Large-Scale and Multi-Structured Databases

LargeB&B

Grillo Alessio
Sfar Omar
Rotelli Luca

Application Highlights

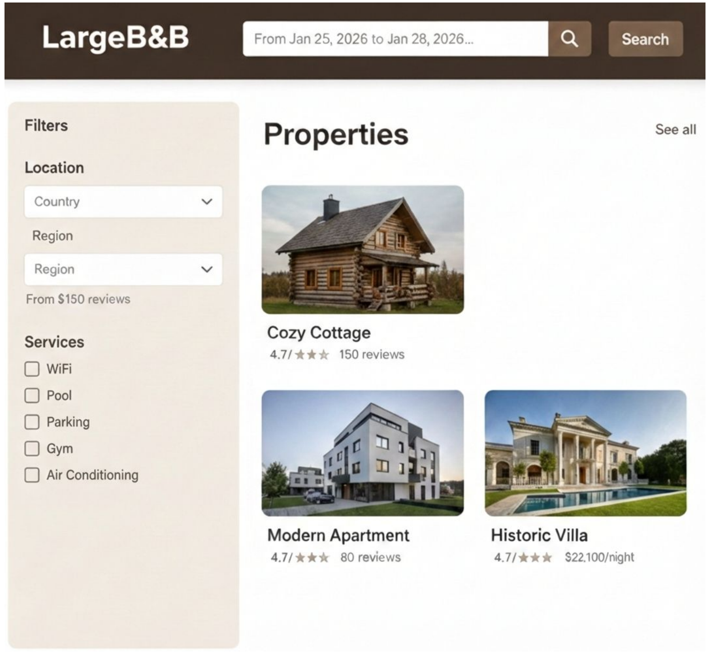
LargeB&B *is a web application that acts as an intermediary platform for booking accommodations. It enables travelers to search, compare, and book hotels, bed & breakfasts, vacation rentals, apartments, and other types of lodging. At the same time, the platform provides property owners and managers with the opportunity to promote their accommodations to a global audience, monitor bookings, and access detailed analytics on the performance of their properties.*

The ***main features*** *of LargeB&B include:*

- *Advanced search and filtering of available rooms based on location and review ratings.*
- *The ability for users to write and read reviews to help other travelers make informed decisions.*
- *Real-time chat communication between customers and property managers, both before and after booking.*
- *Access to analytics and performance insights for property owners regarding the accommodations they have published.*

Actors (i)

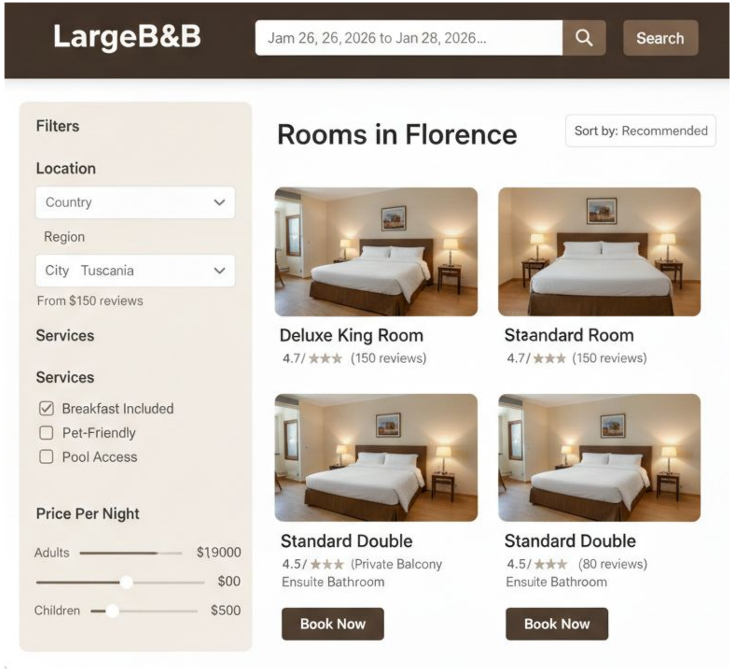
Customer and unregistered user



Page where the user search for properties and filter them.

Actors (ii)

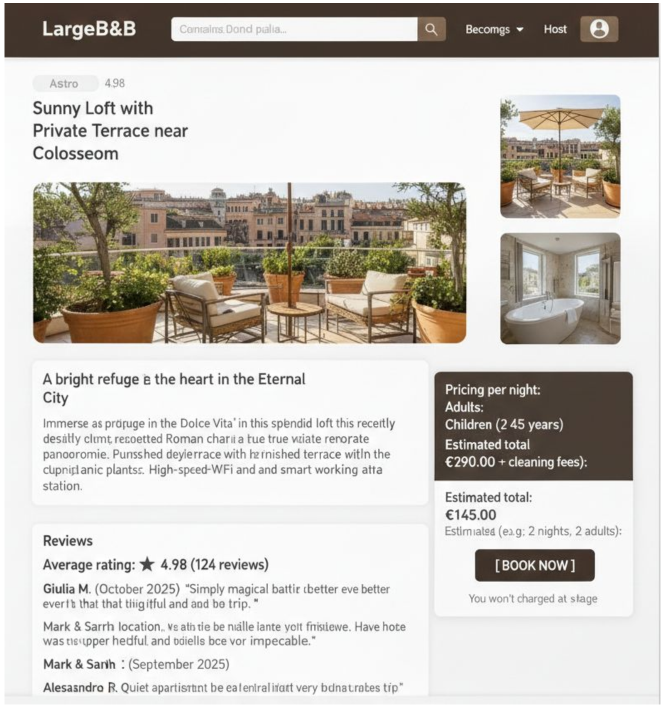
Customer and unregistered user



Page where the user search for rooms and filter them.

Actors (iii)

Customer and unregistered user



Astro 4.98

Sunny Loft with Private Terrace near Colosseom

A bright refuge ɛ the heart in the Eternal City

Immerse as prɑ̈ruge in the Dolce Vita' in this spendid loft this receitly desäitly clımt recoetted Roman charıı a tue true vaiate renorate panooromie. Punsshed deyierrace with hɛrnished terrace witln the clupniɪlanic plantsɛ. High-speed-WFi and and smart working atta station.

Reviews

Average rating: ★ 4.98 (124 reviews)

Giulia M. (October 2025) "Simply magical battir ιbetter eve better evert that that tiiigitful and and be trip. "

Mark & Sarrh location., ve ath tie be nälle lante yolt frristewe. Have hote was tis upper hedful. and bdiells bce vor impecable."

Mark & Sarih : (September 2025)

Alesasndro R. Quiet apartisrtint be ea lenlral irart very bdna trabes tip"

Pricing per night:
Adults:
Children (2 45 years)
Estimated total
€290.00 + cleaning fees):

Estimated total:
€145.00
Estimated (e.g: 2 nights, 2 adults):

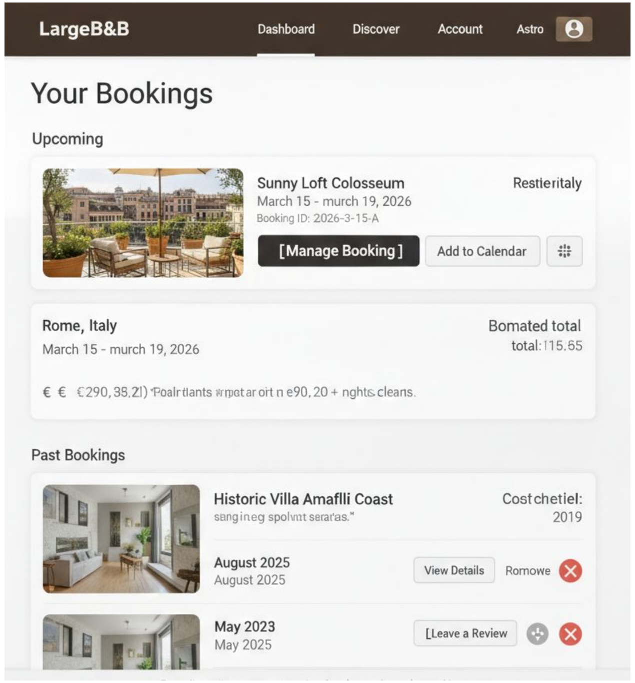
[BOOK NOW]

You won't charged at stage

Page where the user can see the room details.

Actors (iv)

Customer



LargeB&B Dashboard Discover Account Astro

Your Bookings

Upcoming

Sunny Loft Colosseum Restieritaly
March 15 - murch 19, 2026
Booking ID: 2026-3-15-A
[Manage Booking] Add to Calendar

Rome, Italy Bomated total
March 15 - murch 19, 2026 total: 115.65
€ € €290, 38,21) Foalrtlants ırpotar oit n e90, 20 + nghts cleans.

Past Bookings

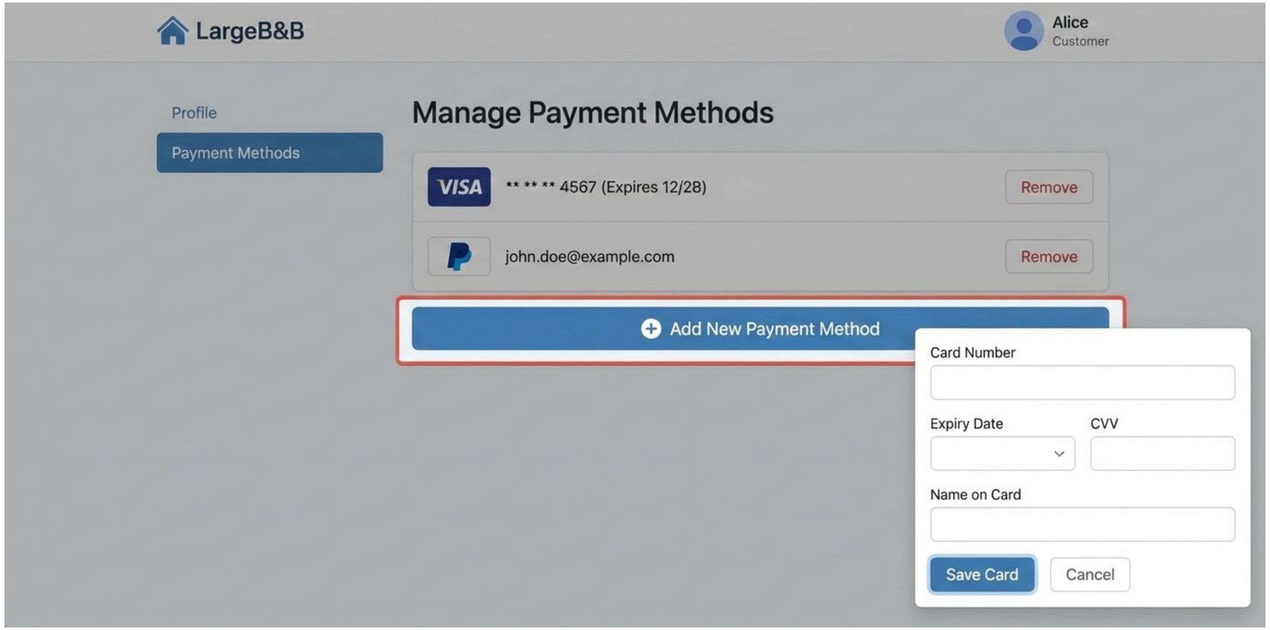
Historic Villa Amaflli Coast Costchetiel: 2019
sangineg spolvnt sera'as."
August 2025
August 2025
View Details Romowe

May 2023
May 2025
[Leave a Review

Page where the customer can see his reservations.

Actors (v)

Customer


LargeB8&B

Alice
Customer

Profile
Payment Methods

Manage Payment Methods

VISA ** ** ** 4567 (Expires 12/28) Remove

john.doe@example.com Remove

Add New Payment Method

Card Number

Expiry Date

CVV

Name on Card

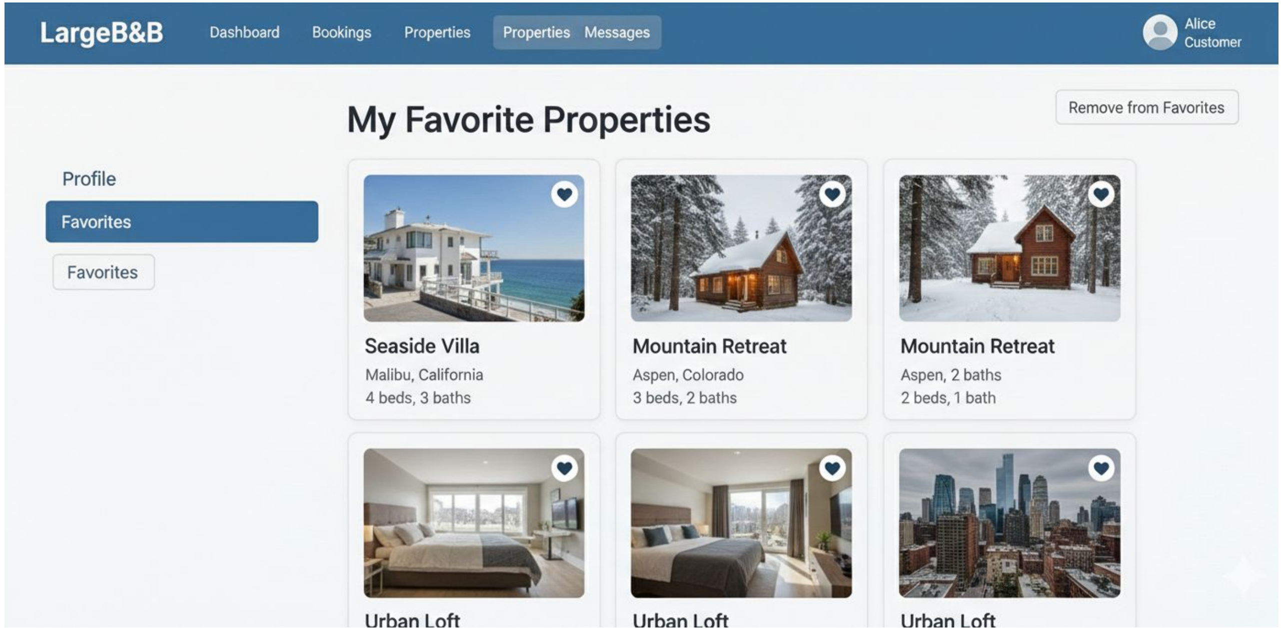
Save Card

Cancel

Page where the customer can manage his payment method.

Actors (vi)

Customer



My Favorite Properties

Seaside Villa
Malibu, California
4 beds, 3 baths

Mountain Retreat
Aspen, Colorado
3 beds, 2 baths

Mountain Retreat
Aspen, 2 baths
2 beds, 1 bath

Urban Loft

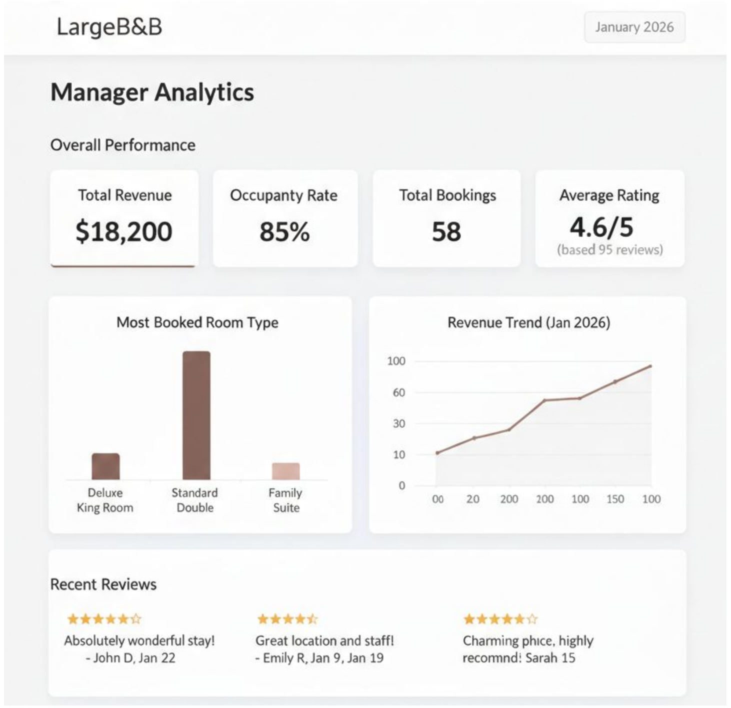
Urban Loft

Urban Loft

Page where the customer can see his favorite properties.

Actors (vii)

Manager



LargeB&B January 2026

Manager Analytics

Overall Performance

Total Revenue	Occupanty Rate	Total Bookings	Average Rating
$18,200	85%	58	4.6/5 (based 95 reviews)

Recent Reviews

★★★★★☆ Absolutely wonderful stay! - John D, Jan 22

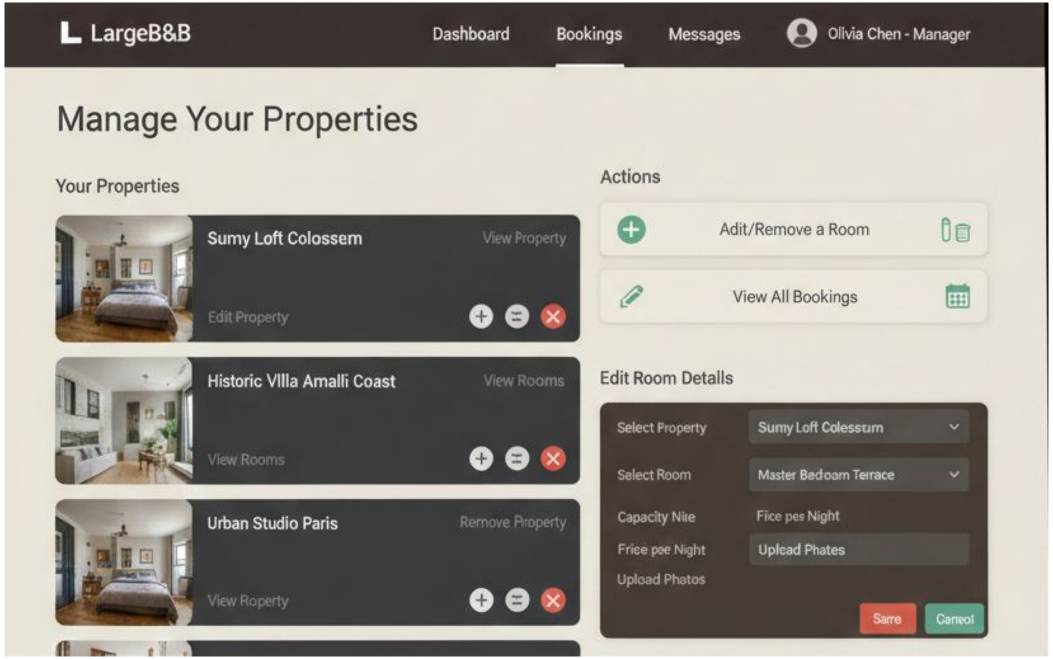
★★★★☆ Great location and staff! - Emily R, Jan 9, Jan 19

★★★★★☆ Charming phice, highly recommnd! Sarah 15

Page where the manager can see his analytics.

Actors (viii)

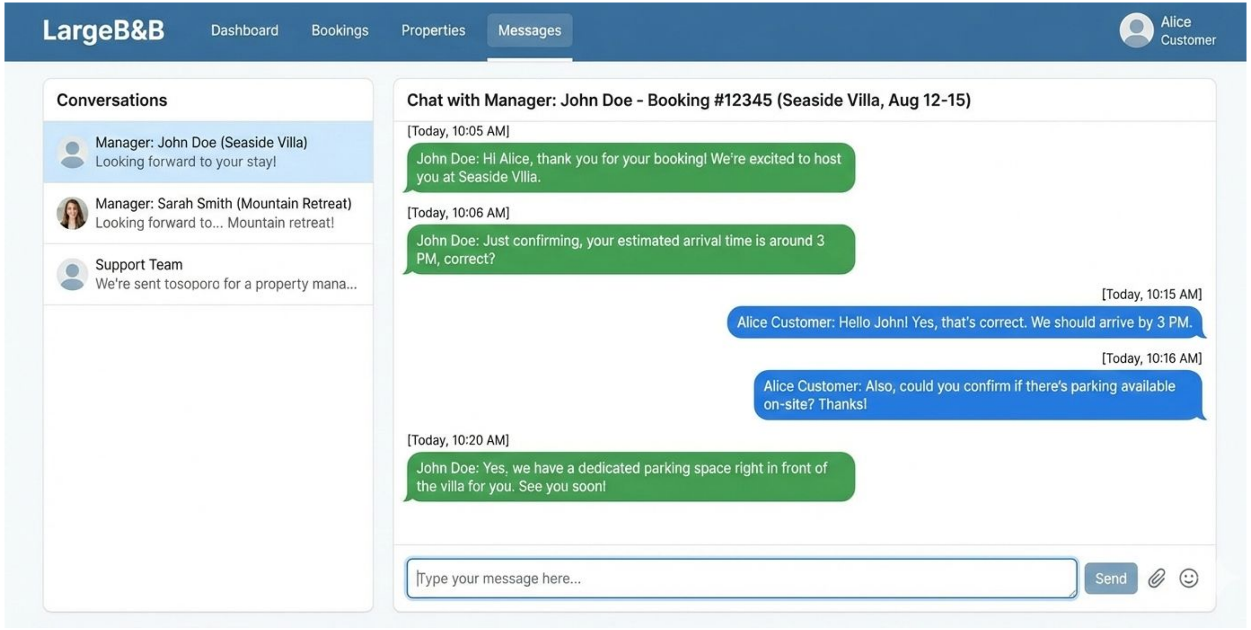
Manager



Page where the manager can manage his properties.

Actors (ix)

Manager and Customer



Conversations

Manager: John Doe (Seaside Villa)
Looking forward to your stay!

Manager: Sarah Smith (Mountain Retreat)
Looking forward to... Mountain retreat!

Support Team
We're sent tosoporo for a property mana...

Chat with Manager: John Doe - Booking #12345 (Seaside Villa, Aug 12-15)

[Today, 10:05 AM]
John Doe: Hi Alice, thank you for your booking! We're excited to host you at Seaside VIlla.

[Today, 10:06 AM]
John Doe: Just confirming, your estimated arrival time is around 3 PM, correct?

[Today, 10:15 AM]
Alice Customer: Hello John! Yes, that's correct. We should arrive by 3 PM.

[Today, 10:16 AM]
Alice Customer: Also, could you confirm if there's parking available on-site? Thanks!

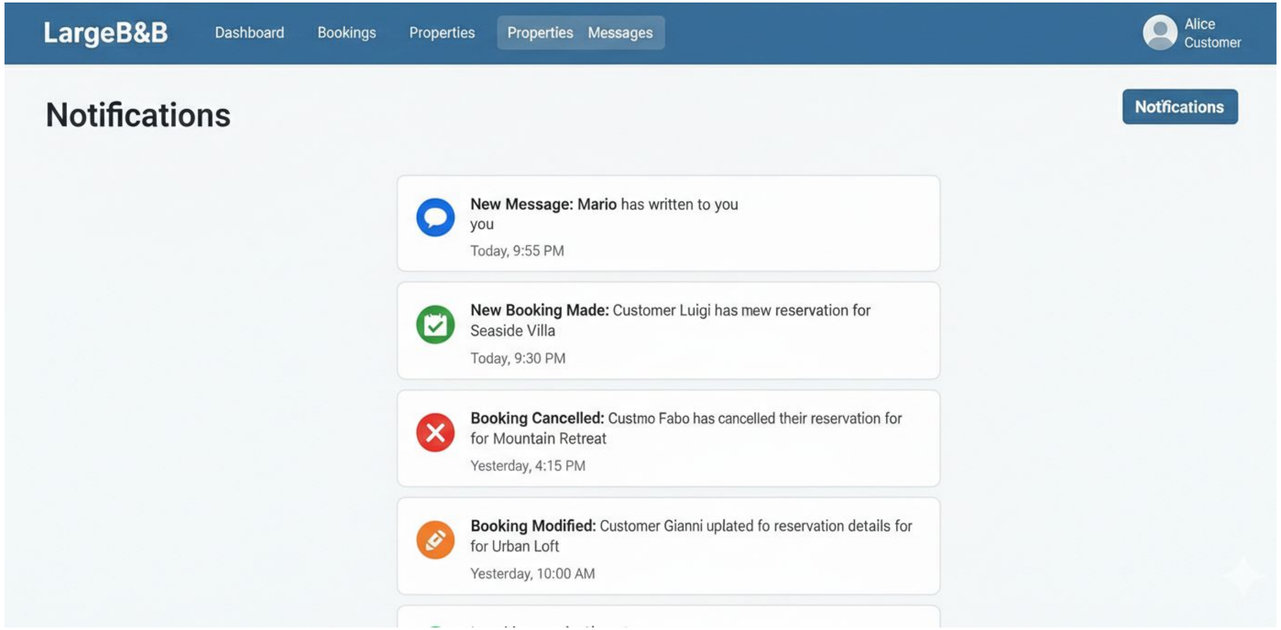
[Today, 10:20 AM]
John Doe: Yes, we have a dedicated parking space right in front of the villa for you. See you soon!

Type your message here...

Page where the manager or customer can send/receive messages.

Actors (x)

Manager and Customer



Notifications

New Message: **Mario** has written to you
you
Today, 9:55 PM

New Booking Made: Customer Luigi has mew reservation for Seaside Villa
Today, 9:30 PM

Booking Cancelled: Custmo Fabo has cancelled their reservation for for Mountain Retreat
Yesterday, 4:15 PM

Booking Modified: Customer Gianni uplated fo reservation details for for Urban Loft
Yesterday, 10:00 AM

Page where the manager or customer can see his notifications (booking notification are only for Manager).

Dataset Description

Sources: *Inside AirBnB, Faker*

Description: *Dataset concerning properties and rooms information, reviews, notifications, messages between customer and manager, and reservations information.*

Volume: *170MB*

Variety: Frequently changing data: real-time room availability, new bookings, and daily reviews.

Dataset Description

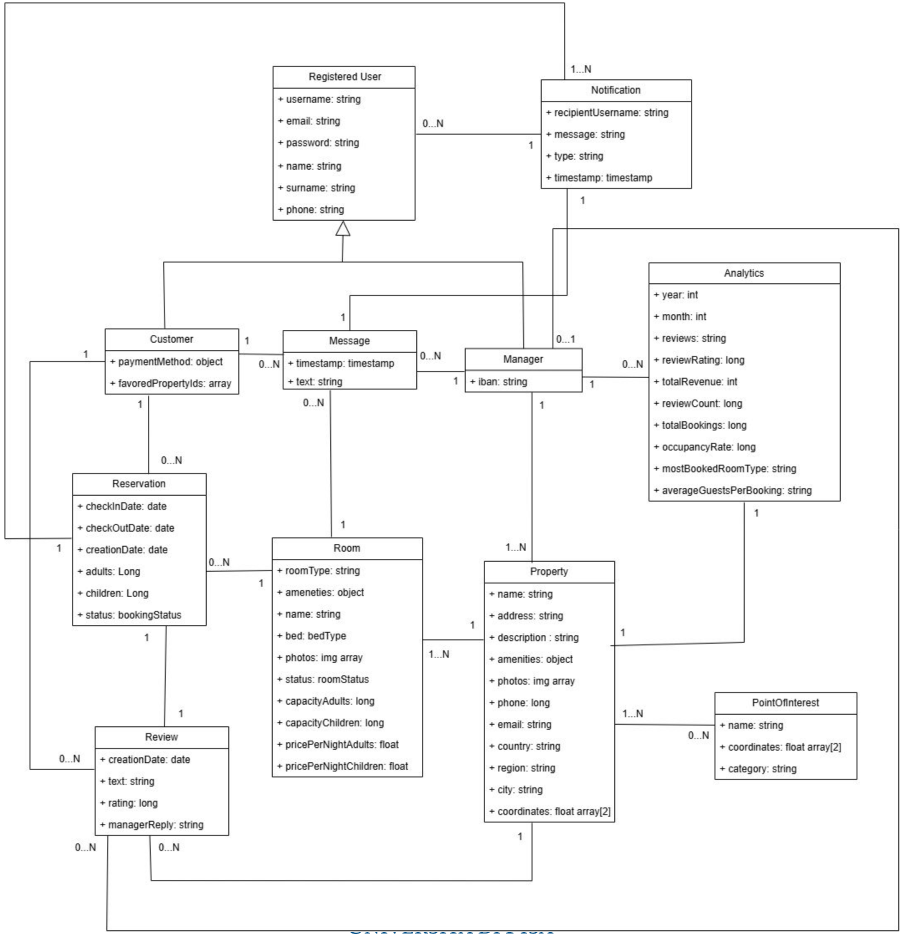
Velocity/Variability:

*The system handles detailed **room and property descriptions**, along with essential booking data such as **guest counts** (adults and children) and **check-in/check-out dates**. It features a comprehensive review system that includes **multi-criteria ratings** for stays and allows for **manager responses**. Additionally, a real-time notification system alerts users to **incoming messages** and provides managers with exclusive updates regarding the **reservation status** of their properties.*

Pre-processing:

***Data Cleaning:** Used Python (Pandas) to clean InsideAirbnb CSVs (handling missing values, removing inactive listings).*
***Data Fusion:** Merging real property data with synthetic Manager profiles.*
***Transformation:** Converting flat CSV records into hierarchical MongoDB Documents (Embedding Rooms/Reviews).*
***Geo-Coding:** Transforming lat/con columns into GeoJSON points objects for spatial indexing.*

UML Design Class Diagram



Application non-functional requirements

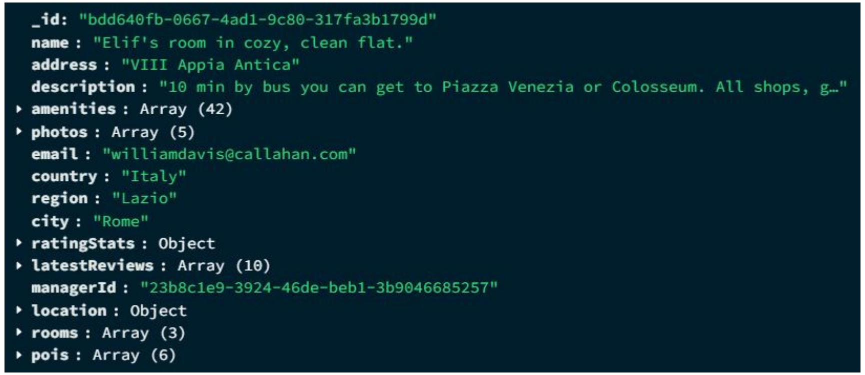
- *Passwords of registered users must be securely encrypted.*
- *The system must provide notifications via application.*
- *The results produced by the system must be consistent.*
- *The system must ensure high data availability.*
- *The system must manage data storage using Document DB, Graph DB, and Key-Value DB.*
- *The system should be able to scale to accommodate an increase in user traffic.*

Application non-functional requirements

- *The DBMS technologies used must be Neo4j, MongoDB, and Redis.*
- *The business logic and data access layers must be implemented using the Spring framework in Java.*
- *The codebase must be easily maintainable, readable, and updatable.*
- *The system must use a document database to store information permanently.*
- *The system must use a key-value database to temporarily store information, manage cache, locks, and user sessions.*

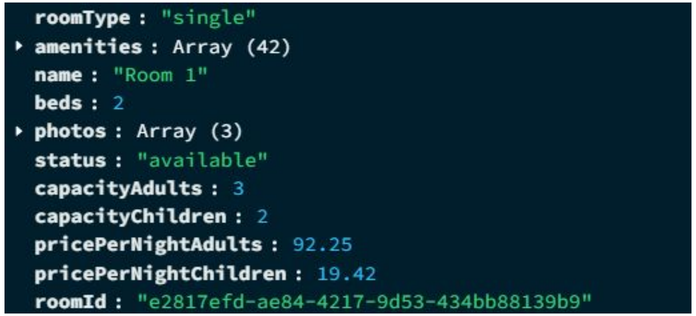
Document DB Design

Properties



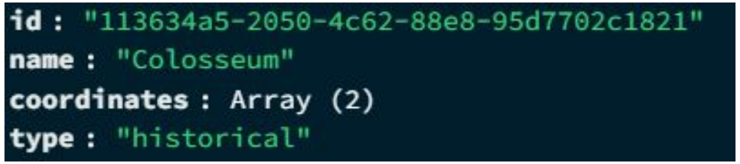
```
  _id: "bdd640fb-0667-4ad1-9c80-317fa3b1799d"
  name : "Elif's room in cozy, clean flat."
  address : "VIII Appia Antica"
  description : "10 min by bus you can get to Piazza Venezia or Colosseum. All shops, g…"
▸ amenities : Array (42)
▸ photos : Array (5)
  email : "williamdavis@callahan.com"
  country : "Italy"
  region : "Lazio"
  city : "Rome"
▸ ratingStats : Object
▸ latestReviews : Array (10)
  managerId : "23b8c1e9-3924-46de-beb1-3b9046685257"
▸ location : Object
▸ rooms : Array (3)
▸ pois : Array (6)
```

Rooms (embedded)



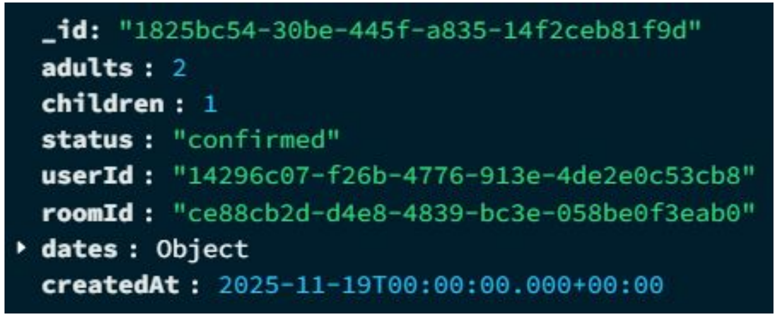
```
  roomType : "single"
▸ amenities : Array (42)
  name : "Room 1"
  beds : 2
▸ photos : Array (3)
  status : "available"
  capacityAdults : 3
  capacityChildren : 2
  pricePerNightAdults : 92.25
  pricePerNightChildren : 19.42
  roomId : "e2817efd-ae84-4217-9d53-434bb88139b9"
```

Pois (embedded)



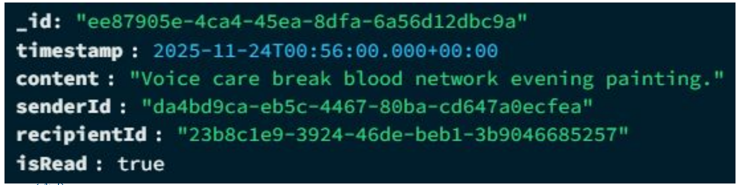
```
id : "113634a5-2050-4c62-88e8-95d7702c1821"
name : "Colosseum"
coordinates : Array (2)
type : "historical"
```

Reservations



```
  _id: "1825bc54-30be-445f-a835-14f2ceb81f9d"
  adults : 2
  children : 1
  status : "confirmed"
  userId : "14296c07-f26b-4776-913e-4de2e0c53cb8"
  roomId : "ce88cb2d-d4e8-4839-bc3e-058be0f3eab0"
▸ dates : Object
  createdAt : 2025-11-19T00:00:00.000+00:00
```

Messages



```
_id: "ee87905e-4ca4-45ea-8dfa-6a56d12dbc9a"
timestamp : 2025-11-24T00:56:00.000+00:00
content : "Voice care break blood network evening painting."
senderId : "da4bd9ca-eb5c-4467-80ba-cd647a0ecfea"
recipientId : "23b8c1e9-3924-46de-beb1-3b9046685257"
isRead : true
```

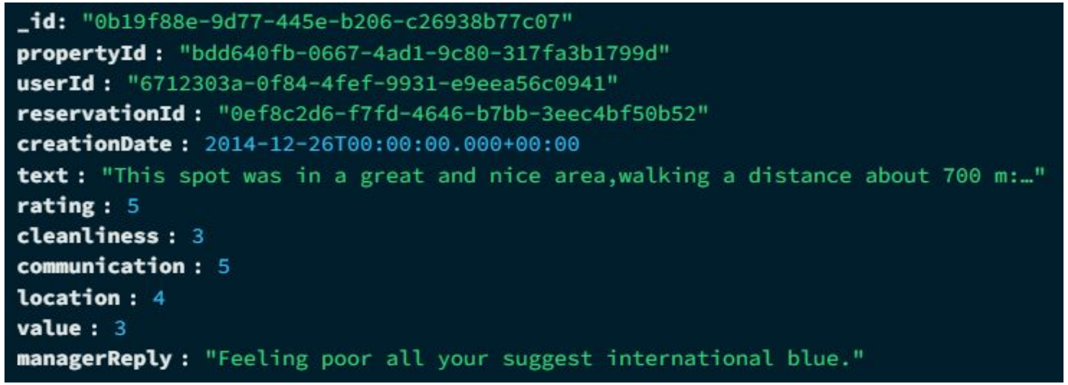
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

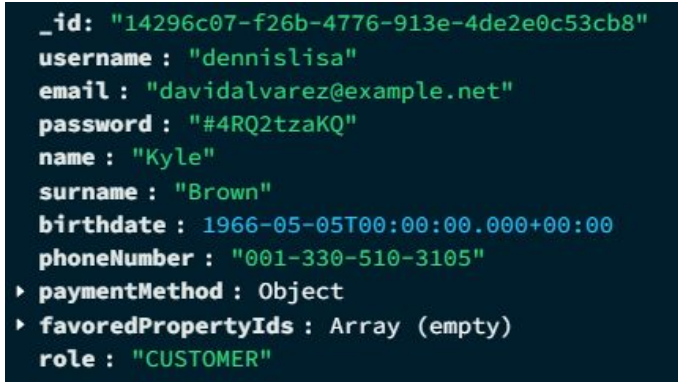
Document DB Design

Reviews



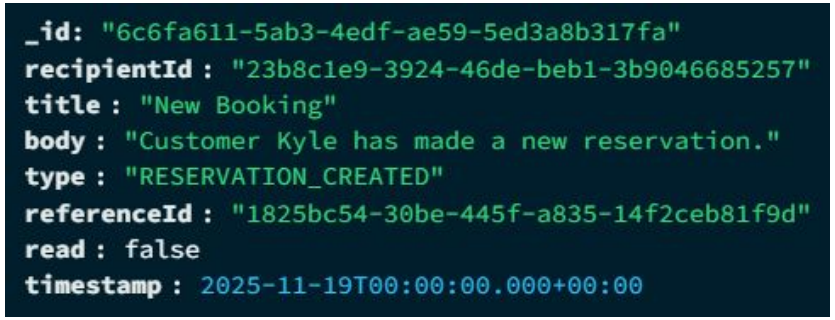
```
_id: "0b19f88e-9d77-445e-b206-c26938b77c07"
propertyId : "bdd640fb-0667-4ad1-9c80-317fa3b1799d"
userId : "6712303a-0f84-4fef-9931-e9eea56c0941"
reservationId : "0ef8c2d6-f7fd-4646-b7bb-3eec4bf50b52"
creationDate : 2014-12-26T00:00:00.000+00:00
text : "This spot was in a great and nice area,walking a distance about 700 m:…"
rating : 5
cleanliness : 3
communication : 5
location : 4
value : 3
managerReply : "Feeling poor all your suggest international blue."
```

Users (Customer)



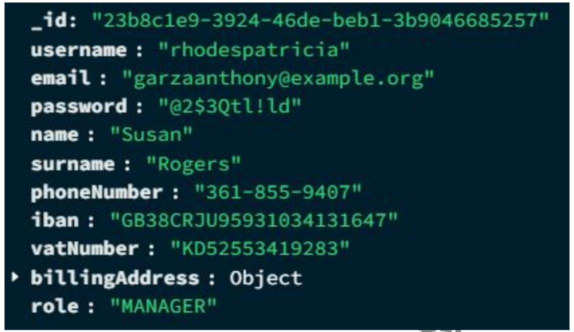
```
  _id: "14296c07-f26b-4776-913e-4de2e0c53cb8"
  username : "dennislisa"
  email : "davidalvarez@example.net"
  password : "#4RQ2tzaKQ"
  name : "Kyle"
  surname : "Brown"
  birthdate : 1966-05-05T00:00:00.000+00:00
  phoneNumber : "001-330-510-3105"
▸ paymentMethod : Object
▸ favoredPropertyIds : Array (empty)
  role : "CUSTOMER"
```

Notifications



```
_id: "6c6fa611-5ab3-4edf-ae59-5ed3a8b317fa"
recipientId : "23b8c1e9-3924-46de-beb1-3b9046685257"
title : "New Booking"
body : "Customer Kyle has made a new reservation."
type : "RESERVATION_CREATED"
referenceId : "1825bc54-30be-445f-a835-14f2ceb81f9d"
read : false
timestamp : 2025-11-19T00:00:00.000+00:00
```

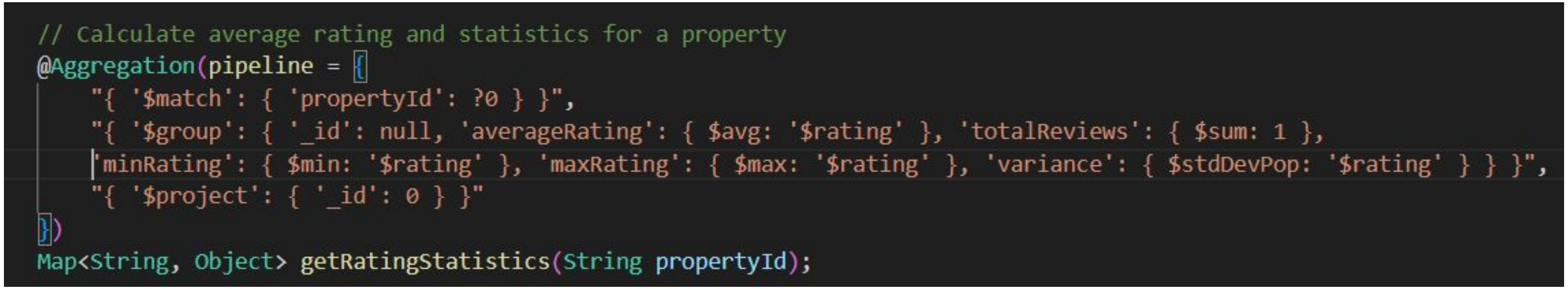
Users (Manager)



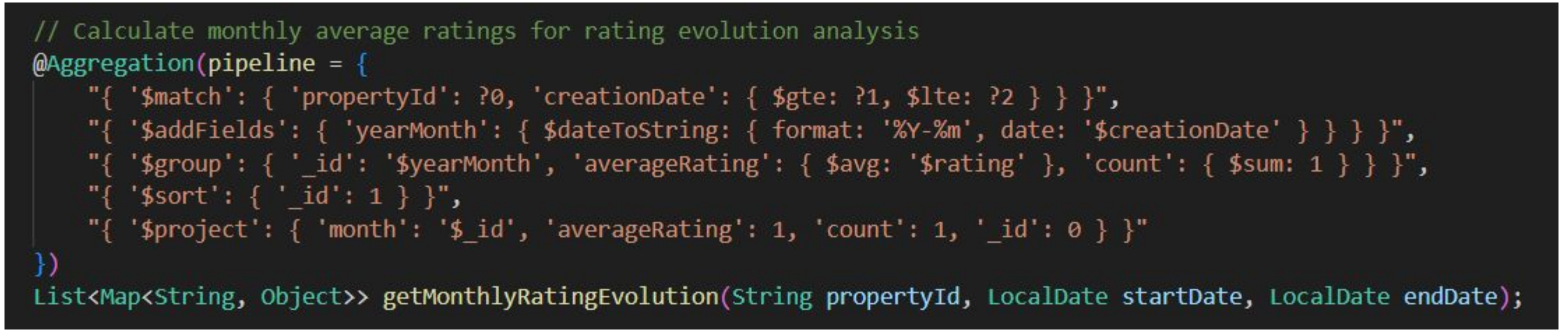
```
  _id: "23b8c1e9-3924-46de-beb1-3b9046685257"
  username : "rhodespatricia"
  email : "garzaanthony@example.org"
  password : "@2$3Qtl!ld"
  name : "Susan"
  surname : "Rogers"
  phoneNumber : "361-855-9407"
  iban : "GB38CRJU95931034131647"
  vatNumber : "KD52553419283"
▸ billingAddress : Object
  role : "MANAGER"
```

Document DB Design

Aggregation Queries Benefiting from Design



```java
// Calculate average rating and statistics for a property
@Aggregation(pipeline = {
    "{ '$match': { 'propertyId': ?0 } }",
    "{ '$group': { '_id': null, 'averageRating': { $avg: '$rating' }, 'totalReviews': { $sum: 1 },
    'minRating': { $min: '$rating' }, 'maxRating': { $max: '$rating' }, 'variance': { $stdDevPop: '$rating' } } }",
    "{ '$project': { '_id': 0 } }"
})
Map<String, Object> getRatingStatistics(String propertyId);
```



```java
// Calculate monthly average ratings for rating evolution analysis
@Aggregation(pipeline = {
    "{ '$match': { 'propertyId': ?0, 'creationDate': { $gte: ?1, $lte: ?2 } } }",
    "{ '$addFields': { 'yearMonth': { $dateToString: { format: '%Y-%m', date: '$creationDate' } } } }",
    "{ '$group': { '_id': '$yearMonth', 'averageRating': { $avg: '$rating' }, 'count': { $sum: 1 } } }",
    "{ '$sort': { '_id': 1 } }",
    "{ '$project': { 'month': '$_id', 'averageRating': 1, 'count': 1, '_id': 0 } }"
})
List<Map<String, Object>> getMonthlyRatingEvolution(String propertyId, LocalDate startDate, LocalDate endDate);
```

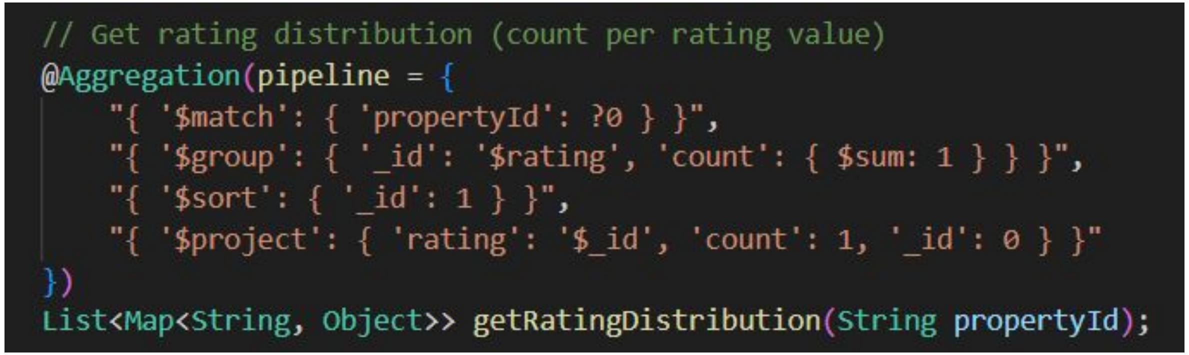
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

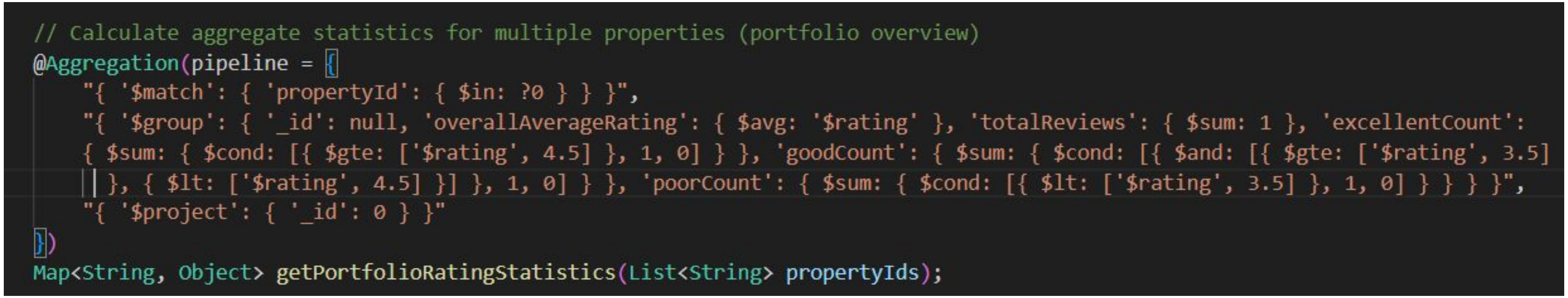
CROSSLAB Innovation for industry 4.0

Document DB Design

Aggregation Queries Benefiting from Design



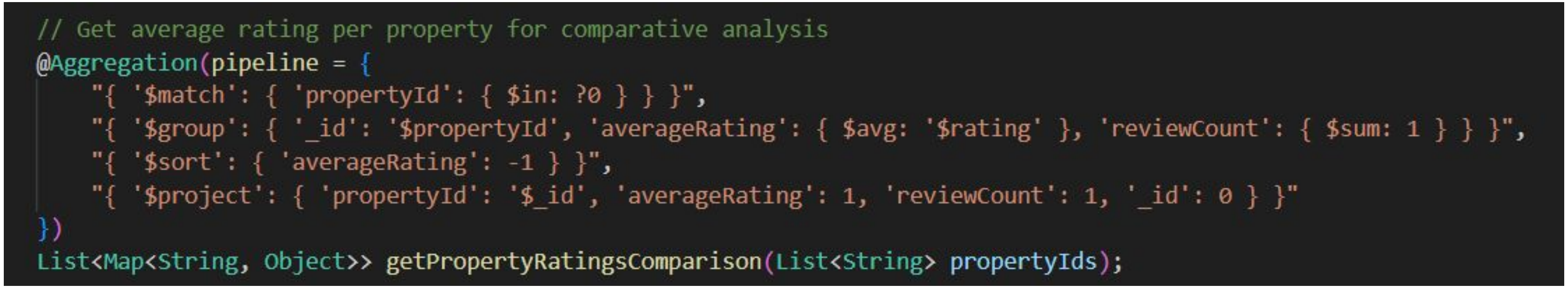
```
// Get rating distribution (count per rating value)
@Aggregation(pipeline = {
    "{ '$match': { 'propertyId': ?0 } }",
    "{ '$group': { '_id': '$rating', 'count': { $sum: 1 } } }",
    "{ '$sort': { '_id': 1 } }",
    "{ '$project': { 'rating': '$_id', 'count': 1, '_id': 0 } }"
})
List<Map<String, Object>> getRatingDistribution(String propertyId);
```



```
// Calculate aggregate statistics for multiple properties (portfolio overview)
@Aggregation(pipeline = {
    "{ '$match': { 'propertyId': { $in: ?0 } } }",
    "{ '$group': { '_id': null, 'overallAverageRating': { $avg: '$rating' }, 'totalReviews': { $sum: 1 }, 'excellentCount':
    { $sum: { $cond: [{ $gte: ['$rating', 4.5] }, 1, 0] } }, 'goodCount': { $sum: { $cond: [{ $and: [{ $gte: ['$rating', 3.5]
    }, { $lt: ['$rating', 4.5] }] }, 1, 0] } }, 'poorCount': { $sum: { $cond: [{ $lt: ['$rating', 3.5] }, 1, 0] } } } }",
    "{ '$project': { '_id': 0 } }"
})
Map<String, Object> getPortfolioRatingStatistics(List<String> propertyIds);
```

Document DB Design

Aggregation Queries Benefiting from Design



```
// Get average rating per property for comparative analysis
@Aggregation(pipeline = {
    "{ '$match': { 'propertyId': { $in: ?0 } } }",
    "{ '$group': { '_id': '$propertyId', 'averageRating': { $avg: '$rating' }, 'reviewCount': { $sum: 1 } } }",
    "{ '$sort': { 'averageRating': -1 } }",
    "{ '$project': { 'propertyId': '$_id', 'averageRating': 1, 'reviewCount': 1, '_id': 0 } }"
})
List<Map<String, Object>> getPropertyRatingsComparison(List<String> propertyIds);
```

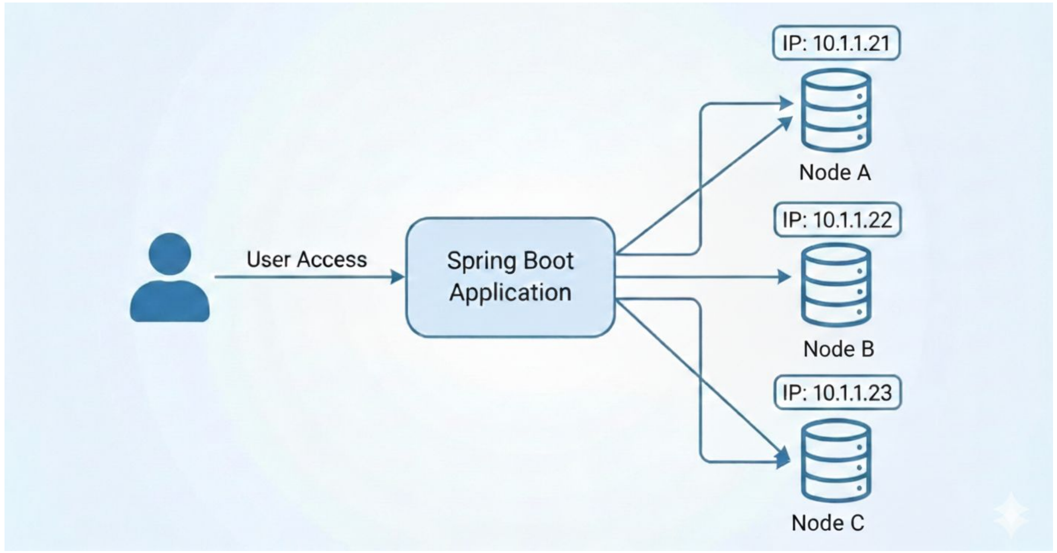
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Cluster Representation

The system deployment involves a 3-node cluster configured to maximize resource efficiency and availability.



MongoDB Replica Set Configuration

We deployed a **MongoDB Replica Set** across all three nodes with a priority-based election strategy: Node A acts as the **Primary** (Priority 5), while Nodes B (Priority 2) and C (Priority 1) serve as Secondaries.

MongoDB Replica Set Configuration

Read Operations (Read Preference: nearest)
Given that the read-to-write ratio in LargeB&B is estimated to be approximately 10:1 (browsing vs. booking), minimizing read latency is crucial. We utilize the nearest read preference. The driver routes the read request to the replica set member with the lowest network latency, regardless of whether it is Primary or Secondary.
This choice provides two main benefits:

- Load Balancing: It distributes the heavy read traffic across all three nodes, preventing the Primary from being overwhelmed.
- Geo-latency Reduction: If the infrastructure is geo-distributed in the future, users will automatically read from the data center closest to them.

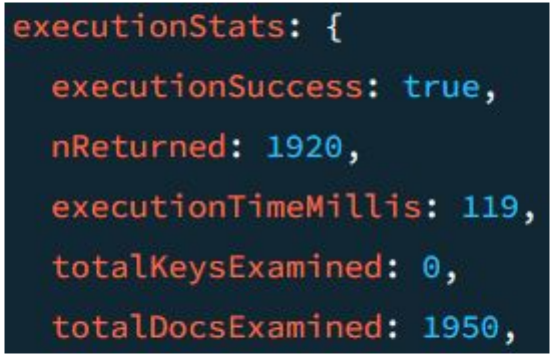
MongoDB Replica Set Configuration

Write Operations (Write Concern: majority):
Although we prioritize availability for reads, write operations—specifically User Registration, Property Creation, and critical Reservation flows—require strict durability. We have configured the Write Concern to majority.

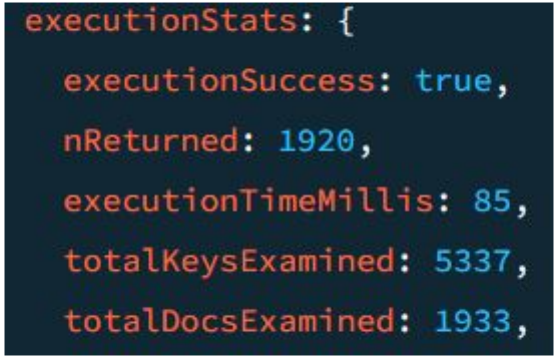
- The system acknowledges a write only after it has been committed to the Primary and propagated to at least one Secondary (2 out of 3 nodes).
- This ensures that even if the Primary node crashes immediately after a write (e.g., after a confirmed payment), the data is safe on a Secondary node that will become the new Primary, preventing data rollback and financial inconsistencies.

MongoDB Indexes

Left image query example with no index & right image query example with index
Properties Collection:
{"rooms.capacityAdults": 1, "rooms.capacityChildren": 1}: It allows the database to efficiently filter properties that contain at least one room capable of accommodating the requested number of adults and children.



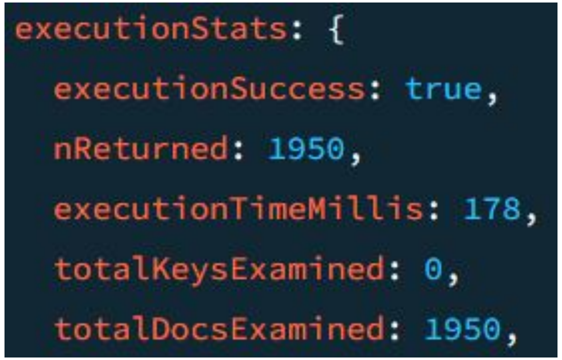
```
executionStats: {
  executionSuccess: true,
  nReturned: 1920,
  executionTimeMillis: 119,
  totalKeysExamined: 0,
  totalDocsExamined: 1950,
```



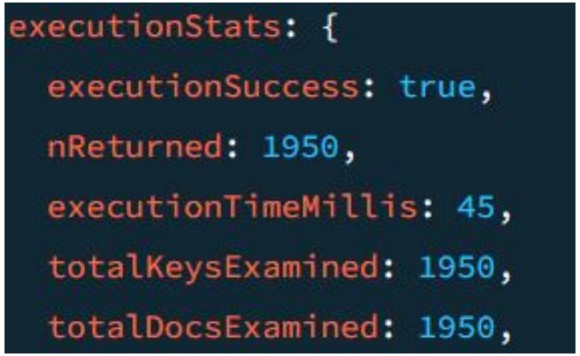
```
executionStats: {
  executionSuccess: true,
  nReturned: 1920,
  executionTimeMillis: 85,
  totalKeysExamined: 5337,
  totalDocsExamined: 1933,
```

MongoDB Indexes

"ratingStats.value": Optimizes the "Sort by Highest Rated" feature, ensuring that high-rated properties are retrieved efficiently without performing expensive in-memory sorting



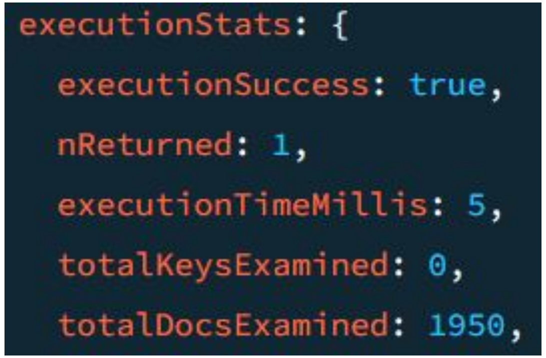
```
executionStats: {
  executionSuccess: true,
  nReturned: 1950,
  executionTimeMillis: 178,
  totalKeysExamined: 0,
  totalDocsExamined: 1950,
```



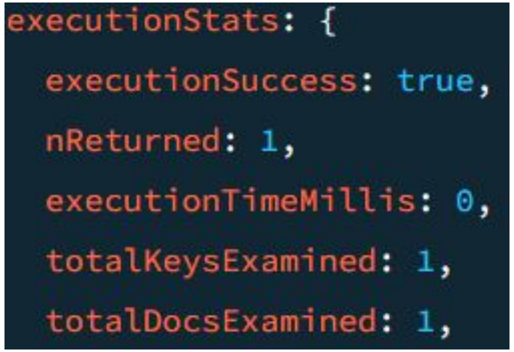
```
executionStats: {
  executionSuccess: true,
  nReturned: 1950,
  executionTimeMillis: 45,
  totalKeysExamined: 1950,
  totalDocsExamined: 1950,
```

MongoDB Indexes

"managerId": Ensures fast retrieval of the specific subset of properties owned by a logged-in manager for the dashboard view



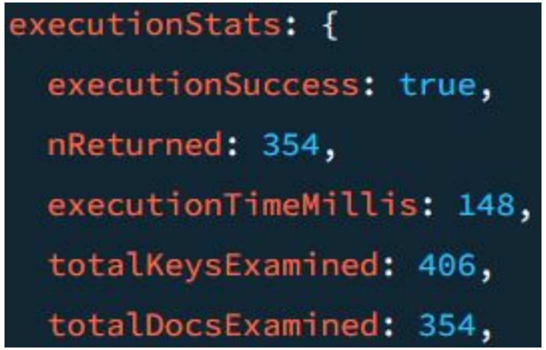
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 5,
  totalKeysExamined: 0,
  totalDocsExamined: 1950,
```



```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 0,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
```

MongoDB Indexes

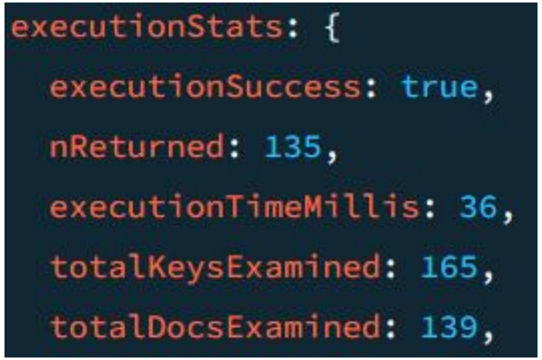
Text-Index:
“name (property) (weight: 3), name (room) (weight: 2), description (weight: 1), city (weight: 1)”: Enables full-text search capabilities with relevance scoring. Matches in the property name are prioritized over matches in the room name, and the room name is prioritized over matches in the description and city, improving search relevance for users



```
executionStats: {
  executionSuccess: true,
  nReturned: 354,
  executionTimeMillis: 148,
  totalKeysExamined: 406,
  totalDocsExamined: 354,
```

MongoDB Indexes

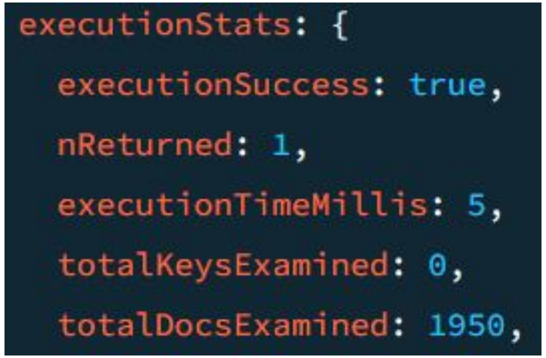
2dsphere Index:
"coordinates": Required to support geospatial queries such as $near and $geoWithin, satisfying the requirement to view properties on a map or find POIs nearby.



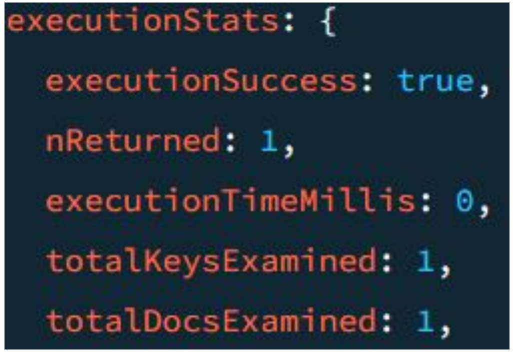
```
executionStats: {
  executionSuccess: true,
  nReturned: 135,
  executionTimeMillis: 36,
  totalKeysExamined: 165,
  totalDocsExamined: 139,
```

MongoDB Indexes

Rooms (Embedded in Property collection):
“pricePerNightAdults”: Optimizes complex budget queries where users filter by adult price



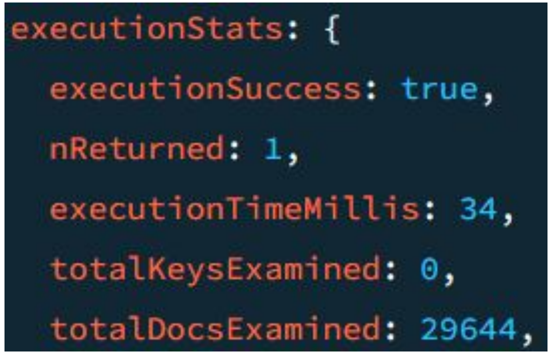
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 5,
  totalKeysExamined: 0,
  totalDocsExamined: 1950,
```



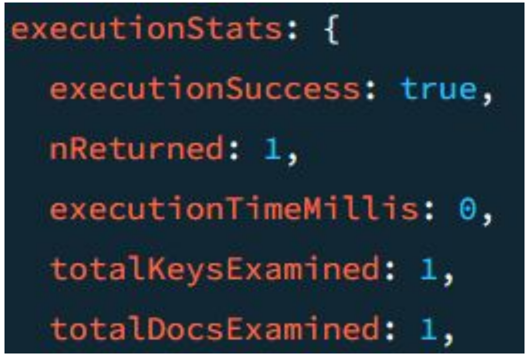
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 0,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
```

MongoDB Indexes

Reservations Collection:
{ "roomId": 1, "dates.checkIn": 1, "dates.checkOut": 1 } : It allows the system to efficiently check if a specific room is occupied during a given date range before confirming a new booking



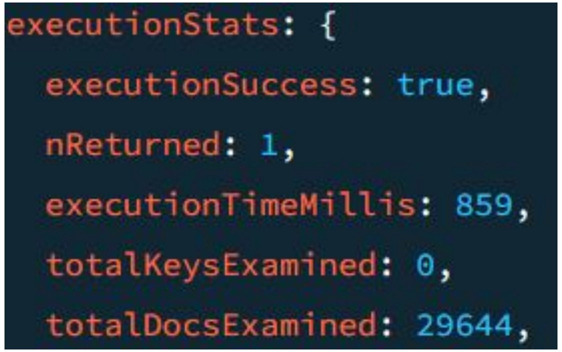
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 34,
  totalKeysExamined: 0,
  totalDocsExamined: 29644,
```



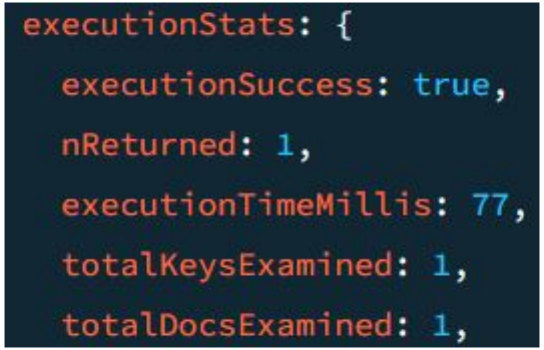
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 0,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
```

MongoDB Indexes

“userId”: Optimizes the retrieval of booking history for the ”My Bookings” page



```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 859,
  totalKeysExamined: 0,
  totalDocsExamined: 29644,
```



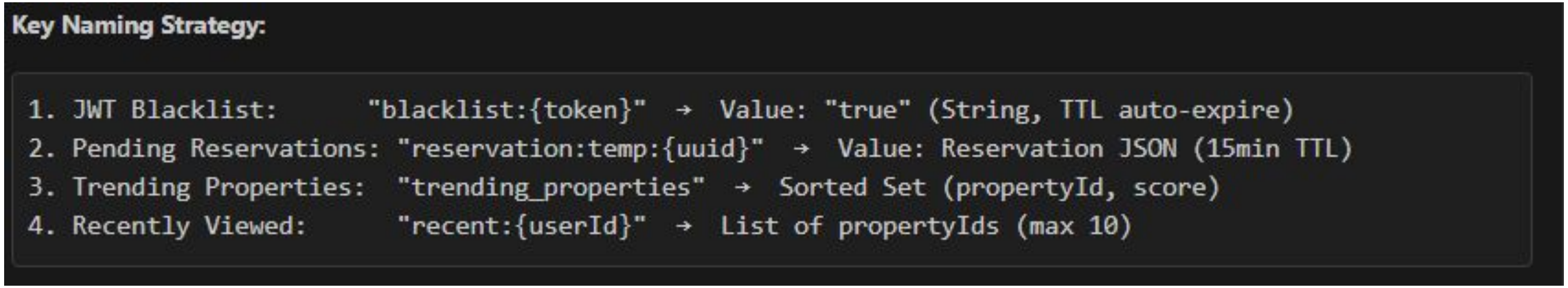
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 77,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
```

Discussion on MongoDB Data Sharding

Since the ***id*** *field is mandatory in every document and is the primary filter for the majority of queries, it was selected as the shard key. This choice enables direct query routing, avoiding the overhead of scatter-gather operations. Consequently, we adopted Hashed Sharding as the partitioning algorithm to ensure uniform distribution.*

Key-value DB Design

Redis Data Model & Key Naming



Key Naming Strategy:

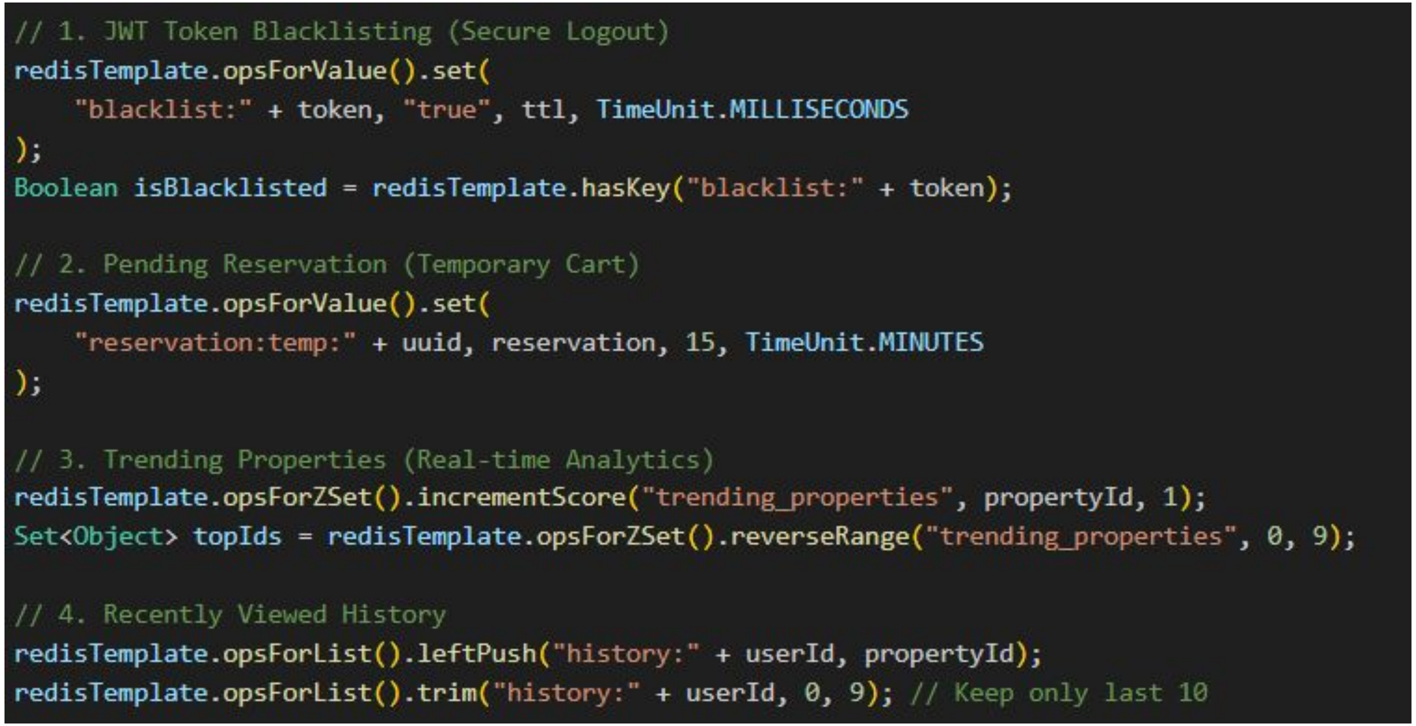
```
1. JWT Blacklist:      "blacklist:{token}"  → Value: "true" (String, TTL auto-expire)
2. Pending Reservations: "reservation:temp:{uuid}"  → Value: Reservation JSON (15min TTL)
3. Trending Properties:  "trending_properties"  → Sorted Set (propertyId, score)
4. Recently Viewed:      "recent:{userId}"  → List of propertyIds (max 10)
```

Design Justification:

- **Namespace prefixes** (blacklist:, reservation:temp:, recent:) prevent key collisions
- **TTL expiration** for temporary data (JWT tokens, pending reservations) ensures automatic cleanup
- **Sorted Set** for trending enables O(log N) scoring and range queries
- **List** for recently viewed maintains insertion order with O(1) head insertion

Key-value DB Design

Implemented Operations



```
// 1. JWT Token Blacklisting (Secure Logout)
redisTemplate.opsForValue().set(
    "blacklist:" + token, "true", ttl, TimeUnit.MILLISECONDS
);
Boolean isBlacklisted = redisTemplate.hasKey("blacklist:" + token);

// 2. Pending Reservation (Temporary Cart)
redisTemplate.opsForValue().set(
    "reservation:temp:" + uuid, reservation, 15, TimeUnit.MINUTES
);

// 3. Trending Properties (Real-time Analytics)
redisTemplate.opsForZSet().incrementScore("trending_properties", propertyId, 1);
Set<Object> topIds = redisTemplate.opsForZSet().reverseRange("trending_properties", 0, 9);

// 4. Recently Viewed History
redisTemplate.opsForList().leftPush("history:" + userId, propertyId);
redisTemplate.opsForList().trim("history:" + userId, 0, 9); // Keep only last 10
```

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Key-value DB Design

System Benefits:

- **Security**: Stateless JWT invalidation without database queries
- **Performance**: Sub-millisecond access for trending properties (vs MongoDB aggregation)
- **UX**: Pending reservations survive page refresh during payment flow
- **Personalization**: Recently viewed history without polluting main DB

Views Supported:

- **ShowProperties** - Displays trending properties banner (reads `trending_properties` ZSet)
- **Checkout Flow** - Retrieves pending reservation during payment (reservation:temp:{id})
- **User Dashboard** - Shows recently viewed properties (recent:{userId} List)

Redis Replica Set Configuration

*For Redis, Node A acts as the **Master** handling both write and read operations, supported by two Read-only Replicas on Nodes B and C. High Availability is guaranteed by **3 Redis Sentinels**, which continuously monitor the cluster's health and automatically promote a Replica to Master in the event of a failure. Data durability and disaster recovery are secured through periodic **RDB snapshots** persisted to disk.*

*To ensure system stability under heavy load, we implemented a strict memory management strategy. We set a maxmemory **limit of 512MB** combined with a **Least Frequently Used (LFU)** eviction policy. When this limit is reached, Redis evicts the least accessed keys first, protecting the system from memory saturation and crashes during traffic spikes.*

Discussion on Redis Data Sharding

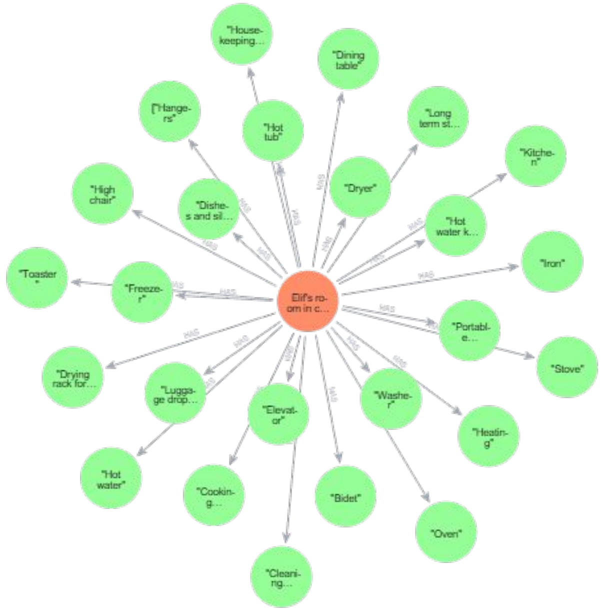
We implemented Sharding in Redis for:

- ***Horizontal Scalability (RAM):*** *It allows storing datasets larger than the memory limit of a single server by distributing data across multiple nodes.*
- ***Increased Write Throughput:*** *Since Redis is single-threaded, sharding splits write operations across multiple CPUs, significantly increasing total cluster performance.*

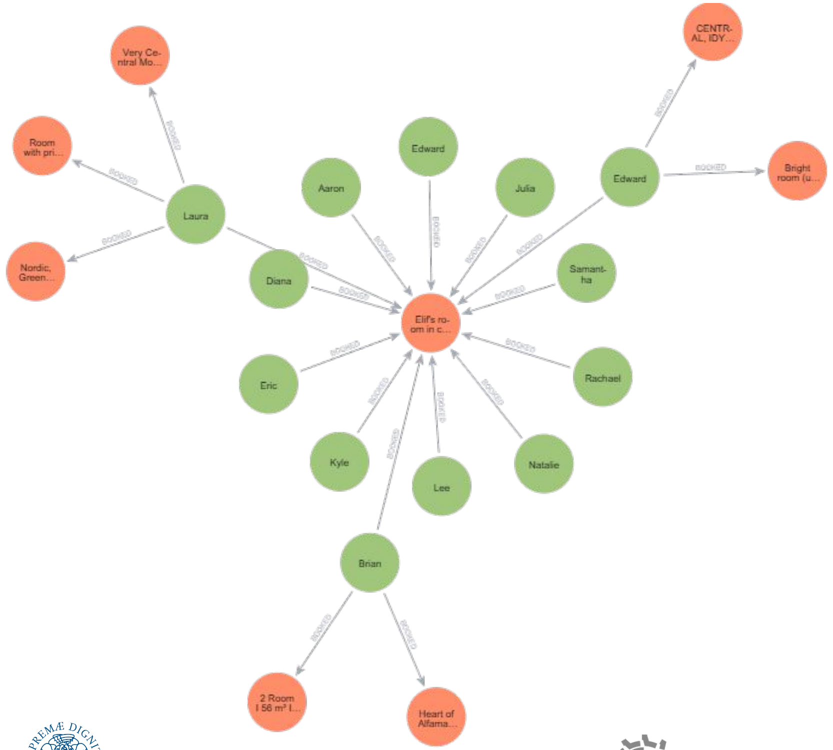
*The **database key** itself serves as the **sharding key**, and the partitioning strategy relies on a **hashing algorithm**.*

Graph DB Design

```
(:Property)-[:HAS]->(:Amenity)
```

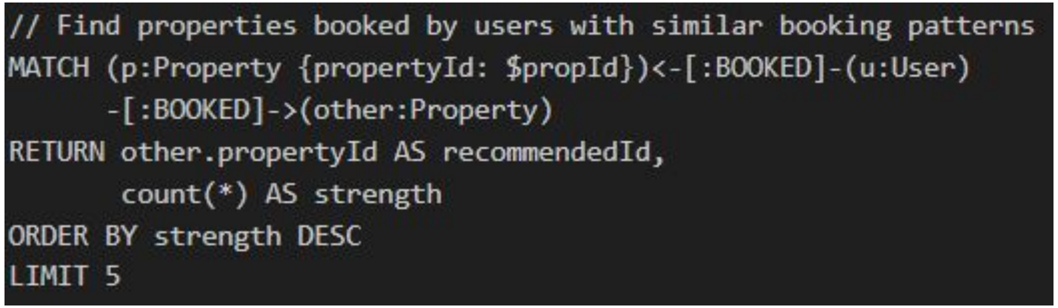


```
(:User)-[:BOOKED]->(:Property)
```



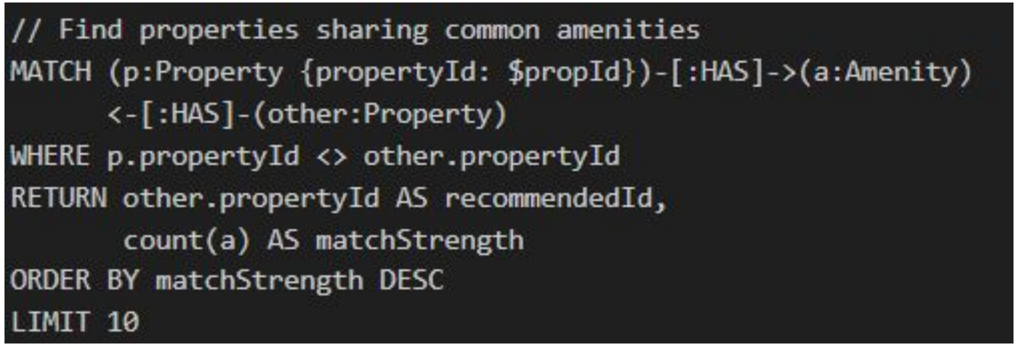
Graph DB Design

Collaborative Filtering (User-Based Recommendations)



```
// Find properties booked by users with similar booking patterns
MATCH (p:Property {propertyId: $propId})<-[:BOOKED]-(u:User)
      -[:BOOKED]->(other:Property)
RETURN other.propertyId AS recommendedId,
       count(*) AS strength
ORDER BY strength DESC
LIMIT 5
```

Content-Based Filtering (Feature Similarity)



```
// Find properties sharing common amenities
MATCH (p:Property {propertyId: $propId})-[:HAS]->(a:Amenity)
      <-[:HAS]-(other:Property)
WHERE p.propertyId <> other.propertyId
RETURN other.propertyId AS recommendedId,
       count(a) AS matchStrength
ORDER BY matchStrength DESC
LIMIT 10
```

Graph DB Design

Performance:

- Graph traversal in O(depth × branching_factor), independent of total dataset size
- 2-hop collaborative filtering: ~10ms vs ~500ms MongoDB aggregation

Query Expressiveness:

- "Users who booked X also booked Y" translates naturally to graph pattern matching
- Amenity-based similarity requires no complex array operations

Decoupled Architecture:

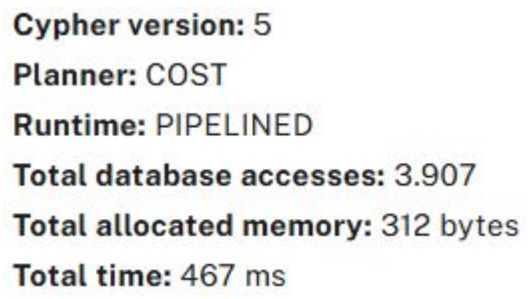
- Neo4j stores projection → can be rebuilt from MongoDB if corrupted
- Read-only for recommendations → no write conflicts with transactional data

Views Supported:

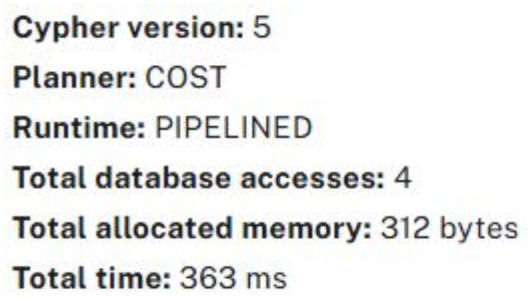
- **Property Details Page** → Shows "Similar Properties" (content-based query)
- **Recommendations Section** → Shows "Users Also Booked" (collaborative filtering)

Neo4j Indexes

Left image query example with no index & right image query example with index
Property Node:
"PropertyId": Both recommendation algorithms begin with a specific property currently being viewed (the "Anchor Node"). This constraint ensures O(1) lookup time to locate this starting point in the graph immediately



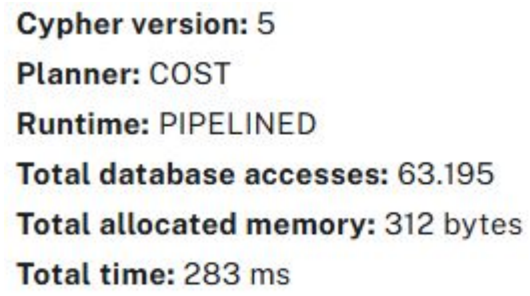
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 3.907
Total allocated memory: 312 bytes
Total time: 467 ms



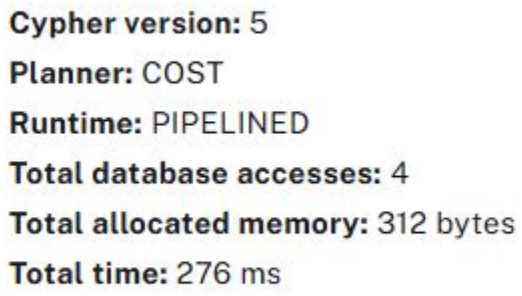
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 4
Total allocated memory: 312 bytes
Total time: 363 ms

Neo4j Indexes

User Node:
"UserId": Essential for the Collaborative Filtering algorithm



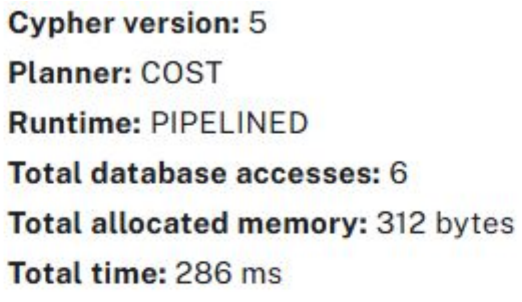
```
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 63.195
Total allocated memory: 312 bytes
Total time: 283 ms
```



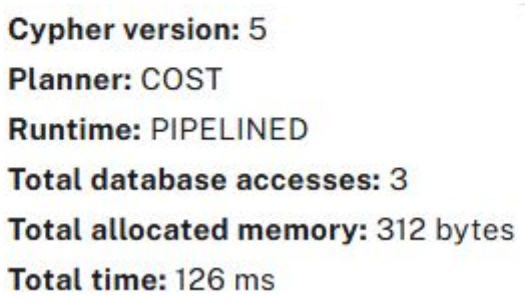
```
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 4
Total allocated memory: 312 bytes
Total time: 276 ms
```

Neo4j Indexes

Amenity Node:
“AmenityName”: Ensures that shared features (e.g., ”WiFi”, ”Pool”) are represented as single, unique nodes in the graph. This allows the traversal algorithm to ”hop” from the current property to related properties via these shared amenity nodes efficiently



```
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 6
Total allocated memory: 312 bytes
Total time: 286 ms
```



```
Cypher version: 5
Planner: COST
Runtime: PIPELINED
Total database accesses: 3
Total allocated memory: 312 bytes
Total time: 126 ms
```

Handling Intra-DB Consistency

Hypothesis: Since MongoDB has 3 replicas with Write Concern "majority", we can assume that a fail on connection or data persistence is almost impossible.

This consistency is managed using **asynchronous propagation** with fire-and-forget pattern:

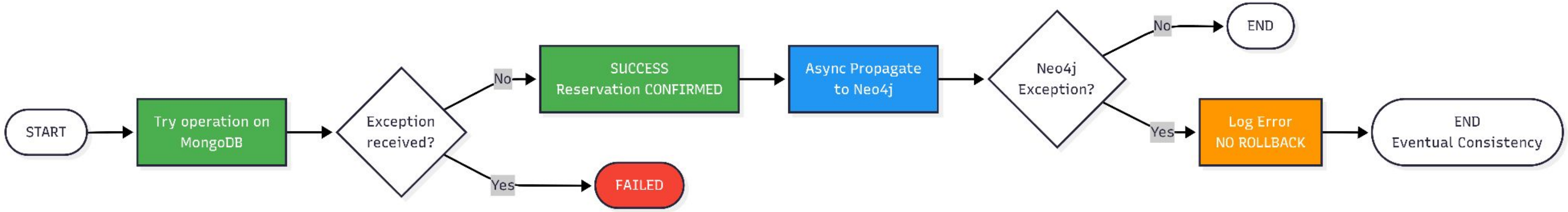
- **MongoDB is the Source of Truth**: All critical write operations (reservations, payments) are persisted to MongoDB FIRST with strong consistency guarantees.

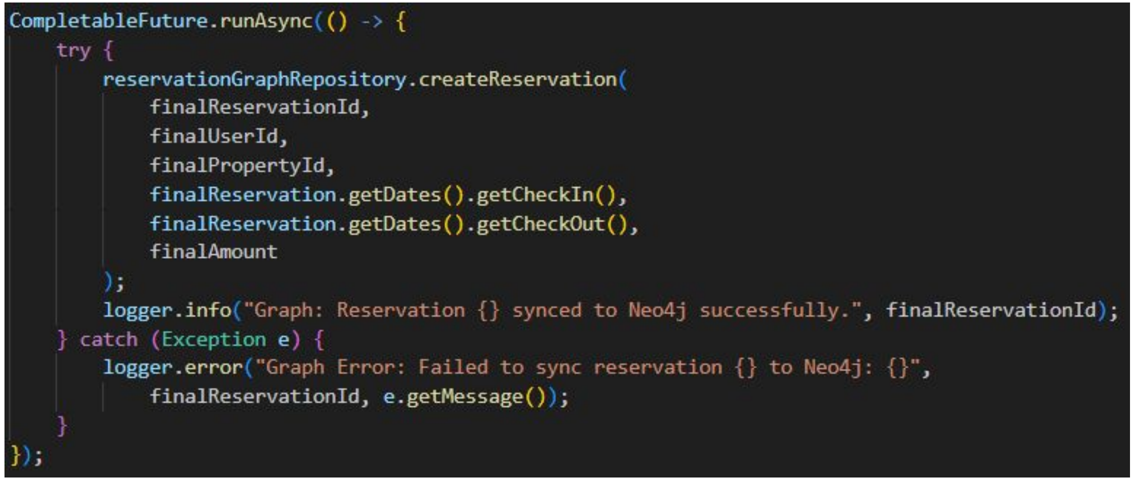
- **Neo4j is a Projection**: After MongoDB confirms the transaction, the relationship is propagated to Neo4j asynchronously using CompletableFuture.

- **Eventual Consistency Model**: If Neo4j propagation fails, the system logs the error but does NOT rollback the MongoDB transaction. The reservation remains valid.

- **Acceptable Inconsistency Window**: There may be a brief delay (milliseconds) before the booking appears in the recommendation graph, but this doesn't impact transactional integrity or user experience.

Handling Intra-DB Consistency

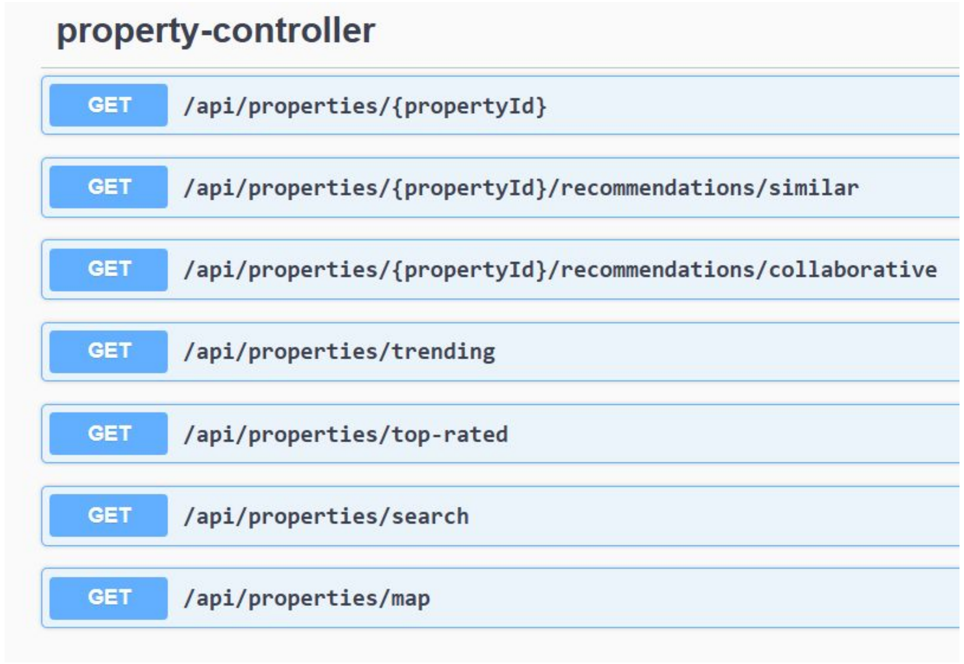




```
CompletableFuture.runAsync(() -> {
    try {
        reservationGraphRepository.createReservation(
            finalReservationId,
            finalUserId,
            finalPropertyId,
            finalReservation.getDates().getCheckIn(),
            finalReservation.getDates().getCheckOut(),
            finalAmount
        );
        logger.info("Graph: Reservation {} synced to Neo4j successfully.", finalReservationId);
    } catch (Exception e) {
        logger.error("Graph Error: Failed to sync reservation {} to Neo4j: {}",
            finalReservationId, e.getMessage());
    }
});
```

Swagger UI REST APIs documentation

Swagger doc url: http://localhost:8080/swagger-ui/index.html



property-controller

GET /api/properties/{propertyId}

GET /api/properties/{propertyId}/recommendations/similar

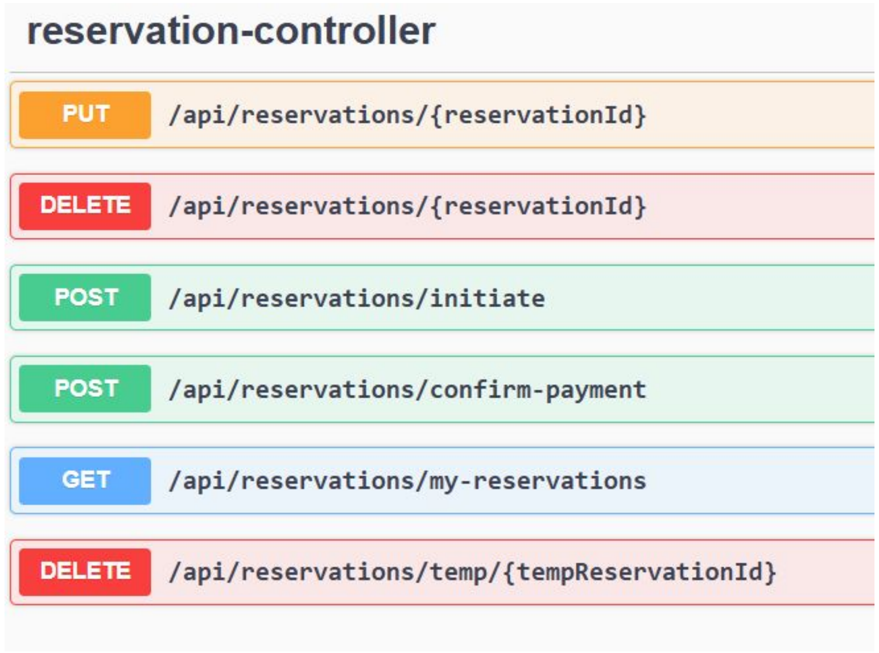
GET /api/properties/{propertyId}/recommendations/collaborative

GET /api/properties/trending

GET /api/properties/top-rated

GET /api/properties/search

GET /api/properties/map



reservation-controller

PUT /api/reservations/{reservationId}

DELETE /api/reservations/{reservationId}

POST /api/reservations/initiate

POST /api/reservations/confirm-payment

GET /api/reservations/my-reservations

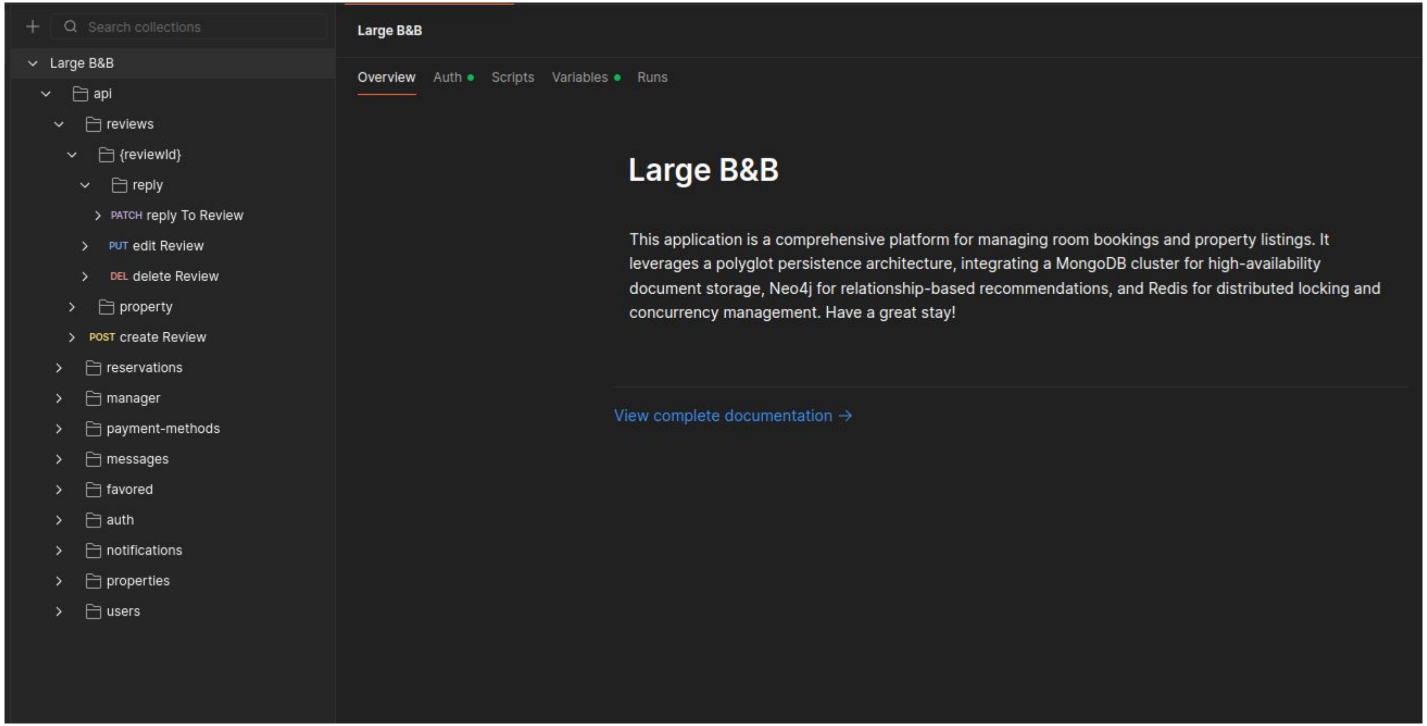
DELETE /api/reservations/temp/{tempReservationId}

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Live Demo with Postman



Large B&B

This application is a comprehensive platform for managing room bookings and property listings. It leverages a polyglot persistence architecture, integrating a MongoDB cluster for high-availability document storage, Neo4j for relationship-based recommendations, and Redis for distributed locking and concurrency management. Have a great stay!

View complete documentation →

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0