

Spark – Big Data Processing

Construção e Padrão de Data Application



Semantix[®]

All about data

\$ whoami



Everton Amaral

Tech lead

Bach. Cienc. Comp. UNASP;

Desenvolvimento +15 anos;

Engenheiro de Dados, 2anos;

Java de DNA;

Contatos

everton.amaral@semantix.com.br

linkedin.com/in/evertonamaralsp



Construção e padrão de Data Application

- O mínimo de conhecimento;
- Entregas;
- Templates de projetos semantix;

O minimo de conhecimento

- 100% dos engenheiros vão precisar lidar com codigo;
- Codigo deve ser feito para ser lido por pessoas;

O minimo de conhecimento

```
def RecebeValor(a,b,c):  
    valorBruto = x.prepare_data().sql("select * from  
    produto where id="+a)[0]['nu_valor_bruto']  
  
    if c > 0.25:  
        res = (valorBruto + b)*c  
    else  
        res = (valorBruto + b)*0.25  
  
    x.prepare_data().sql('insert into produto_fiscal set  
    valorBruto =' + res + ' where id=' + a )
```

O minimo de conhecimento

```
def acrescimo_valor_bruto_produto_mais_taxa(
    int: id_produto,
    double: valor,
    double: tax
):
    """
    Esse metodo aplica o valor do custo do produto real
    mais a taxa de impostos calculada para esse produto
    não permite q a taxa seja menor que taxa base
    """

    prepare_data = connect_session.prepare_data()
    result = prepare_data.sql("select * from produto where id = " +
        id_produto)
    valor_bruto = result.first()["nu_valor_bruto"]
    if tax > TAXA_BASE:
        novo_valor_bruto = (valor_bruto + valor) * tax
    else:
        novo_valor_bruto = (valor_bruto + valor) * TAXA_BASE

    prepare_data.sql(
        "insert into produto_fiscal set valorBruto =" +
        novo_valor_bruto +
        " where id=" + id_produto
    )
```

O minimo de conhecimento

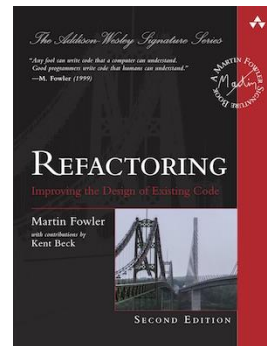
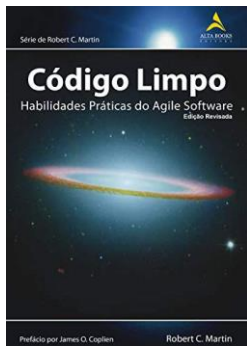
1. Entender a historia da liguagem suas convenções;
 - ex:
 - java,scala variavelEmJava(camel case)
 - python variavel_em_python (snake case)
 - formatação: Java: checkstyle, Python: pep8

O minimo de conhecimento

1. Entender as convenções das linguagens;
2. Entenda as características da linguagem;
 - ex:
 - Java,Scala orientação a objeto para aproveitar o maximo do java entender esse conceito e importante;
 - Python, Scala linguagem funcional

O mínimo de conhecimento

1. Entender as convenções das linguagens;
2. Entenda as características da linguagem;
3. Estudar e aprender com texto, blog seguindo siga bom profissionais na redes



Entregas



Entregas

SCRIPT.py

```
#!/usr/bin/env python:
# coding: utf-8

# Bibliotecas utilizadas no processo

from pyspark.sql import SparkSession
import logging
import sys

def setInfo(msg):
    '''Função que grava informações de informação do processamento no log/YARN
    Parâmetros de Entrada:
    msg - String - Texto da mensagem a gravar na log'''
    logger.info(msg)

def grava_status(spark, tabela, data_execucao, status):
    '''Função que grava o status de carga da tabela passada como parametro
    Parâmetros de Entrada:
    spark - SparkSession - Sessão do spark argv ser utilizada:
    tabela - String - Nome da tabela
    data_execucao - String - Data da execução em formato texto
    status - String - Recebe OK ou NOK
    '''

    spark.sql("insert into table vc.status_cargas values ('{d}', '{t}', '{s}', 0)".format(d = data_execucao, t = tabela, s = status))

if __name__ == '__main__':
    # Definições para o contexto do Spark
    spark = SparkSession.builder.enableHiveSupport().getOrCreate()

    logger = logging.getLogger("")
    logging.basicConfig(level=logging.INFO, format='%(asctime)s: SMART_TABLES %(message)s')

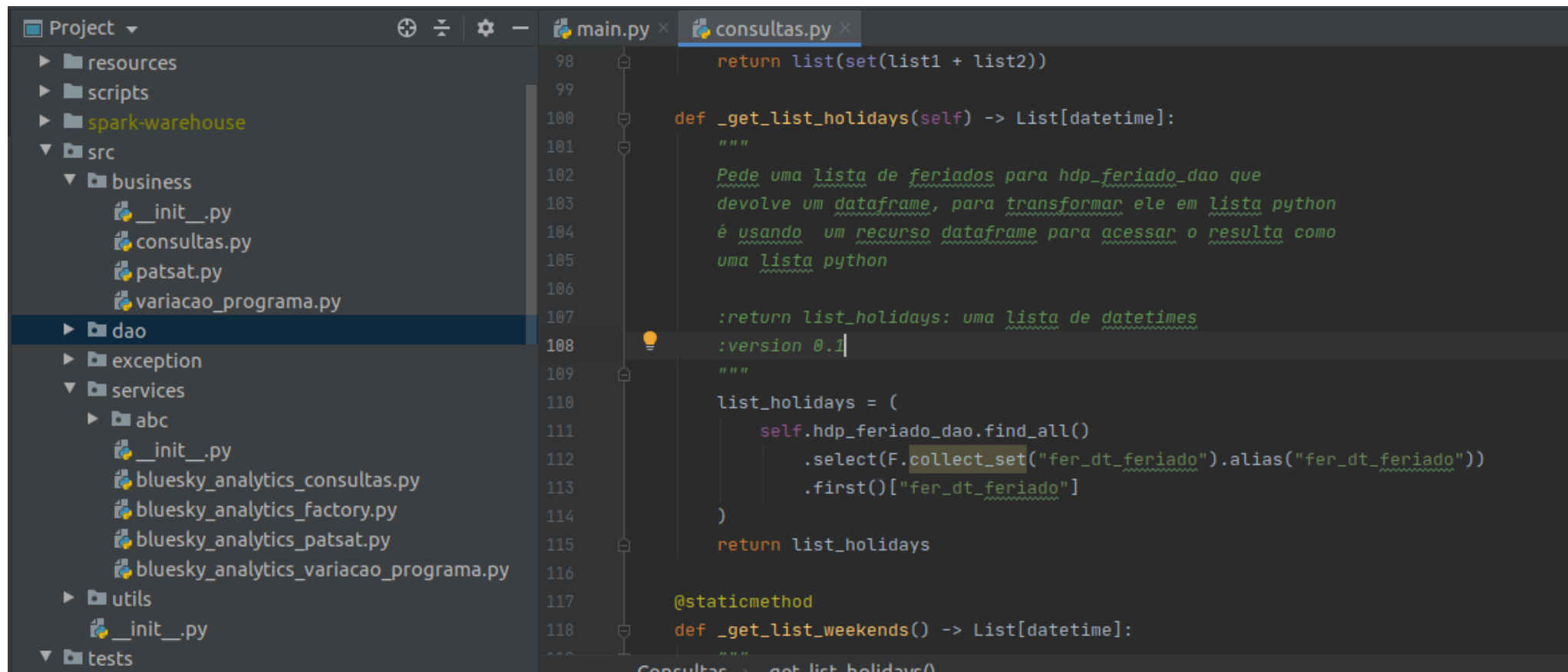
    setInfo("### INICIO DO PROCESSO ###")

    # Recebendo Parâmetros de Entrada
    tabela = sys.argv[1]
    status = sys.argv[2]
    data_execucao = sys.argv[3]

    spark.sql("""set hive.exec.dynamic.partition.mode=nonstrict""")
```

Entregas

PYTHON

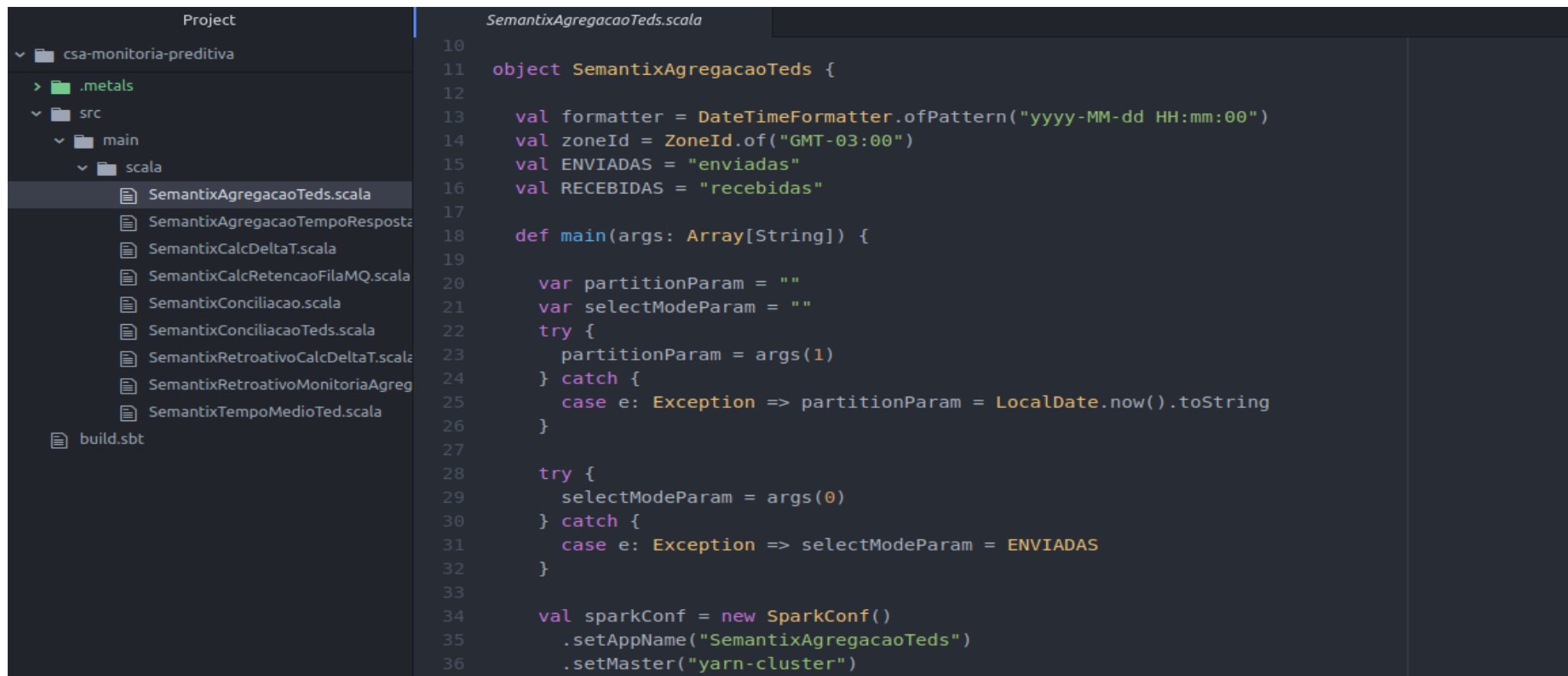


The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer shows a hierarchy: resources, scripts, spark-warehouse, src (business, dao, exception, services, utils, tests). The code editor shows the file consultas.py with the following code:

```
98 return list(set(list1 + list2))
99
100 def _get_list_holidays(self) -> List[datetime]:
101     """
102     Pede uma lista de feriados para hdp_feriado_dao que
103     devolve um dataframe, para transformar ele em lista python
104     é usando um recurso dataframe para acessar o resulta como
105     uma lista python
106
107     :return list_holidays: uma lista de datetimes
108     :version 0.1
109     """
110     list_holidays = (
111         self.hdp_feriado_dao.find_all()
112         .select(F.collect_set("fer_dt_feriado").alias("fer_dt_feriado"))
113         .first()["fer_dt_feriado"]
114     )
115     return list_holidays
116
117 @staticmethod
118 def _get_list_weekends() -> List[datetime]:
119     """
120     """
```

Entregas

SCALA

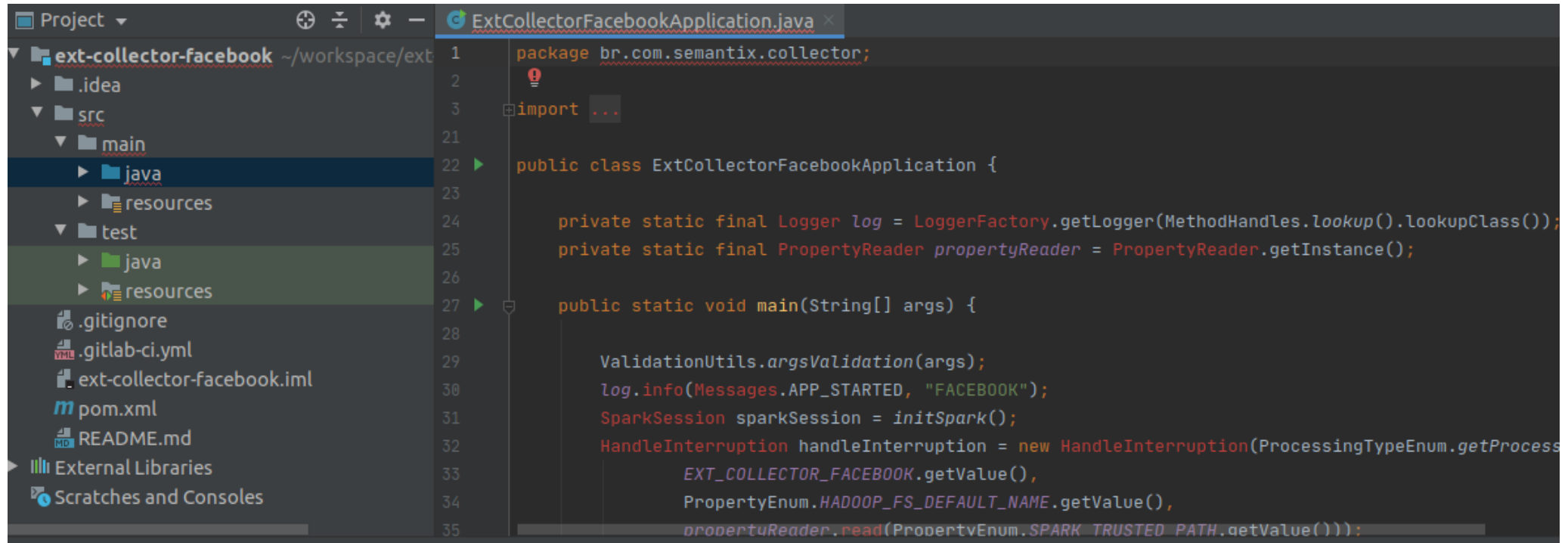


The screenshot shows an IDE with a project structure on the left and a Scala file named `SemantixAgregacaoTeds.scala` open in the editor. The project structure includes a folder `csa-monitoria-preditiva` with subfolders `.metals`, `src`, and `main`. Inside `main`, there is a `scala` folder containing several Scala files, including `SemantixAgregacaoTeds.scala`, which is the file currently open in the editor. The code in the editor defines an object `SemantixAgregacaoTeds` with a `main` method that takes an array of strings as input. The `main` method sets up a `SparkConf` and uses `DateTimeFormatter` and `ZoneId` to format dates. It also defines constants for `ENVIADAS` and `RECEBIDAS`.

```
10
11 object SemantixAgregacaoTeds {
12
13     val formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:00")
14     val zoneId = ZoneId.of("GMT-03:00")
15     val ENVIADAS = "enviadas"
16     val RECEBIDAS = "recebidas"
17
18     def main(args: Array[String]) {
19
20         var partitionParam = ""
21         var selectModeParam = ""
22         try {
23             partitionParam = args(1)
24         } catch {
25             case e: Exception => partitionParam = LocalDate.now().toString
26         }
27
28         try {
29             selectModeParam = args(0)
30         } catch {
31             case e: Exception => selectModeParam = ENVIADAS
32         }
33
34         val sparkConf = new SparkConf()
35             .setAppName("SemantixAgregacaoTeds")
36             .setMaster("yarn-cluster")
```

Entregas

JAVA



The screenshot shows an IDE with a project named 'ext-collector-facebook' located at '~/workspace/ext'. The project structure on the left includes a 'src' directory with 'main' and 'test' subdirectories, each containing 'java' and 'resources' folders. The 'main/java' folder is selected. The main editor displays the file 'ExtCollectorFacebookApplication.java' with the following code:

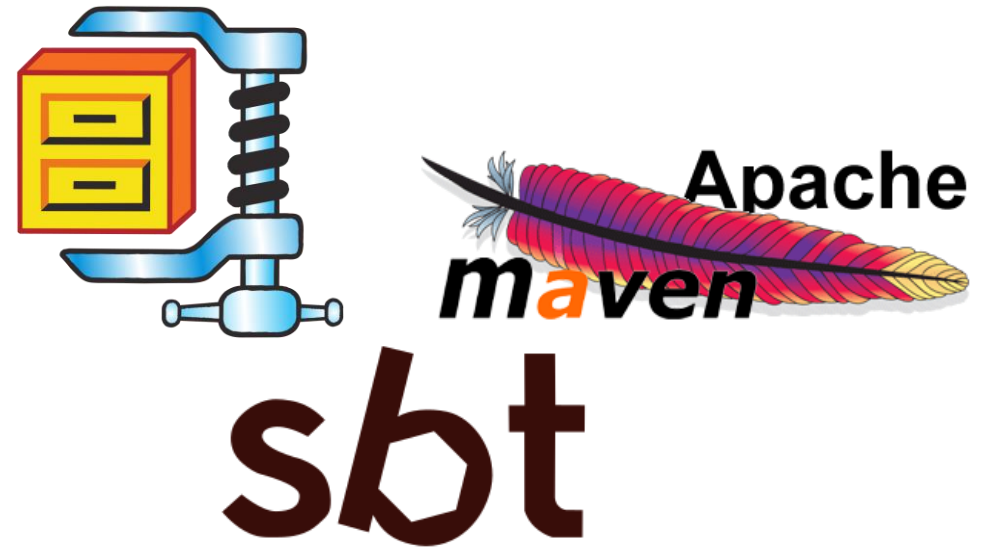
```
1 package br.com.semantix.collector;
2
3 import ...
4
21
22 public class ExtCollectorFacebookApplication {
23
24     private static final Logger log = LoggerFactory.getLogger(MethodHandles.lookup().lookupClass());
25     private static final PropertyReader propertyReader = PropertyReader.getInstance();
26
27     public static void main(String[] args) {
28
29         ValidationUtils.argsValidation(args);
30         log.info(Messages.APP_STARTED, "FACEBOOK");
31         SparkSession sparkSession = initSpark();
32         HandleInterruption handleInterruption = new HandleInterruption(ProcessingTypeEnum.getProcess
33             EXT_COLLECTOR_FACEBOOK.getValue(),
34             PropertyEnum.HADOOP_FS_DEFAULT_NAME.getValue(),
35             propertyReader.read(PropertyEnum.SPARK_TRUSTED_PATH.getValue()));
```

Entregas

- Os artefatos que geramos para os clientes normalmente são divididos em 4 tipos de Arquivos:
 - Scripts
 - JAR
 - Zip
 - .egg

Entregas

- Os artefatos que geramos para os clientes normalmente são divididos em 4 tipos de Arquivos:
 - Scripts
 - JAR
 - Zip
 - .egg



Entregas

- Como executamos

```
spark-submit \  
  --class "br.com.semantix.collector.ExtCollectorAnalyticalApplication" \  
  ext-collector-analytical-1.0.0.jar '20200904' 'RRFT'
```

```
spark-submit --master yarn \  
--py-files semantix_bluesky-0.1.1-py3.6.egg \  
semantix_bluesky-0.1.1-py3.6.egg/__main__.py
```

```
spark-submit --master yarn \  
--py-files sematix_bluesky-0.1.1.zip, __main__.py 1982 reprocess
```

```
spark-submit --master yarn pipeline_oracle_payment.py
```

Entregas

- Ex.: shell

```
spark2-submit \  
--name "semantix-analytical-reprocess" \  
--class "br.com.semantix.collector.ExtCollectorAnalyticalApplication" \  
--master yarn \  
--deploy-mode cluster \  
--conf "spark.dynamicAllocation.enabled=true" \  
--conf "spark.dynamicAllocation.minExecutors=4" \  
--conf "spark.dynamicAllocation.maxExecutors=5" \  
--conf "spark.yarn.maxAppAttempts=1" \  
--files "/opt/semantix/app/lib/log4j.properties" \  
--conf "spark.driver.extraJavaOptions=-Dlog4j.configuration=file:log4j.properties" \  
--conf "spark.executor.extraJavaOptions=-Dlog4j.configuration=file:log4j.properties" \  
--conf spark.yarn.keytab=/keytab_svc_clouderaextcod/svc_clouderaextcod.kt \  
--conf spark.yarn.principal=svc_clouderaextcod@SEMANTIX.CORP \  
--driver-class-path /opt/semantix/app/lib/ojdbc7-12.1.0.3.jar \  
--jars "/opt/semantix/app/lib/ojdbc7-12.1.0.3.jar","/opt/semantix/app/lib/HikariCP-3.4.2.jar" \  
"/opt/semantix/app/collector-analytical/collector-analytical-1.0.0.jar" yyyyMMdd TYPE \  
--
```

Entregas

- o Agendamento:

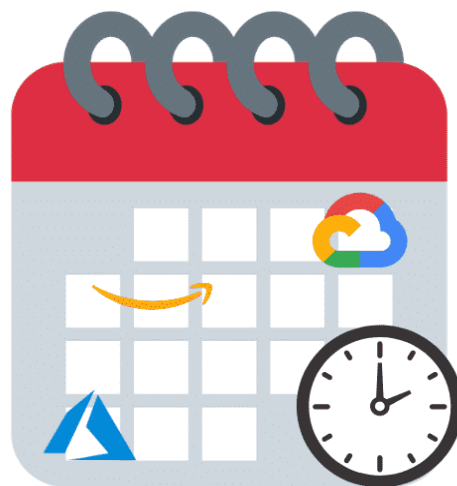
A crontab file has five fields for specifying:

* * * * * command to be executed

```
| | | | |  
| | | | |  
| | | | |  
| | | | | +----- **DAY OF WEEK** (0-6) (Sunday=0)  
| | | | | +----- **MONTH** (1-12)  
| | | | | +----- **DAY OF MONTH** (1-31)  
| | | | | +----- **HOUR** (0-23)  
| | | | | +----- **MINUTE** (0-59)
```



Apache Airflow



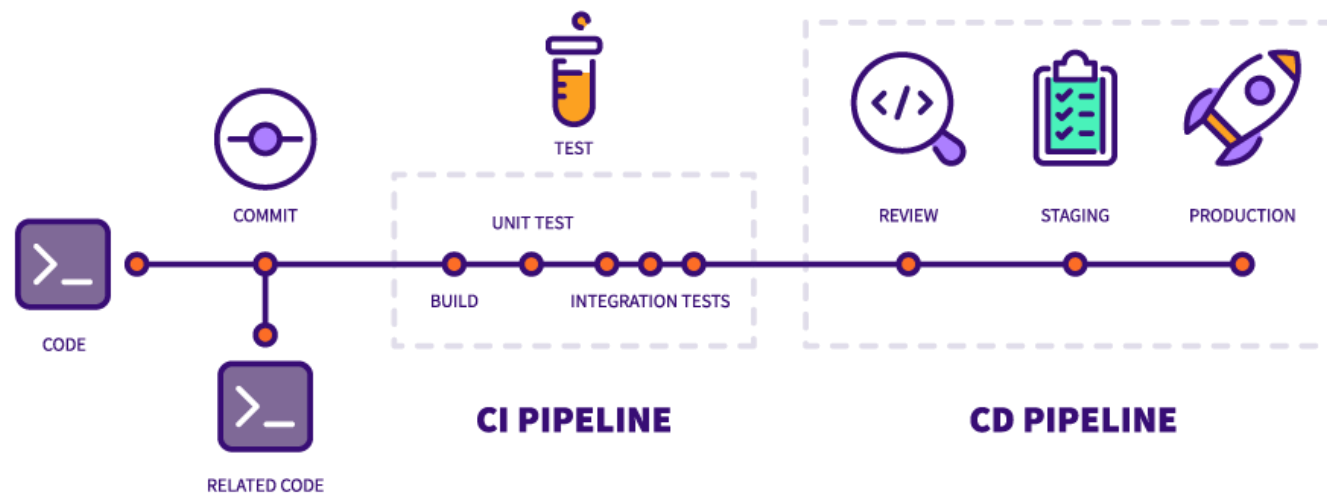
 **bmc**
Control-M

Entregas

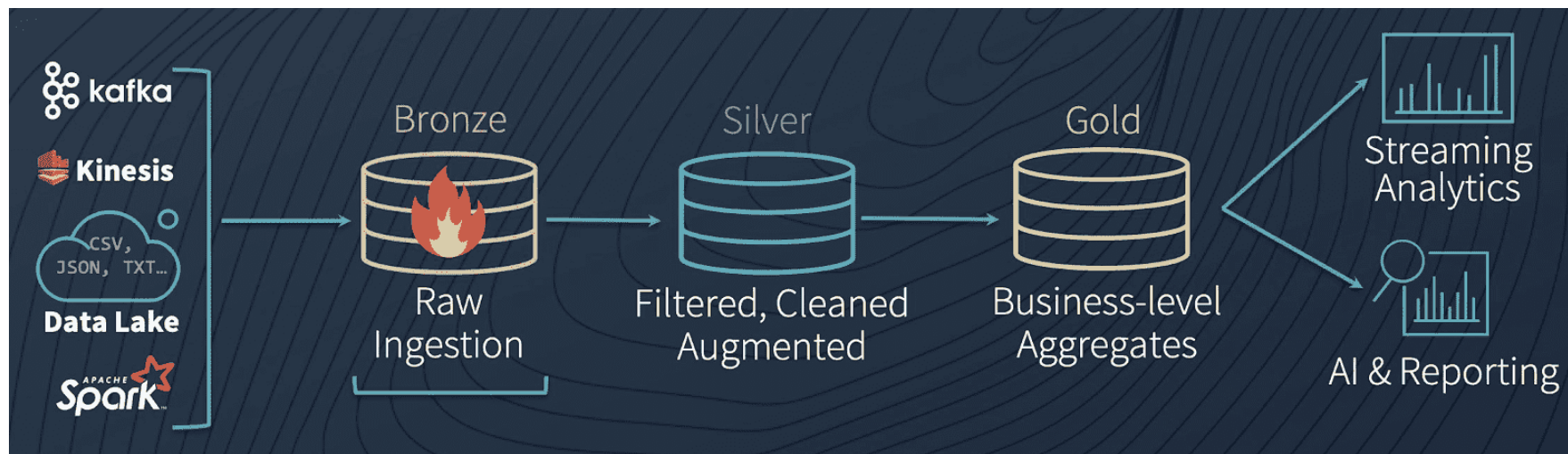
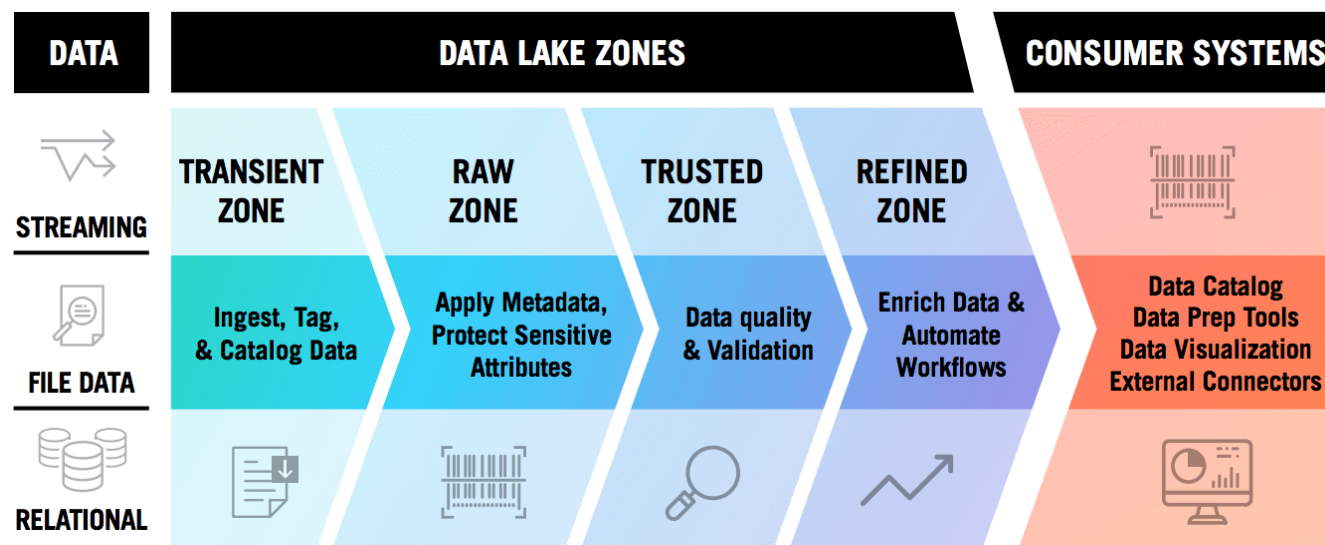
- Deploy:



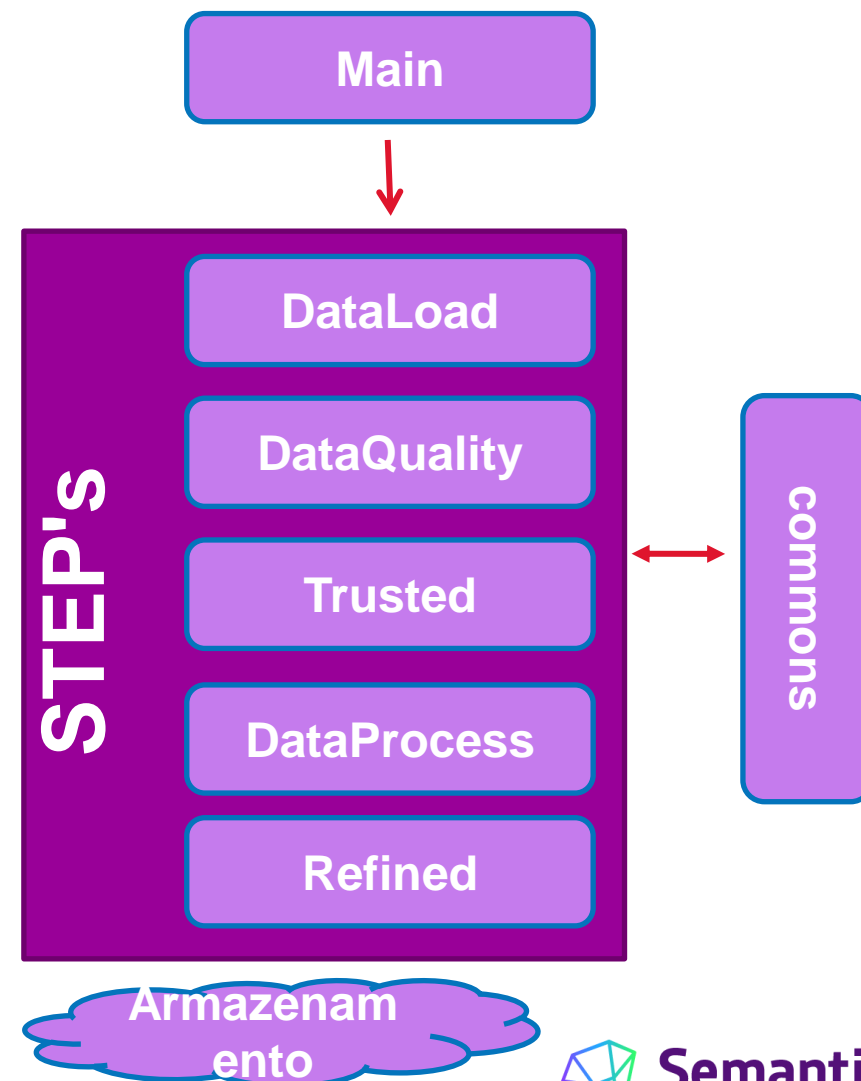
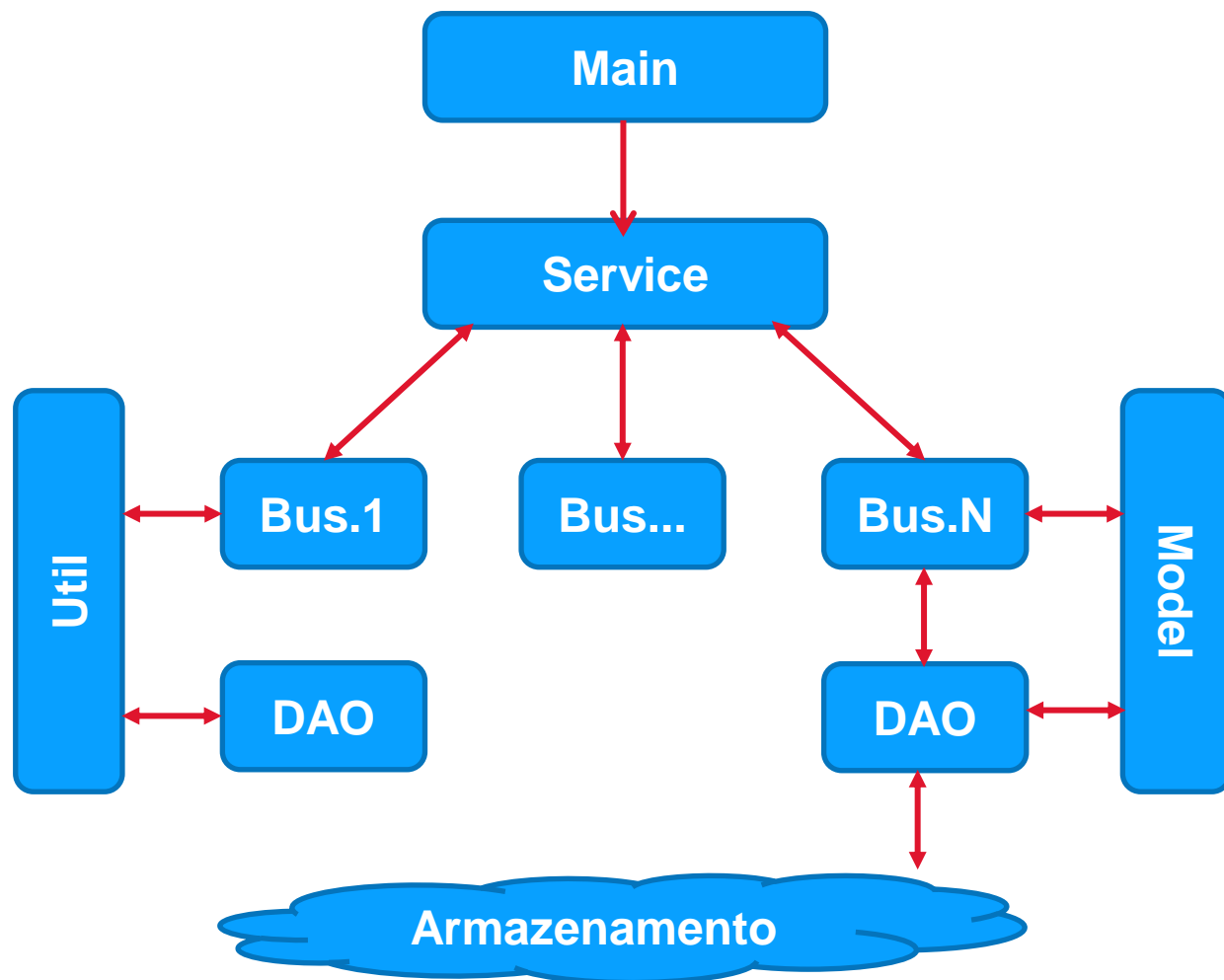
GMUD 



Templates de projetos semantix;



Templates de projetos semantix;



Conclusão

1. Escrevemos código para outras pessoas lerem
2. Precisamos aprender os detalhes da linguagem usada seja: java, python, scala ou R
3. Conceitos de codificação como SOLID nos ajudam a escrever bons códigos
4. Demos um overview sobre etapas de entrega
5. Passamos pelos conceitos de zonas big data
6. Navegamos por modelos produtivos de data application



Semantix[®]

All about data

contato@semantix.com.br

www.semantix.com.br