ReportBeSafe

Simone Gennenzi Matteo Feliziani
Davide Guido Samuele De Rosa

June 2023

Contents

1 The idea **2**

2 User Stories **3**

3 Mockups **4**

4 Non functional requirements **10**

5 Function Points and Cocomo **11**

6 Technologies **12**

7 Software architecture **13**

8 SCRUM **15**
8.1 Release Backlog 17

9 Code **18**

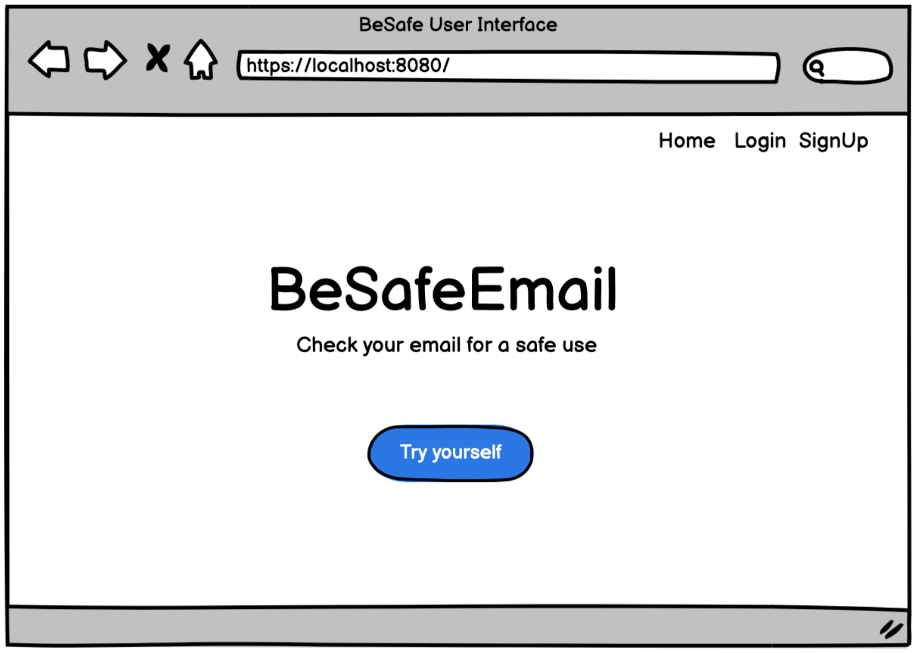
1 The idea

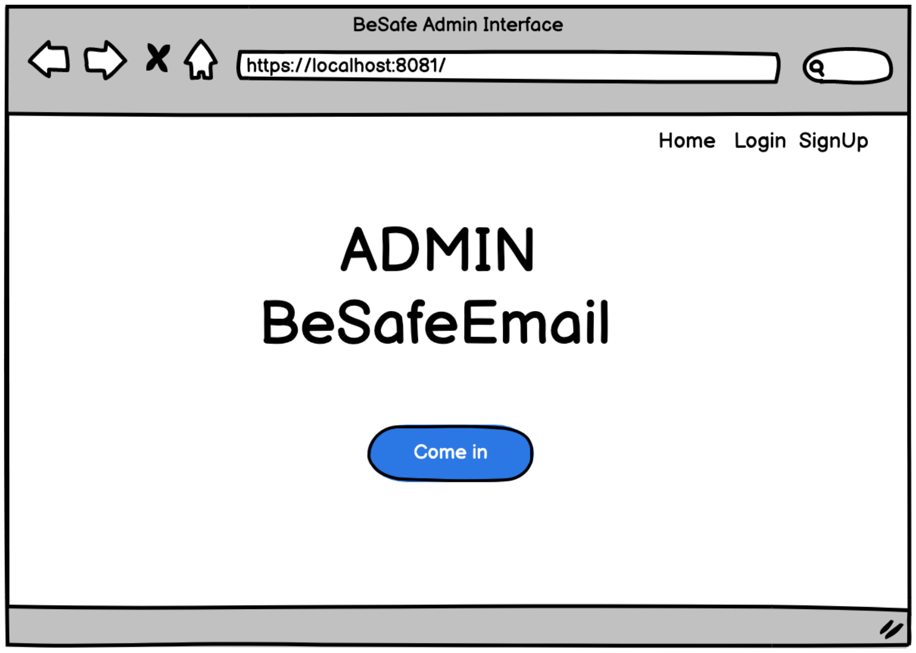
We have developed a distributed application to analyze suspicious email, in particular BSE uses ML based algorithms to classify them in SPAM or not. An use case of the application would be the following one: a client receives a suspicious email, possibly due to the sender's suspected origin or to the content of the email; at this point the user submits sender and the content of the email and these will be processed by two micro-services that analyze independently urls and text.
At the end of the analysis a comprehensive evaluation indicating the probability that the email is SPAM will be reported to the client.
We have implemented also another type of user, the admin, that has the possibility to inspect all the analysis required by the users with their corresponding results.

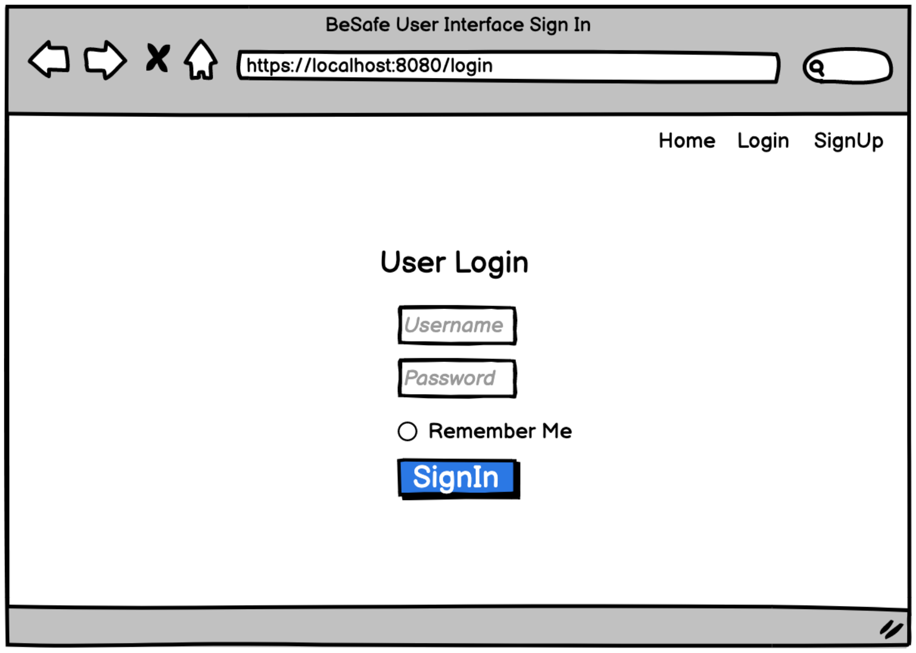
2 User Stories

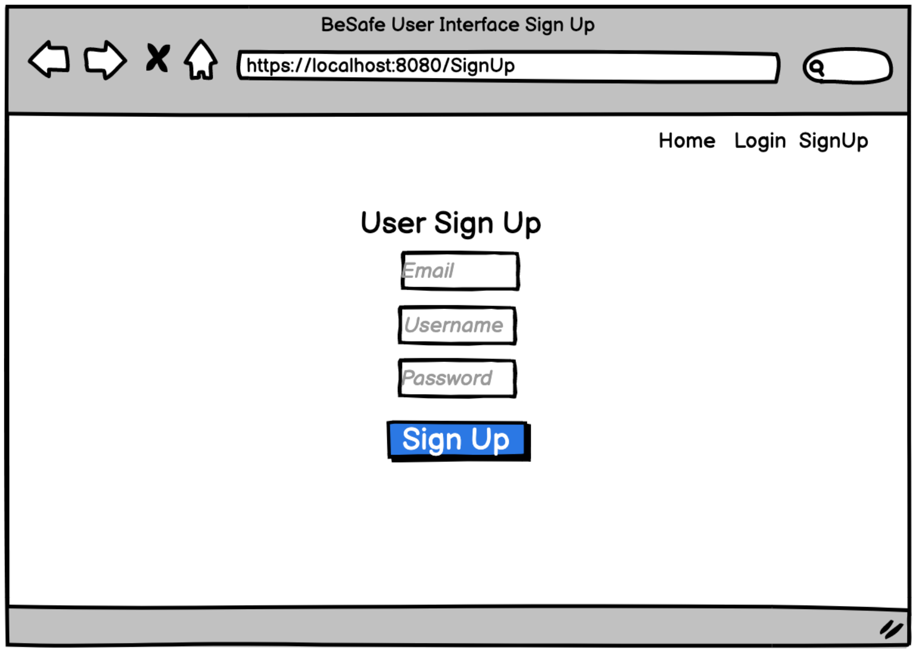
TASK TITLE	**TASK DESCRIPTION**	**ADDED BY**	**DATE ADDED**
User-Interface	As a User I want to have a web interface that I can access so that I can insert email's info	The Team	26/04/23
User Registration	As a User I want to have a page where I can sign up	The Team	26/04/23
User Login	As a User I want to have a page where I can sign in	The Team	26/04/23
History of emails	As a User I want to see the history of all emails I have sent	The Team	10/05/23
Email's danger score	As a User I want to see the email's danger score	The Team	10/05/23
Log out	As a User I want to be able to log out from the site	The Team	10/05/23
Admin-interface	As an Admin I want a dedicated interface completely disconnected from the users' one	The Team	26/04/23
Email's overview	As an Admin I want to have an overview of the danger's score of the emails sent	The Team	30/04/23
Admin Registration	As an Admin I want to have a page where I can sign up	The Team	26/04/23
Admin Login	As an Admin I want to have a page where I can sign in	The Team	26/04/23
Log out	As a User I want to be able to log out from the site	The Team	26/04/23

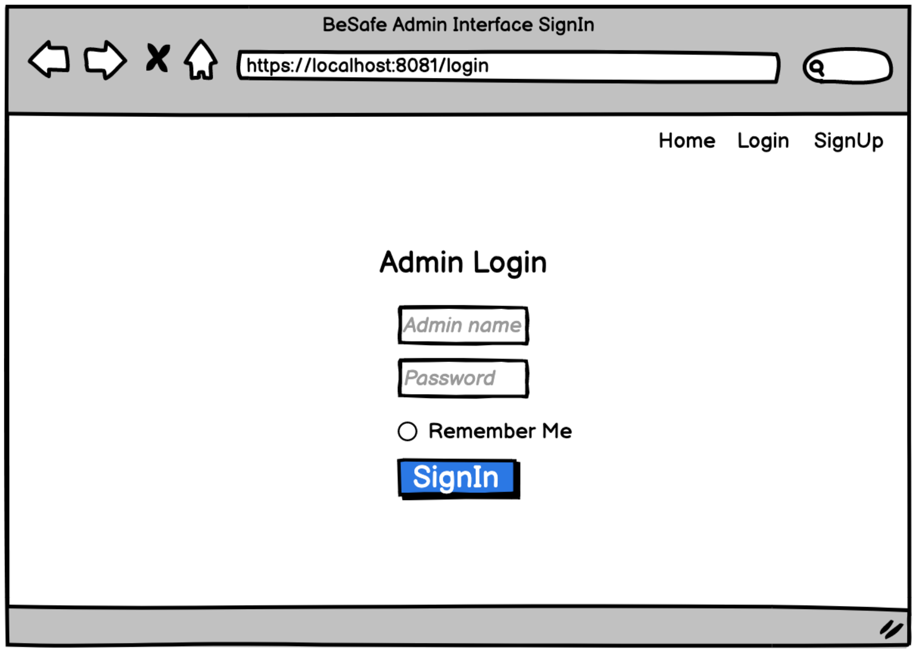
3 Mockups

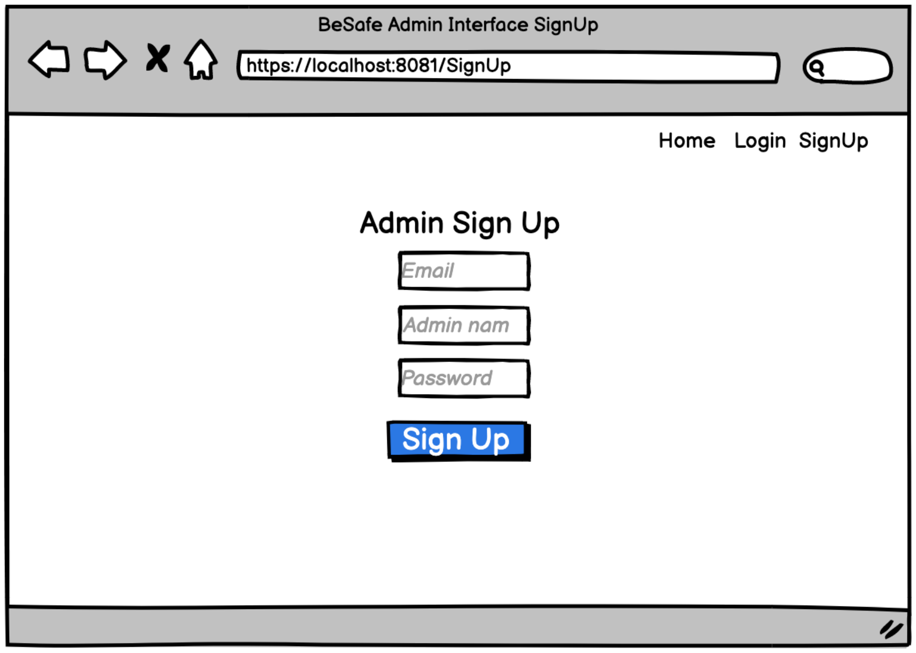

BeSafe User Interface
https://localhost:8080/
Home
Login
SignUp
BeSafeEmail
Check your email for a safe use
Try yourself

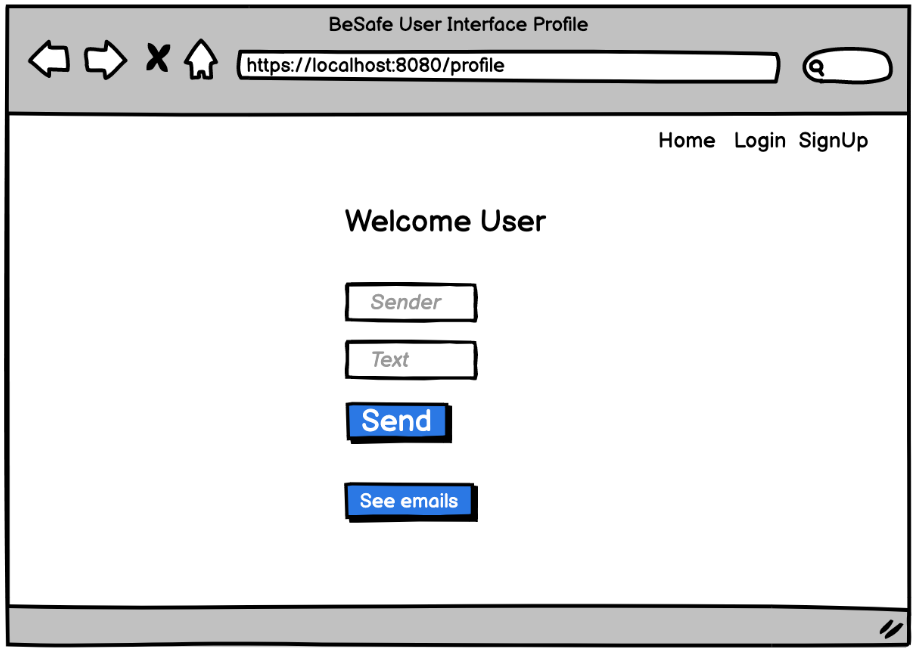

BeSafe Admin Interface
https://localhost:8081/
Home
Login
SignUp
ADMIN
BeSafeEmail
Come in

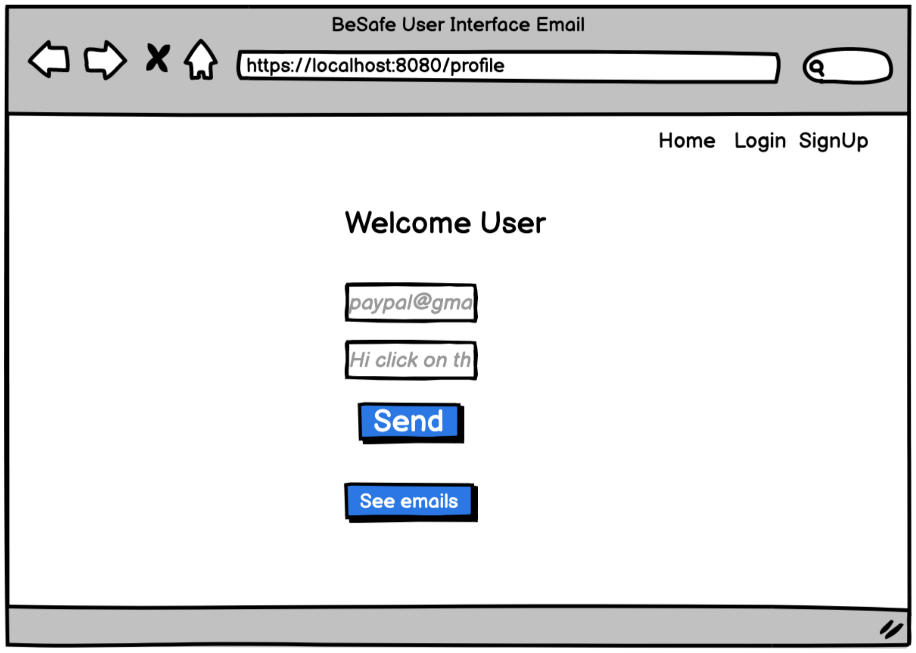

BeSafe User Interface Sign In
https://localhost:8080/login
Home
Login
SignUp
User Login
Username
Password
Remember Me
SignIn

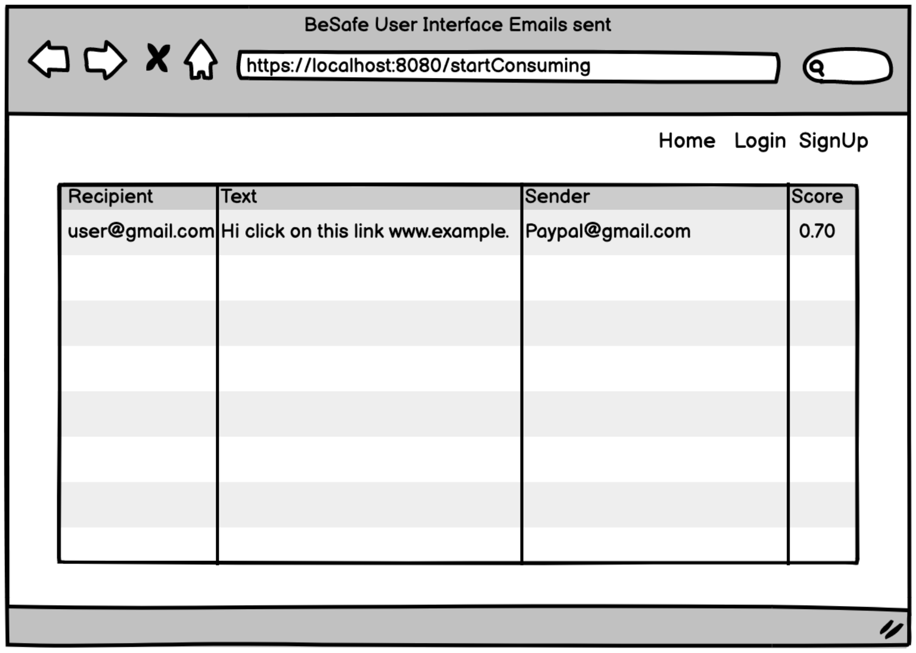

BeSafe User Interface Sign Up
https://localhost:8080/SignUp
Home
Login
SignUp
User Sign Up
Email
Username
Password
Sign Up


BeSafe Admin Interface SignIn
https://localhost:8081/login
Home
Login
SignUp
Admin Login
Admin name
Password
Remember Me
SignIn

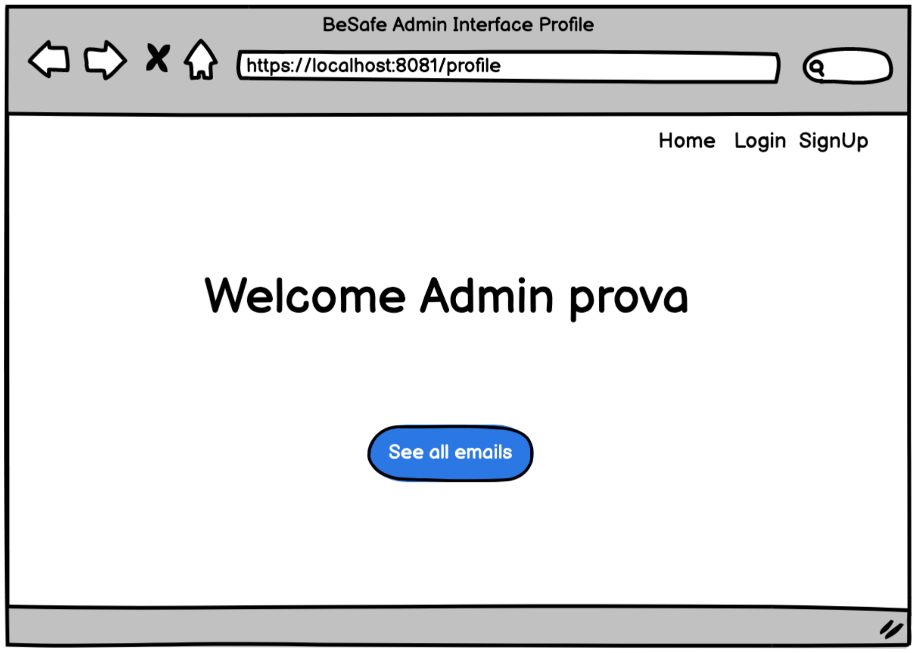

BeSafe Admin Interface SignUp
https://localhost:8081/SignUp
Home
Login
SignUp
Admin Sign Up
Email
Admin nam
Password
Sign Up

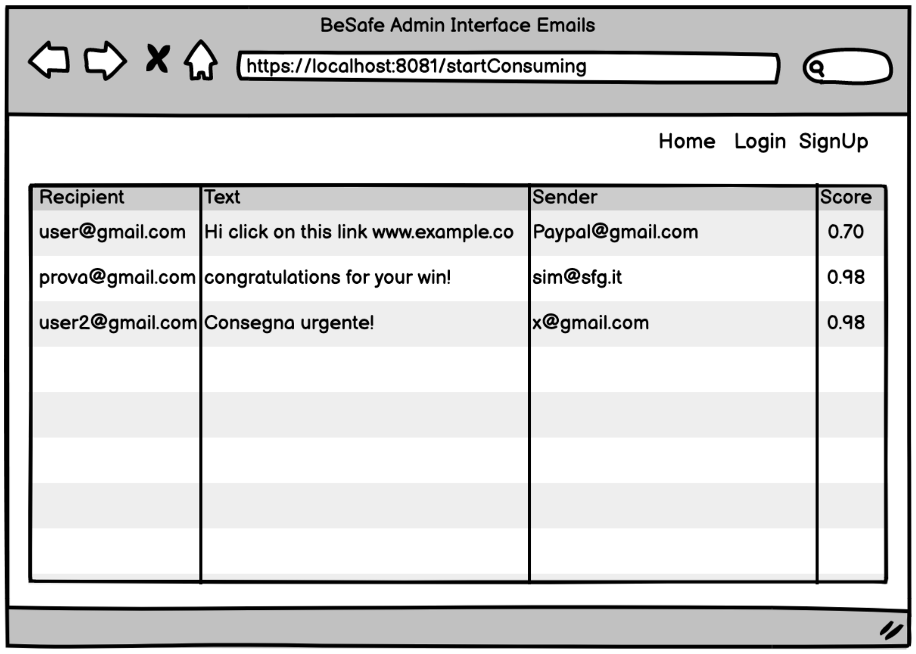

BeSafe User Interface Profile
https://localhost:8080/profile
Home
Login
SignUp
Welcome User
Sender
Text
Send
See emails


BeSafe User Interface Email
https://localhost:8080/profile
Home
Login
SignUp
Welcome User
paypal@gma
Hi click on th
Send
See emails


BeSafe User Interface Emails sent
https://localhost:8080/startConsuming
Home
Login
SignUp

Recipient	Text	Sender	Score
user@gmail.com	Hi click on this link www.example.	Paypal@gmail.com	0.70


BeSafe Admin Interface Profile
https://localhost:8081/profile
Home
Login
SignUp
Welcome Admin prova
See all emails


BeSafe Admin Interface Emails
https://localhost:8081/startConsuming
Home
Login
SignUp

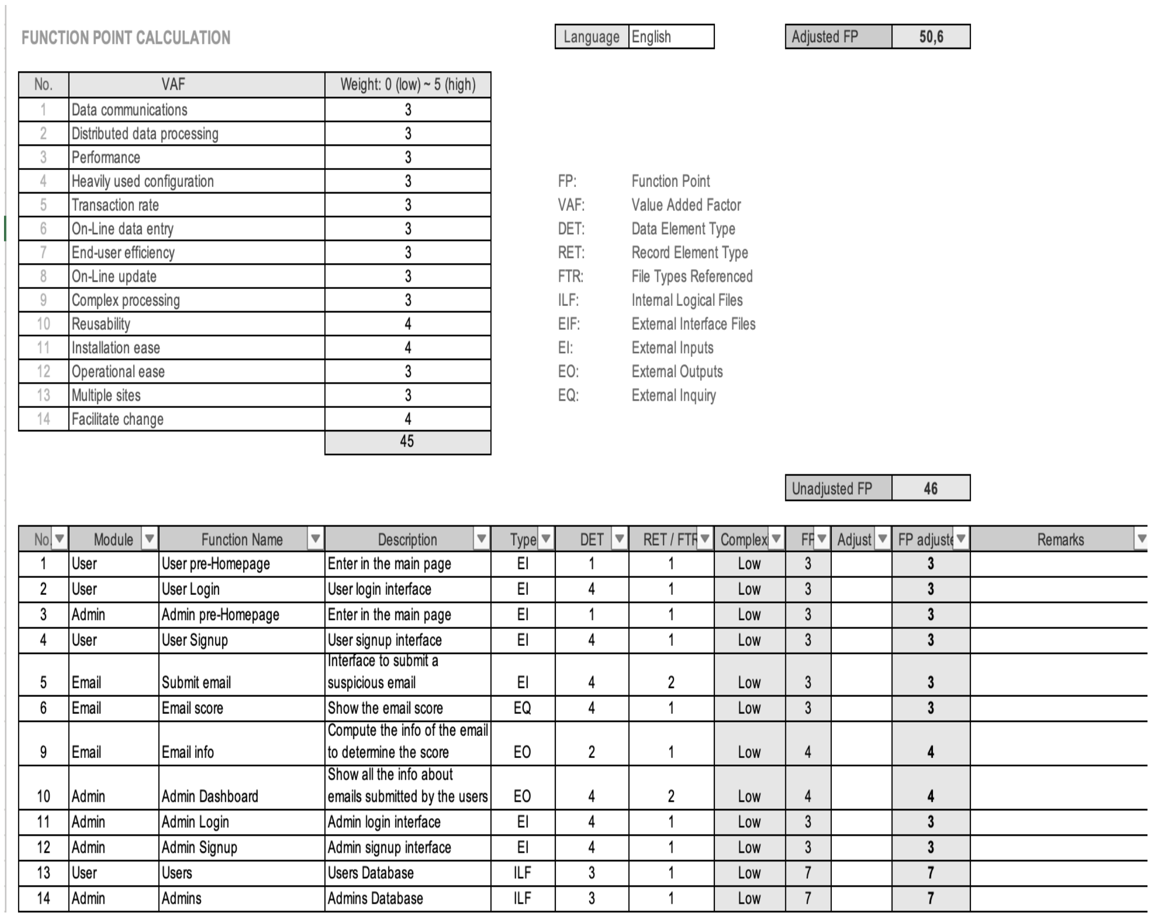
Recipient	Text	Sender	Score
user@gmail.com	Hi click on this link www.example.co	Paypal@gmail.com	0.70
prova@gmail.com	congratulations for your win!	sim@sfg.it	0.98
user2@gmail.com	Consegna urgente!	x@gmail.com	0.98

4 Non functional requirements

Considering also the non functional requirements BeSafe attends to provides the following properties:

1. **Portability:** the application is micro-services oriented, this means that it will be easy to move the micro-services on different machines and hardware in order to match hypothetical business or commercial restrictions.
2. **Scalability:** due to the micro-services architecture the application engages also the scalability properties because it will be easy replicate the several micro-services on more machines.
3. **Usability:** the application should be very easy to use and understand thanks to the very clear implementation of the front-end.
4. **Maintainability:** the application is well modularized, for an external programmer it must be not so difficult to maintain the application
5. **Effectiveness:** the application should accomplish the tasks in the most correct way, the accuracy of the analysis should be as higher as possible.
6. **Efficiency:** the user should not wait too much to have a response from the application.
7. **Reliability:** the application should be as reliable as possible, even in presence of anomalies or crashes.

5 Function Points and Cocomo



FUNCTION POINT CALCULATION

Language	English

Adjusted FP	50,6

No.	VAF	Weight: 0 (low) ~ 5 (high)
1	Data communications	3
2	Distributed data processing	3
3	Performance	3
4	Heavily used configuration	3
5	Transaction rate	3
6	On-Line data entry	3
7	End-user efficiency	3
8	On-Line update	3
9	Complex processing	3
10	Reusability	4
11	Installation ease	4
12	Operational ease	3
13	Multiple sites	3
14	Facilitate change	4
		45

FP:	Function Point
VAF:	Value Added Factor
DET:	Data Element Type
RET:	Record Element Type
FTR:	File Types Referenced
ILF:	Internal Logical Files
EIF:	External Interface Files
EI:	External Inputs
EO:	External Outputs
EQ:	External Inquiry

Unadjusted FP	46

No.	Module	Function Name	Description	Type	DET	RET / FTR	Complex	FP	Adjust	FP adjuste	Remarks
1	User	User pre-Homepage	Enter in the main page	EI	1	1	Low	3		3	
2	User	User Login	User login interface	EI	4	1	Low	3		3	
3	Admin	Admin pre-Homepage	Enter in the main page	EI	1	1	Low	3		3	
4	User	User Signup	User signup interface	EI	4	1	Low	3		3	
5	Email	Submit email	Interface to submit a suspicious email	EI	4	2	Low	3		3	
6	Email	Email score	Show the email score	EQ	4	1	Low	3		3	
9	Email	Email info	Compute the info of the email to determine the score	EO	2	1	Low	4		4	
10	Admin	Admin Dashboard	Show all the info about emails submitted by the users	EO	4	2	Low	4		4	
11	Admin	Admin Login	Admin login interface	EI	4	1	Low	3		3	
12	Admin	Admin Signup	Admin signup interface	EI	4	1	Low	3		3	
13	User	Users	Users Database	ILF	3	1	Low	7		7	
14	Admin	Admins	Admins Database	ILF	3	1	Low	7		7	

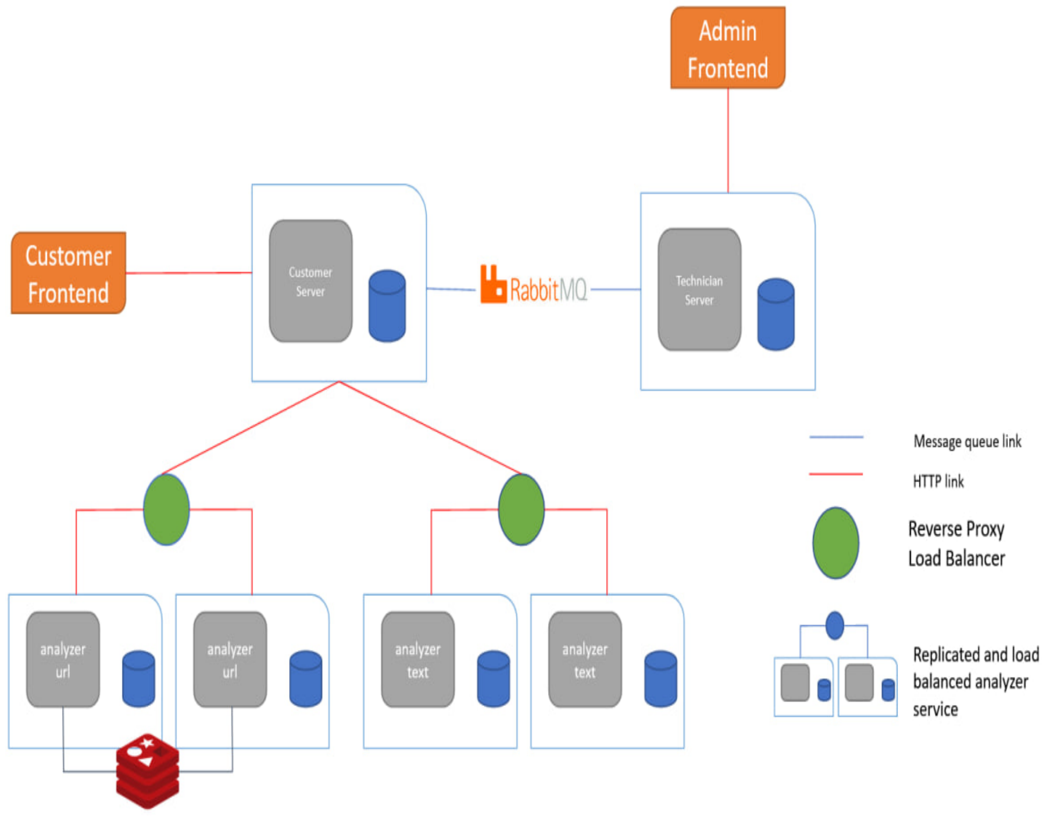
6 Technologies

The technologies implemented in the porject are the following:

- **Docker:** allows us to deploy each micro-service independently from each other, in fact each micro-service has its image and then these images can be orchestrated in order to build a more complex macro-service that leverages the APIs exposed by each micro-service.
- **Flask:** is the web framework with which the REST APIs exposed by the application are implemented
- **RabbitMQ:** it is essentially a Message Broker with several ways of usage, we have used it to implement a persistent queue of message exchanged between the users and the admins.
- **MultinomialNB and CountVectorizer libraries:** is the main logic of the 'SPAManalyzer' service that implement a neural network, already trained, able to classify email in SPAM or not. In order to analyze the email it is need to transform it in an appropriate structure, the 'vectorizer' object accomplish exactly this task: it transforms the content into a numeric feature matrix that is passed as input to the model
- **NGINX:** url-analyzer and text-analyzer services are replicated to guarantee availability of the service even in case of failures and to reduce latency on customer side when there are concurrent requests. To achieve this we used to deploy multiple containers of the url analyzers service and incoming requests are balanced by a NGINX reverse proxy container sitting in front of them
- **Redis:** cache server for fast lookup of already analyzed URLs.
- **MongoDB:** persistent storage for url-analyzer service

7 Software architecture

The high level idea was decomposed into sub jobs and implemented in a microservices pattern. Each microservice has it own responsability and it is decoupled by the others. The customer server offers the interface to the client for accessing the service.



The overall architecture can be described by the following blocks:

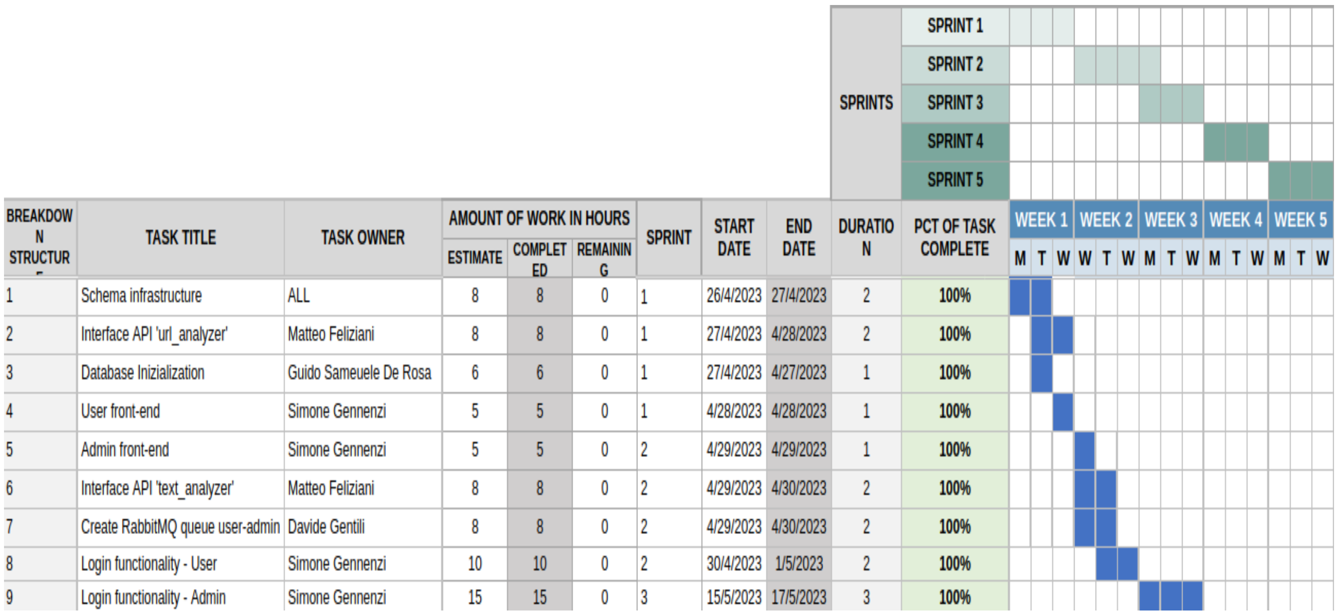
- **CUSTOMER SERVER:** it provides the client of the service as a browser based GUI to submit requests about suspicious email. It extract relevant artifacts from the client's requests and send them to the analyzers, whose results are merged and showed to the client. The result are also written on a queue that the admin can inspect.

- **ADMIN SERVER:** it provides the technician expert a browser based GUI to monitor the analysys in the system, reading in real time from the queue.
- **URLanalyzer:** it implements analysys of urls extracted from emails. It provides its service through REST API and json responses format. It is replicated to guarantee service availability even in case of failures, and to provide responses even in case of high requests from the customer server. The incoming traffic is balanced by a reverse proxy, sitting in front of the replica. Each replica reads and writes persistent data from its own database, but each replica accesses the same instance for fast lookup of already analyzed URLs.
- **SPAManalyzer:** the main logic of this service is based on a ML algorithm that, through a neural network assigns a score to the email analyzed in order to classify it as SPAM-email or not. The network has been trained with supervised learning method, considering examples of emails and their corresponding classification in SPAM or not.

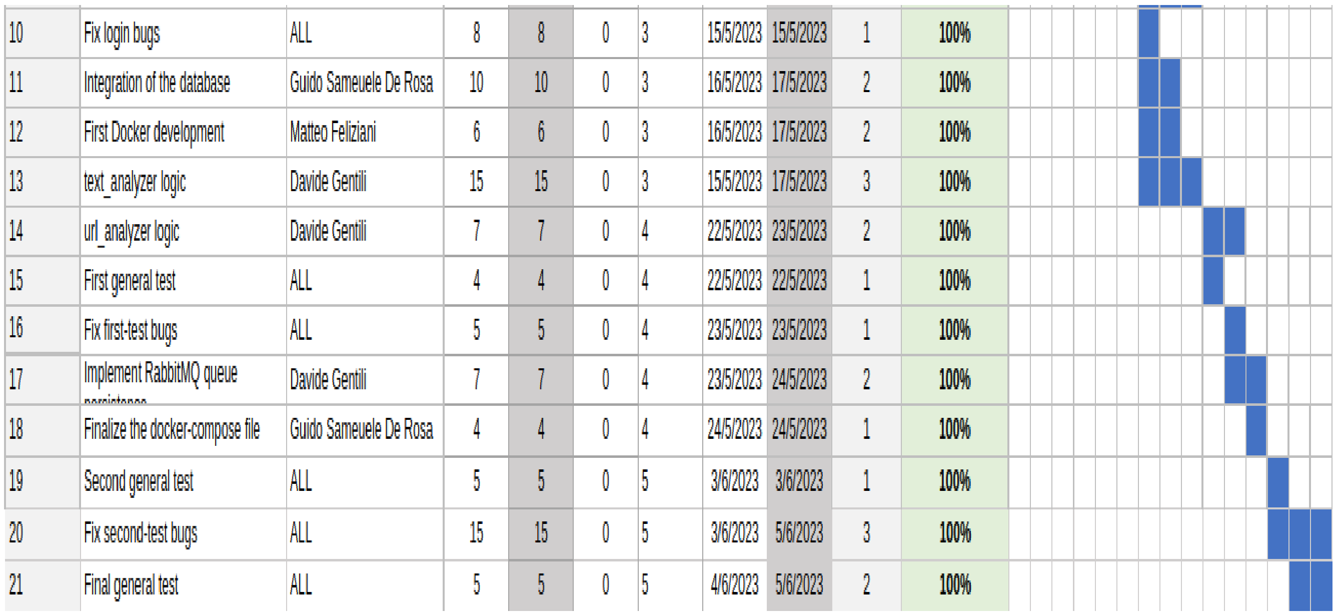
The analyzers services are each other independent, and both replicated to guarantee high availability and fault tolerance. Each replica type receives the requests submitted by the customers from a load balancer sitting in front of them, in order to avoid high load on a single replica without making the customer service to implement the round robin. With this strategy, it could be possible to deploy even more replicas of the same service only by changing some configuration. Each microservice has its own database when needed, and in the case of url-analyzer they also access an instance of Redis for fast response to already processed artifacts.

8 SCRUM

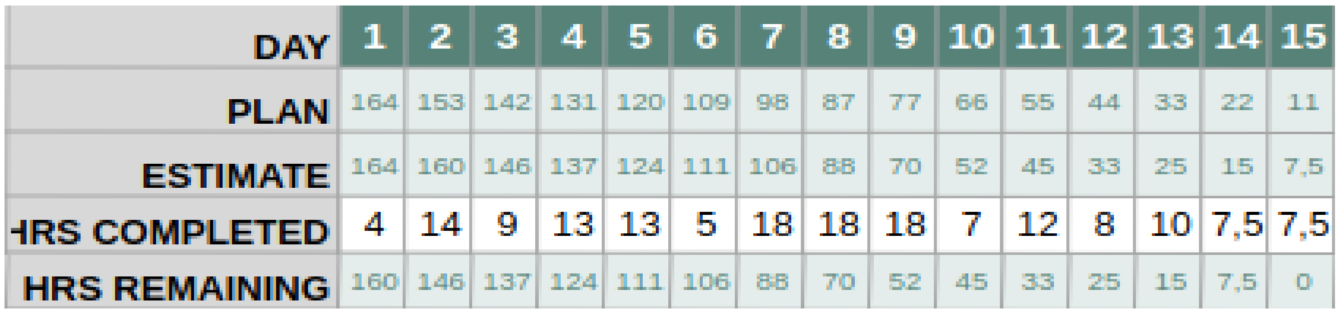
In this section, it is possible to analyze how the overall team-work was divided among the five available periods of time, called *sprint*. We considered a sprint as a 3-day work period, with approximately three hours of work per day. The tasks to be accomplished during the sprints were selected dynamically through team meetings. There was no specific criteria for selecting the tasks; primarily, a task was chosen based on the personal evaluations of the responsible person for that task.



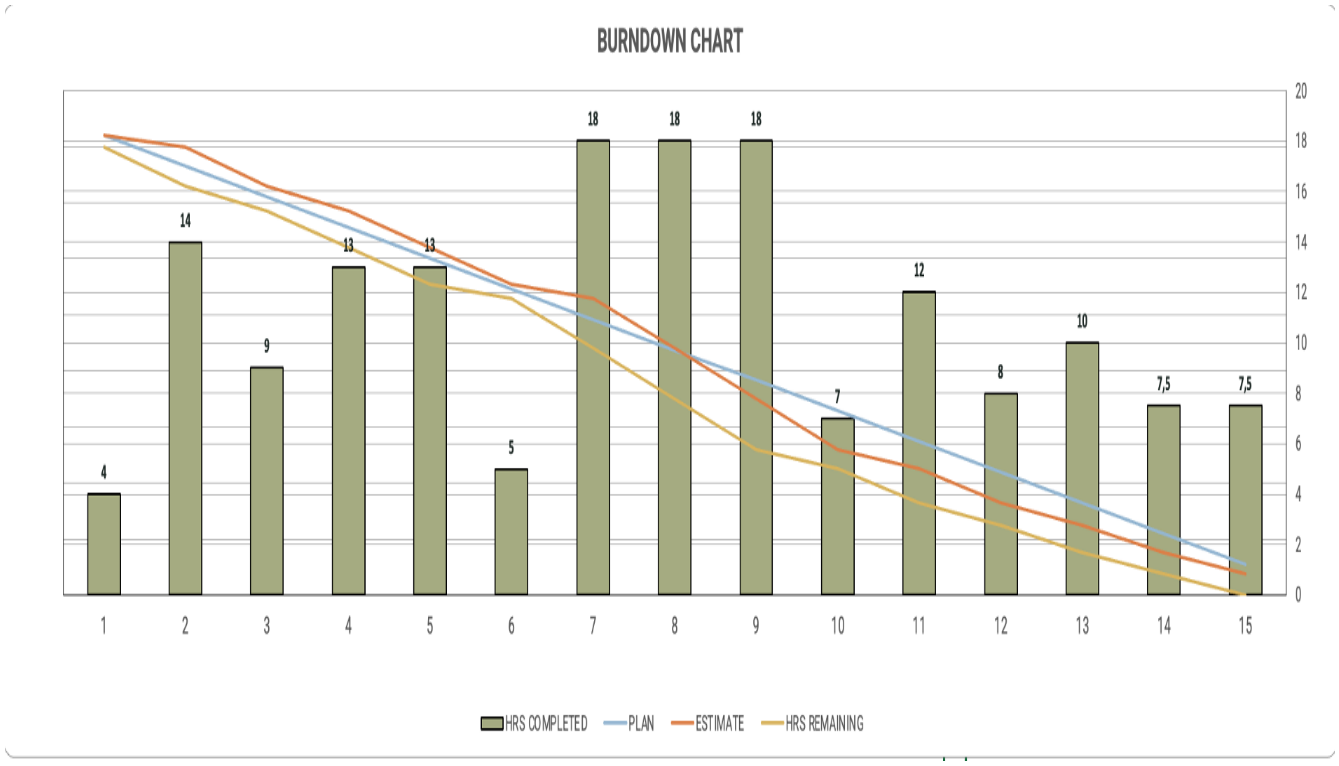
SPRINTS		WEEK 1			WEEK 2			WEEK 3			WEEK 4			WEEK 5		
	SPRINT 1	■	■	■												
	SPRINT 2				■	■	■									
	SPRINT 3							■	■	■						
	SPRINT 4										■	■	■			
	SPRINT 5													■	■	■



BREAKDOWN STRUCTURE	TASK TITLE	TASK OWNER	AMOUNT OF WORK IN HOURS			SPRINT	START DATE	END DATE	DURATION	PCT OF TASK COMPLETE	WEEK 1			WEEK 2			WEEK 3			WEEK 4			WEEK 5		
			ESTIMATE	COMPLETED	REMAINING						M	T	W	M	T	W	M	T	W	M	T	W	M	T	W
1	Schema infrastructure	ALL	8	8	0	1	26/4/2023	27/4/2023	2	100%	■	■													
2	Interface API 'url_analyzer'	Matteo Feliziani	8	8	0	1	27/4/2023	4/28/2023	2	100%		■	■												
3	Database Inizialization	Guido Sameuele De Rosa	6	6	0	1	27/4/2023	4/27/2023	1	100%		■													
4	User front-end	Simone Gennenzi	5	5	0	1	4/28/2023	4/28/2023	1	100%			■												
5	Admin front-end	Simone Gennenzi	5	5	0	2	4/29/2023	4/29/2023	1	100%				■											
6	Interface API 'text_analyzer'	Matteo Feliziani	8	8	0	2	4/29/2023	4/30/2023	2	100%				■	■										
7	Create RabbitMQ queue user-admin	Davide Gentili	8	8	0	2	4/29/2023	4/30/2023	2	100%				■	■										
8	Login functionality - User	Simone Gennenzi	10	10	0	2	30/4/2023	1/5/2023	2	100%					■	■									
9	Login functionality - Admin	Simone Gennenzi	15	15	0	3	15/5/2023	17/5/2023	3	100%							■	■	■						
10	Fix login bugs	ALL	8	8	0	3	15/5/2023	15/5/2023	1	100%							■								
11	Integration of the database	Guido Sameuele De Rosa	10	10	0	3	16/5/2023	17/5/2023	2	100%							■	■							
12	First Docker development	Matteo Feliziani	6	6	0	3	16/5/2023	17/5/2023	2	100%							■	■							
13	text_analyzer logic	Davide Gentili	15	15	0	3	15/5/2023	17/5/2023	3	100%							■	■	■						
14	url_analyzer logic	Davide Gentili	7	7	0	4	22/5/2023	23/5/2023	2	100%										■	■				
15	First general test	ALL	4	4	0	4	22/5/2023	22/5/2023	1	100%										■					
16	Fix first-test bugs	ALL	5	5	0	4	23/5/2023	23/5/2023	1	100%											■				
17	Implement RabbitMQ queue	Davide Gentili	7	7	0	4	23/5/2023	24/5/2023	2	100%											■	■			
18	Finalize the docker-compose file	Guido Sameuele De Rosa	4	4	0	4	24/5/2023	24/5/2023	1	100%												■			
19	Second general test	ALL	5	5	0	5	3/6/2023	3/6/2023	1	100%													■		
20	Fix second-test bugs	ALL	15	15	0	5	3/6/2023	5/6/2023	3	100%													■	■	■
21	Final general test	ALL	5	5	0	5	4/6/2023	5/6/2023	2	100%														■	■



DAY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PLAN	164	153	142	131	120	109	98	87	77	66	55	44	33	22	11
ESTIMATE	164	160	146	137	124	111	106	88	70	52	45	33	25	15	7,5
IRS COMPLETED	4	14	9	13	13	5	18	18	18	7	12	8	10	7,5	7,5
HRS REMAINING	160	146	137	124	111	106	88	70	52	45	33	25	15	7,5	0



8.1 Release Backlog

PRIORITY	SPRINT	FUNCTIONALITY	TASK TITLE	TASK DESCRIPTION	TASK OWNER	HOURS ESTIMATED	STATUS
3	1	General Infrastructure	Schema infrastructure	Draw the blocks-schema representing the infrastructure of the application	ALL	8	Completed
2	1	Microservices	Interface API url_analyzer	Develop the API interface of the url_analyzer microservice	Matteo Feliziani	8	Completed
1	1	User - side	User front-end	Create the web interfaces for the user	Simone Gennenzi	6	Completed
2	1	Microservices	Database initialization	Initialize the database to store emails and results of the analyses	Guido Samuele De Rosa	5	Completed
1	2	Admin - Side	Admin Front-end	Create the web interfaces for the admin	Simone Gennenzi	6	Completed
2	2	Microservices	Interface API text_analyzer	Develop the API interfaces of the text_analyzer microservice	Matteo Feliziani	8	Completed
2	2	General Infrastructure	Create RabbitMQ queue user-admin	Create the RabbitMQ queue to allow the admin to see the email's user analyses	Davide Gentili	8	Completed
3	2	User - Side	Login functionality - User	Build the login functionality to allow the user to login and sign up to the application	Simone Gennenzi	15	Completed
3	3	Admin - Side	Login functionality - Admin	Build the login functionality to allow the admin to login and sign up to the application	Simone Gennenzi	10	Completed
1	3	General Infrastructure	Fix login bugs	Fix bugs discovered in the previous tasks	ALL	8	Completed
1	3	General Infrastructure	Integration of the database	Integrate the database into the application in order to store the emails of the users and the results of the analyses	Guido Samuele De Rosa	10	Completed
3	3	General Infrastructure	First Docker development	Compose the Docker infrastructure of the application	Matteo Feliziani	6	Completed
3	3	Microservices	text_analyzer logic	Implement the logic of the 'text_analyzer' based on ML algorithm	Davide Gentili	15	Completed
2	4	Microservices	url_analyzer logic	Implement the logic of the 'url_analyzer' through external requests	Davide Gentili	7	Completed
2	4	Test	First general test	Test the overall functionalities of the application	ALL	4	Completed
1	4	Test	Fix first-test bugs	Fix bugs discovered in the previous tasks	ALL	5	Completed
1	4	General Infrastructure	Implement RabbitMQ queue persistence	Make the RabbitMQ queue from user to admin persistence	Davide Gentili	7	Completed
2	5	General Infrastructure	Finalize docker-compose file	Finalize the Docker Compose file	Guido Samuele De Rosa	4	Completed
2	5	Test	Second general test	Test the overall functionalities of the application for the second time	ALL	5	Completed
1	5	Test	Fix second-test bugs	Fix bugs discovered in the previous tasks	ALL	15	Completed
3	5	Test	Final general test	Last test of the overall functionalities of the application	ALL	5	Completed

9 Code

The code of the application is available at the following link:
https://github.com/felzmatt/BeSafeEmail
To run the application: clone the repository and exec the command “docker-compose up” inside the main folder.