Large-Scale and Multi-Structured Databases
Project Design
CooGether

Cantini Irene, Perrone Federica,
Tempesti Pietro

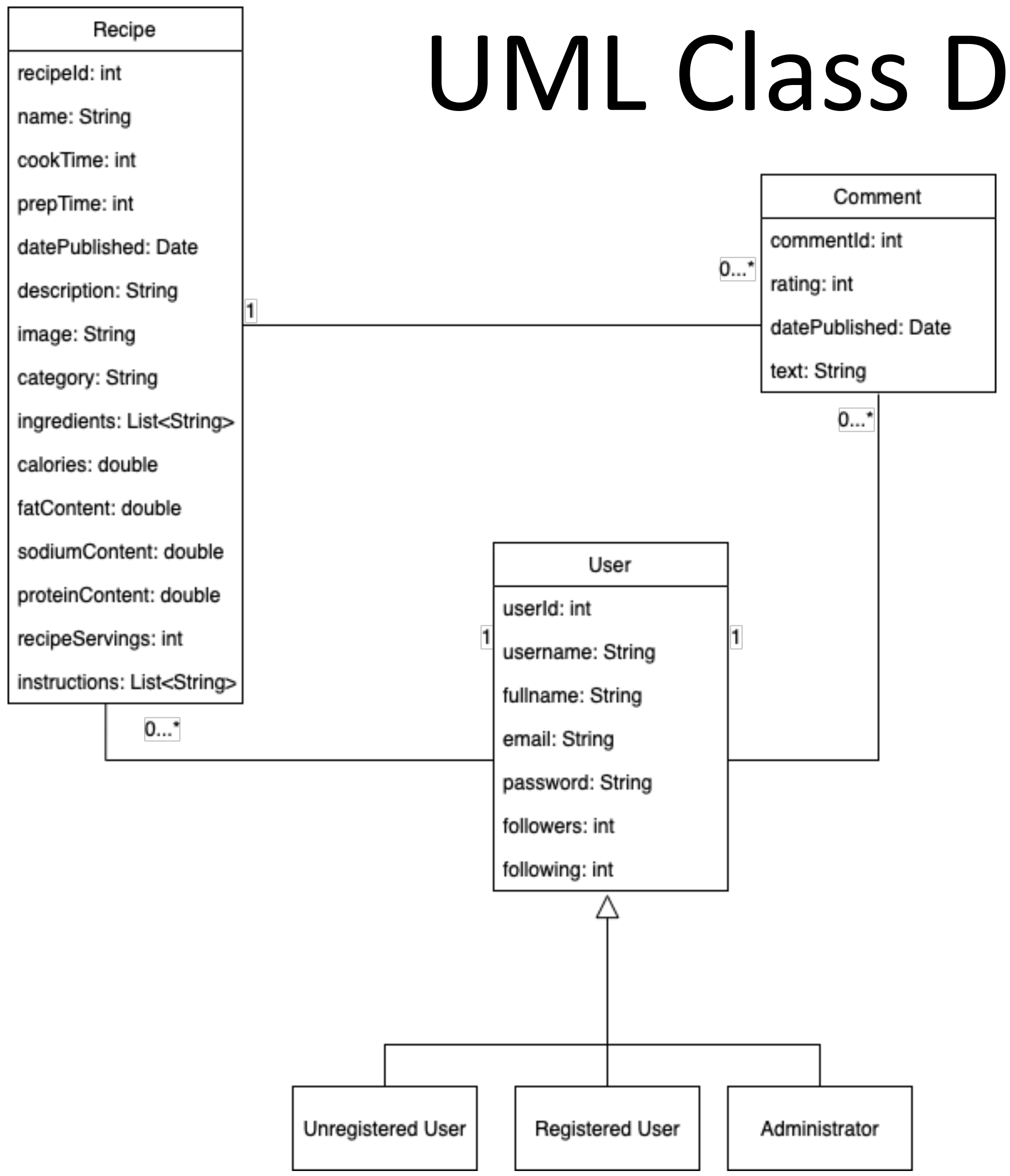
The application

CooGether is an application where users can share their recipes and interact with recipes proposed by other users. In this app, a user can leave a comment and add a review under every recipe: comments should be used as tips/tricks and variations of the original recipe and the owner of the recipe, if a modification proposed is particularly interesting, he can also update the original recipe.

A user can create his network of recipes' uploaders by following other users, in order to see their uploads with a higher priority in the application.

The app has also a ranking system, which allow any user to visualize the users whose recipes have received the best review.

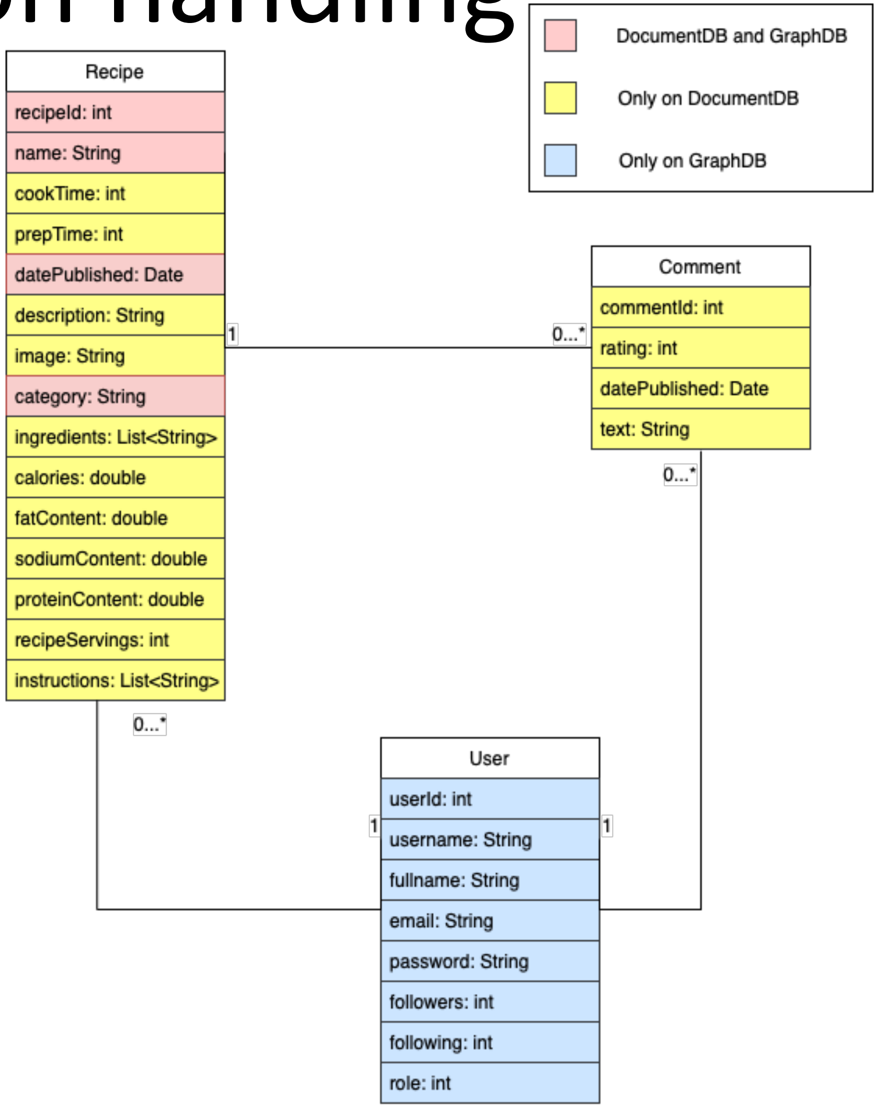
UML Class Diagram



Note: there's a cycle in the class diagram because comments and recipes could be written by different users, and we cannot retrieve these information without the cycle.

Generalization handling

We solve the generalization putting an Integer attribute in the entity User, called **role:** if the user is a registered one, role is 0; if, instead, it is the administrator, role is 1. If the user is an unregistered one, we do not save any information about it.



Dataset Description

Source:
https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews
https://randomuser.me
https://www.kaggle.com/hugodarwood/epirecipes?select=full_format_recipes.json

Description:
Dataset containing recipes, comments and users. We perform some modifications on these datasets, like remove duplicates and useless information, in order to obtain all right information necessary for a correct usage of the application.

Volume:

- 30.000 recipes (44.36MB)
- 170.000 Comments (83.5 MB)
- 65.000 Users (26.13 MB)

Variety:
We use dataset of recipes and comments scraped from food.com and epicurious; we performed a data integration transforming the epicurious recipes into the food.com recipes' format, handling missing values. We also assigned user information with the randomly generated users from randomuser.me

Non-Functional Requirements

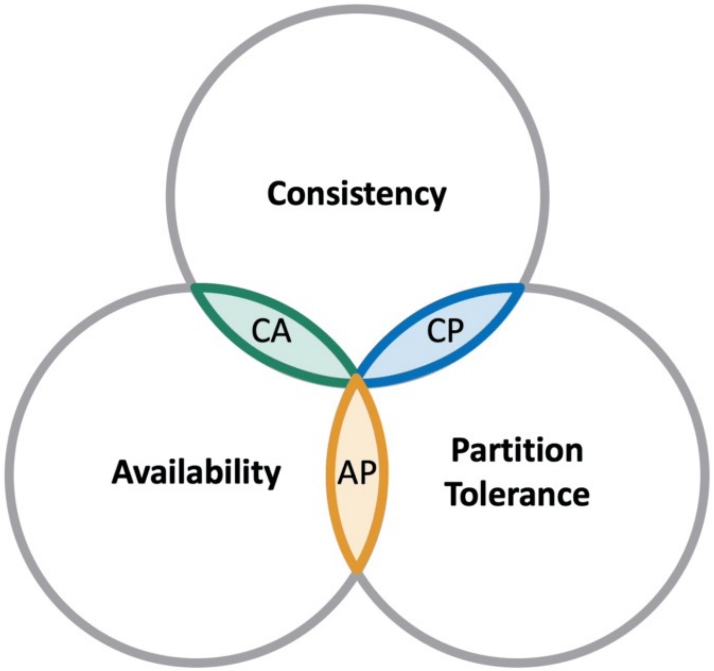
Product requirements

- **Usability:** the application must be easy to use and intuitive
- **Efficiency:** the application must have low response times in navigation
- **Reliability:** the application must handle data inconsistencies and wrong user inputs
- **Data consistency:** the application must handle the data consistency across multiple data sources

Organizational requirements

- If an user is deleted from the application all their recipes must be deleted, but not their comments.

Non-Functional Requirements



In our application, some of the most important requirements are the Availability and the Partition Tolerance, thus is very important to have a distributed system in order to ensure:

- An high availability of the service, so as we are able to response to any query.
- A partition protection, so as failures of individual nodes or connections between nodes do not impact the system as a whole.

MongoDB

Collections:

- Recipe
- Utility

Most relevant queries and analytics

1. Find the top-k healthiest recipes
2. Find the top-k most rated users
3. Find the top-k recipes with the highest number of 5-star reviews
4. Find the k recipes which requires less ingredients
5. Find the top-k fastest recipes
6. Find the recipes with the highest lifespan (recipes which have the highest difference between the date of publication of the first and the latest comment)

Requirements and Entities handled by Graph DB

Entities:

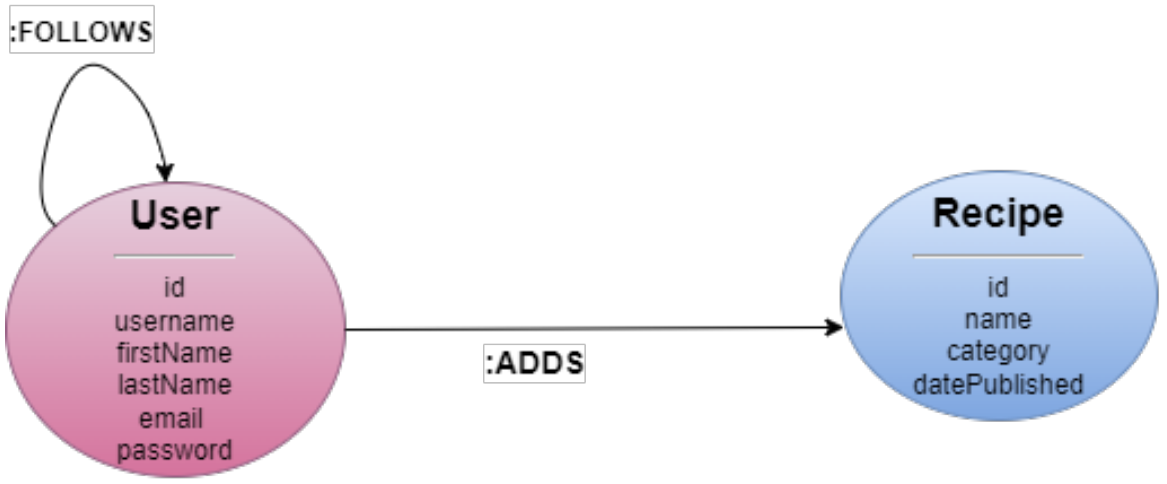
- Recipe
- User

Relations:

- User → User: follows
- User → Recipe: adds

High-level queries:

1. Find the top-k most active users (users which have written the highest number of recipes)
2. Find the top-k most followed users
3. Find suggested recipes (recipes written by users followed by the users that I follow)



Handling comments: a trivial problem

- Comments stored either in neo4j and MongoDB

To get all the information about the recipe with the highest number of comments, we need access to both MongoDB and Neo4j database. The total execution time was:

90ms+28ms=118ms.

- Comments stored only in MongoDB

We need to access only in MongoDB to get all comments of a specific recipe. The execution time was:

35ms

What about document size?

Document Size:

1. The recipe with the maximum number of comments has 2182 comments.
2. The maximum comment's size is 4438 bytes.
3. The maximum recipe size without comments is 9577 bytes
4. The maximum recipe's size is: $9577 bytes + 4438 bytes * 2182 comments$ = $9693293 bytes \simeq 10MB$ < 16MB

We have a good margin with the document size → All the comment information can be stored in MongoDB

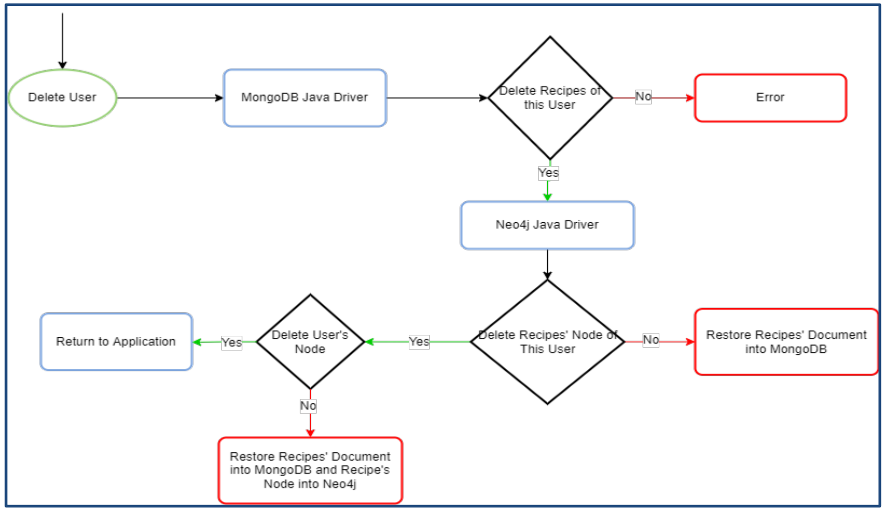
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

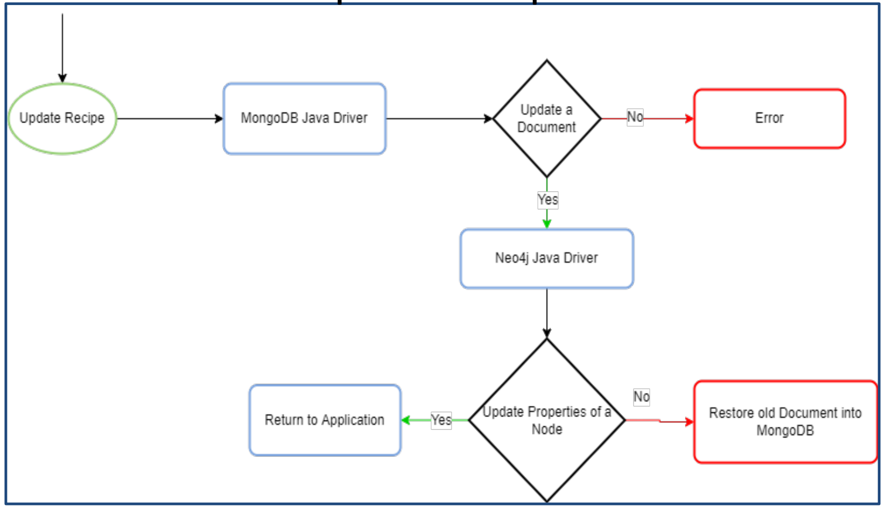
Data consistency

Delete user

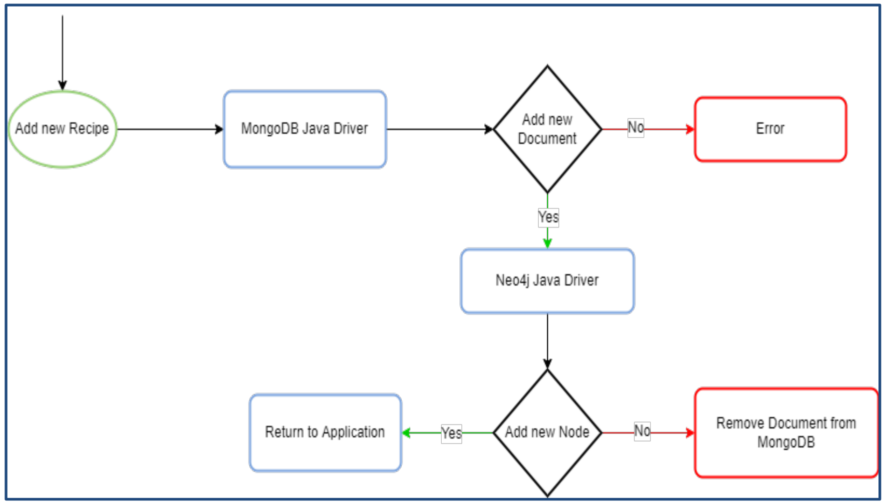


Delete User
MongoDB Java Driver
Delete Recipes of this User
No
Error
Yes
Neo4j Java Driver
Delete Recipes' Node of This User
No
Restore Recipes' Document into MongoDB
Yes
Delete User's Node
Yes
Return to Application
No
Restore Recipes' Document into MongoDB and Recipe's Node into Neo4j

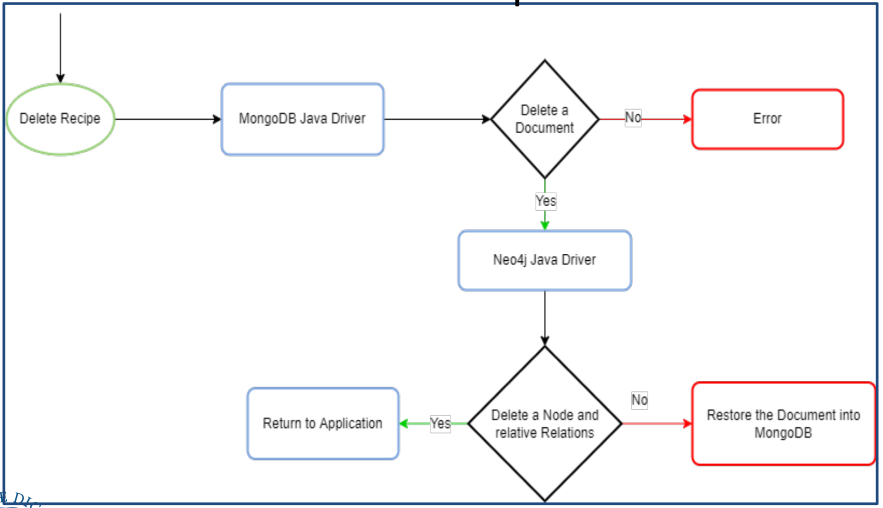
Update recipe



Add recipe



Delete recipe



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

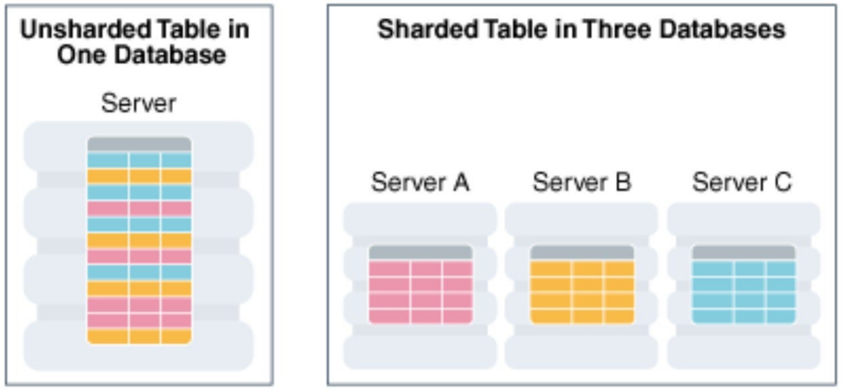
Sharding Proposal

- Split the whole dataset in *three different chunks* starting from the replica set that we have.

- *three replica set for each shard* in order to avoid failures and maintain high availability.

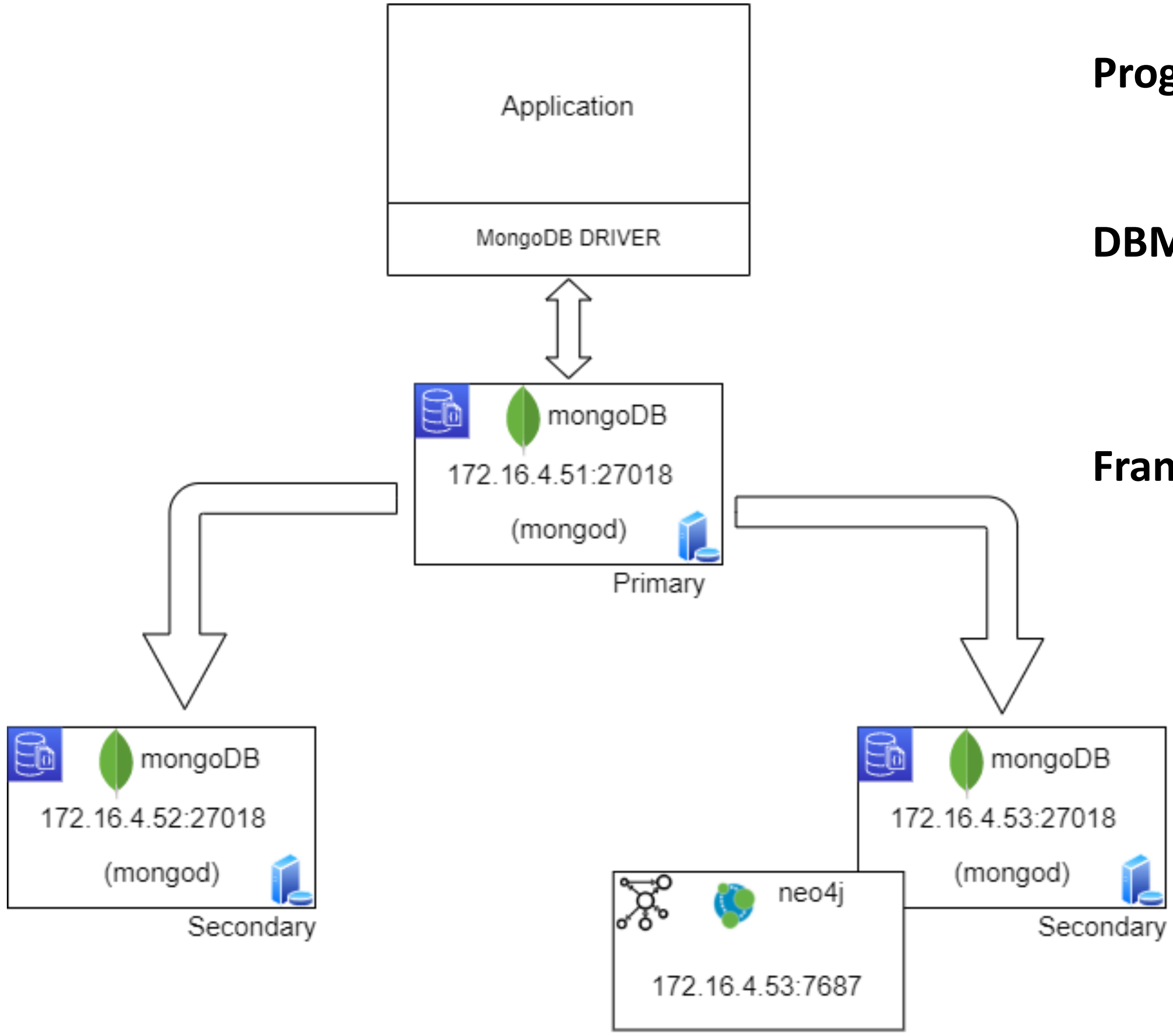
- *sharding key*: **category**, because the queries return, most of the time, recipes of a specific category.

- *Partition algorithm*: **Hashed Strategy**, the field chosen as the sharding key will be mapped through a hash function.



Unsharded Table in One Database
Server
Sharded Table in Three Databases
Server A
Server B
Server C

Software and Hardware Architecture



Programming language:

- Java

DBMSs:

- MongoDB
- Neo4j

Frameworks:

- Maven
- JavaFX for GUI
- Fasterxml.Jackson for JSON handling

CooGether
Cooking Together

Thank you for your attention!

Please ask us for any doubt or clarification, enjoy CooGether!

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0