DESIGN DOCUMENT

- ## Domain Class Model

The following model represents the mapping of the collection of *classes* that better describe our project, for an external viewer. Analyzing the whole structure, some elements have been selected to explain what is behind our *NONSENSE-generator*.

Below here there's a detailed description of every entity you will find in the model at the end of the page:

- **USER:** the main *protagonist*, this entity is purely demonstrative and is used to make important connections between what you, user, can do using our program

- **NONSENSEGenerator:** is the name of our project, the structure that connects the user with everything our application can offer and what the user can decide to do

- **CONFIGURATION:** this object is used to maintain the traces of what the user prefers as configuration for the use of the application, along with the default ones

- **Analyze:** the first main use of the application is to analyze whatever the user wants to type as input, from seeing the structure of the sentence to its syntactic tree

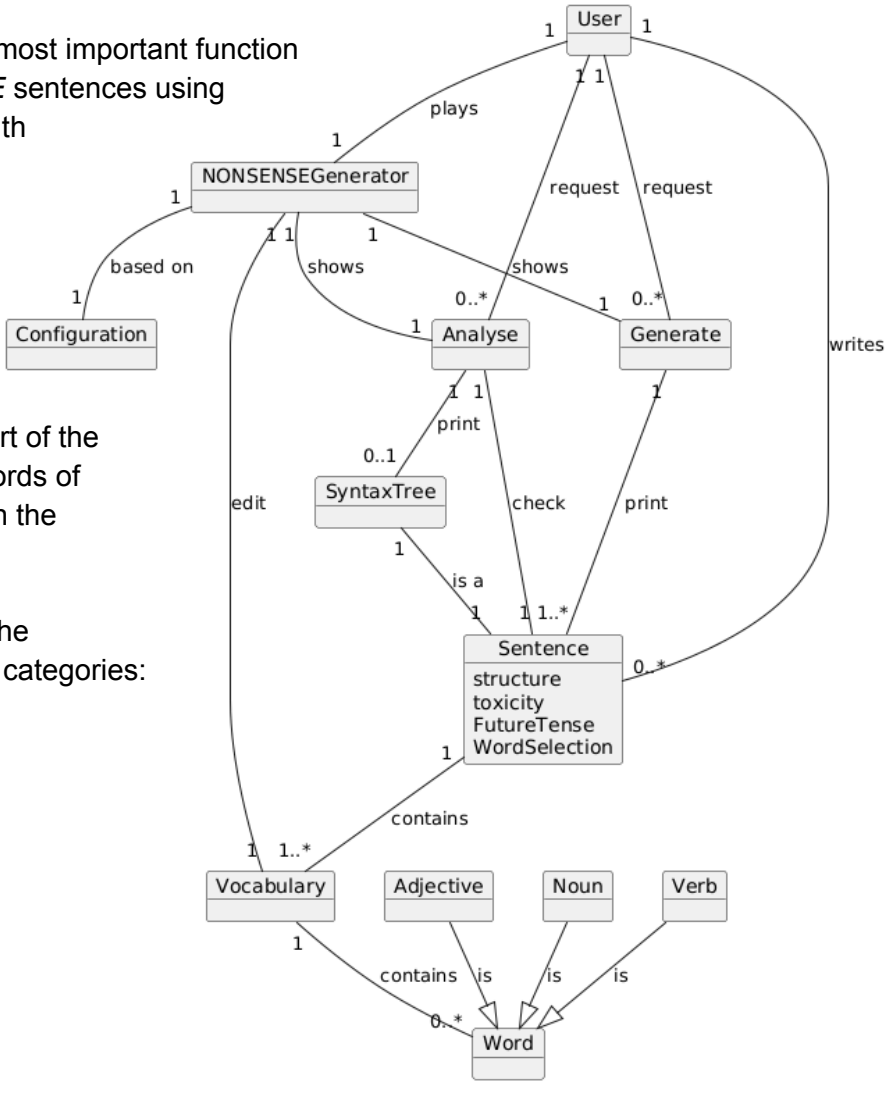
- **SyntaxTree:** this object is used to provide to the user a graphic representation of the syntactic tree of the sentence analyzed

- **Generate:** the second and most important function is to create the *NONSENSE* sentences using inputs from the user and all th configurations he setted

- **Sentence:** the object most used in the program from the input of the user to all the results

- **Vocabulary:** a particular part of the program that contains all words of different kind, used to enrich the generations

- **Word:** the object stored in the vocabulary, divided in three categories: *Adjective*, *Noun*, *Verb*



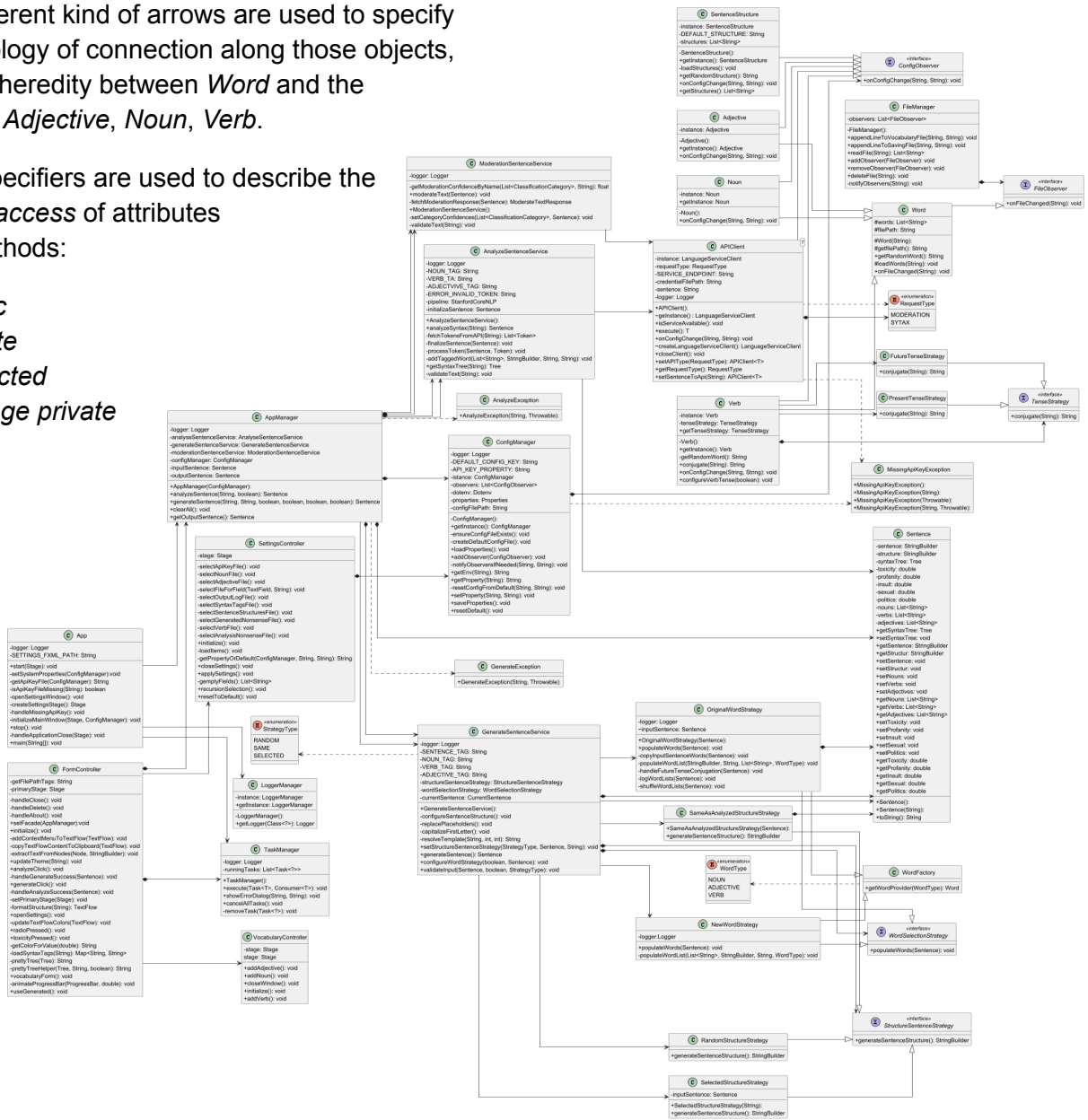
- **Class Model**

The next page offers a detailed structure of the entire project, every entity. Each block in the diagram represents a *Class*, *Interface*, *Enumeration* and more, along with every attribute and methods inside of them.

The different kind of arrows are used to specify the typology of connection along those objects, like the heredity between *Word* and the classes *Adjective*, *Noun*, *Verb*.

More specifiers are used to describe the type of *access* of attributes and methods:

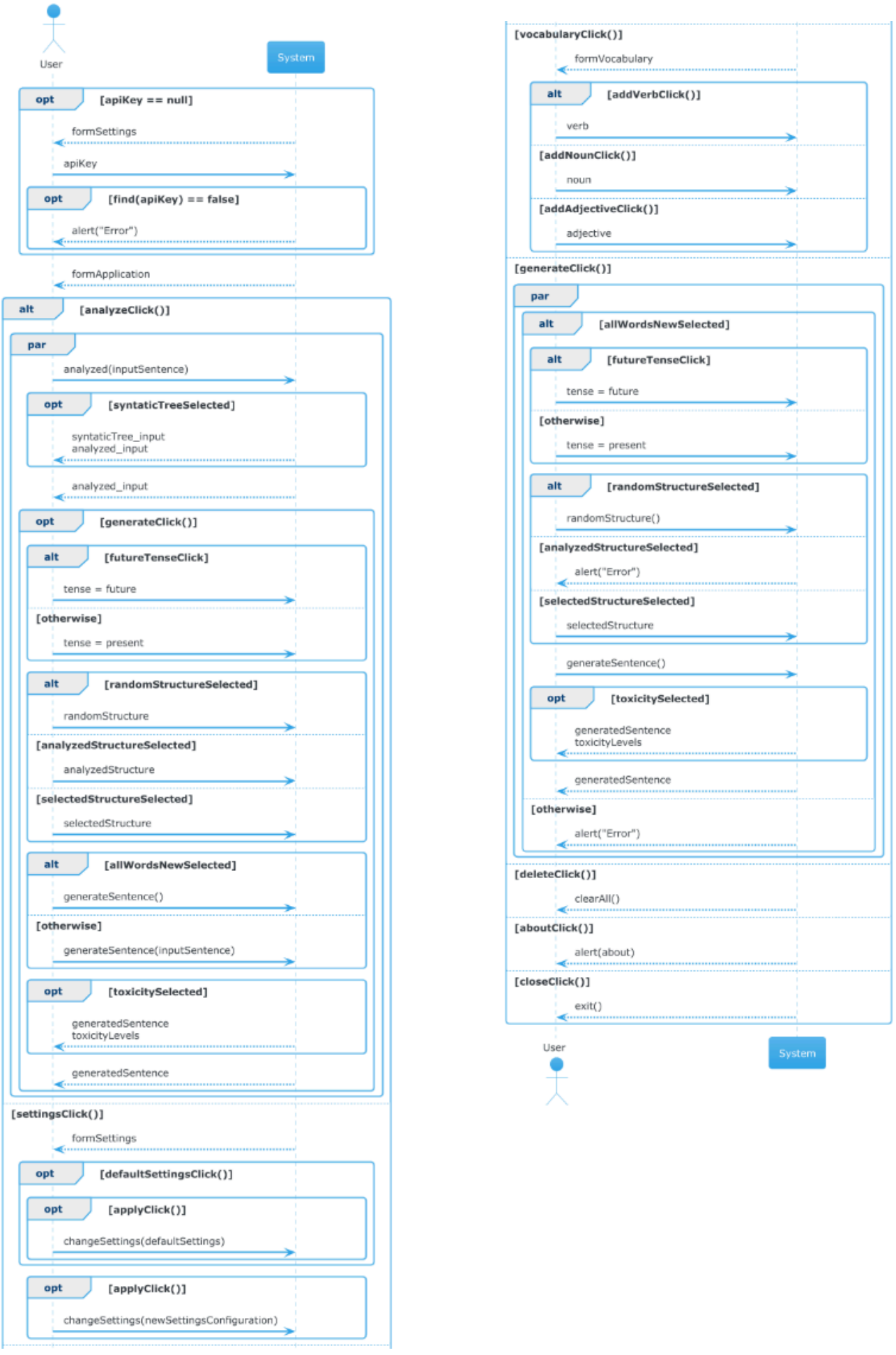
+ *public*
- *private*
*protected*
~ *package private*



To see only the diagram, check on our github at this [link/github]

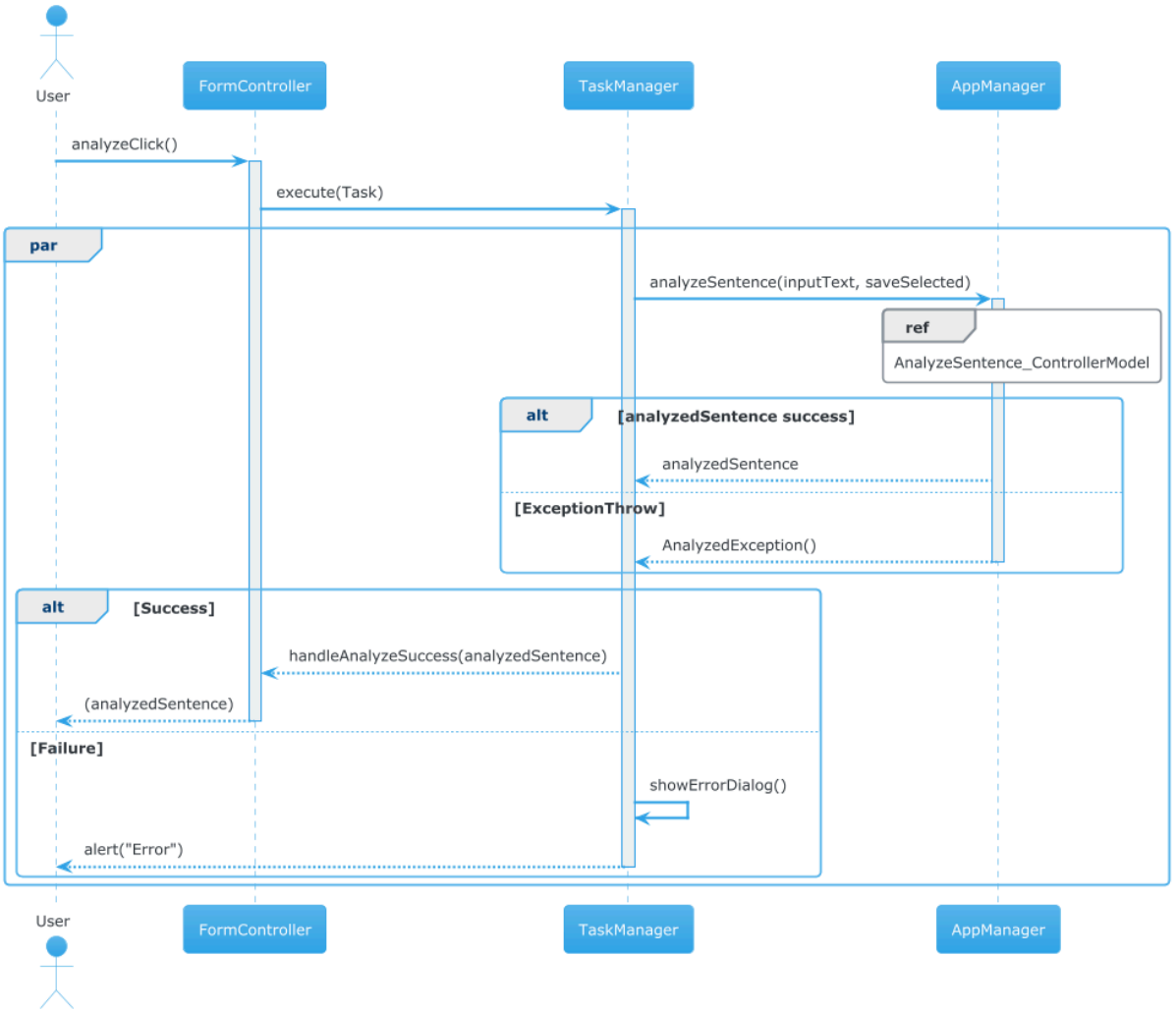
Once opened, search for **documentation → graph → ClassModel.pdf**

- **System Sequence Diagram**

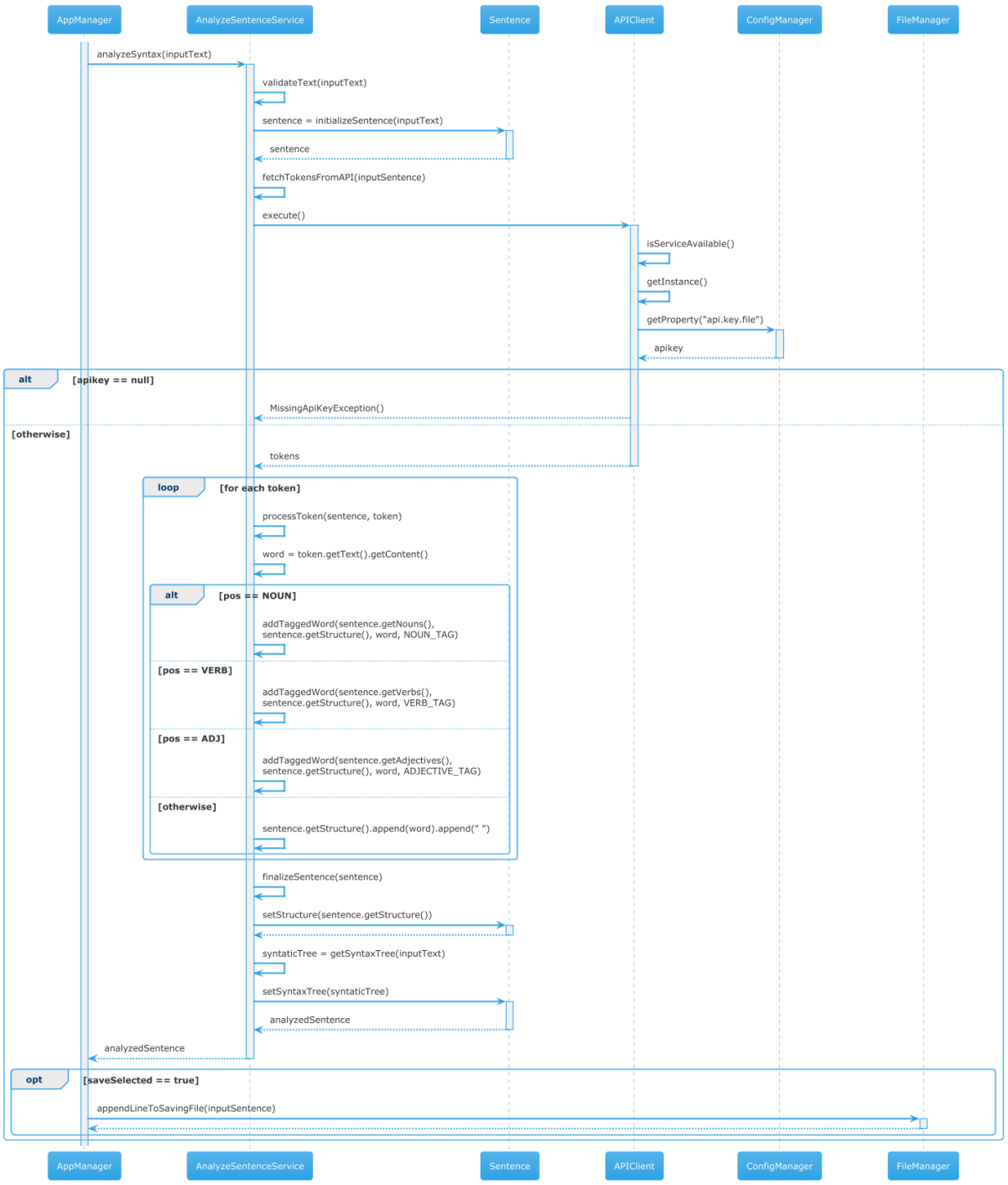


- **Internal Sequence Diagram**

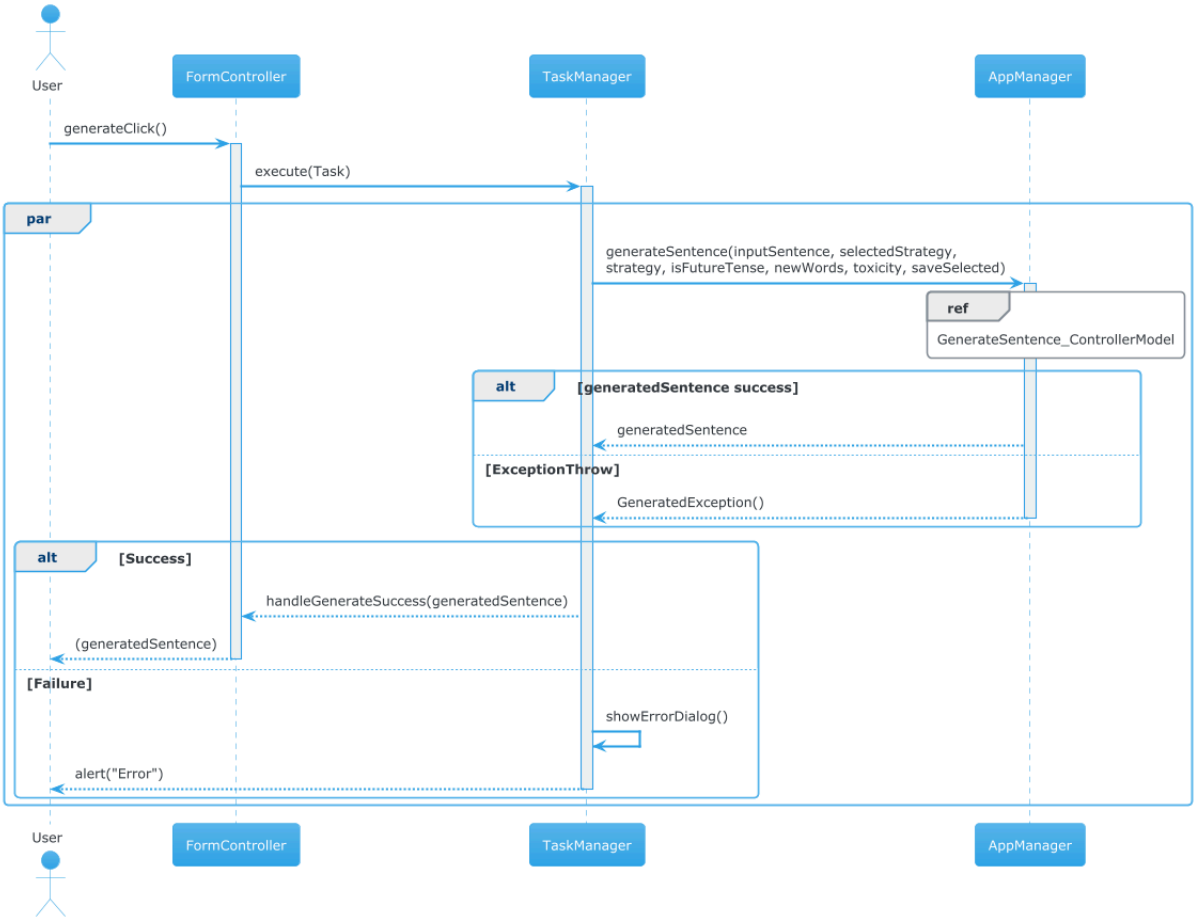
analyzeSentence() (view/controller)



analyzeSentence() (controller/model)



generateSentence() (view/controller)



generateSentence() (controller/model)