NaturalApi

User manual

Version	1.0.0
Approval	Matteo Infantino
Drafting	Francesco Battistella
	Simone Innocente
Check	Francesco Battistella
	Matteo Infantino
State	Approved
Use	External
Addressed to	FourCats
	TealBlue
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
Email	fourcats.unipd@gmail.com

Description
User manual inherent to the NaturalAPI product of the FourCats group, useful for users who want to interact and enjoy the product

Version	Date	Name	Role	Description	Checker	Outcome
1.0.0	14-05-2020	Andrea Polo	Verifier	Additional check	Edoardo Tinto	Positive
	11-05-2020	Andrea Polo	Verifier	Integretion on § 3	Simone Innocente	Positive
0.9.0	07-05-2020	Edoardo Tinto	Verifier	Completed translation § 3	Francesco Battistella	Positive
	06-04-2020	Simone Innocente	Programmer	Updated § 3	Francesco Battistella	Positive
	04-04-2020	Simone Innocente	Programmer	Drafting § 3	Francesco Battistella	Positive
	03-04-2020	Francesco Battistella	Programmer	Drafting § 1 and 2	Matteo Infantino	Positive

Indice

1 Introduction

1.1 Preamble

The current document refers to version 1.0.0 of the product *NaturalAPI*.

1.2 Document purpose

The document purpose is to illustrate the functionality of the software product *NaturalAPI*.
Since the product consists of three independent modules, the functionality for each of them will be shown.

1.3 Product purpuse

The purpose of the product is the creation of a toolkit, *NaturalAPI*, able to automatically generate application programming interfaces (APIs) and related unit tests for a given programming language, starting from features and scenarios in *Gherkin* format and from text documents related to the domain of interest.

1.4 Glossary

At the end of this document there is the appendix A where you can find definitions of terms' definition that may be ambiguous or new for *NaturalAPI* users. These terms are identified by the use of italics.

2 Configuration

2.1 General configuration

The NaturalAPI page is available at the following link https://fourcatsteam.github.io/NaturalAPI_Project/. From here, by following the instructions written on the website, you can download the entire repository or the executable of the individual forms.

2.2 System Requirements

In order to run *NaturalAPI* you need the JVM(Java Virtual Machine) included in the JRE (Java Runtime Environment) available at https://www.java.com/it/download/.
In particular, NaturalAPI is composed of three distinct modules: *NaturalAPI Discover*, *NaturalAPI Design*, *NaturalAPI Develop*
Each module is independent of the others, so you can only run one module at a time.

2.3 NaturalAPI Discover requirements

NaturalAPI Discover needs in input a series of text files in .txt format related to a given domain that can come from different sources.

2.4 NaturalAPI Design requirements

The user must have .feature files in Gherkin format. The following Gherkin keywords are supported:

- Feature;
- Scenario;
- Given;
- When;
- Then;
- And.

In a .feature file all scenarios related to a single actor should be grouped together.
The latter can be specified by placing in the line after the keyword *"Feature:"* the keyword "*As to* ", followed by the name of the actor which the feature is matched.
In the absence of the keyword "*As a*", NaturalAPI Design will assign a default actor "*All*" to all suggestions that will be generated within the feature.
An example .feature file in Gherkin format is shown below.

```
Feature: Guess the word
As a Player
  Scenario: Breaker joins a game
    Given the Maker has started a game with the word "silky"
    When the Breaker joins the Maker's game
    Then the Breaker must guess a word with 5 characters
```

2.5 NaturalAPI Develop requirements

To use NaturalAPI Develop you need to select a valid BAL and PLA from the filesystem. The BAL is the output of NaturalAPI Design and is in JSON format. A valid PLA is in txt format and contains the specifications for each programming language.
In particular, in the first line of this file, the API extension must be present.
Inside the file, moreover, some keywords must be present:

- "**group_action**": indicates the class name;
- "**action_type**": indicates the type of method;
- "**action_name**": indicates the name of the method;
- "**parameter_type**": indicates the type of the method parameter;
- "**parameter_name**": indicates the name of the method parameter;
- "**custom_class**": indicates the name of a custom class;
- "**attribute_type**": indicates the type of an attribute;
- "**attribute_name**": indicates the name of an attribute.
- "**keyword**": indicates the cucucmber keyword for test.
- "**test_stub**": indicates the name of the test.

An example of a PLA file, for the java programming language, is shown below and will be immediately available on NaturalAPI Develop.

```
.java
public class "group_action" {
        public "action_type" "action_name" ("parameter_type" "parameter_name"){

        }
}
custom class
public class "custom_class" {

        private "attribute_type" "attribute_name";

        public void set"attribute_name"("attribute_type" "attribute_name") {
                this."attribute_name" = "attribute_name";
        }

        public "attribute_type" get"attribute_name"() {
                return "attribute_name";
        }

}
test class
        @"keyword" (""test_stub"")
        public void "test_stub"() {
                //please insert an implementation of the test
                //"group_action" "action_name";
                //"action_name"."action_name"("parameter_name");
        }
```

3 Functionalities and use of NaturalAPI

3.1 Preamble

NaturalAPI is composed by three different modules: *NaturalAPI Discover*, *NaturalAPI Design*, *NaturalAPI Develop*. Each module is responsible for some operations:

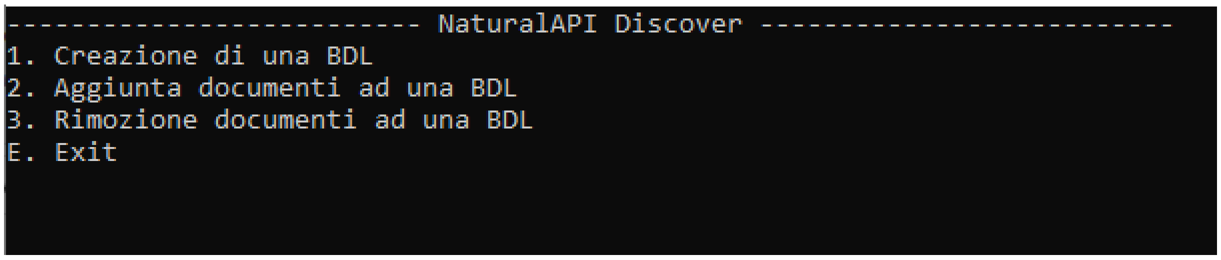
- **NaturalAPI Discover**: the purpose of this module is to find the most used words concerning a specific business domain and how they relate to each other. The first step is to collect business related documents. The result is a Business Domain Language (BDL) which contains the most used terms.
- **NaturalAPI Design**: converts scenarios in Gherkin format into a Business Application Language (BAL) prototype, a pseudolanguage that highlights the various functions required by the scenario and the entities involved. The BAL prototype must then be approved by the user.
  The user can make changes to the prototype to reorganize the functions and make the BAL clearer. Loading a BDL allows the user to see if and how often the suggested and entered terms are present in BDL.
- **NaturalAPI Develop**: uses BAL to produce API in a specific programming language. In order to translate the NaturalAPI Develop a PLA for the target language is required. API suggestions are provided and must be evaluated and accepted by the user before generating the API file.
  NaturalAPI Develop also allows modifing or adding methods to the API if the user deems it necessary.

3.2 NaturalAPI Discover

3.2.1 Start

3.2.1.1 Command Line Interface(CLI)

To run NaturalAPI Discover, go to the product folder and open the terminal. Type *"java -jar -Xmx1280m NaturalAPI_Discover_CLI.jar"*. It will take a few seconds for the program to load all the necessary libraries. After that, a menu will appear in which you will have to choose one of the three functions provided. The home screen will be as follows:



```
-------------------------- NaturalAPI Discover --------------------------
1. Creazione di una BDL
2. Aggiunta documenti ad una BDL
3. Rimozione documenti ad una BDL
E. Exit
```

Figura 1: Menù from NaturalAPI Discover

3.2.1.2 Graphic User Interface(GUI)

To run NaturalAPI Discover with GUI, simply type *"java -jar -Xmx1280m NaturalAPI_Discover_GUI.jar"* in the product folder. You can choose one of the provided features from the new opened menu. (Figure 2)

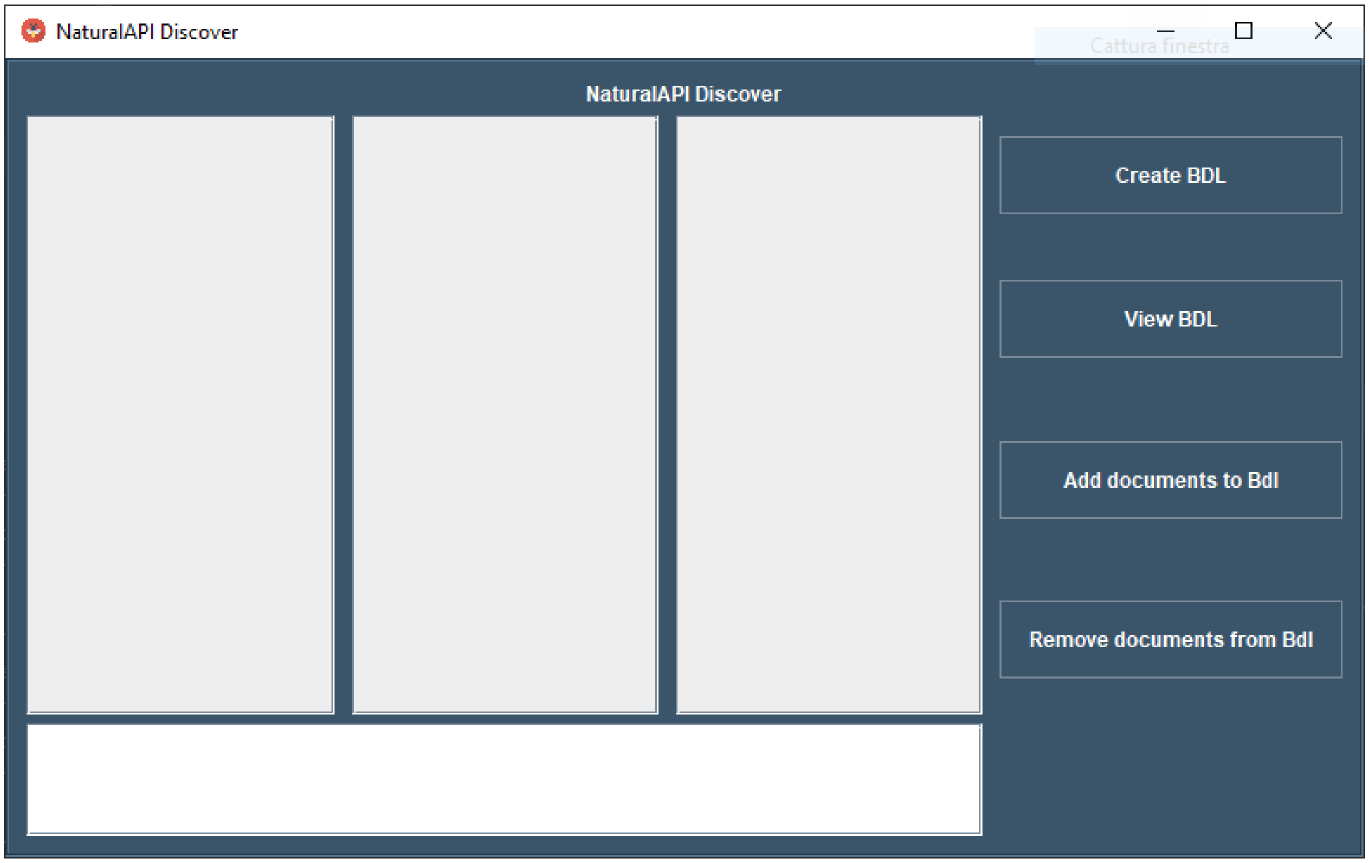


Figura 2: Menù di NaturalAPI Discover GUI

3.2.2 BDL creation

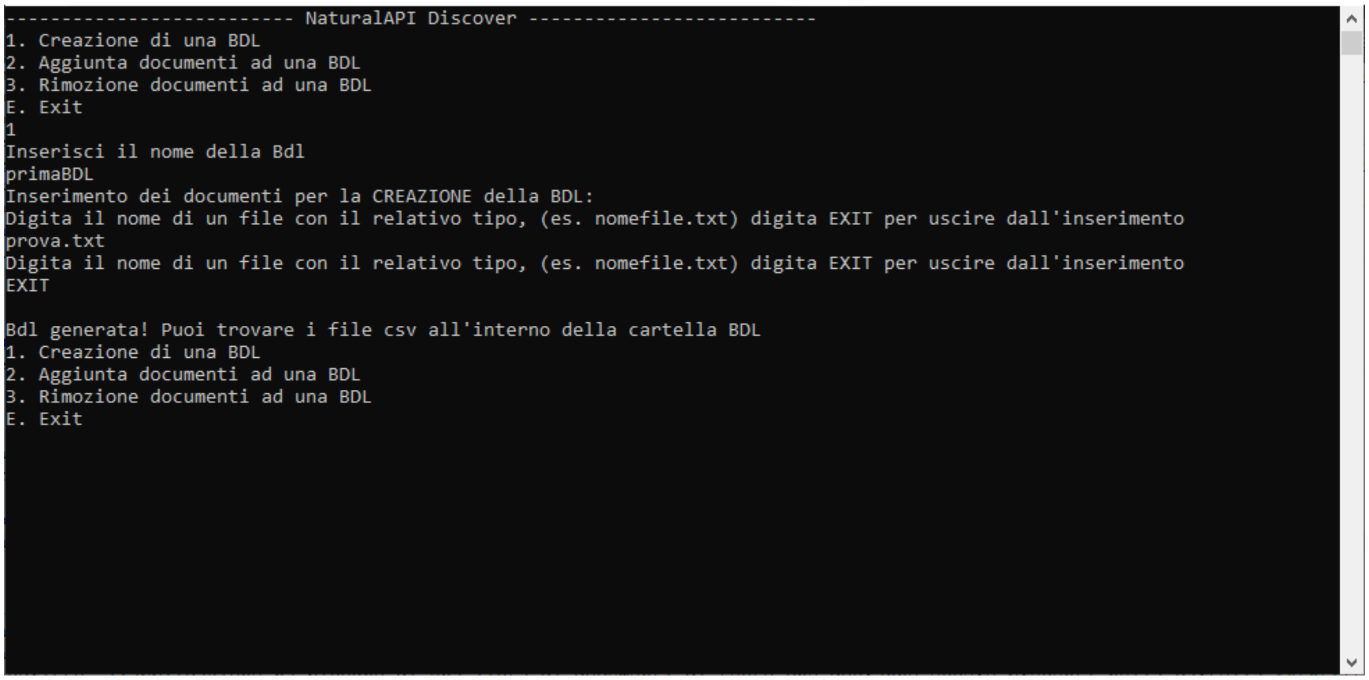
3.2.2.1 Command Line Interface(CLI)

The first functionality made available by NaturalAPI Discover is the creation of a Business Domain Language. The program, before continuing, requires you to type the name you want to give to the BDL. The application also needs a list of text documents that must be present in the folder "*txt_documents*". For the Discover module to parse .txt documents, you must specify the file name and type as follows:

```
documentname.txt
```

N.B. Only .txt files in the folder

are processed. You will be asked for a file name until you decide to type the word "*EXIT*". Then the BDL with the chosen name and documents will be generated. You will find BDL files in the folder "*BDL*" as a .csv files. Here an example of how a BDL is created:



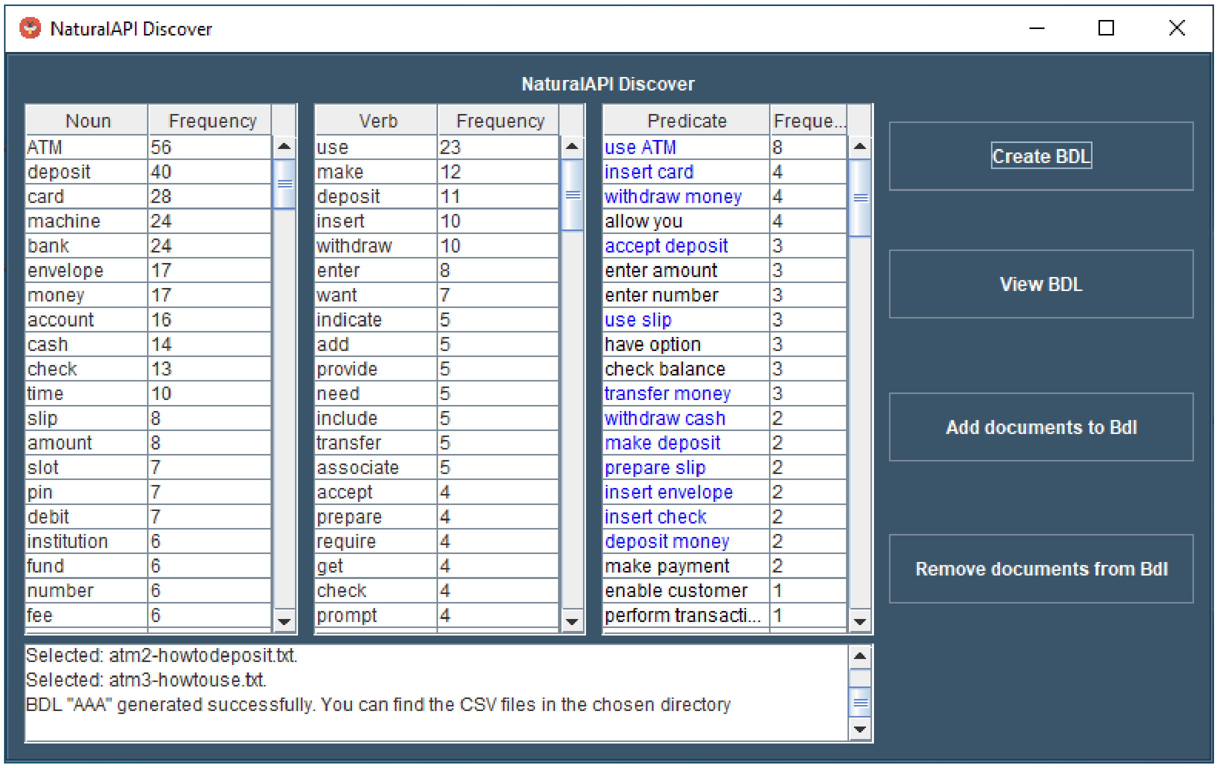
```
------------------------- NaturalAPI Discover -------------------------
1. Creazione di una BDL
2. Aggiunta documenti ad una BDL
3. Rimozione documenti ad una BDL
E. Exit
1
Inserisci il nome della Bdl
primaBDL
Inserimento dei documenti per la CREAZIONE della BDL:
Digita il nome di un file con il relativo tipo, (es. nomefile.txt) digita EXIT per uscire dall'inserimento
prova.txt
Digita il nome di un file con il relativo tipo, (es. nomefile.txt) digita EXIT per uscire dall'inserimento
EXIT

Bdl generata! Puoi trovare i file csv all'interno della cartella BDL
1. Creazione di una BDL
2. Aggiunta documenti ad una BDL
3. Rimozione documenti ad una BDL
E. Exit
```

Figura 3: BDL creation - NaturalAPI Discover

3.2.2.2 Graphic User Interface(GUI)

In order to create a BDL through the graphical interface provided by NaturalAPI Discover, the user must press the "Create BDL" button. A selection window is then displayed, where you can choose the destination directory in which the produced BDL will be saved. Once the folder has been selected and the choice confirmed, a window will open for entering the name of the BDL. In case the user inserts an empty string, he will be asked again to insert the name. Finally a window will be displayed for the insertion of the .txt documents which will be used to produce the BDL. The selection of multiple files is only possible within the same folder. If the .txt files are placed in different folders it is necessary to move them manually or add them later with the "Add documents to BDL" button. **N.B. Only .txt files are processed**. The user can stop the creation at any time before confirming the name by pressing the "Stop" button in the current window. Once the name entry is confirmed, the program will print the name of the documents and the BDL on the screen. Natural API will start to produce the BDL, which will be stored in the destination folder, in the form of three .csv files. If some selected documents are not .txt files a warning will be displayed, the documents will be discarded and the creation will be done with the remaining documents.
The content of the three files (prefixes, nouns and verbs with their relative frequency) is also automatically displayed in a table.
In order to allow the functionality of adding and removing documents to an existing BDL, the .txt files are copied to the destination folder.
The words highlighted in blue, according to the NaturalAPI Discover algorithm, are the most suitable words for a business domain language.



Noun	Frequency
ATM	56
deposit	40
card	28
machine	24
bank	24
envelope	17
money	17
account	16
cash	14
check	13
time	10
slip	8
amount	8
slot	7
pin	7
debit	7
institution	6
fund	6
number	6
fee	6

Verb	Frequency
use	23
make	12
deposit	11
insert	10
withdraw	10
enter	8
want	7
indicate	5
add	5
provide	5
need	5
include	5
transfer	5
associate	5
accept	4
prepare	4
require	4
get	4
check	4
prompt	4

Predicate	Freque...
use ATM	8
insert card	4
withdraw money	4
allow you	4
accept deposit	3
enter amount	3
enter number	3
use slip	3
have option	3
check balance	3
transfer money	3
withdraw cash	2
make deposit	2
prepare slip	2
insert envelope	2
insert check	2
deposit money	2
make payment	2
enable customer	1
perform transacti...	1

```
Selected: atm2-howtodeposit.txt.
Selected: atm3-howtouse.txt.
BDL "AAA" generated successfully. You can find the CSV files in the chosen directory
```

Figura 4: BDL creation - NaturalAPI Discover

3.2.3 Add documents to existing BDL

3.2.3.1 Command Line Interface(CLI)

This feature requires as a pre-condition to have already generated at least one BDL. As for the creation, after choosing this feature you will be asked for a name and a series of documents as described above. These elements are:

- Name: indicates the BDL you want to modify;
- Documents: these are the .txt files you want to add to the BDL.

After entering all inputs, a message will be shown based on the outcome of the operation.
N.B. Documents already added will not be counted

3.2.3.2 Graphic User Interface(GUI)

From the graphical interface the addition of new documents to the BDL will first require the selection of the BDL to be integrated. Pressing the "Add documents to BDL" button will open a selection window inside the source folder where the BDL is stored, then the BDL name is required. In case the user inserts an empty string he is asked again to insert the name of an existing BDL.
Once the name is confirmed, the user is asked to select the documents to add to the BDL, in the same way as described in the creation process.
N.B. Documents already added will not be counted.
The user can stop the creation at any time before confirming the name by pressing the "Stop" button in the current window. Once the selection of the documents has been confirmed, NaturalAPI Discover will proceed with the addition of the documents to the BDL. As for the creation, if some selected documents are not in the correct format they will be discarded and a warning will be displayed. **N.B. To allow the correct addition of documents it is required the presence of an association file named_bdl.json inside the source folder, otherwise an error will be displayed**. Once the process is complete, the .csv files will be updated and the BDL will be automatically displayed in the tables.

3.2.4 Removing documents from a BDL

3.2.4.1 Command Line Interface(CLI)

Removing a document from a BDL (in case it has been added by mistake or if it does not belong to the specific domain) is similar to add documents. The name of the BDL and the documents are selected as explained before. If some of the selected documents have not been used for the production of the BDL, they will be discarded and the removal will continue with the remaining documents.

3.2.4.2 Graphic User Interface(GUI)

As for adding documents, once pressed the "Remove documents from BDL" button, you will be asked to select the source folder of the BDL, enter the name of the BDL and select the documents to be removed in the same way. In this case, however, the documents to be removed must necessarily be selected from the same folder of the BDL.
As for the creation and addition processes, if some selected documents are not in the correct format they will be discarded and a warning will be displayed. **N.B. To allow the correct removal of documents it is required the presence of the association file named_bdl.json inside the source folder, otherwise an error will be displayed**. As for addition process, once the process is complete, the .csv files will be updated and the BDL will be displayed automatically.

3.2.5 Display BDL

3.2.5.1 Graphic User Interface(GUI)

A feature available only in the NaturalAPI Discover GUI is to display a specific BDL. By pressing the "View BDL" button you can select the source folder of the BDL and then type the name of the BDL you want to display. Then NaturalAPI Discover gives you the possibility to choose different types of visualizations:

- **Show All:** Show all words within the BDL;
- **More probable words**: Show all words with total frequency/word ratio greater than 0.01 (probability of encountering the word in documents greater than 1%);
- **First 15 words**: Show the first 15 words sorted by frequency.

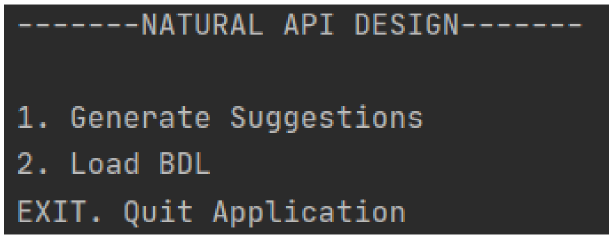
Once the display type is confirmed, the BDL is retrieved and displayed.

3.3 NaturalAPI Design

3.3.1 Start

3.3.1.1 Command Line Interface(CLI)

To run NaturalAPI Design, go to the product folder, open the terminal and type:
"java -jar -Xmx1280m NaturalAPI_Design_CLI.jar".
At startup(which may take some time), you are given the option to load a BDL or to continue directly with the suggestion generation without it.



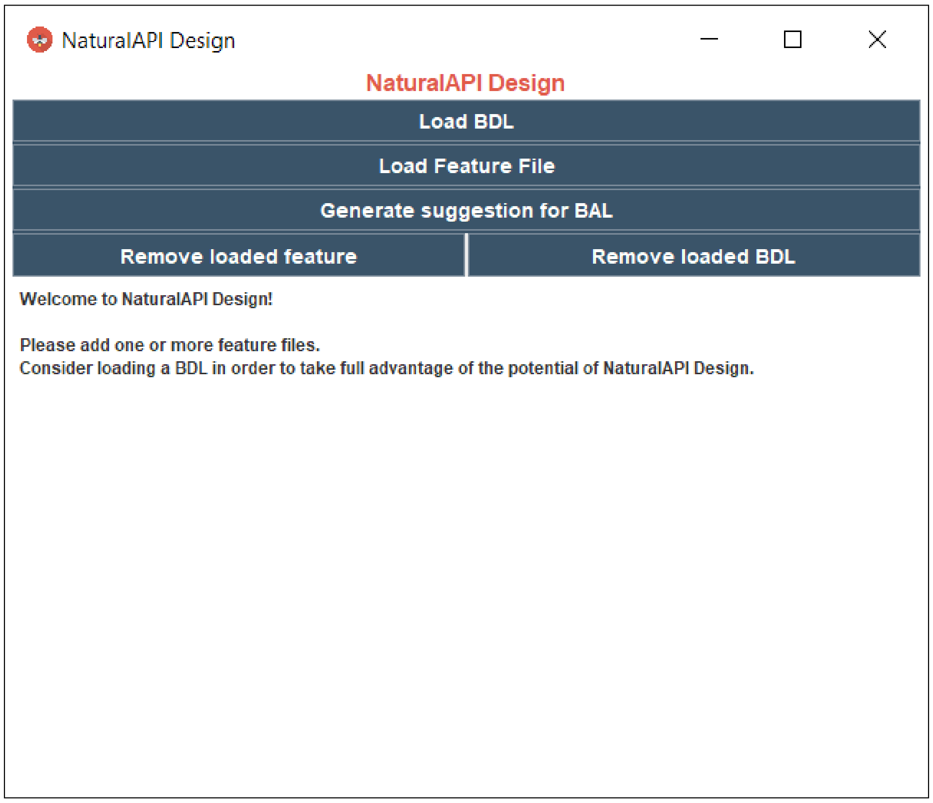
```
-------NATURAL API DESIGN-------

1. Generate Suggestions
2. Load BDL
EXIT. Quit Application
```

Figura 5: Initial window - NaturalAPI Design CLI

3.3.1.2 Graphic User Interface(GUI)

To run NaturalAPI Design, go to the product folder, open the terminal and type:
"java -jar -Xmx1280m NaturalAPI_Design_GUI.jar".
Startup may take some time to allow the necessary libraries to load.
The initial window is the following.



Welcome to NaturalAPI Design!

Please add one or more feature files.
Consider loading a BDL in order to take full advantage of the potential of NaturalAPI Design.

Figura 6: Initial window - NaturalAPI Design GUI

Using the various buttons you can upload Gherkin files, generate BAL creation suggestions and upload BDL.csv files.

3.3.2 Generate suggestions

3.3.2.1 Command Line Interface(CLI)

The application allows you to perform the generation of suggestions given one or more Gherkin files in input.
To continue with the generation, the user is invited to enter the path of the .feature files to be analyzed.

```
Enter the path of the gherkin feature file, digit EXIT when you're done.
```

An input example is the following:

```
C:\Users\username\Desktop\MyFeatureFiles\featureName.feature
```

NaturalAPI Design keeps asking for path until you type "EXIT". Automatically the scenarios of the received files are divided and different suggestions are proposed, for actions that the actor can perform within each scenario. An example of generation without BDL could be the following:

```
----SCENARIO: 0) Withdraw cash
    Given the user is in front of an ATM
    When the user inserts the card in the ATM
    And the user enters the pin of the card
    And the user enters the amount of cash he wants to withdraw
    Then the ATM should check the pin
    And the ATM should give the user the cash

0) @Given the user is in front of an ATM
1) void insert_card (string card)
2) void enter_pin (string pin)
3) void enter_amount (string amount)
4) void check_pin (string pin)
5) void give_cash (string cash)
```

An example with BDL would be the following:

```
----SCENARIO: 0) Withdraw cash
    Given the user is in front of an ATM
    When the user inserts the card in the ATM
    And the user enters the pin of the card
    And the user enters the amount of cash he wants to withdraw
    Then the ATM should check the pin
    And the ATM should give the user the cash

0) @Given the user is in front of an ATM
1) void insert_card[5] (string card[37])
2) void enter_pin[1] (string pin[10])
3) void enter_amount[1] (string amount[7])
4) void check_pin[0] (string pin[10])
5) void give_cash[1] (string cash[17])
```

where terms enclosed in square brackets indicate the frequency of the term or predicate in the BDL.
In both cases a series of options are displayed for the user:

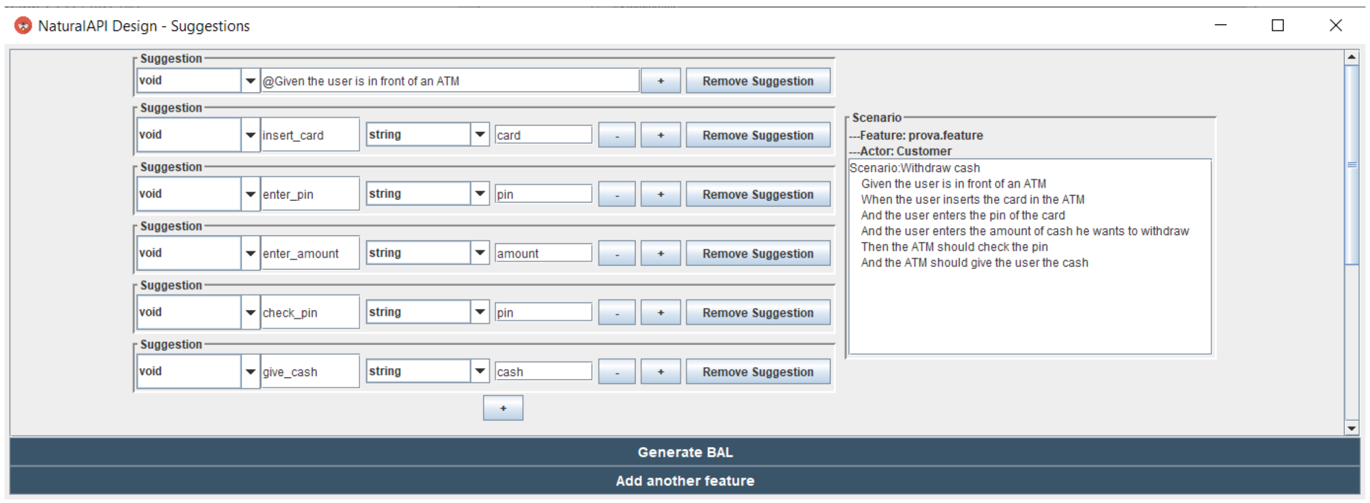
```
What do you want to do?
1. Modify suggestion
2. Delete suggestion
```

```
3. Add suggestion
4. Add new feature
5. Generate BAL
Digit EXIT to abort and go back to the main menu.
```

To access one of the listed functionalities, simply enter the number related to it. The user will be guided through the various operations.
By typing "EXIT" you return to the initial menu, from which you can start a new generation.

3.3.2.2 Graphic User Interface(GUI)

After inserting a Gherkin file via the "Load Feature File" button, you can generate suggestions by pressing the "Generate suggestion for BAL" button. The tool divides the various scenarios received and generates the suggestions. The screenshot displayed will be as follows:
N.B. If no Gherkin file is loaded and you try to generate suggestions, the program generates an error.
The following windows shows the outcome of generating suggestions **WITHOUT** a BDL loaded.

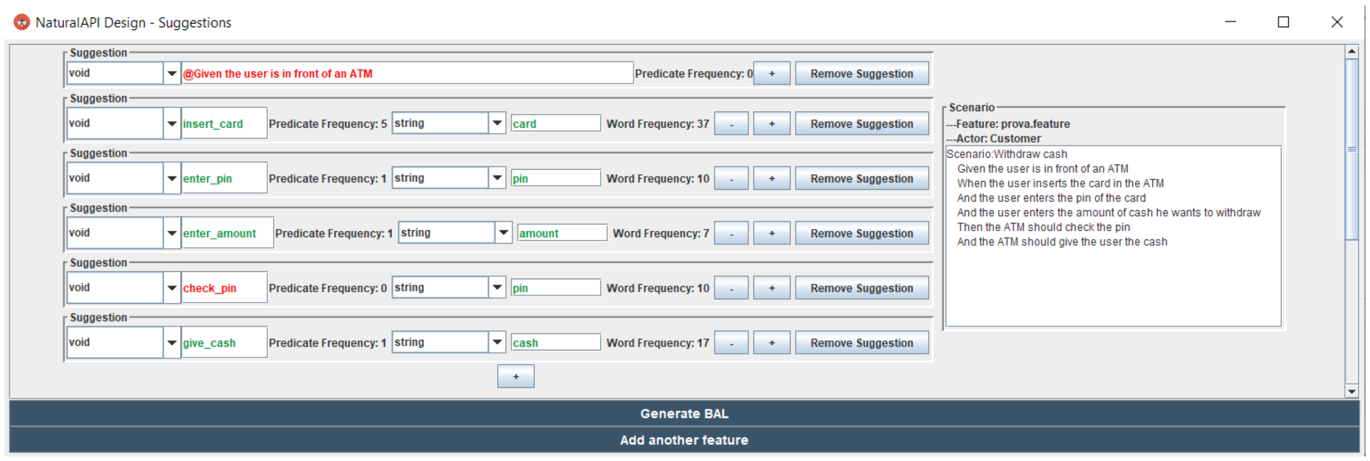


Suggestion				
void	@Given the user is in front of an ATM			
void	insert_card	string	card	
void	enter_pin	string	pin	
void	enter_amount	string	amount	
void	check_pin	string	pin	
void	give_cash	string	cash	

Scenario
---Feature: prova.feature
---Actor: Customer
Scenario:Withdraw cash
Given the user is in front of an ATM
When the user inserts the card in the ATM
And the user enters the pin of the card
And the user enters the amount of cash he wants to withdraw
Then the ATM should check the pin
And the ATM should give the user the cash

Figura 7: Create suggestions for BAL - NaturalAPI Design GUI

The following windows is generated after correctly adding a BDL.



Suggestion					
void	@Given the user is in front of an ATM	Predicate Frequency: 0			
void	insert_card	Predicate Frequency: 5	string	card	Word Frequency: 37
void	enter_pin	Predicate Frequency: 1	string	pin	Word Frequency: 10
void	enter_amount	Predicate Frequency: 1	string	amount	Word Frequency: 7
void	check_pin	Predicate Frequency: 0	string	pin	Word Frequency: 10
void	give_cash	Predicate Frequency: 1	string	cash	Word Frequency: 17

Scenario
---Feature: prova.feature
---Actor: Customer
Scenario:Withdraw cash
Given the user is in front of an ATM
When the user inserts the card in the ATM
And the user enters the pin of the card
And the user enters the amount of cash he wants to withdraw
Then the ATM should check the pin
And the ATM should give the user the cash

Figura 8: Create suggestions for BAL with BDL - NaturalAPI Design GUI

The two screens differ from the addition of the words frequency. Every action and parameter is controlled using the BDL.
If the word appears in the BDL the color turns green and its frequency is extracted, otherwise it turns red.
This provides the user a way to see if the suggestions he chooses are more or less used inside the business documents selected with the Discover.

N.B. The "suggestions" starting with "@" are those for which no predicate has been found to be associated with the step. In this case the user can decide to change the name so that a link between step and action can be inserted in BAL and allow subsequent API and test creation. Otherwise, the default association will only allow you to generate a test that will not have any implementation.
Also note that NaturalAPI Design provides a new suggestion only if it has not already been taken from a previous step (of the same or a previously analyzed scenario), in order to avoid duplication.

3.3.2.3 Delete suggestion

Command Line Interface(CLI)
When you decide to delete a suggestion, the program requires the scenario id and the suggestion id itself. Both are provided when the suggestions are generated and correspond to the number close to the scenario and the suggestion respectively.

```
Please insert the id of the scenario for the suggestion you want to delete.
...

Please insert the id of the suggestion you want to delete.
...
```

After correctly entering the two required inputs, the suggestion is deleted and the updated list is displayed. The program will ask the user again if he wants to delete some suggestions.

Graphic User Interface(GUI)
To delete a suggestion, simply press the "Remove Suggestion" button for the suggestion you want to delete.
The program will ask the user for further confirmation of the deletion. After pressing "Yes", the suggestion is deleted.

3.3.2.4 Add a new feature

Command Line Interface(CLI)
To add a new feature file type the number of the respective option in the menu and enter the path of the file, in the same way as in the generation phase. The program will automatically display the entire updated list of suggestions and propose to the user the list of possible options to execute.

Graphic User Interface(GUI)
From the suggestion windows, to add new features, simply press the "Add another feature" button. After choosing the file, the suggestions are automatically generated and displayed in the table.

3.3.2.5 Generate BAL

Command Line Interface(CLI)
The only input required for BAL generation is the path to save it followed by the name of the BAL itself.

```
Please, enter the path including the name for the BAL
```

Input example:

`C:\Users\username\Desktop\MyBALFiles\BALname`

After entering the path according to the required directions, the BAL will be created and can be found within the specified directory.

Graphic User Interface(GUI)

Through the "Generate BAL" button it will be possible to generate the BAL by choosing the destination folder, or add other Gherkin format files through the "Add another feature" button. This last button will load the received files and generate other suggestions that will be displayed within the program.

3.3.2.6 Modify a suggestion

The possible modifications of a suggestion are the following:

- Modify action name or parameter name;
- Addition or deletion of a parameter;
- Choose other primitive types;
- Create a custom type;
- Add a new action.

Command Line Interface(CLI)

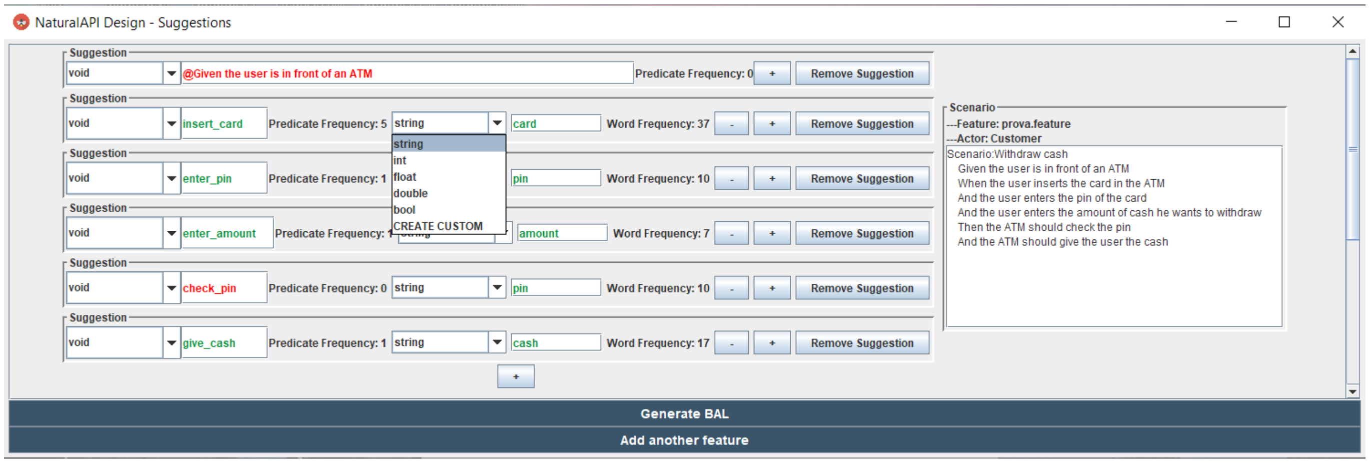
The editing functionalities of the suggestions are accessible after the selection of the respective menu option, shown together with the suggestion generation option. This choice leads to the following menu:

```
1. Modify action name
2. Modify action type
3. Modify parameter name
4. Modify parameter type
5. Add parameter
6. Remove parameter
```

All modification and removal operations will require the insertion of a series of id, all identifiable from the previously generated outputs. The creation of the custom type can be started after choosing to modify a type.

Graphic User Interface(GUI)

All editing operations originate from the suggestion screen.



Suggestion: void | @Given the user is in front of an ATM | Predicate Frequency: 0

Suggestion: void | insert_card | Predicate Frequency: 5 | string | card | Word Frequency: 37

Suggestion: void | enter_pin | Predicate Frequency: 1 | pin | Word Frequency: 10

Suggestion: void | enter_amount | Predicate Frequency: 1 | amount | Word Frequency: 7

Suggestion: void | check_pin | Predicate Frequency: 0 | string | pin | Word Frequency: 10

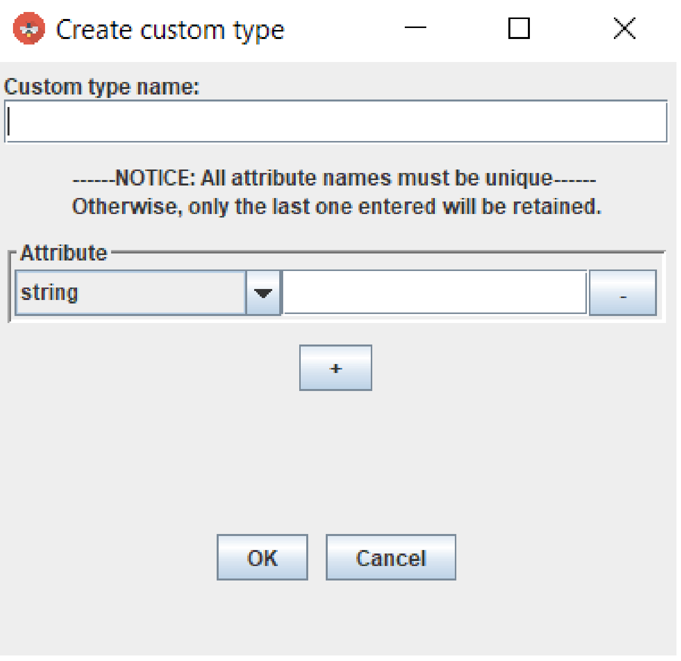
Suggestion: void | give_cash | Predicate Frequency: 1 | string | cash | Word Frequency: 17

Scenario
---Feature: prova.feature
---Actor: Customer

```
Scenario:Withdraw cash
  Given the user is in front of an ATM
  When the user inserts the card in the ATM
  And the user enters the pin of the card
  And the user enters the amount of cash he wants to withdraw
  Then the ATM should check the pin
  And the ATM should give the user the cash
```

Figura 9: Modify suggestion - NaturalAPI Design GUI

Through the text boxes you can change the name of the actions and of the parameters. . The addition of other parameters is done through the "+" button inside the suggestion, you will then be asked the name of the new parameter, then the suggestion is updated with the new parameter inserted. The removal is done by using the "-". button instead. The addition of other actions is done through the "+" button located just below the suggestions or at the side of the scenario if they are not present. You will be asked for the name of the action to add. The possibility of choosing other types for actions and parameters is done through a drop-down menu, which also allows the creation of custom types. You can access this functionality by selecting the last option of the menu, called "CREATE CUSTOM". In this case, the following window will be displayed, where it will be possible to choose the name, add or remove attributes and be modify their names and types.
mbox.



------NOTICE: All attribute names must be unique------
Otherwise, only the last one entered will be retained.

Figura 10: Create a custom type - NaturalAPI Design GUI

3.3.3 Removing feature file and loaded BDL file

3.3.3.1 Command Line Interface(CLI)

The possibility to remove a loaded BDL will be available in the main menu only if the BDL has been loaded. To delete the list of uploaded files after a first generation of suggestions, simply return to the main menu by typing "EXIT".

3.3.3.2 Graphic User Interface(GUI)

To remove the previously loaded feature files or the BDL, you can use the appropriate buttons: "Remove loaded feature" and "Remove loaded BDL". In case you make a mistake while selecting the files, you can remove them quickly and easily.

3.3.3.3 Input example

```
Feature: Bank withdraw
As a Customer
  Scenario: Withdraw cash
   Given the user is in front of an ATM
   When the user inserts the card in the ATM
   And the user enters the pin of the card
   And the user enters the amount of cash he wants to withdraw
   Then the ATM should check the pin
   And the ATM should give the user the cash

  Scenario: Expired card
   Given the user is in front of an ATM
   When the user inserts the card in the ATM
   And the user enters the pin of the card
   And the user enters the amount of cash he wants to withdraw
   Then the ATM should check the validity of the card
   And the ATM should display an error message
```

3.3.3.4 Output example

```
{
"actors" : [ {
        "name" : "Customer",
        "actions" : [ {
                "name" : "check_pin",
                "type" : {
                        "name" : "void",
                        "attributes" : null
                },
                "scenario" : "Withdraw cash",
                "step" : "the ATM should check the pin",
                "objectParams" : [ {
                        "name" : "pin",
                        "type" : {
                                "name" : "int",
                                "attributes" : null
                        },
                        "required" : true
                } ]
        }, {
                "name" : "display_message",
                "type" : {
                        "name" : "void",
                        "attributes" : null
                },
                "scenario" : "Expired card",
                "step" : "the ATM should display an error message",
                "objectParams" : [ {
                        "name" : "message",
                        "type" : {
                                "name" : "CustomMessage",
                                "attributes" : {
                                        "title" : "string",
                                        "content" : "string"
                                }
                        },
                        "required" : true
                } ]
        } ]
} ]
}
```

3.4 NaturalAPI Develop

In the same way as the previous modules, to run NaturalAPI Develop, just go to the product folder, open the terminal and type the command "$java-jarNaturalAPI\_Develop\_CLI.jar$" for the CLI or "$java-jarNaturalAPI\_Develop\_GUI.jar$" for the GUI. This module takes care of the API creation, given a BAL and a PLA text file, as specified in section 2.5.

3.4.1 Start

3.4.1.1 Graphic User Interface(GUI)

The main window of NaturalAPI Develop is the following.

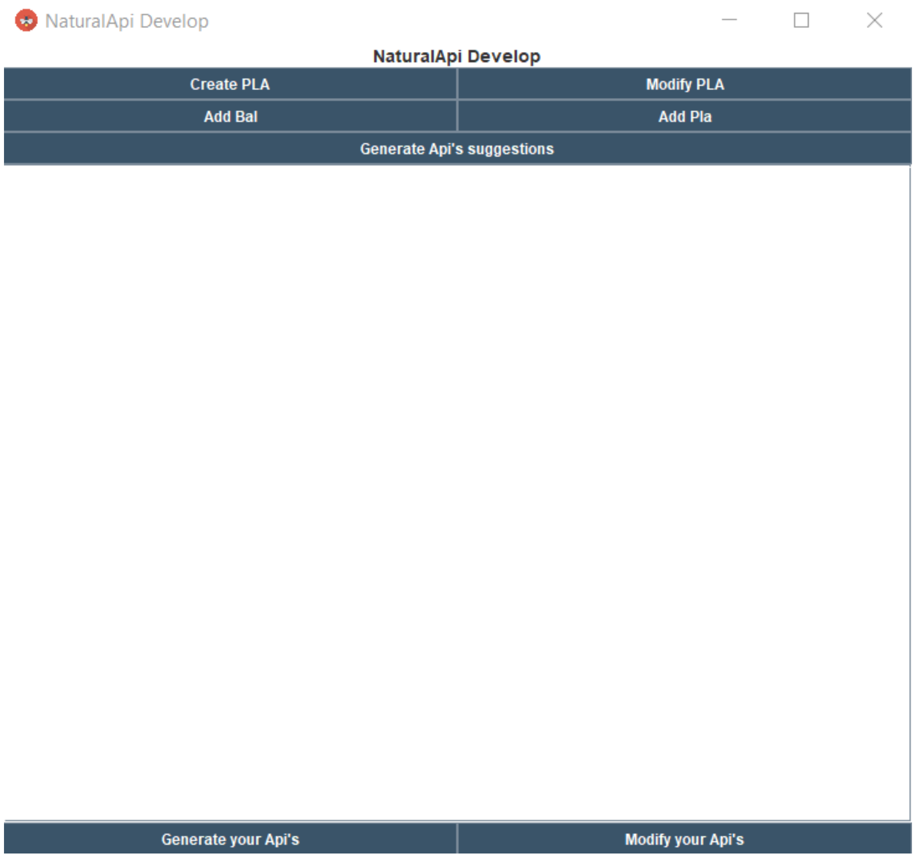
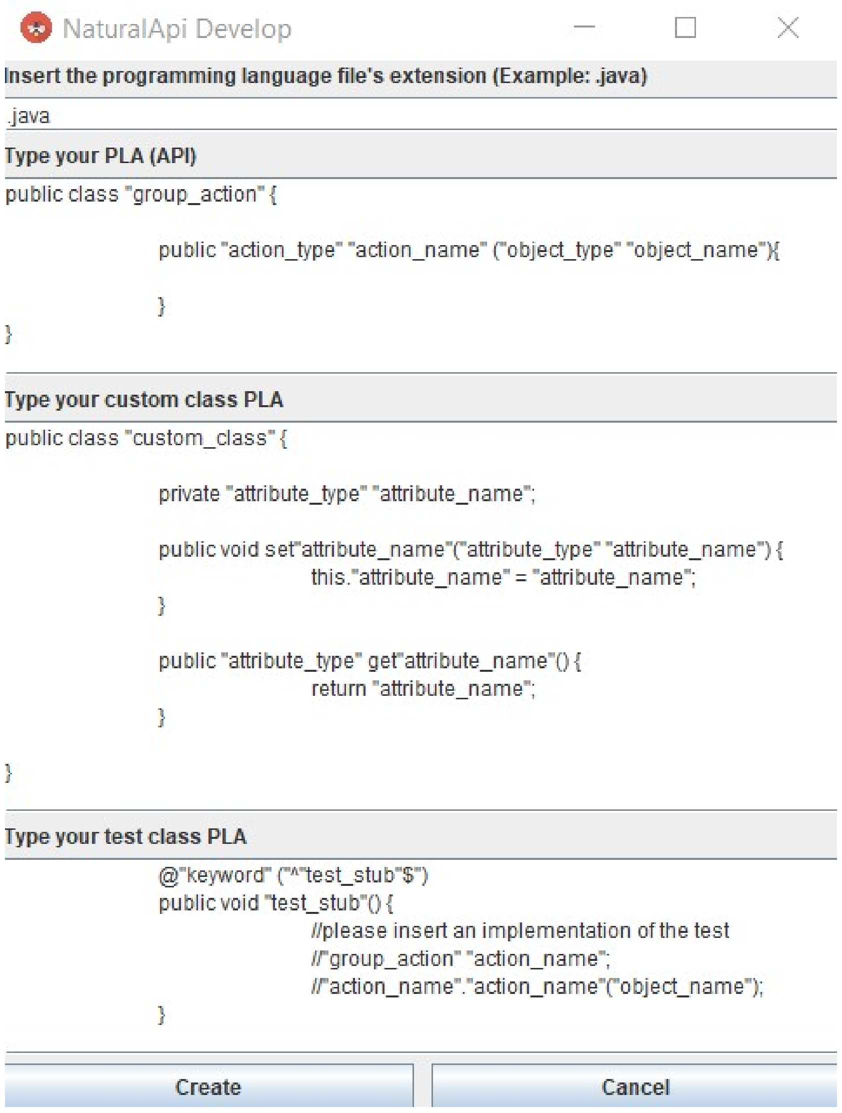


Figura 11: Menu - NaturalAPI Develop GUI

3.4.2 Create a PLA

3.4.2.1 Graphic User Interface(GUI)

Through a graphical interface it is possible to create a PLA for a specific programming language. After pressing the "Create PLA" button another window will appear where you can enter some data.



Insert the programming language file's extension (Example: .java)

.java

Type your PLA (API)

```
public class "group_action" {

            public "action_type" "action_name" ("object_type" "object_name"){

            }
}
```

Type your custom class PLA

```
public class "custom_class" {

            private "attribute_type" "attribute_name";

            public void set"attribute_name"("attribute_type" "attribute_name") {
                        this."attribute_name" = "attribute_name";
            }

            public "attribute_type" get"attribute_name"() {
                        return "attribute_name";
            }

}
```

Type your test class PLA

```
            @"keyword" ("^"test_stub"$")
            public void "test_stub"() {
                        //please insert an implementation of the test
                        //"group_action" "action_name";
                        //"action_name"."action_name"("object_name");
            }
```

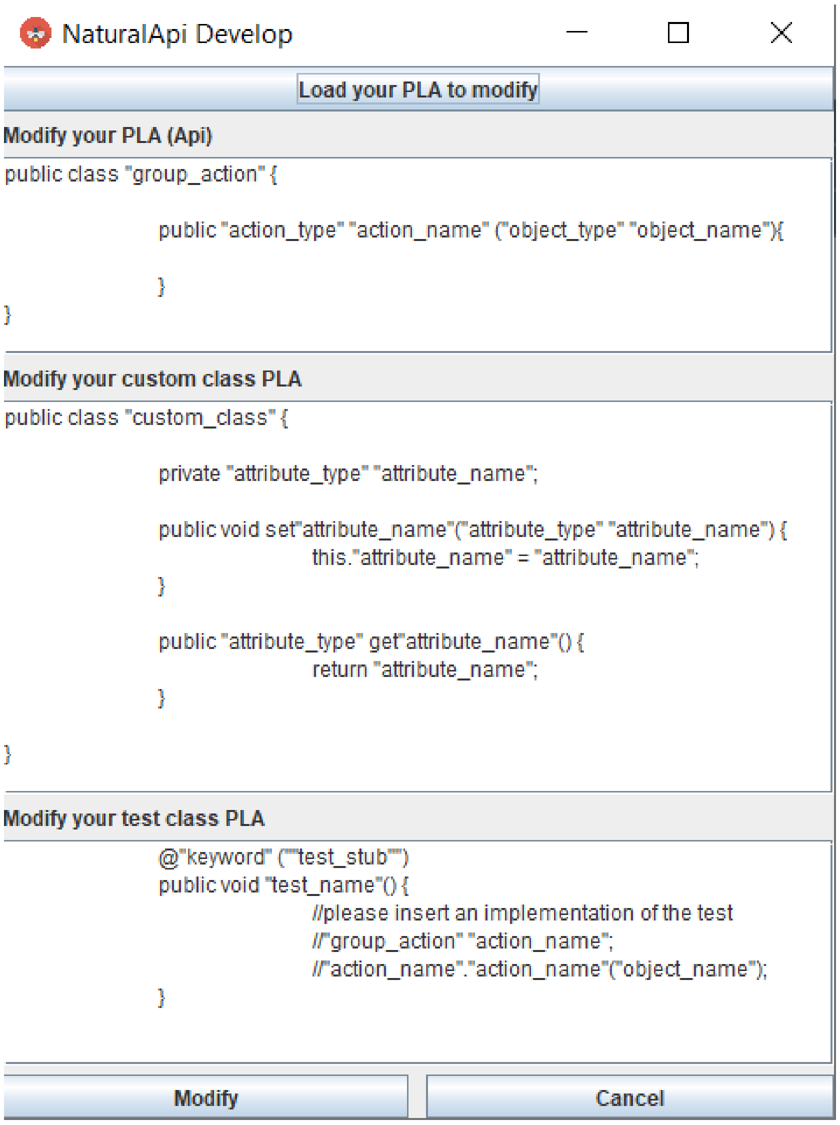
Figura 12: Menu - NaturalAPI Develop GUI

In the first text box you must enter the extension of the chosen programming language.
After that the actual PLA is required. In the first box you should enter the structure of an API, then in the next box you should enter the structure of a class with an attribute and the methods set and get. In the last box you should enter the structure of the tests. **N.B. the PLA must contain the keywords specified in the NaturalAPI Develop Requirements section**. After pressing the ”Create” button, the new PLA will be generated and the program will display whether the process was successful or not.

3.4.3 Modify a PLA

3.4.3.1 Graphic User Interface(GUI)

Another feature offered by NaturalAPI Develop's GUI is the modification of a PLA. By pressing the ”Modify PLA” button the program will display a window to insert modifications. After loading the PLA with the appropriate button, it will be displayed and you will be able to edit it.



```
public class "group_action" {

                public "action_type" "action_name" ("object_type" "object_name"){

                }
}
```

```
public class "custom_class" {

                private "attribute_type" "attribute_name";

                public void set"attribute_name"("attribute_type" "attribute_name") {
                                this."attribute_name" = "attribute_name";
                }

                public "attribute_type" get"attribute_name"() {
                                return "attribute_name";
                }

}
```

```
                @"keyword" (""test_stub"")
                public void "test_name"() {
                                //please insert an implementation of the test
                                //"group_action" "action_name";
                                //"action_name"."action_name"("object_name");
                }
```

Figura 13: Menu - NaturalAPI Develop GUI

3.4.4 Create API suggestions

3.4.4.1 Command Line Interface(CLI)

The first thing the program requires is the pathof the BAL file.
A BAL is a json file that must be specified in this way:

```
C:\Users\pippo\Desktop\nomebal.json
```

In the same way, a .txt PLA file will be requiredr. More information can be found in section 2.5. The PLA file must be specified as follows:

```
C:\Users\pippo\Desktop\nomePLA.txt
```

Then, after entering the required input, the program asks the user if he wants to generate API suggestions.

```
Do you want to create your APIs suggestion? 1: YES or 2:NO
```

Typing 1 the program will show some suggestions derived from the BAL analysis in the programming language chosen by the PLA.
Here is an example.

```
-----------  API ID : 0-----------
public class CheckPin {
        public void checkPin (int pin){}
}

-----------  API ID : 1-----------

public class EnterPin {
        public void enterPin (int pin){}
}

-----------  API ID : 2-----------

public class DisplayMessage {
        public void displayMessage (string message){}
}

-----------  API ID : 3-----------

public class EnterAmount {
        public void enterAmount (double amount){}
}

-----------  API ID : 4-----------

public class CheckValidity {
        public void checkValidity (bool validity){}
}

-----------  API ID : 5-----------

public class WithdrawCash {
        public double withdrawCash (double cash){}
}
```

3.4.4.2 Request adding a new BAL

After generating the suggestions, the user is asked if he wants to add another BAL to the API in the following way:

```
Do you want to add another BAL? 1:YES or 2:NO
```

Typing 1 the Develop module requires two inputs in the same way as shown in creating suggestions in section 3.4.1. Afterwards the previously generated suggestions and those generated by the new BAL will be shown on screen. Finally the program asks again if you want to generate the APIs.

3.4.4.3 Request generating a new API

The user is asked if he wants to generate the API or if he wants to modify it.

```
Do you want to generate your APIs? 1:YES or 2:NO
```

Acceptation of generated API

By typing 1 the user accepts the API generation: the APIs are created in the specified path.

Reject generating API

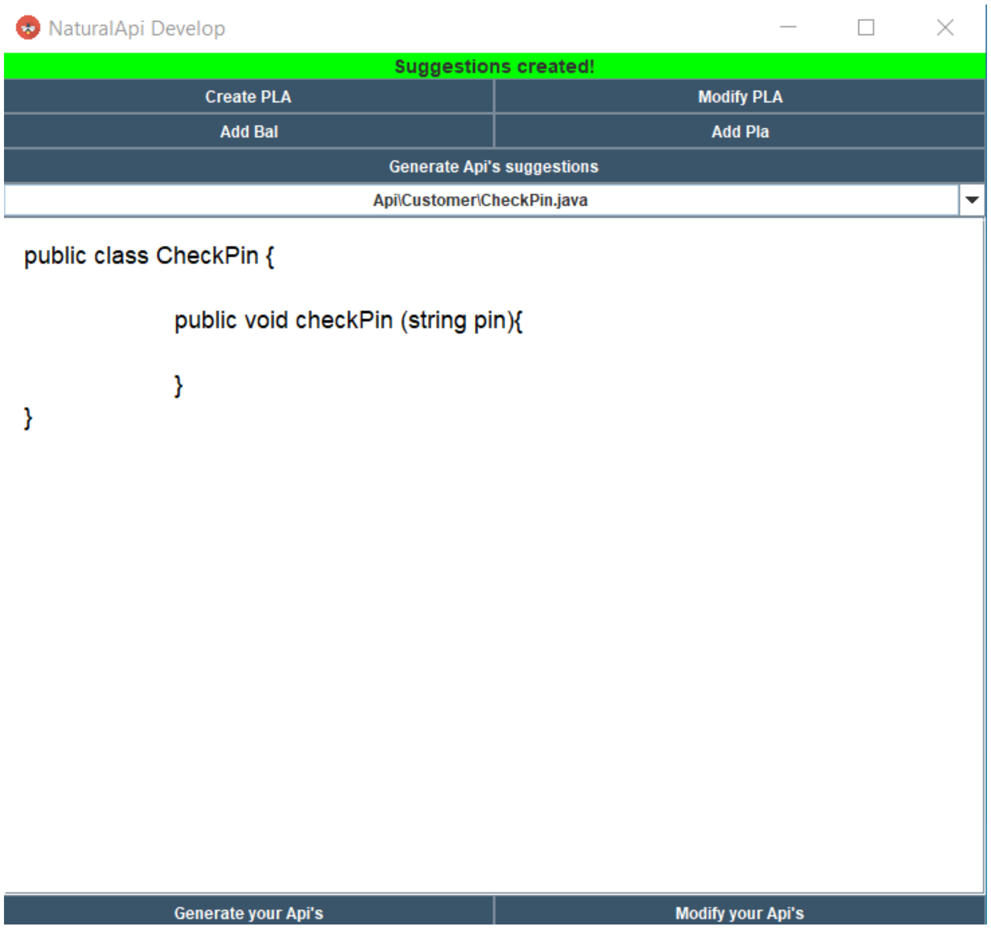
By typing 2 the user refuses to generate the API, and the application makes the following request:

```
Do you want to modify an API? 1:YES or 2:NO
```

N.B The modification only refers to the chosen PLA. Typing 1 the module changes the API according to the instructions described in the next section. Pressing 2 the program will end.

3.4.4.4 Graphic User Interface(GUI)

In order to generate API suggestions, you must first enter a BAL and a PLA using the "Add BAL" and "Add PLA" buttons. After uploading the required files, simply press the "Generate API's Suggestion" button to create suggestions.



```
public class CheckPin {

            public void checkPin (string pin){

            }
}
```

Figura 14: API generation - NaturalAPI Develop GUI

The suggestions can be navigated through the list displayed on video.

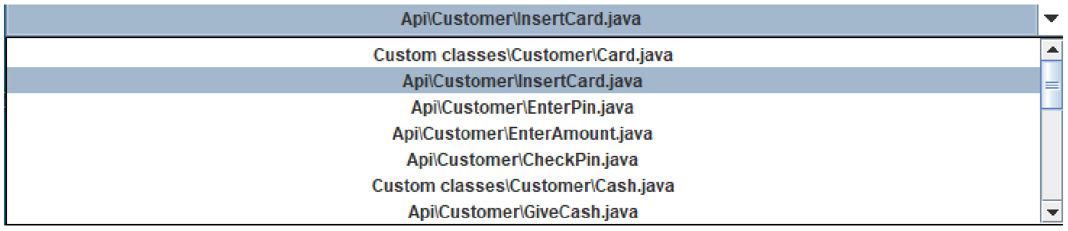

Api\Customer\InsertCard.java
Custom classes\Customer\Card.java
Api\Customer\InsertCard.java
Api\Customer\EnterPin.java
Api\Customer\EnterAmount.java
Api\Customer\CheckPin.java
Custom classes\Customer\Cash.java
Api\Customer\GiveCash.java

Figura 15: API list - NaturalAPI Develop GUI

API generation
To generate the actual API's, just press the "Generate your Api's" button and choose a path for them. The APIs will be saved divided in 3 directories (api, custom_classes and test), each one divided in a directory for each actor.
N.B. You must first generate the suggestions.

3.4.5 Modify an API

3.4.5.1 Command Line Interface(CLI)

When the user decides to modify the API, the API id is requested. The id is displayed when the suggestions are generated.

```
Digit the ID of the first API to modify.

Digit the ID of the first API to modify.
```

Then the PLA file is requested in the same way as the shown in the suggestions creation.
After entering the PLA, the list of new suggestions is automatically displayed on the screen. You are also asked if you want to generate the API.

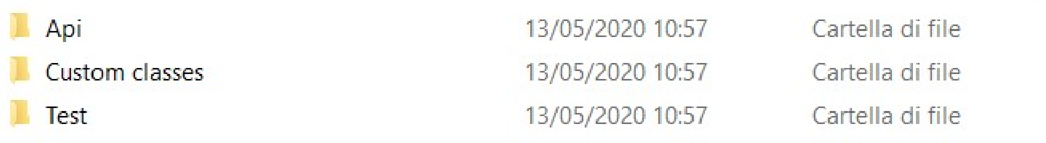
3.4.5.2 Graphic User Interface(GUI)

Since the component where the API is displayed is a text area, you can edit the API as you wish.
To make the changes permanent, however, you need to press the "Modify your Api's" button. All changes will be saved automatically, and when you want to, you can generate the API using the appropriate button.

3.4.6 Input example

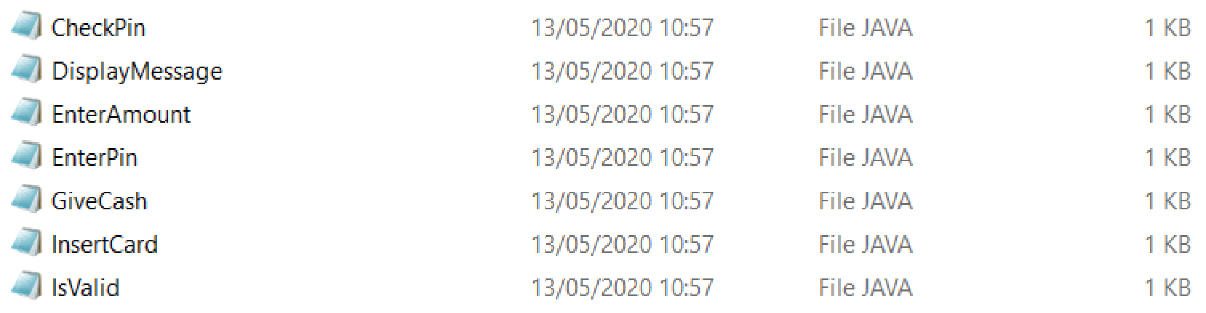
The NaturalAPI Develop input is the output of NaturalAPI Design available in section 3.3.1.4.

3.4.7 Output example



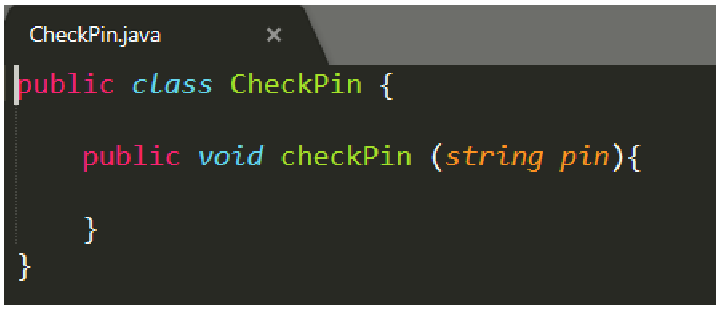
Api	13/05/2020 10:57	Cartella di file
Custom classes	13/05/2020 10:57	Cartella di file
Test	13/05/2020 10:57	Cartella di file

Figura 16: Directories generated by NaturalAPI Develop



CheckPin	13/05/2020 10:57	File JAVA	1 KB
DisplayMessage	13/05/2020 10:57	File JAVA	1 KB
EnterAmount	13/05/2020 10:57	File JAVA	1 KB
EnterPin	13/05/2020 10:57	File JAVA	1 KB
GiveCash	13/05/2020 10:57	File JAVA	1 KB
InsertCard	13/05/2020 10:57	File JAVA	1 KB
IsValid	13/05/2020 10:57	File JAVA	1 KB

Figura 17: API generated by NaturalAPI Develop using Java language



CheckPin.java

```
public class CheckPin {

    public void checkPin (string pin){

    }
}
```

Figura 18: API CheckPin

A Glossary

A.1 A

A.1.1 Action

An operation contained in a BAL suggestion. It can be translated, for example, as a function or as a method of a class.

A.1.2 API

Application Programming Interface - A set of procedures and functions that perform a given task and whose specifications form an interface.

A.2 B

A.2.1 BAL

See Business Application Language.

A.2.2 Business Application Language

Pseudolanguage that interposes itself between the natural language and the programming language. It allows the understanding of actions and objects necessary for APIs even for non-projectors.

A.2.3 BD

See Business Domain.

A.2.4 Business Domain

Aspects of the real world in which the software operates.

A.2.5 BDL

See Business Domain Language.

A.2.6 Business Domain Language

A set of domain-specific terms, consisting of nouns, verbs and predicates and their frequency of use.

A.3 F

A.3.1 Feature

Desired functionality of a product that often includes various behaviors that the software assumes. Behaviors are defined through scenarios.

A.4 G

A.4.1 Gherkin

Standard and widely used format for writing use case scenarios.

A.4.2 Graphic User Interface

Graphic user interface, is a system of interactive visual components for software.

A.4.3 GUI

See Graphic User Interface

A.5 J

A.5.1 Json

Acronym for JavaScript Object Notation, it is a format suitable for exchanging data between client/server applications. It is based on the standard JavaScript language but is independent of it.

A.6 N

A.6.1 NaturalAPI Design

NaturalAPI module that transforms features and scenarios into a BAL.

A.6.2 NaturalAPI Develop

NaturalAPI module that translates BAL into API in the chosen programming language.

A.6.3 NaturalAPI Discover

Modulo di NaturalAPI che estrae il BDL a partire da documenti relativi al dominio.

A.7 P

A.7.1 PLA

See Programming Language Adapter.

A.7.2 Programming Language Adapter

Set of rules that allow the Developer to generate APIs in a specific programming language.

A.8 S

A.8.1 Scenario

Specifies actions and steps that define the behavior of a software using a formal language.