

Procedimiento para reconvertir bases de datos de Centro Salud Higea – Fundación

Base de Datos Sanos (PostgreSQL)

Paso 0. (Preparación)

- Crear copia de seguridad de la base de datos
- Crear un duplicado de la base de datos sanos : CREATE DATABASE sanos_bs WITH TEMPLATE sanos OWNER postgres; (Opcional)
- Dar permiso al usuario **reconversion_user** en **pg_hba.conf**
- **Ejecutar el programa de reconversión (python) con privilegios root**
- **Editar y colocar el nombre de la base de datos en el programa python**

Paso 1. Crear una función en la base de datos a reconvertir (PostgreSQL)

La función, analiza los objetos, tablas y campos de la base de datos, realizando inicialmente sobre las tablas un conteo del número de registros, excluyendo las tablas sin registro, luego hará un análisis sobre los campos de tipo numeric, creando una tabla solo con estos campos, llamada sanos_diccionario, el procedimiento almacenado tiene la facilidad de excluir tablas y campos.

```
CREATE OR REPLACE function sp_sanosDiccionarios (valor INT)
returns void
language plpgsql AS
$$
DECLARE registros INT = 0;
entidad VARCHAR(100) := NULL;
comando VARCHAR(1000) := NULL;
tupla RECORD;
columna RECORD;
-- DECLARE tabla CURSOR FOR
-- SELECT table_schema,table_name
-- FROM information_schema.tables
-- WHERE table_schema NOT IN ('pg_catalog','information schema') and table type = 'BASE TABLE'
-- AND table_name NOT IN ('inventario_articulo_ambulatorio','aps_cliente_especialidad')
-- ORDER BY table_schema,table_name;
-- BEGIN
DROP TABLE IF EXISTS sanos_diccionario;
CREATE TABLE sanos_diccionario (esquema varchar(50), nombretabla varchar(50), nombrecampo varchar(50));
FOR tupla IN tabla LOOP
entidad = tupla.table_schema || '.' || tupla.table_name;
EXECUTE format('SELECT COUNT (*) FROM %s', entidad) INTO registros;
IF registros > 0 THEN
DECLARE campos CURSOR FOR SELECT i.table_schema as esquema, i.table_name as nombre_t, i.column_name as nombre_c
FROM information_schema.columns i
where i.data_type='numeric'
AND i.column_name NOT in ('factor_erp','conteo','equivalencia','ajuste','factor','p_iva','p_descuento')
AND i.column_name NOT ILIKE '%cantidad%'
AND i.column_name NOT ILIKE '%existencia%'
AND i.column_name NOT ILIKE '%porcentaje%'
AND i.column_name NOT ILIKE '%peso%'
AND i.column_name NOT ILIKE '%porc%'
AND i.column_name NOT ILIKE '%consumido%'
AND i.column_name NOT ILIKE '%dosis%'
AND i.column_name NOT ILIKE '%cant%'
AND i.column_name NOT ILIKE '%valor%';
-- BEGIN
-- for columna in campos LOOP
-- insert into sanos_diccionario(esquema,nombretabla,nombrecampo) VALUES (columna.esquema,columna.nombre_t,columna.nombre_c);
-- END LOOP;
-- END;
END IF;
END LOOP;
END;
$$;
```

Paso 2. Ejecutar la función : SELECT sp_sanosDiccionarios (1);

Paso 3. Verificar la creación de la tabla :

```
SELECT * FROM sanos_diccionario;
```

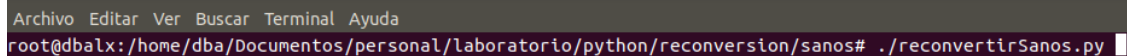
Paso 4. Otorgar permisos al usuario **reconversion_user** sobre toda la base de datos

```
DO $$
DECLARE usuarios VARCHAR(100)[] := array ['reconversion_user'];
DECLARE usuario VARCHAR(100) := NULL;
DECLARE entidad_esquema VARCHAR(100) := NULL;
DECLARE nombre_esquema RECORD;
DECLARE esquema CURSOR FOR SELECT DISTINCT table_schema
FROM information_schema.tables
WHERE table_schema NOT IN ('pg_catalog','information_schema') and table_type = 'BASE TABLE';
BEGIN
FOR login IN array_lower(usuarios, 1)..array_upper(usuarios, 1) LOOP
usuario := usuarios[login];
FOR nombre_esquema IN esquema LOOP
EXECUTE format('GRANT USAGE ON SCHEMA %s TO %s;', nombre_esquema.table_schema,usuario);
EXECUTE format('GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA %s TO %s;', nombre_esquema.table_schema,usuario);
EXECUTE format('GRANT SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA %s TO %s;', nombre_esquema.table_schema,usuario);
EXECUTE format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA %s TO %s;', nombre_esquema.table_schema,usuario);
END LOOP;
END LOOP;
END;
$$
```

Paso 5. Desactivar triggers

```
ALTER TABLE presupuesto.baremo_detalle DISABLE TRIGGER trg_biu_baremo_detalle_hm;
ALTER TABLE inventario_detalle_movimiento DISABLE TRIGGER trg_bidu_invdetmovimiento;
```

Paso 6. Ejecutar el programa **reconvertirSanos.py** para iniciar el proceso de reconversión. Desde un terminal de linux y usuario con privilegio root.



```
Archivo Editar Ver Buscar Terminal Ayuda
root@dbalx:/home/dba/Documentos/personal/laboratorio/python/reconversion/sanos# ./reconvertirSanos.py
```

Paso 7. Habilitar triggers

```
ALTER TABLE presupuesto.baremo_detalle ENABLE TRIGGER trg_biu_baremo_detalle_hm;
ALTER TABLE inventario_detalle_movimiento ENABLE TRIGGER trg_bidu_invdetmovimiento;
```

Programa reconvertirSanos.py

```
1  #!/usr/bin/python3
2  import pandas as pd
3  from datetime import datetime
4
5  import psycopg2 #Postgres
6
7  #Definiciones
8  driver = 'psql'
9  usuario = 'reconversion_user'
10 credenciales = 'xxxxxxxxxx'
11 server = '34.206.176.180'
12 puerto = '5432'
13 dbname = db = 'sanos_test'
14
15
16 formula = 'ROUND({}/1000000,2)'
17 scriptsSQL = []
18
19 ruta = '/home/dba/xxxxxxxxxxxxxxxxxxxxxx/reconversion/sanos/'
20 diccionariocsv = ruta + 'diccionariosanos.csv'
21 updatescsv = ruta + 'updatessanos.csv'
22
23 select_distinct_diccionario = 'SELECT DISTINCT nombretabla,esquema FROM sanos_diccionario ORDER BY 2,1;'
24 select_campos_tablas = 'SELECT nombrecampo FROM sanos_diccionario WHERE nombretabla = \'{}\'' ORDER BY 1;'
25
26 connection = psycopg2.connect(user = usuario, password = credenciales, host = server, port = puerto, database = dbname)
27
28 def leer_tabla_diccionario (cursor):
29     #Ejecutar store procedure para crear la tabla Diccionario con los campos nombretabla, nombrecampo, espk
30     #cursor.execute(execute_storeprocedure)
31     #time.sleep(360)
32     #input('Teclee cualquier tecla para continuar ...')
33     cursor.execute(select_distinct_diccionario)
34     data = cursor.fetchall()
35     tablas = []
```

```

37     # Extraer solo las tablas
38     for t in range(len(data)):
39         tablas.append(list(data[t])[1]+'.'+list(data[t])[0])
40
41     # Extraer los campos por cada tabla
42     camposxtablas = []
43     for t in range(len(tablas)):
44         registro = []
45         cursor.execute(select_campos_tablas.format(tablas[t].split('.')[1]))
46         data = cursor.fetchall()
47         #Tablas con uno o mas campos
48         if len(data) > 0:
49             registro.append(tablas[t])
50             print('Extrayendo ..... +++ {}'.format(tablas[t]))
51             for c in range(len(data)):
52                 registro.append(data[c][0])
53             camposxtablas.append(registro)
54
55     print('Creando archivos ... ')
56     diccionario = open(diccionariocsv, "w")
57     scriptsupdates = open(updatescsv, "w")
58     for t in range(len(camposxtablas)):
59         estructura = camposxtablas[t][0]
60         sentenciaSQL = 'UPDATE ' + camposxtablas[t][0] + ' SET '
61         for c in range(1,len(camposxtablas[t])):
62             sentenciaSQL += camposxtablas[t][c] + ' = ' + formula.format(camposxtablas[t][c]) + ', '
63             estructura += ',' + camposxtablas[t][c]
64         scriptsSQL.append(sentenciaSQL[:-2] + ';')
65         diccionario.write(estructura + '\n')
66         scriptsupdates.write(sentenciaSQL[:-2] + ';' + '\n')
67     diccionario.close()
68     scriptsupdates.close()

```

```

69
70     print('Iniciando ejecucion de updates ... ')
71     for i in range(len(scriptsSQL)):
72         print(scriptsSQL[i])
73         cursor.execute(scriptsSQL[i])
74         connection.commit()
75
76
77
78     def iniciar():
79         tiempo_inicial = datetime.now()
80         try :
81             cursor = connection.cursor()
82             leer_tabla_diccionario(cursor)
83         except (Exception, psycopg2.Error) as error :
84             print ("Error mientras se conecta al Servidor PostgreSQL Server ... Error nro.: ", error.args[0])
85         finally:
86             #Cerrar conexion
87             if(connection):
88                 cursor.close()
89                 connection.close()
90                 tiempo_final = datetime.now()
91                 print('Proceso de reconversion finalizada, tiempo estimado de reconversion : ', tiempo_final - tiempo_inicial)
92
93     if __name__ == '__main__':
94         iniciar()

```

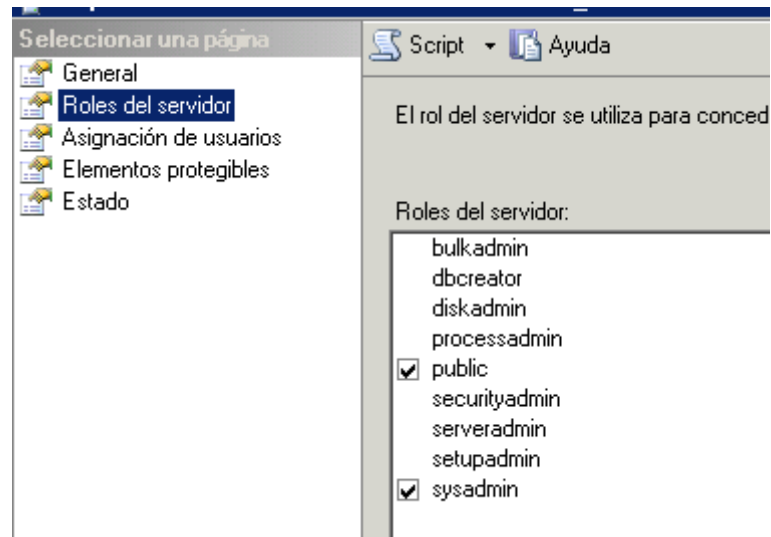
Base de Datos BDNBS | CDHBS (SQL Server)

Paso 0. (Preparación)

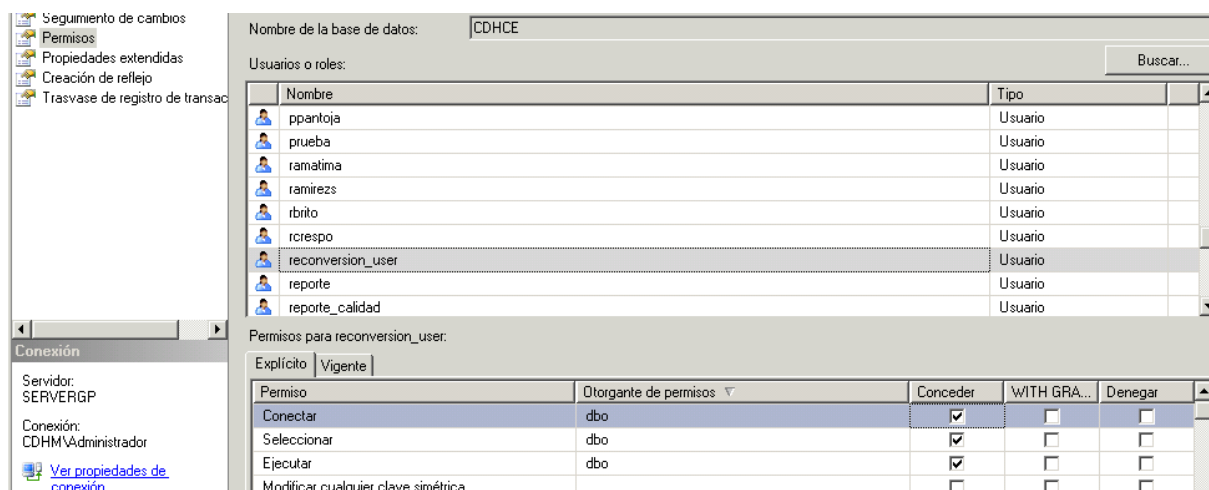
- Verificar que no existan las tablas **BDNDiccionario** y **CDHDiccionario** en la **tempdb** (Opcional)
- Crear copia de seguridad de la base de datos
- Dar permiso al usuario **reconversion_user** en **BDNBS | CDHBS**
- **Ejecutar el programa de reconversión (python) con privilegios root**
- **Editar y colocar el nombre de la base de datos en el programa python**

Asignación del usuario y roles de base de la base de datos

Asignación de roles de servidor



Asignación de permisos



Paso 1. Crear un procedimiento almacenado en la base de datos a reconvertir

El procedimiento almacenado, analiza los objetos, tablas y campos de la base de datos, realizando inicialmente sobre las tablas un conteo del número de registros, excluyendo las

tablas sin registro, luego hará un análisis sobre los campos de tipo numeric, creando una tabla solo con estos campos, llamada **BDNDiccionario** o **CDHDiccionario** en la base de datos tempdb, el procedimiento almacenado tiene la facilidad de excluir tablas y campos.


```

USE [BONCE]
GO
/***** Object: StoredProcedure [dbo].[spBDNDiccionario]    Script Date: 13/09/2021 16:27:51 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[spBDNDiccionario] @atributos INT OUTPUT
AS
BEGIN
DECLARE @query nvarchar(100),
        @parametros nvarchar(100),
        @tabla AS nvarchar(50),
        @campo AS nvarchar(50),
        @sql AS nvarchar(4000),
        @returnAsSelect int = 0,
        @registros AS INT;
IF OBJECT_ID(N'tempdb..BDNDiccionario',N'U') IS NOT NULL
BEGIN
DROP TABLE tempdb..BDNDiccionario;
CREATE TABLE tempdb..BDNDiccionario (id int IDENTITY (1,1) NOT NULL,nombretabla varchar(50),nombrecampo varchar(50),espk int,
CONSTRAINT PK_BDNDiccionario_id PRIMARY KEY CLUSTERED (id));
END;
ELSE
BEGIN
CREATE TABLE tempdb..BDNDiccionario (id int IDENTITY (1,1) NOT NULL,nombretabla varchar(50),nombrecampo varchar(50),espk varchar(2),
CONSTRAINT PK_BDNDiccionario_id PRIMARY KEY CLUSTERED (id));
END
DECLARE Tabla CURSOR FOR SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE='BASE TABLE' AND
TABLE_NAME NOT IN ('aplicacion','ASILOC90','autotemp','BM40100','DTC_MESES','DTT_Version',
'eConnect Out Setup','frrl_acct_code','IGP_REPLICA','IGP_SET000',
'igpsopl0100historica','palbrdy','respaldo RM00401',
'respaldo RM20201','RM00401_RESPALDO','RM20201_RESPALDO',
'ReportSetup','tmp_IGP_REPLICA','usuario','rol','rol_usuario','logxierp',
'CM20100','IV00111','IV00102','DTRMTRX100','DTTSYIMP100','UPR40200')
ORDER BY TABLE_NAME;
OPEN Tabla;
FETCH NEXT FROM Tabla INTO @tabla;

WHILE @@fetch_status = 0
BEGIN
SET @query = N'SET @cuantos_registros = (SELECT COUNT(*) FROM ' + @tabla + ')';
SET @parametros = N'@cuantos_registros INT OUTPUT';
EXECUTE sp_executesql @query, @parametros, @cuantos_registros=@registros OUTPUT;
IF (@registros > 0)
INSERT INTO tempdb..BDNDiccionario(nombretabla, nombrecampo, espk)
SELECT @tabla, COLUMN_NAME, 0 FROM Information_Schema.Columns WHERE TABLE_NAME = @tabla
AND DATA TYPE = 'numeric'
AND COLUMN_NAME NOT IN ('QUANTITY','NOTEINDX','EmailMaxFileSize','HRSPRSHT','DOLRAMNT','DTSEQUUM','FactorUT',
'TXDTLPCT','TDTABMIN','valor_nominal','porc_comision','porc_ret_islr','td_monto_porcentaje',
'td_porcentaje','cantidad_articulo','lista_precio_dcto_prct','islr','descuento_a_factura',
'monto_recargo','monto_descuento')
AND COLUMN_NAME NOT LIKE '%QTY%'
AND COLUMN_NAME NOT LIKE '%ATY%'
AND COLUMN_NAME NOT LIKE '%DLR%'
AND COLUMN_NAME NOT LIKE '%Record%'
AND COLUMN_NAME NOT LIKE '%porcentaje%'
AND COLUMN_NAME NOT LIKE '%porc_%'
AND COLUMN_NAME NOT LIKE '%prcnt_%'
AND COLUMN_NAME NOT LIKE '%RECNUM%';
FETCH NEXT FROM Tabla INTO @tabla;
END;
CLOSE Tabla;
DEALLOCATE Tabla;
SELECT * INTO tempdb.dbo.BDNDiccdatos FROM tempdb..BDNDiccionario;
DECLARE @cuantos INT = (SELECT COUNT(*) FROM tempdb.dbo.BDNDiccdatos);
WHILE @cuantos > 0
BEGIN
DECLARE @id INT = (SELECT TOP(1) id FROM tempdb..BDNDiccdatos ORDER BY id);
DECLARE @nombretabla VARCHAR(50) = (SELECT TOP(1) nombretabla FROM tempdb..BDNDiccdatos ORDER BY id);
DECLARE @nombrecampo VARCHAR(50) = (SELECT TOP(1) nombrecampo FROM tempdb..BDNDiccdatos ORDER BY id);
UPDATE tempdb..BDNDiccionario SET tempdb..BDNDiccionario.espk =
(SELECT COUNT(*)
FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
WHERE COLUMN_NAME = @nombrecampo AND CONSTRAINT_NAME = 'PK' + @nombretabla)
WHERE tempdb..BDNDiccionario.nombretabla = @nombretabla AND tempdb..BDNDiccionario.nombrecampo = @nombrecampo;
DELETE tempdb.dbo.BDNDiccdatos WHERE id = @id;
SET @cuantos = (SELECT COUNT(*) FROM tempdb..BDNDiccdatos);
END;
DROP TABLE tempdb..BDNDiccdatos;
SET @atributos = (SELECT COUNT(*) FROM tempdb..BDNDiccionario);
END;

```

Paso 2. Ejecución del procedimiento almacenado spBDNDiccionario

```

DECLARE @t int = 0;
EXECUTE spBDNDiccionario @t output;
SELECT @t;

```

Paso 3. Ejecutar el programa python **reconvertirHigea.py** para iniciar el proceso de reconversión. Desde un terminal de linux y usuario con privilegio root.

```
dba@dbalx:~/Documentos/personal/laboratorio/python/reconversion/gp$ ./reconvertirFundacion.py
```

Paso 3. Repetir los pasos para la siguiente base de datos a reconvertir

Programa reconvertirFundacion.py

```

1  #!/usr/bin/python3
2
3  import pandas as pd
4  from datetime import datetime
5  import csv
6  import time
7  import pyodbc  #SQL Server
8
9  # Definiciones
10
11  db = 'BDNBS'
12
13  ruta = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
14  diccionariocsv = ruta + 'diccionarioFundacion.csv'
15  updatescsv = ruta + 'updatesFundacion.csv'
16
17  connection = pyodbc.connect("Driver={ODBC Driver 17 for SQL Server};"
18                             "Server=54.81.168.58\\SERVERGP,1433;"
19                             "Database=BDNCE;"
20                             "uid=reconversion_user;"
21                             "pwd=12345678;"
22
23  #Se sugiere ejecutar el store procedure desde el servidor
24  #execute_storeprocedure = 'DECLARE @resultado INT; \
25  #                               EXECUTE ' + db + '..TablasNoVaciasV2 @resultado OUTPUT;\
26  #                               SELECT @resultado AS Resultado;\'
27
28  #formula = 'ROUND({}/1000000,5)'
29  # CAST(ROUND(2325456.4566/1000000.0, 5) AS NUMERIC(19,2))
30  formula = 'CAST(ROUND({}/1000000.0, 5) AS NUMERIC(19,2))'
31  scriptsSQL = []
32
33  #select_diccionario = 'SELECT * FROM tempdb..Diccionario;'
34  select_distinct_diccionario = 'SELECT DISTINCT nombretabla FROM tempdb..BDNDiccionario ORDER BY 1;'
35  select_campos_tablas = 'SELECT nombrecampo FROM tempdb..BDNDiccionario WHERE nombretabla = \'{}\'' AND espk != 1 ORDER BY 1;\'
36

```



```

37 def leer_tabla_diccionario (cursor):
38     #Ejecutar store procedure para crear la tabla Diccionario con los campos nombretabla, nombrecampo, espk
39     #cursor.execute(execute_storeprocedure)
40     #time.sleep(360)
41     #input('Teclee cualquier tecla para continuar ...')
42     cursor.execute(select_distinct_diccionario)
43     data = cursor.fetchall()
44     tablas = []
45
46     # Extraer solo las tablas
47     for t in range(len(data)):
48         tablas.append(list(data[t]))
49
50     # Extraer los campos por cada tabla
51     camposxtablas = []
52     for t in range(len(tablas)):
53         registro = []
54         cursor.execute(select_campos_tablas.format(tablas[t][0]))
55         data = cursor.fetchall()
56         #Tablas con uno o mas campos
57         if len(data) > 0:
58             registro.append(tablas[t][0])
59             print('Extrayendo ..... +++ {} '.format(tablas[t][0]))
60             for c in range(len(data)):
61                 registro.append(data[c][0])
62             camposxtablas.append(registro)
63

```

```

64     print('Creando archivos ... ')
65     diccionario = open(diccionariocsv, "w")
66     scriptsupdates = open(updatescsv, "w")
67     for t in range(len(camposxtablas)):
68         estructura = camposxtablas[t][0]
69         sentenciaSQL = 'UPDATE ' + camposxtablas[t][0] + ' SET '
70         for c in range(1,len(camposxtablas[t])):
71             sentenciaSQL += camposxtablas[t][c] + ' = ' + formula.format(camposxtablas[t][c]) + ', '
72             estructura += ',' + camposxtablas[t][c]
73         scriptssql.append(sentenciaSQL[:-2] + ';')
74         diccionario.write(estructura + '\n')
75         scriptsupdates.write(sentenciaSQL[:-2] + ';' + '\n')
76     diccionario.close()
77     scriptsupdates.close()
78
79     print('Iniciando ejecucion de updates ... ')
80     for i in range(len(scriptssql)):
81         print(scriptssql[i])
82         cursor.execute(scriptssql[i])
83         connection.commit()
84
85 def iniciar():
86
87     print('Iniciando proceso para reconvertir base de datos {}'.format(db))
88     tiempo_inicial = datetime.now()
89     try :
90         cursor = connection.cursor()
91         leer_tabla_diccionario(cursor)
92     except pyodbc.Error as error :
93         print ("Error mientras se conecta al Servidor SQL Server ... Error nro.: ", error.args[0])
94     finally :
95         #Cerrar conexion
96         if (connection) :

```

```

96         if (connection) :
97             cursor.close()
98             connection.close()
99             tiempo_final = datetime.now()
100             print('Proceso de reconversion finalizada, tiempo estimado de reconversion : ', tiempo_final - tiempo_inicial)
101
102
103 if __name__ == '__main__':
104     iniciar()
105

```