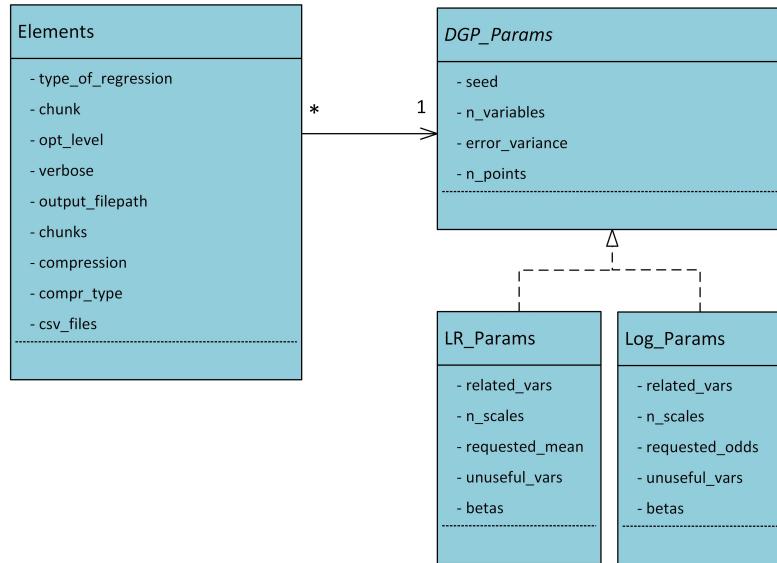


## 0.1 Access

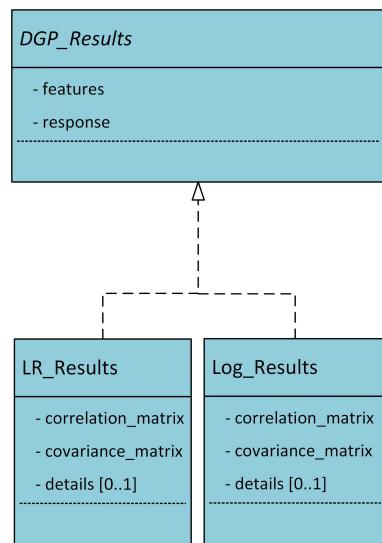
There are three different ways to access the software

- the launch of main without passing arguments, thus being automatically guided in entering the input elements from the terminal;
- the launch of main passing as argument the name of the .json file from which to read the input elements;
- the invocation of the functions of the `textit` Generator class, previously imported, without going through main.

## 0.2 Domain diagrams

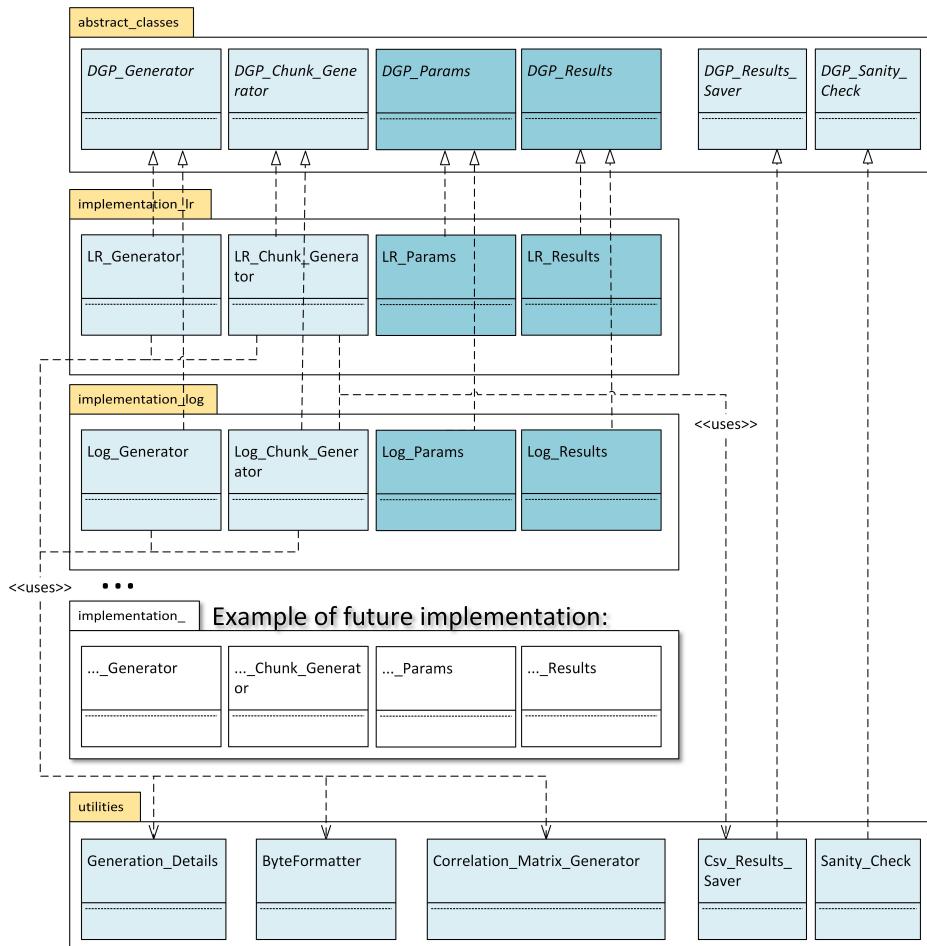


- To represent the different mathematical parameters in the different types of regression we have chosen to construct the abstract class (*DGP\_Params*) and have it implemented by the parameters of the generation varieties that we want to achieve.
- The relation between the *Elements* and *DGP\_Params* classes is “one to many” since each instance of the *Elements* class corresponds to one and only one instance of the *DGP\_Params* class, and since each instance of the *DGP\_Params* class can belong to an indefinite number of Element instances, even none.
- The abstract classes are distinguished from the others, in this diagram and the following ones, by the title in italics

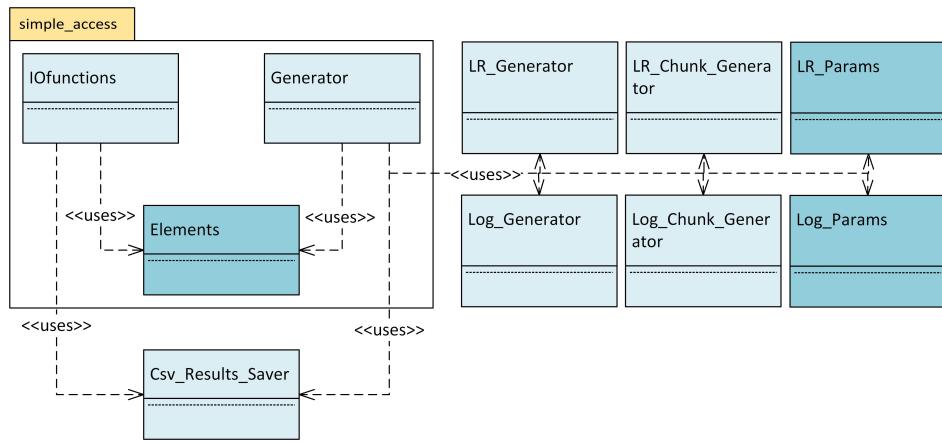


Even if for Linear and Logistic Regression the outputs produced by the generator will be the same (X table, Y series, correlation matrices, covariance matrices and details), it is still good to base the output domain structure on an abstract class, only with the aim of favoring future extensions of the model to types of generations that produce different outputs.

## 0.3 Packages

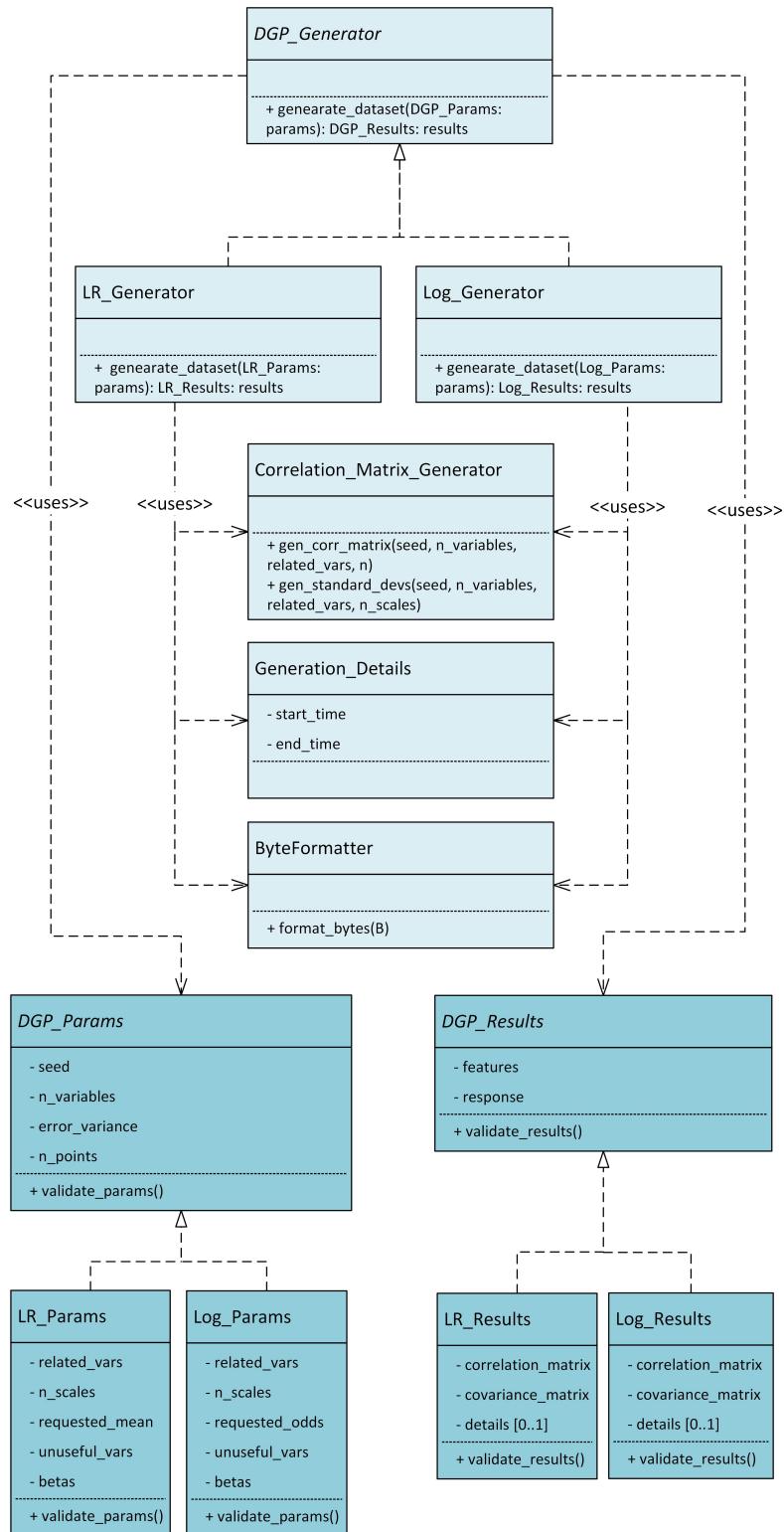


- All the classes that can be useful for the different types of generation will be contained in the `utilities` package. Among these can appear ordinary classes (for example `Correlation_Matrix_Generator`) or implementations of abstract classes (for example `Csv_Results_Saver` ).
- The domain classes, in this diagram and in all subsequent ones, are distinguished from the others because they are colored a darker blue.



The classes of the *simple\_access* package relate, directly or indirectly, to the entire system

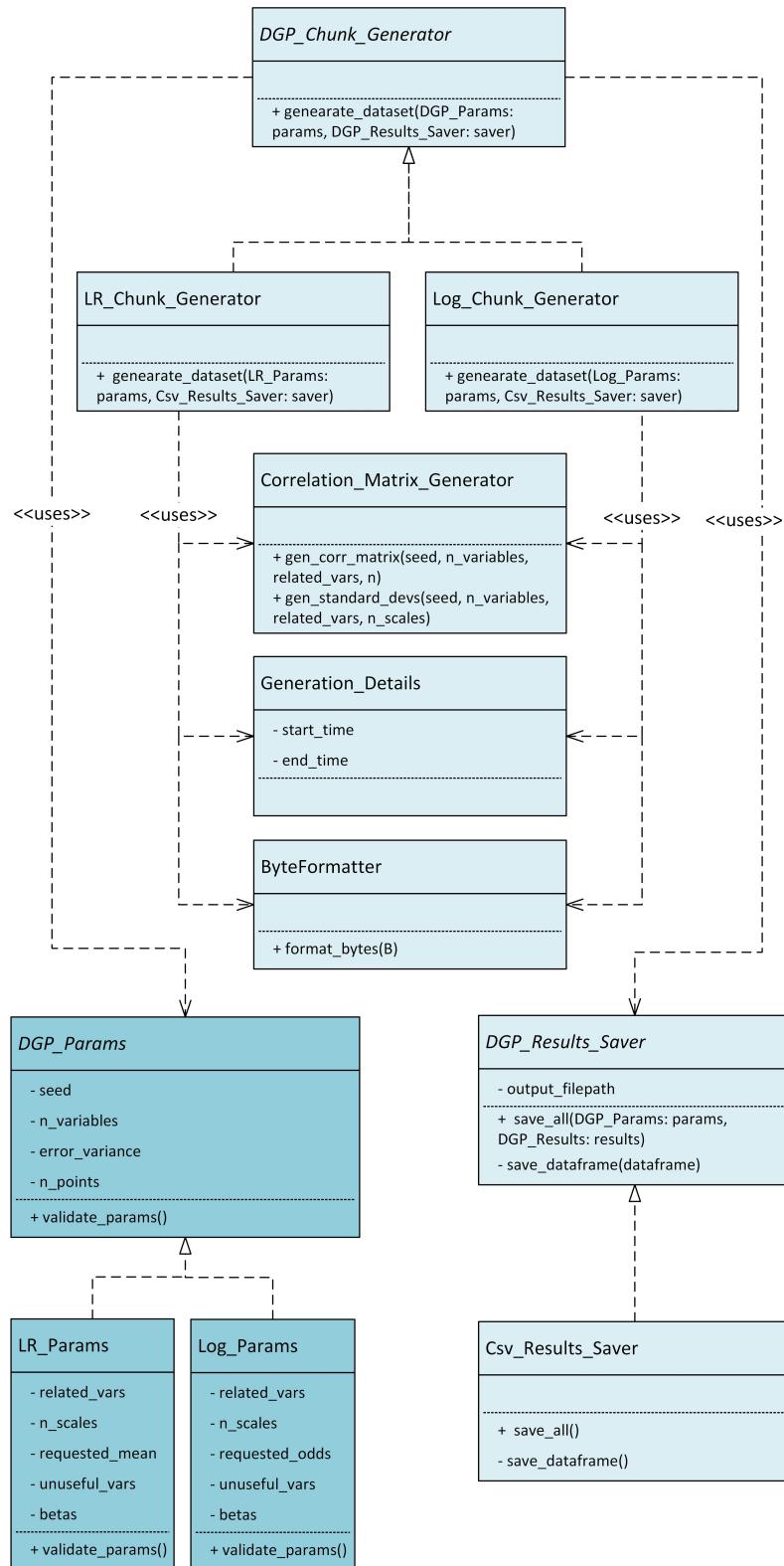
## 0.4 Classes



- The abstract class *DGP\_Generator* represents the core of the software and its implementations will contain all the logic of a specific generation typology.
- The data generations will be carried out on the basis of the user's choices and, above all, starting from an instance of *DGP\_Params* which, with its specific implementations, will provide the mathematical parameters
- The generation function will return to an instance of a *DGP\_Results* class.
- The two generations delegate certain features to the 3 classes *Correlation\_Matrix\_Generator*, *Generation\_Details* and *ByteFormatter*, however they are found in the *utilities* package, therefore they can also be used by the new types of generation that will develop in the future.
- *Correlation\_Matrix\_Generator* is capable of generating matrices that randomly mix correlated and uncorrelated variables.

## 0.4 Classes

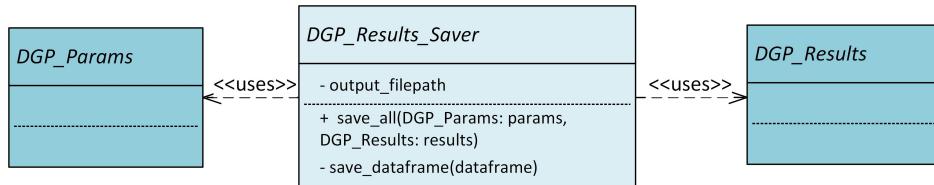
---



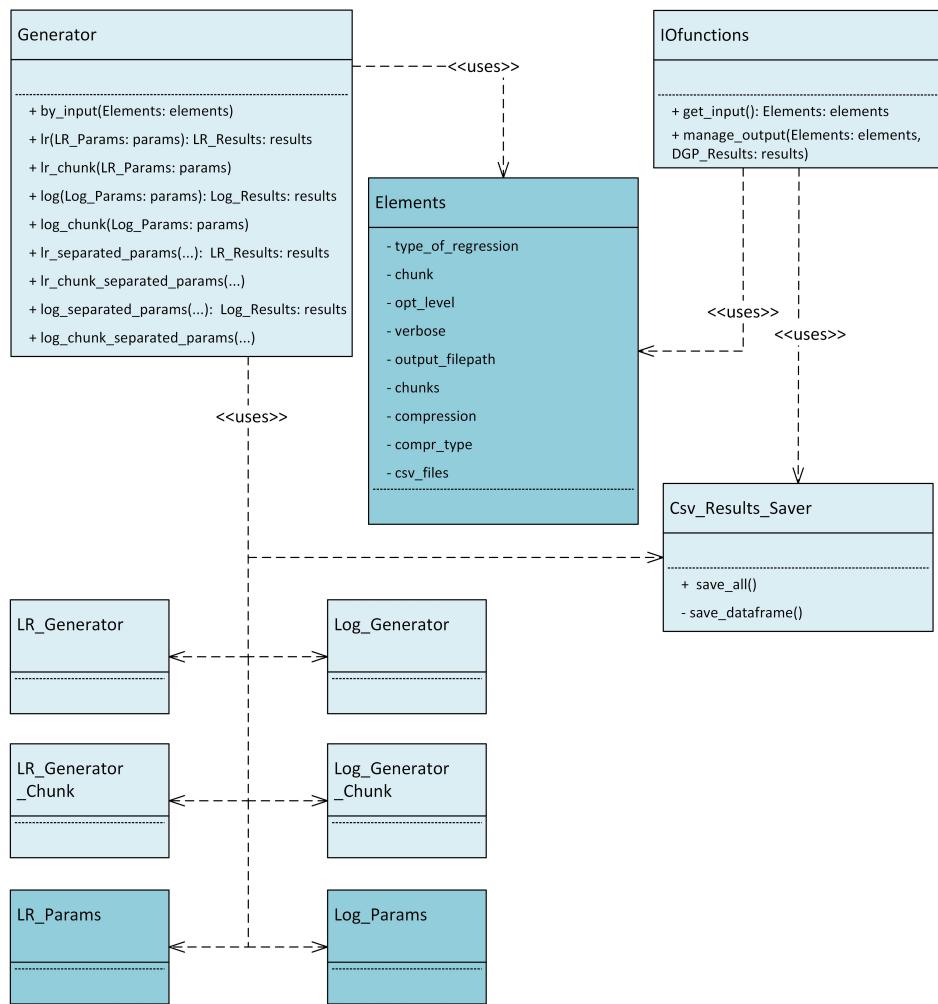
- The consideration of *DGP\_Generator* for the generation in portions and *DGP\_Chunk\_Generator*, this type of generation does not require the use of classes in which to save the results, but requires the 'use of *DGP\_Results\_Saver*.
- The only class implementing *DGP\_Results\_Saver* present at the moment is *Csv\_Results\_Saver*, however the very existence of the abstract class *Csv\_Results\_Saver* opens the door to the implementation of new methods of saving risultati.

## 0.4 Classes

---



- This diagram analyzes the classes involved in saving results.
- The *Csv\_Results\_Saver* class is related to the *DGP\_Params* class as it has among its functions the printing of beta parameters on .csv
- The *Csv\_Results\_Saver* class is related to the *DGP\_Results* class as it has among its functions the printing on .csv of all the attributes present in *DGP\_Results*. Obviously, this functionality is associated only with the standard generation, as the generation in portions does not produce instances of *DGP\_Results*



## *0.4 Classes*

---

- The *IOfuctions* class implements the functions of input from file and terminal and management of the generation results (presentation on terminal, saving on .txt file and possibly on .csv file)
- The *Generator* class provides numerous and simple generator access functions:
  - the *by\_input* function starts generation from the mathematical parameters and choices contained in an instance of *Elements*. Such an instance will typically be obtained from the *get\_input* function of *IOfuctions*. item the functions ending with *\_separated\_params* are the most intuitive of all, in fact they require in input the single mathematical parameters and the generation choices (which can be omitted, as there is default for each). item the remaining functions require as input a class *DGP\_Params* already instantiated and possibly the generation choices.

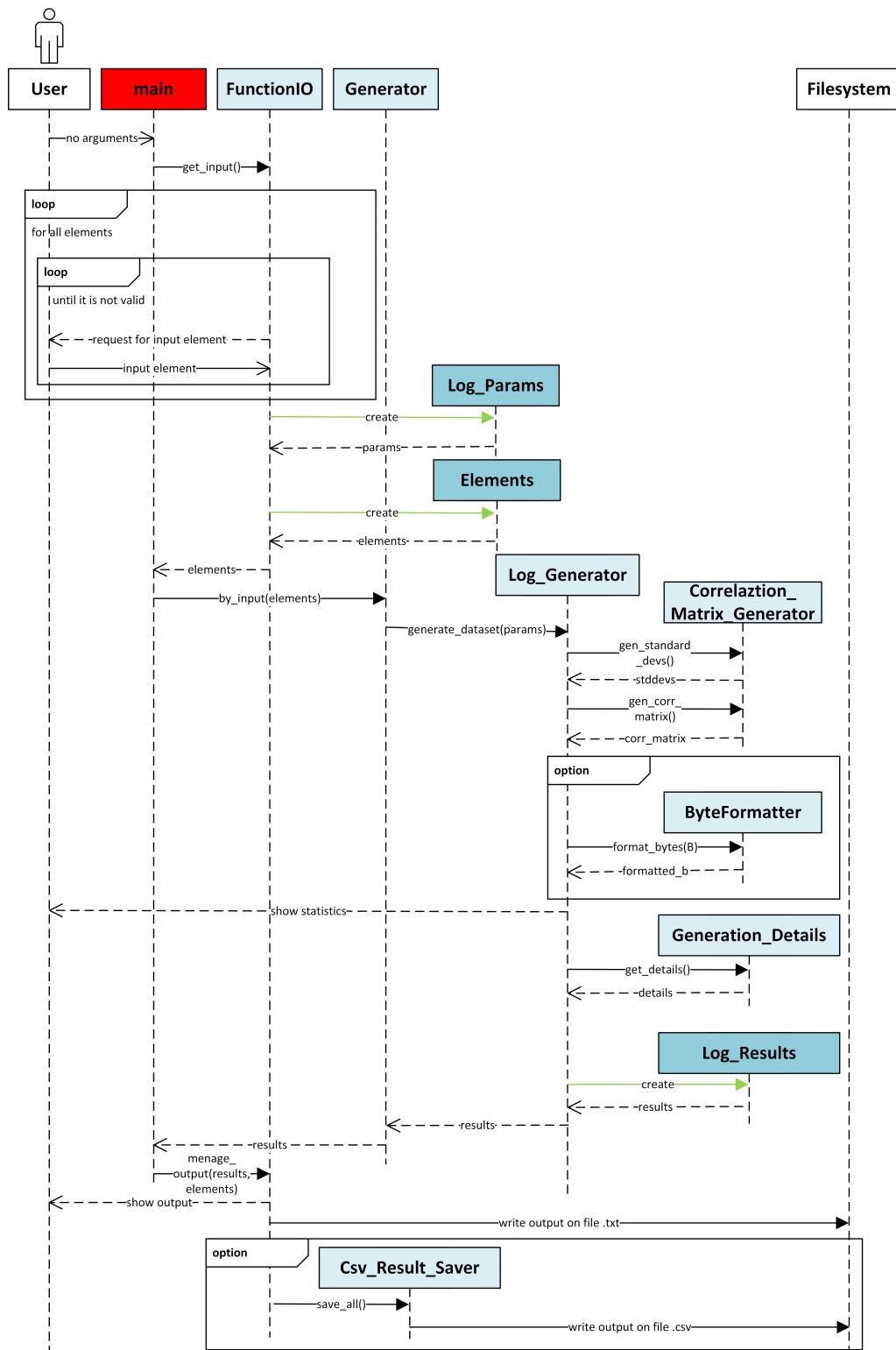
## 0.5 Sequence

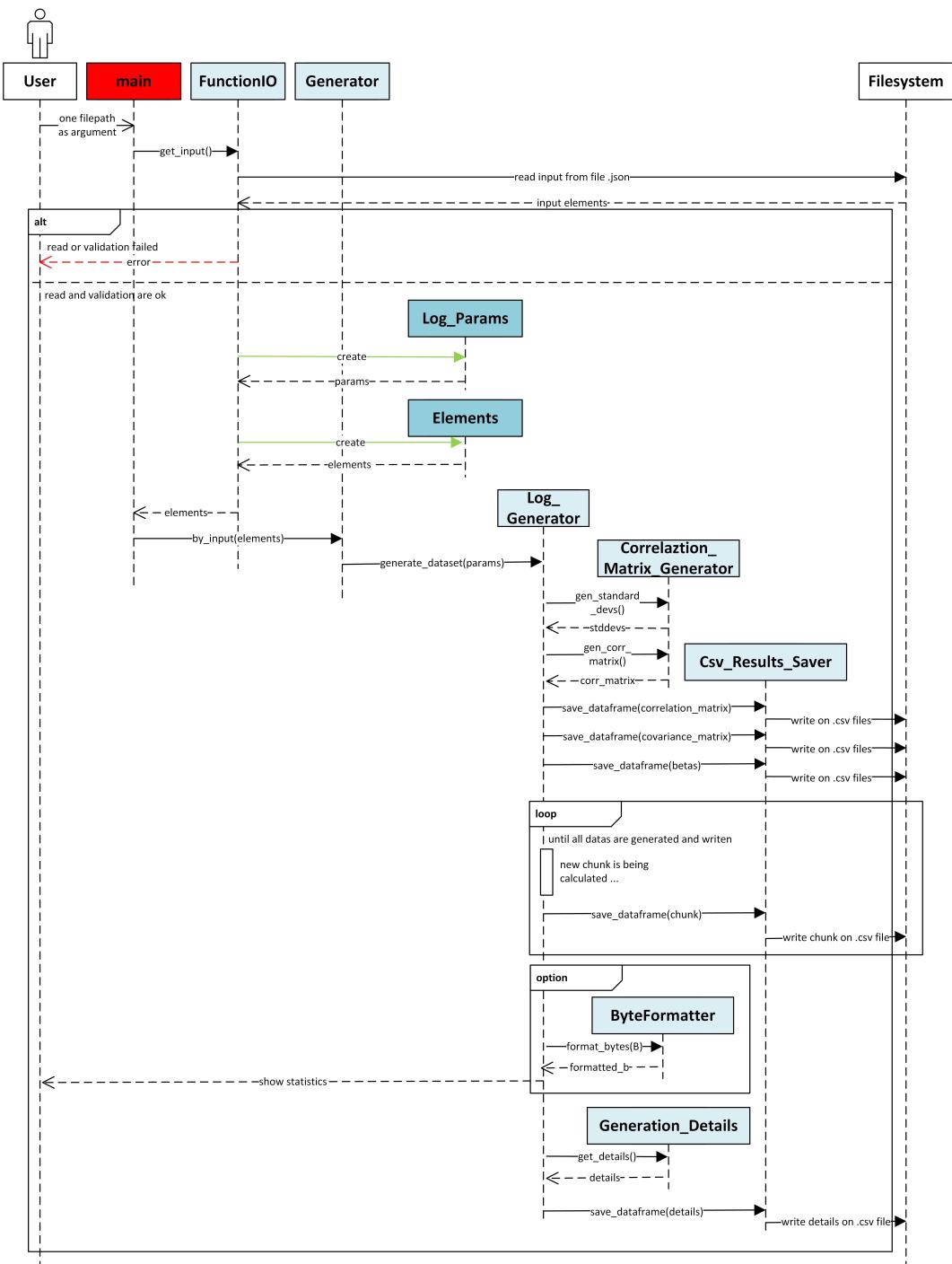
Infine, definito il dettaglio della struttura, si scende nei particolari delle interazioni tra le varie parti del sistema presentando 4 *diagrammi di dettaglio di sequenza* ritenuti significativamente esplicativi.

- Il primo descrive l'immissione dell'input da terminale e la generazione dei dati a partire da Regressione Logistica.
- Il secondo descrive il caricamento dell'input da file .json e la generazione a partire da Regressione Lineare.
- Il terzo descrive l'immissione dell'input da terminale e la generazione dei dati a porzioni sulla base della Regressione Lineare.
- Il quarto descrive il caricamento dell'input da file .json e la generazione dei dati a porzioni sulla base della Regressione Logistica.

## 0.5 Sequence

---





## *0.5 Sequence*

---

- The diagrams presented can easily be translated from a Linear-based generation situation to a Logistic-based situation, in fact all you have to do is replace the creation of class instances of the domain of a regression type with the opposite regression type. *For example:*  
*Log\_Params* will become *LR\_Params*.  
*Log\_Results* will become *LR\_Results*.