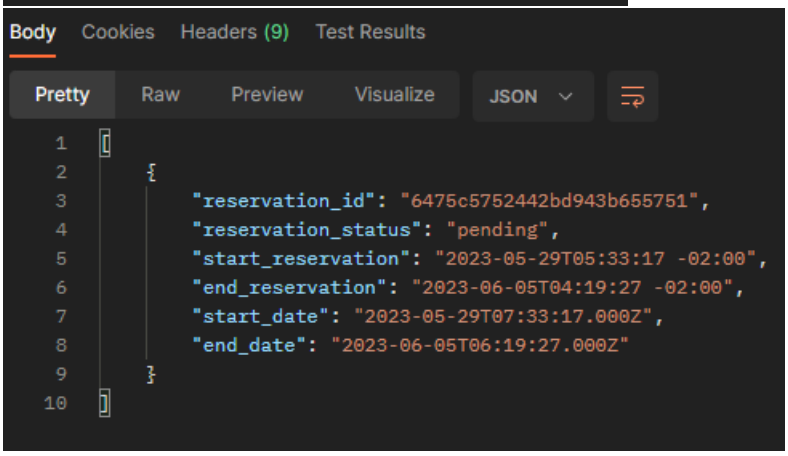
```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const app = express();

const db = require("./app/models");

	db.mongoose
	    .connect(db.url, {
	        useNewUrlParser: true,
	        useUnifiedTopology: true
	    })
	    .then(() => {
	        console.log("Connected to the database!");
	    })
	    .catch(err => {
	        console.log("Cannot connect to the database!", err);
	        process.exit();
	    });

	let corsOptions = {
	    origin: "http://localhost:8081"
	};

	app.use(cors(corsOptions));

	const PORT = process.env.PORT || 8080;
	app.listen(PORT, () => {
	    console.log(`Server is running on port ${PORT}.`);
	});
```

2. Żądania do bazy pobierające odpowiednie dane (metody api) wraz ze screenami potwierdzającymi działanie z Postmana:

a) Żądanie pobiera informacje o rezerwacjach o danym statusie, które zawierają się w podanym przedziale czasowym:

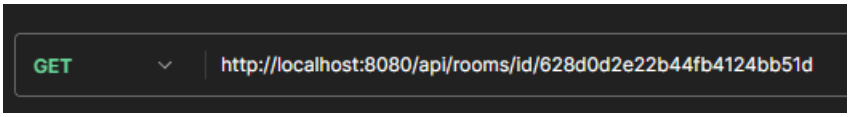
```
exports.getWithStatus = (req, res) => {
    const input_status = req.body.status;
    const input_start_date = new Date(req.body.start);
    const input_end_date = new Date(req.body.end);

    Room.aggregate([
        {$unwind: "$rooms_array"},
        {$unwind: "$rooms_array.room_reservations"},
        {
            $project: {
                _id: 0,
                reservation_id: "$rooms_array.room_reservations._id",
                reservation_status: "$rooms_array.room_reservations.status",
                start_reservation: "$rooms_array.room_reservations.dates.start_date",
                end_reservation: "$rooms_array.room_reservations.dates.end_date",
                start_date: {
                    $toDate: "$rooms_array.room_reservations.dates.start_date"
                },

                end_date: {
                    $toDate: "$rooms_array.room_reservations.dates.end_date"
                }

            }
        },
        {
            $match: {
                $and: [
                    {"start_date": {$gte: input_start_date}},
                    {"end_date": {$lte: input_end_date}},
                    {"reservation_status": {$eq: input_status}}
                ]
            }
        }
    ])
        .then(data => {
            if (!data) {
                res.status(404).send({
                    message: "Not found room types with such status " + input_status
                })
            } else {
                res.send(data);
            }
        })
        .catch(err => {
            res.status(500).send({
                message: "Error retrieving room types with status " + input_status
            })
        })
}
```





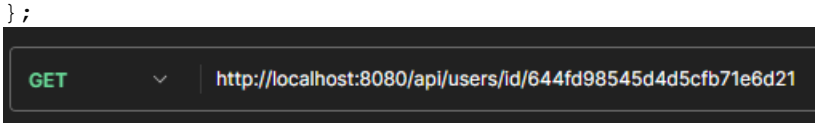
b) Żądanie pobiera informacje na temat danego typu pokoju:

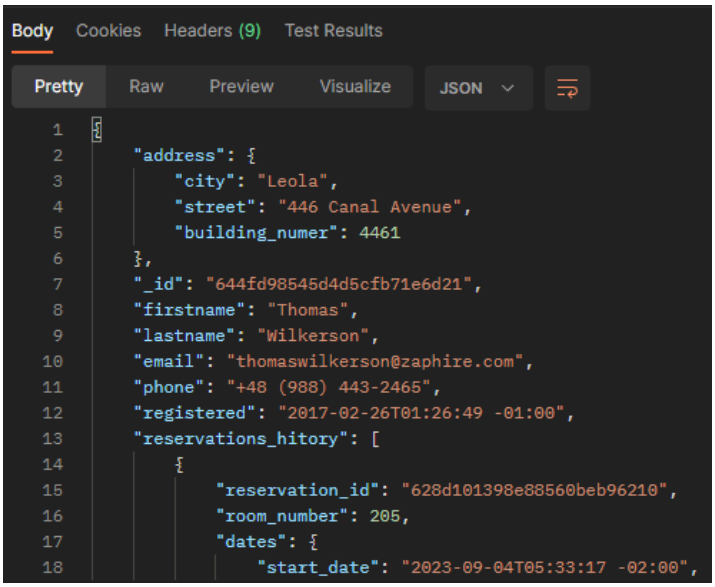
```
exports.getById = (req, res) => {
    const id = req.params.id;

    Room.findById(id)
        .then(data => {
            res.send(data);
        })
        .catch(err => {
            res.status(500).send({
                message:
                    err.message || "Some error occurred while retrieving room type."
            })
        })
};
```



c) Żądanie pobiera informacje na temat danego użytkownika:

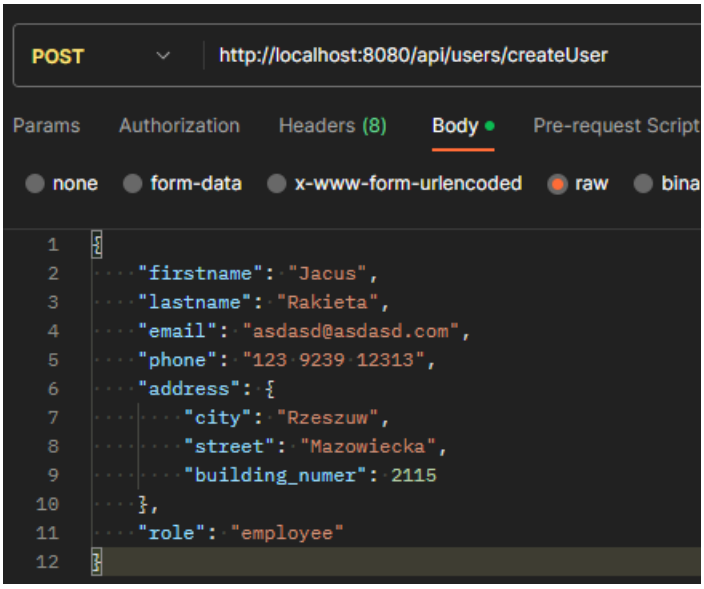
```
exports.getById = (req, res) => {
    const id = req.params.id;

    User.findById(id)
        .then(data => {
            res.send(data);
        })
        .catch(err => {
            res.status(500).send({
                message:
                    err.message || "Some error occurred while retrieving user."
            })
        })
};
```

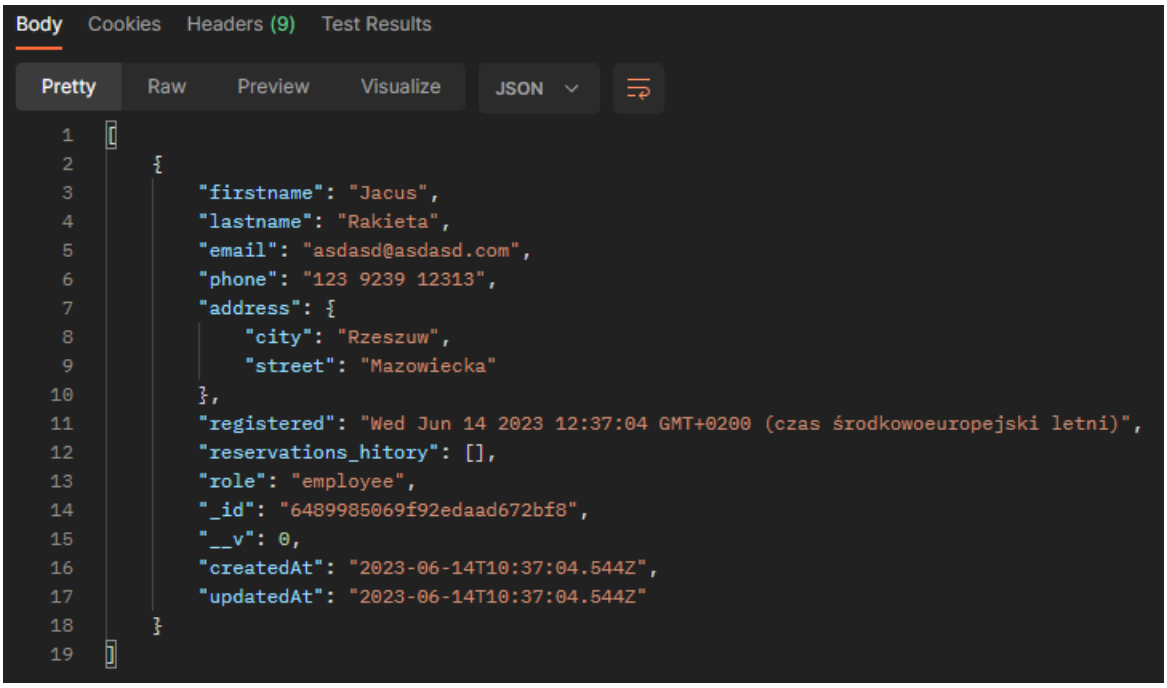




d) Żądanie dodaje nowego użytkownika:

```
exports.createNewUser = (req, res) => {

    User.insertMany({
        firstname: req.body.firstname,
        lastname: req.body.lastname,
        email: req.body.email,
        phone: req.body.phone,
        address: req.body.address,
        registered: new Date(),
        reservations_hitory: [],
        role: req.body.role
    })
    .then(data => {
        res.send(data);
    })
    .catch(err => {
        res.status(500).send({
            message:
                err.message || "Some error occurred while creating the reservation."
        });
    });

}
```


POST
http://localhost:8080/api/users/createUser
Params
Authorization
Headers (8)
Body
Pre-request Script
none
form-data
x-www-form-urlencoded
raw
bina
{
"firstname": "Jacus",
"lastname": "Rakieta",
"email": "asdasd@asdasd.com",
"phone": "123 9239 12313",
"address": {
"city": "Rzeszuw",
"street": "Mazowiecka",
"building_numer": 2115
},
"role": "employee"
}


Body
Cookies
Headers (9)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"firstname": "Jacus",
"lastname": "Rakieta",
"email": "asdasd@asdasd.com",
"phone": "123 9239 12313",
"address": {
"city": "Rzeszuw",
"street": "Mazowiecka"
},
"registered": "Wed Jun 14 2023 12:37:04 GMT+0200 (czas środkowoeuropejski letni)",
"reservations_hitory": [],
"role": "employee",
"_id": "6489985069f92edaad672bf8",
"__v": 0,
"createdAt": "2023-06-14T10:37:04.544Z",
"updatedAt": "2023-06-14T10:37:04.544Z"
}
]

e) Żądanie pobiera numery pokojów o danym typie, które są wolne w danym przedziale czasowym (na dwa sposoby, bo jeden nie działał na początku):

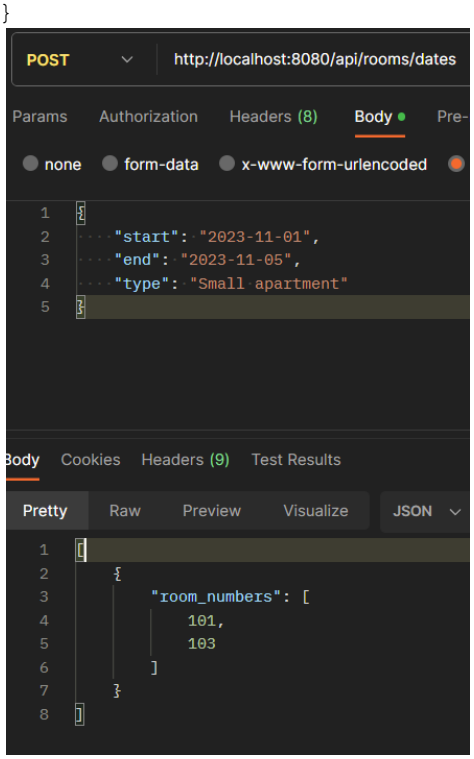
```
exports.getWithAvailableDateAndRoomType = (req, res) => {
    const input_start_date = new Date(req.body.start);
    const input_end_date = new Date(req.body.end);
    const input_type = req.body.type

    Room.aggregate([
        {$unwind: "$rooms_array"},
        {
            $unwind: {
                path: "$rooms_array.room_reservations",
                preserveNullAndEmptyArrays: true
            }
        },
        {
            $project: {

                _id: 0,
                room_type: "$type",
                room_number: "$rooms_array.room_number",
                description: "$description",
                size: "$size",
                price: "$price",
                reservation_status: "$rooms_array.room_reservations.status",
                start_date: {
                    $toDate: "$rooms_array.room_reservations.dates.start_date"
                },

                end_date: {
                    $toDate: "$rooms_array.room_reservations.dates.end_date"
                }

            }
        },
        {
            $match: {

                $and: [
                    {
                        $or: [
                            {
                                $nor: [
                                    {$and:[
                                            {"start_date":{$gte: input_start_date}},
                                            {"end_date": {$lte: input_end_date}},
                                        ]},
                                    {$and:[
                                            {"start_date":{$lte: input_start_date}},
                                            {"end_date": {$gte: input_start_date}},
                                        ]},
                                    {$and:[
                                            {"start_date":{$lte: input_end_date}},
                                            {"end_date": {$gte: input_end_date}},
                                        ]},
                                ]
                            },
                            {"reservation_status": {$eq: "cancelled"}}

                        ]
                    },

                    {"room_type": input_type},

                ]
            }
```

```
        },

        {
            $project: {
                start_date: 0,
                end_date: 0,

            }
        },
        {
            $group: {

                _id: "$room_number"
            }
        }

    ])
        .then(data => {
            if (!data) {
                res.status(404).send({
                    message: "Not found room types with such filters"
                })
            } else {
                res.send(data);
            }
        })
        .catch(err => {
            res.status(500).send({
                message: "Error retrieving room type with such filters"
            })
        })
}
```

Sposób ten okazał się niepoprawny, ponieważ niepotrzebnie przy przypisywaniu do zmiennych input_start_date oraz input_end_date tworzyliśmy obiekty Date za pomocą new Date. Poniżej poprawiona i działająca wersja (z użyciem troszkę innego zapytania):

```
exports.getWithAvailableDateAndRoomType = (req, res) => {
    const input_start_date = req.body.start;
    const input_end_date = req.body.end;
    const input_type = req.body.type

    Room.aggregate([
        {
            $match: {
                type: input_type
            }
        },
        {
            $unwind: "$rooms_array"
        },
        {
            $addFields: {
                "rooms_array.available": {
                    $not: {
                        $anyElementTrue: {
                            $map: {
                                input: "$rooms_array.room_reservations",
                                as: "reservation",
                                in: {
                                    $and: [
                                        { $lte: [input_start_date,
"$$reservation.dates.end_date"] },
                                        { $gte: [input_end_date,
"$$reservation.dates.start_date"] }
                                    ]
                                }
```

```
                                }
                            }
                        }
                    }
                }
            },
            {
                $match: {
                    "rooms_array.available": true
                }
            },
            {
                $group: {
                    _id: null,
                    room_numbers: {
                        $push: "$rooms_array.room_number"
                    }
                }
            },
            {
                $project: {
                    _id: 0,
                    room_numbers: 1
                }
            }
        ])
            .then(data => {
                if (!data) {
                    res.status(404).send({
                        message: "Not found room types with such filters"
                    })
                } else {
                    res.send(data);
                }
            })
            .catch(err => {
                res.status(500).send({
                    message: "Error retrieving room type with such filters"
                })
            })
}
```



3. Widok plików rooms.router.js oraz users.router.js:

Reszta żądań ma podobną strukturę (każde ma wykorzystane zapytanie mongodb przedstawione wcześniej w sprawozdaniu), więc dla zobrazowania ilości żądań pokażemy tylko wspomniane wyżej pliki:

users.routes.js

```
module.exports = app => {

    const user = require("../controllers/users.controller");

    let router = require("express").Router();

    router.get( path: "/", user.getAll);
    router.get( path: "/id/:id", user.getById);
    router.get( path: "/clients", user.getClients);
    router.get( path: "/employees", user.getEmployees);
    router.get( path: "/reservations/:id", user.getReservationsByUserId);

    // router.post("/addReservation", user.addReservationToUser);
    router.post( path: "/createUser", user.createNewUser);
    router.post( path: "/changeStatus", user.changeReservationStatus);

    router.delete( path: "/id/:id", user.delete);
    router.delete( path: "/", user.deleteAll);

    app.use('/api/users', router);

}
```

rooms.routes.js

```
const room = require("../controllers/rooms.controller");

module.exports = app => {
    const room = require("../controllers/rooms.controller.js");

    let router = require("express").Router();

    router.get( path: "/", room.getAll);
    router.get( path: "/id/:id", room.getById)

    router.post( path: "/stars", room.getWithAvgStars);
    router.post( path: "/dates", room.getWithAvailableDateAndRoomType);
    router.post( path: "/status",room.getWithStatus);
    router.post( path: "/addReservation", room.addReservation);
    router.post( path: "/addReview", room.addReview);
    router.post( path: "/changeStatus", room.changeReservationStatus);
    // router.post("/addReservation", room.addReservationToRooms);

    router.delete( path: "/id/:id", room.delete);
    router.delete( path: "/", room.deleteAll);

    app.use('/api/rooms', router);
}
```