Distribution and Integration Technologies

An enterprise distributed system

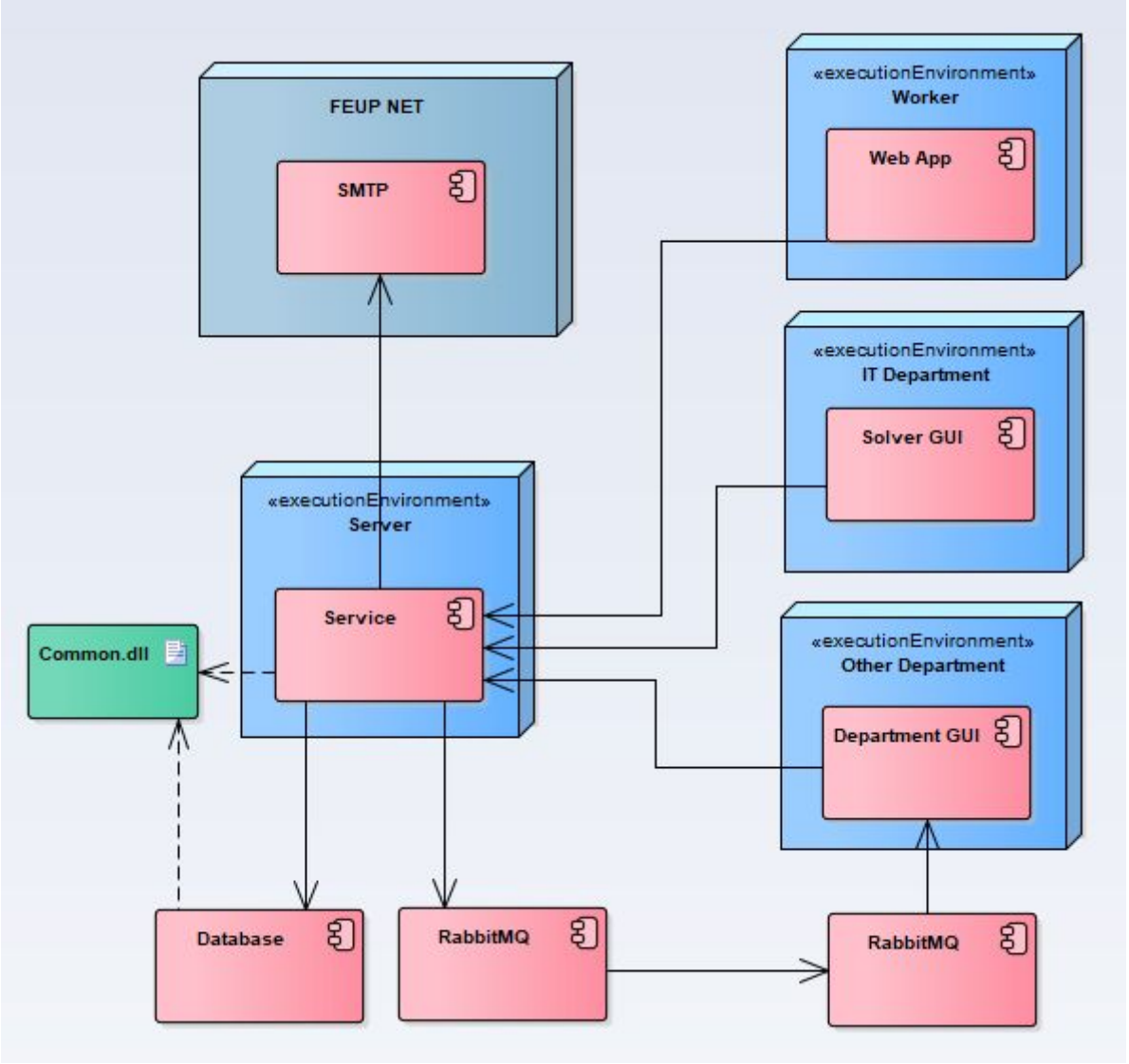
4th year of the Master in Informatics and Computer Engineering

Diogo Duque - up201406274 - up201406274@fe.up.pt
José Carlos Coutinho - up201404293 - up201404293@fe.up.pt

Introduction

This assignment, proposed during the Distribution and Integrated Technologies course, challenged us to create a “trouble ticket” system that could belong to a company. This system should have 3 applications: a Web app for the workers who want to submit trouble tickets; a GUI app for the solvers from the IT department; a GUI app for a specialized solver, who will receive secondary questions from a IT department solver.

Architecture



Features

- Register a solver
- Get all registered solvers
- Register a user
- Get all registered users
- Add trouble ticket
- Get worker's tickets
- Get unassigned tickets
- Assign trouble ticket to solver
- Get solver's assigned tickets (and related secondary questions)
- Associate a secondary question to a ticket
- Get unanswered secondary questions
- Answer a secondary question
- Answer (and solve) the trouble ticket
- Automatically send an email to the worker when a ticket is solved

Technologies

For the backend, we used .NET with WCF to implement a Service, and SQLite for the database where we keep all our information stored.

For the frontend, we used ReactJS in all the apps as it allowed us to recycle a lot of code among them, along with being a easy web development framework. For the 2 GUI apps, Electron was also used, as it works like a desktop application that can render "Web code".

For the message queues, we used RabbitMQ as it provides integration for multiple programming languages, including C# (backend) and JavaScript (frontend).

To send an email, we had to use a predefined account that will send the email from the FEUP servers (up201404293).

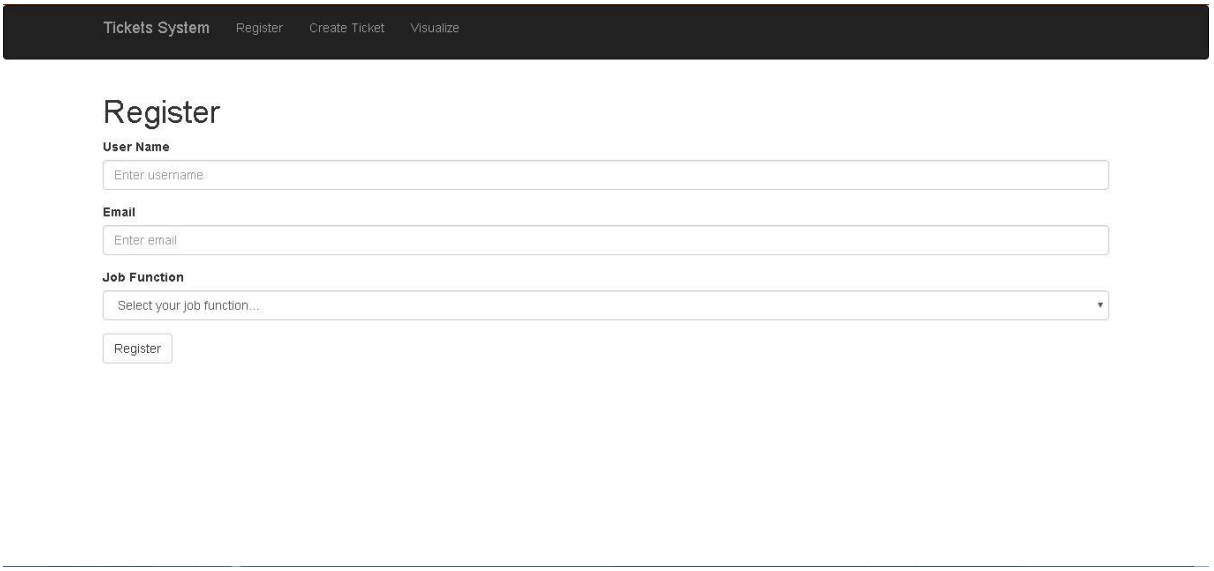
Testing

We did not use an automated test tool/framework. Instead, we used "trial and error". For testing the Service, we made HTTP requests using Postman. For testing the database, we performed direct SQLite queries on it; for testing the frontend apps, we used console logs to keep track of the variables.

User Guide

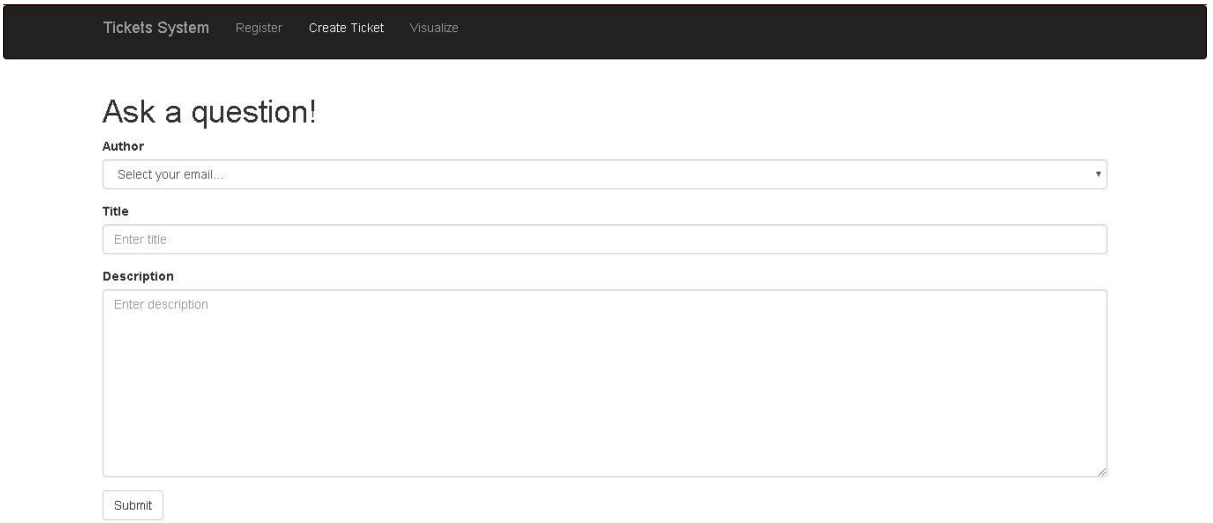
For installing and running instructions, please see the Readme.

Welcome to our Tickets System! Let’s start with the page you’ll be greeted when you try to register as a worker/solver.

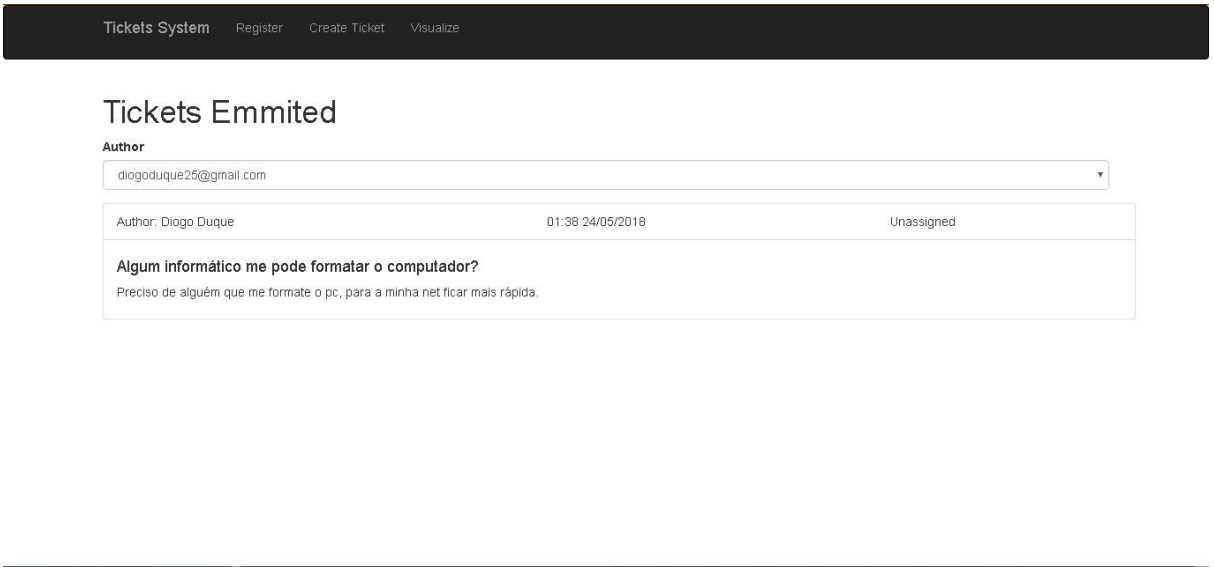


After register, your username will be available from the dropdown on top of the pages you’ll be using.

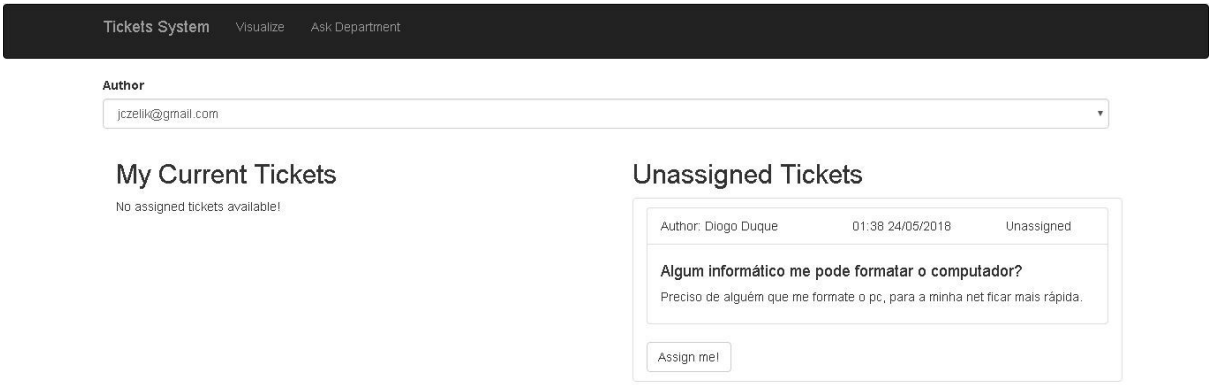
Here we can see how, as a user, we can submit a ticket.



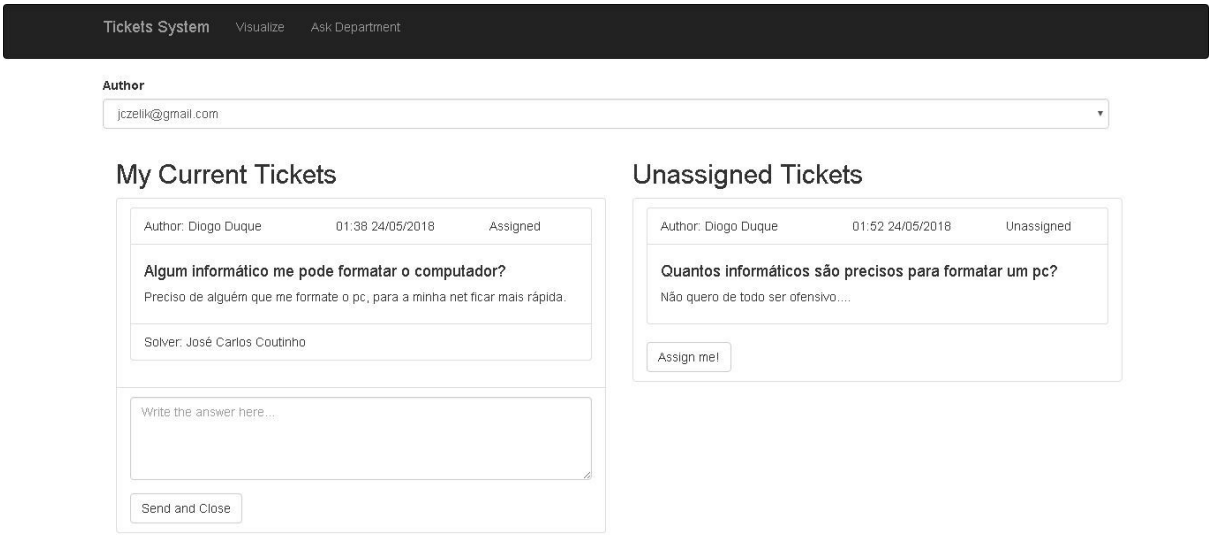
After a ticket is submitted, you can track it (and other tickets you submitted) on this page.



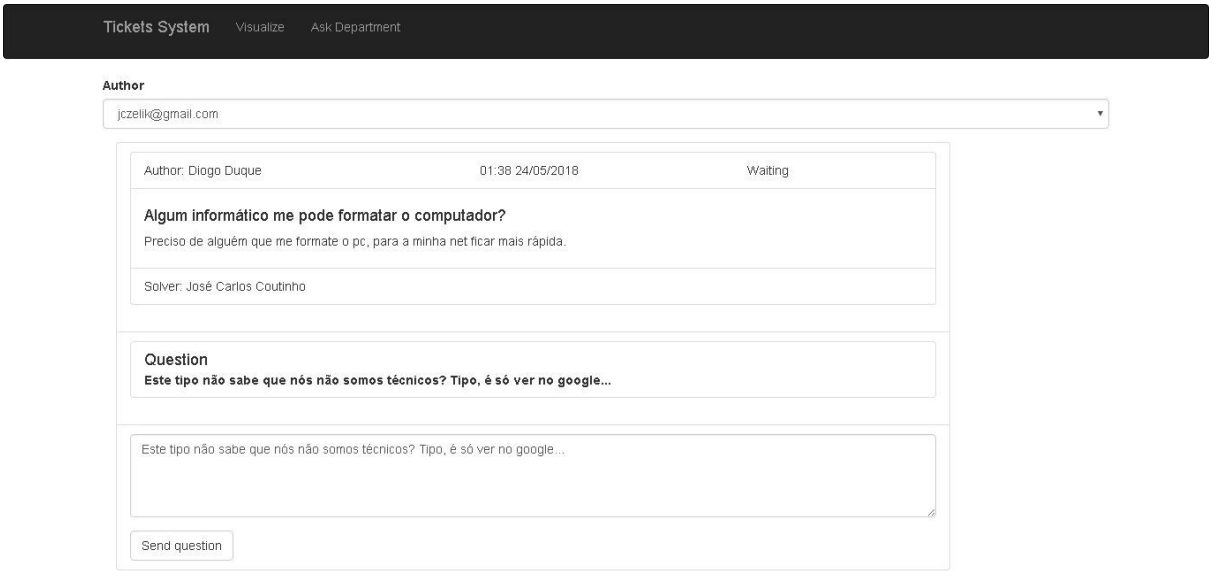
Now moving on to the Solver GUI, on the first page we can see every unassigned tickets and all tickets assigned to the current solver, as well as their status and questions assigned to it.



To assign a ticket to the current solver, click the button below the unassigned ticket. It will move that ticket to “My Current Tickets”. Now a text box should appear below the ticket to answer (by email) and close it.

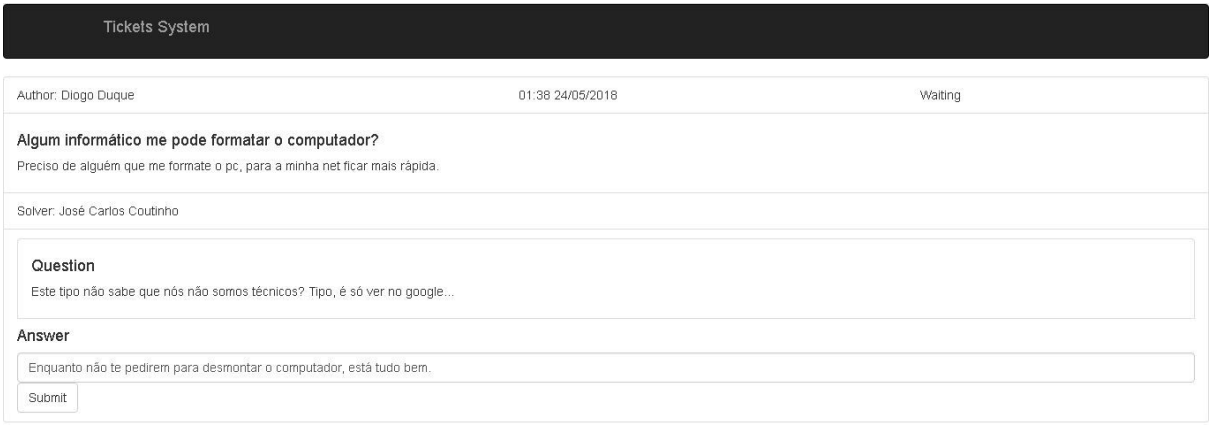


There is also the “Ask Department” page, which allows to ask more specific questions to another Department. You must only make one question at a time.



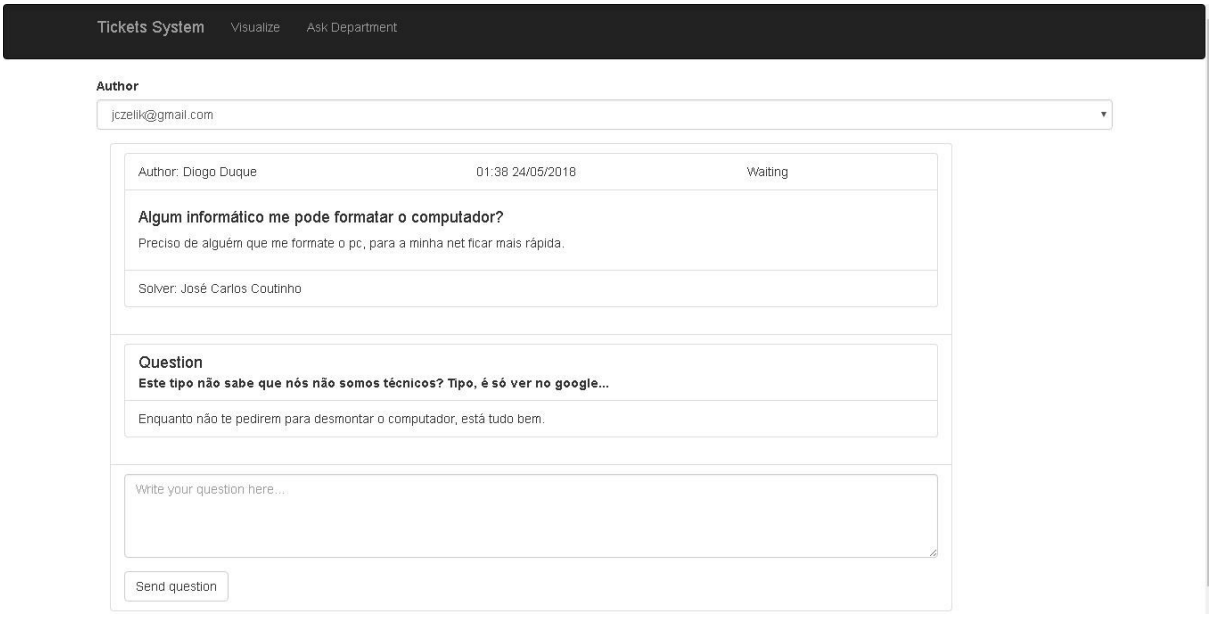
Now let’s see what it’s like in the Department GUI! This GUI has only one page, which shows all the secondary questions needing an answer.

We can select one of them, and start answering it.

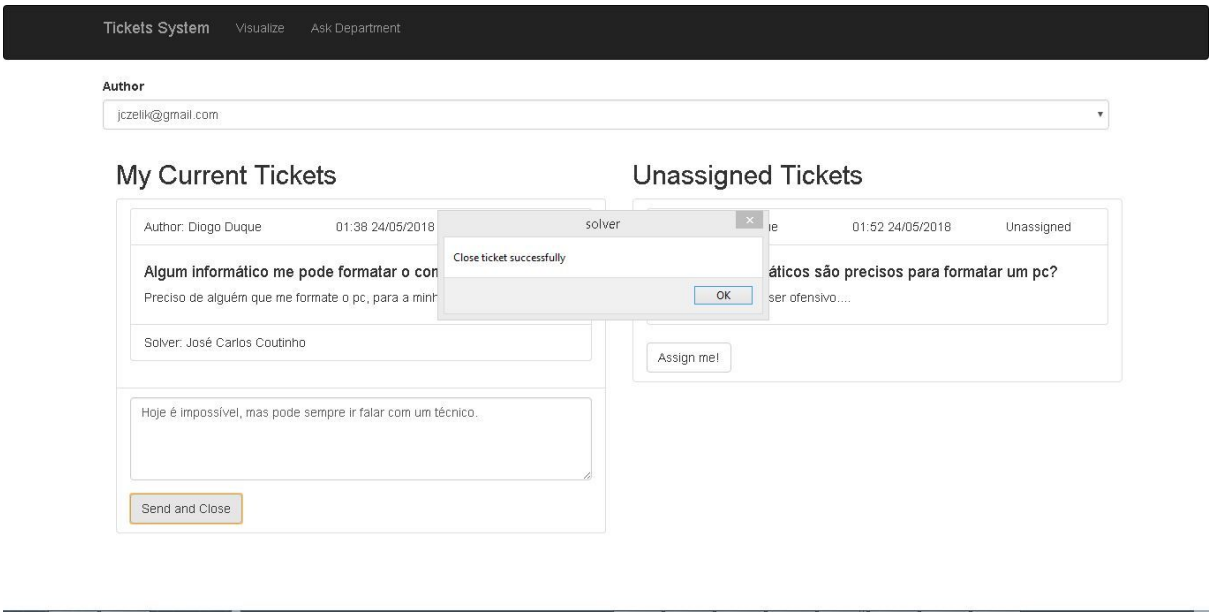


When the answer is finished, you can submit it and it will disappear, as it's no longer needed in this context.

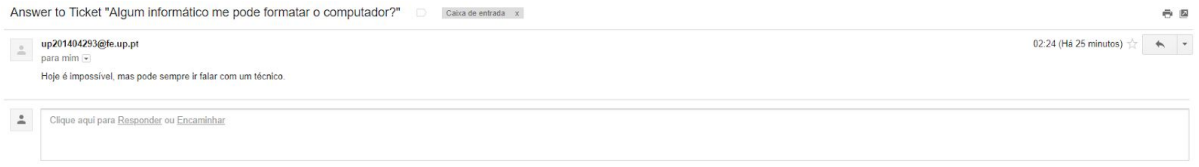
In the Solver GUI, the answer will be shown:



When you know everything you want in order to answer and close the ticket, you can write your answer and click on "Send and Close", and an email will be sent to the author's email address with the answer.



As we can see, the email was received!



Conclusion

This project helped us understand what is a SOA and its applicabilities, as well as strengthen our belief about how helpful message queues can really be.

Though it's already a robust system, further improvements on this work would regard security concerns about validation and login. Even if that wasn't the point of this assignment, it was something we'd like to improve.