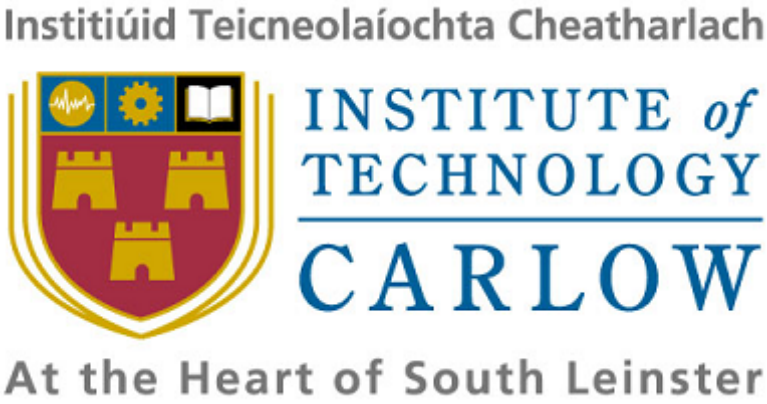


Design Manual:

Code Editing in the Cloud

Author:

Alejandro Borrego Delgado (C00132731)

Final Project,

4th Bach (Honours) in Software Engineering

IT Carlow, 2009/2010

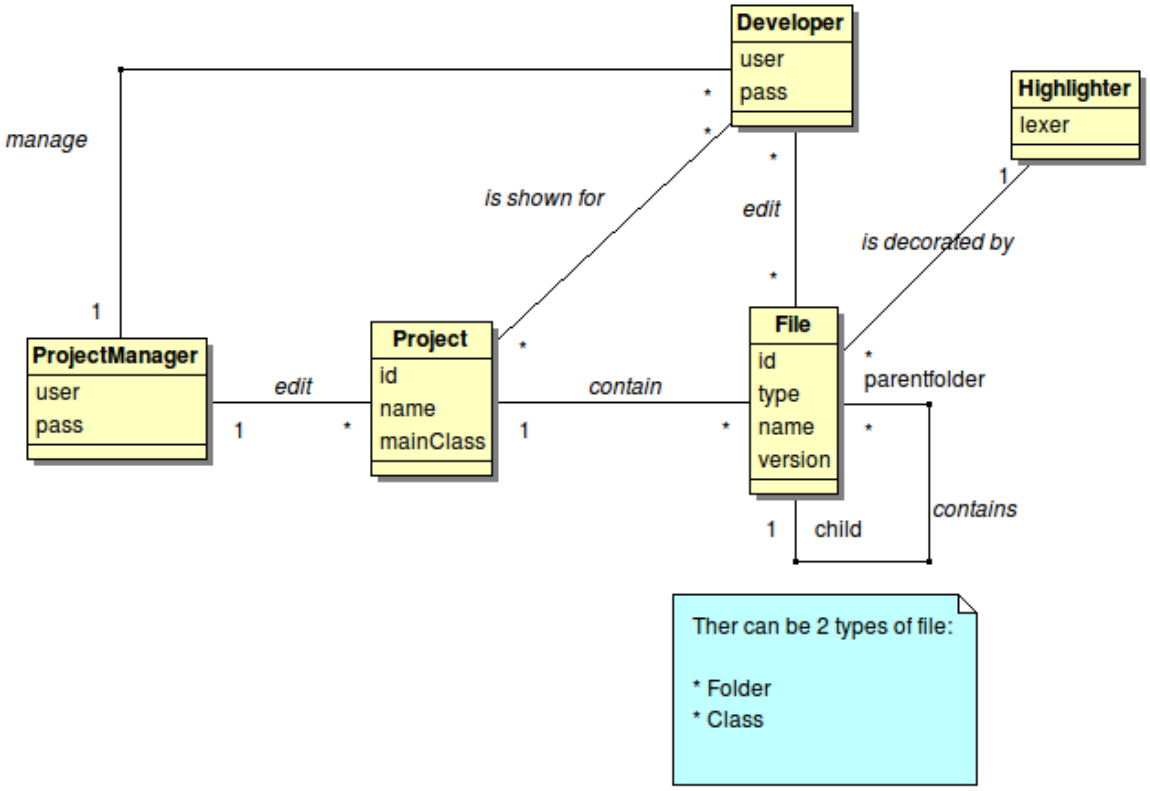
INDEX:

1. Introduction:

This is the final design for our application. This version on the design have evolved from the first version, by completing it with new functionalities and improving already designed parts to end with a better design for our application. The process is simple, iteration after iteration with new implementation, tests, etc. the design have been shown that could be improved and a change in the design was necessary to keep the approach used as agile as possible.

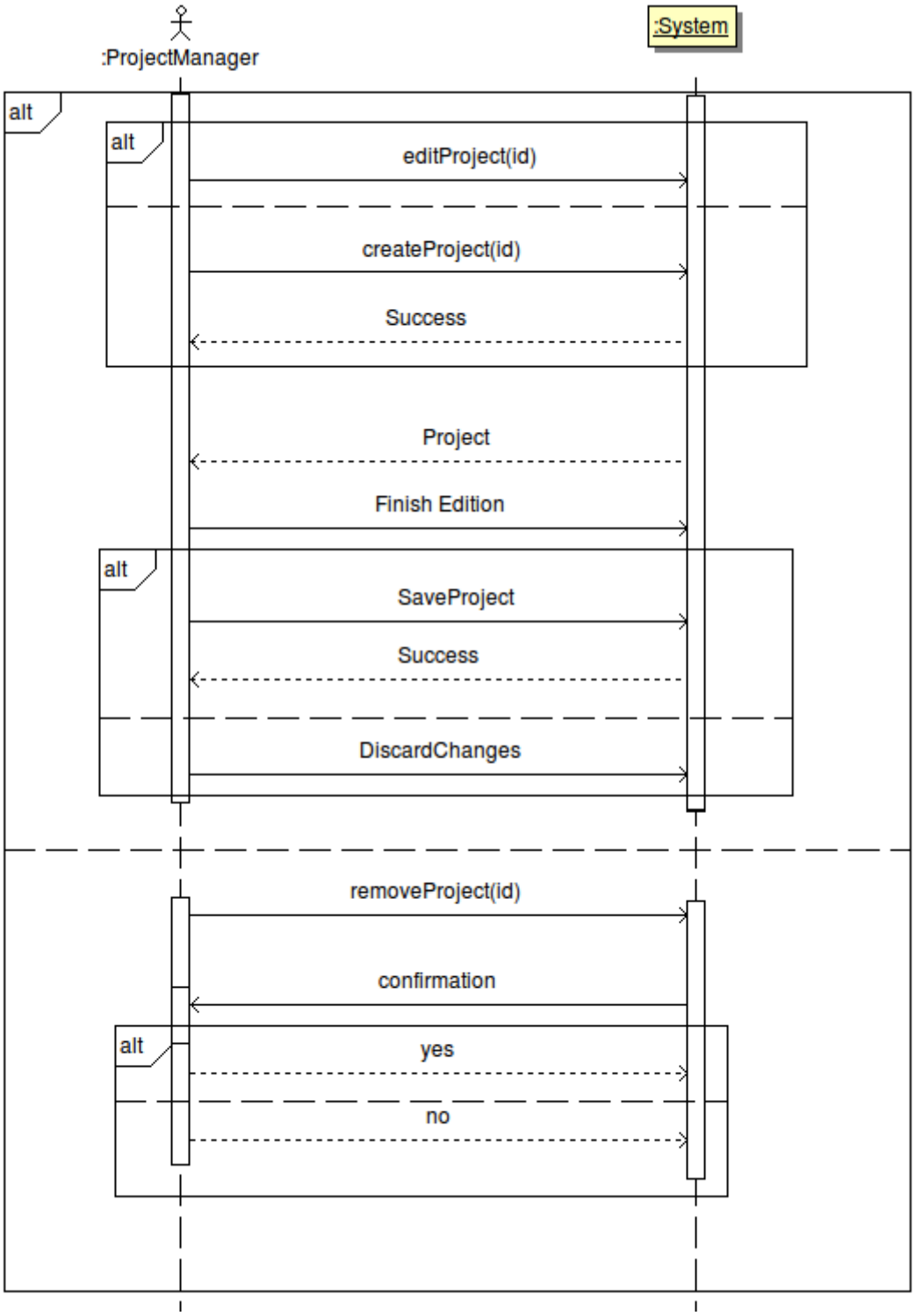
2. Domain Model:

Seeing the diagram, we can understand that there is 3 basic operations. Manage users, manage files, and manage projects.

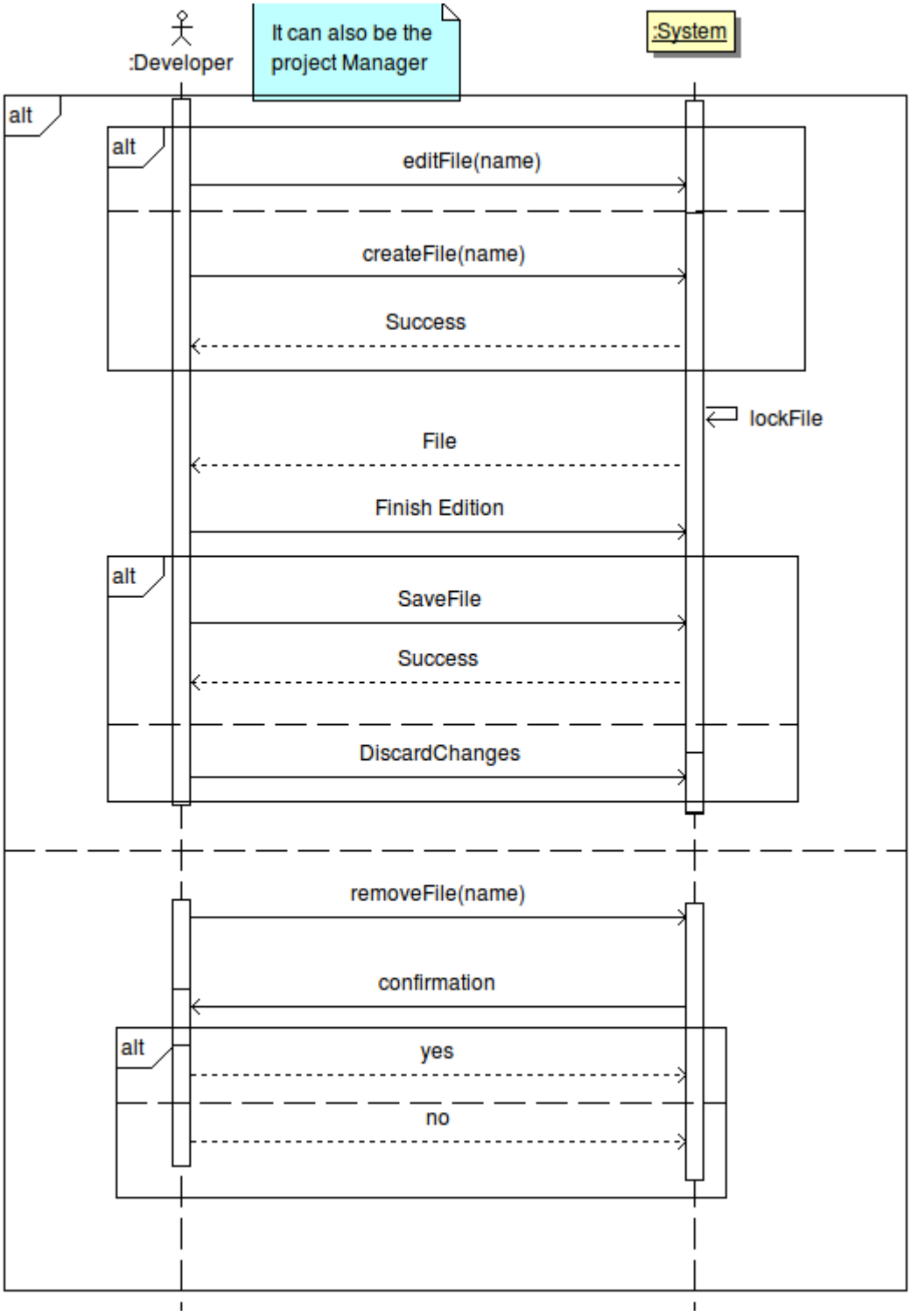


3. System Sequence Diagrams

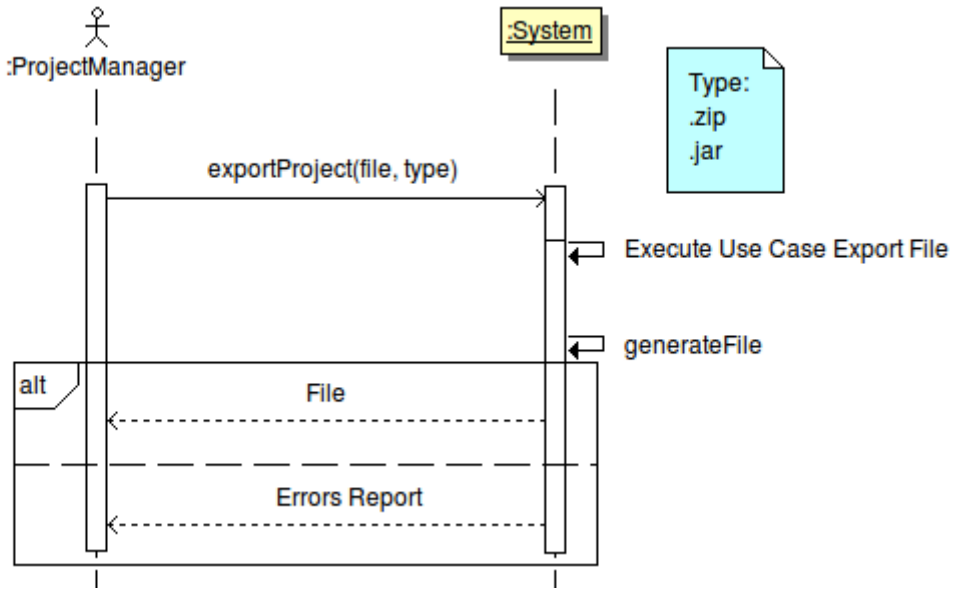
3.1. CRUD Project



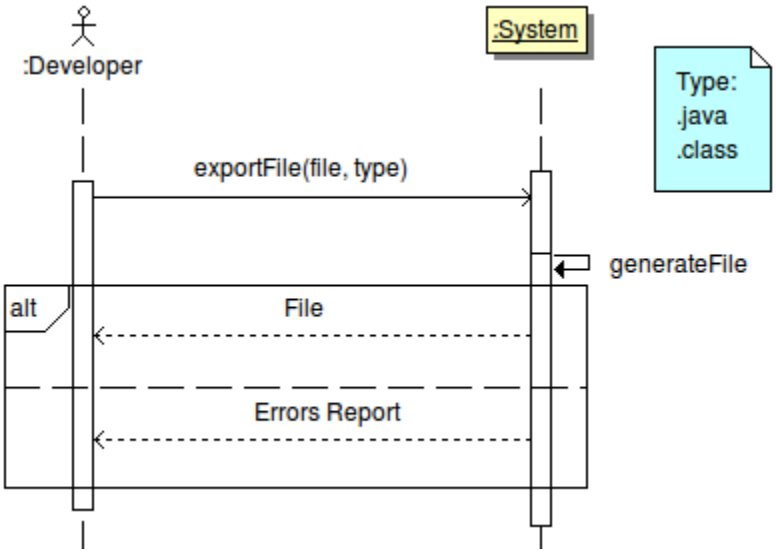
3.2. CRUD File


:Developer
It can also be the project Manager
:System
alt
alt
editFile(name)
createFile(name)
Success
lockFile
File
Finish Edition
alt
SaveFile
Success
DiscardChanges
removeFile(name)
confirmation
alt
yes
no

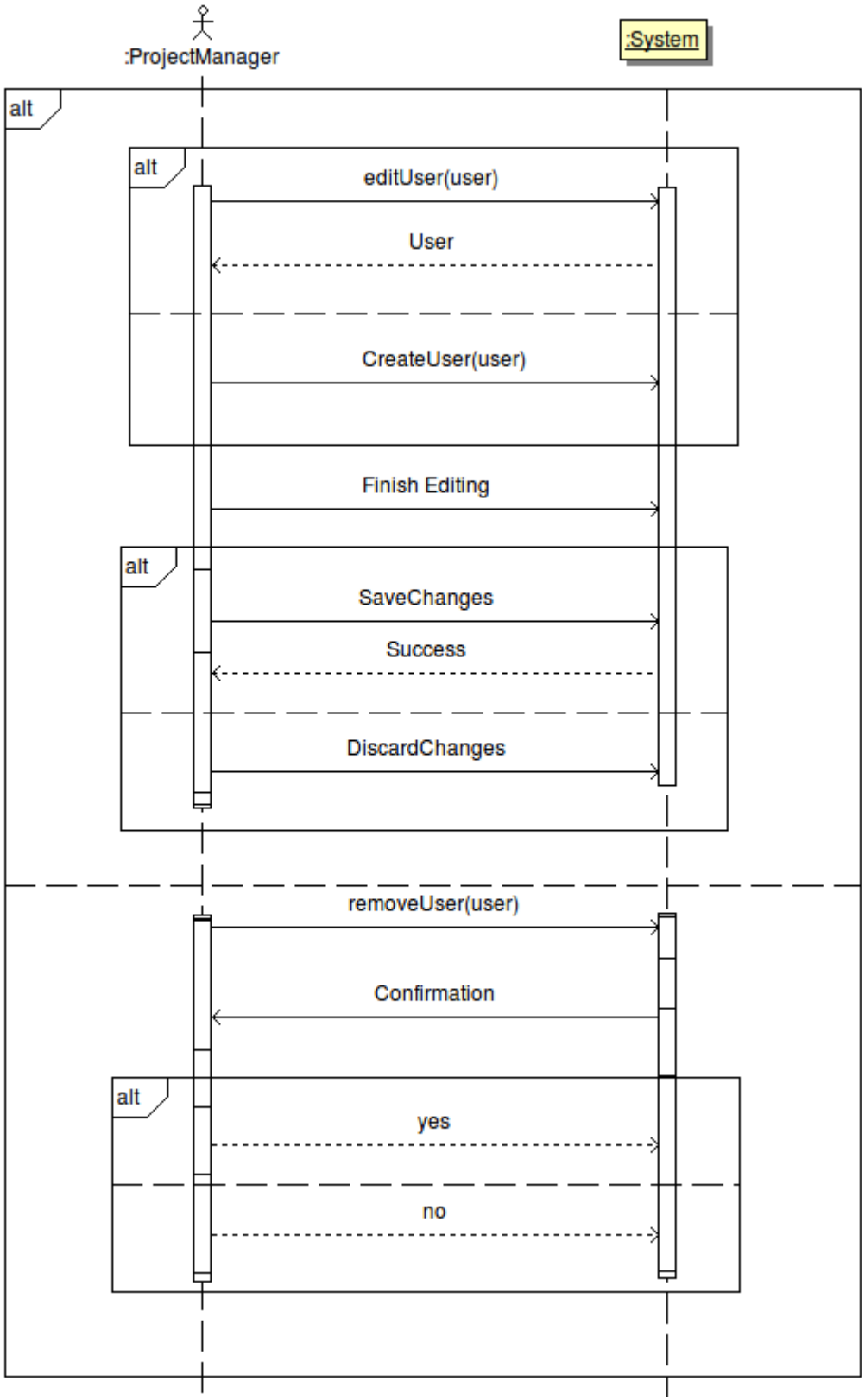
3.3. Export Project



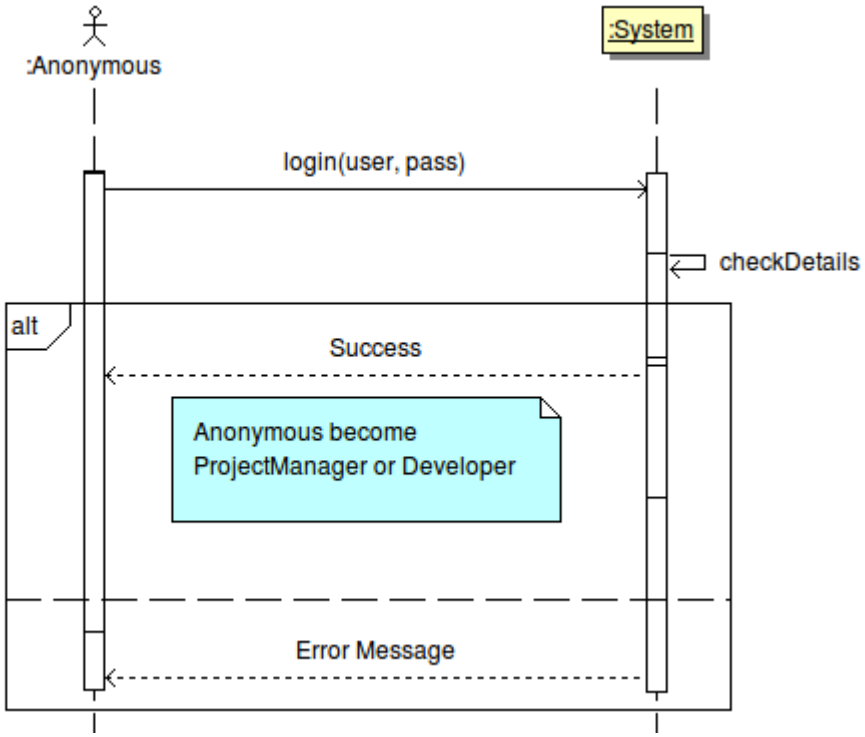
3.4. Export File



3.5. Manage users



3.6. Log-in


:Anonymous
:System
login(user, pass)
checkDetails
alt
Success
Anonymous become
ProjectManager or Developer
Error Message

4. Lexer:

To implement the code highlighter, we need a lexer.

A lexer is an automaton that recognise a stream and divide it in tokens according to their type.

The automaton is a very big one, so in this documents it have been divide in parts, generating the automatons from regular expressions with the “NFA - DFA Generator” a program implemented for another subject in this course but that have been improved so it will fits the specifications to generate the automaton for this project.

The Regular expression is:

/ (~[*]|*~[/])* */ |*

//~[\n]\n*

abstract |

boolean |

break |

byte |

case |

catch |

char |

class |

const |

continue |

default |

do |

double |

else |

extends |

final |

finally |

float |

for |

goto |

if |

implements |

import |

instanceof |

int |

interface |

long |

native |

new |

package |

private |

protected |

public |

return |

short |

static |

strictfp |

super |

switch |

synchronized |

this |

throw |

throws |

transient|

try |

void |

volatile |

while |

wesureal |

true |

false |

null |

*[1-9][0-9]** |

*0x[0-9A-Fa-f]** |

*0[0-7]** |

[1-9][0-9].[0-9]** |

; |

. |

, |

\(|

\) |

{ |

} |

\[|

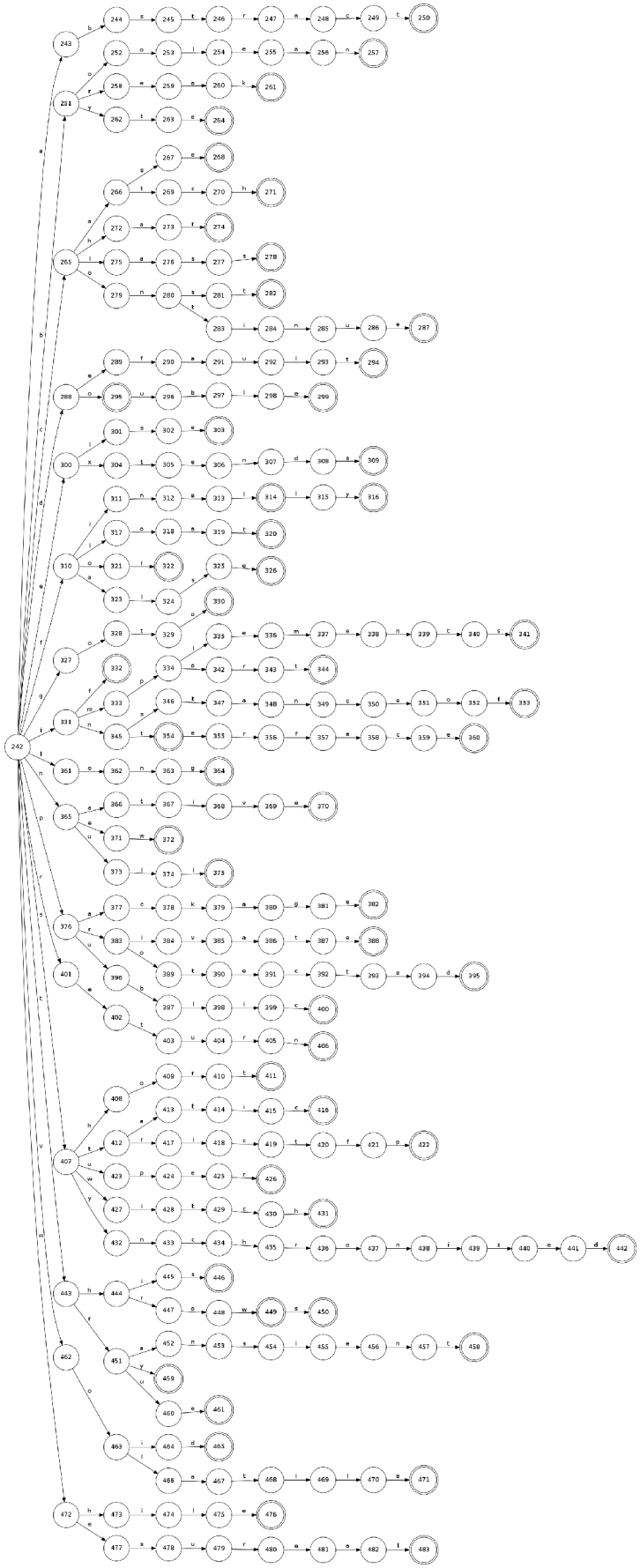
\] |

"(~["]|\\")"* |

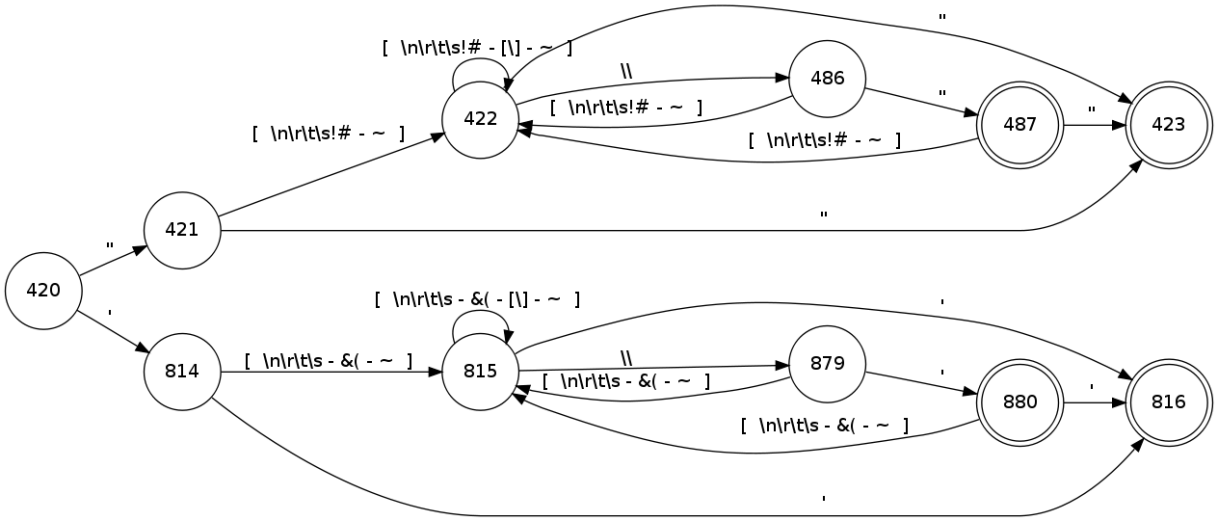
'(~[']|\\')'*

The Automatons generated are:

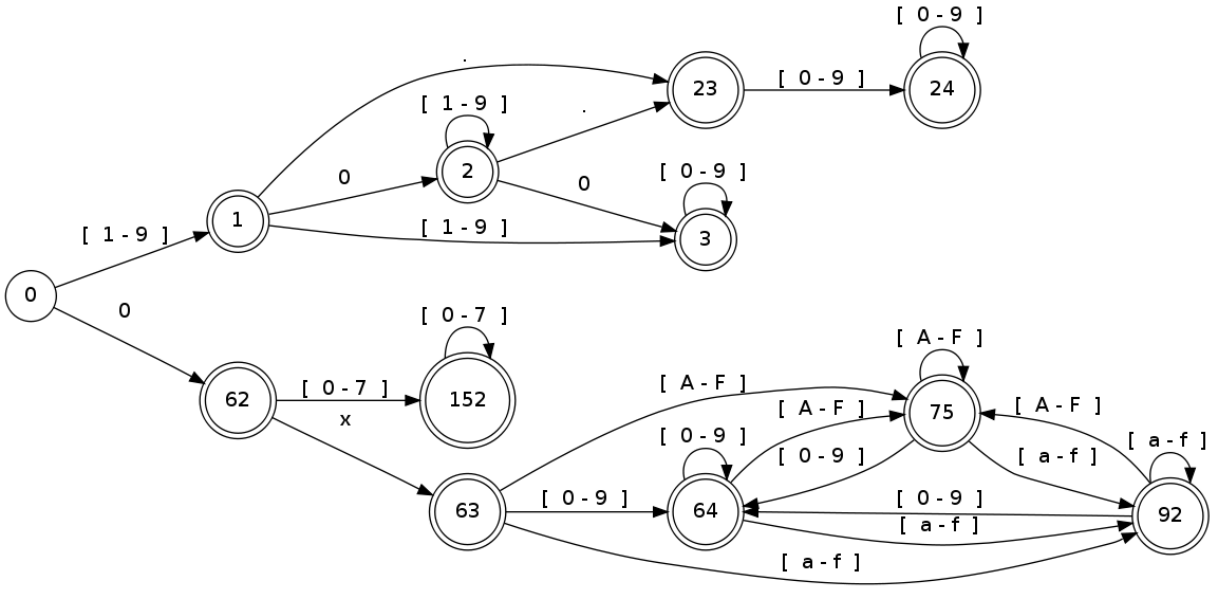
To recognise reserve words:



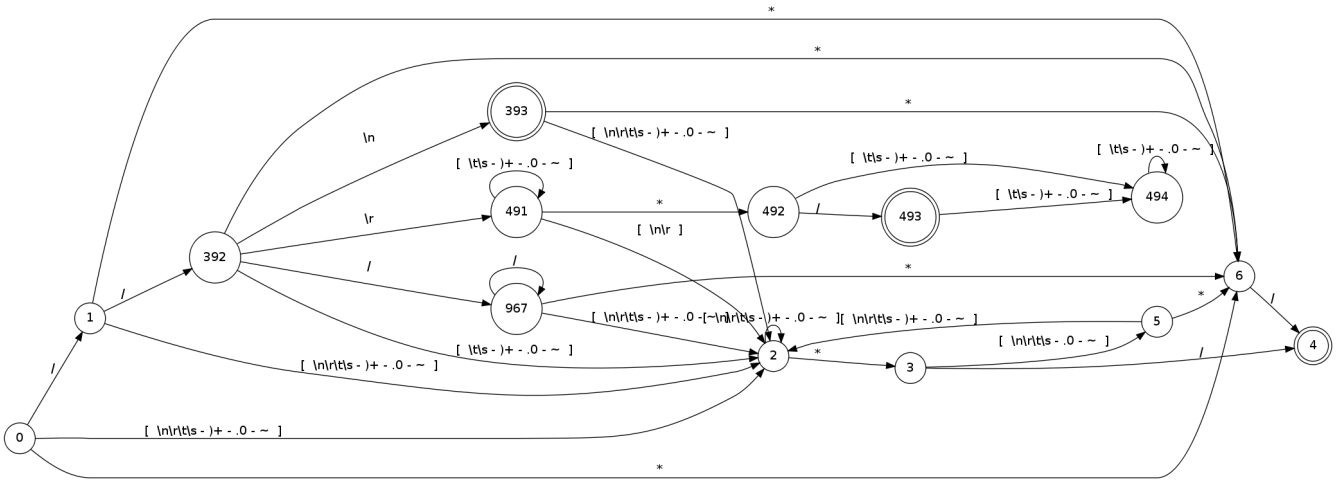
To recognise strings:



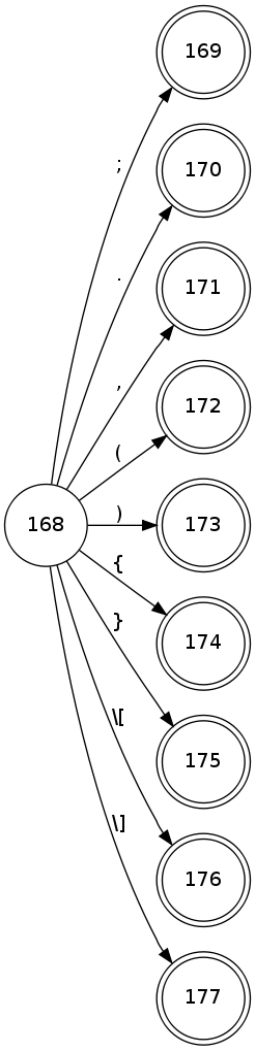
To recognise Numbers:



To recognise Comments:



Other important symbols:

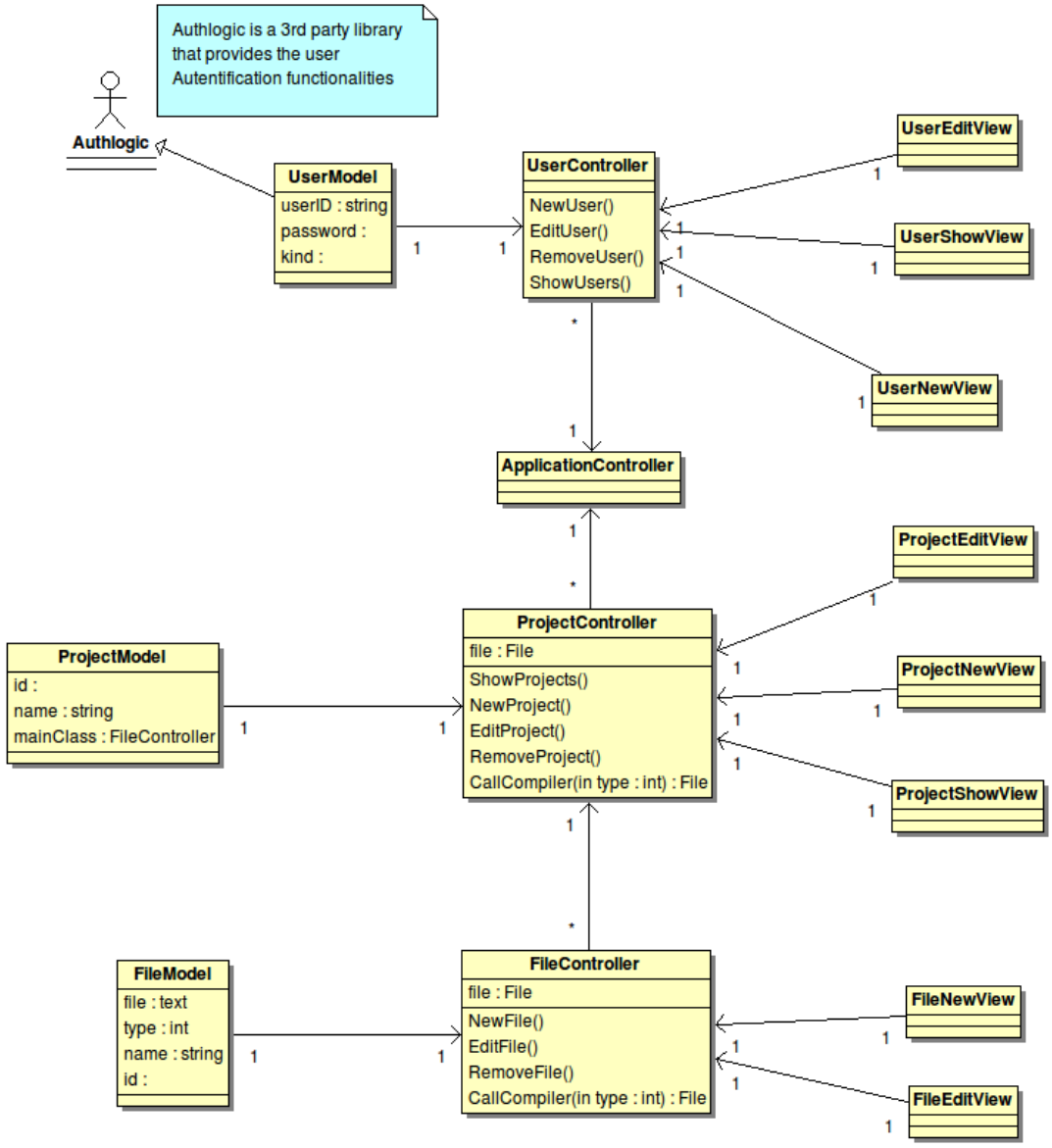


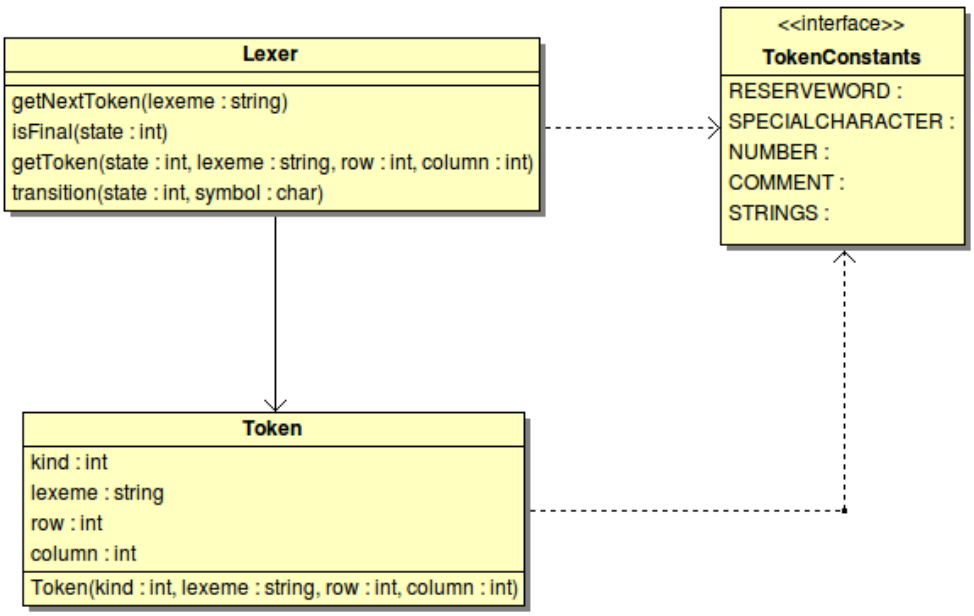
5. Class Diagram:

This is the class diagram for the main application. Evolved from a basic class diagram this version shows a design class diagram for the full application. Functionalities from a 3rd party library (Authlogic) **[*]** is being used, for more information about the Authlogic library please refer to the final report. It has also been provided a class diagram for the Lexer functionalities, which will be called with in the implementation. See reference for further documentation.

The project is implemented under Ruby On Rails technology. Ruby on rails follows the MVC (Model View Controller) structure. As built-in functionalities for managing the Model and Views are already on Ruby On Rails, no further implementation is needed.

[*] http://rdoc.info/projects/binarylogic/authlogic, Authlogic Documentation page




Lexer
getNextToken(lexeme : string)
isFinal(state : int)
getToken(state : int, lexeme : string, row : int, column : int)
transition(state : int, symbol : char)
<<interface>>
TokenConstants
RESERVEWORD :
SPECIALCHARACTER :
NUMBER :
COMMENT :
STRINGS :
Token
kind : int
lexeme : string
row : int
column : int
Token(kind : int, lexeme : string, row : int, column : int)