




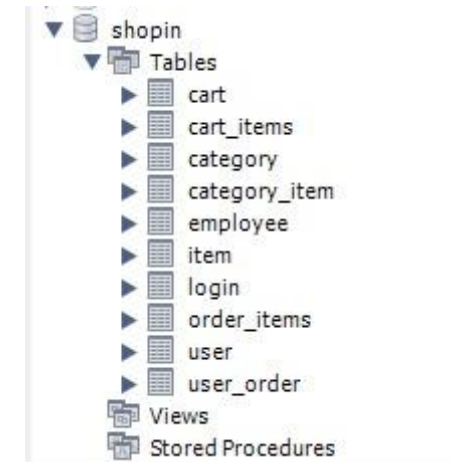


# Collections and Documents

## NoSQL design

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Categories	5	667.6 B	3.3 KB	1	20.0 KB	
Employees	3	192.3 B	577.0 B	1	20.0 KB	
Items	16	183.9 B	2.9 KB	1	20.0 KB	
Orders	5	202.4 B	1012.0 B	1	20.0 KB	
Users	3	571.3 B	1.7 KB	1	20.0 KB	

All mongoDB collections



SQL Table example

```
> {
  "_id": "1",
  "name": "Laptops",
  "description": "Laptops",
  "items": Array
    > 0: Object
    > 1: Object
    > 2: Object
    > 3: Object
    > 4: Object
    > 5: Object
  }
  {
    "_id": "4158eeda-2e64-4ec1-90a2-5189c3fffd75",
    "price": 800,
    "name": "Acer Swift 3",
    "description": "11 Generation Intel® Core",
    "_class": "com.imse.shopin.mongo.MongoCategory"
  }
```

Category collection

```
> {
  "_id": "92b4a851-83b1-4685-ba5c-f247edf0197e",
  "firstName": "Momir",
  "lastName": "Petrov",
  "email": "mome@gmail.com",
  "password": "admin",
  "_class": "com.imse.shopin.mongo.MongoEmployee"
}
```

Employee collection

```

_id: "130ecbc4-2f00-4dbb-bf63-334ba6279328"
price: 822.98
name: "Samsung Curved Monitor"
description: "110 cm (43 Zoll) 3.840 x 1.200 Pixel, 120Hz"
_class: "com.imse.shopin.mongo.MongoItem"

```

### Items collection example

```

_id: "66bc9062-dbc0-441e-a9f7-ad6884bb3017"
orderDate: 2020-11-30T00:00:00.000+00:00
totalPrice: 960
orderItems: Object
  68be01e3-3d1a-4014-af_ : 8
_class: "com.imse.shopin.mongo.MongoOrder"

```

### Order collection example

```

_id: "c0c0de8b-4318-462a-871d-898f9dca8ef0"
email: "mayo@gmail.com"
password: "123456"
cart: Object
  totalPrice: 0
  cartItems: Object
orders: Array
  0: Object
    _id: "66bc9062-dbc0-441e-a9f7-ad6884bb3017"
    orderDate: 2020-11-30T00:00:00.000+00:00
    totalPrice: 960
    orderItems: Object
      68be01e3-3d1a-4014-af92_ : 8
  1: Object
    _id: "d67a79dc-6bef-44e7-8e93-4867337fa130"
    orderDate: 2020-12-09T00:00:00.000+00:00
    totalPrice: 63.480000000000004
    orderItems: Object
      1e40fb47-2f64-40cb-b0e4_ : 6
  2: Object
    _id: "9b75829c-45f7-40a0-b771-64c078721a54"
    orderDate: 2020-12-04T00:00:00.000+00:00
    totalPrice: 1596
    orderItems: Object
      9dd06575-a51a-471d-a569_ : 4
name: "Hasan Gutti"
iban: "AT735277151951290001"
address: "Währinger Gürtel 2"
isLoggedIn: false

```

### User collection example

## Data migration from Sql to NoSql via Java,

The screenshot displays an IDE with three main panels:

- Left Panel (Project Explorer):** Shows the project structure for 'demo [imse master]'. The 'src/main/java' directory contains:
  - `com.imse.shopin`
    - `ShopInApplication.java`
  - `com.imse.shopin.controller`
    - `AdministrationController.java`
    - `BuyController.java`
    - `ItemController.java`
    - `LoginController.java`
    - `Migrator.java` (selected)
  - `com.imse.shopin.corpsShit`
  - `com.imse.shopin.model`
  - `com.imse.shopin.mongo`
  - `com.imse.shopin.mongoRepository`
  - `com.imse.shopin.mongoService`
  - `com.imse.shopin.repository`
  - `com.imse.shopin.service`
- Middle Panel (Code Editor):** Displays the code for `Migrator.java`. The code includes:
  - Imports for `java.util.*`, `com.imse.shopin.model.*`, `com.imse.shopin.service.*`, and `com.imse.shopin.mongo.*`.
  - A `private void migrateUser()` method that:
    - Fetches all users from `userService.findAllUser()`.
    - Iterates over each user, logging them in via `loginService.getLoginByUser(user)`.
    - Creates a `MongoUser` object using user details and the login service.
    - Fetches user orders via `orderService.findOrderByuser(user)`.
    - Iterates over orders, fetching items via `orderItemService.findOrderItemByOrder(order)` and calculating a total price.
    - Creates a `MongoOrder` object with the order details and total price.
    - Inserts the `MongoOrder` into `orderMongoService.insertOrder(mongoOrder)` and updates the `MongoUser` with `mongoUser.update(mongoOrder)`.
    - Finally, inserts the `MongoUser` into `mongoUserService.insertMongoUser(mongoUser)`.
- Right Panel (Class Hierarchy):** Shows the class hierarchy for `com.imse.shopin.controller`. The `Migrator` class is highlighted, showing its methods:
  - `itemServiceSql : ItemService`
  - `itemServiceMongo : MongoItemSe`
  - `categoryService : CategoryService`
  - `categoryServiceMongo : MongoCa`
  - `userService : UserService`
  - `mongoUserService : MongoUserSe`
  - `orderService : OrderService`
  - `orderItemService : OrderItems Servi`
  - `orderMongoService : MongoOrder`
  - `loginService : LoginService`
  - `employeeService : EmployeeService`
  - `mongoEmployeeService : MongoEi`
  - `isMigrated : boolean`
  - `Migrator()`
  - `migrateItems() : void`
  - `migrateEmployee() : void`
  - `migrateUser() : void`
  - `changeMigrate() : void`
  - `isMigrated() : boolean`

User migration example

# NoSQL Implementation of all Use-Cases from Milestone 2

## Main use cases

### Login

Objective: The customer enters his personal customer area, where he can track his orders, payments etc.

Description: In order to join his customer area, the user has to log in with his username and password. If a person doesn't yet have his login credentials, he must register, whereby if everything fits, he will be directly logged into the customer area. For every other login the customer has to use his login credentials.

The image shows a screenshot of an IDE with three panels. The left panel displays the project structure for 'demo [imse master]', showing the package 'com.imse.shopin.controller' and the file 'LoginController.java'. The middle panel shows the Java code for 'LoginController', which is a Spring controller with two endpoints: '/login/{email}/{password}' and '/isUserLoggedIn'. The code includes logic for user authentication, handling both regular users and employees, and returning appropriate messages. The right panel shows the dependencies for 'com.imse.shopin.controller', listing services like 'UserService', 'CartService', 'LoginService', 'ItemService', 'EmployeeService', 'Migrator', 'MongoUserService', and 'MongoEmployeeService'.

```
78 @Transactional
79 @GetMapping("/login/{email}/{password}")
80 public String login(@PathVariable String email, @PathVariable String password){
81     if(!migrator.isMigrated()) {
82         Login login = loginService.getLoginByEmail(email);
83         if(login != null) {
84             if(!login.getPassword().equals(password))
85                 return "Password doesn't match Email!";
86
87             if(!login.isEmployee()) {
88                 if(service.getCurrentLogin() == null) {
89                     service.updateUser(loginService.getLoginByEmail(email).getUser().getId(), true);
90                     service.setCurrentLogin(login.getUser());
91                     return "User successfully LoggedIn!";
92                 }
93                 else return "User Already LoggedIn!";
94             }
95             else if(login.isEmployee()) {
96                 if(employeeService.getCurrentLogin() == null) {
97                     employeeService.setCurrentEmployee(login.getEmployee());
98                     return "Employee successfully LoggedIn!";
99                 }
100                 else return "Employee already LoggedIn!";
101             }
102         }
103     }
104     if(mongoUserService.getMongoUserByEmail(email) != null) {
105         MongoUser mongoUser = mongoUserService.getMongoUserByEmail(email);
106         if(mongoUser.getEmail().equals(email) && mongoUser.getPassword().equals(password)) {
107             mongoUser.setLoggedIn(true);
108             mongoUserService.setCurrentMongoUser(mongoUser);
109             mongoUserService.updateMongoUser(mongoUser);
110
111             return "User successfully LoggedIn!";
112         }
113         else return "Password doesn't match Email!";
114     }
115     else if(mongoEmployeeService.getCurrentEmployee() == null && mongoEmployeeService.findByEmail(email) != null) {
116         mongoEmployeeService.setCurrentEmployee(mongoEmployeeService.findByEmail(email));
117         return "Mongo Admin logged In!";
118     }
119     return "User/Employee not registered!";
120 }
121
122 @CrossOrigin
123 @GetMapping("/isUserLoggedIn")
124 public boolean isUserLoggedIn() {
125     return service.getCurrentLogin() != null || mongoUserService.getCurrentMongoUser() != null;
126 }
127
128 @CrossOrigin
129 @PostMapping("/logout")
130 public String logout() {
```

com.imse.shopin.controller

- service : UserService
- cartService : CartService
- loginService : LoginService
- itemService : ItemService
- employeeService : EmployeeService
- migrator : Migrator
- mongoUserService : MongoUserService
- mongoEmployeeService : MongoEmployeeService
- register(@PathVariable String, @PathVariable String) : boolean
- login(@PathVariable String, @PathVariable String) : boolean
- isUserLoggedIn() : boolean
- logout() : String

## Execute the purchase

Objective: Purchase the products from the Shopping cart

Description: After the customer finished searching for products that he wants to order he can now proceed to purchase these goods.

## Add product to Cart

The screenshot displays an IDE with three panels. The left panel shows the project structure for 'demo [imse master]' with the following hierarchy:

- src/main/java
  - com.imse.shopin
    - com.imse.shopin.controller
      - AdministrationController.java
      - BuyController.java
      - ItemController.java
      - LoginController.java
      - Migrator.java
    - com.imse.shopin.corpsShit
    - com.imse.shopin.model
    - com.imse.shopin.mongo
    - com.imse.shopin.mongoRepository
    - com.imse.shopin.mongoService
    - com.imse.shopin.repository
    - com.imse.shopin.service
  - src/main/resources
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
  - target
    - api.iml
    - api.jar
    - Dockerfile
    - HELP.md
    - mvnw
    - mvnw.cmd

The middle panel shows the source code for the `BuyController` class, with line numbers 58 to 110. The code includes imports for `MongoItemService`, `MongoOrderService`, `ItemService`, `OrderService`, `OrderItemsService`, `Migrator`, `MongoUserService`, `MongoItemService`, and `MongoOrderService`. The `addItemToCart` method is annotated with `@PostMapping("/cartItem/{id}")` and `@PathVariable String id`. It checks if the user is logged in, finds the item, and adds it to the cart. The `mongoAddItem` method is a private helper that updates the cart and the user's cart.

```
58 MongoItemService mongoItemService;  
59 @Autowired  
60 MongoOrderService mongoOrderService;  
61  
62 @PostMapping("/cartItem/{id}")  
63 public String addItemToCart(@PathVariable String id) {  
64  
65     if(migrator.isMigrated()) return mongoAddItem(id);  
66  
67     if (userService.getCurrentLogin() == null) {  
68         return "No user logged in!";  
69     }  
70  
71     Item item = itemService.findItem(id);  
72  
73     Cart cart = userService.getCurrentLogin().getCart();  
74  
75     List<CartItem> cartItems = cartItemService.getCartItemsByCart(cart);  
76     CartItem newItem = null;  
77     if(cartItems != null) {  
78         for(CartItem itemInCart: cartItems) {  
79             if(itemInCart.getCart().equals(cart) && itemInCart.getItem().equals(item)) {  
80                 newItem = itemInCart;  
81                 break;  
82             }  
83         }  
84  
85         if(newItem != null) {  
86             cartItemService.increaseQuantity(newItem);  
87         }else  
88             cartItemService.addItemToCart(new CartItem(item, cart));  
89     }  
90     return "Added to cart";  
91 }  
92  
93  
94  
95 private String mongoAddItem(String id) {  
96  
97     if(mongoUserService.getCurrentMongoUser() != null) {  
98         MongoUser user = mongoUserService.getCurrentMongoUser();  
99         user.getCart().addMongoItemToCart(mongoItemService.findById(id));  
100         mongoUserService.updateMongoUser(mongoUserService.getCurrentMongoUser());  
101         return "Added";  
102     }  
103  
104     return "No User Logged in";  
105  
106  
107 }  
108  
109  
110
```

The right panel shows the class hierarchy for `com.imse.shopin.controller`, with `BuyController` selected. The hierarchy includes:

- com.imse.shopin.controller
  - userService : UserService
  - itemService : ItemService
  - cartItemService : CartItemService
  - orderService : OrderService
  - oiService : OrderItemsService
  - migrator : Migrator
  - mongoUserService : MongoUserService
  - mongoItemService : MongoItemService
  - mongoOrderService : MongoOrderService
  - addItemToCart(@PathVariable String id) : String
  - mongoAddItem(String) : String
  - removeItemFromCart(@PathVariable String id) : String
  - mongoRemoveItem(String) : List<CartItem>
  - orderItems() : String
  - getItemsFromCart() : List<CartItem>
  - getItems() : List<CartItem>
  - getItemsFromOrder() : List<OrderItem>

## Order items from cart

> demo [imse master]

src/main/java

com.imse.shopin

com.imse.shopin.controller

AdministrationController.java

BuyController.java

ItemController.java

LoginController.java

Migrator.java

com.imse.shopin.corpsShit

com.imse.shopin.model

com.imse.shopin.mongo

com.imse.shopin.mongoRepository

com.imse.shopin.mongoService

com.imse.shopin.repository

com.imse.shopin.service

src/main/resources

JRE System Library [JavaSE-11]

Maven Dependencies

src

> target

api.iml

api.jar

Dockerfile

HELP.md

mvnw

mvnw.cmd

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

}

return items;

}

return null;

}

/\*\*

\*

\* @return

\*/

@PostMapping("/order")

public String orderItems() {

if (userService.getCurrentLogin() == null && mongoUserService.getCurrentMongoUser() == null) {

return "No user Logged IN";

}

if(migrator.isMigrated()) {

MongoUser user = mongoUserService.getCurrentMongoUser();

MongoOrder newOrder = new MongoOrder(new Date(), user.getCart().getCartItems(), user.getCart().getTotalPrice());

mongoOrderService.insertOrder(newOrder);

user.update(newOrder);

user.getCart().emptyCart();

mongoUserService.updateMongoUser(user);

return "Mongo items ordered";

}

Order order = new Order(userService.getCurrentLogin(), new Date());

orderService.saveItem(order);

Cart cart = userService.getCurrentLogin().getCart();

List<CartItem> cartItems = cartItemService.getCartItemsbyCart(cart);

for(CartItems itemInCart: cartItems) {

oiService.saveOrder(new OrderItems(itemInCart.getItem(), order, itemInCart.getQuantity()));

cartItemService.deleteCartItem(itemInCart);

}

return "Items ordered";

}

@GetMapping("/allCartItems")

public List<CartDTO> getItemsFromCart(){

if (userService.getCurrentLogin() == null

&& mongoUserService.getCurrentMongoUser() == null)

return null;

return getItems();

}

com.imse.shopin.controller

BuyController

userService : UserService

itemService : ItemService

cartItemService : CartItemService

orderService : OrderService

oiService : OrderItemsService

migrator : Migrator

mongoUserService : MongoUserSe

mongoItemService : MongoItemSe

mongoOrderService : MongoOrder

addItemToCart(@PathVariable String

mongoAddItem(String) : String

removeItemFromCart(@PathVariabl

mongoRemoveItem(String) : List<C

orderItems() : String

getItemsFromCart() : List<CartDTO>

getItems() : List<CartDTO>

getItemsFromOrder() : List<OrderIt

## Reporting use cases

### Most sold product

Objective: Find bestselling products pro customer

Description: Administrator can find out what their best-selling Products are to recommend them, and stock more of them.

```
▼ demo [imse master]
  ▼ src/main/java
    > com.imse.shopin
    ▼ com.imse.shopin.controller
      > AdministrationController.java
      > BuyController.java
      > ItemController.java
      > LoginController.java
      > Migrator.java
    > com.imse.shopin.corpsShit
    > com.imse.shopin.model
    > com.imse.shopin.mongo
    > com.imse.shopin.mongoRepository
    > com.imse.shopin.mongoService
    > com.imse.shopin.repository
    > com.imse.shopin.service
  > src/main/resources
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  > src
  > target
  api.iml
```

```
1 package com.imse.shopin.controller;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 @CrossOrigin
21 @RestController
22 public class AdministrationController {
23
24     @Autowired
25     AdministrationService service;
26
27     @Autowired
28     OrderService orderService;
29
30     @Autowired
31     EmployeeService employeeService;
32
33     @Autowired
34     Migrator migrator;
35
36     @Autowired
37     MongoAdminService mongoAdminService;
38
39     @Autowired
40     MongoEmployeeService mongoEmployeeService;
41
42     // Example: 2020-11-28/2021-01-09
43     @CrossOrigin
44     @GetMapping("/productReport/{from}/{to}")
45     public List<SalesReportDTO> UserItemReport(@PathVariable String from, @PathVariable String to) {
46         if(!migrator.isMigrated()) {
47             if(employeeService.getCurrentLogin() != null)
48                 return service.createSalesRep(from, to);
49             }else {
50                 if(mongoEmployeeService.getCurrentEmployee() != null) {
51                     System.out.println("Sale1");
52                     return mongoAdminService.createSalesReportForMongo(from, to);
53                 }
54             }
55         return null;
56     }
57
58
59 }
```

```
com.imse.shopin.controller
▼ AdministrationController
  ▲ service : AdministrationService
  ▲ orderService : OrderService
  ▲ employeeService : EmployeeService
  ▲ migrator : Migrator
  ▲ mongoAdminService : MongoAdminService
  ▲ mongoEmployeeService : MongoEmployeeService
  ● UserItemReport(@PathVariable String from, @PathVariable String to) : List<SalesReportDTO>
  ● CategoryItemReport(@PathVariable String category, @PathVariable String from, @PathVariable String to) : List<SalesReportDTO>
  ● isUserLoggedIn() : boolean
  ● migrateData() : void
  ● ismigrated() : boolean
```

## Product sales report per category

Objective: Be able to see the top selling product per month (order date) per category.

Description: Employees can see precise statistic on product sales per category per month.

The screenshot displays an IDE with three panels. The left panel shows a project file tree with the following structure:

- com.imse.shopin.controllers
  - AdministrationController.java
  - BuyController.java
  - ItemController.java
  - LoginController.java
  - Migrator.java
- com.imse.shopin.corpsShit
- com.imse.shopin.model
- com.imse.shopin.mongo
- com.imse.shopin.mongoRepository
- com.imse.shopin.mongoService
- com.imse.shopin.repository
- com.imse.shopin.service
- src/main/resources
- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target
- api.iml
- api.jar
- Dockerfile
- HELP.md
- mvnw
- mvnw.cmd
- nom.xml

The middle panel shows the following Java code:

```
52         return mongoAdminService.createSalesReportForMongo(from, to);
53     }
54 }
55     return null;
56 }
57 }
58 }
59 }
60 @CrossOrigin
61 @GetMapping("/categoryReport/{from}/{to}")
62 public List<CategoryReportDTO> categoryItemReport(@PathVariable String from, @PathVariable String to) {
63     if(!migrator.isMigrated()) {
64         if(employeeService.getCurrentLogin() != null)
65             return service.createCategoryReport(from, to);
66     }else {
67         if(mongoEmployeeService.getCurrentEmployee() != null)
68             return mongoAdminService.createCategoryReportForMongo(from, to);
69     }
70     return null;
71 }
72 }
73 }
74 @CrossOrigin
75 @GetMapping("/isEmployeeLoggedIn")
76 public boolean isUserLoggedIn() {
77     return employeeService.getCurrentLogin() != null || mongoEmployeeService.getCurrentEmployee() != null;
78 }
79 }
80 @CrossOrigin
81 @PostMapping("/migrate")
82 public void migrateData() {
83     if(!migrator.isMigrated()) {
84         migrator.changeMigrate();
85         migrator.migrateItems();
86     }
87     System.out.println("### DATA MIGRATED ###");
88 }
89 }
90 }
91 @CrossOrigin
92 @GetMapping("/ismigrated")
93 public boolean ismigrated() {
94     return migrator.isMigrated();
95 }
96 }
97 }
```

The right panel shows a list of dependencies:

- employeeService : EmployeeService
- migrator : Migrator
- mongoAdminService : MongoAdminService
- mongoEmployeeService : MongoEmployeeService
- UserItemReport(@PathVariable String from, @PathVariable String to) : void
- CategoryItemReport(@PathVariable String from, @PathVariable String to) : void
- isUserLoggedIn() : boolean
- migrateData() : void
- ismigrated() : boolean



## **Indices**

Our intention was to sort the orders by date, so that the orders with the most recent dates are at the top and the query is more performant in that way. Unfortunately we couldn't manage to implement this with Spring.

## **Performance implications of the chosen design for the reporting use cases**

When an order is made, it is saved in the user document (see User Collections example) and additionally saved in the orders document (see Orders collections example) as separate objects. We organized it in that way in order to get better performance in the way that we don't have to iterate all users to get their orders for the category report cause all order items are already stored in the user object. For the most sold product report (per user) we just have to iterate through all users and extract the orders to determine the most sold products, when organized in that way.

## Working protocol

Date, Time	Length in minutes	Task	Result	Responsibility	Remarks
22.01.2020, 10:00	120	Mongo Repository and Service impl.	Classes and methods implemented	Adem Mehremic	...
23.01.2020 13:00	280	Mongo Repository and Service impl.	Classes and methods implemented	Nemanja Srdanovic	...
24.01.2020 13:15	360	Mongo Repository and Service impl.	Classes and methods implemented	Adem Mehremic	...
25.01.2020 14:25	360	Mongo Repository and Service impl.	Classes and methods implemented	Adem Mehremic	...
26.01.2020 11:00	430	Frontend extension and presentation	functional frontend component and presentation	Nemanja Srdanovic	...