

Relatório

Tecnologias de Distribuição e Integração

An enterprise distributed system

Mestrado Integrado em Engenharia Informática e
Computação

Maio 2020

Amadeu Pereira up201605646@fe.up.pt

Nuno Lopes up201605337@fe.up.pt

Conteúdo

| | | |
|----------|------------------------------|-----------|
| 1 | Introdução | 2 |
| 2 | Arquitetura | 3 |
| 2.1 | Server | 3 |
| 2.2 | Web App | 5 |
| 2.3 | Solver | 5 |
| 2.4 | Department | 5 |
| 3 | Testes Funcionais | 7 |
| 4 | Modo de Funcionamento | 8 |
| 5 | Instruções | 14 |
| 6 | Conclusão | 15 |
| 7 | Recursos | 16 |
| 7.1 | Bibliografia | 16 |
| 7.2 | Software utilizado | 16 |

1 Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Tecnologias de Distribuição e Integração pertencente ao Mestrado Integrado em Engenharia Informática e Computação. O projeto consiste no desenvolvimento de um serviço interno para resolver problemas dos *workers* de uma organização através da submissão de um *trouble ticket*.

Este sistema distribuído foi desenvolvido tendo por base os princípios de uma Arquitetura Orientada a Serviços (SOA)

No decorrer deste relatório será descrito em pormenor a arquitetura (funcionalidades, módulos e interações), os testes funcionais feitos e a representação gráfica, com capturas de ecrãs, dos principais modos de funcionamento.

2 Arquitetura

Nesta secção vamos abordar a arquitetura adotada no desenvolvimento deste projeto. O projeto encontra-se dividido em 4 módulos que comunicam entre si através de uma API REST e/ou através de uma *asynchronous queue* para garantir a persistência de pedidos feitos a serviços que não se encontrem disponíveis.

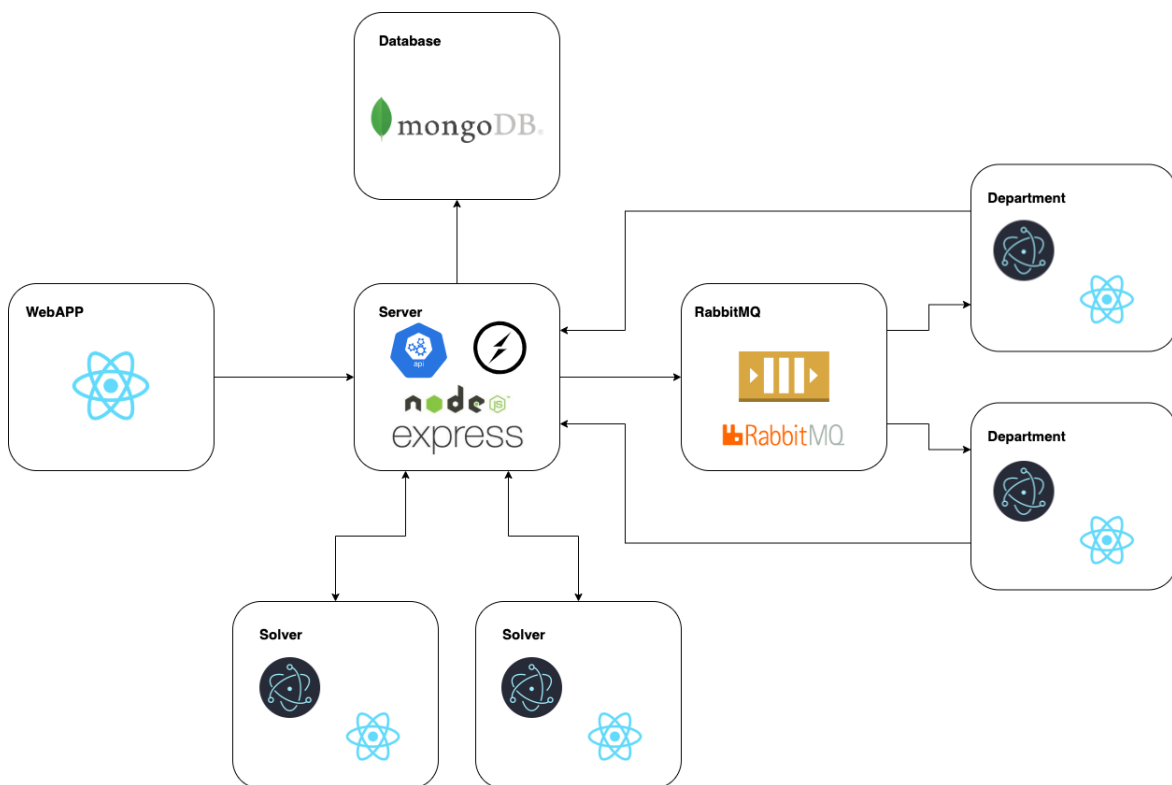


Figura 1: Arquitetura

2.1 Server

O **Server** é o módulo principal deste sistema. Apenas este módulo consegue aceder à base de dados da aplicação, por isso, qualquer outro módulo que lhe queira aceder tem que comunicar com o servidor através da REST API que o mesmo fornece.

Endpoints da API:

- [GET] /api/ticket/:id - Devolve o *ticket* com o id pedido

- [GET] /api/ticket - Devolve todos os *tickets*. Pode ser filtrado se for adicionada uma *query*
- [POST] /api/ticket - Cria um novo *ticket*
- [PUT] /api/ticket/:id - Atualiza o *ticket* com o id fornecido
- [PUT] /api/ticket/:id/solve - Resolve o *ticket* com o id fornecido
- [PUT] /api/ticket/:id/assign - Atribui o *ticket* com o id fornecido a um *worker*
- [GET] /api/ticket/:id/secondary - Devolve todas as *secondary questions* do *ticket* com o id fornecido
- [GET] /api/secondary/:id - Devolve a *secondary question* com o id pedido
- [GET] /api/secondary - Devolve todas as *secondary question*. Pode ser filtrado se for adicionada uma *query*
- [POST] /api/secondary - Cria uma nova *secondary question* de um *ticket*
- [PUT] /api/secondary/:id/solve - Resolve a *secondary question* com o id fornecido
- [PUT] /api/secondary/:id - Atualiza a *secondary question* com o id fornecido

Sempre que uma *secondary question* for submetida o servidor está encarregue de a mandar para a *queue*, na qual o departamento respetivo está à escuta ou, caso não esteja disponível, poderá receber quando o estiver.

O servidor também fornece disponibiliza um *socket* permitindo uma comunicação bi-direcional entre um *solver* e o *server*. Esta abordagem torna possível uma comunicação baseada em eventos que são emitidos pelo servidor para todos os *solvers* conectados quando há uma atualização no estado de um *ticket*.

Quando é recebida informação sobre um *ticket* que foi resolvido o servidor envia um email à pessoa que submeteu sobre a resolução do mesmo.

2.2 Web App

O módulo da *Web Application* fornece aos *workers* uma interface onde eles conseguem, de uma forma intuitiva e fácil, submeter os seus *tickets* e acompanhar o estado de progresso dos mesmos.

Isto é feito através de *requests* à API do servidor para se criar um *ticket* novo e ir buscar os *tickets* filtrados através do nome do autor.

2.3 Solver

Este módulo é uma *GUI* onde os *solvers* conseguem entrar para realizar uma de duas acções: dar *assign* a um *ticket* ou resolver um *ticket*.

No caso de querer dar *assign* a *tickers* ele consegue ver todos os *tickets* que estão *unassigned* e atribuir-se aos mesmos. No caso de querer resolver um *ticket* que está lhe está atribuído, ele pode responder diretamente ao *ticket* ou pode enviar uma mensagem secundária a um *department* que, eventualmente, lhe vai responder.

Todas as trocas de informação são feitas através de pedidos ao servidor, sendo que, quando o *solver* faz uma pergunta a um departamento o servidor envia essa mensagem para a *message queue* onde os departamentos estão a escutar.

Ao mesmo tempo conecta-se ao *socket* do servidor de modo a receber eventos sobre quando um *ticket* foi criado, quando outro *solver* deu *assign* a um *ticket* e quando uma pergunta secundária a um departamento foi respondida. Desta forma os dados são atualizados de forma dinâmica e é notificado quando for respondido.

2.4 Department

O *Department* fornece uma *GUI* onde um departamento entra para receber todas as perguntas que lhe são direccionadas e responder-lhes.

Todas as perguntas lhe são enviadas através da *asynchronous message queue*

onde, seguidamente, são internamente persistidas.

Ao responder a uma *secondary question* a sua resposta é enviada ao servidor através de um pedido ao mesmo onde os *solvers* serão notificados.

3 Testes Funcionais

Foram realizados vários testes ao longo do desenvolvimento deste projeto.

Primeiramente foram-se fazendo testes a cada componente individual de forma a garantir que tinha todas as funcionalidades pretendidas. No servidor também foi necessário certificar que os requisitos do serviço não eram violados (ex: estarem duas pessoas *assigned* ao mesmo *ticket*, atualização do estado do *ticket* para o devido).

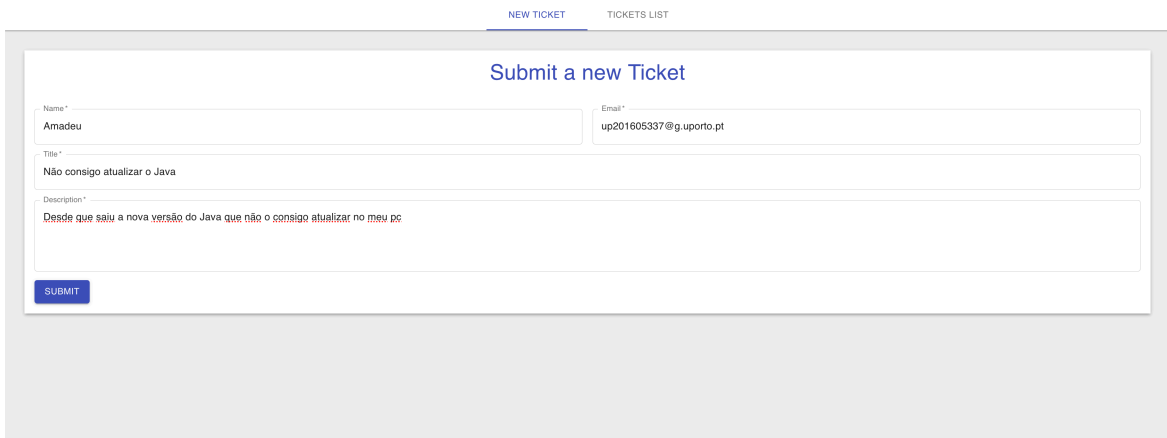
Também tivemos que fazer vários testes a vários serviços de bases de dados e de *queues* para escolher-mos os que melhor se adaptavam ao nosso projeto.

Numa iteração mais final do projeto foi-se testando a ligação entre componentes. Assim garantimos que todos os *requests* feitos ao servidor eram enviados da forma pretendida e que a resposta fosse a esperada. Que a trocas mensagens através da *message queue* fosse de forma assíncrona e que eram recebidas pelo *department* correto.

Também fizemos testes à atualização das interfaces gráficas em tempo real despoletadas através de eventos enviados pelo servidor. Foi necessário certificar que quando um *ticket* é criado ou é atribuído a alguém esse *ticket* aparece ou desaparece, respetivamente, da lista de *tickets* a que um *solver* pode dar *assign*. Garantimos também que quando um *solver* tem uma resposta à sua *secondary question* este recebia uma notificação.

4 Modo de Funcionamento

Instruções para correr tanto todos os módulos podem ser encontradas na secção 5. Depois dos serviços da base de dados, da *queue* e do *server* estarem a funcionar é possível iniciar a Web App e a GUI de *solvers* e Departments. Quando a Web App é aberta aparece um formulário onde um *worker* pode submeter um novo ticket. Para tal este tem de se identificar com o nome e email e escolher um titulo e uma descrição para o ticket.



NEW TICKET TICKETS LIST

Submit a new Ticket

Name *
Amadeu

Email *
up201605337@g.uporto.pt

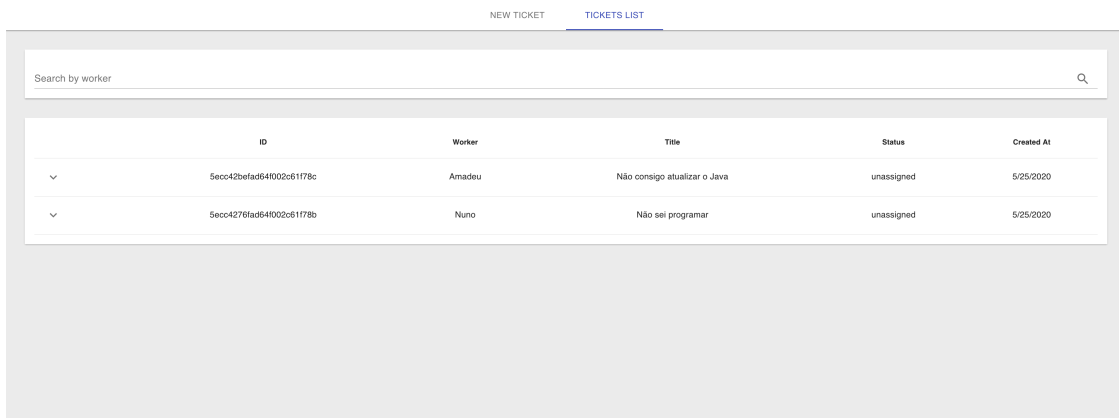
Title *
Não consigo atualizar o Java

Description *
Desde que saiu a nova versão do Java que não o consigo atualizar no meu pc

SUBMIT

Figura 2: Ecrã da Web App para submeter um ticket

Em qualquer altura o *worker* pode utilizar a Web App para verificar o estado dos seus *tickets* bem como o estado dos *tickets* de outros *workers*. Aqui é possível ver a quem os *tickets* estão atribuídos, caso estejam, e também ver as respostas dadas pelos *solvers*.

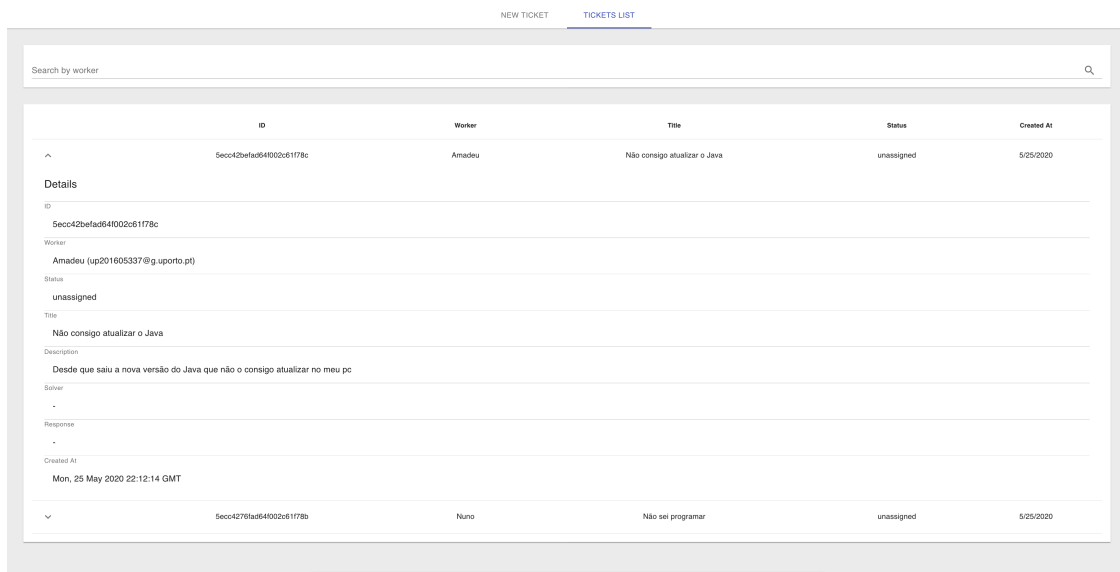


NEW TICKET TICKETS LIST

Search by worker

| | ID | Worker | Title | Status | Created At |
|---|--------------------------|--------|------------------------------|------------|------------|
| ▼ | 5ecc42befad64f002c61178c | Amadeu | Não consigo atualizar o Java | unassigned | 5/25/2020 |
| ▼ | 5ecc4276fad64f002c61178b | Nuno | Não sei programar | unassigned | 5/25/2020 |

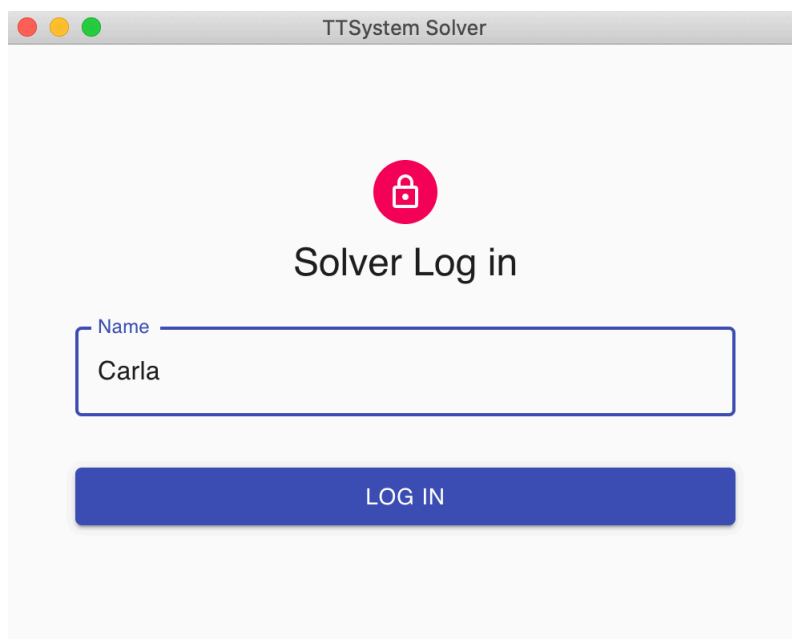
Figura 3: Ecrã da Web App para listar os tickets




| ID | Worker | Title | Status | Created At |
|----------------------------------------------------------------------------|--------|------------------------------|------------|------------|
| 5ecc42befad64f002c61178c | Amadeu | Não consigo atualizar o Java | unassigned | 5/25/2020 |
| Details | | | | |
| ID | | | | |
| 5ecc42befad64f002c61178c | | | | |
| Worker | | | | |
| Amadeu (up201605337@g.uporto.pt) | | | | |
| Status | | | | |
| unassigned | | | | |
| Title | | | | |
| Não consigo atualizar o Java | | | | |
| Description | | | | |
| Desde que saiu a nova versão do Java que não o consigo atualizar no meu pc | | | | |
| Solver | | | | |
| - | | | | |
| Response | | | | |
| - | | | | |
| Created At | | | | |
| Mon, 25 May 2020 22:12:14 GMT | | | | |
| 5ecc4278fad64f002c61178b | Nuno | Não sei programar | unassigned | 5/25/2020 |

Figura 4: Ecrã da Web App para listar em detalhe um ticket

Quando um *solver* inicia a sua aplicação GUI tem de se identificar para poder visualizar os seus *tickets* e os *tickets* que não estão atribuídos a nenhum solver.



TTSystem Solver



Solver Log in

Name

LOG IN

Figura 5: Ecrã de autenticação de um solver

Após um *solver* se identificar uma janela é aberta onde é possível visualizar todos os *tickets* que ainda não estão atribuídos a ninguém, ou seja *tickets* que se encontram no estado *unassigned*. Para cada *ticket* o *solver* pode-se atribuir a si mesmo para o resolver sendo que, quando isto acontece, todos os outros *solvers* online são notificados que houve alterações na lista de *tickets* em tempo real.

Todos os *tickets* atribuídos a um *solver*, que se encontram no estado *assigned*, *waiting* ou *answered*, podem ser encontrados na secção My Tickets. Nesta página o *solver* pode abrir cada um dos *tickets* para obter mais detalhes sobre cada um.

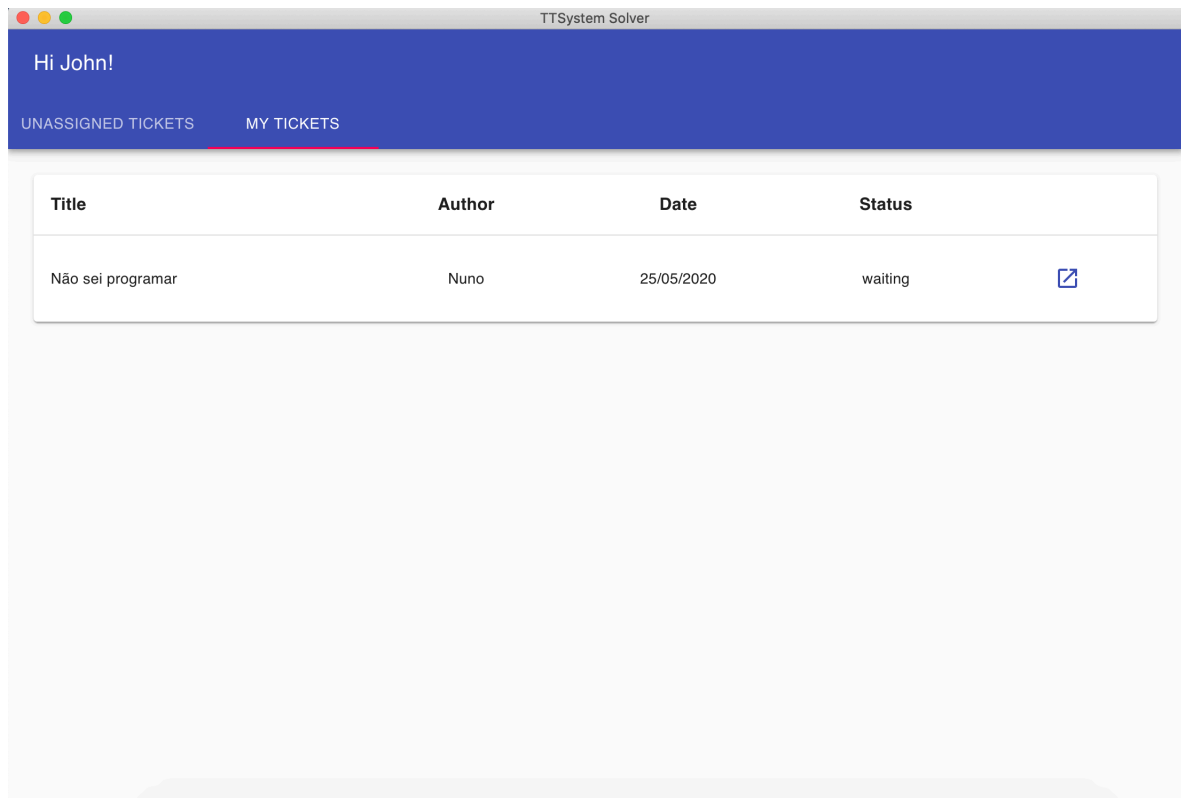


Figura 7: Ecrã de um *solver* que lista os *tickets* atribuídos a si

Dentro da janela de um *ticket* o *solver* pode ver as informações de um *ticket* com maior detalhe, nomeadamente qual foi o *worker* a submeter o ticket, o título e descrição do *ticket* e todas as questões do *solver* feitas a um *department* para este ticket. Para cada um destes campos o *solver* dispõe de uma referência temporal. Além disso o *solver* pode resolver um *ticket* escrevendo uma resposta para a questão do ticket, sendo que o *ticket* passa ao estado *solved* e é removido da GUI, ou abrir uma nova janela para efetuar uma questão a um *department*, sendo que o *ticket* passa ao estado *waiting* até obter uma resposta do *department*.

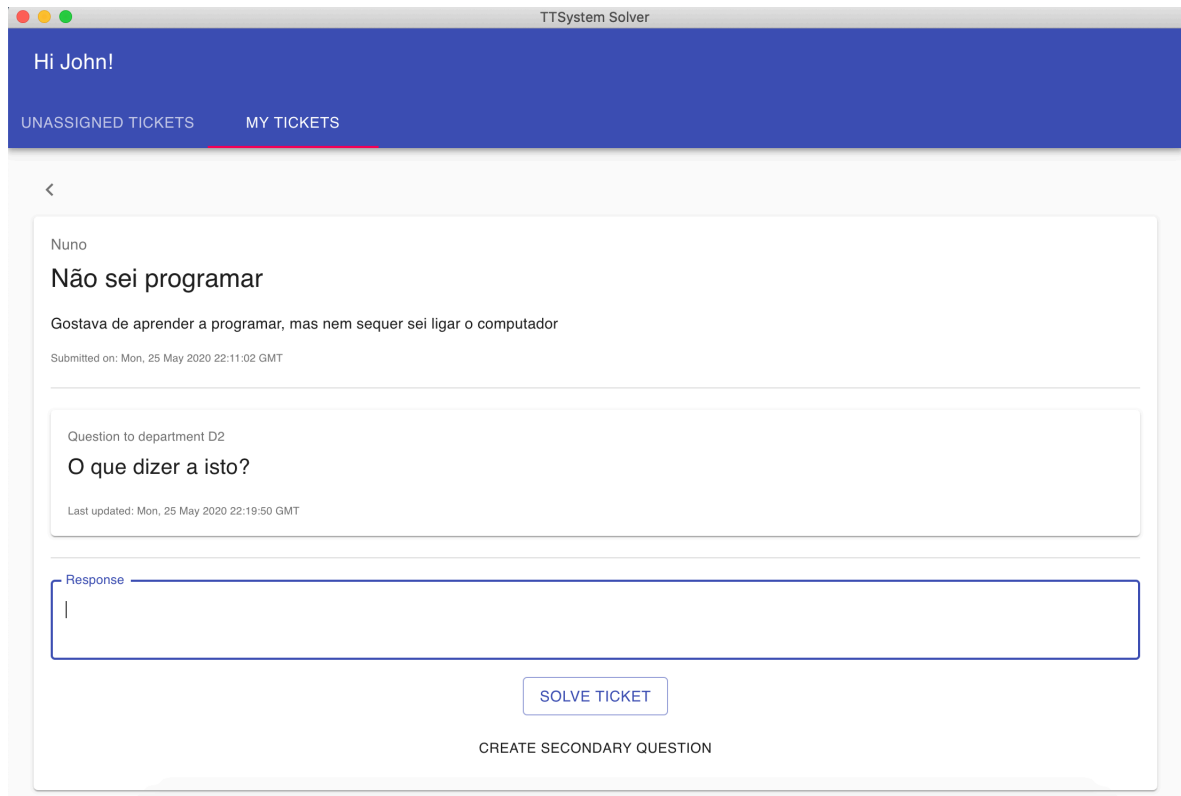


Figura 8: Ecrã de um *solver* que mostra em detalhe um *ticket* atribuído a si

Quando um *solver* resolve um *ticket* é enviado um email com a seguinte estrutura ao *worker* que submeteu o ticket.

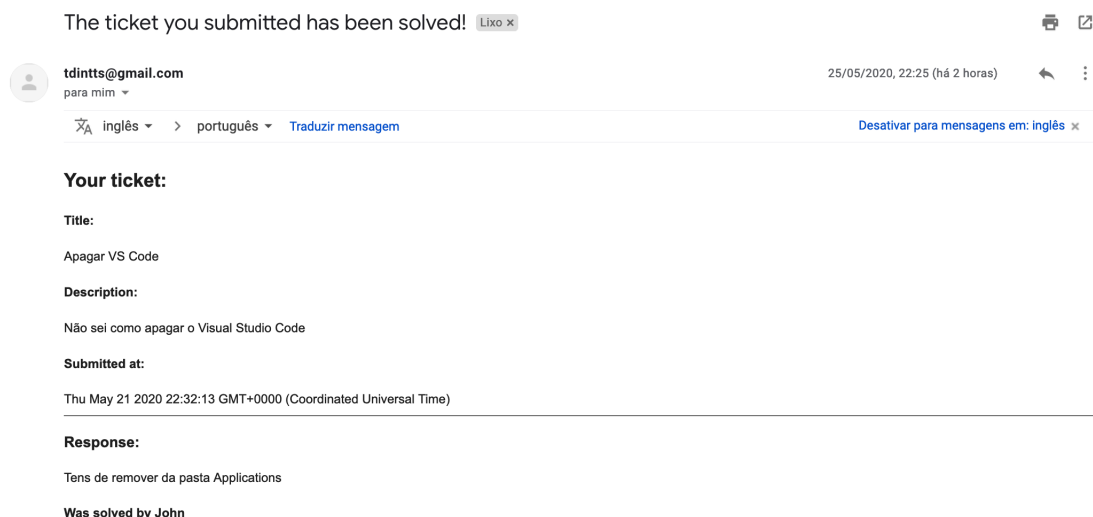
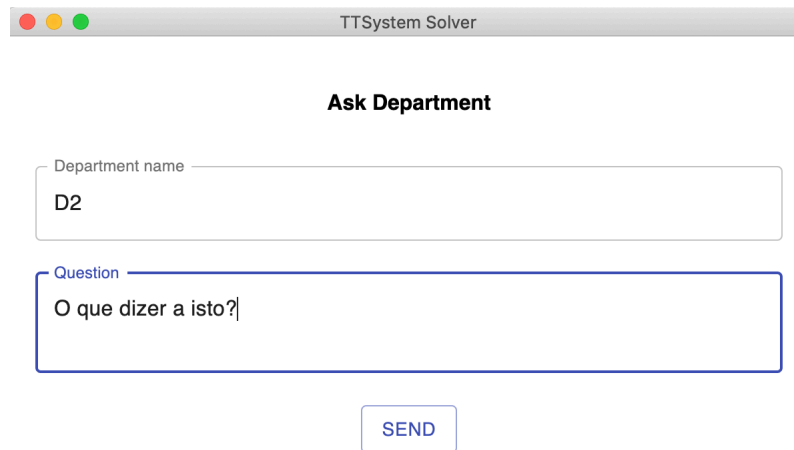


Figura 9: Email enviado a um *worker* quando um *ticket* é resolvido por um *solver*

Para um *solver* efetuar uma questão a um *department* ele tem de identificar o *department* ao qual se quer direccionar e escrever a questão a enviar.



TTSystem Solver

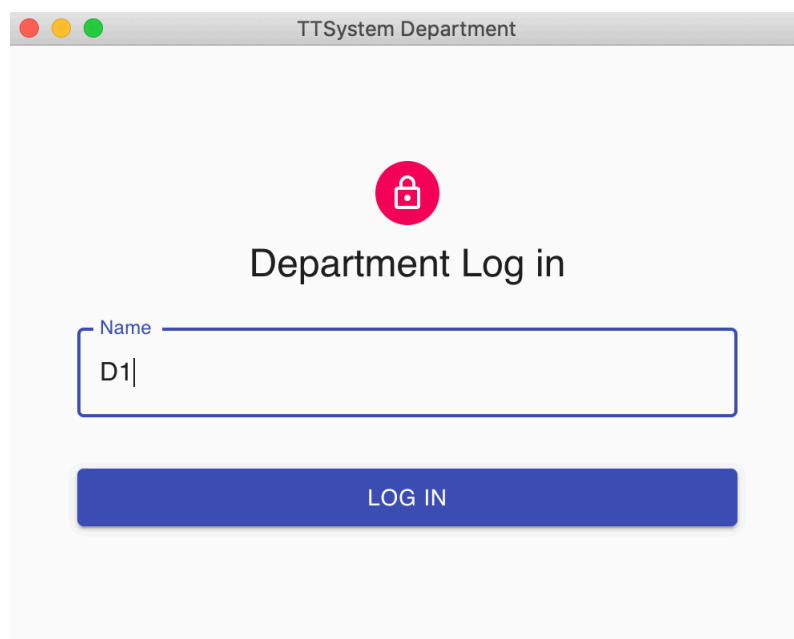
Ask Department

Department name


Question

Figura 10: Ecrã de para um *solver* fazer uma pergunta a um *department*

No caso da GUI de um *department* também, numa primeira interação, é preciso identificar o *department* para receber as mensagens que são direcionadas a si.



TTSystem Department



Department Log in

Name

Figura 11: Ecrã de autenticação de um *department*

Para cada questão efetuada por um *solver* o *department* pode visualizar a questão e também informações sobre o *ticket* que a originou, sendo que no final é possível enviar uma resposta ao *solver*. Quando o *solver* recebe uma resposta de um *department* o estado do *ticket* passa a *answered*.

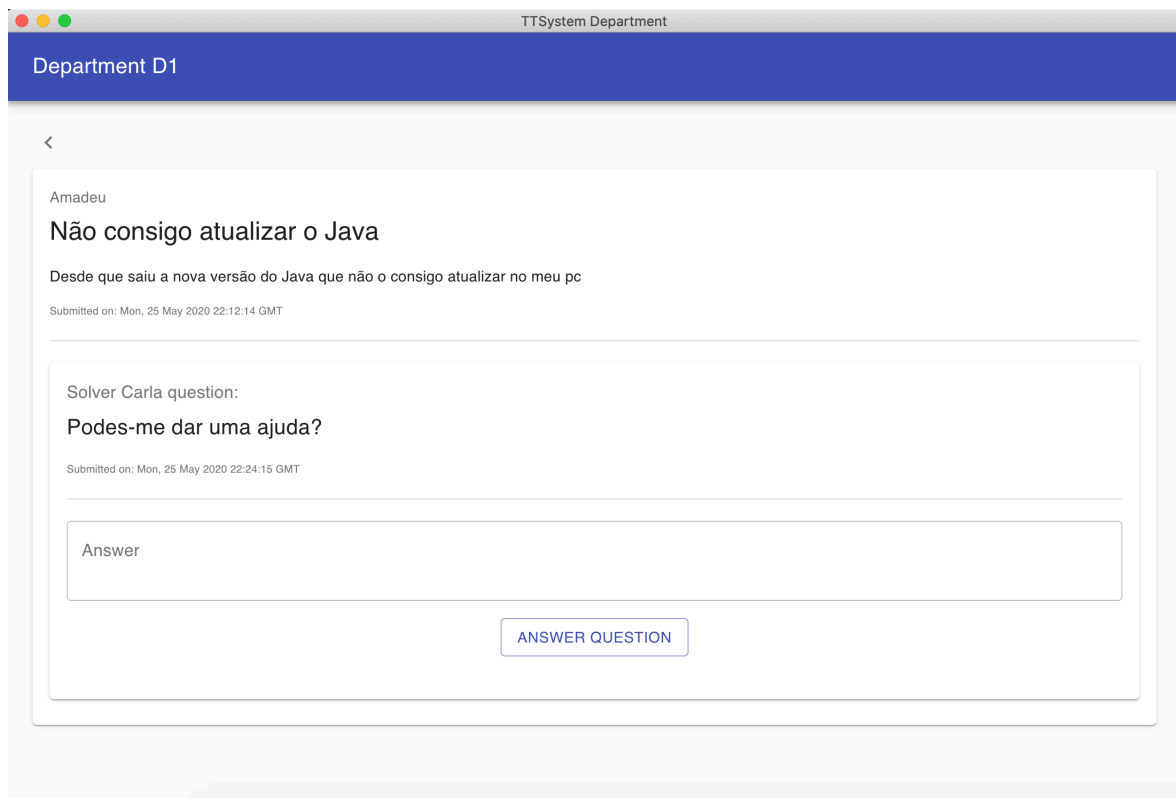


Figura 12: Ecrã de um department que lista em detalhe as perguntas colocadas por um Solver

5 Instruções

Para a execução do projeto é necessário que o computador possua Docker, Docker Compose e Node.

Antes de se iniciar qualquer um dos módulos é necessário gerar executáveis para as interfaces gráficas do *solver* e do *department* compatíveis com sistema operativo onde o projeto vai ser lançado. Dentro das pastas *department/* e *solver/*, individualmente, é preciso executar o seguinte comando:

```
$ npm install && npm run build
```

Este comando gera um executável para o S.O. onde o comando foi corrido dentro das pastas *department/dist/* e *solver/dist/*.

Antes de se iniciarem as GUIs é preciso correr o docker-compose que se encontra na pasta do código fonte do projeto, isto pode ser feito através do comando:

```
$ docker-compose up
```

Neste comando são inicializados os serviços da base de dados, a *queue*, o servidor e a Web App. Após estes serviços estarem devidamente iniciados é possível abrir várias GUIs de *solvers* e de *departments*. A base de dados é persistida localmente, sendo que é possível recuperar o sistema mesmo que este pare o seu funcionamento.

6 Conclusão

Com este projeto o grupo aprendeu a utilizar a Service Oriented Architecture (SOA) para desenvolver uma aplicação distribuída, bem como novas tecnologias às quais o grupo nunca tinha utilizado antes.

O grupo acredita que a aplicação foi desenvolvida com sucesso cumprindo todos os requisitos propostos para o projeto e que consequentemente este seria um produto viável a ser utilizado no mercado como um sistema de *Trouble Tickets*, faltando unicamente uma melhoria no sistema de autenticação sendo que este não era um foco deste projeto.

Com isto, podemos concluir que projeto foi concluído com sucesso sendo que todas as funcionalidades foram implementadas.

7 Recursos

7.1 Bibliografia

- [1] Página da UC de Tecnologias de Distribuição e Integração.
Acedido em maio de 2020.
<https://paginas.fe.up.pt/~apm/TDIN/>

7.2 Software utilizado

- [1] Docker Desktop for Mac. <https://www.docker.com/>
- [2] Docker Compose. <https://docs.docker.com/compose/>
- [3] NodeJS. <https://nodejs.org/>
- [4] ExpressJS. <https://expressjs.com/>
- [5] Electron. <https://www.electronjs.org/>
- [6] RabbitMQ. <https://www.rabbitmq.com/>
- [7] MongoDB. <https://www.mongodb.com/>
- [8] React. <https://reactjs.org/>