**MAQUETACIÓN PARA LA SOLUCIÓN DE APLICATIVO.**

Teniendo en cuenta el reto de desarrollo, se plantea dar solución de manera estructurada, en la cual se tendrán en cuenta factores como, lenguajes (incluyendo tecnologías y framework), bases de datos (Normalización y gestor), y arquitectura de desarrollo.

**Lenguaje y tecnologías:** Java, Spring boot, IntelliJ (IDE)

**Almacenamiento de datos DDBB:** PostgreSql, Hibernate

**Arquitectura:** Modular, orientada a microservicios.

**Manejo de repositorios:** GitHub

**Estructuración de microservicios y definiciones de Path y Endpoint.**

De acuerdo a los requisitos del juego de carros, se definen los microservicios de la siguiente manera:

* Se crearán CRUDS para los players (pilotos), cars y lanes.
* Se creará un Endpoint para el lanzamiento del dato, que generará un numero aleatorio del 1 al 6.
* Se creará un Endpoint para la conversión de kilómetros a metros, calculando el avance del vehículo en cada acción de dado.
* Se creará un Endpoint para definir a los ganadores en el podio.
* Se creará un Endpoint que tendrá el registro de las veces que un mismo player ha ganado.
* Se creará un package (commons), que tendrá todos los elementos transversales al desarrollo, es decir, Interfaces, superclases (clase padre), clases abstractas,ClassEnum.

**Nomenclatura de los Path’s.**

* **/player/search:** listará los players creados
* **/player:** crear un nuevo player
* **/player/{id}:** actualizar (editar) un player creado
* **/player/{id}:** eliminar un player creado.
* **/car/search:** listará los cars creados.
* **/car:** crear un nuevo car
* **/car/{id}:** actualizar (editar) un car creado
* **/car/{id}:** eliminar un car creado.
* **/lane/search:** lista los lanes creados
* **/lane:** crear un lane
* **/lane/{id}:** actualizar (editar) un lane creado
* **/lane/{id}:** eliminar un lane creado.

**Importante:** se crearán validaciones en los casos, de player y car donde no deban existir mas car que player y viceversa. De la misma manera con car y lane.

En cuanto a la creación del repositorio, este llevará el mismo nombre de la solución (KartRock), y estará dividida por Branch y features. Las Branch, Master, Develop, serán las principales. En el caso de las features, tendrán la siguiente nomenclatura**: feature/xxx**, que en el caso de los Path’s anteriores, se crearán las siguientes features, **features/player**, que deberá contener el CRUD, **feature/car**, que deberá contener el CRUD, **feature/lane** que deberá contener el CRUD. En el caso de los Endpoint seguirán la misma nomenclatura, es decir, **feature/winner** que tendrá el podio de ganadores, **feature/recordWinner** que tendrá el histórico de carreras ganadas por player.