

FIAP

PROJETO INTEGRADO

Arquitetura de Dados e Fluxo NiFi | QuantumFinance

YAGO ANGELINI CANDIDO | RM 354173

RAFAEL FERNANDO BATISTA GOMES | RM 354009

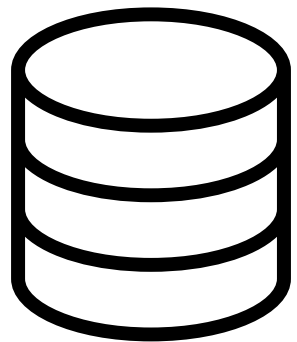
QUANTUMFINANCE



ARQUITETURA DE DADOS (PERSISTÊNCIA POLIGLOTA)

QUANTUMFINANCE

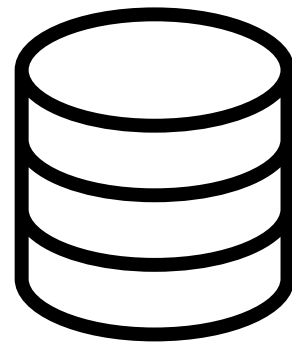
BD CHAVE-VALOR



Dados de sessão do usuário nos aplicativos e site, auditoria



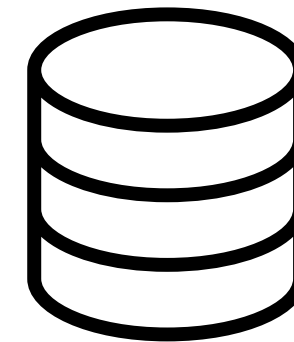
BD RELACIONAL



Transações bancárias, Saldo de Investimentos, Integrações com sistemas externos, Dados de Clientes, Produtos (investimentos)



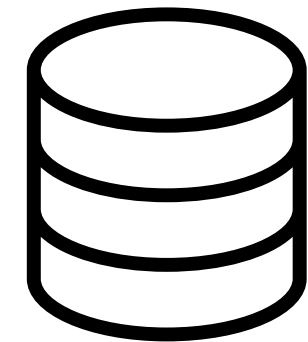
BD DOCUMENTOS



Anexos dos clientes (extratos, documentos, arquivos), logs



BD GRAFOS



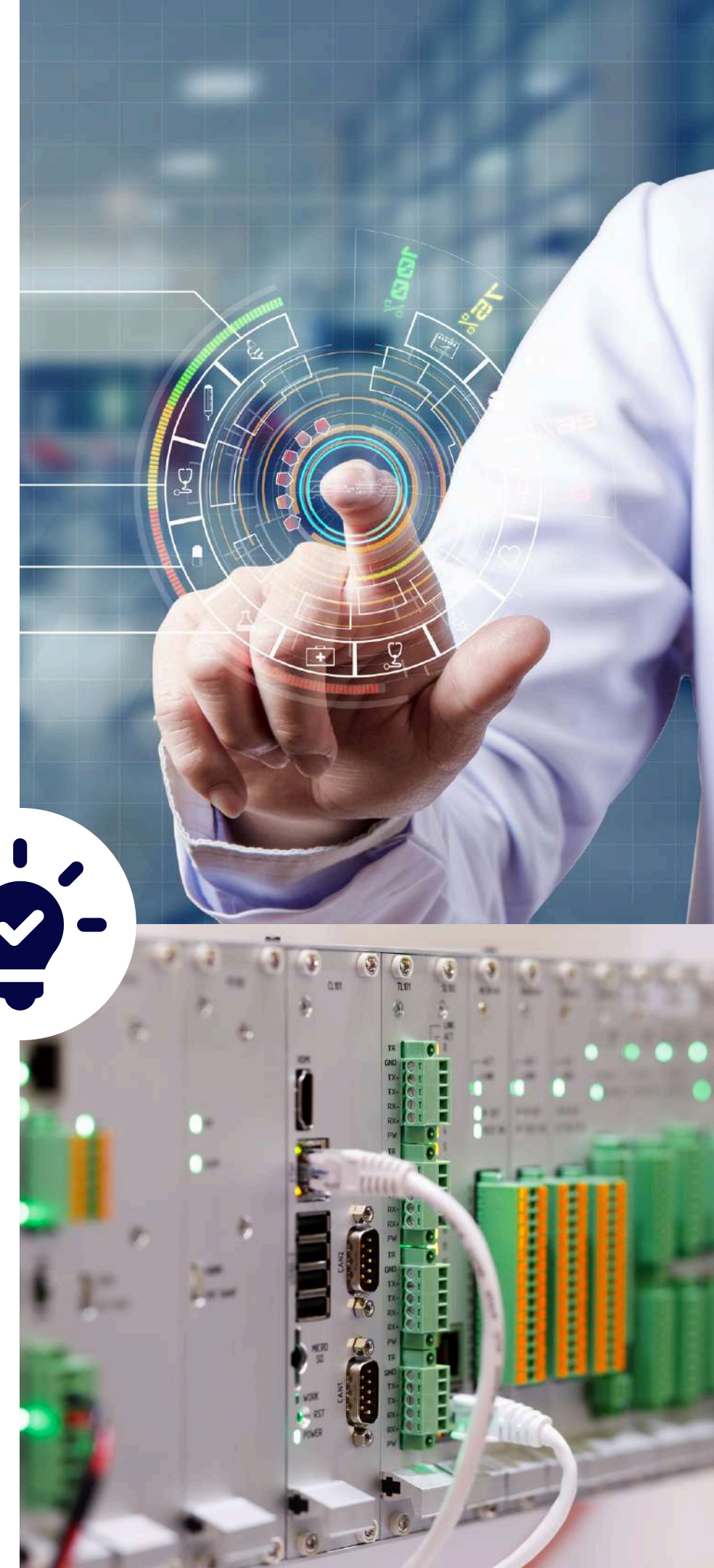
Grafo social do cliente para recomendação de produtos, redes de fraude e conformidade de transações



JUSTIFICATIVAS DA ARQUITETURA #1

Escolhemos o Redis para armazenar dados de sessão do usuário nos aplicativos e no site porque acreditamos que a sua alta performance e baixa latência nos garantem acesso rápido e eficiente às informações de sessão, melhorando significativamente a experiência do usuário, e permitindo um carregamento ágil e interações fluídas nas plataformas da QuantumFinance.

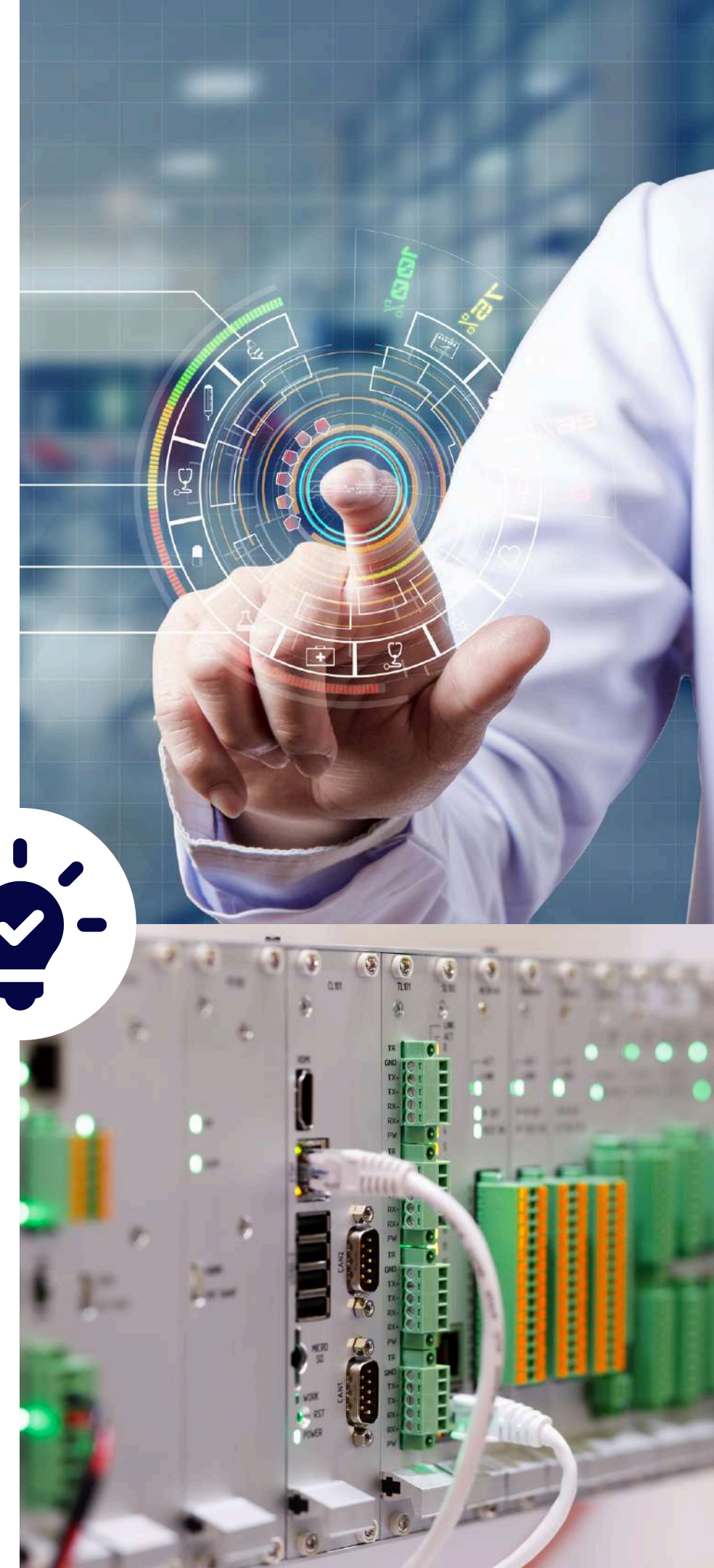
Já para gerenciar transações bancárias, investimentos, integrações com sistemas externos, dados de clientes, produtos, escolhemos o MySQL, pois oferece suporte a transações ACID, garantindo a integridade e consistência dos dados. Sua estrutura relacional nos permitirá gerir dados estruturados e realizar consultas complexas, o que é fundamental para operações bancárias seguras e eficientes.



JUSTIFICATIVAS DA ARQUITETURA #2

A fim de armazenar anexos dos clientes, como extratos, documentos e arquivos, devido à sua flexibilidade para lidar com dados semi-estruturados e documentos com estruturas variáveis, optamos pelo MongoDB. Ele nos permite armazenar e recuperar facilmente diferentes tipos de anexos, garantindo que possamos atender às diversas necessidades documentais dos nossos clientes de maneira eficiente.

Por fim, acreditamos que Neo4j é a melhor opção para gerenciar o grafo social dos clientes, recomendação de produtos, redes de fraude e conformidade de transações, porque sua arquitetura baseada em grafos facilita a modelagem e a consulta de conexões complexas entre diferentes entidades. Isso nos permite identificar rapidamente padrões e interações entre clientes, produtos e transações, melhorando a eficácia dos nossos serviços.



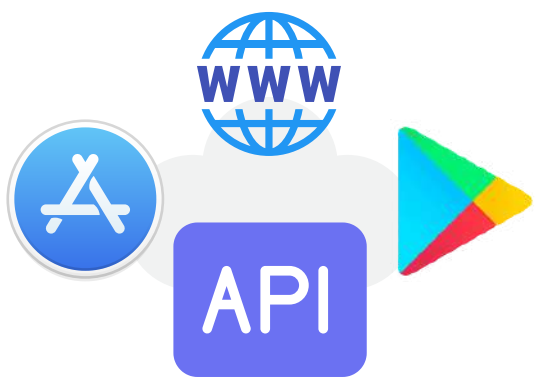
Fontes de Dados

EXEMPLO DE FRAMEWORK DE DADOS

Operacionalização dos Dados



Base de Dados
RELACIONAL / TRANSACIONAL



APIs & Apps
INTERFACES DO CLIENTE



Bases de Dados
NÃO RELACIONAIS (NOSQL)

streaming
batch

parquet
json
csv

sftp

mysql
mongodb

nifi

Ingestão de Dados



Processamento de Dados



Armazenamento de Dados



Visualização de Dados



Acesso de Dados



Consolidação de Dados



EXPLICAÇÃO DO FRAMEWORK #1

A estrutura começa com a coleta de dados de diferentes fontes, como aplicativos, sites e APIs, que são as interfaces dos clientes da QuantumFinance. Esses dados são enviados aos bancos de dados relacional (MySQL) ou não-relacional (MongoDB) de acordo com a característica do dado.

O Apache NiFi é utilizado para a ingestão de dados, automatizando o fluxo entre os bancos de dados e os softwares seguintes de processamento e armazenamento de dados. Ele coleta, transforma e move os dados das fontes, que serão processados pelo Apache Spark, uma plataforma robusta capaz de realizar análises e transformações complexas em grandes volumes de dados em tempo real ou em lote.

Os dados processados são armazenados no Hadoop, que utiliza um sistema de arquivos distribuído (HDFS) para garantir uma armazenagem escalável e redundante. Isso cria um repositório central no Data Lake onde os dados podem ser acessados e manipulados conforme necessário.



EXPLICAÇÃO DO FRAMEWORK #2

Para consolidar os dados, o Hive é utilizado. Ele fornece uma interface semelhante ao SQL que facilita a análise e a síntese de grandes conjuntos de dados armazenados no Hadoop. A partir dessa consolidação, o Presto entra em ação, permitindo consultas analíticas interativas e distribuídas sobre os grandes volumes de dados, e nos garantindo acesso rápido e eficiente aos dados no Data Lake.

Por fim, para visualização e análise interativa desses dados, utilizamos o Power BI. Ele se conecta ao Data Lake para criar dashboards, relatórios e insights baseados nos dados processados, permitindo uma interpretação mais fácil e intuitiva dos nossos dados.

A orquestração de todo esse fluxo de trabalho é gerenciada pelo Apache Airflow, que automatiza as tarefas repetitivas e coordena as pipelines de dados complexos.

Dessa forma, nosso framework garante um fluxo contínuo e eficiente de dados desde a coleta até a análise e visualização, assegurando que grandes volumes de dados possam ser gerenciados e utilizados de maneira eficaz e contemplando soluções para aplicações com necessidades não relacionais, transacionais e de análise e ciência de dados.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...



1. **Alto Desempenho:** Rápido e eficiente em transações.
2. **Escalabilidade:** Cresce conforme a necessidade.
3. **Segurança:** Protege contra acessos não autorizados.
4. **Facilidade de Uso:** Simples de instalar e gerenciar.
5. **Comunidade:** Grande suporte e documentação.
6. **Custo-Benefício:** Open-source e econômico.
7. **Flexibilidade:** Suporte a diferentes tipos de armazenamento.
8. **Compatibilidade:** Funciona bem com várias plataformas e linguagens.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...



1. **Alta performance:** MongoDB oferece rápida execução para operações de leitura e escrita.
2. **Escalabilidade horizontal com Sharding:** Distribui dados em múltiplos servidores para escalabilidade fácil.
3. **Flexibilidade no modelo de dados:** Utiliza documentos JSON para adaptabilidade e agilidade.
4. **Facilidade de uso:** Instalação e configuração simplificadas para uma experiência amigável.
5. **Suporte e documentação extensiva:** Conta com uma grande comunidade para assistência e recursos.
6. **Open-source e econômico:** Solução acessível e sem custos de licenciamento.
7. **Robustez com replicação e failover automáticos:** Garante a segurança e disponibilidade dos dados.
8. **Compatibilidade com várias plataformas e linguagens de programação:** Funciona bem em diversos ambientes e integra-se facilmente.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...



1. **Confiabilidade:** Oferece processamento de dados confiável.
2. **Escalabilidade:** Capaz de lidar com grandes volumes de dados e aumentar sua capacidade conforme necessário.
3. **Segurança avançada:** Implementa recursos robustos de segurança para proteger os dados.
4. **Suporte a várias fontes de dados e protocolos:** Pode integrar-se a uma variedade de fontes e protocolos de dados.
5. **Comunidade ativa:** Beneficia-se de uma comunidade ativa de usuários e desenvolvedores, fornecendo suporte e contribuições contínuas.



JUSTIFICATIVAS

**Para a nossa fintech, ele
é bom porque provê...**



1. **Alta velocidade:** Oferece processamento rápido e eficiente.
2. **Flexibilidade:** Pode trabalhar com diversas fontes de dados e algoritmos.
3. **Análise de big data:** Ideal para análise de grandes conjuntos de dados.
4. **Aprendizado de máquina:** Suporta algoritmos de aprendizado de máquina para análise preditiva e descritiva.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...

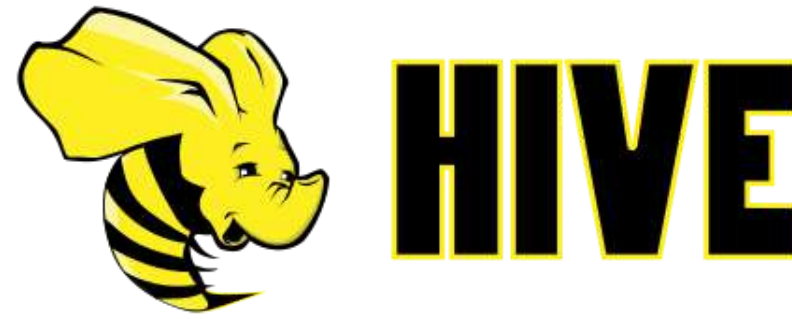


1. **Armazenamento escalável:** Oferece armazenamento escalável para grandes volumes de dados.
2. **Facilita análise de big data:** Permite análises eficientes e confiáveis em grandes conjuntos de dados.
3. **Confiabilidade:** Garante a integridade e disponibilidade dos dados durante o processamento.
4. **Eficiência:** Possibilita o processamento eficiente de dados, mesmo em ambientes com grandes cargas de trabalho.



JUSTIFICATIVAS

**Para a nossa fintech, ele
é bom porque provê...**



1. **Linguagem SQL:** Utiliza uma linguagem familiar, SQL, simplificando o processamento e análise de dados.
2. **Simplifica o processamento de big data:** Facilita a obtenção de insights valiosos de grandes volumes de dados de maneira eficiente.
3. **Eficiência:** Permite realizar operações de forma eficiente, mesmo em grandes conjuntos de dados.



JUSTIFICATIVAS

**Para a nossa fintech, ele
é bom porque provê...**



1. **Escalabilidade para grandes volumes de dados e consultas complexas:** Capaz de lidar com grandes volumes de dados e consultas de alta complexidade.
2. **Compatibilidade com várias fontes de dados:** Integra-se facilmente a diferentes fontes de dados.
3. **Flexibilidade com consultas SQL padrão:** Permite realizar consultas utilizando a linguagem SQL padrão.
4. **Arquitetura distribuída para alta disponibilidade:** Distribui consultas em vários nós para garantir alta disponibilidade.
5. **Comunidade ativa para suporte contínuo:** Beneficia-se de uma comunidade ativa de usuários e desenvolvedores para suporte.
6. **Open-source e customização sem custos de licenciamento:** É uma solução open-source, permitindo customizações sem custos de licenciamento.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...



1. **Integração com diversas fontes de dados:** Permite importar dados de uma variedade de fontes para análise.
2. **Análises avançadas com recursos de inteligência artificial:** Possibilita análises mais profundas com recursos de IA.
3. **Compartilhamento fácil de relatórios e painéis:** Facilita a distribuição de informações através de relatórios e painéis interativos.
4. **Segurança avançada para proteção de dados:** Garante a segurança dos dados durante o compartilhamento e análise.
5. **Solução escalável e acessível:** Adequado para organizações de todos os tamanhos, com custos acessíveis e capacidade de expansão.
6. **Tomada de decisões informadas com insights claros e acionáveis:** Capacita os usuários a tomar decisões com base em informações precisas e relevantes.



JUSTIFICATIVAS

Para a nossa fintech, ele é bom porque provê...



1. **Orquestração de Fluxos de Trabalho:** Ele oferece uma plataforma para orquestrar fluxos de trabalho complexos, permitindo a execução sequencial ou paralela de tarefas, com dependências claras entre elas.
2. **Monitoramento e Gerenciamento Centralizado:** Com sua interface web intuitiva, o Airflow oferece visibilidade e controle sobre o status das tarefas e fluxos de trabalho, facilitando o monitoramento e gerenciamento centralizado.
3. **Reutilização de Código e Modularidade:** Ele promove a reutilização de código através de DAGs (Directed Acyclic Graphs), permitindo a modularidade e a fácil manutenção dos fluxos de trabalho.
4. **Escalabilidade e Tolerância a Falhas:** O Airflow é altamente escalável e tolerante a falhas, podendo lidar com grandes volumes de dados e processos complexos de forma confiável.
5. **Integração com Ecossistema de Dados:** Ele se integra facilmente com outras ferramentas e tecnologias do ecossistema de dados, como Apache Spark, Hadoop, Kubernetes, entre outros.
6. **Comunidade Ativa e Suporte:** O Apache Airflow possui uma comunidade ativa de usuários e desenvolvedores, oferecendo suporte, documentação e contribuições contínuas para o seu desenvolvimento e aprimoramento.



FIAP

FLUXO NIFI

MySQL -> FILE.json



QUANTUMFINANCE



niFi fcff8324-d544-4a8c-a74b-57bf611a5438 LOG OUT

0 / 0 bytes 0 0 9 0 0 3 0 0 0 0 0 0 11:20:49 BRT

```
graph LR; GetClientesFromMySQL[GetClientesFromMySQL] --> NameSuccess1[Name success]; ConvertMySQLToJson[ConvertMySQLToJson] --> NameSuccess2[Name success]; SplitJson[SplitJson] --> NameSplit[Name split]; CollectAttributes[CollectAttributes] --> NameMatched[Name matched]; CalculaAprovacaoEmprestimo[CalculaAprovacaoEmprestimo] --> NameSuccess3[Name success]; GroupAttributes[GroupAttributes] --> NameSuccess4[Name success]; MergeJsonObject[MergeJsonObject] --> NameMerged[Name merged]; ChangeFilename[ChangeFilename] --> NameSuccess5[Name success]; SaveFileJSON[SaveFileJSON];
```

The diagram illustrates a data flow in NiFi for processing MySQL data into JSON and back into MySQL. The flow consists of the following components and connections:

- GetClientesFromMySQL** (QueryDatabaseTable 1.26.0) receives input from the left and outputs to **Name success** (Queued 0 (0 bytes)).
- ConvertMySQLToJson** (ConvertRecord 1.26.0) receives input from **Name success** and outputs to **Name success** (Queued 0 (0 bytes)).
- SplitJson** (Split.Json 1.26.0) receives input from **Name success** and outputs to **Name split** (Queued 0 (0 bytes)).
- CollectAttributes** (Evaluate.JsonPath 1.26.0) receives input from **Name split** and outputs to **Name matched** (Queued 0 (0 bytes)).
- CalculaAprovacaoEmprestimo** (UpdateAttribute 1.26.0) receives input from **Name matched** and outputs to **Name success** (Queued 0 (0 bytes)).
- GroupAttributes** (AttributesToJSON 1.26.0) receives input from **Name success** and outputs to **Name success** (Queued 0 (0 bytes)).
- MergeJsonObject** (MergeContent 1.26.0) receives input from **Name success** and outputs to **Name merged** (Queued 0 (0 bytes)).
- ChangeFilename** (UpdateAttribute 1.26.0) receives input from **Name merged** and outputs to **Name success** (Queued 0 (0 bytes)).
- SaveFileJSON** (PutFile 1.26.0) receives input from **Name success** and outputs to the right.

Each component box displays its name, version, and performance metrics (In, Read/Write, Out, Tasks/Time) over a 5-minute period.

Fluxo do Nifi – OutputClaim PutFile (Processor Final)



View as: original

Filename: arquivo.json
Content Type: application/json

```
1  [{
2    "whatsapp" : "11998765432",
3    "score" : "123",
4    "endereço" : "Rua A, 123",
5    "emprestimoAprovado" : "Não",
6    "cpf_cliente" : "12345678901",
7    "nome" : "Maria Silva",
8    "email" : "maria@email.com",
9    "cep" : "12345678"
10 }, {
11   "whatsapp" : "11987654321",
12   "score" : "456",
13   "endereço" : "Avenida B, 456",
14   "emprestimoAprovado" : "Não",
15   "cpf_cliente" : "23456789012",
16   "nome" : "João Oliveira",
17   "email" : "joao@email.com",
18   "cep" : "87654321"
19 }, {
20   "whatsapp" : "11976543210",
21   "score" : "789",
22   "endereço" : "Praça C, 789",
23   "emprestimoAprovado" : "Sim",
24   "cpf_cliente" : "34567890123",
25   "nome" : "Ana Santos",
26   "email" : "ana@email.com",
27   "cep" : "45678901"
28 }, {
29   "whatsapp" : "11996917284",
30   "score" : "998",
31   "endereço" : "Onde ha alegria",
32   "emprestimoAprovado" : "Sim",
33   "cpf_cliente" : "34567891013",
34   "nome" : "Yago Angelini",
35   "email" : "yagao@wayne.com",
36   "cep" : "06172006"
37 }, {
38   "whatsapp" : "11965432109",
39   "score" : "987",
40   "endereço" : "Alameda D, 012",
41   "emprestimoAprovado" : "Sim",
42   "cpf_cliente" : "45678901234",
43   "nome" : "Pedro Souza"
```


SCRIPT DE CRIAÇÃO DA BASE DE DADOS MYSQL

Table cliente | Database quantumfinance

```
# 1) CRIAÇÃO DO DATABASE quantumfinance|
create database quantumfinance;

# 2) SELECIONAR DATABASE quantumfinance
use quantumfinance;

# 3) CRIAÇÃO DA TABELA cliente
create table cliente(
    cpf_cliente char(11) NOT NULL,
    nome varchar(50) NOT NULL,
    endereco varchar(100) NOT NULL,
    cep char(8) NOT NULL,
    score varchar(4) NOT NULL,
    email varchar(50),
    whatsapp char(11) NOT NULL,
    primary key(cpf_cliente)
);

# 4) INSERÇÃO DE DADOS NA TABELA cliente
INSERT INTO cliente (cpf_cliente, nome, endereco, cep, score, email, whatsapp) VALUES
('12345678901', 'Maria Silva', 'Rua A, 123', '12345678', '123', 'maria@email.com', '11998765432'),
('23456789012', 'João Oliveira', 'Avenida B, 456', '87654321', '456', 'joao@email.com', '11987654321'),
('34567890123', 'Ana Santos', 'Praça C, 789', '45678901', '789', 'ana@email.com', '11976543210'),
('45678901234', 'Pedro Souza', 'Alameda D, 012', '78901234', '987', 'pedro@email.com', '11965432109'),
('56789012345', 'Carla Pereira', 'Travessa E, 345', '90123456', '654', 'carla@email.com', '11954321098'),
('67890123456', 'Fernando Lima', 'Rodovia F, 678', '23456789', '321', 'fernando@email.com', '11943210987'),
('78901234567', 'Juliana Costa', 'Estrada G, 901', '56789012', '231', 'juliana@email.com', '11932109876'),
('89012345678', 'Lucas Oliveira', 'Av. H, 234', '90123456', '564', 'lucas@email.com', '11921098765'),
('90123456789', 'Mariana Almeida', 'Rua I, 567', '34567890', '897', 'mariana@email.com', '11910987654'),
('34567891013', 'Yago Angelini', 'Onde ha alegria', '06172006', '998', 'yagao@wayne.com', '11996917284');
```



EXPLICAÇÃO DO FLUXO

MySQL -> File.json

1. GetClientesFromMySQL

- Esse Processor é responsável por se conectar com a table “cliente” do database “quantumfinance” no banco MySQL que criamos. Ele captura todos os dados que a tabela possui, e traz para o NiFi no formato Avro.

2. ConvertMySQLToJson

- Como precisamos fazer transformações no dado, convertemos do formato Avro para JSON através desse Processor.

3. SplitJson

- Cada objeto do nosso JSON precisa ser tratado individualmente, então dividimos nosso JsonTree em vários objetos, um para cada linha da tabela Cliente.

4. CollectAttributes

- Aqui, mapeamos os atributos do JSON para que pudéssemos fazer transformações sem que eles se percam.



EXPLICAÇÃO DO FLUXO

MySQL -> File.json

5. CalculaAprovacaoEmprestimo

- Esse processor é responsável por criar um atributo novo no objeto JSON, chamado de “emprestimoAprovado”. Ele verifica se o “score” do cliente é acima de 650, se for, ele atribui “Sim” ao campo, caso contrário, “Não”. A instrução usada é: `${score:number():gt(650):ifElse('Sim','Não')}`

6. GroupAttributes

- Os atributos existentes, e o novo atributo são agrupados aqui, para que o arquivo final contenha o conteúdo completo que desenvolvemos.

7. MergeJsonObjects

- Nesse Processor, pegamos os objetos JSON que fizemos o “Split” lá atrás, e fazemos um “Merge” de todos eles novamente.

8. ChangeFilename

- Apenas para fins de visualização, definimos aqui o nome do arquivo final a ser exportado e sua extensão (arquivo.json).

9. SaveFileJSON

- Por fim, exportamos o JSON final para um arquivo que será enviado ao time de Ciência de Dados a fim de ser analisado.



EXPLICAÇÃO DO FLUXO

MySQL -> File.json

Observações Importantes

- A cada **500** segundos, a base de dados será analisada novamente e novos registros serão coletados caso existam.
- Caso o arquivo exportado já exista, não resulta em erro, mas ele dá um **“replace”** no arquivo final, e esse novo arquivo final contém todos os registros atuais e novos da table “cliente”;
- Gostaríamos de ter exportado em um arquivo **.parquet**, uma ótima opção para ciência de dados, pela sua capacidade de armazenar muitos dados, mas com **tamanho de arquivo bem comprimido**, principalmente. Entretanto, ocorreram erros com nosso HDFS, e não foi possível, mas adotamos o JSON, e acreditamos que ele e o CSV são boas opções para análise de dados.
- Nosso fluxo parte do pressuposto que os sistemas da QuantumFinance tem **integração direta** com os bancos e já salvam os dados automaticamente neles. Uma vez que isso acontece, o dado precisa ser transitado para o posterior processamento e armazenamento consolidado. Aí que o NiFi entraria, fazendo essa ingestão praticamente em **tempo real**, trazendo dados dos bancos para as próximas etapas do framework de dados.



RESPOSTA DA EQUIPE DE CIENTISTAS DE DADOS

Prova do Arquivo Final

```
import pandas as pd

clientes_df = pd.read_json("C:/Users/Yago Angelini/Downloads/arquivo_json/arquivo.json")
clientes_df
```

✓ 0.0s

	whatsapp	score	endereco	emprestimoAprovado	cpf_cliente	nome	email	cep
0	11998765432	123	Rua A, 123	Não	12345678901	Maria Silva	maria@email.com	12345678
1	11987654321	456	Avenida B, 456	Não	23456789012	João Oliveira	joao@email.com	87654321
2	11976543210	789	Praça C, 789	Sim	34567890123	Ana Santos	ana@email.com	45678901
3	11996917284	998	Onde ha alegria	Sim	34567891013	Yago Angelini	yagao@wayne.com	6172006
4	11965432109	987	Alameda D, 012	Sim	45678901234	Pedro Souza	pedro@email.com	78901234
5	11954321098	654	Travessa E, 345	Sim	56789012345	Carla Pereira	carla@email.com	90123456
6	11943210987	321	Rodovia F, 678	Não	67890123456	Fernando Lima	fernando@email.com	23456789
7	11932109876	231	Estrada G, 901	Não	78901234567	Juliana Costa	juliana@email.com	56789012
8	11921098765	564	Av. H, 234	Não	89012345678	Lucas Oliveira	lucas@email.com	90123456
9	11910987654	897	Rua I, 567	Sim	90123456789	Mariana Almeida	mariana@email.com	34567890

- A equipe de cientistas de dados agradeceu pela criação do fluxo de ingestão, e disse que os têm ajudado a capturar informações da base do MySQL.
- Ousaram ainda dizer que a criação da coluna “emprestimoAprovado”, caso fosse de um dado real, poderia funcionar como coluna “Target” no treino de um algoritmo de aprendizado supervisionado. Muito bom!



FIAP

Obrigado!

QUANTUMFINANCE

