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 umożliwia 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 30 dokumentów. Poniżej przedstawiono szczegółowy opis każdej kolekcji wraz z przykładowymi dokumentami.
Struktura bazy danych systemu hotelowego w MongoDB

2.1 Kolekcja: guests

Opis: Przechowuje dane gości hotelowych wraz z informacjami kontaktowymi i dodatkowymi atrybutami.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  firstName: String,
  lastName: String,
  email: String,
  phoneNumber: String,
  address: {
```

```
    street: String,
    city: String,
    postalCode: String,
    country: String
  },
  dateOfBirth: Date,
  loyaltyPoints: Number,
  registrationDate: Date
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  firstName: "Jan",
  lastName: "Kowalski",
  email: "jan.kowalski@email.com",
  phoneNumber: "+48123456789",
  address: {
    street: "Kwiatowa 12",
    city: "Warszawa",
    postalCode: "01-234",
    country: "Polska"
  },
  dateOfBirth: ISODate("1985-05-15T00:00:00Z"),
  loyaltyPoints: 120,
  registrationDate: ISODate("2020-03-10T00:00:00Z")
}
```

2.2 Kolekcja: rooms

Opis: Zawiera informacje o pokojach hotelowych, ich typach, wyposażeniu i statusie dostępności.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  roomNumber: String,
  floor: Number,
  type: String,
  capacity: Number,
  pricePerNight: Number,
  amenities: [String],
  status: String,
  lastRenovation: Date
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  roomNumber: "301",
  floor: 3,
  type: "Suite",
  capacity: 4,
  pricePerNight: 600.00,
  amenities: ["TV", "WiFi", "Minibar", "Klimatyzacja",
              "Sejf", "Ekspres do kawy", "Jacuzzi"],
  status: "Dostępny",
  lastRenovation: ISODate("2022-02-15T00:00:00Z")
}
```

2.3 Kolekcja: reservations

Opis: Przechowuje informacje o rezerwacjach gości, zawiera referencje do kolekcji guests i rooms.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  guestId: ObjectId("..."),
  roomId: ObjectId("..."),
  checkInDate: Date,
  checkOutDate: Date,
  numberOfGuests: Number,
  totalPrice: Number,
  paymentStatus: String,
  specialRequests: [String],
  bookingDate: Date,
  status: String
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  guestId: ObjectId("..."),
  roomId: ObjectId("..."),
  checkInDate: ISODate("2023-05-10T00:00:00Z"),
  checkOutDate: ISODate("2023-05-15T00:00:00Z"),
  numberOfGuests: 2,
  totalPrice: 1250.00,
  paymentStatus: "Opłacone",
  specialRequests: ["Późne zameldowanie", "Dodatkowa poduszka"],
  bookingDate: ISODate("2023-04-15T00:00:00Z"),
  status: "Potwierdzona"
}
```

2.4 Kolekcja: employees

Opis: Zawiera dane pracowników hotelu wraz z informacjami o stanowisku, wynagrodzeniu i harmonogramie pracy.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  firstName: String,
  lastName: String,
  position: String,
  email: String,
  phoneNumber: String,
  dateOfBirth: Date,
  hireDate: Date,
  salary: Number,
  department: String,
  workSchedule: [String],
  address: {
    street: String,
    city: String,
    postalCode: String,
    country: String
  }
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  firstName: "Tomasz",
  lastName: "Nowicki",
  position: "Szef kuchni",
  email: "tomasz.nowicki@hotel.com",
  phoneNumber: "+48567890123",
  dateOfBirth: ISODate("1975-03-10T00:00:00Z"),
  hireDate: ISODate("2018-06-01T00:00:00Z"),
  salary: 7500.00,
  department: "Restauracja",
  workSchedule: ["Poniedziałek", "Wtorek", "Środa", "Niedziela"],
  address: {
    street: "Słoneczna 8",
    city: "Pruszków",
    postalCode: "05-800",
    country: "Polska"
  }
}
```

2.5 Kolekcja: services

Opis: Zawiera katalog usług dodatkowych oferowanych przez hotel.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  name: String,
  description: String,
  price: Number,
  category: String,
  availability: [String],
  location: String,
  requiresReservation: Boolean
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  name: "Masaż relaksacyjny",
  description: "60-minutowy masaż relaksacyjny",
  price: 200.00,
  category: "SPA",
  availability: ["Poniedziałek", "Środa", "Piątek", "Sobota"],
  location: "Hotelowe SPA",
  requiresReservation: true
}
```

2.6 Kolekcja: serviceOrders

Opis: Przechowuje zamówienia na usługi dodatkowe składane przez gości hotelowych.
Liczba dokumentów: 5
Struktura dokumentu:

```
{
  _id: ObjectId("..."),
  guestId: ObjectId("..."),
  reservationId: ObjectId("..."),
  serviceId: ObjectId("..."),
  orderDate: Date,
  quantity: Number,
  totalPrice: Number,
  status: String,
  comments: String
}
```

Przykładowy dokument:

```
{
  _id: ObjectId("..."),
  guestId: ObjectId("..."),
  reservationId: ObjectId("..."),
  serviceId: ObjectId("..."),
  orderDate: ISODate("2023-06-18T00:00:00Z"),
  quantity: 1,
  totalPrice: 300.00,
  status: "Potwierdzone",
  comments: "Kolacja na 20:00, rocznica ślubu"
}
```

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 wyszukujące (find)

4.1.1 Dostępne pokoje o pojemności większej niż 2 osoby

```
// Zapytanie wyszukujące dostępne pokoje o pojemności większej niż 2 osoby
db.rooms.find(
  {
    status: "Dostępny",
    capacity: { $gt: 2 }
  }
)
```

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

```
// Zapytanie wyszukujące rezerwacje z zameldowaniem w czerwcu 2023
db.reservations.find(
  {
    checkInDate: {
      $gte: new Date("2023-06-01"),
      $lt: new Date("2023-07-01")
    }
  }
)
```

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

```
// Zapytanie grupujące - statystyki rezerwacji wg typu pokoju
db.reservations.aggregate([
  {
    $lookup: {
      from: "rooms",
      localField: "roomId",
      foreignField: "_id",
      as: "room"
    }
  },
  {
    $unwind: "$room"
  },
  {
    $group: {
```

```
      _id: "$room.type",
      averagePrice: { $avg: "$totalPrice" },
      totalReservations: { $sum: 1 },
      totalRevenue: { $sum: "$totalPrice" }
    }
  },
  {
    $sort: {
      totalRevenue: -1
    }
  },
  {
    $project: {
      roomType: "$_id",
      averagePrice: { $round: ["$averagePrice", 2] },
      totalReservations: 1,
      totalRevenue: 1,
      _id: 0
    }
  }
])
```

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

```
// Zapytanie grupujące - analiza zamówień usług wg kategorii
db.serviceOrders.aggregate([
  {
    $lookup: {
      from: "services",
      localField: "serviceId",
      foreignField: "_id",
      as: "serviceDetails"
    }
  },
  {
    $unwind: "$serviceDetails"
  },
  {
    $match: {
      status: { $in: ["Potwierdzone", "Zrealizowane"] }
    }
  },
  {
    $group: {
      _id: "$serviceDetails.category",
```

```
      totalOrders: { $sum: 1 },
      totalRevenue: { $sum: "$totalPrice" },
      averageQuantity: { $avg: "$quantity" }
    }
  },
  {
    $sort: {
      totalOrders: -1
    }
  },
  {
    $limit: 5
  },
  {
    $project: {
      category: "$_id",
      totalOrders: 1,
      totalRevenue: { $round: ["$totalRevenue", 2] },
      averageQuantity: { $round: ["$averageQuantity", 2] },
      _id: 0
    }
  }
])
```

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)

```
// Zapytanie zliczające pokoje wg statusu dostępności
db.rooms.countDocuments(
  {
    status: "Dostępny",
    pricePerNight: { $lt: 500 }
  }
)
```

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)

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

```
    loyaltyPoints: { $gt: 50 }
  }
)
```

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:

```
// Indeks dla szybkiego wyszukiwania pokoi po numerze
db.rooms.createIndex({ roomNumber: 1 }, { unique: true })

// Indeks dla szybkiego wyszukiwania gości po nazwisku i imieniu
db.guests.createIndex({ lastName: 1, firstName: 1 })

// Indeks dla szybkiego wyszukiwania rezerwacji wg dat
db.reservations.createIndex({ checkInDate: 1, checkOutDate: 1 })

// Indeks dla szybszego łączenia rezerwacji z pokojami
db.reservations.createIndex({ roomId: 1 })

// Indeks dla szybszego łączenia rezerwacji z gośćmi
db.reservations.createIndex({ guestId: 1 })

// Indeks dla szybkiego wyszukiwania usług wg kategorii i ceny
db.services.createIndex({ category: 1, price: 1 })

// Indeks dla szybszego łączenia zamówień usług z usługami
db.serviceOrders.createIndex({ serviceId: 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.