**PROGETTO SE GRUPPO 14:**

**Membri:** Moccaldi Simone Pio, Nicolo Emanuele Alfonso, Ommeniello Giovanni, Pappalardo Francesco **TOOLS**

*Programming Language:* ***Java****.*

*IDE:* ***Visual studio code****.*

*VCS:* ***Git****.*

*Online Repository:* ***GitHub****.*

*Framework unit testing:* ***JUnit****.*

*Visual Editor:* ***SceneBuilder****.*

*Online TaskBoard:* ***Trello.***

**Online** **TaskBoard**: (<https://trello.com/invite/b/Z2q0p9Nk/ATTI6f9fb7b153d83f1da28ceb9fb1d534ed3065B418/gruppo14-iz>)

**Repository:** (<https://github.com/emanuele-02/progettoSe.git>)

***DEFINITION OF DONE***

A user story is done when it has been implemented and has passed all tests, but not necessarily integrated.

***USER STORY***

1. As a customer I want the application must be similar to IFTTT, but it must be designed to run in stand-alone, single-user desktop mode.
2. As user, I want to be able to create rules to create trigger and associated actions.
3. As a customer I want the application to periodically check the rules to see if the trigger can be activated and if so, execute the specified action.
4. As a user, I want to set a specific time to trigger a rule that becomes true when the time is equal to or greater than the specified time, having the ability to decide whether to trigger an audio file or a message in a dialog box, that I can close.
5. As user, I want to be able to delete rules.
6. As user, I want to be able to update rules to change the trigger or the associated action.
7. As user, I want to be able to deactivate and activate a rule.
8. As customer, I want the set of rules to be saved to a file and automatically reloaded when the application is restarted.
9. As user, I want to be able to define whether a rule should be triggered only once after the trigger or whether it can be triggered immediately after or after a specified period of time.
10. As user, I want to do some operations on files (writing a specified string at the end of a specified text file, copying or moving specified file from a specified directory to a specified destination directory, deleting a specified file from a specified directory).
11. As user, I want to execute a specified external program with specified command-line arguments.
12. As a user, I want a 'current day-of-week' trigger that fires when the current day of the week is equal to the one specified (Name day), the 'current day-of-month' trigger that fires when the current day of the month is equal to the one specified (Number day), the 'current date' trigger that fires when the current date is equal to the one specified (format dd/mm/yyyy).
13. As user, I want the following triggers on files: a file with a specified name exists in a specified directory; the size of a specified file is larger than a specified value.
14. As a user, I want a trigger that activates if the exit value of an executed external programme is equal to a specified value.
15. A user, I want to be able to define a set of elementary actions or other sequences of actions as a result of a rule.
16. As user, I want to be able to use logical operators to combine elementary triggers or combined triggers.
17. As user, I want to be able to create, view and edit integers counters.
18. As a user I want to be able to perform the following operations on the counters: set, add a value to this counter, and add the value of one counter to another.
19. As a user, I want to be able to create triggers on counters so that I can use comparison operations with each other or with a specified integer.
20. As a user I want, when using strings, to be able to use a replacement variable to insert the current value of a counter into the string.
21. As customer, I want a graphical user interface.

**USER STORY NAME**

1. Stand-Alone Desktop App like IFTTT.
2. Rule Creation for Trigger-Action. V=21, S=8
3. Periodic Rule Checking and Execution. V=21, S=13
4. Time-Based Rule Trigger with Audio or Dialog Action. V=21, S=13
5. Rule Deletion Capability. V=21, S=1
6. Rule Update for Trigger and Action Modification. V=21, S=5
7. Rule Activation and Deactivation. V=21, S=2
8. Save and Auto-Load Rules on Application Restart. V=21, S=21
9. Rule Trigger Options: Once, Immediately, or After a Set Time. V=13, S=5
10. File Operations: Write, Copy, Move, Delete. V=13, S=8
11. Execute External Program with Command-line Arguments. V=13, S=8
12. Date and Time-Based Triggers. V=8, S=8
13. File Triggers: Existence and Size. V=8, S=8
14. Exit Value External Program Trigger. V=8, S=5
15. Define Sequences of Actions for Rules. V=5, S=21
16. Logical Operator Support for Combined Triggers. V=5, S=3
17. Integer Counter Creation and Management. V=3, S=1
18. Counter Operations: Set, Add, Add to Another Counter. V=2, S=2
19. Triggers on Counters for Comparison Operations. V=2, S=5
20. Use of Replacement Variables to Insert Counter Values into Strings. V=1, S=3
21. Graphical User Interface Implementation. S=21

***ACCEPTANCE CRITERIA***

**Given**: A user in the system.

**When**: The user creates a rule with a trigger and one or more associated actions.

**Then**:

The rule is automatically added to the rule set.

The rule set contains the newly created rule.

**Given**: The application is running.

**When**: The system performs periodic rule checking every 5 seconds.

**Then**: If a trigger can be activated, the specified action(s) associated with the rule are executed.

**Given**: A user in the system.

**When**:

The user sets a specific time for a rule to trigger.

The user specifies whether the trigger should play an audio file (MP3) or show a dialog box with a specific message.

The user provides the path to the audio file.

**Then**:

When the specified time is reached, the rule becomes true.

If the trigger is set to play an audio file, the system plays the specified audio file.

If the trigger is set to show a dialog box, the system displays the dialog box with the specified message, and the user can close it.

**Given**: A user with existing rules.

**When**: The user attempts to delete a rule.

**Then**:

The system prompts the user for confirmation before the final deletion.

If confirmed, the rule is removed from the set of rules.

**Given**: A user with existing rules.

**When**: The user creates a rule.

**Then**:

The newly created rule is activated by default.

The user can choose to deactivate or activate a rule.

**Given**: A user with existing rules.

**When**: The user edits a rule, changing its attributes, trigger, and/or action(s).

**Then**:

Upon successful update, the system provides immediate feedback.

The rule in the rule set reflects the updated attributes, trigger, and actions.

**Given**: A set of rules in the system.

**When**:

The customer requests to save the set of rules to a file.

The application is restarted.

**Then**: The set of rules is automatically reloaded from the saved file.

**Given**: A user creating a rule.

**When**: The user defines whether a rule should be triggered:

Only once after the trigger.

Immediately after the trigger.

After a specified period of time.

**Then**: The rule behaves according to the specified triggering condition.

**Given**: A user interacting with the system.

**When**: The user performs file operations, including:

Writing a specified string at the end of a specified text file.

Copying or moving a specified file from a specified directory to another specified destination directory.

Deleting a specified file from a specified directory.

**Then**: The specified file operations are executed successfully.

**Given**: A user creating a rule.

**When**: The user specifies an external program and command-line arguments to execute.

**Then**: The specified external program is executed with the specified command-line arguments.

**Given**: A user creating a rule.

**When**: The user specifies triggers:

'Current day-of-week' trigger for a specific day.

'Current day-of-month' trigger for a specific day.

'Current date' trigger for a specific date (format dd/mm/yyyy).

**Then**: The triggers activate when the current day, day of the month, or date matches the specified values.

**Given**: A user creating a rule.

**When**: The user defines triggers related to files, including:

A file with a specified name exists in a specified directory.

The size of a specified file is larger than a specified value.

**Then**: The triggers activate based on the specified file conditions.

**Given**: A user creating a rule.

**When**: The user specifies a trigger based on the exit value of an executed external program.

**Then**: The trigger activates if the exit value is equal to the specified value.

**Given**: A user creating a rule.

**When**: The user defines a set of elementary actions or other sequences of actions.

**Then**: The specified actions are executed as a result of the rule.

**Given**: A user creating a rule.

**When**: The user uses logical operators to combine elementary triggers or combined triggers.

**Then**: The rule behaves according to the specified logical conditions.

**Given**: A user interacting with the system.

**When**: The user creates, views, and edits integer counters.

**Then**: The user can perform operations on counters, including setting, adding a value to a counter, and adding the value of one counter to another.

**Given**: A user with existing counters.

**When**: The user performs operations on counters, including:

Setting a counter.

Adding a value to a counter.

Adding the value of one counter to another.

**Then**: The specified operations are performed successfully on the counters.

**Given**: A user creating a rule.

**When**: The user defines triggers on counters, allowing for comparison operations with each other or with a specified integer.

**Then**: The triggers activate based on the specified conditions involving counters.

**Given**: A user creating a rule involving strings.

**When**: The user uses a replacement variable to insert the current value of a counter into the string.

**Then**: The replacement variable is replaced with the current value of the specified counter.

**Given**: A customer using the application.

**When**: The customer interacts with the system.

**Then**: The application provides a graphical user interface (GUI) for easy interaction and usability

***ESTIMATED VELOCITY***

Estimated velocity= TotStoryPoint/Number of sprint=161/3= 54

Estimated velocity with bug/tb/unplanned= 161-(161\*0,20)/3= 129/3 = 43

***SPRINT 1*(TOT STORY POINT 42):**

**2) Rule Creation for Trigger-Action. V=21, S=8.** As a user I want to able to create triggers and actions and rules.  
 A rule consists of a single trigger and one or more actions associated with that trigger.  
When a rule is created it is automatically added to the rule set.

**3) Periodic Rule Checking and Execution. V=21, S=13.**As a customer I want the application to periodically check the rules to see if the trigger can be activated and if so, execute the specified action(s).  
Rule checking is performed every 5 seconds.

**4) Time-Based Rule Trigger with Audio or Dialog Action. V=21, S=13**.  
As a user, I want to set a specific time to trigger a rule that becomes true when the time is equal to or greater than the specified time(hours and minutes), having the ability to decide whether to trigger an audio file(FILE MP3) (specifying the path) or a specific message in a dialog box, that I can close.

**5) Rule Deletion Capability. V=21, S=1.**  
As user, I want to be able to delete a rule from the set of rules. When I attempt to delete a rule, the system prompts me for confirmation before the final deletion.

**7) Rule Activation and Deactivation. V=21, S=2.**As user, I want to be able to deactivate and activate a rule.  
When a rule is created, it’s activated by default.

**6) Rule Update for Trigger and Action Modification. V=21, S=5.**  
As a user I want to be able to edit the rules by changing its attributes, changing its trigger, action(s), or be able to add new actions. Upon successfully updating the trigger or action, I receive immediate feedback from the system.

***SPRINT 1*** **TASK**:(Emanuele=E,Simone=S,Francesco=F,Giovanni=g)

**2) Rule Creation for Trigger-Action.**

-Creation of Rule class with its constructor and the Action interfaces. (E)

-Creation of Trigger interface and TriggerComposite class. (F)

**3) Periodic Rule Checking and Execution.**

-Creation of RuleObserver interface and RuleSubject class. (S)

-Creation of RuleManager class with use of patter singleton and methods relating to add to list, update and execution. (F,E)

**4) Time-Based Rule Trigger with Audio or Dialog Action.**

-Creation of the HourOfDayTrigger class. (G)

-Creation of class DayOfWeekTrigger. (E)

- Creation class DayOfMonthTrigger. (F)

-Creation DateTrigger class. (S)

-Creation AudioAction class. (G)

-Creation DialogBoxAction class. (S)

**5) Rule Deletion Capability.**

-Implementation of removeRule method in RuleManager. (F)

**7) Rule Activation and Deactivation.**

-Add attributes and methods to rule to implement the logic of activating and deactivating a rule. (E)

-Update RuleManager. (G)

**6) Rule Update for Trigger and Action Modification.**

-Implement in Rule the methods: modifyRule,modifyName,modifyTrigger. (F,S)

-Implement in Rule the methods: addAction and removeAction. (G,E)

***USED PATTERNS***

***OBSERVER***

The Observer pattern has been applied to facilitate deacopulation between the Rule subject and the RuleManager observer, as the subject does not depend on any implementation of the observer. This pattern also allows better maintainability of the code, making it possible, for example, to modify the notification logic without directly affecting the observer.

***SINGLETON***

The Singleton pattern has been implemented on the RuleManager class by introducing a static instance attribute and a static getInstance() method. This architectural choice guarantees the existence of a single instance of the RuleManager class throughout the system, providing a global access point to it.

The instance attribute is declared as static to preserve the unique instance of the class, and the getInstance() method returns this instance or creates it if it has not yet been initialised. This approach ensures that all parts of the programme access the same RuleManager instance, facilitating centralised rule management.

The choice to apply the Singleton pattern is particularly advantageous in the context of RuleManager, as this class is responsible for the overall control and management of all rules. The preservation of state between different calls is crucial, especially considering that RuleManager must control the set of rules and manage operations such as saving to a file. The presence of a single instance simplifies state management, avoids duplication of resources, and facilitates control over critical operations, such as saving rules to file.In addition, the centralisation of rule management operations in RuleManager makes the code more orderly and maintainable, allowing future extensions with minimal changes.  
  
The singleton pattern for the same reasons was applied to the MapCounter class, because in our system we only want a MapCounter structure that contains all counters that are a key(name)-value (counter value) pair.

***COMPOSITE***

The Composite pattern was used on the Trigger class to enable the user to create triggers composed of other triggers by means of logical operations (or,not,and). For the realisation of this pattern, the Trigger interface was realised, which contains the common method implemented by all specific triggers (leaves) and the TriggerComposite, wich is the class that allows the union of several triggers into one thanks to the add and remove methods. The use of this pattern favours polymorphism by treating the triggers uniformly without worrying about the specific type, and allows the creation of 'complex' triggers (triggers consisting of two triggers) from other 'complex' triggers

***SOFTWARE ARCHITECTURE***

Immagine che contiene testo, diagramma, Piano, Disegno tecnico

Descrizione generata automaticamente