Documents

Dossier on ISSSR 2017-2018 project

Tiziano Ditoma, Simone Mancini, Luca Menzolini, Andrea Silvi, Alessio Vintari

9 July 2018

Introduction

This document represents the complete documentation produced by Team 2 for ISSSR 2018 course's project. In particular, it has been produced during the project and it has been reorganized and inserted in this dossier in the final stage.

This dossier's aim is to document, in the best possible way, how the team has worked on this project under different aspects: implementation, design and organizational ones.
This document is not only targeted to the presentation of the project, but we hope it could be useful for the future courses.

Dossier is divided into 3 parts:

- **SCRUM Documentation**: the first section explains how the team worked during the project development with a AGILE model (in particular SCRUM);
- **Technical Documentation**: this section concerns an accurate description of the components of the system in a technical way;
- **Design Documentation**: this section concerns the design documentation produced during the project development.

SCRUM Documentation

Dossier on ISSSR 2017-2018 project

Introduction

The purpose of this dossier is to document the implementation of the *SCRUM* approach used to develop the ISSSR 2018 course's project through design choices and activities the team performed during each sprint.

Through graphs and tables, sketched all along the course and organized in the final step, the team wants to show how it worked under implementation aspect, how it organized meetings with the clients and how it decided to developed the project using *AGILE* approach.

The documentation of each Sprint follows carrying particular attention to the assigned tasks, with their complexity and priority, and to the group coordination.

Tools used

For completeness, we firstly show what kink of tools we used to communicate and keeping track of our work progresses:

- **Telegram**: main communication tool used to coordinate implementation and communication aspects and to organize meetings and progresses directions.
- **Google Drive**: main cloud tool used to keep shared relevant documents so that any member of the team could update them.
- **Asana**: desktop and mobile versions; main tool used to keep track of assigned, completed and added tasks in order to have a global view of the project's progresses. Its functionalities helped the team to split the work according to each team member skills.
- **Hangout**: main conference calls tools. Many meetings happened in a telematic way because of different members' commitments;
- **GitHub**: used for code versioning.

Task priority

Priority's range goes from 1 to 5. Depending on this value, we took decisions on transition from Product Backlog to Sprint Backlog.

Effort

Effort's range goes from 1 to 10. The higher the effort is, the higher the task's difficulty should be in terms of working hours. This scale represents an indicative value rather than a specific mapping fin order to image how a task could be complex in comparison to another.

Sprint 1 (24 April - 7 May)

The principal aim of the first Sprint of the project development was requirements elicitation. We tried to define the problem and understand stakeholders' needs through a brainstorming meeting with clients.
Meeting took place at the presence of all the teams in the classroom.

At the end of the meeting, teams kept in touch each other in order to obtain as much necessary information as possible to understand the problem in the next meeting with the stakeholders.

Despite this could not be considered a classical Sprint, but a preliminary meeting necessary to define the project development progresses, the team realized a very simple Ticket CRUD to begin to try the frameworks (such as Spring and AngularJS) we would have used.

Sprint 2 (8 May - 14 May)

The following Sprint can be considered anomalous concerning the Sprint duration (a week) because of the organizational needs of the teams.
Below the evolution of Product Backlog and Sprint Backlog will be shown.

Tasks:

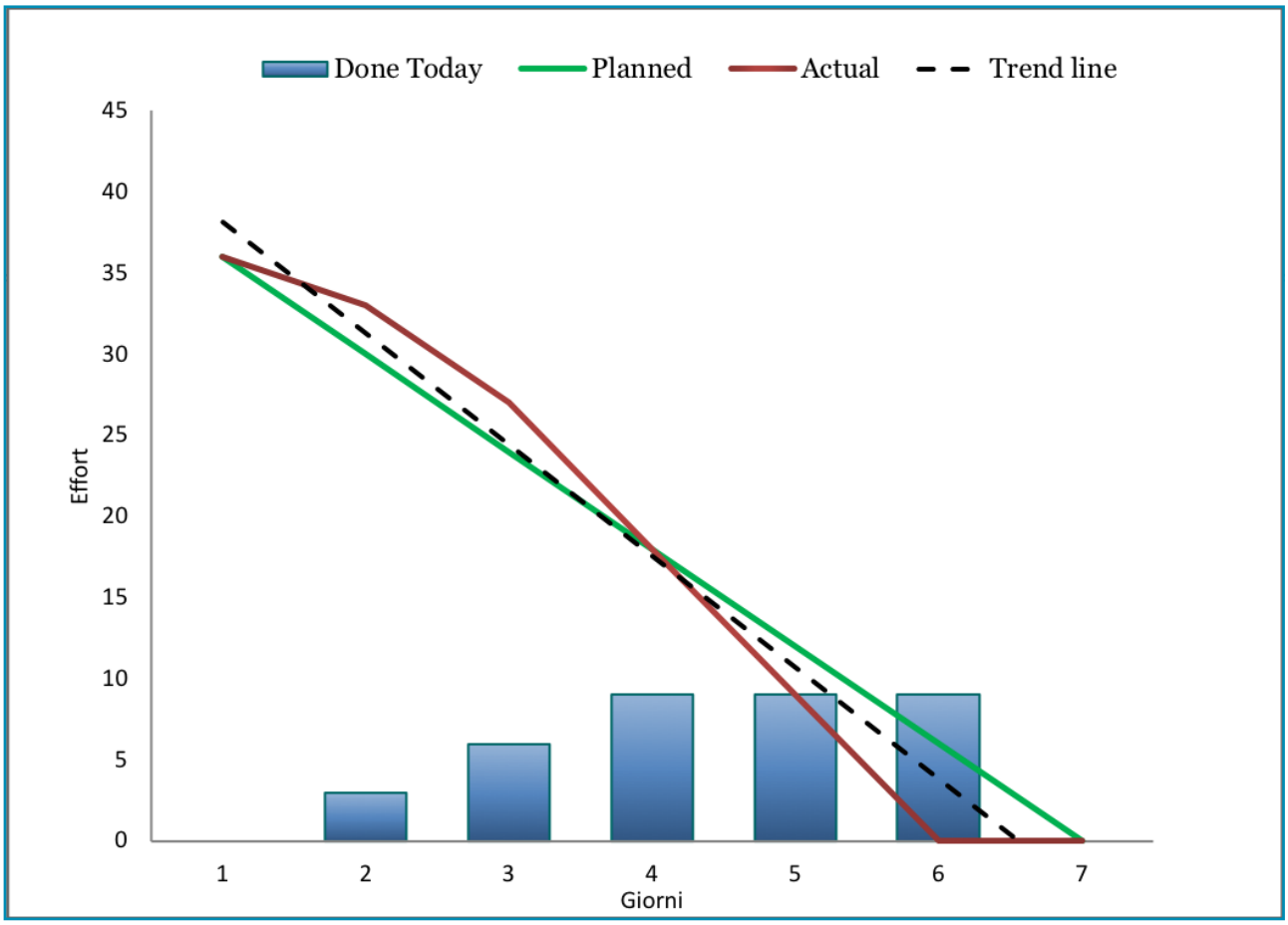
- CRUD Ticket
- CRUD Team
- CRUD Target
- CRUD User

Each CRUD implies the following implementations:

- Front-end
- Back-end
- Testing

Sprint duration					**7 days**
General Tasks	**Specific Tasks**	**Priority**	**Effort**	**Tasks added**	**Tasks completed**
CRUD Ticket	Front-end	5	3		
	Back-end	5	3		
	Test	5	3		
CRUD Team	Front-end	5	3		
	Back-end	5	3		
	Test	5	3		
CRUD Target	Front-end	5	3		
	Back-end	5	3		
	Test	5	3		
CRUD User	Front-end	5	3		
	Back-end	5	3		
	Test	5	3		
			Product Backlog	**Sprint Backlog**	**Increase**
Total Tasks			**12**	**12**	**12**
Total Effort			**36**	**36**	**36**

Burndown Chart



Retrospective Meeting

Meeting results pointed out the following aspects:

- Organizational grade of the team has been satisfying allowing to complete the job
- We should try to complete all the tasks some days before. Work completion anticipation remains a point to evaluate on a 15-days-Sprint.
- Achieved results with Hangout conference calls have been acceptable.
- We must avoid to describe the task too finely. This brings confusion on Asana and make the work division harder.

Sprint 3 (15 May - 28 May)

Tasks:

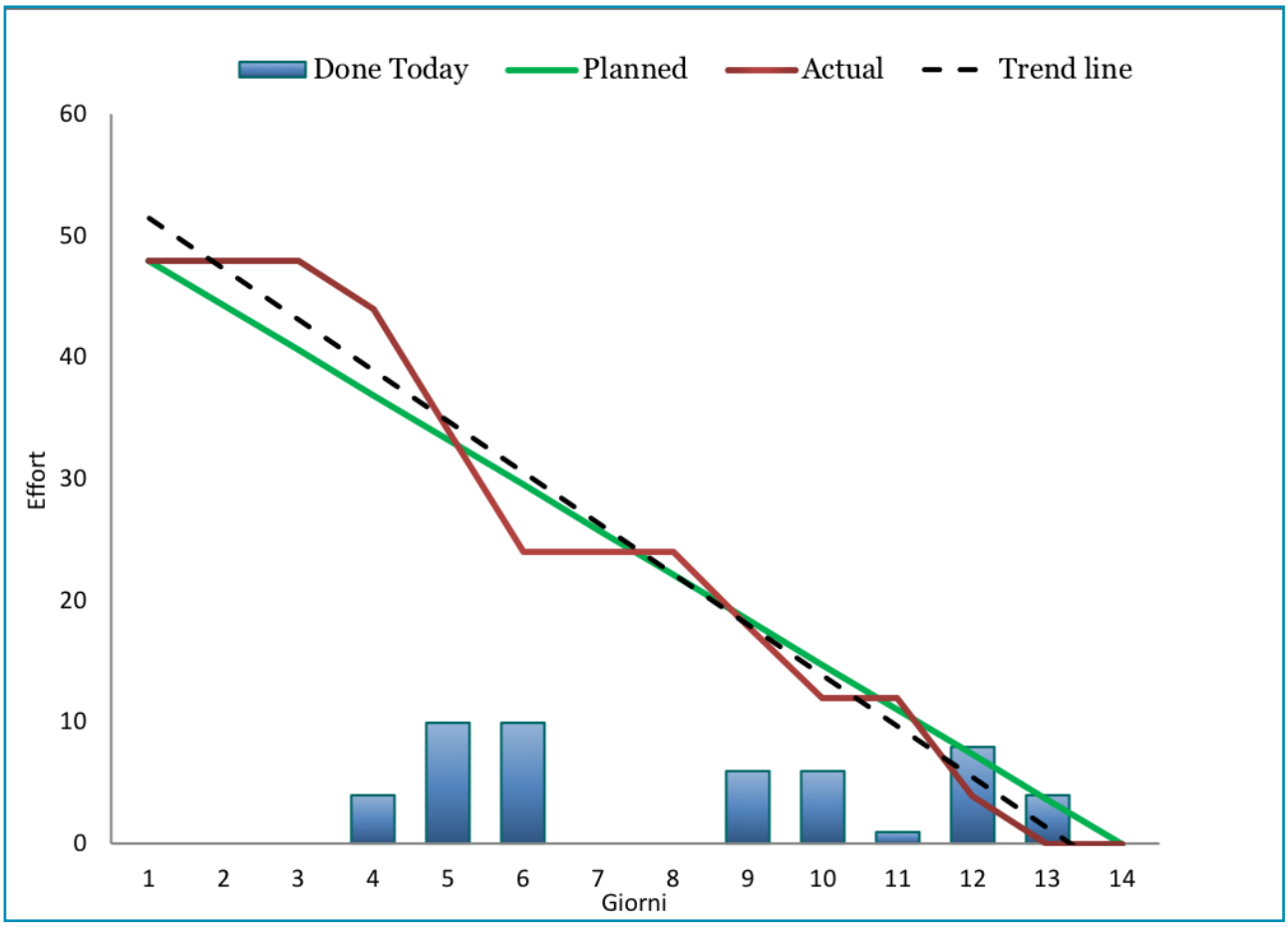
- Bin (delete an element but not remove it from DB)
- Login
- Graphical improvements
    - Graphical effects (e.g. flags)
    - Details explosions
- Opening ticket by mail (start to think a solution)
    - Basic solution (redirect to form)
    - Middle solution (correct formatting implies ticket opening)
    - Complex solution Rete Neurale
- Autogenerated tickets by the system
    - Generate tickets from methods (Spring annotation).
    - Thread pull with Spring (activated by Cron)
    - Any more complex solutions.
- Filtered search
- Pagination

Sprint Planning Meeting

- Regarding opening tickets by mail we decided to keep in the Product Backlog a solution that we define “complex solution”. It consists in an implementation of a very simple neural network able to opening a ticket automatically according to the words in the mail text even if a specific format is not followed. This solution has not been released to the client and in the future we will decide if implement it or not.

Sprint duration					14 days
General Tasks	**Specific Tasks**	**Priority**	**Effort**	**Tasks added**	**Tasks completed**
/	Bin	5	4		
/	Login	5	3		
Graphical improvements	Graphical effects	2	2		
	Details explosion	2	5		
Ticket opening by mail	Basic solution	5	6		
	Middle solution	2	8		**Not released**
	Complex solution	1	10		
Autogenerated tickets by the system	Generate tickets from methods	5	3		
	Thread pull with Spring	3	7		
	Any complex solutions	1	9		
/	Filtered search	4	5		
/	Pagination	4	5		
			Product Backlog	**Sprint Backlog**	**Increase**
Total Tasks			**11**	**9**	**7**
Total Effort			**67**	**48**	**48**

Burndown Chart



Sprint Review Meeting

- Regarding “opening ticket by mail” task’s release, the team has decided not to release the “middle solution” implementation. The client had requested a simple draft of a solution to the problem that has been implemented and released. This harder and more complex implementation has been began but not ended. The team, aware that it could have needed more than one Sprint, decided not to release.

Retrospective Meeting

- Even if it has been a 14-days-Sprint we could not anticipate the tasks’ closure. In fact, we have complete them the day before the release to the client but the idea is to try to begin tasks’ development earlier in the following Sprints.

Sprint 4 (29 May - 12 June)

Tasks:

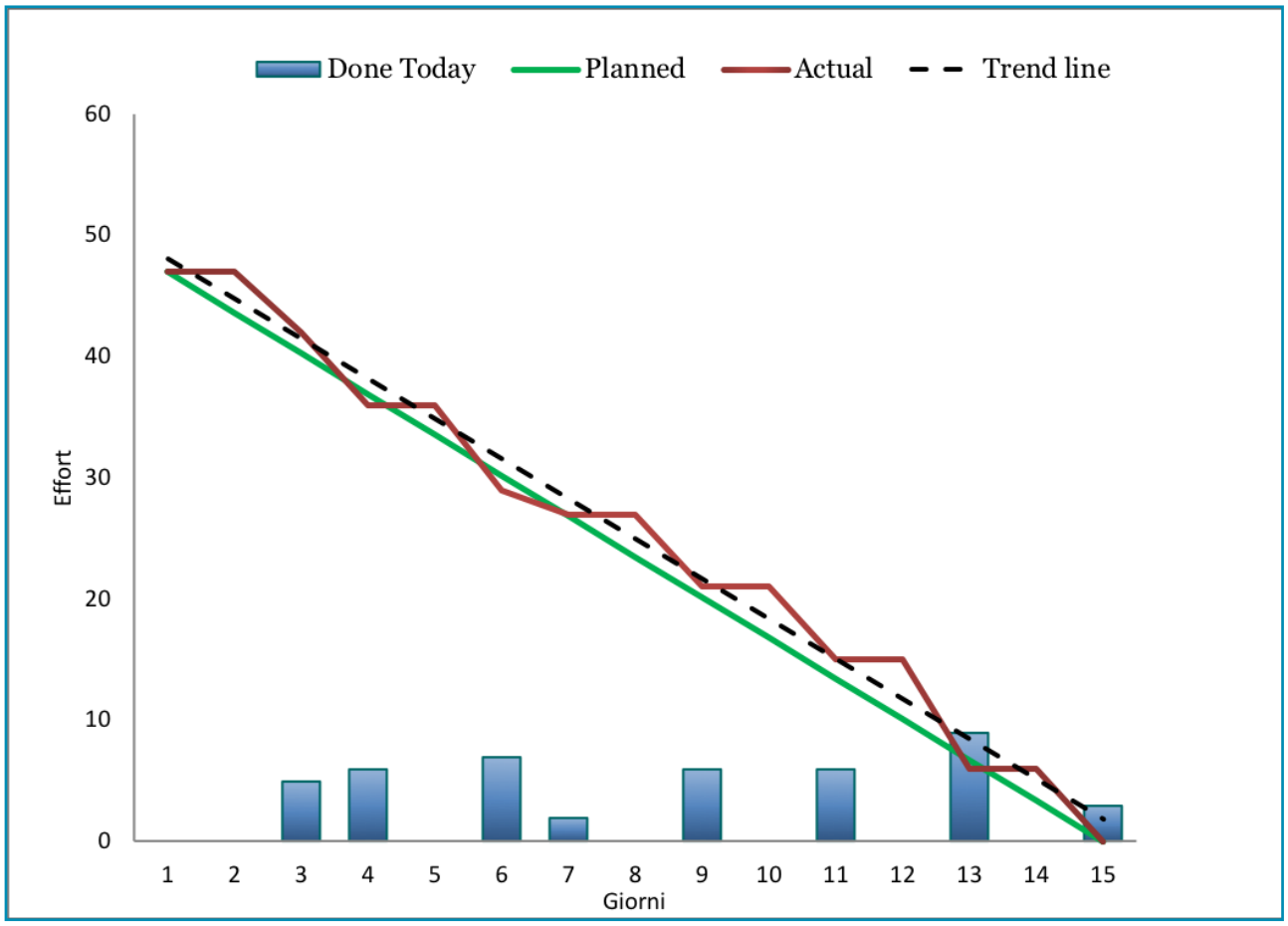
- Log-out management
- FindAllNotDeleted a titolo di ricerca
- Graphical improvements (views presentation)
    - Ticket visualisation
    - Team visualisation
- Ticket opening by mail
    - Mail domain check
    - Middle solution
    - Complex solution
- Automatic tickets generation
    - Inizialization Bean loading
    - Custom Queries insertion (and DB only-read user creation and SQL queries check)
    - Tree structure

Sprint Planning Meeting

- Regarding opening tickets by mail we decided to keep in the Product Backlog a solution that we define “complex solution”. It consists in an implementation of a very simple neural network able to opening a ticket automatically according to the words in the mail text even if a specific format is not followed. This solution has not been released to the client and in the future we will decide if implement it or not.

Sprint duration					15 days
General Tasks	**Specific Tasks**	**Priority**	**Effort**	**Tasks added**	**Tasks completed**
/	Logout	5	3		
/	FindAllNotDeleted	1	5		
Graphical improvements (presentazione schermata)	Ticket visualization	5	5		
	Team visualization	5	5		
Ticket opening by mail	Mail domain check	5	3		
	Middle solution (completion)	4	6		
	Complex solution	2	10		
Autogenerated tickets by the system	Improvements	4	2		
	Custom insertion	4	8		
	Tree structure	4	9		
			Product Backlog	**Sprint Backlog**	**Increase**
Total Tasks			**9**	**8**	**8**
Total Effort			**57**	**47**	**47**

Burndown Chart



Retrospective Meeting

The principal result the meeting pointed out is:

- We tried to anticipate most of the work in the first half of the Sprint more than we have done in the previous Sprints. The team was late on some tasks' implementation because of its members' commitments in other projects. This generated some problems later forcing the team to gather the work at the end of the Sprint. In this Sprint it has been necessary to spend some time dedicated to the team meetings to solve problems related to git branches' merges.

Sprint 5 (13 June - 25 June)

Task:

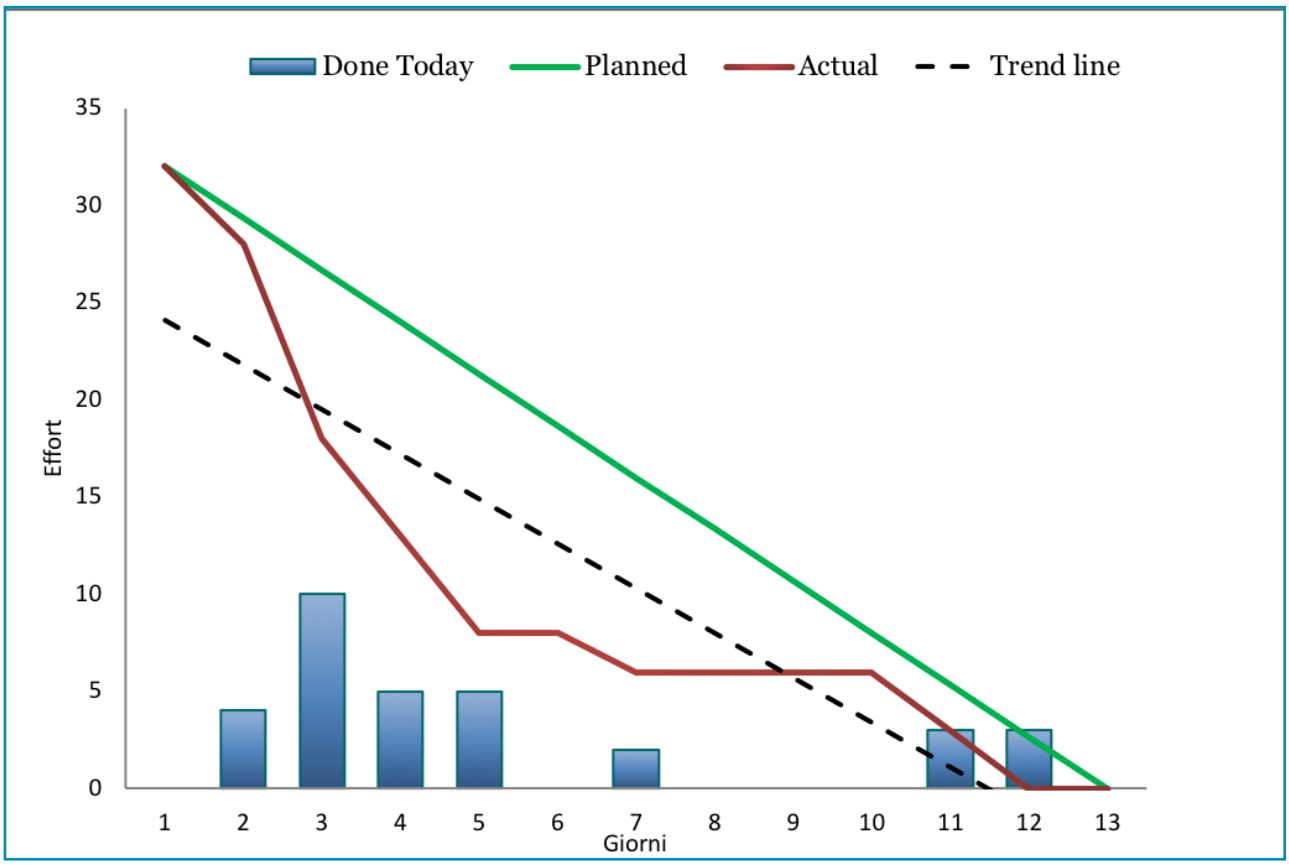
- Cron improvements (months and days check)
- Auditing library integration (other team's library)
- System improvements: configurable tables' prefixes.
- Graphical improvements:
  - Query card dimension
  - Queries enabling and disabling button
- Autogenerated tickets by the system:
  - Improvements (CRUD, query enabling and disabling)
  - Foreign DB approach
- Ticket opening through mail:
  - Complex solution

Sprint Planning Meeting

- We decided to insert in the Product Backlog the "Complex Solution" task referred to "opening ticket by mail" use case. This task has not been completely despatched and its conclusion has been postponed to the following Sprints.

Sprint duration					**13 days**
General Tasks	**Specific Tasks**	**Priority**	**Effort**	**Tasks added**	**Tasks completed**
/	Cron improvements	5	4		
/	Auditing library integration	5	7		
	Improvements		2		
Graphical improvements	Query card dimension	5	2		
	Button	5	2		
Automatic query generation	Improvements	5	5		
	Foreign DB approach	5	5		
Ticket opening by mail	Complex solution (parzialmente)	1	5		**Not released**
			Product Backlog	**Sprint Backlog**	**Increase**
Total Tasks			**7**	**7**	**6**
Total Effort			**32**	**32**	**32**

Burndown Chart



Sprint Review Meeting

We decided not to release the not completed complex solution as we had planned in the meeting at the Sprint beginning.

Retrospective Meeting

Meeting summary is the following:

- Sprint has been completed well in advance, so that the team had the opportunity to improve some details regarding documentations and code refactoring.
- Last days' effort is predominantly due to the implementation of opening ticket task.
- Improvements were expected because of "iteration speed" increase, due to team performance (work division and team communication) increase.

Sprint 6 (26 June - 9 July)

Tasks:

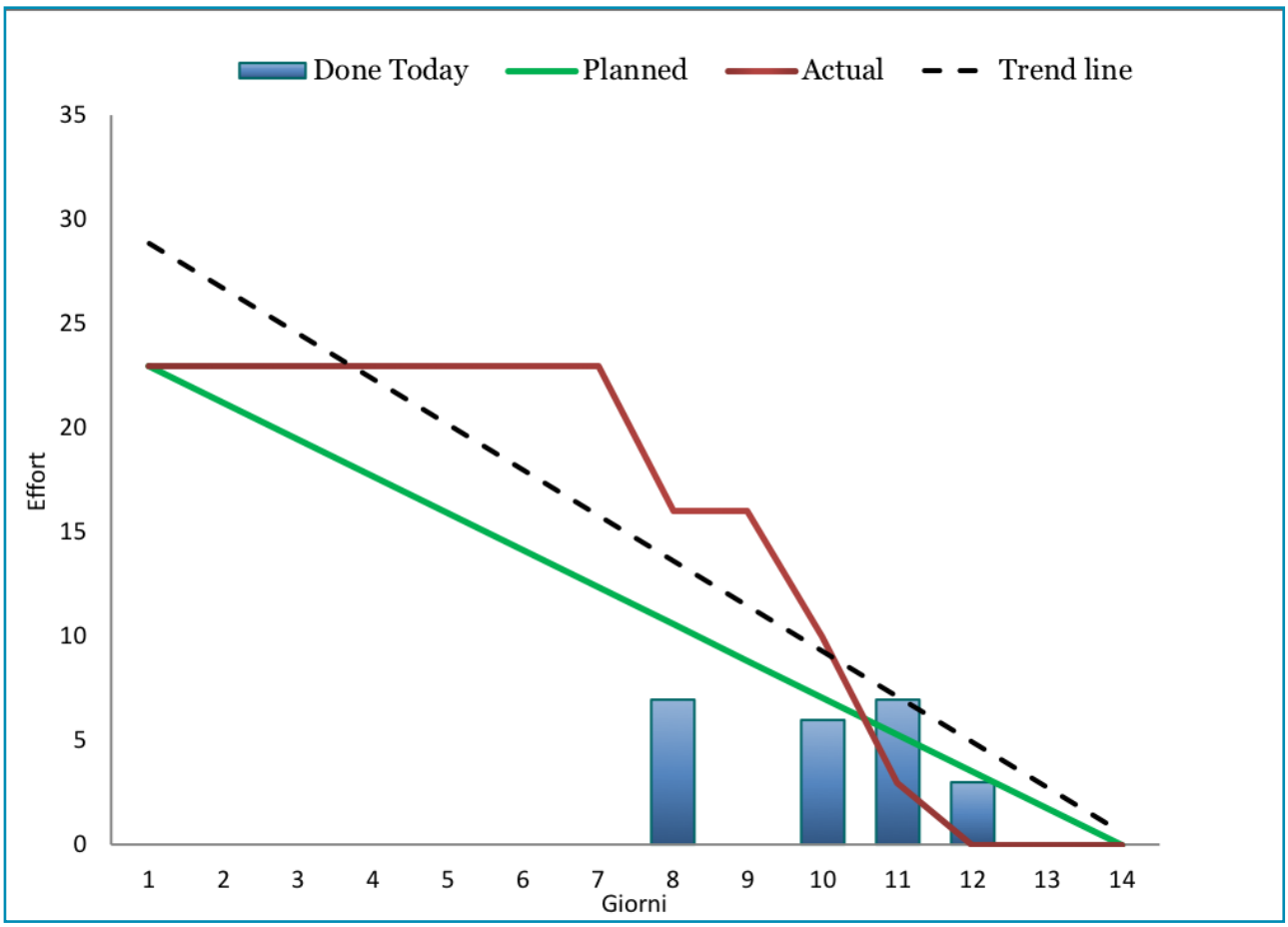
- Final improvements
- DEMO implementation
- Abilitare la configurazione del logging su query automatiche
- Opening ticket by mail
    - Complex solution

Sprint Planning Meeting

We decided not to insert in the Sprint Backlog “opening ticket by mail” task in order to focus on a complete demo creation and because of the team decision to anticipate the final release to the clients.

Sprint duration					**14 days**
General Tasks	**Specific Tasks**	**Priority**	**Effort**	**Tasks added**	**Task completed**
/	Final improvements	4	7		
/	Demo	5	8		
/	Logging configuration	5	8		
Ticket opening by mail	Complex solution (conclusion)	1	10		
			Product Backlog	**Sprint Backlog**	**Increase**
Total Tasks			**4**	**3**	**3**
Total Effort			**33**	**23**	**23**

Burndown Chart



Timesheet

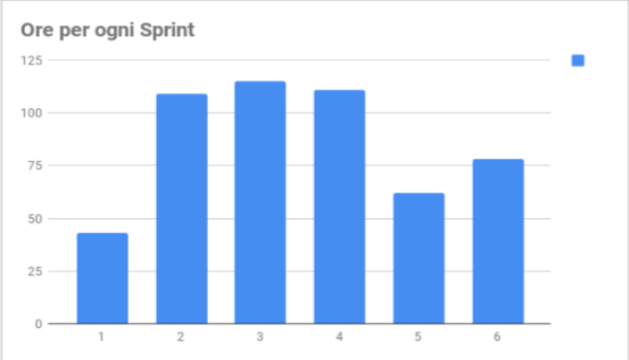
	85	93	84	84	88	84					
DATA	DITOMA	MANCINI	MENZOLINI	OTTAVIANO	SILVI	VINTARI	SPRINT	TOT	TASK	DISCIPLINE (RUP)	NOTE
April 23, 2018	2	2		2			1	6	Intervista agli stakeholder	Requirements	Brainstorming 01 - Elicitazione dei requisti
April 24, 2018		1		1			1	2	Analisi "needs" degli stakeholder	Requirements	Revisione delle needs e invio della mail riepilogativa al cliente
April 29, 2018	2	2	2		2	2	1	10	Sprint Planning Meeting	Project Management	
May 5, 2018		3					1	3	Implementazione	Implementation, Environment	Realizzazione CRUD Ticket
May 6, 2018	2	2		2	2	2	1	10	Sprint Review Meeting	Project Management	Produzione: Glossario, Burndown Chart.
May 7, 2018	2	2	2	2	2	2	1	12	Intervista agli stakeholder	Requirements	Brainstorming 02 - Elicitazione dei requisti (CRUD ticket non rilasciato)
May 8, 2018		1		1			2	2	Revisione documentazione intervista	Requirements	Invio mail riepilogativa
May 8, 2018	1	1				1	2	3	Incontro con altri team	Analysis and Design	Definizione campi dei CRUD User, Target, Team, Ticket
May 9, 2018		2	2	2			2	6	Sprint Planning Meeting	Analysis and Design, Project Management	
May 10, 2018	2			2		2	2	6	Daily SCRUM + Implemetazione	Implementation, Environment	Realizzazione CRUD per BE-Target, FE-Target
May 11, 2018	3				3		2	6	Daily SCRUM + Implemetazione	Implementation, Environment	Inizio realizzazione CRUD BE-User, BE-Team, BE-Ticket
May 12, 2018	5	3	1	5	5		2	19	Daily SCRUM + Implemetazione	Implementation, Environment	Realizzazione CRUD BE-User, BE-Team, BE-Ticket
May 13, 2018		3	1	8	8		2	20	Daily SCRUM + Implemetazione	Implementation, Environment	Realizzazione CRUD FE-User, FE-Team, FE-Ticket
May 13, 2018	3		3			3	2	9	Testing	Testing	Realizzazione test BE
May 14, 2018	3	3	3	3	3	3	2	18	Sprint Review Meeting	Project Management, Deployment	Produzione: Vision, Use Case, Burndown Chart, Class Diagram
May 14, 2018	3	3		3	3	3	2	15	Release al Cliente	Requirements	Rilascio CRUD e riunione con gli staeholders
May 14, 2018	1	1		1	1	1	2	5	Retrospective Meeting	Project Management	
May 15, 2018	1						3	1	Revisione e analisi "needs" degli stakeholder	Requirements, Config and Change Mng	Invio mail riepilogativa
May 15, 2018	2		2			2	3	6	Incontro con altri team	Project Management	Analisi dei punti di contatto tra i team
May 16, 2018	3	3	3	3	3	3	3	18	Sprint Planning Meeting	Analysis and Design, Project Management	
May 18, 2018		4		2			3	6	Daily SCRUM + Implemetazione	Implementation	Realizzazione cestino
May 19, 2018			1		5		3	6	Daily SCRUM + Implemetazione	Implementation	Realizzazione Login, migliroamenti grafici
May 20, 2018	3	5			4		3	12	Daily SCRUM + Implemetazione	Implementation	Realizzazione Ricerche filtrare e paginazione
May 23, 2018			5		1	3	3	9	Daily SCRUM + Implemetazione	Implementation	Inizio realizzazone apertura tramite mail
May 24, 2018		4				5	3	9	Daily SCRUM + Implemetazione	Implementation	Inizio realizzazione creazione ticket automatici
May 26, 2018	1		8		1		3	10	Daily SCRUM + Implemetazione	Implementatio	Realizzazone apertura tramite mail
May 27, 2018		3		1	1	3	3	8	Daily SCRUM + Implemetazione	Implementation	Realizzazione creazione ticket automatici
May 28, 2018	2	2	2	2	2	2	3	12	Sprint Review Meeting	Project Management, Deployment	
May 28, 2018	2	2	2	2	2	2	3	12	Release al Cliente	Requirements	Rilascio apertura tramite mail e ticket automatici e riunione staeholders
May 28, 2018	1	1	1	1	1	1	3	6	Retrospective Meeting	Project Management	
May 29, 2018	2	2		2			4	6	Revisione e analisi "needs" degli stakeholder	Requirements, Config and Change Mng	Invio mail riepilogativa
May 30, 2018	3	3	3	3	3	3	4	18	Sprint Planning Meeting	Analysis and Design, Project Management	
May 31, 2018				1	5		4	6	Daily SCRUM + Implemetazione	Implementation	Realizzazione Logout, miglioramenti grafici
June 1, 2018	1		3		1		4	5	Daily SCRUM + Implemetazione	Implementation	Realizzazione controllo dominio mail
June 3, 2018	2		2		4	2	4	10	Daily SCRUM + Implemetazione	Implementation	Realizzazioni miglioramenti FE
June 4, 2018	1		8	1		8	4	18	Daily SCRUM + Implemetazione	Implementation	Realizzazione avanzamenti apertura tramite mail
June 6, 2018	2		2			4	4	8	Testing	Testing	Testing apertura mail
June 8, 2018		3		3			4	6	Daily SCRUM + Implemetazione	Implementation	Inizio realizzazione inserimento query libera
June 10, 2018		3			1		4	4	Daily SCRUM + Implemetazione	Implementation	Realizzazione inserimento query libera
June 12, 2018	2	2	2	2	2	2	4	12	Sprint Review Meeting	Project Management, Deployment	
June 12, 2018	2	2	2	2	2	2	4	12	Release al Cliente	Requirements	Rilascio miglioramenti mail e ticket automatici e riunione stakeholders
June 12, 2018	1	1	1	1	1	1	4	6	Retrospective Meeting	Project Management	
June 13, 2018	1	1		1			5	3	Revisione e analisi "needs" degli stakeholder	Requirements, Config and Change Mng	
June 13, 2018	1	1	1	1	1	1	5	6	Sprint Planning Meeting	Analysis and Design, Project Management	
June 14, 2018	2				4		5	6	Daily SCRUM + Implemetazione	Implementation	Realizzazione miglioramenti FE
June 15, 2018		4		4			5	8	Daily SCRUM + Implemetazione	Implementation	Realizzazione miglioramento CRON e collegamento a DB esterno
June 16, 2018	1	1	1	2			5	5	Testing	Testing	Testing CRON
June 17, 2018	1		3			3	5	7	Daily SCRUM + Implemetazione	Implementation	Inizio Realizzazione integrazione libreria di log
June 24, 2018			1	1	1		5	3	Daily SCRUM + Implemetazione	Implementation	Realizzazione integrazione libreria di log
June 25, 2018	2	2	2	2	2	2	5	12	Sprint Review Meeting	Project Management, Deployment	
June 25, 2018	1	1	1	1	1	1	5	6	Release al Cliente	Requirements	Rilascio miglioramenti effettuati
June 25, 2018	1	1	1	1	1	1	5	6	Retrospective Meeting	Project Management	
June 26, 2018	1	1	1	1	1	1	6	6	Revisione e analisi "needs" degli stakeholder	Requirements, Config and Change Mng	
June 26, 2018	1	1	1	1	1	1	6	6	Sprint Planning Meeting	Analysis and Design, Project Management	
July 2, 2018	4			4			6	8	Revisione documentazione	Project Management, Deployment	
July 2, 2018		3					6	3	Daily SCRUM + Implemetazione	Implementation	Realizzazione miglioramenti collegamento DB esterno
July 4, 2018				1			6	1	Daily SCRUM + Implemetazione	Implementation	Realizzazione miglioramenti FE
July 5, 2018			4			4	6	8	Daily SCRUM + Implemetazione	Implementation	Inizio realizzazione Demo e configurazione logging su query automatiche
July 6, 2018	3	2	3		2	2	6	12	Daily SCRUM + Implemetazione	Implementation	Realizzazione Demo e configurazione logging su query automatiche
July 7, 2018	2	2		2	2	2	6	10	Daily SCRUM + Implemetazione	Implementation	Realizzazione Demo e miglioramenti FE
July 9, 2018	2	2	2	2	2	2	6	12	Sprint Review Meeting	Project Management, Deployment	
July 9, 2018	2	2	2	2	2	2	6	12	Release al Cliente	Requirements	Presentazione finale

In the Timesheet we inserted DISCIPLINE (RUP) column. This column, inserted after a professor request, represents an attempt to map on RUP model our timesheet's tasks.
We tried to couple each task with RUP disciplines. The team has followed an Agile approach instead of RUP one so, sometimes, coupling may seem forced and sometimes, one of our project's task couples with more than a single RUP discipline.
This attempt, recovered in the next paragraph, is a strong approximation.

Dashboard

Let's extrapolate some relevant data from the timesheet.
Firstly, we show spent hour for each Sprint and the relative histogram.

SPRINT	ORE	%
1	43	8.30%
2	109	21.04%
3	115	22.20%
4	111	21.43%
5	62	11.97%
6	78	15.06%
TOT	518	100.00%



Secondly, we associate each task with a category. This mapping is clearly an internal agreement to the team, because SCRUM does not impose categorization.
For each category, the hours amount and the percentage have been calculated.
The results have been shown on a pie chart.

TASK	ORE	%
Requirements	105	20.27%
Implementation	226	43.63%
Testing	27	5.21%
Organization	50	9.65%
Documentation	81	15.64%
Not Catagorized	29	5.60%
TOT	518	100.00%

Furthermore, on professor's request, a mapping on RUP disciplines has been tried.
Naturally, this attempt results to be an approximation if we consider the timesheet shown above and that the team has followed a SCRUM process.
In the timesheet a SCRUM task is often mapped on two RUP disciplines. In this kink of situations the team decided to split that task's work hours up in a uniform way into the RUP disciplines. This allowed us to create a pie chart that we confirm to be an obvious approximation.

DISCIPLINA	ORE	%
Business Modeling		0.00%
Requirements	79	15.25%
Analysis and Design	86	16.60%
Implementation	185.5	35.81%
Test	22	4.25%
Deployment	33	6.37%
Configuration and Change Management	9	1.74%
Project Management	78	15.06%
Enviroment	25.5	4.92%
Not Catagorized	0	0.00%
TOT	518	100.00%

Technical Documentation

Dossier on ISSSR 2017-2018 project

Introduction

Ticketing System has been developed into two separated project, back-end side and front-end side. Back-end side has been implemented with framework Spring and all of its dependencies can be found into the *POM* file, because it's a Maven Project.
The front-end side has been implemented using AngularJS framework that we learned during the course and all of its dependencies can be found into the *bower.json* file.
Frontend-Backend communication takes place thanks to REST API.

Mail

The following chapter describes in detail the realization and the functionalities concerning e-mails.
An independent interface has been realized so that it can be used as a black box inside the project's execution flows. Shortly, once an e-mail has been received, mail text is parsed according to a well determinate format (customizable) and the wanted values are extracted. Instead, sending e-mails functionality has been thought primarily in order to have notifications activity to the involved users for some operation's output.
"Been registered" or ticket creation notifications are examples of use.

Settings e format

According to make the library more customizable, some settings can be set.
Each setting must be set in the *application.properties* file.
Below an overview of the settings which can be modified is shown. They are divided into 2 groups:

Receiver/Sender server settings

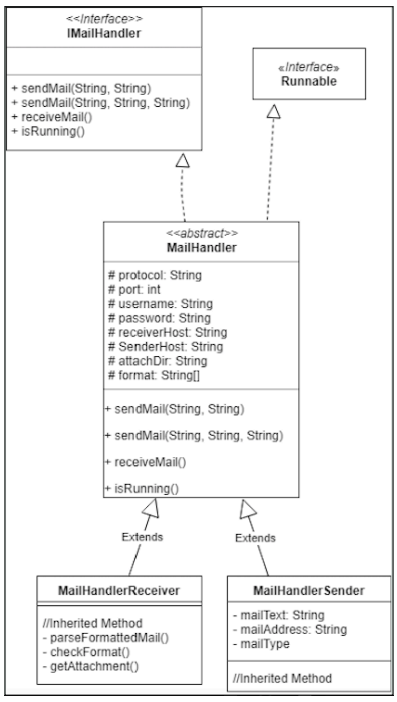
- mail.receiver.protocol: used protocol (IMAP, POP3)
- mail.receiver.host: host where the mail box is registered on.
- mail.receiver.port: provider service port (993, 995)
- mail.receiver.attachmentDirectory: path where attachments can be saved.
- mail.sender.host: sender's host (es. localhost)
- mail.username: account's username
- mail.password: account's password

Mail settings:

- mail.format: values that format must respect to be considered valid
- mail.attach.format: path of the document that shows format template

Sending and receiving mails

In order to receive an e-mail, it's established an IMAP protocol connection to the provider where the mail box is registered on, but it's also possible to set POP3 as protocol.
Receiving operation has been scheduled. On its awakening all unread messages in the INBOX directory are downloaded and a content parsing is done.



The parsing operation verifies that the format inside the application.properties is respected. Regarding this, it is useful to note that as far as a format can be defined with arbitrary number of fields, the format anyway must follow a common template that, in our system's implementation, is like: key: value.
Whenever the format is respected, a ticket is opened. Otherwise, mail is rejected when formatting errors occur and/or wrong information are sent and an automatic reply with the formatting error is sent by the system.
In the generated automatic mail, a .pdf document which contains the explained template is sent as attachment.
Anyway, when a ticket is opened, a confirmation mail is sent.
Further information on the template and the format used by the system are reported into the .pdf document, sent as attachment in the automatic mail.

Thread utilization

Both functionalities (sending and receiving mails) are managed by an asynchronous thread. This design choice is justified by the need to mainly use the "send mail" functionality after front-end operations and it has been considered important to make the system as fluid as possible for the user's point of view.
Furthermore, INBOX directory scanning and the consequent text parsing followed by ticket opening, could bring a relevant overhead as the e-mails amount grows.
A similar scenario brought us to think that a thread could make the system more efficient.

Query

The systems allows to establish conditions under which the tickets are automatically open. It is, at the moment, possible to generate SQL query and compare the result with a reference value.

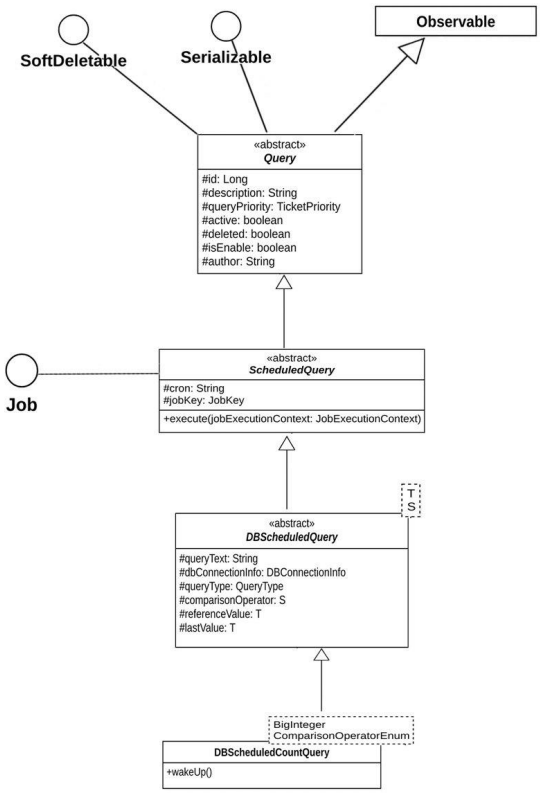
Query logic details

Query class has all the attributes that a verification instance must have:

- *id:* identifier in DB
- *description*: title of the verification instance that appears as description of the possible automatically opened ticket
- *queryPriority:* priority of the ticket that will be opened
- a*ctive:* flag that shows if the query is scheduled or not
- d*elete*: flag that shows if the query has been deleted, but not definitively
- *isEnable*: flag that shows if the logging on the query is enabled
- a*uthor*: mail of the admin who inserted the tuple in the db; it is taken from the session.

It extends the java class ***Observable*** because it's possible to associate observers to it in order to execute verification routines and to find the result.

A *Query* class specialization is ***ScheduledQuery***. It implements the ***Job*** interface for the periodic scheduling. Its relevant attributes are the following: the *cron* that identifies repetition parameters of the task and the *jobKey* that identifies the query inside the Quartz's scheduling.



Scanning the tree we find the ***DBScheduledQuery*** that allows to execute queries scheduled on different SQL databases (at the moment PostgreSQL and MySQL) and to compare the result with a reference value.

Two comparison mechanisms have been implemented: "instant check" and "monitor"; the first one execute SQL query and compare the result directly with the returned value; the second one execute SQL query, subtract the result with the previous detected value, update this value and compare the difference with the reference value.

This class has two generic values (T and S). T indicates the instance type we expect to be returned from the SQL query and so the reference value the results must be compared to and the

attribute where the result is stored into (in monitoring case). On the contrary, S indicates the enumeration used for different kinds of comparisons. Inside the ***DBConnectionInfo*** attribute, DB *url* information are stored through *username* and *password* and the *driver* to use.
It is so possible query different databases (not only the system one).
The first concrete class is the ***DBScheduledCountQuery*** that uses BigInteger and ComparisonOperatorEnum as reference and comparison values. The latter enumerates the following alternatives: "<", ">", "<=", ">=", "=" e "%" (only in monitoring case), operators that can be used with numbers.
When the cron occurs, the job is executed through the method *wakeUp()* that indicates that the object has changed and notifies the observers.
Thanks to the observers it is possible to operate in different ways, for example verification of the execution and the possible ticket creation.
With this kind of structure it is possible to image not scheduled queries that act like listeners, or it can be possible to schedule jobs that do not measure on SQL databases, but, on the contrary, on the system routine execution time.
Furthermore, it is possible to make any kind of comparison with any kind of return value.

Query management

To avoid that unsafe SQL queries on database can be run, at the creation time on front-end side some words check (*INSERT*, *DELETE*, *UPDATE*, *CREATE*, *GRANT*, *DROP*) is done.
In order to ensure security, when the system starts an only-read permissions user is created on the database and all the *DBScheduledQuery* are executed by that user.
Even if this security metrics are used, possible thrown exceptions are monitored in order to notify by mail to the system's admins. Two different exceptions can be thrown:

- ***SQLException***: a mail is sent to the query author to modify the SQL query.
- ***DataAccessException***: a mail is sent to the general admin because the query could violate system constraints; it is also possible to manage exceptions that extend this one (that is abstract) in order to distinguish the behaviours.

In both cases the query is disabled to avoid the exception's throwing whenever it occurs.

Quartz scheduler

Quartz is an open source library for job scheduling, under Apache 2.0 license.

This can be used for those tasks that need periodic or well determined moment executions. It's very useful to execute maintenance or verification jobs.
A task could be any class that implements Job interface.
When a Trigger occurs, the scheduler notifies the jobs belong to a specific group and these ones are enabled through the *execute(JobExecutionContext jobExecutionContext)* method.

There are two different job grouping levels:

1. *CronTrigger*: it is identified by a *CRON_GROUP_NAME* and a *ID;* this groups all those tasks belong to that Trigger.
2. *JobGroup:* each job group has a name; inside the group, the single task is identified with a *ID*

This library has been imported inside the project with Spring and, as a consequence, through the Maven dependency and a configuration class (***SchedulingConfig***) that uses the *SchedulerFactoryBean* to instantiate a scheduler.
The class that manages it is called ***AutoGeneratedTicketService.*** It inserts and removes jobs from the ***TaskScheduler.*** The latter is the *Component* that creates jobs and triggers and couples them inside the scheduler. At the insertion time, the task is mapped in a *JobDetail* through a *JobDataMap* from which the task is later taken to be executed.
Finally, thanks to the *jobKey* related to each task it is possible to remove a job with a single code line, exploiting the Quartz library potency.

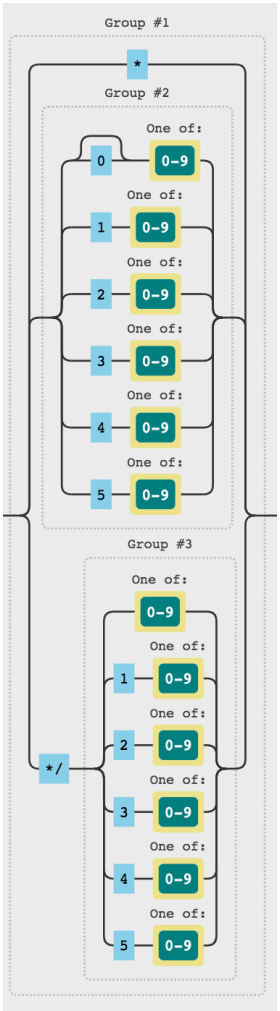
Cron validation

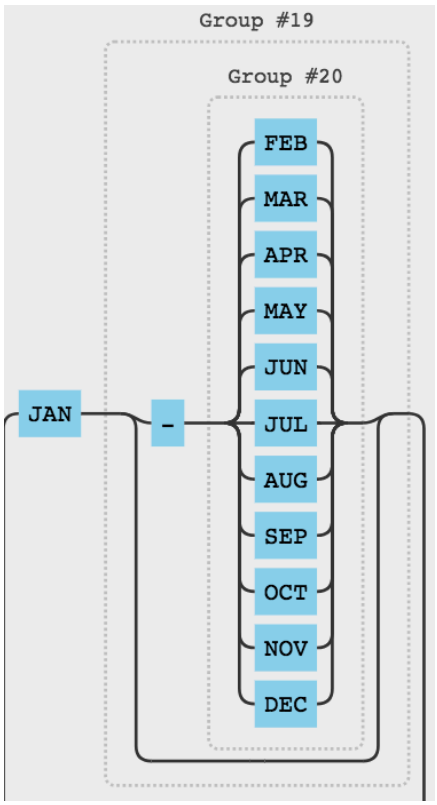
To avoid to have cron not supported by Quartz library, possible configurations range has been limited.
For this reason, it has been realized a regex that verifies the formatting cron on the front-end side.
A 6-values cron is supported and it is possible to insert months and days (even ranges).
The regex string is retrieved below:

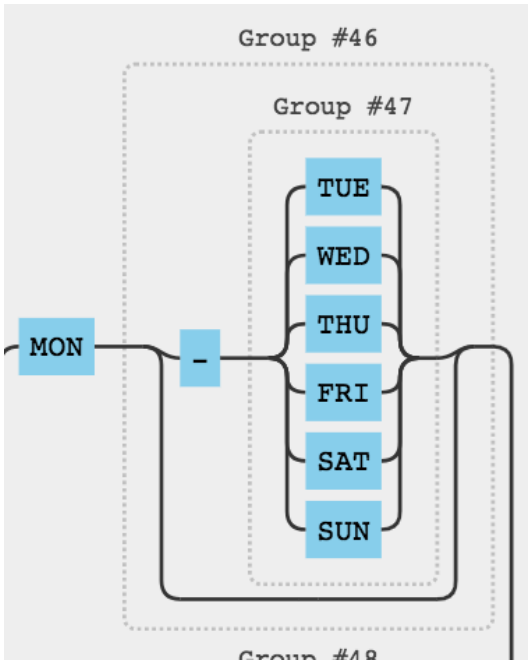
/^(*|(0?[0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9])|*\/([0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9])) (*|(0?[0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9])|*\/([0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9])) (*|(0?[0-9]|1[0-9]|2[0-3])|*\/([0-9]|1[0-9]|2[0-3])) ((* * \?)|(\?\ (((*|(0?[1-9]|1[0-2])|*\/([1-9]|1[0-2])))|(JAN(\-(FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)|)|FEB(\-(JAN|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)|)|MAR(\-(JAN|FEB|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)|)|APR(\-(JAN|FEB|MAR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)|)|MAY(\-(JAN|FEB|MAR|APR|JUN|JUL|AUG|SEP|OCT|NOV|DEC)|)|JUN(\-(JAN|FEB|MAR|APR|MAY|JUL|AUG|SEP|OCT|NOV|DEC)|)|JUL(\-(JAN|FEB|MAR|APR|MAY|JUN|AUG|SEP|OCT|NOV|DEC)|)|AUG(\-(JAN|

FEB | MAR | APR | MAY | JUN | JUL | SEP | OCT | NOV | DEC) |) | SEP(\-(JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | OCT | NOV | DEC) |) | OCT(\-(JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | NOV | DEC) |) | NOV(\-(JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | DEC) |) | DEC(\-(JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV) |))) \ (([1-7]) | (MON(\-(TUE | WED | THU | FRI | SAT | SUN) |) | TUE(\-(MON | WED | THU | FRI | SAT | SUN) |) | WED(\-(MON | TUE | THU | FRI | SAT | SUN) |) | THU(\-(MON | TUE | WED | FRI | SAT | SUN) |) | FRI(\-(MON | TUE | WED | THU | SAT | SUN) |) | SAT(\-(MON | TUE | WED | THU | FRI | SUN) |) | SUN(\-(MON | TUE | WED | THU | FRI | SAT) |)))) | ((0?[1-9] | 1[0-9] | 2[0-9] | 3[0-1]) | *\ / ([1-9] | 1[0-9] | 2[0-9] | 3[0-1])) ((* | (0?[1-9] | 1[0-2]) | *\ / ([1-9] | 1[0-2]))\ \?))$/;

Some validation steps are shown below in a graphical way. To have a complete view of the validation operation, a visual regex generator is required.







Design Documentation

Dossier on ISSSR 2017-2018 project

Vision Document

1. Introduction

The project, born in 2018 for the ISSSR2018 course students, consists in a Ticketing system web app. Students are divided into groups and each group designs, implements and tests a part of the application.
The system required is a ticketing system, a software bought or developed by big companies to provide efficient tools to manage their clients' requests for assistance and to quickly solve the problems and the incidents the clients submit to the company.

2. Positioning

2.1 Problem Statement

The problem of	provide efficient tools to support and help clients simplifying communication with them
affects	big companies, organization and their clients
the impact of which is	costs and revenues, customer retention and public brand image
A successful solution would be	a user-friendly software which collects together in the same place all the requests coming from the clients and allows monitoring in order to manage efficiently and quickly new client's help requests

3. Stakeholder Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibilities
Owner	Software owner	Buy for development and maintenance of the software
Licenser	Organization that needs to use this software to manage help requests from clients	Buy the software license
Customer	Client of the licensers who need to communicate with them to ask for assistance Â	Ask for a ticket to be opened

Name	Description	Responsibilities
Company	Organization that use software bought by Licenser	Ask for a ticket to be opened, could be considered as a unique customer using company domain
Admin	Administrator of the system	Manages users of the system, open automatic tickets
Team member	Member of a licenser team	Resolve a ticket
Team leader	Leader of the team	Control the team
Team coordinator	Manager of the teams	Create teams
Help desk operator	Member of the licenser who assists the clients	Open a ticket on a clientsâ€™ requests

3.2 User Environment

Number of people involved is the number of member of ISSSR 2018 course.
The life of the software development is 3 months.
The used approach is AGILE (in particular SCRUM).
Sprint duration is 2 weeks.
Number of sprints expected: 6.

4. Product Overview

4.1 Needs and Features

Needs	Priority
Admin wants to manage the users	High
Clients want to open a ticket from the system interface	High
Clients want to open a ticket from sending mails	Medium
Admin wants to open automatic tickets	High
Clients want to be notified when a ticket has been received	Low

Features	Priority
The system provides admin panel	High
The system provides the tickets' creation	High
The system allows to receive mails from the clients	Medium
The system provides tools to generate automatic tickets	High
The system provides automatic reply mail	Low

Terms glossary

- **Owner**: Person or company that asks for the develop of the software.
- **Customer**: Licenser's client who ask for assistance
- **Licenser:** Organization that needs to use this software to manage help requests from clients
- **Ticket**: Assistance request from a client. It can be requested in different ways: clients send emails or through web app user interface. The ticket can be opened with no time limitations
- **Automatic Ticket**: Automatically generated tickets by the system because of queries to databases executed by administrator.
- **Target**: Product related to one or more tickets.
- **User**: Account registered to the web app.
- **Company**: Organization that use software bought by Licenser
- **Team**: Group of licenser's employees who manages ticket resolution.
- **Team member**: Member of a licenser's team.
- **Team coordinator**: Licenser's member who split the tickets across the company's teams
- **Team leader**: Team member who coordinate the resolution of the ticket
- **Help desk operator**: Company's member who assists the clients by phone and opens a ticket.
- **Query**: Query on database that generate automatic ticket generation.
- **Administrator**: System admin. As such, he has permissions to do any operation.
- **Mail:** Mail sent from a user to open a ticket and with a well determined format.

Use cases diagram