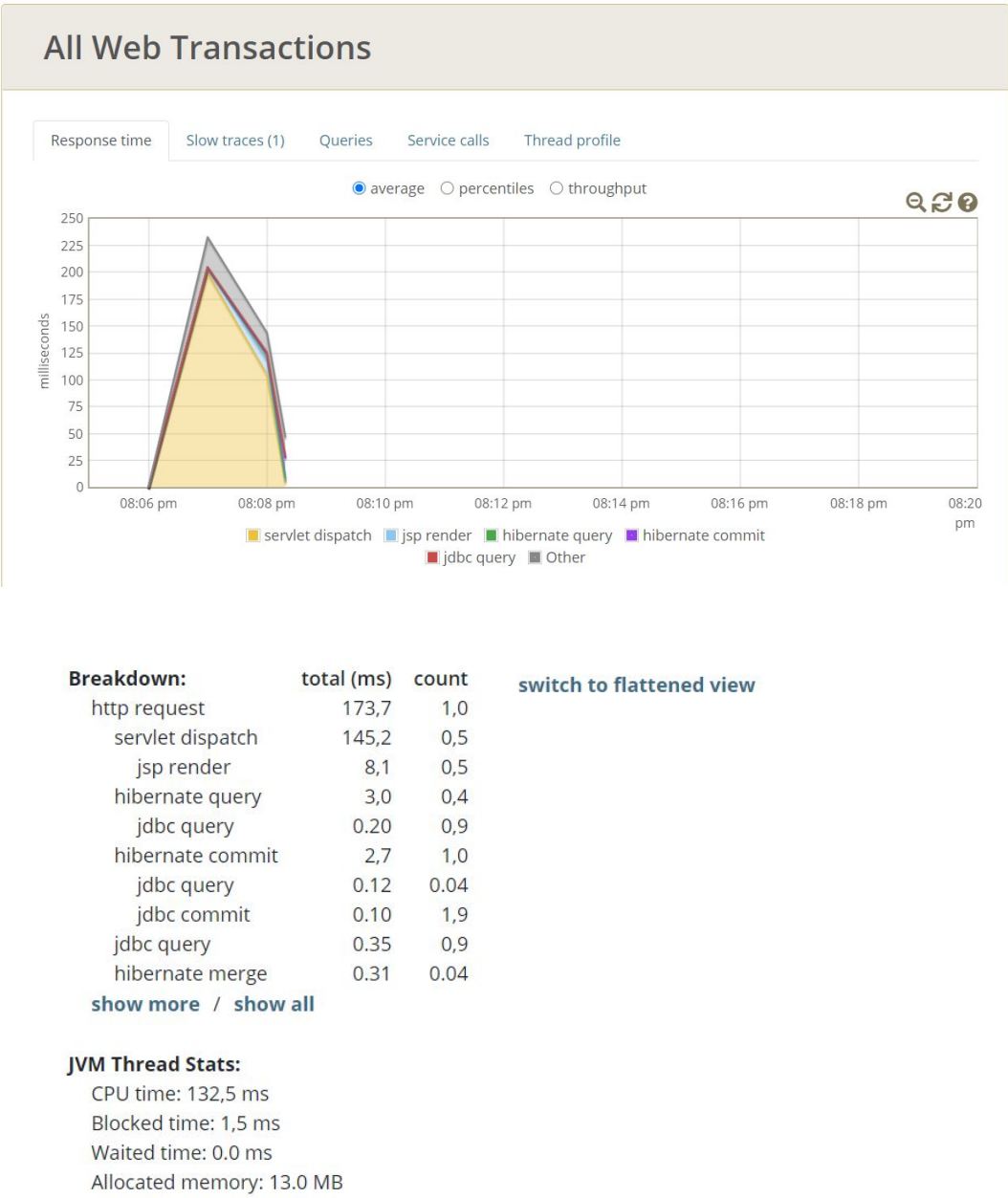


Diseño y Pruebas II

Profiling

Grupo G1-15

Historia 9: Mecánico modifica avería



En estas dos imágenes se muestran los resultados de replicar el escenario de la historia en cuestión de manera manual y una sola vez. En la primera gráfica se ve que la mayor parte del tiempo se fue en el servlet dispatch.

En la segunda foto se observa lo mismo que en la primera. Seguidamente, se encuentra hibernate query e hibernate commit, que hacen alusión a la base de datos, pero tardando unos tiempos despreciables, comparados con los anteriores.

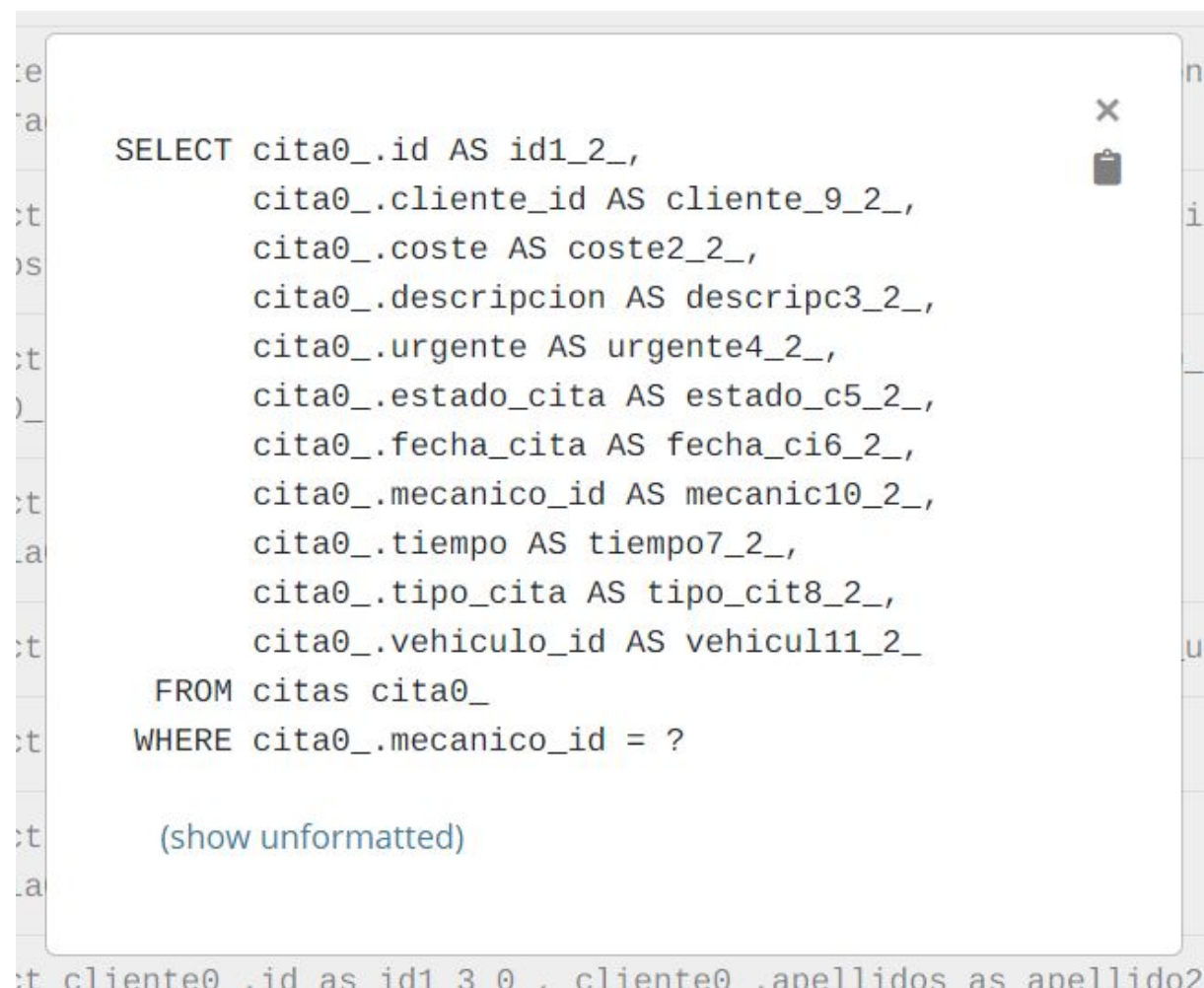
By percent of total time ▾	
All Web Transactions	100.0 %
/	60.0 %
/mecanicos/vehiculos/*/averia/*/edit	13.2 %
/mecanicos/citas	12.7 %
/mecanicos/vehiculos/*/averia	8.5 %
/**	2.9 %
/login	1.5 %
/webjars/**	0.9 %
/favicon.ico	0.4 %

En esta captura se puede observar como la mayor parte del tiempo, un 60%, se va en cargar la página principal. Esta página va seguida en porcentaje de tiempo de la de mecánico actualiza averías, que se corresponde con la página de la historia 9, que es la que nos ocupa. Muy cerca en cuanto a tiempo se encuentra la página de mecánico lista citas, de la historia 11.

Response time	Slow traces (1)	Queries	Service calls	Thread profile				
					Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows
		<code>select mecanico@_.id as col_@_ from mecanicos mecanico@_ where mecanico@_.nombre_usuari...</code>			3,5	9	0.39	1,0
		<code>update averias set cita_id=?, complejidad=?, coste=?, descripcion=?, reparada=?, mecanico...</code>			2,7	1	2,7	1,0
		<code>select cita@_.id as id1_2_, cita@_.cliente_id as cliente_@_2_, cita@_.coste as coste2_2_,...</code>			2,0	6	0.34	1,0
		<code>select cita@_.id as id1_2_@_, cita@_.cliente_id as cliente_@_2_@_, cita@_.coste as coste2...</code>			1,7	7	0.24	1,0
		<code>select averia@_.id as id1_1_, averia@_.cita_id as cita_id@_1_, averia@_.complejidad as co...</code>			1,4	6	0.24	1,0
		<code>select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?</code>			1,3	1	1,3	1,0
		<code>select username, authority from authorities where username = ?</code>			1,2	1	1,2	1,0
		<code>select averia@_.id as id1_1_, averia@_.cita_id as cita_id@_1_, averia@_.complejidad as co...</code>			0.59	4	0.15	1,0
		<code>select cliente@_.id as id1_3_@_, cliente@_.apellidos as apellido2_3_@_, cliente@_.direcci...</code>			0.38	2	0.19	1,0
		<code>select vehiculo@_.id as id1_6_@_, vehiculo@_.activo as activo2_6_@_, vehiculo@_.cliente_i...</code>			0.26	2	0.13	1,0
		<code>select mecanico@_.id as id1_4_@_, mecanico@_.apellidos as apellido2_4_@_, mecanico@_.dire...</code>			0.24	2	0.12	1,0

En esta imagen se puede apreciar el listado de las consultas realizadas a la base de datos. Por tiempo medio, la consulta que más a tardado es la segunda, que se corresponde con el update de avería con los datos nuevos a la base de datos. Sin embargo la que ha consumido más tiempo en total se trata de la primera, que es la que se realiza para asegurar la identidad del mecánico que está realizando la actualización. La consulta no toma mucho tiempo, pero se realiza muchas veces, 9, debido a que cada vez que el mecánico accede a una página, se comprueba su identidad.

La siguiente captura se corresponde a la tercera consulta de la lista:

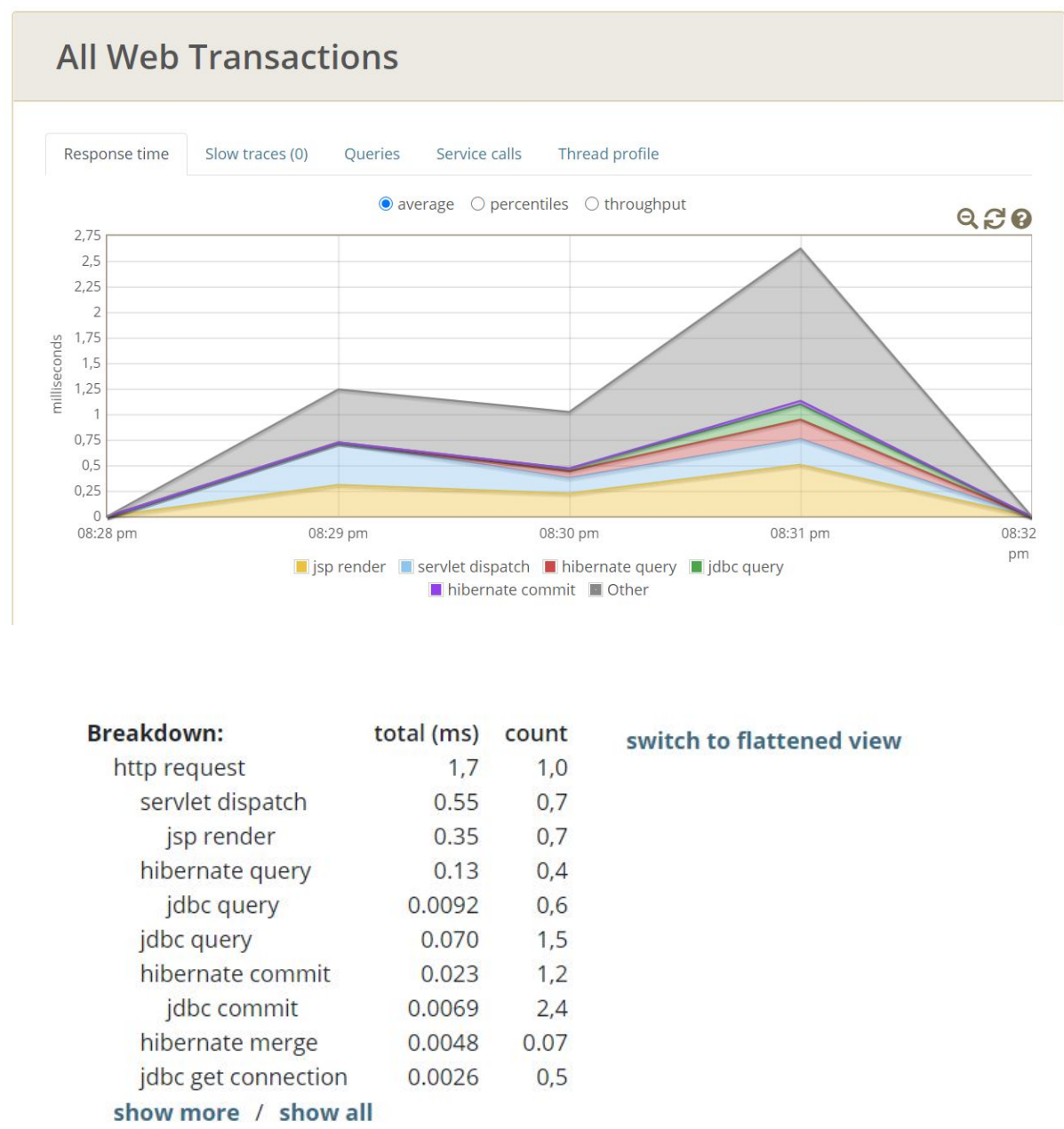


```
SELECT cita0_.id AS id1_2_,
       cita0_.cliente_id AS cliente_9_2_,
       cita0_.coste AS coste2_2_,
       cita0_.descripcion AS descripc3_2_,
       cita0_.urgente AS urgente4_2_,
       cita0_.estado_cita AS estado_c5_2_,
       cita0_.fecha_cita AS fecha_ci6_2_,
       cita0_.mecanico_id AS mecanic10_2_,
       cita0_.tiempo AS tiempo7_2_,
       cita0_.tipo_cita AS tipo_cit8_2_,
       cita0_.vehiculo_id AS vehicul11_2_
FROM citas cita0_
WHERE cita0_.mecanico_id = ?

(show unformatted)
```

Esta consulta se trae el listado de citas asignadas a un mecánico en cuestión. Muchos de estos datos de cada cita que se traen, no se llegan ni a mostrar siquiera en el listado que se le muestra al mecánico.


Posteriormente, realizamos una nueva prueba utilizando gatling y con 600 usuarios. Obtuvimos los siguientes resultados.



En este caso, el servlet dispatch sigue ocupando la mayor parte del tiempo, y los valores de la base de datos son bastante menores en comparación.

Response time	Slow traces (0)	Queries	Service calls	Thread profile		
			Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows
		select mecanico0_.id as col_0_0_ from mecanicos mecanico0_ where mecanico0_.nombre_u...	254,2	2,100	0.12	1,0
		select cita0_.id as id1_2_0_, cita0_.cliente_id as cliente_9_2_0_, cita0_.coste as c...	40,6	2,100	0.019	1,0
		select averia0_.id as id1_1_, averia0_.cita_id as cita_id9_1_, averia0_.complejidad ...	26,3	2,100	0.013	1,0
		select averia0_.id as id1_1_, averia0_.cita_id as cita_id9_1_, averia0_.complejidad ...	16,2	900	0.018	1,0
		select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?	9,5	600	0.016	1,0
		select cita0_.id as id1_2_, cita0_.cliente_id as cliente_9_2_, cita0_.coste as coste...	7,4	900	0.0082	1,0
		select username, authority from authorities where username = ?	3,1	600	0.0052	1,0

En esta captura podemos apreciar como se acusa el tiempo invertido en llamadas de comprobación de la identidad del mecánico, siendo la consulta que más tiempo consume, especialmente en comparación. Se debe en gran parte a que se realiza unas 2100 veces, teniendo un total de 600 usuarios. También se llaman unas 2100 veces al listar de citas y averías, aunque no tardan tanto, pero esto se debe a que hay 2 citas y 2 averías, por lo que en una situación con más ejemplos podría tardar más.



```

SELECT mecanico0_.id AS col_0_0_
FROM mecanicos mecanico0_
WHERE mecanico0_.nombre_usuario LIKE ?

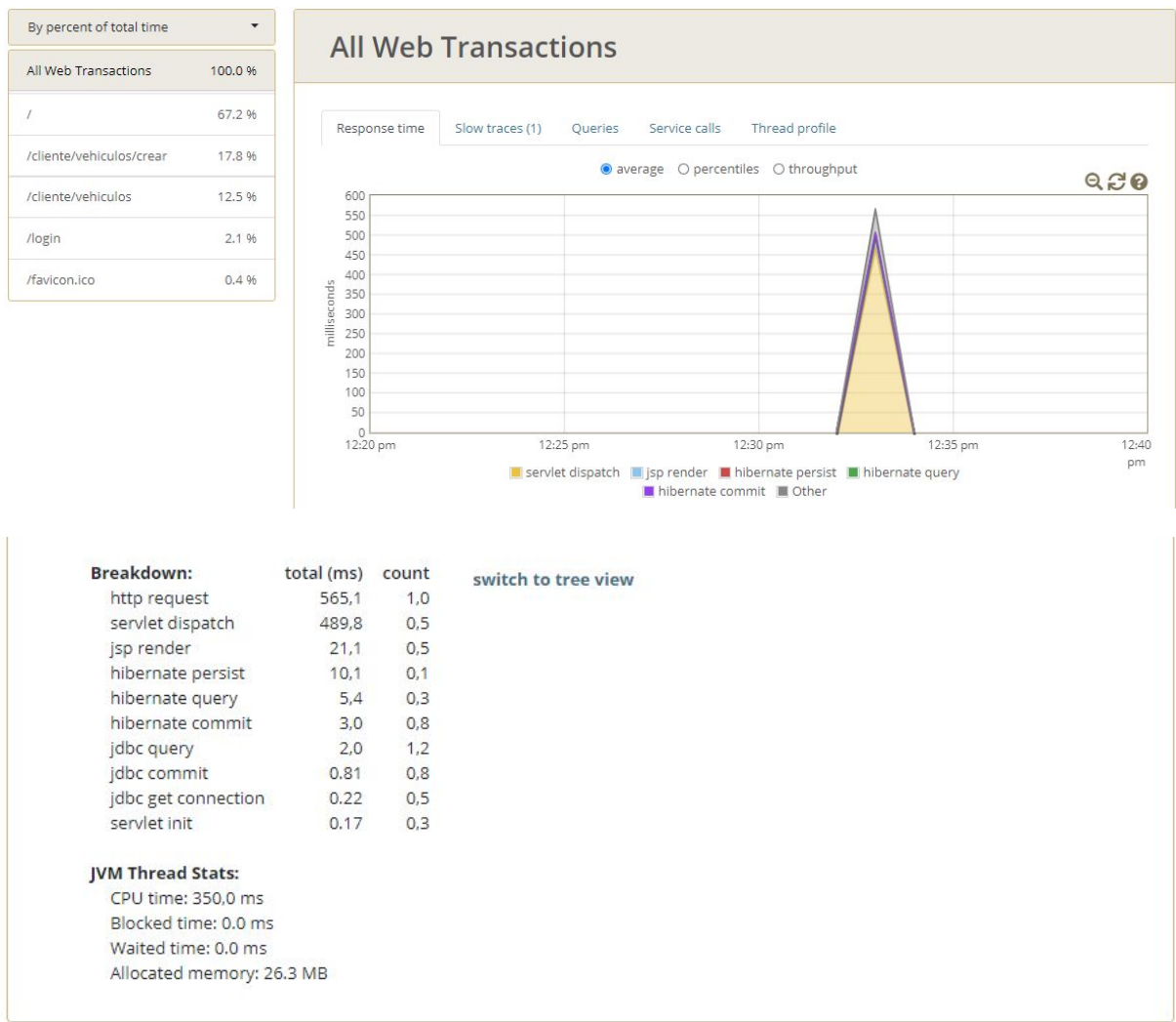
(show unformatted)

```

Esta es la consulta que ha ocupado más tiempo, en la que se comprueba la identidad del mecánico que accede a la página. En ella se comprueban los nombres de usuario de los mecánicos para traernos la id del que coincida, y así poder compararlo con el mecánico asignado a la avería que se vaya a modificar.

En conclusión, teniendo en cuenta los resultados obtenidos, pensamos que convendría indexar los usuarios por nombre, para agilizar la búsqueda del id asociado, además también sería buena idea cachear esos resultados ya que se consultan frecuentemente y no suelen cambiar a menudo. Por otro lado, en cuanto a la consulta del listado de citas y averías, quizás convendría realizar búsquedas personalizadas para no traer atributos innecesarios que luego no se utilicen.

Historia 19 Cliente añade vehículo



En estas dos imágenes se muestran los resultados de replicar el escenario de la historia en cuestión de manera manual y una sola vez. Como se puede observar en la primera gráfica, la mayor parte del tiempo es utilizado por el servlet dispatch.

En la segunda foto se observan los valores de primera gráfica más detalladamente. Seguidamente, se encuentra jsp render e hibernate persist, siendo sus tiempos de 21,1 ms y 10,1 ms respectivamente, por lo que son casi despreciables comparados con los 498,8 ms de servlet dispatch.

Así mismo, en la parte izquierda de la primera imagen podemos observar los tiempos porcentuales que ha tardado en cargar cada página. Según se nos indica en dicha tabla, un 67,2% del tiempo se ha empleado en cargar la página principal. Esta página va seguida en porcentaje de tiempo empleado de la del formulario de creación de vehículos con un 17,8%, que se corresponde con la historia que estamos analizando. Bastante cerca, en cuanto a tiempo, se encuentra la página de la historia 17, que trata de la función de que los clientes listen sus vehículos.

Response time	Slow traces (1)	Queries	Service calls	Thread profile		
			Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows
select cliente0_.id as id1_3_0_, cliente0_.apellidos as apellido2_3_0_, cliente0_.direcci...			6,2	3	2,1	1,0
select vehiculo0_.id as id1_6_, vehiculo0_.activo as activo2_6_, vehiculo0_.cliente_id as...			5,8	3	1,9	2,3
insert into vehiculos (activo, cliente_id, fecha_matriculacion, kilometraje, matricula, m...			3,0	1	3,0	1,0
select cliente0_.id as col_0_0_ from clientes cliente0_ where cliente0_.nombre_usuario li...			2,7	3	0,91	1,0
select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?			1,9	1	1,9	1,0
select username, authority from authorities where username = ?			0,63	1	0,63	1,0

En esta imagen se puede apreciar el listado de las consultas realizadas a la base de datos con sus tiempos. Según el tiempo medio, la consulta que más a tardado es la tercera, que se corresponde con la inserción de un nuevo vehículo a la base de datos y tarda una media de 3 ms. Además de esta, hay otras tres consultas que la siguen de cerca en la categoría de tiempo medio. Estas son: la primera consulta, que busca los datos de un cliente según su id, que tarda unos 2,1 ms de media; la segunda consulta, que busca todos los vehículos dados de alta activos de un cliente, que tarda 1,9 ms de media; y por último, la quinta consulta, que busca la id de un cliente según su nombre, que tarda 1,9 ms de media. Respecto al tiempo total que se tarda en realizar las consultas, la que más tiempo ha tardado es la primera, que aunque tarde unos 2,1 ms de media, al realizarse 3 veces la consulta hace que tarde en realizarse un total de 6,2 ms. Esto se debe a que esta consulta se realiza cada vez que se listan todos los vehículos de un cliente.

La siguiente captura se corresponde a la segunda consulta de la lista:


```

SELECT vehiculo0_.id AS id1_6_,
       vehiculo0_.activo AS activo2_6_,
       vehiculo0_.cliente_id AS cliente_8_6_,
       vehiculo0_.fecha_matriculacion AS fecha_ma3_6_,
       vehiculo0_.kilometraje AS kilometr4_6_,
       vehiculo0_.matricula AS matricul5_6_,
       vehiculo0_.modelo AS modelo6_6_,
       vehiculo0_.tipo_vehiculo AS tipo_veh7_6_
FROM vehiculos vehiculo0_
WHERE vehiculo0_.cliente_id = ?
      AND vehiculo0_.activo = 1

```

(show unformatted)

Esta consulta obtiene el listado de todos los vehículos activos dados de alta de un cliente específico. No se utilizan todos los atributos a la hora de listar los vehículos, pero sí a la hora de mostrar sus detalles.

La siguiente captura se corresponde a la segunda consulta de la lista:

```

SELECT cliente0_.id AS id1_3_0_,
       cliente0_.apellidos AS apellido2_3_0_,
       cliente0_.direccion AS direccio3_3_0_,
       cliente0_.dni AS dni4_3_0_,
       cliente0_.email AS email5_3_0_,
       cliente0_.nombre AS nombre6_3_0_,
       cliente0_.telefono AS telefono7_3_0_,
       cliente0_.nombre_usuario AS nombre_u8_3_0_,
       usuario1_.nombre_usuario AS nombre_u1_5_1_,
       usuario1_.contra AS contra2_5_1_,
       usuario1_.enabled AS enabled3_5_1_
FROM clientes cliente0_
     LEFT OUTER JOIN usuarios usuario1_ ON cliente0_.nombre_usuario = usuario1_.nombre_usuario
WHERE cliente0_.id = ?

```

(show unformatted)

Esta consulta obtiene todos los datos de un cliente específico. Algunos de los atributos, como la contraseña o el nombre de usuario, no se muestran ni siquiera en la función de mecánico muestra cliente.

La siguiente captura se corresponde a la cuarta consulta de la lista:

```
SELECT cliente0_.id AS col_0_0_  
FROM clientes cliente0_  
WHERE cliente0_.nombre_usuario LIKE ?  
  
(show unformatted)
```

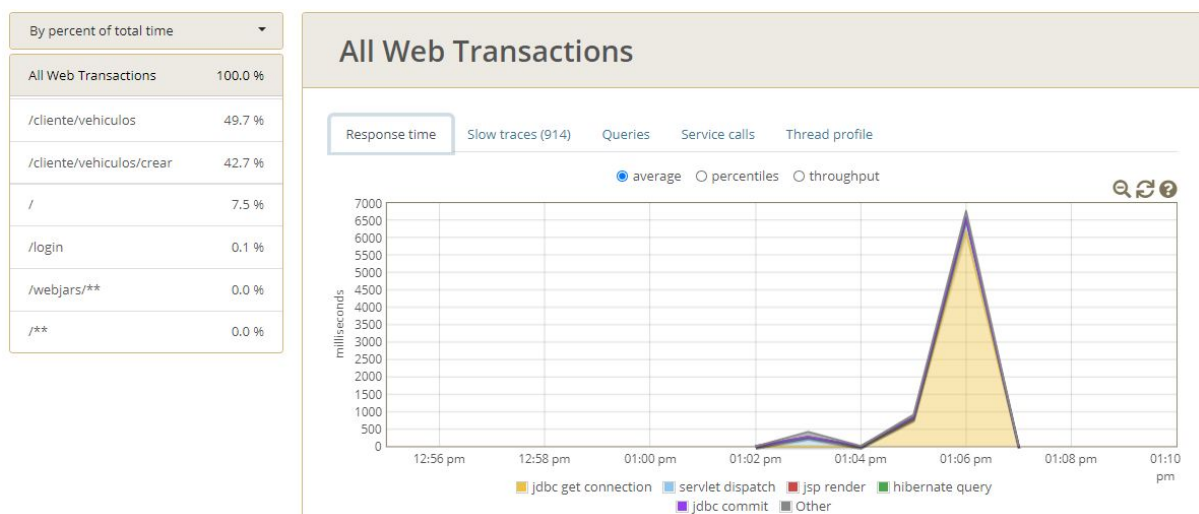
Está consulta obtiene la id de un cliente específico, según su nombre de usuario. Esta función se utiliza para obtener la id del usuario que está utilizando el servicio cada vez que es necesario realizar una comprobación con dicha id.

La siguiente captura se corresponde a la tercera consulta de la lista:

```
insert into vehiculos (activo, cliente_id,  
fecha_matriculacion, kilometraje, matricula,  
modelo, tipo_vehiculo) values (?, ?, ?, ?, ?, ?,  
?)
```

Está consulta inserta un nuevo vehiculo a la base de datos. Esta función se utiliza tras comprobar que todos los datos dentro del formulario de creación son correctos.

A continuación, se van a analizar los datos obtenidos al realizar una nueva prueba, pero esta vez utilizando Gatling e inyectar 500 usuarios.



Breakdown (Main Thread):	total (ms)	count	switch to tree view
http request	773,5	1,0	
jdbc get connection	633,6	0,4	
servlet dispatch	66,6	0,7	
jsp render	12,4	0,7	
hibernate query	11,9	0,3	
hibernate commit	10,1	0,6	
jdbc commit	9,6	0,6	
logging	7,1	0,02	
hibernate persist	3,9	0,03	
jdbc query	3,5	1,0	
show more / show all			

Breakdown (Auxiliary Thread):	total (ms)	count	switch to tree view
auxiliary thread	0,49	0,1	

JVM Thread Stats (Main Thread):

CPU time: 4,9 ms
Blocked time: 83,2 ms
Waited time: 639,9 ms
Allocated memory: 526.8 KB

JVM Thread Stats (Auxiliary Threads):

CPU time: 0.0022 ms
Blocked time: 0.0 ms
Waited time: 0.0 ms
Allocated memory: 45 bytes

En este caso, el servlet dispatch sigue ocupando la mayor parte del tiempo, aunque ha bajado su tiempo a 66,6 ms. Los tiempos de jsp render e hibernate persist también ha disminuido, a 12,4 ms y 3,9 ms respectivamente.

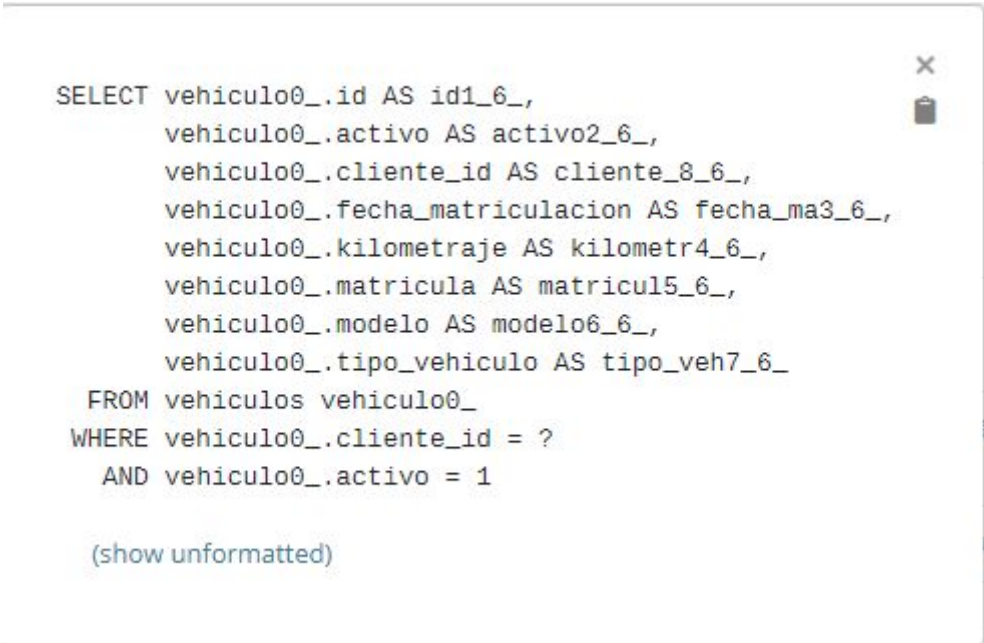
Según podemos observar en el apartado de tiempo percentual según la transacción, la página de listado de los vehículos ha pasado a ser la página en la que se pasa más tiempo con un 49,7%. Siguiéndola de cerca está el formulario de creación de vehículos, con un 42,7%. Esto hace que el tiempo porcentual en el resto de páginas sea ínfimo en comparación con las dos anteriores.

Response time	Slow traces (914)	Queries	Service calls	Thread profile		
			Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows
		select vehiculo0_.id as id1_6_, vehiculo0_.activo as activo2_6_, vehiculo0_.clien...	25.951,1	3,863	6,7	169,4
		insert into vehiculos (activo, cliente_id, fecha_matriculacion, kilometraje, matr...	14.052,0	449	31,3	1,0
		select cliente0_.id as col_0_0_ from clientes cliente0_ where cliente0_.nombre_us...	5153,8	3,863	1,3	1,0
		select cliente0_.id as id1_3_0_, cliente0_.apellidos as apellido2_3_0_, cliente0_...	4031,3	3,863	1,0	1,0
		select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?	1110,6	1,000	1,1	1,0
		select username, authority from authorities where username = ?	859,5	1,000	0.86	1,0

En esta imagen podemos volver a apreciar el listado de las consultas realizadas a la base de datos con sus tiempos, pero esta vez utilizando 500 usuarios. Según el tiempo medio, la consulta que más ha tardado vuelve a ser la que se corresponde con la inserción de un nuevo vehículo a la base de datos, tardando esta vez una media de 31,3 ms. Por otro lado, el resto de consultas tardan aproximadamente 1 ms, excepto la consulta de listado de los vehículos de un cliente, que tarda de media 6,7 ms.

Respecto al tiempo total que se tarda en realizar las consultas, la que más tiempo ha tardado es la la consulta de listado de los vehículos de un cliente, que se realiza 3863 veces la consulta, lo que hace que se dispare el tiempo que utiliza. La segunda consulta que tarda más tiempo total esta vez es la que inserta el vehículo en la base de datos, que aunque se realice una cantidad de veces bastante inferior al resto de consultas, su alto tiempo medio de realización hace que globalmente tarde más que el resto.

La siguiente captura se corresponde a la primera consulta de la lista:




```
SELECT vehiculo0_.id AS id1_6_,
       vehiculo0_.activo AS activo2_6_,
       vehiculo0_.cliente_id AS cliente_8_6_,
       vehiculo0_.fecha_matriculacion AS fecha_ma3_6_,
       vehiculo0_.kilometraje AS kilometr4_6_,
       vehiculo0_.matricula AS matricul5_6_,
       vehiculo0_.modelo AS modelo6_6_,
       vehiculo0_.tipo_vehiculo AS tipo_veh7_6_
FROM vehiculos vehiculo0_
WHERE vehiculo0_.cliente_id = ?
      AND vehiculo0_.activo = 1
```

[\(show unformatted\)](#)

Esta consulta es la que más tiempo total ha utilizado, la cual obtiene el listado de todos los vehículos de un cliente específico.

La siguiente captura se corresponde a la tercera consulta de la lista:



```
SELECT cliente0_.id AS col_0_0_
FROM clientes cliente0_
WHERE cliente0_.nombre_usuario LIKE ?
```

[\(show unformatted\)](#)

Esta consulta es la tercera que más tiempo total ha utilizado, la cual obtiene el id de un cliente según su nombre de usuario. Esta consulta se realiza cada vez que se necesita la id del usuario cliente activo para realizar comprobaciones de cualquier tipo.

La siguiente captura se corresponde a la cuarta consulta de la lista:

```

SELECT cliente0_.id AS id1_3_0_,
       cliente0_.apellidos AS apellido2_3_0_,
       cliente0_.direccion AS direccio3_3_0_,
       cliente0_.dni AS dni4_3_0_,
       cliente0_.email AS email5_3_0_,
       cliente0_.nombre AS nombre6_3_0_,
       cliente0_.telefono AS telefono7_3_0_,
       cliente0_.nombre_usuario AS nombre_u8_3_0_,
       usuario1_.nombre_usuario AS nombre_u1_5_1_,
       usuario1_.contra AS contra2_5_1_,
       usuario1_.enabled AS enabled3_5_1_
FROM clientes cliente0_
LEFT OUTER JOIN usuarios usuario1_ ON cliente0_.nombre_usuario = usuario1_.nombre_usuario
WHERE cliente0_.id = ?

(show unformatted)

```

Esta consulta es la cuarta que más tiempo total ha utilizado, la cual obtiene todos los atributos de un usuario según su id. Esta consulta se realiza cada vez que se necesita utilizar los atributos de un cliente.

La siguiente captura se corresponde a la cuarta consulta de la lista:

```

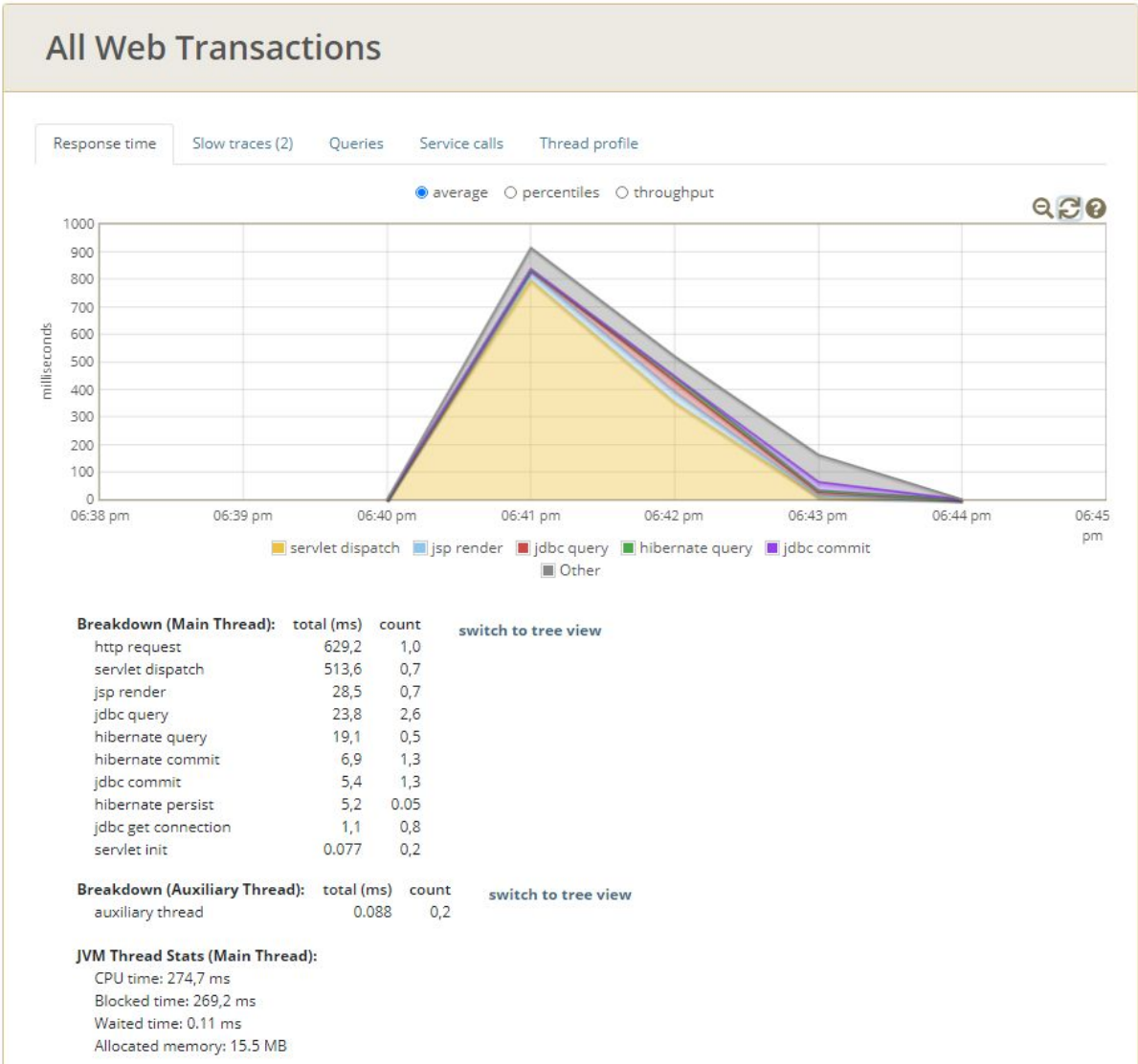
insert into vehiculos (activo, cliente_id,
                      fecha_matriculacion, kilometraje, matricula,
                      modelo, tipo_vehiculo) values (?, ?, ?, ?, ?, ?,
                      ?)

```

Esta es la segunda consulta que más tiempo total ha utilizado, la cual inserta el nuevo vehículo a la base de datos. Esta consulta se utiliza tras comprobar que todos los datos dentro del formulario de creación son correctos.

En conclusión, teniendo en cuenta los resultados obtenidos, pensamos que convendría cachear los datos que se consulten de manera frecuente y no suelen ser modificados, como puede ser la lista de vehículos o los datos de los clientes, ya que estas solamente se modifican al crear, modificar o dar de baja un vehículo, y respecto a los datos del cliente, actualmente no existe ninguna manera de actualizar sus datos desde dentro de la aplicación, por lo que no sería necesario cambiar sus datos hasta que existiera la posibilidad. Además, respecto a los datos del usuario, convendría cambiar la consulta para que solamente obtenga los campos necesarios, ya que para mostrar un cliente no se necesitaría su contraseña o nombre de usuario a no ser que fuera el propio cliente el que accede.

Historia 3 Cliente crea cita



En esta imagen se muestran los resultados de replicar el escenario de la historia en cuestión de manera manual y una sola vez. En la gráfica se ve que la mayor parte del tiempo se fue en el servlet dispatch.

Seguidamente, se encuentra jsp render y jdbc query, que hacen alusión a la carga de las páginas web y a las operaciones sobre la base de datos, respectivamente. Estas dos operaciones han tardado unos tiempos despreciables comparados con los anteriores, el servlet dispatch ha tardado 513.6 ms, mientras que jsp render ha tardado 28.5 ms y jdbc query ha tardado 23.8 ms, por lo que servlet dispatch ha tardado casi 500 ms más.

By percent of total time	
All Web Transactions	100.0 %
/	58.0 %
/cliente/citas/pedir	18.9 %
/cliente/citas	13.7 %
/cliente/citas/vehiculo	5.9 %
/login	3.1 %
/cliente/citas/	0.3 %

En esta captura se puede observar como la mayor parte del tiempo, un 58%, se va en cargar la página principal. Esta página va seguida en porcentaje de tiempo de la página cliente pide cita (donde el cliente crea la cita), con un 18.9%, que se corresponde con la página de la historia 3, qué es la que nos ocupa. La siguiente página trata de la página de cliente lista citas, con un 13.7%, que se corresponde con la página de la historia 1.

All Web Transactions

Response time Slow traces (2) **Queries** Service calls Thread profile

	Total time ↓ (ms)	Total count	Avg time (ms)	Avg rows
<code>select cliente@_.id as col_@_ from clientes cliente@_ where cliente@_.nombre_usuario...</code>	107,9	10	10,8	1,0
<code>select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?</code>	94,2	2	47,1	1,0
<code>select vehiculo@_.id as id1_6_, vehiculo@_.activo as activo2_6_, vehiculo@_.cliente_id...</code>	85,7	6	14,3	2,0
<code>select vehiculo@_.id as id1_6_@_, vehiculo@_.activo as activo2_6_@_, vehiculo@_.client...</code>	45,4	7	6,5	1,0
<code>select cita@_.id as id1_2_, cita@_.cliente_id as cliente_@_2_, cita@_.coste as coste2_...</code>	40,1	4	10,0	2,3
<code>select mecanico@_.id as id1_4_@_, mecanico@_.apellidos as apellido2_4_@_, mecanico@_.d...</code>	31,6	4	7,9	1,0
<code>insert into citas (cliente_id, coste, descripcion, urgente, estado_cita, fecha_cita, m...</code>	28,4	1	28,4	1,0
<code>select cliente@_.id as id1_3_@_, cliente@_.apellidos as apellido2_3_@_, cliente@_.dire...</code>	15,9	11	1,4	1,0
<code>select vehiculo@_.id as id1_6_, vehiculo@_.activo as activo2_6_, vehiculo@_.cliente_id...</code>	2,2	2	1,1	1,0
<code>select username, authority from authorities where username = ?</code>	1,6	2	0,82	1,0

En esta imagen se puede apreciar el listado de las consultas realizadas a la base de datos. Por tiempo medio, la consulta que más ha tardado es la segunda, que se corresponde con la comprobación de la identidad del usuario. Sin embargo la que ha consumido más tiempo en total se trata de la primera, que es la que se realiza para asegurar la identidad del cliente que está realizando la creación. La consulta no toma mucho tiempo, pero se realiza muchas veces, 10, debido a que cada vez que el cliente accede a una página, se comprueba su identidad.

La siguiente captura se corresponde a la tercera consulta de la lista:

```
SELECT vehiculo@_.id AS id1_6_,
       vehiculo@_.activo AS activo2_6_,
       vehiculo@_.cliente_id AS cliente_8_6_,
       vehiculo@_.fecha_matriculacion AS fecha_ma3_6_,
       vehiculo@_.kilometraje AS kilometr4_6_,
       vehiculo@_.matricula AS matricul5_6_,
       vehiculo@_.modelo AS modelo6_6_,
       vehiculo@_.tipo_vehiculo AS tipo_veh7_6_
FROM vehiculos vehiculo@_
WHERE vehiculo@_.cliente_id = ?
AND vehiculo@_.activo = 1
```

(show unformatted)

Esta consulta corresponde con la tercera consulta que más tiempo total ha consumido.

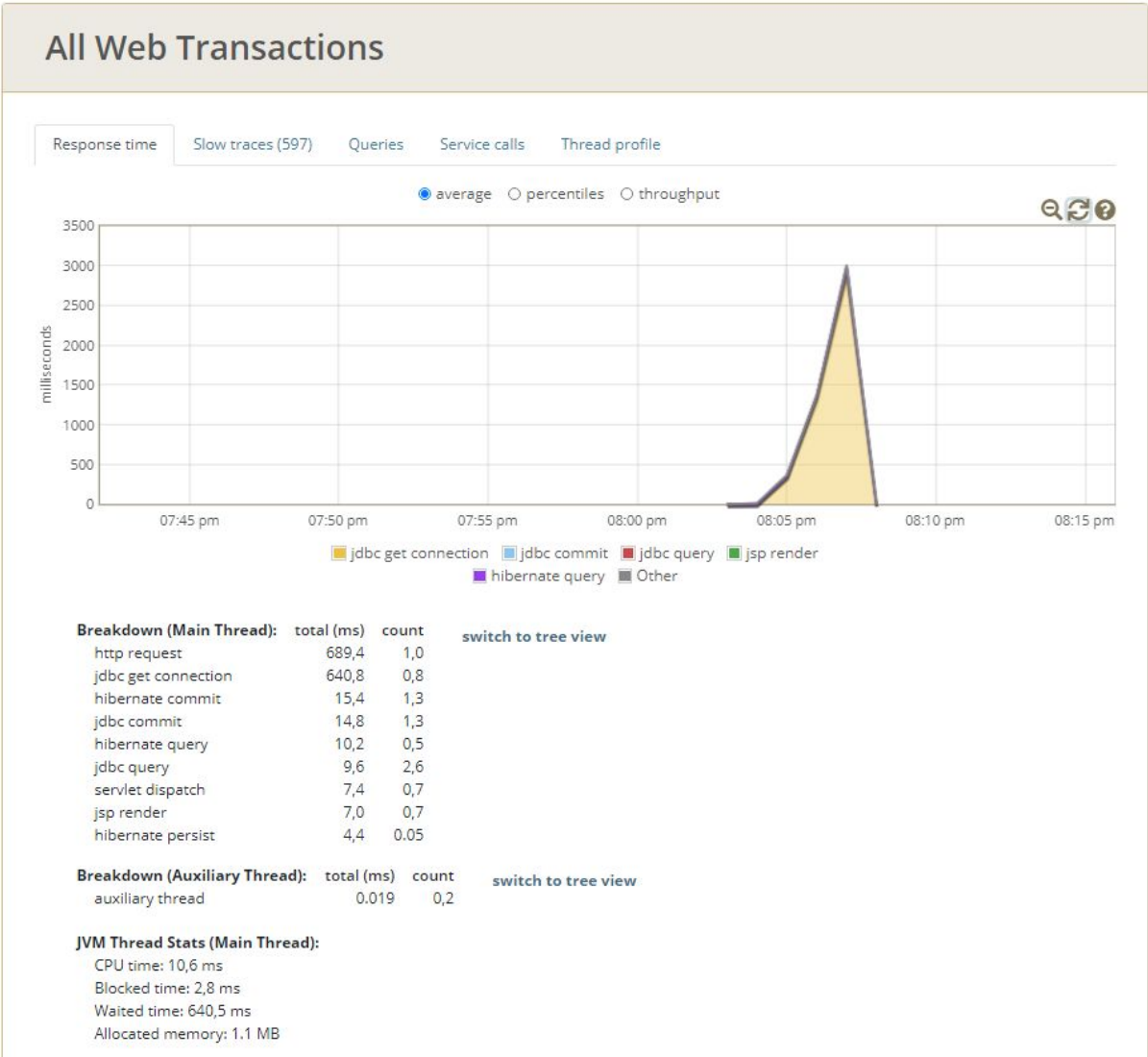
En esta consulta se devuelve los vehículos del cliente que estén en la base de datos, es decir, que se hayan creado con anterioridad. Esto es necesario ya que al crear una cita tienes que asignar obligatoriamente uno de tus vehículos.

La siguiente captura se corresponde a la séptima consulta de la lista:

```
insert into citas (cliente_id, coste, descripcion,
urgente, estado_cita, fecha_cita, mecanico_id,
tiempo, tipo_cita, vehiculo_id) values (?, ?, ?,
?, ?, ?, ?, ?, ?, ?)
```

Esta consulta corresponde con la séptima consulta que más tiempo en total ha consumido. En esta consulta es en la que se crea la cita.

Posteriormente, realizamos una nueva prueba utilizando gatling y con 600 usuarios. Obtuvimos los siguientes resultados.



En este caso, el jdbc get connection es que ocupa la mayor parte del tiempo, tardando 640.8 ms. El tiempo de hibernate commit ha disminuido a 15,4 ms, el tiempo de jdbc commit ha aumentado 14.8 y los tiempos de hibernate query y jdbc query han disminuido a 10.2 y a 9.6 respectivamente.

By percent of total time ▼	
All Web Transactions	100.0 %
/cliente/citas/pedir	54.3 %
/cliente/citas	24.2 %
/cliente/citas/	20.2 %
/	0.6 %
/login	0.6 %
/cliente/citas/vehiculo	0.1 %

Según podemos observar en el apartado de tiempo percentual según la transacción, la página de pedir la cita sigue siendo la página en la que se pasa más tiempo con un 54.3%. La segunda página en la que pasa más tiempo es la de listar las citas con un 24.2%. Siguiéndola de cerca está la página de listar citas cuando ya has creado la cita (Es la misma solo que cambia la url). El tiempo porcentual en el resto de páginas es ínfimo en comparación con las tres anteriores.

All Web Transactions

Response time Slow traces (597) **Queries** Service calls Thread profile

	Total time ▼ (ms)	Total count	Avg time (ms)	Avg rows
insert into citas (cliente_id, coste, descripcion, urgente, estado_cita, fecha_ci...	49.773,3	600	83,0	1,0
select cita0_.id as id1_2_, cita0_.cliente_id as cliente_9_2_, cita0_.coste as co...	34.915,4	2,400	14,5	793,3
select cliente0_.id as id1_3_0_, cliente0_.apellidos as apellido2_3_0_, cliente0_...	6508,1	6,600	0.99	1,0
select vehiculo0_.id as id1_6_0_, vehiculo0_.activo as activo2_6_0_, vehiculo0_.c...	4749,5	4,200	1,1	1,0
select cliente0_.id as col_0_0_ from clientes cliente0_ where cliente0_.nombre_us...	4175,9	6,000	0.70	1,0
select mecanico0_.id as id1_4_0_, mecanico0_.apellidos as apellido2_4_0_, mecanic...	2461,3	2,400	1,0	1,0
select nombre_usuario,contra,enabled from usuarios where nombre_usuario=?	2344,0	1,200	2,0	1,0
select vehiculo0_.id as id1_6_, vehiculo0_.activo as activo2_6_, vehiculo0_.clien...	2332,4	3,600	0.65	2,0
select username, authority from authorities where username = ?	1176,7	1,200	0.98	1,0
select vehiculo0_.id as id1_6_, vehiculo0_.activo as activo2_6_, vehiculo0_.clien...	788,6	1,200	0.66	1,0

En esta imagen podemos volver a apreciar el listado de las consultas realizadas a la base de datos con sus tiempos, pero esta vez utilizando 600 usuarios. Según el tiempo medio, la consulta que más ha tardado vuelve a ser la primera, que se corresponde con la inserción de una nueva cita a la base de datos, tardando esta vez una media de 83.0 ms. Por otro lado, el resto de consultas tardan aproximadamente 1 ms, excepto la consulta de listado de las citas de un cliente, que tarda de media 14.5 ms.

Respecto al tiempo total que se tarda en realizar las consultas, la que más tiempo ha tardado es la que crea la cita y que se ha realizado 600 veces, lo que hace que aumente bastante el tiempo que utiliza. La segunda consulta que tarda más tiempo total es la que lista las citas del cliente, que se realiza 2.400 veces bastante.

La siguiente captura se corresponde a la primera consulta de la lista:

```
insert into citas (cliente_id, coste, descripcion,
urgente, estado_cita, fecha_cita, mecanico_id,
tiempo, tipo_cita, vehiculo_id) values (?, ?, ?,
?, ?, ?, ?, ?, ?, ?)
```

Esta consulta es la que más tiempo total consume y la que más tiempo de media consume. Es la que se encarga de crear la cita en la base de datos.

La siguiente captura se corresponde a la segunda consulta de la lista:

```
SELECT cita0_.id AS id1_2_,
       cita0_.cliente_id AS cliente_9_2_,
       cita0_.coste AS coste2_2_,
       cita0_.descripcion AS descripc3_2_,
       cita0_.urgente AS urgente4_2_,
       cita0_.estado_cita AS estado_c5_2_,
       cita0_.fecha_cita AS fecha_ci6_2_,
       cita0_.mecanico_id AS mecanic10_2_,
       cita0_.tiempo AS tiempo7_2_,
       cita0_.tipo_cita AS tipo_cit8_2_,
       cita0_.vehiculo_id AS vehicul11_2_
FROM citas cita0_
WHERE cita0_.cliente_id = ?
```

(show unformatted)

Esta consulta se encarga de listar todas las citas del cliente. Es la segunda que más tiempo total consume y que más media de tiempo consume.