This document presents the results from the tests made to the developed apps.

The goal of this project was to accomplish a Go application that was able to collect CPU and RAM usage each second and store it in a local database, create a simulator of an external device generating (each second) a sample composed by four variables and store them in a local database, and, finally, to provide an interface between the user and the samples data through the console allowing multiple processing from that data.

To allow the coexistence between the database continuous fill, and, at the same time, its interaction, it was created two separated apps: DataCollector.exe and Interface.exe.

Therefore, the tests presented in this document are divided by:

1- Database periodic actualization
2- Interaction between the Interface and the database

1. Database periodic actualization

In the Figure 1 and 2, it is possible to see, for different time moments, the update of table "OScollector" (CPU and RAM usage) as well as in Figure 3 and 4 for table "ExtDevice" for the same time difference, using DB Browse program.

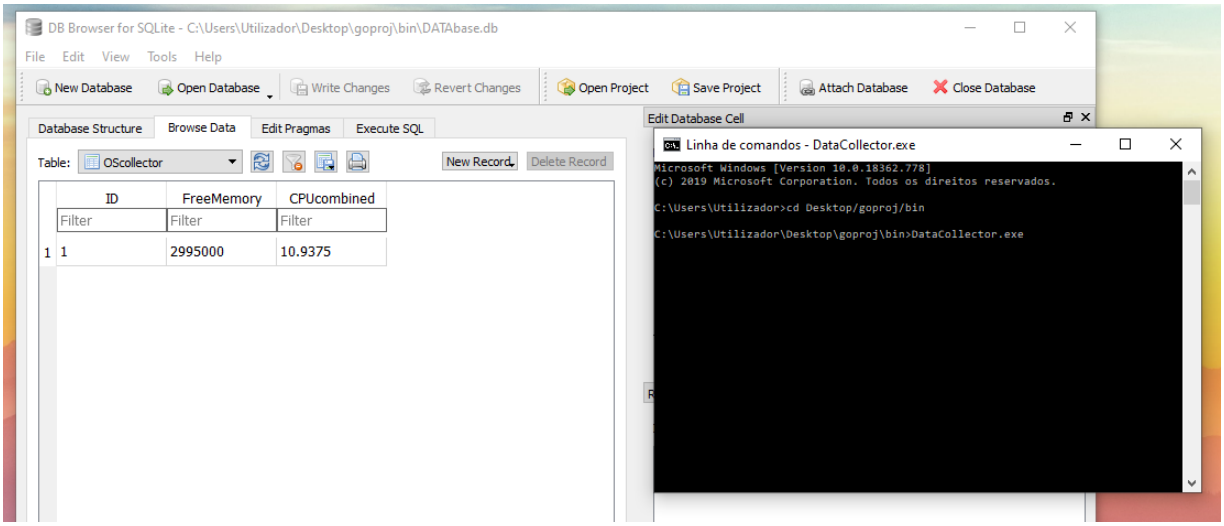

Figure 1- left: database entries, right: program executing

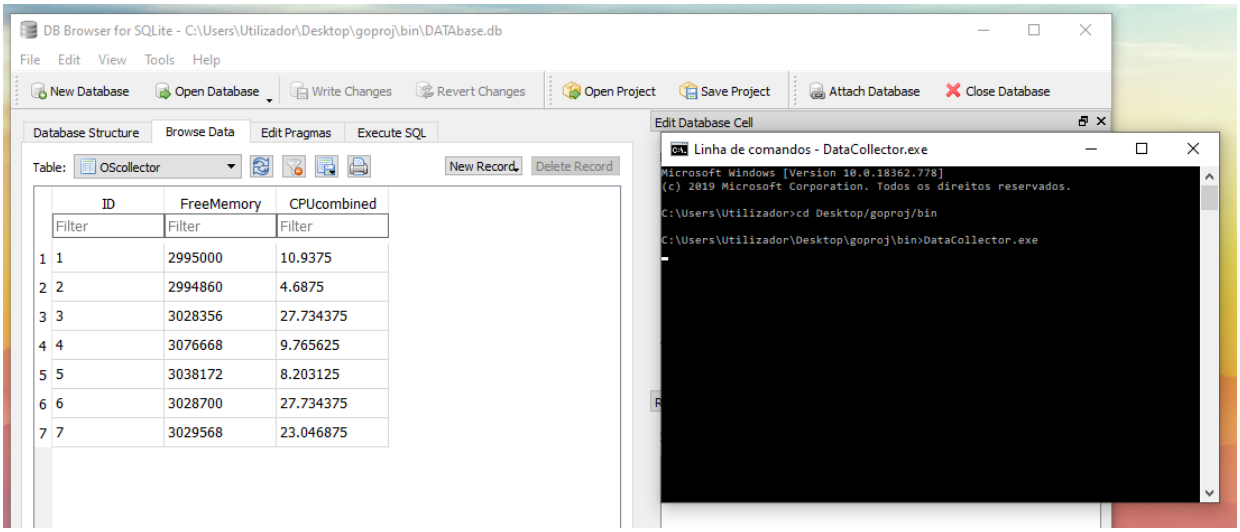

Figure 4 - left: database new entries, right: program executing

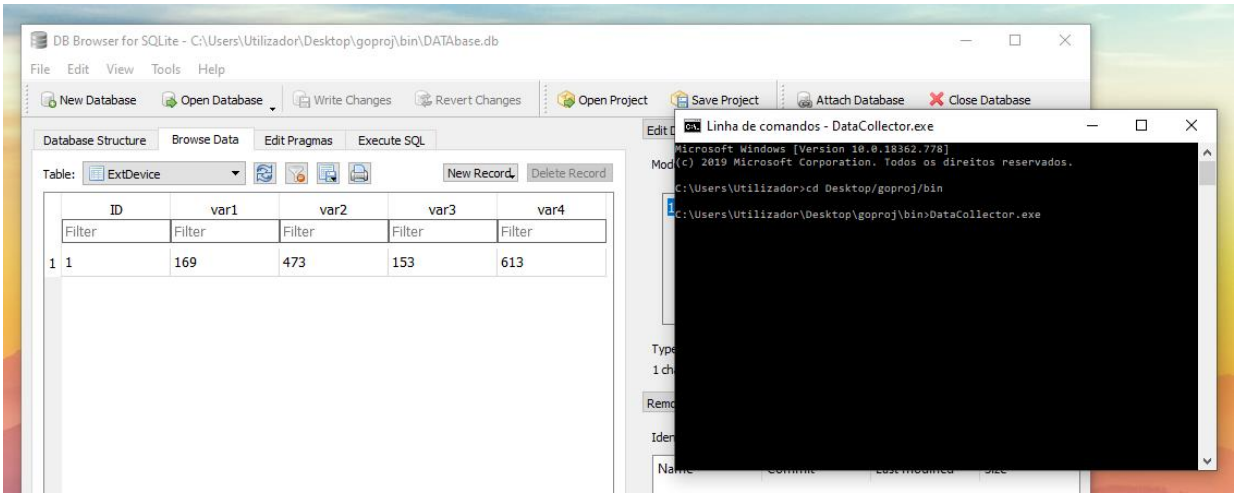

Figure 3 - left: database entries, right: program executing

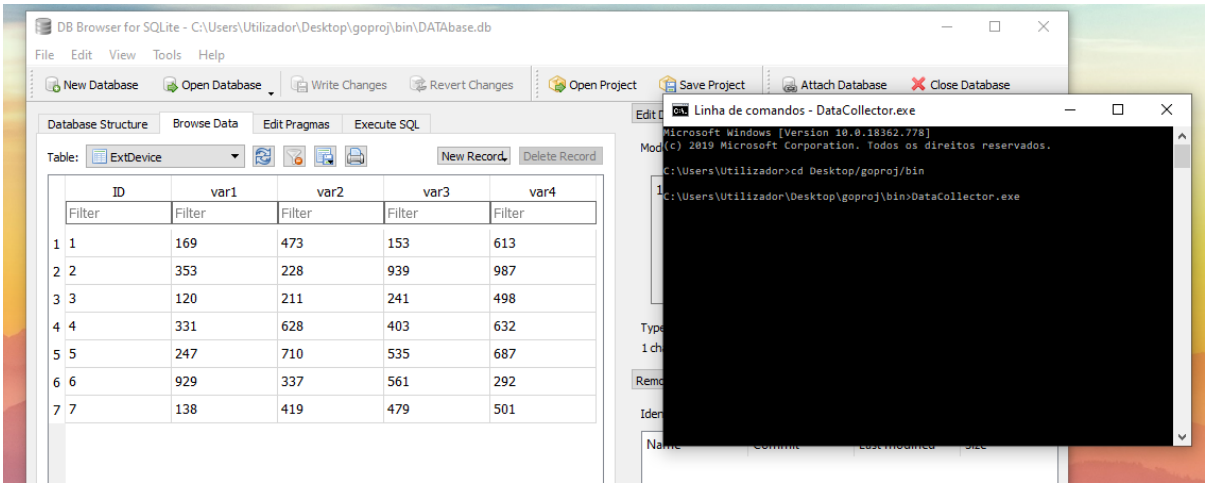

Figure 2 - left: database new entries, right: program executing

2. Interaction between the Interface and the database

In Figure 5 it is possible to see the “DataCollector” app running in one terminal, while the Interface gets different results in two different time moments for a “metrics all” command, presenting the last 5 samples for all variables.

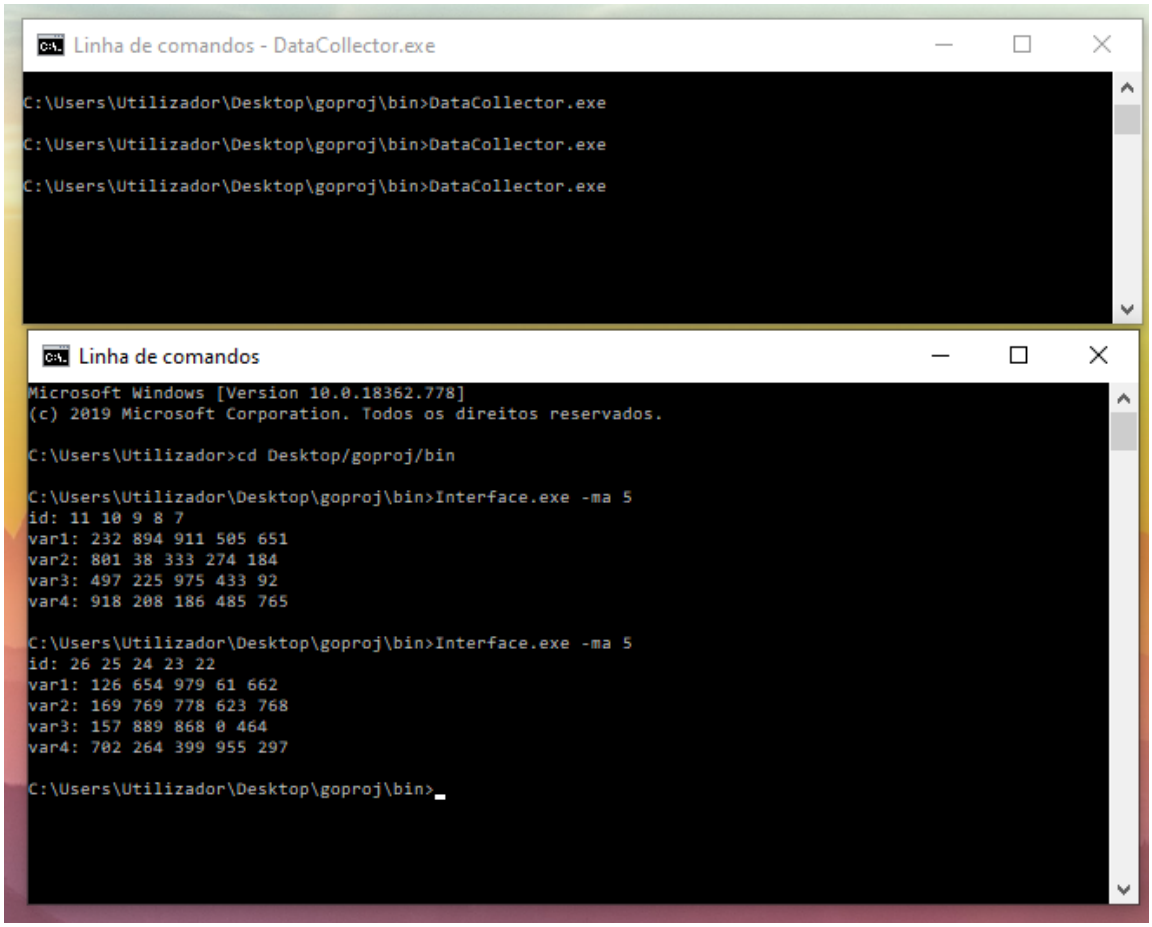

Figure 5 - Interface for a “metrics all” command

Next figure shows the test for a “metrics” command for two variables (var1 and var2) in three samples.

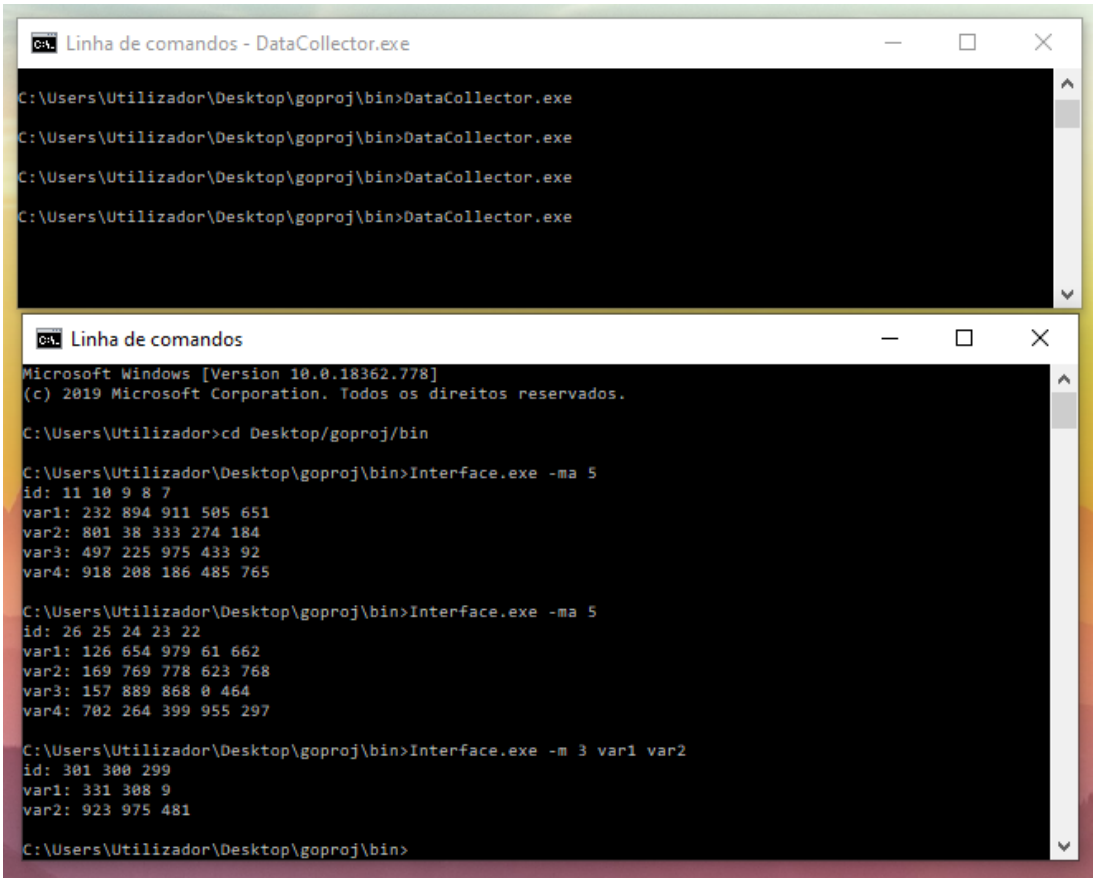

Figure 6 - Interface for a “metrics” command

Last, in Figure 7, it is presented the test to the “average” command of all the four variables

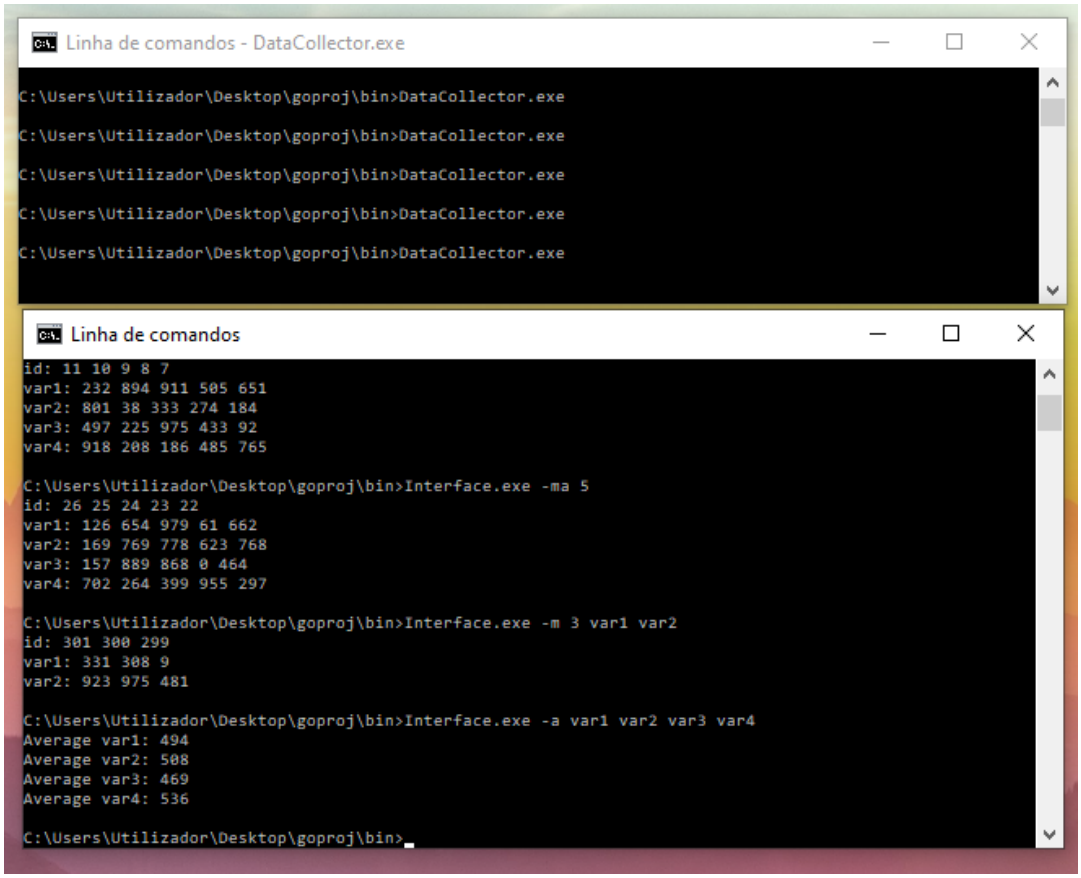

Figure 7 - Interface for an “average” command