Regras Para a Elaboração do Relatório de Projeto/Estágio da LEEC

Pedro João De-Francesco Resende Fortuna Assis



Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2020

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores

Candidato: Pedro João De-Francesco Resende Fortuna Assis, Nº 1234567, pfa@isep.ipp.pt

Orientação científica: Pedro João De-Francesco Resende Fortuna Assis, pfa@isep.ipp.pt

Empresa: xxxxxx

Orientador: Pedro João De-Francesco Resende Fortuna Assis, pfa@isep.ipp.pt



Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

7 de julho de 2020

Agradecimentos

Agradecimentos que sejam devidos. Esta secção é opcional!

Resumo

No trabalho efetuado foi proposto que projetasse e implementasse que proporcionasse um melhor controlo das diferentes linhas de produção, pretendia-se que o sistema encurtasse a necessidade de controlo de produção e supervisão, redução do tempo de paragem das linhas, e aumento do controlo de qualidade, com o objetivo de melhorar o rendimento da produção.

O projeto foi dividido em três partes principais, a primeira consistia na conversão do sistema previamente existente para funcionar numa máquina com sistema operativo baseado em Linux.

A segunda parte consistiu na realização de uma interface Web que disponibilizasse, em tempo real, os dados de saída do sistema de controlo de produção para os funcionários e para o supervisor.

A última fase correspondeu á implementação de uma interface gráfica web que permitisse visualizar os dados de forma mais simpática e organizada, que permitisse a sua melhor compreensão por parte dos responsáveis.

Palavras-Chave

Lista, separadas por vírgulas, de palavras, frases, ou acrónimos chave no âmbito do trabalho descrito neste texto.

Abstract

The abstract should summarize the report’s contents, in what concerns the problem identification and/or the formulated hypotheses. The solution, its validation and assessment should also be briefly focused. In all, it should be less than 2 pages in length.

Keywords

We would like to encourage you to list your keywords, key phrases and most relevant acronyms in this section.

Índice

Agradecimentos i

Resumo iii

Abstract v

Índice vii

Índice de Figuras ix

Índice de Tabelas xi

Acrónimos xiii

1. Introdução 1

1.1. Contextualização 3

1.2. Objectivos 4

1.3. Calendarização 4

1.4. Organização do relatório 5

2. Formatos & Estilos 7

2.1. Dimensões da página 8

2.2. Capa 8

2.3. Contracapa 8

2.4. Agradecimentos, resumo e abstract 9

2.5. Índices 9

2.6. Acrónimos 9

2.7. Lemas, proposições e teoremas 10

2.8. Figuras e tabelas 10

2.9. Fórmulas 11

2.10. Extractos de programas 12

2.11. Listas 12

2.12. Notas de rodapé 12

2.13. Referências 13

2.14. Anexos 13

3. Conclusões 15

Referências Documentais 17

Anexo A. Colector baseado em JMS 19

Histórico 21

Índice de Figuras

Figura 1 Arquitetura do OSS 5620 SAM-O [1] 4

Figura 2 Diagrama de estados [7] 11

Índice de Tabelas

Tabela 1 Calendarização do projeto 5

Tabela 2 Descrição dos campos da Notificação de Alarme [7] 11

Acrónimos

|  |  |  |
| --- | --- | --- |
| API | – | Application Programming Interface |
| ASCII | – | American Standard Code for Information Interchange |
| ASN.1 | – | Abstract Syntax Notation - One |
| ASR | – | Alcatel Service Router |
| ATM | – | Asynchronous Transfer Mode |
| CADREDE | – | Sistema de Gestão do Cadastro de Rede da PT |
| COTS | – | Components Of The Shelf |
| CPU | – | Communications Processor Unit |
| CRC | – | Cyclic Redundancy Check |
| CRM | – | Customer Relationship Management |
| CSMA/CD | – | Carrier Sense Multiple Access/Collision Detection |
| ER | – | Elemento de Rede |
| eTOM | – | Enhanced Telecom Operations Model |
| FAB | – | Fulfillment, Assurance & Billing |
| FCAPS | – | Fault, Configuration, Accounting, Performance, Security |
| FCS | – | Frame Check Sequence |
| FIFO | – | First In First Out |

1. Introdução
   1. Contextualização

Este projeto [7] surgiu do desejo de realizar um trabalho no âmbito da gestão de redes baseadas na infraestrutura Simple Network Management Protocol (SNMP) [4][5][8][9], mas que simultaneamente se integrasse na plataforma de alarmes do Sistema de Gestão de Alarmes (SGA) da Portugal Telecom (PT).

Por coincidência, a PT tinha acabado de adquirir (à Alcatel) a nova rede de banda larga Internet Protocol (IP)—Multi Protocol Label Switching (MPLS), e assim surgiu a necessidade de integrar os alarmes gerados por esta infraestrutura (Figura 1) no SGA.



1. Arquitetura do OSS 5620 SAM-O [1]

Dada a pouca experiência, e conhecimento, sobre a aquisição de alarmes para este tipo de tecnologias, foi natural a proposta de incluir a integração, e tratamento, dos alarmes desta rede no âmbito do projeto. Esta inclusão refletiu-se no desenvolvimento de um processo de aquisição dos alarmes gerados e de um programa que formate os relatórios de alarme para o formato (normalizado) da notificação de alarme utilizado no SGA.

* 1. Objetivos

O principal objetivo deste projeto é melhorar a qualidade da supervisão de seis linhas de produção e implementar um sistema web que permita apresentar a informação recolhida durante o processo de produção de forma amigável e organizada ao utilizador. Dada a complexidade do objetivo, sentiu-se a necessidade de o subdividir em múltiplas tarefas mais simples, tais como:

* Estudo do modo de funcionamento e configuração de base de dados MySQL
* A conversão do sistema de recolha de dados para o sistema operativo Linux
* Estudo de algumas ferramentas usadas para desenvolvimento Web (JavaScript, JQuery, Python, …)
* Desenvolvimento de uma interface gráfica para display de dados em tempo real
* Desenvolvimento da aplicação Web para analises estatísticas da informação, e ajuda na tomada de decisões
  1. Calendarização

Para uma melhor gestão do tempo disponível para a realização foi concebida a calendarização da tabela 1, ainda na fase de planeamento. Esta inclui tanto as semanas para aprendizagem de conceitos como para o desenvolvimento efetivo do trabalho. Tal como podemos ver no mês de março foi iniciada a primeira parte do projeto com o estudo da linguagem c++, assim como do sistema já implementado na Indmei, e gradualmente foi feita a conversão para Linux.

O mês de abril deu início á segunda fase o projeto, que corresponde ao estudo de algumas linguagens de programação tal como, python, html, css, e JavaScript. Neste mês foi também desenvolvida a interface gráfica destinada às linhas de produção e supervisores.

A terceira fase do trabalho começou no início do mês de maio e prolongou-se até ao final da terceira semana de junho. Esta correspondeu ao desenvolvimento da aplicação web destinada á apresentação e analise estatística dos dados de produção.

**Tabela 1 Calendarização do projeto**

Uma imagem com computador, branco, armário, portátil

Descrição gerada automaticamente

* 1. Organização do relatório

No primeiro capítulo é feita uma breve introdução ao projeto realizado no âmbito da unidade curricular de Projeto/Estágio do 3º ano da licenciatura em engenharia eletrotécnica e de computadores (LEEC), assim como a contextualização do problema a resolver, calendarização, e apresentados os principais objetivos propostos.

No capítulo seguinte, 2, analisa-se a arquitetura do sistema, onde são apresentadas as diferentes tecnologias utilizadas e são explicadas as interações entre estas. É também, feito o enquadramento destas no sistema em geral.

No capítulo 3, são expostos os principais algoritmos, etapas, e processos utilizados no desenvolvimento do projeto, assim como uma descrição detalhada dos mesmos.

No capítulo seguinte, 4, procede-se á mostra e analise dos resultados obtidos no trabalho.

Por fim, no último capítulo, 5, são retiradas as principais conclusões e apresentadas perspetivas que poderão ser abordadas para desenvolvimentos futuros.

1. Arquitetura do sistema

Neste capítulo é feita a apresentação e descrição das tecnologias utilizadas no desenvolvimento do projeto, assim como a explicação para a escolha das mesmas. Todas as tecnologias utilizadas são gratuitas (open source) mas apresentam muita robustez e, algumas, elevado nível de complexidade. Este aspeto permite reduzir os custos de desenvolvimento do projeto mantendo a sua qualidade.

A componente de recolha de dados já estava implementada e precisou apenas de uma conversão para o seu correto funcionamento em sistema operativo Linux. Esta recolha é feita através de um Arduíno que está constantemente a escrever para a porta serie do PC. Posteriormente o processamento e validação dos dados é realizado por um aplicativo escrito em c++.

Na figura seguinte está representado o diagrama do sistema implementado, onde podemos ver as diferentes tecnologias utilizadas e algumas das suas interações, este foi pensado ainda na fase de planeamento e mais tarde corroborado pela fase de analise dos resultados.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

**Figura 2 Esquema geral do sistema**

Tal como podemos verificar na figura 2, a base de dados MySQL encontrasse no centro do diagrama, tornando evidente a sua importância no sistema. Numa primeira fase os dados recolhidos pelo Arduíno são enviados para a porta serie do PC e posteriormente são processados e validados pelo aplicativo, que é responsável por inserir as entradas de informação na base de dados. Posteriormente a aplicação web comunica com o MySQL para obter os dados relativos á produção, esta comunicação realizasse através de uma application programming interface (API) do Mysql para a linguagem Python.

* 1. MySQL

O MySQL é um sistema de gestão de base de dados relacional (RDBMS) que adota maioritariamente um modelo de cliente-servidor, em que o cliente gera os pedidos e o servidor processa esses pedidos e constrói as respetivas respostas, e a Structured Query Language (SQL). Os comandos SQL dividem-se quatro grandes grupos, Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), e Transaction Control Language (TCL). Neste projeto os grupos mais utilizados são o DDL, uma vez que esta lida com a estrutura e descrição dos dados dentro da base de dados, e o DML que, faz a manipulação dos dados, este inclui os comandos mais conhecidos, tais como SELECT, INSERT, UPDATE, etc..

* 1. Python

Python é uma das linguagens de programação mais interessantes e usadas a nível mundial. De acordo com as tabelas de classificação da TIOBE, python é a terceira linguagem mais popular, ficando apenas atrás de c e java. As principais vantagens desta linguagem em comparação a outras incluem:

* É uma linguagem interpretada, o que significa que não precisa de ser compilada para instruções máquina de baixo nível
* O Python Package Index (PyPI) contêm vários módulos de terceiros que a tornam capaz de interagir com a maioria das outras linguagens e plataformas
* Extensas librarias de suporte: oferece uma grande biblioteca padrão que inclui áreas como protocolos de internet, operações de string, ferramentas de desenvolvimento Web e interfaces de sistema operacional. Muitas tarefas de alto uso já foram escritas na biblioteca padrão o que reduz significativamente o comprimento do código a ser escrito.

[https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/]

Outra vantagem do Python é a sua grande flexibilidade de programação Web, graças a todas as estruturas (frameworks) que tornam o desenvolvimento Web rápido e fácil, como por exemplo Django e Flask. Uma framework é um pacote que contém pacotes e módulos que fornecem cetras funcionalidades que podem ser alteradas de forma seletiva para criar um software específico de aplicação.

1. **Flask**

Uma Web Application Framework é uma coleção de módulos e librarias que possibilitam aos desenvolvedores escrever aplicações com maior nível de abstração em relação a detalhes de baixo nível, tal como protocolos e gestão de threads. [https://pythonbasics.org/what-is-flask-python/]

Flask é uma Web Framework desenvolvida por Armin Roonacher, que lidera uma equipa internacional de entusiastas Python chamados Poocco. Esta é baseada no Wekzeg web server gateway interface (WSGI) kit e no mecanismo de modelo Jinja2. [https://pythonbasics.org/what-is-flask-python/]

![Uma imagem com captura de ecrã, pássaro

Descrição gerada automaticamente]()

**Figura 3 Modelo Jinja2**

O Jinja2 é um mecanismo de modelo para Python usado para criar HyperText Markup Language (HTML), eXtensible Markup Language (XML), ou outros formatos de marcação, que são retornados ao usuário por meio de uma resposta HyperText Transfer Protocol (HTTP). Um modelo contém variáveis que são substituídas pelos valores passados quando o modelo é renderizado, tal como mostra a figura 3. As variáveis são uteis com dados dinâmicos.

Para que a FlaskApp funcione de forma correta é preciso que os ficheiros sigam a organização apresentada na figura 4.

![Uma imagem com mesa

Descrição gerada automaticamente]()

**Figura 4 Estrutura FlaskApp**

A estrutura da aplicação ajuda a manter o código organizado e acessível, por defeito o flask espera que o diretório “templates” esteja na mesma pasta onde o módulo foi criado, no entanto pode ser especificado uma localização diferente através do comando mostrado abaixo.

app = Flask(\_\_name\_\_, template\_folder='../pages/templates')

[https://codeburst.io/jinja-2-explained-in-5-minutes-88548486834e]

* 1. JavaScript

JavaScript é uma linguagem de script que roda do lado do cliente, ou seja, ao invés de rodar remotamente em servidores na internet, o JavaScript tem como característica principal correr os programas localmente, sendo estes executados pelo navegador web. Assim sendo, o JavaScript fornece às páginas web a possibilidade de programação, transformação e processamento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HMTL e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas, o que torna a experiência do utilizador mais interessante e interativa.

JavaScript é uma linguagem de thread única, o que significa que possui uma única Call Stack e, portanto, apenas tem a capacidade de fazer uma coisa de cada vez.

O facto desta linguagem receber e enviar pedidos através da rede para servidores remotos, sem ter a necessidade de recarregar a totalidade da página conforme as entradas do usuário foi um elemento decisivo quanto á sua escolha para este projeto.

[<https://www.educba.com/how-javascript-works/>]

1. **JQuery**

JQuery é uma biblioteca JavaScript rápida, pequena e rica em recursos que simplifica o processo de passagem e manipulação de documentos HTML, eventos, animação e AJAX. Tem uma API fácil de usar que funciona em quase todos os navegadores. [<https://jquery.com/>]

O principal motivo da utilização desta tecnologia no projeto é a simplicidade de execução dos pedidos AJAX (asynchronous JavaScript and XML). Estes pedidos possuem grande importância porque permitem a troca de dados entre o front-end e o back-end sem usar alguma variedade de linguagem no lado do servidor para injetar os dados necessário no JavaScript e sem ter de navegar para uma página nova sempre que for preciso atualizar o conteúdo. [<https://www.quora.com/Why-is-JSON-and-AJAX-important-for-JavaScript?share=1>]

$.ajax({

url: "/getconf",

success: function (result) {

console.log(result.postos)

n\_postos = result.postos

n\_turnos = result.turnos

$('#n\_postos').append(n\_postos)

$('#n\_turnos').append(n\_turnos)

}

})

No estrato de código acima podemos ver a estrutura básica de um pedido AJAX, neste caso a propriedade “url” contém o destino para onde o pedido (request) é enviado. A propriedade “success” permite chamar uma função caso o pedido tenha sido bem-sucedido. Para esta função são passados três argumentos: o conteúdo retornado pelo servidor; o objeto jqXHR (XMLHttpRequest); e uma string que descreve o estado do pedido. [<https://api.jquery.com/jQuery.ajax/>]

1. **JSON (JavaScript Object Notation)**

JSON é uma formatação leve de troca de dados, é fácil de ler e escrever para o programador, e fácil de interpretar e gerar para máquinas. Está baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro – 1999. JSON é em formato de texto e completamente independente da linguagem, pois usa convenções que são idênticas às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Python, Perl e muitas outras. Estas propriedades fazem com que o JSON seja um formato ideal para troca de dados.

O formato do conteúdo pode mudar de acordo com a estrutura dos dados. As duas estruturas mais usadas são:

* Um objeto, que é um conjunto desordenado de pares nome/valor. O objeto começa com uma chave de abertura (“{”) e termina com uma chave de fechamento (“}”), cada nome é seguido por “:” e os pares nome/valor são separados por virgula, tal como pode ser observado na figura 5. Este tipo de estrutura foi o mais utilizado no desenvolvimento do projeto.
* Um array, sequencia de dados listados de forma ordenada numa lista. Não existe o mapeamento nome/valor.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

**Figura 5 Estrutura de um objeto JSON**

1. Desenvolvimento do sistema

Como referido anteriormente, a realização deste projeto foi escalonada em três fases distintas. Numa primeira fase procedeu-se á conversão do código do sistema de aquisição de dados para poder ser executado numa máquina com sistema operativo Linux. Na segunda fase foi desenvolvida uma aplicação gráfica para visualização dos dados de produção, número de peças produzidas e tempo de paragem, em tempo real. Por último, foi desenvolvida uma aplicação Web para efetuar cálculos estatísticos e permitir uma visualização mais amigável dos dados adquiridos ao longo do tempo.

1. **. Aquisição de dados**

A aquisição dos dados é feita através de um Arduíno que comunica com o computador através de uma ligação USB. Sempre que um posto produz uma nova peça o Arduíno envia para o PC um número inteiro correspondente ao número do posto, por exemplo, se o posto número 6 produzir uma peça é enviado o número 6 pela conexão USB. Posteriormente o aplicativo escrito em c++ deteta que existe informação nova na porta série e procede á sua interpretação e validação, caso a informação esteja correta os dados são atualizados e inseridos numa base de dados, tal como podemos verificar no diagrama da figura 6.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

**Figura 6 Estrutura de um objeto JSON**

Nesta fase do projeto é necessário converter o código do aplicativo c++ para que este rode num sistema Linux. Os principais problemas encontrados nesta fase foram: o código ter sido construído com base na API do Windows (WinAPI), e o facto de a comunicação série se processar de forma completamente diferente no sistema Linux. Por estas razoes foi necessário proceder ao estudo da WinAPI e da comunicação série em sistema operativo Linux.

1. **Comunicação série**

A comunicação série, em Linux, realiza-se através de um sistema semelhante ao de leitura e escrita para um ficheiro, isto é, as portas serie são tratados como se fossem um ficheiro e apresentam nomes como “/dev/ttys0”, “/dev/ttys1”. Neste caso, a comunicação série com o Arduíno acontece sempre na porta “/dev/ttyUSB0”. [<https://www.eltima.com/serial-programming-for-windows-and-linux.html>] Para efetuar a comunicação entre o aplicativo e as portas série foi usada a tecnologia termios, o que corresponde a uma API para entradas e saídas de terminal para sistemas Unix. A estrutura do programa para realizar as operações de entradas e saídas serie com a ajuda de termios é a seguinte:

1. Abrir o dispositivo série através da chamada standard do sistema Unix “open(2)”.
2. Configuração dos parâmetros e propriedades da comunicação serie com a ajuda de funções e estruturas de dados especificas de termios.
3. Uso das funções standard do sistema Unix “read(2)” ou “write(2)”, para ler e escrever para a porta serie.
4. Fechar o dispositivo série através da chamada standard “close(2)”

No processo de desenvolvimento do código de leitura da porta serie começou-se por criar uma classe denomina “serial” que recebe por argumento uma string com o nome da porta serie que se pretende utilizar, neste caso “/dev/ttyUSB0”. De seguida, procede-se á abertura do ficheiro recorrendo á chamada standard “open(2)”, tal como mostrado no extrato de código abaixo:

int serial\_port = open(“/dev/ttyUSB0”, O\_RDWR);

Uma imagem com texto

Descrição gerada automaticamente

**Figura 7 Fluxograma função ConnectComPort ()**

Na figura 7 está representado o fluxograma da função “ConnectComPort”. Nesta função começa-se por criar a variável “com1” do tipo “serial” e logo de seguida executa-se o método “com1.com\_conection()” que é responsável pela abertura do ficheiro da porta serie e que retorna o correspondente descritor de arquivo. Posteriormente é chamado o método “com1.conf\_conection” que recebe o descritor por argumento. Neste método procede-se á configuração dos parâmetros da comunicação serie utilizando as estruturas da API termios, como apresentado no extrato de código abaixo.

tty.c\_cflag &= ~PARENB;

tty.c\_cflag &= ~CSTOPB;

tty.c\_cflag |= CS8;

tty.c\_cflag |= CREAD | CLOCAL;

tty.c\_lflag &= ~ICANON;

tty.c\_lflag &= ~ECHO;

tty.c\_lflag &= ~ECHOE;

tty.c\_lflag &= ~ECHONL;

tty.c\_lflag &= ~ISIG;

tty.c\_iflag &= ~(IXON | IXOFF | IXANY);

tty.c\_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL);

tty.c\_oflag &= ~OPOST;

tty.c\_oflag &= ~ONLCR;

tty.c\_cc[VMIN] = 18;

tty.c\_cc[VTIME] =10;

Neste fragmento de código é possível observar os cinco grandes campos da estrutura termios, o “tcflag\_t c\_iflag” que é responsável pelos modos de entrada; o “tcflag\_t c\_oflag” que atua ao nível dos modos de saída; o “tcflag\_t c\_cflag” que atua sobre os modos de controlo; o “tcflag\_t c\_lflag” responsável pela configuração dos modos locais; e o “cc\_t c\_cc[NCCS]” encarrega dos carateres especiais. [https://man7.org/linux/man-pages/man3/termios.3.html]

As configurações com maior importância são as que incluem os seguintes parâmetros:

* “~PARENB” – desabilita o bit de paridade
* “~CSTOPB” – implementa um stop bit na comunicação
* “CS8” – define o tamanho da máscara para 8 bits
* “~ICANON” – ativa funcionamento no modo não canônico
* “VMIN” – define o número mínimo de caracteres lidos no modo não canónico, neste caso são 18
* “VTIME” -define o tempo máximo em decimas de segundo da leitura no modo não canónico, neste caso 1 segundo (10 decimas de segundo). O que significa que ao fim de um segundo é retornado o conteúdo lido na porta serie independentemente do número de caracteres ter atingido o valor definido em “VMIN” ou não.

A definição da baud rate de leitura é conseguida através da chamada de “cfsetispeed()”, esta deve ser igual á que foi definida nas configurações do arduíno, neste caso 9600 bits por segundo.

Como se pode verificar no fluxograma da Figura 7, após serem definidas as configurações da comunicação série o programa entra num ciclo infinito. Neste ciclo o programa está constantemente a ler dados da porta serie por meio do método “com1.read\_data”, e sempre que encontra dados novos estes são processados e validados pelo restante sistema. O método “com1.read\_data” usa a chamada standard “read(2)” que escreve os dados lidos numa variável global chamada “read\_buffer” e retorna o número de bytes lidos. Se o valor retornado for inferior a zero significa que houve um erro de leitura e os dados são rejeitados.

1. **Interface gráfica para supervisionamento**

Nesta fase, a Indmei propôs a elaboração de uma interface gráfica que permitisse ver os dados de produção em tempo real. Foi sugerido que implementasse uma página Web que mostrasse o número de peças e o tempo total de paragem por hora para o turno atual.

Como referido anteriormente, para o desenvolvimento em back-end foi escolhida a linguagem de programação Python, para o front-end optou-se por utilizar JavaScript.

Uma imagem com captura de ecrã, relógio, desenho

Descrição gerada automaticamente

**Figura 8 Diagrama de interações da interface gráfica**

Na figura 8, observa-se das diferentes interações entra as tecnologias usadas. Os ficheiros escritos em Python apresentam três tarefas de extrema importância: a gestão e manutenção do servidor, feita com recurso á tecnologia referida anteriormente Flask; a leitura e processamento dos dados armazenados nas bases de dados; e a passagem do conteúdo para as tecnologias de front-end. O JavaScript é utilizado para atribuir características dinâmicas e atualizar a informação da página html.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

**Figura 9 Mapa de diretórios da aplicação**

A microestrutura flask necessita que os ficheiros sigam a estrutura apresentada na figura 9, o programa principal “interface.py” tem de estar dentro do diretório principal “/interface”; os ficheiros html modelo devem ser colocados numa pasta com o nome “templates”; e os ficheiros do tipo js (javaScript) e css dentro do diretório “static”.

**3.2.1. Desenvolvimento Back-end**

No desenvolvimento desta aplicação o servidor apenas precisou de dois “routes”, o primeiro para proceder á renderização do modelo interface.html e o último para obter os dados que são visualizados na página html, tal como mostra o código abaixo.

app = Flask(\_\_name\_\_)

@app.route("/")

@app.route("/home")

def home():

return render\_template('interface.html')

@app.route("/getData")

def getData():

posto = request.args.get('posto')

mycursor = mysql.connection.cursor()

turnos = infogeral()

c\_turno = get\_today\_turno(turnos)

pecas, tempo, horas = get\_today\_data(c\_turno, posto)

return jsonify({'turno':c\_turno['turno'], 'pecas':pecas, 'tempo':tempo, 'hora':str(c\_turno['inicio']), 'h':horas})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host="0.0.0.0", port=5000, debug=True)

Uma imagem com captura de ecrã

Descrição gerada automaticamente

**Figura 10 Fluxograma da função “getData()”**

A função “getData()” tem com principais tarefas recolher os dados de produção do turno atual, para o posto que é passado por argumento; e organizar esses dados por hora. O primeiro passo para que tal aconteça é estabelecer uma conexão á base de dados, o que requer o módulo flask\_mysqldb. No extrato de código abaixo pode-se verificar como são definidas as configurações dessa conexão.

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'indmei'

app.config['MYSQL\_PASSWORD'] = '\*\*\*\*\*\*\*\*\*'

app.config['MYSQL\_DB'] = 'indmei'

mysql = MySQL(app)

Como mostra a figura 10, após a ligação ser estabelecida, é executada a função “infogeral()” que recolhe a informação relativa a cada turno da tabela “def\_turnos”, recorrendo á seguinte query:

Select \* from def\_turno

Consequentemente essa informação é passada para a função “get\_now\_turno()” que com base na hora do sistema determina o turno instantâneo. De seguida, procede-se á chamada de “get\_today\_data()” que seleciona a informação para o turno pretendido e para o dia atual. Por fim os dados são retornados num objeto JSON, seguindo a seguinte estrutura:

{

'turno':c\_turno['turno'],

'pecas': pecas,

'tempo': tempo,

}

* **“turno”** – contém os detalhes do turno selecionado (hora de início, hora do fim, e nome do turno)
* **“pecas”** – contém um vetor com o número de peças para cada hora do turno
* **“tempo”** – contém um vetor com o tempo total de paragem da linha por cada hora do turno

**3.2.2. Desenvolvimento Front-End**

Para efetuar a visualização dos dados em tempo real, a Indmei propôs que a página html fosse composta por seis tabelas, e cada tabela devia apresentar pelo menos oito linhas (uma linha para cada hora do turno) e duas colunas (a primeira para o numero de peças produzidas nessa hora, e a segunda para o tempo total de paragem nessa hora).

Uma imagem com símbolo, desenho

Descrição gerada automaticamente

**Figura 11 Fluxograma do ficheiro “interface.js”**

Na figura 11 está representado o fluxograma do ficheiro “interface.js”, como podemos observar assim que a página html carrega é executado um ciclo “for” com a duração igual ao número de postos existentes, neste caso seis. Durante cada ciclo são executadas duas funções, “drawtables()” e “updateData()”. Na primeira procede-se ao desenho da tabela para o posto correspondente, e na segunda é efetuado um pedido AJAX que obtém os dados a partir do processo definido em back-end “/getData” e atualiza as tabelas correspondentes.

$.ajax({

url: "/getData",

data: {

'posto': posto,

},

success: function (result) {

…

$('#Tbody'+posto).empty();

$('#Tbody'+posto).append(rowDados)

}

})

Pela interpretação do extrato de código apresentado acima, percebe-se que para efetuar o pedido AJAX foi utilizada a tecnologia JQuery (mencionada no capítulo 2) para facilitar a troca de dados.

1. **Aplicação Web para análise de dados**

Nesta última fase do projeto foi desenvolvida uma interface que permite a visualização de grandes quantidades de dados de forma mais organizada e percetível.

**3.3.1. Sistema de autenticação e autorização**

O método de autenticação por JSON Web Token (JWT) foi o escolhido para fazer a gestão do sistema de login. O JWT é um pedaço de texto com algumas informações codificadas que permitem identificar o usuário para o qual este foi gerado. Para o correto funcionamento é necessário efetuar os seguintes passos:

1. O utilizador deve introduzir o seu nome de utilizador e palavra passe
2. Verificar que estes estão corretos no interior da aplicação Flask
3. Gerar o JWT com a identificação do usuário codificada
4. Enviar o JWT para o utilizador
5. Sempre que o utilizador efetuar um request para a aplicação deve enviar o JWT gerado anteriormente. Assim é possível verificar que o utilizador que enviou o JWT é o mesmo para o qual este foi gerado.

O último passo é muito importante, pois quando o servidor recebe um JWT valido pode verificar a informação codificada no seu interior e validar o utilizador.

Para implementar este sistema é preciso importar o modulo jwt para o interior do ficheiro onde a aplicação está definida. Posteriormente é necessário definir uma “secret\_key” que é usada para assinar o token e garantir que este foi gerado pela aplicação. Por norma a “secret\_key” deve ter 256 bits de tamanho, neste projeto, de cada vez que a aplicação é iniciada, gera-se uma nova “secret\_key” aleatória, usando a seguinte instrução:

app.config['SECRET\_KEY'] = ''.join(choice(string.ascii\_uppercase + string.ascii\_lowercase + string.digits) for \_ in range(256))

Após a verificação dos dados inseridos pelo utilizador (username e password) cria-se o objeto JWT usando a função “jwt.encode()”, onde são passados por argumento a identificação do usuário, o tempo de duração do token (neste projeto é de 30 minutos), e a “secret\_key”, tal como no extrato de código seguinte:

token = jwt.encode({'user': user, 'exp': datetime.utcnow()+timedelta(minutes=20)},app.config['SECRET\_KEY'])

A verificação do token é feita numa função á parte que deve ser chamada em todos os “routes” que se quer proteger. Nessa função o token é descodificado usando a chamada “jwt.decode()” que recebe por argumentos o próprio token e a “secret\_key” e retorna o “payload”, tal como mostra o extrato de código abaixo. O “payload” é o nome técnico para a informação que está contida no token (id do usuário).

@app.route('/dashboard')

@token\_required

def dashboard():

return render\_template('dashboard.html')

Os dados de utilizador estão guardados numa base de dados chamada “Users” que contém apenas a tabela “usersLog” com duas colunas (nome de utilizador e password). As passwords são guardadas com recurso á encriptação sha256. No código da aplicação a comparação da palavra passe introduzida com a que esta guardada na base de dados é feita da seguinte forma:

sha256\_crypt.verify(inserted\_password, password)

**3.3.2. Algoritmos de recolha de dados**

No desenvolvimento do programa existem dois algoritmos de recolha de dados que são crucias para o funcionamento da aplicação. O primeiro faz o retorno dos dados de produção para o dia e postos passados por argumento; e o segundo retorna a informação relativa à produção por hora para o dia atual.

Uma imagem com texto

Descrição gerada automaticamente

**Figura 12 Fluxograma do algoritmo “get\_data\_day()”**

Na figura 12 está representado o fluxograma da função “get\_data\_day()”. Esta carrega grande importância pois é a base que suporta quase todas as funcionalidades da aplicação. Este algoritmo recebe por argumento o dia e o posto para os quais deve recolher a informação presente na base de dados. O primeiro passo a efetuar é iniciar a conexão á “database” através do modulo “mysql.connection.cursor()”, tal como já foi visto no capítulo 3.2.1.. De seguida o programa entra num ciclo que permite recolher os dados para todos os turnos, o comando SQL utilizado foi o seguinte:

Select total\_pecas\_bloco, tempo\_paragem\_total from data%s where id\_turno=%s and data='%s'" % (posto, turno, dia)

Após a execução deste comando verifica-se se foi retornado algum conteúdo, no caso de a resposta não conter informação as variáveis “pecas” e “tempo” são igualadas a zero; caso a resposta apresente conteúdo é efetuado um novo ciclo que faz a soma das peças produzidas e do tempo de paragem e guarda os valore nas variáveis “pecas” e “tempo”. Por último, estas variáveis são adicionadas às listas “total\_pecas” e “total\_tempo” utilizando a seguinte operação:

total\_pecas.append(int(pecas))

total\_tempo.append(int(tempo))

No final do algoritmo estas listas devem apresentar a informação para cada um dos turnos.

Uma imagem com texto, mapa

Descrição gerada automaticamente

**Figura 13 Fluxograma do algoritmo “get\_data\_hour()”**

O segundo algoritmo, representado pela figura 13, recebe apenas o número do posto por argumento. Neste algoritmo é executado um ciclo “for” que percorre todas as 24 horas do dia e preenche uma lista com os dados de produção para cada hora. O comando SQL utilizado para recolher a informação da base de dados foi:

Select nr\_pecas\_periodo, tempo\_paragem\_total from data%s where hora\_inicio\_periodo = '%s' and data='%s'" % (posto, i, today)

**3.3.3. Visualização dos dados**

Para facilitar a observação da informação pela parte do cliente foi usado o JavaScript com o intuito de tornar as páginas mais dinâmicas e interativas. A informação é apresentada maioritariamente em tabelas ou gráficos. A atualização do conteúdo é feita através de pedidos AJAX, tal como foi explicado e exemplificado no capítulo 3.2.2.. Para a apresentação em gráficos foi usada a libraria “Chart.js”.

A API “Chart.js” permite desenhar os diferentes tipos de gráficos dentro de um elemento do tipo canvas em HTML5. Neste projeto foram utilizados dois tipos de gráficos: de barras, e de linha. Uma das características mais interessantes desta libraria é que os gráficos são do tipo responsivo, ou seja, adaptam-se ao espaço disponível na página.

O primeiro passo para desenhar um gráfico passa por criar um elemento canvas na página html, para tal usa-se o código seguinte:

<canvas id="mychart" style="display: inline;"></canvas>

De seguida, no ficheiro de JavaScript, usasse o método “getElementById()” para obter o elemento canvas onde se pretende desenhar o gráfico. Consequentemente tem de ser criada uma variável do tipo “new chart” que recebe por parâmetros o elemento canvas e um objeto que contém os dados e as configurações do gráfico, tal com mostra o código abaixo:

bigctx = document.getElementById(“mychart”);

bigchart= new Chart (bigctx, bigchartinfo);

Para o desenho do gráfico é preciso passar o objeto “bigchartinfo” que deve conter as seguintes propriedades:

1. “type” – especifica o tipo de gráfico que deve ser desenhado (exemplos: barras, linha)
2. “labels” - contém um vetor de strings que especifica o nome das instâncias a ser comparadas no gráfico
3. “datasets” – é um objeto que deve conter no mínimo os dados que se pretende introduzir no gráfico e uma “label” que especifica o tipo de dados

var bigchartinfo = {

type: 'bar',

data: {

labels: ['semana 1','semana 2'],

datasets: [{

data: [250,300],

label: “Posto 1”

},

{

data: [300,260],

label: “Posto 2”

}]

}

1. Análise dos resultados

Nesta fase

Referências Documentais

1. Alcatel—*SAM-O Service Aware Manager Open Interface, Release 3.0, OSS Interface Developer Guide*. Alcatel, 2005.
2. AZEVEDO, Carlos; AZEVEDO, Ana—*Metodologia Científica: contributos práticos para a elaboração de trabalhos académicos*. 3ª ed. Porto: Ed. C. Azevedo, 1996. pp 67-91
3. Hofmann, Alfred; Barth, Ursula; Beyer, Ingrid; Günther, Christine; Holzwarth, Frank; Kramer, Anna; Siebert-Cole, Erika—*Lecture Notes in Computer Science: Authors’ Instructions for the Preparation of Camera-Ready Contributions to LNCS/LNAI/LNBI Proceedings*. Springer-Verlag, Computer Science Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany
4. Mauro, Douglas; Schmidt, Kevin—*Essential SNMP, Help for System and Network Administrators*. O’Reilly, 2001. ISBN 0-596-00020-0.
5. Net-SNMP, Version: 5.2.1, <http://www.net-snmp.org/>
6. PINTO, J. A. Cardoso—*RETA, Manual de Referência*. PT-Comunicações, 2004.
7. RODRIGUES, José—*gestAlarm: Um Colector de Alarmes Para a Rede IP/MPLS da PT*. Dissertação de Licenciatura em Engenharia Eletrotécnica—Eletrónica e Computadores orientada pelo Eng.º Pedro Assis e apresentada no Instituto Superior de Engenharia do Instituto Politécnico do Porto, na especialidade de Telecomunicações em 2006.
8. ROSE, Marshall T. —*The Simple Book: An Introduction to Networking Management, Readings in Simple Network Management Protocol*. Prentice Hall, Inc, 1996. ISBN 0-13-451659-1.
9. Schönwälder, Jürgen—*Internet Management Technologies*. International University Bremen, Germany, <http://www.faculty.iu-bremen.de/schoenw>
10. Coletor baseado em JMS

Neste anexo são descritos alguns dos pormenores do desenvolvimento e testes efetuados para o coletor de alarmes pela interface JMS [1] e respetivos resultados.

Histórico

* 13 de Dezembro de 2006, Versão 1.0, <mailto:pfa@isep.ipp.pt>
* 21 de Junho de 2007, Versão 1.0.a, <mailto:pfa@isep.ipp.pt>

$Id:PESTA\_LEEC\_RELATORIO.dot v1.0.a Date:21-06-2007$