Projeto Câmera Aberta

Especificação

Sumário

[Data Lake 2](#_Toc189061290)

[RAW 2](#_Toc189061291)

[GTFS 2](#_Toc189061292)

[APIs 2](#_Toc189061293)

[TRUSTED 3](#_Toc189061294)

[STOP\_TIMES 3](#_Toc189061295)

[TRIPS 3](#_Toc189061296)

[SHAPES 3](#_Toc189061297)

[POSICAO 4](#_Toc189061298)

[PREVISAO 4](#_Toc189061299)

[GARAGEM 4](#_Toc189061300)

[BUSINESS 4](#_Toc189061301)

[CHEGADA 4](#_Toc189061302)

[Métricas e KPIs 6](#_Toc189061303)

[Filtros 6](#_Toc189061304)

[Garagem 6](#_Toc189061305)

[Veículos 6](#_Toc189061306)

[Linhas 7](#_Toc189061307)

[Paradas 7](#_Toc189061308)

[Ranking Veículos maior atraso 7](#_Toc189061309)

[Ranking Linhas maior atraso 7](#_Toc189061310)

[Ranking Paradas maior atraso 7](#_Toc189061311)

[Mapa 7](#_Toc189061312)

[Tabelas dinâmicas 7](#_Toc189061313)

# Data Lake

## RAW

A camada RAW deverá armazenar dados públicos disponibilizados pela SPTrans no site <https://www.sptrans.com.br/desenvolvedores> a partir de arquivos “GTFS” e “APIs”.

### GTFS

Os dados disponibilizados nos arquivos “GTFS” são considerados “estáticos”, não são atualizados em near real time. Sendo assim, utilizaremos esses dados apenas para fins cadastrais, atualizando-os uma vez ao dia de forma completa (substituindo o conteúdo anterior).

Deverão ser armazenados os dados extraídos dos seguintes arquivos:

#### stops

#### stop\_times

#### routes

#### trips

#### shapes

### APIs

Os dados disponibilizados a partir das APIs “Olho Vivo” são near real time. Sendo assim, serão carregados de forma incremental na camada RAW a cada 2 minutos.

Deverão ser armazenados os dados extraídos das seguintes APIs:

#### Empresas/GET

Utilizar método GET e armazenar dados em EMPRESAS

#### Linhas/BUSCAR

Utilizando método BUSCAR e armazenar os dados em LINHAS

#### Posição/GET

Utilizar método GET e armazenar os dados em POSICAO

#### Previsão/Linha

Utilizar método Linha, passando no parâmetro codigoLinha o conteúdo de TRIPS-TRIP\_ID e armazenar os dados em PREVISAO

#### Previsão/Garagem

Utilizar método Garagem, passando no parâmetro codigoEmpresa o conteúdo de EMPRESAS-E\_C e armazenar os dados em GARAGEM (inclusive o campo E\_C)

## TRUSTED

A camada TRUSTED armazenará dados extraídos da camada RAW após serem tratados/enriquecidos de acordo com as regras descritas abaixo.

### STOP\_TIMES

Criar os campos “STOP\_LAT” e “STOP\_LON”, preenchendo-os com o mesmo conteúdo do arquivo stops, buscando a partir do campo STOP\_ID

### TRIPS

(1) Criar os campos “ROUTE\_LONG\_NAME” e “AGENCY”, preenchendo-os com o mesmo conteúdo do arquivo ROUTES, buscando a partir do campo ROUTE\_ID.

(2) Criar o campo “DURATION\_TIME”, preenchendo-o com o tempo de duração da viagem, calculando a duração a partir dos tempos da entidade stop\_times (ARRIVAL\_TIME de STOP\_ID = 1 – DEPARTURE\_TIME do último STOP\_ID) a partir do campo TRIP\_ID.

### SHAPES

(1) criar os campos “TRIP\_ID”, “ROUTE\_ID”, “ROUTE\_LONG\_NAME”, “AGENCY” com o mesmo conteúdo de trips, buscando a partir do campo TRIP\_ID.

(2) criar os campos “STOP\_ID” e “STOP\_SEQUENCE”, preenchendo-os a partir de uma rotina.

Regra de negócio para campos “STOP\_ID” e “STOP\_SEQUENCE”:

Comparação da latitude e longitude dos pontos de parada de uma linha com a latitude e longitude do percurso dessa mesma linha (SHAPE\_PT\_LAT - STOP\_TIMES-STOP\_LAT ; SHAPE\_PT\_LON - STOP\_LON), marcando qual ponto do percurso é o mais aproximado de um determinado ponto de parada.

Para isso, a rotina abaixo deve ser executada para cada registro da entidade TRIPS (para cada linha).

Selecionar os pontos do percurso e os pontos de parada da linha em questão (STOP\_TIMES-TRIP\_ID = TRIPS-TRIP\_ID e SHAPES-TRIP\_ID = TRIPS-TRIP\_ID), ordenando pela sequencia de paradas (STOP\_TIMES-STOP\_SEQUENCE) e pela sequencia do percurso (SHAPES-SHAPE\_PT\_SEQUENCE).

O código da primeira parada (STOP\_TIMES-STOP\_ID quando STOP\_TIMES-STOP\_SEQUENCE=1) deve ser registrada como sendo a parada equivalente ao primeiro ponto do percurso (SHAPES-STOP\_ID quando SHAPES-SHAPE\_PT\_SEQUENCE = 1).

Para os demais pontos de parada da linha (STOP\_TIMES-STOP\_SEQUENCE>1)...

Para cada registro da entidade STOP\_TIMES, buscar o registro do percurso mais próximo ao ponto da parada, comparando a latitude longitude da parada (STOP\_TIMES-STOP\_LAT, STOP\_TIMES-STOP\_LON) com a latitude e longitude do percurso (SHAPES-SHAPE\_PT\_LAT, SHAPES-SHAPE\_PT\_LON). Preencher no registro da entidade percurso (SHAPES) os campos SHAPES-STOP\_ID e SHAPES-STOP\_SEQUENCE com os dados da parada.

Executar essa rotina até a última parada da linha (STOP\_TIMES-TRIP\_ID = TRIPS-TRIP\_ID).

Processar a próxima linha (TRIPS-TRIP\_ID), até terminarem as linhas.

### POSICAO

Preencher campo DATE a partir do conteúdo campo VS\_TA

### PREVISAO

Preencher campo DATE a partir do conteúdo do campo VS\_TA

### GARAGEM

**C**riar os campos “ROUTE\_ID”, “ROUTE\_LONG\_NAME”, “AGENCY” com o mesmo conteúdo de trips, buscando TRIP\_ID a partir do campo L\_CL.

## BUSINESS

Na camada BUSINESS serão armazenados dados das camadas TRUSTED ou RAW, após serem transformados de acordo com regras de negócio específicas.

### CHEGADA

Para mensurar a assertividade da previsão de localização dos veículos, deverá ser criada uma tabela para armazenar o horário aproximado da chegada dos veículos nos pontos de parada. A posição recebida pela API não necessariamente é a mesma de um ponto de parada e nem com as posições do percurso recebidas na API Previsão ou nos arquivos GTFS. Sendo assim, será necessário a criação de uma rotina para dar visibilidade para a gestão da operação de linhas, veículos ou pontos que precisam de atenção e uma possível ação visando otimizar a operação e qualidade do transporte público.

Regra de negócio para criação dos registros da tabela chegada:

Comparação da latitude e longitude da posição dos veículos de uma linha com a latitude e longitude do percurso/paradas dessa mesma linha (SHAPES-SHAPE\_PT\_LAT - POSICAO-VS\_PY; SHAPES-SHAPE\_PT\_LON – POSICAO-VS\_PX), marcando o horário aproximado da passagem do veículo por uma ponto do percurso equivalente a um ponto de parada.

Para os registros ainda não processados da API POSICAO (quando POSICAO-PROC\_DATE\_TIME vazio)...

Ordenar por horário, linha e veículo (POSICAO-HR, POSICAO-L\_CL, POSICAO-VS\_P)

Para cada combinação de horário, linha e veículo de POSICAO...

Selecionar os pontos do percurso da linha (SHAPES-TRIP\_ID = POSICAL-L\_CL) ordenando pela sequencia do percurso (SHAPES-SHAPE\_PT\_SEQUENCE).

Selecionar os pontos de chegada da linha/veículo em questão (CHEGADA-L\_CL = POSICAO-L\_CL e CHEGADA-VS\_P = POSICAO-VS\_P) registrados no dia ou dia anterior ( CHEGADA-DATE = data do campo POSICAO-VS\_TA ou dia anterior), ordenando pela sequencia de chegadas (CHEGADA-STOP\_SEQUENCE)

Selecionar a previsão da linha/veículo em questão (PREVISAO-L\_CL = POSICAO-L\_CL e PREVISAO-VS\_P = POSICAO-VS\_P) referentes ao dia ou dia anterior (PREVISAO-DATE = data do campo POSICAO-VS\_TA ou dia anterior), ordenando pelo horário decrescente, linha, veículo e parada (PREVISAO-HR,L\_CL, VS\_P, P\_CP)

Buscar o registro do percurso (SHAPES) mais próximo a posição do veículo, comparando a latitude longitude do percurso (SHAPES-SHAPE\_PT\_LAT, SHAPES-SHAPE\_PT\_LON) com a latitude longitude da posição (POSICAO-VS\_PY, POSICAO-VS\_PX)

Se o ponto do percurso encontrado não possui parada associada (SHAPES-STOP\_ID<>””), ler ponto anterior do percurso (SHAPES-SHAPE\_SEQUENCE=SHAPES-SHAPE\_SEQUENCE-1), até encontrar um registro com uma parada associada (SHAPES-STOP\_ID<>””).

Buscar em CHEGADA se o último registro para essa linha/veículo é referente a parada encontrada (CHEGADA-STOP\_ID=SHAPES-STOP\_ID).

Caso encontre, não é necessário criar um novo registro em CHEGADA.

Caso não encontre, será necessário criar um novo registro em CHEGADA. Para isso...

Buscar em PREVISAO o horário da última previsão da combinação linha/veículo/parada (PREVISAO-L\_CL = POSICAO-L\_CL, PREVISAO-VS\_P = POSICAO-VS\_P, PREVISAO-P\_CP = SHAPES-STOP\_ID)

Criar registro na entidade CHEGADA com os dados da posição, percurso/parada e previsão:

VS\_TA = POSICAO

L\_CL = POSICAO

CP = POSICAO

VS\_P = POSICAO

SHAPE\_ID = SHAPES

SHAPE\_PT\_SEQUENCE = SHAPES

STOP\_SEQUENCE = SHAPES

HR = POSICAO

VS\_PY = POSICAO

VS\_PX = POSICAO

ROUTE\_ID = SHAPES

VS\_T = PREVISAO

VS\_T\_DESVIO = Diferença entre planejado e realizado (VS\_T – VS\_TA) em minutos

ROUTE\_ID = SHAPES

ROUTE\_LONG\_NAME = SHAPES

AGENCY = SHAPES

YEAR = Ano de VS\_TA

MONTH = mês de VS\_TA

DATE = data de VS\_TA

WEEK\_DAY = dia da semana de VS\_TA

Preencher o conteúdo do campo POSICAO-PROC\_DATE dos registros processados em POSICAO, utilizando a data/hora atual.

Executar essa rotina para todos os registros ainda não processados da API POSICAO (quando POSICAO-PROC\_DATE\_TIME vazio)...

# Métricas e KPIs

## Filtros

Permitir filtro por Linha, Parada, Veículo, Data, Mês, Ano e Dia da Semana.

Por padrão, filtrar a data de hoje

## Garagem

Contar GARAGEM-VS\_P distintos, quando GARAGEM-HR = último horário encontrado em GARAGEM

## Veículos

Contar POSICAO-VS\_P distintos, quando POSICAO-HR = último horário encontrado em POSICAO

## Linhas

Contar PREVISAO-L\_CL distintos, quando PREVISAO-HR = último horário encontrado em PREVISAO

## Paradas

Contar PREVISAO-P\_CP distintos, quando PREVISAO-HR = último horário encontrado em PREVISAO

## Ranking Veículos maior atraso

Soma de CHEGADA-VS\_T\_DESVIO decrescente, agrupado por VS\_P

## Ranking Linhas maior atraso

Soma de CHEGADA-VS\_T\_DESVIO decrescente, agrupado por L\_CL

## Ranking Paradas maior atraso

Soma de CHEGADA-VS\_T\_DESVIO decrescente, agrupado por CP

## Mapa

Última localização dos veículos registradas em POSICAO

## Tabelas dinâmicas

Disponibilizar dados das tabelas SHAPES, POSICAO, PREVISAO, GARAGEM e CHEGADA para “discovery”. Limitar tabela para apresentar no máximo 1000 registros