

Construção de uma infraestrutura para a coleta e armazenamento de dados sobre a Covid19 e a educação brasileira

Kálvin Antunes de Almeida

Orientador(a): João Paulo Barbosa Nascimento

31/12/2021





KÁLVIN ANTUNES DE ALMEIDA

INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO

RELATÓRIO DO PROJETO APLICADO

Construção de uma infraestrutura para a coleta e armazenamento de dados sobre a Covid19 e a educação brasileira

Relatório de Projeto Aplicado desenvolvido para
fins de conclusão do curso de Engenharia de
Dados.

Orientador (a): João Paulo Barbosa Nascimento

Caxias do Sul

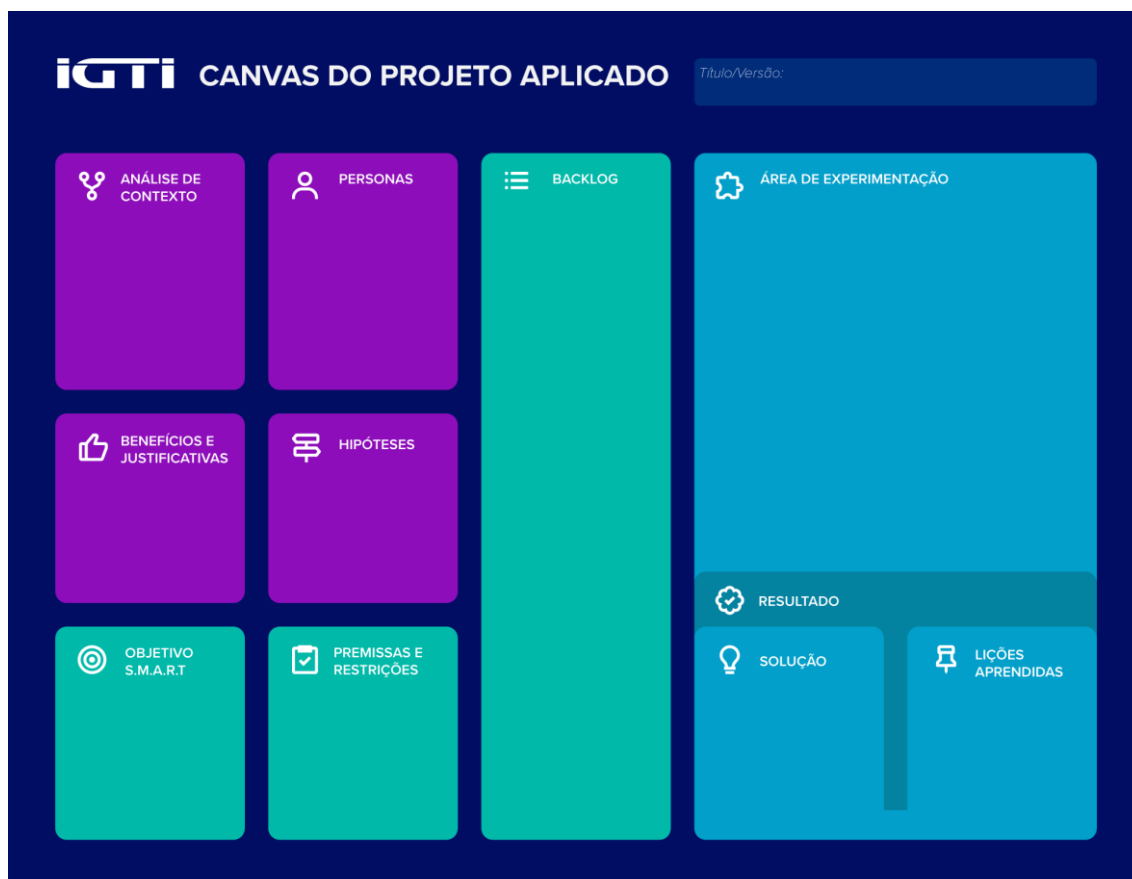
31/12/2021

Sumário

1. CANVAS do Projeto Aplicado	4
1.1 Desafio	5
1.1.1 Análise de Contexto	5
1.1.2 Personas	7
1.1.3 Benefícios e Justificativas	8
1.1.4 Hipóteses	10
1.2 Solução	12
1.2.1 Objetivo SMART	12
1.2.2 Premissas e Restrições	13
1.2.3 Backlog de Produto	14
2. Área de Experimentação	15
2.1 Sprint 1	15
2.1.1 Solução	15
• Evidência do planejamento:	15
• Evidência da execução de cada requisito:	15
• Evidência dos resultados:	23
2.1.2 Lições aprendidas	25
2.2 Sprint 2	26
2.2.1 Solução	26
• Evidência do planejamento:	26
• Evidência da execução de cada requisito:	26
• Evidência dos resultados:	31
2.2.2 Lições aprendidas	35
2.3 Sprint 3	36
2.3.1 Solução	36
• Evidência do planejamento:	36
• Evidência da execução de cada requisito:	36
• Evidência dos resultados:	42
2.3.2 Lições aprendidas	45
3. Considerações Finais	46
3.1 Resultados Finais	46
3.2 Contribuições	47
3.3 Próximos passos	47

1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado.



1.1 Desafio

1.1.1 Análise de Contexto

No início de 2020, a OMS declarou que o surto do novo coronavírus seria tratado nos mais altos níveis de importância internacional. Em meados de março, a Covid-19 foi caracterizada pela OMS como uma pandemia. Desde o início desta até os dias atuais, o coronavírus causou diversos impactos negativos no Brasil e no mundo. Dentre esses impactos, uma das áreas mais afetadas foi a educação.

A pandemia pelo novo coronavírus provocou um cenário inédito de isolamento social, com rápida transição para o ensino à distância e um enorme impacto no aspecto emocional de milhões de estudantes, educadores e famílias. Ou seja, é possível identificar sem análises mais profundas as consequências do novo coronavírus, contudo não é possível encontrar de uma forma mais específica sem um estudo mais detalhado. Sem essas informações, não é possível combater os reais problemas causados pela pandemia na educação brasileira.

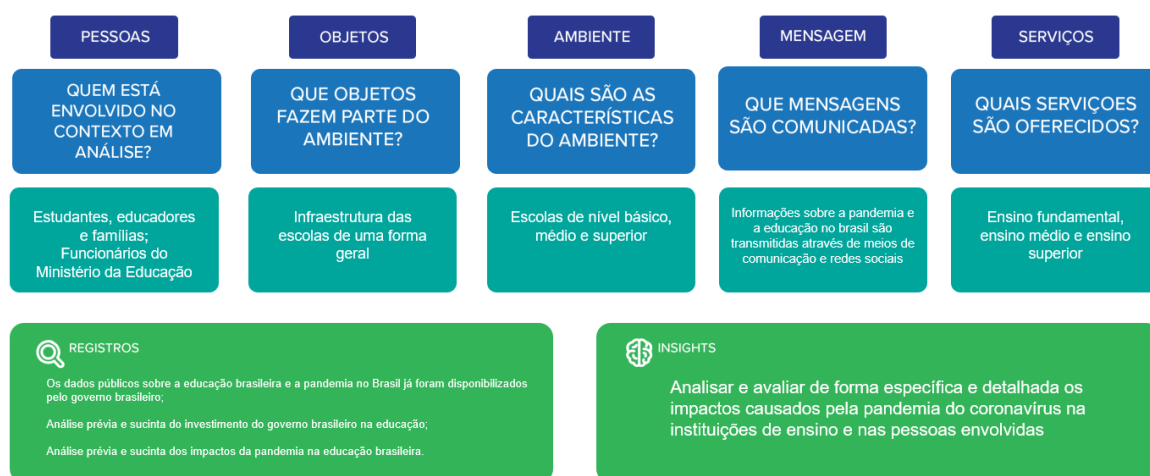
Por mais que o Brasil seja um dos países que mais investe em educação no mundo, investindo cerca de 5,7% do seu PIB, o investimento é insuficiente para sanar todas as necessidades. Segundo especialistas, o investimento deve ser feito com mais eficiência, atacando os reais problemas e tendo um retorno adequado. Portanto, o primeiro passo para um investimento mais eficiente é descobrir onde esse capital deve ser destinado. Para iniciar o processo dessa análise, foi elaborada a Matriz CSD para entender o problema que deverá ser resolvido.

Figura 1 - Matriz CSD

		CERTEZAS	SUPOSIÇÕES	DÚVIDAS
DIFERENTES ÓTICAS DE ANÁLISE	ATORES	Estudantes, educadores e famílias sofreram um grande impacto por causa da pandemia	A classe social das pessoas analisadas são diretamente ligadas aos impactos sofridos pelas mesmas	Qual o nível de impacto causado nas pessoas analisadas?
	CENÁRIOS		Investimento na educação brasileira é feito de forma ineficiente e ineficaz	
	REGRAS	É possível melhorar a eficiência e a eficácia dos recursos disponíveis do governo na educação brasileira	Falta de dados concretos para auxiliar a tomada de decisão de investimentos na área da educação	Todos os impactos analisados serão de fato ocasionados pela pandemia?

Com a análise da Matriz CSD, é possível perceber que o problema de investimento inadequado pode ser resolvido utilizando nas informações sobre a educação brasileira e sobre a pandemia no Brasil. Portanto, a melhor alternativa é efetuar análises dos dados da educação cruzando informações com os dados da pandemia. Para termos uma visão mais sistêmica do problema e da solução, foi desenvolvida a tabela da observação POEMS.

Figura 2 - Observação POEMS



Para fazer essa análise, deverá ser criada uma *pipeline* para coletar os dados já disponíveis nos portais públicos, transformá-los em informações concretas e disponibilizar as mesmas para os profissionais habilitados para que sejam feitas os devidos desenvolvimentos para levantar as questões do desafio. Com os desenvolvimentos feitos, serão disponibilizadas informações relevantes aos gestores do Ministério da Educação que podem gerar conhecimento sobre as áreas mais afetadas dentro da educação brasileira e, assim, definir qual a melhor forma de destinar os recursos humanos e financeiros para sanar os problemas apontados nesses estudos. Essas informações serão disponibilizadas em forma de *dashboards* interativos, no qual todos os gestores responsáveis terão acesso. Contudo, o foco deste projeto se limitará em apenas criar e gerenciar a *pipeline* de dados e em disponibilizar estes dados para as consultas que serão feitas por outros profissionais.

1.1.2 Personas

A persona é um ser fictício que possui as características de um cliente ideal para a solução proposta, sendo apresentada de uma forma mais humanizada e personalizada. Para que seja possível visualizar isso, foi criado o Mapa de Empatia abaixo:

Figura 3 - Mapa de Empatia



Para poder tipificar as personas, foram criados os exemplos a seguir, ambos fictícios e ideias para a ser os clientes finais da proposta.

Persona 1: Lucas Silva

Profissão: Gestor na Ministério da Educação

Idade: 36

Sexo: Masculino

Estado Civil: Solteiro

Filhos(as): Não

Educação: Pós-Graduado

Objetivos: Melhorar os indicadores de performance dos investimentos do ministério da educação, iniciar o seu mestrado em uma universidade federal

Desafios: Encontrar informações relevantes para traçar o Plano de Ação do ministério

Como a solução proposta pode ajudá-la: Com as informações geradas nesse projeto, será possível efetuar o Plano de Ação, definindo como os recursos disponíveis serão divididos dentro da educação brasileira e, caso precise, solicitar mais recursos para o ministério utilizando as informações do projeto como embasamento

Persona 2: Maria Lúcia

Profissão: Professora universitária

Idade: 38

Sexo: Feminino

Estado Civil: Casada

Filhos(as): Sim, 2 filhos

Educação: Doutoranda

Objetivos: Finalizar o doutorado, passar mais tempo com sua família

Desafios: Melhorar sua forma de ministrar as aulas e de gerenciar seu tempo entre trabalho e família

Como a solução proposta pode ajudá-la: Facilitar a forma que ministra suas aulas a distância, conseguir melhores recursos das instituições de ensino para melhorar a qualidade de suas aulas

Persona 3: João Pedro

Profissão: Aluno do ensino fundamental

Idade: 6

Sexo: Masculino

Reside com: Mora com seu pai e sua mãe

Educação: Ensino fundamental incompleto

Objetivos: Terminar o ensino fundamental, aproveitar ao máximo a escola e seus colegas de turma

Desafios: Voltar a ter uma educação presencial de qualidade

Como a solução proposta pode ajudá-lo: Facilitar as instituições de ensino a melhorar sua forma de ensino, sua infraestrutura e seus recursos humanos para que ele possa vencer o desafio citado

1.1.3 Benefícios e Justificativas

O mundo vive uma época muito conturbada devido a pandemia. Muitos países, se não todos, passam por grandes dificuldades socioeconômicas. Conforme já explicado, uma das áreas mais afetadas foi a da educação. No Brasil, além de todos os problemas causados devido ao Covid-19, temos ainda mais dois fatores agravantes se tratando de educação: o investimento insuficiente e ineficaz e a falta de detalhamento técnico nas pesquisas efetuadas para determinar as reais causas de problemas.

Portanto, conhecer com maior exatidão as áreas dentro da educação brasileira que mais foram afetadas é a principal justificativa para esse projeto, visto que, através desse conhecimento, será possível utilizar os recursos disponíveis de forma mais eficiente e mais eficaz. Ou seja, os recursos humanos e financeiros dedicados à educação brasileira serão destinados aos reais problemas que estão ocorrendo, uma vez que essas informações serão concretizadas com as análises dos dados propostos. Para explorar mais o problema e o desafio, foi elaborado o *Business Design Blueprint*, capaz de mostrar o produto em detalhes.

Figura 4 - Business Design Blueprint

ITENS	DETALHAMENTO
Objetivos	Buscar as melhores formas de efetuar o investimento na educação de forma eficiente e eficaz
Atividades	Análise informações em dados genéricos da educação e da saúde de forma separada
Questões	Devo investir na infraestrutura? Em recursos humanos? Em métodos de ensino? Em questões externas relacionadas educação?
Barreiras	Falta de recursos financeiros e falta de dados concretos
Ações do Cliente	Efetua o investimento na educação da melhor forma possível com as ferramentas e dados que tem disponível
Funcionalidades	Exploração de informações concretas, de fácil acesso e de fácil interpretação
Integração	Dashboard com possibilidade de alterar as visualizações conforme necessidade
Mensagem	Se melhores informações e conhecimento, então melhor investimento
Onde Ocorre	Órgãos do governo, instituições de ensino de nível fundamental, médio e superior.
Tarefas Aparentes	Dashboard interativo
Tarefas Escondidas	Estrutura de Data Lake e cluster com engine de consulta dessas informações
Processos de Suporte	Estrutura de servidores locais em boas condições ou financiamento para estruturas na nuvem
Saída Desejável	Informações relevantes e de fácil compreensão disponibilizadas em Dashboard interativo

Além do *Business Design Blueprint*, foi desenvolvido também o Canvas de Proposta de Valor, capaz de delimitar o valor da solução proposta relacionando-se aos desejos e necessidades do cliente.

Figura 5 - Canvas de Proposta de Valor



1.1.4 Hipóteses

Uma vez definido o contexto do desafio, agora é necessário apresentar as hipóteses relacionadas ao mesmo que vão direcionar o desenvolvimento da solução. Para isso, foi elaborada a Matriz de Observações abaixo.

Figura 5 - Matriz de Observações

OBSERVAÇÕES	HIPÓTESE
A tomada de decisão será melhor quando os dados foram mais específicos e detalhados	Disponibilizar informações específicas e detalhadas através de dados públicos
Dificuldade em verificar aonde os recursos financeiros e humanos serão distribuídos nos planos de ação	Realizar análises das informações disponibilizadas para facilitar a distribuição de recursos
Um dashboard interativo facilita a visualização das informações	Disponibilizar as informações em dashboards interativos de fácil acesso e manuseio

Com essas observações feitas, fica clara a necessidade de se possuir dados específicos e detalhados, que foram devidamente tratados, apresentados e disponíveis para consumo, sejam para os profissionais responsáveis pelas consultas e desenvolvimentos ou para os usuários finais.

Para isso, há três maneiras possíveis, que são apresentadas nas propostas a seguir:

- 1) Coletar, transformar e disponibilizar os dados necessários em um ambiente na nuvem;
- 2) Coletar, transformar e disponibilizar os dados necessários em um ambiente local;
- 3) Coletar, transformar e disponibilizar os dados necessários em um ambiente híbrido, usando tanto ambiente local quanto na nuvem.

Para analisar as três maneiras disponíveis, será utilizada a Matriz de Priorização de Ideias. Essa matriz é a ferramenta ideal neste cenário para selecionar a ideia que apresenta a maior viabilidade para implantação. Nesta matriz, as ideias selecionadas serão avaliadas nos critérios de avaliação em notas, no qual a ideia que possuir o maior somatório de notas será a escolhida. Para definir os critérios e pontuação, será utilizada a tabela balizadora de acrônimo BASICO.

Figura 6 - Tabela BASICO

Escala	B - Benefícios	A - Abrangência	S - Satisfação	I - Investimentos	C - Cliente	O - Operacionalidade
5	De vital importância	Total (de 70 a 100%)	Muito grande	Pouquíssimo investimento	Nenhum impacto	Muito fácil
4	Significativo	Muito grande (de 40 a 70%)	Grande	Algum investimento	Impacto pequeno	Fácil
3	Razoável	Razoável (de 20 a 40%)	Média	Médio investimento	Médio impacto	Média facilidade
2	Poucos benefícios	Pequena (de 5 a 20%)	Pequena	Alto investimento	Impacto grande	Difícil
1	Algum benefício	Muito pequena	Quase não é notada	Altíssimo investimento	Impacto muito grande no cliente	Muito difícil

Figura 7 - Matriz de Priorização de Ideias

Ideias	Critérios de Avaliação						Soma
	C1	C2	C3	C4	C5	C5	
I1	5	5	5	2	4	5	26
I2	4	3	4	4	4	4	23
I3	4	3	4	4	5	2	22

Portanto, após a análise na Matriz de Priorização de Ideias, podemos concluir que coletar, transformar e disponibilizar os dados necessários em um ambiente na nuvem é a melhor alternativa considerando o cenário atual.

1.2 Solução

1.2.1 Objetivo SMART

Determinada a ideia vencedora na Matriz de Priorização de Ideias, é necessário agora definir o objetivo que esse projeto almeja, considerando as premissas e restrições existentes. O objetivo deve ser capaz de alinhar as expectativas, de maneira clara e objetiva, visando maximizar as chances de alcançar os resultados esperados.

O objetivo, de forma geral, é criar um *Data Lake* em uma estrutura *Cloud*, utilizando ferramentas de extração, transformação e carregamento (*ETL*) capaz de coletar os dados públicos sobre a educação (Censo Escolar) e a saúde (especificamente a pandemia do Covid-19) no Brasil e, após isso, disponibilizar os dados para consultas através de uma estrutura de *data warehouse*. Contudo, esse objetivo deve ser atingindo de forma SMART: Específico (*Specific*), Mensurável (*Mensurable*), Atingível (*Attainable*), Relevante (*Relevant*) e Temporal (*Time based*).

- Objetivo Específico: disponibilizar de forma fácil e intuitiva os dados públicos coletados para que possam ser feitas análises e consultas;
- Objetivo Mensurável: conseguir definir através dos dados disponibilizados quais as melhores alternativas para as tomadas de decisão;
- Objetivo Atingível: utilizar as ferramentas de *ETL*, *Data Lake* e *Data Warehouse* para disponibilizar os dados tratados;
- Objetivo Relevante: oferecer informações relevantes e específicas sobre o assunto para a posterior análise e consulta por parte dos gestores;
- Objetivo Temporal: implementar todas as etapas, até a disponibilização de dados no *data warehouse*, em até 60 dias corridos.

1.2.2 Premissas e Restrições

Com os objetivos definidos, é necessário estabelecer as premissas e as restrições do projeto. As premissas são todos os fatores dentro do projeto que são assumidos como verdadeiros sem que haja uma demonstração prévia, ou seja, são consideradas hipóteses. Já as restrições são os fatores internos e externos que limitam as opções de atuação dentro do projeto. Em outras palavras, são como se fossem os requisitos mínimos de um projeto.

1) Premissas:

- Não será necessária alteração na infraestrutura interna;
- Eventuais erros que ocorrerem durante a implantação e testes serão corrigidos no decorrer do projeto antes da entrega final;
- Toda a infraestrutura será feita no *Google Cloud Platform (GCP)*;
- Todos os softwares utilizados serão *Open Source* dentro da infraestrutura estabelecida;
- A linguagem padrão utilizada durante o desenvolvimento das *pipelines* de dados e nos processos de *ETL* será o *Python*.
- Posteriormente, serão definidos os Cientistas de Dados e Analistas de Dados responsáveis por elaborar as consultas e análises após os dados serem disponibilizados (não será feito neste projeto).

2) Restrições:

- Não poderão ser utilizados dados não oficiais (não disponibilizados pelo governo brasileiro) no projeto;
- O Engenheiro de Dados é o único responsável pela criação e manutenção das *pipeline* de dados e dos processos de *ETL*;
- O trabalho será feito diariamente, todos os dias da semana, com pelo menos duas horas de trabalho, salvo exceções;
- Eventuais problemas de infraestruturas não serão consideradas no cronograma do projeto.

Quanto aos riscos do projeto, foi redigida uma Matriz de Risco. Nesta tabela foram listados os riscos identificados, o impacto potencial que eles podem gerar, quais as ações preventivas que podem ser tomadas para minimizar as chances de ocorrerem e, caso algum deles ocorra, como adotar as devidas ações de correção.

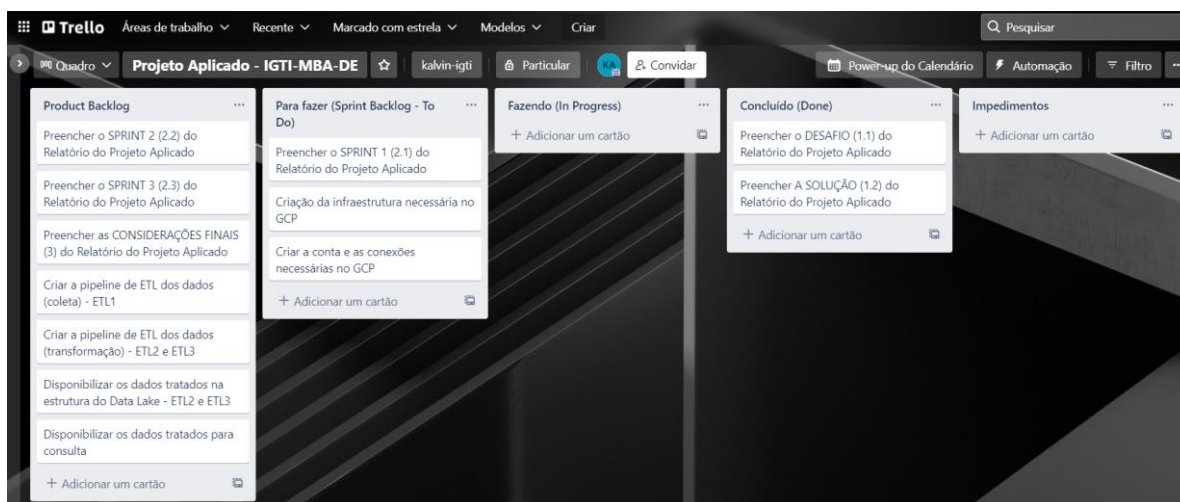
Figura 8 - Matriz de Risco

Risco Identificado	Impacto Potencial	Ações Preventivas	Ações Corretivas
Ausencia do responsável do projeto	As etapas do projeto não serão realizadas a tempo	Adiantar etapas do projeto para que não ocorram demasiados atrasos no projeto	O responsável deverá, sempre que possível, avisar sobre a sua ausência
Problemas de acesso a internet	Não acesso as plataformas Cloud	Utilizar acessos de internet via 4G / 5G	Possuir um acesso de internet backup
Falta de tempo para finalizar as etapas do projeto	Atraso na entrega do projeto final	Utilizar mais horas de trabalho diárias para suprir as necessidades de trabalho	Reorganizar o cronograma do projeto para contemplar as tarefas atrasadas
Engenharia de dados definida ser ineficiente	A parte técnica do projeto não atingirá os objetivos estabelecidos	Buscar outros profissionais para verificar o trabalho já efetuado e alterado naquilo que for preciso	Buscar alternativas de correção ou de melhor implantação com outros profissionais ou em outros repositórios

1.2.3 Backlog de Produto

O Backlog de Produto é uma lista de funcionalidades e requisitos que deverão ser entregues ao final do projeto. Essa lista é priorizada e ordenada, no qual os envolvidos no projeto deverão trabalhar conforme a priorização de cada item da lista.

Figura 9 - Backlog de Produto (Início)



2. Área de Experimentação

2.1 Sprint 1

Neste primeiro Sprint, o primeiro objetivo é criar a conta no *Google Cloud Platform* que será usada para hospedar toda a infraestrutura e códigos necessários para colocar esse projeto na prática. Lembrando que o uso do *GCP* é uma premissa já definida anteriormente. Após isso, o segundo objetivo é criar toda a infraestrutura em nuvem necessária para o projeto em si. Por fim, o terceiro objetivo é criar o *script* de *ETL* para buscar e armazenar todos os dados públicos necessários para as futuras consultas.

2.1.1 Solução

- Evidência do planejamento:

Figura 10 - Backlog de Produto

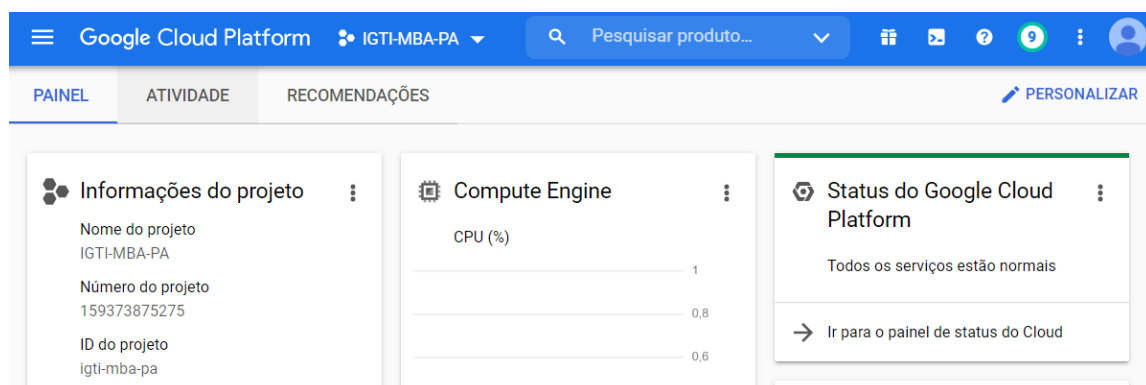


- Evidência da execução de cada requisito:

Para o primeiro objetivo, foi criada uma conta padrão no *Google*, no qual é possível acessar o *console GCP* para que seja possível acessar o seu ambiente *Cloud*. O *GCP* oferece US\$ 300 de crédito para uma avaliação gratuita, ou seja, esse projeto não terá custos em seu primeiro momento.

Com a conta funcional, foi criado o novo projeto no *GCP* que vai armazenar toda a infraestrutura deste trabalho. O nome do projeto foi intitulado: *igti-mba-pa*. Abaixo, é possível verificar na Figura 11 o console do *GCP* com o projeto criado.

Figura 11 - Projeto criado no *GCP*

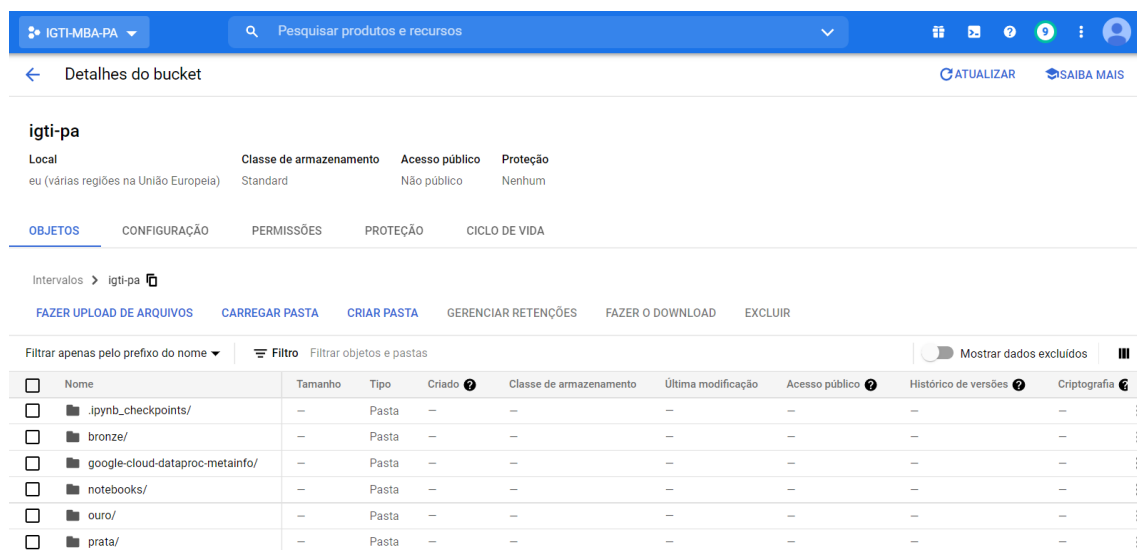


Para o segundo objetivo, foi necessário definir a melhor infraestrutura para atender as necessidades do projeto. Após uma análise geral das ferramentas disponíveis no *GCP* que são *open-source* e com um custo dentro do planejamento inicial, foi definida a infraestrutura ideal que será explicada a seguir.

Para o armazenamento, foi criado um *bucket* chamado *igti-pa* no *Cloud Storage* (Figura 12). Nesse diretório vão ficar salvos todos códigos de *ETL*, arquivos temporários de processos e todos os dados necessários para o projeto. Para armazenar esses dados, foi definido um estrutura similar ao de um *Data Lake* (arquitetura *LakeHouse*):

- Pasta BRONZE: local aonde os dados brutos serão ingeridos
- Pasta PRATA: local aonde os dados serão manipulados e tratados
- Pasta OURO: local aonde os dados finais para consulta serão disponibilizados

Figura 12 - Cloud Storage - Bucket igti-pa



igti-pa

Local: eu (várias regiões na União Europeia) | Classe de armazenamento: Standard | Acesso público: Não público | Proteção: Nenhum

OBJETOS | CONFIGURAÇÃO | PERMISSÕES | PROTEÇÃO | CICLO DE VIDA

Intervalos > igti-pa

FAZER UPLOAD DE ARQUIVOS | CARREGAR PASTA | CRIAR PASTA | GERENCIAR RETENÇÕES | FAZER O DOWNLOAD | EXCLUIR

Filtrar apenas pelo prefixo do nome | Filtro: Filtrar objetos e pastas | Mostrar dados excluídos

Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões	Criptografia
.ipynb_checkpoints/	—	Pasta	—	—	—	—	—	—
bronze/	—	Pasta	—	—	—	—	—	—
google-cloud-dataproc-metainfo/	—	Pasta	—	—	—	—	—	—
notebooks/	—	Pasta	—	—	—	—	—	—
ouro/	—	Pasta	—	—	—	—	—	—
prata/	—	Pasta	—	—	—	—	—	—

Para orquestrar as *pipelines* de dados, foi criado um *cluster* no *Google Dataproc* (Figura 13) para o processamento de dados com *Apache Spark* e *Apache Hadoop* de forma rápida e escalável, no qual é possível criar *scripts* de *ETL* usando *Python* ou *PySpark*. Para isso, foi definida as seguintes configurações do *cluster*:

- Máquina: n2-standard-4
- Imagem do Sistema Operacional: 2.0.24-centos8
- Disco: 500 GB
- Componentes: JUPYTER
- Nós do Cluster: um nó mestre e nenhum nós de trabalho

Figura 13 - *Dataproc - Cluster ETL*

Pesquisar produtos e recursos

+ IGTI-MBA-PA

<

Detalhes do cluster

+ ENVIAR JOB

A ATUALIZAR

> INICIAR

■ INTERROMPER

[X] EXCLUIR

[≡] VER REGISTROS

Região	us-east1
Zona	us-east1-c
Escalaonamento automático	Desativado
Metastore do Dataproc	Nenhum
Exclusão programada	Desativado
Nó mestre	Nó único (1 mestre, 0 worker)
Tipo de máquina	n2-standard-4
Número de GPUs	0
Tipo de disco principal	pd-standard
Tamanho do disco principal	500 GB
SSDs locais	0
Inicialização segura	Desativada
VTPM	Desativada
Monitoramento de integridade	Desativada
Bucket de preparação do Cloud Storage	igti-pa
Sub-rede	default
Tags de rede	Nenhum
Apenas IP interno	Não
Versão da imagem ?	2.0.24-centos8
Acesso ao projeto	Permita acesso à API para todos os serviços do Google Cloud no mesmo projeto
Criado em	17 de nov. de 2021 16:54:09
Componentes opcionais	JUPYTER
Propriedades	Mostrar propriedades
Segurança avançada	Desativado
Marcadores	<div>goog-datap... : etl ▼</div>
Tipo de criptografia	Chave gerenciada pelo Google

Neste projeto, toda a infraestrutura definida e explicada até então foi desenvolvida com o conceito de *IaC (Infrastructure as Code)*. A Infraestrutura como Código é um processo de provisionamento e gerenciamento de infraestrutura usando arquivos de configuração (*scripts*) ao invés de ferramentas de configuração ou configurações interativas. A decisão de usar o conceito de *IaC* foi principalmente a agilidade de implantação, já que o *script* cria toda a infraestrutura necessária em apenas uma execução. Porém, existem outras duas vantagens que foram decisivas para a sua utilização no projeto:

- **Consistência:** o *script* utilizado sempre irá gerar a mesma infraestrutura, o que garante maior segurança e controle sobre o ambiente
- **Reduz possíveis erros humanos:** com a implantação automatizada, não há como fazer alterações no ambiente já definido, ao menos que seja alterado diretamente no *script*

Neste projeto, foi utilizada a ferramenta *open-source Terraform* que permite definir os recursos e a infraestrutura em arquivos de configuração e gerenciar o ciclo de vida do projeto em si. Para a sua utilização, foi efetuado o *download* do seu arquivo executável e executado os comandos através do programa *Visual Studio Code*, um editor de código-fonte da Microsoft que é construído em código aberto e não possui custos para sua utilização.

Para a conexão com o *GCP*, foi necessário realizar o *download* do arquivo de credenciais da conta principal do projeto (que possui todas as permissões necessárias). Esse arquivo possui a extensão *JSON* (*pa-igti.json*) e ficará salvo dentro da *workspace* deste projeto. Assim, basta definirmos no *script* o parâmetro das credenciais, apontando para esse arquivo em si, conforme a Figura 14:

Figura 14 - Conexão entre o *Terraform* e o *GCP*



```

main.tf
Terraform > main.tf
1 provider "google" {
2   credentials = file(var.key)
3   project = var.project
4   region = var.region
5 }

variables.tf
Terraform > variables.tf
1 variable "project" {
2   default = "igti-mba-pa"
3 }
4
5 variable "key" {
6   default = "pa-igti.json"
7 }
8
9 variable "region" {
10  default = "us-east1"
11 }
  
```

Além dela, foi também utilizado o *Google Cloud SDK*, uma aplicação que contém ferramentas e bibliotecas para interagir com os produtos e serviços do *Google Cloud*. Para isso, foi efetuada a instalação padrão da ferramenta e configurada a conexão com o *GCP*, conforme a documentação da *Google*. Para a utilização no *Terraform*, basta acessar via *CMD* o diretório do seu arquivo executável e então digitar os comandos desejados. Abaixo, a Figura 15 comprova devida configuração do *Google Cloud SDK*:

Figura 15 - Google Cloud SDK

```
gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [pa-igti] are:
accessibility:
  screen_reader: 'False'
core:
  account: [redacted]@gmail.com
  disable_usage_reporting: 'True'
  project: igti-mba-pa
```

Com todas essas etapas concluídas, foi possível então criar o *script* de implantação da infraestrutura no GCP conforme a definição explicada anteriormente. Como boa prática, foi utilizado o arquivo principal (*main.tf*) e um arquivo com todas as variáveis (*variables.tf*). Segue abaixo o código do *main.tf* e do *variables.tf* (Figura 16), respectivamente:

Figura 16 - *main.tf* e *variables.tf*

<pre> 1 provider "google" { 2 credentials = file(var.key) 3 project = var.project 4 region = var.region 5 } 6 7 resource "google_storage_bucket" "storage_bucket" { 8 name = var.bucket_name 9 location = var.location 10 force_destroy = true 11 uniform_bucket_level_access = true 12 } 13 14 resource "google_dataproc_cluster" "etl" { 15 name = var.nome_etl 16 region = var.region 17 project = var.project 18 provider = google-beta 19 20 cluster_config { 21 staging_bucket = var.bucket_etl 22 temp_bucket = var.bucket_etl 23 24 gce_cluster_config { 25 zone = var.zone_etl 26 service_account = var.service_account 27 } 28 29 master_config { 30 num_instances = 1 31 machine_type = var.machine_etl 32 disk_config { 33 boot_disk_type = "pd-standard" 34 boot_disk_size_gb = 500 35 num_local_ssd = 0 36 } 37 } 38 39 worker_config { 40 num_instances = 0 41 } 42 43 software_config { 44 optional_components = ["JUPYTER"] 45 image_version = var.image_etl 46 override_properties = [47 "dataproc:dataproc.allow-zero-workers" = "true" 48] 49 } 50 51 endpoint_config { 52 enable_http_port_access = "true" 53 } 54 } 55 } 56 57 resource "google_storage_bucket_object" "bronze" { 58 name = "bronze/" 59 content = "Ingestão de Dados" 60 bucket = "\${google_storage_bucket.storage_bucket.name}" 61 } 62 63 resource "google_storage_bucket_object" "prata" { 64 name = "prata/" 65 content = "Dados Pré-Qualificados" 66 bucket = "\${google_storage_bucket.storage_bucket.name}" 67 } 68 69 resource "google_storage_bucket_object" "ouro" { 70 name = "ouro/" 71 content = "Dados Enriquecidos" 72 bucket = "\${google_storage_bucket.storage_bucket.name}" 73 } </pre>	<pre> 1 variable "project" { 2 default = "igti-mba-pa" 3 } 4 5 variable "key" { 6 default = "pa-igti.json" 7 } 8 9 variable "region" { 10 default = "us-east1" 11 } 12 13 variable "service_account" { 14 default = "Admin-igti@igti-mba-pa.iam.gserviceaccount.com" 15 } 16 17 variable "zone" { 18 default = "us-central1-c" 19 } 20 21 variable "location" { 22 default = "EU" 23 } 24 25 variable "bucket_name" { 26 default = "igti-pa" 27 } 28 29 variable "credentials" { 30 default = "pa-igti.json" 31 } 32 33 variable "nome_etl" { 34 default = "etl" 35 } 36 37 variable "bucket_etl" { 38 default = "igti-pa" 39 } 40 41 variable "machine_etl" { 42 default = "n2-standard-4" 43 } 44 45 variable "image_etl" { 46 default = "2.0-centos8" 47 } 48 49 variable "zone_etl" { 50 default = "us-east1-c" 51 } 52 53 54 </pre>
---	---

Com o *script* pronto, basta iniciar o ambiente no *Terraform* (comando *init*), verificar o plano de ação do *script* (comando *plan*) e, por final, aplicar o *script* para que ele o execute (comando *apply*). Assim, a infraestrutura explicada inicialmente neste tópico será criada no ambiente *GCP*. Lembrando que, caso precise refazer o ambiente, basta seguir novamente os comandos citados. Caso deseje deletar toda a estrutura criada, basta executar o comando *destroy* no *Terraform*.

Após termos a infraestrutura no *GCP* pronta, iniciamos o processo de criação do *script* de *pipeline* de dados. Esse *script* será dividido em partes, sendo cada parte um processo independente dentro do macroprocesso de *ETL*. Na primeira parte, que é o objetivo final deste Sprint, o *script* realizará as seguintes ações:

- Instalar e carregar as bibliotecas necessárias para a execução do *script*
- Realizar a conexão com o *Google Storage*
- *Download* dos dados públicos sobre a COVID 19 no Brasil
- *Download* dos dados públicos sobre o Censo Escolar 2019
- *Download* dos dados públicos sobre o Censo Escolar 2020
- Salvar os dados dentro da pasta Bronze no *bucket igti-pa*

O *script* em questão foi desenvolvido em linguagem *Python*, conforme já definido como premissa do projeto. Para isso, foi utilizada a plataforma do *Jupyter Notebook*, um aplicativo *web open-source* onde é possível compilar trechos de códigos de forma independente. Essa ferramenta foi instanciada no cluster *ETL* conforme evidenciado anteriormente. O *script* em questão está dividido em trechos, no qual cada um deles desempenha uma função específica dentro do processo de *ETL* como um todo, conforme já explicado no parágrafo anterior. O código desenvolvido pode ser observado na Figura 17:

Figura 17 - Código ETL1

ETL1 - SCRIPT DE PIPELINE DE DADOS

Script responsável por baixar os dados do CENSO ESCOLAR 2019, CENSO ESCOLAR 2020 e COVID19 e salvá-los no Cloud Storage

```
In [ ]: # Baixar e atualizar as bibliotecas necessárias
# Necessário executar apenas uma vez
!pip install --upgrade pandas
!pip install --upgrade numpy
!pip install --upgrade requests
!pip install --upgrade shutil
!pip install --upgrade gzip
!pip install --upgrade google-cloud-storage

In [ ]: # Importa as bibliotecas necessárias
import pandas as pd
import numpy as np
import zipfile
import gzip
import requests
from io import BytesIO
import os
import shutil
from google.cloud import storage

In [ ]: # Cria a conexão com o bucket do GCP e define o bucket a ser usado
storage_client = storage.Client()
bucket = storage_client.bucket('igti-pa')

In [ ]: # Define as variáveis do script - diretório de dados, url dos dados do censo escolar e do covid
dir_local = '/dados/'
url_covid = 'https://data.brasil.io/dataset/covid19/caso_full.csv.gz'
url_censo_2019 = 'https://download.inep.gov.br/microdados/microdados_educacao_basica_2019.zip'
url_censo_2020 = 'https://download.inep.gov.br/dados_abertos/microdados_censo_escolar_2020.zip'

In [ ]: # Cria o diretório temporário no qual serão salvos os arquivos localmente
try:
    os.mkdir(dir_local)
except OSError as e:
    print(e)
else:
    print("O diretório '{}' foi criado com sucesso!".format(dir_local))

In [ ]: # Realiza o download dos dados do COVID19
try:
    filename_covid = dir_local + url_covid.split("/")[-1]
    with open(filename_covid, "wb") as f:
        r = requests.get(url_covid)
        f.write(r.content)
    print("Dados do COVID19 baixados com sucesso!")
except:
    print("Ocorreu um erro ao baixar os dados do COVID19!")

In [ ]: # Realiza o upload dos dados do COVID19 para o bucket do GCP
try:
    blob_covid = bucket.blob('bronze/covid19.gz')
    blob_covid.upload_from_filename(filename_covid)
    print("O upload dos dados do COVID19 foi concluído com sucesso!")
except:
    print("Ocorreu um erro ao realizar o upload dos dados do COVID19!")

In [ ]: # Realiza o download dos dados do CENSO 2019
try:
    filebytes = BytesIO(requests.get(url_censo_2019).content)
    myzip = zipfile.ZipFile(filebytes)
    myzip.extractall(dir_local)
    print("Dados do CENSO 2019 baixados com sucesso!")
except:
    print("Ocorreu um erro ao baixar os dados do CENSO 2019!")

In [ ]: # Realiza o upload dos arquivos CSV do CENSO 2019 para o bucket do GCP
try:
    censo_2019 = dir_local + 'microdados_educacao_basica_2019/DADOS/'
    for diretorio, subpastas, arquivos in os.walk(censo_2019):
        for arquivo in arquivos:
            if '.CSV' in arquivo:
                blob = bucket.blob('bronze/censo_2019/' + arquivo)
                blob.upload_from_filename(diretorio + arquivo)
                print(arquivo + " carregado com sucesso no GCP")
    print("O upload dos dados do CENSO 2019 foi concluído com sucesso!")
except:
    print("Ocorreu um erro ao realizar o upload dos dados do CENSO 2019!")

In [ ]: # Realiza o download dos dados do CENSO 2020
try:
    filebytes = BytesIO(requests.get(url_censo_2020).content)
    myzip = zipfile.ZipFile(filebytes)
    myzip.extractall(dir_local)
    print("Dados do CENSO 2020 baixados com sucesso!")
except:
    print("Ocorreu um erro ao baixar os dados do CENSO 2020!")

In [ ]: # Realiza o upload dos arquivos CSV do CENSO 2020 para o bucket do GCP
try:
    censo_2020 = dir_local + 'microdados_educacao_basica_2020/DADOS/'
    for diretorio, subpastas, arquivos in os.walk(censo_2020):
        for arquivo in arquivos:
            if '.CSV' in arquivo:
                blob = bucket.blob('bronze/censo_2020/' + arquivo)
                blob.upload_from_filename(diretorio + arquivo)
                print(arquivo + " carregado com sucesso no GCP")
    print("O upload dos dados do CENSO 2020 foi concluído com sucesso!")
except:
    print("Ocorreu um erro ao realizar o upload dos dados do CENSO 2020!")

In [ ]: # Deleta o diretório temporário no qual estão salvos os arquivos baixados
try:
    shutil.rmtree(dir_local)
except OSError as e:
    print(e)
else:
    print("O diretório '{}' foi deletado com sucesso!".format(dir_local))
```

FIM DO SCRIPT ETL1

- Evidência dos resultados:

Com o *script* ETL1 pronto, foram executados todos os trechos do código de forma separada, no qual não foram encontrados erros no processo. Ou seja, todos os dados necessários para a continuação deste projeto foram baixados e armazenados no *Google Storage* (pasta Bronze) com sucesso. Segue as imagens evidenciando a execução de um dos trechos do *script* (Figura 18) e os dados armazenados no *bucket* como evidência dos resultados (Figura 19):

Figura 18 - Código ETL1 executado com sucesso

```
In [8]: # Realiza o download dos dados do CENSO 2020
try:
    filebytes = BytesIO (requests.get(url_censo_2020).content)
    myzip = zipfile.ZipFile(filebytes)
    myzip.extractall(dir_local)
    print("Dados do CENSO 2020 baixados com sucesso!")
except:
    print("Ocorreu um erro ao baixar os dados do CENSO 2020!")

Dados do CENSO 2020 baixados com sucesso!

In [9]: # Realiza o upload dos arquivos CSV do CENSO 2020 para o bucket do GCP
try:
    censo_2020 = dir_local + 'microdados_educacao_basica_2020/DADOS/'
    for diretorio, subpastas, arquivos in os.walk(censo_2020):
        for arquivo in arquivos:
            if '.CSV' in arquivo:
                blob = bucket.blob('bronze/censo_2020/'+arquivo)
                blob.upload_from_filename(diretorio+arquivo)
                print(arquivo + ' carregado com sucesso no GCP')
    print("O upload dos dados do CENSO 2020 foi concluído com sucesso!")
except:
    print("Ocorreu um erro ao realizar o upload dos dados do CENSO 2020!")

docentes_co.CSV carregado com sucesso no GCP
docentes_nordeste.CSV carregado com sucesso no GCP
docentes_norte.CSV carregado com sucesso no GCP
docentes_sudeste.CSV carregado com sucesso no GCP
docentes_sul.CSV carregado com sucesso no GCP
escolas.CSV carregado com sucesso no GCP
gestor.CSV carregado com sucesso no GCP
matricula_co.CSV carregado com sucesso no GCP
matricula_nordeste.CSV carregado com sucesso no GCP
matricula_norte.CSV carregado com sucesso no GCP
matricula_sudeste.CSV carregado com sucesso no GCP
matricula_sul.CSV carregado com sucesso no GCP
turmas.CSV carregado com sucesso no GCP
O upload dos dados do CENSO 2020 foi concluído com sucesso!
```

Figura 19 - Dados armazenados no Google Storage

igti-pa

Local

eu (várias regiões na União Europeia)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > igti-pa > bronze

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES

FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

Filtro Filtrar objetos e pastas

Mostrar dados excl

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões
<input type="checkbox"/>	censo_2019/	—	Pasta	—	—	—	—	—
<input type="checkbox"/>	censo_2020/	—	Pasta	—	—	—	—	—
<input type="checkbox"/>	covid19.gz	73,9 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—

igti-pa

Local

eu (várias regiões na União Europeia)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > igti-pa > bronze > censo_2019

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES

FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

Filtro Filtrar objetos e pastas

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões	Criptografia	Data de v
<input type="checkbox"/>	DOCENTES_CO.CSV	279,2 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	DOCENTES_NORDESTE.CSV	942 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	DOCENTES_NORTE.CSV	334 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	DOCENTES_SUDESTE.CSV	1,4 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	DOCENTES_SUL.CSV	586,8 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	ESCOLAS.CSV	106 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	GESTOR.CSV	40,7 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	MATRICULA_CO.CSV	952,7 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	MATRICULA_NORDESTE.CSV	3,6 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	MATRICULA_NORTE.CSV	1,2 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	MATRICULA_SUDESTE.CSV	4,6 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	MATRICULA_SUL.CSV	1,6 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	TURMAS.CSV	440,4 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—

igti-pa

Local

eu (várias regiões na União Europeia)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > igti-pa > bronze > censo_2020

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES

FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

Filtro Filtrar objetos e pastas

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões	Criptografia	Data de val
<input type="checkbox"/>	docentes_co.CSV	272,1 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	docentes_nordeste.CSV	894,7 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	docentes_norte.CSV	328,8 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	docentes_sudeste.CSV	1,4 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	docentes_sul.CSV	591,7 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	escolas.CSV	106,1 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	gestor.CSV	41,3 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	matricula_co.CSV	919,9 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	matricula_nordeste.CSV	3,4 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	matricula_norte.CSV	1,2 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	matricula_sudeste.CSV	4,5 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	matricula_sul.CSV	1,6 GB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—
<input type="checkbox"/>	turmas.CSV	429,7 MB	text/csv	29 de no...	Standard	29 de nov. de 20...	Não público	—	Google-managed key	—

2.1.2 Lições aprendidas

Ao final do Sprint 1, foi possível comprovar que o projeto pode ser realizado com as premissas e requisitos previamente determinados, no qual todas as execuções e resultados foram conquistados dentro do planejamento. Todo o planejamento inicial encontra-se dentro do cronograma previsto. Além disso, todos os objetivos e os resultados dos objetivos do Sprint 1 foram alcançados:

- Criar a conta e as conexões necessárias no *GCP*
- Criação da infraestrutura necessária no *GCP*
- Criar a *pipeline* de *ETL* dos dados

Durante o planejamento da Sprint 1, foi necessária realizar uma alteração na infraestrutura prevista. Inicialmente, o planejamento era orquestrar as *pipelines* de dados no *software open-source Apache Airflow*, instalado e configurado no *Google Composer*. Contudo, por uma questão de custos e simplificação de projeto, essa ideia foi descartada.

O custo da utilização do *Google Composer* com o *Apache Airflow* instalado é de US\$ 0,10 por hora, enquanto ao do *cluster* no *Dataproc* é de US\$ 0,04 por hora. Além disso, levando em consideração que esse *script* será executado apenas uma vez, não há necessidade nem de agendamentos de tarefas nem de orquestração de grandes atividades. Portanto, o *Apache Airflow* acabou se tornando uma ferramenta inapropriada para o projeto, sendo substituída pela execução do *script* diretamente na ferramenta *Jupyter Notebook*.

2.2 Sprint 2

Neste Sprint, o principal objetivo é criar o *script* de *pipeline* de dados para efetuar as transformações necessárias nos dados coletados no Sprint 1, preparando esses dados para serem disponibilizados posteriormente para consultas. Para isso, esse *script* foi dividido em três *scripts* distintos: o primeiro (ETL1) já foi criado e explicado no Sprint 1; o segundo (ETL2) vai ser responsável por ler, transformar e armazenar os dados referentes ao Covid19 no Brasil; o terceiro (ETL3) vai ser responsável por ler, transformar e armazenar os dados do Censo Escolar 2019 e do Censo Escolar 2020.

2.2.1 Solução

- Evidência do planejamento:

Figura 20 - Backlog de Produto



- Evidência da execução de cada requisito:

O *script* ETL2 tem como principal objetivo realizar o processo de *ETL* para os dados coletados sobre a Covid19. Nele, foi utilizada novamente a linguagem *Python* usando principalmente a biblioteca *PANDAS*, conforme já determinado nas premissas deste projeto. Esse *script* se divide em duas partes distintas.

Na primeira parte, é efetuada a leitura dos dados coletados sobre o Covid19, salvando o mesmo em um *Dataframe*. Após isso, são realizadas as devidas transformações nesses dados, sendo elas: a exclusão de colunas não relevantes para o estudo; o filtro dos dados apenas até o dia 31/12/2020; a renomeação das colunas para um melhor entendimento; as mudanças nos tipos de dados de cada coluna. Por fim, o *Dataframe* resultante é armazenado na zona Prata do *Cloud Storage* em formato *PARQUET*. Podemos visualizar as etapas descritas na Figura 21:

Figura 21 - ETL2 (Primeira Parte)

ETL2 - SCRIPT DE PIPELINE DE DADOS

Script responsável por buscar os dados coletados do COVID19 que estão no Cloud Storage, transformar eles para atender as necessidades do projeto e, por fim, salvá-los na zona de ouro do Cloud Storage em formato PARQUET

PRIMEIRA PARTE

```
In [ ]: # Importa as bibliotecas necessárias
import pandas as pd
import numpy as np
import gzip
import requests
import os
import datetime

In [ ]: # Salva os dados contidos no arquivo covid19.gz em um dataframe chamado DADOS
dados = pd.read_csv("gs://igti-pa/bronze/covid19.gz", compression='gzip', sep=',')

In [ ]: # Verifica as informações básicas do dataframe criado
dados.info()

In [ ]: # Exclui as colunas que não serão utilizadas
dados.drop(['estimated_population', 'estimated_population_2019', 'is_last', 'is_repeated',
            'last_available_confirmed_per_100k_inhabitants', 'order_for_place',
            'last_available_date', 'last_available_death_rate'], axis=1, inplace=True)

In [ ]: # Filtra os dados apenas até 2020 - após 2020 não há necessidade de análise
dados = dados[dados['date'] <= '2020-12-31']

In [ ]: # Existem algumas coletas de dados que não tiveram identificação de cidade, então ficou salvo no campo como NaN
# Altera todos os valores NaN para 0 (int)
dados['city_ibge_code'].fillna(0, inplace=True)

In [ ]: # Altera o código IBGE primeiro para INT para retirar o (.) e, após isso, altera o campo para STR
dados = dados.astype({"city_ibge_code": int})
dados = dados.astype({"city_ibge_code": str})

In [ ]: # Altera o campo do ano+semana da pandemia para STR
dados = dados.astype({"epidemiological_week": str})

In [ ]: # Renomeia as colunas para português para melhor entendimento
dados = dados.rename(columns={'city': 'cidade', 'city_ibge_code': 'codigo_ibge', 'date': 'data',
                              'epidemiological_week': 'ano_semana_pandemia',
                              'last_available_confirmed': 'qtd_ultimos_casos_disponiveis',
                              'last_available_deaths': 'qtd_ultimas_mortes_disponiveis',
                              'state': 'estado', 'new_confirmed': 'novos_casos_confirmados',
                              'new_deaths': 'novas_mortes_confirmados'})

In [ ]: # Verifica novamente as informações básicas do dataframe após as alterações efetuadas
dados.info()

In [ ]: # Mostra os primeiros três registros do dataframe transformado
dados.head(3)

In [ ]: dados.to_parquet('gs://igti-pa/prata/dados_covid19.parquet')
```

Na segunda parte, é feita a leitura dos dados salvos na primeira parte do *script* e realizado o devido tratamento deles para que seja possível analisar os dados do Covid19 nas perspectivas de Cidades e de Estados, sendo cada um desses um novo *Dataframe*. Após isso, cada um deles é armazenado na zona Ouro do *Cloud Storage* em formato *PARQUET*. Podemos verificar essas etapas na Figura 22:

Figura 22 - ETL2 (Segunda Parte)

```

SEGUNDA PARTE

In [ ]: # Faz a Leitura dos dados do COVID19 no bucket em formato PARQUET
dados = pd.read_parquet('gs://igti-pa/prata/dados_covid19.parquet')

In [ ]: # Cria um novo dataframe chamado ESTADOS e salva apenas as informações conforme o campo "place_type"
estados = dados[dados['place_type'] == 'state']
estados = estados.reset_index(drop=True)

In [ ]: # Cria um novo dataframe chamado CIDADES e salva apenas as informações conforme o campo "place_type"
cidades = dados[dados['place_type'] == 'city']
cidades = cidades.reset_index(drop=True)

In [ ]: # Exclui as colunas que não serão utilizadas
estados.drop(['cidade', 'place_type'], axis=1, inplace=True)

In [ ]: # Exclui as colunas que não serão utilizadas
cidades.drop(['place_type'], axis=1, inplace=True)

In [ ]: # Mostra os primeiros três registros do dataframe ESTADOS transformado
estados.head(3)

In [ ]: # Mostra os primeiros três registros do dataframe CIDADES transformado
cidades.head(3)

In [ ]: # Verifica as informações básicas do dataframe CIDADES após as alterações efetuadas
estados.info()

In [ ]: # Verifica as informações básicas do dataframe CIDADES após as alterações efetuadas
cidades.info()

In [ ]: # Salva o novo dataframe ESTADOS no bucket do GCP
estados.to_parquet('gs://igti-pa/ouro/estados_covid19.parquet')

In [ ]: # Salva o novo dataframe CIDADES no bucket do GCP
cidades.to_parquet('gs://igti-pa/ouro/cidades_covid19.parquet')

In [ ]: # Lê as primeiras três linhas do arquivo ESTADOS.parquet criado para verificar a integridade do arquivo
pd.read_parquet('gs://igti-pa/ouro/estados_covid19.parquet')

In [ ]: # Lê as primeiras três linhas do arquivo CIDADES.parquet criado para verificar a integridade do arquivo
pd.read_parquet('gs://igti-pa/ouro/cidades_covid19.parquet')

```

FIM DO SCRIPT ETL2

Já o *script* ETL3 tem como objetivo principal realizar o processo de *ETL* para os dados coletados sobre o Censo Escolar 2019 e o Censo Escolar 2020. Nele, foi utilizada novamente a linguagem *Python*. Contudo, foi utilizada a biblioteca *PYSPARK*, utilizando o *SPARK* no cluster *ETL* (já criado e configurado no Sprint 1). Foi optado a utilização do *SPARK* pois há uma grande quantidade de registros coletados, o que tornaria inviável a transformação deles usando as bibliotecas e o cluster padrão. Esse *script* foi dividido em cinco partes, porém, apenas as três primeiras foram criadas no Sprint 2. As outras duas partes serão criadas e detalhadas no Sprint 3.

Na primeira parte foram definidos todas as variáveis gerais do *script*, ou seja, todos os parâmetros do *SPARK* que serão utilizados na hora da leitura de cada um dos arquivos do Censo Escolar e todos os campos que não serão utilizados para esse projeto. Vale ressaltar que o dicionário de dados de ambos censos são similares, no qual as poucas diferenças puderam ser tratadas de uma forma geral. Portanto, essas definições vão servir tanto para os anos de 2019 quanto para 2020. Na Figura 23 podemos constatar todas as variáveis que serão utilizadas nesse *script*:

Figura 23 - ETL3 (Primeira Parte)

ETL3 - SCRIPT DE PIPELINE DE DADOS

Script responsável por buscar os dados coletados do CENSO 2019 e CENSO 2020 que estão no Cloud Storage, transformar eles para atender as necessidades do projeto e, por fim, salvá-los na zona de outro do Cloud Storage em formato PARQUET

```
In [ ]: # Importa as bibliotecas necessárias
from pyspark.sql import SparkSession
import pyspark.sql.types as t
import pyspark.sql.functions as f

In [ ]: # Obtem ou Cria uma Spark Session
spark = SparkSession.builder.getOrCreate()

DEFINIÇÕES GERAIS

In [ ]: # Define todas as colunas dos dados dos docentes que não serão utilizadas neste projeto
cols_docente_drop = ('NU_ANO_CENSO', 'ID_DOCENTE', 'NU_IDADE', 'IN_BAIXA_VISAO', 'IN_CEGUEIRA', 'IN_DEF_AUDITIVA', 'IN_DEF_FISICA',
                    'TP_NORMAL_MAGISTERIO', 'TP_SITUACAO_CURSO_1', 'CO_AREA_CURSO_1', 'CO_CURSO_1', 'IN_LICENCIATURA_1', 'IN_COM_PED',
                    'TP_SITUACAO_CURSO_2', 'CO_AREA_CURSO_2', 'CO_CURSO_2', 'IN_LICENCIATURA_2', 'IN_COM_PEDAGOGICA_2', 'NU_ANO_INIC',
                    'CO_CURSO_3', 'IN_LICENCIATURA_3', 'IN_COM_PEDAGOGICA_3', 'NU_ANO_INICIO_3', 'NU_ANO_CONCLUSAO_3', 'TP_TIPO_IES',
                    'IN_DISC_CIENTIAS', 'IN_DISC_LINGUA_PORTUGUESA', 'IN_DISC_LINGUA_INGLES', 'IN_DISC_LINGUA_ESPANHOL', 'IN_DISC_L',
                    'IN_DISC_EDUCACAO_FISICA', 'IN_DISC_HISTORIA', 'IN_DISC_GEOGRAFIA', 'IN_DISC_FILOSOFIA', 'IN_DISC_ENSINO_RELIG',
                    'IN_DISC_INFORMATICA_COMPUTACAO', 'IN_DISC_PROFISSIONALIZANTE', 'IN_DISC_ATENDIMENTO_ESPECIAIS', 'IN_DISC_DIVE',
                    'IN_ESPECIFICO_ANOS_FINALS', 'IN_ESPECIFICO_ENS_MEDIO', 'IN_ESPECIFICO_EJA', 'IN_ESPECIFICO_ED_ESPECIAL', 'IN_E',
                    'IN_ESPECIFICO_DIV_SEXUAL', 'IN_ESPECIFICO_DIR_ADOLESC', 'IN_ESPECIFICO_AFRO', 'IN_ESPECIFICO_OUTROS', 'IN_ESPE',
                    'CO_REGIAO', 'CO_MESORREGIAO', 'CO_MICRORREGIAO', 'CO_UF', 'CO_MUNICIPIO', 'CO_DISTRITO', 'TP_DEPENDENCIA', 'TP_LO',
                    'IN_MANT_ESCOLA_PRIVADA_SIST_S', 'IN_MANT_ESCOLA_PRIVADA_S_FINS', 'TP_REGULAMENTACAO', 'TP_LOCALIZACAO_DIFEREN',
                    'CO_AREA_COMPL_PEDAGOGICA_2', 'CO_AREA_COMPL_PEDAGOGICA_3', 'IN_DISC_PORT_SEGUNDA_LINGUA', 'IN_DISC_ESTAGIO_SU

In [ ]: # Define todas as colunas dos dados das matrículas que não serão utilizadas neste projeto
cols_matricula_drop = ('NU_IDADE', 'ID_ALUNO', 'NU_DURACAO_TURMA', 'NU_DUR_ATIV_COMP_MESMA_REDE', 'NU_DUR_ATIV_COMP_OUTRAS_REDES',
                    'IN_DEF_AUDITIVA', 'IN_SURDOCEGUEIRA', 'IN_DEF_FISICA', 'IN_DEF_INTELLECTUAL', 'IN_DEF_MULTIPLA', 'IN_AUTISMO',
                    'IN_SINDROME_ASPIRGER', 'IN_SINDROME_RETT', 'IN_SUPERDOTACAO', 'IN_RECURSO_LEDOR', 'IN_RECURSO_TRANSCRITAO',
                    'IN_RECURSO_INTERPRETE', 'IN_RECURSO_LIBRAS', 'IN_RECURSO_LABIAL', 'IN_RECURSO_BRILLE',
                    'TP_INGRESSO_FEDERATS', 'IN_REGULAR', 'IN_EJA', 'IN_PROFISSIONALIZANTE', 'CO_CURSO_EDUC_PROFISSIONAL',
                    'CO_DISTRITO', 'TP_DEPENDENCIA', 'TP_LOCALIZACAO', 'TP_CATEGORIA_ESCOLA_PRIVADA', 'IN_CONVENIADA_PP',
                    'IN_MANT_ESCOLA_PRIVADA_EMP', 'IN_MANT_ESCOLA_PRIVADA_ONG', 'IN_MANT_ESCOLA_PRIVADA_SIND',
                    'TP_LOCALIZACAO_DIFERENCIADA', 'IN_EDUCACAO_INDIGENA', 'NU_ANO_CENSO', 'IN_RECURSO_AMPLIADA_18',
                    'IN_AEE_LINGUA_PORTUGUESA', 'IN_AEE_INFORMATICA_ACESSIVEL', 'IN_AEE_BRILLE', 'IN_AEE_CAA',
                    'IN_AEE_DESEN_COGNITIVO', 'IN_AEE_MOBILIDADE', 'IN_TRANSP_BICICLETA', 'IN_TRANSP_MICRO_ONIBUS',
                    'IN_TRANSP_EMBAR_ATES', 'IN_TRANSP_EMBAR_SA15', 'IN_TRANSP_EMBAR_15A35', 'IN_TRANSP_EMBAR_35',

In [ ]: # Define todas as colunas dos dados das turmas que não serão utilizadas neste projeto
cols_turma_drop = ('TX_HR_INICIAL', 'TX_MI_INICIAL', 'NU_DURACAO_TURMA', 'IN_REGULAR', 'IN_EJA', 'IN_PROFISSIONALIZANTE',
                    'IN_DIA_SEMANA_TERCA', 'IN_DIA_SEMANA_QUARTA', 'IN_DIA_SEMANA_QUINTA', 'IN_DIA_SEMANA_SEXTA',
                    'CO_TIPO_ATIVIDADE_4', 'CO_TIPO_ATIVIDADE_5', 'CO_TIPO_ATIVIDADE_6', 'IN_BRILLE', 'IN_RECURSOS_BAIXA_VISAO',
                    'IN_INFORMATICA_ACESSIVEL', 'IN_PORT_ESCRITA', 'IN_AUTONOMIA_ESCOLAR', 'IN_DISC_QUIMICA', 'IN_DISC_FISICA',
                    'IN_DISC_LINGUA_INGLES', 'IN_DISC_LINGUA_ESPANHOL', 'IN_DISC_LINGUA_FRANCES', 'IN_DISC_LINGUA_OUTRA',
                    'IN_DISC_FILOSOFIA', 'IN_DISC_ENSINO_RELIGIOSO', 'IN_DISC_ESTUDOS_SOCIAIS', 'IN_DISC SOCIOLOGIA',
                    'IN_DISC_ATENDIMENTO_ESPECIAIS', 'IN_DISC_DIVER_SOCTO_CULTURAL', 'IN_DISC_LIBRAS', 'IN_DISC_PEDAGOGICAS',
                    'TP_DEPENDENCIA', 'TP_CATEGORIA_ESCOLA_PRIVADA', 'IN_CONVENIADA_PP', 'TP_CONVENIO_PODER_PUBLICO',
                    'IN_MANT_ESCOLA_PRIVADA_SIST_S', 'IN_MANT_ESCOLA_PRIVADA_S_FINS', 'TP_REGULAMENTACAO', 'TP_LOCALIZACAO_DIFERENCI',
                    'IN_DISC_ESTAGIO_SUPERVISIONADO', 'IN_MANT_ESCOLA_PRIVADA_OSCIP', 'IN_MANT_ESCOLA_PRIV_ONG_OSCIP', 'NU_ANO_CENSO

In [ ]: # Define todas as colunas dos dados das escolas que não serão utilizadas neste projeto
cols_escola_drop = ('NU_ANO_CENSO', 'TP_FALTANTE', 'IN_MANT_ESCOLA_PRIVADA_EMP', 'IN_MANT_ESCOLA_PRIVADA_ONG', 'IN_MANT_ESCOLA_PRIVA',
                    'CO_ESCOLA_SEDE_VINCULADA', 'CO_TES_OFERTANTE', 'TP_REGULAMENTACAO', 'IN_LOCAL_FUNC_PREDIO_ESCOLAR', 'TP_OCUPACA',
                    'IN_COMUM_PRE', 'IN_COMUM_FUND_AT', 'IN_COMUM_FUND_AF', 'IN_COMUM_MEDIO_MEDIO', 'IN_COMUM_MEDIO_INTEGRADO', 'IN_C',
                    'IN_ESP_EXCLUSIVA_FUND_AF', 'IN_ESP_EXCLUSIVA_MEDIO_MEDIO', 'IN_ESP_EXCLUSIVA_MEDIO_INTEGR', 'IN_ESP_EXCLUSIVA',
                    'IN_ESP_EXCLUSIVA_EJA_FUND', 'IN_ESP_EXCLUSIVA_EJA_MEDIO', 'IN_ESP_EXCLUSIVA_EJA_PROF', 'IN_COMUM_PROF',
                    'IN_ORGAO_CRENTO_ESTUDANTIL', 'IN_ORGAO_OUTROS', 'IN_ORGAO_NENHUM', 'IN_RESERVA_PPI', 'IN_RESERVA_RENDA',
                    'CO_LINGUA_INDIGENA_2', 'CO_LINGUA_INDIGENA_3', 'IN_MATERIAL_PED_MULTIMIDIA', 'IN_MATERIAL_PED_INFANTIL',
                    'IN_MATERIAL_PED_ARTISTICAS', 'IN_MATERIAL_PED_DESPORTIVA', 'IN_MATERIAL_PED_INDIGENA', 'ID_MATRICULA',
                    'IN_FUNDAMENTAL_CICLOS', 'IN_GRUPOS_NAO_SERIADOS', 'IN_MODULOS', 'IN_FORMACAO_ALTERNANCIA')

In [ ]: # Define os parâmetros OPTIONS usados em todas as Leituras de dados
config_file = {
    "header": True,
    "inferSchema": True,
    "delimiter": ";"
}
```

Na segunda parte é efetuada a leitura e a primeira parte da transformação dos dados do Censo Escolar 2019. Nessa etapa, é feita a leitura de cada um dos arquivos CSV coletados, sendo quatro deles principais: docentes, matrículas (alunos), escolas e turmas. Após isso, são realizadas as devidas transformações nesses dados, sendo elas: a exclusão de colunas não relevantes para o estudo e o filtro dos dados apenas para os brasileiros natos (foco do projeto). Ocorre apenas uma peculiaridade para os dados relativo aos docentes e matrículas, no qual é necessário fazer a união dos dados, já que os mesmos foram disponibilizados de forma separada por região. Por fim, os dados tratados são armazenados na zona Prata do *Cloud Storage* em formato *PARQUET*. Abaixo, temos como exemplo a leitura e transformação dos dados dos docentes (Figura 24) e das turmas (Figura 25) do Censo Escolar 2019:

Figura 24 - ETL3 (Segunda Parte - Exemplo 1)

PRIMEIRA PARTE - DADOS DO CENSO 2019	Continuação...
<pre>In []: ##### DADOS DOS DOCENTES - 2019 In []: # Salva os dados referentes aos docentes - centroeste docentes_centroeste_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/DOCENTES_CO.CSV")) In []: # Salva os dados referentes aos docentes - nordeste docentes_nordeste_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/DOCENTES_NORDESTE.CSV")) In []: # Salva os dados referentes aos docentes - norte docentes_norte_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/DOCENTES_NORTE.CSV")) In []: # Salva os dados referentes aos docentes - sudeste docentes_sudeste_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/DOCENTES_SUDESTE.CSV")) In []: # Salva os dados referentes aos docentes - sul docentes_sul_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/DOCENTES_SUL.CSV")) In []: # Dropa todas as colunas que não serão utilizadas docentes_centroeste_2019 = docentes_centroeste_2019.drop(*cols_docente_drop) docentes_nordeste_2019 = docentes_nordeste_2019.drop(*cols_docente_drop) docentes_norte_2019 = docentes_norte_2019.drop(*cols_docente_drop) docentes_sudeste_2019 = docentes_sudeste_2019.drop(*cols_docente_drop) docentes_sul_2019 = docentes_sul_2019.drop(*cols_docente_drop)</pre>	<pre>In []: # Faz a união de todos os dados coletados dos docentes docentes_2019 = docentes_centroeste_2019.unionAll(docentes_nordeste_2019).unionAll(docentes_norte_2019).unionAll(docentes_sudeste_2019).unionAll(docentes_sul_2019) In []: # Filtra os dados apenas com os Brasileiros Natos (definido no projeto) matriculas_2019 = matriculas_2019.filter("TP_NACIONALIDADE == '1'") In []: # Dropa as tabelas referente a nacionalidade após a execução do filtro matriculas_2019 = matriculas_2019.drop("TP_NACIONALIDADE", "CO_PAIS_ORIGEM") In []: # Salva os dados das MATRICULAS 2019 na pasta PRATA do bucket como arquivo PARQUET matriculas_2019.write.parquet("gs://igti-pa/prata/matriculas_2019.parquet") In []: ##### DADOS DAS TURMAS - 2019 In []: # Salva os dados referentes as turmas turmas_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/TURMAS.CSV")) In []: # Dropa todas as colunas que não serão utilizadas turmas_2019 = turmas_2019.drop(*cols_turma_drop) In []: # Salva os dados das TURMAS 2019 na pasta PRATA do bucket como arquivo PARQUET turmas_2019.write.parquet("gs://igti-pa/prata/turmas_2019.parquet") In []: ##### DADOS DAS ESCOLAS - 2019 In []: # Salva os dados referentes as escolas escolas_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/ESCOLAS.CSV")) In []: # Dropa todas as colunas que não serão utilizadas escolas_2019 = escolas_2019.drop(*cols_escola_drop) In []: # Salva os dados das ESCOLAS 2019 na pasta PRATA do bucket como arquivo PARQUET escolas_2019.write.parquet("gs://igti-pa/prata/escolas_2019.parquet")</pre>

Figura 25 - ETL3 (Segunda Parte - Exemplo 2)

<pre>In []: ##### DADOS DAS TURMAS - 2019 In []: # Lê os dados referentes as turmas turmas_2019 = (spark .read .format('csv') .options(**config_file) .load("gs://igti-pa/bronze/censo_2019/TURMAS.CSV")) In []: # Dropa todas as colunas que não serão utilizadas turmas_2019 = turmas_2019.drop(*cols_turma_drop) In []: # Salva os dados das TURMAS 2019 na pasta PRATA do bucket como arquivo PARQUET turmas_2019.write.parquet("gs://igti-pa/prata/turmas_2019.parquet")</pre>
--

Por fim, na terceira parte é efetuada a leitura e a primeira parte da transformação dos dados do Censo Escolar 2020. Essa transformação ocorre de maneira idêntica aos dados coletados de 2019. Ou seja, o *script* é basicamente o mesmo, apenas trocando os arquivos que serão feitas as leituras. Na Figura 26 é possível observar a leitura e transformação dos dados das turmas e das escolas do Censo Escolar 2020:

Figura 26 - ETL3 (Terceira Parte)

```
In [ ]: # Lê os dados referentes as turmas
turmas_2020 = (
    spark
    .read
    .format('csv')
    .options(**config_file)
    .load("gs://igti-pa/bronze/centro_2020/turmas.CSV")
)

In [ ]: # Dropa todas as colunas que não serão utilizadas
turmas_2020 = turmas_2020.drop(*cols_turma_drop)

In [ ]: # Salva os dados das TURMAS 2019 na pasta PRATA do bucket como arquivo PARQUET
turmas_2020.write.parquet('gs://igti-pa/prata/turmas_2020.parquet')

In [ ]: ##### DADOS DAS ESCOLAS - 2020

In [ ]: # Lê os dados referentes as escolas
escolas_2020 = (
    spark
    .read
    .format('csv')
    .options(**config_file)
    .load("gs://igti-pa/bronze/centro_2020/escolas.CSV")
)

In [ ]: # Dropa todas as colunas que não serão utilizadas
escolas_2020 = escolas_2020.drop(*cols_escola_drop)

In [ ]: # Salva os dados das ESCOLAS 2019 na pasta PRATA do bucket como arquivo PARQUET
escolas_2020.write.parquet('gs://igti-pa/prata/escolas_2020.parquet')
```

- Evidência dos resultados:

Com o *script* ETL2 pronto, foram executados todos os trechos de códigos da parte um e, posteriormente, da parte dois. Com isso, todos os dados coletados sobre a Covid19 no Brasil foram devidamente tratados e armazenados no *Google Storage* (pasta Prata) e, após isso, foram divididos por Cidades e Estados e salvos novamente no *Google Storage* (pasta Ouro) com sucesso.

Na Figura 27 podemos visualizar o trecho do código responsável por filtrar os dados nulos, transformar os campos nos tipos adequados, renomear as colunas e mostrar o resultado dessas transformações. Na Figura 28 podemos observar o trecho do código responsável por dividir os dados da Covid19 entre Cidades e Estados e deletar os campos que não serão mais utilizados. Por fim, a Figura 29 mostra todos esses dados tratados salvos dentro da estrutura do *Google Storage* (pastas Prata e Ouro):

Figura 27 - ETL2 (Primeira Parte) executado com sucesso

```
In [7]: # Existem algumas coletas de dados que não tiveram identificação de cidade, então ficou salvo no campo como NaN
# Altera todos os valores NaN para 0 (int)
dados['city_ibge_code'].fillna(0, inplace=True)

In [8]: # Altera o código IBGE primeiro para INT para retirar o (.) e, após isso, altera o campo para STR
dados = dados.astype({'city_ibge_code': int})
dados = dados.astype({'city_ibge_code': str})

In [9]: # Altera o campo do ano+semana da pandemia para STR
dados = dados.astype({'epidemiological_week': str})

In [10]: # Renomeia as colunas para português para melhor entendimento
dados = dados.rename(columns={'city': 'cidade', 'city_ibge_code': 'codigo_ibge', 'date': 'data',
                              'epidemiological_week': 'ano_semana_pandemia',
                              'last_available_confirmed': 'qtd_ultimos_casos_disponiveis',
                              'last_available_deaths': 'qtd_ultimas_mortes_disponiveis',
                              'state': 'estado', 'new_confirmed': 'novos_casos_confirmados',
                              'new_deaths': 'novas_mortes_confirmados'})

In [11]: # Verifica novamente as informações básicas do dataframe após as alterações efetuadas
dados.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1320832 entries, 0 to 3144339
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   cidade                1312890 non-null object
1   codigo_ibge           1320832 non-null object
2   data                  1320832 non-null object
3   ano_semana_pandemia   1320832 non-null object
4   qtd_ultimos_casos_disponiveis  1320832 non-null int64
5   qtd_ultimas_mortes_disponiveis  1320832 non-null int64
6   place_type            1320832 non-null object
7   estado                1320832 non-null object
8   novos_casos_confirmados  1320832 non-null int64
9   novas_mortes_confirmados  1320832 non-null int64
dtypes: int64(4), object(6)
memory usage: 110.8+ MB
```

Figura 28 - ETL2 (Segunda Parte) executado com sucesso

```
In [15]: # Cria um novo dataframe chamado ESTADOS e salva apenas as informações conforme o campo "place_type"
estados = dados[dados['place_type'] == 'state']
estados = estados.reset_index(drop=True)

In [16]: # Cria um novo dataframe chamado CIDADES e salva apenas as informações conforme o campo "place_type"
cidades = dados[dados['place_type'] == 'city']
cidades = cidades.reset_index(drop=True)

In [17]: # Exclui as colunas que não serão utilizadas
estados.drop(['cidade', 'place_type'], axis=1, inplace=True)

In [18]: # Exclui as colunas que não serão utilizadas
cidades.drop(['place_type'], axis=1, inplace=True)

In [21]: # Verifica as informações básicas do dataframe CIDADES após as alterações efetuadas
estados.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7942 entries, 0 to 7941
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   codigo_ibge           7942 non-null  object
1   data                  7942 non-null  object
2   ano_semana_pandemia   7942 non-null  object
3   qtd_ultimos_casos_disponiveis  7942 non-null  int64
4   qtd_ultimas_mortes_disponiveis  7942 non-null  int64
5   estado                7942 non-null  object
6   novos_casos_confirmados  7942 non-null  int64
7   novas_mortes_confirmados  7942 non-null  int64
dtypes: int64(4), object(4)
memory usage: 496.5+ KB

In [22]: # Verifica as informações básicas do dataframe CIDADES após as alterações efetuadas
cidades.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1312890 entries, 0 to 1312889
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   cidade                1312890 non-null object
1   codigo_ibge           1312890 non-null object
2   data                  1312890 non-null object
3   ano_semana_pandemia   1312890 non-null object
4   qtd_ultimos_casos_disponiveis  1312890 non-null int64
5   qtd_ultimas_mortes_disponiveis  1312890 non-null int64
6   estado                1312890 non-null object
7   novos_casos_confirmados  1312890 non-null int64
8   novas_mortes_confirmados  1312890 non-null int64
dtypes: int64(4), object(5)
memory usage: 90.1+ MB
```


Figura 29 - Dados sobre o Covid19 salvos no *Cloud Storage* (Pastas Prata e Ouro)

igti-pa

Local

eu (várias regiões na União Europeia)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > igti-pa > prata

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES


FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

Filtro Filtrar objetos e pastas

Mostrar dados excluídos

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões
<input type="checkbox"/>	 dados_covid19.parquet	10,6 MB	application/octet-stream	4 de dez...	Standard	4 de dez. de 202...	Não público	—

igti-pa

Local

eu (várias regiões na União Europeia)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > igti-pa > ouro

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES



FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

Filtro Filtrar objetos e pastas

Mostrar dados excluídos

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões
<input type="checkbox"/>	 cidades_covid19.parquet	4,8 MB	application/octet-stream	4 de dez...	Standard	4 de dez. de 202...	Não público	—
<input type="checkbox"/>	 estados_covid19.parquet	119,2 KB	application/octet-stream	4 de dez...	Standard	4 de dez. de 202...	Não público	—

Por fim, com o *script* ETL3 parcialmente pronto, foram executados todos os trechos de códigos da parte um (definições gerais), parte dois (Censo Escolar 2019) e, posteriormente, da parte três (Censo Escolar 2020). Com isso, todos os dados coletados sobre os censos foram devidamente tratados e armazenados no *Google Storage* (pasta Prata) com sucesso.

A Figura 30 exibe o trecho do *script* responsável por ler os dados das matrículas (Censo Escolar 2019), unificar os dados de todas as regiões, excluir as colunas que não serão utilizadas, filtrar os dados apenas para brasileiros natos ou naturalizados e armazenar os dados tratados no *Google Storage* (pasta Prata) em formato *PARQUET*. Já a Figura 31 mostra o trecho do *script* responsável por ler os dados parcialmente tratados das turmas (Censo Escolar 2020), deletar as colunas que não serão utilizadas e armazenar os dados tratados no *Google Storage* (pasta Prata) em formato *PARQUET*. Por fim, a Figura 32 apresenta todos os dados dos Censos Escolares 2019 e 2020 previamente tratados dentro da pasta Prata no *Google Storage*.

Figura 30 - ETL3 (Segunda Parte) executado com sucesso

```
In [23]: # Lê os dados referentes as matrículas - sudeste
matriculas_sudeste_2019 =(
    spark
    .read
    .format('csv')
    .options(**config_file)
    .load("gs://igti-pa/bronze/censo_2019/MATRICULA_SUDESTE.CSV")
)

In [24]: # Lê os dados referentes as matrículas - sul
matriculas_sul_2019 =(
    spark
    .read
    .format('csv')
    .options(**config_file)
    .load("gs://igti-pa/bronze/censo_2019/MATRICULA_SUL.CSV")
)

In [25]: # Dropa todas as colunas que não serão utilizadas de todos os dados coletados até então
matriculas_centroeste_2019 = matriculas_centroeste_2019.drop(*cols_matricula_drop)
matriculas_nordeste_2019 = matriculas_nordeste_2019.drop(*cols_matricula_drop)
matriculas_norte_2019 = matriculas_norte_2019.drop(*cols_matricula_drop)
matriculas_sudeste_2019 = matriculas_sudeste_2019.drop(*cols_matricula_drop)
matriculas_sul_2019 = matriculas_sul_2019.drop(*cols_matricula_drop)

In [26]: # Faz a união de todos os dados coletados das matrículas
matriculas_2019 = matriculas_centroeste_2019.unionAll(matriculas_nordeste_2019).unionAll(matriculas_norte_2019).unionAll(matriculas_sudeste_2019).unionAll(matriculas_sul_2019)

In [27]: # Filtra os dados apenas com os Brasileiros Natos (definido no projeto)
matriculas_2019 = matriculas_2019.filter('TP_NACIONALIDADE == "1"')

In [28]: # Dropa as tabelas referente a nacionalidade após a execução do filtro
matriculas_2019 = matriculas_2019.drop('TP_NACIONALIDADE', 'CO_PAIS_ORIGEM')

In [29]: # Salva os dados das MATRÍCULAS 2019 na pasta PRATA do bucket como arquivo PARQUET
matriculas_2019.write.parquet('gs://igti-pa/prata/matriculas_2019.parquet')
```

Figura 31 - ETL3 (Terceira Parte) executado com sucesso

```
In [47]: # Lê os dados referentes as turmas
turmas_2020 =(
    spark
    .read
    .format('csv')
    .options(**config_file)
    .load("gs://igti-pa/bronze/censo_2020/turmas.CSV")
)

In [48]: # Dropa todas as colunas que não serão utilizadas
turmas_2020 = turmas_2020.drop(*cols_turma_drop)

In [49]: # Salva os dados das TURMAS 2019 na pasta PRATA do bucket como arquivo PARQUET
turmas_2020.write.parquet('gs://igti-pa/prata/turmas_2020.parquet')
```

Figura 32 - Dados sobre o Censo 2019/2020 salvos no *Cloud Storage* (Pasta Prata)

igti-pa

Local	Classe de armazenamento	Acesso público	Proteção
eu (várias regiões na União Europeia)	Standard	Não público	Nenhum

OBJETOS CONFIGURAÇÃO PERMISSÕES PROTEÇÃO CICLO DE VIDA

Intervalos > igti-pa > prata

FAZER UPLOAD DE ARQUIVOS CARREGAR PASTA CRIAR PASTA GERENCIAR RETENÇÕES FAZER O DOWNLOAD EXCLUIR

Filtrar apenas pelo prefixo do nome **Filtro** Filtrar objetos e pastas ☐ Mostrar dados excluídos

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões	
<input type="checkbox"/>	dados_covid19.parquet	10,6 MB	application/octet-stream	4 de dez...	Standard	4 de dez. de 202...	Não público	—	⬇ ⋮
<input type="checkbox"/>	docentes_2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	docentes_2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	escolas_2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	escolas_2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	matriculas_2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	matriculas_2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	turmas_2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	turmas_2020.parquet/	—	Pasta	—	—	—	—	—	⋮

2.2.2 Lições aprendidas

Ao final do Sprint 2, o projeto segue dentro do cronograma do planejamento inicial, no qual todos os objetivos traçados e os resultados almejados foram alcançados. Lembrando que esse Sprint é o mais complexo de ser executado, considerando todas as atividades planejadas. Os objetivos alcançados são:

- Criar a *pipeline* de dados ETL2
- Criar a *pipeline* de dados ETL3 - Primeira Parte
- Execução dos *scripts* com sucesso

Durante o Sprint 2 não foi necessário nenhum tipo de alteração da infraestrutura, premissa ou objetivo. Todos os custos até então também estão dentro do planejamento inicial do projeto.

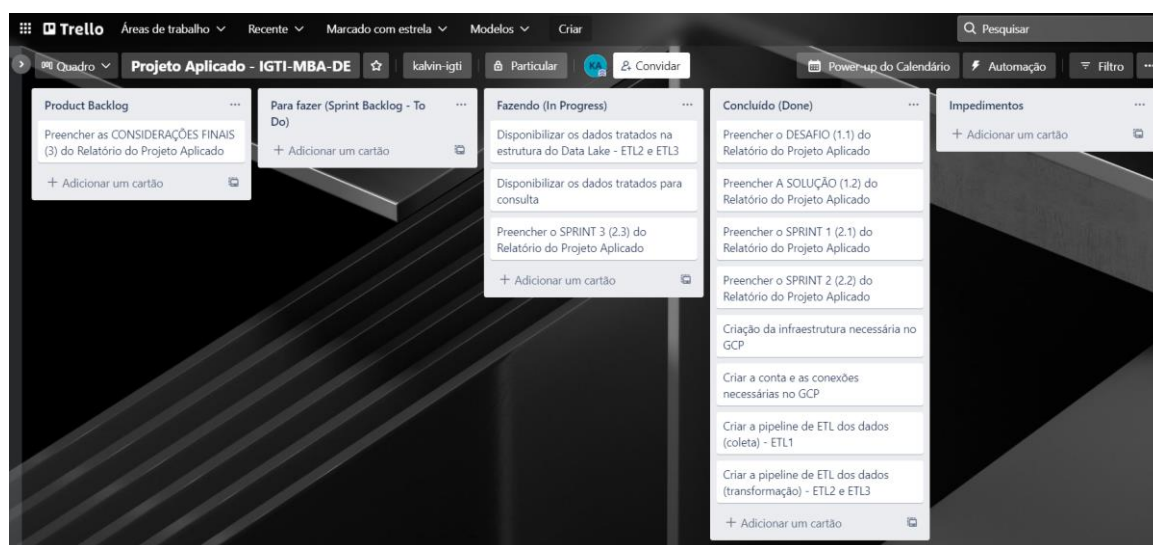
2.3 Sprint 3

No terceiro e último Sprint, o primeiro objetivo é finalizar a transformação dos dados do Censo Escolar 2019 e Censo Escolar 2020 através do *script* ETL3. Com todos os dados tratados, o segundo objetivo será construir um novo *script* responsável por criar toda a estrutura de tabelas no *BigQuery* e, por fim, armazenar todos os dados tratados dentro dessa estrutura. Ao final, será demonstrada algumas instruções simples no *BigQuery* para comprovar a integridade e disponibilidade de todos os dados tratados nos processos de *ETL* explicados anteriormente.

2.3.1 Solução

- Evidência do planejamento:

Figura 33 - Backlog de Produto



- Evidência da execução de cada requisito:

A primeira, segunda e terceira parte do *script* ETL3 (explicado na Sprint 2) já foram concluídas. Portanto, é necessário agora finalizar a quarta e quinta parte do *script* ETL3.

A quarta parte do ETL3 é responsável pelo tratamento final dos dados do Censo Escolar 2019. Já a quinta parte é responsável pelo tratamento final dos dados do Censo Escolar 2020. Nestas etapas, o *script* faz a leitura dos dados na pasta Prata do *Cloud Storage*, define a ordenação das colunas para melhor visualização e análise dos dados, transforma todos os campos para os seus devidos tipos (*String*, *Int* e *Data*) e, por fim, armazena os dados tratados no *Google Storage* (pasta Ouro). Este processo utilizando os dados das turmas do Censo Escolar de 2020 pode ser visualizado na Figura 34.

Figura 34 - ETL3 (Quinta Parte - Turmas do Censo Escolar 2020)

```
In [109]: ##### DADOS DAS TURMAS - 2020

In [110]: # Faz a leitura dos dados na pasta PRATA do bucket GCP
turmas2020 = spark.read.parquet('gs://igti-pa/prata/turmas_2020.parquet')

In [111]: # Redefine a ordem das colunas
cols = ['ID_TURMA', 'TP_LOCALIZACAO', 'TP_ETAPA_ENSINO', 'QT_MATRICULAS', 'NU_DIAS_ATIVIDADE', 'TP_MEDIACAO_DIDATICO_PEDAGO', 'TP_TIPO',
turmas2020 = turmas2020[cols]

In [112]: # Transforma todas as demais colunas para o tipo STRING - MENOS A COLUNA QUANTIDADE (INT)
turmas2020 = turmas2020.withColumn("ID_TURMA", turmas2020["ID_TURMA"].cast('string'))
turmas2020 = turmas2020.withColumn("TP_MEDIACAO_DIDATICO_PEDAGO", turmas2020["TP_MEDIACAO_DIDATICO_PEDAGO"].cast('string'))
turmas2020 = turmas2020.withColumn("NU_DIAS_ATIVIDADE", turmas2020["NU_DIAS_ATIVIDADE"].cast('string'))
turmas2020 = turmas2020.withColumn("TP_TIPO_ATENDIMENTO_TURMA", turmas2020["TP_TIPO_ATENDIMENTO_TURMA"].cast('string'))
turmas2020 = turmas2020.withColumn("TP_TIPO_LOCAL_TURMA", turmas2020["TP_TIPO_LOCAL_TURMA"].cast('string'))
turmas2020 = turmas2020.withColumn("TP_ETAPA_ENSINO", turmas2020["TP_ETAPA_ENSINO"].cast('string'))
turmas2020 = turmas2020.withColumn("IN_ESPECIAL_EXCLUSIVA", turmas2020["IN_ESPECIAL_EXCLUSIVA"].cast('string'))
turmas2020 = turmas2020.withColumn("QT_MATRICULAS", turmas2020["QT_MATRICULAS"].cast('int'))
turmas2020 = turmas2020.withColumn("CO_ENTIDADE", turmas2020["CO_ENTIDADE"].cast('string'))
turmas2020 = turmas2020.withColumn("TP_LOCALIZACAO", turmas2020["TP_LOCALIZACAO"].cast('string'))

In [113]: # Salva os dados das TURMAS 2019 na pasta OURO do bucket como arquivo PARQUET
turmas2020.write.parquet('gs://igti-pa/ouro/turmas2020.parquet')
```

Nas Figuras 35 e 36, é possível observar o código responsável por realizar o processo descrito acima para os dados dos docentes (Censo Escolar 2019) e das matrículas (Censo Escolar 2020). Contudo, nesse código há uma peculiaridade: a data de nascimento dos docentes e dos alunos (matrículas). Nos *datasets* disponibilizados pelo governo federal, não há a informação sobre o DIA, apenas sobre o MÊS e o ANO. Então, para manter a integridade dos dados e facilitar os meios de pesquisa e análise, foi definido para todos os campos do tipo *DATA* o dia “01”, mantendo a informação original do mês e ano. Assim, foi possível criar um novo campo com essa informação.

Figura 35 - ETL3 (Quarta Parte - Docentes do Censo Escolar 2019)

```
In [58]: ##### DADOS DOS DOCENTES - 2019

In [59]: # Faz a leitura dos dados na pasta PRATA do bucket GCP
docentes2019 = spark.read.parquet('gs://igti-pa/prata/docentes_2019.parquet')

In [60]: # Concatena as colunas NU_MES e NU_ANO para facilitar a visualização da DATA_NASCIMENTO
docentes2019 = docentes2019.withColumn('DATA_NASCIMENTO', f.concat_ws('-', 'NU_MES', 'NU_ANO').cast('string'))

In [61]: # Dropa as colunas NU_MES e NU_ANO após a concatenação
docentes2019 = docentes2019.drop('NU_MES', 'NU_ANO')

In [62]: # Redefine a ordem das colunas
cols = ['DATA_NASCIMENTO', 'NU_IDADE_REFERENCIA', 'TP_SEXO', 'TP_COR_RACA', 'CO_UF_NASC', 'CO_MUNICIPIO_NASC', 'CO_UF_END', 'CO_MUNICIPIO_END', 'TP_ZONA_RESIDENCIAL', 'TP_LOCAL_RESID_DIFERENCIADA', 'IN_NECESSIDADE_ESPECIAL', 'TP_ESCOLARIDADE', 'TP_ENSINO_MEDIO', 'IN_COMPLEMENTACAO_PEDAGOGICA', 'IN_ESPECIALIZACAO', 'IN_MESTRADO', 'IN_DOUTORADO', 'IN_POS_NENHUM', 'IN_ESPECIFICO_GESTAO', 'ID_TURMA', 'TP_TIPO_DOCENTE', 'TP_TIPO_CONTRATACAO', 'TP_ETAPA_ENSINO', 'IN_ESPECIAL_EXCLUSIVA', 'CO_ENTIDADE']
docentes2019 = docentes2019[cols]

In [63]: # Transforma a coluna DATA_NASCIMENTO para o tipo DATA
docentes2019 = docentes2019.withColumn('DATA_NASCIMENTO', f.to_date(docentes2019.DATA_NASCIMENTO, 'M-yyyy').alias('DATA_NASCIMENTO'))

In [64]: # Altera o formato do tipo DATA da coluna DATA_NASCIMENTO
docentes2019 = docentes2019.withColumn('DATA_NASCIMENTO', f.date_format('DATA_NASCIMENTO', 'dd-MM-yyyy').alias('DATA_NASCIMENTO'))

In [65]: # Transforma todas as demais colunas para o tipo STRING
docentes2019 = docentes2019.withColumn("NU_IDADE_REFERENCIA", docentes2019["NU_IDADE_REFERENCIA"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_SEXO", docentes2019["TP_SEXO"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_COR_RACA", docentes2019["TP_COR_RACA"].cast('string'))
docentes2019 = docentes2019.withColumn("CO_UF_NASC", docentes2019["CO_UF_NASC"].cast('string'))
docentes2019 = docentes2019.withColumn("CO_MUNICIPIO_NASC", docentes2019["CO_MUNICIPIO_NASC"].cast('string'))
docentes2019 = docentes2019.withColumn("CO_UF_END", docentes2019["CO_UF_END"].cast('string'))
docentes2019 = docentes2019.withColumn("CO_MUNICIPIO_END", docentes2019["CO_MUNICIPIO_END"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_ZONA_RESIDENCIAL", docentes2019["TP_ZONA_RESIDENCIAL"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_LOCAL_RESID_DIFERENCIADA", docentes2019["TP_LOCAL_RESID_DIFERENCIADA"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_NECESSIDADE_ESPECIAL", docentes2019["IN_NECESSIDADE_ESPECIAL"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_ESCOLARIDADE", docentes2019["TP_ESCOLARIDADE"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_ENSINO_MEDIO", docentes2019["TP_ENSINO_MEDIO"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_COMPLEMENTACAO_PEDAGOGICA", docentes2019["IN_COMPLEMENTACAO_PEDAGOGICA"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_ESPECIALIZACAO", docentes2019["IN_ESPECIALIZACAO"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_MESTRADO", docentes2019["IN_MESTRADO"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_DOUTORADO", docentes2019["IN_DOUTORADO"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_POS_NENHUM", docentes2019["IN_POS_NENHUM"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_ESPECIFICO_GESTAO", docentes2019["IN_ESPECIFICO_GESTAO"].cast('string'))
docentes2019 = docentes2019.withColumn("ID_TURMA", docentes2019["ID_TURMA"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_TIPO_DOCENTE", docentes2019["TP_TIPO_DOCENTE"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_TIPO_CONTRATACAO", docentes2019["TP_TIPO_CONTRATACAO"].cast('string'))
docentes2019 = docentes2019.withColumn("TP_ETAPA_ENSINO", docentes2019["TP_ETAPA_ENSINO"].cast('string'))
docentes2019 = docentes2019.withColumn("IN_ESPECIAL_EXCLUSIVA", docentes2019["IN_ESPECIAL_EXCLUSIVA"].cast('string'))
docentes2019 = docentes2019.withColumn("CO_ENTIDADE", docentes2019["CO_ENTIDADE"].cast('string'))

In [66]: # Salva os dados dos DOCENTES 2019 na pasta OURO do bucket como arquivo PARQUET
docentes2019.write.parquet('gs://igti-pa/ouro/docentes2019.parquet')
```

Figura 36 - ETL3 (Quinta Parte - Matrículas do Censo Escolar 2020)

```
In [100]: ##### DADOS DAS MATRÍCULAS - 2020

In [101]: # Faz a leitura dos dados na pasta PRATA do bucket GCP
matriculas2020 = spark.read.parquet('gs://igti-pa/prata/matriculas_2020.parquet')

In [102]: # Concatena as colunas NU_MES e NU_ANO para facilitar a visualização da DATA_NASCIMENTO
matriculas2020 = matriculas2020.withColumn('DATA_NASCIMENTO', f.concat_ws('-', 'NU_MES', 'NU_ANO').cast('string'))

In [103]: # Dropa as colunas NU_MES e NU_ANO após a concatenação
matriculas2020 = matriculas2020.drop('NU_MES', 'NU_ANO')

In [104]: # Redefine a ordem das colunas
cols = ['ID_MATRICULA', 'DATA_NASCIMENTO', 'NU_IDADE_REFERENCIA', 'TP_SEXO', 'TP_COR_RACA', 'CO_UF_NASC', 'CO_MUNICIPIO_NASC', 'CO_UF_END', 'CO_MUNICIPIO_END', 'TP_ZONA_RESIDENCIAL', 'TP_LOCAL_RESID_DIFERENCIADA', 'IN_NECESSIDADE_ESPECIAL', 'TP_ETAPA_ENSINO', 'IN_COMPLEMENTACAO_PEDAGOGICA', 'IN_ESPECIALIZACAO', 'IN_MESTRADO', 'IN_DOUTORADO', 'IN_POS_NENHUM', 'IN_ESPECIFICO_GESTAO', 'ID_TURMA', 'TP_TIPO_DOCENTE', 'TP_TIPO_CONTRATACAO', 'TP_ETAPA_ENSINO', 'IN_ESPECIAL_EXCLUSIVA', 'CO_ENTIDADE']
matriculas2020 = matriculas2020[cols]

In [105]: # Transforma a coluna DATA_NASCIMENTO para o tipo DATA
matriculas2020 = matriculas2020.withColumn('DATA_NASCIMENTO', f.to_date(matriculas2020.DATA_NASCIMENTO, 'M-yyyy').alias('DATA_NASCIMENTO'))

In [106]: # Altera o formato do tipo DATA da coluna DATA_NASCIMENTO
matriculas2020 = matriculas2020.withColumn('DATA_NASCIMENTO', f.date_format('DATA_NASCIMENTO', 'dd-MM-yyyy').alias('DATA_NASCIMENTO'))

In [107]: # Transforma todas as demais colunas para o tipo STRING
matriculas2020 = matriculas2020.withColumn("ID_MATRICULA", matriculas2020["ID_MATRICULA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("NU_IDADE_REFERENCIA", matriculas2020["NU_IDADE_REFERENCIA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_SEXO", matriculas2020["TP_SEXO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_COR_RACA", matriculas2020["TP_COR_RACA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("CO_UF_NASC", matriculas2020["CO_UF_NASC"].cast('string'))
matriculas2020 = matriculas2020.withColumn("CO_MUNICIPIO_NASC", matriculas2020["CO_MUNICIPIO_NASC"].cast('string'))
matriculas2020 = matriculas2020.withColumn("CO_UF_END", matriculas2020["CO_UF_END"].cast('string'))
matriculas2020 = matriculas2020.withColumn("CO_MUNICIPIO_END", matriculas2020["CO_MUNICIPIO_END"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_ZONA_RESIDENCIAL", matriculas2020["TP_ZONA_RESIDENCIAL"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_LOCAL_RESID_DIFERENCIADA", matriculas2020["TP_LOCAL_RESID_DIFERENCIADA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_NECESSIDADE_ESPECIAL", matriculas2020["IN_NECESSIDADE_ESPECIAL"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_ETAPA_ENSINO", matriculas2020["TP_ETAPA_ENSINO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_COMPLEMENTACAO_PEDAGOGICA", matriculas2020["IN_COMPLEMENTACAO_PEDAGOGICA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_ESPECIALIZACAO", matriculas2020["IN_ESPECIALIZACAO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_MESTRADO", matriculas2020["IN_MESTRADO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_DOUTORADO", matriculas2020["IN_DOUTORADO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_POS_NENHUM", matriculas2020["IN_POS_NENHUM"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_ESPECIFICO_GESTAO", matriculas2020["IN_ESPECIFICO_GESTAO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("ID_TURMA", matriculas2020["ID_TURMA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_TIPO_DOCENTE", matriculas2020["TP_TIPO_DOCENTE"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_TIPO_CONTRATACAO", matriculas2020["TP_TIPO_CONTRATACAO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("TP_ETAPA_ENSINO", matriculas2020["TP_ETAPA_ENSINO"].cast('string'))
matriculas2020 = matriculas2020.withColumn("IN_ESPECIAL_EXCLUSIVA", matriculas2020["IN_ESPECIAL_EXCLUSIVA"].cast('string'))
matriculas2020 = matriculas2020.withColumn("CO_ENTIDADE", matriculas2020["CO_ENTIDADE"].cast('string'))

In [108]: # Salva os dados das MATRÍCULAS 2019 na pasta OURO do bucket como arquivo PARQUET
matriculas2020.write.parquet('gs://igti-pa/ouro/matriculas2020.parquet')
```

Por fim, são obtidos todos os dados tratados do Censo Escolar 2019 e do Censo Escolar 2020 após a execução da quarta e quinta parte do ETL3. Lembrando que, para fins de código, essas partes são muito similares. Portanto, não há a necessidade de apresentar todo o código do ETL3.

Com o *script* ETL3 concluído, o primeiro objetivo do Sprint 3 foi atingido. Agora, é necessário realizar o segundo objetivo. Para isso, foi criado um novo *script* chamado de “LOAD”. Primeiramente, foi importada a biblioteca necessária para a comunicação com o *Google BigQuery*, criado o cliente de conexão e configurado os parâmetros dos Jobs de execução, conforme mostra a Figura 37.

Figura 37 - Código LOAD (Início)

LOAD - BigQuery

Script responsável por criar os Datasets e inserir todos os dados tratados do Covid19, Censo 2019 e Censo 2020 no BigQuery

```
In [1]: # Importa a biblioteca necessária
        from google.cloud import bigquery

In [2]: # Cria o cliente do BQ
        client = bigquery.Client()

In [3]: # Configura os parâmetros do Job do BQ
        job_config = bigquery.LoadJobConfig(source_format=bigquery.SourceFormat.PARQUET,)
```

Após isso, é necessário definir e criar os conjuntos de dados que vão ficar armazenados todos os dados tratados até então. Lembrando que cada *dataset* criado pelos *scripts* ETL1, ETL2 e ETL3 geraram uma tabela em formato *PARQUET* que será importada dentro dos conjuntos de dados definidos. Considerando todo o contexto deste projeto, foram definidos três conjuntos:

- covid19
- censo2019
- censo2020

Com os conjuntos definidos, é necessário criar os mesmos dentro do projeto *igti-mba-pa* do *Google BigQuery*. Para isso, foi implementado o código conforme a Figura 38. Neste código também foi determinado que a região que serão armazenados esses conjuntos de dados e essas tabelas é a “US” (segundo a documentação da Google, ficam armazenadas em diversas regiões do Estados Unidos).

Figura 38 - Código LOAD (Construção dos Conjuntos de Dados)

```
In [4]: ##### CONSTRUÇÃO DOS DATASETS NO BIGQUERY

In [5]: # Define os três datasets que serão criados no projeto
dataset_covid = "igti-mba-pa.covid19".format(client.project)
dataset_censo2019 = "igti-mba-pa.censo2019".format(client.project)
dataset_censo2020 = "igti-mba-pa.censo2020".format(client.project)

In [6]: # Faz a construção do Dataset do Covid19
dataset = bigquery.Dataset(dataset_covid)

# Define qual a Localidade do DS - será utilizado "Estados Unidos - Várias Localidades"
dataset.location = "US"

# Faz a criação do Dataset no BQ através da API
dataset = client.create_dataset(dataset, timeout=30) # Make an API request.
print("Dataset criado: {}".format(client.project, dataset.dataset_id))

Dataset criado: igti-mba-pa.covid19

In [7]: # Faz a construção do Dataset do Censo 2019
dataset = bigquery.Dataset(dataset_censo2019)

# Define qual a Localidade do DS - será utilizado "Estados Unidos - Várias Localidades"
dataset.location = "US"

# Faz a criação do Dataset no BQ através da API
dataset = client.create_dataset(dataset, timeout=30) # Make an API request.
print("Dataset criado: {}".format(client.project, dataset.dataset_id))

Dataset criado: igti-mba-pa.censo2019

In [8]: # Faz a construção do Dataset do Censo 2020
dataset = bigquery.Dataset(dataset_censo2020)

# Define qual a Localidade do DS - será utilizado "Estados Unidos - Várias Localidades"
dataset.location = "US"

# Faz a criação do Dataset no BQ através da API
dataset = client.create_dataset(dataset, timeout=30) # Make an API request.
print("Dataset criado: {}".format(client.project, dataset.dataset_id))

Dataset criado: igti-mba-pa.censo2020
```

Assim, com os conjuntos de dados devidamente criados, a última etapa é efetuar a inserção dos dados do Covid19, Censo Escolar 2019 e Censo Escolar 2020 dentro dos seus respectivos conjuntos. Para isso, foi implementado os códigos conforme a Figura 39, 40 e 41.

Figura 39 - Código LOAD (Inserção dos dados do Covid19 no BigQuery)

```
In [9]: ##### INSERÇÃO DOS DADOS DO COVID19 NO BQ

In [10]: # Define as tabelas que serão incorporadas ao BQ para o Dataset COVID19
estados = "igti-mba-pa.covid19.estados"
cidades = "igti-mba-pa.covid19.cidades"
uri_estados = "gs://igti-pa/ouro/estados_covid19.parquet"
uri_cidades = "gs://igti-pa/ouro/cidades_covid19.parquet"

In [11]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_estados, estados, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(estados)
print("Foram inseridas {} linhas no DS Covid19 Estados.".format(destination_table.num_rows))

Foram inseridas 7942 linhas no DS Covid19 Estados.

In [12]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_cidades, cidades, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(cidades)
print("Foram inseridas {} linhas no DS Covid19 Cidades.".format(destination_table.num_rows))

Foram inseridas 1312890 linhas no DS Covid19 Cidades.
```


Figura 41 - Código LOAD (Inserção dos dados do Censo Escolar 2019 no *BigQuery*)

```
In [9]: ##### INSERÇÃO DOS DADOS DO COVID19 NO BQ

In [10]: # Define as tabelas que serão incorporadas ao BQ para o Dataset COVID19
estados = "igti-mba-pa.covid19.estados"
cidades = "igti-mba-pa.covid19.cidades"
uri_estados = "gs://igti-pa/ouro/estados_covid19.parquet"
uri_cidades = "gs://igti-pa/ouro/cidades_covid19.parquet"

In [11]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_estados, estados, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(estados)
print("Foram inseridas {} linhas no DS Covid19 Estados.".format(destination_table.num_rows))

Foram inseridas 7942 linhas no DS Covid19 Estados.

In [12]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_cidades, cidades, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(cidades)
print("Foram inseridas {} linhas no DS Covid19 Cidades.".format(destination_table.num_rows))

Foram inseridas 1312890 linhas no DS Covid19 Cidades.
```

Figura 42 - Código LOAD (Inserção dos dados do Censo Escolar 2020 no *BigQuery*)

```
In [19]: ##### INSERÇÃO DOS DADOS DO CENSO 2020 NO BQ

In [20]: # Define as tabelas que serão incorporadas ao BQ para o Dataset CENSO 2020
docentes = "igti-mba-pa.censo2020.docentes"
matriculas = "igti-mba-pa.censo2020.matriculas"
turmas = "igti-mba-pa.censo2020.turmas"
escolas = "igti-mba-pa.censo2020.escolas"

uri_docentes = "gs://igti-pa/ouro/docentes2020.parquet/part*"
uri_matriculas = "gs://igti-pa/ouro/matriculas2020.parquet/part*"
uri_turmas = "gs://igti-pa/ouro/turmas2020.parquet/part*"
uri_escolas = "gs://igti-pa/ouro/escolas2020.parquet/part*"

In [21]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_docentes, docentes, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(docentes)
print("Foram inseridas {} linhas no DS Censo 2020 Docentes.".format(destination_table.num_rows))

Foram inseridas 9288283 linhas no DS Censo 2020 Docentes.

In [22]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_matriculas, matriculas, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(matriculas)
print("Foram inseridas {} linhas no DS Censo 2020 Matriculas.".format(destination_table.num_rows))

Foram inseridas 49194223 linhas no DS Censo 2020 Matriculas.

In [23]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_turmas, turmas, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(turmas)
print("Foram inseridas {} linhas no DS Censo 2020 Turmas.".format(destination_table.num_rows))

Foram inseridas 2353351 linhas no DS Censo 2020 Turmas.

In [24]: # Configura o Job que vai carregar os dados no DS
load_job = client.load_table_from_uri(uri_escolas, escolas, job_config=job_config)

# Aguarda a execução do Job por completo
load_job.result()

# Faz a inserção dos dados do GCP (arquivo parquet) no BQ
destination_table = client.get_table(escolas)
print("Foram inseridas {} linhas no DS Censo 2020 Escolas.".format(destination_table.num_rows))

Foram inseridas 224229 linhas no DS Censo 2020 Escolas.
```

- Evidência dos resultados:

Após a execução de todas as etapas e códigos descritos nas evidências de execução, foi possível construir a zona de Ouro (pasta) dentro da estrutura de *Data Lake* no *Cloud Storage* e a estrutura de conjuntos de dados no *BigQuery* contendo todos os dados coletados e tratados até então. Ou seja, após executar todas as etapas do Sprint 1, Sprint 2 e Sprint 3, foi possível atingir o objetivo final deste projeto: disponibilizar os dados devidamente tratados sobre a Covid19, o Censo Escolar 2019 e o Censo Escolar 2020 para consultas dentro de um *Data Warehouse* em uma estrutura *cloud*.

Na Figura 43, é possível visualizar a estrutura da pasta Ouro dentro do *Cloud Storage* com todos os dados tratados durante esse projeto. Já na Figura 44 é possível visualizar a estrutura do *BigQuery* com os conjuntos de dados e suas respectivas tabelas.

Figura 43 - Dados finais no *Cloud Storage* (Pasta Ouro)

igti-pa

Local: eu (várias regiões na União Europeia) | Classe de armazenamento: Standard | Acesso público: Não público | Proteção: Nenhum

OBJETOS | CONFIGURAÇÃO | PERMISSÕES | PROTEÇÃO | CICLO DE VIDA

Intervalos > igti-pa > ouro

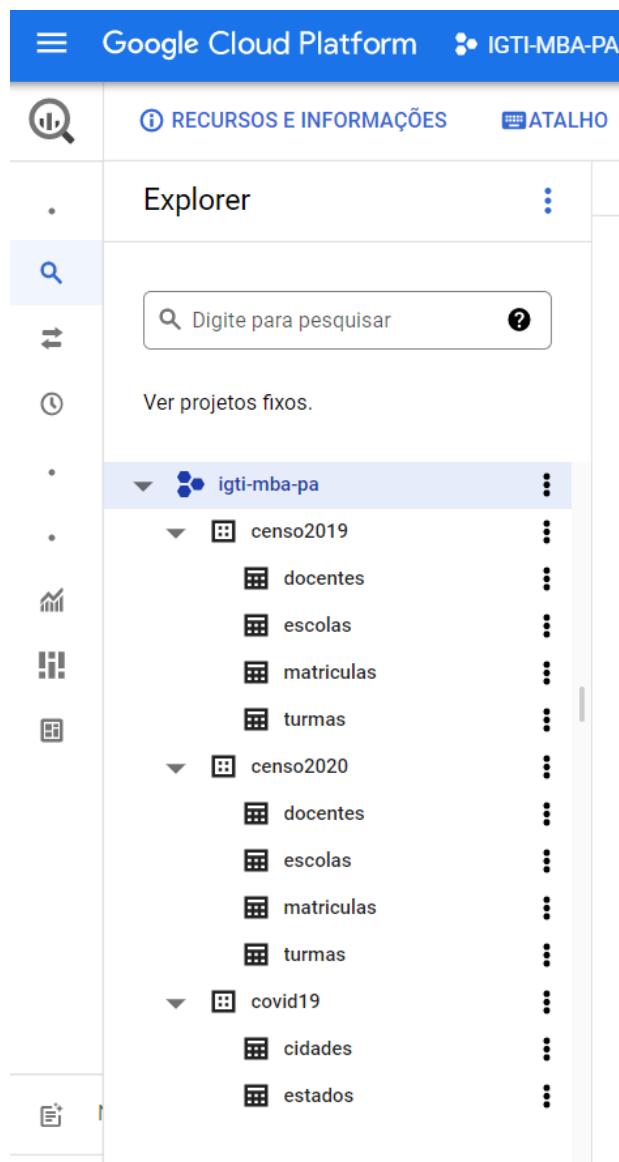
FAZER UPLOAD DE ARQUIVOS | CARREGAR PASTA | CRIAR PASTA | GERENCIAR RETENÇÕES | FAZER O DOWNLOAD | EXCLUIR

Classificar e filtrar | Filtro: Filtrar objetos e pastas | Mostrar dados excluídos

A classificação e o filtro de objeto podem tornar o navegador do Storage mais lento. Para um desempenho mais rápido, selecione **Filtrar apenas pelo prefixo do nome** no menu para aplicar filtros. DISPENSAR

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado	↑	Classe de armazenamento	Última modificação	Acesso público	
<input type="checkbox"/>	docentes2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	docentes2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	escolas2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	escolas2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	matriculas2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	matriculas2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	turmas2019.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	turmas2020.parquet/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	estados_covid19.parquet	119,2 KB	application/octet-stream	4 de dez. de ...	—	Standard	4 de dez. de 2021 11:...	Não público	⬇ ⋮
<input type="checkbox"/>	idades_covid19.parquet	4,8 MB	application/octet-stream	4 de dez. de ...	—	Standard	4 de dez. de 2021 11:...	Não público	⬇ ⋮

Figura 44 - Estrutura no *BigQuery*



Por fim, foi realizada duas consultas nos conjuntos com o intuito de comprovar a integridade e consistência dos dados carregados. Na Figura 45, foi efetuada uma consulta mostrando a quantidade de matrículas feitas nos anos de 2019 e 2020. Além da consulta, vale ressaltar a velocidade que foi executada essa query no qual mais de 100 milhões de registros foram lidos, no qual a consulta retornou o resultado em 0.6 segundos.

Figura 45 - BigQuery (Primeira Consulta)

Q *CONSUL...

X

criar nova consulta

EXECUTAR

SALVAR

PROGRAMAÇÃO

MAIS

Esta consulta processará 0 B quando executada.

```
select count(*) as NUM_MATRICULAS from igti-mba-pa.censo2019.matriculas
union all
select count(*) as NUM_MATRICULAS from igti-mba-pa.censo2020.matriculas
|
```

Local de processamento: US

Resultados da consulta

SALVAR RESULTADOS

EXPLORAR DADOS

Consulta finalizada (tempo decorrido:0,6 s, bytes processados: 0 B)

Informações do job

Resultados

JSON

Detalhes da execução

Linha	NUM_MATRICULAS
1	49194223
2	51011521

Já a Figura 46 mostra uma consulta que mostra as cidades que mais tiveram casos de Covid19 comprovados no ano de 2020 junto com a quantidade de matrículas ocorridas nas escolas da mesma cidade e do mesmo ano. Novamente, vale ressaltar que essa consulta foi efetuada usando três tabelas e dois conjuntos de dados diferentes, processando 1Gb de dados, retornando o resultado em aproximadamente 3.8 segundos.

Figura 46 - BigQuery (Segunda Consulta)

Q *CONSUL...

X

criar nova consulta

EXECUTAR

SALVAR

PROGRAMAÇÃO

MAIS

Esta consulta processará 1 GiB quando executada.

```
select distinct a.cidade as CIDADE, a.qtd_ultimos_casos_disponiveis as QTD_CASOS_COVID, count(b.ID_MATRICULA) as QTD_MATRICULAS from igti-mba-pa.covid19.cidades a
inner join igti-mba-pa.censo2020.escolas c on a.codigo_ibge = c.CO_MUNICIPIO
inner join igti-mba-pa.censo2020.matriculas b on c.CO_ENTIDADE = b.CO_ENTIDADE
where a.data = '2020-12-31'
group by a.CIDADE, a.qtd_ultimos_casos_disponiveis
order by a.qtd_ultimos_casos_disponiveis desc
limit 10;
```

Local de processamento: US

Resultados da consulta

SALVAR RESULTADOS

EXPLORAR DADOS

Consulta finalizada (tempo decorrido:3,8 s, bytes processados: 1 GB)

Informações do job

Resultados

JSON

Detalhes da execução

Linha	CIDADE	QTD_CASOS_COVID	QTD_MATRICULAS
1	São Paulo	401718	2798679
2	Brasília	220482	720936
3	Rio de Janeiro	165079	1349052
4	Salvador	109887	518492
5	Fortaleza	82294	635323
6	Manaus	82218	543635

2.3.2 Lições aprendidas

Com o final do Sprint 3, o projeto cumpriu todos o seu escopo estipulado dentro do prazo e das premissas e restrições definidas. Todas as execução dos códigos criados foram efetuadas com sucesso, cumprindo totalmente a sua finalidade. Ao final, os objetivos alcançados no Sprint 3 foram:

- Criar a *pipeline* de dados ETL2 - Parte Final
- Disponibilizar os dados tratados na estrutura do *Data Lake*
- Disponibilizar os dados tratados para consulta no *BigQuery*

Durante o Sprint 3 não foi necessário nenhum tipo de alteração da infraestrutura, premissa ou objetivo. Além disso, todos os custos estão dentro do planejamento inicial do projeto. Por fim, a velocidade e a capacidade de processamento das consultas no *BigQuery*, sejam simples ou complexas, se mostrou mais performática do que o esperado, agregando mais valor ao projeto.

3. Considerações Finais

3.1 Resultados Finais

Com o termino da parte prática, foi possível alcançar todos os objetivos previamente definidos obedecendo todas as premissas e restrições também definidas no início do projeto. Ou seja, foi possível criar uma infraestrutura em *cloud* (*GCP*) utilizando o conceito de *IaC* (Infraestrutura como Código) e, após isso, desenvolver códigos responsáveis por buscar, extrair, transformar e armazenar os dados referentes ao Covid19 e a educação brasileira (Censo Escolar 2019 e 2020).

Esse projeto teve alguns pontos positivos que merecem destaques. O primeiro é a questão do custo. Esse projeto como um todo não teve custos, uma vez que a *GCP* disponibiliza US\$300 gratuitos para uso (um dos principais motivos de ter sido escolhido o provedor da Google). Contudo, caso fosse gerar alguma cobrança, não seria gasto mais que R\$200 para realizar todo o projeto, conforme o relatório de faturamento gerado pela própria *GCP*. O segundo ponto positivo é a facilidade, segurança e rapidez de se trabalhar na *GCP*, o que favoreceu na hora da criação e configuração do ambiente *cloud* e na criação e execução dos diversos códigos (detalhados nas Sprints). Por fim, o terceiro ponto positivo é a velocidade na execução das instruções efetuadas no *BigQuery* que, por mais que não seja o objetivo deste trabalho, vai auxiliar plenamente os profissionais responsáveis pelas instruções que serão feitas nesses conjuntos de dados, melhorando sua experiência de uso e deixando o seu trabalho mais performático. Pode-se considerar também um ponto positivo esse projeto não ter pontos negativos, já que todos os objetivos foram cumpridos dentro do prazo, respeitando todas as premissas e restrições impostas.

Entretanto, por mais que não existam pontos negativos relevantes a serem apontados, ocorreram duas dificuldades durante esse trabalho que merecem destaque. A primeira delas são os dados disponibilizados pelo governo brasileiro, principalmente sobre o Censo Escolar 2019 e 2020, no qual várias colunas que estão especificadas no dicionário de dados não existem nos dados disponibilizados, além de vários dados não coletados ou então coletados de forma errônea. Isso demandou uma grande carga de trabalho em cima desses dados, para que fosse possível tratá-los da melhor forma possível para assim então disponibilizá-los para futuras consultas. A segunda foi a mudança de infraestrutura, no qual foi definido não usar mais o *Apache Airflow* para o orquestramento das *pipelines* de dados e sim apenas a utilização do *Jupyter Notebook* e execução manual dos *scripts*. Essa mudança acarretou em diversas mudanças no projeto e nos códigos já previamente desenvolvidos, no qual uma alteração geral foi necessária, demandando uma grande carga de trabalho e tempo.

Por fim, é possível mencionar duas lições aprendidas com o decorrer do projeto. A primeira é a importância de um planejamento detalhado e bem específico, tanto a parte teórica quanto a parte prática, sendo necessário ainda revisar todo esse plano posteriormente. Com isso, as dificuldades relacionadas a alteração de infraestrutura não teriam ocorrido. E a segunda é seguir fielmente tudo aquilo que foi programado, garantindo assim o sucesso do projeto. No caso deste trabalho, por mais que tenha ocorrido essa alteração de infraestrutura, todos os objetivos foram alcançados por conta de um ótimo planejamento e uma execução fiel àquilo que foi programado.

3.2 Contribuições

Este projeto demonstrou que é possível fornecer para os Cientistas de Dados e Analistas de Dados os dados necessários para que os mesmos construam as mais diversas instruções para extrair *insights* importantes para a tomada de decisão dos gerentes e dos Ministros da Educação do Brasil. Por mais que isso já seja feito de forma genérica, esse projeto proporcionou dados muito mais específicos, além da possibilidade de cruzamento entre todos os dados disponíveis de forma simples e eficaz.

Esse projeto não prevê a criação dessas instruções nem a definição dos profissionais para a continuação do trabalho, conforme já definido ainda na parte inicial. Contudo, toda a parte de Engenharia de Dados referente ao macroproblema exposto foi solucionada neste trabalho.

3.3 Próximos passos

Para os próximos passos, será necessário realizar a busca, extração, transformação e armazenamento dos dados do Censo Escolar 2021 e dos dados do Covid19 de 2021 (assim que o governo brasileiro os disponibilizar). Assim, esse projeto terá uma base de dados ainda mais ampla, contendo os dados educacionais do Brasil de 2019, 2020 e 2021, além dos dados atualizados sobre a Covid19. Assim, será possível efetuar novas consultas nos conjuntos de dados, aumentando ainda mais as possibilidades de se encontrar *insights* importantes para a tomada de decisão por parte do Ministério da Educação.