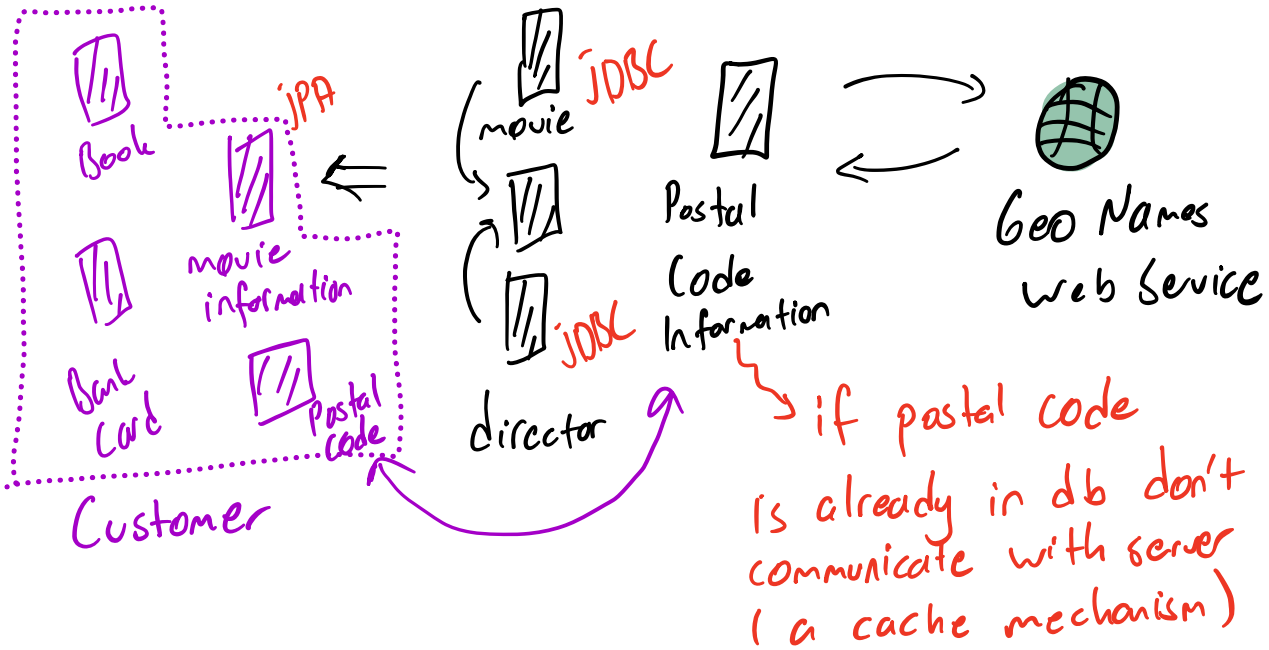
This application covers lots of attempts related to spring boot fundamentals and i listed some of them below.

- Spring Boot side
  - exception handling
  - basic principles like mapstruct
  - different update approaches, layered architec. etc.
  - custom exception handling layer by layer
- DB Side
  - testing / database file / h2 memory db / PostgreSQL using Test containers
  - a basic cache mechanism between entity and web service. if the address is in the db, data is fetched directly from db without interacting with web service.
  - JDBC principles, using queries and functions in spring boot and connection with JPA Repositories
  - Hibernate and JPA principles, table relations include native, jpql, repository queries
- Testing Side
  - Unit and Integration tests using different dbs and docker container
- Security Side
  - Basic and Bearer Token



- movie

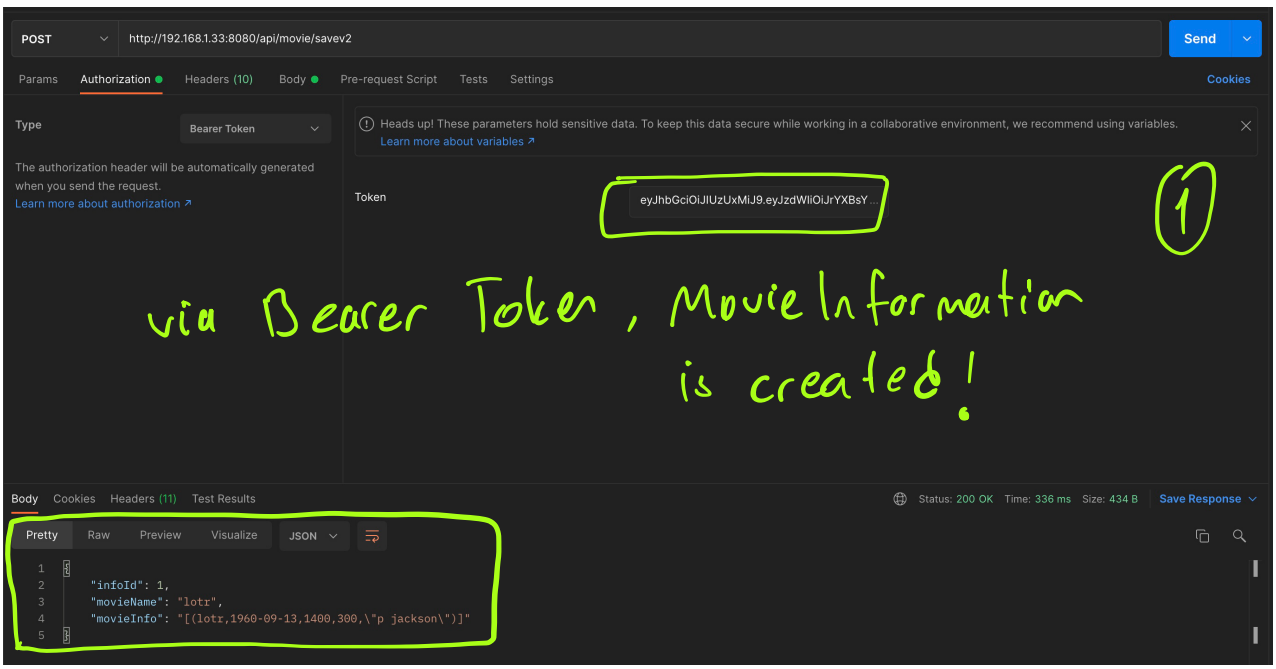
	movie_id	name	scene_time	rating	cost	imdb
1	1	Inception	2022-03-07	1000	500	8.6
2	2	kaplan's movie	20-01-01	400	900	4.5
3	3	lotr	1960-09-13	1400	300	8.7

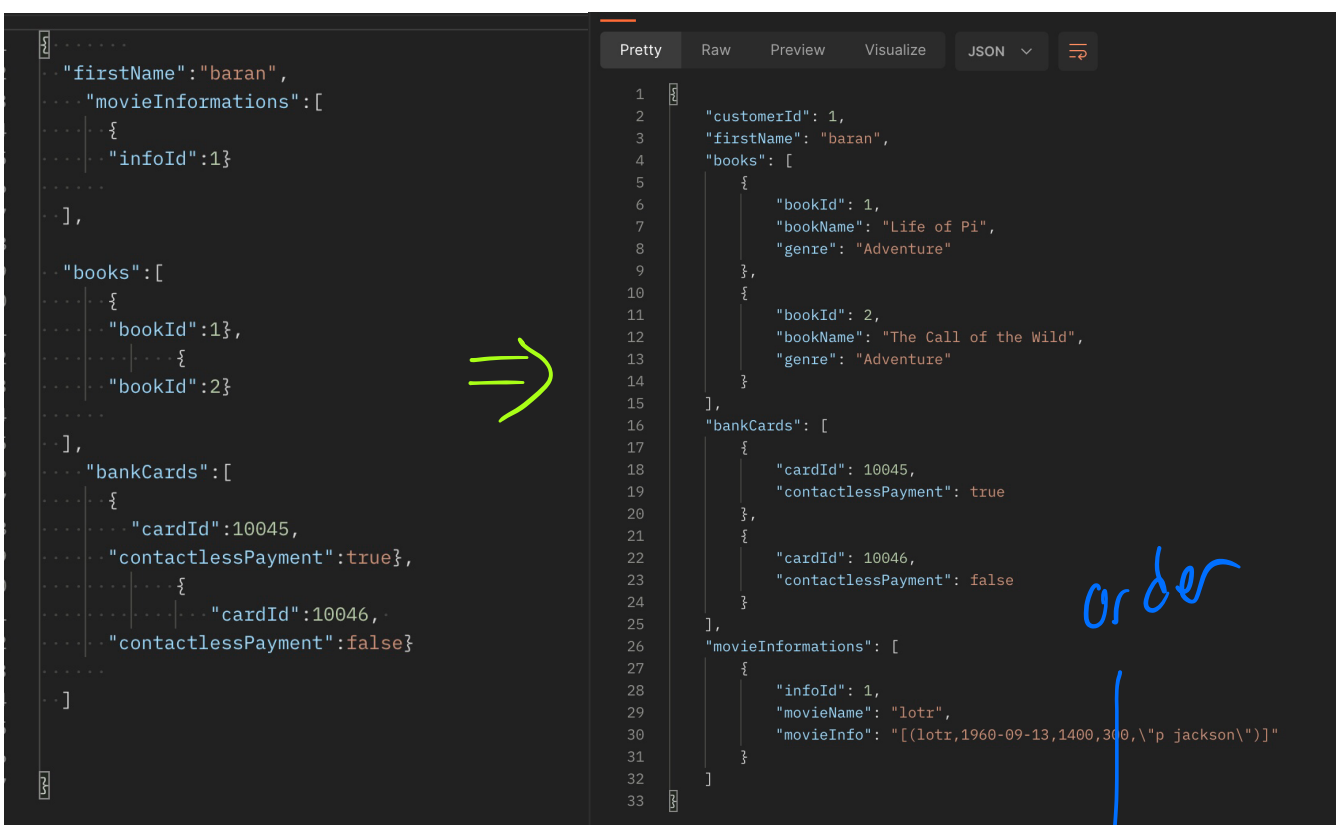
- director

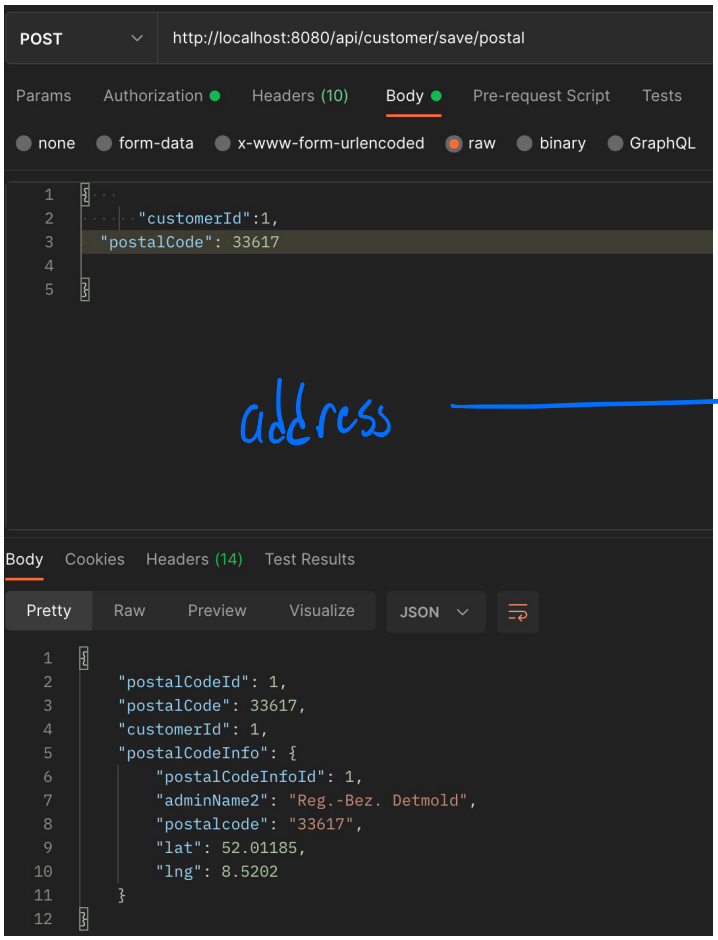
	director_id	name	birt_date
1	1	kaplan	1989-10-05
2	2	nolan	1970-07-30
3	3	p jackson	1961-10-31

- movie_to_director

	movies_to_director_id	movie_id	director_id	movie_count
1	1	1	1	4
2	2	2	2	4
3	3	3	3	4


POST
http://192.168.1.33:8080/api/movie/savev2
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Type
Bearer Token
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.
Learn more about variables
Token
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJrYXBsY
1
via Bearer Token, Movie Information is created!
Body
Cookies
Headers (11)
Test Results
Status: 200 OK
Time: 336 ms
Size: 434 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"infoId": 1,
"movieName": "lotr",
"movieInfo": "[(lotr,1960-09-13,1400,300,\"p jackson\")]"
}


{
"firstName":"baran",
"movieInformations":[
{
"infoId":1}
],
"books":[
{
"bookId":1},
{
"bookId":2}
],
"bankCards":[
{
"cardId":10045,
"contactlessPayment":true},
{
"cardId":10046,
"contactlessPayment":false}
]
}
Pretty
Raw
Preview
Visualize
JSON
{
"customerId": 1,
"firstName": "baran",
"books": [
{
"bookId": 1,
"bookName": "Life of Pi",
"genre": "Adventure"
},
{
"bookId": 2,
"bookName": "The Call of the Wild",
"genre": "Adventure"
}
],
"bankCards": [
{
"cardId": 10045,
"contactlessPayment": true
},
{
"cardId": 10046,
"contactlessPayment": false
}
],
"movieInformations": [
{
"infoId": 1,
"movieName": "lotr",
"movieInfo": "[(lotr,1960-09-13,1400,300,\"p jackson\")]"
}
]
}
order



Customer

has order and address