Actor Bug: Communication - Response

Actors perform work and change state through their responses to different types of messages. This category is for the errors that occur as a result of improper responses to different communication based operations.

Lagom had issues with connecting to a Cassandra Data Base when connection attempts were performed before the server had initialized yet. These connection attempts would be formed into prepared statements that could be cached. The problem was when the data base isn't running, these statements would fail and by caching the failed statement, the service would never try to reconnect. The bug was solved by adding a case to clear the prepared statement from the cache when one failed.

The code below shows that originally prepared statements would be stored in an array of futures using flatmap with no regards to their outcomes. This was modified with a filter that would collect the prepared statements and then check them for failures. If a failure was found, they were removed for the collection of statements. This would prevent failed statements to be used.

Link: https://github.com/lagom/lagom/issues/91

Code:

lagom/internal/persistence/cassandra/CassandraSessionImpl.scala

```
   private val computePreparedStatement = new JFunction[String, Future[PreparedStatement]] {
     override def apply(key: String): Future[PreparedStatement] =
-      underlyingSession().flatMap(_.prepareAsync(key).asScala)
+      underlyingSession().flatMap { s =>
+        val prepared = s.prepareAsync(key).asScala
+        prepared.onFailure {
+          case _ =>
+            // this is async, i.e. we are not updating the map from the compute function
+            preparedStatements.remove(key)
+        }
+        prepared
+      }
   }
```