

ZTBD - sprawozdanie z projektu

2025-05-24

Podstawowe informacje

Autorzy projektu:	Mikołaj Kaczmarek (124942) Piotr Wilma (124832)
Temat:	Hotel
Wybrany zakres (ocena):	5,0
Baza danych:	ztbd124942

1 Wprowadzenie

Projekt obejmuje stworzenie bazy danych MongoDB dla hotelu. System zarządzania hotelem wymaga przechowywania danych o gościach, pokojach, rezerwacjach, pracownikach oraz usługach dodatkowych. Opracowana baza danych umożliwi śledzenie aktualnego stanu rezerwacji, zarządzanie dostępnością pokoi oraz realizację dodatkowych zamówień gości hotelowych. Rozwiązanie takie znajduje zastosowanie w każdym obiekcie hotelowym potrzebującym wszechstronnego systemu do zarządzania codzienną działalnością.

Główne funkcjonalności realizowane przez bazę danych obejmują:

- Przechowywanie informacji o gościach wraz z ich danymi kontaktowymi
- Zarządzanie pokojami hotelowymi i ich atrybutami
- Obsługę rezerwacji z informacją o terminach pobytu i statusie płatności
- Ewidencję pracowników hotelu z przypisaniem do działów
- Katalog usług dodatkowych oraz zamówień na te usługi

2 Schemat bazy - kolekcje

Opracowana baza danych składa się z 6 kolekcji, łącznie zawierających 70 dokumentów. Poniżej przedstawiono szczegółowy opis każdej kolekcji wraz z przykładowymi dokumentami.

2.1 Kolekcja: guests

Opis: Przechowuje dane gości hotelowych wraz z informacjami kontaktowymi i dodatkowymi atrybutami.

Liczba dokumentów: 15

Struktura dokumentu:

```
1 {  
2   _id: ObjectId("..."),  
3   firstName: String,  
4   lastName: String,  
5   email: String,  
6   phoneNumber: String,  
7   address: {  
8     street: String,
```

```

9      city: String,
10     postalCode: String,
11     country: String
12 },
13     dateOfBirth: Date,
14     loyaltyPoints: Number,
15     registrationDate: Date
16 }

```

Przykładowy dokument:

```

1  {
2    _id: ObjectId("..."),
3    firstName: "Jan",
4    lastName: "Kowalski",
5    email: "jan.kowalski@email.com",
6    phoneNumber: "+48123456789",
7    address: {
8      street: "Kwiatowa 12",
9      city: "Warszawa",
10     postalCode: "01-234",
11     country: "Polska"
12 },
13     dateOfBirth: ISODate("1985-05-15T00:00:00Z"),
14     loyaltyPoints: 120,
15     registrationDate: ISODate("2020-03-10T00:00:00Z")
16 }

```

2.2 Kolekcja: rooms

Opis: Zawiera informacje o pokojach hotelowych, ich typach, wyposażeniu i statusie dostępności.

Liczba dokumentów: 12

Struktura dokumentu:

```

1  {
2    _id: ObjectId("..."),
3    roomNumber: String,
4    floor: Number,
5    type: String,
6    capacity: Number,
7    pricePerNight: Number,
8    amenities: [String],
9    status: String,
10    lastRenovation: Date
11 }

```

Przykładowy dokument:

```

1  {

```

```

2   _id: ObjectId("..."),
3   roomNumber: "301",
4   floor: 3,
5   type: "Suite",
6   capacity: 4,
7   pricePerNight: 600.00,
8   amenities: ["TV", "WiFi", "Minibar", "Klimatyzacja",
9               "Sejf", "Ekspres do kawy", "Jacuzzi"],
10  status: "Dostępny",
11  lastRenovation: ISODate("2022-02-15T00:00:00Z")
12 }

```

2.3 Kolekcja: reservations

Opis: Przechowuje informacje o rezerwacjach gości, zawiera referencje do kolekcji guests i rooms.

Liczba dokumentów: 10

Struktura dokumentu:

```

1  {
2    _id: ObjectId("..."),
3    guestId: ObjectId("..."),
4    roomId: ObjectId("..."),
5    checkInDate: Date,
6    checkOutDate: Date,
7    numberOfGuests: Number,
8    totalPrice: Number,
9    paymentStatus: String,
10   specialRequests: [String],
11   bookingDate: Date,
12   status: String
13 }

```

Przykładowy dokument:

```

1  {
2    _id: ObjectId("..."),
3    guestId: ObjectId("..."),
4    roomId: ObjectId("..."),
5    checkInDate: ISODate("2023-05-10T00:00:00Z"),
6    checkOutDate: ISODate("2023-05-15T00:00:00Z"),
7    numberOfGuests: 2,
8    totalPrice: 1250.00,
9    paymentStatus: "Opłacone",
10   specialRequests: ["Późne zameldowanie", "Dodatkowa poduszka"],
11   bookingDate: ISODate("2023-04-15T00:00:00Z"),
12   status: "Potwierdzona"
13 }

```

2.4 Kolekcja: employees

Opis: Zawiera dane pracowników hotelu wraz z informacjami o stanowisku, wynagrodzeniu i harmonogramie pracy.

Liczba dokumentów: 8

Struktura dokumentu:

```
1  {
2    _id: ObjectId("..."),
3    firstName: String,
4    lastName: String,
5    position: String,
6    email: String,
7    phoneNumber: String,
8    dateOfBirth: Date,
9    hireDate: Date,
10   salary: Number,
11   department: String,
12   workSchedule: [String],
13   address: {
14     street: String,
15     city: String,
16     postalCode: String,
17     country: String
18   }
19 }
```

Przykładowy dokument:

```
1  {
2    _id: ObjectId("..."),
3    firstName: "Tomasz",
4    lastName: "Nowicki",
5    position: "Szef kuchni",
6    email: "tomasz.nowicki@hotel.com",
7    phoneNumber: "+48567890123",
8    dateOfBirth: ISODate("1975-03-10T00:00:00Z"),
9    hireDate: ISODate("2018-06-01T00:00:00Z"),
10   salary: 7500.00,
11   department: "Restauracja",
12   workSchedule: ["Poniedziałek", "Wtorek", "Środa", "Niedziela"],
13   address: {
14     street: "Słoneczna 8",
15     city: "Pruszków",
16     postalCode: "05-800",
17     country: "Polska"
18   }
19 }
```

2.5 Kolekcja: services

Opis: Zawiera katalog usług dodatkowych oferowanych przez hotel.

Liczba dokumentów: 10

Struktura dokumentu:

```
1 {
2   _id: ObjectId("..."),
3   name: String,
4   description: String,
5   price: Number,
6   category: String,
7   availability: [String],
8   location: String,
9   requiresReservation: Boolean
10 }
```

Przykładowy dokument:

```
1 {
2   _id: ObjectId("..."),
3   name: "Masaż relaksacyjny",
4   description: "60-minutowy masaż relaksacyjny",
5   price: 200.00,
6   category: "SPA",
7   availability: ["Poniedziałek", "Środa", "Piątek", "Sobota"],
8   location: "Hotelowe SPA",
9   requiresReservation: true
10 }
```

2.6 Kolekcja: serviceOrders

Opis: Przechowuje zamówienia na usługi dodatkowe składane przez gości hotelowych.

Liczba dokumentów: 15

Struktura dokumentu:

```
1 {
2   _id: ObjectId("..."),
3   guestId: ObjectId("..."),
4   reservationId: ObjectId("..."),
5   serviceId: ObjectId("..."),
6   orderDate: Date,
7   quantity: Number,
8   totalPrice: Number,
9   status: String,
10  comments: String
11 }
```

Przykładowy dokument:

```

1 {
2   _id: ObjectId("..."),
3   guestId: ObjectId("..."),
4   reservationId: ObjectId("..."),
5   serviceId: ObjectId("..."),
6   orderDate: ISODate("2023-06-18T00:00:00Z"),
7   quantity: 1,
8   totalPrice: 300.00,
9   status: "Potwierdzone",
10  comments: "Kolacja na 20:00, rocznica ślubu"
11 }

```

3 Referencje między dokumentami

W bazie danych zaimplementowano następujące referencje między dokumentami:

1. Rezerwacje (reservations) - Goście (guests)

- Atrybut `guestId` w kolekcji `reservations` zawiera referencję do identyfikatora dokumentu z kolekcji `guests`
- Modeluje relację: "Gość dokonuje rezerwacji" (jeden gość może mieć wiele rezerwacji)

2. Rezerwacje (reservations) - Pokoje (rooms)

- Atrybut `roomId` w kolekcji `reservations` zawiera referencję do identyfikatora dokumentu z kolekcji `rooms`
- Modeluje relację: "Rezerwacja dotyczy konkretnego pokoju" (jeden pokój może mieć wiele rezerwacji w różnych terminach)

3. Zamówienia usług (serviceOrders) - Goście (guests)

- Atrybut `guestId` w kolekcji `serviceOrders` zawiera referencję do identyfikatora dokumentu z kolekcji `guests`
- Modeluje relację: "Gość zamawia usługę dodatkową" (jeden gość może złożyć wiele zamówień)

4. Zamówienia usług (serviceOrders) - Rezerwacje (reservations)

- Atrybut `reservationId` w kolekcji `serviceOrders` zawiera referencję do identyfikatora dokumentu z kolekcji `reservations`
- Modeluje relację: "Zamówienie usługi jest przypisane do konkretnej rezerwacji" (jedna rezerwacja może mieć wiele zamówień usług)

5. Zamówienia usług (serviceOrders) - Usługi (services)

- Atrybut `serviceId` w kolekcji `serviceOrders` zawiera referencję do identyfikatora dokumentu z kolekcji `services`
- Modeluje relację: "Zamówienie dotyczy konkretnej usługi" (jedna usługa może być zamawiana wielokrotnie)

4 Zapytania

4.1 Zapytania wyszukiujące (find)

4.1.1 Dostępne pokoje o pojemności większej niż 2 osoby

```
1 // Zapytanie wyszukiujące dostępne pokoje o pojemności większej niż 2 osoby
2 db.rooms.find(
3   {
4     status: "Dostępny",
5     capacity: { $gt: 2 }
6   }
7 )
```

Zapytanie zwraca wszystkie pokoje, które są obecnie dostępne i mogą pomieścić więcej niż 2 osoby. Wykorzystuje operator porównujący `$gt` (greater than).

4.1.2 Rezerwacje rozpoczynające się w określonym przedziale czasowym

```
1 // Zapytanie wyszukiujące rezerwacje z zameldowaniem w czerwcu 2023
2 db.reservations.find(
3   {
4     checkInDate: {
5       $gte: new Date("2023-06-01"),
6       $lt: new Date("2023-07-01")
7     }
8   }
9 )
```

Zapytanie zwraca wszystkie rezerwacje z datą zameldowania w czerwcu 2023. Wykorzystuje operatory porównujące `$gte` (greater than or equal) i `$lt` (less than).

4.2 Zapytania grupujące (aggregate)

4.2.1 Średnia cena i liczba rezerwacji według typu pokoju

```
1 // Zapytanie grupujące - statystyki rezerwacji wg typu pokoju
2 db.reservations.aggregate([
3   {
4     $lookup: {
5       from: "rooms",
6       localField: "roomId",
7       foreignField: "_id",
8       as: "room"
9     }
10  },
11  {
12    $unwind: "$room"
13  },
14  {
15    $group: {
```

```

16     _id: "$room.type",
17     averagePrice: { $avg: "$totalPrice" },
18     totalReservations: { $sum: 1 },
19     totalRevenue: { $sum: "$totalPrice" }
20   }
21 },
22 {
23   $sort: {
24     totalRevenue: -1
25   }
26 },
27 {
28   $project: {
29     roomType: "$_id",
30     averagePrice: { $round: ["$averagePrice", 2] },
31     totalReservations: 1,
32     totalRevenue: 1,
33     _id: 0
34   }
35 }
36 ]))

```

Zapytanie łączy dane z kolekcji rezerwacji i pokoi, a następnie grupuje je według typu pokoju. Dla każdego typu pokoju obliczane są: średnia cena rezerwacji, liczba rezerwacji i całkowity przychód. Wyniki są sortowane według przychodu malejąco. Wykorzystane operatory to \$group, \$lookup, \$sort i \$project.

4.2.2 Analiza zamówień usług dodatkowych według kategorii

```

1  // Zapytanie grupujące - analiza zamówień usług wg kategorii
2  db.serviceOrders.aggregate([
3    {
4      $lookup: {
5        from: "services",
6        localField: "serviceId",
7        foreignField: "_id",
8        as: "serviceDetails"
9      }
10   },
11   {
12     $unwind: "$serviceDetails"
13   },
14   {
15     $match: {
16       status: { $in: ["Potwierdzone", "Zrealizowane"] }
17     }
18   },
19   {
20     $group: {
21       _id: "$serviceDetails.category",

```

```

22     totalOrders: { $sum: 1 },
23     totalRevenue: { $sum: "$totalPrice" },
24     averageQuantity: { $avg: "$quantity" }
25   }
26 },
27 {
28   $sort: {
29     totalOrders: -1
30   }
31 },
32 {
33   $limit: 5
34 },
35 {
36   $project: {
37     category: "$_id",
38     totalOrders: 1,
39     totalRevenue: { $round: ["$totalRevenue", 2] },
40     averageQuantity: { $round: ["$averageQuantity", 2] },
41     _id: 0
42   }
43 }
44 ])
```

Zapytanie analizuje zamówienia usług dodatkowych, łącząc dane z kolekcji zamówień i usług. Filtruje zamówienia o statusie "Potwierdzone" lub "Zrealizowane", grupuje je według kategorii usługi i oblicza statystyki: liczbę zamówień, całkowity przychód i średnią ilość zamówionych usług. Wyniki są sortowane według liczby zamówień i ograniczone do 5 najlepszych kategorii. Wykorzystane operatory to `$lookup`, `$match`, `$group`, `$sort`, `$limit` i `$project`.

4.3 Zapytanie zliczające (count)

```

1 // Zapytanie zliczające pokoje wg statusu dostępności
2 db.rooms.countDocuments(
3   {
4     status: "Dostępny",
5     pricePerNight: { $lt: 500 }
6   }
7 )
```

Zapytanie zwraca liczbę pokoi, które są obecnie dostępne i kosztują mniej niż 500 zł za noc. Wykorzystuje metodę `countDocuments` z parametrem `query`.

4.4 Zapytanie o wartości unikalne (distinct)

```

1 // Zapytanie o unikalne kraje pochodzenia gości
2 db.guests.distinct(
3   "address.country",
4   {
```

```

5     loyaltyPoints: { $gt: 50 }
6   }
7 )

```

Zapytanie zwraca listę unikalnych krajów pochodzenia gości, którzy posiadają więcej niż 50 punktów lojalnościowych. Wykorzystuje metodę `distinct` z parametrem query.

5 Indeksy

W celu optymalizacji zapytań w bazie danych utworzono następujące indeksy:

```

1  // Indeks dla szybkiego wyszukiwania pokoi po numerze
2  db.rooms.createIndex({ roomNumber: 1 }, { unique: true })
3
4  // Indeks dla szybkiego wyszukiwania gości po nazwisku i imieniu
5  db.guests.createIndex({ lastName: 1, firstName: 1 })
6
7  // Indeks dla szybkiego wyszukiwania rezerwacji wg dat
8  db.reservations.createIndex({ checkInDate: 1, checkOutDate: 1 })
9
10 // Indeks dla szybszego łączenia rezerwacji z pokojami
11 db.reservations.createIndex({ roomId: 1 })
12
13 // Indeks dla szybszego łączenia rezerwacji z gośćmi
14 db.reservations.createIndex({ guestId: 1 })
15
16 // Indeks dla szybkiego wyszukiwania usług wg kategorii i ceny
17 db.services.createIndex({ category: 1, price: 1 })
18
19 // Indeks dla szybszego łączenia zamówień usług z usługami
20 db.serviceOrders.createIndex({ serviceId: 1 })
21
22 // Indeks dla szybszego łączenia pracowników z działami i stanowiskami
23 db.employees.createIndex({ department: 1, position: 1 });

```

Indeksy przyspieszają operacje wyszukiwania i łączenia danych, szczególnie w przypadku złożonych zapytań agregujących. Na przykład, indeks `roomNumber` zapewnia szybkie wyszukiwanie pokoju po jego numerze, a indeksy na polach `guestId` i `roomId` w kolekcji `reservations` przyspieszają operacje łączenia danych z różnych kolekcji.