INTERNSHIP KEY POINTS

- Pragna Sree Pallam

1.CRUD Operations:

Api’s: create order

Read order payload

Update Order

Delete Order

2.Cancel Orders

Api’s:

CRUD + cancel order api(changes the status in order payload and reflects in cancelled orders table)

3.Spring Scheduler:

Used CRON Expressions to schedule tasks

4.Quartz scheduler:

Key points:

Quartz dependency in pom.xml

Job(Class)

Job Detail(metion about what our job is i.e,Job class)

Job Trigger(Triggers the job at customized timings /intervals)

5.Email scheduling Application(Quartz):

Schedules an email with required Subject, Body to be sent at a particular time

6.Microservices:

Cancel order -> uses the sendMail api from email schedulers to send the cancelled order mails and updates the db using a flag.

6.**Mini OMS(Order Create + Order Validate):**

Api’s:

CURD

Databases:  
Order(Oreder payload)  
Entity\_proxy\_table  
Service\_Status\_detail

Retry:  
Quartz

Quartz pools the failed order(Entity\_proxy\_T) for regular intervals of time and reprocess the order  
Once the order successfully completes the tasks in create, the order is moved to validate.  
Used Webclient for communication between create and validate applications.

7.CAMUNDA

Workflows worked on:

POMS\_AZW\_ASYNC\_RETRY\_OR\_NA  
 POMS\_AZW\_SMF\_RETRY\_STANDARD\_SMFOR\_NA  
 POPS\_OMS\_DAILY

Installation:

Camunda c8run  
[Release c8run-8.7 · camunda/camunda · GitHub](https://github.com/camunda/camunda/releases/tag/c8run-8.7/)

Camunda Modeller:

[Download The Camunda BPMN / DMN Process Modeler | Camunda](https://camunda.com/download/modeler/?_gl=1*1l46izu*_gcl_au*MjAyNzA2ODEwNS4xNzQ0MzUyOTUx*FPAU*MjAyNzA2ODEwNS4xNzQ0MzUyOTUx*_ga*NTc5NjAwNTcuMTc0NDM1Mjk1Mw..*_ga_4EYN8X5FNR*czE3NDY2OTA0MDYkbzM2JGcxJHQxNzQ2NjkwNDEzJGowJGwwJGgxNjM1OTU0MDg.)

Needs docker

Conclusion on Camunda:

Positives:

* Perfectly inclines with the workflow
* Simple to define and work with jobs
* Have connectors where we can directly give the api’s that need to be triggered. No much configurations required
* Good UI/dashboard for monitoring the jobs,instances,status
* Connector for pageduty integration for alerts
* Custom feature design(user task) can integrate with springboot and perform any customized tasks
* Gives error message description when any job us failed.
* Monitor & retry failed jobs
* Failure diagnostics
* Enterprise-grade + Low-code
* Cost efficiency

Drawbacks:

* Loging (when not using springboot and directly configuring api’s through connector, the error messages sometimes may not be enough to debug the problem, all the logs might be required)
* Pause/resume the scheduler(Camunda 7 has the toggle button to pause/resume the timer, but for Camunda 8, the option is not available yet).
* Once the job is failed, the instances were not being stopped until it gets fixed. i.e, scheduler keeps working and triggering new instances which will eventually fail(can do something if we can stop the scheduler with logic)

8.Kestra

Did not get a chance to go deep but kestra has a really good dashboard, meets all the requirements. Has a wide range of extensions to connect with various softwares like pagerduty. No problem with pause/resuming the scheduler. Works majorly with yaml scripting. Could be a potential alternative