

# Report MP - CBSE

What is CQRS?

**CQRS** stands for **Command and Query Responsibility Segregation**, a pattern that separates read and update operations for a data store. Implementing CQRS in your application can maximize its *performance*, *scalability*, and *security*. The flexibility created by migrating to CQRS allows a system to better evolve over time and prevents update commands from causing merge conflicts at the domain level.

Technologies ?

ElasticSearch for Query microservice:

Elasticsearch is a highly scalable open-source **full-text search and analytics engine**. It allows you to store, search, and analyze big volumes of data quickly and in near real time. It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.

MongoDB for Command microservice:

MongoDB is a **document database used to build highly available and scalable internet applications**. With its flexible schema approach, it's popular with development teams using agile methodologies. **It has ability to handle large amounts of unstructured data when it comes to speed.**

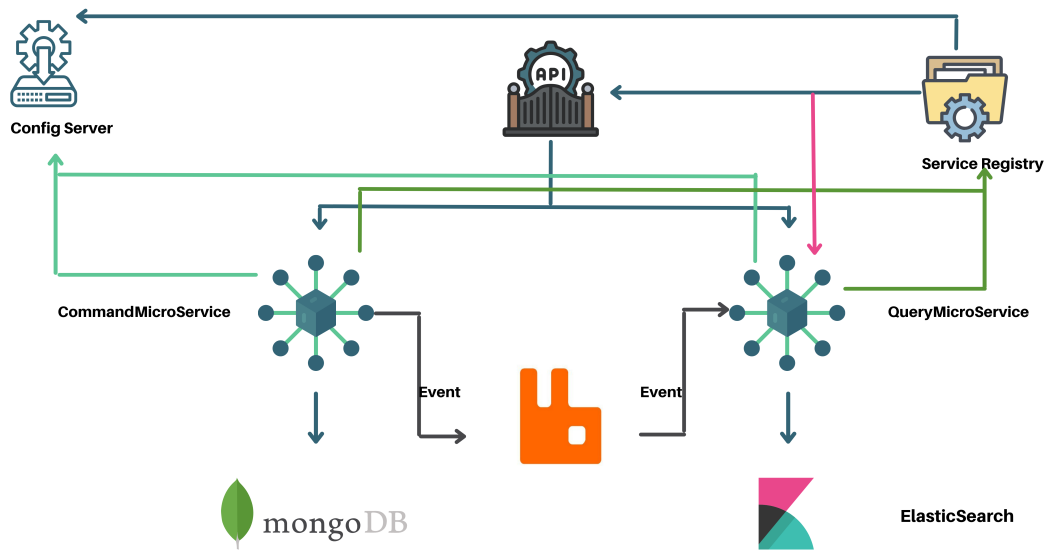
SpringBoot :

Spring Boot **helps developers to start coding right away without wasting time on preparing and configuring the environment**. In contrast to other Java frameworks, it provides flexible XML configurations, robust batch processing, database transactions, easy workflow, along with a wide variety of tools for development.

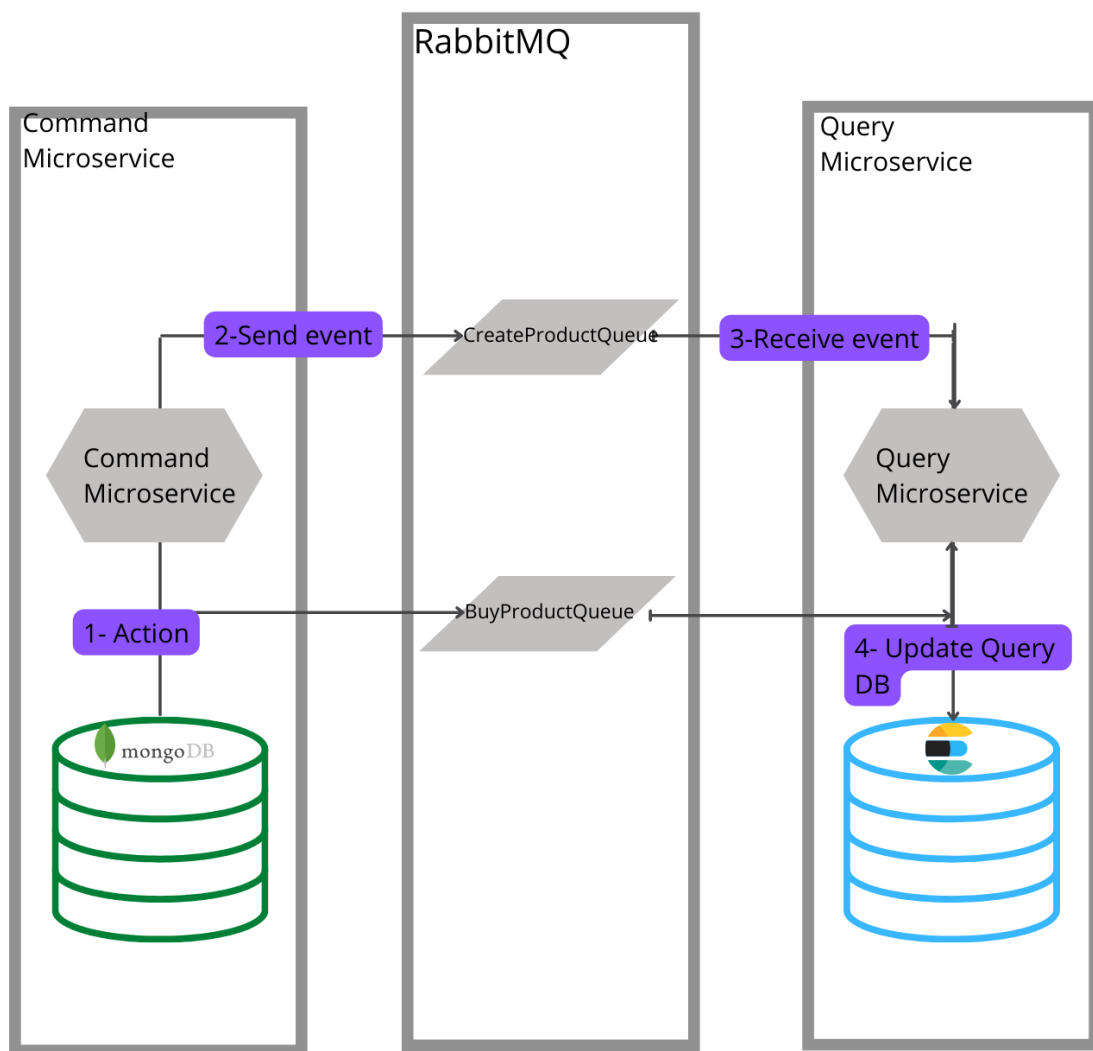
RabbitMQ:

The message broker that will allow us to synchronize between both databases by providing events

Project's Architecture :



Project's microservices :



It's easy! All you have to do is open your terminal in the root of the project and use the following command.

```
docker-compose build
docker-compose up
```




After starting the project, visit this to check if the services are registered or not;

<http://localhost:8761/>




The **Product** entity.

```
"Product": {
  "ref": "str",
  "name": "str",
  "description": "str",
  "price": "float",
  "quantity": "int"
}
```

### Commands:

 Command	 Method	 Request
<a href="#">Create Product</a>	Post	<a href="localhost:8080/command/create">localhost:8080/command/create</a>
<a href="#">Buy Product</a>	Post	<a href="localhost:8080/command/buy/{product-ref}">localhost:8080/command/buy/{product-ref}</a>

### Queries:

 Query	 Method	 Request
<a href="#">Get All Products</a>	Get	<a href="localhost:8080/query/">localhost:8080/query/</a>
<a href="#">Get Product By Reference</a>	Get	<a href="localhost:8080/query/{product-ref}">localhost:8080/query/{product-ref}</a>

| to purge both databases;

| localhost:8080/query/purge

| localhost:8080/command/purge