**DP2-G3-14**

**SPRINT 2,**

**ENTREGABLE 1:**

**LUNES 30 DE MARZO DE 2020**

[**https://github.com/fersolesp/DP2-G3-14**](https://github.com/fersolesp/DP2-G3-14)

Contenido

[1. Nivel de acabado 2](#_Toc36407537)

[2. Cómo hemos alcanzado los requisitos de cada nivel 3](#_Toc36407538)

[3. Análisis retrospectivo 4](#_Toc36407539)

[4. Cambios y consideraciones 7](#_Toc36407540)

[5. Problemas 8](#_Toc36407541)

# 1. Nivel de acabado

Como grupo nos hemos puesto de acuerdo para desarrollar una aplicación que cumple con los dos siguientes estándares definidos en la información/normas del proyecto:

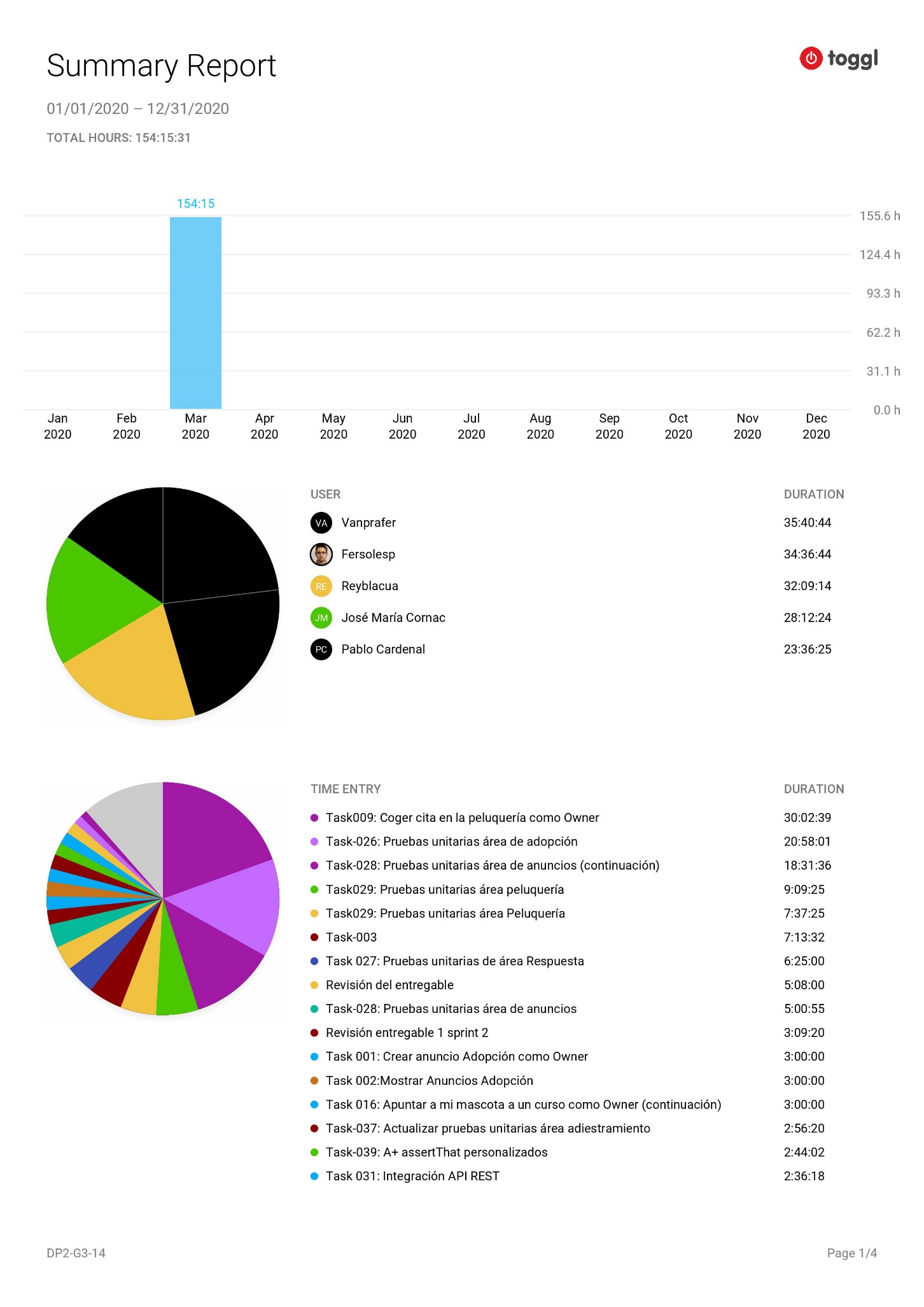
* **Aplicación de Nivel 2:** Hemos extendido el sistema de información web que viene por defecto, “PetClinic”, más específicamente hemos añadido 8 nuevas entidades con relaciones con otras entidades de la aplicación. Además, tenemos un total de 22 HU, por lo que hemos superado las 20 requeridas en dicho nivel.
* **Nivel hasta 10 puntos:**Aplicamos todos los requisitos necesarios hasta el nivel 9 y además hemos realizado un ejercicio adicional que requiere de investigación extra, en este caso siguiendo la recomendación de la asignatura hemos implementado el uso de “AssertThat personalizados”.

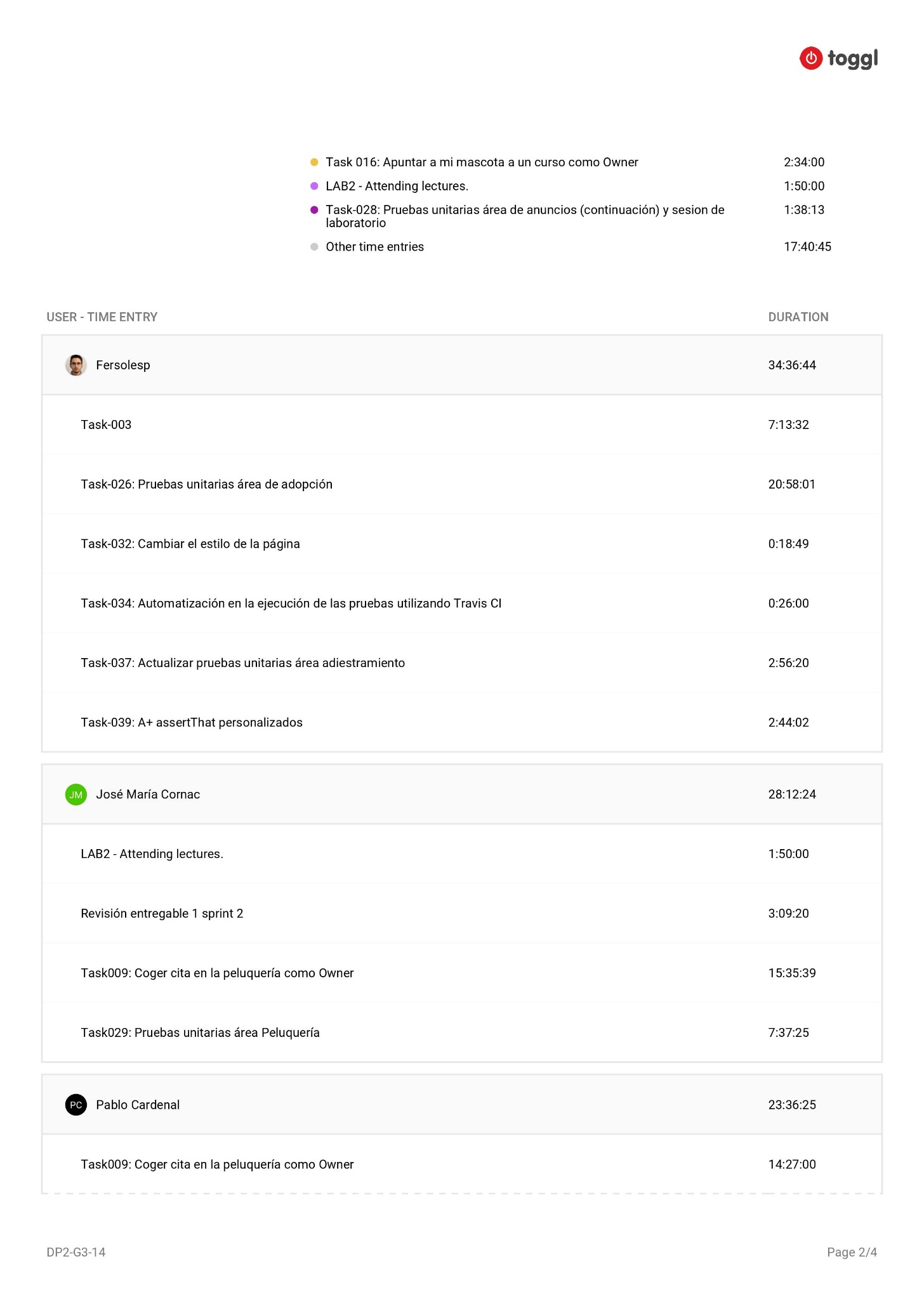
# 2. Cómo hemos alcanzado los requisitos de cada nivel

* **Nivel hasta 6 puntos*:*** Todos hemos tomado parte en la implementación de la nueva lógica y las pruebas unitarias, más específicamente:
* Área de Adiestramiento: Desarrollado por **Vanessa** y **Reyes**, probado por **Fernando**.
* Área de Peluquería: Desarrollado por **Pablo** y **J. María**, probado por **J. María** (parte de **Pablo**) y por **Pablo** (parte de **J. María**).
* Anuncios: Desarrollado por **Reyes** (al ser impares tenemos la desventaja de que habrá tareas que solo las desarrolle 1 persona), probado por **Vanessa**.
* Respuestas: Desarrollado por **Vanessa** y **Fernando**, probado por **Reyes**.
* **Nivel hasta 8 puntos*:*** La automatización en la ejecución de las pruebas utilizando Travis CI la ha llevado a cabo **Fernando**.
* **Nivel hasta 9 puntos*:*** Todos hemos llevado a cabo pruebas parametrizadas en la medida de lo posible para probar todos los casos posibles. Además, **Reyes** ha implementado una API que nos provee de datos y curiosidades sobre gratos en la página “Home” de la aplicación.
* **Nivel hasta 10 puntos*:*** El ejercicio extra, más específicamente “AssertThat personalizados” lo ha realizado **Fernando**.

# 3. Análisis retrospectivo

* Dado que **José María** y **Pablo** han hecho un único “merge” con todo su contenido del Sprint, su parte ha sido bastante opaca de cara al resto del grupo, y no se ha podido conocer los detalles de implementación de su parte hasta los últimos días, dificultando así el entendimiento del proyecto completo como conjunto. Para el próximo “Sprint” nos hemos propuesto hacer, al menos, un “merge” semanal por pareja, de forma que podamos acceder todos a los progresos que vamos realizando.
* Hemos tenido un pequeño problema en la organización y reparto de las tareas, ya que, **Pablo** y **José María** han desarrollado la lógica de su parte, así como también han implementado las pruebas de la misma, aunque sea cruzándose las partes de cada uno, no es la organización idónea. Teniendo en cuenta el hecho de que somos un número impar de integrantes en el grupo, y que esto dificulta la formación de parejas, pretendemos mejorar el reparto de tareas en el próximo “Sprint” de forma que esto no vuelva a suceder.
* En cuanto al rendimiento de las parejas, consideramos que hemos trabajado adecuadamente, siguiendo el concepto de “programación en parejas” y pensamos que ha sido beneficioso en el sentido de que cada uno de los miembros de la pareja aportaba sus ideas para cada aspecto que se iba a implementar o probar.
* Los esfuerzos en horas que hemos realizado son:





# 4. Cambios y consideraciones

* Hemos implementado todas las historias de usuario a excepción de:
  + 4; 5 y 6 (validaciones de Answer, que deben ser probadas)
  + 9, 10, 11, 12, 14, 15, que corresponden a la parte de creación de Appointment.

Se trata casi en su mayoría de historias de usuario de reglas de negocio, por lo que estimamos que el proyecto está implementado en torno a un 90 % del total.

* Hemos cambiado la historia de usuario número 21 que hablaba acerca de que sólo habría cursos para perros y gatos, por otra en la que especificamos que no se puedan crear inscripciones a un curso ya completo.
* Hemos añadido una historia de usuario número 22, de forma que se contemple el control de acceso de los usuarios a los datos de otros usuarios, ya que le hemos dedicado bastante tiempo y pensamos, es buena idea recogerlo como historia de usuario, aunque pudiera pensarse que fuera implícito.
* Respecto a los “mocks” en los “tests” de servicio, hemos considerado, finalmente, que no es necesario hacer uso de ellos en estas pruebas, puesto que, esto no supondría ninguna ventaja aparente y conllevaría la creación de numerosos objetos “dummy”. Por ejemplo, si en esta prueba del servicio de “inscription” optamos por usar “mocks”, tendríamos algo así:

@Test

**void** shouldFindInscriptionsGivingOwner() {

Owner owner1 = **this**.createDummyOwner(1, "George", "Franklin", "110 W. Liberty St.", "Madison",**false**,**true**,**false**,**true**, "6085551023", "owner1", "0wn3r");

Mockito.*when*(**this**.ownerService.findOwnerByUserName("owner1")).thenReturn(owner1);

Owner owner = **this**.ownerService.findOwnerByUserName(owner1.getUser().getUsername());

Iterable<Inscription> inscriptions =

**this**.inscriptionsService.findInscriptionsByOwner(owner);

org.assertj.core.api.Assertions.*assertThat*(inscriptions).hasSize(1);

}

Como podemos ver, supondría crear un “dummy” y que el servicio “mockeado” nos lo devuelva, algo que no aporta nada.

Tendría sentido usar “mocks” en estos “tests” siempre que se prueben métodos que no accedan a repositorios, pero en nuestro caso, todos lo hacen.

# 5. Problemas

**Problema**: En las pruebas de controlador, para hacer pruebas de tipo “post”, en los que es necesario enviar por el formulario un objeto Java, no encontramos forma de hacerlo, no se podía pasar por “.param()”, forzaba a pasar un “String”.

**Solución**: La clave es indicar el id del objeto que queremos pasar, en nuestro caso el “pet”:

“.param("pet", this.lillie.getId().toString())” --- siendo lillie un dummy pet.

Entonces, si previamente al “dummy” le hemos asignado el Id, podrá recuperarlo una vez se haga el Post.

**Problema**: Para realizar los “AssertThat” personalizados, la clase que lo implementaba necesitaba extender a “AbstractAssert” del paquete “AssertJ”. Sin embargo, aún estando en las dependencias de “Maven” y en las librerías, “Eclipse” no lograba importarlo.

**Solución**: Añadimos a “pom.xml” la dependencia de “AssertJ” e indicamos que tenga “scope” de tipo “provider”, porque si no es así, el paquete tiene “scope” de tipo “test” y no puede ser importado fuera del paquete de pruebas del proyecto. Por tanto, lo que deberíamos añadir a “pom.xml”:

<dependency>

<groupId>org.assertj</groupId>

<artifactId>assertj-core</artifactId>

<scope>provide</scope>

</dependency>