

PROFILINGS

Historia de Usuario 21 a la que se le aplica profile	Yo COMO administrador QUIERO poder crear nuevos veterinarios PARA aumentar la plantilla en caso de ser necesario.
Escenario	Un administrador crea un Veterinario.
Tipo de Profile	N+1 queries

En un principio, nuestros tiempos tras ejecutar dicha operación eran los siguientes:

<i>/login</i>	1.3 %	<pre>select diagnoses0_vet_id as vet_id5_3_0_, diagnoses0_id as id1_3_0_, diagnoses0_id ...</pre>	611.8	243	2.5	0.1
<i>/error</i>	1.1 %	<pre>select specialtie0_vet_id as vet_id1_15_0_, specialtie0_specialty_id as specialt2_15...</pre>	276.9	243	1.1	0.9
<i>/favicon.ico</i>	0.0 %	<pre>insert into users (enabled, password, username) values (?, ?, ?)</pre>	176.8	30	5.9	0.03
<i>/logout</i>	0.0 %	<pre>select specialty0_id as id1_11_, specialty0_name as name2_11_ from specialties speci...</pre>	114.0	92	1.2	3.0
		<pre>select user0_username as username1_14_0_, user0_enabled as enabled2_14_0_, user0.pa...</pre>	110.6	115	0.96	1.0
		<pre>select visits0_pet_id as pet_id6_17_0_, visits0_id as id1_17_0_, visits0_id as id1...</pre>	73.8	32	2.3	1.0
		<pre>select diagnoses0_pet_id as pet_id4_3_0_, diagnoses0_id as id1_3_0_, diagnoses0_id ...</pre>	59.1	32	1.8	1.0
		<pre>select username,password,enabled from users where username = ?</pre>	50.5	31	1.6	1.0
		<pre>select vet0_id as id1_16_, vet0.first_name as first_na2_16_, vet0.last_name as last...</pre>	42.2	32	1.3	7.6
		<pre>select username, authority from authorities where username = ?</pre>	38.7	31	1.2	1.0

Como podemos apreciar, cada vez que vamos a crear un nuevo veterinario, primero tenemos que pasar por la vista del listado de veterinarios que existen, y después acceder al formulario de creación del nuevo veterinario. Bien, pues aquí se puede ver que tenemos un caso de N+1 queries, ya que si multiplicamos 32(total count) * 7'6 (avg rows) obtenemos 243'2, que es el total de la consulta de especialidades y diagnósticos.

Para cada veterinario se consultan sus diagnósticos y especialidades con un elevado tiempo. Además, es importante saber que, en todo este proceso, resulta innecesario consultar los diagnósticos asociados a cada veterinario, ya que esto solo nos interesaría si accediésemos siendo veterinario.

En cuanto a los tiempos medios, no son muy malos, pero en cuestión al número medio de filas que nos traen cada uno si vemos algunos problemas, ya que para los veterinarios nos trae 7'6, pero para los diagnósticos y especialidades nos trae 0'1 y 0'9 respectivamente.

Cambios realizados para mejorar el rendimiento de esta historia de usuario:

Para mejorar esta historia de usuario, haremos varias cosas:

En primer lugar, como he dicho anteriormente, no nos interesa consultar los diagnósticos de los veterinarios que ya tenemos creados, ya que esto solo se aplica si el que está usando la aplicación es un veterinario. Lo que debemos hacer es que los diagnósticos se consulten solamente cuando se necesiten. Para conseguir hacer esto he hecho algún cambio en el código siguiendo el ejemplo implementado en el vídeo de clase:

```
// Añadido para Diagnosis
@OneToMany(cascade = CascadeType.ALL, mappedBy = "vet", fetch = FetchType.LAZY)
private Set<Diagnosis> diagnoses;
```

Simplemente cambio el fetchType de EAGER A LAZY, para que solo realice la consulta de los diagnósticos cuando sea necesario.

Por otra parte, con las especialidades hago esto mismo y también implemento una nueva query para que cuando se haga una consulta de los veterinarios, se obtengan todos estos junto a sus especialidades:

```
@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "vet_specialties", joinColumns = @JoinColumn(name = "vet_id"), inverseJoinColumns = @JoinColumn(name = "specialty_id"))
private Set<Specialty> specialties = new HashSet<>();
```

Cambio el fetchType de EAGER A LAZY, para que cuando se realice una consulta de las especialidades, se haga una consulta tras otra, disminuyendo así el tiempo.

```
@Override
@Query("SELECT DISTINCT v FROM Vet v LEFT JOIN FETCH v.specialties sp")
Collection<Vet> findAllWithSpecialties();
```

Además añadido un método adicional, con esta query que se muestra arriba, para que así seleccione todos los veterinarios y me los una junto a sus especialidades. Como los veterinarios pueden tener especialidad o no, usamos LEFT JOIN FETCH, y el DISTINCT para que no se repita ningún veterinario.

Tiempos tras realizar dichos cambios en nuestro código:

select visits0.pet_id as pet_id0_17_0, visits0.id as id0_17_0, visits0.id as id0_...	25.9	2	12.9	0.5
select username, authority from authorities where username = ?	23.3	35	0.67	1.0
select specialties0.id as id0_13_0, specialties0.name as name0_13_0, from specialties speci...	17.3	33	0.52	1.0
select pets0.id as id0_8_0, pets0.name as name0_8_0, pets0.birth_date as birth_date0_...	15.5	2	7.8	1.0
insert into vets (first_name, last_name, username) values (?, ?, ?)	15.1	4	3.8	1.0
insert into vet_specialties (vet_id, specialty_id) values (?, ?)	7.4	4	1.8	1.0
insert into authorities (authority, username) values (?, ?)	6.4	4	1.6	1.0
select owners0.id as id0_5_0, pets0.id as id0_6_0, owners0.first_name as first_name0_...	3.1	1	3.1	2.0
select authorities0.username as username0_0_0, authorities0.authority as authority0_0_...	2.1	4	0.53	0
select diagnoses0.pet_id as pet_id0_3_0, diagnoses0.id as id0_3_0, diagnoses0.id ...	1.9	2	0.94	0.5
select pettypes0.id as id0_13_0, pettypes0.name as name0_13_0, from types pettypes0_ order...	1.8	1	1.8	6.0
select vets0.id as id0_16_0, vets0.first_name as first_name0_16_0, vets0.last_name as last...	1.6	2	0.78	1.0
select specialties0.vet_id as vet_id0_13_0, specialties0.specialty_id as specialty0_13_...	1.1	2	0.54	1.0
select visits0.id as id0_17_0, visits0.visit_date as visit_date0_17_0, visits0.description...	1.0	2	0.52	0

Podemos observar mejoras considerables en los tiempos, ya que por ejemplo la consulta de los diagnósticos ni aparece al no ser necesaria en esta operación.

Además, el tiempo de la consulta de especialidades se ha reducido mucho, aumentando así el rendimiento de nuestra historia de usuario.

Historia de Usuario 17 a la que se le aplica profile	Yo COMO owner QUIERO poder registrar una mascota a mi nombre PARA poder tener un registro de las mismas.																																													
Escenario	Ver sus mascotas y crear una nueva																																													
Tipo de Profile	Projections																																													
En un principio, nuestros tiempos tras ejecutar dicha operación eran los siguientes:																																														
Tras haber lanzado Gatling con 30 usuarios en 20 segundos los resultados es Glowroot son los siguientes:																																														
	<table><thead><tr><th></th><th>Total time ▼ (ms)</th><th>Total count</th><th>Avg time (ms)</th><th>Avg rows</th></tr></thead><tbody><tr><td>select visits0_.pet_id as pet_id5_17_0_, visits0_.id as id1_17_0_, visits0_.id as id1_...</td><td>715,1</td><td>298</td><td>2,4</td><td>0</td></tr><tr><td>select diagnoses0_.pet_id as pet_id4_3_0_, diagnoses0_.id as id1_3_0_, diagnoses0_.id ...</td><td>535,9</td><td>298</td><td>1,8</td><td>0</td></tr><tr><td>select owner0_.id as id1_5_0_, pets1_.id as id1_6_1_, owner0_.first_name as first_na2_...</td><td>489,7</td><td>93</td><td>5,3</td><td>3,2</td></tr><tr><td>select pettype0_.id as id1_13_, pettype0_.name as name2_13_ from types pettype0_ order...</td><td>272,8</td><td>123</td><td>2,2</td><td>6,0</td></tr><tr><td>select user0_.username as username1_14_0_, user0_.enabled as enabled2_14_0_, user0_.pa...</td><td>153,7</td><td>93</td><td>1,7</td><td>1,0</td></tr><tr><td>select username,password,enabled from users where username = ?</td><td>151,4</td><td>30</td><td>5,0</td><td>1,0</td></tr><tr><td>insert into pets (name, birth_date, owner_id, room_id, sitter_id, type_id) values (?, ...</td><td>125,5</td><td>3</td><td>41,8</td><td>1,0</td></tr><tr><td>select username, authority from authorities where username = ?</td><td>49,1</td><td>30</td><td>1,6</td><td>1,0</td></tr></tbody></table>		Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows	select visits0_.pet_id as pet_id5_17_0_, visits0_.id as id1_17_0_, visits0_.id as id1_...	715,1	298	2,4	0	select diagnoses0_.pet_id as pet_id4_3_0_, diagnoses0_.id as id1_3_0_, diagnoses0_.id ...	535,9	298	1,8	0	select owner0_.id as id1_5_0_, pets1_.id as id1_6_1_, owner0_.first_name as first_na2_...	489,7	93	5,3	3,2	select pettype0_.id as id1_13_, pettype0_.name as name2_13_ from types pettype0_ order...	272,8	123	2,2	6,0	select user0_.username as username1_14_0_, user0_.enabled as enabled2_14_0_, user0_.pa...	153,7	93	1,7	1,0	select username,password,enabled from users where username = ?	151,4	30	5,0	1,0	insert into pets (name, birth_date, owner_id, room_id, sitter_id, type_id) values (?, ...	125,5	3	41,8	1,0	select username, authority from authorities where username = ?	49,1	30	1,6	1,0
	Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows																																										
select visits0_.pet_id as pet_id5_17_0_, visits0_.id as id1_17_0_, visits0_.id as id1_...	715,1	298	2,4	0																																										
select diagnoses0_.pet_id as pet_id4_3_0_, diagnoses0_.id as id1_3_0_, diagnoses0_.id ...	535,9	298	1,8	0																																										
select owner0_.id as id1_5_0_, pets1_.id as id1_6_1_, owner0_.first_name as first_na2_...	489,7	93	5,3	3,2																																										
select pettype0_.id as id1_13_, pettype0_.name as name2_13_ from types pettype0_ order...	272,8	123	2,2	6,0																																										
select user0_.username as username1_14_0_, user0_.enabled as enabled2_14_0_, user0_.pa...	153,7	93	1,7	1,0																																										
select username,password,enabled from users where username = ?	151,4	30	5,0	1,0																																										
insert into pets (name, birth_date, owner_id, room_id, sitter_id, type_id) values (?, ...	125,5	3	41,8	1,0																																										
select username, authority from authorities where username = ?	49,1	30	1,6	1,0																																										

Breakdown:	total (ms)	count
http request	52,6	1,0
servlet dispatch	27,6	0,2
jsp render	4,5	0,2
jdbc query	2,8	1,1
hibernate query	1,3	0,2
jdbc query	0,36	0,2
hibernate commit	0,47	0,3
jdbc commit	0,24	0,3
jdbc get connection	0,40	0,2
hibernate persist	0,24	0,004
show more / show all		

By percent of total time	
All Web Transactions	100.0 %
/owner/pets/new	36.4 %
/webjars/**	31.5 %
/**	19.0 %
/owner/pets	9.1 %
/	2.8 %
/login	0.9 %
/favicon.ico	0.3 %

Como vemos las consultas que más tiempo han tardado han sido las relacionadas con ver la lista de mascotas ya que para ello pido a la base de datos los datos del owner completos con todas sus mascotas incluyendo estás datos que realmente no vamos a mostrar al usuario.

Aunque el que mayor porcentaje de tiempo se lo lleve owner/pets/new este es un método post que va a crear una mascota, por lo que tardará mucho más que el resto de métodos, por lo que en este caso nos vamos a concentrar en la consulta de owner/pets que es la que se trae todas las mascotas del owner.

Cambios realizados para mejorar el rendimiento de esta historia de usuario:

Cuando nosotros como owner listamos nuestras mascotas le estamos pidiendo a la base de datos que traiga todos los datos de ese owner, eso incluye su nombre, nombre de usuario, mascotas y todo lo que esté relacionado con esta, para evitar esto hemos creado una clase llamada OwnerPets en la carpeta de projections, esa clase incluirá solo los parámetros que vamos a necesitar:

```
package org.springframework.samples

import java.time.LocalDate;

public interface OwnerPets {

    String getName();
    LocalDate getBirthDate();
    String getType();
    Integer getId();
    LocalDate getVisitDate();
    String getVisitDescription();
}
```

Una vez creada esta clase vamos a añadir al repositorio la consulta findOwnerPets(String nameOwner) al que le pasaremos el nombre el owner y nos devolverá únicamente los datos que se incluyan en OwnerPets, la query SQL es la siguiente

```
@Query("SELECT p.id AS id, p.name AS name, p.birthDate AS birthDate, "
+ "t.name AS type, v.description AS visitDescription, v.date AS visitDate"
+ " FROM Pet p INNER JOIN p.type t LEFT JOIN p.visits v WHERE p.owner.user.username =:nameOwner")
```

Posteriormente añadimos dicho método al servicio y modificamos el controlador para que realice esa consulta, transformando los datos de OwnerPets a un Owner que solo contendra los datos que se necesiten.

Tiempos tras realizar dichos cambios en nuestro código:

Luego de realizar los cambios anteriores volvimos a lanzar Gatling con 30 usuario en 20 segundos, esta vez los resultados fueron los siguientes:

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select visits0_.pet_id as pet_id5_17_0_, visits0_.id as id1_17_0_, visits0_.id as id1_...	409,3	240	1,7	0
select diagnoses0_.pet_id as pet_id4_3_0_, diagnoses0_.id as id1_3_0_, diagnoses0_.id ...	396,0	240	1,7	0
insert into pets (name, birth_date, owner_id, room_id, sitter_id, type_id) values (?, ...	209,0	3	69,7	1,0
select pettype0_.id as id1_13_, pettype0_.name as name2_13_ from types pettype0_ order...	150,1	123	1,2	6,0
select owner0_.id as id1_5_0_, pets1_.id as id1_6_1_, owner0_.first_name as first_na2...	107,1	60	1,8	4,0
select pet0_.id as col_0_0_, pet0_.name as col_1_0_, pet0_.birth_date as col_2_0_, pet...	96,4	33	2,9	2,2
select user0_.username as username1_14_0_, user0_.enabled as enabled2_14_0_, user0_.pa...	65,4	60	1,1	1,0
select owner0_.id as col_0_0_ from owners owner0_ where owner0_.username=?	34,3	33	1,0	1,0
select username,password,enabled from users where username = ?	20,6	30	0.69	1,0

			By percent of total time ▾	
			All Web Transactions	100.0 %
Breakdown:	total (ms)	count	/owner/pets/new	44.0 %
http request	18,5	1,0	/webjars/**	28.5 %
servlet dispatch	4,5	0,2	/**	18.8 %
jsp render	0.94	0,2	/owner/pets	5.2 %
jdbc query	1,4	0,9	/	2.6 %
hibernate commit	0.74	0,3	/login	0.8 %
jdbc commit	0.17	0,3		
hibernate query	0.46	0,2		
jdbc query	0.33	0,2		
hibernate persist	0.29	0.004		
jdbc query	0.28	0.004		

Como vemos hay una mejora considerable en los tiempos reduciendo además el número de consultas que se realizan. Si nos fijamos en el porcentaje de tiempos vemos que /owner/pets a pasado de ocupar un 9.1% a ocupar un 5.2% por lo tanto hemos hecho que el poder listar las mascotas sea mas optimo.

Historia de Usuario 7 a la que se aplica profile	7. Yo COMO usuario QUIERO ver la lista de causas aceptadas PARA poder realizar mis donaciones.
Escenario	Un owner visualiza la lista de causas
Tipo de Profile	Caches

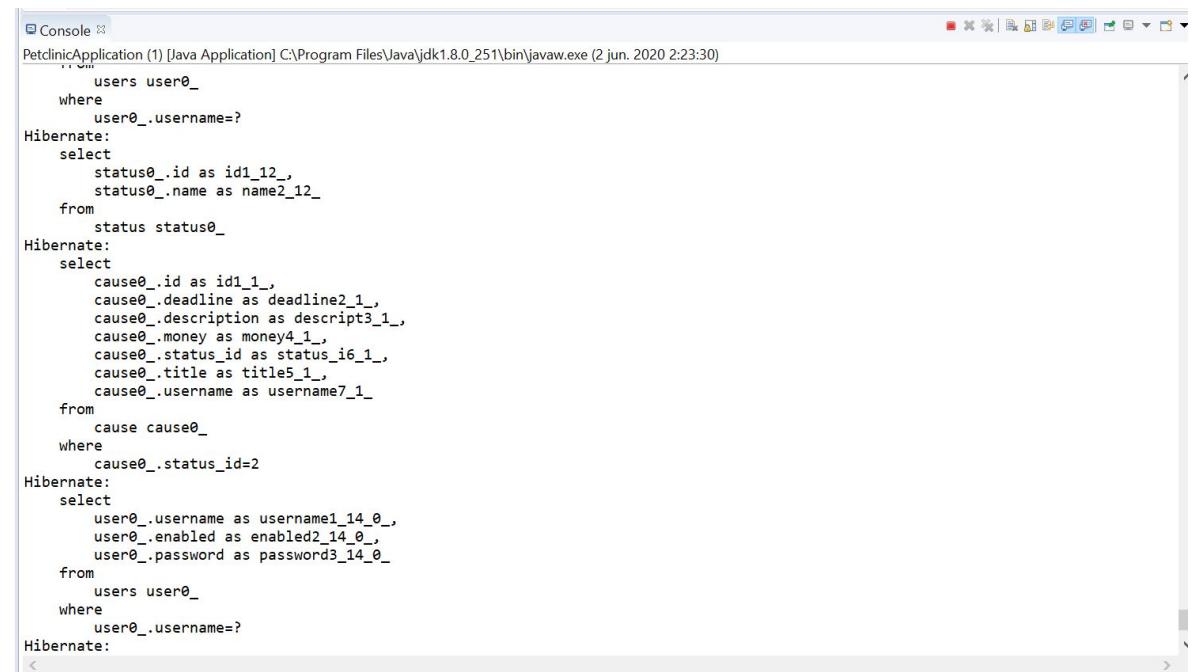
En un principio, los tiempos resultantes al realizar esta operación son los siguientes:

Aquí podemos ver reflejada los tiempos que ha tardado la aplicación sin realizar aún ningún cambio en nuestro sistema.



No es un tiempo desorbitado pero intentaremos mejorarlo con la implementación del

profiling enfocado en la caché. Además, si miramos la consola de nuestra aplicación, podemos ver que cada vez que entremos dentro de esta funcionalidad se vuelven a cargar todos los datos dado que no se quedan guardados en la caché de nuestro sistema.



```
Console
PetclinicApplication (1) [Java Application] C:\Program Files\Java\jdk1.8.0_251\bin\javaw.exe (2 jun. 2020 2:23:30)

users user0_
where
  user0_.username=?
Hibernate:
select
  status0_.id as id1_12_,
  status0_.name as name2_12_
from
  status status0_
Hibernate:
select
  cause0_.id as id1_1_,
  cause0_.deadline as deadline2_1_,
  cause0_.description as descript3_1_,
  cause0_.money as money4_1_,
  cause0_.status_id as status_i6_1_,
  cause0_.title as title5_1_,
  cause0_.username as username7_1_
from
  cause cause0_
where
  cause0_.status_id=2
Hibernate:
select
  user0_.username as username1_14_0_,
  user0_.enabled as enabled2_14_0_,
  user0_.password as password3_14_0_
from
  users user0_
where
  user0_.username=?
Hibernate:
```

Cambios realizados para mejorar el rendimiento de esta historia de usuario:

Pues hemos realizado distintos cambios dentro de nuestro código para implementar la caché que nos ayude a acelerar este proceso. En primer lugar, buscamos el método que vamos a cachear y hacemos lo siguiente:

```
@Transactional
@Cacheable("AcceptedCauses")
public Collection<Cause> findAcceptedCauses() throws DataAccessException {
    return this.causeRepository.findAcceptedCauses();
}
```

Con esta anotación, señalamos que este método será al que le añadiremos el caché. Además, añadiremos esto al método save para que cada vez que guardemos una causa la caché se borre automáticamente:

```
@Transactional
@CacheEvict(cacheNames = "AcceptedCauses", allEntries = true)
public void saveCauses(@Valid final Cause cause) throws DataAccessException {
    this.causeRepository.save(cause);
}
```

Además añadiremos dos clases al paquete de configuración que son los siguientes:

```

CauseController.java  CacheConfiguration.java  CacheLogger.java  PetclinicApplication.java  CauseSen
1
2 package org.springframework.samples.petclinic.configuration;
3
4 import org.springframework.cache.annotation.EnableCaching;
5
6
7 @Configuration
8 @EnableCaching
9 public class CacheConfiguration {
10
11 }
12

```

En la clase CacheLogger determina que es lo que aparecerá en consola al realizar el método al que le estamos añadiendo el caché. Nos mostrará el tipo de evento, el valor antiguo guardado en caché y el nuevo valor que obtenemos.

```

CauseController.java  CacheConfiguration.java  *CacheLogger.java  PetclinicApplication.java  CauseService.java  ehc
1
2 package org.springframework.samples.petclinic.configuration;
3
4 import org.ehcache.event.CacheEvent;
5
6
7 public class CacheLogger implements CacheEventListener<Object, Object> {
8
9     private final Logger log = LoggerFactory.getLogger(CacheLogger.class);
10
11     @Override
12     public void onEvent(final CacheEvent<?, ?> event) {
13         this.log.info("Key: {} | EventType: {} | Old value: {} | New Value: {}",
14             event.getKey(), event.getType(), event.getOldValue(), event.getNewValue());
15     }
16 }
17
18
19
20
21
22

```

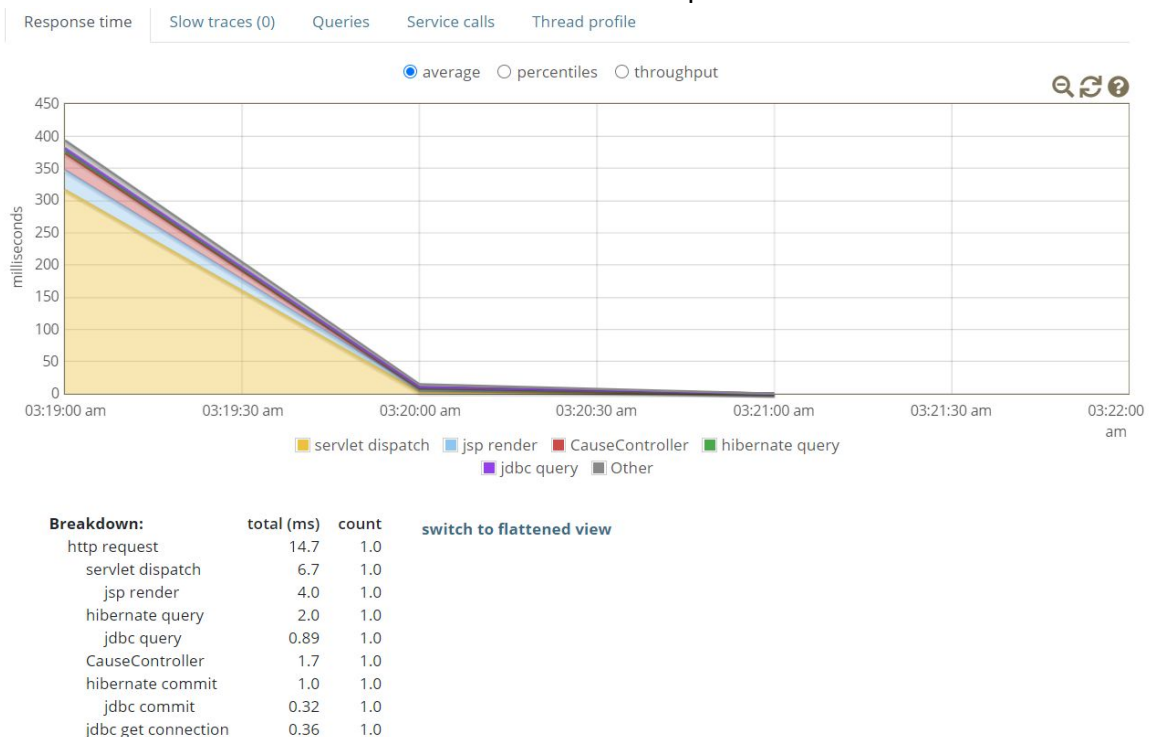
Crearemos también un ehcache3.xml en el que definiremos las distintas características de nuestro caché, como por ejemplo, de que cada vez que pasen 120 segundos la caché se borre automáticamente. El archivo .xml es el siguiente:

```
1 <config
2     xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
3     xmlns='http://www.ehcache.org/v3'
4     xsi:schemaLocation="
5         http://www.ehcache.org/v3
6         http://www.ehcache.org/schema/ehcache-core-3.7.xsd">
7
8     <!-- Persistent cache directory -->
9     <!--< persistence directory="spring-boot-ehcache/cache" />-->
10
11     <!-- Default Cache Template -->
12     <cache-template name="default">
13         <expiry>
14             <ttl unit="seconds">120</ttl>
15         </expiry>
16         <listeners>
17             <listener>
18                 <class>org.springframework.samples.petclinic.configuration.CacheLogger</class>
19                 <event-firing-mode>ASYNCHRONOUS</event-firing-mode>
20                 <event-ordering-mode>UNORDERED</event-ordering-mode>
21                 <events-to-fire-on>CREATED</events-to-fire-on>
22                 <events-to-fire-on>EXPIRED</events-to-fire-on>
23                 <events-to-fire-on>EVICTED</events-to-fire-on>
24             </listener>
25         </listeners>
26         <resources>
27             <heap>1000</heap>
28         </resources>
29     </cache-template>
30
31     <cache alias="AcceptedCauses" uses-template="default">
32         <key-type>org.springframework.cache.interceptor.SimpleKey</key-type>
33         <value-type>java.util.Collection</value-type>
34     </cache>
35 </config>
```

Por último, añadiremos una modificación en el application.properties para finalizar la configuración necesaria para el profiling de caché.

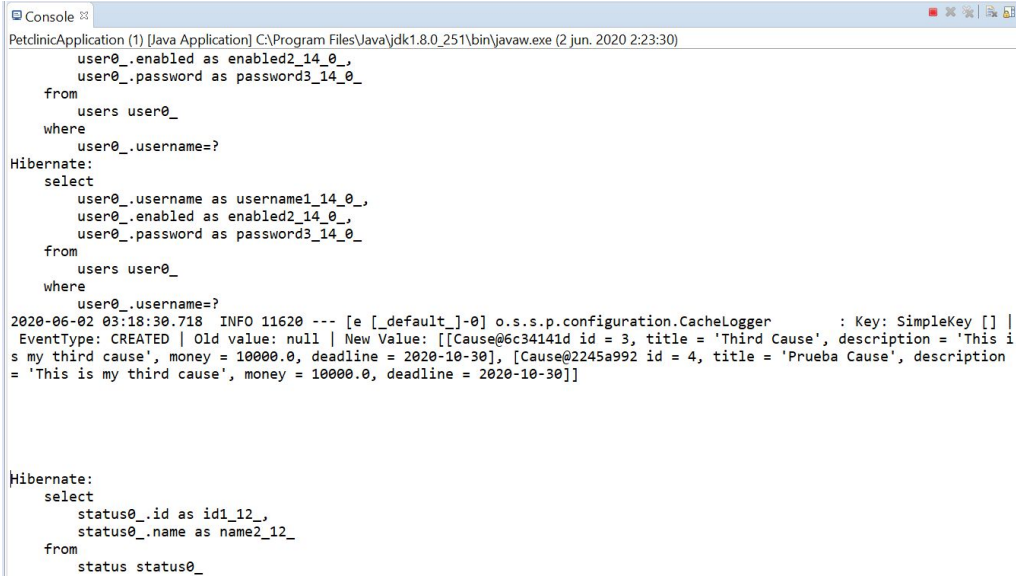
Cambios obtenidos al terminar el profiling de caché:

Como podemos ver, hemos obtenido mejores resultados tras la realización de los cambios comentados anteriormente. Este sería el tiempo detallado:



Podemos visualizar en la consola de nuestro sistema que el profiling se ha realizado con éxito dado que muestra el Log comentado en la clase CacheLogger y que nos verifica su

correcto funcionamiento. Además, podemos ver que si volvemos a buscar las causas antes de los 2 minutos (que es el tiempo definido para que se borre la caché), el sistema no vuelve a buscar las causas dado que los valores están guardados:



The screenshot shows a Java console window titled "Console" with the following content:

```
PetclinicApplication (1) [Java Application] C:\Program Files\Java\jdk1.8.0_251\bin\javaw.exe (2 jun. 2020 2:23:30)

    user0_.enabled as enabled2_14_0_,
    user0_.password as password3_14_0_
  from
    users user0_
  where
    user0_.username=?
Hibernate:
  select
    user0_.username as username1_14_0_,
    user0_.enabled as enabled2_14_0_,
    user0_.password as password3_14_0_
  from
    users user0_
  where
    user0_.username=?
2020-06-02 03:18:30.718 INFO 11620 --- [e [_default_]-0] o.s.s.p.configuration.CacheLogger      : Key: SimpleKey [] |
EventType: CREATED | Old value: null | New Value: [[Cause@6c34141d id = 3, title = 'Third Cause', description = 'This i
s my third cause', money = 10000.0, deadline = 2020-10-30], [Cause@2245a992 id = 4, title = 'Prueba Cause', description
= 'This is my third cause', money = 10000.0, deadline = 2020-10-30]]
Hibernate:
  select
    status0_.id as id1_12_,
    status0_.name as name2_12_
  from
    status status0_
```