Ejercicio carrito API Rest

1.- Breve descripción del ejercicio

En este ejercicio se pide completar la implementación del carrito de la compra desarrollado en el primer seminario estableciendo todas sus comunicaciones entre los módulos de este mediante el uso de una API Rest.

Hemos basado nuestra versión inicial del carrito para poder desarrollar la API Rest, por lo que se puede mantener la visión de módulos del carrito. Actualmente, esta "app" sigue constando de diferentes archivos/módulos, siguiendo con su versión modular que permite cambiar y conectarse a los diferentes módulos según se vayan necesitando en el proyecto.

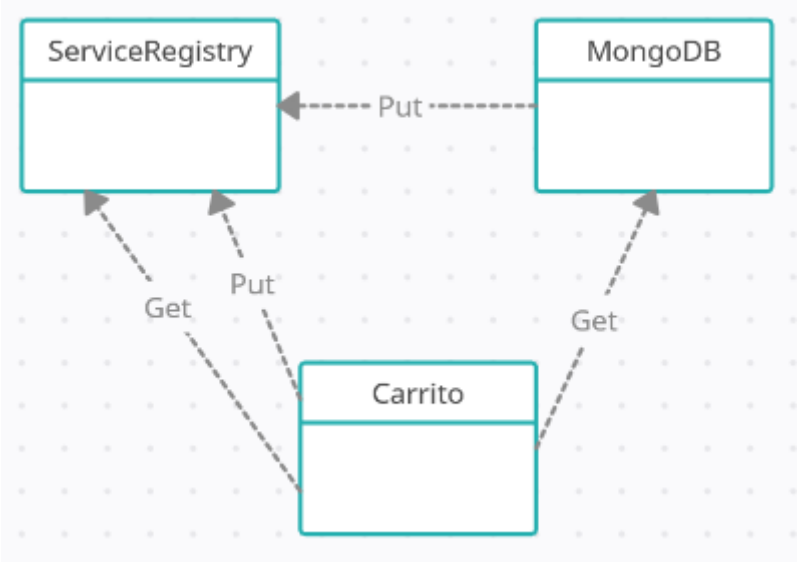

Figura 1.- Diagrama de clases de la solución presentada.

Para desarrollar esta actividad se ha decidido reescribir el código programado en *TypeScript* y pasarlo a *JavaScript*.
Además, en vez de almacenar los datos localmente, se ha optado por utilizar el servicio de base de datos en la nube de Mongo, llamado *MongoDB Atlas*, para almacenar los productos insertados por el script que gestiona las conexiones con MongoDB.

2.- Explicación de las clases propuestas en la solución

En este caso la solución explorada cuenta con 4 archivos principales escritos en *Javascript* y sus correspondientes servicios y archivos para ejecutarlos. A continuación, revisaremos la funcionalidad de cada una de ellas:

- `MongoDB`: se incorporan todos aquellos métodos correspondientes al acceso a la base de datos.
- `Producto`: definición de la clase producto. Contiene atributos, constructores, *getters* y *setters* y el método `toString()`.

- `Carrito`: define el carrito propiamente. Se encarga de gestionar los productos añadiendo o borrando nuevos, además de tener control sobre ellos. Además permite mostrar al usuario los productos almacenados.
- `ServiceRegistry`: este script se encarga de registrar cada servicio en la API Rest, en este caso siendo el `Carrito` y `MongoDB`. El `ServiceRegistry`, si detecta que un servicio no se ha comunicado con él, pasados 10 segundos, lo borra, por tanto se establece un intervalo para cada servicio, en el que cada 5 segundos, se enviará una petición Rest `Put` para actualizar el *timestamp* de dicho servicio en el `ServiceRegistry`.
- Además se cuentan con clases `Service` y `Run` para cada uno de las clases principales, exceptuando `Producto`, que se encargan de gestionar las peticiones HTTP Rest y de ejecutarlos respectivamente.

3.- Pasos a seguir para su compilación y ejecución

Para probar esta funcionalidad se deberán ejecutar cada uno de los archivos `Run` en un orden determinado. Este orden será el siguiente: `ServiceRegistry` → `MongoDB` → `Carrito`. Se deberá esperar unos segundos entre la ejecución de cada uno de ellos para que las peticiones puedan encontrar su destino correspondiente (se recomienda 5 segundos). Para ejecutarlos se realizará el siguiente comando en las carpetas correspondientes a sus servicios: `node run.js`

Por último, para proveer el código fuente, se han adjuntado tanto un zip con los archivos como un link al repositorio de GitHub mostrado a continuación: https://github.com/pabletefest/SAD_MUIINF