

Prueba técnica: Data Engineer

Nombre: Cirino Martínez / Data Engineer.

Sección 1: Data pipeline.

Herramientas utilizadas:

1. Base de datos relacional. Postgres
2. Herramienta para ETL. Spark 2.4.
3. Lenguaje de programación. Scala 2.11
4. Java 1.8

1.1 Carga de información.

Elegí la base de datos de Postgres, porque es una base de datos relacional, es open source, se adapta para trabajar en diversas plataformas, tienen buen performance en cuanto al procesamiento del volumen de datos, brinda la seguridad de control a los datos, así como es una herramienta amigable.

1.2 Extracción

El dataset proporcionado **data_prueba_tecnica_(2)_(1).csv**, tiene el siguiente formato:

```
id,name,company_id,amount,status,created_at,paid_at
,,,,,
48ba4bdbfb56ceebb32f2bd0263e759be942af3d,MiPasajefy,cbf1c8b09cd5b549416d49d220a40cbd317f952e,3,voided,19/03/2019,
,,,,,
05fc6f5ac66b6ee7e4253aa5d0c2299eb47aaaf4,MiPasajefy,cbf1c8b09cd5b549416d49d220a40cbd317f952e,3,pending_payment,06/05/2019,
,,,,,
2cdce231c1fc6a2061bfa2f1d978351fe217245d,MiPasajefy,cbf1c8b09cd5b549416d49d220a40cbd317f952e,3,voided,22/02/2019,
,,,,,
```

Fig.1.0. Estructura del archivo data_prueba_tecnica_(2)_(1).csv.

Para trabajar la parte ETL de este set de datos, elegí las siguientes herramientas:

- Spark 2.4.
- Scala 2.11
- Java 1.8

Por qué Spark permite el procesamiento de datos estructurados y no estructurados, además permite procesar volúmenes muy altos de datos (big data), permite utilizar funciones SQL, se integra fácilmente con lenguajes como Scala, Java, Python, R; mantiene una tolerancia a fallos, es ideal para el procesamiento distribuido, además de que es open source y por qué estoy familiarizado con la herramienta.

- Lectura de **data_prueba_tecnica_(2)_(1).csv**

Usando Spark y lenguaje de Scala.

```
def readCSVFile(pathFile: String)(implicit spark: SparkSession): DataFrame = {
  val dfEmpty=spark.emptyDataFrame
  try {
    spark.read
      .option("inferSchema", "true") // Make sure to use string version of true
      .option("header", true)
      .option("dateFormat", "yyyy-MM-dd")
      .option("sep", ",")
      .csv(pathFile)
  }
  catch {
    case ex: AnalysisException =>
      logger.error(s"Check path about $ex ")
      dfEmpty
  }
}
```

Fig.1.1. Funcion readCSVFile en lenguaje de Scala.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	save at Writer scala:37	2020/10/11 20:30:34	2.5 min	1/2	76/201 (2 running)
1	csv at Reader scala:21	2020/10/11 20:30:25	3 s	1/1	1/1
0	csv at Reader scala:21	2020/10/11 20:30:21	3 s	1/1	1/1

Fig.1.2. Tarea de csv at Reader.scala

- Revisar el esquema del dataset:

```
ETLJobOperator x
20/10/11 20:41:22 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all
20/10/11 20:41:22 INFO DAGScheduler: Job 1 finished: csv at Reader.scala:21, took 0.1
root
|-- id: string (nullable = true)
|-- company_name: string (nullable = true)
|-- company_id: string (nullable = true)
|-- amount: decimal(16,2) (nullable = true)
|-- status: string (nullable = true)
|-- created_at: timestamp (nullable = true)
|-- updated_at: timestamp (nullable = true)
```

Fig.1.3. PrintSchema del dataset.

- Así como una visualización de los datos.

id	company_name	company_id	amount	status	created_at	updated_at
893ed946f477bd580a481987e35a10ac463dea42	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	179.71	paid	2019-04-12 00:00:00	2019-04-12 00:00:00
ac8f6474e31d8ccdfc24f69dac9469505038c42	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	3.00	voided	2019-02-03 00:00:00	null
e0416a3061b6cdb955ca6a11fac3344bd4109fd	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	61.29	pending_payment	2019-03-24 00:00:00	null
52ca5cd87accaea8bcc2238ba989188d359bec94	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	73.38	paid	2019-01-30 00:00:00	2019-01-30 00:00:00
26dc18bd602481eb00419bb46913d7394d0ea822	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	89.48	paid	2019-05-04 00:00:00	2019-05-04 00:00:00
a7672f29d782971a8a7c50a962d61735288dc921	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	180.83	paid	2019-04-26 00:00:00	2019-04-26 00:00:00
3e52752b27bc7b7f5365d6888cd7f4216f09c6eb	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	35.27	paid	2019-03-06 00:00:00	2019-03-06 00:00:00
e0ecda7395c577b9a9945277c280328d4bd9252e	MiPasajefy	cbf1c8b09cd5b549416d49d228a40cbd317f952e	215.58	paid	2019-02-06 00:00:00	2019-02-06 00:00:00

Fig.1.4. Show del dataset.

1.3 Transformación.

```

package cms.test.seccion1.joboperation

import ...

object ETLJobOperator extends Attributes(
  ...
) {
  val PATR = "resources/cms/test/resources/data_prueba_tecnica.csv"
  val ETLCargo = new EtlTableCargo()
  val LOGGER = Logger.getLogger(getClass.getName)
  val SPARK = SparkSession
    .builder()
    .config("spark.service.user.postgresql.pass", "cmsdb")
    .config("spark.service.user.postgresql.user", "cms")
    .config("spark.service.user.postgresql.database", "db_test_daen")
    .config("spark.service.user.postgresql.port", "5432")
    .config("spark.service.user.postgresql.host", "localhost")
    .master("local")
    .getOrCreate()

  ETLJobOperator.main(args: Array[String])
}

```

Seccion1. Código de Scala – Spark (ETL)

- Transformación del campo de **created_at**:

Debido a que el campo en el origen tiene diferentes formatos:

created_at
01/05/2019
2019-02-27T00:00:00
20190516
20190121

Fig.1.5. Diferentes formatos en el origen.

Implementé una función para corregirlo.

```

/**
 *
 * @param DateField Nombre de la columna que contiene el dato de fecha en string.
 * @return Column
 */
def transformDateField(DateField: String): Column = {
  when(to_date(col(DateField), fmt = "yyyy-MM-dd").isNotNull,
    to_date(col(DateField), fmt = "yyyy-MM-dd"))
  .when(to_date(col(DateField), fmt = "yyyyMMdd").isNotNull,
    date_format(to_date(col(DateField), fmt = "yyyyMMdd"), format = "yyyy-MM-dd"))
  .when(to_date(col(DateField), fmt = "MM/dd/yyyy").isNotNull,
    to_date(col(DateField), fmt = "MM/dd/yyyy"))
  .when(to_date(col(DateField), fmt = "yyyy MMMM dd").isNotNull,
    to_date(col(DateField), fmt = "yyyy MMMM dd"))
  .when(to_date(col(DateField), fmt = "yyyy MMMM dd E").isNotNull,
    to_date(col(DateField), fmt = "yyyy MMMM dd E"))
  .otherwise(value = "Unknown Format")
  .alias(DateField)
}

```

Fig.1.6. Función en Scala, para formatear el campo de fecha.

- Transformación del campo de **amount**:
Se convirtió a decimal (16,2) el campo.
- Filtros para evitar **Null**:
En los campos de company_id, para evitar null.

Nota. Se realizaron cambios en la definición del esquema propuesto.

id varchar(24) NOT NULL – Se cambió a 40 caracteres, debido a la longitud de los datos.

company_id varchar(24) NOT NULL – Se cambió a 40 caracteres, debido a la longitud de los datos.

```

CREATE TABLE db_test_daen.cargo(
  id varchar(40) NOT NULL,
  company_name varchar(130) NOT NULL,
  company_id varchar(40) NOT NULL,
  amount decimal(16,2) NOT NULL,
  status varchar(30) NULL,
  created_at timestamp not NULL,
  updated_at timestamp null)

```

db_test_daen.cargo	
ABC id	varchar(40) NOT NULL
ABC company_name	varchar(130) NOT NULL
ABC company_id	varchar(40) NOT NULL
123 amount	numeric(16,2) NOT NULL
ABC status	varchar(30)
🕒 created_at	timestamp NOT NULL
🕒 updated_at	timestamp

Fig.1.7. Creación de la tabla cargo en Postgres.

1.4 Dispersión de los datos.

Se crearon las tablas **charges** y **companies**.

Se muestra el diagrama.

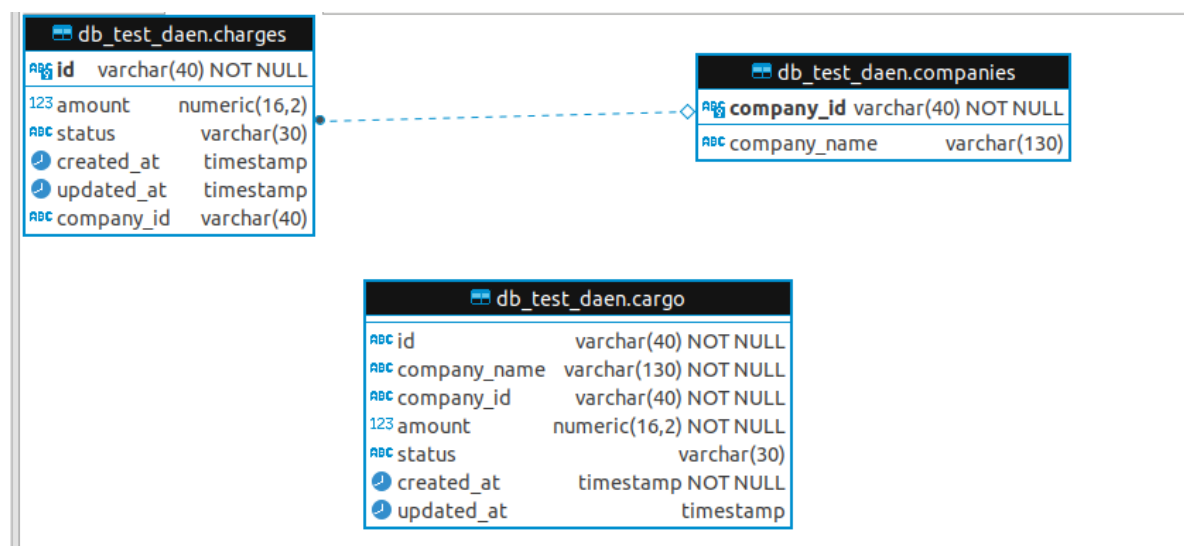


Fig.1.8. Creación de las tablas companies y charges en Postgres.

A manera de ilustración los datos se guardaron en formato Parquet, compresión snappy.

```
def saveResultSetAsParquet(df: DataFrame, saveMode: String, format: String, companyId: String, createdAt: String,
    pathToSave: String): Unit = {
    try {
        df.write
            .mode(saveMode)
            .partitionBy(createdAt, companyId)
            .option("compression", "snappy")
            .format(format)
            .save(pathToSave)
        logger.info("The Data save into the path : ".concat(pathToSave))
    }
    catch {
        case ex: SQLException =>
            logger.error(s"Check table about $ex ")
    }
}
```


Jobs
Stages
Storage
Environment
Executors
SQL

SQL

Running Queries: 1

Completed Queries: 2

Running Queries (1)

ID	Description	Submitted
2	save at Writer.scala:36 +details	2020/10/11 21:57:15

Fig.1.9. Función en Scala para escribir en parquet.













Nombre	Tamaño	Modificación
 part-00001-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.0 kB	21:57
 part-00002-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57
 part-00003-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.0 kB	21:57
 part-00004-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.1 kB	21:57
 part-00013-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57
 part-00016-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57
 part-00017-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.1 kB	21:57
 part-00021-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.0 kB	21:57
 part-00026-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	2.0 kB	21:57
 part-00029-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57
 part-00031-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57
 part-00038-07cbb2cc-28a8-4a01-aa24-5e7e30573ed7.c000.snappy.parquet	1.9 kB	21:57

Fig.1.10. Salida de resultados.

Escritura en la tabla cargo de Postgres.

```
def saveResultJDBC(df: DataFrame, saveOption: String, jdbcUrl: String, jdbcUser: String, jdbcPassword: String,
  tableToWrite: String): Unit = {
  val connectionProperties = new Properties
  connectionProperties.put("user", jdbcUser)
  connectionProperties.put("password", jdbcPassword)
  try {
    df.write
      .mode(saveOption)
      .jdbc(jdbcUrl, tableToWrite, connectionProperties)
  }
  catch {
    case ex: IOException =>
      logger.error(s"Check the path about $ex ")
  }
}
```

Spark 2.4.0 Jobs Stages Storage Environment Executors SQL ETL - Data Pipeline - Test appli

SQL

Running Queries: 1
Completed Queries: 3

Running Queries (1)

ID	Description	Submitted	Duration	Running Job IDs	Succeeded Job IDs	Failed Job IDs
3	jdbc at Writer.scala:42	+details 2020/10/11 21:59:32	19 s	[4]		

Completed Queries (3)

ID	Description	Submitted	Duration	Job IDs
2	save at Writer.scala:36	+details 2020/10/11 21:57:15	2.3 min	[5]
1	show at ESTableCargo.scala:42	+details 2020/10/11 21:57:10	5 s	[4]
0	csv at Reader.scala:26	+details 2020/10/11 21:57:03	3 s	[4]

Fig.1.11. Función en Scala para escribir Postgres por medio de jdbc.

```
select * from db_test_daen.cargo
limit 100;
```

cargo

select * from db_test_daen.cargo limit 100

ID	company_name	company_id	amount	status	created_at	updated_at
893ed946f477bd580a481907e35a10ac463dea42	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	179.71	paid	2019-04-12 00:00:00	2019-04-12 00:00:00
ac8f6474e31d8ccedfc24f69dac9469505038c42	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	voided	2019-02-03 00:00:00	
e0416a3061b6cd955ca60a11fac3344bd4109fd	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	61.29	pending_payment	2019-03-24 00:00:00	
52ca5cd87accaea8bcc2238ba989180d359bec94	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	73.38	paid	2019-01-30 00:00:00	2019-01-30 00:00:00
26dc18bd0692481eb00419bb46913d7394d0ea822	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	89.48	paid	2019-05-04 00:00:00	2019-05-04 00:00:00
a7672f29d782971a8a7c50a962d61735288dc921	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	180.83	paid	2019-04-26 00:00:00	2019-04-26 00:00:00
3e52752b7bc7b7f5365d688cd7f4216f99c6eb	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	35.27	paid	2019-03-06 00:00:00	2019-03-06 00:00:00
0eecd47395c577b9a9945277c2803284d9d9252e	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	215.58	paid	2019-02-06 00:00:00	2019-02-06 00:00:00
c905ae13e48d749cc63a20dac8f5ea9d38f49d	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	197.31	paid	2019-01-25 00:00:00	2019-01-25 00:00:00
ca66d879b49116854cd92e2385900a2c072bd00	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	voided	2019-05-19 00:00:00	
693a1683517be99ca5eb130e0d339754c4265473	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	pending_payment	2019-04-13 00:00:00	
0933af171e7c4f8dac3f5073540773417de1fe35	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	94.47	paid	2019-04-21 00:00:00	2019-04-21 00:00:00
14f56db8bba9973c08fec3ea4bc08e975fa3e9a6	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	voided	2019-01-14 00:00:00	
557e0eddfdc2c2ad00b129521860fb08f9259e	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	57.06	voided	2019-02-13 00:00:00	
44d8756ca27f9f57ec83b060d0747738674ce98	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	145.37	paid	2019-04-09 00:00:00	2019-04-09 00:00:00
34895570786c9df04ed6383a02dfef9ffbac2020	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	56.49	voided	2019-02-15 00:00:00	
e3641f953a24604d031b410b166c63756f141cd0	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	pending_payment	2019-01-23 00:00:00	
2d54b127ab4c1ebd1204614db0d5213db698abda	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	3.00	pending_payment	2019-04-05 00:00:00	
710e11ca1ea376843b28fcc9aaf78aa8454c5b1c	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	54.45	paid	2019-01-06 00:00:00	2019-01-06 00:00:00
c620733387f5901c7ab441b304ce77008ee61c5c	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	82.21	paid	2019-01-28 00:00:00	2019-01-28 00:00:00
51343f71910672f320da9b66628ff555b754796d	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	61.81	paid	2019-04-22 00:00:00	2019-04-22 00:00:00
89969104241b2bf32bf8e8ad7bb5b548d159bb6a	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	124.73	paid	2019-03-13 00:00:00	2019-03-13 00:00:00
c5bc25cb4fd0e384a3cf9f39e31b97663440f04	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	188.75	paid	2019-01-22 00:00:00	2019-01-22 00:00:00
bf0e5f5a067b17992c56cf6a57c411de23d6ea	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	30.24	paid	2019-05-14 00:00:00	2019-05-14 00:00:00
8341fa3927d5dedb8d8d5f9af8173511a52d842c	MiPasajefy	cbf1c8b09cd5b549416d49d220a40cbd317f952e	56.45	naid	2019-02-25 00:00:00	2019-02-25 00:00:00

Save Cancel Script 200 100 100 row(s) fetched - 3ms (+1ms)

Fig.1.12. Salida de resultados en postgres.

Nota. Debido a que existían nombre diferentes nombres de companias, asociados al mismo Company_id, realice la unificación.

```
-- update table cargo
update db_test_daen.cargo
set company_name = 'MiPasajef'
where company_id = 'cbf1c8b09cd5b549416d49d220a40cbd317f952e';

table companies
```

Fig.1.13. Update de Company_name

company_id	company_name
cbf1c8b09cd5b549416d49d220a40cbd317f952e	MiPasajef
8f642dc67fccf861548dfe1c761ce22f795e91f0	Muebles chidos

Fig.1.14. carga de datos a la tabla companies.

select * from db_test_daen.charges limit 100;

id	amount	status	created_at	updated_at	company_id
747abd994e947b3373838a7647ef9bef79ea8809	3.00	voided	2019-01-12 00:00:00		cbf1c8b09cd5b549416d49d220a40cbd317f952e
f4cc3a4197fd5d91c8a8c33fc7b601131c3abc94	3.00	voided	2019-04-09 00:00:00		cbf1c8b09cd5b549416d49d220a40cbd317f952e
850e9a3e88bb8d7dcf3012ad6ea48b872e5745b	35.02	paid	2019-05-14 00:00:00	2019-05-14 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
33cc43a62425c62ff771eccc31f004595c88f9	60.81	paid	2019-03-19 00:00:00	2019-03-19 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
6ef48f01704eb1ae0f776a42da8187d7005a6f59	3.00	voided	2019-05-12 00:00:00		cbf1c8b09cd5b549416d49d220a40cbd317f952e
7482f24a83a85b20c1c6ac3f5ed3102e5c7825b	44.60	paid	2019-02-24 00:00:00	2019-02-24 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
630e5d61d9ae1990d16e7847a6c11dda968a6113	289.94	paid	2019-03-05 00:00:00	2019-03-05 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
c6de5513cf50156ad0f46b2355ca056032f1ed94	599.00	paid	2019-01-14 00:00:00	2019-01-14 00:00:00	8f642dc67fccf861548dfe1c761ce22f795e91f0
7e0ec9b99e509144928e7687700a4a95e36e031	169.15	paid	2019-02-28 00:00:00	2019-02-28 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
cd183e41890f9ec9631ef2040dea10fb8804c16	11969.00	pending_payment	2019-03-19 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
83add3efce6fddf5177c175ef0d60ea1e34d97977	549.00	paid	2019-02-07 00:00:00	2019-02-07 00:00:00	8f642dc67fccf861548dfe1c761ce22f795e91f0
0f47af9c103ef9dd9b116f55ea559360a4a101630	8299.00	paid	2019-02-18 00:00:00	2019-02-18 00:00:00	8f642dc67fccf861548dfe1c761ce22f795e91f0
4fe10d6e06dd47592a7115bd4c03ef48faa74b86	147.50	paid	2019-04-28 00:00:00	2019-04-28 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
8e8e49d7ee9689c2dfa89626b3303d97f66917df	42243.00	pending_payment	2019-03-29 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
ec451874eed8db562b1754a1d324194ee693735a	8058.00	pending_payment	2019-01-24 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
be812306ba75709cc284b97d6a3c22592bcf978f	11489.10	pending_payment	2019-04-29 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
5b8b024cb126f002b75783e6c6a63c7ad2b577c	549.00	pending_payment	2019-02-05 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
082deb4f921183879882ec193e87d5288632dac7	13828.00	pending_payment	2019-01-23 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
fbf1af9c73ccebada1e05c80c9587c0824529c8f5	10648.00	pending_payment	2019-02-23 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
fb3f640e303df3b1f7f3cb09177b70a7c431b053	34973.00	paid	2019-04-16 00:00:00	2019-04-17 00:00:00	8f642dc67fccf861548dfe1c761ce22f795e91f0
fb2e581dd06ed4c51dc4daa35d580939787ca648	7698.00	pending_payment	2019-03-04 00:00:00		8f642dc67fccf861548dfe1c761ce22f795e91f0
6401eed3c9645e82721029ec4798f6dae3b6d91a	86.67	paid	2019-01-07 00:00:00	2019-01-07 00:00:00	cbf1c8b09cd5b549416d49d220a40cbd317f952e
cc6e8ed7a47209df3a1475285ef400e0b73e2cc4	8099.10	paid	2019-05-14 00:00:00	2019-05-14 00:00:00	8f642dc67fccf861548dfe1c761ce22f795e91f0

Save Cancel Script 200 100 100 row(s) fetched - 3ms

Fig.1.14. carga de datos a la tabla charges.

1.5 SQL.

Creación de la vista

```
create view vw_monto_transaccionado AS
select company_id, created_at, sum(amount) as "monto_transaccion"
from db_test_daen.charges
group by (company_id, created_at )
```

Statistics

create view vw_monto_transaccionado AS select cc | Enter a SQL expression to filter results (use Ctrl+Space)

ame	Value
Updated Rows	0
Query	create view vw_monto_transaccionado AS select company_id, created_at, sum(amount) as "monto_transaccion" from db_test_daen.charges group by (company_id, created_at)
Finish time	Sun Oct 11 22:16:50 CDT 2020


```
graph LR
    charges[db_test_daen.charges] -.-> companies[db_test_daen.companies]
    cargo[db_test_daen.cargo] -.-> companies
    vw[db_test_daen.vw_monto_transaccionado] -.-> charges
```

db_test_daen.charges

ABC	id	varchar(40) NOT NULL
123	amount	numeric(16,2)
ABC	status	varchar(30)
ABC	created_at	timestamp
ABC	updated_at	timestamp
ABC	company_id	varchar(40)

db_test_daen.companies

ABC	company_id	varchar(40) NOT NULL
ABC	company_name	varchar(130)

db_test_daen.cargo

ABC	id	varchar(40) NOT NULL
ABC	company_name	varchar(130) NOT NULL
ABC	company_id	varchar(40) NOT NULL
123	amount	numeric(16,2) NOT NULL
ABC	status	varchar(30)
ABC	created_at	timestamp NOT NULL
ABC	updated_at	timestamp

db_test_daen.vw_monto_transaccionado

ABC	company_id	varchar(40)
ABC	created_at	timestamp
123	monto_transaccion	numeric

select * from db test daen.vw monto transaccionado
limit 100;

vw_monto_transaccionado

select * from db_test_daen.vw_monto_transaccion

company_id	created_at	monto_transaccion
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-30 00:00:00	6070.39
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-03-18 00:00:00	4050.64
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-02-14 00:00:00	31542.00
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-01-30 00:00:00	3743.10
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-03-26 00:00:00	5703.31
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-05 00:00:00	5184.97
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-22 00:00:00	3650.23
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-04 00:00:00	6349.69
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-01-21 00:00:00	19592.00
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-04-18 00:00:00	6617.00
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-17 00:00:00	7058.25
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-03-14 00:00:00	15489.00
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-01-07 00:00:00	26754.30
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-04-09 00:00:00	63863.00
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-03-17 00:00:00	22818.00
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-03-22 00:00:00	7654.85
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-04-17 00:00:00	12273.05
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-04-17 00:00:00	5819.90
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-02-11 00:00:00	8218.57
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-03-01 00:00:00	44466.58
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-04-26 00:00:00	13828.33
cbf1c8b09cd5b549416d49d220a40cbd317f952e	2019-05-05 00:00:00	3112.96
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-02-28 00:00:00	11416.05
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-03-04 00:00:00	7698.00
8f642dc67fccf861548dfe1c761ce22f795e91f0	2019-02-19 00:00:00	10997.00

Fig.1.15. Resultados de la vista vw_monto_transaccionado

Sección 2: Scala.

Desarrollado en Scala 2.11

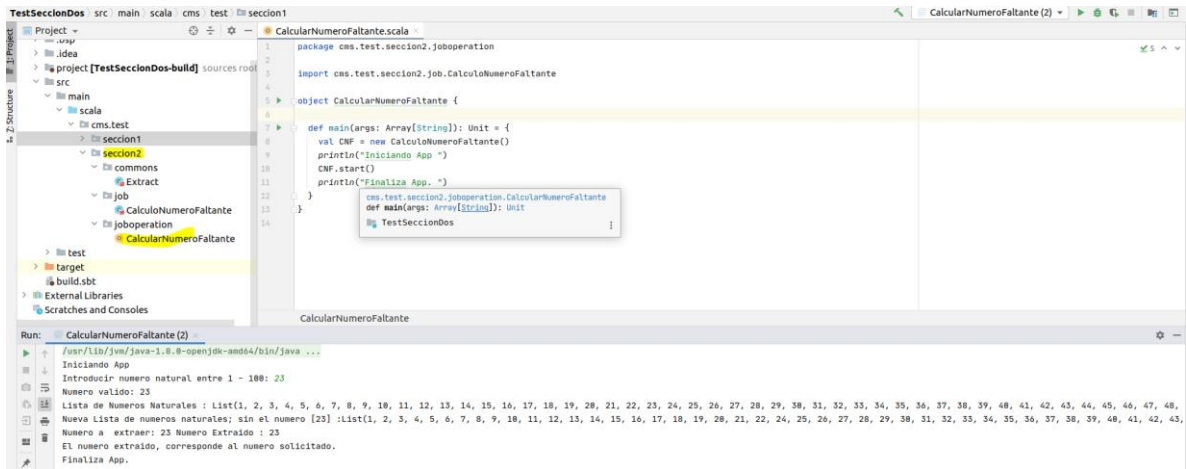


Fig.2.0. Seccion2

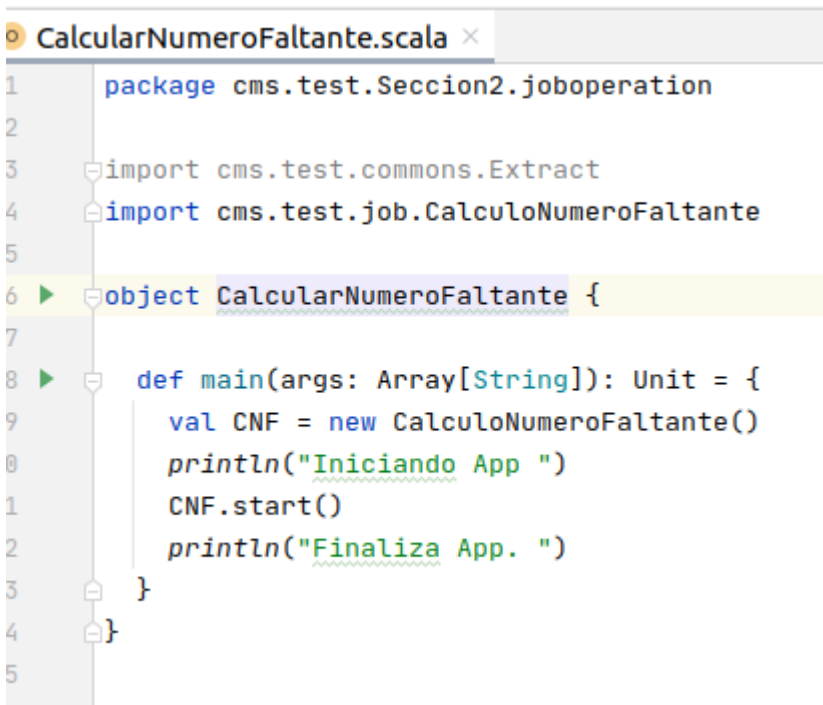


Fig.2.1. Método main()

```
Run: CalcularNumeroFaltante (2)
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Iniciando App
Introducir numero natural entre 1 - 100: 5055F5F
El caracter introducido no corresponde a un valor entero.
Numero invalido [ 5055F5F ]. (Favor de introducir un numero entre 1 - 100: -45454
Numero invalido [ -45454 ]. (Favor de introducir un numero entre 1 - 100: 3436565
Numero invalido [ 3436565 ]. (Favor de introducir un numero entre 1 - 100: 6
El caracter introducido no corresponde a un valor entero.
Numero invalido [ 6 ]. (Favor de introducir un numero entre 1 - 100: 15
Numero valido: 5
Lista de Numeros Naturales : List[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
Nueva Lista de numeros naturales; sin el numero 5: List[1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
Numero a extraer: 5 Numero Extraido : 5
El numero extraido, corresponde al numero solicitado.
Finaliza App.

Process finished with exit code 0
```

Fig.2.2. Implementación de las funcionalidades requeridas.