ALFACONVERTER

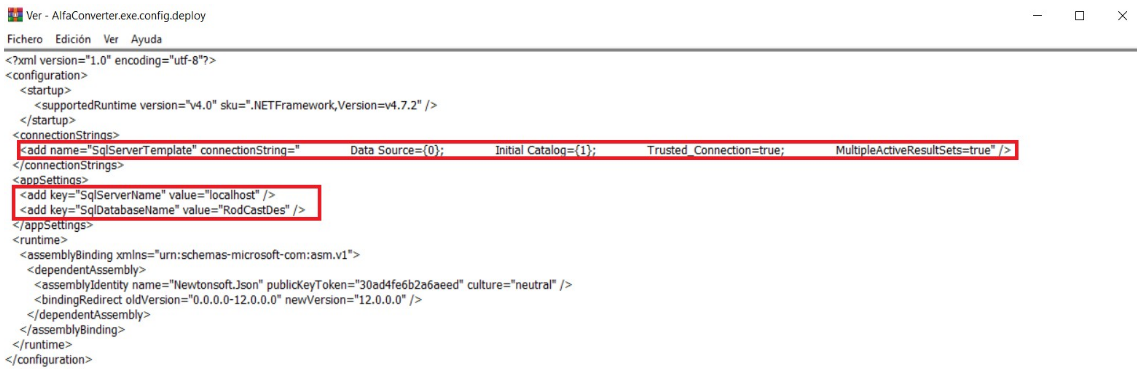
User Guide

FEBRUARY

KYBELE

Initial Setup

In the application directory there is a file called “**AlfaConverter.exe.config**” whose content allows you to configure some basic parametersThe most important are:


Ver - AlfaConverter.exe.config.deploy
Fichero Edición Ver Ayuda

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
 <connectionStrings>
  <add name="SqlServerTemplate" connectionString="          Data Source={0};          Initial Catalog={1};          Trusted_Connection=true;          MultipleActiveResultSets=true" />
 </connectionStrings>
 <appSettings>
  <add key="SqlServerName" value="localhost" />
  <add key="SqlDatabaseName" value="RodCastDes" />
 </appSettings>
 <runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
    <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="12.0.0.0" />
   </dependentAssembly>
  </assemblyBinding>
 </runtime>
</configuration>
```

Figure 1: Important fields for AlfaConverter configuration

configuration/connectionStrings

Contains a database connection string with name “SqlServerTemplate”, which is used by the application to connect to an SQL Server (parameter {0}) and to a database (parameter {1}).

configuration/appSettings

They contain key-value type configurations. There are two:

- **SqlServerName**: Name of the server to which it connects by default at startup.
- **SqlDatabaseName**: Name of the database to which it connects by default at startup.

Alfaconverter use

When you open the desktop application, it initially connects to the server database configured in the App.config file. For example, see the following figure:

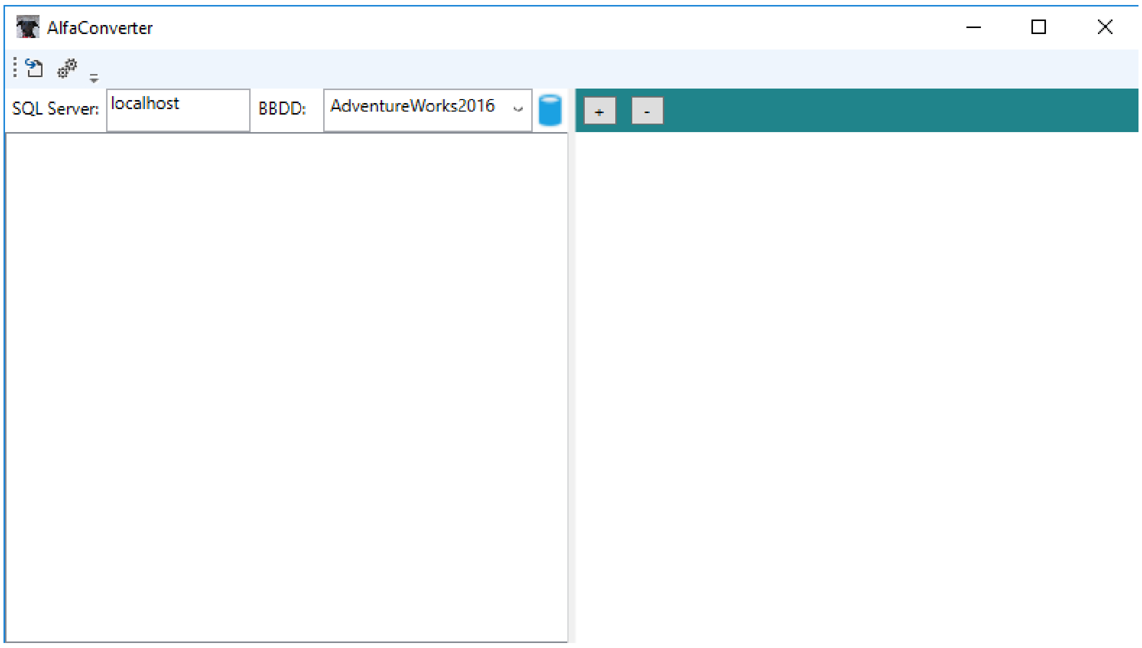
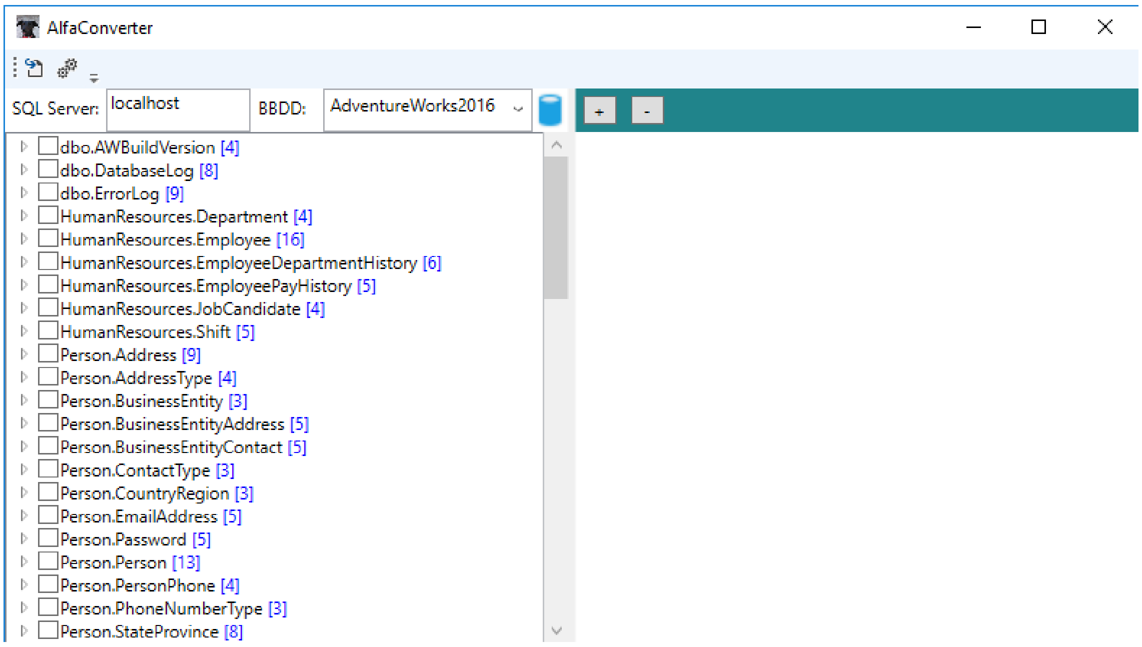


Figure 2: Initial opening of the application

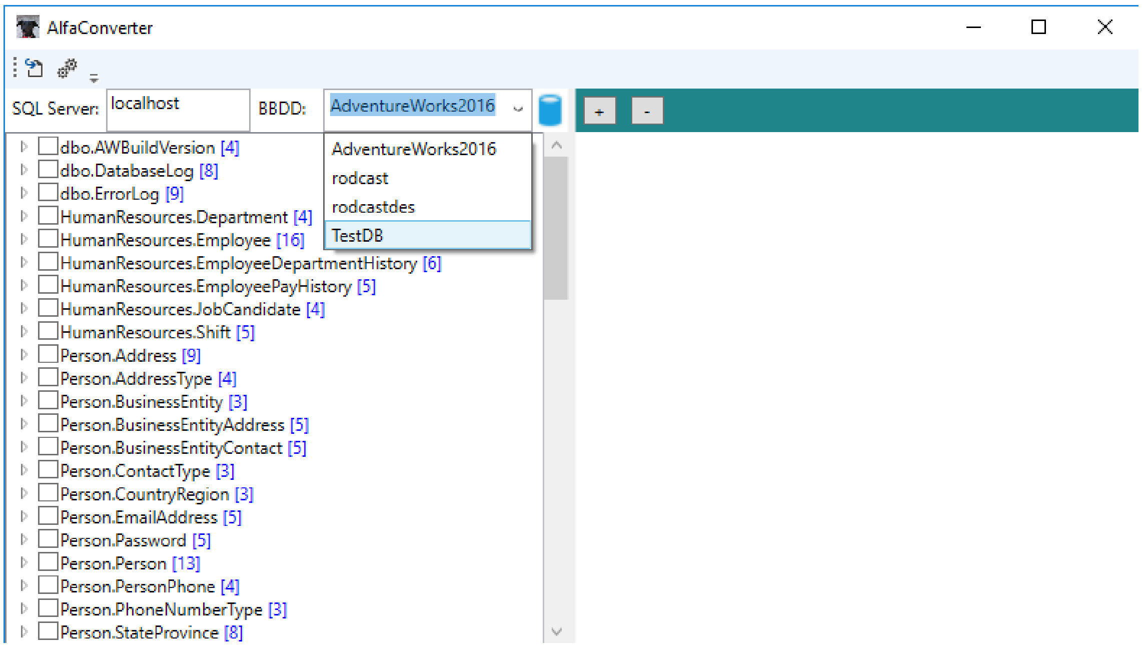
Because in the connection process you perform an analysis of the database in search of the tables and their columns, it may take a while to finish displaying the result in the left panel. This result can be seen below:



dbo.AWBuildVersion [4]
dbo.DatabaseLog [8]
dbo.ErrorLog [9]
HumanResources.Department [4]
HumanResources.Employee [16]
HumanResources.EmployeeDepartmentHistory [6]
HumanResources.EmployeePayHistory [5]
HumanResources.JobCandidate [4]
HumanResources.Shift [5]
Person.Address [9]
Person.AddressType [4]
Person.BusinessEntity [3]
Person.BusinessEntityAddress [5]
Person.BusinessEntityContact [5]
Person.ContactType [3]
Person.CountryRegion [3]
Person.EmailAddress [5]
Person.Password [5]
Person.Person [13]
Person.PersonPhone [4]
Person.PhoneNumberType [3]
Person.StateProvince [8]

Figure 3: Initial load of the tables of the configured database

In this state, the user can modify the name of the server, in which case the drop-down list of databases is updated with the new listing. The user can also select another database if desired:


AlfaConverter

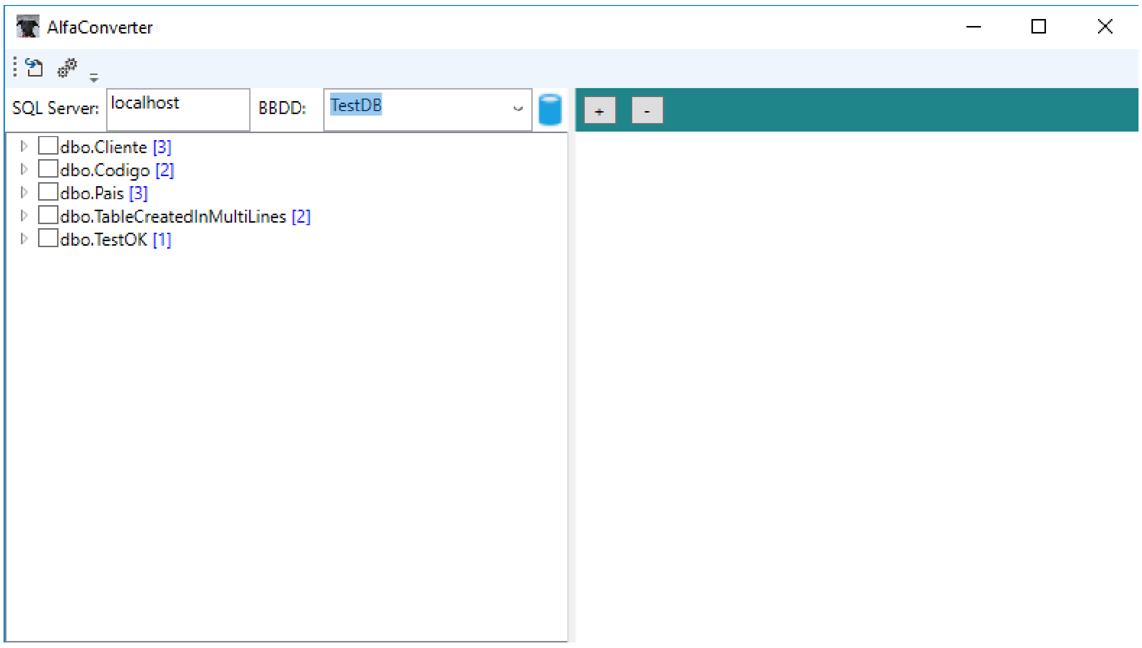
SQL Server: localhost BBDD: AdventureWorks2016

AdventureWorks2016
rodcast
rodcastdes
TestDB

dbo.AWBuildVersion [4]
dbo.DatabaseLog [8]
dbo.ErrorLog [9]
HumanResources.Department [4]
HumanResources.Employee [16]
HumanResources.EmployeeDepartmentHistory [6]
HumanResources.EmployeePayHistory [5]
HumanResources.JobCandidate [4]
HumanResources.Shift [5]
Person.Address [9]
Person.AddressType [4]
Person.BusinessEntity [3]
Person.BusinessEntityAddress [5]
Person.BusinessEntityContact [5]
Person.ContactType [3]
Person.CountryRegion [3]
Person.EmailAddress [5]
Person.Password [5]
Person.Person [13]
Person.PersonPhone [4]
Person.PhoneNumberType [3]
Person.StateProvince [8]

Figure 4: Displaying the BD list

When selecting a different database, the list of tables in the left pane is updated as can be seen below:

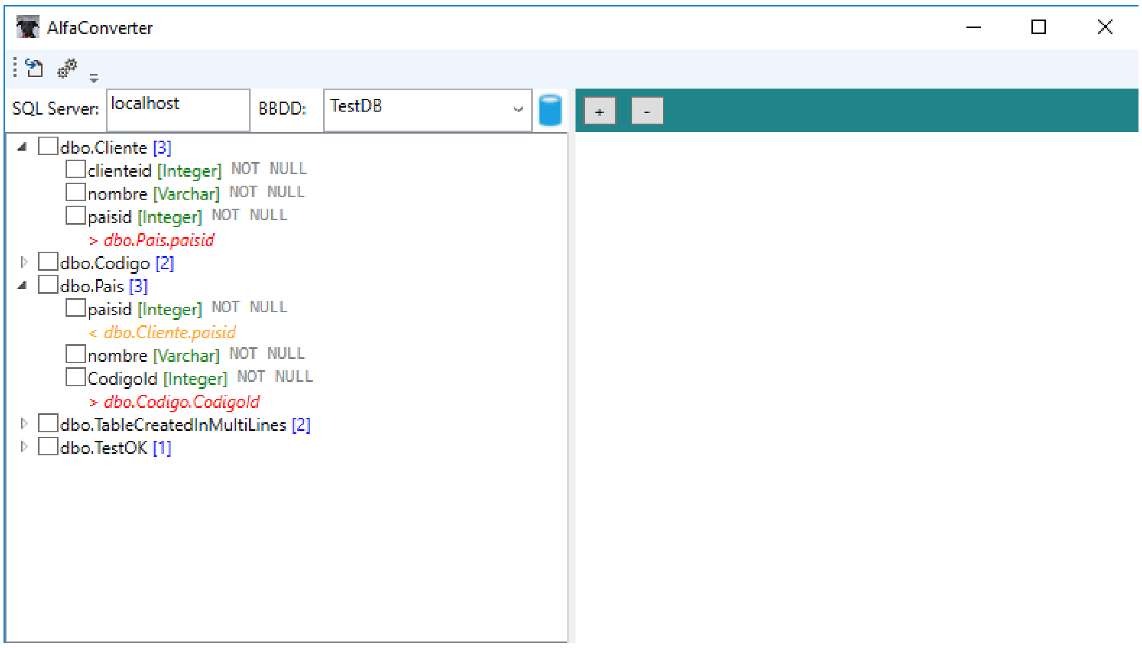

AlfaConverter

SQL Server: localhost BBDD: TestDB

dbo.Cliente [3]
dbo.Codigo [2]
dbo.Pais [3]
dbo.TableCreatedInMultiLines [2]
dbo.TestOK [1]

Figure 5:Selection of another database

The left panel allows an agile exploration of the database through its tables. Display a table shows your list of columns:



```
SQL Server: localhost   BBDD: TestDB

◢ ☐dbo.Cliente [3]
    ☐clienteid [Integer] NOT NULL
    ☐nombre [Varchar] NOT NULL
    ☐paisid [Integer] NOT NULL
      > dbo.Pais.paisid
▷ ☐dbo.Codigo [2]
◢ ☐dbo.Pais [3]
    ☐paisid [Integer] NOT NULL
      < dbo.Cliente.paisid
    ☐nombre [Varchar] NOT NULL
    ☐CodigoId [Integer] NOT NULL
      > dbo.Codigo.CodigoId
▷ ☐dbo.TableCreatedInMultiLines [2]
▷ ☐dbo.TestOK [1]
```

Figure 6:Showing list of columns in tables

In the previous figure it can be seen that in addition to the list of columns, additional information is also shown differentiated by a color code:

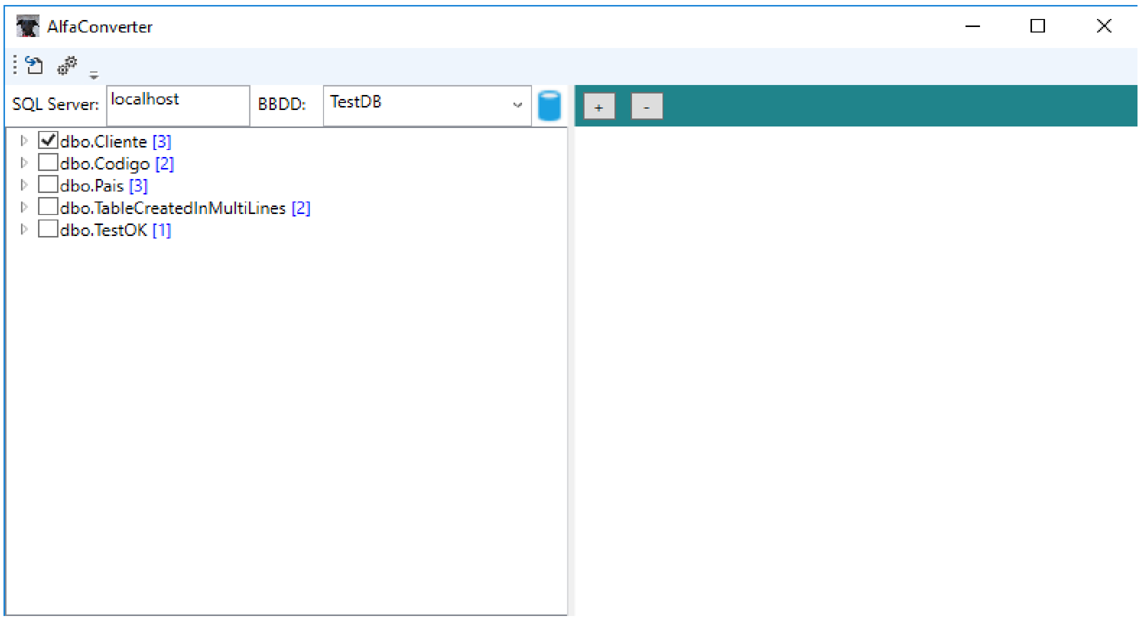
- Green: Column type.
- Grey: Indicates if the field can be NULL.
- Red: Indicate to which table a foreign key points.
- Orange: Indicates the list of tables that point to the current table field.

SQL to JSON conversion

Being easy to obtain important information from the Country table, this table will be processed to JSON detailing the fields it has:

- It has three fields: pais, nombre and CodigoId. You can also quickly see their types and if they can be NUL.
- The field Pais.CodigoId points to the Codigo.CodigoId.
- The field Pais.paisid is referred to by the Cliente.paisid.

Next, the user can select one or more tables:

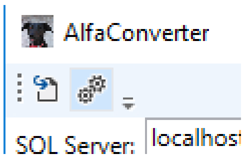

AlfaConverter

SQL Server: localhost BBDD: TestDB

- ☑ dbo.Cliente [3]
- ☐ dbo.Codigo [2]
- ☐ dbo.Pais [3]
- ☐ dbo.TableCreatedInMultiLines [2]
- ☐ dbo.TestOK [1]

Figure 7:Selecting an SQL table

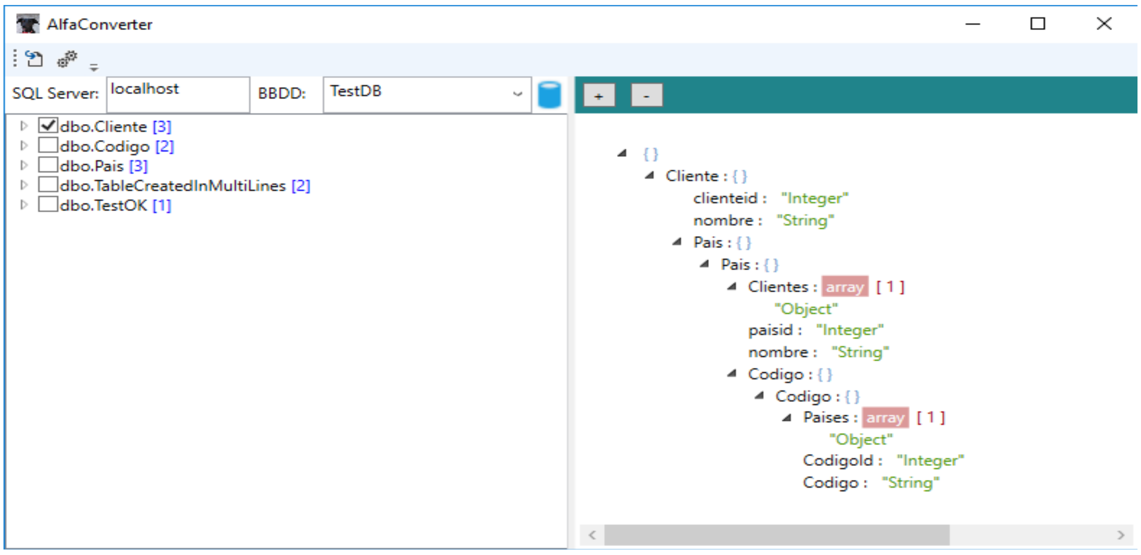
By clicking on the process icon (Gear):


AlfaConverter

SQL Server: localhost

Figure 8:"Process" button in the AlfaConverter tool

The application performs an analysis of the selected table and generates an equivalent JSON in the right panel:


AlfaConverter

SQL Server: localhost BBDD: TestDB

- ☑ dbo.Cliente [3]
- ☐ dbo.Codigo [2]
- ☐ dbo.Pais [3]
- ☐ dbo.TableCreatedInMultiLines [2]
- ☐ dbo.TestOK [1]

```
{ }
  Cliente : { }
    clienteid :  "Integer"
    nombre :  "String"
    Pais : { }
      Pais : { }
        Clientes : array [ 1 ]
          "Object"
        paisid :  "Integer"
        nombre :  "String"
        Codigo : { }
          Codigo : { }
            Paises : array [ 1 ]
              "Object"
            CodigoId :  "Integer"
            Codigo :  "String"
```

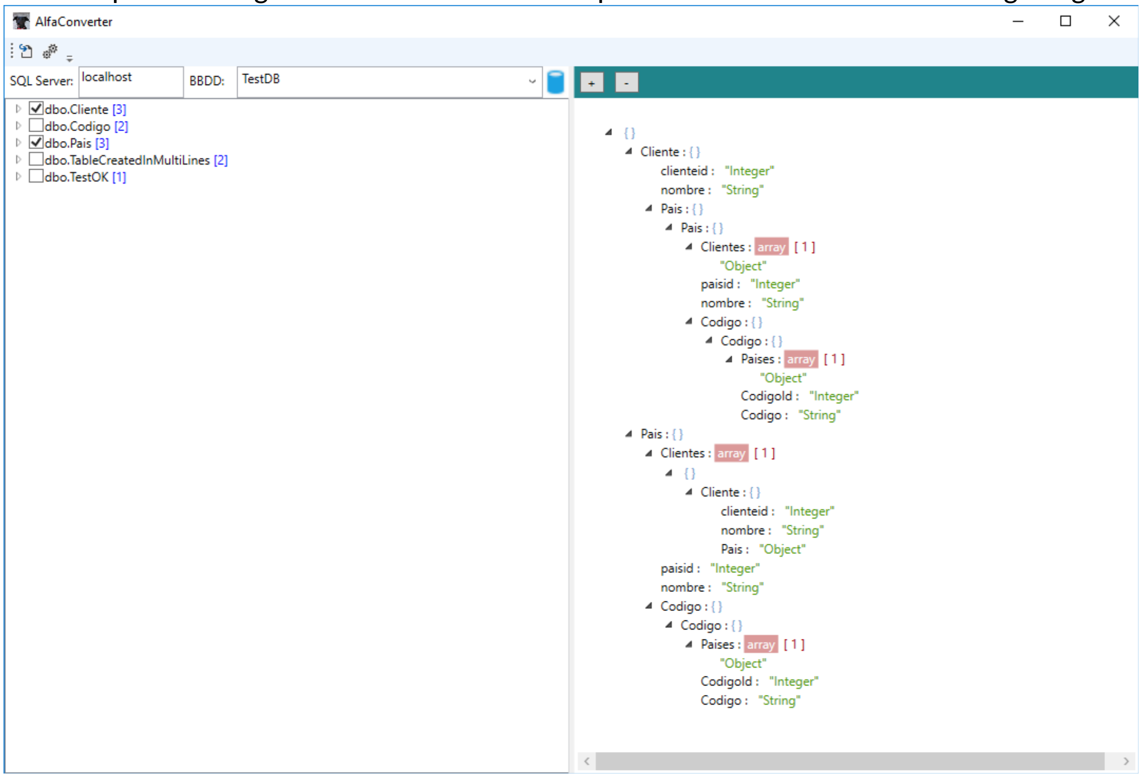
Figure 9:Conversion of an SQL table to JSON

As can be seen in Figure 8, the JSON that has been generated explains the SQL relationships originating in the Client table:

- As Cliente.paisid is a foreign key to the Pais table, in the JSON structure it translates into a field called “Pais” which in turn contains fields according to its table.
- As Pais.CodigoId is a foreign key to the Codigo table, in the JSON structure it translates into a field called "Codigo" within the "Pais" field which in turn contains fields according to its table.
- As Pais.paisid is referred to by the Cliente table, a “Cliente” field is created (note the automation of the plural), which is of the array type. As the "Cliente" object has already been defined at a higher level, to avoid infinite loops, this branch of the JSON structure is stopped here.
- The same thing happens with the “Pais” field that is autogenerated in the plural within the “Codigo” field. As the “Pais” structure has been generated at higher levels, the definition of the branch is also stopped here to avoid the infinite loop.
- SQL types are converted to NoSQL types:
    - Integer translates to Integer.
    - Varchar translates to String.

Conversion of several SQL tables to JSON

It is also possible to generate a JSON from multiple tables as seen in the following image:



SQL Server: localhost
BBDD: TestDB

- [✓] dbo.Cliente [3]
- [ ] dbo.Codigo [2]
- [✓] dbo.Pais [3]
- [ ] dbo.TableCreatedInMultiLines [2]
- [ ] dbo.TestOK [1]

```
{ }
  Cliente : { }
      clienteid :  "Integer"
      nombre :  "String"
    Pais : { }
      Pais : { }
        Clientes : array [ 1 ]
            "Object"
          paisid :  "Integer"
          nombre :  "String"
        Codigo : { }
          Codigo : { }
            Paises : array [ 1 ]
                "Object"
              CodigoId :  "Integer"
              Codigo :  "String"
  Pais : { }
    Clientes : array [ 1 ]
      { }
        Cliente : { }
            clienteid :  "Integer"
            nombre :  "String"
            Pais :  "Object"
      paisid :  "Integer"
      nombre :  "String"
    Codigo : { }
      Codigo : { }
        Paises : array [ 1 ]
            "Object"
          CodigoId :  "Integer"
          Codigo :  "String"
```

Figure 10:Conversion of two SQL tables to JSON structure

In this case, the structure for the Pais table is generated following the relationships taking that table as a starting point. By jointly generating Cliente and Pais, it is easy to realize the action of infinite loop protection:

- The “Pais” field slows its definition in the “Clientes” field because it was already defined at a higher level.
- However, when the “Pais” structure is defined in parallel, the “Clientes” structure is defined.

Import SQL script

In addition to being able to access an existing database, the application has the possibility to import an SQL script with DDL statements using the import button as shown in the following image:

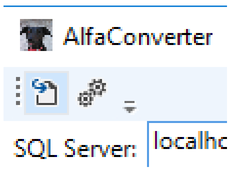
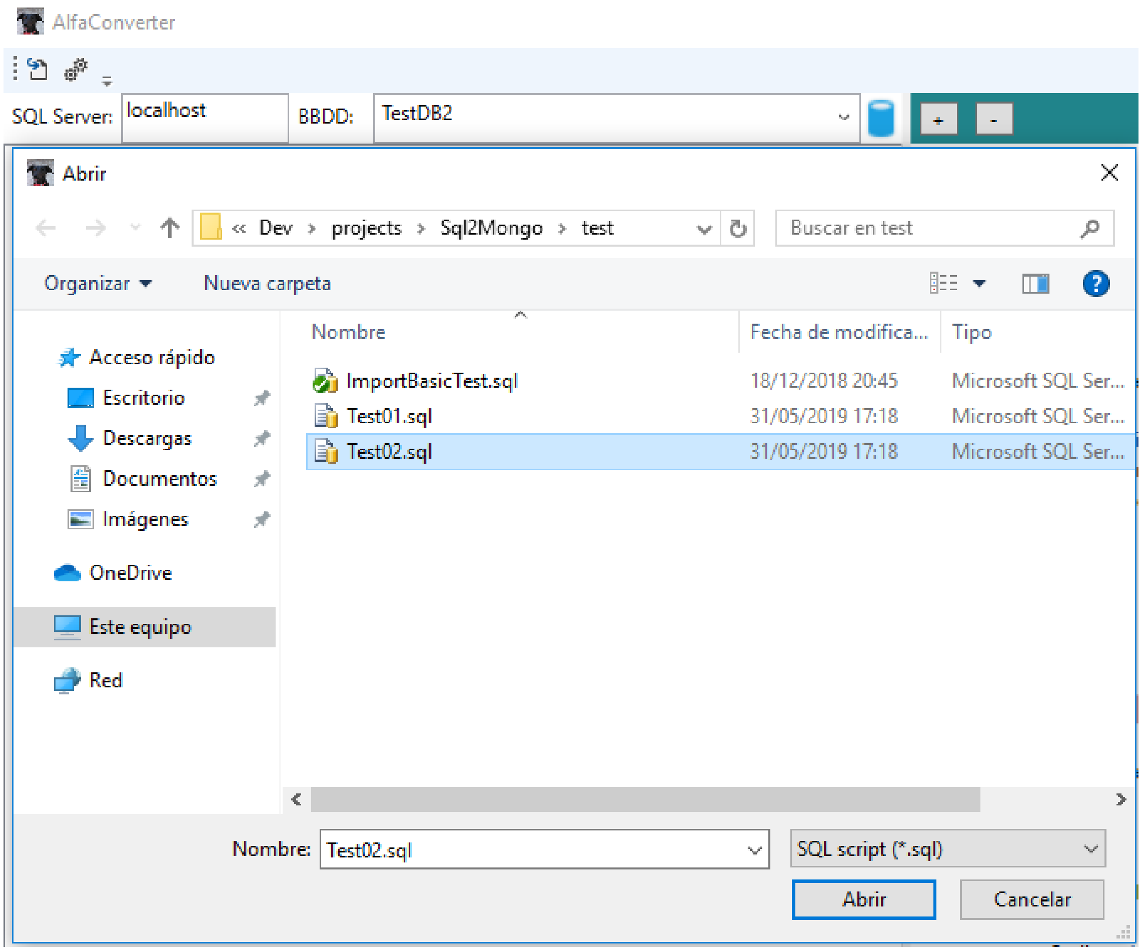


Figure 11:"Import" button of the AlfaConverter tool



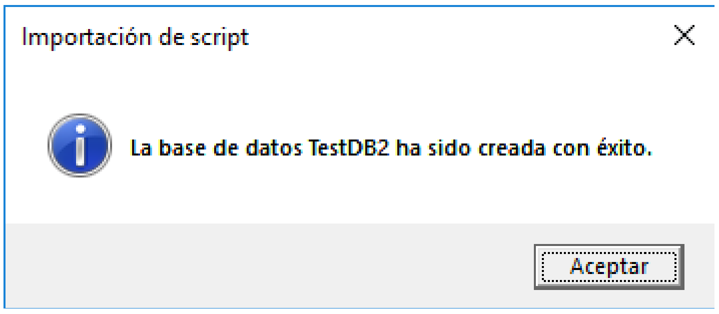
Nombre	Fecha de modifica...	Tipo
ImportBasicTest.sql	18/12/2018 20:45	Microsoft SQL Ser...
Test01.sql	31/05/2019 17:18	Microsoft SQL Ser...
Test02.sql	31/05/2019 17:18	Microsoft SQL Ser...

Nombre: Test02.sql

SQL script (*.sql)

Figure 12:Importing a SQL script into AlfaConverter

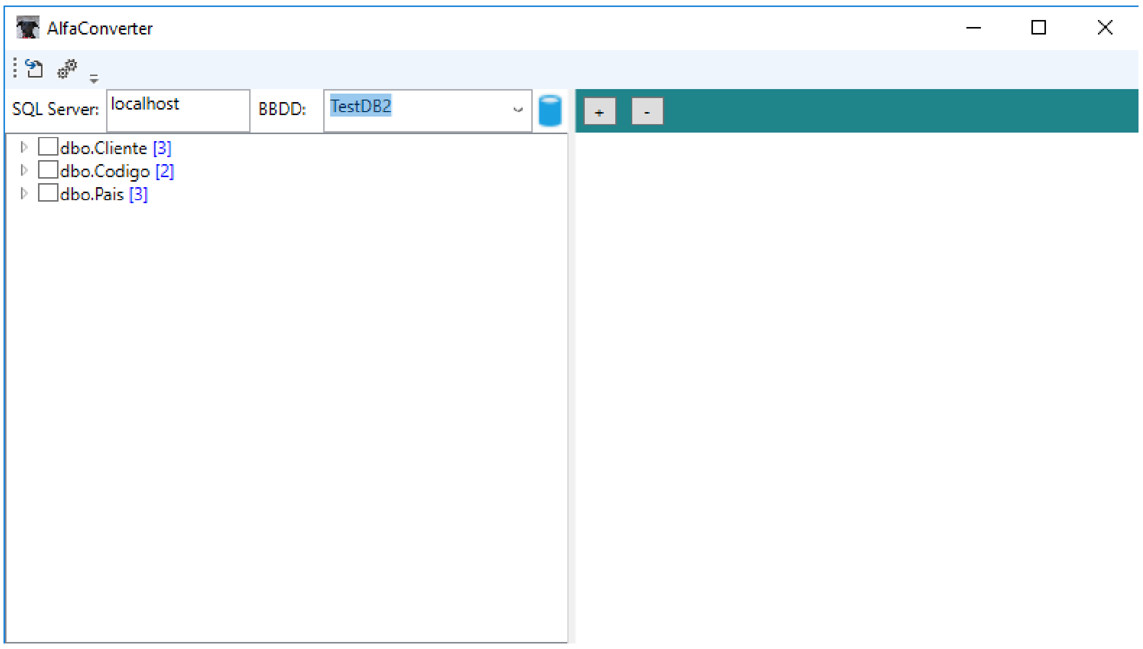
If the import was successful, the application will display a message similar to the one below as a notification to the user:


Importación de script

La base de datos TestDB2 ha sido creada con éxito.

Figure 13:Notification message by AlfaConverter tool

After which, it is possible to select the new database as one more:



SQL Server: localhost BBDD: TestDB2

dbo.Cliente [3]
dbo.Codigo [2]
dbo.Pais [3]

Figure 14:Viewing a new imported data database