**Grupo 2**

**Integrantes**

* Amanda De Jesus Garcia
* Alter Vitor Soares Rodrigues
* Gabriela Monteiro Xavier
* Henrique Sales Kouyoumdjian
* Thiago Ruiz Aniceto

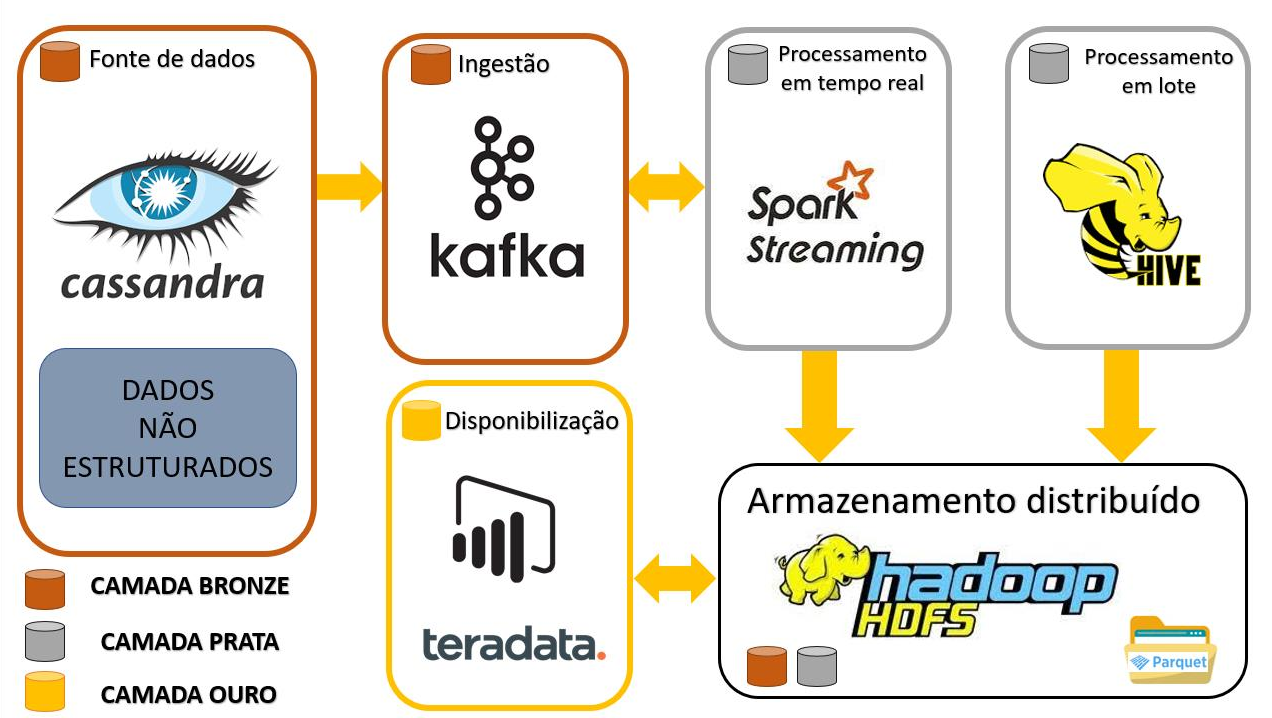
1. **Definição de um problema (Podem escolher “a gosto”)**

* **Contexto**: Somos uma empresa de e-commerce que possui vendas de vários produtos em diversas regiões no país e não tem um controle sobre os dados que são gerados de seus compradores.
* **Problema**: Identificar os melhores compradores por região e por produto em tempo real. Isso significa que a empresa precisa ter um controle eficiente sobre os dados, com o objetivo de identificar e acompanhar em tempo real os clientes que são considerados os melhores compradores em cada região e em relação a cada produto específico.

1. **Definição ingestão em dados (Local da origem, pode já existir ou pode ser gerado, qual ferramenta que vão usar)**

* **Apache Cassandra:** As transações de compra e venda são rastreadas e registradas no Apache Cassandra, que atua como **fonte de dados** do processo.
* **Kafka**: responsável por receber os dados do Cassandra e fazer a **ingestão** em tempo real. Ele atua como um intermediário entre as fontes externas e o processamento stream.
* **Spark Streaming**: **processa** os dados em tempo real da fonte Kafka e realiza as operações de transformação de dados, como filtragem, mapeamento e agregação por comprador, região e etc... Após esse processamento, armazenamos os resultados no HDFS para fins de armazenamento seguro e durável.
* **HDFS**: O Spark Streaming envia os dados para o HDFS , que é um sistema de arquivos distribuído projetado para **armazenar** grandes volumes de dados, garantindo que os dados dos compradores sejam armazenados de forma segura. Utilizaremos o formato **Parquet.**
* **HIVE**: uma camada de consulta e análise de dados construída sobre o Hadoop que vai permitir **escrever** consultas para **extrair** informações dos dados armazenados no HDFS sobre os compradores.

1. **Definição arquitetura (desenho no mínimo 2 partes do todo ingestão e armazenamento como será)**

****

1. **Subir no GIT**

**5) Definição De Uma Camada De Processamento**

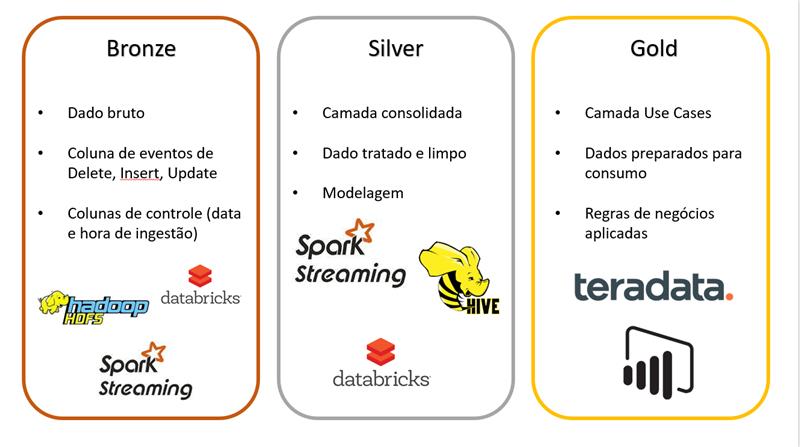
* Camada Ingestão (Bronze)
* Dado bruto + Colunas de controle (data e hora de ingestão)
* Databricks/Spark

**6) Definição De Uma Camada De Exploração**

* Camada Consolidada (Silver)
* Modelagem dos dados, dados tratados e limpos
* Hive e/ou Databricks/Spark

**7) Definição De Uma Camada De Disponibilização**

* Camada Use Cases (Gold)
* Dados preparados para consumo por parte das áreas de negócios, com as regras de negócios aplicadas nos dados
* Power bi e Teradata



**8) Pesquisa De Ferramentas (Ex Qual A Motivação De Usar As Ferramentas Escolhidas) - 1**

* **Apache Cassandra:** utilizaremos o Cassandra por oferecer algumas vantagens:
  + Alta **disponibilidade**;
  + Performance;
  + Extremamente tolerante a falhas;
  + **Escalabilidade** linear: se o banco atende 100K de requisições, para atender 200K basta dobrar a infraestrutura;
  + Sem nenhum ponto único de falha;
  + Altamente **distribuído**;
  + Suporta N datacenters nativamente.
* **Kafka**: Utilizaremos o Kafka pois ele é amplamente utilizado para ingestão de dados em tempo real,se destacando por sua **escalabilidade**, alta **disponibilidade e** baixa **latência**.
* **Spark Streaming**: é uma extensão da API principal do Spark que dá suporte ao processamento de dados em tempo real com **tolerância a falhas,** alto **desempenho** e de forma **escalável.** Nós escolhemos o Spark por conta da **velocidade**, **facilidade** para usar (Java, Scala, Python, R e SQL)
* **HDFS**: o HDFS oferece várias vantagens significativas. Destacamos 4 que serão úteis no desenvolvimento da nossa arquitetura e que foram determinantes para nossa escolha:
  + Armazenamento **distribuído** e **econômico**
  + Armazenamento **seguro**
  + **Escalável**, podendo lidar com grandes volumes de dados adicionando mais servidores ao cluster;
  + **Tolerante a falhas**, lidando muito bem com a perda de nós individuais sem interromper as operações.
* **HIVE**: O Apache HIVE será usado para consultar e fazer análises dos nossos dados estruturados/semi-estruturados. Os motivos que fizeram nosso grupo utilizá-lo foi: Escalabilidade, Processamento distribuído, Integração com o ecossistema Hadoop, suporte a formato de dados variados e a comunidade ser ativa e dar suporte.