Design document

Version	Start date	End date	Description
1.0	05.10.2022	07.10.2022	Adding how I guarantee SOLID, UML, C4 diagram to layer 3 and, reasons for using spring boot and what other frameworks I am using in my software solution.

How I guarantee SOLID:

I guarantee solid the following way:

S - Single responsibility: I have classes that are ment to do only one thing. In my case I have classes that request or respond with data, as an example I give *"CreateStaffRequest"*, his only purpose is to get the data required to hire staff. I also have use case classes such as *"CreateStaffUseCase"*, which only does the creation of the staff.

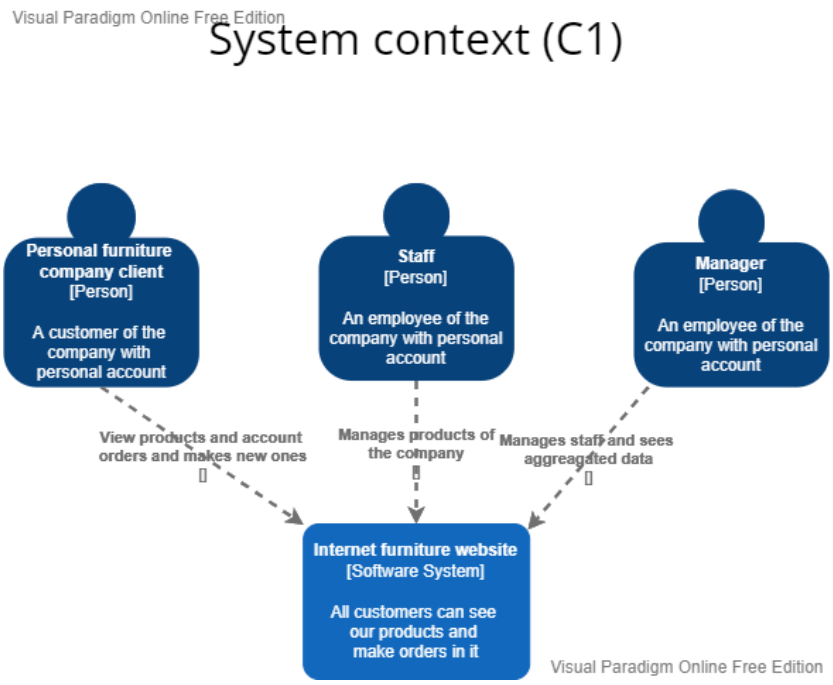
O - Open/Closed: Currently I don't have example but as soon as I add real database in the project, I will have one. I will have interface with all the methods that are same between the real database and the fake one and they will only have their implementation in them. This way if it happens that the company want a new way of storing their data, I can easily add it.

L - Liskov substitution: In my case I don't have classes that inherit partly their interface (leaving override methods empty or throwing errors if they are called).

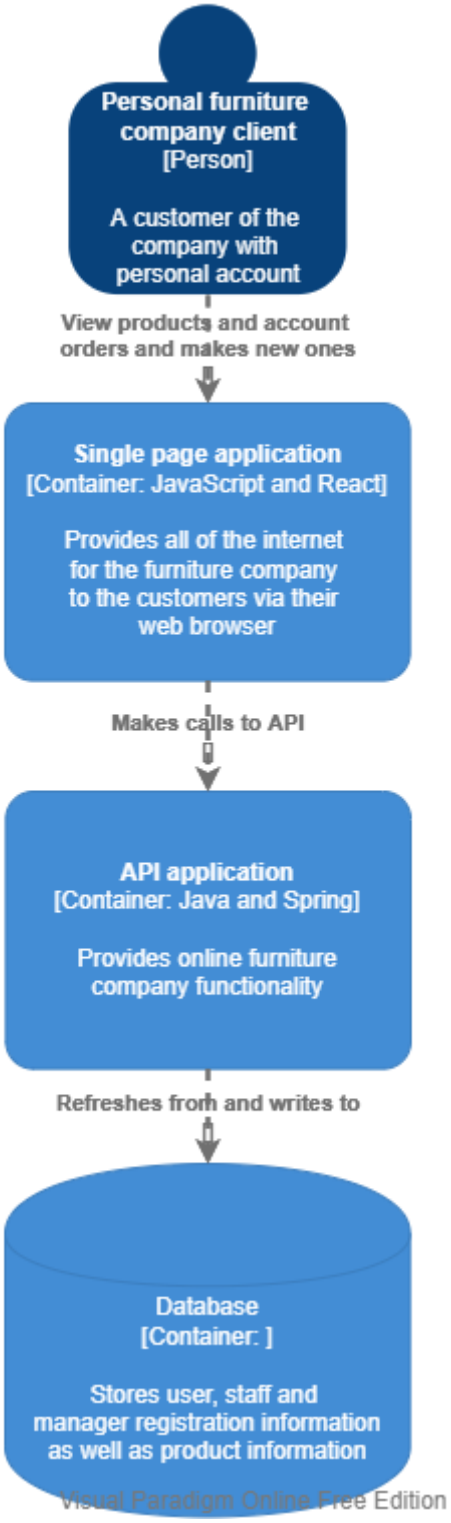
I - Interface segregation: I don't force my clients to use classes that own methods that have nothing to do with them (the clients). I achieve this by separating my program base on users and their needs.

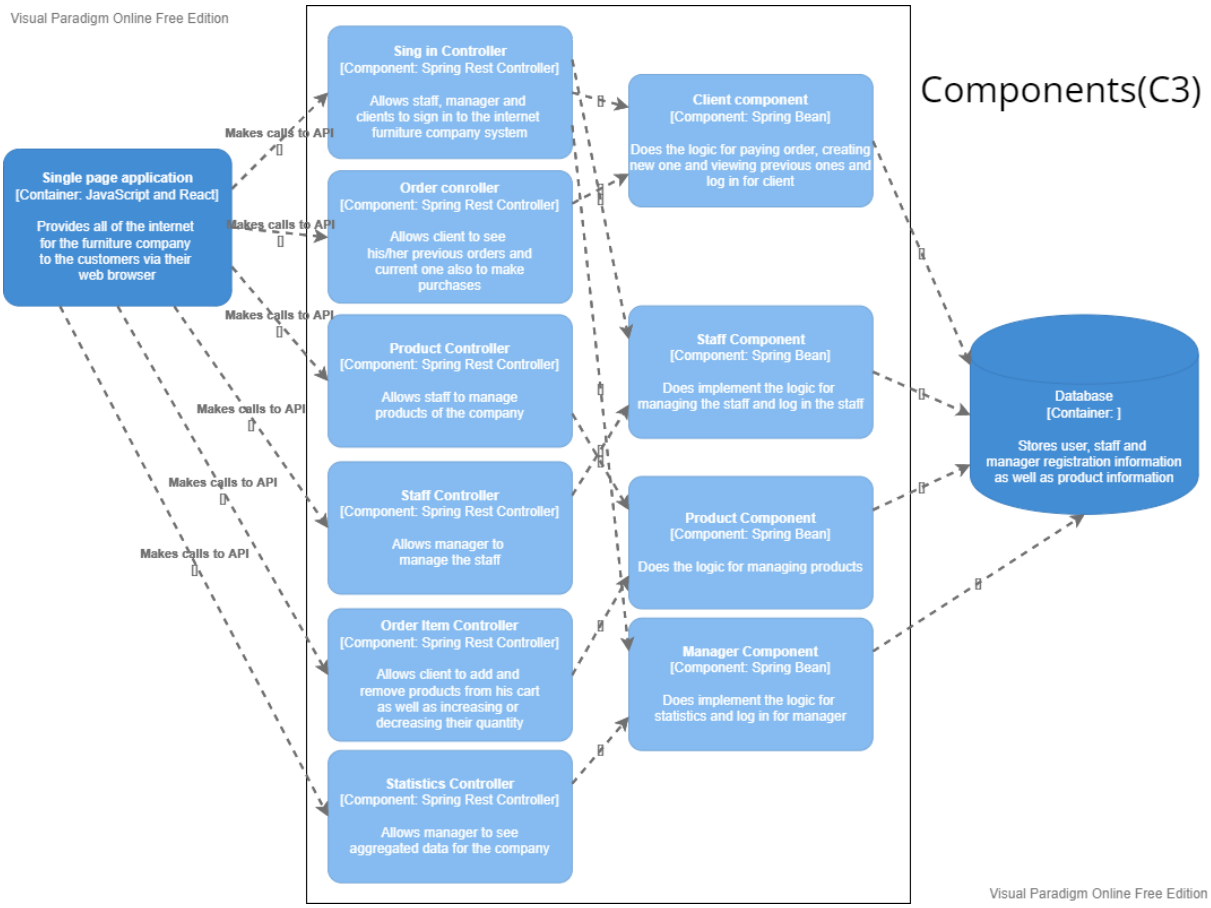
D - Dependency inversion: I guarantee it by using interfaces as an example I can give the following one: When one class has to reference to another class at another package it does not do it directly but through interface which is inherited by the targeted class. This can be seen with my controllers calling mu use case methods.

C4 architecture diagrams:



Containers (C2)





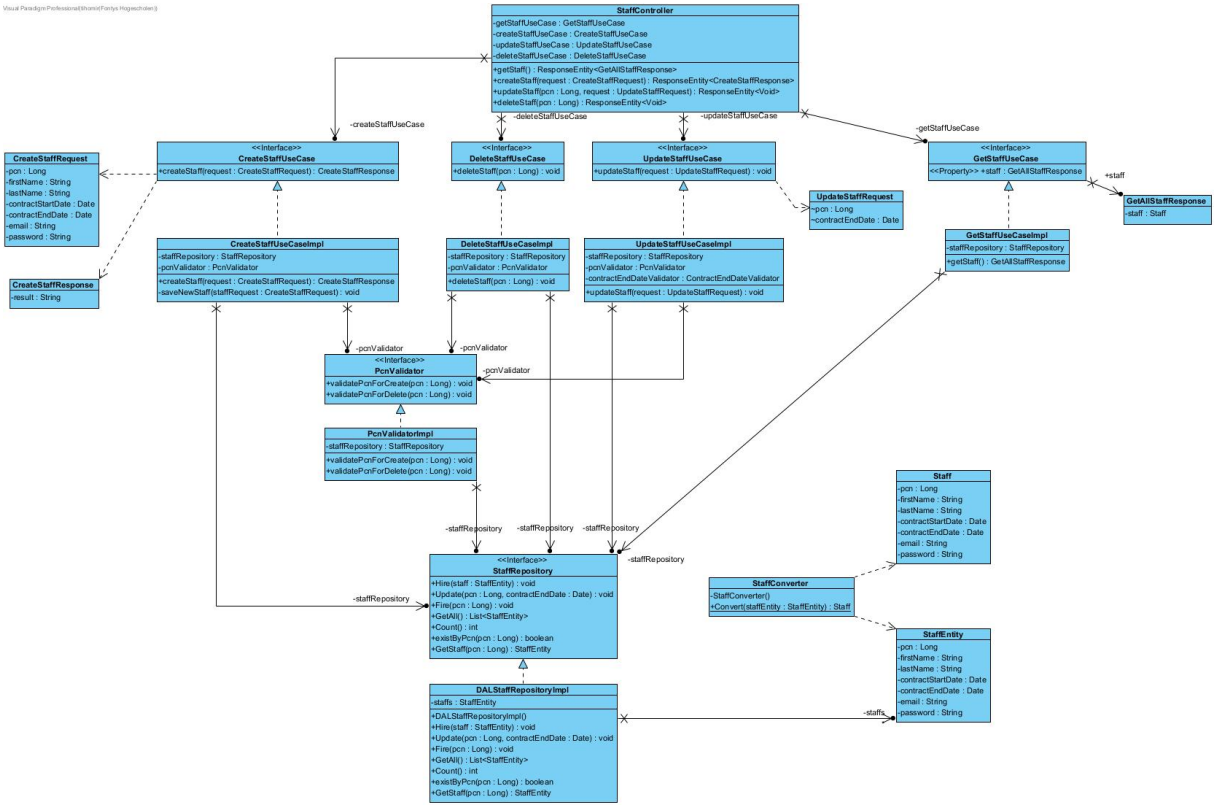
Important design decisions:

Why I am using spring boot: Spring has flexible libraries (Lombok) that are trusted by many developers also has very good frameworks at its core such as Dependency Injection (DI). Also because of its popularity you can easily find what you need. Another thing is that all major IDEs provide support for it. Spring allows to easily link parts of the program and later with same ease test them.

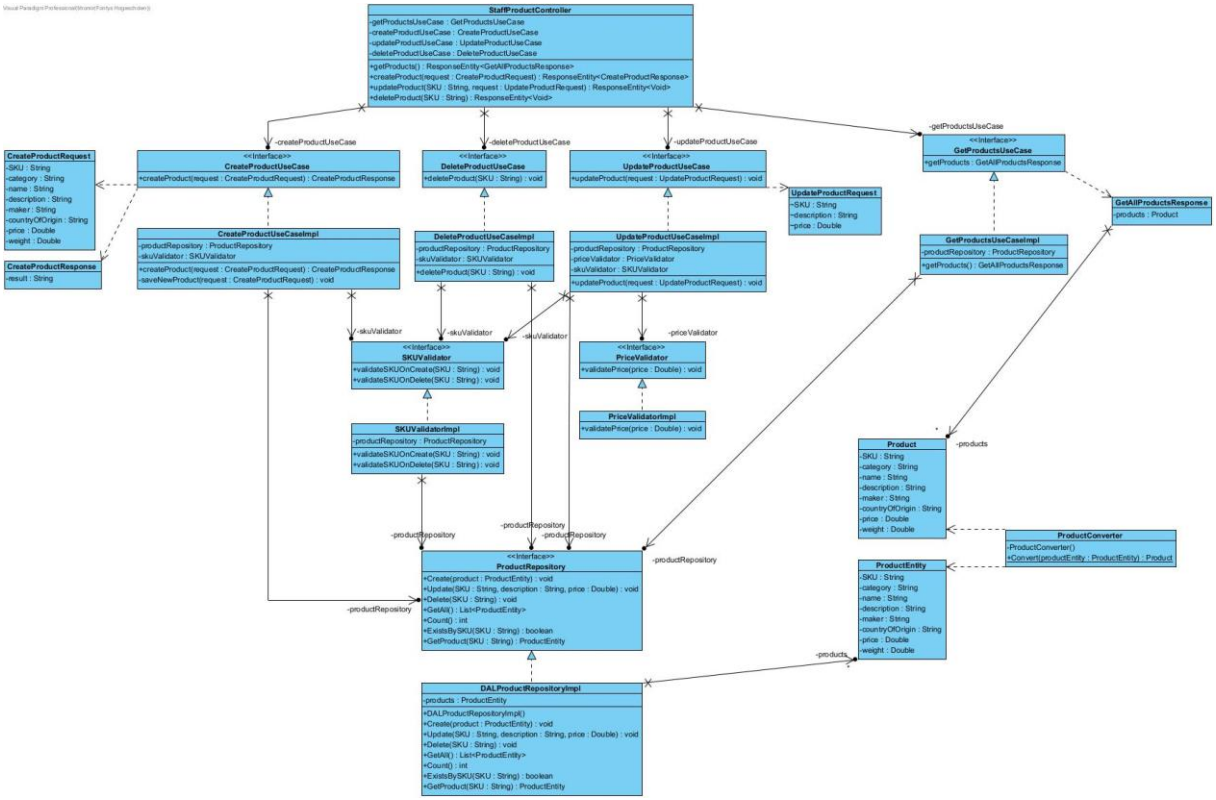
What frameworks I am using and why: I am using spring framework as I already said. Another one is React. The reasons that I am using it are its superior performance for running apps, its usability also you can reuse components of your program that will help building it quickly and with less code, also its easy to learn and it the most used web framework, so it is easy to find what you need.

UML class diagrams:

UML for staff part of the program:



UML for product part of the program:


StaffProductController
-getProductsUseCase : GetProductsUseCase
-createProductUseCase : CreateProductUseCase
-updateProductUseCase : UpdateProductUseCase
-deleteProductUseCase : DeleteProductUseCase
+getProducts() : ResponseEntity<GetAllProductsResponse>
+createProduct(request : CreateProductRequest) : ResponseEntity<CreateProductResponse>
+updateProduct(SKU : String, request : UpdateProductRequest) : ResponseEntity<Void>
+deleteProduct(SKU : String) : ResponseEntity<Void>
-createProductUseCase
-deleteProductUseCase
-updateProductUseCase
-getProductsUseCase
CreateProductRequest
-SKU : String
-category : String
-name : String
-description : String
-maker : String
-countryOfOrigin : String
-price : Double
-weight : Double
<<Interface>>
CreateProductUseCase
+createProduct(request : CreateProductRequest) : CreateProductResponse
<<Interface>>
DeleteProductUseCase
+deleteProduct(SKU : String) : void
<<Interface>>
UpdateProductUseCase
+updateProduct(request : UpdateProductRequest) : void
UpdateProductRequest
-SKU : String
-description : String
-price : Double
<<Interface>>
GetProductsUseCase
+getProducts : GetAllProductsResponse
GetAllProductsResponse
-products : Product
CreateProductResponse
-result : String
CreateProductUseCaseImpl
-productRepository : ProductRepository
-skuValidator : SKUValidator
+createProduct(request : CreateProductRequest) : CreateProductResponse
-saveNewProduct(request : CreateProductRequest) : void
DeleteProductUseCaseImpl
-productRepository : ProductRepository
-skuValidator : SKUValidator
+deleteProduct(SKU : String) : void
UpdateProductUseCaseImpl
-productRepository : ProductRepository
-priceValidator : PriceValidator
-skuValidator : SKUValidator
+updateProduct(request : UpdateProductRequest) : void
GetProductsUseCaseImpl
-productRepository : ProductRepository
+getProducts() : GetAllProductsResponse
-skuValidator
-skuValidator
-skuValidator
-priceValidator
<<Interface>>
SKUValidator
+validateSKUOnCreate(SKU : String) : void
+validateSKUOnDelete(SKU : String) : void
<<Interface>>
PriceValidator
+validatePrice(price : Double) : void
SKUValidatorImpl
-productRepository : ProductRepository
+validateSKUOnCreate(SKU : String) : void
+validateSKUOnDelete(SKU : String) : void
PriceValidatorImpl
+validatePrice(price : Double) : void
-products
Product
-SKU : String
-category : String
-name : String
-description : String
-maker : String
-countryOfOrigin : String
-price : Double
-weight : Double
ProductConverter
-ProductConverter()
+Convert(productEntity : ProductEntity) : Product
-productRepository
-productRepository
-productRepository
-productRepository
-productRepository
<<Interface>>
ProductRepository
+Create(product : ProductEntity) : void
+Update(SKU : String, description : String, price : Double) : void
+Delete(SKU : String) : void
+GetAll() : List<ProductEntity>
+Count() : int
+ExistsBySKU(SKU : String) : boolean
+GetProduct(SKU : String) : ProductEntity
ProductEntity
-SKU : String
-category : String
-name : String
-description : String
-maker : String
-countryOfOrigin : String
-price : Double
-weight : Double
-products
DALProductRepositoryImpl
-products : ProductEntity
+DALProductRepositoryImpl()
+Create(product : ProductEntity) : void
+Update(SKU : String, description : String, price : Double) : void
+Delete(SKU : String) : void
+GetAll() : List<ProductEntity>
+Count() : int
+ExistsBySKU(SKU : String) : boolean
+GetProduct(SKU : String) : ProductEntity