# Entrega para el Code Chanllenge

La solucion planteada esta compuesta por 2 apis llamadas:

* machine-administrator
* machine-software

Ambas estan desarrolladas sobre un arquetipo base generado con Spring Initializr para Maven con Java 11, y empleando Swagger Ui para documentar la api.

# Machine-Software

Es el encargado de la gestión de cada máquina expendedora.

### Base de datos

Esta integrado con una base de datos embebida H2 y configurado en modo create-drop para hacer sensillo el testing.

Luego de cada ejecucion el sistema inicia sin datos y ejecuta el script data.sql .

Si se deseara que los datos quedaran persistidos luego de finalizar el tiempo de ejecución la misma se puede cambiar desde el aplication.properties → spring.jpa.hibernate.ddl-auto=create-drop

### **Ejecucion del software**

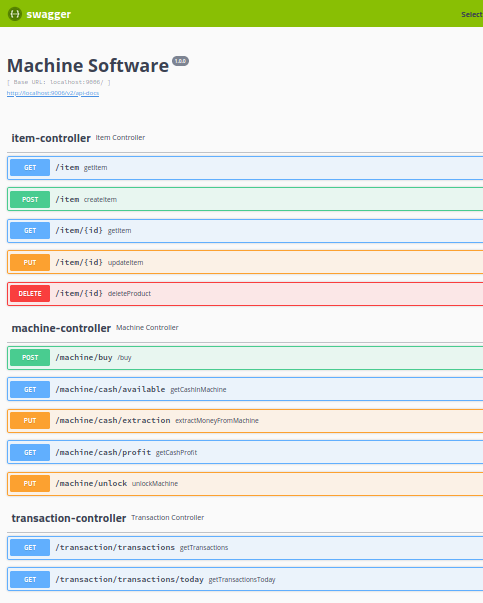
Se desarrollo un data set para emular 6 máquinas expendedoras, que pueden correrse de forma individual o simultanea.

Para ejecutar los diferentes máquinas debe parametrizarse el profile de spring desde la linea de comando para la maquina que se quiera correr.

Mas info https://docs.spring.io/spring-boot/docs/1.2.0.M1/reference/html/boot-features-profiles.html

### **Acceso a la API**

Cada maquina expone un swagger para visualizar los metodos disponibles y la paramtrizacion de los mismo. La maquina por defecto corre en el puerto 9006, y la url de su swagger es: <http://localhost:9006/swagger-ui.html>

****

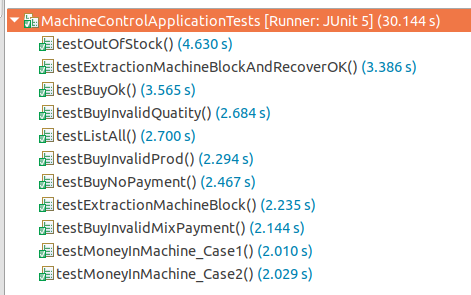
### **Consideraciones especiales**

* Se tomo como unidad de valor el centavo, de forma tal que todo los valores numericos representan centavos.
* Si se deseara pagar con tarjeta de credito, el numero ingresado debe pertenecer a una tarjeta VISA ya que la integracion realizada solo prevee este proveerdor, de lo contrario acusara que la tarjeta es invalida.

### **Mejoras y pendientes**

Dado la extension del problema, no pudieron realizarse algunas o ejecutarse apropiadamente los requerimientos detallados a continuacion:

* La cobertura de tests es escasa, mas alla de eso se opto por implementar tests end to end para cubrir la completitud de los flujos elegidos, desde la invocacion a traves del controller hasta la persistencia en la BD. Queda pendiente el desarrollo de los restantes test para darse solidez al desarrollo.



* Refactorizar el error handling, hoy en dia todas las exepciones son capturadas y expresadas como un error 400 aunque lo correcto seria que cada una retorne el codigo de error correspondiente
* Falta modelar las RIR en el script sql para mejorar le integridad de datos.
* Seria interesante mejorar la integracion entre las maquinas y su administrador implementando un diseno de microservicios donde cada maquina toma un puerto aleatorio y ser registra mediante una instancia de eureka a su administrador, y mediante la misma se comunican.
* Seria interesante aplicar Spring Security y JWT para evitar almacenar las passwords crudas.

# Machine-**Administrator**

Al igual que la api antes detallada cuenta con un Swagger para visualizar los servicios expuestos y puede accederse a traves de la url http://localhost:8080/swagger-ui.html

