# Invoice System

* 24 May, I am thinking about a project to work on over the summer, I have a couple of ideas but that’s the problem, too much thinking and nothing gets done
* I have finally decided to start working on Invoice System, 2 points for my creativity there I worked on a system like this for my 2nd year project in Java.
* I hated it though as we were forced to use Java swing in 2023. Anyway Since I have the database structure sorted out I will challenge myself to learn Springboot instead.
* 1 problem, I have to figure out how I will deal with role based authentication. I have a table for invoices, customers products etc. but it is meant for a single user for the system.
* It doesn’t sound like too much of a challenge but It is something I have to think about.
* I am yet to decide on whether or not I should use vanillaJs, state management will be an issue though, I haven’t thought about checking out how that is done. Easier to use a framework
* Angular/Vuejs I will decide later on depending on the learning curve. I like how the structured angular programs are.
* So the task for the next two weeks during my off days is to learn about micro services and how the MVC model is implemented in spring.
* Ok I have basic understanding of how the micro services work, well you can only learn so much from reading.
* I dived into code basically the idea is to have a gateway running on port 8080
* After building the gateway we will then use the forward url value to determine which micro service we are targeting
* Port 8081 will be for customers so to access this via our default gateway the api call will be localhost:8080/customer
* Every URL that starts like this will take us to the customer micro service.
* I have done the necessary CRUD operations for customer (I am using soft delete)
* Once we introduce role based authentication later on then I will add permanent delete and viewing of all soft deleted fields.
* Then next challenge is to work on the CRUD operations for products micro service which will run on Port 8082.
* The api call from the gateway will be localhost:8080/products
* Let me get to it then we I will deal with the challenges I foresee already i.e. getting the gateway to make the right call
* Kind of weird, when using a custom update or delete query, you have to specify that the query is being modified with an annotation and also for the update to be successful the method in the service must be annotated with Transaction.
* There is a lot to learn in this spring framework
* Now we need to consider communication between the micro services, yes: the time has arrived.
* We don’t want a lot of queries to come from the front end targeting different api’s, in my limited knowledge that seems like bad practice.
* An example we are going for is the front end will hit the api: localhost:8083/invoices
* This will get all invoices with and the data should be as follows: [{status, due date, date created, invoice number, customer name, amount due}
* As you can see customer name is not part of the customer table, so we need to use the customerId in the invoice micro-service, communicate with the customer micro-service to get the customer name with that ID.
* I will need to figure out if I should extract the whole customer object or I can just get a response with just the name
* So next step is understanding how Rest Template works (although I hear it will be deprecated to use asynchronous java programming. What a time to be alive)
* Also service discovery
* When front end hits the api: localhost:8083/invoice/{invoiceId}
* This will get data as follows: {customerObject, InvoiceObject, ProductObject} ont end
* 4 microservices will be affected in this call namely customer, invoice, product & inv\_prod
* Tonight I have a lot of work cut out for me.
* Jumping into frontend now. I am behind time on a couple of things but I should start having visuals on the front end. I will keep the design simple.
* A dashboard powered by Angular material, I will start with customers.
* We will use the concept of lazy loading to try and improve the apps performance. This will be achieved by having modules for each component then when the associated is routed is activated the component will then be loaded
* Ok I have the base setup but I don’t like it, I mind abandon the idea of using angular material and just use bootstrap but I will come back to that.
* Managed to work on service discovery using Netflix-Eureka-Server/Client.
* Everything seems like a challenge till you actually do it, it wasn’t as bad as I thought it will be in terms of complexity.
* The invoice service has been setup I now need work on then making calls amongst the services to get the data I need to send to and from the client.