Students:

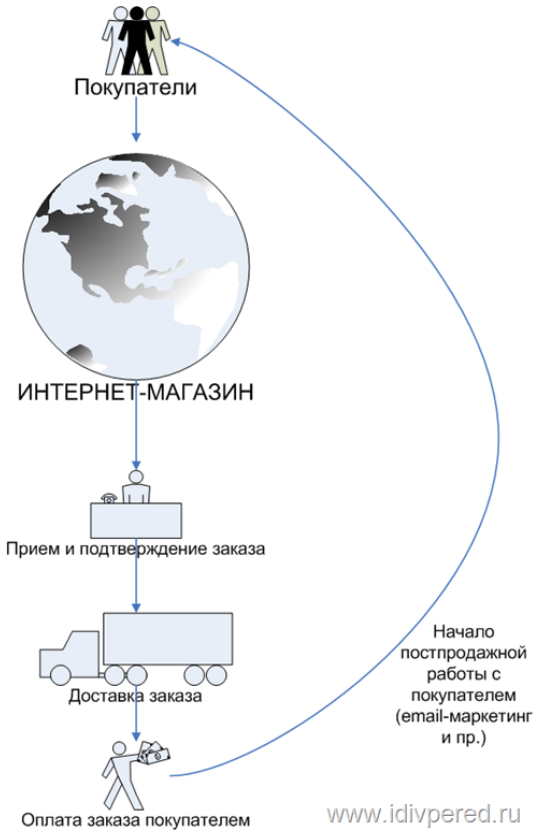
1) Serikbay Duman
2) Kanatbek Adilet

Topic is: **Online - magazine**

The structure of the online store is quite simple and with a certain approach does not cause any difficulties. We have depicted everything graphically for ease of understanding. And at the end of the article, you will find the most detailed scheme of the online store.

Briefly, the scheme of work is as follows:

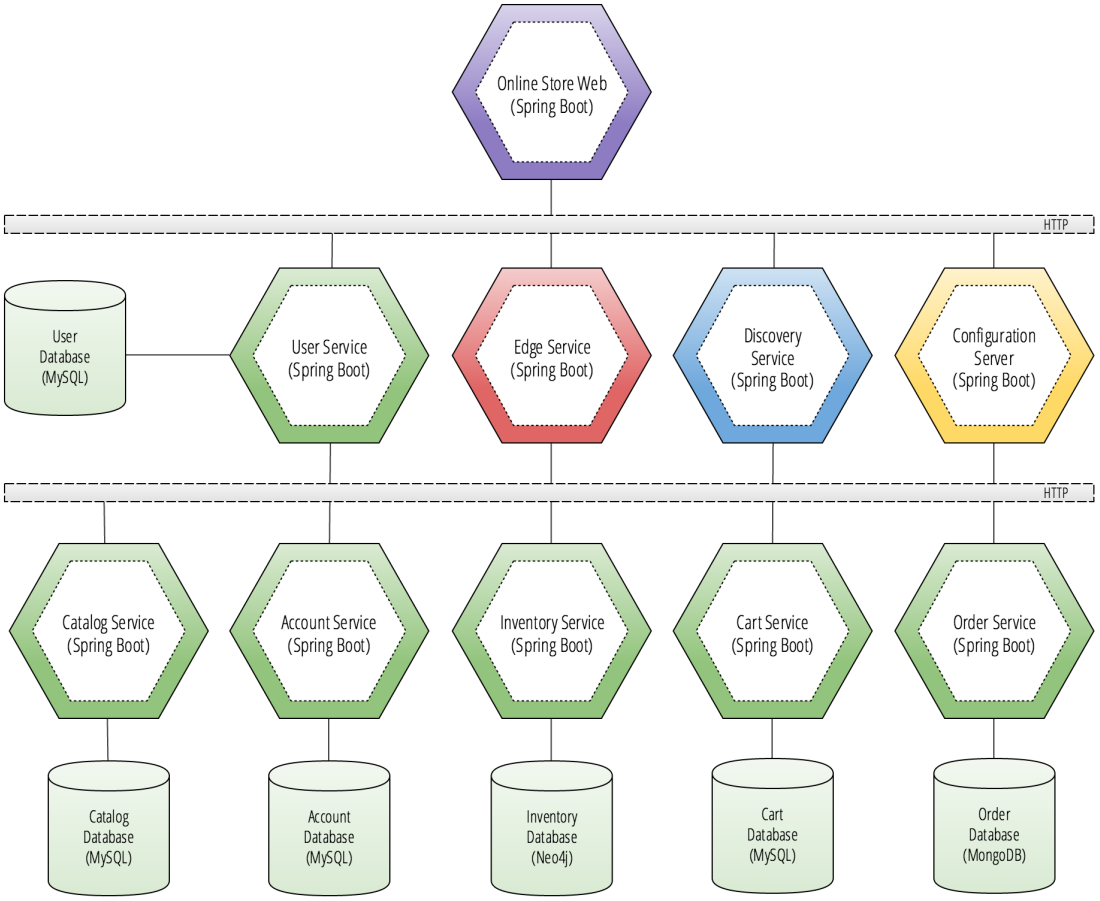
- The buyer comes to the site of your online store;
- Places an order;
- The online store confirms the order and sends it to the delivery service;
- Courier service or mail delivers the goods;
- The buyer pays for it;
- Start of post-sale work with a customer (email marketing, advertising, etc.)
- If the buyer makes an advance payment, then in this scheme it is enough to simply change the points of delivery and payment among themselves.



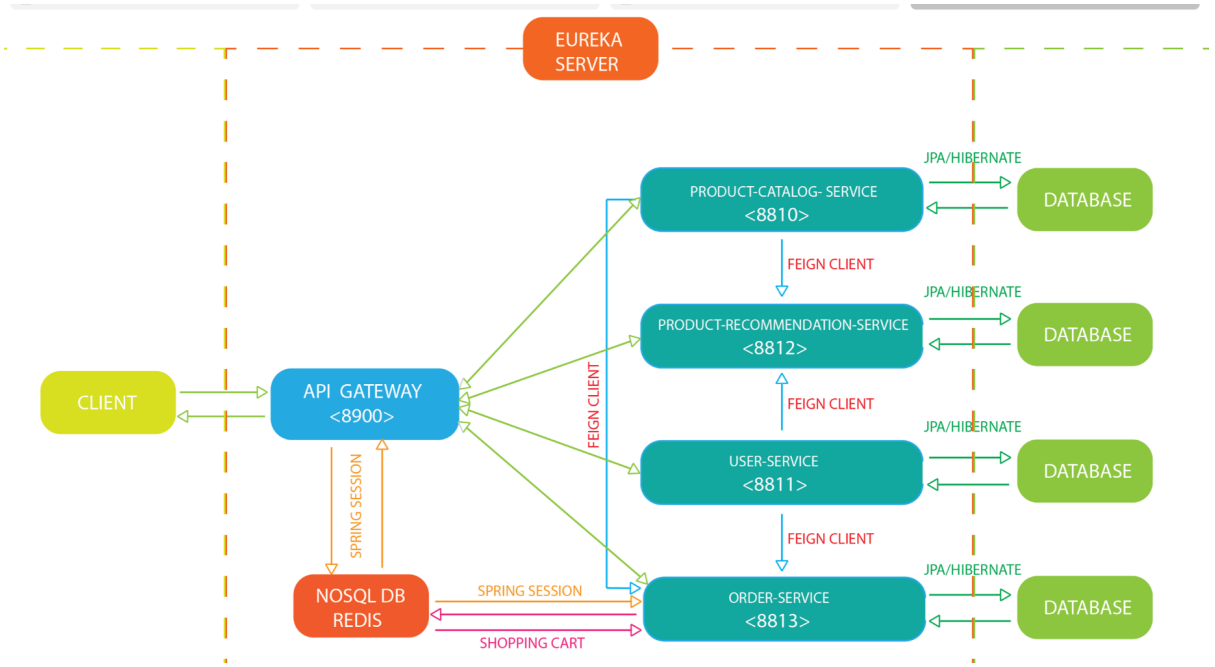
Let's describe the purchase of goods step by step:

- The buyer comes to the website of the online store;
- Sees the product or products he is interested in;
- Puts them in the basket;
- In the shopping cart you offer to buy accessories or similar products for these products;
- The buyer chooses something additionally and refills the cart;
- The buyer places an order;
- The employee of the online store contacts the buyer and confirms the order, then transfers it to the delivery service.

UML Diagram



Project System architecture



Tools and Technologies

- Java 8
- Spring Boot
- Spring Web MVC
- SQL Database engine
- Maven

Features

1. Administrator:
   - Users management
   - Products management
   - Orders management
   - Recommendations management

2. User:
   - Registration
   - Shopping cart (for guest or logged user)
   - Order

- Product recommendation
- Product catalog

Order Component Service Architecture

This component will be implemented as a Spring-Boot micro-service with a REST API and utilizing a NoSQL: MongoDB. This micro-service will be implemented using an MVC architecture and will have its own UI components that will be incorporated into the complete web application. The diagram above shows the architecture of the Order micro- services component interfaced with other services.

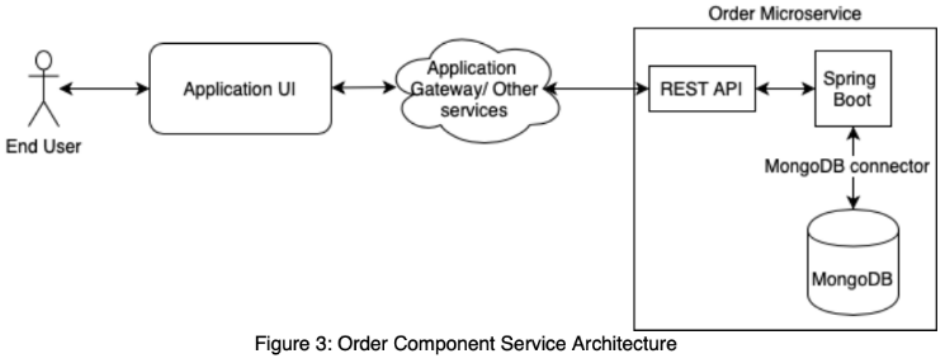


Figure 3: Order Component Service Architecture

- Front-End: HTML, CSS, JS, Bootstrap
- Rendering Engine: Thymeleaf
- Microservices: Catalog, order, cart, eureka, customer, shipping
- Back-End: JAVA 1.8
- Database: MongoDB

Microservice Integration

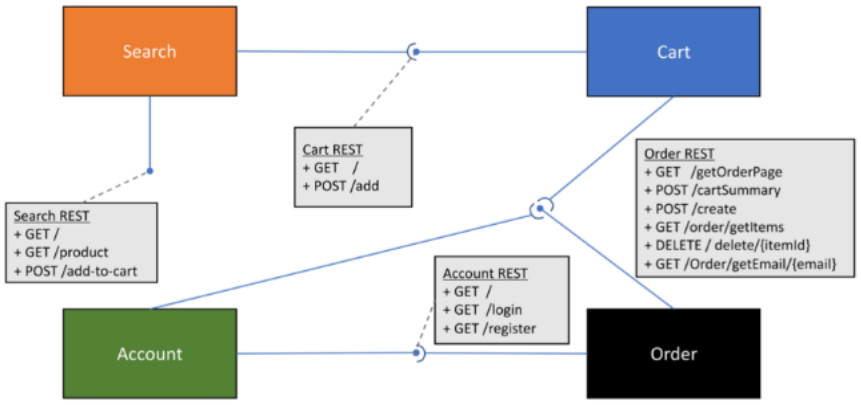


Figure 4: Microservices integration API

Once the cart service has the list of products a user is interested in, it coordinates with order microservice, where:

- POST /cartSummary/: Cart service posts cart items to order service
- DELETE /delete/{itemId}/: Cart deletes the item by itemId on order database if that item is deleted in cart
- GET /getOrderPage/: Returns Order page with Items that were in the cart as order summary

Once the order service receives items from cart, then order service coordinates with account service, where:

- GET /login/ : Order service calls the login page of account service
- - GET /getItems/ : Account service gets the items that was posted by cart from order service.

Order Component REST API

Endpoint	/create	/Order/getBy/{email}/	/cartSummary	/getOrderPage	/deleteItem/{itemId}	/Order/getItems
Purpose	Adds Order information to database	Other services can access order information by email	Adds cart items information to database	Displays item summary on order UI from database and returns order page	Deletes items on order db by itemId	Other services can access items that are ordered
Method	POST	GET	POST	GET	DELETE	GET
Data Format	JSON	JSON	JSON		URL Encoded	JSON
Expected Data	fullName, email, address, city, state, zip, nameOnCard, CCNum, expMon, expYear, CVV	fullName, email, address, city, state, zip, nameOnCard, CCNum, expMon, expYear, CVV	itemId, title, description, price		itemId to delete item from db	

Example Data	{"fullName" : "string", "email":"string", "address":"string", "city" : "string", "state":"string", "zip": number, "nameOnCard": number, "CCNum": number, "expMon": number, "expYear": number, "CVV": number
	
	}
		{
		
		"fullName" : "string", "email":"string", "address":"string", "city" : "string", "state":"string", "zip": number, "nameOnCard": number, "CCNum": number, "expMon": number, "expYear": number, "CVV": number
		
		}
			{
			
			"itemid": "Long", "name": "string", "description": "string", "prince": "Double"
			
			}
					/deleteItem/{itemId}	{
						
						"ItemId": "long", "title": "String", "description": "String", " price": "Double"
						
						}

Table 1: Order Service RESR endpoints