A close up of a logo

Description automatically generated

**Relatório**

**Comunicação de Dados**

**Alunos**

João Apresentação (21152)

Pedro Simões (21140)

Augusto Pereira (21136)

**Docente**

Miguel Lopes

**Licenciatura Engenharia Sistemas Informáticos**

Barcelos, junho de 2022

Indice

[2. Índice de ilustrações 3](#_Toc105353024)

[3. Resumo 5](#_Toc105353025)

[4. Base de dados 6](#_Toc105353026)

[5. Back-End 7](#_Toc105353027)

[5.1. Models 7](#_Toc105353028)

[5.2. Controllers 7](#_Toc105353029)

[5.2.1. Utilizador 7](#_Toc105353030)

[5.2.2. Máquina 16](#_Toc105353031)

[5.2.3. Conexao 21](#_Toc105353032)

[5.3. Funcionalidades para plano de produção 26](#_Toc105353033)

[5.3.1. Automático 26](#_Toc105353034)

[5.3.2. Manual 28](#_Toc105353035)

[6. Front-End 29](#_Toc105353036)

[7. Bibliografia 30](#_Toc105353037)

# Índice de ilustrações

[Base de Dados 1 - Estrutura 6](#_Toc105352182)

[Back-end 1 - Get Utilizadores 7](#_Toc105352194)

[Back-end 2 - Hide Password 8](#_Toc105352195)

[Back-end 3 - Get Utilizador por token 9](#_Toc105352196)

[Back-end 4 - Get Utilizador 9](#_Toc105352197)

[Back-end 5 - Login 10](#_Toc105352198)

[Back-end 6 - Verifica Conta 10](#_Toc105352199)

[Back-end 7 - Cria token admin 11](#_Toc105352200)

[Back-end 8 - Cria token user 11](#_Toc105352201)

[Back-end 9 - Post utilizador 12](#_Toc105352202)

[Back-end 10 - Patch utilizador 13](#_Toc105352203)

[Back-end 11 - Recuperar password 14](#_Toc105352204)

[Back-end 12 - Arquivar utilizador 15](#_Toc105352205)

[Back-end 13 - Delete utilizador 16](#_Toc105352206)

[Back-end 14 - Get maquina 16](#_Toc105352207)

[Back-end 15 - Get maquina by id 17](#_Toc105352208)

[Back-end 16 - Get maquina por jobop 17](#_Toc105352209)

[Back-end 17 - Post maquina 18](#_Toc105352210)

[Back-end 18 - Patch maquina 19](#_Toc105352211)

[Back-end 19 - Delete maquina 20](#_Toc105352212)

[Back-end 20 - Get Conexões ativas 21](#_Toc105352213)

[Back-end 21 - Get conexões inativas 21](#_Toc105352214)

[Back-end 22 - Get conexões by user id 22](#_Toc105352215)

[Back-end 23 - Get simulação por user id 22](#_Toc105352216)

[Back-end 24 - Post conexões 23](#_Toc105352217)

[Back-end 25 - Patch conexão 24](#_Toc105352218)

[Back-end 26 - Delete conexão admin 24](#_Toc105352219)

[Back-end 27 - Delete conexão admin e funcionario 25](#_Toc105352220)

[Back-end 28 - Planner 26](#_Toc105352221)

[Back-end 29 - Algoritmo Escalonamento 27](#_Toc105352222)

[Back-end 30 - Planner manual 28](file:///C:\Users\toxic\Documents\GitHub\API_REST_Escalonamento\Relatorio\Relatorio_CD.docx#_Toc105352223)

[Front-end 1 - Página de registo 29](#_Toc105352224)

[Front-end 2 - Página da simulação 29](#_Toc105352225)

# Resumo

Em suma, o trabalho prático da cadeira de Comunicação de Dados teve como propósito a montagem de uma API capaz de correr serviços web com determinadas funcionalidades. Era pretendido a implementação para o problema do escalonamento FJSSP (flexible job shop problem).

De uma forma mais descritiva, a existência de várias simulações, constituídas por Jobs, que por sua vez constituídos por operações, e cada operação será desempenhada por uma máquina com uma determinada duração. O escalonamento tem por objetivo encontrar um plano de produção mais eficiente ao nível de duração.

Esta aplicação oferece de uma forma geral os seguintes serviços:

* Gestão de Simulações;
* Tabela de Produção;
* Plano de Produção;
  + Automatizado;
  + Manual;
* Gestão de Utilizadores;

Este programa oferece a possibilidade de construção de simulações, á qual poderá se fazer download, simular manualmente e automaticamente, além de muitas outras funcionalidades.

Este terá duas categorias de permissões:

* Funcionário;
* Administrador;

O Administrador tem todas as funcionalidades iguais ás do Funcionário, mas este ainda pode gerir os funcionários e as máquinas.

Para a resolução da mesma recorreu-se ao uso:

* linguagem C# para o Back-End realizado no Visual Studio 19;
* Html, Css e TypeScript para o Front-End elaborado em Visual Studio Code;
* Base de dados em Microsoft SQL Server.
* Testes de request realizados em Postman;

# Base de dados

Para construção da base de dados, tal como o referido acima, construído em Microsoft SQL Server levou a seguinte estrutura:

Uma imagem com mesa

Descrição gerada automaticamente

Base de Dados - Estrutura

Dentre as 3 tabelas existe a do utilizador e da máquina que são independentes da existência de dados na tabela das conexões. A tabela das conexões representa cada vez que um utilizador cria uma simulação com respetivas associações. Esta tabela será usada para a grande maioria das funcionalidades e apresentações.

# Back-End

Para a criação do Back-End realizado em Visual Studio 19, foi criado uma ligação á base de dados através de .NET (Entity-Framework).

Este contem Controllers e os Models.

## Models

Os models são criados através da linkagem com a base de dados e contem todos os atributos e respetiva tabela.

## Controllers

Os Controllers são criados para se poder realizar os requests e poderem ser executados para o Front-End. Desta forma seguem-se os Controllers criados e suas respetivas funcionalidades:

## Utilizador

A função que se segue, realiza um Get para devolver a lista dos utilizadores através da base de dados, e esta só será autorizada para um user que tenha token de administrador.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Get Utilizadores

Esta ainda usa uma função auxiliar para esconder passwords:

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Hide Password

Temos uma funcionalidade Get, para buscar um utilizador através da token apanhada no Front-End através do login realizado. Este verifica através do seu mail e este é retornado.

Uma imagem com texto

Descrição gerada automaticamente

Back-end 3 - Get Utilizador por token

Para quando se necessitar dos dados de um utilizador em específico, é retornado através de um Get que leva por parâmetro o id do user em questão.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Get Utilizador

Para o login do utilizador é feito um Post para autenticar este através das credenciais inseridas. Este verifica o utilizador, e no caso deste ser verificado, é gerada a token de sessão dependendo da sua rule.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Login

A funções auxiliares para a token e verificação da conta são as seguintes:Uma imagem com texto

Descrição gerada automaticamente

Back-end - Verifica Conta

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Cria token admin

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Cria token user

Para o utilizador ser criado é usado um POST do objeto Utilizador passado por parâmetro. Todos os seus dados são armazenados, a sua password é criptografada, e uma role é lhe atribuído. No fim o utilizador é armazenado na lista.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Post utilizador

No caso de os dados do utilizador serem alterados é realizado Patch. É enviado por parâmetro o id do utilizador e um objeto com os novos dados a serem alterados. Este recebe uma verificação de autorizações, os dados são alterados e atualizados.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Patch utilizador

Para o caso de se alterar a password, a função que se segue cumpre. Este recebe o id do utilizador e o objeto utilizador que contem a nova password. A nova password é criptografa e armazenada.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Recuperar password

Caso o utilizador queira remover a sua conta, este terá o estado da sua conta alterado para “Inativo”, ou seja, este terá a sua conta arquivada. É feito uma verificação da sua autorização, pois este só poderá remover a sua própria conta.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Arquivar utilizador

Outro delete existente é da remoção de um utilizador através do admin. Este terá autorização para remover os seus funcionários.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Delete utilizador

## Máquina

De forma a conseguir listar as máquinas todas com estado “Ativo” é utilizado um Get:

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Get maquina

Para a listagem de uma máquina em específico é passado por parâmetro o seu id:

Uma imagem com texto

Descrição gerada automaticamente

Back-end 15 - Get maquina by id

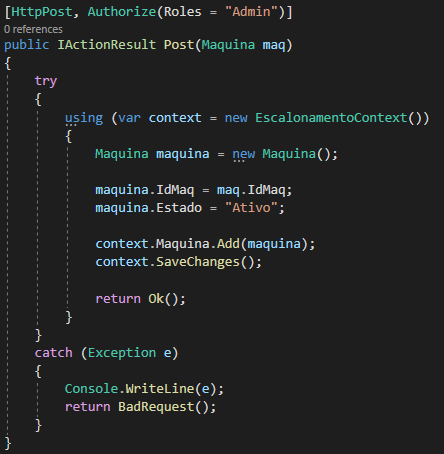
Para a obtenção de uma máquina e sua duração em uma operação, de um job, de uma simulação é enviado um objeto do tipo Conexao que contem os ids do user, simulação, job e operação. Pesquisando depois na base de dados este retorna a conexão que contem a máquina e a duração a serem apresentadas ao utilizador.

Uma imagem com texto

Descrição gerada automaticamente

Back-end 16 - Get maquina por jobop

Inserção de uma máquina pelo Admin na base de dados:



Back-end - Post maquina

O método que se segue atualiza os dados de uma máquina, mais especificamente o seu estado:Uma imagem com texto

Descrição gerada automaticamente

Back-end - Patch maquina

Para a remoção de uma máquina pelo Admin, este envia por parâmetro o id da máquina e altera o seu estado para “Inativo”. No caso desta ser desativada, todas as conexões que a contenham, o id máquina e a duração serão convertidas a null. Mais tarde o user pode verificar que tal ocorreu e substituir por máquinas e durações novas.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Delete maquina

## Conexao

Para buscar as conexões temos os seguintes métodos:

Este retorna para o Admin as conexões cujo estado sejam verdade (por outras palavras ativa para uso).

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Get Conexões ativas

Este retorna todas as conexões falsas (ou inativas) para o Admin:

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Get conexões inativas

Para o caso de o utilizador querer ver as suas conexões, este através do seu id, consegue retornar todas que lhe sejam pertencidas. O admin também as pode consultar:

Uma imagem com texto

Descrição gerada automaticamente

Back-end 22 - Get conexões by user id

No caso de se querer obter os dados de uma simulação pertencente a um utilizador, esta função envia os dados dessa simulação em específico:Uma imagem com texto

Descrição gerada automaticamente

Back-end 23 - Get simulação por user id

Na criação de uma nova conexão é feito um Post dos dados passados por um objeto do tipo Conexão:

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Post conexões

Para o caso de o utilizador querer alterar o id de máquina e duração de uma conexão em específico, este terá de enviar todos os ids de user, sim, job e operação que o identifiquem e os novos valores a substituir:

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Patch conexão

Para o caso de remover uma simulação de uma conexão, este recebe por parâmetros id do user e id da simulação e onde estes forem encontrados na base de dados é lhes atribuído o estado falso (ou inativo).

Para o Admin:Uma imagem com texto

Descrição gerada automaticamente

Back-end - Delete conexão admin

Para Admin e Funcionário:

Uma imagem com texto

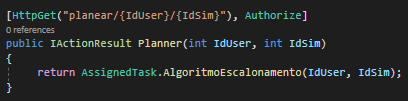
Descrição gerada automaticamente

Back-end - Delete conexão admin e funcionario

## Funcionalidades para plano de produção

## Automático

Para automático, este chama uma função que representa o algoritmo (Encontrado no AssignedTask.cs) por trás do escalonamento e este retorna um objeto que será utilizado no Front-End:



Back-end - Planner

Algoritmo para o automático:

Como uma fase inicial do algoritmo, de forma a adequar a nossa solução, foi feita uma construção de uma lista (allJobs) de listas, em que cada lista representa um job e contem id da máquina e duração. O resto do algoritmo é igual ao fornecido pela google.

Uma imagem com texto

Descrição gerada automaticamente

Back-end - Algoritmo Escalonamento

## Manual

Para o algoritmo manual, visto que neste caso é inserido manualmente os instantes iniciais de cada operação no Front-End, este recebe como parâmetro uma lista de conexões, que contem todas as sequências e respetivo instante inicial dessa sequência (id user, id simulação, id job, id operação, id da máquina, duração e instante inicial). Como verificações dos valores inseridos dos instantes serem corretos para uma boa criação do plano, são feitos os seguintes passos:

* Primeiramente é criada uma estrutura para guardar o estado do plano e uma mensagem/valor para retornar neste método.
* De seguida é criado inicialmente uma variável que vai guardar o valor do tempo de conclusão da última operação da simulação.
* É verificado se alguma operação está a ser executada antes da operação anterior do mesmo job ter terminado.
* Uma imagem com texto

  Descrição gerada automaticamenteÉ verificado também se alguma máquina está a ser usada mais do que uma vez no mesmo intervalo de tempo.

Back-end - Planner manual

# Front-End

O projeto foi criado em angular (v13) composto pelos seguintes componentes:

* Admin – Perfil de administrador
* Funcionários – Gestão de funcionários (Apenas o admin tem acesso)
* Login – Autenticação
* Máquinas – Página que mostra as máquinas (Apenas o admin tem acesso)
* New-Sim – Página para adicionar uma nova operação numa determinada simulação
* Show-Sims – Gestão de simulações e execução de planeamentos manuais e automatizados

Uma imagem com texto, eletrónica, captura de ecrã

Descrição gerada automaticamente

Front-end - Página de registo

Uma imagem com texto

Descrição gerada automaticamente

Front-end - Página da simulação

# Bibliografia

<https://developers.google.com/optimization/scheduling/job_shop> (Google, s.d.)