**Sprint Review**

1. Sprint (previsto e realizado)

O planejamento foi seguido conforme estipulado até o 2º dia. No entanto, não foi possível finalizar a solução de armazenamento no dia 2, por problemas de infraestrutura e conexão entre Java e Cassandra, o que causou atraso no desenvolvimento da API.

Assim, a API foi desenvolvida e testada no dia 5 e o desenvolvimento do .bat (mudança do .sh por estar trabalhando com Windows) ocorreu no dia 6.

1. Tecnologias utilizadas

SO: Windows 10

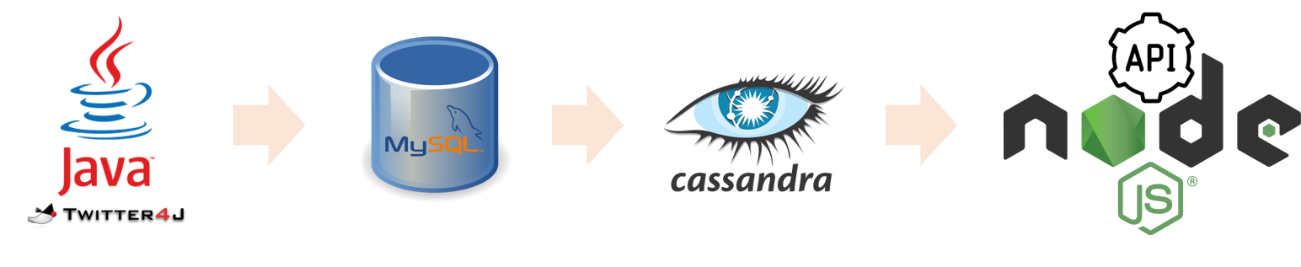
Linguagens: Java, Node.js, SQL e CQL

IDE’s: Eclipse Oxygen.3a, Sublime Text 3

Programas: MySQL Community Server 5.7.21, Cassandra 3.0.9

API: twitter4j 4.0.6

1. Diagrama da Solução



Foi utilizado o Java 8 para desenvolvimento da aplicação que coleta as últimas postagens do Twitter. Em conjunto foi utilizada a API twitter4j, para extrair os dados requeridos.

Como solução de armazenamento para processamentos, foi utilizado o MySQL, por ter um bom desempenho com a quantidade de dados manipulados neste exercício e por ser um SGBD leve e confiável. Também foi considerado o Hadoop/Hive, porém pela pequena quantidade de dados, a performance das queries ficaria comprometida; além disso, houve problemas com a instalação do ambiente, o que impossibilitou um teste com essa solução.

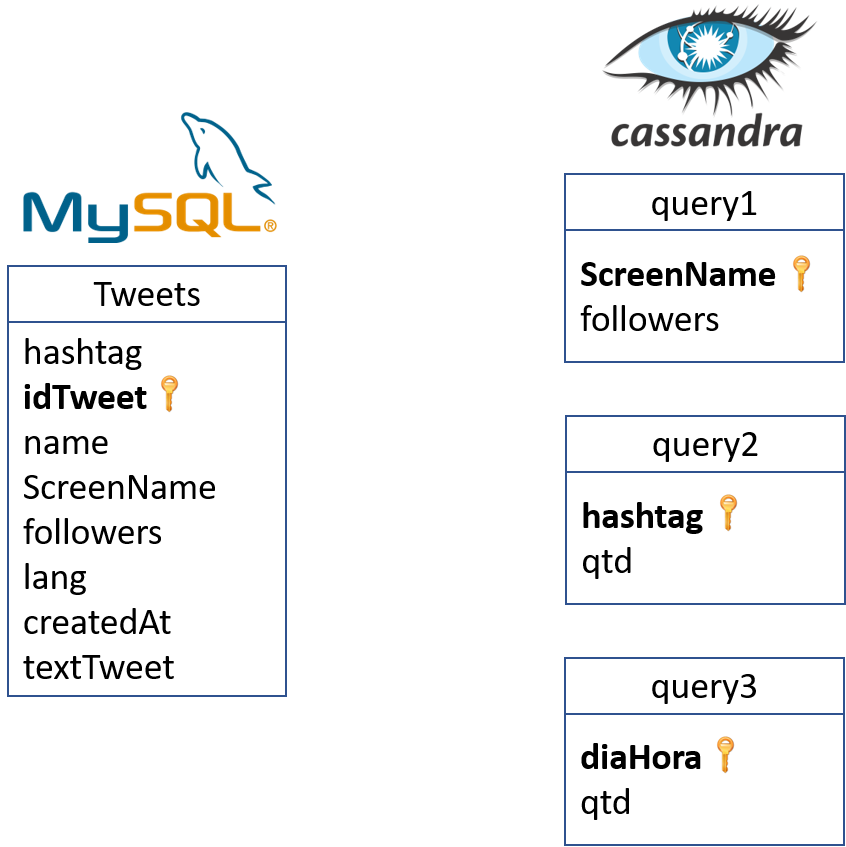
Como solução de armazenamento para consultas, foi utilizado o Cassandra, por ter alta disponibilidade, ser tolerante a falhas, ser orientado a query e por ser possível escalar horizontalmente de acordo com a necessidade. O Cassandra também facilita e aumenta a velocidade de resposta para uma API. Foi inserido no Cassandra o resultado final das queries solicitadas no exercício.

Por fim, como camada final, foi utilizado o node.js para construção da API Rest.

Observação: a conexão entre a aplicação Java e o Cassandra não foi realizada, pois o comando copy não funciona chamado a partir da aplicação. Como sugestão de melhoria futura, seria interessante testar o cassandra-loader. Como alternativa, também foi desenvolvida uma API consultando direto do MySQL.

1. Modelagem de dados

No MySQL foi utilizada uma tabela e no Cassandra 3 tabelas, conforme queries finais.



1. Referências utilizadas

Foram utilizados diversos sites para auxiliar no desenvolvimento do exercício, segue abaixo alguns deles:

<http://jkoder.com/twitter-search-api-get-tweets-and-tweets-count-hashtag-java-client-twitter4j/>

<https://examples.javacodegeeks.com/core-java/sql/import-csv-file-to-mysql-table-java-example/>

<https://stackoverflow.com/questions/27348315/how-to-create-a-java-application-to-read-twitter-feeds/40691881#40691881>

<https://stackoverflow.com/questions/5455555/twitter4j-how-to-get-tweet-text-with-anchor-to-hash-tag-and-urls?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa>

<http://www.luiztools.com.br/post/como-usar-nodejs-mysql/>

<https://academy.datastax.com/resources/getting-started-apache-cassandra-and-nodejs>

<http://www.julianomarcon.com.br/post/20160322-node-e-cassandra-parte3-criando-uma-api-simples/>

<https://docs.datastax.com/en/cql/3.3/cql/cql_reference/cqlshCopy.html>