**Scheme generator**

**Context**

Many of today's applications use a database to store the data that is needed in the applications. However, when the applications are in a development environment they need data to prove that the system is working properly, however, this involves registering data to populate the database and that these can be manipulated by the system. That said, most developers need to generate random data and make it as real as possible, the problem is that, if they want to have a database with thousands of records, making them by hand takes a lot of time.

According to DTM software, a data generator is: “A software product that produces data rows and scheme objects for testing purposes: test database population, performance analyzing, QA testing or loading tests fulfillment. The generator was designed to provide developers and quality assurance engineers with high quality and realistic tests arrays. It automatically creates data values and optional schema objects (tables, views, procedures, triggers, etc.)” (DTM, 2018)

That is why data generators are used to generate different data and that these can be stored in a database, there are currently many of these generators on the internet, but the problem with these generators is that they only generate data in one language (Spanish, English, etc.) or the amount of data is limited. However, generators are a great tool that optimizes many tasks for developers.

**Solution**

The solution to this problem was to create an application in Scheme that could generate different types of data for different languages, giving users the possibility to register their own catalogs so that other developers can make use of these types of data.

The application creates a CSV file in which the name of the columns and the type of data that each of the rows fill are recorded. The number of lines is defined by the user, as well as the language and the name of the output file.

The following diagram shows how the system works.

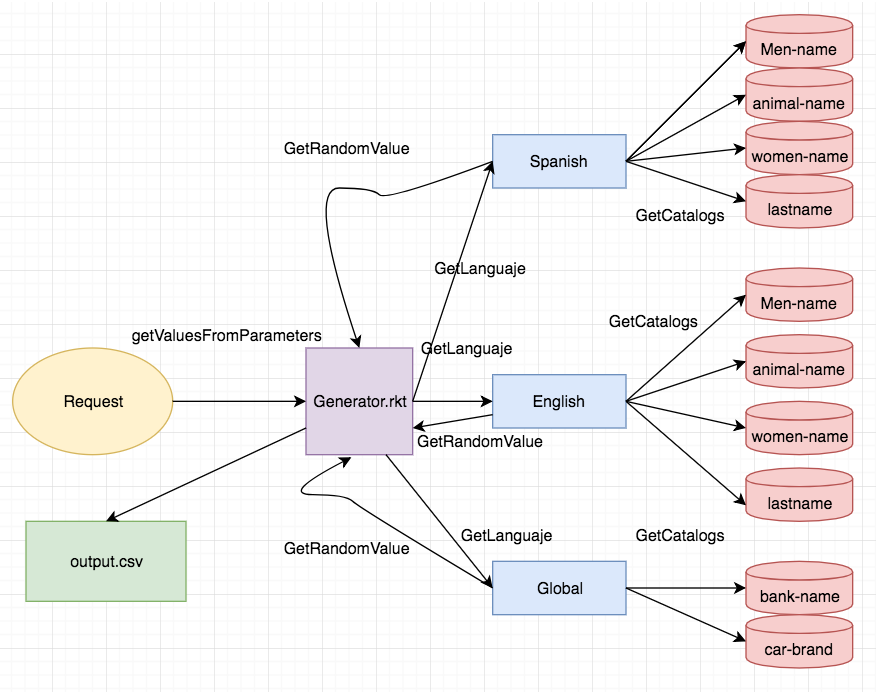


Image 1.1 – Flow diagram for scheme generator.

The catalogs are in folders named with a simple abbreviation of the languages (es –Spanish, en - English). But also exists a folder called global which contains catalogs that are the same for Spanish and English data, because are data that you can’t translate because they are names of business or brands.

For this program are used three files:

Generator.rkt: This file is the main file which generates the csv file.

Get-values.rkt: This file is used to get the values given in parameters from input.

Get-catalogs.rkt: This file is in charge to load the catalogs from the differents catalogs.

**Results**

The scheme generator shows and example of input, the available languages and datatypes for each language.

The program also validates if the input is valid, if you put valid languages and the filename for csv file.

Note: The program doesn’t validate yet, if the datatypes and if the parenthesis are correctly used. This a technical debt that will be fixed then. So, it is expected that the user understands the basic instructions that the program shows.

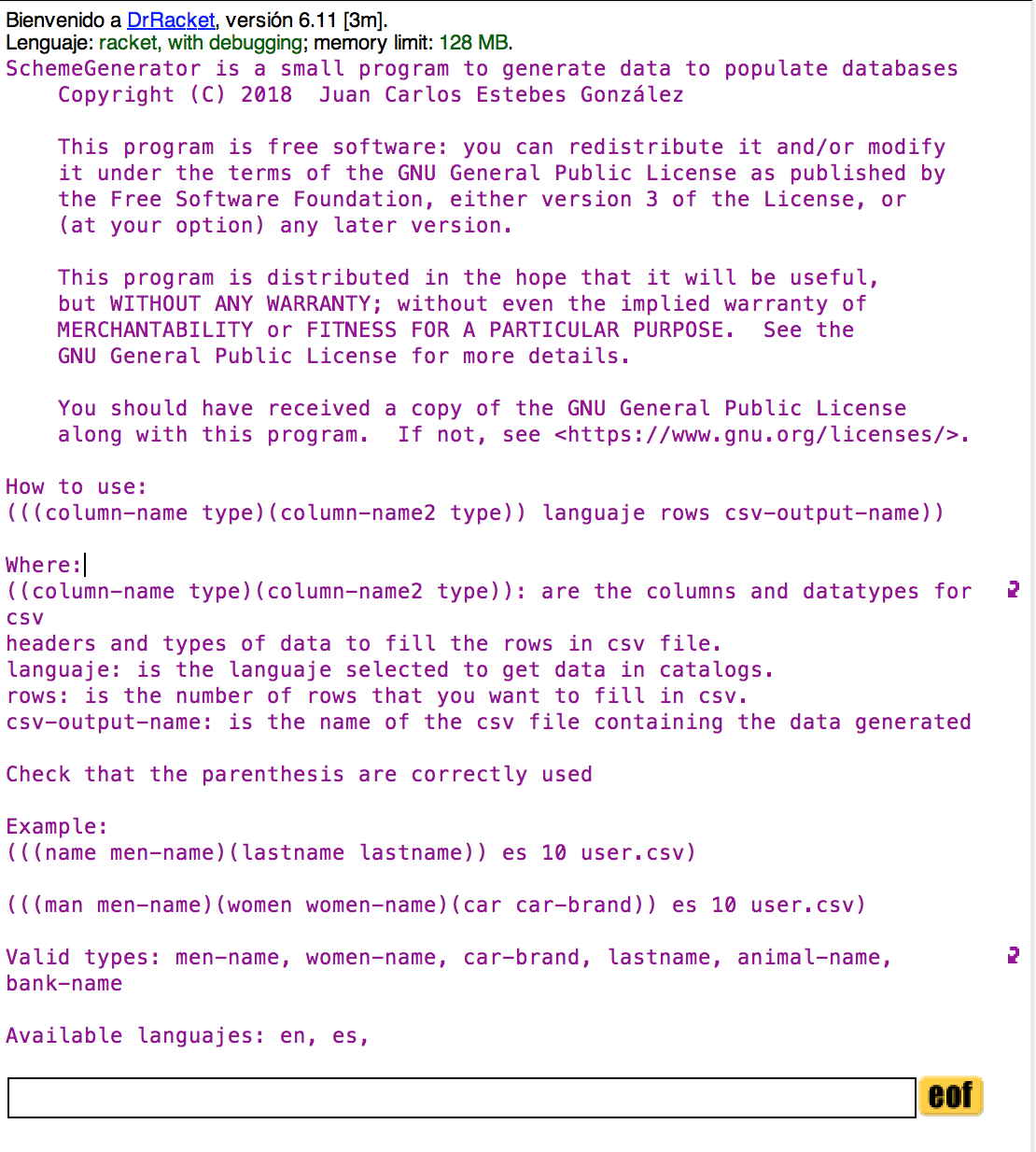


Image 1.2 – User interface

One example could be generating a user.csv file contain 10 rows with Spanish data, and the column names are user and lastname, so the type of data for these two columns will be man-name and lastname respectively:

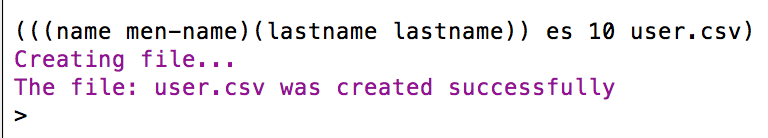


Image 1.3 – Example of valid input.

The previous example show how the program works with valid values, but there are other scenarios in with the program evaluates the right output. For example, if you want to create a csv file with the same name of an existing one in the same directory or if you want to generate data selecting an inexistent language.

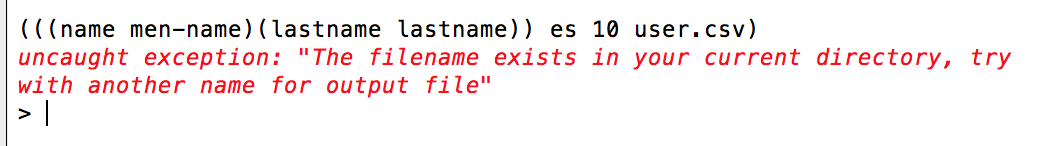


Image 1.4 – Message for existing file with the same name.

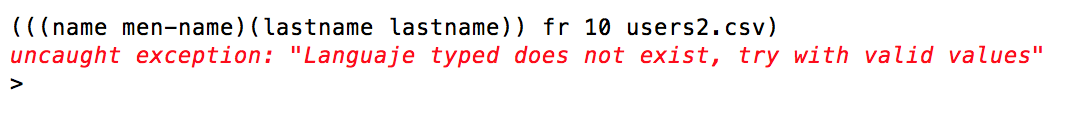


Image 1.5 – Message for inexistent language.

**Tests Cases**

For tests-cases, I create different outputs files with different inputs.

Test cases are in:

<https://github.com/Estebes10/SchemeGenerator/tree/develop/test-cases>

**Conclusions**:

I have developed a small program that could be useful for developers that are working on projects that use databases and they want to populate their databases with real data. On the other hand, this program allows to the users the possibility of adding their own catalogs and other user can use it later. This program allows me to understand more in detail how the functional paradigm programming works, because more of the functions that I have implemented in the generator was using recursion, that is the important loop control and the order of execution in this paradigm is of low importance compared to imperative programing paradigm. Finally, this project is not finished because is something that I want to improve to use in future projects, adding more functionality and catalogs, but I enjoy developing this system and I hope that other users use this program.

**Setup instructions**

Open DrRacket IDE (if you don’t have it, download from it here: <https://download/racketlang.org)>

Open and execute the program “generator.rkt”

The program will display an example of how to use the program, what are the available languages and the available types for each language.

To use the program, you must type something like this:

*(((column\_name1 datatype) (column\_name2 datatype2)) es 10 salida.csv)*

where:

* column\_name1 and column\_name2 are the columns of the table that you want to populate and they will be the headers of the csv file.
* Datatype and datatype2 are the type of data that you want to generate to populate column1 and column2.
* es: is the language selected to get data from catalogs
* 10: is the number of rows that will be registered in the csv file
* salida.csv: is the filename for output csv file.

Example:

*(((Username men-name) (last-name lastname)) es 50 users.csv)*

That input will generate an users.csv file containing 50 rows of users with name and last-name in Spanish data.

**References**:

DTM Soft. (2018). DTM Data Generator. Test Data Generator Overview. Retrieved from: <http://www.sqledit.com/dg/>