INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

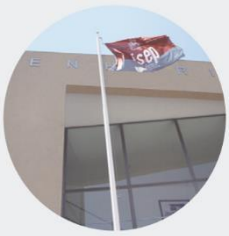



# Solução Industrial para Controlo de OEE

**PEDRO DANIEL MAIA FALCÃO**
Outubro de 2015

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




# Solução Industrial para Controlo de OEE

**PEDRO DANIEL MAIA FALCÃO**
Outubro de 2015

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

# Solução Industrial para Controlo de OEE

**Pedro Daniel Maia Falcão**

**Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia**

**Orientador: Doutora Maria Fátima Coutinho Rodrigues**

**Júri**:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, outubro 2015

# Resumo

É cada vez mais forte e natural o desenvolvimento de aplicações móveis. Hoje em dia qualquer pessoa seja de forma amadora ou profissional tenta tirar o máximo partido do seu dispositivo móvel, através de aplicações desenvolvidas para as mais diversas áreas.

Um dos sistemas operativos que mais programadores apostam é no *Android* devido à sua versatilidade e ás suas capacidades enquanto sistema operativo dentro de um *smartphone*. Para além dessas mais-valias, desenvolver uma aplicação para *Android* não tem qualquer custo, apenas terá, caso seja uma aplicação para adicionar a *Google Play Store*, caso contrário pode desenvolver-se qualquer tipo de aplicação sem custos, o que não acontece com outros sistemas operativos.

Tendo em conta os custos, é normal as pequenas e médias empresas tentarem investir neste tipo de desenvolvimento, pois evitam gastos em licenciamentos para produzir os seus produtos. Seguindo este contexto, esta dissertação insere-se nesse perfil, isto é, tentar encontrar uma solução com baixos custos e que produza os resultados pretendidos. Através das mais diversas *API's* é possível adaptar o sistema operativo *Android* a qualquer área apenas basta enquadrar da melhor maneira ideias e dar asas à imaginação.

Desta forma, esta dissertação irá centrar-se na área industrial, na qual o *Android* pode e deve ser bastante útil, se for usado com rigor. Irá ser abordado o desenvolvimento de aplicações móveis e serão analisadas as tecnologias envolventes no projeto assim como algumas das principais soluções já implementadas e desenvolvidas por empresas no setor e para o sector industrial. O projeto aborda também de que forma é possível aliar o sistema operativo *Android* á área industrial de forma a permitir um melhor desempenho por parte de todos em prol da empresa á qual está associado. A necessidade de criação desta aplicação, surgiu numa perspectiva de melhoria contínua, com o objectivo de abandonar o procedimento instituído, que era realizado manualmente, passando a fazê-lo de uma forma automática e mais interativa. O processo será realizado pelos colaboradores e terá sempre o seu contributo, mas de uma forma mais rigorosa, simples e eficiente, aumentando a qualidade do serviço.

**Palavras-chave**: OEE, Aplicações móveis, Usabilidade, Near Field Communication, Android

# Abstract

Nowadays the development of mobile applications is growing exponentially. It's so natural, that any person tries to get the most out of their mobile device, through the applications developed for the most diverse areas.
Android, it's now the most widely used smartphone operating system in the world, and the one that more developers are betting, not only because of its versatility, but also because of their multiple capabilities as an operating system inside a smartphone.
In addition to these capital gains, develop an application for Android has no cost, it only has when added to the Google Play Store, otherwise you can develop any type of application for free, which doesn't happen with the other operating systems.
Taking into account the costs, it is normal that small and medium sized companies try to invest in this type of development, because they prevent spent on licensing processes to produce their products. Following this context, this thesis is part of this profile, that is, try to find a low cost solution that produce the desired results. Through the most varied API's is possible to adapt the Android operating system to any area, you only need to fit your ideas in the project and a little bit of imagination.
In this way, this thesis will focus on industrial area, in which the operating system Android can, and must be very useful, if used with precision. Will be approached the development of mobile applications and also analyzed the surrounding technologies in the project, as well as some of the main solutions already implemented and developed by companies in the sector and to the industrial sector. The project also discusses how is possible to combine the Android operating system to the industrial area, in order to allow a better performance of the company which is associated with. The need of the creation of this application, emerged in a perspective of continuous improvement, with the intention of abandon the procedure established, which was performed manually. The process will be carried out by employees and will always have their contribution, but in a more rigorous, simply and efficiently way, increasing the quality of the service.

**Keywords**: OEE, Mobil Applications, Usability, Near Field Communication, Android

# Agradecimentos

À minha orientadora, Doutora Fátima Rodrigues, pela disponibilidade, por todos os esclarecimentos, paciência e incentivo para fazer mais e melhor, ao longo deste longo percurso.

À minha mãe, pela teimosia e persistência, pelas palavras e também pelo incentivo demonstrado ao longo deste percurso, para que este chegassa a bom porto.

À minha irmã, pela ajuda incondicional no desenvolvimento desta tese, pela motivação, insistência e disponibilidade demonstrada para que todo este projeto chegasse a um fim.

Ao meu pai, que apesar de já não estar presente entre nós, é a ele a quem dedico inteiramente todo o meu trabalho, foi ele que me ajudou e continua a ajudar mesmo não estando cá.

À minha namorada, Elisabete Alves, por todo o incentivo e apoio em todos os momentos deste percuro, que foram muitas vezes conturbados, mas sempre ultrapassados com a ajuda dela, em conjunto com o resto da minha família.

Um muito obrigado a todos, e Pai esta é para ti!

# Índice

# Lista de Figuras

# Lista de Tabelas

# Acrónimos

## Lista de Acrónimos

| | |
|---|---|
| **OEE** | Overall Equipment Effectiveness |
| **OHA** | Open Handset Alliance |
| **MWT** | Mean Waiting Time |
| **MTTR** | Mean Time to Repair |
| **MTBF** | Mean Time Between Failure |
| **NFC** | Near Field Communication |
| **JSON** | JavaScript Object Notation |
| **SOAP** | Simple Object Access Protocol |
| **XML** | Extensible Markup Language |
| **3DES** | Triple Data Encryption Algorithm |
| **RDBMS** | Relational Database Management System |
| **HTTP** | HyperText Transfer Protocol |
| **NDEF** | Data Exchange Format |
| **UL** | UltraLight |
| **ULC** | UltraLight Cs |
| **LLCP** | Logical Link Control Protocol |
| **URI** | Uniform Resource Identifier |
| **ERP** | Enterprise Resource Planning |
| **IIS** | Internet Information Services |
| **DBMS** | Data Base Management System |
| **SDK** | Software Development Kit |

# 1 Introdução

## 1.1 Enquadramento e Motivação

Atualmente, a maioria das pequenas e médias empresas tenta apostar em tecnologias *open-source* para desenvolver os seus produtos e usá-los no dia-a-dia em prol dos seus objetivos, evolução e garantias de excelência para com os seus clientes. Nesse sentido, cada vez mais o uso de *smartphones* ou *tablets* tem vindo a crescer nas empresas, fazendo com que o número de aplicações móveis desenhadas especificamente para cada área de negócio seja também cada vez maior, por forma a satisfazer as suas necessidades e aumentar a produtividade das organizações.

O projeto descrito nesta tese, foi concebido numa empresa na área da engenharia industrial. Nesta área, as empresas ainda preferem apostar em tecnologias que já deram provas de ser viáveis e poderosas, do que em apostar em tecnologias recentes, embora possam vir a tornar-se melhores do que aquelas já estão implementadas no mercado.

O objetivo final de qualquer empresa industrial, é integrar todas as tecnologias que dispõe e que possam vir a surgir, de forma a conseguir o controlo de tudo o que é produzido, comprado e vendido. Neste projeto será abordado o controlo do que é produzido, sendo o tema principal deste mesmo projeto, o *Overall equipment effectiveness* (OEE) [OEE, 2013].

O OEE consiste numa forma de medição da eficácia dos equipamentos presentes na fábrica. O OEE pressupõe o conhecimento de três parâmetros para obter o seu resultado final, desta forma, quanto maior for o seu resultado final, melhor teoricamente, a empresa irá operar, contribuindo para obter melhores resultados e melhorar o seu rendimento.

Os três parâmetros pelos quais o OEE se rege são: a disponibilidade, a eficiência e a qualidade. A disponibilidade, refere-se ao tempo em que a máquina está em funcionamento durante o horário laboral da fábrica, subtraindo todo o tempo perdido em paragens para refletir esse princípio.

A velocidade, mede-se em conjugação com a sua disponibilidade. Ao conjugar os dois parâmetros, torna-se possível verificar se a máquina está a produzir corretamente, ou está fora dos parâmetros aceitáveis, os quais são calculados através do tempo de ciclo de cada máquina.

No que diz respeito à qualidade, serve para verificar se todos os produtos produzidos cumprem os parâmetros para serem transportados para o mercado. Por vezes, este parâmetro de medição, não é o mais relevante, podendo ser assumido sempre o seu valor a 100%, de modo a dar mais atenção aos dois primeiros parâmetros.

## 1.2 Objetivos e Resultados esperados

O principal objetivo deste projeto é fornecer a todos os intervenientes neste processo a medição do OEE, que antes do desenvolvimento desta aplicação, ainda era realizada manualmente em papel. Este procedimento fazia com que os trabalhadores perdessem demasiado tempo, tanto no seu preenchimento, como na sua leitura, provocando diversos erros, que por sua vez levariam a correções. Através da utilização desta aplicação, este processo será simplificado, assim como bastarão apenas alguns cliques no ecrã para que todos os dados sejam recolhidos e de uma forma mais fiável.

Uma outra vantagem da utilização da aplicação será a facilidade com que os resultados das medições OEE poderão ser consultadas, para além de que a nível de produção, através da aplicação é possível obter os resultados em tempo real, relativos a cada máquina, podendo assim o responsável intervir e melhorar os seus procedimentos e quaisquer problemas adjacentes.

A aplicação será uma mais-valia para o desenvolvimento da fábrica, pelas inúmeras funcionalidades que possui e pela informação que permite obter, de uma forma simples e eficaz.

## 1.3 Documento

A dissertação encontra-se dividida em cinco capítulos, onde será descrito todo o processo de implementação e da solução do presente projeto.

No primeiro capítulo, será apresentada uma pequena introdução, que reflete as motivações para a proposta desta solução e o seu enquadramento. De seguida, serão enunciados os principais objetivos deste projeto, assim como os resultados a alcançar.

No segundo capítulo irá ser exposto o sistema operativo envolvido na proposta, o *Android*, juntamente com uma reflexão explicativa sobre o motivo pelo qual cada vez mais utilizadores usam as aplicações móveis, no quotidiano.

Após a abordagem ao sistema operativo Android, será enunciado o tema principal da proposta, o OEE, e a sua forma de cálculo, onde serão demonstrados alguns cálculos exemplificativos. Por fim, será feita uma breve descrição de uma das tecnologias usadas neste projeto, o NFC, a qual irá permitir a extração de um dos cálculos extras ao OEE, relativamente aos cálculos da manutenção, como por exemplo o tempo médio entre avarias, sendo que todos os dados são recolhidos pela aplicação.

Relativamente ao terceiro capítulo, será possível visualizar toda a proposta esquematizada, através do seu modelo conceptual, assim como os requisitos necessários para a sua implementação. Para além dos requisitos, irão ser abordadas as tecnologias utilizadas, bem como as funcionalidades presentes na aplicação móvel e as interações que os utilizadores irão ter com a mesma.

Após o desenho de toda a aplicação, no quarto capítulo será exposta toda a base onde a aplicação móvel se desenvolveu, assim como o seu *layout*. Posteriormente, será redigida a forma como o sistema foi implementado, dando exemplos de código, que sustentam todas as variantes desta proposta.

Por fim, no quinto capítulo, será realizada uma reflexão crítica, de toda a proposta, dando a conhecer para além das implementações já concretizadas, aquelas que poderão ocorrer no futuro, de modo a melhorar todo a aplicação.

# 2 Estado Arte

## 2.1 Android

O *Android* [História em Detalhe, 2015] é um sistema operativo móvel baseado em Linux. Este foi criado pela *Android inc.,* empresa à qual pertenciam *Andy Rubin*, *Rich Miner*, *Nick Sears* e *Chris White*, os pioneiros do projeto. Inicialmente, o objetivo não era criar um sistema operativo para *smartphones*, mas sim para câmaras digitais, para que estas pudessem aceder facilmente a serviços informáticos. No entanto, constataram que iriam tirar pouco proveito com esse tipo de desenvolvimento, e optaram pela área dos dispositivos móveis.

Em 2005, a Google apropriou-se dessa pequena empresa criada por *Andy Rubin*, de modo a tentar competir com a sua opositora, a Apple. Apesar de nessa altura a Apple ainda não ter lançado para o mercado qualquer dispositivo móvel, estaria a preparar-se para o lançamento do seu primeiro *iPhone*, com teclado no ecrã, que ocorreu passado dois anos, em 2007.

Nesse mesmo ano, a Google iniciou a *Open Handset Alliance* (OHA), que consistiu numa junção de várias empresas para um mesmo fim, isto é, a criação de padrões para dispositivos móveis, todos estes livres. Entre as empresas da *Open Handset Alliance*, destacam-se a *T-Mobile*, empresa de telecomunicações, a *Motorola* e *HTC*, empresas de fabrico de dispositivos móveis, e a *Texas Instruments e Qualcomm*, fabricantes de chip's.

Em setembro de 2008 surge então o primeiro dispositivo móvel *Android*, denominado o *HTC Dream* (G1). Este já incluía a integração com os serviços do Google, um browser web, o denominado *Android Market, a atual Play Store*, *Wi-Fi*, *Bluetooh* e multitarefa.

No entanto, embora tenha conseguido evoluir após a sua criação, apenas na versão 2.3 (*GingerBread*) alcançou 20% da quota do mercado, e assim tornar-se no que é atualmente, uma referência nos sistemas operativos móveis. Devido ao seu crescimento o *Android*, não estagnou no desenvolvimento dos apenas nos dispositivos móveis, estando também presente noutros dispositivos, como televisões, relógios, óculos, tablets e netbooks. Alguns destes têm vindo a ganhar uma maior relevância, nomeadamente o Tablet.

Para além do investimento por parte dos fabricantes de dispositivos móveis em transformar estes dispositivos em verdadeiros computadores, quer em termos de capacidade, velocidade de processamento, como de memória, o *Android*, foi progredindo e aperfeiçoado ao longo do tempo e com ele surgiram novas funcionalidades que tornaram o quotidiano mais simplificado para os utilizadores, assim como a adoção de um estilo cada vez mais percetível e prático para todas as faixas etárias, mesmo para aqueles que têm pouco ou nenhum conhecimento sobre tecnologias.

A figura 1, representa a taxa de penetração no mercado dos vários sistemas operativos para dispositivos móveis, onde podemos constatar que apesar de ter baixado, o *Android* continua a ter a percentagem mais elevada com 79%, seguido do *iOS* com 18%, o *Windows* com 3% e o *Blackberry* e os outros com cerca de 1%.

O Android, para além de possuir uma maior percentagem de penetração no mercado, também é dos sistemas mais vendidos, com cerca de 265 000 unidades vendidas. Estes dados referem-se ao primeiro trimestre do ano de dois mil e quinze.

**Worldwide Smartphone Sales to End Users by Operating System in 1Q15 (Thousands of Units)**

| Operating System | 1Q15 Units | 1Q15 Market Share (%) | 1Q14 Units | 1Q14 Market Share (%) |
|---|---|---|---|---|
| Android | 265,012 | 78.9 | 227,549 | 80.8 |
| iOS | 60,177 | 17.9 | 43,062 | 15.3 |
| Windows | 8,271 | 2.5 | 7,580 | 2.7 |
| Blackberry | 1,325 | 0.4 | 1,714 | 0.6 |
| Other OS | 1,268.7 | 0.4 | 1,731.0 | 0.6 |
| **Total** | **336,054.4** | **100.0** | **281,636.9** | **100.0** |

Source: Gartner (May 2015)

Figura 1 – Quota de Mercado e Unidades Vendidas *Android* [Gartner, 2014]

Quando falamos do sistema operativo *Android*, associamos a um produto único, no entanto o mesmo está dividido em várias versões, como demonstra a Figura 2.

É possível concluir que a versão que os dispositivos mais usam é a versão 4.4 (API 19) "*KitKat*" com 39%. Logo depois encontra-se a versão 4.x (API 16 a API 18) "*Jelly Bean*" 37%. De salientar que a versão mais recente do sistema operativo *Android*, anda só está disponível em 13% dos dispositivos.

| Version | Codename | API | Distribution |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.3% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 5.6% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 5.1% |
| 4.1.x | Jelly Bean | 16 | 14.7% |
| 4.2.x | | 17 | 17.5% |
| 4.3 | | 18 | 5.2% |
| 4.4 | KitKat | 19 | 39.2% |
| 5.0 | Lollipop | 21 | 11.6% |
| 5.1 | | 22 | 0.8% |

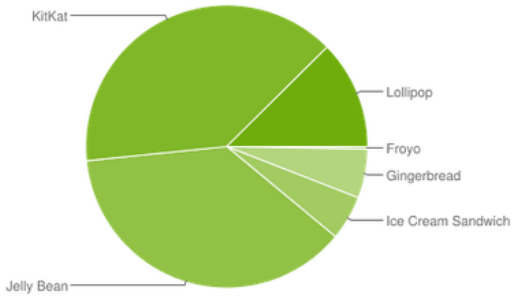



Figura 2 – Distribuição Android a nível de Sistema Operativo [Developers , 2015]

## 2.2 Aplicações Móveis

Vivemos na era dos dispositivos móveis, sejam eles tablet's, smartphones e/ou computadores portáteis, onde a dependência da tecnologia é cada vez maior, sendo por isso necessário que a informação que é apresentada nestes dispositivos seja concebida de uma forma simples e eficaz.

Devido a todos estes fatores, as aplicações móveis adquiriram um papel preponderante, trazendo consigo uma diversidade de funcionalidades que possibilita aos utilizadores estarem à distância de um clique daquilo que mais gostam, ou necessitam, seja trabalho ou lazer.

Cada vez mais as pequenas empresas apostam neste tipo de aplicações, pois necessitam de chegar aos consumidores de uma forma mais simples**,** proporcionando aos utilizadores finais uma maior interação, assim como uma aproximação da realidade do seu negócio, contribuindo para expandir o nome das empresas pelo mercado de forma simples e rápida.

Existem vários estudos [Gartner, 2014] que apontam que cada vez mais os utilizadores usam aplicações móveis, e que essa tendência tem vindo a crescer muito rapidamente. Estima-se que em 2018 os *tablets* passem a ser o dispositivo preferido dos utilizadores, deixando assim para segundo lugar o conhecido desktop.

## 2.3 Aplicações Móveis na Indústria

Uma importante forma de melhoria contínua a nível empresarial, é criar e apostar em ferramentas que melhorem os resultados e a produtividade, por forma a alcançar os objetivos. Quando surge a ideia de cálculo do *OEE*, aparecem também os vários métodos para esse efeito, tais como:  o registo manual, registo em base de dados, registo via autómatos, etc.

Neste sentido, as mais variadas formas de cálculo têm evoluído de acordo com a evolução e introdução da tecnologia nas indústrias. Umas apostam em tecnologias já com alguma maturidade e com provas dadas, outras apostam em tecnologias novas e recentes na área da indústria, mas que têm bastante potencial para trazer bastantes melhorias a todos os níveis.

### 2.3.1 Versacall VT 3000

Um dos exemplos de aposta em tecnologias recentes como Android, é o software *Versacall*. Este apresenta o sistema *VT 3000* [VT 3000 System] que engloba dois componentes principais no seu sistema, o Android, e um dispositivo que ligado à máquina controla os input's e output's. Estes dois equipamentos efetuam a sua ligação através de Bluetooth de forma a que o dispositivo ligado à máquina, possa comunicar com o dispositivo *Android*.

O dispositivo conectado à máquina tem como função registar todas as ocorrências relacionadas com a mesma, assim como o estado da máquina e os erros que possam surgir a nível de software.

Por outro lado, o dispositivo *Android*, tem como objetivo servir de intermediário entre o operador e o sistema. O operador tem como função registar as operações efetuadas assim como, começar nova produção, acabar produção, iniciar novo Setup, chamar técnicos da manutenção, identificar paragens e registar através de um código de barras o que produziu.


Figura 3 – Imagens da Aplicação *Versacall VT 3000*

Para além da recolha de todos os dados pela aplicação móvel *Android*, a mesma possui uma opção que permite visualizar os relatórios sobre as máquinas onde foram efetuados os registos de produção e paragem.

## 2.4 OEE

As empresas recorrem a inúmeros indicadores de modo a avaliar o seu desempenho a nível de mercado e da sua posição competitiva. No entanto, nem todas possuem dados sobre todos os processos, desde a conceção até ao produto final, mas apenas sobre o cariz económico e financeiro. Desta forma será necessário interpretar os resultados financeiros e analisar a possibilidade de obter melhores resultados através da melhoria dos processos de operações.

Neste sentido, em 1970, *Seiichi Nakajima* desenvolveu um meio que possibilita não só a avaliação do desempenho dos equipamentos, mas também uma forma de progredir ou aperfeiçoar progressivamente os processos envolvidos continuamente. Através deste conceito foram muitas as empresas que adotaram o *OEE*, tornando-se uma referência mundial para a medição dos equipamentos industriais.

O *OEE* [Vorne, 2008] é composto por três parâmetros: disponibilidade, eficiência e qualidade.

- Disponibilidade – Mede o tempo em que o equipamento está disponível para produção

$$\boldsymbol{Disponibilidade} = \frac{\text{Tempo Total de Trabalho}}{\text{Tempo Total Planeado}}$$

Este fator é afetado pelas paragens efetuadas ao longo da produção como por exemplo:

  - Falta de Material
  - Falta de Operadores
  - Alterações ao Plano
  - Avarias
  - Falhas de Energia
  - Controlo de Qualidade
  - Setup

- Eficiência – Mede a competência de o equipamento produzir à velocidade estabelecida previamente

$$\boldsymbol{Eficiência} = \frac{\text{Tempo de Produção}}{\text{Tempo de Produção Esperada}}$$

Este fator baseia-se em velocidades já pré-estabelecidas por equipamento, podendo variar, ou porque a velocidade encontra-se incorreta, sendo necessário ajustar ainda o equipamento quando opera a uma velocidade maior, obtém um tempo de produção menor.

- Qualidade – Mede a diferença entre os artigos que cumprem os critérios de qualidade (artigos bons) e o total de artigos provenientes de cada equipamento

$$\boldsymbol{Qualidade} = \frac{\text{Artigos Bons}}{\text{Total de Artigos}}$$

Este procedimento torna-se complexo, pois é quase impossível medir em tempo real a qualidade do produto. Regra geral, apenas uma pequena percentagem de cada artigo é considerado como artigo rejeitado, ou seja, com defeito, o restante, mesmo que aproveitado para outros fins, é considerado produto bom para o sistema. Tendo em conta este fator, a medição é assumida maioritariamente que o valor aproximado é 100%. Daí a importância recair sobre os dois primeiros indicadores, a disponibilidade e a eficiência.

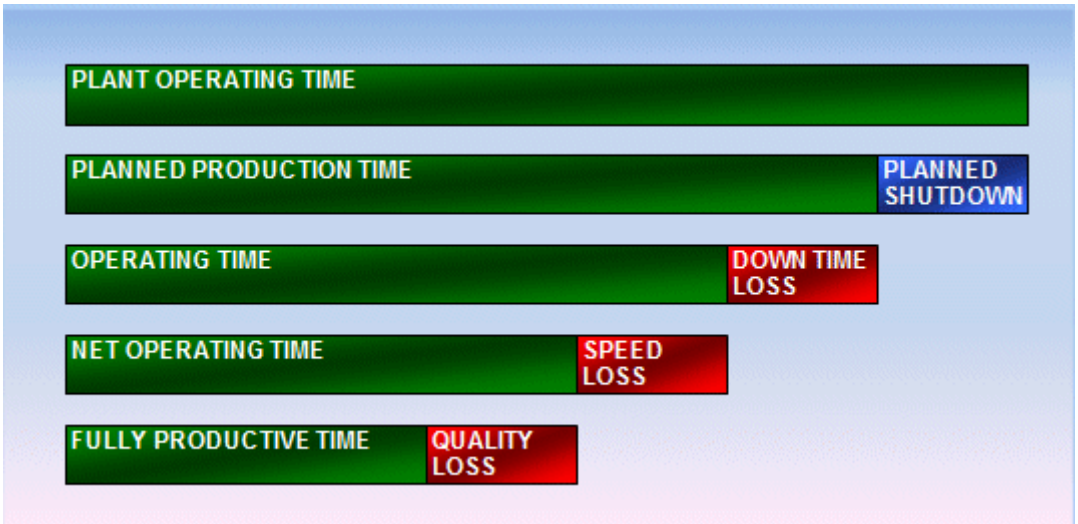



Figura 4 – OEE Cálculo

### 2.4.1 Exemplo de Cálculo

Assumindo o cálculo do OEE como sendo a análise de um determinado período de tempo em que o equipamento vai produzir. Tendo como exemplo um tempo total de produção de 480 minutos, ao qual se subtraem 30 minutos para paragens planeadas, perfazendo um tempo total de produção de 450 minutos.

Ao tempo total de produção irá subtrair-se o tempo total de paragens não planeadas que neste caso será de 70 minutos, obtendo um tempo de operação de 380 minutos.

$$\boldsymbol{Disponibilidade} = \frac{480 - 30 - 70}{480} \times 100 \cong 79{,}1\ \%$$

Será a partir desta etapa que a disponibilidade do equipamento se obtém, a qual será calculada através do tempo total de operação, a dividir pelo tempo total de produção, o que corresponde, no exemplo dado, a uma disponibilidade de 79%.

No parâmetro eficiência temos a velocidade, à qual está sempre associado um valor por equipamento, e obtendo já à partida o tempo total de operação, apenas será necessário o valor total de peças produzidas e a velocidade do equipamento.

Por exemplo, podemos assumir que ao produzir 10 000 peças, num equipamento em que a velocidade estabelecida é de 40 peças por minuto, podemos efetuar o cálculo dividindo o total de peças produzidas, pelo tempo total de operação, seguido da divisão desse resultado pela velocidade estabelecida de 40 peças por minuto, no qual obtemos 65 % de eficiência.

$$\boldsymbol{Eficiência} = \frac{10\,000}{380} = 26{,}3\ \frac{26{,}3}{40} \cong 65\ \%$$

Por fim, falta apenas o cálculo da qualidade das peças produzidas, sendo que no total das 10 000 peças, 8 000 eram consideradas boas. Assim sendo, divide-se o total de peças boas, 8 000 peças, pelo total de peças produzidas, 10 000 peças, onde o resultado da qualidade irá ser de 80%.

$$\boldsymbol{Qualidade} = \frac{8000}{10000} \times 100 \cong 80\ \%$$

Após a realização dos cálculos dos três parâmetros, é necessário multiplicar os resultados finais da disponibilidade, eficiência e qualidade. O OEE daquele equipamento, neste caso, será de 79%, multiplicando por 65% e multiplicando por 80%, o que perfaz um OEE de 41%.

$$\boldsymbol{OEE} = 79\ \% \times 65\% \times 80\% \cong 41\ \%$$

## 2.5 NFC

O *Near Field Communication (NFC)* é uma tecnologia que permite a interação entre dispositivos via rádio frequência a uma curta distância. Esta tecnologia remonta aos anos 80, onde *Charles Walton* registou a primeira patente acerca de um identificador de rádio frequência portátil.

Em 2004, surgiu o *NFC* fórum [NFC Forum, 2015], levado a cabo pela Nokia, Philips e a Sony, atualmente possui mais de 160 membros, que trabalham para o mesmo objetivo, promover o *NFC* e certificar-se que os dispositivos de marcas diferentes conseguem comunicar sem nenhum constrangimento.

Em 2006, as especificações do *NFC* surgiram e com elas apareceram também os primeiros cartazes inteligentes, e mais tarde surge o primeiro dispositivo móvel a suportar esta

tecnologia. Com o decorrer dos anos e com a adesão de cada vez mais membros ao *NFC* Fórum, a tecnologia evoluiu, e em 2009 apareceu o *Standard Peer-to-Peer* para transferência de qualquer tipo de ficheiros.

O primeiro dispositivo móvel surge em 2010, já com *Android* e com *NFC,* o chamado *Nexus S*.


Figura 5 – *Nexus S Android GingerBeard NFC*

### 2.5.1 Como funciona

O *NFC* comparativamente com outras tecnologias de transmissão de dados, foi desenhado com o objetivo de ser um meio de transmissão de dados com mais fiabilidade. Para que seja efetuada a transmissão entre dois objetos, é necessário que os mesmos se aproximem para que se proceda à troca de informações. De forma a existir comunicação são necessários dois objetos, sendo o funcionamento possível de duas formas, modo passivo e modo ativo.

O modo passivo, define a passagem de informação apenas de um objeto para outro, ou seja, um dos objetos emite um sinal e o outro apenas o recebe. No modo ativo, ambos os objetos comunicam entre si, ou seja, um objeto emite um sinal para o outro objeto, e esse mesmo emite outro sinal de resposta.

Para além destes dois modos de funcionamento, o *Near Field Communication (NFC)* caracteriza-se por três modos de operações:  o modo Escrita/Leitura, *modo Peer-to-Peer* e o modo emulação de cartão.

No modo escrita/leitura, o dispositivo *NFC* tem a capacidade de ler/escrever qualquer credencial incorporada, seja uma tag, ou um cartaz inteligente. No *modo Peer-to-Peer*, os dois dispositivos podem trocar dados entre si, tal como se pode fazer através de *Blueetooth*. Por fim, no modo emulação de cartão, o dispositivo *Near Field Communication (NFC)* funciona como um objeto a parte, assim como um cartão tradicional de crédito. Apenas emula, como o

próprio nome sugere, e permite efetuar pagamentos sem alterar a estrutura de receção já existente.

### 2.5.2 Tipos de Chip

Para além dos vários modos de funcionamento de operação, o *Near Field Communication (NFC)* também foi evoluindo a nível de chips existentes. Hoje em dia, existem chips específicos para a área em que estão a ser comercializados. Entre os vários tipos, destacam-se o *NTAG 203*, o *UltraLight*, o *UltraLight C* e o *Standard 1k* [NFC Chip, 2015].

O *NTAG 203* é um chip com boa capacidade de memória (168bytes), ideal para quase todo o tipo de aplicações. Tem um bom desempenho a nível de leitura, sendo menos eficiente ao nível da segurança de transmissão dos dados, pois neste tipo de chip não existe qualquer tipo de criptografia associada.

O *Ultralight* é uma versão mais acessível e é utilizado para transmitir pequena quantidade de informação. É utilizado em cartazes inteligentes e na maior parte das aplicações, apenas executando algumas ações. Em termos de capacidade, não é equiparável ao *NTAG 203*, mas ambos partilham o mesmo problema ao nível da segurança, uma vez que não possuem qualquer tipo de criptografia associada.

O *Ultralight C* é uma evolução do *Ultralight*, permitindo para além de maior capacidade de armazenamento (192 bytes) comparativamente aos (64 bytes) do *Ultralight*, também traz consigo um padrão de encriptação, chamada *Triple Data Encryption Standard (3DES)*. Apesar da maior capacidade, a sua leitura torna-se mais difícil e é mais baixa relativamente a sua versão anterior, o *Ultralight*.

Por fim, o Standard 1k, permite até 1024 bytes de capacidade de armazenamento, o que faz com que seja muito utilizado para a transmissão de grandes quantidades de informação, assim como os bem conhecidos *Vcard* - cartão de contatos existente nos dispositivos móveis.

Em termos de segurança, não possuem o padrão *Triple Data Encryption Standard (3DES)*, mas sim, o padrão *Crypto-1,* o que faz com que a transmissão dos dados neste tipo de chip seja segura. Tem uma boa capacidade de leitura, perdendo na falta de compatibilidade com dispositivos móveis. É inferior aos restantes tipos de chips devido a esta incompatibilidade.

Tabela 1 – Comparação entre Chips [NFC Chip , 2015]

| | UltraLight | UltraLight C | Standard 1k | NTAG 203 |
|---|---|---|---|---|
| **Capacidade de Memória** | 64 bytes | 192 bytes | 1024 bytes | 168 bytes |
| **Comprimento de Texto** | 39 Caracteres | 130 Caracteres | 709 Caracteres | 130 Caracteres |
| **Compatibilidade Móvel** | Sim | Sim | Não | Sim |
| **Utilizações** | Cartazes Inteligentes e maior parte das aplicações | Apenas para aplicações que usem criptografia | VCards | Ideal para todos tipos de aplicações |
| **Fórum NFC** | Sim | Sim | Não | Sim |
| **Criptografia** | Não | 3DES | Crypto-1 | Não |
| **Leitura** | Boa | Má | Boa | Muito Boa |

### 2.5.3 Protocolos

#### *2.5.3.1 NDEF*

O *NFC Data Exchange Format (NDEF)* [NDEF, 2008] é um formato de dados normalizado, que é usado para troca de dados entre dois objetos *Near Field Communication (NFC)*. Este segue todas as normas do Fórum NFC, e é usado maioritariamente para troca de URIs e textos pequenos. Este formato é composto por Mensagem NDEF e Registos NDEF.

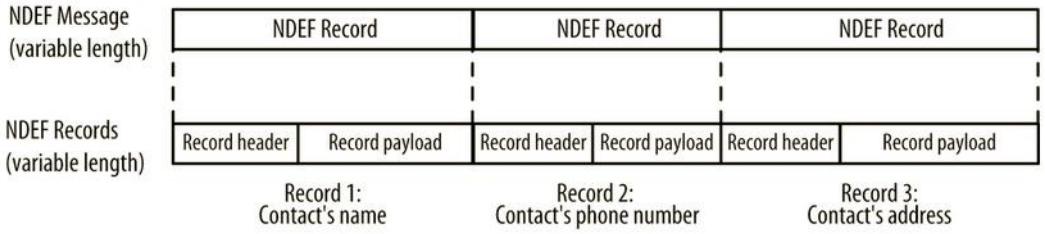



Figura 6 – Estrutura NDEF

Cada mensagem é constituída por um cabeçalho, que contém todos os dados referentes á mensagem, tipo de registo e comprimento.

Este tipo de transmissões são geralmente curtas, devido ao limite de espaço da mensagem e também porque normalmente cada *Tag* apenas contém uma mensagem, o que faz com que o processo seja rápido. Estas mensagens estão limitadas a $2\,32^{-1}$ bytes, no entanto os registos podem ser encadeados de forma a passar objetos mais longos.

Todos os registos *NDEF* são compostos por 4 campos:

- *Type Name Format* – Indica como o recetor interpreta o que recebe
- *Record Type Definition* – Descreve o formato do registo
- *ID* – Identificador do registo
- *Payload* – O conteúdo da mensagem

Analisando os 4 campos existentes nos registos *NDEF*, o *Type Name Format* é definido por 8 tipos:

- *Empty* – Indica que não existe mensagem associada ao registo
- *Well Known Type* – Indica que o valor segue no formato definido pelo NFC Fórum
- *Media* – Indica que o valor é do media-type [RFC 2046]
- *Absolute URI* - Indica que o valor é do tipo URI [RFC 3986]
- *External* – Indica que o valor segue no formato definido pelo NFC Fórum
- *Unknown* – Indica que o tipo de mensagem é desconhecido
- *Unchanged* – Indica que é a continuação do registo anterior
- *Reserved* - Reservado

Já o *Record Type Definition (RTD),* define o formato de registo para algumas aplicações já conhecidas, tais como, cartazes inteligentes, texto, URI, ou controlo genérico.

#### *2.5.3.2 LLCP*

Para melhorar o *Peer-to-Peer* e incluir este tipo de comunicação nos dispositivos, o NFC Fórum especificou um protocolo de rede conhecido por *Logical Link Control Protocol (LLCP)*. Neste protocolo é necessário que exista uma comunicação bidirecional entre as aplicações.

Esta especificação define dois tipos de serviços, o serviço sem conexão (*ConnectionLess*) e o serviço de conexão orientada (*Connection-oriented*).

O primeiro serviço, o sem conexão (*ConnectionLess*), disponibiliza uma configuração mínima e sem fiabilidade ou controlo de fluxo, deixando todas as operações sob o domínio das aplicações e do ISO/IEC 18092 E ISO/IEC 14443.

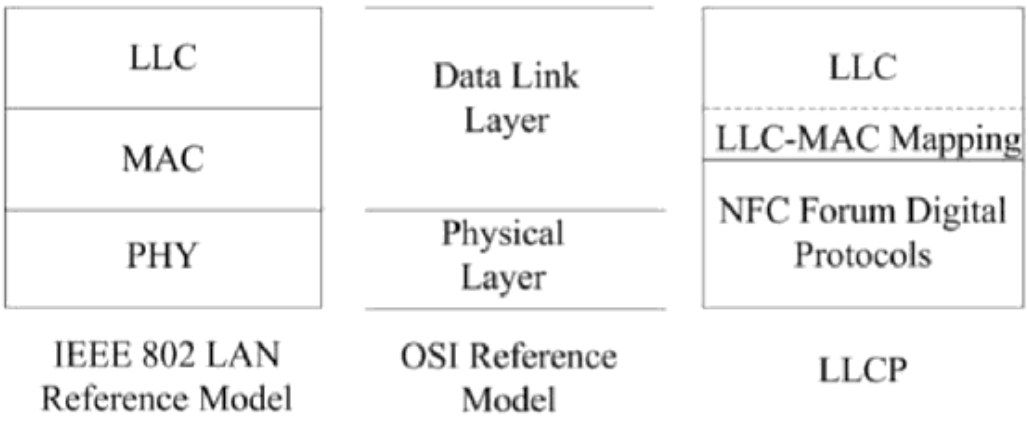



Figura 7 – LLCP Protocolo

Já o serviço de conexão orientada (*Connection-oriented*), disponibiliza a fiabilidade, controlo de fluxo e ainda uma camada de multiplexação baseada em sessão.

O *Logical Link Control Protocol (LLCP)* é baseado no *IEEE 802.2*, e está projetado para suportar tanto pequenas aplicações com pouco fluxo de dados, como para suportar protocolos de rede que oferecem um serviço mais abrangente para as aplicações.

O *NFC Logical Link Control Protocol (LLCP)*, oferece então uma boa base para *aplicações Peer-to-Peer*, melhorando significativamente o já disponibilizado pelo *ISO/IEC 18092*, não interferindo na interoperabilidade entre aplicações *NFC*.

# 3 Desenho da Aplicação

Este projeto consiste na criação de uma aplicação para medição do OEE, focada no sistema operativo *Android*, e com a tecnologia *NFC*. O projeto centra-se numa aplicação móvel, que regista todos os dados de uma qualquer máquina industrial, de uma forma simples e intuitiva.

Esta aplicação, para além do registo desses mesmos dados, também possui uma área administrativa, onde poderão ser consultados todos os dados registados.

Para além do foco da aplicação ser o sistema operativo android, esta também conta com um ambiente web, desenvolvido em *HTML* E *Javascript*, para facilitar o acesso a todos os que não possuam um dispositivo *Android*.

## 3.1 Tecnologia Utilizada

Antes de enunciar todas as suas funcionalidades, é necessário reter alguns conceitos inerentes ao desenvolvimento da aplicação.

- **JSON** – É um formato de troca de informações entre sistemas. Ele caracteriza-se por ser leve, exigindo menos das ligações, e pelo seu formato de resposta, tornando-o de fácil leitura pelo destinatário. É uma alternativa bastante poderosa ao já conhecido *Extensible Markup Language (XML)*.

- **WebService** – É uma solução que permite a interligação e comunicação entre sistemas diferentes. Os *webservices* podem trazer consigo bastante agilidade para efetuar as comunicações entre os mais variados sistemas, e principalmente, são concebidos de forma segura e controlada, trazendo muitos benefícios para quem os

utiliza. Outra vantagem da sua utilização, é a possibilidade de as respostas serem dadas numa linguagem universal, percetíveis para quaisquer sistemas (*XML* ou *JSON*).

- **SOAP** – é um protocolo de troca de mensagens usando como base o XML. *O Simple Object Access Protocol (SOAP)* permite então a interoperabilidade entre os mais variados sistemas. Divide-se em 3 partes: o envelope, que contém e identifica toda a mensagem; o cabeçalho (*Header*) que contém o título, e o body que corresponde ao corpo da mensagem.

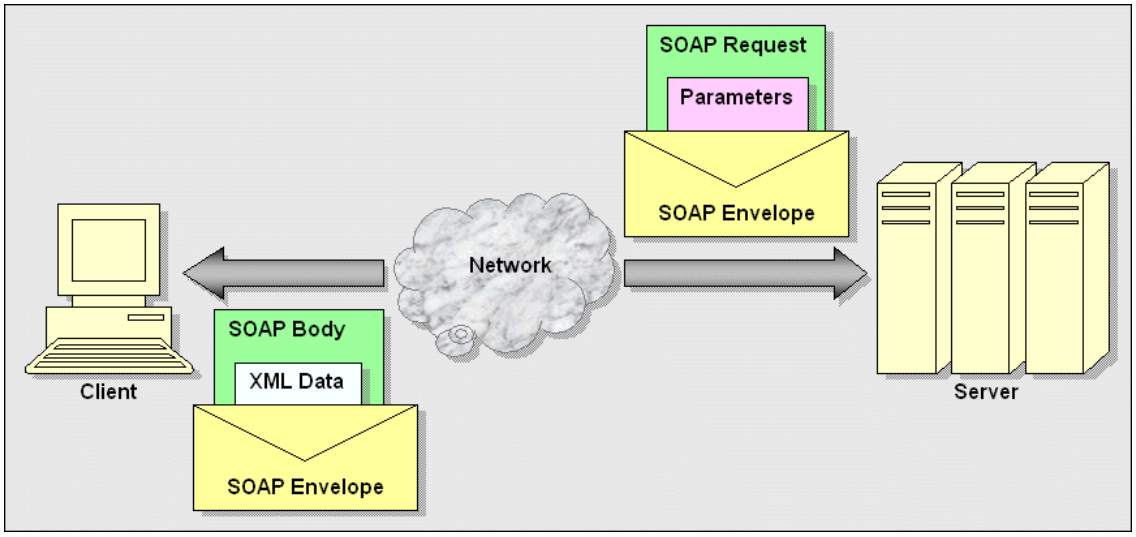



Figura 8– SOAP Protocolo

- **NFC** – É uma tecnologia sem fios que permite a comunicação entre objetos a curta distância (aproximadamente até 10cm) é rápida e eficaz no seu processo. Ela é usada para efetuar compras, identificar objetos, pode servir ainda de guia turístico, conjugado com a realidade aumentada.

- **JavaScript** - o *JavasScript* é uma linguagem de programação do lado do cliente. Com ele podemos dar ao utilizador (cliente) uma maior interatividade e melhor integração dos componentes de uma página normal *HTML*. Todo o seu processamento está dependente do cliente, mas isso não a impede de ser uma das maiores e mais poderosas ferramentas a nível Web.

## 3.2 Requisitos Aplicação

Como já referido anteriormente, a aplicação contém três módulos distintos, e cada módulo tem requisitos diferentes. O módulo operativo, requer sempre um tablet, com pelo menos sete polegadas de dimensão e uma versão igual ou superior a API 17 (JELLY BEAN,4.2). Já o módulo administrativo é desenhado para ser utilizado num *smartphone* com uma versão

idêntica á do *tablet*. Ambos os módulos requerem que o dispositivo possua a tecnologia NFC, caso contrário a aplicação não irá funcionar corretamente.

Para além destes requisitos, é necessária uma ligação *Wifi*, sem necessidade de conexão à internet, para efetuar maior parte das operações existentes. Sendo que o módulo operativo poderá funcionar sem que haja ligação *Wifi*, ou seja, sem necessidade de conexão à internet.

No último modulo, o modulo SMS, não é necessária a tecnologia *NFC*, apenas existe a necessidade de uma ligação à rede móvel, por forma a realizar as operações para que foi desenhado e que contenha uma versão igual ou superior a *API 16 (JELLY BEAN,4.1).* É necessária a ligação *Wifi*, pois esta irá sempre recolher os dados referentes à base de dados. Caso não haja ligação por algum motivo, a aplicação perderá todas as funcionalidades.

Na componente web não existe qualquer requisito específico, é possível aceder através de qualquer Web Browser.

## 3.3 Modelo Conceptual

Em seguida apresenta-se o modelo conceptual que visa descrever a informação que o sistema vai gerir.

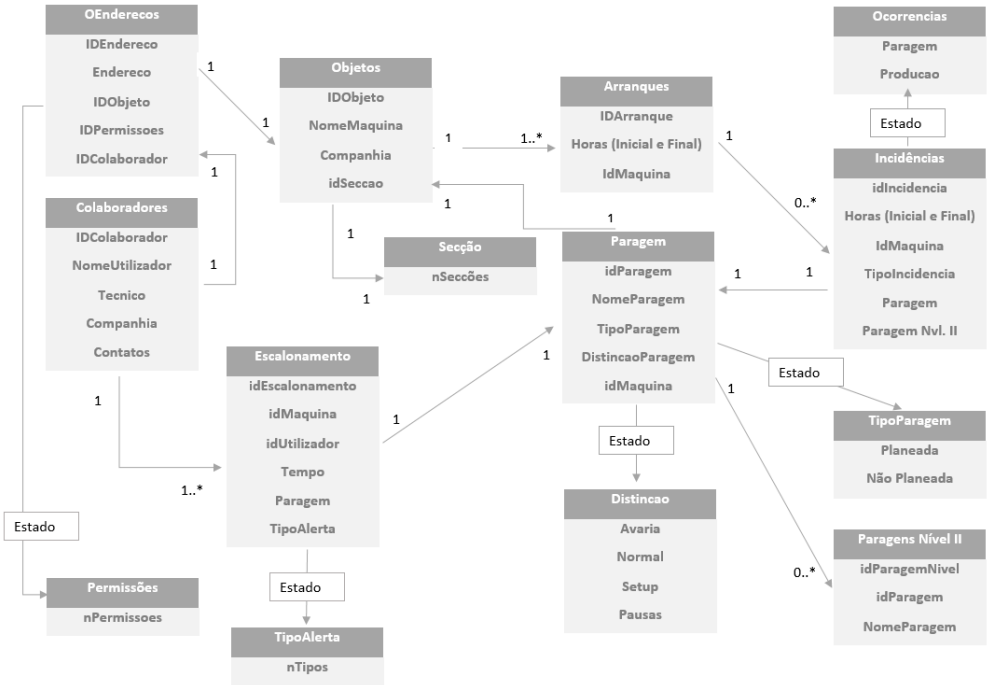



Figura 9– Modelo Conceptual

### 3.3.1 Descrição Modelo Conceptual

- **OEndereços -** A tabela OEndereços é um elemento fundamental na distinção de dispositivos. Nela podemos definir as funções de cada dispositivo, atribuindo sempre a um objeto. Servindo como base de utilização quer para funções administrativas, quer para funções operativas.

- **Objetos -** Associado a cada endereço há um objeto. Para além de conter todas as informações sobre os objetos, possui também um campo denominado secção que permite a subdivisão dos objetos.

- **Colaboradores -** Para além de haver necessidade de definir todos os dispositivos, é também necessário enunciar todos os utilizadores que podem aceder à aplicação. Estes são distinguidos pela empresa em que trabalham e também pela sua formação técnica.

- **Escalonamento -** Cada vez que ocorre uma paragem em qualquer objeto, é necessário haver um tempo útil de resolução para cada paragem. É para este efeito que existe o escalonamento, que corresponde ao tempo que cada técnico terá até ser alertado para se deslocar ao local do objeto, para posterior resolução do problema. Para cada escalonamento é atribuída uma paragem, um tipo de alerta definido e uma máquina, podendo haver para o mesmo utilizador várias paragens, e vários tempos.

- **Arranques -** Para cada máquina existe um ou n arranques. Estes consistem num turno de trabalho, ou seja, começam a partir da hora de início do turno, até ao seu término.

- **Ocorrências -** Dentro de cada período de trabalho existem n ocorrências sobre tudo aquilo que qualquer máquina executa nesse período. Estas podem ser distinguidas por produção ou paragem. Para além desta distinção podem também conter um tipo de paragem associado, e, não obrigatoriamente, uma paragem de segundo nível.

- **Paragem -** Quando ocorre uma incidência do tipo paragem, é necessário definir a que paragem se associa. E como o próprio nome indica, representa o nome de todas as paragens associadas à máquina. Podem ser distinguidos em 4 tipos: Avaria, Normal, Setup e Pausas. Podemos ainda distinguir o tipo de paragem, ela poderá ser planeada ou não planeada.

- **Paragens Segundo Nível -** Para conseguir aprofundar o motivo da ocorrência de uma paragem, é necessário descrever pormenorizadamente a razão das paragens. Este conceito representa as paragens de segundo nível. Estas estão sempre associadas a uma paragem, e podem conter n paragens segundo nível.

Este segundo nível existe para simplificar a gestão de cada máquina, permitindo obter informações precisas sobre as paragens, podendo assim intervir da melhor forma para que este tipo de situações não volte a ocorrer.

## 3.4 Funcionalidades

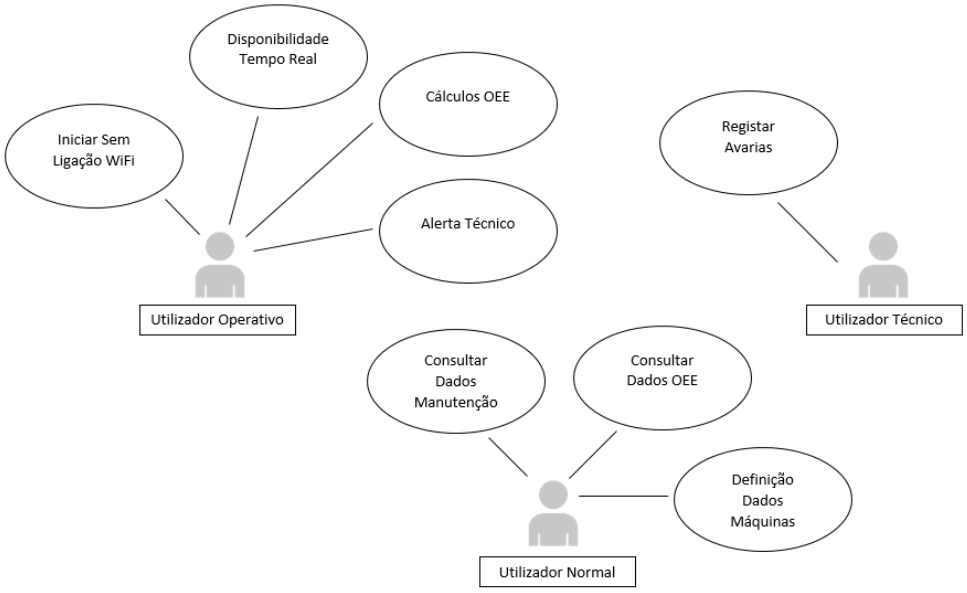



Figura 10 – Funcionalidades

### 3.4.1 Descrição das Funcionalidades da Aplicação

- **INICIAR EM MODO OFFLINE**

Esta funcionalidade está disponível apenas após uma primeira recolha dos dados por parte da aplicação. Ajuda caso haja algum problema na rede, ou falta de ligação ao servidor central. Antes de esta funcionalidade estar disponível, a aplicação irá sempre tentar conectar-se à rede.

Etapas:

1. Iniciar Aplicação
2. Antes de efetuar qualquer início de registo, existe uma verificação da conexão
3. Não existindo conexão, e caso o dispositivo já possua os dados para dar início ao arranque, a aplicação permite iniciar.

- **DISPONIBILIDADE EM TEMPO REAL**

A partir do momento que se inicia qualquer arranque no dispositivo, esta funcionalidade entra em ação. Ela calculará, segundo a segundo, um parâmetro do OEE, a disponibilidade, baseando-se sempre no que regista o utilizador. Cada vez que é iniciada uma nova ocorrência, seja produção ou paragem, é iniciado também um contador, que demonstra o tempo total da ocorrência, permitindo ao longo do tempo o cálculo da disponibilidade em tempo real.

Etapas:

1. Arranque da aplicação
2. Registo das incidências ocorrentes (Produções/Paragens)
    a. Contador desde o início das incidências
    b. Cálculo da Disponibilidade

- **CÁLCULOS OEE**

No final de cada arranque é possível obter um dos parâmetros do OEE, a disponibilidade, depois dessa informação, é necessário calcular os restantes dois parâmetros, eficiência e qualidade. Para realizar esse cálculo recorre-se a um serviço web, que faz ligação com o *Enterprise Resource Planning (ERP)*, onde é feito todo o reporte do que é produzido em qualquer máquina.

Etapas:

1. Fechar arranque
2. Serviço Web solicita dados da eficiência e qualidade
3. Aplicação calcula, já com a disponibilidade, o resultado final do OEE

- **ALERTAR TÉCNICO**

Cada vez que o utilizador presente no dispositivo inicia uma paragem no dispositivo, e após a escolha do tipo de paragem que iniciou, é possível alertar um técnico, caso haja necessidade, para que o mesmo se desloque à máquina e verifique o problema que provocou a paragem.

Etapas:

1. Início de paragem
2. Seleção de Paragem
3. Procura na base de dados do dispositivo (Local), o técnico com o menor tempo de escalonamento para a paragem selecionada

4. Comunicação com o sistema central
5. Alerta do Técnico

- **REGISTAR AVARIAS**

A cada paragem registada pelo utilizador presente no dispositivo há a possibilidade de um utilizador que possua uma etiqueta NFC se identificar no local onde se encontra o dispositivo. Ao deslocar-se este utilizador poderá especificar com um maior detalhe a paragem efetuada através das Paragens de Segundo Nível. Para além de especificar ao detalhe a paragem, é dada a possibilidade de acrescentar as observações que o técnico considere importante realçar.

Etapas:

1. Iniciar paragem (Utilizador Operativo)
2. Identificação técnica com etiqueta NFC (Inicio da Intervenção)
3. Verificação na base de dados do dispositivo se corresponde a um técnico identificado
4. Registo do detalhe avaria/paragem
5. Registo das Observações
6. Fecho da intervenção

- **CONSULTA DADOS MANUTENÇÃO**

Através do registo por parte do utilizador operativo, é possível a extração de vários tipos de cálculos que ajudam a interpretar melhor os problemas de cada máquina onde o dispositivo se encontra. Esta funcionalidade, permite a quem visualiza os dados tanto no dispositivo, como na web, alguns cálculos inerentes a nível de manutenção, tais como, o Tempo Médio entre Falhas (MTBF), Tempo Médio de Reparação (MTTR) e o Tempo Médio de Resposta (MWT).

Etapas:

1. Utilizador acede ao menu correspondente aos cálculos (MTBF/MTTR/MWT)
2. Seleciona a máquina ou todas as máquinas existentes
3. Escolhe o período que pretende verificar os cálculos
4. A aplicação recolhe os dados no servidor central
5. É visualizado o resultado na aplicação

- **CONSULTA CÁLCULOS OEE**

Para além de informações referentes à manutenção, é também possível extrair dados relativos ao OEE. Estes cálculos permitem avaliar o desempenho de cada máquina, e verificar formas de melhorar os resultados obtidos. Nos cálculos, é possível verificar o total de

paragens planeadas, o total de paragens não planeadas, tempo médio de Setup, total de avarias, evolução do OEE, e o OEE por máquina.

Etapas:

1. Utilizador acede ao menu correspondente aos cálculos
2. Seleciona a máquina ou todas as máquinas existentes
3. Escolhe o período que pretende verificar os cálculos
4. A aplicação recolhe os dados no servidor central
5. É visualizado o resultado na aplicação

## 3.5 Interações

Há vários tipos de utilizador a interagirem com a aplicação, os quais podemos dividir em 3 grupos:

- Utilizadores Operativos
- Utilizadores Técnicos
- Utilizadores

Todos contribuem para o desenrolar do processo, com o objetivo de superar os melhores resultados e obter um melhor e maior rendimento de cada máquina.

### 3.5.1 Utilizadores Operativos/Aplicação

Para iniciar, é necessário haver uma configuração inicial da aplicação, escolhendo o ambiente que é possível trabalhar, seja num paradigma de testes, seja num paradigma de produção, assim como o nome do dispositivo. Estas configurações são realizadas na aplicação. No *back office*, configura-se o endereço a atribuir a cada dispositivo, para que seja definida a função a desempenhar.

Após estas configurações iniciais, o utilizador operativo poderá dar início à aplicação. No início de cada horário de trabalho, o utilizador operativo é obrigado a selecionar a opção arranque para poder dar início ao seu trabalho. Após dar início, o próprio terá três opções de escolha: Produção, Paragem e Fecho.

Entende-se como produção, sempre que o objeto começa a produzir algum produto. Em termos de paragem, entende-se como parado, o objeto sempre que haja necessidade de efetuar, por exemplo, algum ajuste ou preparar o objeto para uma nova produção. Por fim, no fecho, que coincide com o termino do horário laboral, é quando toda a produção e/ou paragem tem o seu fim. Após o fecho é onde irão ser apresentados todos os resultados relativos ao seu período laboral.

Estas três opções surgem no início da aplicação, logo após o arranque, mas, caso o operador escolha uma paragem, ser-lhe-á apresentado um outro ecrã, onde o mesmo terá de introduzir a paragem efetuada. Ao selecionar a paragem efetuada, a aplicação avança para um outro ecrã, onde o utilizador operativo irá aguardar, caso necessite, de um técnico para a resolução da paragem, ou o mesmo utilizador poderá finalizar a paragem e prosseguir com o seu trabalho.

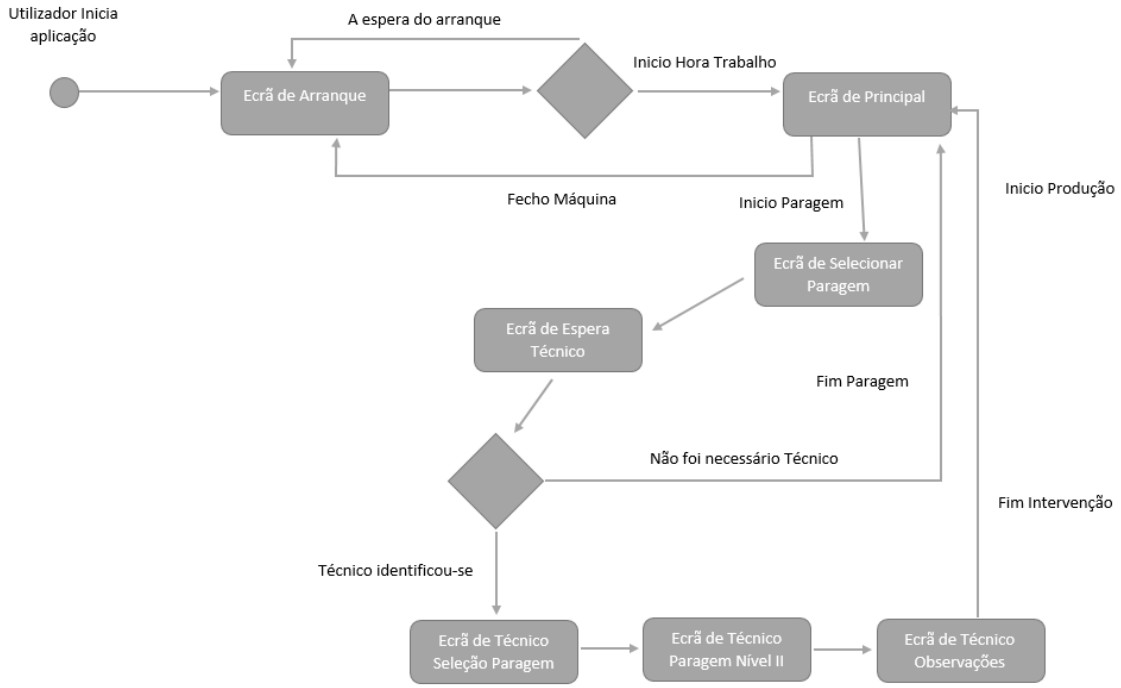



Figura 11 – Diagrama Utilizadores Operativos / Aplicação

### 3.5.2 Utilizadores Técnicos/Aplicação

A cada paragem, caso haja a necessidade de intervenção por parte de um técnico, é necessário que o mesmo se identifique através da sua etiqueta *NFC*. Após a sua identificação, surge um ecrã onde será possível selecionar a paragem que considere mais correta, corrigindo caso necessite a paragem selecionada por parte do utilizador operativo. Após essa seleção, caso exista uma paragem mais específica (Paragem de Segundo Nível), o próprio técnico irá selecioná-la, podendo no final redigir algumas observações sobre a paragem. Após as observações, procede-se ao encerramento da paragem, podendo o utilizador operativo continuar com a aplicação.

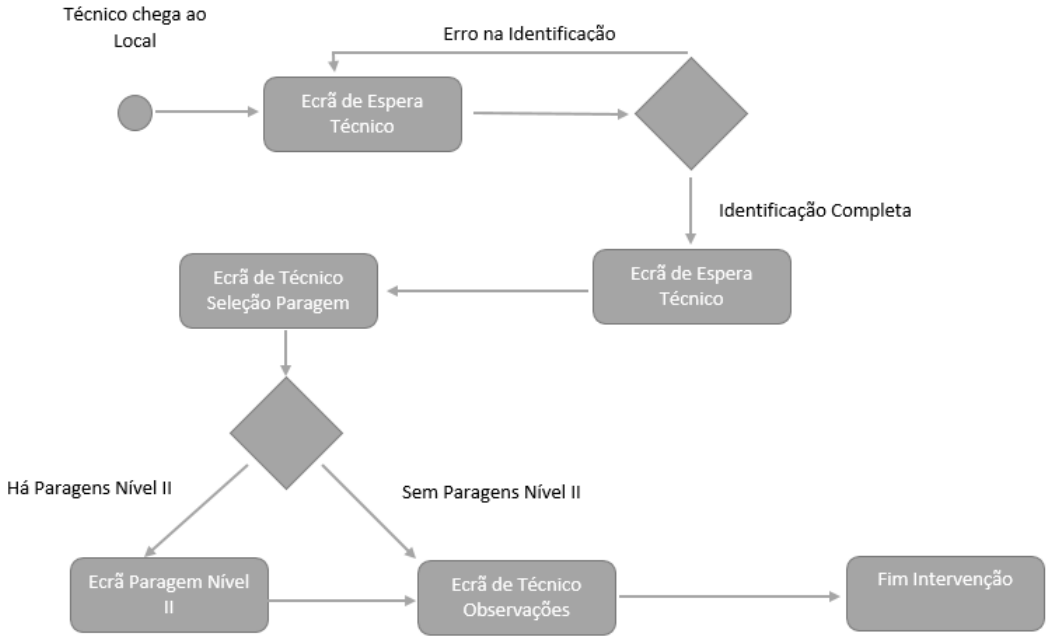



Figura 12 – Diagrama Utilizadores técnicos / Aplicação

### 3.5.3 Utilizadores/Aplicação

A consulta dos dados poderá ser feita pelos utilizadores, quer através da aplicação, quer via web. Na aplicação móvel, a autenticação rege-se através do IP atribuído, sendo os gestores da rede, quem decide. Já na plataforma web, inicialmente, bastará apenas o endereço web para acederem aos dados disponíveis na plataforma.

No entanto, em ambos os casos, ao iniciar qualquer ambiente, seja web, seja aplicação móvel, irá ser apresentada uma *dashboard*, onde se encontram os dispositivos ativos e todas as informações sobre da sua atividade em tempo real, assim como um resumo do total de paragens efetuadas até ao momento.

Após a visualização da *dashboard*, os utilizadores poderão efetuar consultas dos cálculos já referidos, assim como realizar a introdução dos parâmetros necessários para o correto funcionamento dos dispositivos.

# 4 Sistema Desenvolvido

## 4.1 Arquitetura Sistema

Neste capítulo é descrita a forma como foi idealizada e desenvolvida a arquitetura de todo o sistema. Irão ser abordados os requisitos para o servidor, que possui todos os dados centralizados, assim como os *WebServices* necessários para poder executar qualquer operação na base de dados. Para além do servidor, irá ser abordada a aplicação móvel, assim como a plataforma web existente para manter o sistema.

### 4.1.1 Arquitetura Servidor

O servidor escolhido para a realização deste protótipo foi o *IIS (Internet Information Services)*. É um servidor que permite a publicação de vários tipos de tecnologias, assim como o *PHP*, *ASPX*, *HTML*. Para além do *IIS*, o sistema foi planeado para ter a base de dados em *SQL Server*.

Para além de alojar toda a plataforma web, para uma maior simplicidade na manutenção da aplicação, aloja também alguns *webservices,* que serão utilizados pela aplicação móvel de forma a esta poder visualizar os dados existentes na base de dados. Cada *Webservice* terá a sua função, podendo ser usado tanto para a plataforma web, como nas aplicações móveis, de forma a simplificar todos os processos.

Após o envio de todos os dados por parte dos dispositivos móveis, existem na maior parte das tabelas da base dados, os *Triggers*, que permitem que o processo de automatização do sistema seja mais facilitado, tratando os dados de forma imediata sempre que este cumpra as parametrizações existentes nesse pequeno excerto de código. Para além dos *triggers*, existem também algumas *Standard Procedures*, que permitem efetuar algumas operações que envolvem um maior número de dados, assim como um maior número de cálculos. Têm como

vantagem o fato de se encontrarem pré-compiladas, permitindo economizar tempo e recursos ao servidor e ao utilizador.

Para que a aplicação SMS funcionasse em pleno, foi necessário acrescentar ao *SQL Server* alguns trabalhos recorrentes, como por exemplo, a verificação do estado das máquinas, que ocorre minuto a minuto, por forma a verificar as atualizações constantes sobre o estado de cada aplicação móvel (mais concretamente de cada máquina).

- **Verificação Estados das Máquinas** - Esta rotina ocorre a cada minuto e irá verificar o estado de cada máquina, para que, caso haja um tempo de escalonamento ultrapassado, seja possível inserir numa tabela um ou vários registos para a lista de espera de envio de mensagens. A partir do momento que o registo se encontra nessa lista, a aplicação que trata das mensagens irá proceder ao envio dos respetivos dados existentes.

  A Figura 13 representa o esquema da base de dados no servidor.

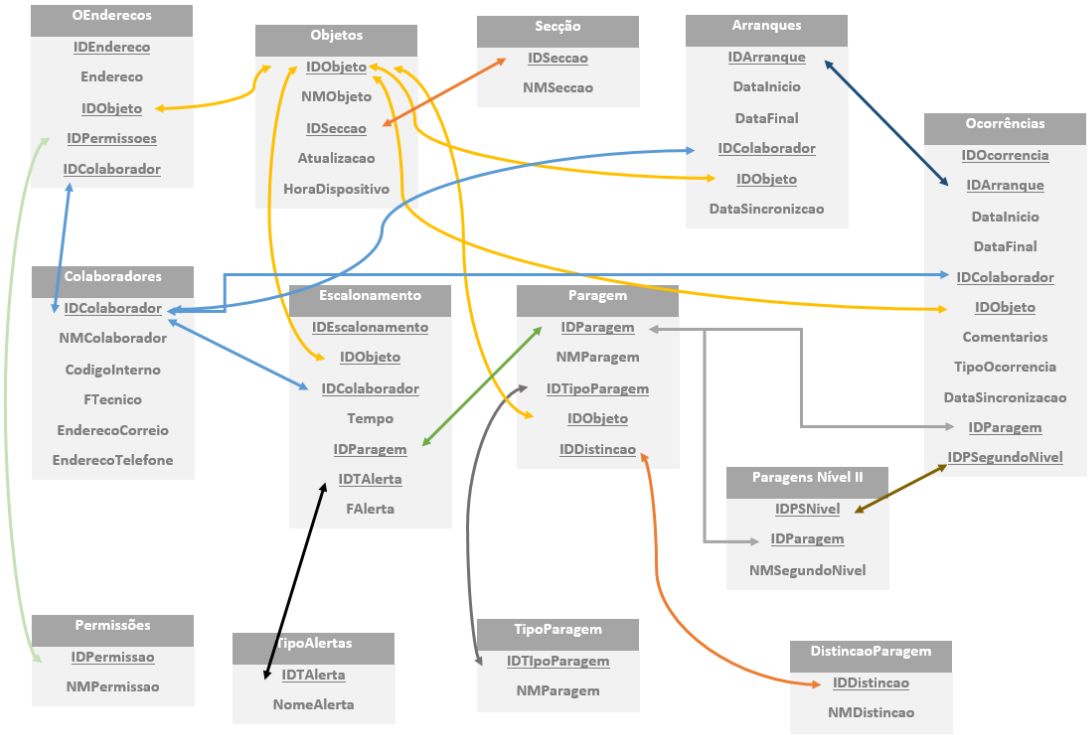



Figura 13 – Estrutura da Base de Dados no Servidor

### 4.1.2 Arquitetura Aplicação Móvel

O desenvolvimento da aplicação móvel foi efetuado em *Java*. Para o desenvolvimento da aplicação móvel, foi necessário recorrer a algumas bibliotecas *open-source* já existentes para *Android*, de modo a facilitar e proporcionar uma melhor experiência ao utilizador que irá fazer uso da aplicação no dia-a-dia. A Figura 14 representa a estrutura da aplicação no *Android Studio*

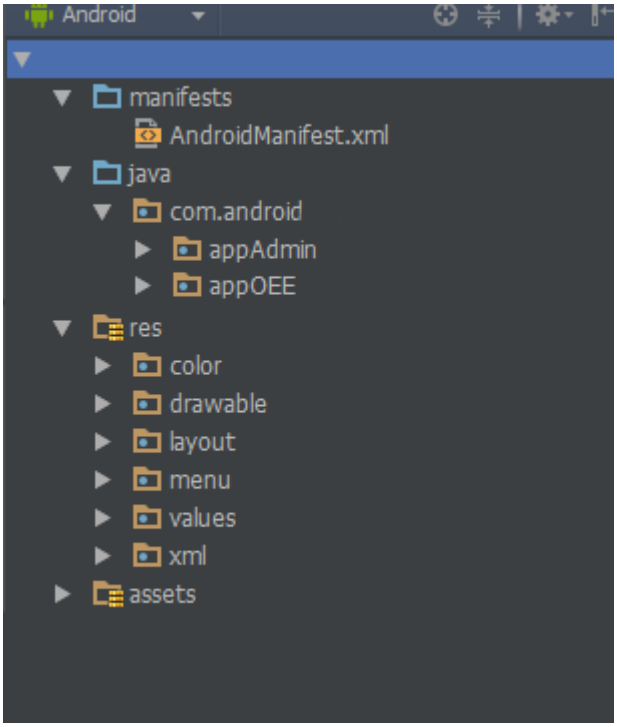



Figura 14 – Estrutura de Projeto Aplicação Móvel *Android Studio*

#### *4.1.2.1 Estrutura Base de Dados SQLite*

A base de dados existente na aplicação móvel difere ligeiramente daquela que se encontra no servidor. A base de dados da aplicação móvel apenas contém os dados necessários para a aplicação funcionar, isto é, a informação mais atualizada, referentes às tabelas (Objetos, Colaboradores, Escalonamento, Paragem e Paragens Segundo Nível). Contudo a base de dados poderá ainda conter dados referentes aos últimos trabalhos realizados na aplicação referentes às tabelas de Arranques e Ocorrências. Apenas estas se encontram na base de dados da aplicação móvel.

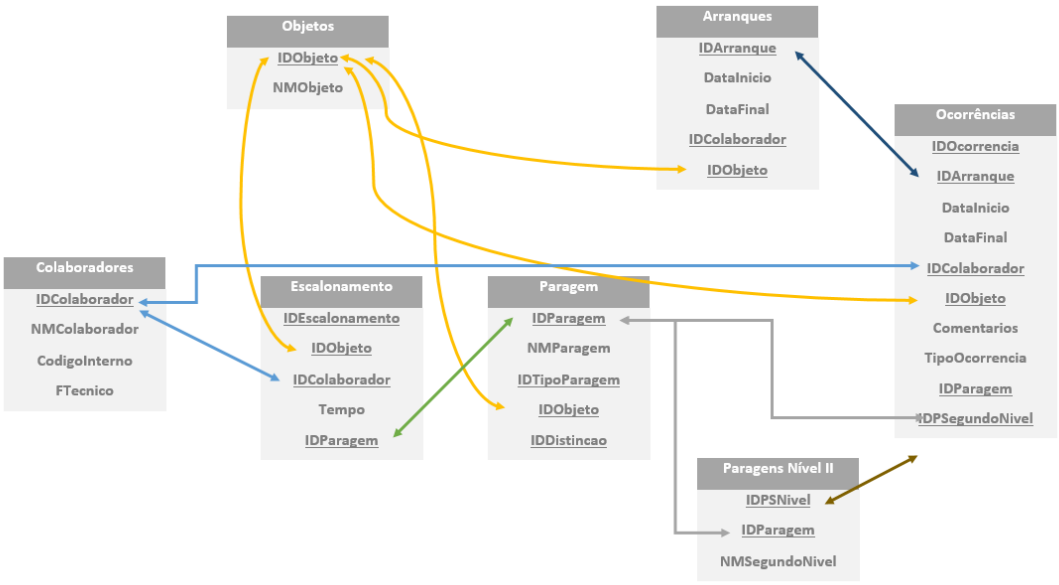



Figura 15 – Estrutura da Base de Dados Aplicação Móvel

#### *4.1.2.2 Bibliotecas Auxiliares*

- **SQLite –** É uma pequena biblioteca [SQLIte, 2015], desenvolvida em *C*, que permite armazenar dados assim como qualquer *SGBD*. Distingue-se por ser *open-source*, e pela sua simplicidade na forma como lê e se escreve para a base de dados existente no disco, a qual se encontra num único ficheiro. A base de dados pode armazenar até 2 Terabytes, e para a sua utilização não é necessário qualquer instalação, configuração ou administração. O SQLite implementa a maioria do *SQL92*, tornando-se fácil o manuseamento das consultas e escritas, sendo muito semelhante ao *SQL Server*. O SQLite está incorporado em vários sistemas móveis, facilitando a criação e usabilidade de base de dados, que se encontram localmente no dispositivo.

- **Joda Time –** Esta biblioteca [Joda Time, 2013] é *open-source* e permite melhorar substancialmente a maneira como tratamos no dispositivo móvel as classes data e hora. Esta biblioteca é fácil de usar, pois contém vários métodos simples e de fácil acesso ao resultado pretendido, que é o melhor tratamento de datas e horas no *Android*. Fornece 8 calendários e permite de uma forma fácil aplicar qualquer fuso horário ou diferenças entre datas. Consegue-se obter um excelente desempenho, tratando rapidamente todos os cálculos, ou dados que se pretende obter, fazendo com que esta biblioteca seja usada na aplicação móvel.

Alguns métodos que se destacam na biblioteca:

- **LocalDate –** Data sem o tempo
- **LocalTime –** Tempo sem a data
- **Datetime –** Data completa com o fuso horário
- **Formatter-Parser –** Permite formatar data/hora

- **KSOAP –** Dada a utilização de *webservices*, e pelo facto de estes responderem com uma *SOAP message*, foi necessária a utilização de uma biblioteca que permitisse a leitura dessas mensagens da forma mais simples possível. Com esse intuito, esta biblioteca foi desenvolvida. A biblioteca *KSOAP* [KSOAP, 2015] permite ler todas as mensagens *SOAP* de uma forma simples e eficaz em qualquer dispositivo *Android*. Esta biblioteca é *open-source* e as suas atualizações são feitas de forma bastante recorrente. Esta biblioteca é usada tanto na aplicação móvel com na aplicação SMS.

- **AchartEngine –** Trata-se de uma biblioteca [AchartEngine, 2011] para *Android*, que permite desenhar no ecrã gráficos. Esta biblioteca disponibiliza vários tipos de gráficos, como por exemplo, gráfico de barras, circulares, bolhas, etc. De acordo com os dados e com a forma que pretendemos apresentá-los, recorrendo a código nativo do *Android*. Esta biblioteca permite diversas personalizações, tais como, cores, zoom, formatações, eixos, podendo equiparar-se a um *Google Charts*, com a vantagem de poder ser utilizada sem aceder à Internet, bastando apenas aceder ao servidor central para visualizar todos os gráficos.

- **Acra –** Esta biblioteca [Acra, 2013] permite gerir várias versões de aplicações, sendo bastante útil para quem programa e disponibiliza diversas versões móveis. Ajuda a encontrar erros decorrentes do uso da aplicação em qualquer dispositivo, guardando todos os dados relevantes sobre esse mesmo erro. Disponibiliza também uma interface web de fácil acesso, que inclui relatórios sobre os erros, permitindo uma análise da periocidade de cada um, chamada *Acralyzer* [Acralyzer, 2014].

#### *4.1.2.3 Estrutura Aplicação Android*

Qualquer projeto desenvolvido através da ferramenta *Android Studio* tem com estrutura base a exemplificada na Figura 16.

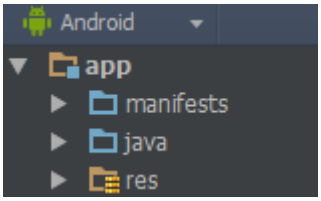



Figura 16 – Estrutura de Projeto Standard *Android Studio*

- **Manifests –** onde se encontra o ficheiro XML de configuração de todas as atividades, serviços ou *Receivers* existentes na aplicação
- **Java –** Todos os pacotes e ficheiros java que irão ajudar no desenrolar da aplicação
- **Res –** Os recursos utilizados pela aplicação, assim como, Strings, imagens, layout's, menu's, dimensões, etc.

A partir da pasta Res, podemos ainda acrescentar subpastas que possam conter as bibliotecas usadas pela aplicação, assim como outro tipo de ficheiro que ajudem na sua execução.

#### *4.1.2.4 Termos Associados a Aplicações Android*

Cada projeto desenvolvido para *Android* terá sempre a definição da estrutura do projeto, este ficheiro tem o nome de *AndroidManifest.xml*. Este ficheiro possui toda a informação sobre a estrutura de atividades que existem no projeto e que de alguma maneira irão interagir durante o desenvolvimento da aplicação. Para além das atividades, estão também enunciados os serviços existentes, caso existam, assim como os *Receivers*. Para tornar o ficheiro completo, fica a faltar a versão do SDK a qual a aplicação se destina. Tendo em conta que o sistema operativo *Android* está dividido por *API's* é necessário indicar entre quais, um mínimo e um máximo, as *API's* em que a aplicação poderá ser executada.

Depois de enumerar as atividades, estas são associadas sempre a um ficheiro *XML*, que contém um desenho daquilo que se pretende mostrar no ecrã do dispositivo. É possível incluir, uma panóplia de objetos, tais como botões, linhas para inserção de texto, imagens, etc.

A estes objetos, podemos ainda atribuir algumas características dessa mesma *Activity*, por exemplo um modo pouco utilizado, mas que neste projeto é usado com frequência, o Modo Imersivo.

Neste modo é possível esconder a barra de notificações, assim como as três teclas existentes na parte inferior de qualquer *Android*, melhorando a experiência ao nível do funcionamento da aplicação, não havendo contato com as restantes funcionalidades e permitindo ao utilizador um foco maior na aplicação.

Os *Receiver's* ou *BroadCast Receiver*, são uma parte importante da aplicação, caso o programador necessite de executar alguma operação no início, no fim ou no decorrer da aplicação. Existem imensas possibilidades, no entanto, estes *Receiver* estão associados a eventos do próprio sistema operativo, no qual podemos efetuar algumas ações a partir do momento que estes acontecem. Registando sempre antes o *Receiver*, ou seja, iniciá-lo para que este aguarde até que determinada ação aconteça. Seguem abaixo alguns exemplos:

- android.intent.action.BATTERY_CHANGED
- android.intent.action.BOOT_COMPLETED
- android.intent.action.REBOOT

Os *Services* no Android trazem consigo mais funcionalidades que os *Receivers*. Estes podem servir a aplicação da mesma forma que os *Receivers*, mas normalmente são usados para outro tipo de operações. Estes serviços podem ser iniciados e terminados sempre que seja necessário, neles podem ser executados diversos tipos de código, sempre em background, para que estes não afetem o desenrolar da aplicação.

Em qualquer aplicação *Android* é possível guardar os dados genéricos, para que o utilizador não tenha que no início da aplicação definir esses mesmos dados. Para esse fim, existem as *SharedPreferences*. Esta classe providencia uma *framework* que permite gravar e ler alguns dados introduzidos no formato Key-Value, ou seja, cada preferência tem uma Key associada a um value. É possível gravar vários tipos de dados primitivos, tais como: *floats*, *Strings*, *ints*.

Para além das *SharedPreferences*, existem outros métodos e formas para guardar os dados no dispositivo. Uma delas pode verificar-se no uso da base de dados disponibilizada pelo *Android*, o *SQLite*. A classe *SQLiteOpenHelper,* ajuda no manuseamento dos dados. Nesta classe definem-se os vários métodos de leitura, gravação ou inserção na base de dados, tornando-se disponíveis para qualquer atividade da aplicação, ou seja, sempre que é necessário fazer alguma consulta à base de dados. É também aqui que se define a estrutura da base de dados, ou seja, tabelas, campos, tipo de dados, nome da base de dados, caminho físico onde se encontra base de dados, etc.

Quando é necessário executar um pedido extenso à base de dados local ou então um pedido a um servidor web, o termo *Mutlitasking* surge como forma para solucionar o problema, de forma a que o processo principal nunca seja afetado. Para ajudar no *Multitasking*, o *Android* disponibiliza para além das *threads*, que permitem alterar o aspeto gráfico, caso sejam iniciadas pela mesma *thread* que iniciou a interface, a classe *AsyncTask*. Esta classe permite executar operações numa *thread* à parte, ficando a *thread* principal à espera do resultado final da execução da *AsyncTask*. Esta é composta por quatro fases, o *onPrexecute*, onde se executa normalmente os pré-requisitos necessários para executar a tarefa. *DoInBackground*, onde decorrem todas as ações que precisam de ser executadas, ou seja, onde se encontram todos os cálculos ou chamadas a serviços externos.

Durante este processo existe ainda o *onProgressUpdate*, que permite saber qual o estado de evolução das tarefas a serem executadas no *DoInBackground*.

Por fim, temos o *onPostExecute*, que já depois de todas as tarefas terem terminado, executa ações finais, de modo a prosseguir com o restante da aplicação.

### 4.1.3 Arquitetura Plataforma Web

O desenvolvimento da Plataforma Web foi concebido em *HTML* juntamente com *JavaScript*. Para o seu desenvolvimento foi necessário utilizar algumas bibliotecas *open-source* para facilitar a conceção da plataforma. Através do uso destas bibliotecas, a plataforma foi melhorada a nível de interatividade, assim como ao nível de funcionalidades.

A Figura 17 representa a estrutura da Plataforma Web.

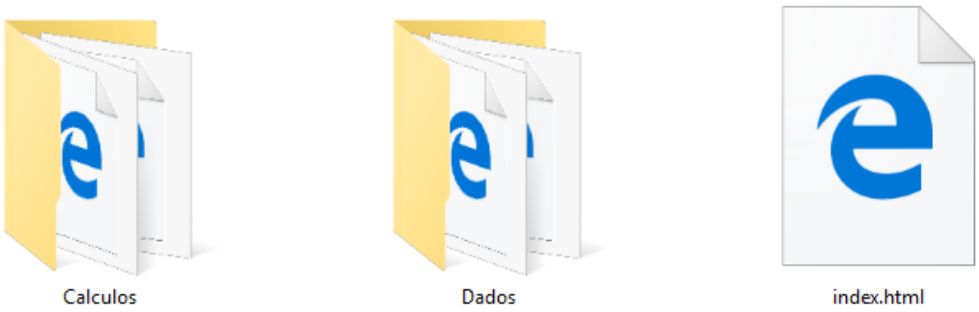



Figura 17 – Estrutura diretório Plataforma Web

#### 4.1.3.1 Bibliotecas Auxiliares

- **Bootstrap –** Projetada [BootStrap, 2015] para ajudar no desenvolvimento de aplicações web. É uma *framework front end* que inclui estilos *CSS* para ícones, botões, tabelas, navegação, etc. Juntamente com *plug-ins jQuery*, torna as aplicações web mais interativas e mais fáceis na sua usabilidade.

- **Bootbox –** Uma pequena biblioteca [BootBox, 2015] que ajuda na criação de caixas de diálogo, utilizando a biblioteca *bootstrap* em conjunto com as suas *modals*. A vantagem do uso desta biblioteca verifica-se na criação e manuseamento das caixas de diálogo, uma vez que a biblioteca gere automaticamente todas as opções. Um exemplo disso, podemos ver na Figura 18, onde bastaram três linhas de código, para criar uma caixa de diálogo.

```
bootbox.alert("Olá Mundo!", function() {
    //função CALLBACK
});
```

Figura 18 – Exemplo de criação de uma caixa de diálogo

- **HighCharts -** Baseada em *HTML5*, esta biblioteca [HighCharts, 2015] desenvolvida em *JavaScript*, permite adicionar a qualquer página web gráficos interativos, de uma forma simples. Esta é compatível com qualquer browser web e permite gráficos de vários tipos, entre os quais se destacam o gráfico de linhas, área e colunas. A sua configuração é simples e dinâmica, dando a possibilidade de definir vários tipos de configuração, dependendo sempre dos dados que se pretende demonstrar.

- **Highcharts-export-client-side –** Esta biblioteca serve de apoio ao *HighCharts*, permitindo a exportação dos gráficos desenhados pelo *HighCharts*. A biblioteca *Highcharts* já contém o módulo de exportação, apenas com uma dependência, esta precisa de obter ligação ao *WebSite* da biblioteca para a sua exportação. Com este módulo à parte da biblioteca mãe, essa ligação torna-se desnecessária, pois facilmente se consegue a exportação na parte do cliente, sem necessidade de aceder um servidor externo. Este módulo permite as mesmas exportações que a biblioteca mãe, ou seja, exportação para um ficheiro de imagem, exportação para PDF, exportação para Excel, etc.

- **Tablesorter –** Com a criação de tabelas nas páginas web, torna-se necessária a criação de ferramentas para a sua ordenação. Esta biblioteca permite após a criação de tabelas, associar a essa mesma tabela as respetivas funções de ordenação. Para além da ordenação, permite configurar à partida, os campos possíveis para ordenação assim como múltiplas ordenações. No que diz respeito à ordenação, permite a ordenação de vários tipos de campos, sejam eles, datas, números, textos, URL's, etc.

- **Jquery -** Esta biblioteca [Jquery, 2015] tem como principal objetivo facilitar a usabilidade na criação de *Script's* e no manuseamento de objetos *HTML*, *CSS*, efeitos, *AJAX*, etc. A interação torna-se bastante mais fácil assim que a incluímos em qualquer página web, qualquer objeto HTML está acessível com poucas linhas de código e a atribuição de estilos CSS a qualquer objeto HTML, também fica acessível com poucas linhas de código. Para além de estilos e objetos *HTML*, as chamadas *AJAX* são bastantes simplificadas e a sua utilização é completamente personalizada.

- **Bootstrap-select –** Na utilização de listas de seleção, esta biblioteca oferece mais funcionalidades, para além das normais que todas as listas possuem originalmente. Por norma as listas possuem uma lista de objetos, e apenas um é selecionável, utilizando esta biblioteca torna-se possível a multiselecção de campos, assim como atribuir um estilo idêntico à biblioteca *BootStrap*, já referida anteriormente. Para além da multiselecção,

esta permite agrupar dentro das listas, os campos de modo a distingui-los de forma a facilitar a identificação do que selecionar na lista apresentada.

- **Canvas-tools –** Para o desenho dos gráficos pela biblioteca *HighCharts*, é necessário ter na página web a biblioteca *Canvas-tools*, que serve de apoio ao desenho dos gráficos. Esta biblioteca, para além de ser uma ajuda no desenho dos gráficos, permite desenhar objetos dentro do espaço *<canvas>* de acordo com as necessidades do utilizador.

## 4.2 Implementação do Sistema

Neste capítulo será apresentada toda a implementação do sistema. De início, será abordada a implementação dos *webservices*, de seguida a aplicação móvel OEE, prossegue-se com a exemplificação da aplicação móvel SMS e por fim será apresentada a plataforma web.

### 4.2.1 Implementação *WebServices*

A implementação ao nível do servidor, foi realizada de acordo com o que se encontra representado na Figura 19.

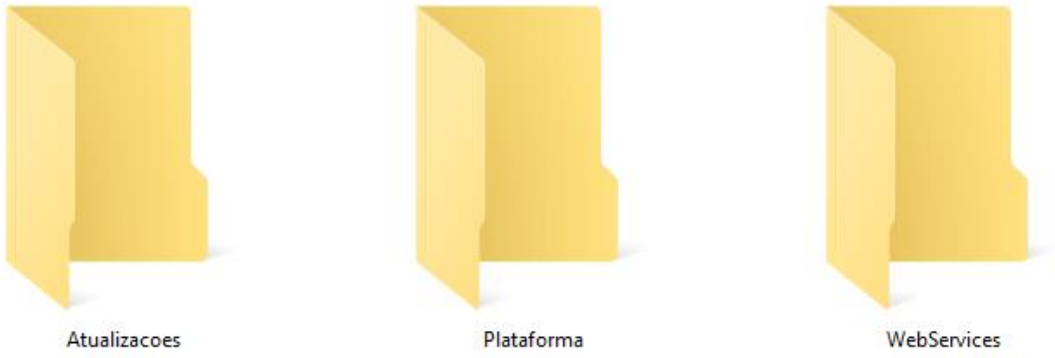



Figura 19 – Raiz do Servidor

Esta estrutura está dividida em três diretórios, como indicado na Figura 19, em que a pasta "Atualizacoes", representa a pasta onde se encontram todas as atualizações referentes à aplicação, tornando a aplicação autonomamente atualizável, não havendo necessidade de a publicar numa outra plataforma. Associado a este diretório está um *Webservice* correspondente, que verifica sempre qual a última versão existente nesse mesmo diretório.

A pasta “Plataforma”, contempla o diretório onde existem todas as páginas web relacionadas com a plataforma Web que irá ser disponibilizada para consulta dos dados recolhidos por todos os dispositivos.

Por fim, a pasta “WebServices”, é o local onde se encontram os ficheiros referentes aos *webservices*, que irão ajudar na consulta, inserção de dados e na atualização da versão da aplicação móvel.

```
internal string ObterVersao(string url)
        {
            var filepath = string.Empty;
            var filename = string.Empty;
            var lista = new List<WebDavResource>();
            try
            {
                //ACEDE A PASTA DADO PELO URL COM DADOS DE LOGIN E PASSWORD
                var webDav = new WebDavManager {Credential = new WebDavCredential('USERNAME', 'PASSWORD', AuthType.Ntlm)};
                lista = webDav.List(url);

                var startDate = DateTime.MinValue;
                //IRÁ VERIFICAR TODOS OS FICHEIROS DA PASTA CORRESPONDENTE
                foreach (var webDavResource in lista)
                {
                    if (webDavResource.IsDirectory) continue;

                    if (webDavResource.Modified <= startDate) continue;

                    startDate = webDavResource.Modified;
                    filename = webDavResource.Name;
                }

                var localpath = new Uri(url + filename).LocalPath;
                filepath = "\\\\localhost" + localpath.Replace("/", @"\");

                //USO DE UMA FERRAMENTE PARA VERIFICACAO DO NOME DA VERSAO E CODIGO DA VERSAO
                var info = ApkReader.ReadApkFromPath(filepath.ToString(CultureInfo.InvariantCulture));
                //DADOS A RETORNAR PELO METODO
                return filename + info.VersionName + info.VersionCode;
            }
            catch (Exception ex)
            {
                throw new Exception('Retorna o Erro');
            }
        }
```

Figura 20 – Exemplo de Código de obtenção da última versão

Na Figura 20, está representado um código exemplo do *webservice* que será executado sempre que a aplicação móvel seja iniciada. Este irá sempre procurar dentro do diretório “Updates”, a versão mais recente em termos de data, e caso exista, irá comparar as versões dentro de cada ficheiro *APK* existente nesse mesmo diretório.

```
internal string ConsultarDados(string query, string connStr)
        {
            using (var BD = new SqlConnection(connStr))
            {
                if (BD.State != ConnectionState.Open)
                    BD.Open();

                BD.InfoMessage += CnInfoMessage;

                var result = BD.Execute(query);

                while (BD.State != ConnectionState.Closed)
                    BD.Close();
                var err = ErrosSQL.ToArray();
                var obj = new Dictionary<string, object[]> { { result.ToString(CultureInfo.InvariantCulture), err} };

                var objJson = JsonConvert.SerializeObject(obj);

                return objJson;
            }
        }
```

Figura 21 – Exemplo de Código de Consulta de Dados

Na Figura 21, é representado um código exemplo do *webservice* relativo à consulta de dados. Este recebe dois parâmetros, um deles, uma *connectionString*, com os dados de conexão à base de dados, e outro com a *query* a executar na base de dados. Este mesmo *webservice* devolverá uma resposta em *JSON*, de forma a ser percetível e poder ser utilizada, quer na plataforma Web, como nas aplicações móveis (OEE e SMS).

```
internal int InsercaoDados(string query, string connStr)
        {
            using (var BD = new BaseDataContext(connStr))
            {
                    if (BD.Connection.State != ConnectionState.Open)
                        BD.OpenConnection();

                    var result = BD.ExecuteNonQuery(query);

                    while (BD.Connection.State != ConnectionState.Closed)
                        BD.CloseConnection();
                    return result;
            }
        }
```

Figura 22 – Exemplo de Código de Inserção de Dados

Na Figura 22, está representado um código exemplo semelhante ao código já demonstrado na consulta de dados, Figura 21, recebendo também dois parâmetros, a *connectionString* e a *query* a executar, apenas divididos para separar a quando de cada chamada de modo a poder-se facilmente distinguir o que qualquer código irá executar na chamada ao *webservice*.

### 4.2.2 Implementação Aplicação Móvel OEE

#### *4.2.2.1 Activitys*

As *Activitys* são usadas em quase todos os projetos e são uma representação visual de uma aplicação. Estas atividades podem conter inúmeras *views* e *fragments* para criar uma interface, através da qual o utilizador final irá interagir para usufruir da aplicação.

As *Activitys* são independentes umas das outras, podendo ser chamadas no momento da execução do código *startActivity*. Ao executar este código, a *Activity* existente, caso haja, irá passar para um outro estado, parada, de forma a que a próxima *Activity* apareça no ecrã.

```
Intent proximaatividade = new Intent(NomeAtividadepresente.this, NomeParagemSeguinte.class);
//FLAGS possíveis para acrescentar quando se transita entre Atividades
proximaatividade.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
proximaatividade.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
proximaatividade.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
//Inicia a nova Atividade
NomeAtividadepresente.this.startActivity(proximaatividade);
```

Figura 23 – Exemplo de mudança entre Atividades

#### *4.2.2.2 AsyncTask*

Para poder comunicar com o servidor, é necessário usar uma classe disponibilizada pelo *Android* denominada *Asynctask*. Esta é executada numa *Thread* à parte da *Activity,* a qual se encontra ativa, tornando possível o uso das duas em simultâneo.

Esta classe possui 4 métodos dos quais se pode tirar partido para fazer uso da mesma. Esta possui uma sequência bem definida, que começa pelo método *onPreExecute*.

```
@Override
protected void onPreExecute() {
    //INICIO DA ASYNCTASK
    super.onPreExecute();
}
```

Figura 24 – Exemplo Código *onPreExecute*

Dentro deste método, é necessário preparar todo um conjunto de variáveis que podem ser usadas no método a seguir, o *doInBackground*.

```
@Override
protected Void doInBackground(Void... arg0) {
//DURANTE O PROCESSO TODO
//TODOS OS CALCULOS E CHAMADAS MAIS LONGAS
    ....
}
```

Figura 25 – Exemplo Código *doInBackground*

É neste que se deve concentrar maior parte das ações, a tomar e executar, assim como pedidos ao servidor, cálculos, etc. Durante estas ações, a classe *AsyncTask* permite dar feedback ao utilizador que solicitou o pedido através da interface da aplicação.

```
@Override
protected void onProgressUpdate(Void... values) {
    //ONDE PODEMOS TER ACESSO AO GUI E MOSTRAR AO UTILIADOR O PROGRESSO
    //DO METODO DOINBACKGROUND
}
```

Figura 26 – Exemplo Código *onProgressupdate*

Aqui surge o método *onProgressUpdate*, o único método que tem a possibilidade de atualizar a interface existente na *thread* principal, de forma a informar o utilizador em que estado se encontra o seu pedido.

Por fim, temos o método *onPostExecute*, que é responsável pelas tarefas finais, já com as tarefas mais longas e pesadas executadas, este apenas finaliza e prepara o resultado para o utilizador final poder visualizar na sua interface na aplicação.

```
@Override
protected void onPostExecute(Void result) {
    //NO FINAL DE CADA ASYNCTASK
    ....
    super.onPostExecute(result);
}

-
```

Figura 27 – Exemplo Código *onPostExecute*

#### *4.2.2.3 Base de Dados SQLite*

Como já demonstrado anteriormente, na Figura 14, a base de dados da aplicação móvel é composta por 7 tabelas. A tabela objetos, é responsável por conter os dados referentes ao dispositivo que está a ser utilizado. Esta tabela apenas terá um único registo, servindo apenas para identificar o dispositivo e dar informações acerca do mesmo.

Na tabela Colaboradores, Escalonamento, Paragem, Paragens Segundo Nível, serão listadas todas as informações acerca dos quatro temas referidos por cada título de tabela, ou seja, na parte dos Colaboradores, o dispositivo irá ter a informação de todos os Colaboradores existentes no sistema. Na parte do Escalonamento, Paragem e Paragem Segundo Nível, apenas irá conter dados acerca do dispositivo em questão. Todos estes dados são sempre atualizados a cada finalização de um arranque.

As tabelas restantes, Arranques e Ocorrências, serão preenchidas ao longo do decorrer da aplicação, e terão informações detalhadas sobre cada arranque que ocorreu no dispositivo, assim como sobre cada ocorrência que existiu referente a esse mesmo arranque. No fim de cada Arranque para além de atualizar as restantes tabelas, estas serão enviadas para o servidor, de modo a que os registos existentes no dispositivo possam ser eliminados.

Para esta função, é necessário existir uma classe de ajuda à manipulação dos dados existentes e à inserção dos mesmos. Dentro desta classe existem sempre dois métodos, um de iniciação de ligação à base de dados e um de fecho de ligação à base de dados. Pelo meio existem métodos dos mais variados tipos, por forma ser possível trabalhar a informação. Na Figura 28 é possível visualizar um exemplo de como está representada essa classe.

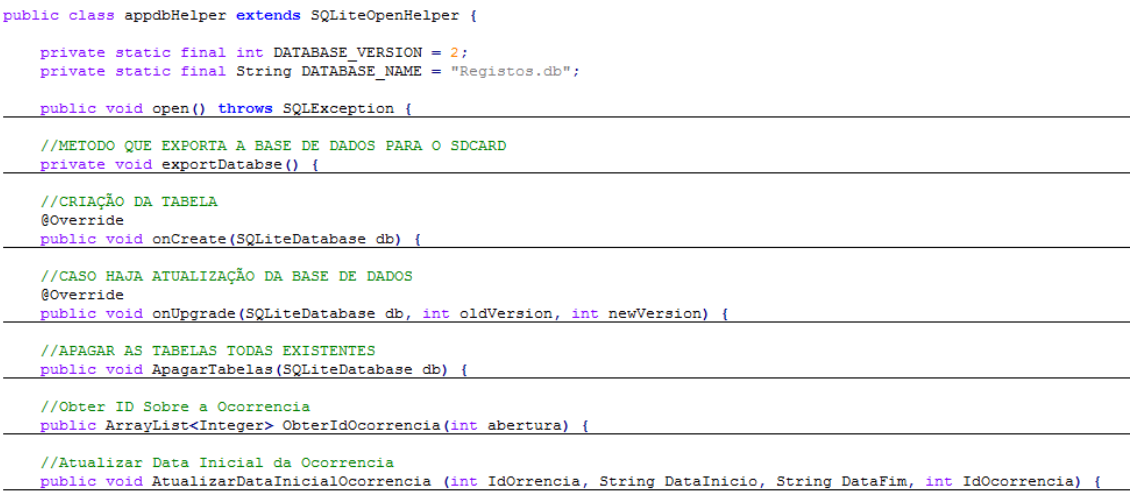



Figura 28 – Exemplo Código Manuseamento Base de Dados *SQLite*

#### *4.2.2.4 WebServices*

A chamada aos *WebServices* está sempre relacionada com o uso da classe *Asynctask*, como já referido anteriormente. Assim como no manuseamento da base de dados, existe uma classe apenas para tratamento deste tipo de chamadas. Dentro desta mesma classe podem coexistir métodos diferentes, com diferentes parâmetros, de forma a satisfazer as necessidades do utilizador. Todos os *Webservices* existentes que a aplicação usará retornam sempre um objeto *JSON*, no entanto é necessário ler o envelope enviado em *SOAP* para chegar ao objeto em *JSON*. Como já dito anteriormente, para esse efeito utiliza-se a biblioteca *KSOAP* e a partir dessa biblioteca é possível obter a mensagem desse envelope recebido, que contém um objeto *JSON* pronto a ser consumido pela aplicação móvel.

Para além de interpretar os objetos *JSON* do servidor é necessário também existir métodos de envio de dados para esse mesmo servidor, dos quais podem divergir dos métodos de receção dos objetos.

Na Figura 29, está representado um método de envio de dados para o servidor.

```
public List<String> getJSON(String query, String namespace, String method, String soap, String conn, String url) {
    List<String> resultado = new ArrayList<String>();
    //CRIAÇÃO DO OBJETO SOAP
    //METHOD - METODO EXISTENTE NO WEBSERVICE
    //NAMESPACE - NORMALMENTE http://tempuri.org/
    SoapObject request = new SoapObject(namespace, method);
    //ADICIONAR OS DOIS CAMPOS NECESSÁRIOS NA EXECUTAÇÃO DO METÓDO
    request.addProperty("query", query);
    request.addProperty("connStr", conn);
    //SERIALIZAÇÃO DO ENVELOPE A ENVIAR
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.setOutputSoapObject(request);
    envelope.dotNet = true;
    try {
        //CHAMADA AO WEBSERVICE
        final HttpTransportSE androidHttpTransport = new HttpTransportSE(url,20000);
        androidHttpTransport.debug = true;
        androidHttpTransport.call(soap, envelope);
        //RECOLHEMOS O ENVELOPE SOAP
        SoapObject result = (SoapObject) envelope.bodyIn;
        if (result != null) {
            //TRANSFORMAÇÃO DO ENVELOPE EM EM JSONARRAY
            final JSONArray a = new JSONArray(result.getPrimitivePropertyAsString(getResources().getString(R.string.jsonresult)));
            //PERCORRER OS DADOS TODOS DO JSONARRAY
            for (int i = 0; i < a.length(); i++) {
                final JSONArray innerJsonArray = a.getJSONArray(i);
                for (int j = 0; j < innerJsonArray.length(); j++) {
                    final JSONObject jsonObject = innerJsonArray.getJSONObject(j);
                    resultado.add(jsonObject.getString("Value"));
                }
            }
        } else {
            resultado.add('Dados Inválidos');
            return resultado;
        }
    } catch (Exception e) {
        resultado.add('Erro na comunicação');
        e.printStackTrace();
    }
    //Retorno do resultado
    return resultado;
}
```

Figura 29 – Exemplo Código de Chamada ao *WebService*

#### *4.2.2.5 Comunicação NFC*

Para realizar a comunicação entre a etiqueta NFC e o dispositivo de escrita apenas é necessário o uso de umas das *API's* já existentes no *Android*, o *NFCAdapter*. É nela que se baseiam as classes de ajuda a leitura da etiqueta e configuração da mesma na aplicação móvel.

Para criarmos uma mensagem, para além do dispositivo conter o sistema *NFC* e de este estar ativo, é necessário criar uma mensagem para a etiqueta ser reconhecida na aplicação móvel. Os dados preenchidos na etiqueta, estão codificados em base64, e são apenas percetíveis pela aplicação, para evitar qualquer tipo de falsificação.

Na Figura 30, podemos visualizar um exemplo de criação de uma mensagem para uma etiqueta *NFC*.

```
private static boolean EscreverNFCTag(Context context, Tag tag, String data) {

    //ENCODIFICAÇÃO DA MENSAGEM
    data = Base64.encodeToString(data.getBytes(), DEFAULT);
    NdefRecord appRecord = NdefRecord.createApplicationRecord(context.getPackageName());
    NdefRecord relayRecord = new NdefRecord(NdefRecord.TNF_WELL_KNOWN, null, null, data.getBytes());
    NdefMessage message = new NdefMessage(new NdefRecord[] {relayRecord, appRecord});
    try {
        Ndef ndef = Ndef.get(tag);

        if(ndef != null) {
            ndef.connect();
            //VERIFICAÇAO SE É POSSIVEL ESCREVER NA ETIQUETA
            if(!ndef.isWritable()) {

                return false;
            }
            //VERIFICAÇÃO DO TAMANHO TOTAL DA MENSAGEM A ESCREVER
            int size = message.toByteArray().length;
            if(ndef.getMaxSize() < size) {

                return false;
            }
            //ESCRITA DA MENSAGEM NA ETIQUETA
            try {
                ndef.writeNdefMessage(message);
                return true;
            } catch (TagLostException tle) {
                return false;
            } catch (IOException ioe) {
                return false;
            } catch (FormatException fe) {
                return false;
            }
        } else {
            NdefFormatable format = NdefFormatable.get(tag);
            if(format != null) {
                try {
                    format.connect();
                    format.format(message);

                    return true;
                } catch (TagLostException tle) {
                    return false;
                } catch (IOException ioe) {
                    return false;
                } catch (FormatException fe) {
                    return false;
                }
            } else {
                return false;
            }
        }
    } catch(Exception ignored) {
    }

    return false;
}
```

Figura 30 – Exemplo Código Escrita NFC

Esta irá conter os dados já existentes na aplicação móvel na parte administrativa, e irá preencher com o nome e número do colaborador respetivo nessa mesma etiqueta.

Para leitura da mesma, como já referido, é necessário descodificar de base 64 a mensagem recebida, e é esse método que é exemplificado na Figura 30. A partir do momento que se obtém a mensagem na íntegra, é possível verificar se aquele colaborador tem permissões para executar a tarefa desejada pela aplicação.

```
protected void onNewIntent(Intent intent) {
    //AO SER RECONHECIDA A TAG DISPUTA UM INTENT
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction()) ||
        NfcAdapter.ACTION_TECH_DISCOVERED.equals(intent.getAction()) ||
        NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction())) {
        NdefMessage[] msgs;
        Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
        if (rawMsgs != null) {
            msgs = new NdefMessage[rawMsgs.length];
            for (int i = 0; i < rawMsgs.length; i++) {
                msgs[i] = (NdefMessage) rawMsgs[i];
            }
            NdefRecord[] recs = msgs[0].getRecords();
            byte[] value = recs[0].getPayload();

            try {
                value = Base64.decode(value, DEFAULT);
                String str = new String(value, "UTF-8");
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }

}
```

Figura 31 – Exemplo Código Leitura NFC

#### *4.2.2.6 Biblioteca de Gráficos*

Uma das funcionalidades da parte administrativa é poder visualizar na própria aplicação gráficos que representam os valores obtidos por cada objeto na aplicação.

Para isso, e como já referido anteriormente, é necessário a biblioteca *achartengine*, que é responsável pelo desenho dos gráficos. Os dados necessários para a elaboração dos gráficos são recolhidos através de *webservice*. A Figura 32, mostra um exemplo com dados aleatórios e sem recorrer a *webservices,* de como se pode construir um gráfico de maneira facilitada com a biblioteca *achartengine*.

```
void DesenharGrafico(int param1, int param2, int param3, int param4, int param5, String param6)
{
    //Calculos e preparação dos Arrays para demonstrar no grafico

    int[] colors = new int[] { Color.parseColor("#558ED5"), Color.parseColor("#C0504D")};
    XYMultipleSeriesRenderer renderer = buildBarRenderer(colors);
    setChartSettings(renderer, 'Titulo do Gráfico', "", 'Percentagem (%)', 0.5,
            6, 0, 110, Color.BLACK, Color.BLACK, titles2);
    //Propriedades ao fazer renderer do Gráfico
    renderer.setXLabels(12);
    renderer.setYLabels(10);
    renderer.setDisplayChartValues(true);
    renderer.setZoomButtonsVisible(false);
    renderer.setClickEnabled(false);
    renderer.setExternalZoomEnabled(false);
    renderer.setInScroll(true);
    renderer.setShowLegend(false);
    renderer.setBarSpacing(0.5f);
    renderer.setChartTitleTextSize(25);
    renderer.setZoomEnabled(false, false);
    renderer.setPanEnabled(true, false);
    renderer.setXLabels(0);
    renderer.setXLabelsAlign(Paint.Align.CENTER);
    renderer.setXLabelsColor(Color.BLACK);
    renderer.setShowAxes(false);
    renderer.setChartTitleTextSize(25f);

    if (mChartView == null) {
        //Adiciona o Gráfico a View caso ele não existe
        mChartView = ChartFactory.getBarChartView(getActivity(), buildBarDataset(titles, values), renderer, BarChart.Type.STACKED);
        chartperiodo.addView(mChartView, new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT, 1000));
    } else {
        //Caso Exista já algum Gráfico na View ele execute o repaint() com os novos dados
        mChartView.repaint();
    }
}
```

Figura 32 – Exemplo de Código Desenho Gráficos

```
void setChartSettings(XYMultipleSeriesRenderer renderer, String title, String xTitle,
                      String yTitle, double xMin, double xMax, double yMin, double yMax, int axesColor,
                      int labelsColor, String[] titles) {
    renderer.setChartTitle(title);
    renderer.setChartTitleTextSize(25f);
    renderer.setYLabelsAlign(Paint.Align.CENTER);
    renderer.setXTitle(xTitle);
    renderer.setYTitle(yTitle);
    renderer.setChartValuesTextSize(15f);
    renderer.setXAxisMin(xMin);
    renderer.setXAxisMax(xMax);
    renderer.setYAxisMin(yMin);
    renderer.setYAxisMax(yMax);
    renderer.setAxisTitleTextSize(20f);
    renderer.setMargins(new int[] { 10, 65, 10, 15 });
    renderer.setAxesColor(axesColor);
    renderer.setLabelsColor(labelsColor);
    renderer.setShowAxes(false);
    renderer.setXLabels(0);
    renderer.setXLabelsAlign(Paint.Align.CENTER);
    renderer.setXLabelsColor(Color.BLACK);

    for (int i = 1; i < titles.length; i++) {
        renderer.addXTextLabel(i, titles[i]);
    }

}
```

Figura 33 – Exemplo de Código configuração Gráficos

### 4.2.3 Implementação Aplicação Móvel SMS

#### *4.2.3.1 Envio de SMS*

Já fora de aplicação móvel OEE, existe a aplicação das SMS, que de certa forma complementa a aplicação móvel OEE. É responsável pelo envio das SMS geradas pela aplicação OEE e garante que todos receberão os avisos a tempo.

Esta aplicação recorre a uma API já existente no *Android* para o envio das SMS, denominada *SMSManager*. Para que funcione corretamente, é necessário garantir que existe um cartão SIM ativo e com possibilidade de enviar SMS. A partir desse momento, a aplicação irá entrar num ciclo com um tempo configurável, para verificar a lista de espera existente, numa tabela do servidor principal.

```
public void GetSMS()
{
    //CASO HAJA WIFI LIGADO
    WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
    int state = wifi.getWifiState();
    if(state == WifiManager.WIFI_STATE_ENABLED) {

       //   QUERY AO WEBSERVICE
        ArraList<String> resposta = new ArrayList();
        reposta.addAll('Resultado query Webservice');
        if (resposta.size() != 0)
        {
                EnvioSMS(Parametros);
        }
    }
    else
    {
        //Envia SMS de Erro
        new AsyncWarnSMS().execute("Wifi desativo");
    }
```

Figura 34 – Exemplo de Código de recolha de mensagem da Lista de Espera

Na Figura 35, é exemplificado o método de envio das SMS, após receção de dados do servidor.

```
private void sendSMS(String mobNo, String message, int IDD, String user, String machine,
 String APP, String TipoSMS,String NomeSMS,int idsms, Boolean cc) {
        ArrayList<String> Mensagem = new ArrayList<String>(1);
        //CRIAÇÃO DA MENSAGEM
        Mensagem.add("[" + APP + "] " + TipoSMS + " \n" + message);
        String smsSent = "SMS_SENT";
        String smsDelivered = "SMS_DELIVERED";
        final String SENT = "SMS_SENT";
        final String DELIVERED = "SMS_DELIVERED";
        ArrayList<PendingIntent> deliveryIntents = new ArrayList<PendingIntent>();
        ArrayList<PendingIntent> sentIntents = new ArrayList<PendingIntent>();
        final SmsManager sms = SmsManager.getDefault();
        //NECESSÁRIO CRIAR DOS INTENT'S
        PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
        PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0, new Intent(DELIVERED), 0);
        ArrayList<String> parts = sms.divideMessage(Mensagem.get(0));
        int messageCount = parts.size();
        for (String part : parts) {
            sentIntents.add(sentPI);
            deliveryIntents.add(deliveredPI);
        }
        // RECEIVER PARA VERIFICAR O ESTADO DO ENVIO DA SMS
        sender = new BroadcastReceiver() {
        registerReceiver(sender, new IntentFilter(smsSent));

        // RECEIVER PARA VERIFICAR SE FOI ENTREGUE A SMS
        receiver = new BroadcastReceiver() {
        registerReceiver(receiver, new IntentFilter(smsDelivered));

        ArrayList<String> msgStringArray = sms.divideMessage(Mensagem.get(0));
        sms.sendMultipartTextMessage(mobNo, null, msgStringArray, sentIntents, deliveryIntents);
    }
```

Figura 35 – Exemplo de Envio de SMS

### 4.2.4 Implementação Plataforma Web

#### *4.2.4.1 Desenho Gráficos Plataforma*

Para criar o desenho sobre a página web disponibilizada pela plataforma é utilizada a biblioteca *Highcharts*. O seu funcionamento está sempre dependente da chamada através da função *.highchart().* A mesma poderá ser parametrizada conforme demonstra a Figura 36.

```
function DesenharGrafico(categorias, valoresdatasetum, titulodatasetum, titulodatasetdois, valuesdatasetdois,titulografico)
    {
        $('#container').highcharts({
            chart: {
            type: 'column'
        },
        title: {
            text: titulografico
        },
        xAxis: {
            categories: categorias
        },
        credits: {
            enabled: false
        },
        exporting: {
            filename: titulografico,
            enabled: false,
            sourceWidth: 1280,
            sourceHeight: 720
        },
        yAxis: {
            title: {
                text: 'Percentagem (%)'
            },
            stackLabels: {
                enabled: false,
                style: {
                    fontWeight: 'bold',
                    color: (Highcharts.theme && Highcharts.theme.textColor) || 'gray'
                }
            }
        },
        tooltip: {
            formatter: function () {
                return '<b>' + this.x + '</b><br/>' +
                    this.series.name + ': ' + this.y + '%<br/>';
            }
        },
        plotOptions: {
            column: {
                stacking: 'normal'
            },
            series: {
                        borderWidth: 0,
                        dataLabels: {
                            enabled: true,
                            formatter: function () {
                                return (this.y == null)? '' : this.y + '%<br/>';
                            },
                            color : 'white'
                        }
                    }
        },
        series: [{ name : titulodatasetum, data: valoresdatasetum},{ name : titulodatasetdois, data: valuesdatasetdois}]
    });
    }
```

Figura 36 – Código de Desenho de Gráfico na Plataforma

Após a atribuição de uma parte da página, uma * *, é possível configurar a biblioteca *HighCharts* de acordo com as necessidades.

#### *4.2.4.2 Chamadas ao Webservice*

Antes de desenhar qualquer gráfico, é necessário recolher dados para apresentar nesses mesmos gráficos. Incluído com o *jQuery*, a função *$. ajax* permite efetuar uma chamada aos *WebServices* existentes de forma a ir recolher dados a base de dados consoante o pedido efetuado. Assim como o desenho de qualquer gráfico, esta função também pode ser parametrizada da maneira mais conveniente, assim como demonstra a Figura 37.

```
$.ajax({
    contentType: "application/json; charset=utf-8",
    type: "POST",
    crossDomain: true,
    url: URLDESTINO,
    headers: { "Access-Control-Allow-Origin": "*" },
    data: "{query : " + JSON.stringify(QuerySistema) + ",connStr : '" + CONNECTIONSTRING + "'}",
    cache: false,
    success: function (msg) {
    var o = jQuery.parseJSON(msg.d);
    },
    error: function (xmlHttpRequestMainContentResult, textStatusMainContentResult, errorThrownMainContentResult) {

        if (xmlHttpRequestMainContentResult.status != 0 && xmlHttpRequestMainContentResult.responseText != '') {
            alert(xmlHttpRequestMainContentResult.status);
            alert(xmlHttpRequestMainContentResult.responseText);
        }
        textStatusMainContentResult.toString();
        errorThrownMainContentResult.toString();
    }
});
```

Figura 37 – Exemplo de chamada ao *WebServices*

## 4.3 Layout do Protótipo

### 4.3.1 Layout da Aplicação Móvel OEE

Após toda a análise efetuada sobre todo o sistema no background, é necessário desenhar a interface pelo qual os utilizadores finais irão ter acesso para usarem e contribuírem para que todo o sistema funcione na normalidade.

#### *4.3.1.1 Aplicação Móvel OEE (Operativa)*

Para que todos os utilizadores presentes no dispositivo possam exercer as suas funções, irão dispor de alguns ecrãs para efetuarem todos os registos. Cada vez que chegarem aos postos de trabalho, no dispositivo irão ter sempre um ícone da aplicação para executarem. Depois de executar, esta irá sempre verificar se existe alguma atualização disponível, caso não haja, e o endereço de IP esteja com a permissão Operativa, esta irá prosseguir com a aplicação até ao ecrã principal, apresentado na Figura 38.

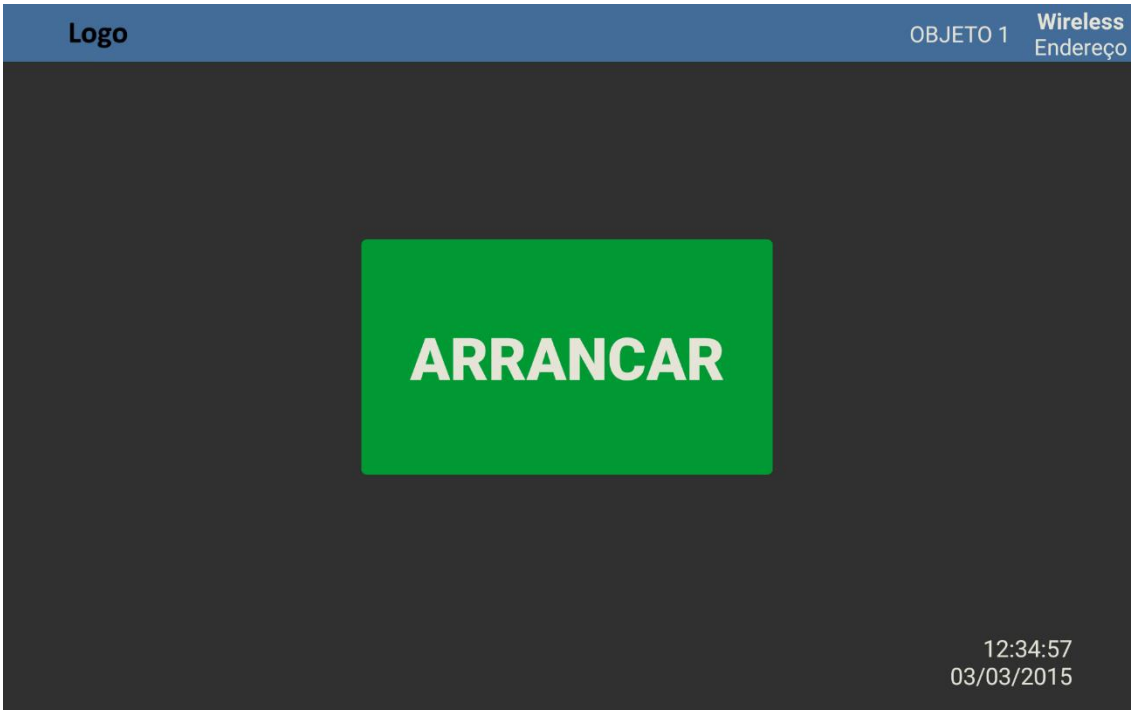



Figura 38 – Layout do Ecrã de Arranque

Neste ecrã ser-lhe-ão apresentadas algumas informações sobre qual o objeto em questão, horas, e um botão para darem início o seu trabalho.

Ao iniciarem o seu trabalho passam para um outro ecrã, Figura 39, o qual apresenta os três estados possíveis de cada objeto, estado em produção, o estado em paragem, e o estado fechado.


Figura 39 – Layout Ecrã Principal

Caso o estado seja em Produção, o ecrã não irá sofrer nenhuma alteração, permitindo ao colaborador prosseguir normalmente com a sua atividade. Caso este entre em estado paragem, nesse caso o ecrã sofre alterações, conforme pode verificar-se na Figura 40, permitindo que o operador possa efetuar o registo no sistema sobre o tipo de paragem em questão.


Figura 40 – Layout Ecrã das Paragens

Após a seleção do tipo de paragem, passa para outro ecrã da aplicação, como ilustrado na Figura 41, aguardando pelo técnico, caso seja necessário. Caso algum técnico seja incumbido de alguma tarefa, terá de se identificar perante a máquina, através da sua Etiqueta NFC.


Figura 41 – Layout Ecrã Espera Técnico

Ao identificar-se, a aplicação passa para um ecrã, ao qual só os técnicos terão acesso, mas idêntico ao já demonstrado na Figura 40. Após a resolução do problema por parte do técnico, este terá, antes de o colaborador poder prosseguir com a sua atividade, identificar o motivo da sua presença, podendo para além de atribuir dados à paragem, fornecer mais pormenores através das Paragens de Segundo Nível. Para finalizar a sua tarefa, este poderá também acrescentar comentários à paragem, de maneira a que o responsável pela análise dos dados possa ter uma intervenção mais eficaz na resolução do problema a posteriori. Este ecrã está representado na Figura 42.

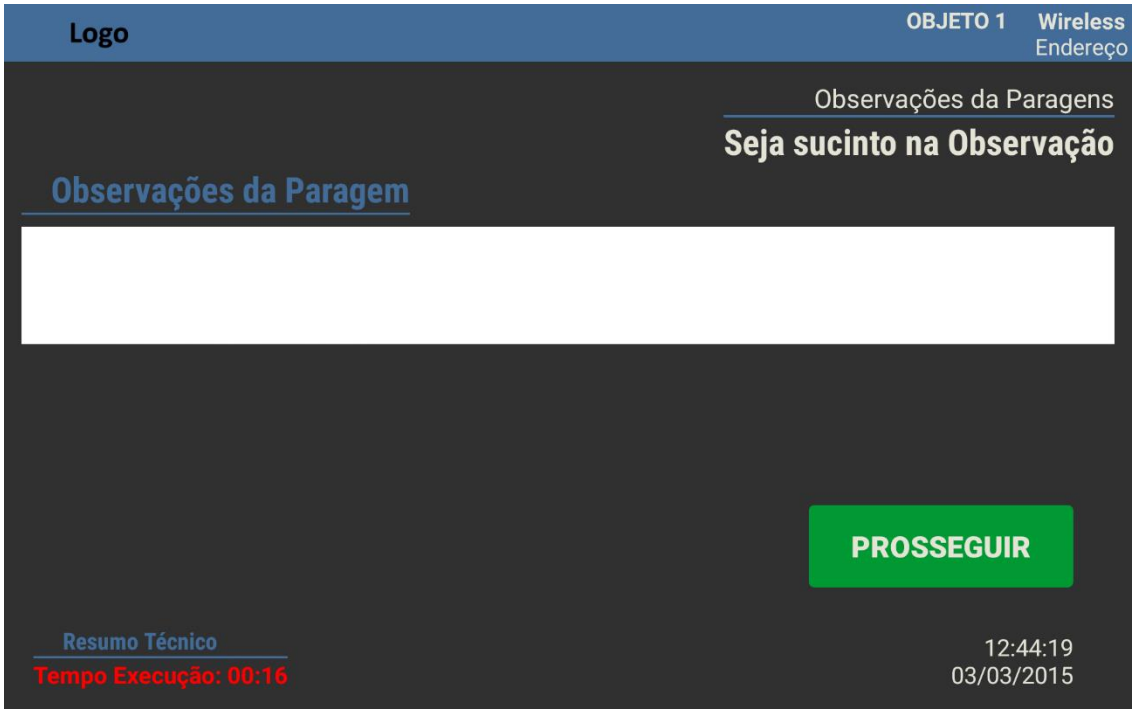



Figura 42 – Layout Ecrã Comentários Técnico

Caso não seja necessário o técnico ou este não compareça no local, o utilizador pode voltar ao ecrã da Figura 39, para prosseguir com as suas atividades. No termino das atividades, o colaborador é obrigado a entrar em estado fechado, procedendo então a amostragem dos resultados finais do seu trabalho, assim como demonstra a Figura 43, passando para outro ecrã, onde vão ser apresentados os resultados do mesmo, em termos das medidas do OEE, e o resultado final de OEE.

Após visualização dos seus resultados, regressa ao ecrã principal, e é nesta transição que todos os dados serão enviados para o servidor principal. Após o envio, e voltando ao ecrã principal, o ciclo de toda a atividade recomeça, dando início a um novo período de trabalho.

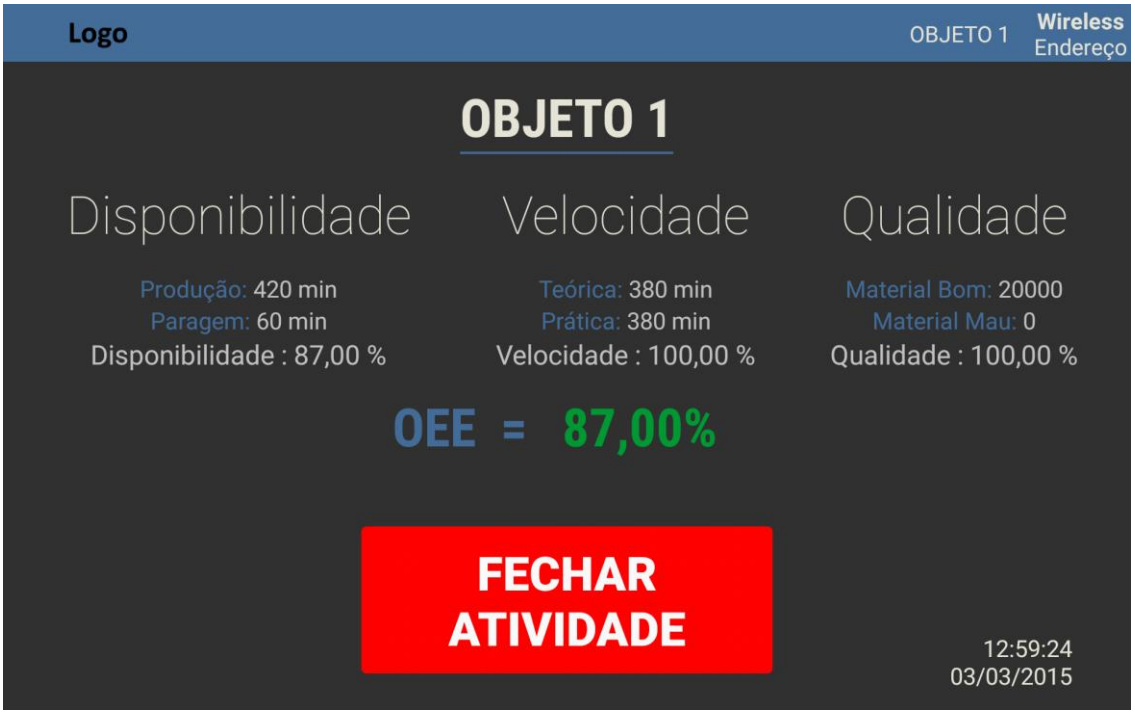



Figura 43 – Layout Ecrã Término Atividade

#### *4.3.1.2 Aplicação Móvel OEE (Administrativa)*

Na parte Administrativa, o inicio é idêntico à parte Operativa, visto fazerem parte da mesma aplicação, a única diferença está nas permissões, onde apenas alguns utilizadores têm acesso.

Após as verificações já conhecidas, atualizações e permissões, a aplicação prossegue para o ecrã principal da parte administrativa, como demonstrado na Figura 44.

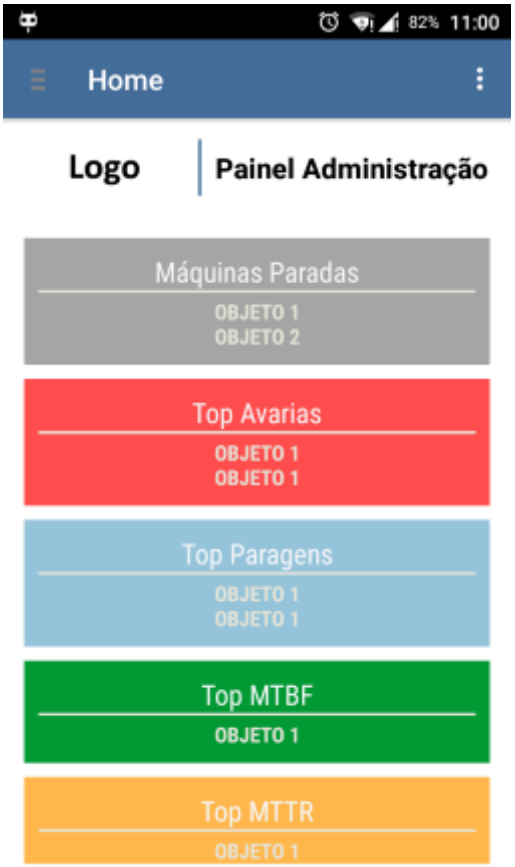



Figura 44 – Layout Ecrã *DashBoard*

A partir deste momento, o utilizador poderá navegar através de um menu existente na parte esquerda do ecrã, Figura 45, selecionando a opção que pretender. Todos os submenus, mostram os resultados acerca do objeto pretendido, apresentando um gráfico para cada pesquisa efetuada. Os ecrãs apresentam um aspeto semelhante ao da Figura 46.

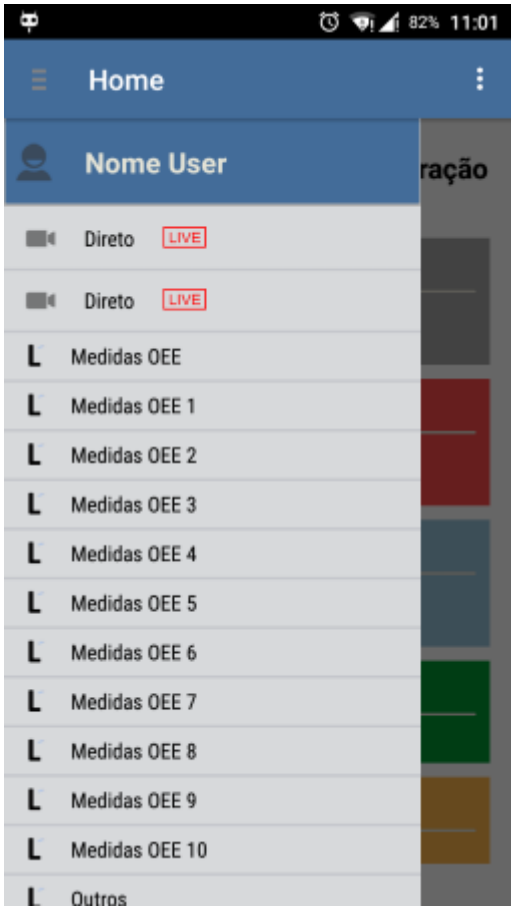



Figura 45 – Layout Ecrã com Menu

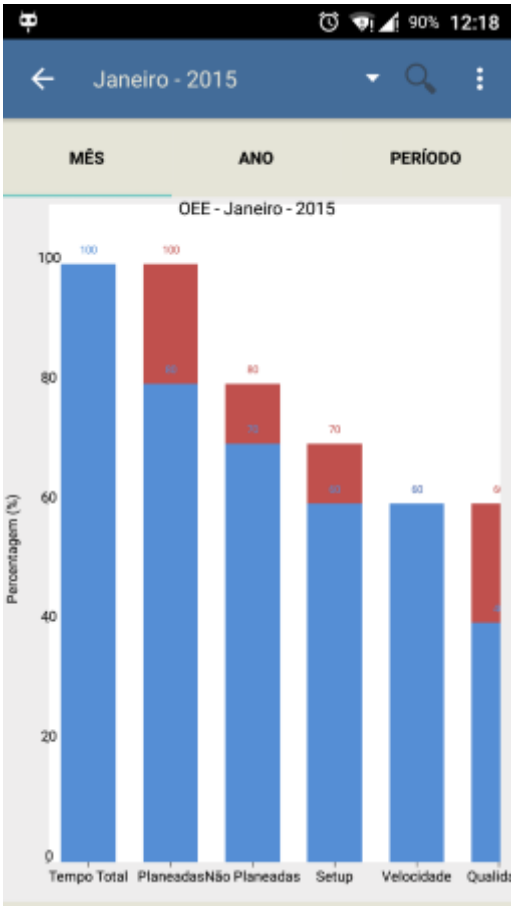



Figura 46 – Layout Ecrã demonstração dos dados

Ainda na parte administrativa, é o único local onde é possível configurar as etiquetas *NFC*, permitindo a quem gere o sistema poder atribuir etiquetas, assim como verificar se as mesmas estão perfeitamente legíveis e com os dados corretos, como demonstra a Figura 47.

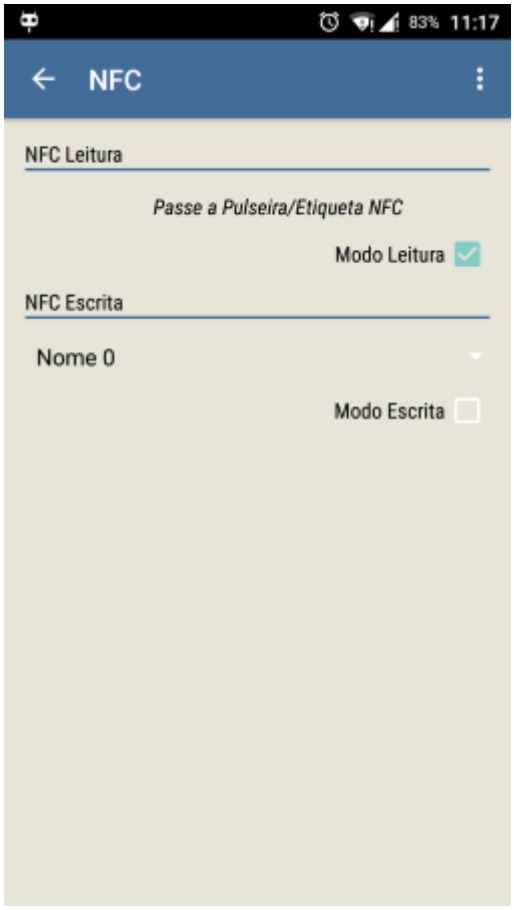



Figura 47 – Layout Ecrã Leitura e Escrita *NFC*

Para além de visualizar os dados, também é possível gerir toda a base de dados caso seja um dispositivo que opere sempre na parte operativa, isto permite verificar sempre se há dados em falta ou quais os dados que não foram devidamente enviados.

### 4.3.2 Layout da Plataforma Web

Na plataforma Web, a aplicação é muito idêntica à parte administrativa no dispositivo móvel, com a diferença que em vez de ser visualizada num dispositivo móvel, pode ser visualizada em qualquer browser web. A plataforma web contempla um menu com as opções disponíveis para o utilizador, assim como uma *dashboard* em tempo real de todos os dispositivos ligados, medições acerca do OEE, configuração de Paragens, paragens segundo nível, escalonamento, etc.

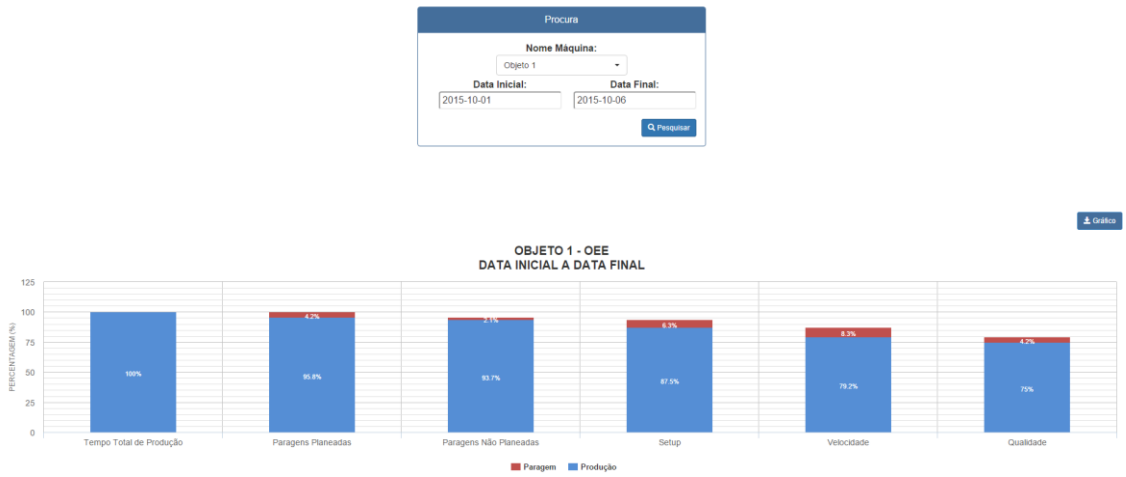



Figura 48 – Layout Página Consulta Dados

Nesta plataforma, podemos distinguir dois tipos de páginas, aquelas que são apenas para consulta, Figura 48, em que o layout é sempre idêntico, contendo um gráfico, onde são apresentados os resultados atingidos por cada objeto selecionado, num período específico. Para além de visualizar o gráfico na plataforma, há também a possibilidade de exportar o gráfico selecionado para um formato de imagem pré-definido, o JPEG.

Por outro lado, as páginas de manuseamento dos dados, onde se encontram as tabelas que contêm os registos existentes no servidor, divididas por cada objeto, podendo editá-los, removê-los ou adicionar mais dados para esses mesmos objetos, como demonstra a Figura 49.

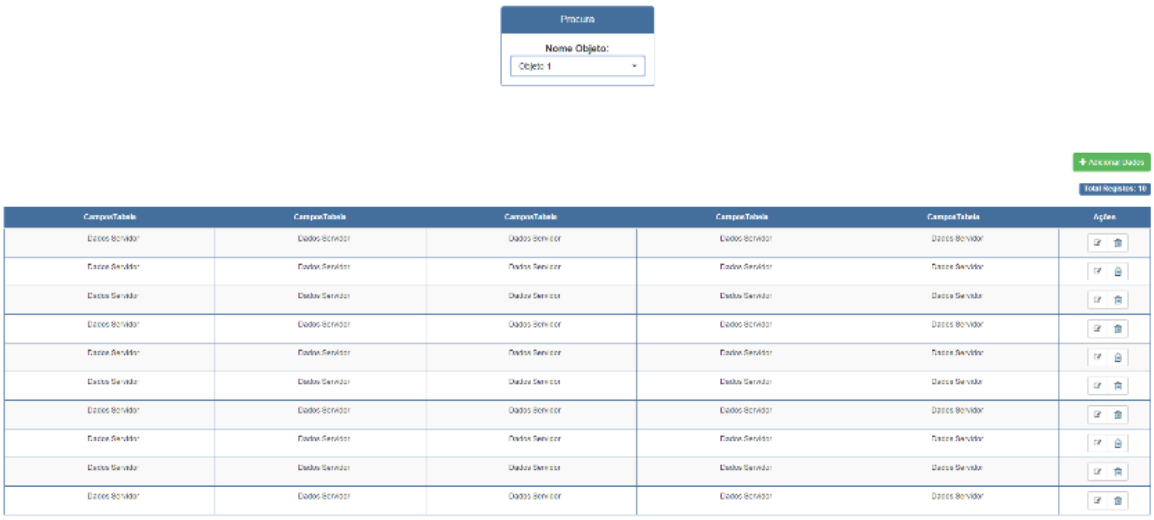



Figura 49 – Layout Página Manuseamento dos Dados

# 5 Conclusão

O desenvolvimento deste projeto, que originou esta dissertação, foi precedido de uma análise de todos os problemas que poderiam surgir, assim como, a melhor forma para obter os melhores resultados. É de salientar que existiu um especial cuidado, para que toda a aplicação móvel desenvolvida pudesse ser utilizada por qualquer pessoa, incluindo aqueles que nunca tiveram contato com tecnologias. Após a fase de análise, deu-se início ao desenvolvimento de quatro módulos que iriam contemplar todos os processos envolvidos.

O primeiro módulo, consiste no desenvolvimento da aplicação móvel, na sua vertente operativa. Esta foi desenvolvida numa fase inicial do projeto, sem ter em conta ainda a integração da parte administrativa da mesma. Na fase inicial do desenvolvimento, foram tidos em consideração sempre, os principais utilizadores da vertente operativa, aqueles que poderiam sentir dificuldades na utilização da aplicação móvel. Para que não existisse qualquer tipo de dificuldade na utilização, a simplicidade dos processos e as cores que a aplicação apresenta foram especificamente pensados para que fosse possível o uso da aplicação de uma forma simples e intuitiva.

O segundo módulo, a parte administrativa da aplicação, foi desenvolvida numa segunda fase, com o objetivo de integrar tudo numa só aplicação. Foi desenvolvida com o mesmo tipo de preocupações, no entanto, já exige outro tipo de conhecimento por parte do utilizador, ou seja necessita de saber que resultados irá procurar para cada objeto, e de que forma interpretar os mesmos, para tal necessita de ter conhecimentos no manuseamento de dispositivos móveis.

Ainda numa segunda fase de desenvolvimento, foi desenvolvido o terceiro módulo do projeto, a aplicação de envio de SMS. A mesma já não contempla o mesmo pressuposto de simplicidade que se encontra nas outras duas vertentes, mas sim um pressuposto de funcionalidade deixando de parte questões de design.

Com o objetivo de preencher a lacuna existente, ter uma plataforma web acessível em qualquer computador, e não apenas num dispositivo móvel, a aplicação web foi desenvolvida para que os restantes utilizadores que não fazem um uso frequente do dispositivo móvel, possam de uma forma simples aceder a toda a informação, verificando todo o processo produtivo, através da consulta dos cálculos das medições efetuadas a cada máquina.

Este projeto foi desenvolvido em três fases, que se complementam e têm como principal objetivo a criação de uma aplicação mais robusta, completa e acessível a qualquer utilizador.

Esta aplicação pode ser introduzida em qualquer tipo de indústria onde ainda não disponham de uma forma automatizada para efetuar o cálculo do OEE.

## 5.1 Implementações Futuras

Como qualquer aplicação, seja móvel ou fixa, necessita sempre de manutenção e de melhorias, tanto a nível de desempenho, como a nível de interatividade com o utilizador. São estes os pilares fundamentais a ter em conta para implementações futuras.

Em conjunto com estas melhorias, há vários aspetos onde a mesma pode ser melhorada, um dos mais relevantes é a necessidade de deixar de contemplar a intervenção humana no desenrolar da aplicação, tirando partido da automação, o dispositivo passará a estar conectado ao objeto, recolhendo tanto as paragens como produções, deixando apenas para o utilizador presente no dispositivo a confirmação da paragem. Com a introdução desta melhoria, os resultados finais seriam ainda mais precisos, do que aqueles que a aplicação apresenta nesta primeira fase.

Ao nível de interatividade, deverá ter melhorias em implementações futuras, mas sempre utilizando o feedback dos seus utilizadores. Neste aspeto, a intervenção humana é fulcral para a evolução da aplicação.

No que diz respeito à plataforma web, uma das possíveis melhorias seria passá-la para um paradigma diferente, podendo dessa forma implementar mais segurança, ou seja, implementá-la em *.Net*, mais propriamente em *ASPX*, ou então em *PHP*. São linguagens que permitem uma maior segurança do que em simples páginas *HTML* mais *JavaScript*. Numa primeira fase, iriam manter-se todas as funcionalidades, mudando apenas a linguagem em que se apresentam os dados. Numa segunda fase, seria recolhido o feedback dos utilizadores da aplicação, tornando a plataforma uma ferramenta mais eficaz e completa.

# Referências


| | |
|---|---|
| [Android Developers, 2015] | Android Developers, http://developer.android.com/index.html [último acesso: Set 2015] |
| [NFC Forum, 2015] | NFC Forum http://nfc-forum.org/ [ultimo acesso: Set 2015] |
| [OEE , 2013] | OEE http://www.leanproduction.com/oee.html [ultimo acesso: Set 2015] |
| [Vorne , 2008] | The Fast Guide to OEE http://www.vorne.com/pdf/fast-guide-to-oee.pdf [ultimo acesso: Jan 2015] |
| [NDEF , 2008] | NFC Data Exchange Format (NDEF) http://www.eet-china.com/ARTICLES/2006AUG/PDF/NFCForum-TS-NDEF.pdf [ultimo acesso: Fev 2015] |
| [Developers , 2015] | Dashboard Android Developers https://developer.android.com/about/dashboards/index.html [ultimo acesso: Ago 2015] |
| [História em Detalhe , 2015] | História em Detalhe https://www.google.com/about/company/history/ [ultimo acesso: Jan 2015] |
| [VT 3000 System , S/ Data] | VT 3000 System http://versacall.com/ [ultimo acesso: Mai 2015] |
| [Implementing OEE , 2012] | Implementing OEE http://www.oee.com/implementing-oee.html [ultimo acesso: Jan 2015] |
| [Iqity Solutions , 2012] | OEE WhitePaper http://www.iqitysolutions.com/wp-content/uploads/2012/02/IQity_OEE_WhitePaper.pdf [ultimo acesso: Fev 2015] |
| [NFC Chip , 2015] | NFC Chip http://rapidnfc.com/which_nfc_chip [ultimo acesso: Jan 2015] |
| [MTBF and MTTR , 2015] | MTBF and MTTR http://world-class-manufacturing.com/KPI/mtbf.html [ultimo acesso: Fev 2015] |
| [Gestão Manutenção, S/Data] | Gestão Manutenção https://fenix.tecnico.ulisboa.pt/downloadFile/3779571412644/C [ultimo acesso: Fev 2015] |
| [BootStrap , 2015] | BootStrap Documents http://bootstrapdocs.com/v3.2.0/docs/ [ultimo acesso: Mar 2015] |
| [Joda Time , 2013] | Joda Time APi http://joda-time.sourceforge.net/apidocs/ [ultimo acesso: Mar 2015] |
| [AchartEngine , 2011] | AchartEngine API http://achartengine.org/content/javadoc/ [ultimo acesso: Mar 2015] |
| [KSOAP, 2015] | KSOAP Project http://simpligility.github.io/ksoap2-android/index.html [ultimo acesso: Mar 2015] |
| [Acra , 2013] | Acra http://www.acra.ch/ [ultimo acesso: Mar 2015] |
| [Acralyzer , 2014] | Acralyzer https://github.com/ACRA/acralyzer/wiki [ultimo acesso: Mar 2015] |
| [SQLIte , 2015] | SQLite Documentation https://www.sqlite.org/docs.html [ultimo acesso: Mar 2015] |
| [BootBox , 2015] | BootBox Documentation http://bootboxjs.com/documentation.html [ultimo acesso: Mar 2015] |
| [HighCharts , 2015] | HighCharts Documentation http://www.highcharts.com/docs [ultimo acesso: Mar 2015] |
| [Jquery , 2015] | Jquery API https://api.jquery.com/ [ultimo acesso: Mar 2015] |
| [ASP.NET Web API , 2015] | APS.NET Web API http://www.asp.net/web-api [ultimo acesso: Mar 2015] |

[wearehathway, 2015] Benefits of Native Mobilie APP Development http://wearehathway.com/blog/benefits-of-native-mobile-app-development/ [ultimo acesso: Ago 2015]

[Gartner, 2014] Gartner Says By 2018, More Than 50 Percent of Users Will Use a Tablet or Smartphone First for All Online Activities http://www.gartner.com/newsroom/id/2939217 [ultimo acesso: Fev 2015]

[Gartner, 2014] Gartner Says By 2018, More Than 50 Percent of Users Will Use a Tablet or Smartphone First for All Online Activities http://www.gartner.com/newsroom/id/2939217 [ultimo acesso: Fev 2015]

[TechRunch, 2014] Mobile App Usage Increases In 2014, As Mobile Web Surfing Declines http://techcrunch.com/2014/04/01/mobile-app-usage-increases-in-2014-as-mobile-web-surfing-declines/ [ultimo acesso: Fev 2015]

[Statista, 2014] Statistics and facts about Mobile App Usage http://www.statista.com/topics/1002/mobile-app-usage/ [ultimo acesso: Fev 2015]