# PROJECT PROPOSAL

## Project Description :

Simulation of an online shopping platform that allows for the users to buy their items in internet with two different options: the first one is to check if the item is available in a shop and reserve it. The other option is to buy directly on internet and to have it delivered to a locker. We will use 5 different project which will have different roles : the client, the storehouse, the shops, the lockers and the manager. Each of those projects will have their own components which will interact between each other.

### Description of EACH of the controller operations:

Each of those controllers will belong to their own project, (i.e. manager for the manager project, like for the micro 1, 2, 3 lab we did) that will run together.

Manager : /shop/manager

Add item (name, quantity, price, description) -> Add an Article in the storehouse  
Remove item (name, quantity) -> Remove article from storehouse  
Remove All Item (name) -> Remove all article from storehouse

Clients : /shop/clients

Ask Availability in Shop(name) -> Will return a boolean if yes or no the article is available in shops  
Reserve Item In Shop (name, quantity, shop)   
Buy and Deliver Item (name, quantity) -> return token + lockerID  
Take from locker (lockerId, token)  
SignIn (Name, password)  
Register (Name, password, Register)

Storehouse : /shop/storehouse  
 Send item (name, quantity)

Shop: /shop/storehouse  
 Say availability (name, quantity)  
 Ask item (name, quantity)  
 Reserve item (name, quantity)

Lockers :/shop/locker  
 Lock (lockerId)  
 Unlock (lockerId, token)

### Description of all the Components and their jobs :

Manager’s project:

ArticleComponent:  
 @PostContruct INIT  
 Check if items in DB, if not add items into DB  
StoreHouseComponent :

@PostContruct INIT  
 Check if items in DB, if not add items into DB

Add Item : Add new Item (name, quantity, price, description)

Remove Item : Remove Item (name, quantity)  
LockerComponent :

@PostConstruct Init   
 Check if empty, if yes, init lockersDb.

Clients’s project:

LockerComponent:

Reserve Locker : Reserve Locker () -> return lockerID, token   
 Take from locker : Take from locker (lockerId, token)

StorehouseComponent:  
 Remove Item : Buy and Deliver Item (name, quantity)

ShopComponent:  
 Ask availability : Ask Availability in Shop(name)  
 Reserve Item : Reserve Item In Shop (name, quantity, shop)

ClientComponent:  
 SignIn : SignIn (Name, password)  
 Register : Register (Name, password, Register)

StoreHouse’s project:

ShopComponent:  
 Send Item to shop : Send item (name, quantity)

Shop’s project:  
 StoreHouseComponent:  
 Ask item : Ask item (name, quantity)

ClientComponent:  
 Say availability : Say availability (name, quantity)   
 Reserve Item : Reserve item (name, quantity)

Locker’s project:  
 LockerComponent:

Lock : Lock (clientId)  
 Unlock : Unlock (lockerId, token)

### Descriptions of the Entities and Repositories :

*Those datas can be NULL.*

Clients:  
Long id, String name, String password, String email, String token, Long lockerId, Long CommandId

Articles :  
Long id, String name, String description, Double price

StoreHouse:  
Long idArticles, String articleName, Double articlePrice, Integer quantity

Shops:  
Long Id, long idArticles, String articleName, Integer quantity, Double articlePrice

Lockers:  
Long Id, String token, Boolean isFree

Only magic queries used in this project.

In this zip file you can find the beginning of our implementation, we have the 5 different project which can run simultaneously. For the moment they are all connected to the same database because we didn’t create the methods to communicate between the project.  
But in the future they will have different database.  
We have implemented all the entities and their corresponding repository with the queries we needed for the moment.  
Each component has only one method for the moment in order to test if we the project receive ou Postman request.

You will also find a Postman Collection in order to test all the component. Take care that the client ID for the locker request may be different for you (check in the DB).