E-Z Beach

Simplify Your Day at the Beach

DATABASE DESIGN

Dustin Kent
07/16/2023

Overview

I chose MongoDB as the database type for my beach setup application because it perfectly suits the needs of my application. MongoDB's flexibility and document-oriented nature make it a seamless choice for storing and retrieving complex data structures, which is essential for handling the dynamic requirements of my application. It also offers scalability and high performance, ensuring that my application can handle many users and a lot of data without any issues. The powerful querying capabilities of MongoDB, in addition to its extensive ecosystem of resources, tools, and community support, equip me with the tools to fulfill the requirements of my application.

To organize my data effectively, I have designed separate collections for users, locations, items, reservations, and jobs. Each collection stores the relevant information about its corresponding resource. For example, the “users” collection holds user details, the “locations” collection contains beach location information, the “items” collection stores available beach items, and so on. This structured approach enables easy retrieval, manipulation, and management of data, which is crucial for a RESTful architecture.

By choosing MongoDB for my application, I can ensure that it aligns well with RESTful principles and allows for seamless resource management. It offers efficient querying capabilities and the ability to scale as my application grows. With MongoDB as the backbone of my application, I can create a user-friendly API that facilitates smooth communication between users and employees for beach item setup.

Users

Document Structure

```
{
 "_id": ObjectId("created by MongoDB"),
 "name": "John Doe",
 "email": "johndoe@example.com",
 "password": "hashedpassword",
 "phone": "123-456-7890",
 "address": {
  "street": "123 Main Street",
  "city": "City",
  "state": "State",
  "postalCode": "12345",
  "country": "Country"
 }
 // Additional employee-specific fields as found necessary

}
```

Employees

Document Structure

```
{
 "_id": ObjectId("created by MongoDB”),
 "name": "John Smith",
 "email": "johnsmith@example.com",
 "password": "hashedpassword",
 "phone": "123-456-7890",
 "address": {
  "street": "123 Main Street",
  "city": "City",
  "state": "State",
  "postalCode": "12345",
  "country": "Country"
 },
 "assignedJobs": [
  {
   "jobId": ObjectId("<job_id_1>"),
   "status": "assigned"
  },
  {
   "jobId": ObjectId("<job_id_2>"),
   "status": "completed"
  },
  {
   "jobId": ObjectId("<job_id_3>"),
   "status": "canceled"
  }
 ],
```

```
  // Additional employee-specific fields as found necessary

}
```

Locations

Document Structure

```
{
  "name": "Beach A",
  "address": "123 Beach Street",
  "coordinates": {
    "latitude":  ######,
    "longitude": ######
  },
  // Additional location-specific fields as found necessary
}
```

Items

Document Structure

```
{
  "name": "Beach Umbrella",
  "description": "Provides shade on the beach",
  "price": 4.99,
  “quantity”: 4,
  // Additional item-specific fields as found necessary
}
```

Reservations

Document Structure

```
{
  "userId": "<user_id>",
  "itemIds": [
    "<item_id_1>",
    "<item_id_2>"
  ],
  "timing": {
    "start": "2023-07-15T10:00:00Z",
    "end": "2023-07-15T18:00:00Z"
  },
  "status": "confirmed",
  "cancellable": true,
  "cancellationWindowStart": "2023-07-15T10:00:00Z",
  "cancellationWindowEnd": "2023-07-15T10:02:00Z"
}
```

Jobs

Document Structure

```
{
  "locationId": "<location_id>",
  "employeeId": "<employee_id>",
  "itemIds": [
    "<item_id_1>",
    "<item_id_2>"
  ],
  "timing": {
    "start": "2023-07-16T09:00:00Z",
    "end": "2023-07-16T17:00:00Z"
  },
  "status": "accepted/refused"
}
```

Purpose, Implementation, and Interactions

Users

Purpose: The Users collection serves the purpose of storing user information and enabling personalized experiences, authentication, and communication related to their request.

Implementation: Each user is represented by a document in the collection, containing fields such as name, email, password, phone, and address. The address field is structured to include street, city, state, postal code, and country.

Interaction: The Users collection is interacted with during the account creation process, user login and authentication, and for accessing and managing user-specific features and information within the application. User will see and be able to choose location, dates, times, and items for setup and make/cancel reservations.

Employees

Purpose: The Employees collection is designed to store employee information and track their assigned, completed, and canceled jobs.

Implementation: Each employee is represented by a document in the collection, including fields like name, email, password, phone, address, and an assignedJobs array. The assignedJobs array holds job information, including the job ID and status (assigned/completed/cancelled).

Interaction: The Employees collection is interacted with to manage employee profiles, track job assignments and status, and view the job history of each employee. Once logged in, employees will see available jobs and be able to choose one if available.

Locations

Purpose: The Locations collection is responsible for storing beach location information and facilitating location-based services within the application.

Implementation: Each beach location is represented by a document in the collection, containing fields such as name, address, and coordinates (latitude and longitude). (Pinning Locations is important)

Interaction: The Locations collection is interacted with to display available beach locations, set default and preferred locations for users, and provide location-related services within the application. Users will be able to choose their location, and employees will be able to see the locations where jobs are requested.

Items

Purpose: The Items collection is used to store information about the available beach setup items (chairs, umbrellas, coolers, tents, toys, carts, radios, etc).

Implementation: Each item is represented by a document in the collection, including fields like name, description, and price, as well as quantity.

Interaction: The Items collection is interacted with to present users with a range of available items for beach setup, provide item descriptions, quantities, and pricing information, and manage item-related operations. User will select the details of what they need setup and employees will be able to see what has been requested.

Reservations

Purpose: The Reservations collection is responsible for storing user reservation information, including the selected items, timing, and cancellation details.

Implementation: Each reservation is represented by a document in the collection, containing fields such as the user ID, an array of item IDs, timing details, status, and cancellation window information.

Interaction: The Reservations collection is interacted with for creating and managing user reservations, tracking timing details, confirming reservation status, and handling cancellations within the designated window.

Jobs

Purpose: The Jobs collection stores information about setup jobs at beach locations and tracks their assignment and completion status.

Implementation: Each job is represented by a document in the collection, including fields such as location ID, employee ID, item IDs, timing details, and status.

Interaction: The Jobs collection is interacted with to manage job assignments, track job details, and ensure coordination between employees and users for setting up beach items.