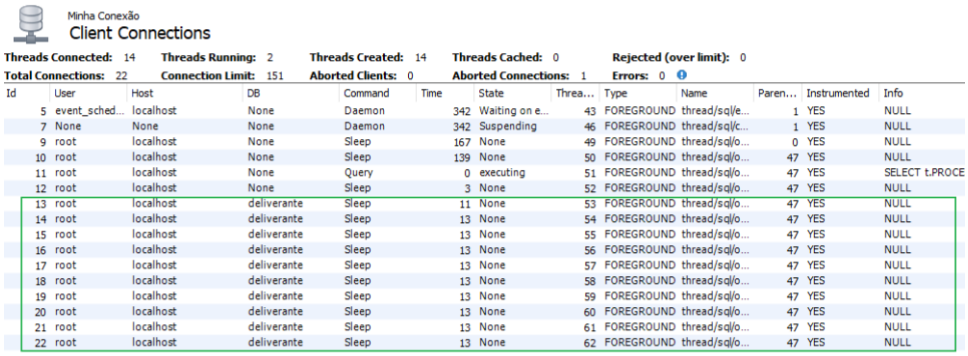
Pool de conexões – Hikari

Ao iniciar a aplicação **Spring** dá pra olhar no SGDB as conexões que são feitas com ele, essas conexões ficam ociosas até serem chamadas:



Minha Conexão

Client Connections

Threads Connected: 14 **Threads Running:** 2 **Threads Created:** 14 **Threads Cached:** 0 **Rejected (over limit):** 0

Total Connections: 22 **Connection Limit:** 151 **Aborted Clients:** 0 **Aborted Connections:** 1 **Errors:** 0

Id	User	Host	DB	Command	Time	State	Threa...	Type	Name	Paren...	Instrumented	Info
5	event_sched...	localhost	None	Daemon	342	Waiting on e...	43	FOREGROUND	thread/sql/e...	1	YES	NULL
7	None	None	None	Daemon	342	Suspending	46	FOREGROUND	thread/sql/c...	1	YES	NULL
9	root	localhost	None	Sleep	167	None	49	FOREGROUND	thread/sql/o...	0	YES	NULL
10	root	localhost	None	Sleep	139	None	50	FOREGROUND	thread/sql/o...	47	YES	NULL
11	root	localhost	None	Query	0	executing	51	FOREGROUND	thread/sql/o...	47	YES	SELECT t.PROCE
12	root	localhost	None	Sleep	3	None	52	FOREGROUND	thread/sql/o...	47	YES	NULL
13	root	localhost	deliverante	Sleep	11	None	53	FOREGROUND	thread/sql/o...	47	YES	NULL
14	root	localhost	deliverante	Sleep	13	None	54	FOREGROUND	thread/sql/o...	47	YES	NULL
15	root	localhost	deliverante	Sleep	13	None	55	FOREGROUND	thread/sql/o...	47	YES	NULL
16	root	localhost	deliverante	Sleep	13	None	56	FOREGROUND	thread/sql/o...	47	YES	NULL
17	root	localhost	deliverante	Sleep	13	None	57	FOREGROUND	thread/sql/o...	47	YES	NULL
18	root	localhost	deliverante	Sleep	13	None	58	FOREGROUND	thread/sql/o...	47	YES	NULL
19	root	localhost	deliverante	Sleep	13	None	59	FOREGROUND	thread/sql/o...	47	YES	NULL
20	root	localhost	deliverante	Sleep	13	None	60	FOREGROUND	thread/sql/o...	47	YES	NULL
21	root	localhost	deliverante	Sleep	13	None	61	FOREGROUND	thread/sql/o...	47	YES	NULL
22	root	localhost	deliverante	Sleep	13	None	62	FOREGROUND	thread/sql/o...	47	YES	NULL

Essa quantidade de ***conexões do pool não é explicitamente exposta***, quem decidiu essa quantidade foi a própria aplicação através da dependência de starter do Data JPA:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Starter é um conjunto de dependências, entrando nele tem a **dependência do JDBC** e no **JDBC** tem a dependência do **HikariCP** que é o pool de conexões padrão de projetos **Spring:**

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
  <version>2.6.2</version>
  <scope>compile</scope>
</dependency>
    <dependency>
      <groupId>com.zaxxer</groupId>
      <artifactId>HikariCP</artifactId>
      <version>4.0.3</version>
      <scope>compile</scope>
    </dependency>
```

==========

Configurando o pool

Por mais que essas configurações sejam padrões, elas ainda sim podem ser alteradas. O **número máximo** e mínimo de conexões do pool do Hikari podem ser configuradas pelo application propeties ***(dessa maneira que está eu digo que ele vai ter no máximo 5 conexões e vai começar com 3)***:

```
spring.datasource.hikari.maximum-pool-size = 5
spring.datasource.hikari.minimum-idle = 3
```

Se por acaso eu precisei de conexões a mais que meu mínimo definido, eu ***posso configurar também o que vai acontecer com essas conexões excedentes***, como por exemplo um **timeout recebendo milisegundos** *(no mínimo 10000):*

```
spring.datasource.hikari.maximum-pool-size =5
spring.datasource.hikari.minimum-idle =3
spring.datasource.hikari.minimum-idle-timeout =10000
```

Para fazer testes no pool dá pra usar a ferramenta **apache http server**, ele vem com um cara chamado **ab(apache bench)** que permite simular requisições através do terminal.

Um comando para se usar é **"ab -n quantidade de requisições -c de quantos em quantos vai ser passado local da api"** = *"ab -n 60 -c 10 localhost:8080/endpoint"* .