Large-Scale and Multi-Structured Databases

Project Design

BeatBuddy

Luca Arduini

Giovanni Enrico Loni

Lorenzo Mancinelli

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

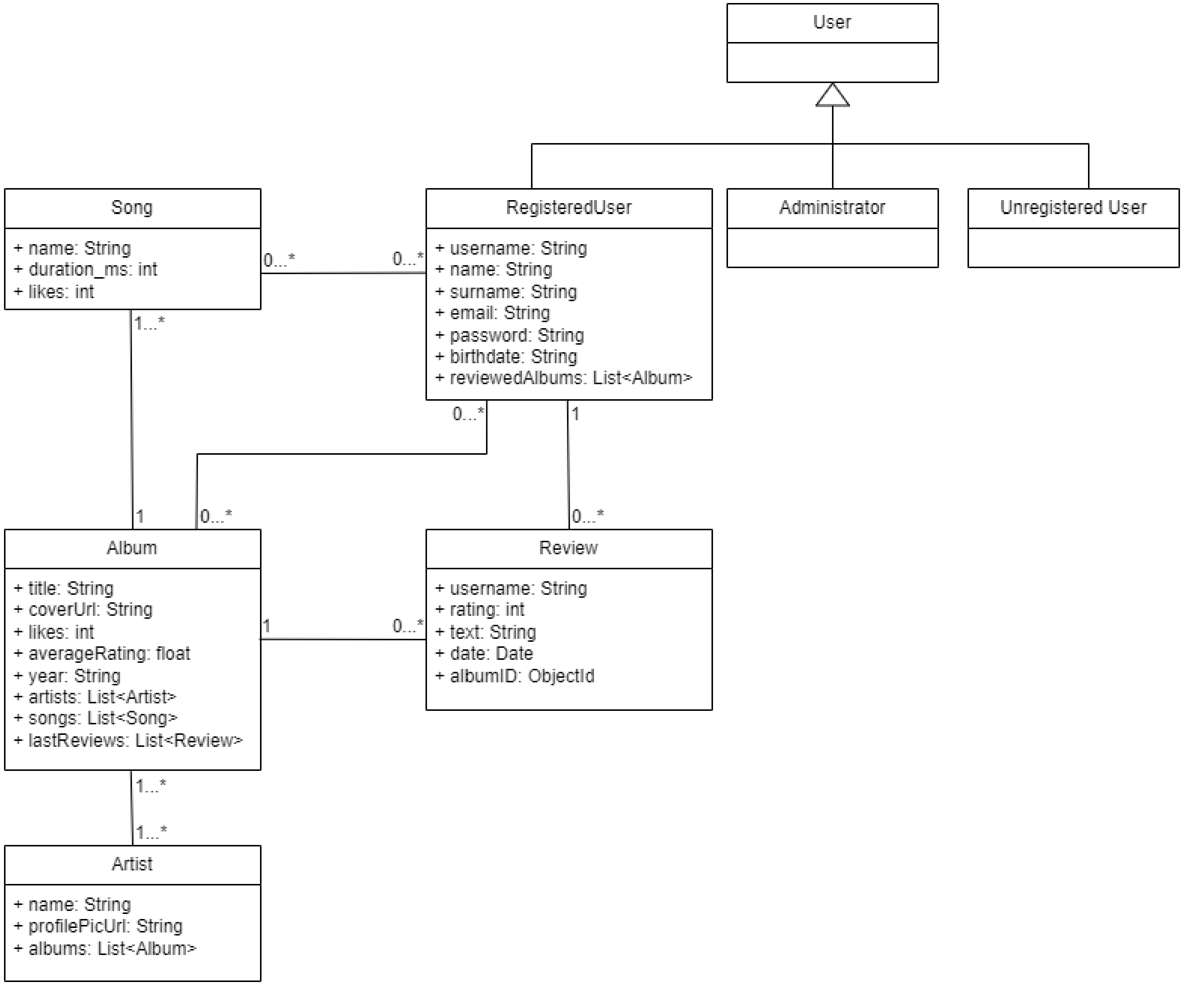
Application Highlights

BeatBuddy merges the world of social networking with a passion for music, highlighting two key features:

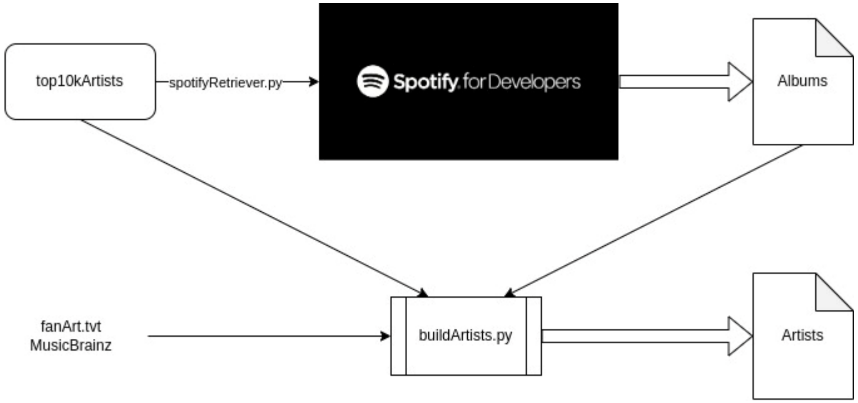
- **Search**: Our users can effortlessly explore albums, songs, and artists. Each search leads to a dedicated page brimming with details such as track listings, related album collections, and reviews from other music lovers.
- **Discover**: We're all about helping you find fresh sounds. The app shines a spotlight on trending tracks and offers personalized recommendations, either tailored to your own taste or influenced by your friends' musical preferences.

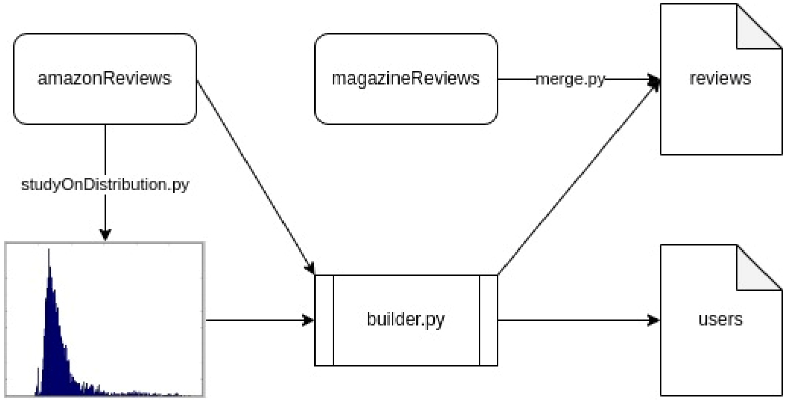
And behind the scenes, our admins keep the rhythm going with powerful analytics, staying in tune with how the app is used and ensuring the best experience for everyone.

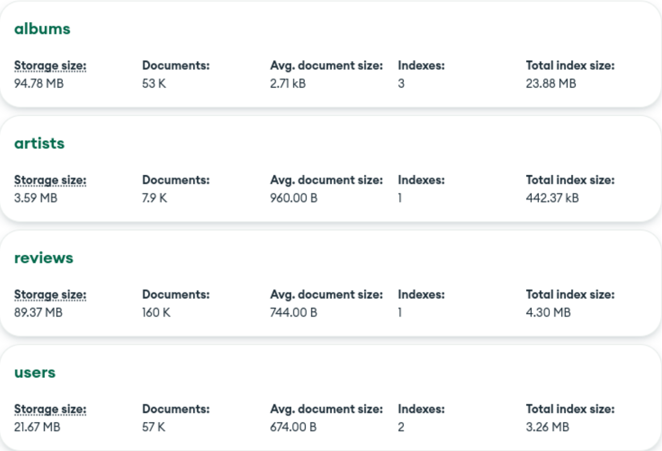
UML Class Diagram



Dataset Description







albums

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
94.78 MB	53 K	2.71 kB	3	23.88 MB

artists

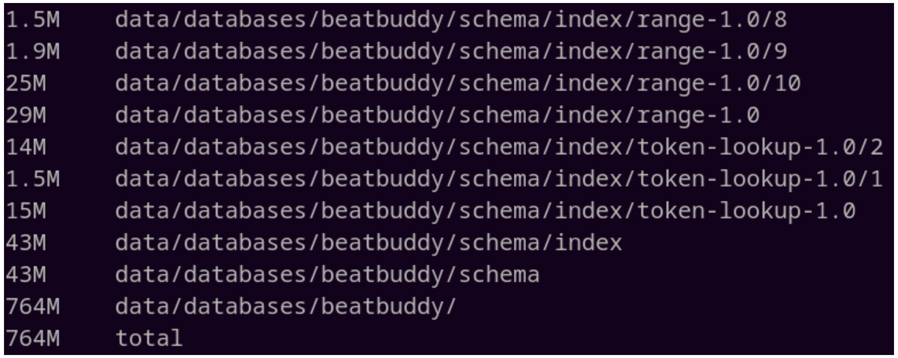
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
3.59 MB	7.9 K	960.00 B	1	442.37 kB

reviews

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
89.37 MB	160 K	744.00 B	1	4.30 MB

users

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
21.67 MB	57 K	674.00 B	2	3.26 MB

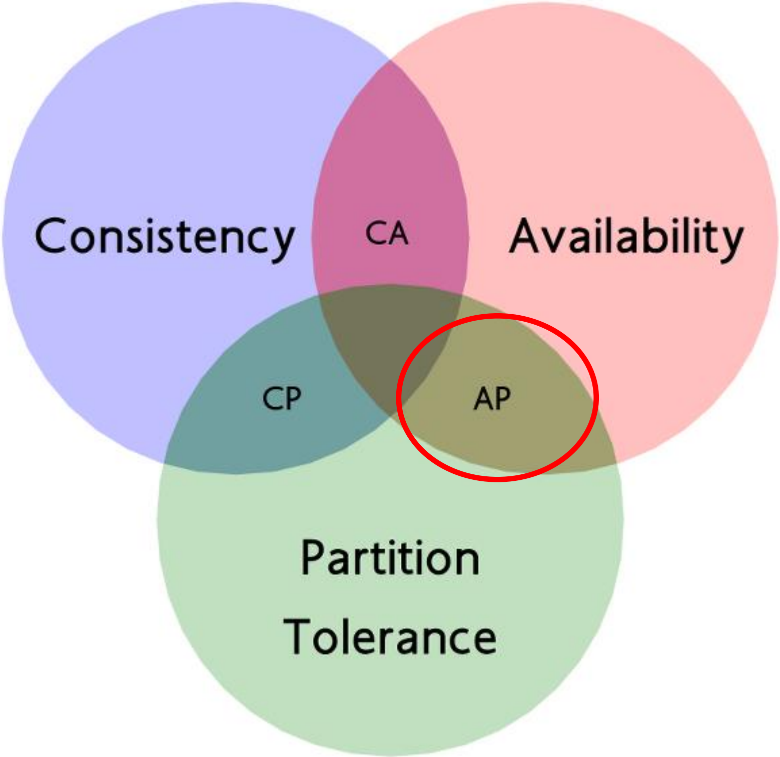


```
1.5M    data/databases/beatbuddy/schema/index/range-1.0/8
1.9M    data/databases/beatbuddy/schema/index/range-1.0/9
25M     data/databases/beatbuddy/schema/index/range-1.0/10
29M     data/databases/beatbuddy/schema/index/range-1.0
14M     data/databases/beatbuddy/schema/index/token-lookup-1.0/2
1.5M    data/databases/beatbuddy/schema/index/token-lookup-1.0/1
15M     data/databases/beatbuddy/schema/index/token-lookup-1.0
43M     data/databases/beatbuddy/schema/index
43M     data/databases/beatbuddy/schema
764M    data/databases/beatbuddy/
764M    total
```

Non-Functional Requirements

- Implementation as a web application.
- Minimization of single points of failure.
- Emphasis on high availability, even with occasionally outdated data.
- Use of Object-Oriented Programming languages for code development.
- Tolerance for data loss.
- Encryption of user passwords.

CAP Theorem Issue



1. The application needs to be available 24/7

2. We need to avoid a single point of failure

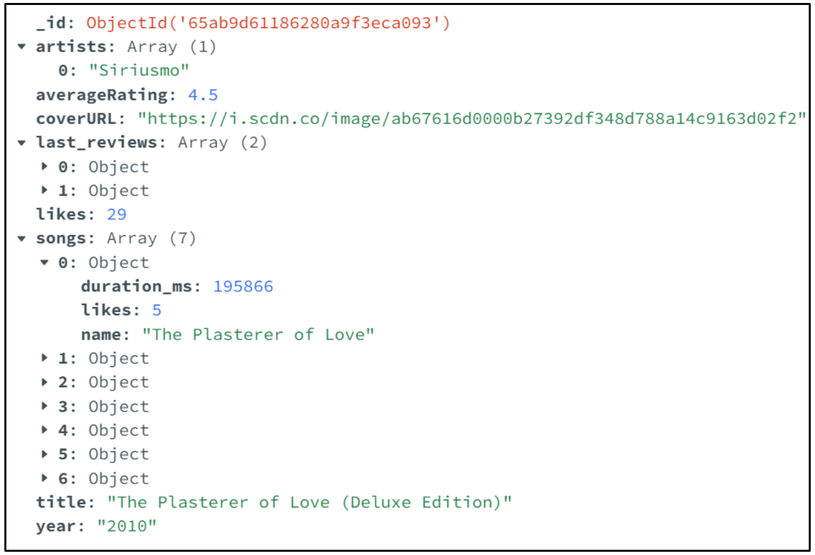
3. We accept potentially out-of-date version of data

↓

WE STAY ON THE AP SIDE OF THE TRIANGLE

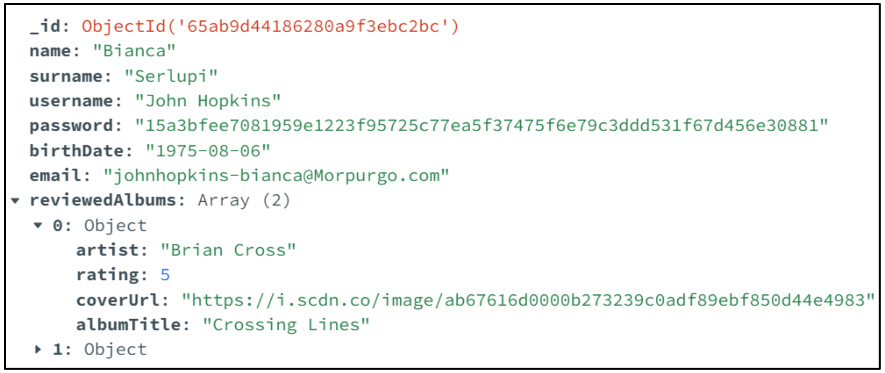
MongoDB design

Album's collection



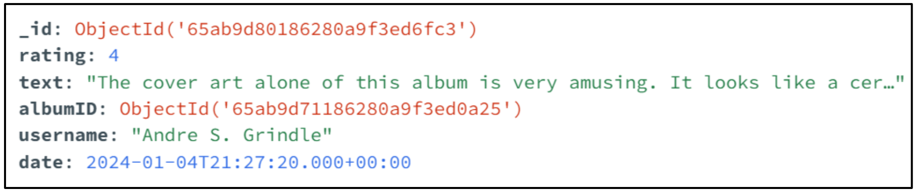
```
  _id: ObjectId('65ab9d61186280a9f3eca093')
▾ artists: Array (1)
    0: "Siriusmo"
  averageRating: 4.5
  coverURL: "https://i.scdn.co/image/ab67616d0000b27392df348d788a14c9163d02f2"
▾ last_reviews: Array (2)
  ▸ 0: Object
  ▸ 1: Object
  likes: 29
▾ songs: Array (7)
  ▾ 0: Object
      duration_ms: 195866
      likes: 5
      name: "The Plasterer of Love"
  ▸ 1: Object
  ▸ 2: Object
  ▸ 3: Object
  ▸ 4: Object
  ▸ 5: Object
  ▸ 6: Object
  title: "The Plasterer of Love (Deluxe Edition)"
  year: "2010"
```

User's collection



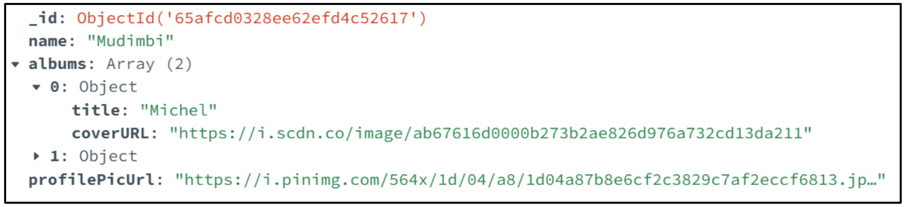
```
  _id: ObjectId('65ab9d44186280a9f3ebc2bc')
  name: "Bianca"
  surname: "Serlupi"
  username: "John Hopkins"
  password: "15a3bfee7081959e1223f95725c77ea5f37475f6e79c3ddd531f67d456e30881"
  birthDate: "1975-08-06"
  email: "johnhopkins-bianca@Morpurgo.com"
▾ reviewedAlbums: Array (2)
  ▾ 0: Object
      artist: "Brian Cross"
      rating: 5
      coverUrl: "https://i.scdn.co/image/ab67616d0000b273239c0adf89ebf850d44e4983"
      albumTitle: "Crossing Lines"
  ▸ 1: Object
```

Review's collection



```
_id: ObjectId('65ab9d80186280a9f3ed6fc3')
rating: 4
text: "The cover art alone of this album is very amusing. It looks like a cer…"
albumID: ObjectId('65ab9d71186280a9f3ed0a25')
username: "Andre S. Grindle"
date: 2024-01-04T21:27:20.000+00:00
```

Artist's collection



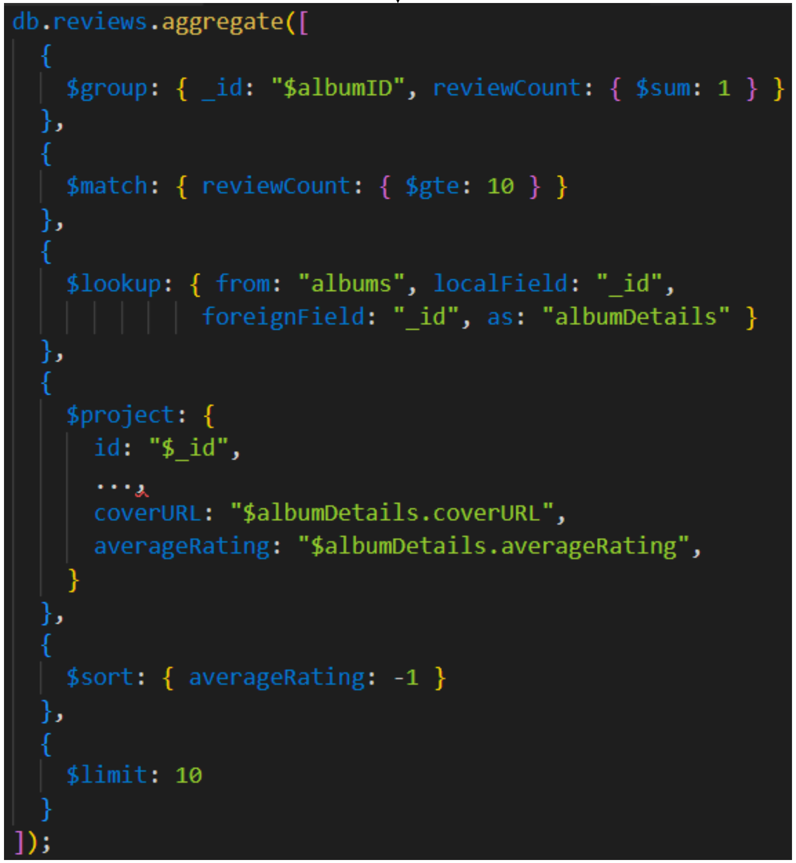
```
  _id: ObjectId('65afcd0328ee62efd4c52617')
  name: "Mudimbi"
▾ albums: Array (2)
  ▾ 0: Object
      title: "Michel"
      coverURL: "https://i.scdn.co/image/ab67616d0000b273b2ae826d976a732cd13da211"
  ▸ 1: Object
  profilePicUrl: "https://i.pinimg.com/564x/1d/04/a8/1d04a87b8e6cf2c3829c7af2eccf6813.jp…"
```

Relevant MongoDB queries

Aggregations:

- **getAlbumsWithMinReviewsByAvgRating_AllTime**

Compiles the ranking of albums with the highest average rating.



```
db.reviews.aggregate([
  {
    $group: { _id: "$albumID", reviewCount: { $sum: 1 } }
  },
  {
    $match: { reviewCount: { $gte: 10 } }
  },
  {
    $lookup: { from: "albums", localField: "_id",
               foreignField: "_id", as: "albumDetails" }
  },
  {
    $project: {
      id: "$_id",
      ...,
      coverURL: "$albumDetails.coverURL",
      averageRating: "$albumDetails.averageRating",
    }
  },
  {
    $sort: { averageRating: -1 }
  },
  {
    $limit: 10
  }
]);
```

- **getSongsByLikes_AllTime**

Creates the list of the most-liked songs of all time

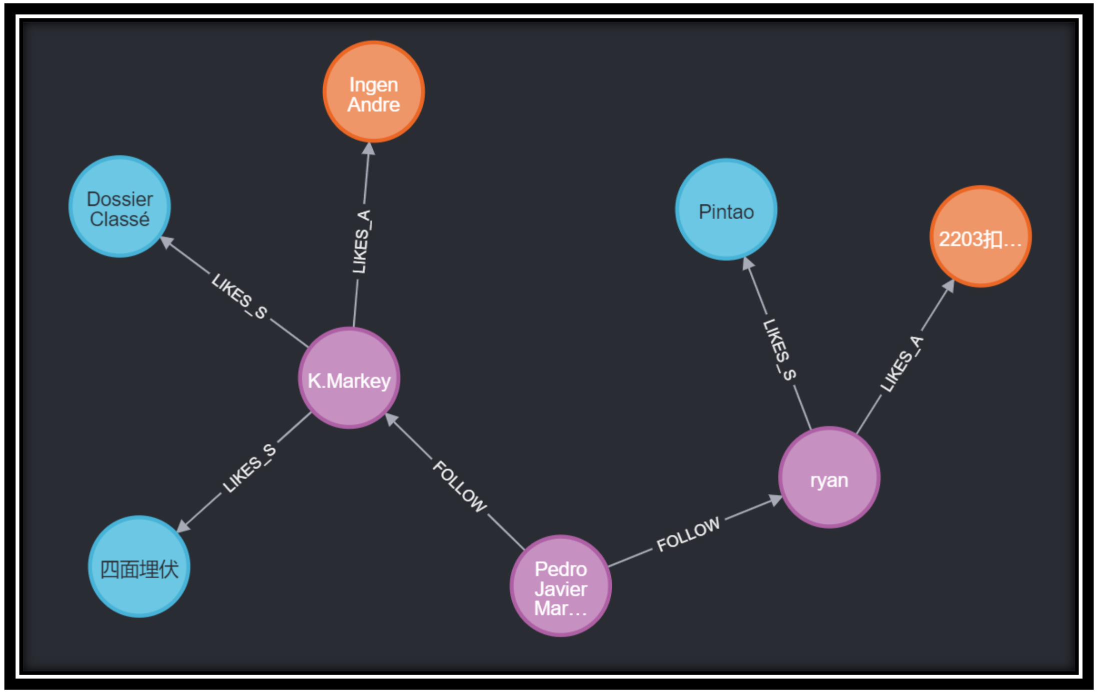
- **getAverageRatingForRecentReviews**

Calculates the average rating for albums that have received a review in the last 24 hours.

Relevant operations:

- Display album detail page
- Add a new review
- Search for a song by substring

Neo4J design



Entities:

- **User** (username)
- **Album** (albumName, artistName, coverURL)
- **Song** (albumName, artistName, coverUrl, songName)

Relationships:

- **FOLLOW** (User-User)
- **LIKES_A** (User-Album)
- **LIKES_S** (User-Song)

Relevant Neo4j queries

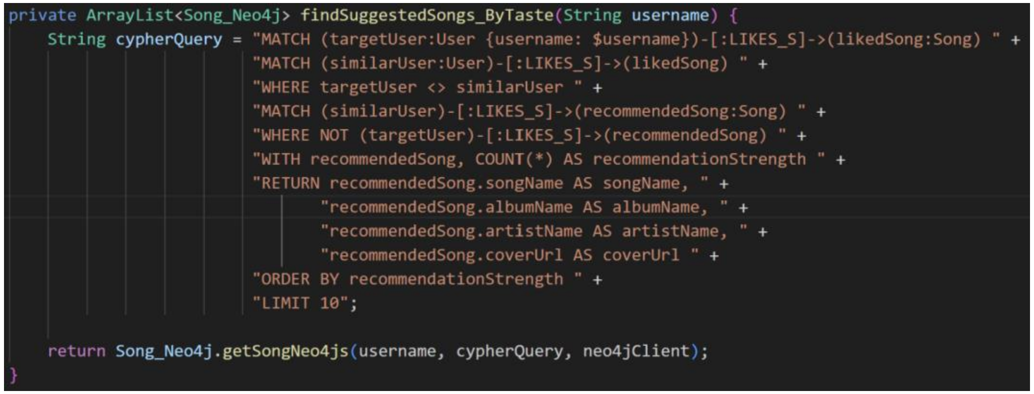
SUGGESTIONS:

Users*:*

- Based on the *users already followed*.

Songs*:*

- Based on *favourite songs among the followed users*.
- **Based on *user's liked songs***

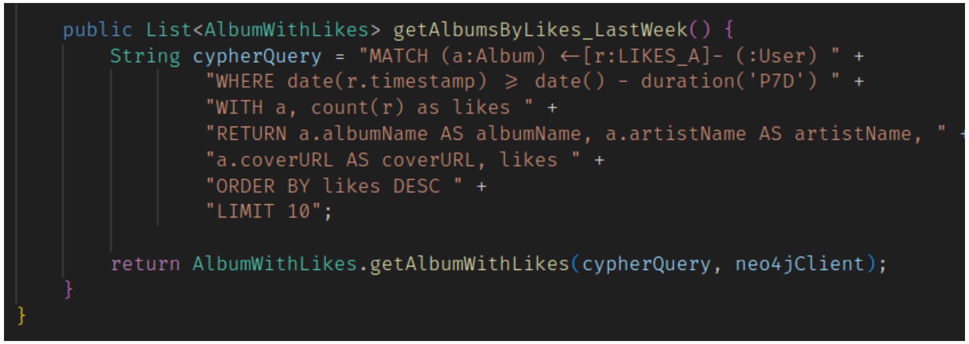


```
private ArrayList<Song_Neo4j> findSuggestedSongs_ByTaste(String username) {
    String cypherQuery = "MATCH (targetUser:User {username: $username})-[:LIKES_S]->(likedSong:Song) " +
                         "MATCH (similarUser:User)-[:LIKES_S]->(likedSong) " +
                         "WHERE targetUser <> similarUser " +
                         "MATCH (similarUser)-[:LIKES_S]->(recommendedSong:Song) " +
                         "WHERE NOT (targetUser)-[:LIKES_S]->(recommendedSong) " +
                         "WITH recommendedSong, COUNT(*) AS recommendationStrength " +
                         "RETURN recommendedSong.songName AS songName, " +
                                "recommendedSong.albumName AS albumName, " +
                                "recommendedSong.artistName AS artistName, " +
                                "recommendedSong.coverUrl AS coverUrl " +
                         "ORDER BY recommendationStrength " +
                         "LIMIT 10";

    return Song_Neo4j.getSongNeo4js(username, cypherQuery, neo4jClient);
}
```

STATISTISC:

- Count all-time likes (Songs and Albums)
- **Count likes in the past week** (Songs and **Albums**)
- Count all-time likes for entities that received likes in the past day



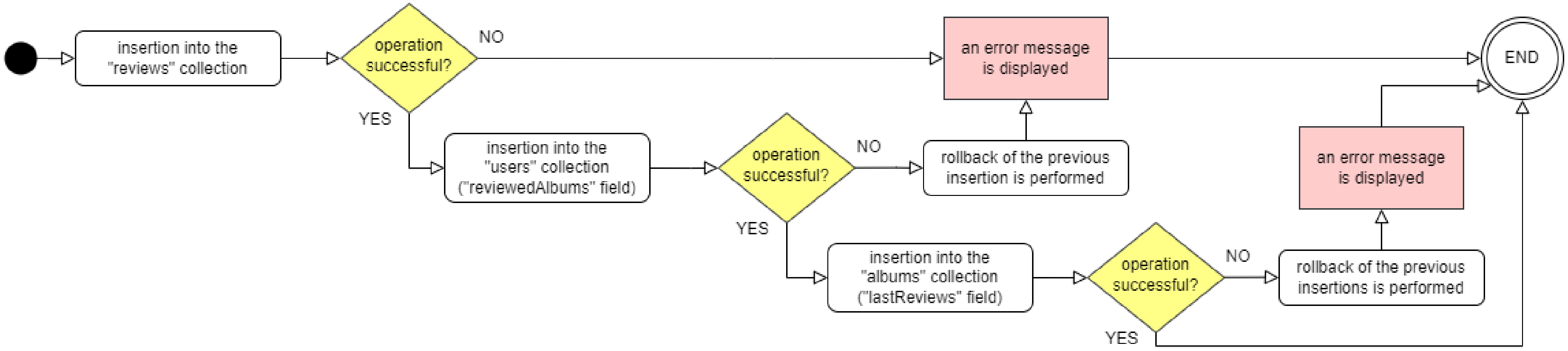
```
    public List<AlbumWithLikes> getAlbumsByLikes_LastWeek() {
        String cypherQuery = "MATCH (a:Album) ←[r:LIKES_A]- (:User) " +
                "WHERE date(r.timestamp) ⩾ date() - duration('P7D') " +
                "WITH a, count(r) as likes " +
                "RETURN a.albumName AS albumName, a.artistName AS artistName, " +
                "a.coverURL AS coverURL, likes " +
                "ORDER BY likes DESC " +
                "LIMIT 10";

        return AlbumWithLikes.getAlbumWithLikes(cypherQuery, neo4jClient);
    }
}
```

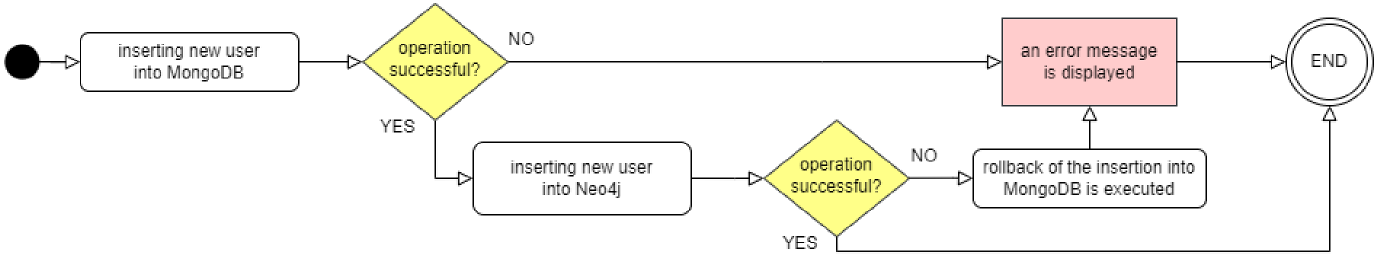
Data Consistency

We use Spring Boot's **`@Transactional`** to ensure consistency in multiple insertions. If an insertion fails, previous ones in the transaction are automatically rolled back. Let's see how it works in action.

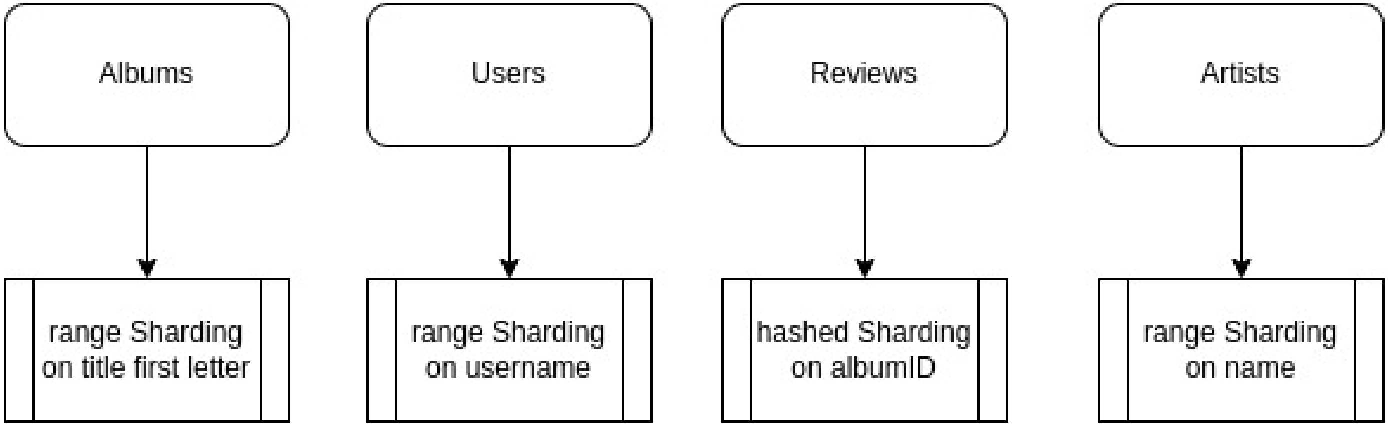
- example 1: Insertion of a new review → three insertions in MongoDB


insertion into the "reviews" collection
operation successful?
NO
YES
an error message is displayed
END
insertion into the "users" collection ("reviewedAlbums" field)
operation successful?
NO
YES
rollback of the previous insertion is performed
an error message is displayed
insertion into the "albums" collection ("lastReviews" field)
operation successful?
NO
YES
rollback of the previous insertions is performed

- example 2: Registration of a new user → one insertion in MongoDB and one in Neo4j


inserting new user into MongoDB
operation successful?
NO
YES
an error message is displayed
END
inserting new user into Neo4j
operation successful?
NO
YES
rollback of the insertion into MongoDB is executed

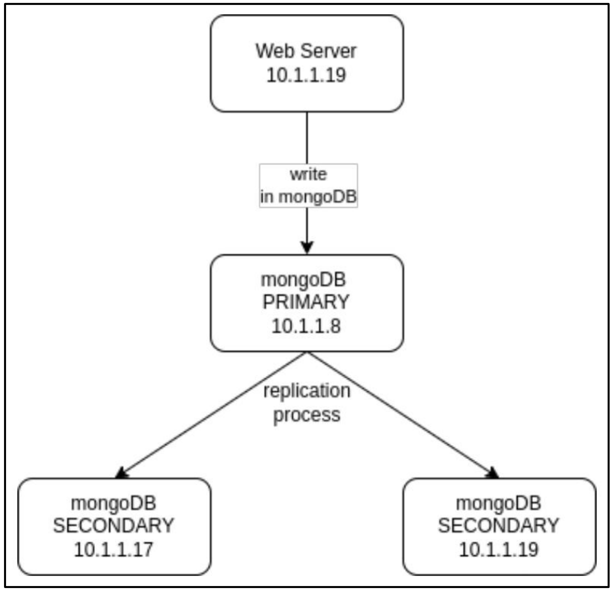
Data Sharding Proposal


Albums
Users
Reviews
Artists
range Sharding on title first letter
range Sharding on username
hashed Sharding on albumID
range Sharding on name

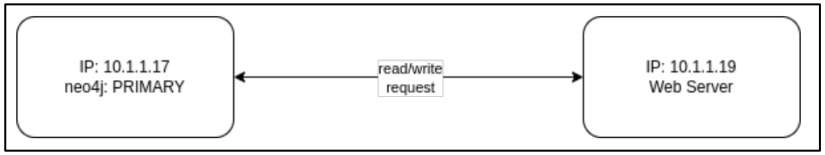
Software and Hardware Architecture

MongoDB

Write Concern = 1



Neo4J



mongoDB neo4j

spring BOOT

Java python

Thymeleaf

HTML CSS JS