Dokumentacja projektu bazy danych StreetBall App

Autor: Vladyslav Liulka
Grupa: IO 6.8

Spis treści

1. Opis projektu

StreetBall App to aplikacja internetowa do organizowania i uczestniczenia w ulicznych grach sportowych w Polsce, z początkowym naciskiem na streetball i koszykówkę, z możliwością rozszerzenia na inne sporty (piłka nożna, siatkówka itp.). Głównym celem aplikacji jest uproszczenie procesu wyszukiwania gier i boisk, a także organizowania własnych gier.

System ma ułatwić entuzjastom sportów ulicznych znalezienie dostępnych boisk, dołączanie do istniejących gier oraz tworzenie własnych wydarzeń sportowych. Początkowo aplikacja skupia się na grach typu streetball i koszykówka, ale struktura danych została zaprojektowana z myślą o przyszłej rozbudowie o inne sporty.

2. Wymagania funkcjonalne bazy danych

Przechowywanie informacji o użytkownikach:

- Podstawowe dane profilu (nazwa użytkownika, e-mail, hasło)
- Ustawienia powiadomień
- Linki do utworzonych i dołączonych gier

Przechowywanie informacji o obiektach sportowych:

- Współrzędne geograficzne do wyświetlenia na mapie
- Szczegółowe informacje o boisku (rodzaj nawierzchni, oświetlenie itp.)
- Obsługiwane dyscypliny sportowe
- Oceny i recenzje

Przechowywanie informacji o grze:

- Czas i czas trwania
- Lokalizacja (boisko)
- Format gry (3x3, 5x5 itp.)
- Lista uczestników
- Status gry (zaplanowana, w toku, zakończona, anulowana)

Przechowywanie powiadomień:

- Śledzenie przeczytanych/nieprzeczytanych powiadomień
- Komunikacja z użytkownikami i grami
- Typ powiadomienia (przypomnienie, dołączenie gracza itp.)

Dodatkowo, baza danych musi umożliwiać:

- Wyszukiwanie dostępnych boisk w okolicy
- Filtrowanie gier według poziomu zaawansowania, formatu i typu sportu
- Monitorowanie zaplanowanych gier i przypomnienia o nich
- Analizę aktywności użytkowników i popularności boisk

3. Struktura kolekcji

Baza danych MongoDB składa się z następujących kolekcji:

Kolekcja users:

Przechowuje informacje o użytkownikach aplikacji:

```
{
  _id: ObjectId,
  username: String,
  email: String,
  password: String (hashed),
  fullName: String,
  avatar: String,
  phone: String,
  createdGames: [ObjectId], // referencje do kolekcji games
```

```
  joinedGames: [ObjectId],  // referencje do kolekcji games
  notifications: {
    email: Boolean,
    push: Boolean,
    reminderTime: Number
  },
  createdAt: Date,
  updatedAt: Date
}
```

Kolekcja courts:

Przechowuje informacje o boiskach dostępnych w systemie:

```
{
  _id: ObjectId,
  name: String,
  location: {
    type: String,
    coordinates: [Number, Number], // długość i szerokość
geograficzna
    address: String
  },
  sportTypes: [String],
  photos: [String],
  description: String,
  features: {
    covered: Boolean,
    lighting: Boolean,
    surface: String,
    changingRooms: Boolean
  },
  workingHours: {
    monday: { open: String, close: String },
    // ... dla pozostałych dni tygodnia
  },
  rating: Number,
  reviews: [
    {
      user: ObjectId, // referencja do kolekcji users
      text: String,
      rating: Number,
      date: Date
    }
```

```
  ],
  createdAt: Date,
  updatedAt: Date
}
```

Kolekcja games:

Przechowuje informacje o grach utworzonych przez użytkowników:

```
{
  _id: ObjectId,
  court: ObjectId, // referencja do kolekcji courts
  creator: ObjectId, // referencja do kolekcji users
  sportType: String,
  dateTime: Date,
  duration: Number, // w minutach
  format: String, // np. "3x3", "5x5"
  maxPlayers: Number,
  currentPlayers: [
    {
      user: ObjectId, // referencja do kolekcji users
      joinedAt: Date
    }
  ],
  status: String, // "scheduled", "in_progress", "completed",
"cancelled"
  description: String,
  skillLevel: String, // "beginner", "intermediate",
"advanced", "any"
  isPrivate: Boolean,
  inviteCode: String, // opcjonalnie dla prywatnych gier
  createdAt: Date,
  updatedAt: Date
}
```

Kolekcja notifications:

Przechowuje informacje o powiadomieniach wysyłanych do użytkowników:

```
{
  _id: ObjectId,
  user: ObjectId, // referencja do kolekcji users
  game: ObjectId, // referencja do kolekcji games
  type: String, // typ powiadomienia
```

```
  message: String,
  isRead: Boolean,
  scheduledFor: Date, // opcjonalnie dla przypomnień
  createdAt: Date,
  updatedAt: Date
}
```

4. Implementacja bazy danych

Poniżej przedstawione są skrypty MongoDB, które zostały wykorzystane do utworzenia i wypełnienia bazy danych przykładowymi danymi.

Tworzenie kolekcji i wypełnianie danymi

Skrypt tworzący kolekcje i wypełniający je przykładowymi danymi:

javascript

```
// Przełączenie na schemat lub jego utworzenie
use streetball-db

// Usunięcie kolekcji (jeśli istnieją)
db.users.drop()
db.courts.drop()
db.games.drop()
db.notifications.drop()

// Utworzenie kolekcji
db.createCollection("users")
db.createCollection("courts")
db.createCollection("games")
db.createCollection("notifications")

// Dodawanie dokumentów do kolekcji users
db.users.insertMany([
  {
    username: "adamkoszy",
    email: "adam@example.pl",
    password:
"$2a$10$qnPGv0opAm70OETJCu6qXuIjbwKS7aSQi69ZcP2aYKk2Wj3j7Fq/i"
, // "password123"
    fullName: "Adam Kowalski",
    avatar: "default-avatar.png",
    phone: "+48 501 234 567",
```

```
    createdGames: [],
    joinedGames: [],
    notifications: {
      email: true,
      push: true,
      reminderTime: 60
    },
    createdAt: new Date(),
    updatedAt: new Date()
  },
  // ... pozostali użytkownicy
])

// Dodawanie danych do kolekcji courts, games, notifications
// ... (szczegóły dostępne w pliku fill_with_data_script.txt)
```

Skrypt tworzy cztery kolekcje i wypełnia je przykładowymi danymi, które umożliwiają testowanie funkcjonalności aplikacji. Dodane są przykładowe dane:

- 5 użytkowników
- 5 boisk o różnych cechach (nawierzchni, oświetleniu, itp.)
- 5 gier o różnych formatach (3x3, 5x5, itp.) i poziomach zaawansowania
- 5 powiadomień różnych typów

5. Zapytania do bazy danych

Poniżej przedstawione są przykłady zapytań wykonywanych na bazie danych, które realizują wymagane funkcjonalności aplikacji:

Zapytanie 1: Wyszukiwanie boisk obsługujących określone sporty

javascript

```
// Boiska do koszykówki i siatkówki
db.courts.find({
  sportTypes: { $all: ["streetball", "volleyball"] }
})
```

To zapytanie zwraca boiska, które obsługują zarówno streetball, jak i siatkówkę.

Zapytanie 2: Wyszukiwanie gier o określonym formacie w konkretnym dniu

javascript

```
// Gry 3x3 zaplanowane na jutro
const tomorrow = new Date();
```

```
tomorrow.setDate(tomorrow.getDate() + 1);
tomorrow.setHours(0, 0, 0, 0);
const dayAfter = new Date(tomorrow);
dayAfter.setDate(dayAfter.getDate() + 1);

db.games.find({
  format: "3x3",
  dateTime: { $gte: tomorrow, $lt: dayAfter },
  status: "scheduled"
})
```

To zapytanie wyszukuje wszystkie zaplanowane gry w formacie 3x3, które odbywają się następnego dnia.

Zapytanie 3: Wyszukiwanie boisk o określonych cechach

javascript

```
// Zadaszone boiska z drewnianą podłogą
db.courts.find({
  "features.covered": true,
  "features.surface": "wood"
})
```

To zapytanie wyszukuje wszystkie kryte boiska z drewnianą nawierzchnią.

Zapytanie 4: Wyszukiwanie boisk o wysokiej ocenie i z recenzjami

javascript

```
// Boiska z oceną > 4.5 oraz posiadające review
db.courts.find({
  rating: { $gt: 4.5 },
  "reviews.0": { $exists: true }
})
```

To zapytanie zwraca boiska o ocenie powyżej 4.5, które mają co najmniej jedną recenzję.

Zapytanie 5: Wyszukiwanie użytkowników z nieprzeczytanymi powiadomieniami

javascript

```
// Użytkownicy, którzy mają nieprzeczytane powiadomienia
const userIds = db.notifications.distinct("user", { isRead:
false });

db.users.find({ _id: { $in: userIds } })
```

To zapytanie znajduje wszystkich użytkowników, którzy mają nieprzeczytane powiadomienia.

Zapytanie 6: Agregacja - Liczba zaplanowanych gier na boiskach

javascript

```
// Liczenie zaplanowanych gier dla boisk
db.games.aggregate([
  { $match: { status: "scheduled" } },
  { $group: {
    _id: "$court",
    gameCount: { $sum: 1 }
  }},
  { $lookup: {
    from: "courts",
    localField: "_id",
    foreignField: "_id",
    as: "courtInfo"
  }},
  { $project: {
    courtName: { $arrayElemAt: ["$courtInfo.name", 0] },
    gameCount: 1,
    _id: 0
  }},
  { $sort: { gameCount: -1 } }
])
```

Ta agregacja liczy zaplanowane gry dla każdego boiska i zwraca boiska posortowane według liczby gier.

Zapytanie 7: Agregacja - Średni poziom zapełnienia gier według formatów

javascript

```
// Liczenie AVG liczby ludzi dla formatu gier
db.games.aggregate([
  { $match: { status: "scheduled" } },
  { $project: {
    format: 1,
    occupancyRate: {
      $multiply: [
        { $divide: [{ $size: "$currentPlayers" },
"$maxPlayers"] },
        100
      ]
    }
```

```
  }},
  { $group: {
    _id: "$format",
    averageOccupancy: { $avg: "$occupancyRate" },
    gameCount: { $sum: 1 }
  }},
  { $sort: { averageOccupancy: -1 } }
])
```

Ta agregacja oblicza średni poziom zapełnienia (w procentach) dla gier w różnych formatach.

Zapytanie 8: Agregacja - Analiza aktywności użytkowników

javascript

```
// Analiza aktywności użytkowników: liczba utworzonych gier i
gier, do których się dołączyli
db.users.aggregate([
  { $project: {
    username: 1,
    createdGamesCount: { $size: "$createdGames" },
    joinedGamesCount: { $size: "$joinedGames" }
  }},
  { $addFields: {
    totalActivity: { $add: ["$createdGamesCount",
"$joinedGamesCount"] }
  }},
  { $sort: { totalActivity: -1 } }
])
```

Ta agregacja analizuje aktywność użytkowników, zliczając liczbę utworzonych i dołączonych gier, a następnie sortuje użytkowników według całkowitej aktywności.

Zapytanie 9: Agregacja - Liczba boisk dla każdego sportu

javascript

```
// Liczba boisk dla każdego sportu
db.courts.aggregate([
  { $unwind: "$sportTypes" },
  { $group: {
    _id: "$sportTypes",
    courtCount: { $sum: 1 }
  }},
  { $sort: { courtCount: -1 } }
])
```

Ta agregacja liczy, ile boisk obsługuje każdy z typów sportów.

Zapytanie 10: Agregacja - Analiza najbardziej popularnych godzin gry

javascript

```
// Analiza najbardziej popularnych godzin
db.games.aggregate([
  { $match: { status: "scheduled" } },
  { $project: {
    hour: { $hour: "$dateTime" },
    dayOfWeek: { $dayOfWeek: "$dateTime" },
    players: { $size: "$currentPlayers" }
  }},
  { $group: {
    _id: { hour: "$hour", dayOfWeek: "$dayOfWeek" },
    gameCount: { $sum: 1 },
    totalPlayers: { $sum: "$players" }
  }},
  { $sort: { totalPlayers: -1 } },
  { $limit: 5 },
  { $project: {
    _id: 0,
    hour: "$_id.hour",
    dayOfWeek: "$_id.dayOfWeek",
    gameCount: 1,
    totalPlayers: 1,
    averagePlayers: { $divide: ["$totalPlayers", "$gameCount"]
}
  }}
])
```

Ta agregacja analizuje, które godziny i dni tygodnia są najbardziej popularne do gry, bazując na liczbie graczy.

6. Operacje aktualizacji danych

Poniżej przedstawione są przykłady operacji aktualizacji danych w bazie:

Aktualizacja 1: Zmiana cech boiska

javascript

```
// Zmiana godzin pracy, dodanie oświetlenia i zmiana opisu
db.courts.updateOne(
  { name: "Park Jordana" },
```

```
  {
    $set: {
      "features.lighting": true,
      "workingHours.friday.open": "07:00",
      "workingHours.friday.close": "23:00",
      "description": "Odnowiony park z profesjonalnym boiskiem
do koszykówki, doskonałą nawierzchnią i nowoczesnym
oświetleniem LED"
    }
  }
)
```

Ta operacja aktualizuje godziny otwarcia, dodaje oświetlenie i zmienia opis boiska "Park Jordana".

Aktualizacja 2: Dodanie recenzji i aktualizacja oceny boiska

javascript

```
// Zwiększenie oceny boiska i dodanie nowego review
const userId = db.users.findOne({ username: "piotr_pro"
})._id;

db.courts.updateOne(
  { name: "Szkoła Podstawowa nr 45" },
  {
    $inc: { rating: 0.2 },
    $push: {
      reviews: {
        user: userId,
        text: "Po remoncie nawierzchnia jest znacznie lepsza,
polecam na treningi!",
        rating: 5,
        date: new Date()
      }
    }
  }
)
```

Ta operacja dodaje nową recenzję boiska i zwiększa jego ocenę o 0.2.

Aktualizacja 3: Zmiana statusu gry i usunięcie gracza

javascript

```
// Zmiana statusu gry oraz usunięcie gracza
```

```
db.games.updateOne(
  { sportType: "streetball", format: "3x3", status:
"scheduled" },
  {
    $set: { status: "in_progress" },
    $pull: {
      currentPlayers: {
        user: db.users.findOne({ username: "kasia_basket"
})._id
      }
    }
  }
)
```

Ta operacja zmienia status gry na "w trakcie" i usuwa jednego z graczy z listy uczestników.

Aktualizacja 4: Dodanie nowego sportu i zmiana nazwy pola

javascript

```
// Dodanie nowego sportu oraz zmiana nazwy pola
db.courts.updateOne(
  { name: "Centrum Młodzieżowe" },
  {
    $addToSet: { sportTypes: "football" },
    $rename: { "features.covered": "features.isIndoor" }
  }
)
```

Ta operacja dodaje nowy sport (piłka nożna) do boiska i zmienia nazwę pola z "covered" na "isIndoor".

Aktualizacja 5: Aktualizacja wielu dokumentów jednocześnie

javascript

```
// Zmiana ustawień notification dla wszystkich użytkowników i
zwiększenie czasu notification
db.users.updateMany(
  { "notifications.reminderTime": { $lt: 60 } },
  {
    $set: { "notifications.push": true },
    $inc: { "notifications.reminderTime": 15 }
  }
)
```

Ta operacja aktualizuje ustawienia powiadomień dla wszystkich użytkowników, którzy mają czas przypomnienia mniejszy niż 60 minut, ustawiając powiadomienia push na true i zwiększając czas przypomnienia o 15 minut.

7. Zarządzanie użytkownikami bazy danych

W ramach zabezpieczenia bazy danych utworzono trzech użytkowników o różnych poziomach uprawnień:

Administrator bazy danych

javascript

```
// Admin dla całej DB
use admin
db.createUser(
  {
    user: "dbAdmin",
    pwd: "secure_admin_password",
    roles: [
      { role: "userAdminAnyDatabase", db: "admin" },
      { role: "dbAdminAnyDatabase", db: "admin" },
      { role: "readWriteAnyDatabase", db: "admin" }
    ]
  }
)
```

Ten użytkownik ma pełne uprawnienia do zarządzania bazą danych, w tym tworzenia, modyfikowania i usuwania kolekcji oraz dokumentów w dowolnej bazie.

Użytkownik z uprawnieniami do odczytu i zapisu

javascript

```
// Użytkownik read/write tylko dla streetball-db
use streetball-db
db.createUser(
  {
    user: "streetball_dev",
    pwd: "dev_password",
    roles: [
      { role: "readWrite", db: "streetball-db" }
    ]
  }
)
```

Ten użytkownik może odczytywać i modyfikować dane w bazie streetball-db, ale nie ma uprawnień administracyjnych.

Użytkownik tylko do odczytu

javascript

```
// Użytkownik read tylko dla streetball_db
use streetball-db
db.createUser(
  {
    user: "streetball_reader",
    pwd: "read_only_password",
    roles: [
      { role: "read", db: "streetball-db" }
    ]
  }
)
```

Ten użytkownik może tylko odczytywać dane z bazy streetball-db, bez możliwości wprowadzania zmian.

8. Eksport i kopie zapasowe bazy

Wykonano eksport danych z bazy w celu zabezpieczenia przed utratą danych:

Eksport danych

bash

```
mongodump --db streetball-db --out exported_collections
```

Komenda eksportuje wszystkie kolekcje z bazy streetball-db do katalogu exported_collections.

Tworzenie kopii zapasowej

bash

```
mongodump --db streetball-db --out streetball-db
```

Komenda tworzy kopię zapasową bazy streetball-db.

Podsumowanie

Baza danych StreetBall App została zaprojektowana i zaimplementowana zgodnie z wymaganiami funkcjonalnymi. Zawiera wszystkie niezbędne kolekcje i relacje między nimi, co umożliwia efektywne zarządzanie danymi aplikacji.

Przeprowadzone operacje i zapytania potwierdzają, że baza danych spełnia wszystkie wymagania dotyczące:

- Przechowywania informacji o użytkownikach, boiskach, grach i powiadomieniach
- Wyszukiwania i filtrowania danych według różnych kryteriów
- Analizy danych za pomocą agregacji
- Aktualizacji i modyfikacji danych
- Zabezpieczenia bazy przez system uprawnień użytkowników
- Tworzenia kopii zapasowych i eksportu danych

Struktura bazy jest elastyczna i pozwala na przyszłe rozszerzenie o dodatkowe sporty oraz funkcjonalności, zgodnie z wymaganiami projektu.