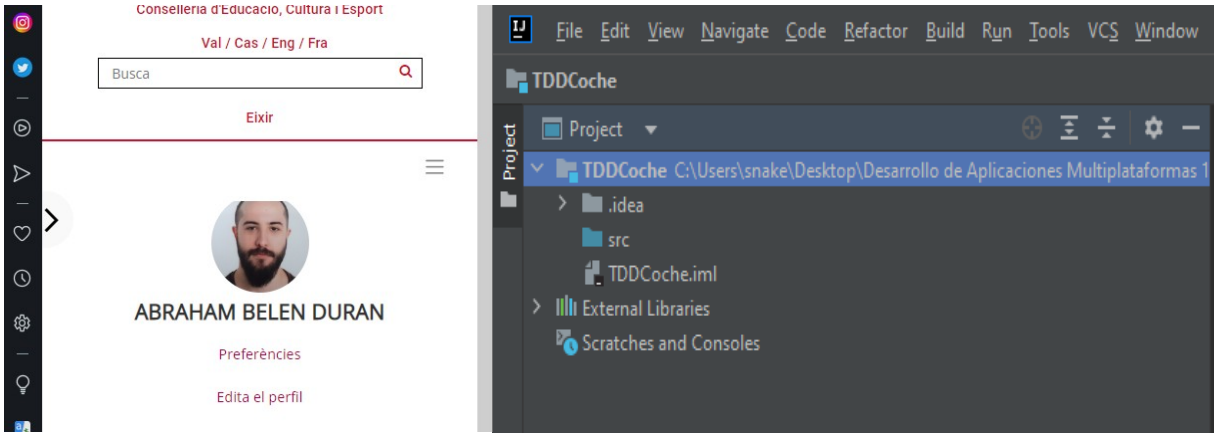
Entornos de Desarrollo

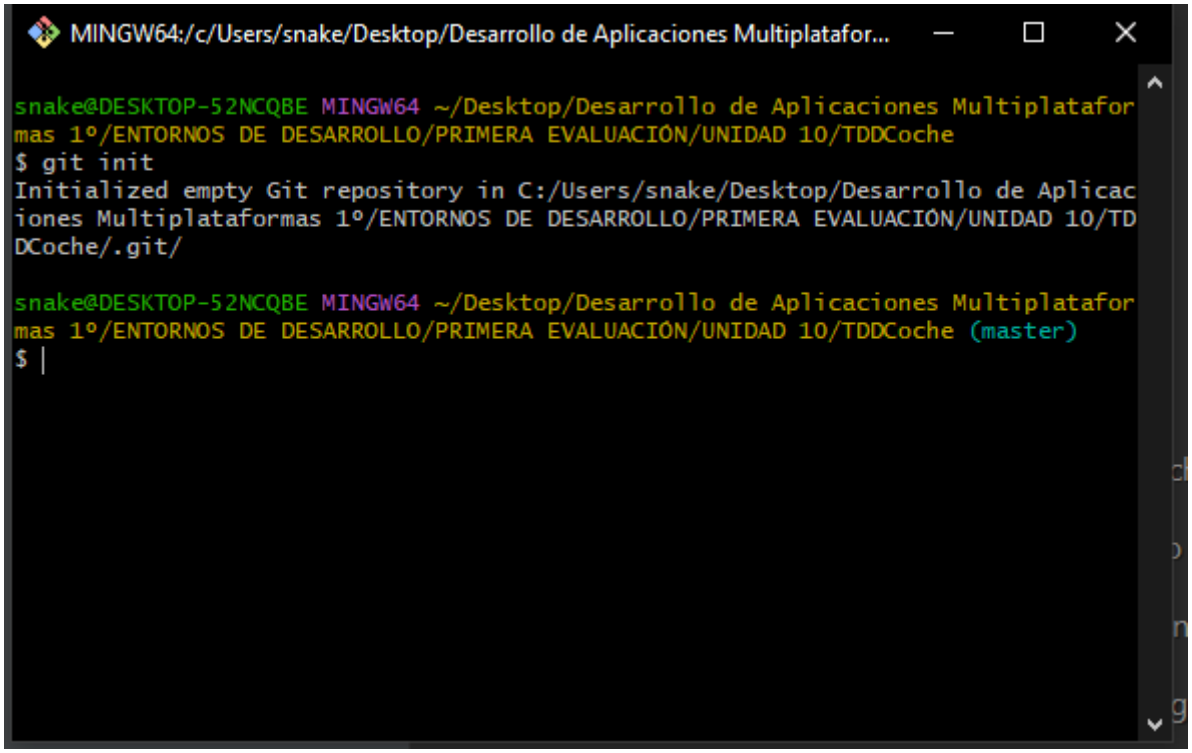
U.D 10: Patrones de refactorización.

Mi primer TDD.

Lo primero que debemos hacer es crear un proyecto de Java en IntelliJ IDEA.

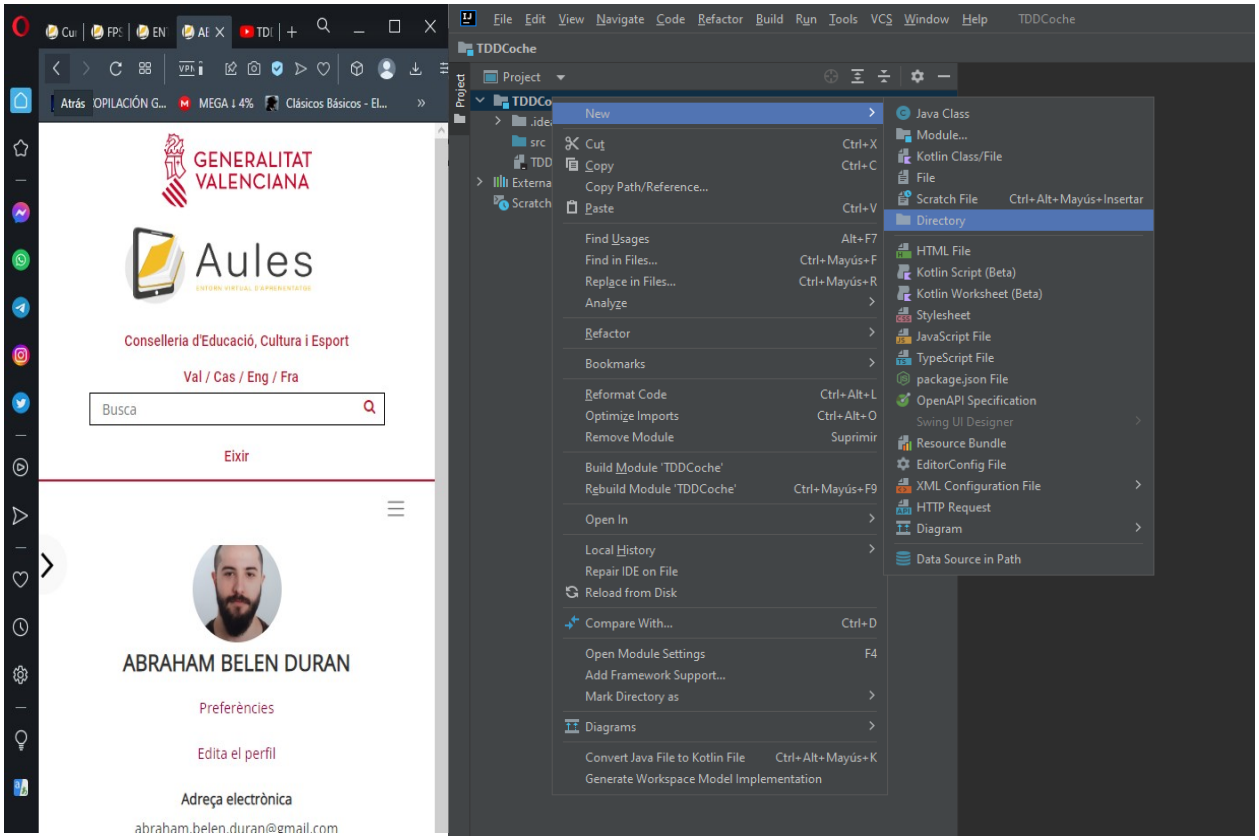


Aquí se aprecia el proyecto creado. A continuación debemos iniciar el control de versiones como se aprecia en la siguiente captura.

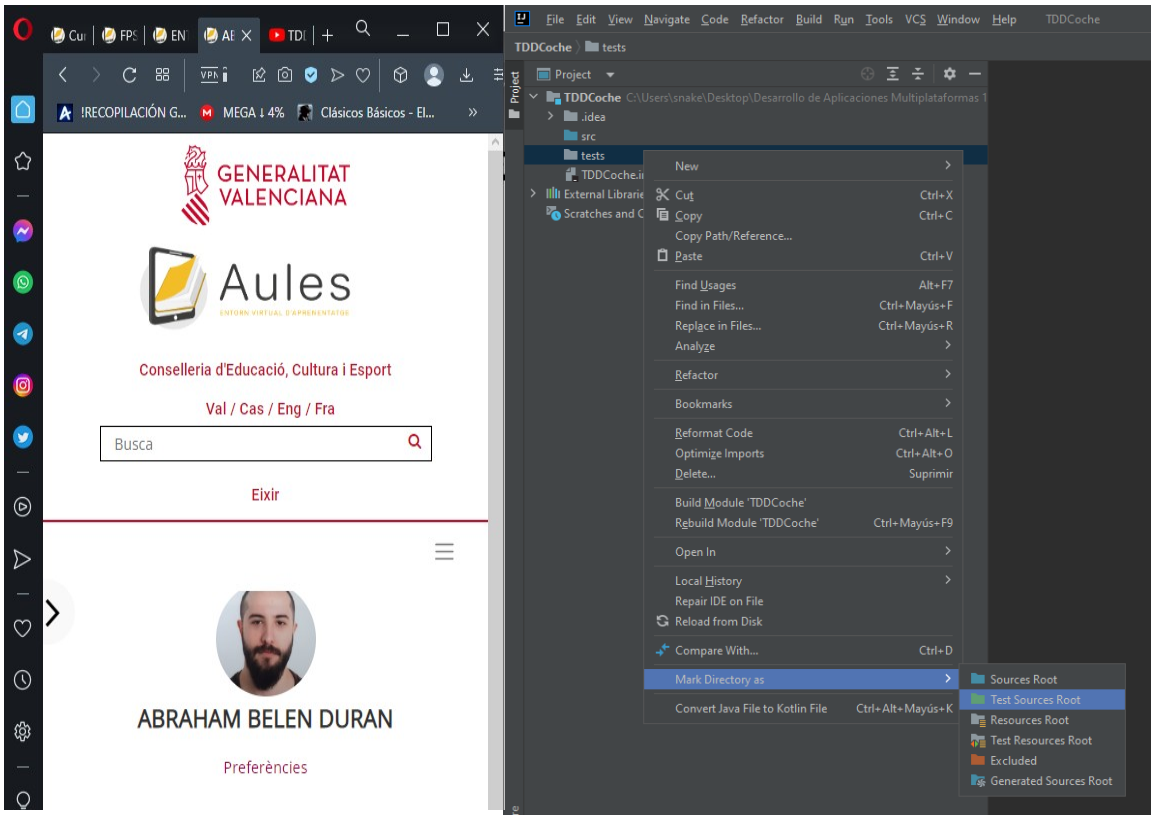


Bien, una vez hecho esto hay que seguir con el tutorial.

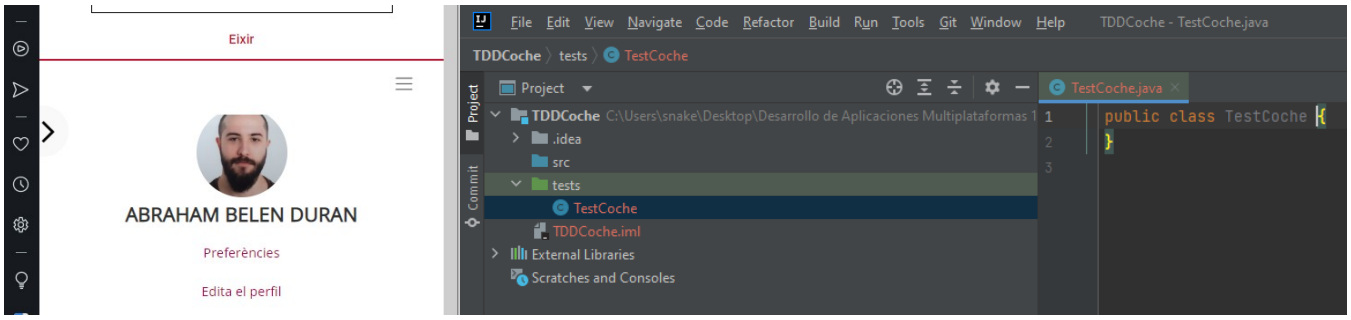
Para empezar hay que crear un nuevo directorio en la carpeta principal del proyecto a la que llamaremos ‘tests’. Para crear el directorio debemos hacer ‘click derecho’ sobre la carpeta principal del proyecto, darle a la opción ‘New’ y seguidamente ‘Directory’.



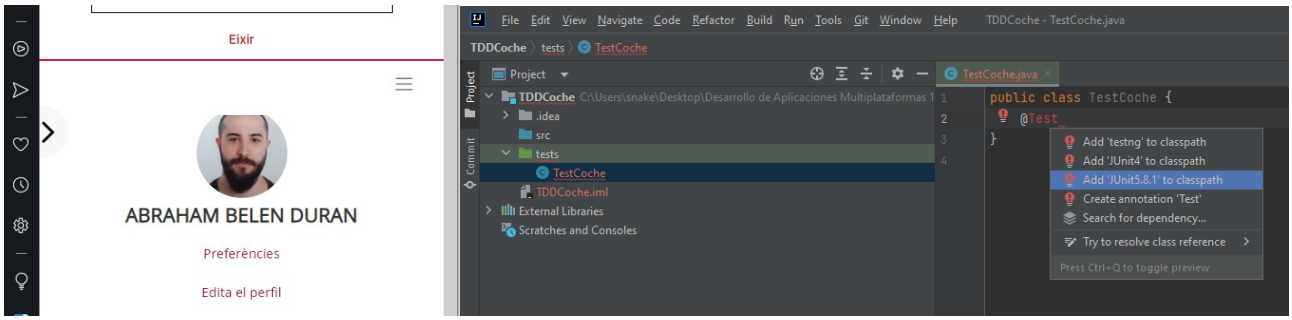
Una vez creada la carpeta ‘tests’, haremos ‘Click derecho’ en ella, desplazaremos el puntero hasta la opción ‘Mark Directory as’ y seleccionaremos la opción ‘Test Sources Root’



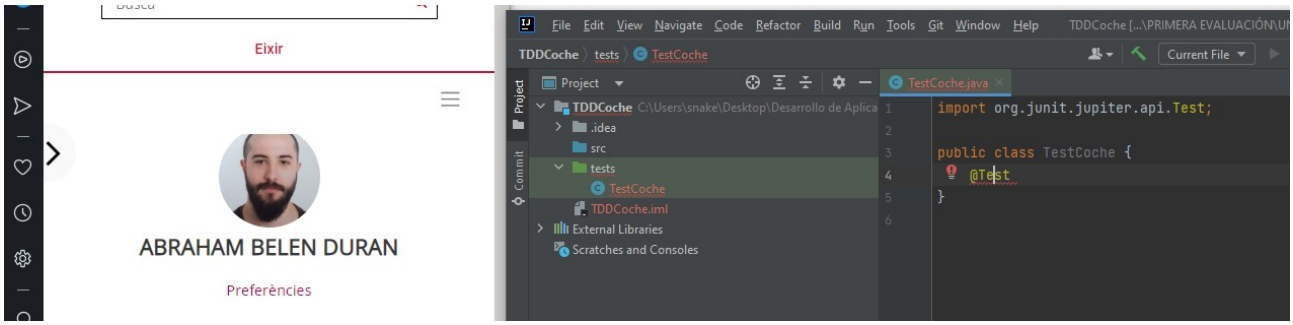
Y una vez creado como un directorio de ‘test’, haremos ‘Click derecho’ sobre dicho directorio y crearemos una nueva clase de Java. En la siguiente captura se aprecia la creación de la clase dentro del directorio ‘tests’.



Como lo que queremos es crear un test, debemos indicarlo. Para ello escribiremos @Test, en este momento nos dará un error, pero si pulsamos las teclas ALT+Intro se nos abrirá una pequeña pestaña para iniciar JUnit 5.8.1. En la siguiente imagen se puede apreciar.

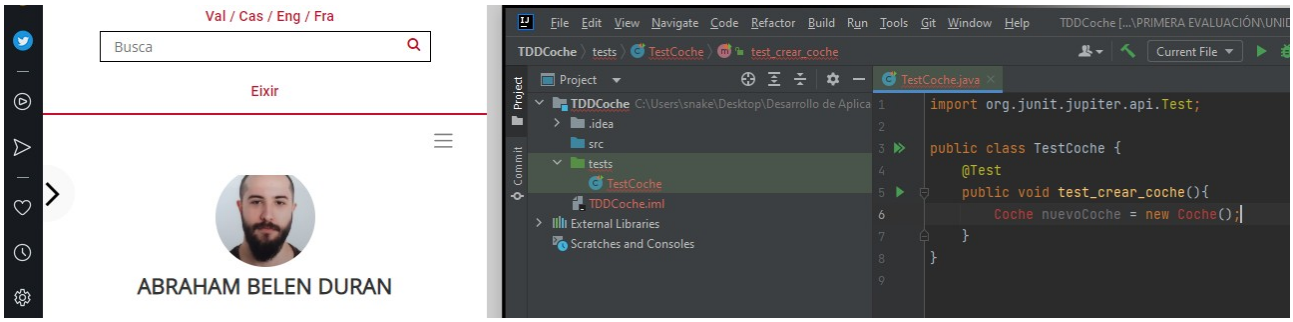


Así se ve una vez iniciado. Además de que se importa el paquete necesario para su funcionamiento.

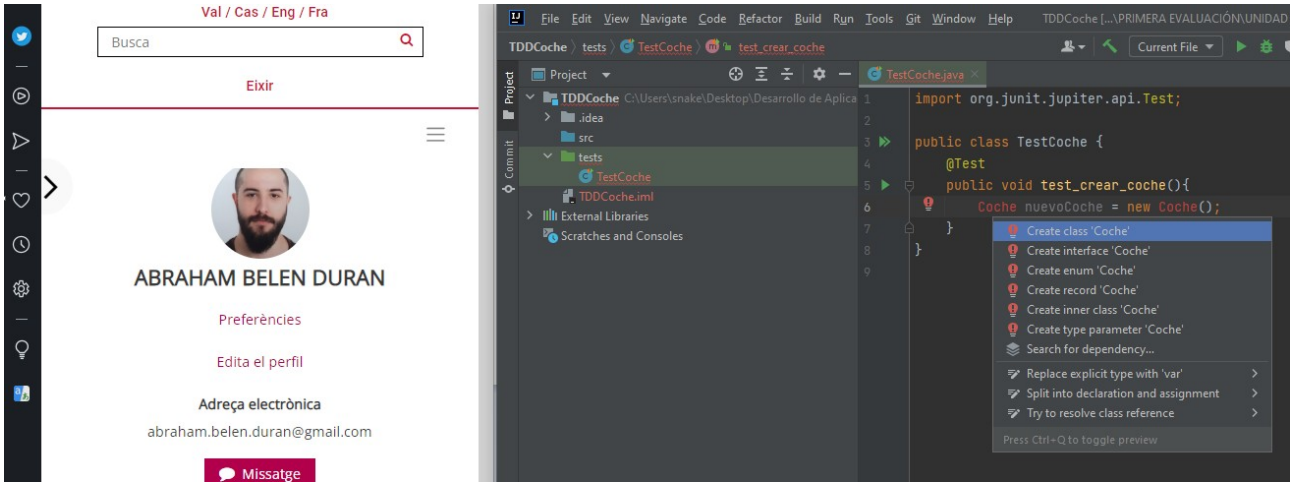


A continuación crearé mi primer Test.

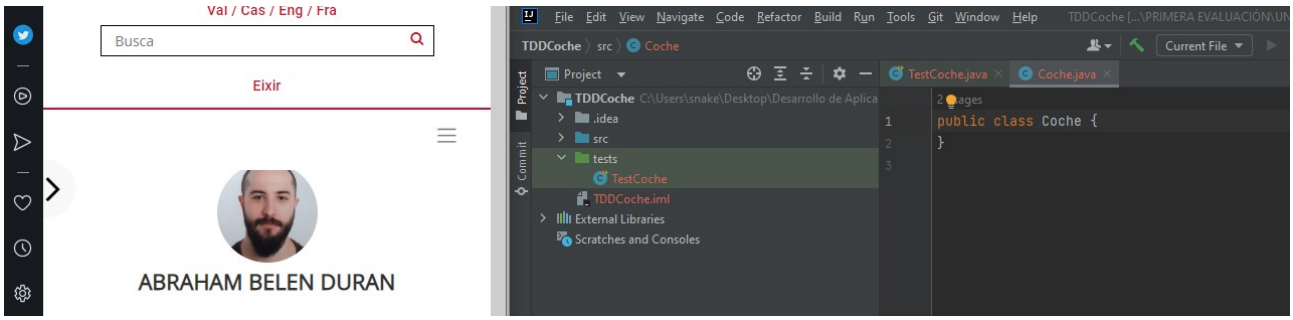
En la siguiente captura se aprecia la creación del primer test. Da error a la hora de encontrar la clase coche, pero es lo que buscamos.



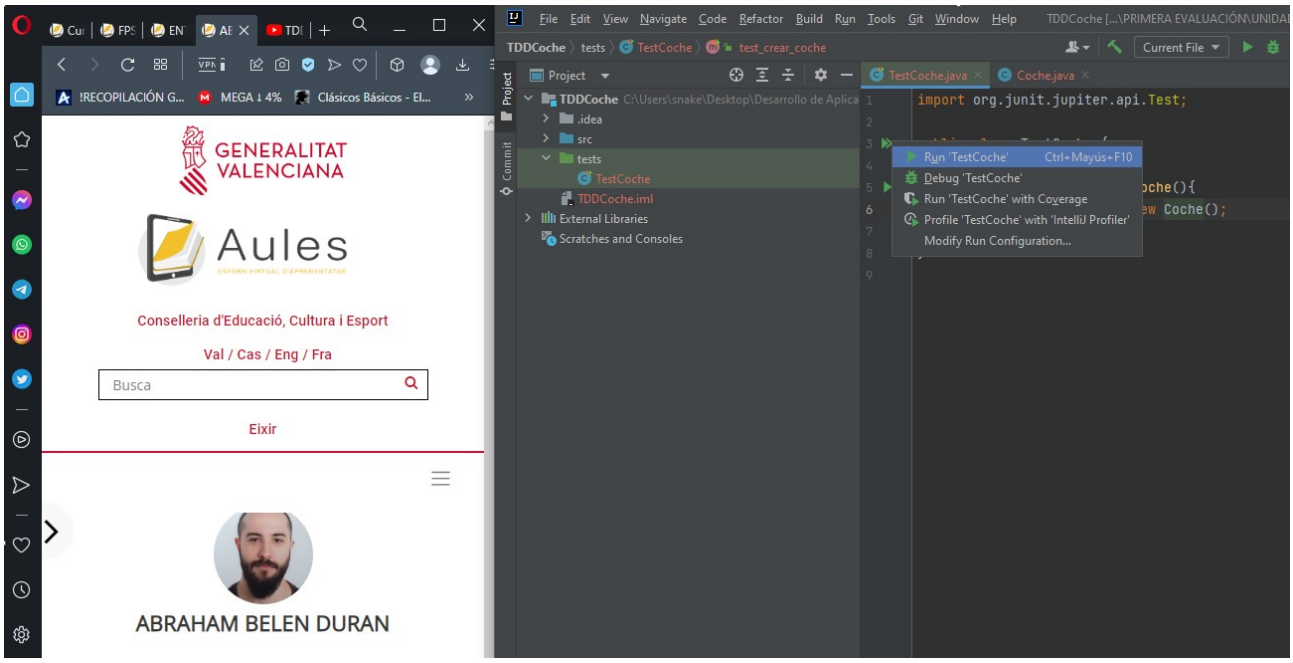
Si pasamos el puntero por encima de la clase que no existe, el propio entorno de desarrollo nos indica para crearla automáticamente. Vamos a hacerlo.



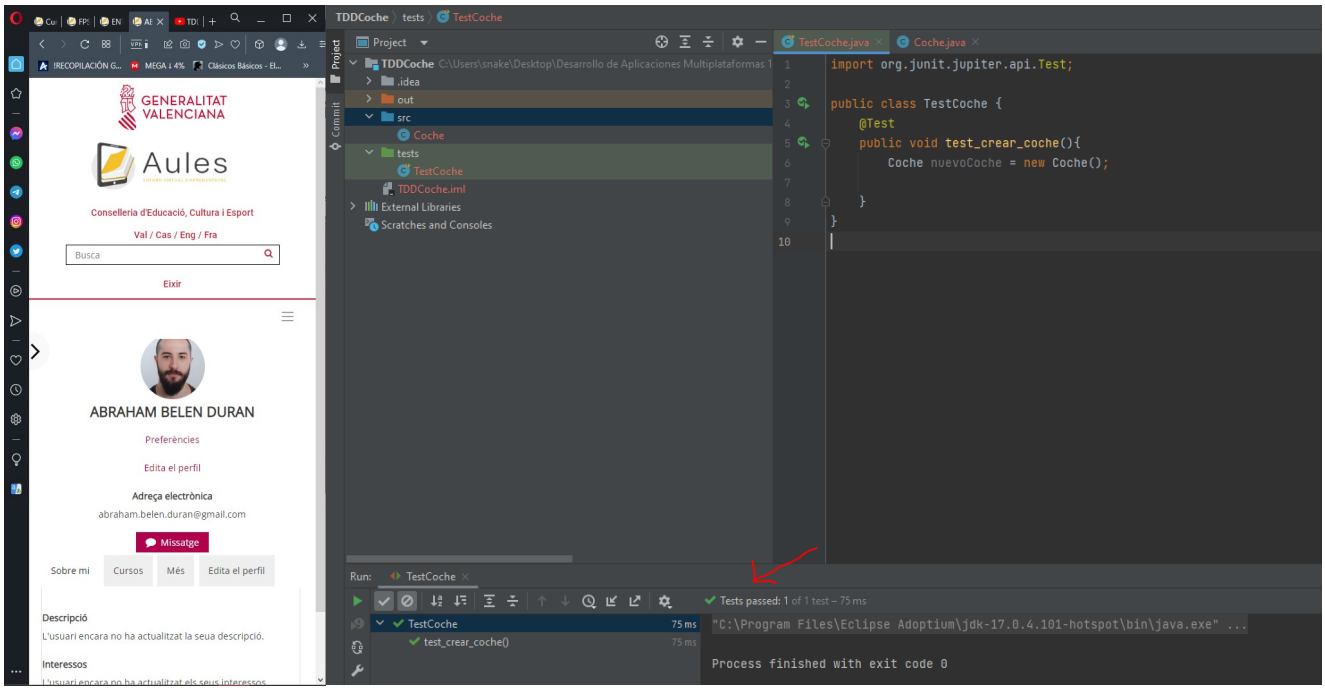
En la siguiente captura se aprecia la clase Coche creada.



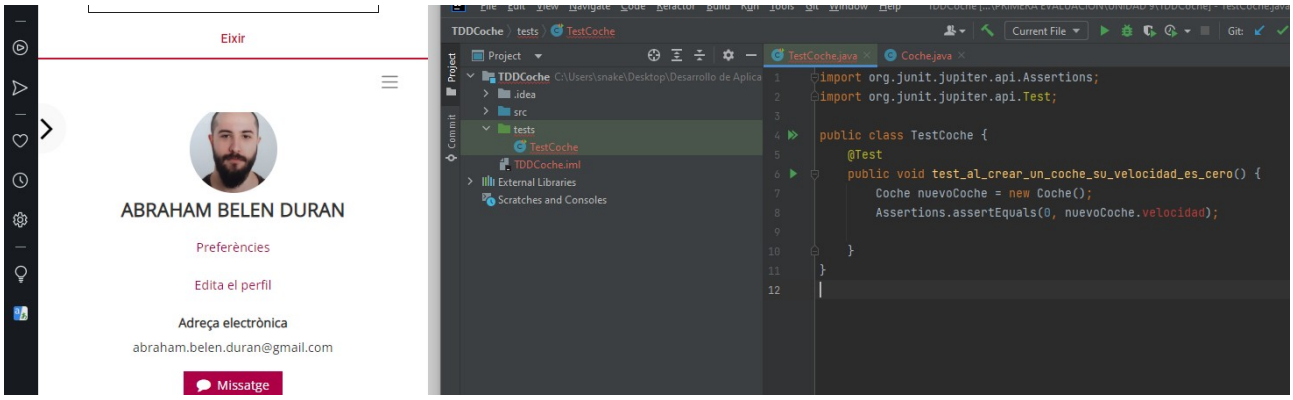
A continuación ejecutaremos nuestro primer test.



Una vez ejecutado veremos que hemos pasado nuestro primer test, tal y como se muestra en la imagen siguiente.

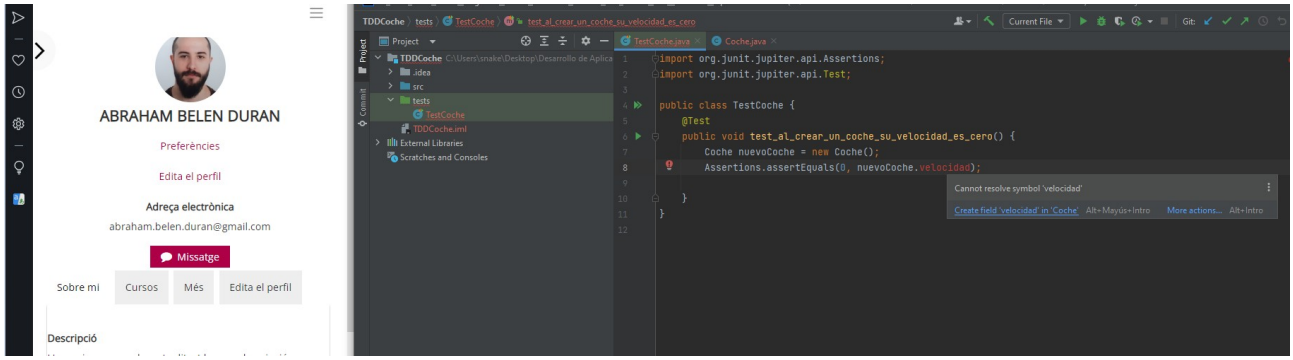


Bien, este test era bastante sencillo. Ahora lo complicaremos un poco. Crearemos un nuevo test para que cuando creemos un nuevo objeto de tipo coche, este se inicie con velocidad a cero.

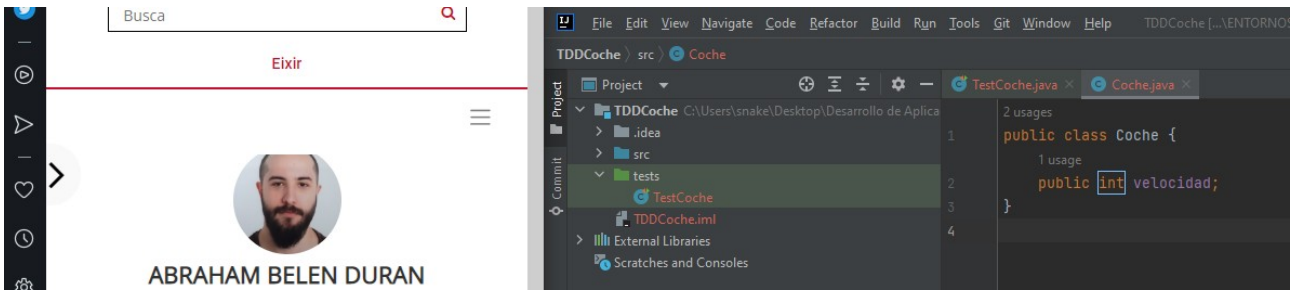


Esto lo que está haciendo es comparar que la velocidad de nuevoCoche sea igual al 0, si no es así, entonces no pasaremos el test.

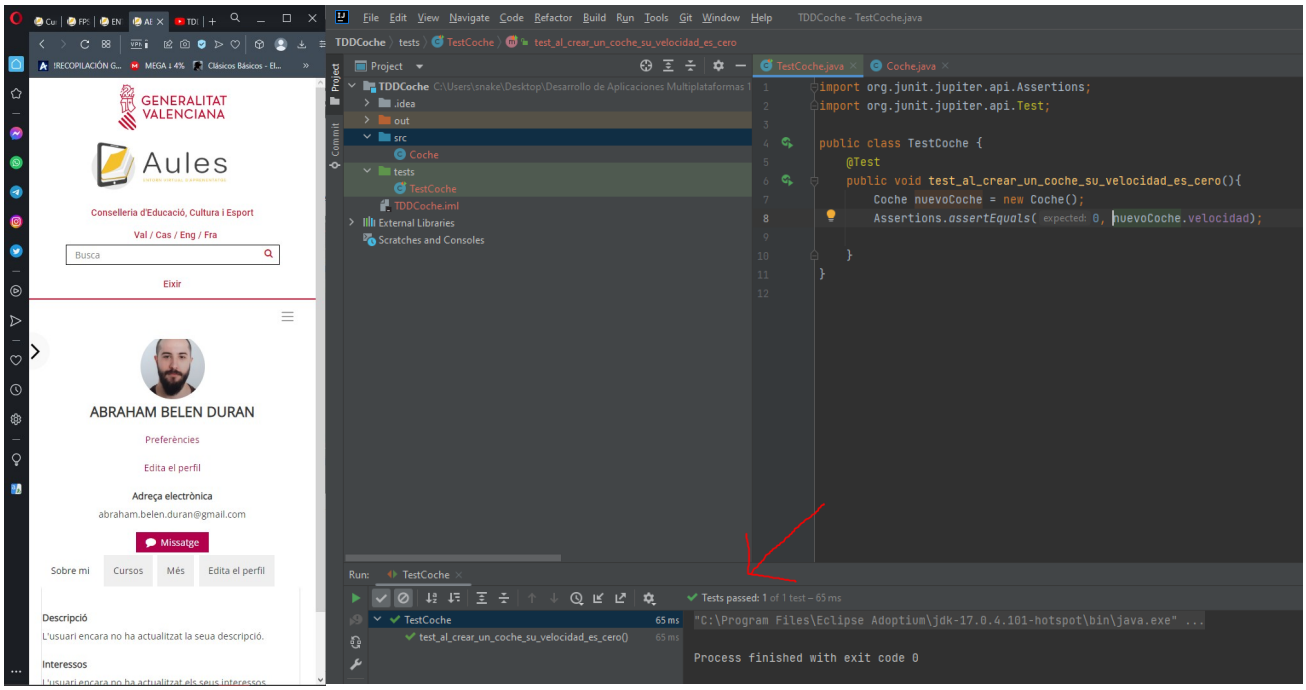
A continuación deberemos crear la variable velocidad para que este test pueda funcionar. Como hemos hecho antes, el propio entorno de desarrollo nos da la opción de crearlo automáticamente, así que lo haremos.



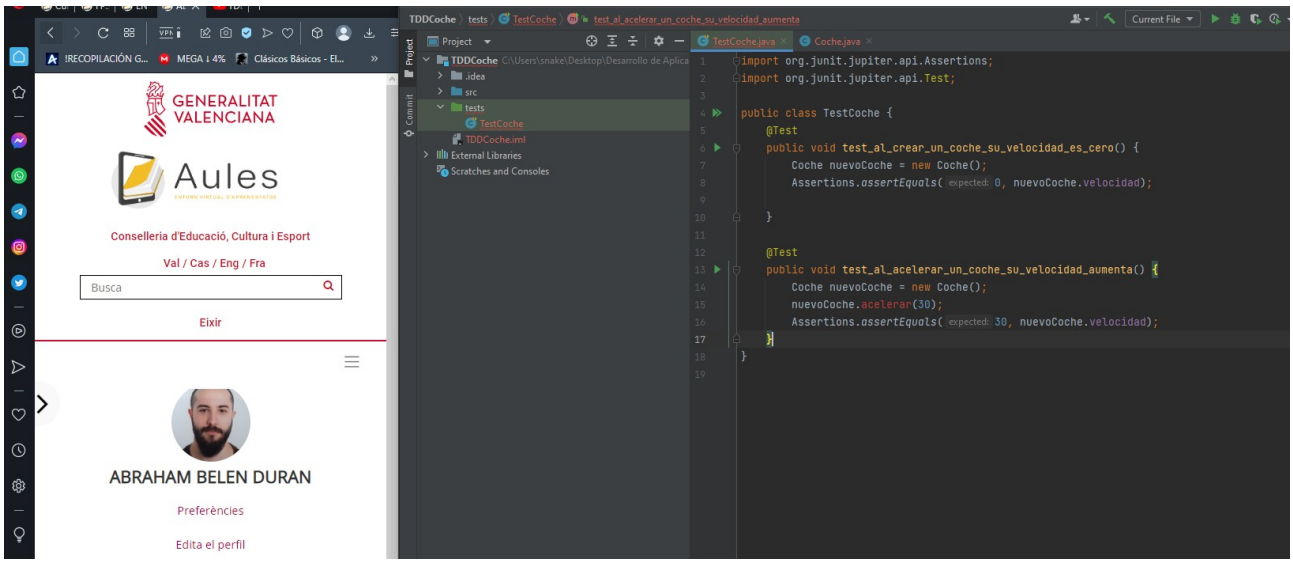
En este caso me lo ha creado de tipo int, aunque podríamos crearlo del tipo que deseemos.



Volvemos a ejecutar el test y podemos apreciar que lo hemos pasado satisfactoriamente.

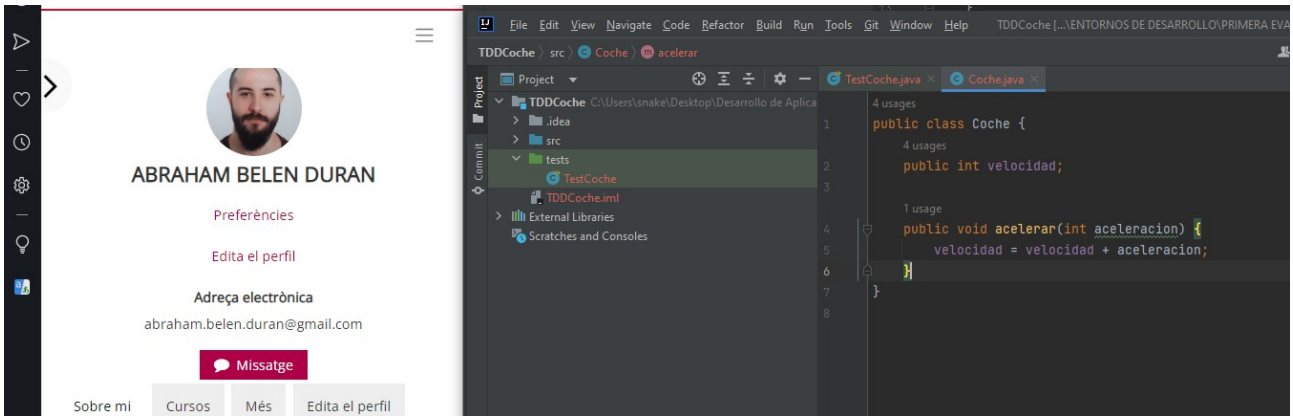


Una vez hecho esto, vamos ha realizar otro método distinto, esta vez crearemos el método acelerar para cambiar la velocidad del coche.

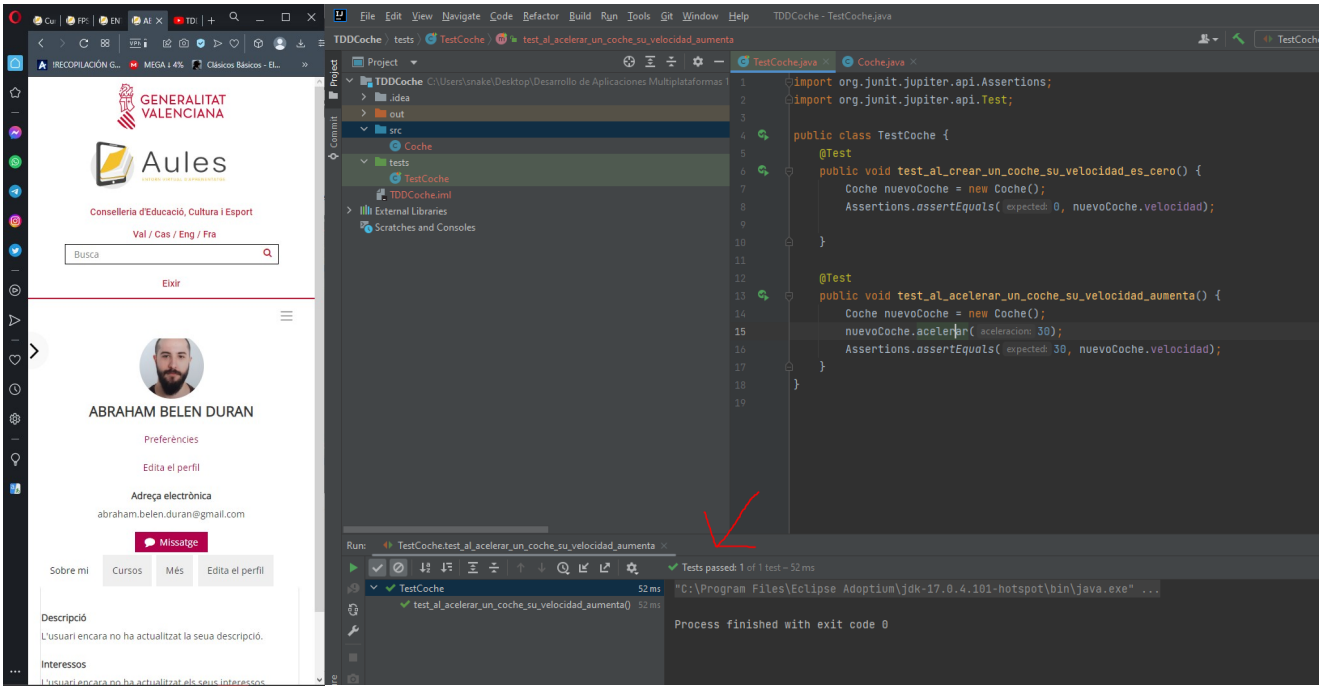


Le daremos una velocidad de aceleración de 30, por lo cual, la velocidad que deberá tener en Assertions debe de ser 30. Como se aprecia en la imagen, no tenemos el método 'acelerar' así que le indicaremos al entorno de desarrollo que nos cree dicho método de manera automática.

En la siguiente imagen se aprecia el método acelerar ya creado.

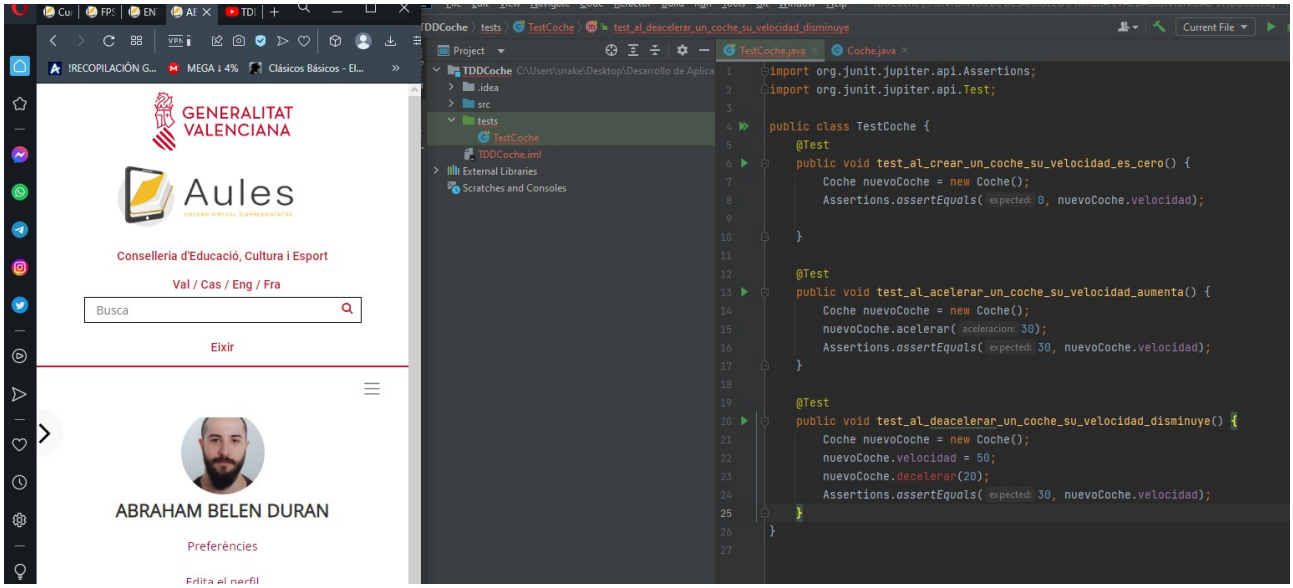


Ahora solo quedaría comprobar si el test funciona una vez que lo ejecutemos.



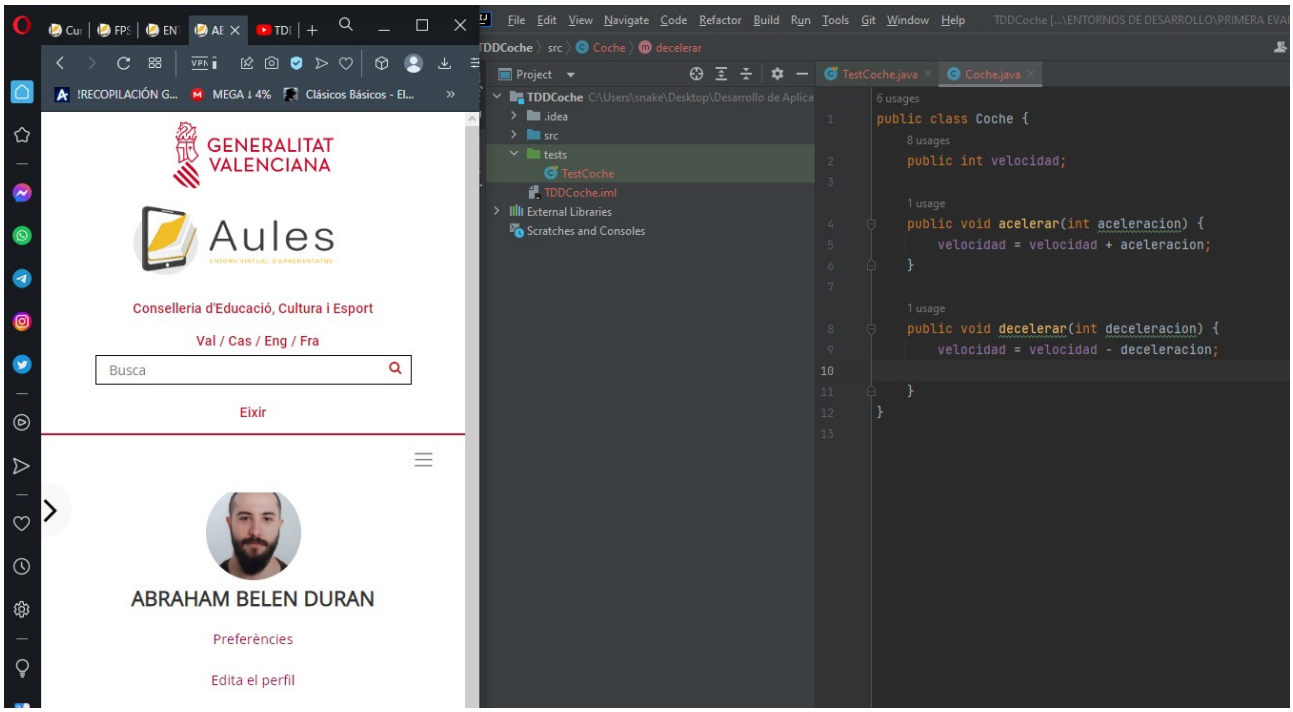
Como podemos apreciar, el test ha sido pasado satisfactoriamente.

Bien, a continuación vamos a crear el método decelerar, lo que hará que si el coche decelera, disminuirá su velocidad.



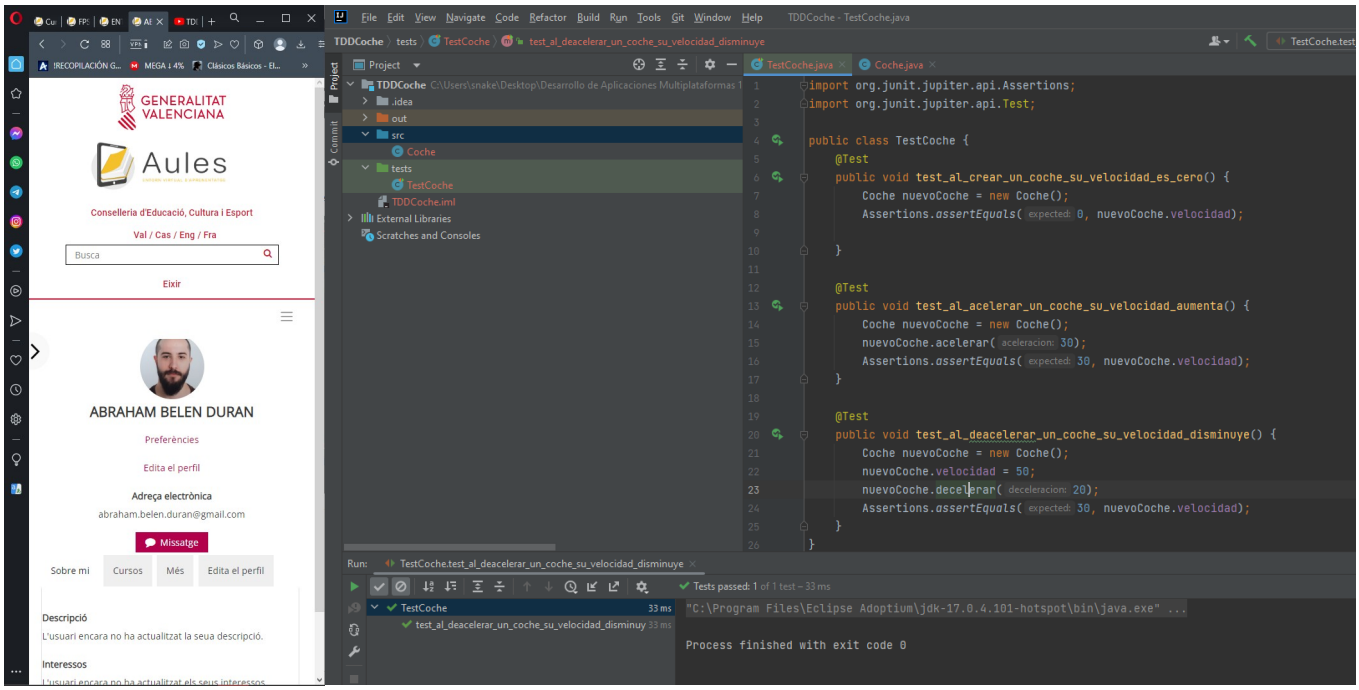
Aquí lo que he hecho ha sido crear un nuevoCoche con una velocidad de 50, porque cuando creamos un nuevoCoche su valor inicial de velocidad es 0 pero si así fuese, este nuevo método no funcionaría porque no se puede decelerar menos que 0.
Después he hecho que la deceleración sea de 20 para que el resultado final sea 30.
Ahora solo quedaría crear en la clase Coche el método decelerar.

Así quedaría el método decelerar junto con su argumento y el cálculo correspondiente.

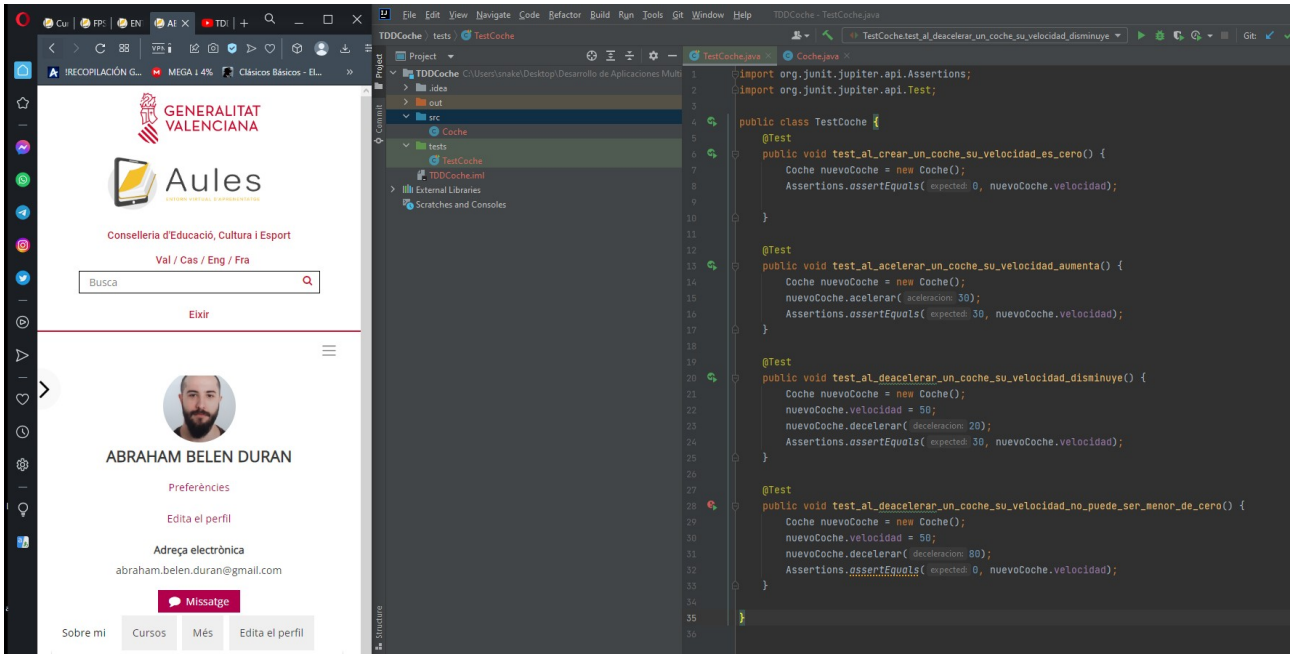


Ahora solo quedaría ver si funciona.

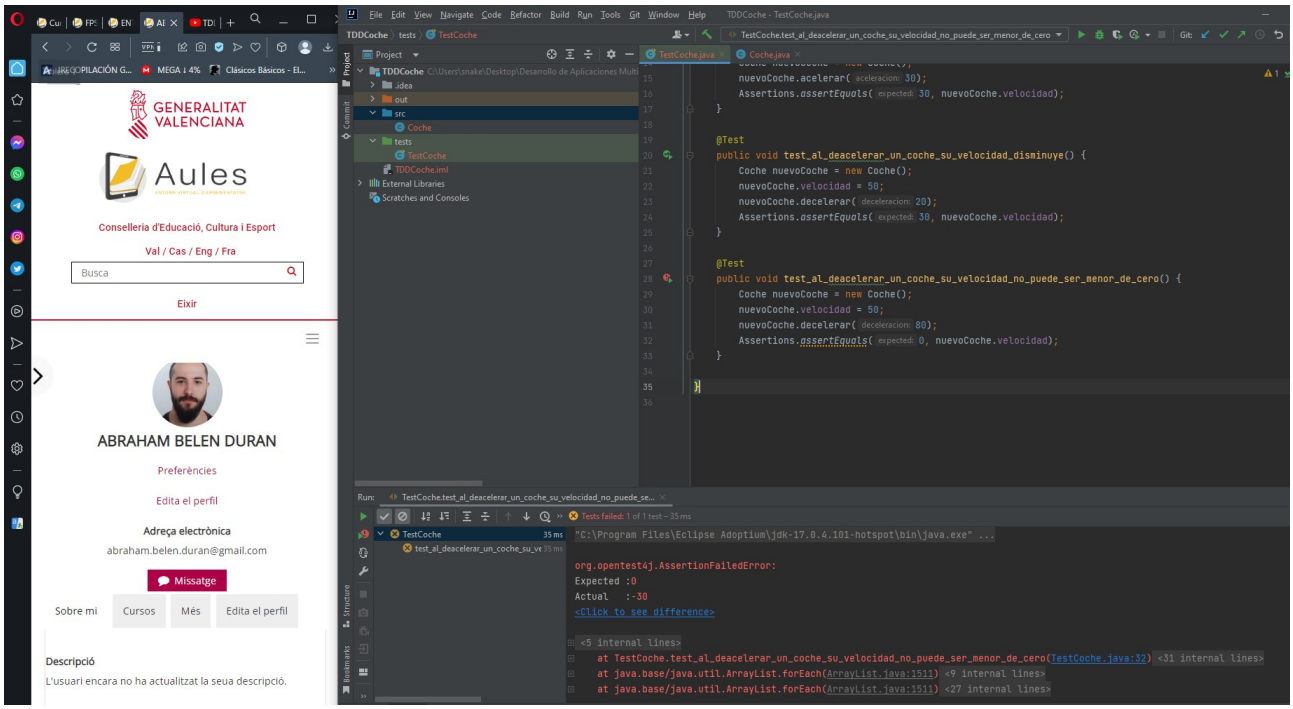
Y como se aprecia en la siguiente imagen, todo funciona a la perfección.



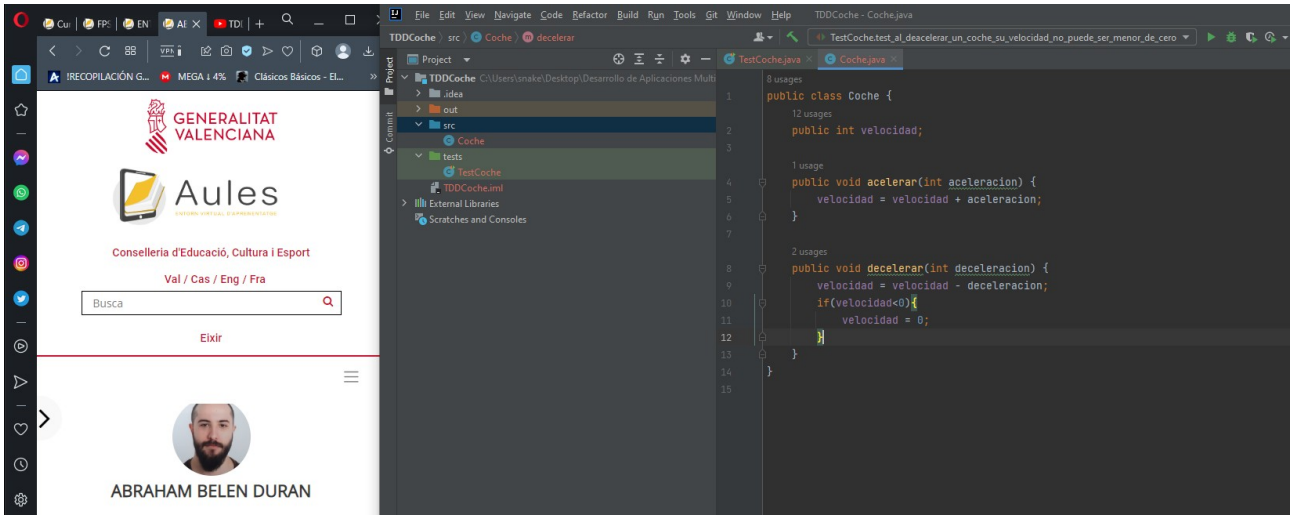
A continuación crearemos un último test que hará que si un coche decelera su velocidad no podrá ser inferior a cero.



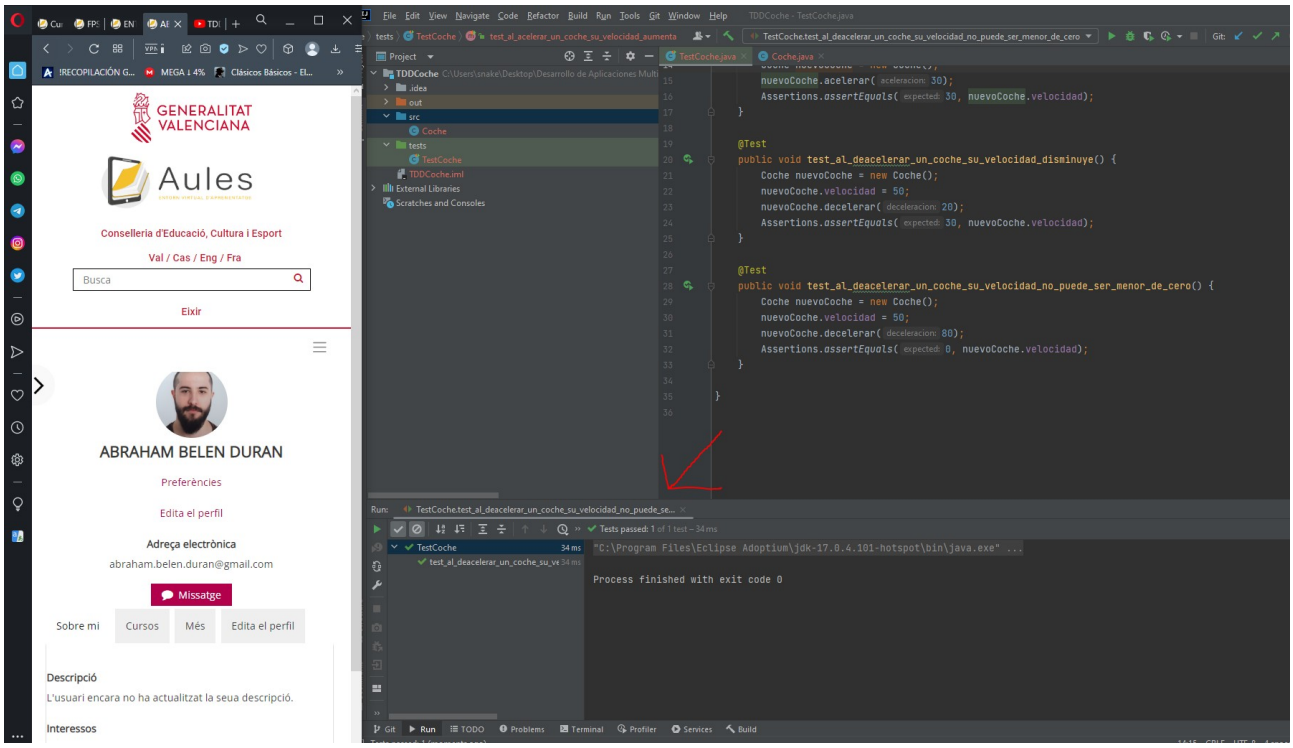
Si ejecutamos el test nos dará error. Consigue compilar pero da error. Como se puede ver en la captura siguiente, nos dice el error que da, en este caso se esperaba un valor de 0 y sin embargo el valor real ha sido -30. Que sería el resultado de la velocidad(50) menos decelerar(80).



Así que la solución a todo esto es hacer que la deceleración no pueda ser menor que 0, todo ello se debe modificar en la clase coche, tal y como muestro en la siguiente captura.



Esta sería la forma de solucionar el error que nos daba el test anterior, ahora solo queda probar a ver si funciona.



Y como se aprecia en esta captura, todo funciona a la perfección.

REFACTORIZACIÓN

En las siguientes capturas se aprecian los métodos refactorizados.