**Objetivo y alcance**

Como proyecto fin de master se desarrollará una aplicación web que servirá para que un criador de perros pueda gestionar sus perros adultos, camadas, sus clientes y sus ventas. Esta aplicación inicialmente estará enfocada a la cría de perros, pero el diseño se planteará de tal manera que si el criador decide ampliar su actividad empresarial a otros tipos de animales no sea necesario plantear ningún cambio importante en el diseño.

Esta fuera del alcance de esta aplicación la gestión de las facturas, contabilidad y pago de impuestos derivados de la cría de animales. Solamente existirá el concepto de venta que permitirá al criador identificar sus ingresos brutos.

El desarrollo de esta aplicación está orientado a la parte back, por lo cual no se prestará especial atención al diseño de las páginas web, dejando este punto pendiente para versiones posteriores.

La aplicación estará securizada para que no sean accesibles las urls si no hay un usuario logeado correctamente. Además, está fuera del alcance de esta primera versión la creación de funcionalidades que permiten administrar a los usuarios mediante perfiles.

**Modelo de Datos y Base de Datos**

Como base de datos se utilizará MySql que correrá bajo MAMP, el nombre de la base de datos será *“feederddbb”* y estará disponible en la siguiente url “*//localhost:3306/feederddbb*”.

Se ha decidido utilizar el idioma ingés para dar nombre a las tablas/entidades de la aplicación.

La primera entidad del modelo de datos será “*User”* que tendrá los siguientes campos: username, password, email, name, surnames y role. Esta entidad no tendrá relación con ninguna otra.

La siguiente entidad del modelo de datos será “*Animal”* que tendrá los siguientes campos: idAnimal, name, born\_date, sex, weight, status,color, father, mother, death\_date, idBreed, IdClient e idSale.

La siguiente entidad del modelo de datos será *“Breed”* que reflejará las razas de los animales. Los campos de esta entidad serán: idBreed, name, junior\_price, senior\_price y specy\_id.

La siguiente entidad será *“Specy”* que reflejará las especies, es decir, una misma especie podrá tener diferentes razas. Los campos que tendrá esta entidad serán: idSpecy, name y type.

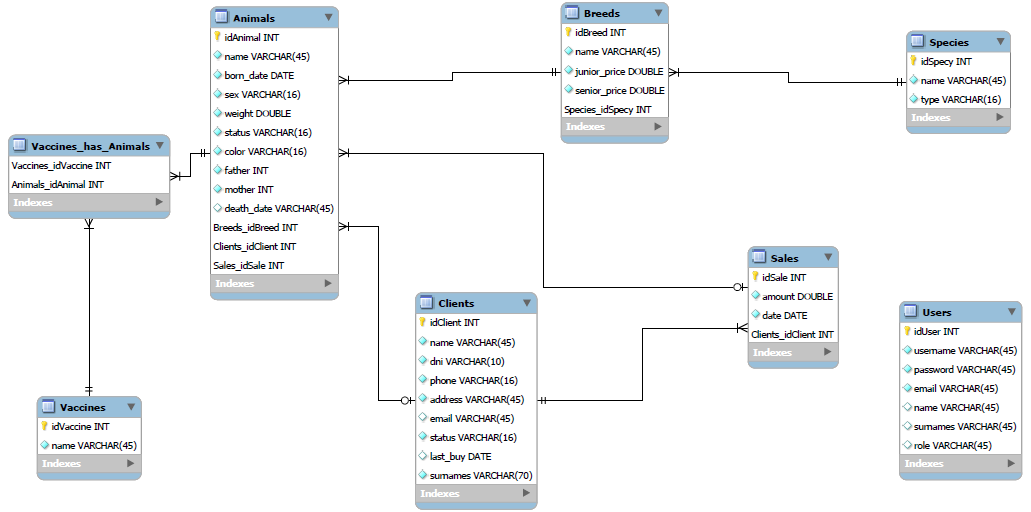
La siguiente entidad será *“Client”* y tendrá los siguientes campos: idClient, name, surnames, dni, pone, address, email, status y last\_buy.

La siguiente entidad será *“Sale”* y tendrá los siguientes campos: idSale, amount, date e idClient.

En el modelo de datos existe otra entidad con el nombre de “Vaccine” que pretende reflejar las vacunas que se aplican a cada animal pero que no se utilizará en esta primera versión de la aplicación.

En el fichero “*Scripts\_ddbb*”, se encuentran todos los scripts de creación de las tablas junto con algunos datos de prueba. La tabla *“Users”* inicialmente estará vacía y será necesario crear a través de la GUI un usuario con el que poder acceder a la aplicación.

A continuación aparece el esquema de las relaciones entre las tablas que definen el modelo de datos de la aplicación:



Al utilizar MySql se ha detectado un problema a la hora de la nomenclatura de los campos que son claves de otras tablas. El problema consiste en que para cada una de las tablas solo puede haber un campo con el literal “id” (el problema está en la dependencia de MySql de Spring Boot), por lo cual se utiliza “id” como identificador de la entidad y para el resto de campos que tenían “id” en el nombre, es necesario borrarlo. Por ejemplo, en la tabla “Animals” el campo “id” hace referencia a “idAnimal” y los campos “id\_Client”, “id\_Sale” e “id\_Breed” se han tenido que renombrar a “client”, “sale” y “breed”.

**Arquitectura**

Se ha decidido utilizar para el desarrollo de la aplicación una arquitectura de microservicios bajo el ecosistema de Spring Boot en Java 8. Todos los microservicios compatirán una única base de datos. El ecosistema de Spring Boot contará con los siguientes microservicios:

1. Config Server – Este microservicio correrá en el puerto 8888 en localhost y la uri donde estarán todos los ficheros de properties del resto de microservicios será <https://github.com/ddfdesign2018/SpringConfig>. El nombre del proyecto para este microservicio será ms-ConfigServer y el nombre del microservicio será “*config-server*”.
2. Eureka – Este microservicio correrá en el puerto 8761 en localhost. El nombre del proyecto para este microservicio será ms-EurekaServer y el nombre del microservicio será “*eureka-server*”.
3. Zuul – Este microservicio correrá en el puerto 8762 en localhost. El nombre del proyecto para este microservicio será ms-Zuul y el nombre del microservicio será “*zuul-server*”.
4. Users – Este microservicio correrá en el puerto 8760 en localhost. El nombre del proyecto para este microservicio será ms-users y el nombre del microservicio será “*users-ms*”.
5. Animals – Este microservicio correrá en el puerto 8763 en localhost. El nombre del proyecto para este microservicio será ms-animals y el nombre del microservicio será “*animals-ms*”.
6. Species – Este microservicio correrá en el puerto 8764 en localhost. El nombre del proyecto para este microservicio será ms-species y el nombre del microservicio será “*species-ms*”.
7. Breeds – Este microservicio correrá en el puerto 8765 en localhost. El nombre del proyecto para este microservicio será ms-breeds y el nombre del microservicio será “*breeds-ms*”.
8. Clients – Este microservicio correrá en el puerto 8766 en localhost. El nombre del proyecto para este microservicio será ms-clients y el nombre del microservicio será “*clients-ms*”.
9. Sales – Este microservicio correrá en el puerto 8767 en localhost. El nombre del proyecto para este microservicio será ms-sales y el nombre del microservicio será “*sales-ms*”.

La secuencia de arranque de los microservicios debe ser la siguiente:

1) Arrancar Config Server.

2) Arrancar Eureka.

3) Arrancar Zuul que debe registrarse en Eureka y cargar configuración desde Config Server.

4) Arrancar Users que debe registrarse en Eureka y cargar configuración desde Config Server.

5) Arrancar Animals que debe registrarse en Eureka y cargar configuración desde Config Server.

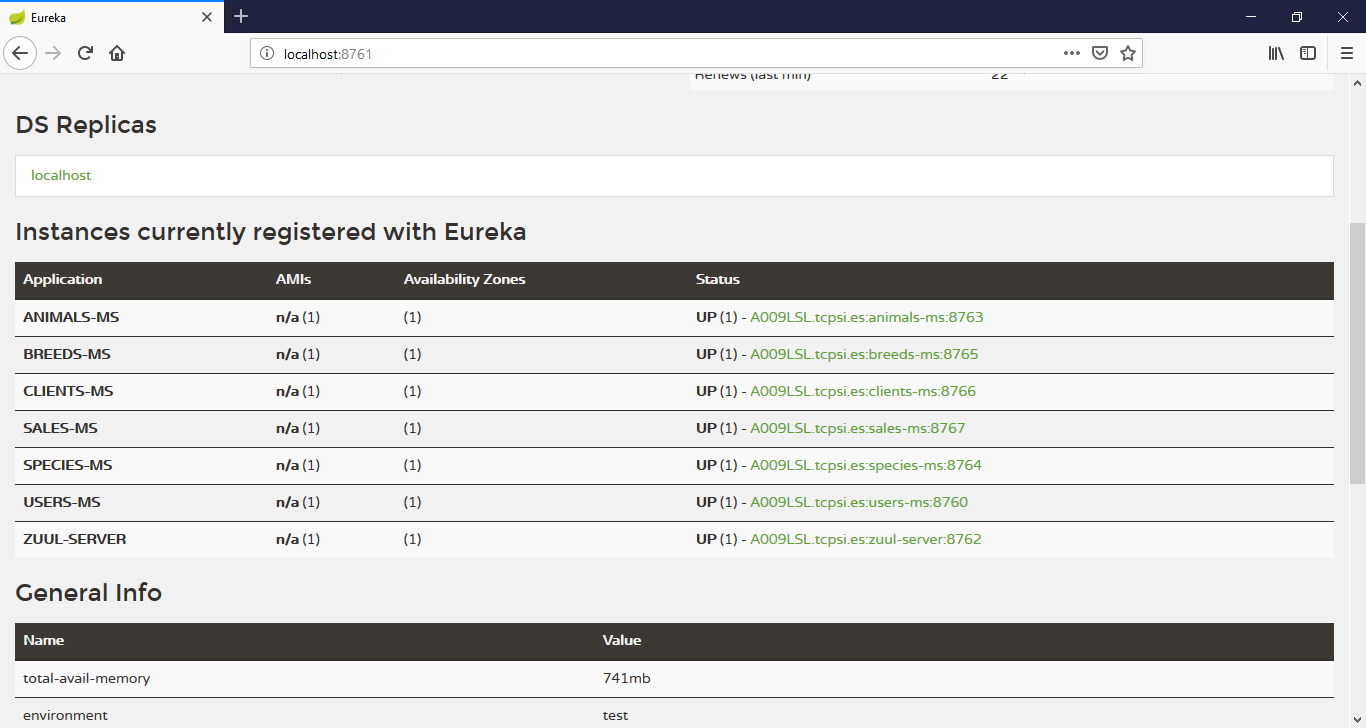
6) Arrancar Breeds que debe registrarse en Eureka y cargar configuración desde Config Server.

7) Arrancar Species que debe registrarse en Eureka y cargar configuración desde Config Server.

8) Arrancar Clients que debe registrarse en Eureka y cargar configuración desde Config Server.

1. Arrancar Sales que debe registrarse en Eureka y cargar configuración desde Config Server.

Al consultar la consola de Eureka deben aparecer registrados todos los microservicios excepto el propio Eureka y Config Server.



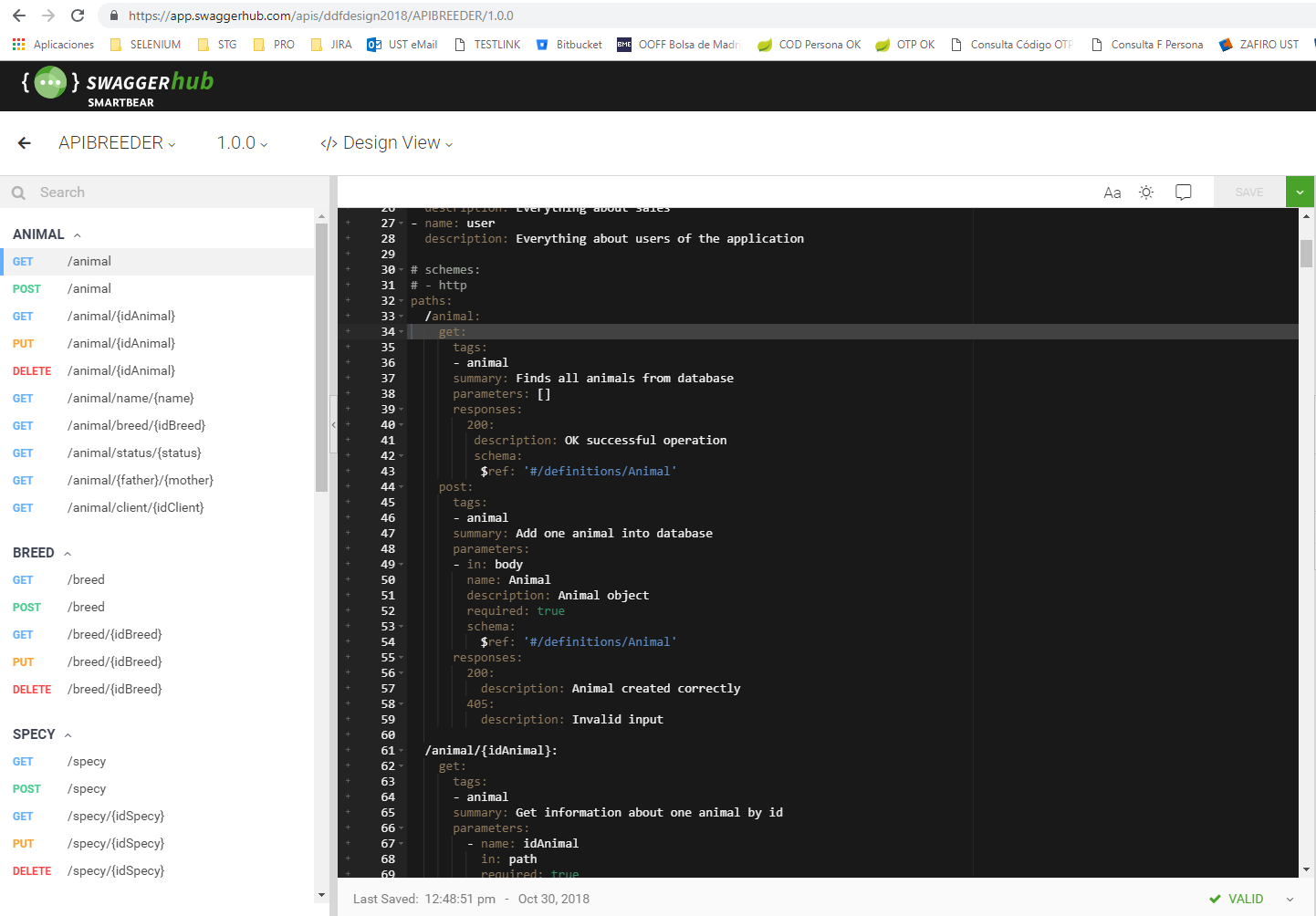
Se configura Zuul para que todos los microservicios sean accesibles desde un único punto de acceso. En la sección de funcionamiento se mostrará un ejemplo de como se puede acceder a los microservicios directamente desde su puerto o desde Zuul.

**API Rest definitions**

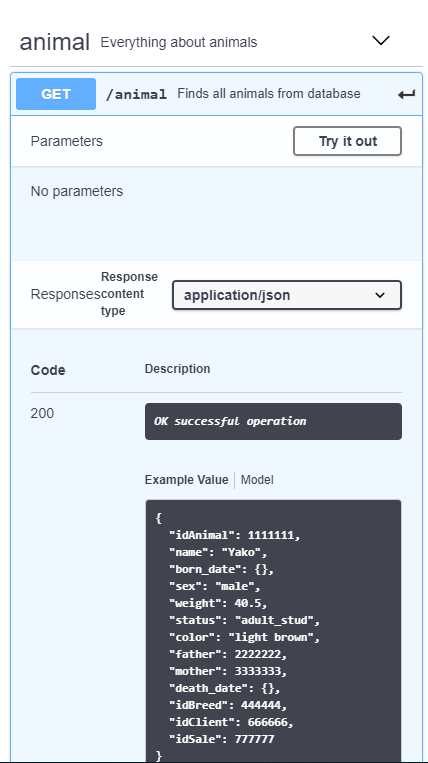
Se ha utilizado swaggerhub para la definición de todas las API rest de los microservicios que componen el ecosistema Spring Boot.

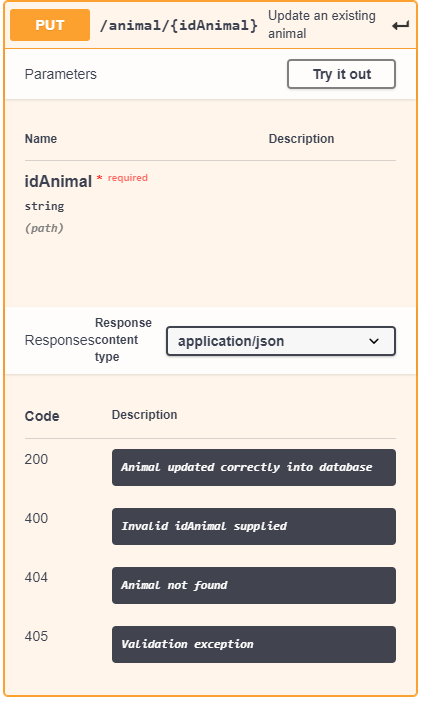
Si se accede a la url que figura a continuación se puede revisar el contrato de cada una de las APIs: <https://app.swaggerhub.com/apis/ddfdesign2018/APIBREEDER/1.0.0>

A modo de ejemplo se adjuntan pantallazos del API de Animals:

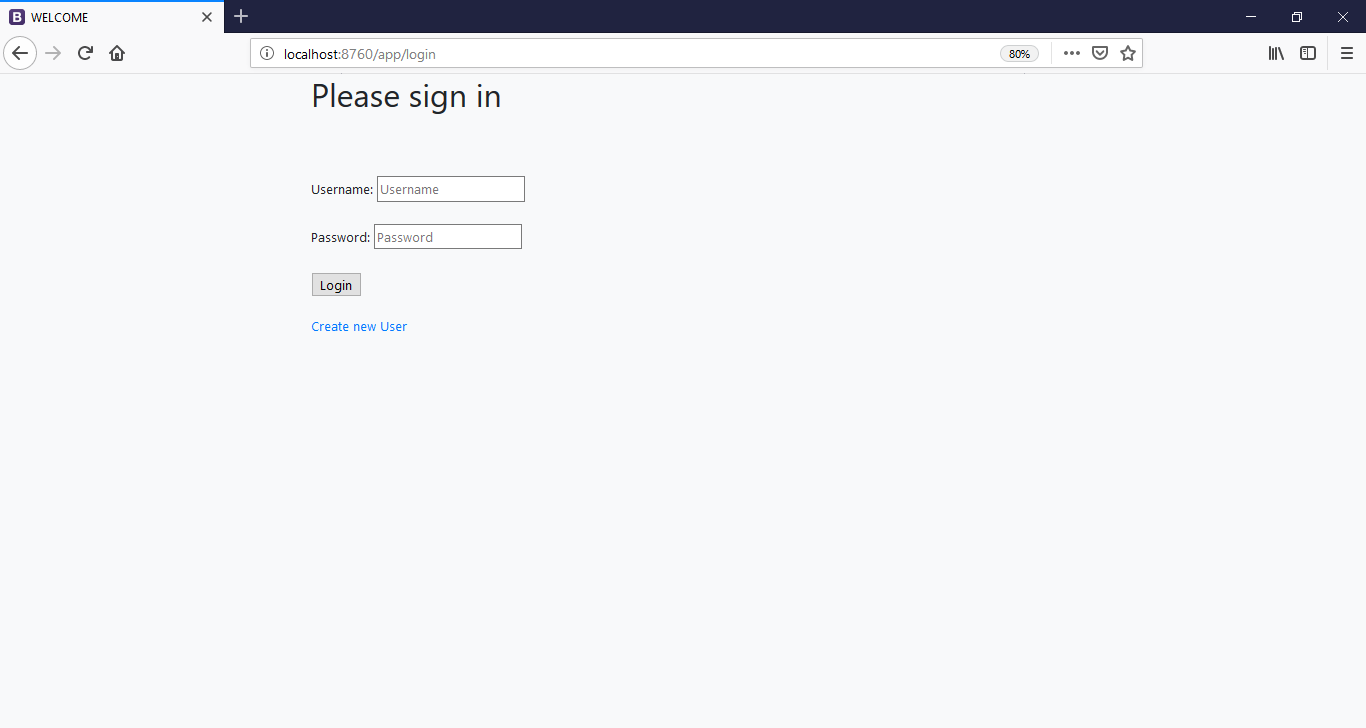


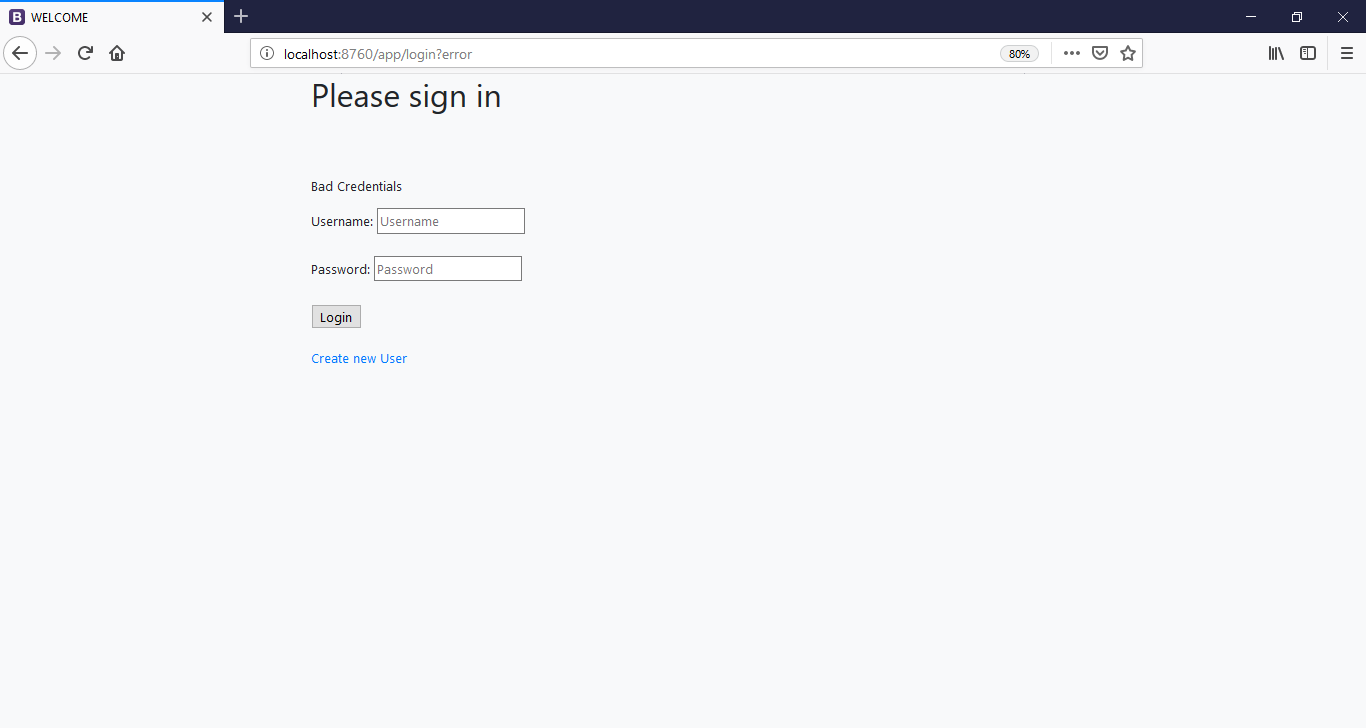


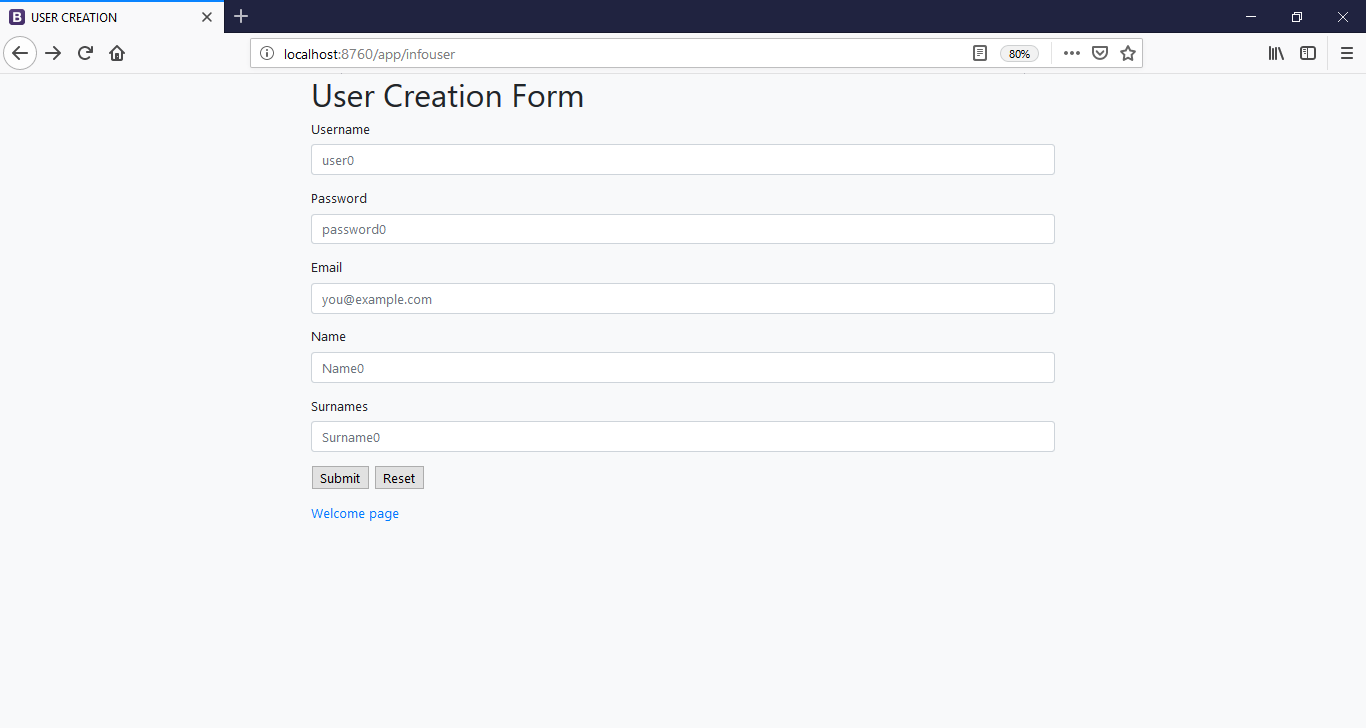


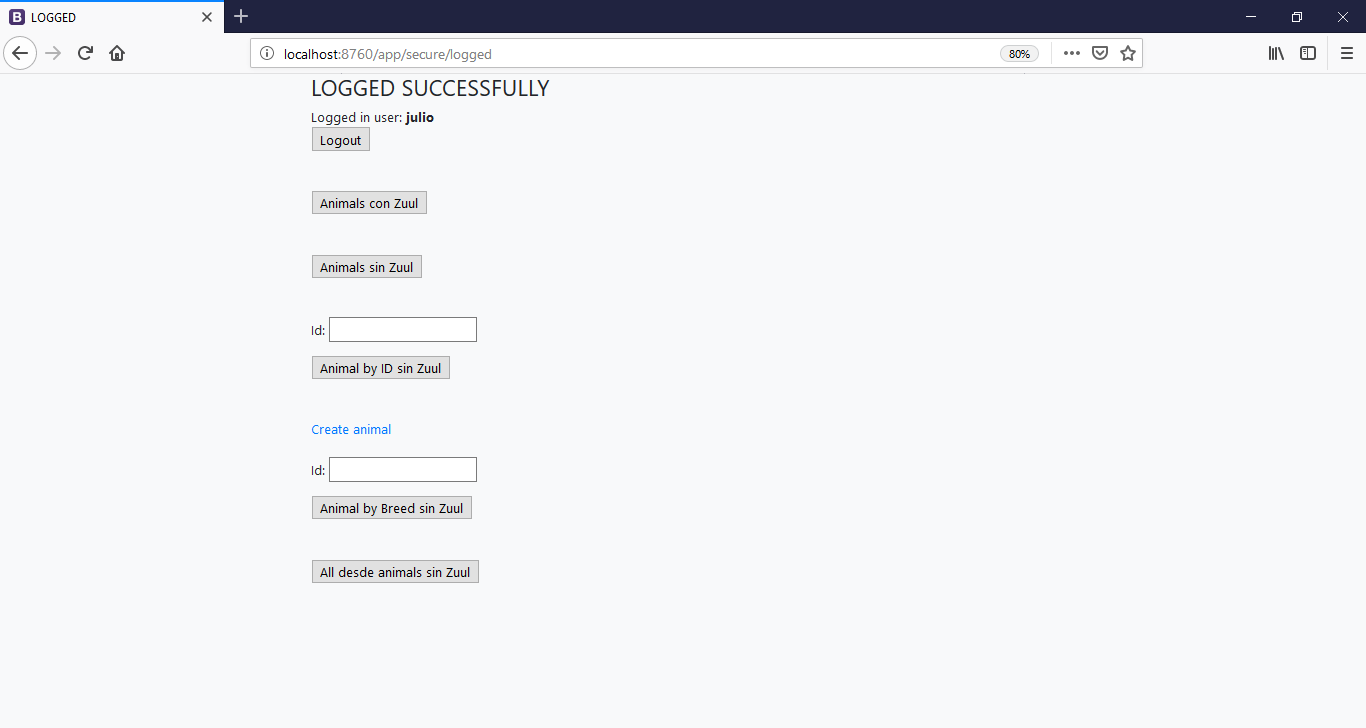


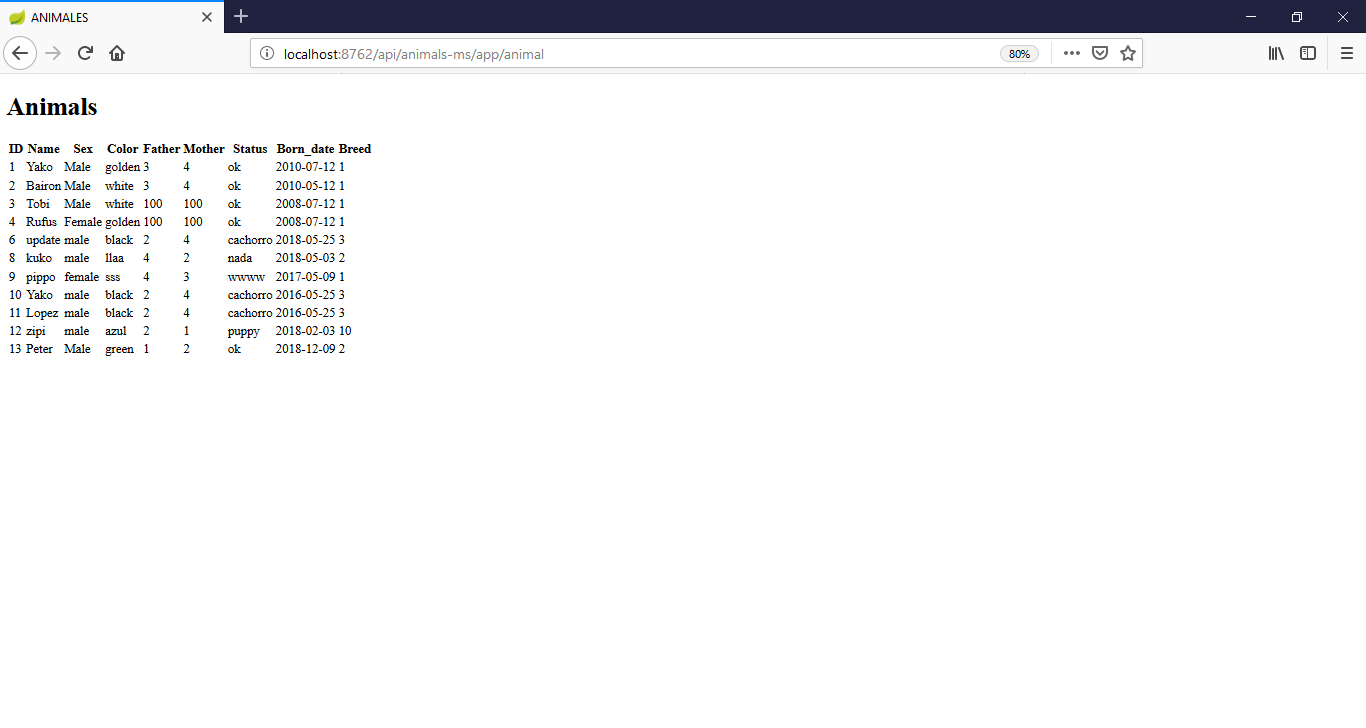
**Funcionamiento**

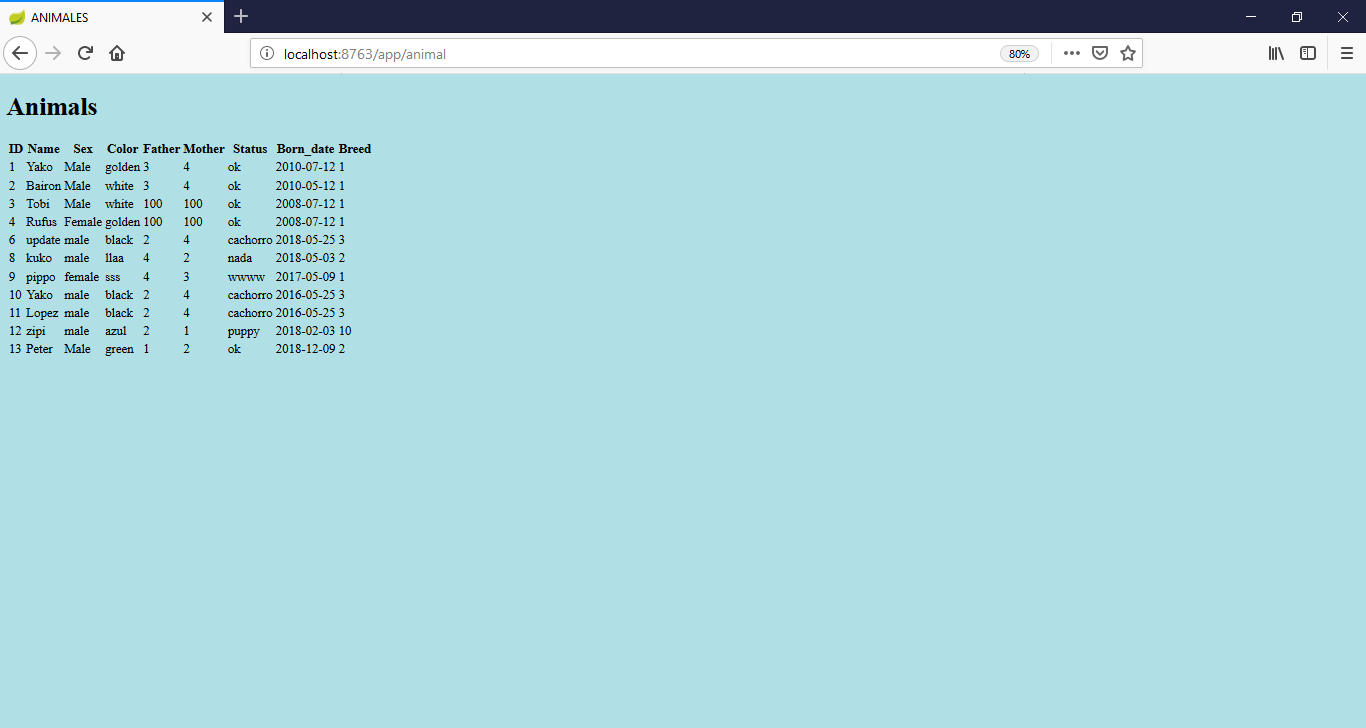


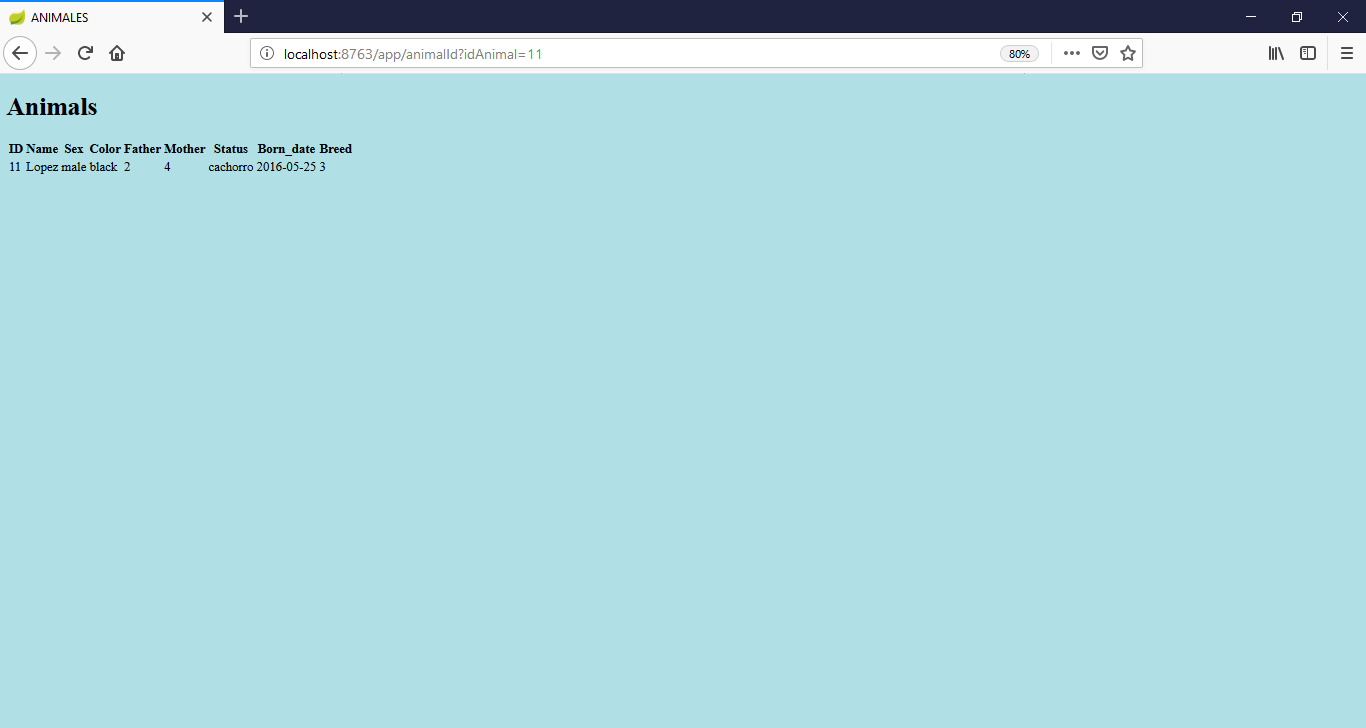


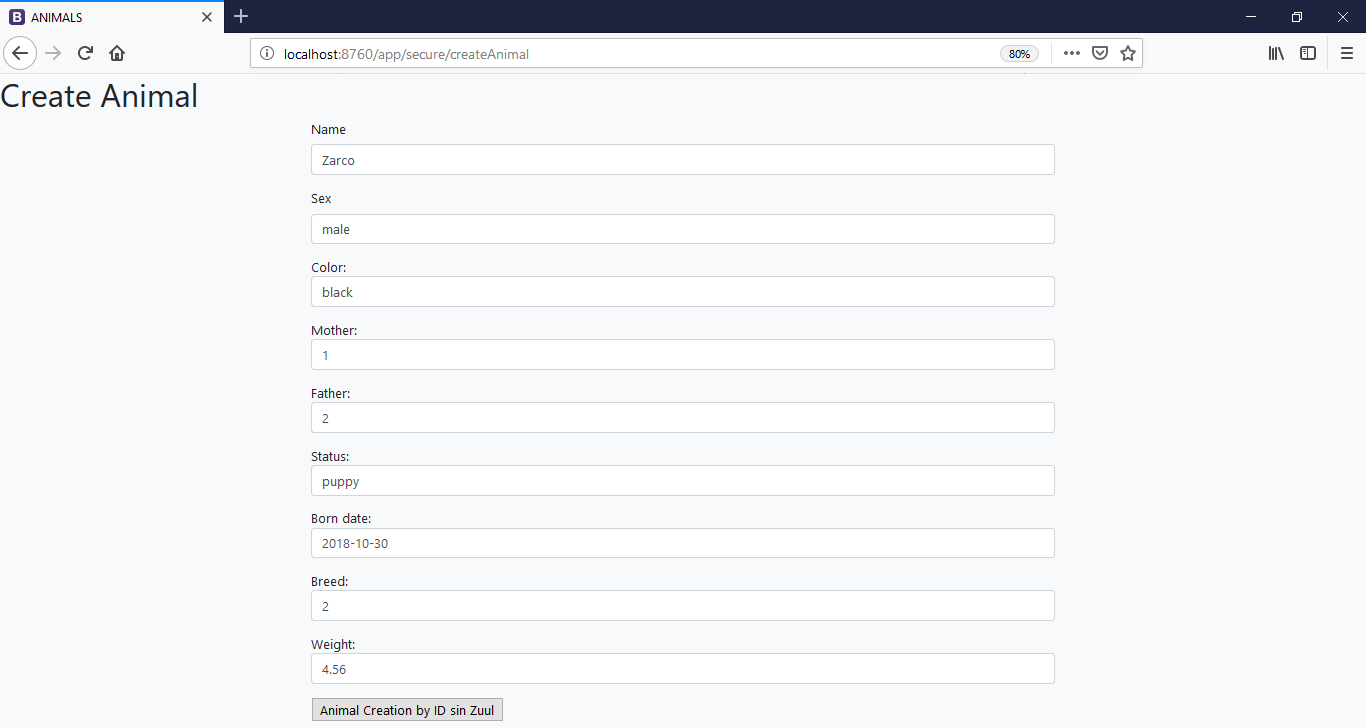


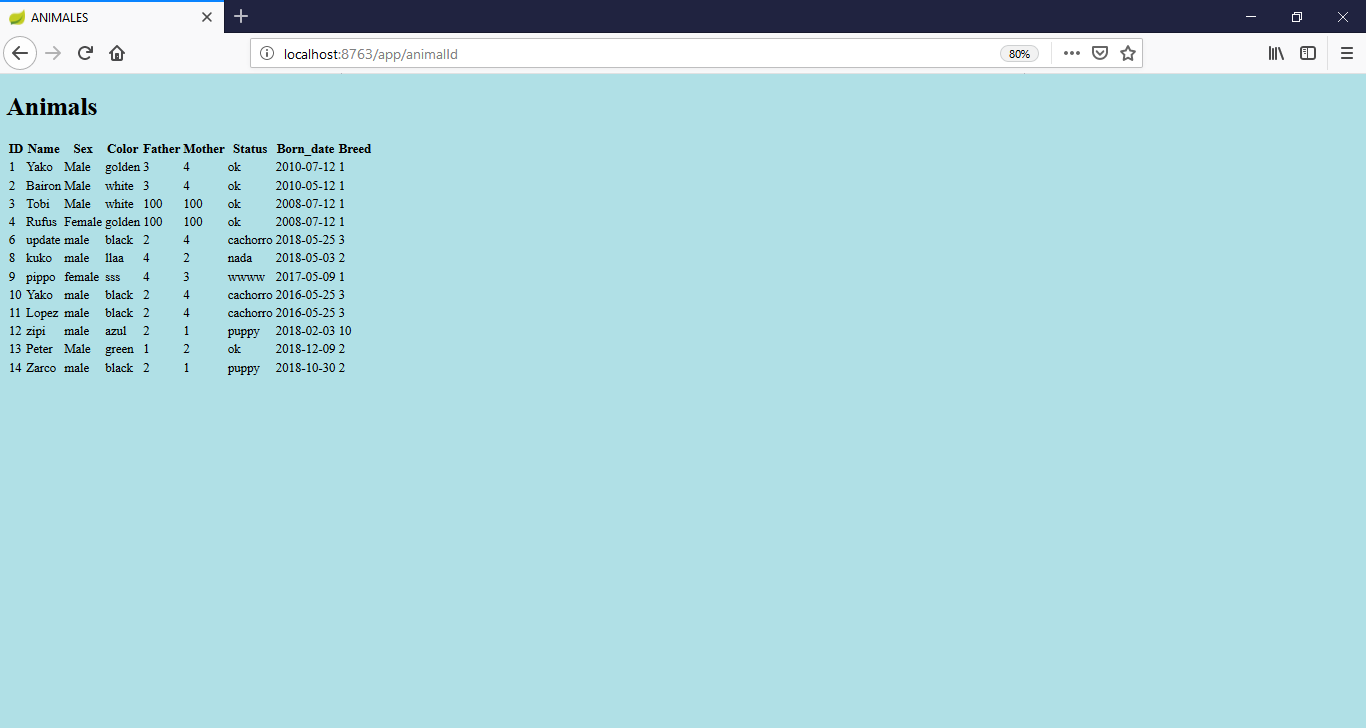


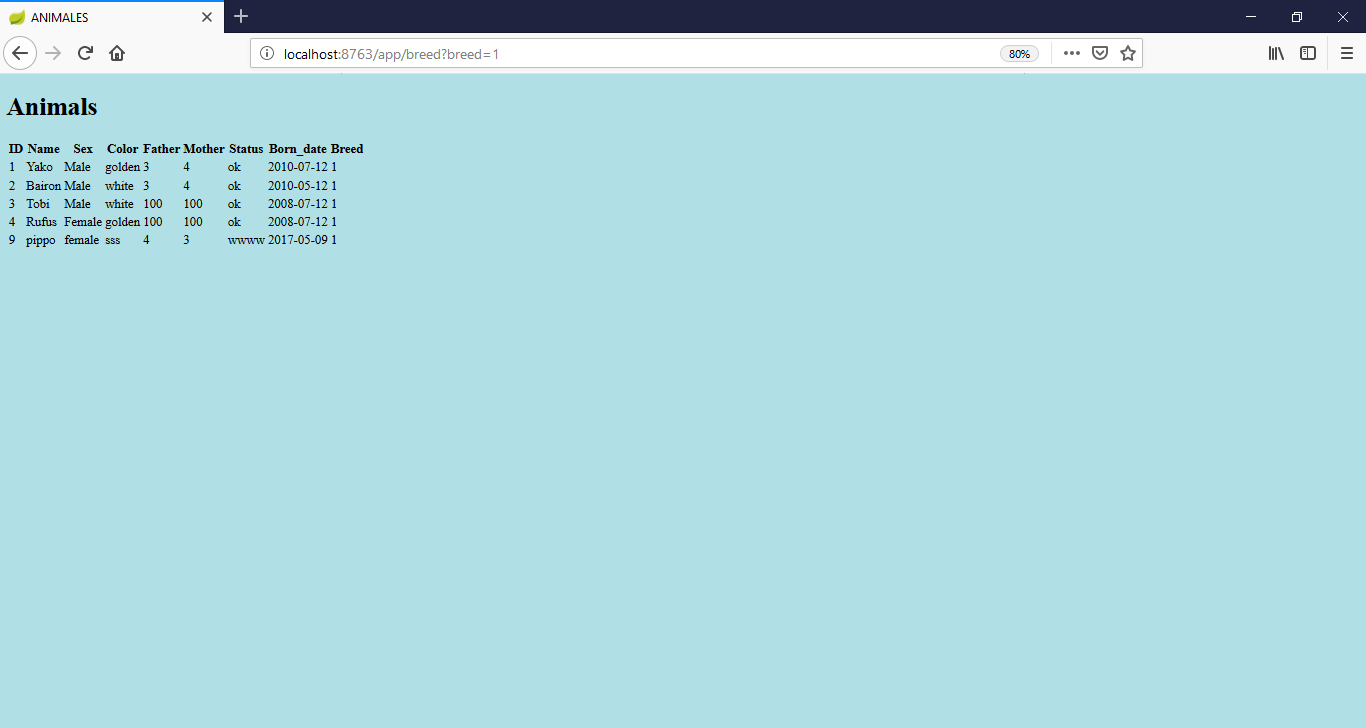


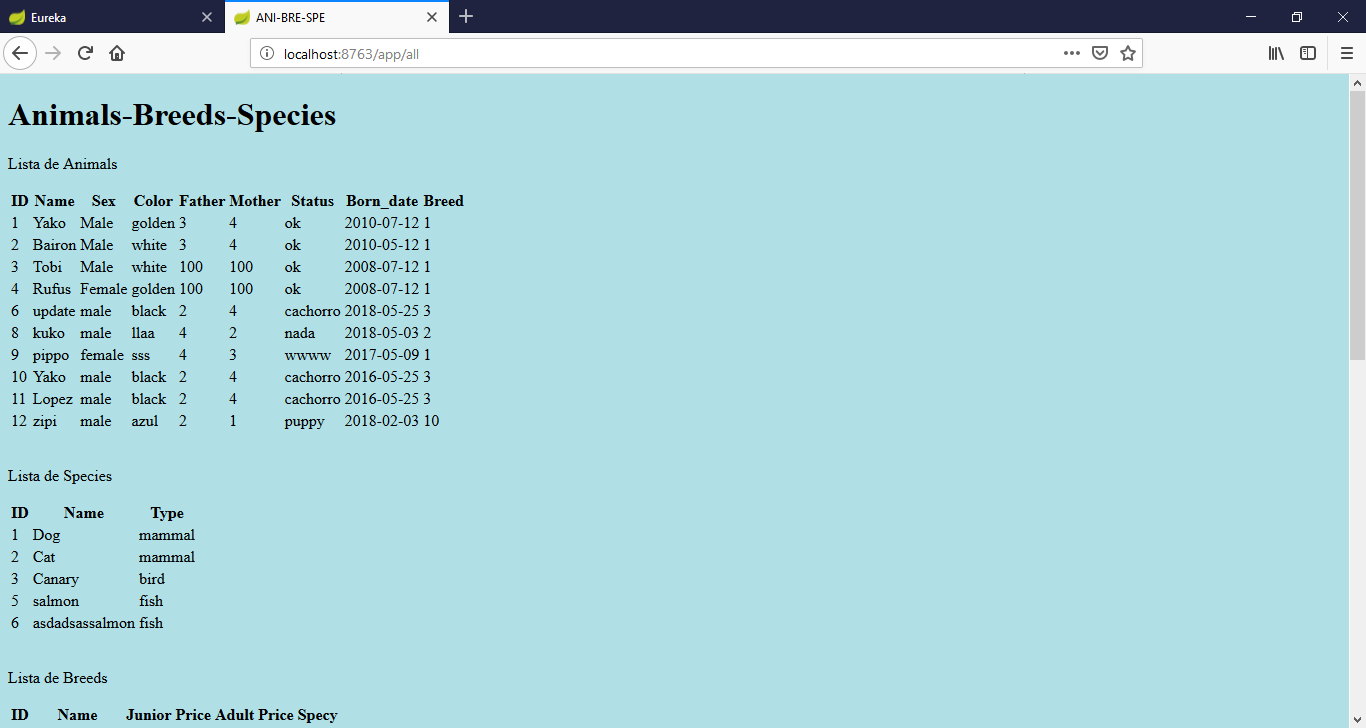


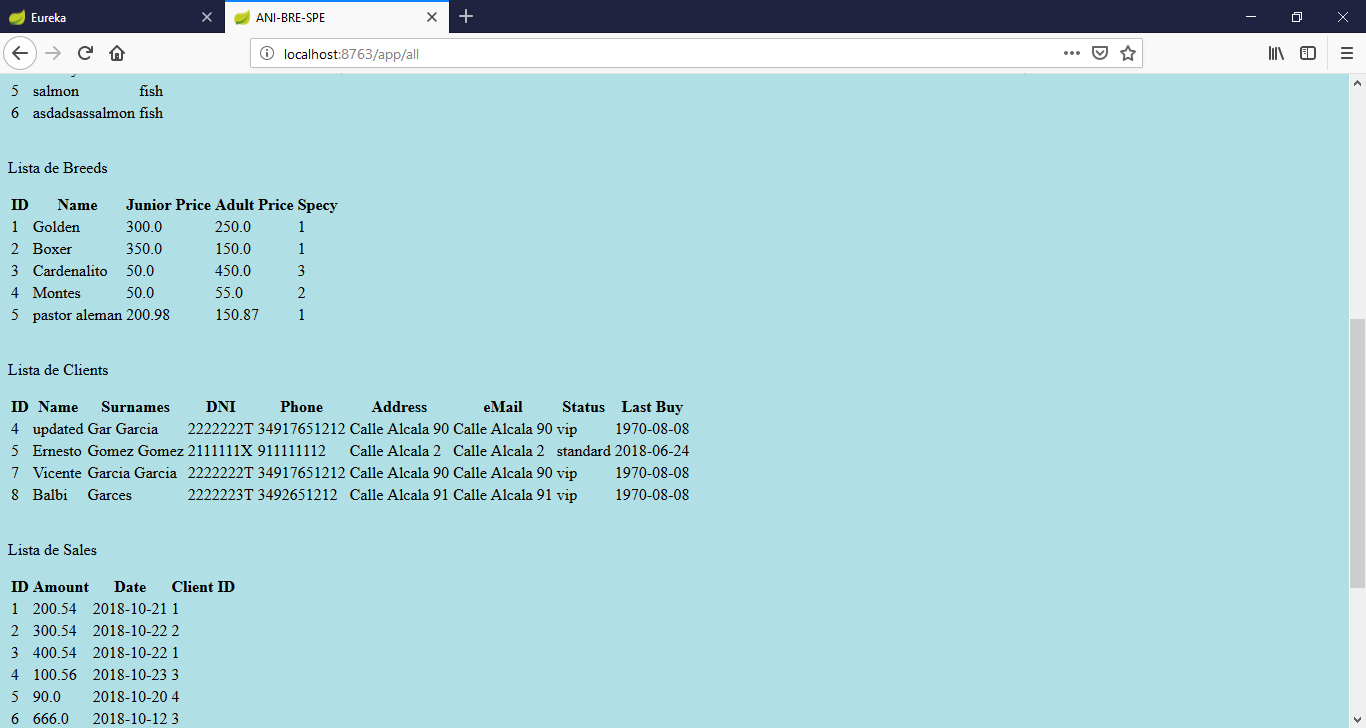


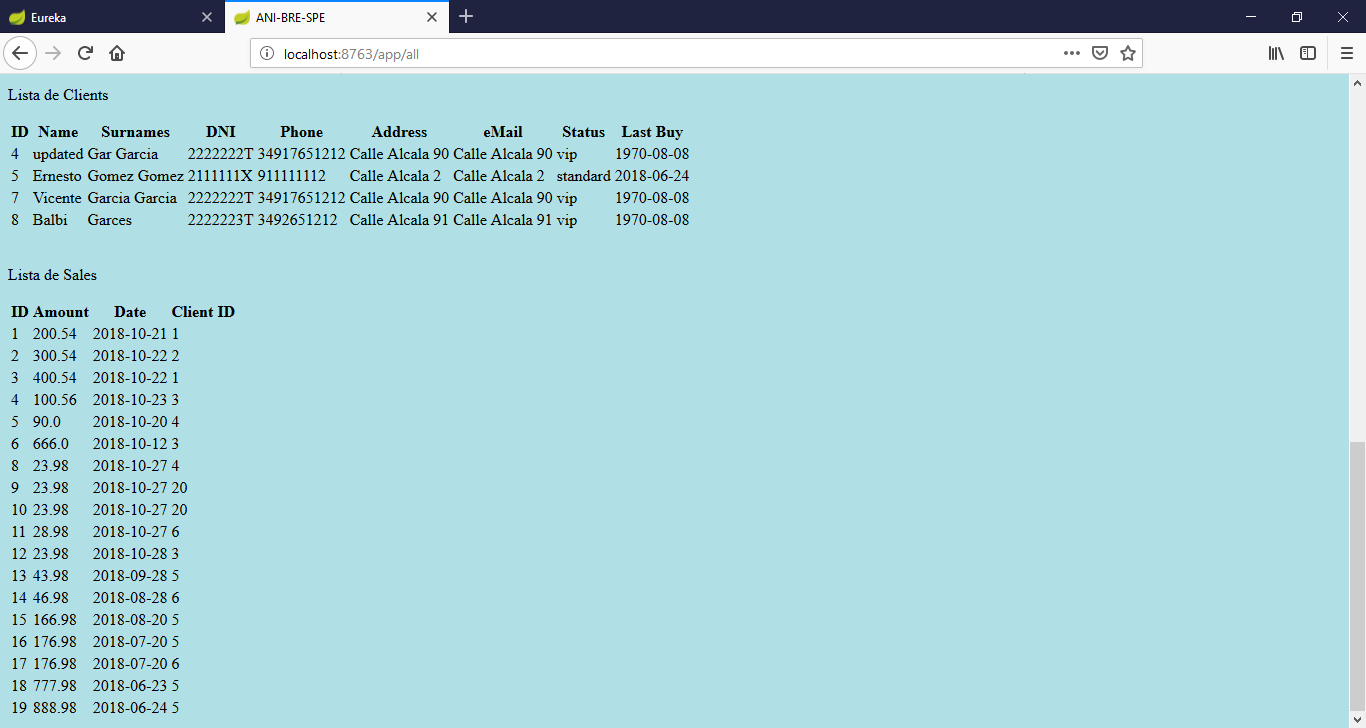












**Mejoras en próximas releases**

Para próximas versiones se puede plantear la creación de varios microservicios que permitan gestionar todos los costes derivados de la cría de animales. También se podrán crear más microservicios que permitan gestionar las vacunas y tratamientos veterinarios de los animales.

La aplicación podrá ser revisada para que pueda ser utilizada en otros idiomas a parte del inglés.

Se podrá realizar un re-styling a nivel Front de la aplicación para que sea más atractiva e intuitiva a los usuarios.

A nivel de seguridad, solamente están securizadas las url que gestiona el microservicio ms-usuarios. Se podría extender el nivel de seguridad a todas las urls de todos los microservicios.

Se puede implementar la funcionalidad para gestionar los perfiles de los usuarios y segregar el acceso a funcionalidades dependiendo del perfil.

No se han implementado logs para cada uno de los microservicios, quedaría este punto pendiente para futuras releases. También quedaría pendiente para próximas versiones el control de errores de todos los microservicios que se han creado.

A la hora de crear animal y breed no se comprueba que los ids de las otras entidades que forman parte de sus tablas existen. Es decir, a la hora de crear un animal se debería comprobar que existe el breed por ejemplo.

Queda pendiente para próximos releases la creación de formularios para poder crear las siguientes entidades: breed, specy, client y sale. Ahora mismo solamente se pueden crear por interfaz users y animals. Sin embargo, de los API rest de cada microservicio sí está disponible el CRUD de todas las entidades.