Mestrado em Instrumentação Biomédica

Desenvolvimento de uma Aplicação Móvel para a Monitorização Remota dos Equipamentos de Esterilização – Estágio na PROHS, S.A.

Relatório de estágio apresentado para a obtenção do grau de
Mestre em Instrumentação Biomédica

Autor
Vitaliy Hladchuk

Orientadores
Professora Doutora Fernanda de Madureira Coutinho
Professora do Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia de Coimbra
Professor Doutor Inácio de Sousa Adelino da Fonseca
Professor do Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia de Coimbra

Supervisor
Engenheiro Milton de Deus Rodrigues
PROHS – Equipamentos Hospitalares e Serviços Associados

Coimbra, abril, 2019

À minha família

Agradecimentos

Quero expressar, aqui, a minha gratidão e reconhecimento:

- À minha família, por me permitirem estagiar longe de casa com apoio financeiro e moral;
- À minha companheira de vida, Patrícia Duarte, pelo acompanhamento ao longo de todo o processo com carinho, paciência e ajuda;
- Aos meus orientadores de estágio, a professora Fernanda Coutinho e o professor Inácio Fonseca por toda a ajuda que conseguiram oferecer. Em especial, à professora Fernanda Coutinho por tornar possível este estágio e pelas horas não dormidas para correção deste documento;
- A toda equipa da PROHS, pelo acolhimento e condições propícias de trabalho. Em especial ao Jorge Lima, que permitiu a existência deste estágio, e ao meu supervisor, Milton Rodrigues, pelo suporte, preocupação e orientação ao longo do estágio;
- Ao Carlos Silva e ao Gustavo Santos pelo tempo e conhecimento disponibilizado e pela contribuição no desenvolvimento do lado do esterilizador;
- À comunidade de desenvolvedores Android pelo todo o conteúdo informativo disponibilizado na Internet e licenças permissivas dos seus projetos e bibliotecas, que pouparam muitas horas de trabalho neste estágio;
- À Omron, em especial ao João Silva e Nuno Rodrigues, pela formação oferecida e ajuda na resolução de problemas relacionados com as suas consolas;
- Por fim, mas não menos importante, quero agradecer aos meus amigos por todo o apoio prestado durante este percurso. Em especial, ao Nelson Mendes, grande colega de estudos, por todos os momentos passados, enorme apoio durante todo o processo e ajuda na escrita deste documento.

Vitaliy Hladchuk

Resumo

Nos últimos anos, a área da saúde tem vindo a adotar, cada vez mais, as tecnologias digitais, entrando assim na era da informação. Exemplo disso, é a integração de um equipamento médico na rede de comunicação de uma instituição (*e.g.*, hospital, laboratório clínico), abrindo caminho à sua monitorização remota e automática. Esta abordagem, integrada no espírito da Indústria 4.0, permite não só reduzir o tempo despendido pelos profissionais de saúde ou outros recursos humanos da instituição, como também promove a eficácia, a qualidade e o tempo de resposta dos equipamentos e dos serviços.

Por conseguinte, são exigidas, cada vez mais, aos fabricantes de equipamento médico, que os seus produtos estejam dotados da capacidade de monitorização remota e em tempo real. Os fabricantes de equipamento de esterilização médico não são uma exceção.

O trabalho desenvolvido e apresentado neste documento, resultou de um protocolo de cooperação entre o ISEC e a empresa PROHS – fabricante de equipamento de esterilização médico, e foi realizado no âmbito do estágio curricular do Mestrado em Instrumentação Biomédica do ISEC. O principal objetivo foi o desenvolvimento de uma aplicação para dispositivos móveis Android, capaz de fazer a monitorização remota dos esterilizadores horizontais deste fabricante. A aplicação desenvolvida possui duas funcionalidades principais: a monitorização remota em tempo real da execução do ciclo de um esterilizador e o acesso ao seu histórico de ciclos.

O resultado final correspondeu a uma aplicação intuitiva e robusta, que soluciona os problemas apontados pelo fabricante e que vai ao encontro dos requisitos técnicos e funcionais identificados. Para além disso, a solução desenvolvida não requer instalação de infraestruturas adicionais, o que constitui uma mais valia para a empresa e distingue-a de outras soluções comerciais concorrentes.

Palavras-chave: Aplicação móvel Android, App, Desenvolvimento Android, Monitorização em tempo real, Esterilizadores horizontais, Modbus TCP/IP.

Abstract

Recently, the healthcare system has been increasingly adopting digital technologies, joining the age of information. An example of this is the integration of medical equipment within the institution's network (e.g., hospital, laboratory) allows its remote monitoring. This approach provides mean for time optimization of: healthcare professionals and institution's human resources, also enhancing the efficiency, quality and response time of equipments and services.

Thus, the tools for medical equipment's remote monitoring has increasingly been demanded to be provided by the manufacturers of said equipments, from which the manufacturers of sterilization equipment are not an exception.

The undertaken assignment that is described in this document, emerged from cooperative protocol between ISEC *and* PROHS *– manufacturer of medical sterilization equipment. And was performed in the span of Masters Degrees in Biomedical Instrumentation internship. The main target of this internship was the development of the mobile application for Android devices, which allows remote monitoring of* PROHS*' horizontal sterilizers. The developed application has two main features: the real time remote monitoring of the sterilizer's cycle execution and the access to its cycle history.*

The final outcome corresponds to an intuitive and robust mobile application, which solves the issues pointed by PROHS *as well as, conforming with technical and functional requirements identified. Above all, the developed system does not require installation of any additional infrastructure, being cost effective for the company, and makes it stand out from the already existing solutions on the market.*

Key-Words: *Mobile Android application, App, Android development, Real time monitoring, Horizontal sterilizer, Modbus TCP/IP.*

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADU	*Application Data Unit*
AOT	*Ahead-of-time*
API	*Application Programming Interface*
APK	*Android Package*
App	Aplicação Móvel
ART	*Andoid Runtime*
CPU	*Central Processing Unit*
CRC	*Cyclic Redundancy Check*
CSV	*Comma-Separated Values*
DSL	*Domain Specific Language*
FHD	*Full High Definition*
FTP	*File Transfer Protocol*
HAL	*Hardware Abstraction Layer*
HMI	*Human Machine Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
ISEC	Instituto Superior de Engenharia de Coimbra
JSM	José dos Santos Monteiro Lda.
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MBAP	*Modbus Application Protocol header*
MIB	Mestrado em Instrumentação Biomédica
NDK	*Native Development Kit*
NoSQL	*No Structured Query Language*
OS	*Operating System*
PDU	*Protocol Data Unit*
PLC	*Programmable Logic Controller*
SCE	Serviço Central de Esterilização
SDK	*Software Development Kit*
StU	*Sterlization Unit*
TCP	*Transmission Control Protocol*
UID	*Unique Identifier*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
VNC	*Virtual Network Computing*
XML	*Extensible Markup Language*

Capítulo 1 - Introdução

Este relatório descreve o estágio curricular realizado pelo aluno Vitaliy Hladchuk, no âmbito da Unidade Curricular de Projeto/Estágio do 2º ano do Mestrado em Instrumentação Biomédica (MIB), do Instituto Superior de Engenharia de Coimbra (ISEC). A empresa PROHS – Equipamento Hospitalar e Serviços Associados, S.A., daqui para a frente designada apenas por PROHS, foi a Instituição de Acolhimento deste estágio curricular. A sua área de atividade principal é o Projeto e Desenvolvimento de Centrais de Esterilização para uso hospitalar.

Este estágio teve a orientação científica da Professora Doutora Fernanda Coutinho e do Professor Doutor Inácio Fonseca do ISEC. Foi supervisionado pelo Engenheiro Milton Rodrigues da PROHS. Decorreu entre fevereiro e agosto de 2018, inclusive.

O principal objetivo deste estágio foi o desenvolvimento de uma aplicação móvel (App), para dispositivos Android, capaz de fazer a monitorização remota dos equipamentos de esterilização da PROHS, mais especificamente dos esterilizadores horizontais conferindo, ao mesmo tempo, um nível de segurança adequado para o profissional que manipula o equipamento.

1.1 Motivação e Enquadramento

A PROHS é uma Pequena e Média Empresa (PME) do setor metalomecânico que atua na área da saúde e que se dedica ao projeto, fabrico e instalação de centrais de esterilização (PROHS, 2018).

Com o alargamento dos mercados de atuação e modernização hospitalar, cada vez mais, nos concursos públicos, é exigido um acesso de monitorização remoto, ou seja, um mecanismo de monitorização à distância dos equipamentos instalados, nomeadamente dos esterilizadores, a partir de dispositivos móveis. Esta exigência surge da necessidade de otimização dos recursos humanos que operam este tipo de equipamento, com o objetivo de libertá-los para a execução de outras tarefas no serviço, visto não haver necessidade de uma constante presença do operador junto do equipamento durante a execução do ciclo de esterilização, para verificação do estado de execução do programa.

Os esterilizadores da PROHS estão equipados com consolas NB7 da Omron, constituindo estas a interface homem-máquina (HMI - *Human Machine Interface*). Por sua vez, estas consolas possuem uma interface que permite o seu acesso remotamente, através de um servidor web (*NB Web Interface*). Este servidor permite monitorizar e operar a consola remotamente a partir de um computador, acedendo num navegador de *Internet* à página do servidor *Web* ou através de uma aplicação, *Omron Remote Viewer* (OMRON, 2018d), para *smartphones*. No entanto, atualmente, esta funcionalidade encontra-se de-

sativada por questões de segurança, pois não possui mecanismos de controlo e restrição da sua utilização para além de um único *login*. Ou seja, qualquer indivíduo que possua credencias para aceder ao servidor *Web* pode interferir com o esterilizador, de forma consciente ou não, e colocar em perigo a integridade do profissional que possa estar a manusear o equipamento no momento. Por exemplo, supondo que o esterilizador está a ser controlado remotamente, através da consola, enquanto que um outro profissional está a desempenhar uma ação de limpeza rotineira ao interior da câmara do esterilizador. Se, inadvertidamente, o primeiro der ordem de fecho da porta do esterilizador, e pese embora este tenha mecanismos de segurança próprios, a verdade é que essa ação pode colocar em risco o bem-estar do executante da ação de limpeza, podendo até sofrer lesões graves.

Neste sentido foi proposto este estágio para o desenvolvimento de um mecanismo de monitorização capaz de conferir maior segurança e robustez, indo assim ao encontro dos problemas e requisitos existentes.

1.2 Objetivos

O principal objetivo da realização deste estágio foi contactar e compreender o mundo profissional, em contexto empresarial, bem como desenvolver um *software* para dispositivos móveis que tenha a capacidade de monitorizar, de forma remota e segura, os equipamentos de esterilização da PROHS (Figura 1.1).

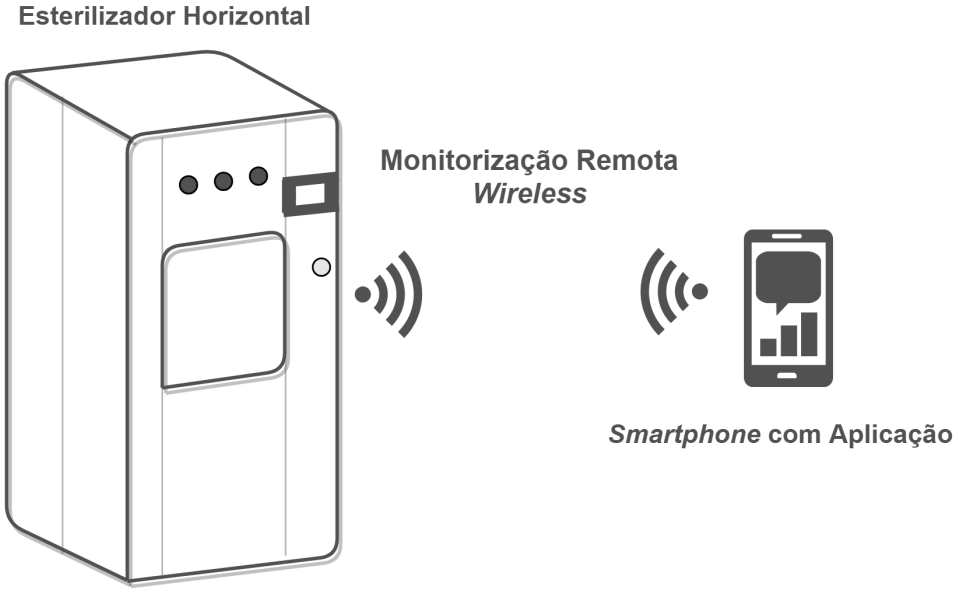


Figura 1.1: Esquema simplificado da solução desenvolvida no âmbito do estágio - monitorização remota de um esterilizador, com ações de segurança implementadas.

Para tal foram traçados os seguintes objetivos específicos:

- Desenvolver competências, dentro da empresa, na Plataforma Android;
- Estudar as possibilidades dos esterilizadores no que respeita ao acesso remoto;
- Estudar as soluções disponibilizadas pela concorrência para resolver o problema descrito;
- Definir as capacidades e o *layout* da aplicação a desenvolver;
- Desenvolver o *software* pretendido;
- Efetuar testes com a aplicação em ambiente semelhante ao real.

1.3 Instituição de Acolhimento

A PROHS foi fundada no ano 2000, num processo estratégico de modernização e renovação da imagem empresarial, sendo uma empresa vocacionada para a produção e comercialização de equipamentos de desinfeção e esterilização hospitalar e laboratorial, assim como outros equipamentos complementares de forma a abranger todas as necessidades de um Serviço Central de Esterilização (SCE). Dentro do Grupo da empresa PROHS está agregada a empresa José dos Santos Monteiro Lda. (JSM), que labora nesta área desde 1967, produzindo e comercializando equipamentos de desinfeção e esterilização e equipamento hospitalar em aço inoxidável com a marca JSM. No seu quadro, conta com a presença de 48 colaboradores, engenheiros e técnicos especializados nas seguintes áreas: projeto, instalação, formação, manutenção e pós-venda, o que permite responder às constantes exigências da desinfeção e esterilização. O organigrama geral da empresa está representado na Figura 1.2.

Em 2003 a PROHS adquire 55% da empresa JSM, passando assim a controlar as duas vertentes, a produção e a distribuição, com vista à internacionalização. O seu processo de internacionalização teve início em 2004 com a venda de dois esterilizadores para Angola, como consequência da sua participação como empresa expositora na Feira Internacional de Luanda, FILDA 2004, em Luanda. Em 2005 penetra no mercado Moçambicano ao apresentar uma proposta a um concurso internacional para o fornecimento de material hospitalar, que resultou na venda de 75 esterilizadores verticais. Contudo, é em 2006 que a PROHS investe fortemente na internacionalização ao marcar presença como expositor na maior feira internacional da especialidade – Medica, em Düsseldorf. Foi a primeira empresa portuguesa de dispositivos médicos a estar presente na referida feira, abrindo portas a novos mercados e iniciando relações comerciais com mercados profícuos como Marrocos, Síria, Argélia, Turquia, Egito e França. Atualmente, a PROHS é uma empresa de referência nacional e internacional no seu campo de atividade, colaborando em mais de 50 países e em diversas áreas geográficas e com parceiros locais nos 5 continentes.

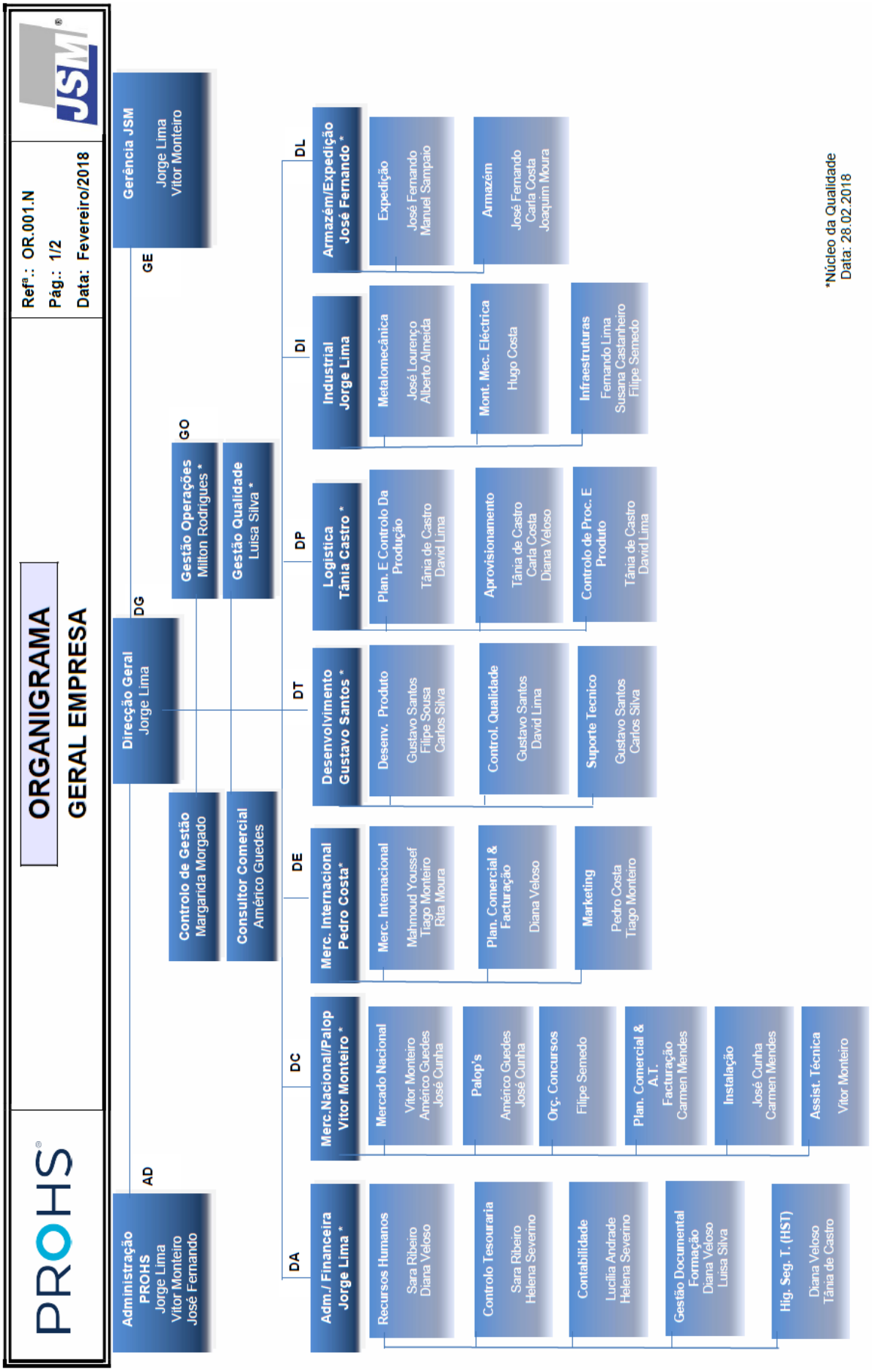


Figura 1.2: Organigrama geral da PROHS.

A PROHS é certificada pela norma NP EN ISO 9001:2008 - Sistema de Gestão da Organização de Empresa e ISO 13485:2012 - Sistema de Gestão da Qualidade de fabricantes de Dispositivos Médicos. Os produtos são fabricados de acordo com todas as normas de higiene, segurança no trabalho e controlo de qualidade por técnicos certificados e qualificados. Ao longo das diversas fases de produção todos os produtos são submetidos a testes e ensaios de acordo com as normas e diretivas aplicáveis, de forma a garantir a qualidade e fiabilidade dos produtos. Os dispositivos médicos produzidos pela PROHS possuem marcação CE concedida pelo organismo notificado SGS UK (CE 0120) no âmbito da Diretiva 93/42/EEC. No que respeita aos esterilizadores, tratando-se de equipamentos sob pressão, estão abrangidos pela Diretiva 2014/68/UE, tendo sido a marcação CE concedida pela SGS Portugal (CE 1155) – Equipamentos sob Pressão.

O SCE tem um papel fundamental no Controlo de Infeções em ambiente hospitalar. Por isso, o seu bom planeamento e funcionamento é essencial para a segurança dos pacientes. Como tal a PROHS dispõe de um departamento de projeto e desenvolvimento especializado em SCE, que acompanha cada projeto desde a fase de conceção até à sua entrega, completamente finalizada e testada, atendendo às especificações do cliente e às normas legais de cada país.

A sede da empresa (Figura 1.3) está localizada no norte do país, na zona industrial de Maia (Figura 1.4). Atualmente, as suas instalações contam com mais de $3.600\,m^2$, divididas em duas naves industriais adjacentes, resultado de uma recente expansão com a finalidade de procurar uma organização ergonómica do espaço de produção.

Figura 1.3: Sede da PROHS.

Figura 1.4: Localização da PROHS no Google Maps, retângulo vermelho.

1.4 Principais Tarefas do Estágio

O estágio decorreu durante seis meses e esteve principalmente focado no desenvolvimento de uma aplicação para a monitorização remota, denominada como "PROHS Remote View". O seu desenvolvimento ficou organizado em cinco componentes principais: histórico de ciclos, navegação pela aplicação, monitorização em tempo real, sistema de autenticação e *back-end* e instruções básicas de utilização dentro da aplicação. No entanto, para além do desenvolvimento da aplicação foram realizadas tarefas diversas, atribuídas ao estagiário ao longo do estágio com o intuito de se integrar na empresa, de modo a interagir com a sua equipa e seus parceiros externos.

De modo a realizar as diversas tarefas, ao longo do estágio, houve necessidade de estudar e adquirir novos conceitos. Entre eles destaco a esterilização com vapor de água, o serviço central de esterilização e os constituintes do esterilizador (principalmente HMI e controlador lógico programável (PLC - *Programmable Logic Controller*)). Houve, ainda, necessidade de adquirir novas competências e aprofundar as já existentes, nomeadamente no que diz respeito à programação em Android, à linguagem Java, ao Modbus, à programação das consolas NB da Omron, ao sistema de controlo de versões (git) e ao Firebase para a autenticação e *back-end* da aplicação. Posto isto, na Figura 1.5 está representado o cronograma das tarefas desempenhadas no estágio. Todas estas tarefas estão expostas

ao longo do presente documento.

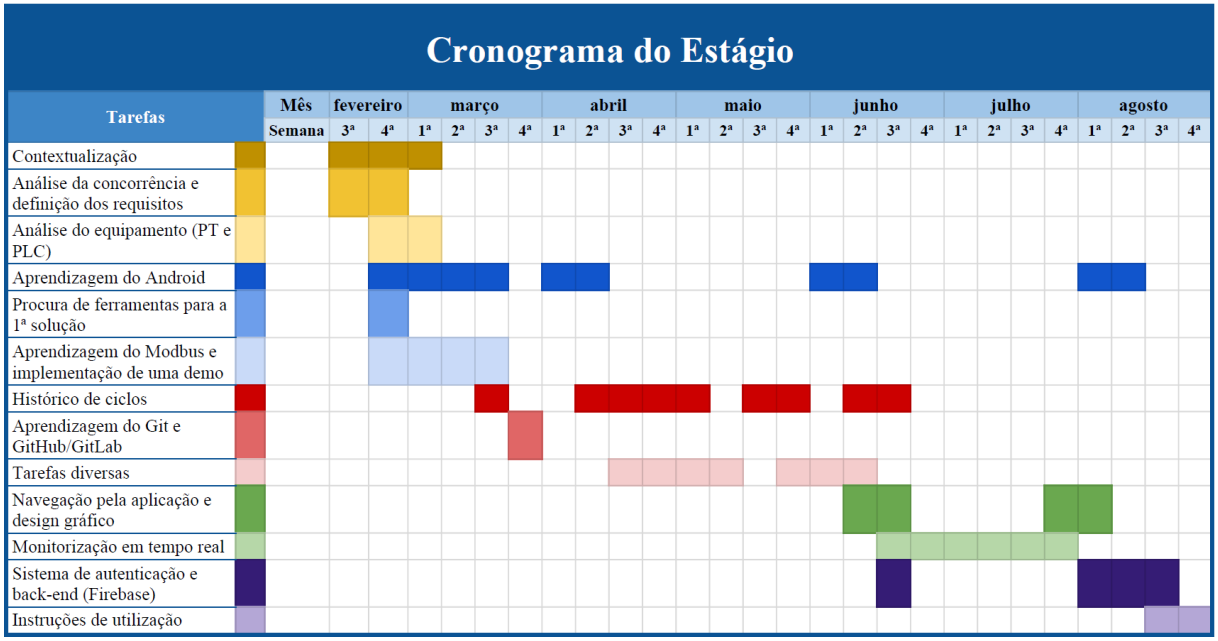


Figura 1.5: Cronograma do estágio.

1.5 Organização do Documento

Este documento está estruturado de forma a apresentar aos leitores o trabalho realizado durante o estágio, com foco no desenvolvimento da aplicação para monitorização remota dos esterilizadores. Assim, o documento está dividido em sete capítulos, sendo eles:

- **Capítulo 1 – Introdução** - é exposta uma breve apresentação do estágio, um enquadramento e motivação, bem como os objetivos traçados, a instituição de acolhimento e as principais tarefas realizadas no estágio. Por fim, é apresentada a estrutura e a organização do presente documento;
- **Capítulo 2 – Conceitos e Fundamentos Teóricos** - são abordados os princípios básicos de esterilização com vapor de água, o serviço central de esterilização, o equipamento e conectividade dos esterilizadores, os conceitos base da programação em Android e, por fim, as considerações finais do capítulo;
- **Capítulo 3 – Análise da Concorrência e Definição dos Requisitos** - são analisados os principais concorrentes em função das soluções de monitorização remota dos seus equipamentos e definição dos requisitos para a aplicação;
- **Capítulo 4 – Propostas de Solução** - são abordadas as três propostas de solução para monitorização remota;
- **Capítulo 5 – Desenvolvimento** - é abordado o desenvolvimento de todas as componentes da aplicação, começando pelo histórico de ciclos e seguindo com a nave-

gação pela aplicação e *design* gráfico, a monitorização em tempo real, o sistema de autenticação e *back-end* da aplicação, as instruções de utilização e, por fim, considerações finais do capítulo;

- **Capítulo 6 – Conclusão** - são expostas as conclusões do estágio e propostas para trabalho futuro.

Capítulo 2 - Conceitos e Fundamentos Teóricos

Por forma a concretizar o objetivo expresso na Secção 1.2 - desenvolvimento de uma aplicação móvel para a monitorização remota dos esterilizadores da PROHS - foi necessário aprofundar e também adquirir novas competências teóricas sobre diversos tópicos, nomeadamente: o processo de esterilização; o meio envolvente; o material com que se ia trabalhar e as suas capacidades; a plataforma, o software e o sistema operativo onde se ia desenvolver a aplicação.

Por conseguinte, este capítulo apresenta, de forma sumária, os principais conceitos teóricos e tecnológicos que foi necessário explorar e adquirir com vista ao desenvolvimento do trabalho. Esta descrição está dividida em quatro secções:

- Princípios Básicos de Esterilização com Vapor de Água (Secção 2.1), onde são abordados os fatores críticos, inerentes à esterilização com vapor de água, e o ciclo de esterilização;
- Serviço Central de Esterilização (Secção 2.2), - onde é feita uma apresentação do que é e qual a sua organização
- Equipamento e Conectividade dos Esterilizadores (Secção 2.3), onde são abordados os equipamentos utilizados no esterilizador (PLC e HMI) e as formas de estabelecer a ligação com o esterilizador;
- Programação Android (Secção 2.4), onde são apresentados os fundamentos do sistema operativo Android, como é feita a programação e em que consiste a aplicação Android.

Por fim as Considerações Finais (Secção 2.5), onde são abordadas tópicos extra, relacionados com os fundamentos teóricos, que não foram mencionadas ao longo do capítulo.

2.1 Princípios Básicos de Esterilização com Vapor de Água

Nesta secção são abordados os princípios básicos de esterilização pelo vapor de água, necessários para uma melhor compreensão dos alguns termos utilizados no resto do documento.

De uma forma geral, a esterilização entende-se como um conjunto de operações destinadas a eliminar ou matar a maior quantidade de formas de seres vivos, contidos num objeto/substância (Acosta-Gnass e Stempliuk, 2009). A esterilização pelo vapor de água é um processo através do qual se submetem os microrganismos à ação do calor mediante a injeção de vapor de água saturado seco e sob pressão. É o método mais utilizado nas centrais de esterilização hospitalares, sendo adequado quando os materiais

a esterilizar são resistentes às temperaturas elevadas e à humidade.

2.1.1 Fatores Críticos

Segundo Dion e Parker (2013), existem seis fatores particularmente críticos para o sucesso da esterilização a vapor de água, nomeadamente Tempo, Temperatura, Humidade, Contacto, Vácuo e Secagem. Nesta secção[1] são descritos cada um destes fatores, sumariamente.

Tempo

O tempo de esterilização é um fator crítico porque nem todos microrganismos morrem ao mesmo tempo. Este processo segue uma lei exponencial, sendo a curva típica ilustrada na Figura 2.1. Consequentemente, para a eliminação total é necessária uma esterilização por tempo "infinito", o que é impraticável, por isso o tempo para cada ciclo de esterilização é ajustado de forma a eliminar uma quantidade significativa dos microrganismos.

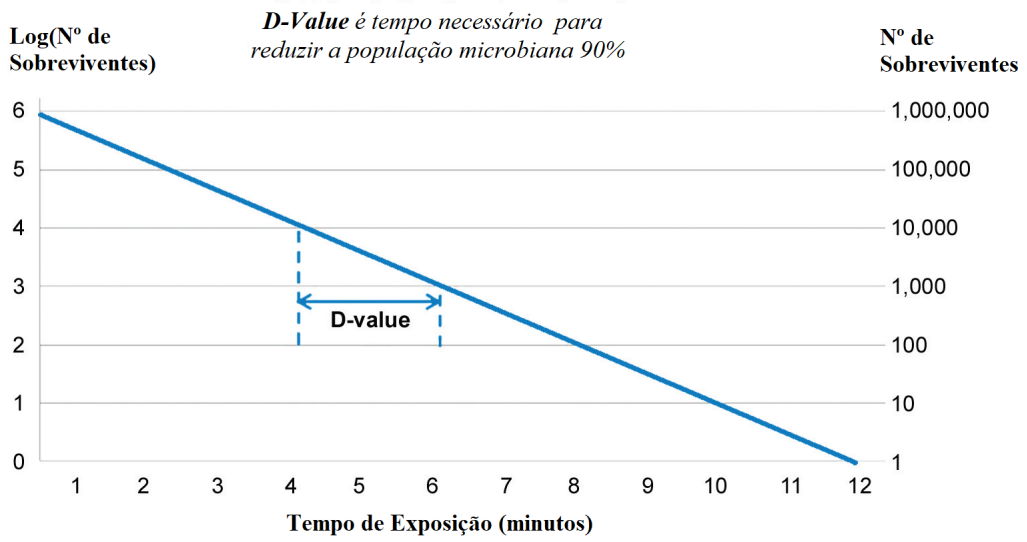


Figura 2.1: Curva típica de sobreviventes na escala logarítmica.

Temperatura

O segundo fator crítico é a temperatura na câmara do esterilizador. O aumento da temperatura diminui drasticamente o tempo de esterilização, o que é claramente visível na Figura 2.2. Nesta é possível observar-se os tempos aproximados para várias temperaturas de referência de modo a atingir os mesmos resultados de esterilização.

[1] O conteúdo da Secção 2.1.1, incluindo as figuras, é baseada nos trabalhos de Dion e Parker (2013) e Acosta-Gnass e Stempliuk (2009).

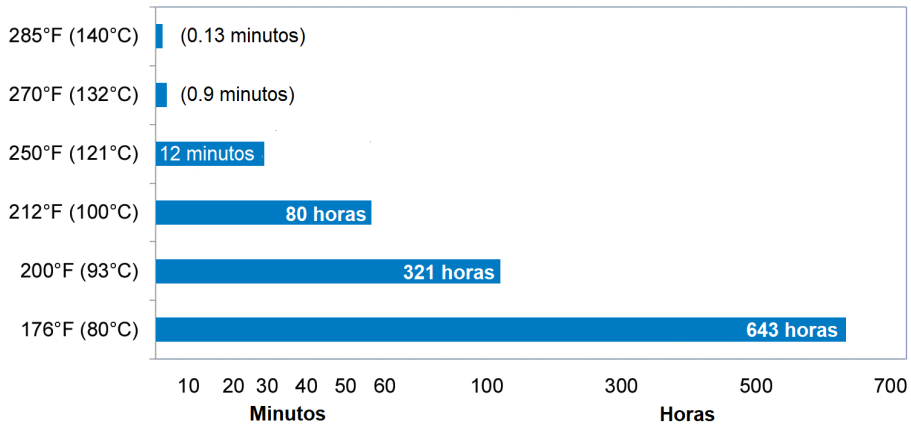


Figura 2.2: Tempo de esterilização *versus* temperatura.

Humidade

A esterilização por vapor de água está baseada no poder da humidade de desnaturar ou coagular proteínas, o que causa a morte dos microrganismos. Para se conseguir o melhor resultado é necessário utilizar o vapor saturado seco (*i.é.*, quando o vapor contém a concentração máxima de água). Para tal é fundamental entrar em consideração com a pressão. Com o aumento da temperatura, a quantidade máxima de água contida no vapor aumenta com o aumento da pressão, existindo assim uma relação direta entre temperatura e pressão para o vapor saturado seco (Figura 2.3).

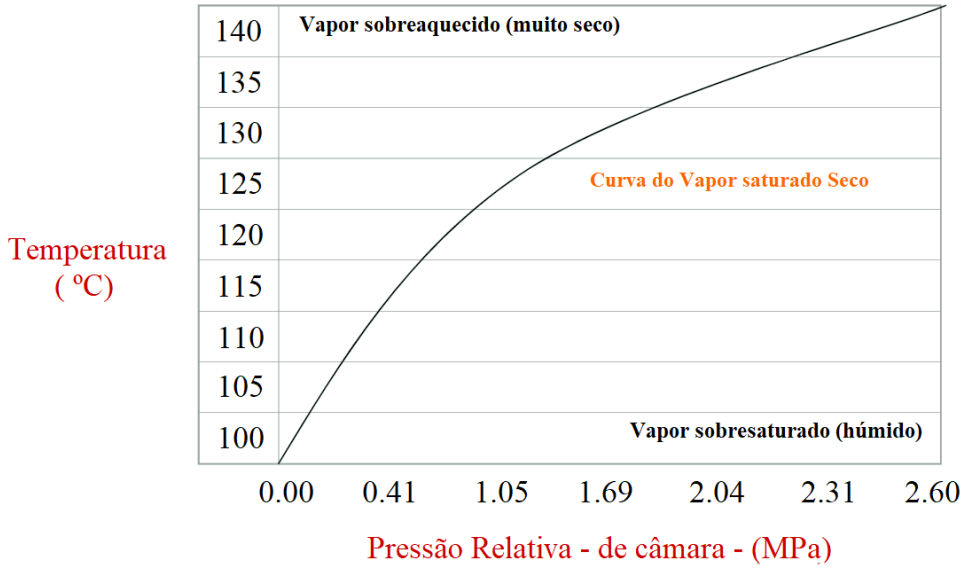


Figura 2.3: Relação pressão-temperatura para vapor saturado seco.

Caso a relação pressão-temperatura não seja conseguida, o vapor poderá ser considerado sobreaquecido - tendo pressão abaixo da necessária, ou vapor sobressaturado - tendo pressão acima da necessária. Em ambos os casos a esterilização será dificultada, não havendo humidade necessária para a desnaturação das proteínas ou existindo formação de condensados indesejáveis, respetivamente para o primeiro e o segundo caso.

Contacto

Outro fator importante é o contacto direto do vapor com a superfície dos objetos a esterilizar. Toda a esterilização por vapor é baseada no calor latente e condensação do vapor sobre superfícies com as quais entra em contacto. O calor latente é uma forma de energia térmica que não provoca alteração de temperatura, mas sim causa mudança de estado físico nos corpos. O vapor ao entrar em contacto com material mais frio, despende energia em forma de calor latente, o que causa a formação de condensados na superfície do material e o seu aquecimento. Este é o mecanismo que torna possível a desnaturação das proteínas dos microrganismos, causando a sua morte. Devido às altas temperaturas e pressão, o condensado rapidamente volta ao estado gasoso e o calor latente é reabsorvido pelo vapor. Esta troca de energia entre o vapor e os objetos é a base da esterilização.

Vácuo

O ar é considerado um dos melhores isolantes térmicos e o principal obstáculo na esterilização a vapor. A presença do ar na câmara do esterilizador impede a difusão do vapor e consequentemente o contacto direto com os objetos. Para além disso também causa redução da temperatura o que afeta a esterilização. Por estas razões existe necessidade de remoção do ar presente na câmara antes do início da esterilização, o que é conseguido através de uma série de pré-vácuos no início do ciclo de forma a remover a maior quantidade de ar. Na Figura 2.4 na página ao lado está presente um gráfico típico de um ciclo de esterilização com os pré-vácuos no início.

Secagem

Os objetos para a esterilização devem estar selados em recipientes adequados para a esterilização. Para remover os objetos do esterilizador de forma asséptica, é feita uma inspeção visual dos recipientes com o intuito de verificar a ausência de humidade (gotas visíveis). A presença de condensados nos recipientes pode causar recontaminação dos objetos durante a remoção. Este facto justifica a presença de secagem no final do ciclos de esterilização. A secagem é conseguida através de uma redução de pressão dentro da câmara para aproximadamente 6,9 kPa a 13,8 kPa. Posto isto, o ponto de ebulição da água para estas pressões é de aproximadamente 38,7^{o}C, o que causa evaporação, e consequentemente a secagem dos objetos.

2.1.2 Ciclo de Esterilização

O ciclo de esterilização, visível na Figura 2.4, pode ser dividido em 3 etapas (pré-tratamento, esterilização e pós-tratamento), e geralmente é constituído por 6 fases (pulsados, injeção do vapor, esterilização, descompressão, secagem e equalização da pressão).

Fases do Ciclo

1. **Pulsados** - consiste numa sequência de vácuo e injeção de vapor, numa série de pulsados, no sentido de promover a eliminação do ar na câmara;
2. **Injeção do vapor** - é injetado o vapor na câmara até que sejam atingidos os valores de pressão e temperatura de esterilização correspondentes;
3. **Esterilização** - são mantidos os parâmetros de pressão e temperatura durante o tempo estipulado para a morte de maior quantidade de microrganismos;
4. **Descompressão** - consiste na remoção do vapor da câmara até atingir os valores de pressão adequados à secagem;
5. **Secagem** - durante esta fase o material é arrefecido e seco até a remoção de toda a humidade;
6. **Equalização da pressão** - é injetado o ar estéril até à equalização da pressão.

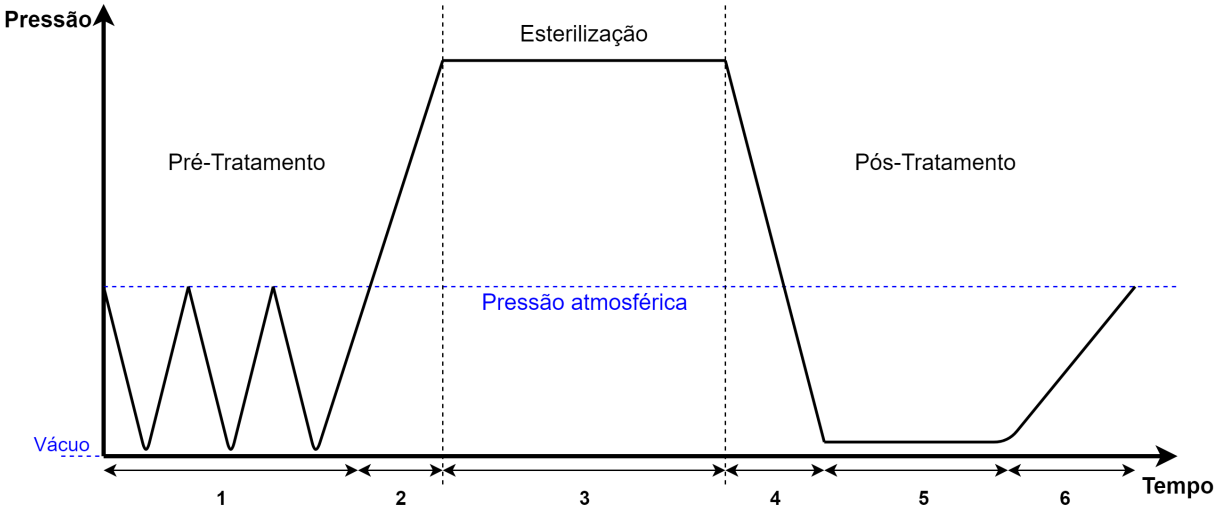


Figura 2.4: Gráfico típico do ciclo de esterilização.

Ciclos básicos

Os esterilizadores vêm programados com alguns ciclos ou programas básicos, no entanto é possível adicionar outros desde que passem a validação. Os programas básicos disponibilizados nos esterilizadores são:

- **Pré-Aquecimento** - utilizado apenas para pré-aquecer o esterilizador;
- **Têxteis** - destinado a têxteis (cargas porosas);
- **Sensíveis** - utilizado para material termo-sensível como borracha e vidro;
- **Instrumentos** - destinado para instrumentos embalados;
- **Contentores** - utilizado quando os instrumentos estão dentro de contentores metálicos;

- **Priões** - utilizado quando o material é contaminado com priões (Creutzfeldt-Jakob), seres mais resistentes à esterilização;
- **Rápido** - utilizado somente no bloco operatório para instrumentos não embalados;
- **Bowie & Dick** - teste diário para a verificação da eficiência da remoção do ar da câmara.

Também existe um programa para verificação do correto funcionamento do esterilizador, "Ensaio às Fugas de Ar", que consiste na produção de vácuo durante 15 minutos e deteção de flutuações na pressão.

2.2 Serviço Central de Esterilização

Os esterilizadores horizontais são o principal equipamento de esterilização no serviço central de esterilização. Por definição, uma central de esterilização (Figura 2.5) é um serviço que recebe, prepara, processa, controla e distribui têxteis, dispositivos biomédicos e instrumentos para todos os setores do hospital, para serem utilizados de forma segura nos utentes (Acosta-Gnass e Stempliuk, 2009). As imagens constantes nesta secção são propriedade da PROHS, tendo sido autorizada a sua utilização neste documento para uma melhor ilustração dos processos aqui descritos.

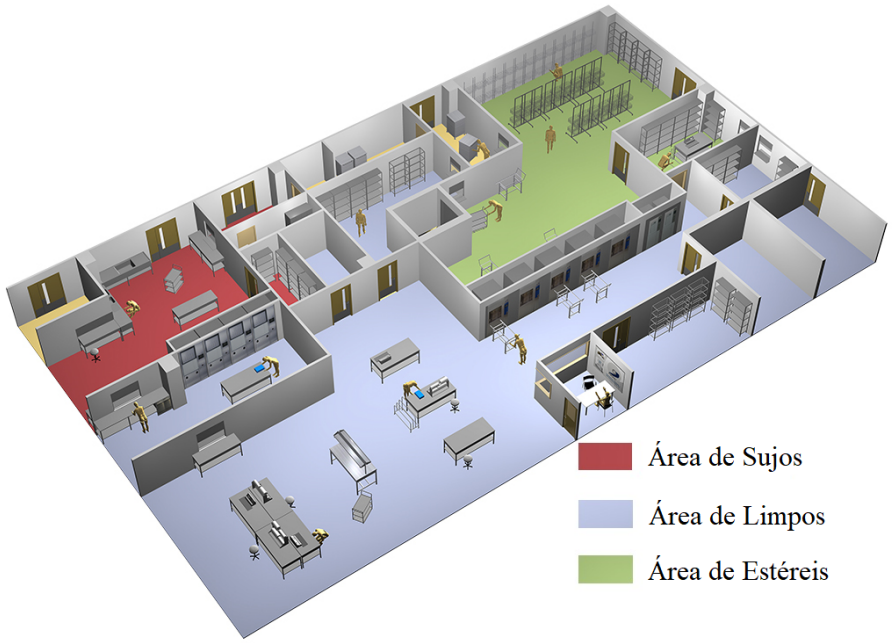


Figura 2.5: Mapa da central de esterilização.

A centralização dos serviços de esterilização segundo Acosta-Gnass e Stempliuk (2009) traz vantagens significativas, nomeadamente:

- **Eficiência** - quando devidamente organizado, este sistema permite uma fácil supervisão de todo o processo de tratamento do material biomédico;

- **Economia** - é evitada a duplicação de equipamento dispendioso (esterilizadores, máquinas de lavar, etc) e é aumentada a longevidade dos instrumentos, devido a eficiente manipulação (limpeza, preparação, esterilização) supervisionada por pessoal especializado;
- **Segurança** - com a centralização e pessoal especializado é reduzida a probabilidade de falhas nos processos (limpeza, preparação, esterilização) e exposição de materiais e instrumentos a métodos impróprios de esterilização.

Uma central de esterilização é dividida em 3 áreas principais - área de descontaminação (área de sujos), área de empacotamento de esterilizados (área de limpos) e área de armazenamento de esterilizados (área de estéreis). Cada uma delas pode ser dividida em sub-áreas e por tarefas que são desempenhadas nas mesmas. Na Figura 2.6 está representado o percurso do material dentro de uma central de esterilização.

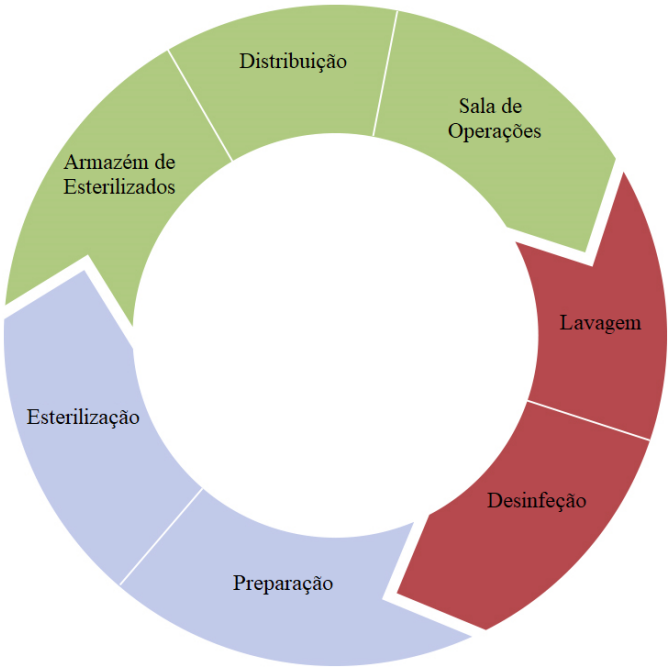


Figura 2.6: Diagrama de fluxo do material dentro do serviço central de esterilização.

Área de Sujos

Área de sujos, também designada de área para limpeza e descontaminação de materiais, é responsável por receber o material contaminado, proveniente do resto do hospital, para ser processado. Como tal, esta área deve ser facilmente acessível do corredor exterior à SCE. Nesta área (Figura 2.7(a)) é removida a maior carga microbiana. Sem uma limpeza adequada não é permitida a esterilização. A área de sujos deve ser fisicamente separada do resto das zonas da SCE, de modo a evitar recontaminação do material limpo por correntes de ar. A barreira física (Figura 2.7(b)) que separa esta área das outras é conseguida com equipamento adequado à passagem de material entre as áreas adjacentes, sem comprometer a esterilidade da área de limpos.

(a)

(b)

Figura 2.7: (a) Bancada de limpeza; (b) Máquina de lavar e desinfetar do lado esquerdo e guiché de transferência entre salas de sujos/limpos do lado direito.

Área de Limpos

Esta área é destinada a condicionamento, embalagem e preparação do material para esterilização. Na área de limpos somente podem ser admitidos materiais completamente limpos e secos, como tal, a primeira zona desta área é destinada a inspeção dos materiais provenientes da área de sujos, após limpeza e descontaminação. De seguida o material passa para a zona de preparação e embalagem (Figura 2.8(a)), onde é devidamente acondicionado e embalado para o processo de esterilização a que vai ser sujeito posteriormente. Após a preparação, o material é dirigido, com auxílio de carros de carga próprios, à zona de esterilização (Figura 2.8(b)), onde o material é sujeito a esterilização. Muitas vezes, os próprios esterilizadores servem de barreira física entre a área de limpos e a área de estéreis, o que é possível quando os esterilizadores horizontais são equipados com duas portas, uma de carga e outra de descarga, do lado oposto.

(a)

(b)

Figura 2.8: (a) Zona de preparação e embalagem; (b) Zona de esterilização com dois esterilizadores horizontais e um carro de carga.

Área de Estéreis

Por fim, a área de estéreis pode ser dividida em área de descarga (Figura 2.9(a)) do material proveniente dos esterilizadores e área para o armazenamento de esterilizados (Figura 2.9(b)). Esta área é separada fisicamente da restante central por uma barreira sanitária, para onde somente é admitido pessoal devidamente vestido.

(a) (b)

Figura 2.9: (a) Área de descarga; (b) Área de armazenamento.

2.3 Equipamento e Conectividade dos Esterilizadores

Uma das primeiras tarefas do estágio foi compreender com que equipamentos a aplicação a desenvolver precisaria de estabelecer ligação remota e de que forma tal poderia ser feito. Esta secção contém uma breve descrição do equipamento utilizado nos esterilizadores horizontais, relacionados com a comunicação remota.

Os esterilizadores horizontais da PROHS, na configuração básica mais recente, são controlados por um PLC da Omron, modelo CJ2M. Vem equipado com uma unidade de processamento central (CPU - *Central Processing Unit*) - CPU12, e também com uma consola interativa da Omron, um HMI da série NB com um ecrã de sete polegadas.

2.3.1 Controlador Lógico Programável

O autómato é um equipamento eletrónico, que pode ser considerado como um computador cuja arquitetura, sistema operativo, linguagem de programação, entradas/saídas e forma construtiva estão especialmente adaptados para aplicações de controlo em tempo real de processos sequenciais em ambiente industrial. Está concebido para funcionar em ambientes industriais agressivos (temperatura, vibrações, humidade, ruído elétrico, etc.), por isso, é também um equipamento muito robusto (Francisco, 2003).

O autómato, através das suas entradas, recebe informação do processo, vindas dos sensores, e, de acordo com o programa armazenado na sua memória, envia ordens, através

das saídas, para os atuadores que, por sua vez, atuam sobre o processo (Figura 2.10) (Francisco, 2003).

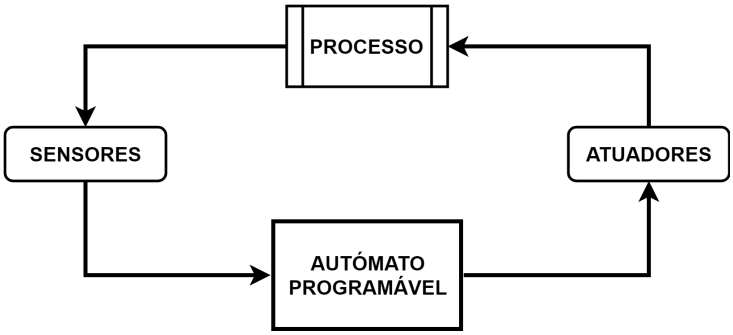


Figura 2.10: Funcionamento do autómato.

Os sinais que o autómato recebe dos sensores e que fornecem as informações ao programador denominam-se variáveis externas de entrada. Os sinais que o autómato fornece aos atuadores, que atuam sobre a parte operativa da instalação, denominam-se por variáveis externas de saída. Os sinais que o autómato utiliza como resultado de operações lógicas ou aritméticas realizadas pelo programa, denominam-se variáveis internas (Francisco, 2003). As variáveis externas, podem ser analógicas ou digitais.

2.3.1.1 PLC CJ2M-CPU12

No que diz respeito ao PLC utilizado pela PROHS, o CJ2M é um PLC modular e considerado de gama média (Figura 2.11(a)). Este autómato, por ser modular, permite acrescentar módulos ou cartas que são inseridas numa unidade de expansão (Figura 2.11(b)), sendo desta forma possível controlar os custos de instalação deste equipamento e ter uma fácil adaptação ao processo a controlar. O modelo da CPU é CPU12 com as características técnicas representadas na Tabela 2.1.

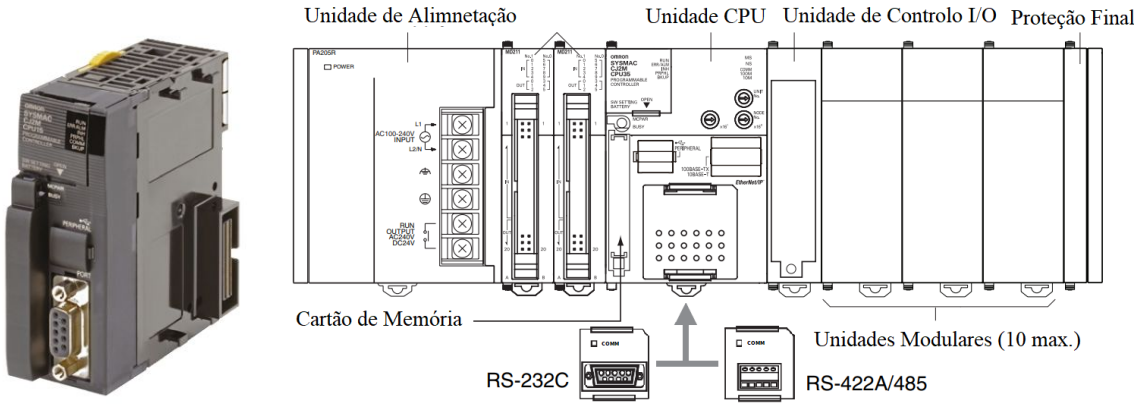


Figura 2.11: (a) PLC CJ2M da OMRON[2]; (b) Esquema da constituição do PLC (OMRON, 2017).

Tabela 2.1: Dados técnicos da CPU12 do PLC modular, CJ2M (OMRON, 2017).

Temperatura ambiente de operação	0 a 55ºC
Humidade ambiente de operação	10% a 90%
Consumo	5 V_{DC}, 500 mA
Vida da bateria	5 anos a 25ºC
Capacidade do programa	10K *steps*
Capacidade da memória de dados	64K *words*
Capacidade para montagem de unidades	40 unidades (10 por unidade de expansão)
Tipo de comunicações	*Universal Serial Bus* (USB) + RS-232C
Tempo de execução	0,04µs instruções básicas
	0,06µs instruções especiais
Modos de operação	*Run*, Monitorização e Programação
Cartão de memória	128 MB, 256 MB, ou 512 MB

Para suportar a configuração do esterilizador são necessárias 7 entradas digitais, 4 entradas analógicas e 16 saídas. As entradas analógicas são destinadas aos sensores de pressão e de temperatura (atualmente são 2 sensores de temperatura dentro da câmara e 2 sensores de pressão: um dentro da câmara e outro na camisa (Figura 2.12)). O esterilizador é um recipiente de corpo duplo, o exterior chamado de camisa e o interior de câmara.

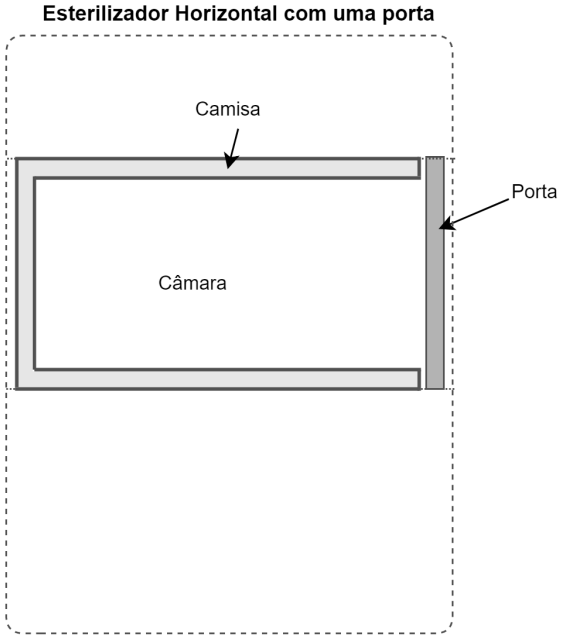


Figura 2.12: Esquema simplificado da localização da camisa num esterilizador horizontal.

Como tal, são utilizados 3 módulos: CJ1W-ID201, CJ1W-AD04U e CJ1W-OC211, estando as suas características técnicas indicadas nas tabelas 2.2, 2.3 e 2.4, respetivamente.

[2]Imagem retirada de: `https://uk.rs-online.com/web/p/plc-cpus/7462878/`. Consultado em novembro de 2018.

O módulo das entradas analógicas utilizado é um módulo universal, em que é possível configurar cada entrada separadamente para um tipo de entrada específico (sensores de temperatura, corrente ou tensão).

Tabela 2.2: Dados técnicos do módulo das entradas digitais, CJ1W-ID201 (OMRON, 2017).

Número de entradas	8 digitais
Tensão e corrente de entrada	12 a 24 VDC, 10 mA
Nº de *words* alocados	1 *word*

Tabela 2.3: Dados técnicos do módulo das entradas analógicas, CJ1W-AD04U (OMRON, 2017).

Número de entradas	4 analógicas
Tipos de entradas	Termopar (K, J, T, L, R, S, B)
	Termómetro (Pt100, JPt100, Pt1000)
	Corrente (4 a 20 mA, 0 a 20 mA)
	Tensão (1 a 5 V, 0 a 5 V, 0 a 10 V)
Velocidade de conversão	250 ms/4 entradas

Tabela 2.4: Dados técnicos do módulo das saídas digitais, CJ1W-OC211 (OMRON, 2017).

Número de saídas	16 digitais a relé
Tensão e corrente nominais	250VAC/24 VDC, 2 A
Nº de *words* alocados	1 *word*

2.3.2 Interface Homem-Máquina

Tipicamente, uma HMI consiste numa consola com ecrã tátil, que serve de interface gráfica entre o utilizador e um sistema de controlo industrial, neste caso um esterilizador. No nosso caso, é através de uma consola destas que o utilizador pode interagir com o equipamento de uma forma intuitiva e monitorizar o estado de execução do ciclo de esterilização.

Em relação à consola, a PROHS optou por uma HMI da Omron da série NB, uma versão *Wide Screen* de 7 polegadas, mais concretamente NB7W-TW01B (Figura 2.13) que possui uma entrada *Ethernet* incorporada. Na Tabela 2.5 estão sumariadas as principais características técnicas desta consola.

Figura 2.13: Consola HMI da Omron, NB7 - TW01B.

Tabela 2.5: Características técnicas da consola NB7 - TW01B (OMRON, 2018c).

Tipo de ecrã	7" LCD TFT
Resolução de ecrã	800 × 480
Número de cores	65536
Tipo de iluminação	LED
Vida útil dos LED	50000h (6 anos) em tempo de operação a 25ºC
Painel de toque	Membrana resistiva, duração 1000000 toques
Memória interna	128Mb
Interface de memória	USB
Tipo de comunicações	*Ethernet*, RS232C, RS485/422A
Proteção frontal	IP65
Potência de consumo	11W
Temperatura ambiente de trabalho	0-50ºC
Humidade relativa do ambiente	10-90% sem condensação

As consolas da série NB estão acompanhadas com um *software* gratuito *NB Designer*, sendo através deste que se configura a comunicação com o PLC e se cria todo o ambiente gráfico de interface para o utilizador.

2.3.3 Conectividade

Em relação à conectividade do esterilizador, a consola e o autómato comunicam via comunicação série RS232, visto a ausência de uma entrada *Ethernet* incorporada no autómato, como tal, não existe um acesso direto do exterior ao autómato. Assim a consola HMI NB7 é única opção para estabelecer a ligação com o esterilizador, utilizando a rede *Ethernet*. A NB7 dispõe de quatro formas distintas de acesso remoto à consola HMI:

- **Servidor Modbus** - a NB7 suporta o protocolo industrial Modbus TCP/IP (onde: TCP - *Transmission Control Protocol*; IP - *Internet Protocol*), que por sua vez permite estabelecer a ligação do exterior do esterilizador;
- Três interfaces de acesso remoto:
  - **Servidor *Web*** - disponível a partir da versão 1.30 do *NB Designer*;

– **Servidor *Virtual Network Computing*** (**VNC**) e **servidor *File Transfer Protocol*** (**FTP**) - ambos adicionados em Abril de 2018 numa das últimas atualizações ao *software* das consolas NB e a *NB Designer* (versão 1.46).

2.3.3.1 Modbus TCP/IP

O Modbus é um protocolo de comunicação de dados em sistemas de automação industrial, desenvolvido em 1979 pela empresa Modicon e que atualmente pertence ao grupo Schneider Electric. Modbus é um dos primeiros protocolos desenvolvidos, considerado como padrão, e é o mais utilizado em redes industriais para a comunicação com PLC (Modbus.org, 2012). A sua arquitetura está baseada numa comunicação cliente/servidor (mestre/escravo), onde o servidor fica à escuta, à espera de solicitações do cliente. As solicitações podem ser tanto de leitura como de escrita de dados. A comunicação é levada a cabo através de uma troca de tramas Modbus (Figura 2.14): uma parte da trama específica do protocolo Modbus designada de *Protocol Data Unit* (PDU), independente do meio de comunicação e uma parte preenchida que depende do meio de comunicação, servindo de encapsulamento da PDU, designada de *Application Data Unit* (ADU). O endereçamento adicional é composto por um *byte* para identificação do dispositivo destino.

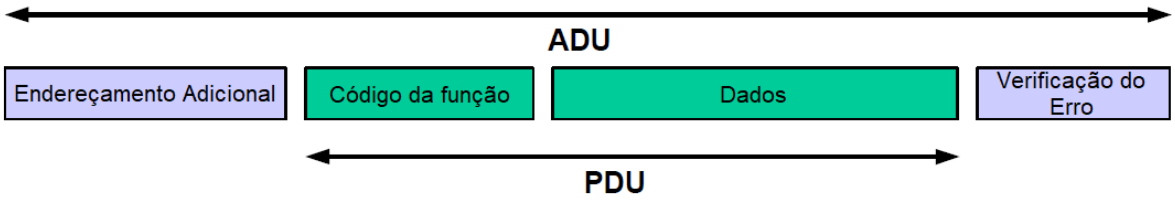


Figura 2.14: Trama geral do protocolo Modbus (Modbus.org, 2012).

Para distinção da ação pretendida pelo cliente, o primeiro *byte* da PDU é preenchido com o código da função (podendo tomar valores entre 1 e 127). O código da função, quando é feita uma solicitação pelo cliente, é geralmente acompanhado pelo campo de dados (pode não existir), que contém informação adicional como endereços de registos, quantidade de dados para serem manipulados e contador de *bytes* presentes no campo. No caso de resposta do servidor, o campo de dados pode conter os dados solicitados ou o código de erro. O campo de verificação do erro é composto por 2 *bytes* gerados pelo algoritmo de *Cyclic Redundancy Check* (CRC[3]) para verificação cíclica de redundância. O tamanho da PDU está limitado a 253 *bytes* devido à primeira implementação do Modbus em comunicação série, onde o tamanho máximo da ADU é 256 *bytes*. Para campos de endereçamento e de dados, o Modbus utiliza o método 'big-Endian', o qual significa que valores numéricos maiores que um *byte* são transferidos sequencialmente, onde o *byte* mais significativo é enviado primeiro (Modbus.org, 2012).

[3]Para mais informação acerca deste algoritmo, pode ser lido artigo da IEEE escrito por Sheng-Ju (2015)

O Modbus TCP/IP, que por sua vez é uma extensão do Modbus, utiliza a rede *Ethernet* como meio de comunicação para transferência de pacotes via TCP/IP. É utilizado por omissão o porto TCP 502, reservado para comunicações Modbus. Neste caso a trama (Figura 2.15) é ligeiramente diferente. O endereçamento adicional é substituído pelo cabeçalho *Modbus Application Protocol header* (MBAP), de 7 *bytes*, e a verificação do erro é retirada da ADU, pois o protocolo TCP/IP possui uma verificação por omissão. O cabeçalho MBAP contém 2 *bytes* para identificador de transação, 2 *bytes* para identificador do protocolo, 2 *bytes* para o tamanho da mensagem e 1 *byte* para identificar a unidade de destino (Modbus.org, 2006).

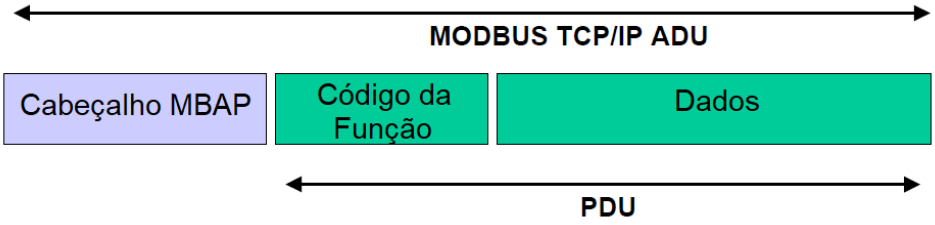


Figura 2.15: Trama do protocolo Modbus TCP/IP (Modbus.org, 2006).

A consola NB configurada como servidor em Modbus TCP/IP, suporta um número limitado de códigos de função, relacionados com tipo de endereços 0X e 4X, correspondentes a *Coils* (*bits*) e *Holding Registers* (*words* de 16 *bits*), descritos na Tabela 2.6. Os endereços dos registos 0X e dos registos 4X podem tomar valores entre 1 e 9000, que correspondem aos registos locais LB (0-8999) e LW (0-8999), nas consolas NB (OMRON, 2018b).

Tabela 2.6: Tabela de códigos de função disponíveis para consolas NB em modo *slave* do Modbus TCP/IP (OMRON, 2018b).

Código da Função	**Nome**	**Função**	**Tipo de Endereço**
01	*Read coil status*	Leitura de N *bits*	0X
03	*Read multiple registers*	Leitura de N *words*	4X
05	*Write single coil*	Escrita de um *bit*	0X
06	*Write single register*	Escrita de uma *word*	4X
15	*Write multiple coils*	Escrita de N *bits*	0X
16	*Write multiple registers*	Escrita de N *words*	4X

Na comunicação Modbus TCP/IP existem algumas regras a considerar ao criar o cliente Modbus (Modbus.org, 2006), como:

- É recomendada a implementação da gestão de ligações TCP automática;
- É recomendado manter a mesma ligação TCP aberta para todas as transações;

- É recomendado para cada cliente Modbus criar o mínimo de ligações possíveis com o servidor (um por aplicação).

2.3.3.2 Servidor *Web*

O servidor *Web*, também designado pela Omron como *NB Web Interface,* é uma funcionalidade que as consolas NB possuem e, quando ativada, permite monitorizar a tela representada pelo seu ecrã utilizando um navegador *Web* (Internet Explorer, Google Chrome, Opera, etc.) a partir de um dispositivo que esteja ligado à consola via *Ethernet*.

O *NB Web Interface* possui as seguintes funcionalidades:

- **Monitorização Remota** - consiste numa receção periódica de imagens (*print screen*) a partir do ecrã em tempo real;
- **Operação Remota ou Controlo Remoto** - tal como na monitorização, consiste na receção das imagens mas com possibilidade de interação. As imagens estão sob uma área que deteta os *clicks* efetuados pelo utilizador, de forma a registar o local e a duração do *click*, que informa a consola em tempo real sobre esta interação;
- **Acesso à memória USB** - é possível introduzir uma *pen* USB na consola para gravar as capturas de ecrã e/ou tabelas com dados. Esta funcionalidade permite aceder à informação guardada na *pen* sem necessidade de removê-la da consola e ligar a um computador.

O servidor *Web* utiliza o protocolo *Hypertext Transfer Protocol* (HTTP) para estabelecer a ligação e a comunicação. Para estabelecer a ligação com a consola é necessário inicializar o navegador *Web* e introduzir na barra de navegação o endereço IP da consola, que por sua vez irá abrir uma janela de autenticação, para introduzir *username* e *password*. Após autenticação, o utilizador acederá ao menu do *NB Web Interface* (Figura 2.16), onde terá quatro opções: as três funcionalidades mencionadas e uma janela de configurações do servidor.

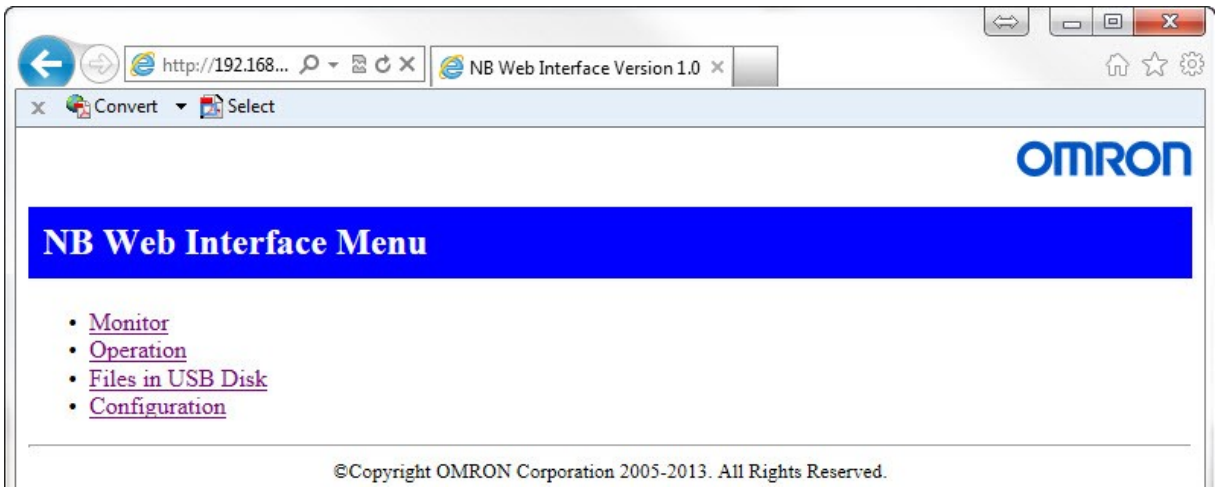


Figura 2.16: Menu da NB *Web Interface* (OMRON, 2014).

Na janela de configurações (Figura 2.17) para Monitorização Remota é possível ajustar o período de atualização da imagem (entre 2 a 99 segundos), o formato da imagem (BMP, JPEG ou PNG) e o valor de compressão da imagem, caso seja selecionado o formato PNG. Para além disto é possível desativar a autenticação e representar a data e a hora da captura do ecrã (imagem). Para a Operação Remota, para além do pedido de atualização, é possível especificar o atraso depois do *click* (0 a 99 segundos) para leitura da imagem da consola depois da ação. Também existe a possibilidade de desativar o ecrã tátil da consola enquanto esta esteja a ser operada a partir da interface *Web*. Adicionalmente, existem opções para configurar o servidor, como alterar o título da página do servidor, a janela inicial após autenticação e a porta TCP para acesso ao servidor. Também é possível ativar ou desativar a utilização de *JavaScript* para atualização automática das janelas de Monitorização e Operação.

Para alterar as credencias de acesso ao servidor é necessário aceder à NB-Manager, à secção "*Web Interface Operation*" (Figura 2.18), onde é possível modificar o *username* e *password*, assim como voltar a usar os parâmetros por omissão (*username*: *default* e *password*: *default*).

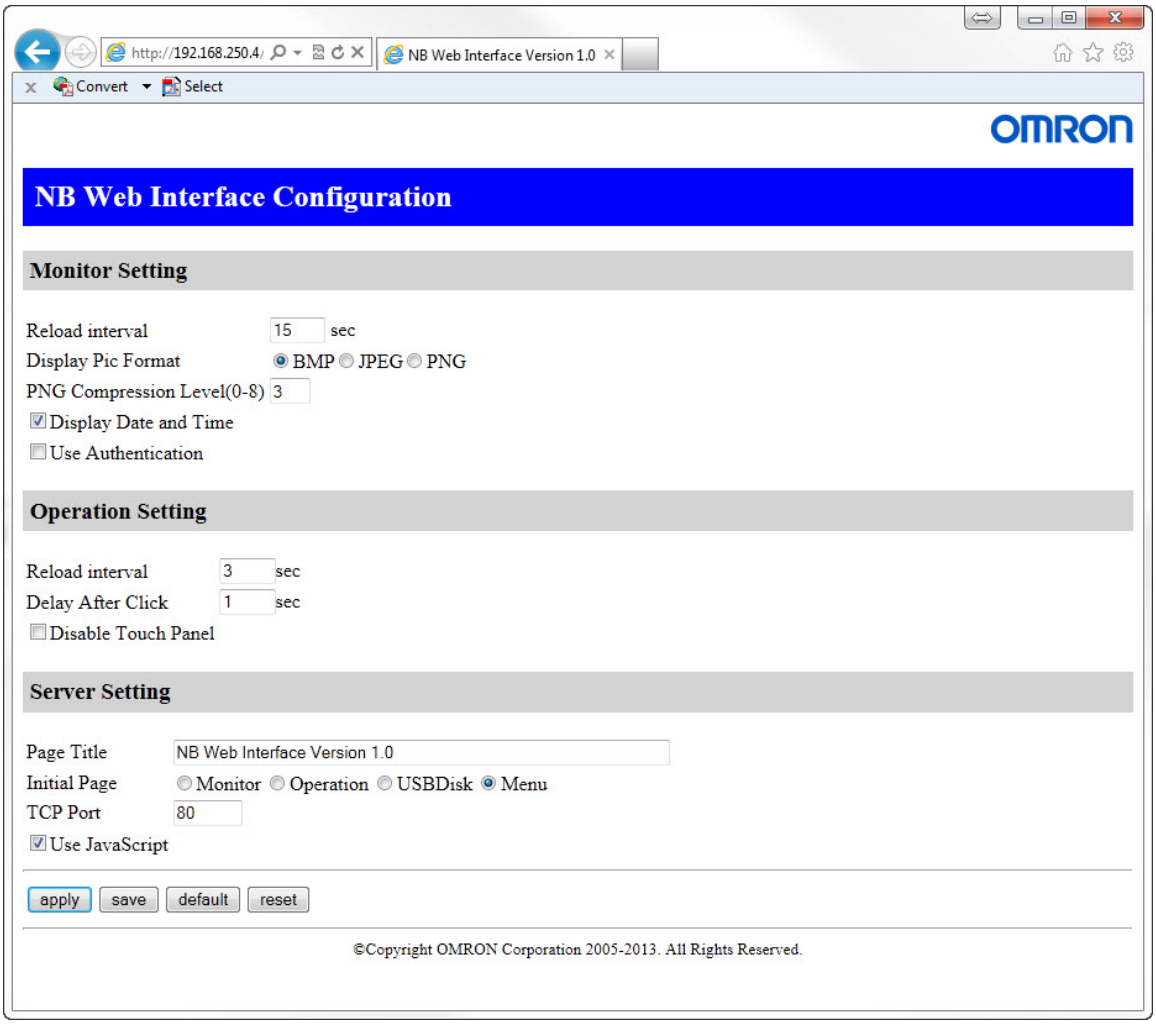


Figura 2.17: Configurações do NB *Web Interface* (OMRON, 2014).

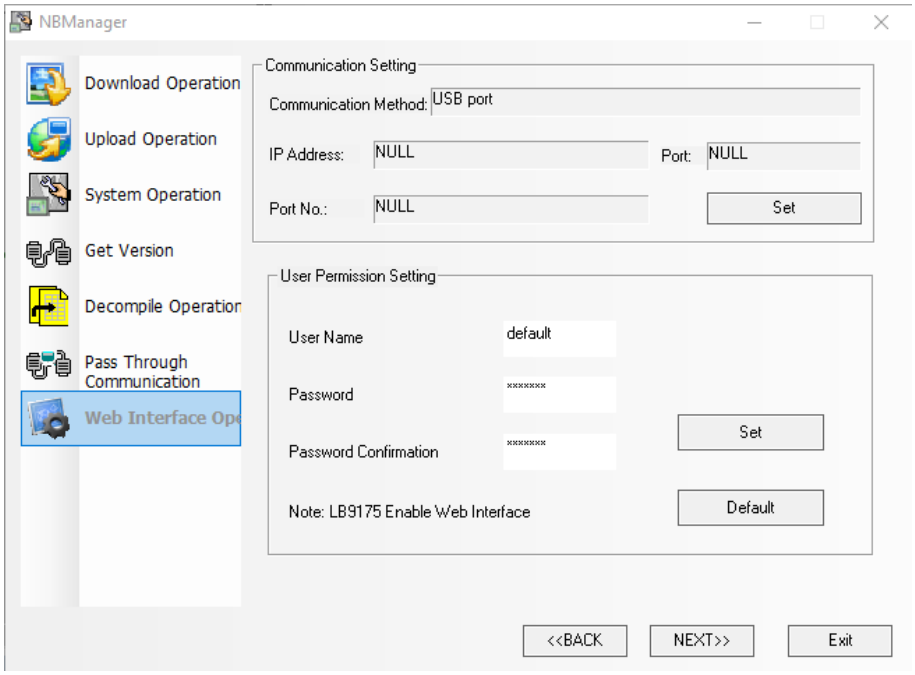


Figura 2.18: *NB-Manager* na secção do *Web Interface Operation.*

2.3.3.3 Servidor VNC

VNC é um sistema "*thin client*" baseado num *remote-display protocol* que é independente da plataforma e do sistema operativo. Este sistema permite obter acesso e visualização de interfaces gráficas remotas através de uma ligação segura (Richardson et al., 1998). Para estabelecer a ligação com o servidor VNC é necessário recorrer a um *software* VNC *client*, também designado VNC *viewer*, introduzir o endereço IP do servidor e autenticar-se com uma *password*. O porto TCP do servidor VNC é 5900 (OMRON, 2018a). Pode ser utilizado qualquer VNC *viewer* (gratuito ou pago) disponível para a plataforma, a partir da qual se pretende aceder ao servidor. Para dispositivos móveis a Omron possui uma solução gratuita de acesso remoto - Omron Remote Viewer para aceder às consolas NB (OMRON, 2018d), que pode ser utilizado tanto para aceder ao servidor *Web* como ao servidor VNC.

Nas consolas NB, o VNC veio como uma alternativa mais segura e "potente" do servidor *Web* para a Monitorização e Operação remota. VNC corrige as falhas que o servidor *Web* possui, como encriptação de dados na comunicação, possibilidade de permitir ou proibir a operação remota e tornar a experiência de acesso remoto mais fluida e cómoda, pois a imagem é atualizada com uma maior frequência, minimizando atrasos existentes no servidor *Web*.

Para ativar VNC nas consolas NB é necessário aceder à janela de configuração da HMI no *NB Designer* (Figura 2.19), onde existe um campo de ativação do VNC em conjunto com campos para introdução de *passwords* para os modos de Monitorização e Operação. As *passwords* somente suportam valores numéricos, podendo ser utilizado qualquer número no intervalo de 0 a 4294967295 (valor por omissão 888888). A proibição

ou a permissão do modo de Operação remota somente pode ser feita de forma programada ao ativar ou desativar o bit LB9291, respetivamente (OMRON, 2018a).

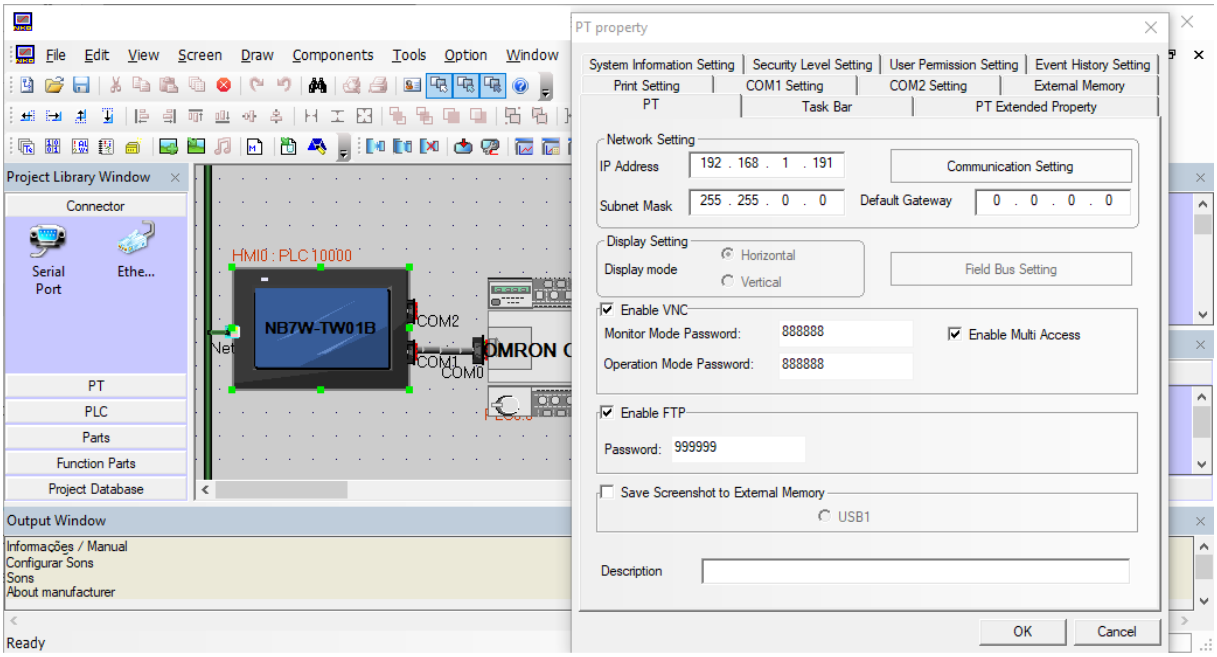


Figura 2.19: *NB Designer*, janela de configuração da HMI.

2.3.3.4 Servidor FTP

O FTP é um mecanismo padrão para transferência de ficheiros na rede disponibilizada pelo protocolo TCP/IP (Forouzan, 2009). Por sua vez, o servidor FTP disponibiliza um serviço de acesso, para utilizadores, a uma memória com ficheiros (disco rígido ou servidor de arquivos), que no caso deste projeto é uma *pen* USB. Para além disto, tem dois canais de ligação: um de controlo (utiliza o porto TCP 21) e um de dados (utiliza o porto TCP 20). De uma forma simples, a ligação de controlo, após estabelecimento de ligação, serve para enviar comandos e receber respostas FTP entre o cliente e servidor, e a ligação de dados serve para a transferência de ficheiros. Para estabelecer ligação é necessário definir o endereço IP do servidor e as credenciais de acesso, que são enviadas para o porto 21 do servidor, utilizando um *software* dedicado à comunicação FTP como FileZila (um *software* gratuito *open source*), ou linha de comandos Linux ou Windows. Também é possível aceder ao servidor FTP utilizando um navegador *Web*. Nas consolas NB esta ferramenta/funcionalidade veio como substituição da funcionalidade do servidor *Web* para aceder aos ficheiros da *pen* USB.

O protocolo HTTP é também considerado como um protocolo de transferência de ficheiros no entanto o FTP é um protocolo estritamente dedicado a esta causa, portanto é mais eficiente e flexível. Em conjunto com o servidor VNC é possível abdicar na totalidade do servidor *Web*. Para ativar o servidor FTP, tal como com o servidor VNC, é necessário aceder à janela de configuração da HMI (Figura 2.19). Na mesma secção da janela é possível definir a *password* de acesso, que somente pode ser um valor numérico

compreendido entre 0 e 99999999. O *username* é, por omissão, 'root', e não pode ser alterado.

2.4 Programação Android

O Android é um sistema operativo (*Operating System*, OS) móvel, *open-source*, desenvolvido pela Google principalmente para dispositivos móveis tácteis, como os *smartphones* e *tablets*. Segundo Statista (2018), desde 2009 que o Android tem vindo a crescer no mercado de *smartphones*, Em 2017, 85.9% de todos os *smartphones* vendidos possuíam o sistema operativo Android. A sua popularidade é justificada pelo facto de ser *open-source*, o que atrai grandes empresas de produção e desenvolvimento de tecnologia, para acelerar a inovação da tecnologia móvel de modo a oferecer aos consumidores uma melhor, mais rica e menos dispendiosa experiência (Open Handset Alliance, 2007). Tudo isto promove a formação de uma ampla comunidade de desenvolvedores e uma detalhada documentação, disponibilizando de forma gratuita tutoriais para programação Android, facilitando assim a aprendizagem e o aparecimento de novos desenvolvedores.

2.4.1 Arquitetura da Plataforma Android

A arquitetura do Android OS, representada na Figura 2.20, é considerada como uma pilha de *software* que consiste em diferentes camadas (Android Developers, 2018j), como:

- **Aplicações do Sistema** - é a camada superior que consiste num conjunto de aplicações base, com que todos os dispositivos Android vêm equipados, tais como navegador de *Internet*, contactos, *e-mail*, aplicação de envio de mensagens, calendário, entre outros (Android Developers, 2018j);
- **Estrutura da Java *Application Programming Interface*** (**API**) - é a camada que interage diretamente com as aplicações do sistema, e que consiste num conjunto de ferramentas e blocos de código reutilizáveis, que servem de base ao desenvolvimento de aplicações. Os desenvolvedores possuem acesso total a estes blocos de código para a criação de aplicações, utilizando-os da forma que lhes for mais conveniente (Android Developers, 2018j);
- **Bibliotecas C/C++ nativas** - são bibliotecas requeridas para suporte dos principais componentes e serviços do sistema Android, como por exemplo *Hardware Abstraction Layer* (HAL) e Andoid *Runtime* (ART). O acesso a funcionalidades desta camada é feito através da camada superior, que faz ponte com aplicações via Java APIs (Android Developers, 2018j). Exemplo destas bibliotecas é o Webkit, que disponibiliza ferramentas para a navegação *Web*; o OpenGL, que é utilizado para *rendering* do conteúdo gráfico 2D e 3D nos ecrãs; o SQLite, um mecanismo

leve de base de dados relacional, utilizado para fins de armazenamento de dados no Android; entre outras;

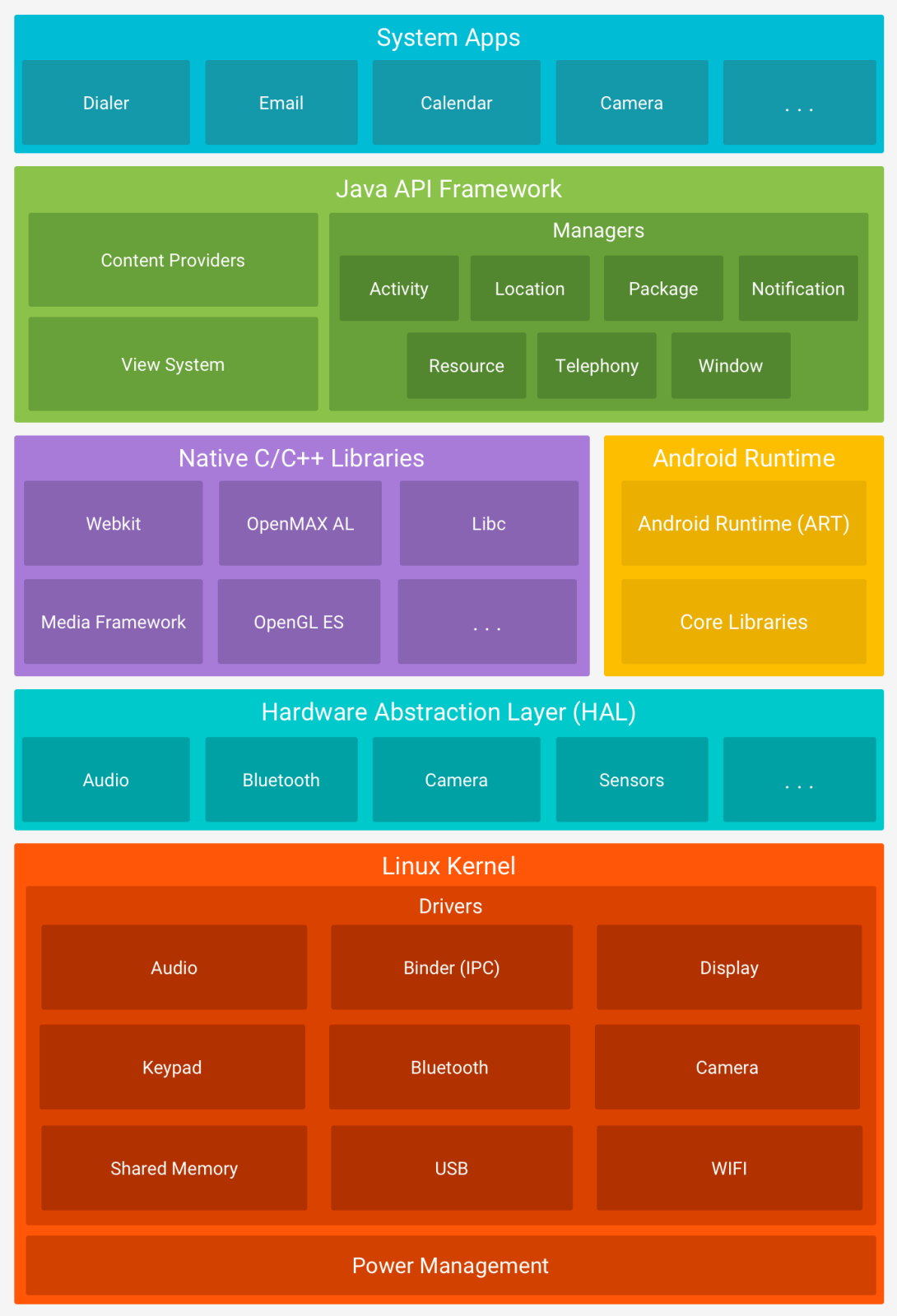


Figura 2.20: A pilha de *software* do Android (Android Developers, 2018j).

- **Kernel do Linux** - é a camada base do Android, que contém os *drivers* essenciais de *hardware* e interage diretamente com o *hardware* do dispositivo. A ART recorre ao

Kernel do Linux para efetuar tarefas de baixo nível como gestão de processos, gestão de memória, opções de segurança, gestão de ficheiros, *networking,* etc (Android Developers, 2018j).

- **Android *Runtime*** - responsável por compilar e correr as aplicações em máquinas virtuais nos dispositivos, otimizada de forma a utilizar os recursos disponíveis de forma eficiente. A ART, como o seu predecessor Dalvik *Virtual Machine*, foi especialmente desenvolvida para o Android (Android Developers, 2018j). A sua principal característica é a compilação *Ahead-of-time* (AOT), que traz uma melhoria no desempenho das aplicações. Esta camada também inclui as principais bibliotecas *runtime* para suportar as funcionalidades da linguagem de programação Java (Android Developers, 2018d);
- **Camada de abstração de *hardware* (HAL)** - consiste num conjunto de módulos com bibliotecas, que implementam uma interface para um tipo específico de *hardware* (exemplo: módulos de câmara e *bluetooth*), com intuito de expor as capacidades do *hardware* do dispositivo às camadas superiores (Android Developers, 2018j);

2.4.2 Fundamentos das Aplicações Android

A principal linguagem de programação em Android é o Java, no entanto, é possível desenvolver aplicações utilizando Kotlin, uma linguagem interoperável com Java, adicionada recentemente a partir da versão 3.0 do Android Studio (Android Developers, 2018h). Também se pode utilizar o C++ recorrendo ao Android *Native Development Kit* (NDK), para reutilizar as bibliotecas escritas em C++ ou mesmo criar aplicações, no entanto, não é recomendado para iniciantes e pode não compensar a complexidade adicional (Android Developers, 2018f).

O principal ambiente de desenvolvimento integrado (IDE - *Integrated Development Environment*) para o desenvolvimento de aplicações Android é o Android Studio da Google, baseado em Intellij IDEA (Android Developers, 2018i), que integra todas as ferramentas necessárias como Android *Software Development Kit* (SDK), emulador, *software* de controlo de versões, entre outros. O Android Studio é uma excelente plataforma de desenvolvimento, com constante suporte e atualizações, sendo que a última versão à data, 3.2, foi lançada em setembro de 2018 (Android Developers, 2018a). As aplicações e todo o seu conteúdo são compiladas pelo Android SDK para um arquivo Android *Package* (APK), que consiste num ficheiro único com extensão ".apk", que por sua vez é utilizado pelos dispositivos Android para a instalação das aplicações (Android Developers, 2018c). Cada aplicação após a instalação "vive" no seu próprio ambiente seguro, sendo isto conseguido porque o Android OS é um sistema multi-utilizador do Linux, em que cada aplicação é um utilizador único (*unique* Linux *user ID*). O sistema atribui permissões aos ficheiros da aplicação, o que garante que somente aplicações com o *user ID* especificado nas

permissões tenham acesso aos mesmos.

Por omissão cada aplicação é executada num único processo Linux, que é inicializado quando alguma componente da aplicação necessita de ser executada e é terminado quando já não é necessário ou quando o sistema necessite de recuperar a memória. Para além disto, cada processo possui uma máquina virtual própria, de modo a que o código de cada aplicação seja executado de forma isolada do resto das aplicações (Android Developers, 2018c).

O Android OS implementa o princípio do menor privilégio (*principle of least privilege*), isto é, cada aplicação somente tem acesso a componentes necessários para o seu correto funcionamento. Assim sendo, é conseguido um ambiente seguro onde é necessário atribuir explicitamente as permissões à aplicação para que tenha acesso a diferentes partes do sistema (Android Developers, 2018c).

2.4.2.1 Componentes da Aplicação

Uma aplicação Android é essencialmente composta por elementos básicos, denominados de *Application Components*, em que cada um deles possui um propósito distinto e comportamento característico. Existem quatro tipos de *Application Components,* em que uma aplicação é obrigatoriamente composta por pelo menos um destes componentes (Android Developers, 2018c):

- **Atividades -** principais pontos de interação com o utilizador, onde cada atividade representa um ecrã único com conteúdo gráfico de interface com o utilizador. Geralmente uma aplicação é constituída por mais de uma atividade com diferentes propósitos e com possibilidade de navegação entre elas;
- **Serviços -** componentes que não possuem interface com o utilizador e são executados em segundo plano, muitas vezes de forma oculta, para realizar operações de longa duração ou trabalho com processos remotos, como a sincronização de dados com um servidor remoto;
- ***Content providers* -** tem como objetivo a gestão do conjunto de dados compartilhados da aplicação. Os dados podem ser armazenados no sistema de ficheiros interno, numa base de dados SQLite, na *web*, ou em qualquer outro lugar de armazenamento persistente, que garante acesso à aplicação. Este componente é especialmente útil para uma leitura e escrita, de forma segura, dos dados privados da aplicação, sem que haja conhecimento para o exterior da mesma;
- ***Broadcast receivers* -** permitem a subscrição a certos eventos, transmitidos pelo sistema ou pela própria aplicação, como notificação de baixo nível da bateria, desligar do ecrã e términos de alguma operação programada pela aplicação (exemplo: fim do *download* de um ficheiro do servidor remoto).

O sistema Android foi projetado de forma a que uma aplicação possa inicializar um

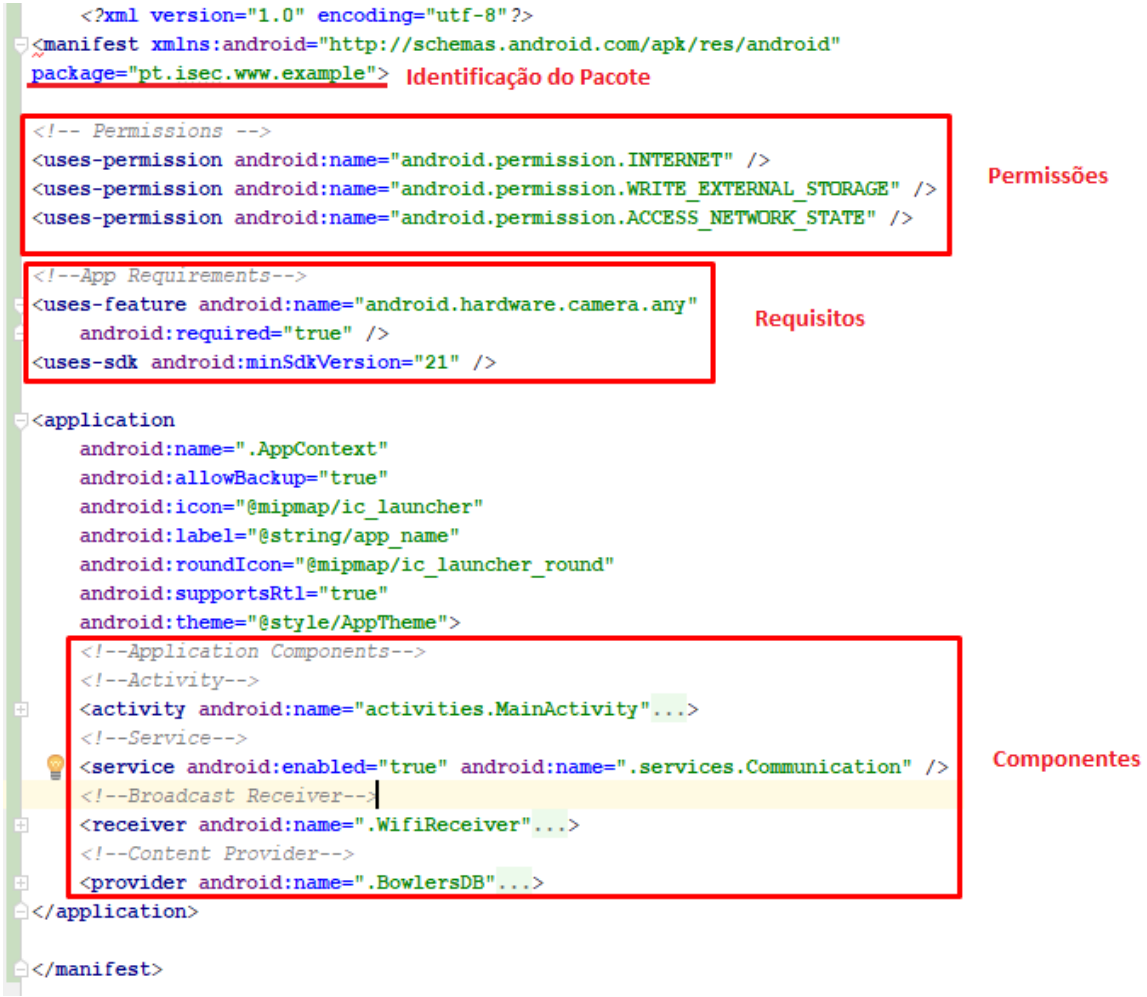
componente de outra aplicação, isto é, se um determinado componente de uma aplicação for capaz de realizar a tarefa pretendida, não há necessidade de desenvolvê-lo na própria aplicação, basta chamá-lo para a realização da tarefa.

A inicialização dos componentes é conseguida através de intenções, em inglês *Intents*, que são mensagens assíncronas que interligam diferentes componentes individuais durante a sua execução (Android Developers, 2018c). Os *Intents* podem ser implícitos ou explícitos, ou seja, definem uma mensagem para a inicialização de um tipo de componente ou de um componente específico, respetivamente. No caso de um *Intent* implícito, é inicializado qualquer componente que seja capaz de realizar a tarefa e, se houver mais do que uma aplicação com o componente pretendido, o utilizador escolhe qual pretende utilizar (Android Developers, 2018g). Geralmente, os *Intents* são utilizados para mostrar ou enviar algo, como por exemplo visualizar a localização no mapa ou abrir uma página *web*. No entanto em alguns casos é possível enviar um *Intent* de forma a receber algum resultado, como a captura de uma fotografia, em que o resultado é um *Uniform Resource Identifier* (URI) da localização da fotografia.

Os componentes de uma aplicação, para serem reconhecidos, devem ser declarados no ficheiro *manifest*. Cada aplicação possui, na sua raiz, um ficheiro *manifest*, ***AndroidManifest.xml***, que é um ficheiro *Extensible Markup Language* (XML) que apresenta a informação essencial da aplicação para o sistema Android. Este ficheiro contém informação sobre Android Developers (2018c):

- **Permissões** - para que uma aplicação tenha acesso a informações e dados externos e a ferramentas específicas, é necessário declarar as permissões no ficheiro *manifest*. Por exemplo, acesso à *Internet*, leitura e/ou escrita para a memória interna/externa, acesso a contactos, etc.;
- **Pacote** - designado de *package name*, é utilizado como identificador único do pacote Java da aplicação;
- **Requisitos** - são indicados os requisitos para os dispositivos Android para que possam executar a aplicação, tal como o nível mínimo da API, o tamanho do ecrã, o tipo do dispositivo, presença da câmara, etc.;
- ***Application Components*** - todos os componentes da aplicação devem ser declarados e descritos para informar o sistema Android e especificar a forma e circunstâncias em que são inicializados;
- **Bibliotecas** - devem ser listadas todas as bibliotecas para além das bibliotecas base, como por exemplo a biblioteca do *Google Maps*, caso seja utilizada.

Na Figura 2.21 está representado um exemplo de um ficheiro *manifest* com identificação das partes mencionadas anteriormente.




```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="pt.isec.www.example">

<!-- Permissions -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<!--App Requirements-->
<uses-feature android:name="android.hardware.camera.any"
    android:required="true" />
<uses-sdk android:minSdkVersion="21" />

<application
    android:name=".AppContext"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <!--Application Components-->
    <!--Activity-->
    <activity android:name="activities.MainActivity"...>
    <!--Service-->
    <service android:enabled="true" android:name=".services.Communication" />
    <!--Broadcast Receiver-->
    <receiver android:name=".WifiReceiver"...>
    <!--Content Provider-->
    <provider android:name=".BowlersDB"...>
</application>

</manifest>
```

Figura 2.21: Exemplo de um ficheiro *Manifest*.

2.4.2.2 Sistema de Compilação Gradle

O Android Studio utiliza o sistema de compilação Gradle, uma ferramenta para automação e gestão do processo de compilação. O Gradle é responsável por combinar todos os recursos e código-fonte da aplicação (ficheiros .Java e .xml) para compactar e converter tudo num ficheiro APK. Para este efeito existe um ficheiro ***build.gradle***, tal como o ficheiro *manifest,* localizado na raiz do projeto da aplicação. Algumas informações, como o nível mínimo da API, o nível da API dos dispositivos-alvo da parte dos requisitos do *manifest* e as bibliotecas, são declaradas no ficheiro ***build.gradle*** em vez do *manifest*, para maior conveniência na sua gestão. Para o preenchimento do ficheiro build.gradle é utilizada a linguagem sem formatação *Domain Specific Language* (DSL), para a descrição e manipulação da lógica de compilação que, posteriormente, é convertida em Groovy utilizada pelo *Java Virtual Machine* (JVM) (Android Developers, 2018e).

2.4.2.3 Recursos da Aplicação Android

A aplicação, para além do código Java, é constituída por ficheiros de recurso, que podem ser imagens, ficheiros de áudio, bem como todos os ficheiros relacionados com o

aspeto visual da aplicação. A partir dos recursos é possível definir as animações, os menus, os estilos, as cores, os temas e os esboços da interface com utilizador das atividades, recorrendo à linguagem XML. Os recursos ficam localizados na pasta '/res' na raiz da aplicação (Android Developers, 2018b; Android Developers, 2018c), sendo as principais sub-pastas:

- ***drawable*** - contém ficheiros com conteúdo gráfico, como imagens e ficheiros .xml, que definem algum objeto;
- ***layout*** - contém os ficheiros .xml que definem a aparência das atividades com interface de utilizador;
- ***menu*** - contém os ficheiros .xml que definem o aspeto e conteúdo de um menu;
- ***raw*** - onde geralmente são colocados ficheiros brutos como vídeos, ficheiros de áudio, ficheiros de base de dados, etc.;
- ***values*** - contém ficheiros .xml que definem vários valores, como as cores, as dimensões, os textos e estilos.

A utilização dos recursos torna a atualização e modificação da aplicação mais simples e facilitada, sem necessidade de alterar o código Java, como por exemplo a definição do tema geral da aplicação ou tamanho do texto numa determinada atividade. Um dos aspetos essenciais da utilização dos recursos é o facto de ser possível criar recursos alternativos para o suporte de diferentes configurações dos dispositivos. Isto é, para dispositivos que apresentem tamanhos de ecrã diferentes, rotação de ecrã e utilização de diversas línguas, é possível criar ficheiros .xml para cada caso específico, que sejam posteriormente reconhecidos pelo sistema e selecionados consoante necessidade. De forma a distinguir os ficheiros entre si, são adicionados qualificadores ao nome da pasta que os contém, por exemplo, quando é pretendido o suporte de várias línguas, são criadas pastas com os valores com o qualificador da língua ("/res/values-pt/" para o caso do português). Quando não existem ficheiros para um caso específico, são utilizados os ficheiros por omissão ("/res/values/" no exemplo anterior).

2.5 Considerações Finais

Para além dos tópicos abordados neste capítulo, para a aquisição de conhecimentos e preparação para o desenvolvimento, foram realizadas algumas atividades extra, tais como: formação acerca das consolas NB da Omron, cursos online gratuitos em programação Android e Java e aprendizagem de um sistema de controlo de versões.

Por estar diretamente relacionado com os seus equipamentos, a Omron ficou interessada na solução desenvolvida, pelo que ofereceram uma formação[4] de um dia sobre as HMI da série NB deles, como forma de apoio ao desenvolvimento da aplicação. A forma-

[4]Para mais informações sobre a formação pode aceder à página *web* da Omron (2018).

ção consistiu na introdução ao NB Designer e as principais funcionalidades das consolas, apoiados com uma componente prática. Foi uma excelente oportunidade para adquirir conhecimentos na parte de automação, que teve um impacto positivo no desenvolvimento do projeto e no estágio em geral.

A Secção 2.4 está integralmente baseada na informação disponibilizada pela Android na sua página *web* de desenvolvimento. No entanto, a aprendizagem sobre a programação, conceitos práticos e formas de a fazer de forma adequada foram conseguidas graças à vasta comunidade de desenvolvedores, que publicam tutoriais em fóruns (como StackOverflow, GitHub, entre outros), YouTube e *blogs*. Foram realizados cursos *online* gratuitos sobre a programação em Android e Java, dos quais se destacam os cursos da Udacity[5] e do SlideNerd[6], e que se revelaram importantes e úteis na aquisição e consolidação de conhecimentos.

Para além da aprendizagem do Android, foi fundamental adquirir conhecimentos na utilização de sistemas de controlo de versões, por ser uma prática importante no desenvolvimento de *software* de uma dimensão considerável, o que permite, em qualquer momento, rever e/ou reverter para versões anteriores, de forma cómoda e rápida, sempre que for desejável e/ou útil. Posto isto, optou-se por aprender a utilizar o git[7], com recurso a um curso *online* gratuito (Buckey e Spikes, s.d), disponibilizado pela Udacity, como um curso complementar ao desenvolvimento em Android.

Depois de explorados os conceitos teóricos, e antes de passar à implementação propriamente dita da solução, é necessário fazer um levantamento dos requisitos. Para realizar essa tarefa, antes, foi feito um levantamento das principais concorrentes da PROHS na tentativa de explorar e melhor identificar os requisitos a implementar na nossa solução. Este trabalho é apresentado no capítulo seguinte.

[5]Galpin et al. (s.d), Fujiwara et al. (s.d) e Nurik, Williams e Butcher (s.d)
[6]Ramesh (2015b), Ramesh (2016b), Ramesh (2015a) e Ramesh (2016a)
[7]Página oficial do The Git Project (2018)

Capítulo 3 - Análise da Concorrência e Definição dos Requisitos

Antes de iniciar a especificação e o desenvolvimento da aplicação, foi importante explorar e definir os requisitos da mesma, nomeadamente o que diz respeito ao modo como iria ser feita a monitorização remota do esterilizador. Um dos contributos para este processo foi recolhido através da identificação e análise das principais empresas (e respetivos produtos comerciais) concorrentes, ajudando assim a entender as suas soluções, nomeadamente quais as interfaces (que informação mostram e que tipo de ações permitem) e quais os meios tecnológicos envolvidos. O conteúdo deste capítulo foi condicionado à informação pública disponibilizada pelos respetivos fabricantes (através dos *sites* e brochuras), por questões de privacidade da informação. Para tal, foi elaborada uma lista de potenciais concorrentes e locais na *Internet*, como feiras de dispositivos médicos (MEDICA (2018) na Alemanha) onde se reúnem muitos produtores desta área e onde se poderiam encontrar novos concorrentes. Os principais concorrentes identificados são: MMM Group (Secção 3.1.1), Tuttnauer (Secção 3.1.2), Getinge (Secção 3.1.3), Steris (Secção 3.1.4), Steelco (Secção 3.1.5) e Matachana (Secção 3.1.6). O capítulo termina com uma secção de definição dos requisitos e considerações finais.

3.1 Análise da Concorrência

3.1.1 MMM Group

A Münchener Medizin Mechanik é uma empresa alemã, produtora de equipamento de limpeza, desinfeção e esterilização para ambiente hospitalar, que disponibiliza os seus serviços e equipamentos desde 1954. Tal como a PROHS, especializa-se em SCE e propõe soluções completas desde o planeamento e a instalação à consultoria e manutenção. O principal produto concorrente que se pode destacar é o autoclave horizontal Selectomat PL (MMM, 2018) disponível em vários tamanhos, com capacidade entre 2 e 18 *sterilization unit* (StU), com um HMI de 10".

Em relação ao *software,* relacionado com a monitorização remota, segundo a brochura da MMM (2016), dispõem de um *software*, *Intelligent Service Advisor* (ISA), para o auxílio da gestão dos equipamentos do SCE e o aumento da eficiência do pessoal do serviço. O ISA é responsável por enviar SMS ou *emails* com informação relevante acerca do estado do processo de esterilização como a ocorrência de um erro, o fim de execução e notificações de manutenção. Da análise e interpretação da brochura, o ISA consiste num mini-computador ligado diretamente ao equipamento com um servidor ISA, um programa para Windows, que recebe a informação do esterilizador e redireciona as notificações aos destinatários pretendidos (definidos na configuração).

3.1.2 Tuttnauer

A Tuttnauer é uma empresa holandesa, especialista em autoclaves, que opera na área da esterilização desde 1925. Para além das autoclaves, produz esterilizadores de plasma e distribui os restantes equipamentos de controle de infeção, como máquinas de lavar e desinfetar (Tuttnauer, 2018b). Destacam-se com esterilizadores horizontais, de baixa capacidade (1-4 StU) da série 55, de média capacidade (4-8 StU) da série 66 e de alta capacidade (7-12 StU) da série 69.

Para a monitorização remota dispõem de um *software* para computador, *R.PC.R Software* (Tuttnauer, 2018a). Este *software* (Figura 3.1) permite aceder ao estado atual do esterilizador (*real-rime autoclave display*), ou seja, é visível a imagem direta do ecrã do HMI. Também possui um histórico de ciclos dos parâmetros utilizados, onde é possível visualizá-los de forma gráfica ou numérica numa tabela em conjunto com ferramentas para a documentação. Também é referida a possibilidade de monitorizar até 8 esterilizadores ao mesmo tempo e que o *software* pode operar de forma *online* via *Ethernet* e *offline* via *USB flash drive*. Com isto, pode concluir-se que o *software* opera somente dentro da mesma rede e que a parte do histórico é responsável por apresentar a informação de forma agradável. Não foi possível apurar ao certo onde está guardada a informação dos ciclos (PLC ou HMI), mas é percetível que existem duas formas de a descarregar: pela *Ethernet* ou guardando diretamente num dispositivo de memória (*pen*), que posteriormente pode ser ligado a um computador com o *R.PC.R Software* para a visualização da informação. No catálogo (Tuttnauer, 2015) é, ainda, mencionado que é possível ter acesso remoto via *Internet* para o suporte técnico e que requer o *R.PC.R Software* e a instalação de um *router* celular no local para o acesso à *Internet*, o que confirma que este *software* de monitorização funciona somente dentro da mesma rede.

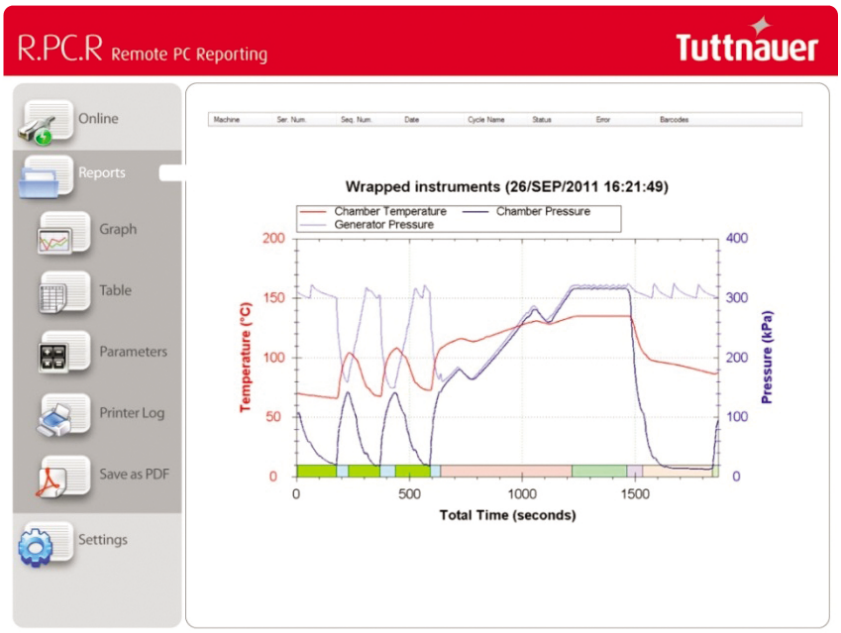
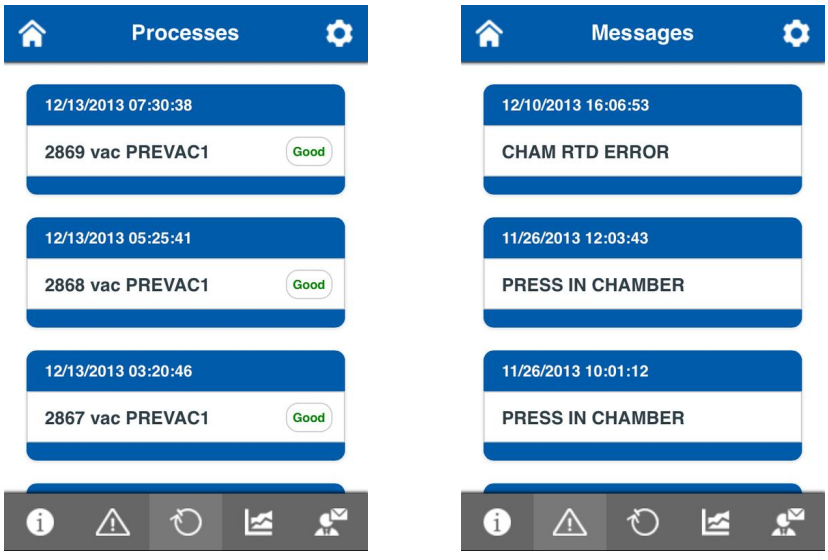


Figura 3.1: Captura de ecrã do *software* R.PC.R (retirado de Tuttnauer (2018a)).

3.1.3 Getinge

A Getinge é uma empresa sueca, atualmente de grande dimensão que apresenta várias soluções na área da saúde, como equipamentos de anestesia, ventilação, iluminação, equipamentos para o bloco operatório, e também na área da limpeza, desinfeção e esterilização. A empresa Getinge começou a prestar serviços na área de saúde em 1932 como produtor de esterilizadores para hospitais, embora a empresa já existisse enquanto produtora de equipamento agrícola desde 1904. Da área de esterilização podem destacar-se os esterilizadores horizontais *Getinge GSS67H* e *Getinge HS66*, ambos com vários tamanhos e capacidades entre 4 a 12 StU (Getinge, 2018).

Para a monitorização remota oferecem uma solução, *Getinge Online* (Getinge, 2013), que é uma plataforma de monitorização dos equipamentos (esterilizadores e máquinas de lavar) através de uma página *Web*. O *Getinge Online* permite monitorizar o estado atual do equipamento, visualizar a estatística do funcionamento, aceder ao histórico de erros e ao histórico dos ciclos de esterilização. Também permite o envio automático de notificações com informação relevante via SMS, *email* ou *pager*. Para usufruir desta plataforma é instalada uma carta de comunicações no equipamento que, através da infraestrutura de rede existente, envia informação via *Internet* para os servidores da Getinge, que disponibilizam a informação processada numa página *Web* (*Getinge Costumer Portal*). Segundo Getinge (2015) existe também uma aplicação para os *smartphones* (Figura 3.2), no entanto com algumas limitações em comparação ao acesso via *browser* à página web.



(a) (b)

Figura 3.2: Capturas de ecrã da aplicação Getinge Online para *smartphones*: (a) Histórico de ciclos; (b) Histórico de erros (imagens retiradas de Getinge (2015)).

3.1.4 Steris

A Steris é uma empresa americana, fundada em 1985, que adquiriu *Amsco International* em 1996, uma empresa pioneira e especialista na área de esterilização, que operava desde 1894. Os principais produtos que a Steris disponibiliza são equipamentos de limpeza, desinfeção e esterilização e também algum equipamento para o bloco operatório como iluminação e mesas de cirurgia (Steris, 2018). Dos equipamentos de esterilização pode destacar-se o *AMSCO Evolution Steam Sterilizer*, um esterilizador horizontal com capacidade entre 6 e 15 StU.

A Steris possui um *software CS-iQ® Sterile Processing Workflow Management Software* (Figura 3.3), composto por vários módulos, entre os quais um módulo de gestão do desempenho do esterilizador (Steris, 2013). Neste módulo é possível visualizar toda a estatística dos ciclos de esterilização incluindo indicadores de temporização e de fase, fitas do ciclo, gráficos, indicadores de pressão e de temperatura na câmara, tempo total dos ciclos e resultados dos testes de fuga. Também permite o envio de notificações para a equipa de manutenção sobre tarefas de manutenção preventiva e visualização do estado atual do equipamento. Este *software* insere-se numa infraestrutura dedicada, criada pela Steris, para a gestão e otimização do processo de esterilização, que requer uma instalação de *hardware* especifico nos esterilizadores e de servidores no hospital (Clark, 2015). Deduz-se que é um *software* multi-plataforma e que não haja acesso à informação para o exterior das instalações, uma vez que indicam existir outro *software* para apoio técnico, que requer uma instalação de uma infraestrutura dedicada que possa servir de *gateway*, ligado via *Ethernet* a todos os equipamentos, com endereços IP fixos, e que também deva ter acesso à *Internet*. Desta forma, Steris fica com acesso a toda informação relevante do equipamento e sempre que hajam anomalias pode intervir para as solucionar ou ajudar na sua solução.

Figura 3.3: Fotografia do *CS-iQ® Sterile Processing Workflow Management Software* num *tablet* (retirado de Steris (2013)).

3.1.5 Steelco

A Steelco, é uma empresa italiana, dedicada à produção de dispositivos e equipamentos de limpeza, desinfeção e esterilização (Steelco, 2018a), sendo a data da sua fundação desconhecida. Em termos de esterilização, dispõem de esterilizadores horizontais Steelco da série VS, com capacidades entre 1 a 18 StU.

Em relação à monitorização remota, a informação é escassa, no entanto, nas brochuras dos seus equipamentos, esterilizadores e máquinas de lavar, mencionam que existe um sistema de controlo, *SteelcoData*, ao qual é possível ligar a maioria dos seus equipamentos para uma monitorização remota (Steelco, s.d[a]; Steelco, s.d[b]). Este *software* permite guardar e visualizar o histórico de ciclos e monitorização remota do estado atual dos equipamentos. Mencionam, ainda, que é facilmente interligado com a infraestrutura do cliente e que é utilizado o servidor do hospital para guardar o histórico dos *backups*. A campanha de publicidade deste *software* está mais direcionada ao reprocessamento de endoscópios, *SteelcoData ARES software* (Steelco, 2018b), daí se deduzir que o *software* é destinado a computadores, operando somente dentro da instituição. Pensa-se, igualmente, que *SteelcoData* não é um *software* independente mas sim integrado numa solução de gestão do SCE.

3.1.6 Matachana

A Matachana é uma empresa espanhola fundada em 1962, que desenvolve, produz, comercializa e representa outras marcas internacionais de equipamentos para SCE, anatomia patológica e tratamento de resíduos biológicos contaminados (J. A. Matachana, 2017). Como produto concorrente pode destacar-se o esterilizador horizontal Matachana da série S1000, com capacidades entre 4 a 12 StU.

Para monitorização remota, a Matachana apresenta um *software* de monitorização e *traceability* dos equipamentos e material reprocessado no SCE, *EasyLOOK*. Este *software* possui várias *features*, das quais está a visão geral de até 5 equipamentos ligados (número da série, data do último ciclo, alarmes, número de ciclos, parâmetros do ciclo, etc.) e acesso à documentação do processo de esterilização, como o histórico de ciclos (Matachana, 2018). Para usufruir desta ferramenta é necessário a instalação de alguma infraestrutura no hospital, operando exclusivamente dentro da rede da instituição.

3.2 Definição dos Requisitos

Com a análise dos principais concorrentes da PROHS, sob a perspetiva de *software* de monitorização remota, foi possível obter uma noção geral das principais *features* que um *software* deste tipo possa precisar, assim como meios pelos quais foram conseguidos. Na Tabela 3.1 está representado o resumo desta análise.

Tabela 3.1: Resumo da análise da concorrência.

Concorrente	***Software* de Mon. Remota**[a]	***Features***	**Inf. Ad.**[b]	**Suporte Disp. Móveis**[c]	**Área de funcionamento**
MMM Group	ISA	Envio de notificações (ocorrência de erro, fim da execução do ciclo e necessidade de manutenção) via email ou SMS.	Sim	N/A[d]	Sem restrições (Email / SMS)
Tuttnauer	R.PC.R	Visualização do ecrã do HMI em tempo real; Histórico de ciclos; Ferramentas de documentação.	Sim	Não	Rede local
Getinge	Getinge Online	Visualização do estado dos equipamentos em tempo real (somente via página *Web*); Estatística de funcionamento do equipamento; Histórico de ciclos; Histórico de erros; Envio de notificações do estado do equipamento via email, SMS ou *pager*.	Sim	Sim	Sem restrições (*Internet*)
Steris	CS-iQ®	Visualização do estado dos equipamentos em tempo real (somente via página Web); Estatística de funcionamento do equipamento; Histórico de ciclos.	Sim	Sim	Rede local
Steelco	SteelcoData	Visualização do estado dos equipamentos em tempo real; Histórico de ciclos.	Sim	Não	Rede local
Matachana	EasyLOOK	Histórico de ciclos.	Sim	Não	Rede local

[a] *Software* de Monitorização Remota
[b] Requer a Infraestrutura Adicional
[c] Suporte para Dispositivos Móveis
[d] Não Aplicável

Na sua maioria, o *software* é destinado a funcionar nos computadores e dentro da mesma rede *Ethernet* das instituições. Para conseguir acesso à informação fora das instalações é sempre necessário algum mecanismo de retransmissão e proteção que desempenhe o papel de *gateway*. Adicionalmente, para um processamento da informação mais complexo, como dados estatísticos, é inevitável o recurso a um servidor.

Todas as soluções encontradas são apresentadas como opcionais, visto que, de alguma forma, requerem a instalação de uma infraestrutura adicional, que por sua vez implica mais despesa, tanto para os produtores como para os consumidores, que certamente terão custos adicionais para usufruir destas ferramentas.

Compilando a informação obtida da análise efetuada, para esta aplicação pode considerar-se como relevante a seguinte informação:

- Imagem direta do ecrã do HMI;
- Indicadores de interesse durante o ciclo (tempos, fase do ciclo, temperatura, pressão);
- Acesso ao histórico de ciclos (com representação dos dados em forma de tabela ou alternativamente em gráfico);
- Notificações do término do ciclo, tal como existência de falha/erro;
- Informação do estado da máquina (número de ciclos efetuados, tempo de funcionamento, tempo desde último Bowie-Dick e teste de fugas, dados estatísticos, etc.).

As principais *features* que, de alguma forma são comuns em todas as soluções, são a informação em tempo real do estado do ciclo e o acesso ao histórico de ciclos.

De realçar que o desenvolvimento foi planeado de forma a contemplar a integração das principais *features* recorrendo essencialmente a ferramentas e infraestruturas existentes (equipamento adicional, servidores dedicados, entre outros), minimizando assim os custos à PROHS e tornando esta aplicação, exclusiva a dispositivos móveis, num complemento à monitorização dos seus esterilizadores.

Capítulo 4 - Propostas de Solução de Monitorização Remota

Inicialmente, a instituição de acolhimento do estágio propôs que fosse desenvolvida uma plataforma capaz de efetuar a monitorização e manipulação remotas do esterilizador. No entanto, com o decorrer do estágio, concluiu-se que não haveria interesse na manipulação remota do equipamento. O motivo prende-se com o facto de nos momentos em que pudesse ser necessário manipular o esterilizador, haveria sempre um funcionário que o pudesse fazer no local, seja para introduzir material para esterilização e iniciar o ciclo, ou, para abrir a porta no final do ciclo para retirar o material esterilizado.

A primeira possível solução foi pensada antes do início do estágio, com vista a conseguir ambas as vertentes, monitorização e manipulação. Desta forma, em conjunto com os orientadores do ISEC, elaborou-se uma proposta de solução, de forma a aproveitar ao máximo a infraestrutura existente, *Web Interface* e *software* existente na HMI. A utilização do Modbus ficou para último recurso, pois era indispensável o redesenho de todas as janelas da HMI para conseguir a monitorização e manipulação completa do esterilizador.

Posteriormente, ao chegar-se à conclusão que a manipulação não tem interesse para esta aplicação, pensou-se na utilização do Modbus, uma vez que na monitorização, a única parte importante é a execução do ciclo, que se resume numa única janela da HMI. Como tal, a primeira solução ficou também mais simplificada. Posto isto, surge a segunda proposta de solução, em contrapartida à primeira, que recorre somente ao Modbus.

Por fim, a terceira e última proposta de solução, que engloba ambas as vertentes, surge após confirmação da possibilidade de criação de um histórico de ciclos, sendo a partir desta que se desenrola o desenvolvimento da aplicação.

4.1 Primeira Proposta

Como mencionado anteriormente, esta proposta foi pensada em conjunto com os orientadores do ISEC. Num dos laboratórios do ISEC (Laboratório de Redes) existia uma consola NB5, facto este que permitiu a preparação de um ambiente de testes.

Para a elaboração da proposta, a primeira etapa consistiu, inicialmente, na instalação do *NB-Designer* e de um simulador do *Android OS*, com a finalidade de averiguar o funcionamento do *NB Web Interface* e da aplicação da Omron para acesso remoto à consola. O recurso a um simulador deveu-se à grande limitação do *Omron Remote Viewer* em suportar, somente, dispositivos com ecrã maior que 7", algo que tanto o aluno como o professor não possuíam. Também foi necessário a familiarização mínima com o *software* de desenvolvimento, para criar um projeto simples para testes. Este projeto consistia

numa janela com dois indicadores LED e dois botões para os acionar. Após a configuração de um dispositivo virtual no simulador e a instalação da aplicação, prosseguiu-se para a comparação da visualização do funcionamento da mesma e acesso ao *NB Web Interface* a partir de um *browser* no computador. Concluiu-se deste modo que, após o estabelecimento da ligação e a escolha da forma de acesso, monitorização ou operação, a aplicação simplesmente apresentava a página *web* do *NB Web Interface*, exatamente igual como se tivesse sido acedida a partir de um *browser*.

De seguida tentou-se estudar o *NB Web Interface*, essencialmente em que consiste, como funciona e se é possível configurá-lo de alguma forma para conseguir o pretendido. Na sua documentação, a informação estava limitada às indicações de como ativar o *NB Web Interface* e à descrição das suas funcionalidades. Posto isto, seguiu-se para a análise das páginas-fonte do *NB Web Interface*, escritas em *HyperText Markup Language* (HTML), com o objetivo de compreender o funcionamento da interface gráfica. Após esta análise, verificou-se a possibilidade de replicar os pedidos da imagem e respostas dos *clicks* à consola, no caso do modo de operação, tal como as configurações da janela de configurações. Desta forma surgiu a ideia de utilizar máscaras invisíveis sobre a imagem proveniente da HMI, que delimitem regiões de interação, como botões, em conjunto com um sistema de autenticação e permissões para limitar o controlo remoto do esterilizador. Com a análise, foi igualmente percetível a possibilidade de ligar uma *pen* à consola e aceder ao conteúdo da mesma, e deste modo, ser possível guardar na *pen* as máscaras para cada janela do programa da HMI em forma de tabela, *Comma-Separated Values* (CSV). Posteriormente, consoante a janela que esteja a ser representada pela HMI, retirar e aplicar a máscara correta à imagem recebida. No entanto, para além da necessidade de criar máscaras para cada janela, é necessário criar algum mecanismo de identificação da mesma, pois quando é recebida a imagem pelo servidor *Web*, não existe forma de saber a qual janela corresponde a imagem, o que impede a correta colocação da máscara. Para contornar este problema, pensou-se adicionar algum identificador discreto num dos cantos de cada janela e, posteriormente, na aplicação com recurso a um processamento da imagem, identificá-la.

Para implementar um sistema de autenticação e permissões, é possível utilizar um servidor com uma base de dados, que pode estar localizado na PROHS, ou seja, para utilizar a aplicação, o utilizador necessitaria de autenticar-se com o servidor da PROHS, o qual, posteriormente, transmitiria de forma oculta as permissões e as credencias de acesso ao servidor *Web* da HMI. A partir do momento em que o utilizador esteja autenticado, pode estabelecer a ligação com o equipamento e monitorizá-lo ou operá-lo, de acordo com as suas permissões. A Figura 4.1 representa um esquema do funcionamento desta proposta, com indicações numéricas relativas à sequência de acontecimentos.

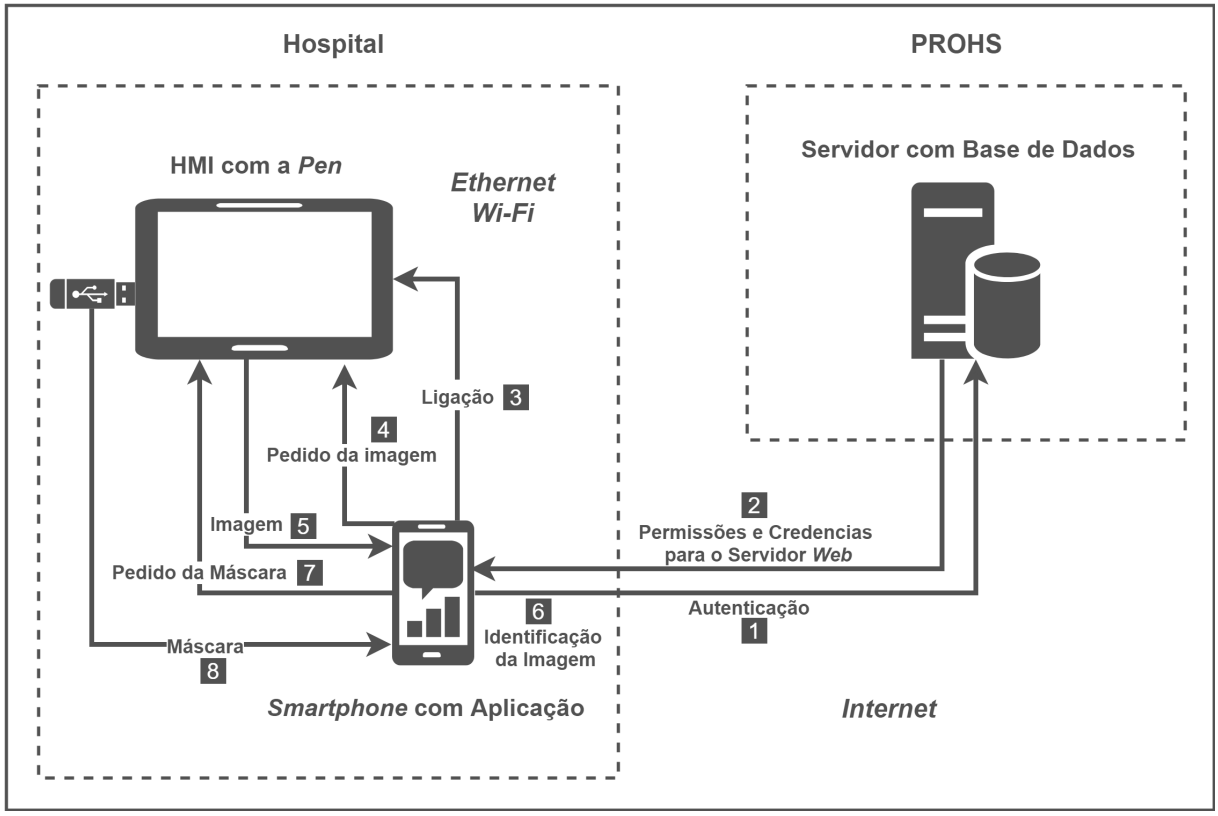


Figura 4.1: Esquema de funcionamento da primeira proposta.

4.2 Segunda Proposta

A partir do momento em que a utilização do Modbus passou a ser uma opção, iniciou-se o estudo do protocolo Modbus, o seu funcionamento nas consolas NB e, o mais importante, o Modbus em conjunto com o Android. Para as primeiras duas tarefas estudou-se a documentação publicamente disponível, sendo que para a terceira, no momento do estudo, a documentação era inexistente. Como tal, a pesquisa consistiu na procura de aplicações existentes que recorressem a Modbus, de preferência com código fonte *open-source* para compreender o seu funcionamento e a implementação da comunicação. Desta pesquisa, foi encontrado um projeto *Mobile Modbus* no github, criado por Ben Catlin (2013), que implementa um cliente Modbus em Android a funcionar no modo de *polling*. Este projeto serviu como ponto de partida para a compreensão do funcionamento do Modbus em Android e, apesar de ser antigo e desse modo impedir a compilação do projeto para observar a aplicação em funcionamento, trouxe informação importante, ou seja, a existência de bibliotecas Modbus em Java, como a biblioteca Modbus4J (Haggar e Bihler, 2016), utilizada por Ben Catlin. Esta biblioteca, desenvolvida e disponibilizada por Infinite Automation Systems e Serotonin Software, não trazia documentação, que por sua vez tornou a sua compreensão mais desafiadora, levando à procura de exemplos no *forum* de discussões na plataforma online dos proprietários da biblioteca (Infinite Automation Systems, 2018). No entanto após a pesquisa exaustiva, para além do *forum*, foi encon-

trada no github uma implementação desta biblioteca para Android, *Modbus4Android*, por um desenvolvedor, conhecido como zgkxzx (2018). O zgkxzx disponibiliza exemplos de utilização da sua implementação, o que acelerou imenso a aprendizagem e permitiu a criação de um projeto teste para confirmar a possibilidade de realizar o pretendido.

A partir do momento em que foram encontradas ferramentas para estabelecimento da ligação Modbus com a HMI e troca de informação, foi elaborada a segunda proposta. Esta proposta está exclusivamente baseada em comunicação Modbus, o que requer a criação de uma interface gráfica na aplicação Android para representar os dados pretendidos em tempo real. Tal como na primeira proposta, é também neste caso necessário um servidor com base de dados para o controlo de utilizadores e informação a que tem acesso, ou seja, informação necessária para estabelecer a ligação com o esterilizador. A partir do momento em que é estabelecida a ligação com a HMI (HMI desempenha o papel de servidor/escravo e o *smartphone* o cliente/mestre), é feito o *polling* dos dados. Na Figura 4.2 está representado um esquema de funcionamento da segunda proposta com indicações numéricas relativas à sequência de acontecimentos.

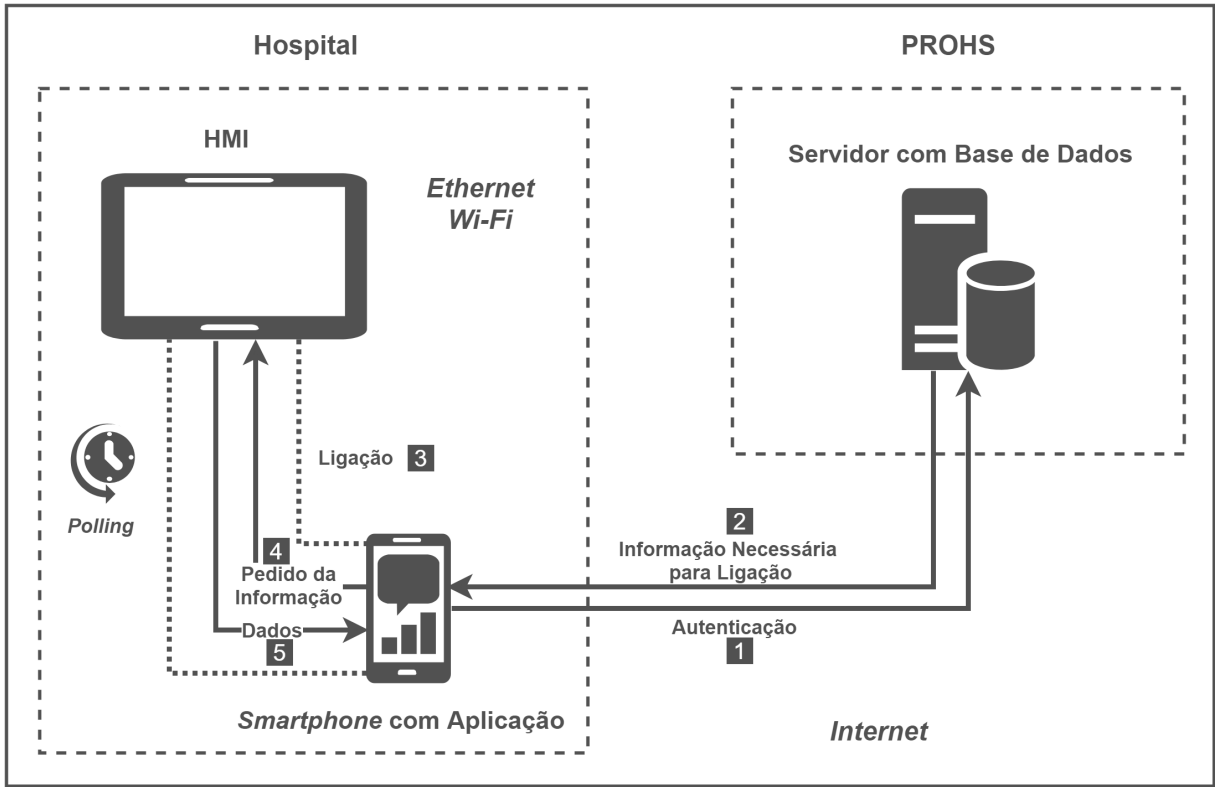


Figura 4.2: Esquema do funcionamento da segunda proposta.

A utilização do Modbus traz algumas vantagens em comparação com a utilização do servidor *Web*, das quais se destacam:

- Acesso direto a registos da consola - é possível aceder aos mesmo a qualquer altura independentemente do estado da máquina, desde que esteja ligada. Por exemplo, durante o ciclo o utilizador somente tem acesso a duas janelas, janela com indicadores do ciclo e janela com gráfico do ciclo, impossibilitando acesso a outras informações

que possam ser de interesse. Com o Modbus é possível aceder aos registos mesmo durante o ciclo;

- Quaisquer acontecimentos durante o ciclo, como erros e o término do ciclo, podem ser escutados separadamente em *background* e o utilizador pode ser notificado, por exemplo com uma notificação sonora ou vibração, o que não é possível com o servidor *Web*;
- Velocidade de transmissão dos dados - é possível amostrar os dados com maior frequência, tornando a experiência de utilização mais fluida.

Como desvantagens pode-se anotar a impossibilidade de aceder à imagem direta da HMI nem ao conteúdo da *pen*.

4.3 Terceira Proposta

Por fim, a terceira proposta, consistia em aproveitar ambas as vertentes, o servidor *Web* e o Modbus, com intuito de incorporar uma *feature* adicional à aplicação, para além da monitorização em tempo real - o histórico de ciclos, tendo-se optado por guardar o histórico de ciclos no próprio esterilizador.

A PROHS dispõe de uma solução para criação de um histórico de ciclos designada de *SpaceLogger*. O *SpaceLogger* é um pequeno arquivador de dados, que armazena os dados num cartão CompactFlash em formato digital, para posterior acesso a partir de um computador. No entanto, o *SpaceLogger* é uma ferramenta opcional, que pode ser instalada nos esterilizadores da PROHS, a pedido do cliente, o que envolve a articulação e integração com a estrutura tecnológica do cliente.

Da documentação do NB Designer (OMRON, 2018a) e do *forum* da Omron (OMRON, 2016), concluiu-se que existe a possibilidade de guardar dados em forma de tabela (em ficheiro CSV) na *pen* e, da primeira proposta, aceder e obter facilmente os ficheiros guardados na *pen*. Posto isto surgiu a ideia de guardar os ciclos na *pen*, de forma organizada após cada ciclo e desta forma criar o histórico de ciclos, exclusivo para cada máquina. O profissional da empresa, à data responsável pela programação das consolas, foi o responsável pela concretização desta ideia, com o auxílio do pessoal técnico da Omron, tendo elaborado o primeiro "protótipo" do histórico de ciclos, o que tornou a terceira proposta realizável. O "protótipo" consistia numa macro, que é executada na HMI após o término e impressão do ciclo, escrita em linguagem C e responsável por criar um ficheiro CSV e preenchê-lo com os dados do ciclo e do equipamento.

A terceira proposta consiste na comunicação Modbus TCP/IP para a monitorização em tempo real da execução do ciclo e, no servidor *Web*, a obtenção da imagem direta da HMI e acesso aos ficheiros da *pen* para o histórico de ciclos. Tal como nas propostas anteriores, é necessário um mecanismo de autenticação dos utilizadores e o controlo da

informação a que tem acesso, ou seja, conseguirem, exclusivamente, estabelecer a ligação com os seus esterilizadores. Nesta proposta, a pedido do responsável pela segurança de rede na PROHS, é evitado o uso do servidor interno, utilizando um servidor fora da rede principal ou procurando um serviço em *cloud*. Na Figura 4.3 está representado o esquema de funcionamento simplificado da terceira proposta. Os utilizadores devem autenticar-se de modo a obter a informação necessária para estabelecer as ligações com os equipamentos e o acesso às *features* da aplicação, desde que o dispositivo esteja na mesma rede que o esterilizador. No caso da monitorização em tempo real, deve ser criada uma interface gráfica para representar a informação proveniente do esterilizador. Para obtenção da imagem direta devem ser descarregadas as capturas de ecrã de forma periódica. Relativamente ao histórico de ciclos deve ser implementada uma navegação intuitiva pelo histórico e uma representação do ciclo.

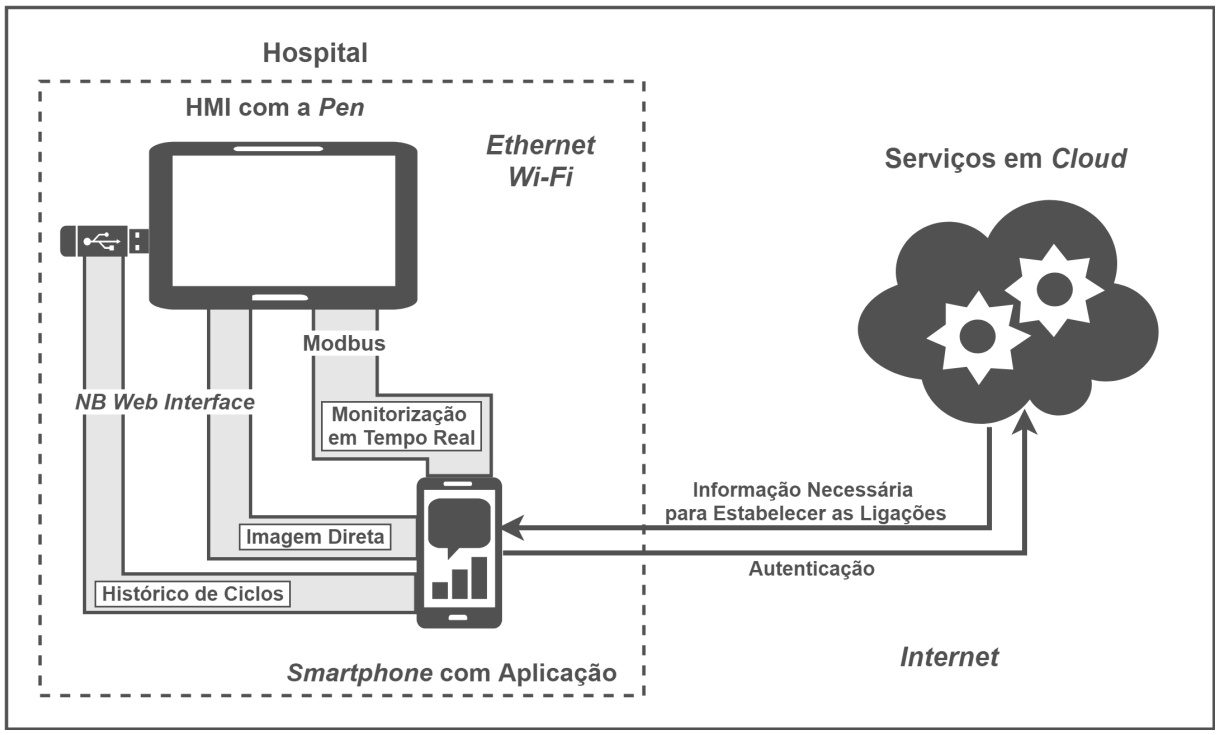


Figura 4.3: Esquema de funcionamento simplificado da terceira proposta.

4.4 Considerações Finais

Neste capítulo foram abordadas as diversas soluções para a resolução do problema pretendido, numa forma de percurso até à definição da solução final (terceira solução). O desenvolvimento desta terceira solução está descrito no capítulo seguinte (Capítulo 5).

Capítulo 5 - Desenvolvimento

O desenvolvimento segue a terceira proposta de solução, em que o esquema geral do sistema está representado na Figura 5.1. O esterilizador está ligado através da HMI à rede da instituição (onde for instalado: hospital, laboratório, etc.), via cabo *Ethernet*. O *smartphone* com a aplicação estabelece a ligação com o esterilizador, via *Wi-Fi* da rede, para a monitorização remota (monitorização em tempo real, via Modbus; e histórico de ciclos, via servidor *Web*). O utilizador deve autenticar-se com o servidor da Firebase e de ter acesso a *Internet* (através dos dados móveis ou da rede *Wi-Fi* da instituição) durante a inicialização da aplicação, de modo a obter a informação necessária para estabelecimento da ligação com o esterilizador.

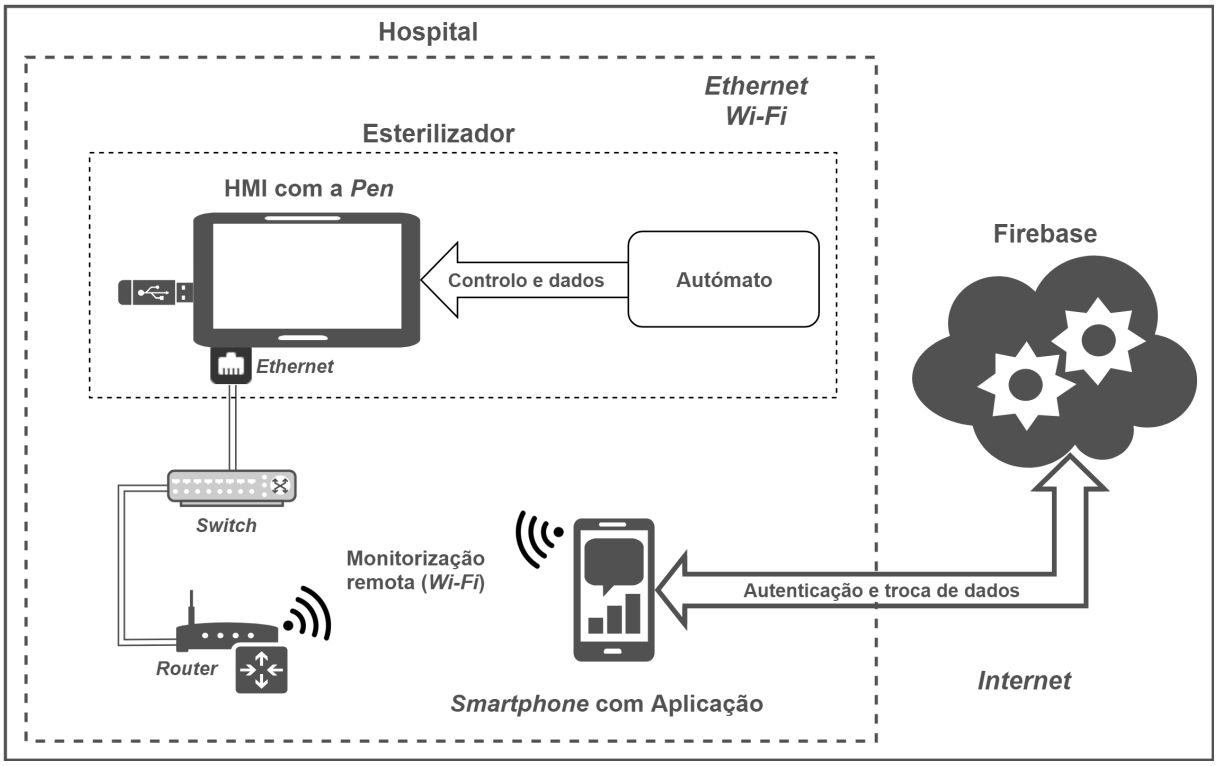
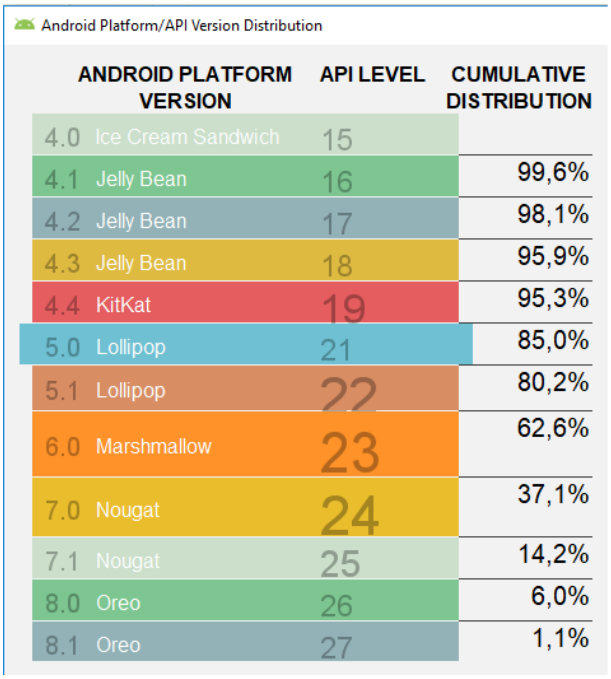


Figura 5.1: Esquema geral do funcionamento do sistema.

De modo a suportar a maior quantidade de dispositivos, sem complexidade acrescida no código, escolheu-se a API 21 como a API mínima (corresponde à versão 5.0 da plataforma Android). A versão alvo da aplicação é sempre a versão mais recente, como tal é a API 27 (versão mais recente à data). Durante a criação do projeto, o Android Studio ajuda a escolher a API mínima e em dezembro de 2018 estimou-se suportar 85% de todos os dispositivos Android (Figura 5.2).


Android Platform/API Version Distribution

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,6%
4.2 Jelly Bean	17	98,1%
4.3 Jelly Bean	18	95,9%
4.4 KitKat	19	95,3%
5.0 Lollipop	21	85,0%
5.1 Lollipop	22	80,2%
6.0 Marshmallow	23	62,6%
7.0 Nougat	24	37,1%
7.1 Nougat	25	14,2%
8.0 Oreo	26	6,0%
8.1 Oreo	27	1,1%

Figura 5.2: Distribuição comulativa das versões da plataforma Android nos dispositivos em funcionamento (dezembro de 2018).

Todo o desenvolvimento da aplicação foi realizado em inglês, incluindo os comentários dentro do código. O inglês é muitas vezes considerado como linguagem universal e é amplamente utilizado na comunidade de desenvolvedores Android, o que facilitará posterior interpretação do código. Deste modo a linguagem por omissão na aplicação é o inglês.

Por questões limitativas do *hardware* (portátil) utilizado no desenvolvimento, não foi possível recorrer aos simuladores de dispositivos Android para os testes do código. Como tal, recorreu-se a um dispositivo físico, *smartphone* do estagiário, um Huawei Honor 7, com versão 6.0 *Marshmallow* da plataforma Android (API 23). Este *smartphone* possui um ecrã de 5.2 polegadas com resolução *full high definition* (FHD) e com 423 dpi de densidade de pixeis. Assim, todas as capturas de ecrã realizadas neste capítulo foram feitas através deste dispositivo.

Este capítulo está organizado segundo a sequência cronológica do desenvolvimento durante o estágio:

- Histórico de Ciclos (Secção 5.1) - onde são abordados todos os passos do seu desenvolvimento, desde a consola NB à representação final na aplicação. O desenvolvimento desta componente, inicialmente definida como adicional/opcional, por questões de conveniência teve que ser desenvolvida em primeiro lugar;
- Navegação e *Design* Gráfico da Aplicação (Secção 5.2) - onde é abordada a forma de navegação pela aplicação e definido o seu aspeto visual;

- Monitorização em Tempo Real (Secção 5.3) - onde são abordados todos os passos do seu desenvolvimento, funcionamento e *layout* gráfico;
- Autenticação e *Back-end* (Firebase) (Secção 5.4) - onde são abordados todos os passos na implementação do sistema de autenticação, da base de dados e abordado o funcionamento final da aplicação;
- Instruções de Utilização (Secção 5.5) - onde são abordadas as instruções básicas de utilização da aplicação.

Por fim, são apresentadas as considerações finais deste capítulo (Secção 5.6).

5.1 Histórico de Ciclos

O desenvolvimento desta componente está dividido em quatro partes, organizadas de forma cronológica:

- Criação do Histórico de Ciclos na Consola (Secção 5.1.1);
- *Parsing* do Ficheiro CSV em Android (Secção 5.1.2);
- Aquisição dos Ficheiros CSV e Navegação pelo Histórico de Ciclos (Secção 5.1.3);
- Representação do Ciclo (Secção 5.1.4);

Por fim uma secção com as funcionalidades adicionais (Secção 5.1.5), que não foram mencionadas na parte do desenvolvimento.

5.1.1 Criação do Histórico de Ciclos na Consola

Antes de avançar com a programação, foram estudados os dados que podem e devem fazer parte do histórico. Este histórico fica guardado em ficheiro CSV tendo-se optado por salvaguardar os dados presentes na Tabela 5.1. Esta tabela apresenta também dos dados salvaguardados aqueles que devem ser visualizados na aplicação Android. Todos os dados possuem tamanho fixo, à exceção dos dados da temperatura e pressão na câmara e pressão na camisa, sendo estes dados dinâmicos, ou seja, não é possível prever à priori quantas linhas irão ocupar. Como tal, a estrutura do ficheiro CSV ficou definida de forma a ter um cabeçalho de tamanho fixo (17 linhas) com todos os dados com tamanho imutável, cada um em linha separada e depois seguido pelos dados de gráfico em que cada amostra ocupa uma linha e três colunas para cada tipo de dados (temperatura e pressão na câmara e pressão na camisa).

Tabela 5.1: Dados disponíveis para o histórico de ciclos.

Dados	**Visível (Sim/Não)**	**Dados**	**Visível (Sim/Não)**
Programa	Sim	Dado do ciclo 1[a]	Sim
Entidade	Não	Dado do ciclo 2[b]	Sim
Nome da máquina	Não	Temperatura máxima[c]	Sim
Número da série	Não	Temperatura mínima[c]	Sim
Utilizador	Não	Validade do ciclo	Sim
Número do ciclo	Sim	Número de erro	Sim
Hora de início	Sim	Temperatura na câmara	Sim
Hora do fim	Sim	Pressão na câmara	Sim
Data	Sim	Pressão na camisa	Sim

[a] Geralmente é a temperatura de esterilização. Em casos particulares pode ser a temperatura de aquecimento ou o tempo de estabilização dependendo dos ciclos, respetivamente o pré-aquecimento e o teste de fugas.
[b] Dependendo do ciclo pode ser o tempo de esterilização (ciclo de esterilização ou teste de Bowie&Dick), tempo de aquecimento (pré-aquecimento) ou tempo de teste (teste de fugas).
[c] Temperatura máxima e mínima durante a fase de esterilização.

Para além da estrutura do ficheiro CSV foi necessário definir de que forma estarão organizados e identificados os ficheiros dentro da memória externa. Foi decidido agrupar os ciclos por dia de trabalho do esterilizador e identificá-los por número do ciclo, ou seja, dentro da memória externa, no diretório principal é criada a pasta "*Cycle History*", que irá conter pastas com ciclos para cada dia de trabalho do esterilizador. Estas pastas são identificadas por data em que dia, mês e ano estão sem qualquer separador, tal como representado na Figura 5.3. Como por exemplo a pasta "03072018", que representa a pasta do dia 03/07/2018, contendo cinco ciclos, com os números de ciclo de 23 a 27.

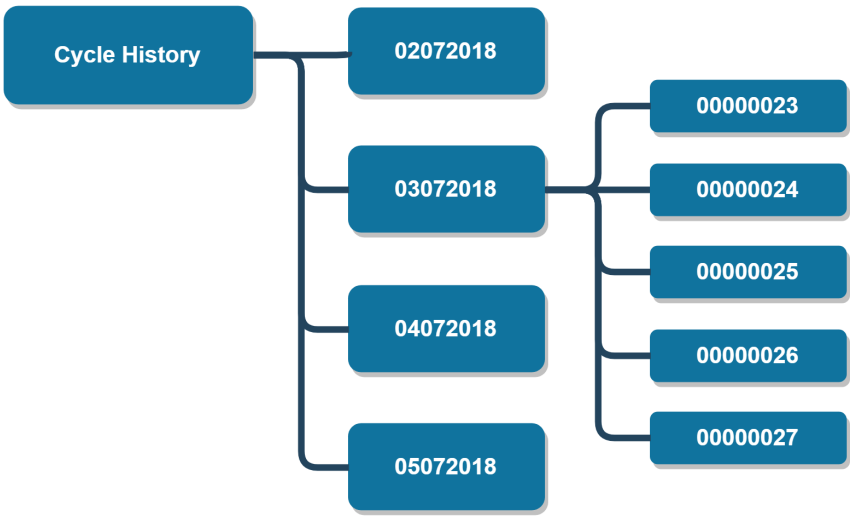
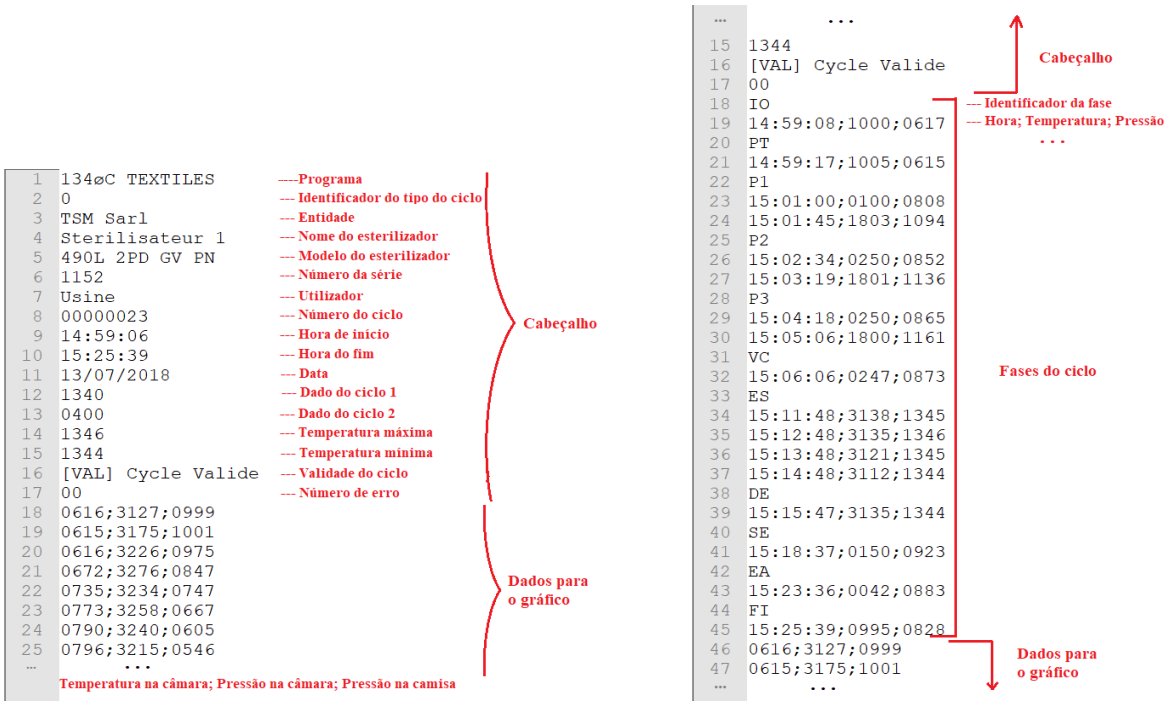


Figura 5.3: Diagrama de organização da pasta com histórico de ciclos.

Após estar tudo definido, a PROHS avançou com a preparação de macros para guardar os dados e em paralelo foi feita a pesquisa em como ler e interpretar os ficheiros CSV no Android.

Posteriormente, durante a implementação do histórico de ciclos entendeu-se que os dados para o histórico de ciclos mencionados anteriormente não eram suficientes para a representação do ciclo. Isto é, porque para a construção da tabela com as fases do ciclo, inicialmente, pensou-se em deduzir os valores a partir dos dados destinados para o gráfico do ciclo, mas como estes dados são amostrados a cada 4 segundos os valores deduzidos não corresponderiam aos momentos exatos das fases, desta forma divergindo dos valores impressos pelo esterilizador e tornando o histórico de ciclos incredível. Como tal, foi necessário acrescentar aos dados os valores das fases do ciclo. A dada altura a tarefa de atualização do código na consola ficou ao cargo do estagiário. A tarefa consistiu na análise do código existente, para compreensão mais detalhada do seu funcionamento, e na criação de um algoritmo para guardar os dados das fases de forma organizada e que facilite o seu *parsing* do lado da aplicação Android. Como o código para o *parsing* dos ficheiros CSV já estava desenvolvido, para a versão anterior do histórico de ciclos (Figura 5.4(a)), para minimizar a quantidade de alterações no mesmo, decidiu-se em incorporar os dados das fases dos ciclos entre o cabeçalho e os dados do gráfico (Figura 5.4(b)). Na Figura 5.4 estão representadas as versões dos ficheiros CSV dos ciclos para o histórico de ciclos, sem e com os dados das fases do ciclo incluídos.



(a) Versão antiga, sem os dados das fases do ciclo incluídos.

(b) Versão mais recente, com os dados das fases do ciclo incluídos.

Figura 5.4: Versões dos ficheiros CSV dos ciclos para o histórico de ciclos com legendas.

As fases do ciclo na sua grande maioria são representados por uma amostra de dados (hora, pressão e temperatura na câmara) no instante inicial da fase, à exceção dos pulsados, da esterilização, do aquecimento e de teste de fugas. Os pulsados são representados por duas amostras de dados, instantes do mínimo e do máximo do pulsado. A fase de esterilização, a fase de aquecimento e a fase de teste de fugas estão representadas por uma quantidade de amostras que depende da sua duração, com uma amostragem a cada minuto. Para guardar as fases do ciclo no ficheiro CSV optou-se por utilizar identificadores das fases (Tabela 5.2), em vez das suas designações por extenso, devido às designações das fases dependerem da configuração da língua na HMI. Em virtude da imutabilidade dos identificadores são evitadas as complicações no *parsing* nesta parte na aplicação. Foram reutilizados os identificadores criados pelo profissional da PROHS, à data responsável pelo departamento técnico, para o protocolo de comunicação com o *SpaceLogger*, evitando assim a introdução de novos dados no programa do autómato.

Tabela 5.2: Correspondência entre as fases existentes e os seus identificadores imutáveis.

Fase	**Identificador**	**Fase**	**Identificador**
Inicio da Operação	IO	Secagem	SE
Aquecimento	AQ	Entrada de Ar	EA
Pré-Tratamento	PT	Fim	FI
Pulsado #[a]	P#[a]	Vácuo	VA
Vapor na Câmara	VC	Teste de Fugas	TE
Esterilização	ES	Arrefecimento	AR
Exaustão Lenta	EL	Ondulação	OW
Descompressão	DE	Erro	ER

[a] O # corresponde ao número do pulso, onde durante o ciclo de esterilização podem existir até 8 pulsos, com o mínimo de 1 pulso para o caso de programa rápido.

5.1.2 *Parsing* do Ficheiro CSV em Android

De forma a desenvolver código reutilizável e transversal, foram criados métodos e objetos para o processamento da informação proveniente do ficheiro CSV, independentemente de como e onde foi obtido o ficheiro.

Inicialmente foram criados objetos que irão conter as variáveis para cada dado do ciclo e os métodos para o seu preenchimento e posterior acesso. De seguida foi criado o método (getParsedData) para processar o ficheiro e guardar os dados, cada um na sua respetiva variável dentro do objeto (CycleSample) que represente o ciclo. O método está dividido em três algoritmos responsáveis pela leitura (sequencial de cada linha), processamento e armazenamento: dos dados do cabeçalho, das fases do ciclo e dos dados para o gráfico. Na Figura 5.5 está representada a visão geral do *parsing* do ficheiro CSV com os dados do ciclo.

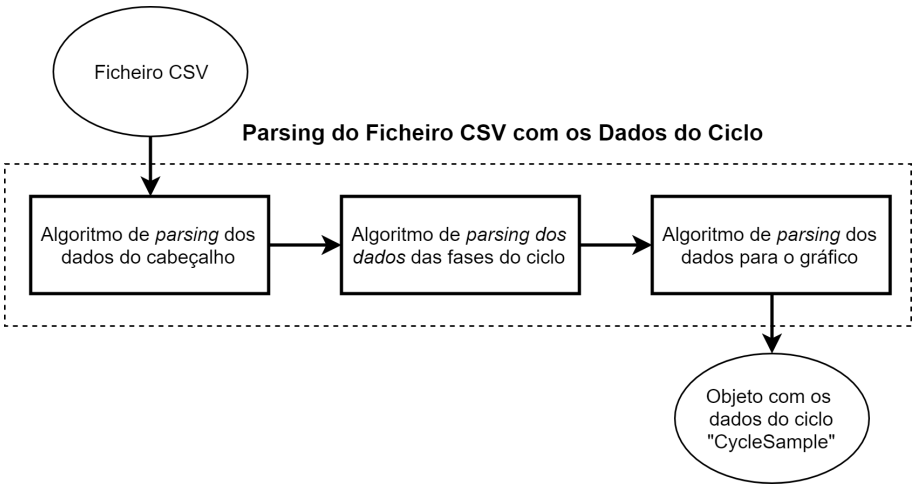


Figura 5.5: Processo resumido do *parsing* do ficheiro CSV em Android.

Algoritmo de *Parsing* dos Dados do Cabeçalho

Devido à estrutura e tamanho fixo do cabeçalho, a localização dos dados é bem conhecida, pelo que o algoritmo para o *parsing* do cabeçalho consiste na leitura, transformação dos dados para unidades corretas (caso necessário) e posterior armazenamento dos dados de cada linha.

Algoritmo de *Parsing* dos Dados das Fases do Ciclo

Para processamento dos dados das fases ao contrário do cabeçalho e dos dados para o gráfico é necessário construir um mecanismo para o correto processamento, principalmente porque a quantidade dos dados para cada fase do ciclo é variável, o que por sua vez também traz um desafio para a sua correta representação a seguir.

Podem-se considerar duas estratégias diferentes para superar esta dificuldade, uma primeira mais direta que consistiria na construção de tabelas, prevendo todas as situações possíveis para a quantidade de dados e de estrutura do ciclo, e uma segunda que consiste na criação de um algoritmo independente capaz de fazer o *parsing* desta parte, de forma a conseguir reconhecer os identificadores, os dados e a quantidade de dados pertencente a cada fase. A primeira estratégia torna-se inviável com projetos de maior dimensão, devido ao tempo e esforço para a previsão de todos os casos possíveis ser elevado e à elevada probabilidade de não contemplar algum caso. Para além disto, tornaria o código inflexível e frágil a alterações e/ou situações ocasionais (como por exemplo, ocorrência de um *bug* ou falha). Como tal, optou-se por projetar e construir um algoritmo minimamente sofisticado, *future-proof* e que faça o *parsing* dos dados de forma esquemática, capaz de analisar e guardar eventuais imprevistos, sem que seja provocado um erro (exceção).

Para além dos identificadores e dos dados das fases, acrescentou-se mais uma variável durante o *parsing*, para um efeito explicativo designada de "Quantidade", que representará a quantidade de linhas de dados para cada fase, estando inicialmente definida a zero.

Esta variável é incrementada após a leitura duma linha com dados da fase e o seu valor é guardado em conjunto com os restantes dados durante o *parsing* dos dados do ficheiro CSV.

A análise dos dados das fases é iniciada logo após o *parsing* do cabeçalho, com a leitura da linha seguinte do ficheiro CSV. De acordo com a Figura 5.4(b), as linhas podem conter unicamente uma ou três colunas com dados, separados por um ponto e vírgula. Assim:

- Caso a linha contenha somente uma coluna, o conteúdo presente na linha corresponde ao identificador da fase. Nesta situação, é guardado o valor da variável "Quantidade" em conjunto com o identificador da fase, à exceção da primeira iteração, onde a variável "Quantidade" ainda não sofreu alterações, sendo somente guardado o identificador, prosseguindo com a leitura da linha seguinte;
- Caso contenha três colunas, correspondem aos dados da fase precedente, pela respetiva ordem: a hora, a pressão e a temperatura. Nesta situação, é incrementada a variável "Quantidade" e guardados os dados presentes na linha.

Os identificadores das fases, as quantidades e todos os dados da fase são guardados nas listas separadas. Como a quantidade de dados para cada fase não é igual e dependendo do ciclo, algumas fases possuem a quantidade de dados variável (como o caso da fase de esterilização), a lista com quantidades serve de ponte para interligar a lista com as fases às listas com os dados correspondentes (Figura 5.6). A leitura e análise da parte das fases do ciclo continua até não encontrar o identificador de fim do ciclo (FI) ou o identificador de erro (ER). Na Figura 5.7 está representado o fluxograma do funcionamento do algoritmo de *parsing* dos dados das fases do ciclo.

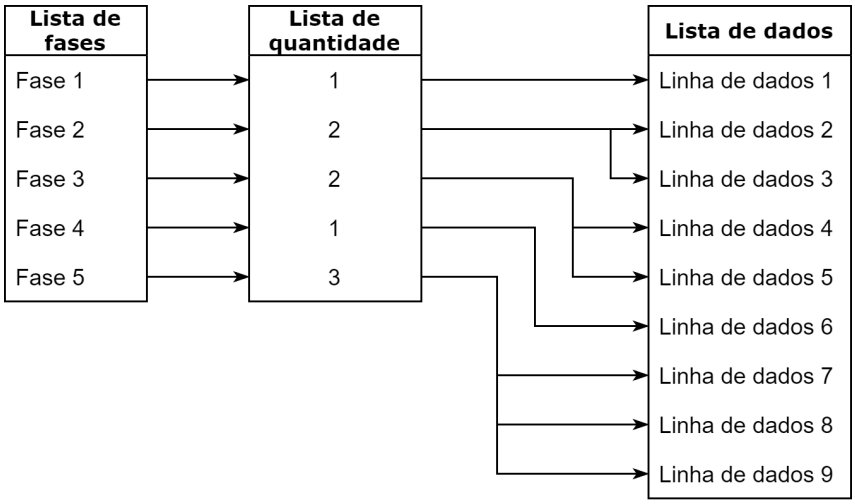


Figura 5.6: Diagrama de correspondência entre a fase e os seus dados .

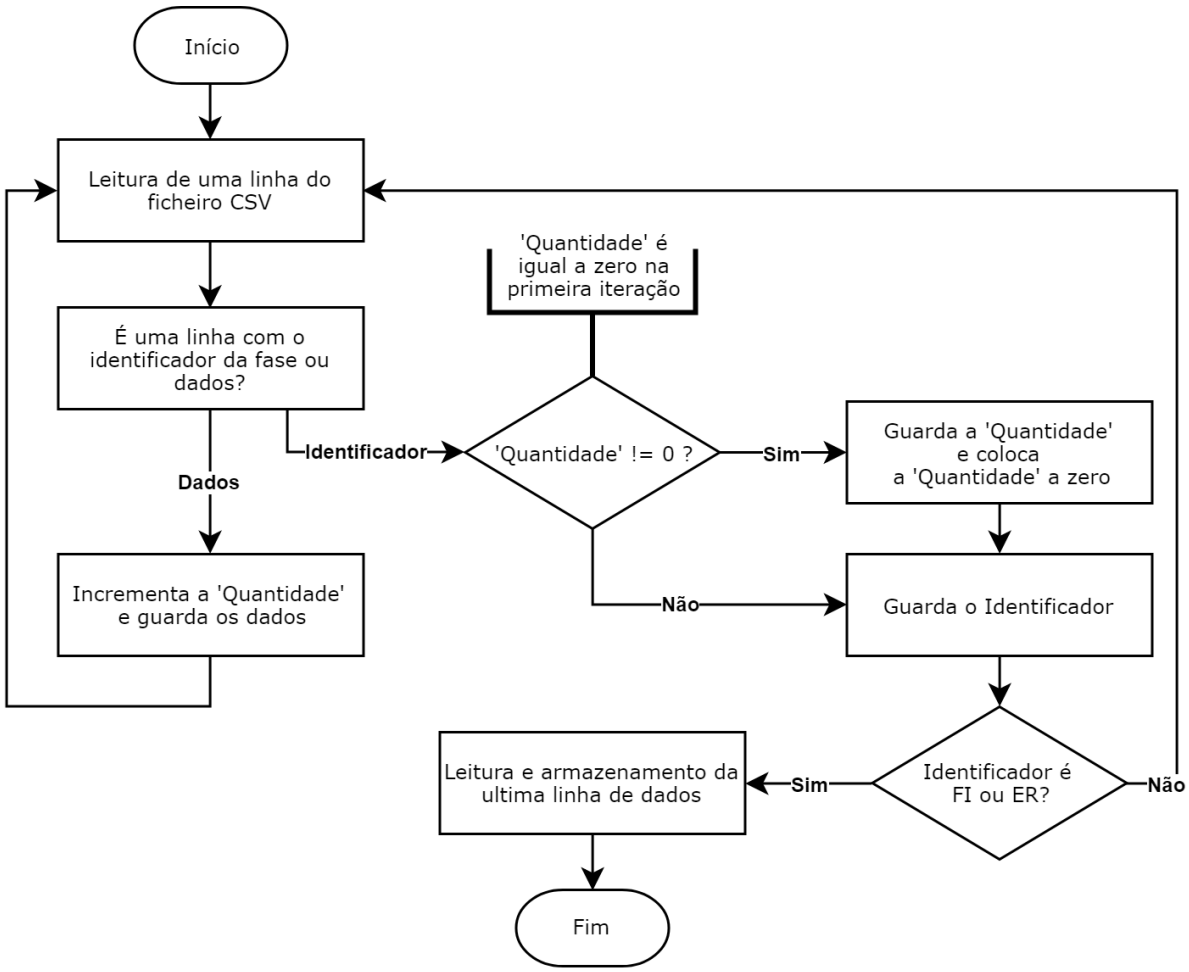


Figura 5.7: Fluxograma da análise da parte das fases do ciclo do ficheiro CSV.

Algoritmo de *Parsing* dos Dados para o Gráfico

A parte dos dados para o gráfico está estruturada de forma uniforme, começando logo após os dados das fases do ciclo e finalizando com uma linha em branco. A partir desta estrutura, o algoritmo de *parsing* consiste num *loop* que termina ao encontrar a linha em branco, sendo lida uma linha de cada vez, processando os dados de forma a guardá-los na unidade correta e nas listas correspondentes. Por exemplo uma linha de dados (0252;1009;0077) é guardada como 25.2 (corresponde aos ºC), 1.009 e 0.077 (correspondem a Bar). Para posteriormente estes dados poderem ser representados em forma de gráfico, é criada mais uma lista com a hora em que os dados foram amostrados. Esta lista é preenchida em conjunto com os restantes dados, em que se recorre à hora do início do ciclo para a primeira amostra e soma-se sucessivamente 4 segundos para as próximas.

5.1.3 Aquisição dos Ficheiros CSV e Navegação pelo Histórico de Ciclos

Para a parte do *parsing* foi utilizado um ficheiro CSV de testes guardado dentro da aplicação, em forma de recurso *raw*, e após o *parsing* estar bem sucedido, a etapa seguinte consistia na aquisição dos ficheiros da *pen* através do servidor *Web*. Esta etapa

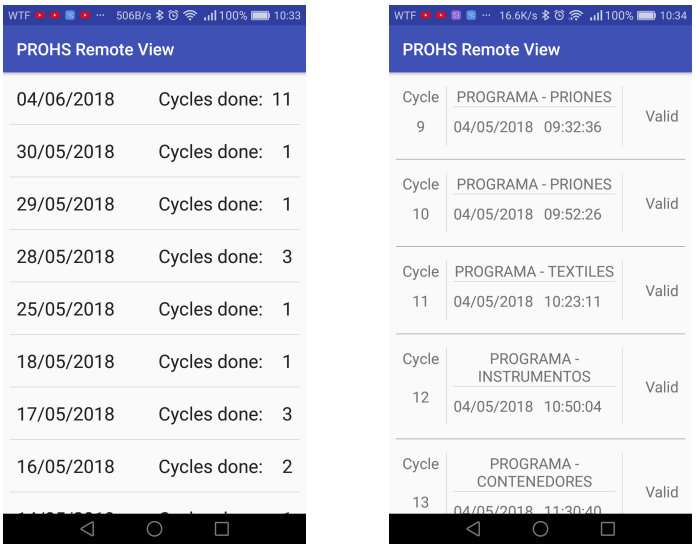
resumia-se essencialmente na navegação pelo histórico e *download* dos ficheiros com ciclos. Para este efeito, a primeira tarefa estava direcionada na análise das páginas HTML do servidor *Web* correspondentes ao acesso à pen, com intuito de entender se existe alguma forma de questionar o servidor para obter a informação desejável, como a quantidade de pastas, a quantidade de ficheiros dentro e os seus nomes. A informação das páginas está organizada na forma de uma tabela (Figura 5.8, retângulo azul), com as hiperligações para cada subsequente pasta ou ficheiro em conjunto com a data e hora da sua criação e, no final da tabela, o número total das pastas/ficheiros. Após esta análise, não foi possível obter qualquer indicação sobre como foi criada a tabela, nem como aceder às informações individualmente.

Como tal, restou uma solução em fazer o *parsing* das páginas HTML para obter a informação de interesse, pelo que se recorreu à biblioteca jsoup, uma biblioteca Java *open-source* desenvolvida por Jonathan Hedley capaz de analisar, extrair e manipular os dados presentes num documento HTML (Hedley, 2018). A API da jsoup possui métodos para estabelecer a ligação com a página *Web* (recorrendo ao *Uniform Resource Locator* (URL) da mesma), extrair o documento HTML da mesma e repartir o documento em elementos dependendo da *query* realizada. São utilizados seletores nas *queries* para identificar os elementos desejáveis, neste caso são $'a[href]'$ e $'p'$, em que o primeiro é um seletor combinado para identificar as hiperligações das pastas e o segundo para obter o número de pastas/ficheiros presentes. Na Figura 5.8 está representada a captura de ecrã da página fonte da raiz da *pen* acedida pelo servidor *Web*, com as informações de interesse delimitadas e sublinhadas a vermelho. Neste caso não está presente a pasta com o histórico de ciclos, devido ao intuito desde exemplo consistir em justificar a escolha dos seletores e de demonstrar a estrutura típica das páginas, em que a primeira hiperligação é da pasta mãe (serve para retroceder), seguida pelas hiperligações das pastas existentes e, no final da tabela, o número total das pastas/ficheiros no diretório presente.

Figura 5.8: Exemplo de uma página fonte de acesso à *pen*.

De acordo com a estrutura do histórico de ciclos definida anteriormente e representada na Figura 5.3, para a navegação foram desenvolvidas duas representações em forma

de lista do conteúdo da pasta "*Cycle History*", que representam os dias de funcionamento do equipamento e o seu conteúdo (os ciclos realizados), respetivamente. Para a representação do dia utilizou-se a data e a quantidade de ciclos realizados no respetivo dia (Figura 5.9(a)), enquanto que para a representação dos ciclos na lista, utilizou-se o número e designação do ciclo, a data e a hora do início do ciclo e o indicador de validade do ciclo (Figura 5.9(b)).



(a) Lista de dias de funcionamento do esterilizador.

(b) Lista de ciclos.

Figura 5.9: Representação das pastas do histórico de ciclos em *portrait*.

Para o caso da lista de dias de funcionamento, a informação é retirada diretamente das páginas HTML, ou seja, a data que é extraída da hiperligação e que corresponde ao nome da pasta, e a quantidade de ciclos que é obtida acedendo ao interior da pasta. O esquema para a obtenção dos dados necessários para a pasta "*Cycle History*" está representado na Figura 5.10. O *parsing* da pasta do histórico de ciclos começa pela a obtenção do seu URL, formada a partir do endereço IP do equipamento e da porção fixa do diretório do "*Cycle History*": $'http://<endereçoIP>/disk/usb1/CycleHistory'$

Logo de seguida, acede-se à página *Web* a partir do URL e obtém-se o documento HTML representativo, sendo repartido o documento de forma a retirar somente os elementos com a hiperligação para as pastas existentes. Caso não sejam identificados elementos com hiperligação, significa que o histórico de ciclos está vazio, ou seja, o esterilizador ainda não operou para gerar ciclos. Caso contrário, é feito *parsing* de todos os elementos encontrados, um de cada vez, começando pela obtenção do nome da pasta a partir da hiperligação (corresponde ao último elemento do diretório), por exemplo: $'http://<endereçoIP>/disk/usb1/CycleHistory/03072018'$, do exemplo do diagrama de organização da Figura 5.3. De seguida, acede-se ao interior da pasta (03072018, do exemplo anterior), recorrendo à hiperligação, e obtém-se o elemento que representa a quantidade, a partir do documento HTML. Obtidos todos os dados necessários, o URL da

pasta, o nome da pasta em forma de data e a quantidade de ciclos presentes no interior, são agrupados e guardados num objeto próprio. O URL é necessário para permitir o acesso à pasta, quando o utilizador selecionar o dia ao qual pretender aceder. O *parsing* termina quando todos os elementos forem analisados e os seus objetos guardados numa lista, para a posterior representação na aplicação (Figura 5.9(a)), com data do dia no lado esquerdo e a quantidade de ciclos presentes na pasta no lado direito.

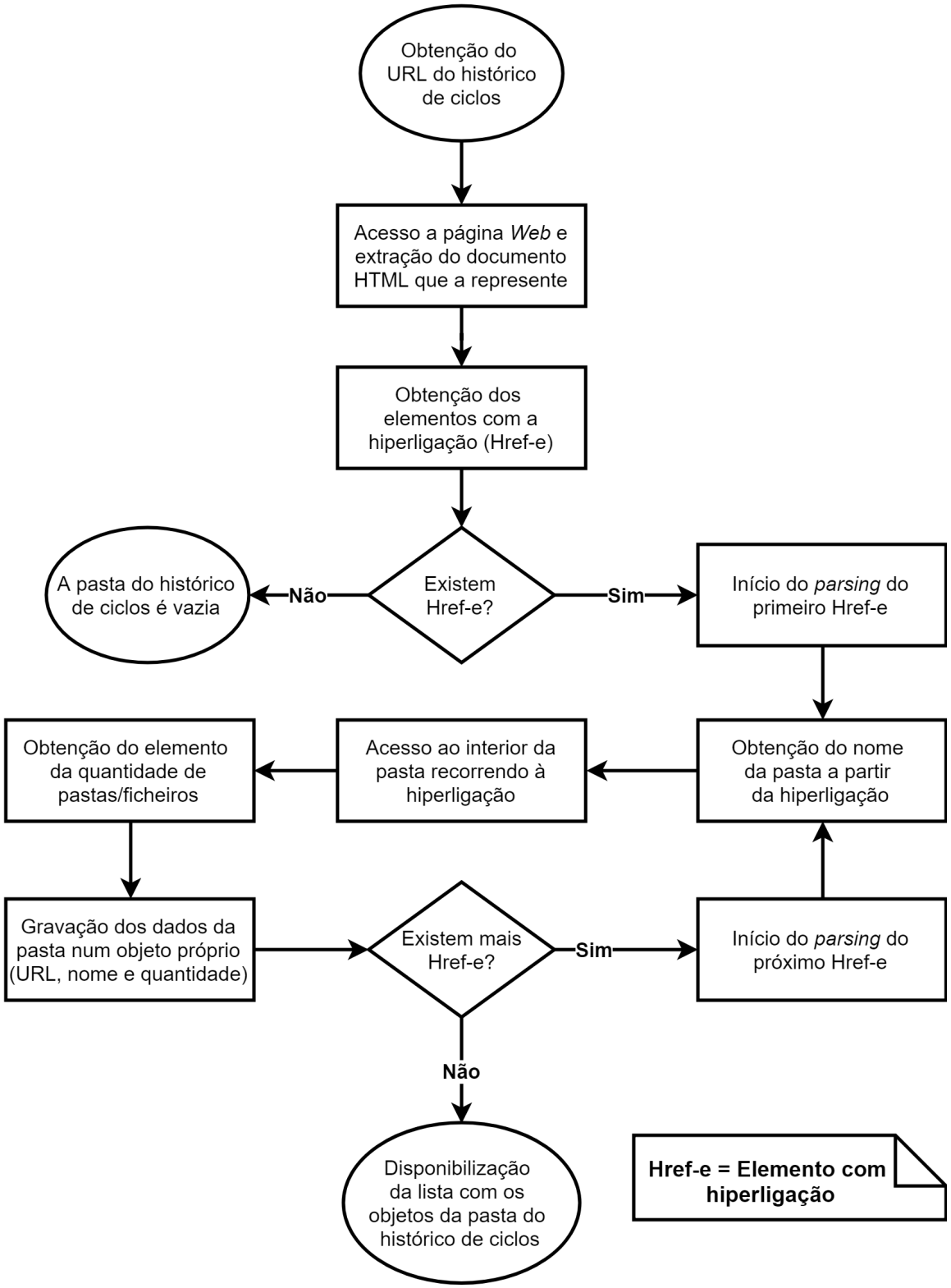


Figura 5.10: Esquema do *parsing* da pasta do histórico de ciclos.

Para o caso da lista dos ciclos, praticamente toda a informação é retirada a partir dos ficheiros CSV (Figura 5.11).

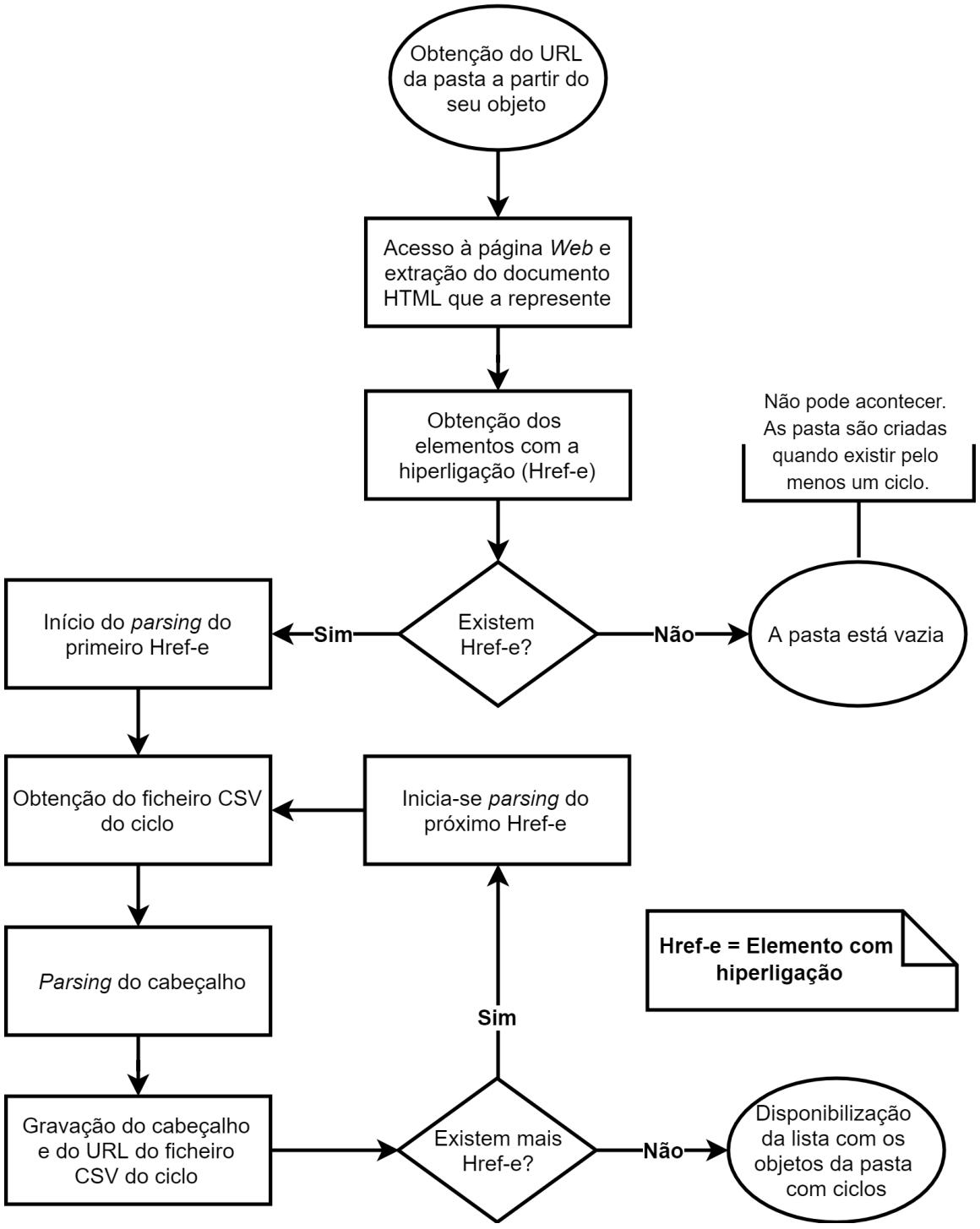


Figura 5.11: Esquema do *parsing* da pasta com os ciclos.

O *parsing* começa quando o utilizador seleciona o dia que pretende visualizar, sendo retirado o URL do objeto, que representa este dia, e é acedida a página *Web* da pasta. De seguida, é feita a repartição do documento de forma a retirar somente os elementos com a hiperligação para as pastas existentes. Caso não sejam encontrados elementos, significa que a pasta está vazia devido à ocorrência de alguma falha na criação do histórico, isto

porque as pastas que representam o dia de funcionamento do esterilizador são somente criadas após o primeiro ciclo do dia ter terminado, ou seja, terá que existir pelo menos um ciclo dentro da pasta para que esta exista. No decorrer normal, inicia-se o *parsing* de todos os elementos com hiperligação, um de cada vez, o que consiste na obtenção do ficheiro CSV do ciclo, consequente *parsing* e gravação do seu cabeçalho em conjunto com o URL (para posterior acesso ao ciclo) num objeto dedicado a esse propósito. O *parsing* termina quando todos os elementos forem analisados e os seus objetos guardados numa lista, para posterior representação na aplicação (Figura 5.9(b)). Na representação do ciclo na lista optou-se por dividir em quatro regiões:

- O número do ciclo, do lado esquerdo;
- O indicador de validade do ciclo, do lado direito;
- A designação do ciclo, no centro e em cima;
- A data e hora do início do ciclo, no centro e em baixo.

Quando o item da lista de ciclos é pressionado, é obtido o URL do ciclo a partir do seu objeto e descarregado o ficheiro CSV do ciclo correspondente a esse item. Na primeira instância, para a verificação do correto funcionamento, a representação do ciclo consistia na apresentação direta, sem qualquer processamento, do ficheiro CSV e posterior representação do seu objeto após o *parsing*.

5.1.4 Representação do Ciclo

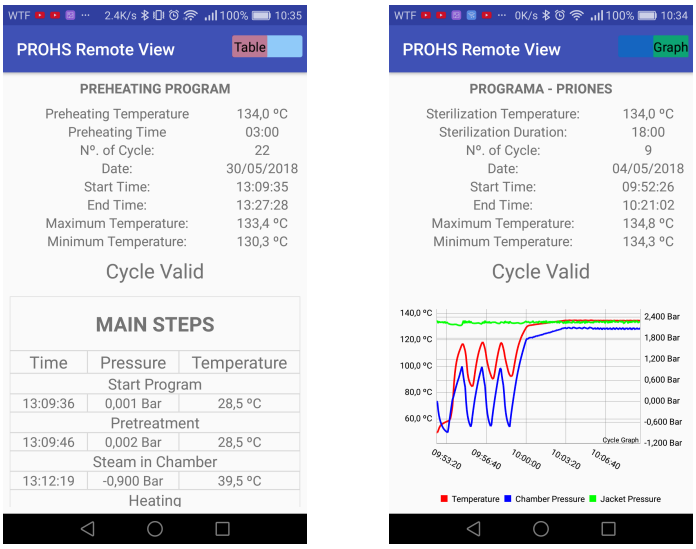
A partir do momento em que a aquisição e o *parsing* do ficheiro CSV foram concluídos, o próximo passo consistiu na representação do ciclo (Figura 5.12) selecionado na lista. A representação é iniciada pelo cabeçalho com o intuito de verificar o espaço necessário para a representação dos dados de maior relevância (Tabela 5.1). Optou-se por utilizar uma estrutura semelhante àquela nas fitas impressas após o ciclo, começando com a designação do respetivo ciclo (título), seguido pelos restantes dados (onde o tipo do dado está do lado esquerdo e o correspondente dado do lado direito), terminando com o indicador de validade do ciclo. Devido à existência de diferentes tipos de ciclos, a representação do cabeçalho possui algumas particularidades, em que "dado do ciclo 1" e "dado do ciclo 2" (Tabela 5.1 na página 54) representam diferentes informações, como tal, é necessário representar estes ciclos de forma coerente. Para superar este desafio, foi adicionado um indicador do tipo de ciclo ao ficheiro CSV, a seguir ao nome do ciclo (Figura 5.4(a) na página 55), com o intuito de simplificar o processamento durante a representação do ciclo. Isto porque, caso fosse utilizado o nome do ciclo para distinguir os tipos de ciclo, seria necessário implementar muitas verificações, devido à existência de vários ciclos do mesmo tipo e porque o nome do ciclo, guardado no ficheiro CSV, depende da língua selecionada

na consola, o que requereria verificações para todas as línguas existentes na consola. O indicador do tipo de ciclo é um valor numérico que pode tomar três valores distintos:

- Valor '0' - ciclos de esterilização e teste de Bowie&Dick. Neste caso, o "dado do ciclo 1" representa a temperatura de esterilização e o "dado do ciclo 2" representa o tempo de esterilização, tal como representado na Figura 5.12(b);
- Valor '1'- teste de fugas. Neste caso, o "dado do ciclo 1" representa o tempo de estabilização e o "dado do ciclo 2" representa a duração do teste. Os dados da temperatura máxima e mínima são ignorados e não representados, pois durante o teste de fugas não há esterilização;
- Valor '2'- pré-aquecimento. Neste caso, o "dado do ciclo 1" representa a temperatura de aquecimento e o "dado do ciclo 2" representa o tempo de aquecimento, tal como representado na Figura 5.12(a).

Inicialmente pensou-se em representar os dados do cabeçalho de forma separada dos dados das fases do ciclo e o gráfico, adicionando, no final, dois botões para aceder aos mesmos. Porem após a representação do cabeçalho foi notório a existência de muito espaço em branco e que é possível condensar a informação para ocupar somente metade do ecrã e desta forma libertar a outra metade para representar os dados das fases e/ou gráfico.

Primeiro foi desenvolvida a representação do ciclo em forma de gráfico, devido aos dados das fases terem sido adicionados posteriormente no decorrer do desenvolvimento do histórico de ciclos. A tarefa inicial consistia na pesquisa de bibliotecas que facilitassem a criação do gráfico, com um aspeto agradável e que suportasse vários conjuntos de dados ao mesmo tempo, para evitar a criação de gráficos separados para a temperatura e a pressão. As bibliotecas internas da Google não respeitavam os requisitos mencionados, como tal foi encontrada uma biblioteca externa, MPAndroidChart, desenvolvida por Philipp Jahoda, disponível num repositório no GitHub (Jahoda, 2018). A configuração do gráfico, para além da atribuição dos dados, consistia na adição de dois eixos verticais, um para representar a escala da temperatura (lado esquerdo) e o segundo para representar a escala de pressão (lado direito). Foram criados métodos para as escalas serem representadas com as unidades corretas, ºC no caso da temperatura e Bar no caso da pressão. O eixo horizontal representa o tempo e para que a sua legenda não se sobreponha, configurou-se de forma a ser representada com um ângulo de 30º. Também foi adicionada legenda, por baixo do gráfico, para cada conjunto de dados representados no mesmo: a linha a vermelho e a azul são a temperatura e a pressão na câmara, respetivamente, e a linha a verde é a pressão na camisa. O gráfico permite realizar o zoom nas regiões de interesse com o ajuste automático das escalas (visível no exemplo da Figura 5.12(b), onde o gráfico está focado na região dos pulsados e numa porção inicial da fase de esterilização).



(a) Representação do ciclo com a tabela com as fases do ciclo.

(b) Representação do ciclo com o gráfico do ciclo.

Figura 5.12: Exemplos da representação do ciclo em *portrait*.

Após a representação do gráfico do ciclo, foi desenvolvida a representação dos dados das fases do ciclo, tendo-se optado por representá-los na forma de uma tabela e, pelo facto das fases e a sequência das mesmas diferirem entre ciclos, foi necessário desenvolver um algoritmo para a criação automática da tabela, recorrendo às listas criadas no *parsing* (Figura 5.6 na página 58 da Secção 5.1.2). O título da tabela e as designações das três colunas (tempo, pressão e temperatura) são fixas, enquanto que o resto da tabela é criada progressivamente, de modo a que as designações das fases ocupem as três colunas, de forma centrada, seguidas pelos respetivos dados. O algoritmo para a criação da tabela está representado na Figura 5.13, que começa pela extração das listas com a informação do objeto que representa o ciclo, seguido pela determinação da quantidade das fases, presentes na lista com os identificadores das fases. O próximo passo do algoritmo consiste no processamento de todos os indicadores presentes na lista (um de cada vez), por meio de um *loop*. Dentro deste *loop* existe um segundo *loop* responsável pelo processamento das amostras de dados para cada fase. Utilizando o exemplo representado no diagrama de correspondência na Figura 5.6 na página 58, na primeira iteração do primeiro *loop*, é lido o primeiro elemento da lista com identificadores. Este identificador é transformado no nome completo da fase, recorrendo à tabela de correspondência (Tabela 5.2 na página 56), que é adicionado à tabela com devida configuração gráfica. Logo de seguida, é também retirado o primeiro elemento da lista de quantidades, com o propósito de saber quantas vezes será corrido o segundo *loop*, onde são lidos e processados os dados e adicionados na forma de uma linha à tabela. Para esta tarefa foi criada uma variável complementar, designada de apontador, para apontar o elemento das listas de dados que deverá ser lido. Este apontador é incrementado após cada adição de linha de dados, sendo que após a adição de todas as linhas de dados correspondentes à fase, o *loop* recomeça e é feita a próxima

iteração, ou seja, para este exemplo é lido o segundo elemento da lista com identificadores e assim sucessivamente.

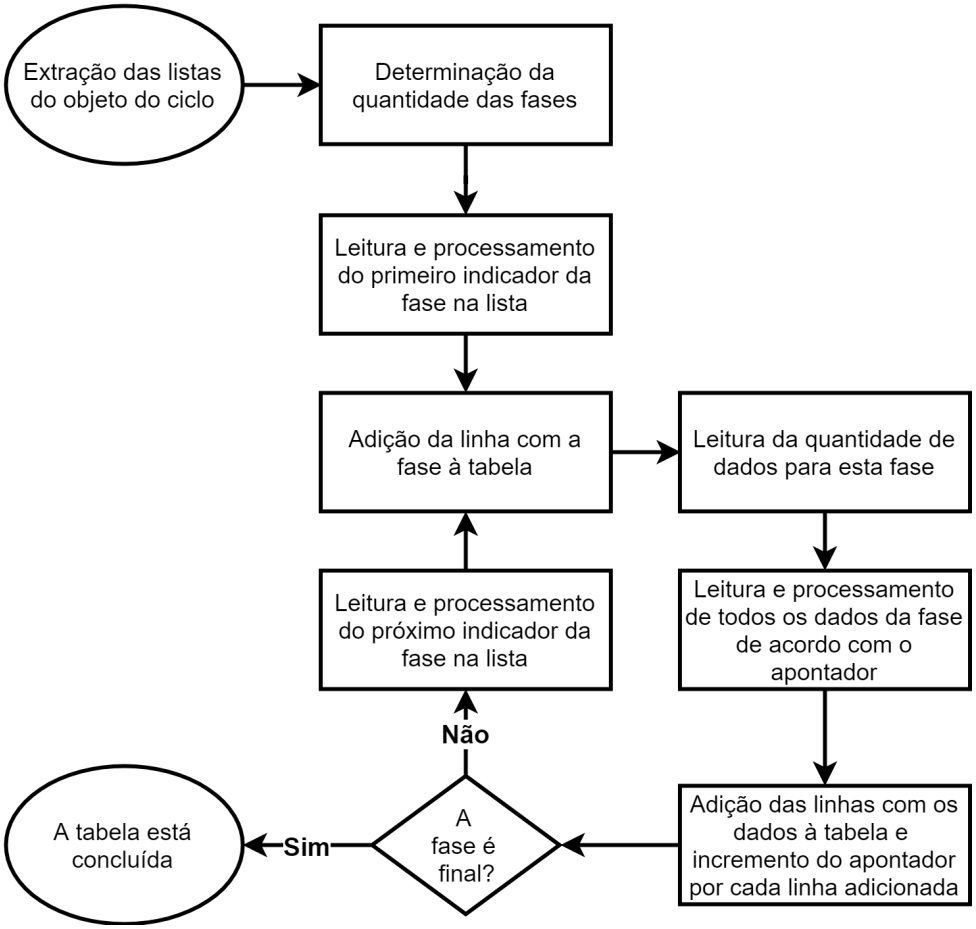


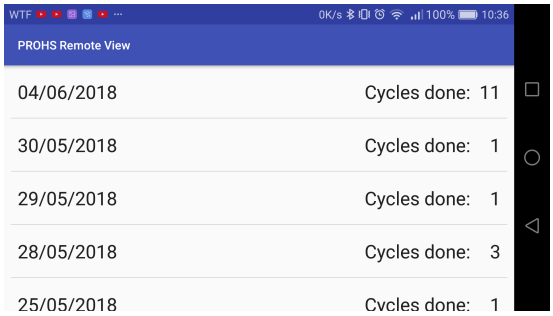
Figura 5.13: Algoritmo para a criação da tabela com os dados das fases do ciclo.

O tamanho da tabela é variável e em todos os ciclos os dados das fases ocupam mais de metade do ecrã, com exceção da ocorrência de uma falha prematura durante o ciclo, tornando a tabela mais curta. Como tal, foi adicionado um *scroll* à tabela, de modo a deslizar a tabela para cima e para baixo para permitir visualizar todas as fases.

No *smartphone* com um ecrã relativamente pequeno, não é possível representar toda a informação do ciclo ao mesmo tempo, como tal, foi necessário criar um mecanismo, claro e intuitivo, para intercalar entre a representação do ciclo no formato de uma tabela e de um gráfico. Para este efeito, decidiu-se criar um *switch* personalizado e colocá-lo no *Toolbar* da aplicação, de forma a não ocupar o espaço da representação do ciclo, o qual é escasso. Na primeira instância, o *switch*, com o intuito de aprendizagem, possuía diferentes cores para cada parte e estado do mesmo, com uma legenda sobre o estado selecionado, para indicar se foi selecionada a tabela ou o gráfico (canto superior direito da Figura 5.12(a) e Figura 5.12(b)), sendo que posteriormente foi ajustado o design e forma do *switch*, que será abordado na secção de design (Secção 5.2). Para alternar entre gráfico e tabela, é necessário deslizar o *switch* na direção desejada, o que substituirá a tabela pelo gráfico e vice-versa.

5.1.5 Funcionalidades Adicionais

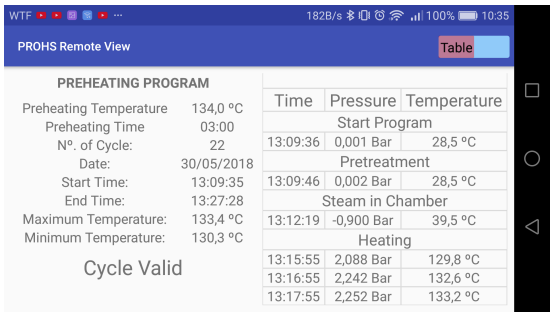
No histórico de ciclos, para além das funcionalidades principais, o objetivo consistiu na criação de um *layout* flexível, de modo a ocupar o espaço de forma homogénea, capaz de adaptar-se ao tamanho de ecrã, com vista a adicionar, futuramente, um suporte para dispositivos com outros tamanhos do ecrã e densidade de pixeis. Por sua vez, este tipo de *layout*, facilitou a implementação de um suporte de rotação do ecrã, ajustando-se de forma correta ao espaço horizontal acrescido[1], para o caso das listas na navegação pelo histórico de ciclos (Figura 5.14). No caso da representação do ciclo, foi necessário reorganizar a disposição da informação, ou seja, distribui-la no espaço horizontal, colocando os dados do cabeçalho do lado esquerdo e a representação da tabela/gráfico do lado direito (Figura 5.15).



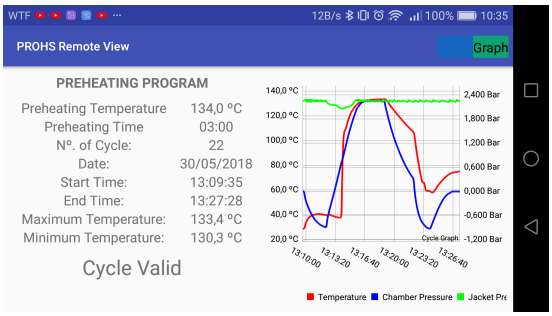
(a) Lista de dias de funcionamento do esterilizador.

(b) Lista de ciclos.

Figura 5.14: Representação das pastas do histórico de ciclos em *landscape*.



(a) Representação do ciclo com a tabela das fases do ciclo.



(b) Representação do ciclo com o gráfico do ciclo.

Figura 5.15: Exemplos da representação do ciclo em *landscape*.

Na representação do ciclo, em casos de ciclos inválidos, foi implementado um mecanismo para apresentar informação detalhada sobre o erro que causou a invalidade do ciclo. Como o espaço no ecrã é escasso e não existe possibilidade de introduzir novos dados, optou-se por tornar o indicador de validade num elemento interativo e criar um *pop-up* para apresentar a informação detalhada sobre o erro (Figura 5.16), que aparece

[1] O mesmo princípio foi implementado em toda a aplicação.

sobre a atividade quando o indicador é pressionado. O *pop-up* é composto por um título, que indica o número de erro, um *icon* do erro em conjunto com o nome do erro, uma descrição detalhada do erro e por fim um botão 'OK' para fechar o *pop-up*.

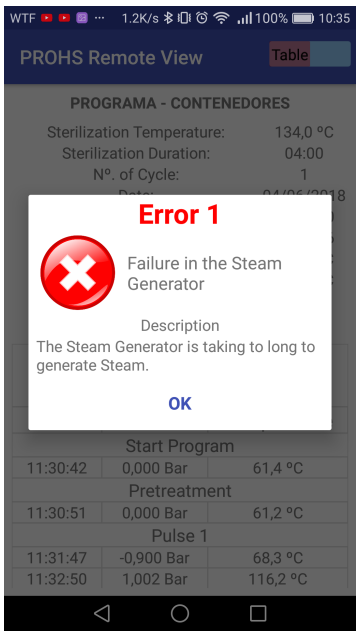


Figura 5.16: Exemplo do *pop-up* com a informação detalhada do erro.

Por fim, adicionou-se um *swipe-to-refresh* às listas na navegação pelo histórico, para os casos em que as listas não sejam carregadas corretamente (não são apresentadas) ou quando se está a aguardar pelo ciclo mais recente, recarregando a lista com um deslizar de cima para baixo. Ao realizar o *swipe-to-refresh* aparecerá um símbolo de atualização, que permanecerá (a rodar) até que a lista seja recarregada ou ocorra um *timed-out*, em casos de problemas na comunicação (Figura 5.17).

Figura 5.17: Exemplo do *swipe-to-refresh* na lista com ciclos.

5.2 Navegação e *Design* Gráfico da Aplicação

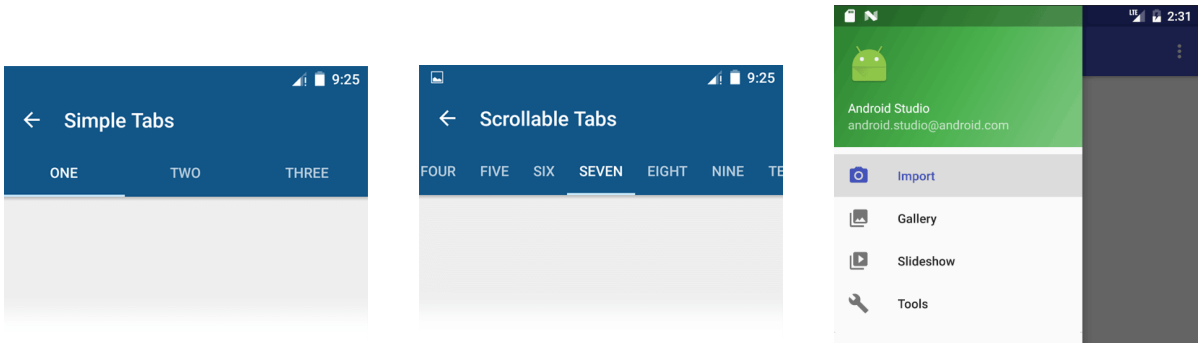
Antes de avançar para o desenvolvimento da componente da monitorização em tempo real, foi necessário pensar na forma de navegação pela aplicação e em como ter acesso a todas as componentes desenvolvidas para o *debugging*, e definir a estrutura geral da aplicação em conjunto com o seu *design* gráfico. Para tal, foram efetuadas pesquisas sobre formas de navegação existentes e análise das aplicações no Play Store da Google para definir a forma mais adequada de navegação para esta aplicação. Foram encontradas quatro formas diferentes de implementar a navegação:

1. Atividade com botões para aceder a todas as funcionalidades - é a forma mais simples de conseguir a navegação. No entanto, possui as suas desvantagens, como a necessidade de voltar para esta atividade sempre que se queira mudar entre formas de monitorização;

2. *Swipe tabs* (Figura 5.18(a)) - consistem em separadores fixos na parte superior do ecrã, junto ao *Toolbar*, onde é possível trocar de separador ao pressionar no separador desejado ou ao deslizar com o dedo para um dos lados. Os *swipe tabs* são uma forma plausível e intuitiva para a navegação em casos de poucos separadores, de modo a que sejam todos representados, sem sobreposição. A grande desvantagem, para esta aplicação, é o facto de ocupar uma porção considerável do espaço no ecrã, impondo assim limitações de espaço para o resto da aplicação;

3. *Scrollable tabs* (Figura 5.18(b)) - semelhante aos *swipe tabs*, aplicável em aplicações com muitos separadores, em que os separadores que não tiveram espaço no ecrã ficam escondidos, reaparecendo de um lado à medida que é feito o *scroll*, enquanto desaparecem do lado oposto, os separadores inicialmente visíveis. Esta solução é menos intuitiva para o utilizador do que a solução anterior e apresenta a mesma desvantagem;

4. *Navigation drawer* (Figura 5.18(c)) - é uma forma universal para uma navegação principal dentro da aplicação e acesso a todas funcionalidades da mesma, utilizada na grande maioria das aplicações modernas, incluindo o próprio Play Store. Consiste num painel normalmente escondido no bordo esquerdo do ecrã, quando não está a ser utilizado. Para que se torne visível, é necessário deslizar o dedo a partir do bordo esquerdo para o interior do ecrã ou pressionar no *icon* correspondente na *Toolbar*. O *navigation drawer* permite grande customização da sua aparência e conteúdo. Como desvantagem, pode ser considerada a complexidade acrescida para a sua implementação.



(a) *Swipe tabs* (retirado de Tamada (2015)). (b) *Scrollable tabs* (retirado de Tamada (2015)). (c) *Navigation drawer* (*Template* no Android Studio).

Figura 5.18: Exemplos de diferentes formas de navegação.

Após esta pesquisa, foi decidido avançar com a implementação de um *navigation drawer*, pelo facto desta solução ser a única que não apresentava desvantagens de usabilidade para a aplicação, tornando-a, pelo contrário, mais robusta, intuitiva e apelativa.

Em conjunto com o *navigation drawer*, foi definido o tema geral da aplicação, que consistiu na alteração das principais cores da aplicação, definidas por omissão, para as cores da PROHS: azul escuro, azul claro e branco. Na Figura 5.19 está representada a implementação inicial do *navigation drawer*, que consistia num cabeçalho com a imagem e slogan da PROHS e um único menu com os possíveis destinos. Os itens do menu são compostos por um *icon* representativo da funcionalidade, seguidos pelo seu nome, com uma cor azul escura e um fundo branco. O item selecionado fica destacado pela mudança da cor para azul claro e com um sombreado cinzento. Para além desta forma de destacar o item, foi colocado um subtítulo no *Toolbar*, por baixo do título da aplicação, para indicar a funcionalidade selecionada.

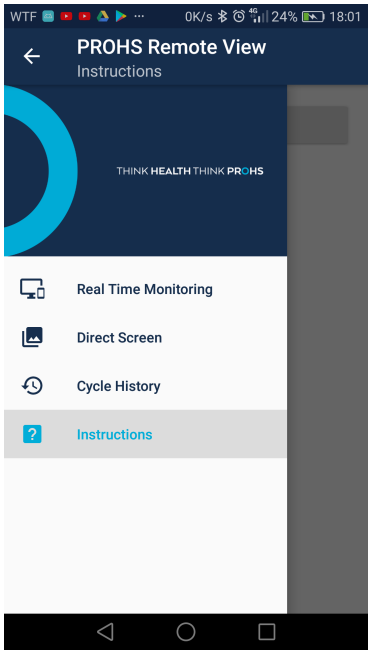


Figura 5.19: Implementação inicial do *navigation drawer*.

Nesta fase de implementação do *navigation drawer*, ficou decidido que a atividade principal (que será a primeira a ser apresentada ao aceder à aplicação) é a atividade com as instruções de utilização. Para navegar para outras atividades, o utilizador deve abrir o *navigation drawer* e selecionar, no menu, o item da funcionalidade a qual pretende aceder.

A partir do momento em que o *navigation drawer* ficou implementado e adicionado a todas as atividades, o próximo passo consistiu na implementação das transições entre as atividades, de modo a serem suaves, agradáveis e percetíveis. Por omissão, as transições eram bruscas, em que a atividade corrente, em conjunto com a *Toolbar*, deslizavam para o lado esquerdo, substituídas pela atividade de destino, o que provocava sensação de falta de integridade na aplicação. Para a transição entre atividades foi selecionado o efeito *fade*, onde a atividade que está a ser substituída torna-se gradualmente transparente até desaparecer e a atividade de destino torna-se gradualmente visível. Para atingir o efeito de integridade na aplicação, as transições foram devidamente temporizadas e os elementos comuns a ambas as atividades (como barra de notificações, *Toolbar* e o título) excluídos das mesmas. A sequência da transição é a seguinte:

1. Seleção do destino;
2. O *navigation drawer* fecha;
3. Desaparecimento da atividade atual e aparecimento da atividade seguinte, com duração de 500 ms.

O comportamento das atividades foi configurado de modo a que somente as atividades da funcionalidade selecionada, possam permanecer no *backstack*[2], à exceção das instruções (atividade principal, que permanece sempre como último elemento no *backstack*). O *backstack* é do género de um histórico de atividades: quando o utilizador abre uma nova atividade, essa atividade é adicionada ao *backstack*. O utilizador ao pressionar o botão para retroceder, termina a atividade atual e ela é retirada do *backstack*, e de seguida é apresentada a atividade anterior. Quando o *backstack* possuir somente uma atividade, ao retroceder a última atividade esta é terminada e a aplicação é fechada.

Na Figura 5.20 está representado um exemplo do funcionamento do *backstack* implementado para a aplicação. Ao contrário do comportamento habitual do *backstack*, após a navegação entre as funcionalidades (X e Y) a atividade da funcionalidade da qual foi realizada a navegação é terminada e retirada do *backstack*, ou seja, ao retroceder aparecerá sempre a atividade principal. Existe sempre possibilidade de retroceder ou de navegar de uma funcionalidade para a atividade principal, no caso da navegação, em vez de ser criada uma nova instância da atividade principal, é retomada a atividade principal existente no *backstack*, e quando se retrocede a partir da atividade principal a aplicação é fechada.

[2]Para informações mais detalhadas acerca do *backstack* pode ser consultado Android Developers (2018k).

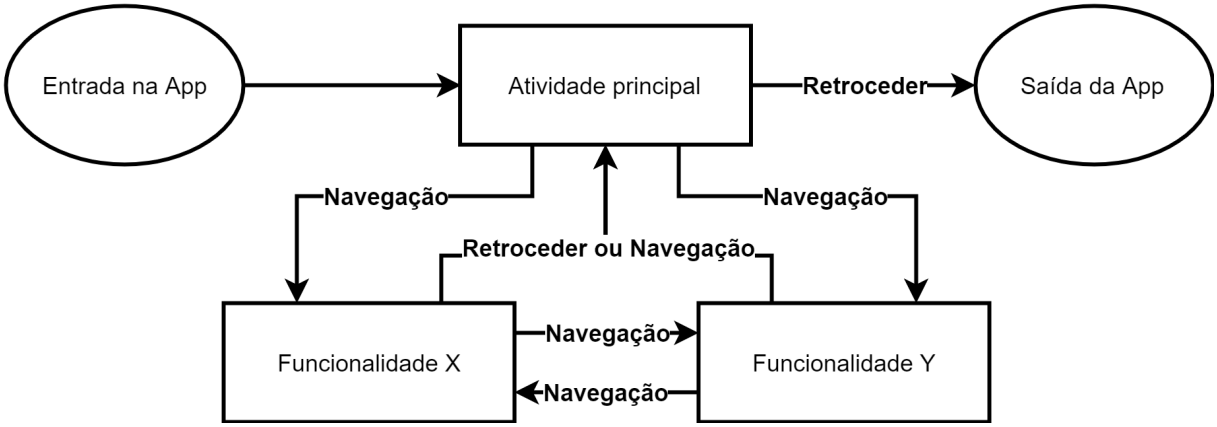


Figura 5.20: Esquema do comportamento das atividades na aplicação.

Dentro do histórico de ciclos, o *backstack* possui o seu comportamento habitual, ou seja, é possível retroceder da representação do ciclo para a lista dos ciclos e a seguir a lista dos dias de funcionamento, sem qualquer problema (Figura 5.21). No entanto, se retroceder a partir da lista dos dias de funcionamento, aparecerá a atividade com as instruções, ou se for feita uma navegação para outra funcionalidade qualquer, todas as atividades do histórico de ciclos serão terminadas e não haverá possibilidade de retroceder.

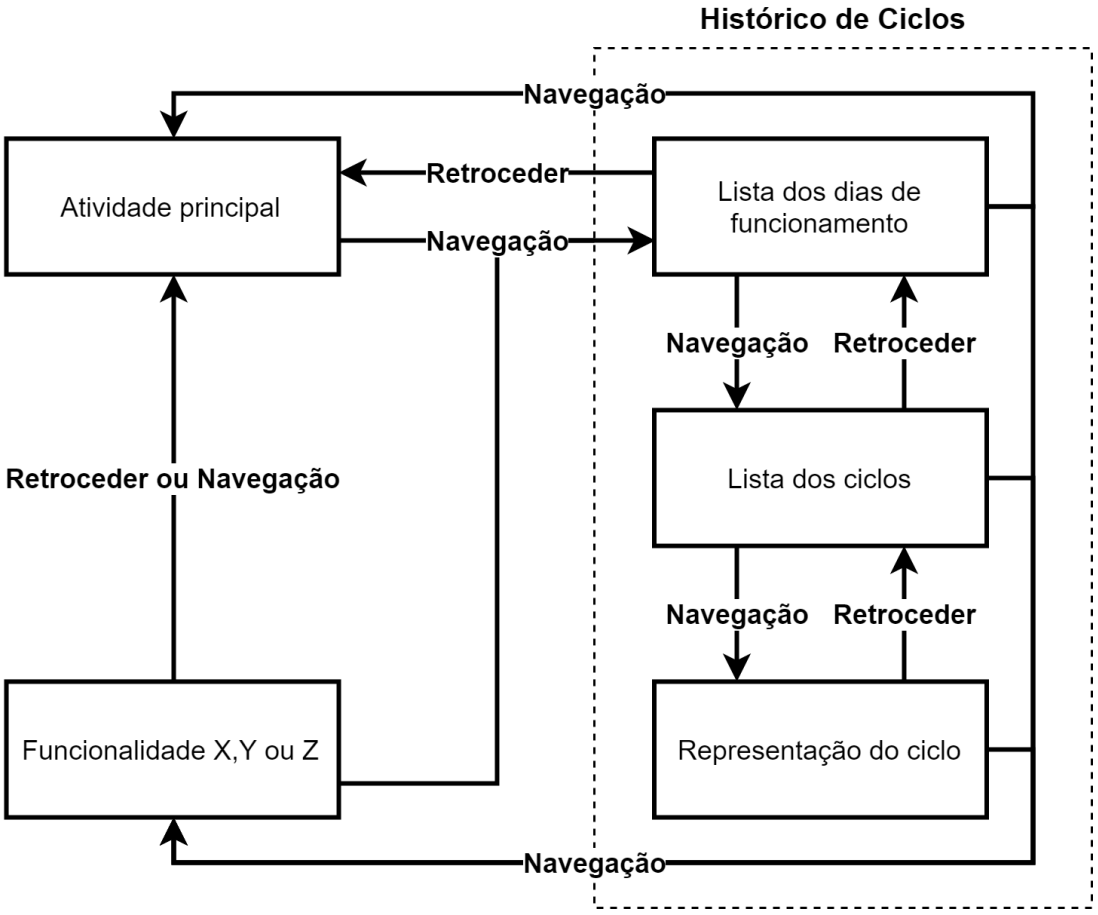
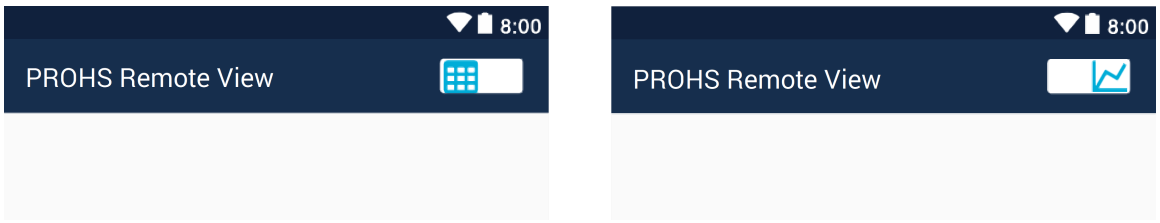


Figura 5.21: Esquema do comportamento das atividades na aplicação.

Para o histórico de ciclos foram implementadas transições particulares, mais especificamente para a navegação pelas listas. A transição de saída das listas é uma combinação de efeitos de explosão e de *fade*, em que após a seleção do elemento da lista, este elemento desaparece gradualmente (tornando-se invisível) e os restantes elementos da lista deslizam para fora do ecrã (para cima ou para baixo, dependendo da sua posição ao centro do ecrã). Depois da transição de saída terminar, aparecerá com efeito *fade* a atividade seguinte (lista dos ciclos ou representação do ciclo).

Com a definição do tema para a aplicação, foi possível definir o *design* final do *switch* (para intercalar entre a tabela e o gráfico) na representação do ciclo. O novo *design* (Figura 5.22) consistiu na alteração das cores do *switch* para as cores do tema e substituição das legendas por *icons* representativos, uma tabela e um gráfico, respetivamente.



(a) Aparência do *switch* (tabela). (b) Aparência do *switch* (gráfico).

Figura 5.22: Novo *design* do *switch* para o histórico de ciclos.

Na representação do ciclo, para além das alterações do *switch*, foram adicionadas transições suaves para a substituição da tabela pelo gráfico e vice-versa. Para estas transições foi utilizado o efeito *slide*, que consiste no deslizar do elemento para fora do ecrã numa das direções. As transições foram configuradas e temporizadas de modo a que estivesse de acordo com o movimento do *switch*, ou seja, quando é feita a troca de tabela para o gráfico, a tabela desliza para fora, na direção do bordo direito do ecrã, e o gráfico aparece deslizando a partir do bordo esquerdo do ecrã. O mesmo acontece para o caso da troca de gráfico para a tabela, só que no sentido contrário.

5.3 Monitorização em Tempo Real

O primeiro passo para o desenvolvimento da funcionalidade responsável pela monitorização em tempo real, do processo de esterilização, consistiu na transferência e adaptação do código, desenvolvido para os testes com o Modbus durante a preparação da segunda proposta de solução (Secção 4.2 na página 47). Em conjunto com este passo desenvolveu-se o *layout* gráfico (Figura 5.23) para esta funcionalidade, tendo-se optado por aproximar o *design* àquele que está na HMI (Figura 5.24), com o objetivo de o tornar familiar para o utilizador.

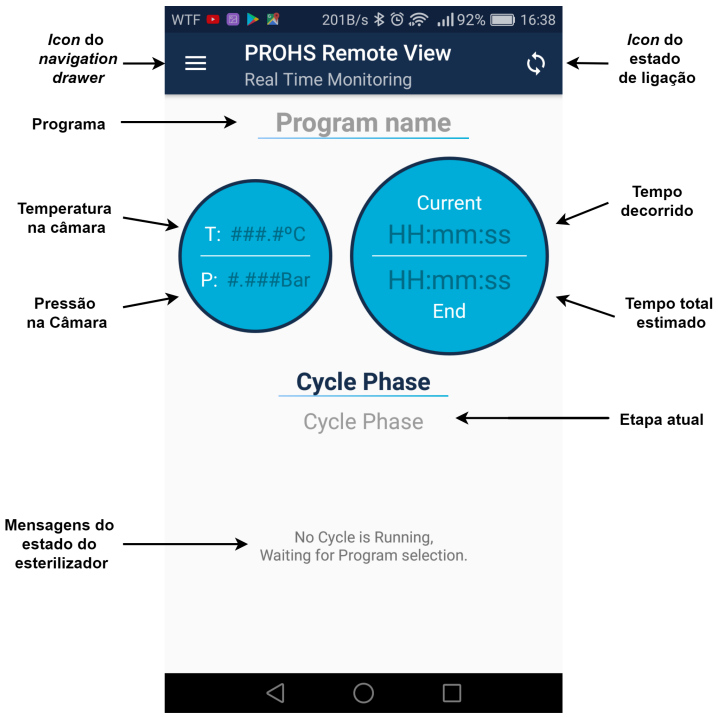


Figura 5.23: *Layout* gráfico desenvolvido para a monitorização em tempo real.

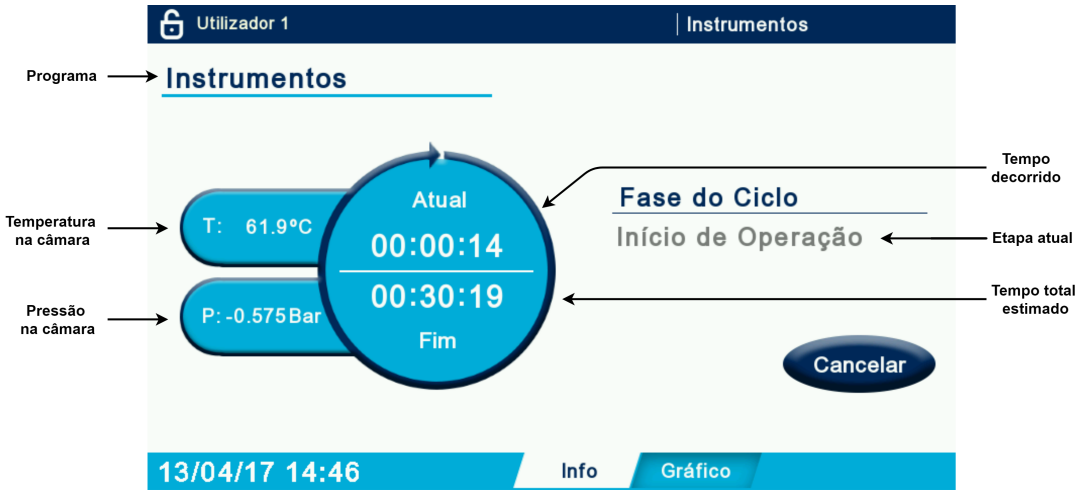


Figura 5.24: Interface de utilizador da HMI durante execução de um ciclo.

Para além das partes em comum nos dois *designs* (indicador do programa selecionado, da temperatura e pressão na câmara, do tempo decorrido e tempo total estimado e da fase), considerou-se importante colocar duas componentes adicionais. Um campo com as mensagens relevantes acerca do estado do esterilizador, isto é, acerca de todos os acontecimentos desde a seleção do programa até ao seu término e mensagens relativas à ligação. Inicialmente este campo foi colocado para um *debugging* desta componente, no entanto, foi decidido mantê-lo, pois proporciona uma mais valia para experiência do utilizador, apresentando informações úteis tais como a indicação:

- De que de momento não está a correr nenhum ciclo (Figura 5.23);
- Que o esterilizador está a aguardar a pressão na camisa;

- Que estão a ser feitas preparações para iniciar o ciclo;
- Que o ciclo está a correr;
- Do término do ciclo;
- Da ocorrência de um erro e que está-se a aguardar pelo término do ciclo de segurança;
- Que foi perdida a ligação e as possíveis causas (Figura 5.26);
- Que não foi possível estabelecer a ligação e as possíveis causas (Figura 5.25).

Este campo está situado na parte de baixo do ecrã, ocupando o espaço que restou após a interface gráfica criada para representação do ciclo. Para além disto, foi colocado um *icon* para indicar o estado de ligação, que também serve de botão para reestabelecer a ligação. Este *icon* possui três estados:

 - ligação estabelecida e existe troca de informação;

 - ligação foi perdida e será reestabelecida automaticamente, caso os problemas existentes forem eliminados (Figura 5.26);

 - ligação não foi estabelecida. Para reestabelecer a ligação o *icon* deve ser pressionado (Figura 5.25).

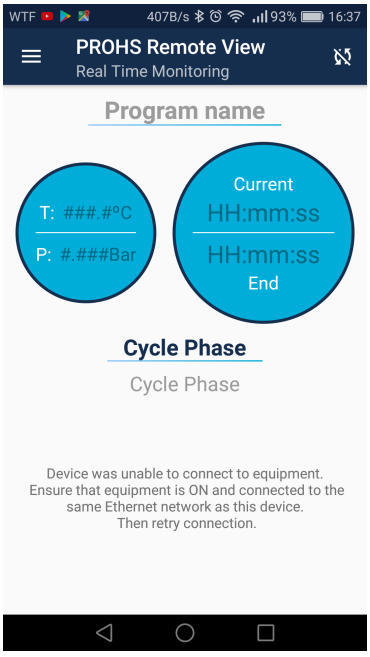


Figura 5.25: Captura de ecrã da monitorização em tempo real durante a ocorrência de problemas no estabelecimento da ligação inicial.

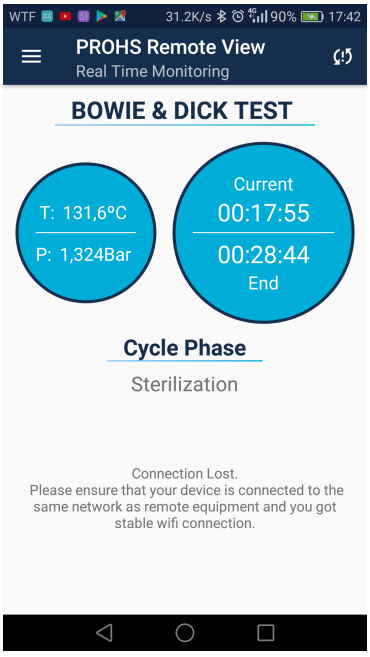


Figura 5.26: Captura de ecrã da monitorização em tempo real durante a perda da ligação.

O segundo passo consistiu na disponibilização dos dados por parte do esterilizador, começando com uma análise sobre a localização, no autómato, de todos os dados necessários para a monitorização e a localização de endereços de memória livres na HMI. De seguida foi criada uma macro para fazer a transferência dos dados do autómato para a HMI e no autómato foi criada uma rotina para controlar a execução periódica da macro durante a execução do ciclo. Após alguns testes com a frequência de execução da macro, deparou-se com um problema em que a atualização dos dados acontecia no mínimo a cada 2 segundos, de forma inconstante, consequência da execução lenta da macro. Como o *NB Designer* é um *software* gratuito e sem quaisquer ferramentas de *debugging*, teve-se que utilizar métodos não exatos para o despiste e correção do problema. Foram colocados dois *LED* na janela de execução do ciclo, um para indicar o *trigger* para execução da macro, proveniente do autómato, e um segundo para indicar a execução da macro. Todas as alterações feitas pela macro entram em vigor somente no final da execução. Desta forma, para estimar o tempo de execução da macro, é ligado o *LED* imediatamente antes da execução da macro e desligado no final da sua execução, e com o auxílio de um cronómetro de bolso, é medida a duração do *LED* ligado. Chegou-se à conclusão que o tempo de execução da macro ronda os 2-3 segundos e que caso o *trigger* ocorra durante a execução da macro, ele é ignorado e a consola fica a espera do próximo *trigger* para voltar a executar a macro. Caso este processo não esteja bem sincronizado, a atualização dos dados torna-se inconstante. Para diminuir o tempo de execução da macro, decidiu-se separar os dados constantes dos dados variáveis em duas macros, uma que será executada logo no início e outra periodicamente durante o ciclo, e para não sobrecarregar a consola com as conversões colocaram-se todos os valores, no autómato, no mesmo formato binário (BIN). Com estas alterações foi possível diminuir a execução da macro para, sensivelmente, 1 segundo. Este valor é considerado aceitável para a monitorização em tempo real, no entanto, devido à falta de mecanismos de sincronização, o problema do *trigger* em acontecer durante a execução da macro permanece, o que causa pequenos atrasos na atualização dos dados, percetíveis a olho desarmado.

Como com as macros não foi possível atingir os resultados desejáveis, efetuou-se uma análise profunda do manual operacional do *NB Designer*, com o objetivo de encontrar uma forma alternativa da transferência de dados do autómato para a HMI ou de melhorar o tempo de execução da macro. Com esta análise foi possível localizar uma pequena particularidade dos *timers*, nomeadamente para transferência de dados, ou seja, é possível configurar o *timer* para a transferência periódica de dados. Para uma transferência de dados eficiente (minimizar a quantidade de *timers*) foi necessário reorganizar os dados no autómato de forma a ficarem em endereços consecutivos. A Tabela 5.3 contém todos os dados para a monitorização em tempo real. Para a sua correta transferência recorreu-se a quatro *timers* e a três macros para as trocas do estado do esterilizador:

- Primeiro *timer* - transferência pontual dos dados constantes (nome do programa e tempo total estimado) no início da execução do ciclo;

- Segundo *timer* - transferência periódica (a cada 200ms) dos dados variáveis (tempo decorrido, número da fase, temperatura e pressão na câmara) durante a execução do ciclo;
- Terceiro *timer* - transferência pontual do número de erro, após a sua ocorrência;
- Quarto *timer* - transferência periódica (a cada 200ms) do valor do progresso da insuflação da pressão na camisa durante o pré-ciclo;
- Três macros - uma macro para cada *bit* de estado, para atualizá-los de acordo com o estado do esterilizador. O estado do esterilizador é indicado pelo autómato e é este que controla a execução destas macros na consola.

Tabela 5.3: Dados para a monitorização em tempo real.

Dados	**Tamanho**	**Descrição**	**Tipo**
Nome do programa	12 *words*	Carateres ASCII, com um máximo de 24 carateres	Constante
Tempo total estimado	3 *words*	1 *word* para cada unidade de tempo (hora, minuto, segundo)	Constante
Tempo decorrido	3 *words*	1 *word* para cada unidade de tempo (hora, minuto, segundo)	Variável
Número da fase	1 *word*	Valores entre 1 e 23 de acordo com as fases existentes (Tabela 5.2)	Variável
Temperatura na câmara	1 *word*	Valor da temperatura (0250 = 25.0ºC)	Variável
Pressão na câmara	1 *word*	Valor da pressão absoluta em $mBar$	Variável
Número de erro	1 *word*	Valores entre 0 e 35 (zero = ausência de erro)	Constante[a]
Progresso da insuflação da pressão à camisa	1 *word*	Valor numérico correspondente à percentagem de 0 à 100	Variável
Estado do esterilizador	3 *bits*	4 estados diferentes (000 - esterilizador parado; 100 - ciclo em execução; 010 - pré-ciclo[b]; 001 - ocorreu um erro[c])	Variável

[a] O número de erro é constante desde a sua ocorrência até ao término do ciclo de segurança.
[b] Pré-ciclo corresponde a todas as preparações para iniciar o ciclo, como insuflação da pressão à camisa.
[c] Execução do ciclo de segurança.

O terceiro passo consistiu no desenvolvimento de métodos e mecanismos para a correta aquisição, processamento e representação dos dados provenientes da HMI, via ligação Modbus. As aquisições dos dados foram separadas em quatro tipos, consoante a sua origem e tempo de ocorrência:

- Primeiro tipo - aquisição dos dados do ciclo (constantes e variáveis) durante a execução do ciclo;
- Segundo tipo - aquisição do número de erro, quando este acontece;

- Terceiro tipo - aquisição do valor do progresso da insuflação da pressão à camisa durante o pré-ciclo;
- Quarto tipo - aquisição dos *bits* do estado do esterilizador.

Para distinguir entre estes tipos de aquisição foi desenvolvida uma estrutura de mensagens e um método para o seu correto processamento. As mensagens são constituídas pelos dados adquiridos (também pode ser uma mensagem de erro) e acompanhados com o indicador do tipo de aquisição e de um indicador de sucesso da aquisição. Estas mensagens são construídas após a aquisição dos dados provenientes da HMI e enviadas para o método que as interpreta e encaminha para os métodos de processamento correto. Em casos em que a aquisição não foi bem sucedida (ligação perdida), são feitas tentativas de restabelecimento da ligação de forma automática e é indicado ao utilizador, no campo de mensagens, que foi perdida a ligação com o esterilizador (Figura 5.26). Na Figura 5.27 está representado o esquema simplificado do método mencionado anteriormente.

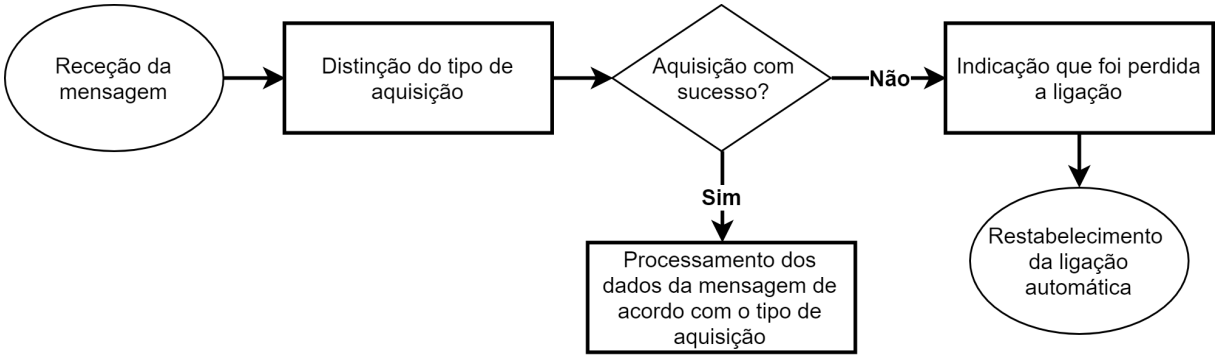


Figura 5.27: Esquema simplificado do funcionamento do método para a interpretação das mensagens.

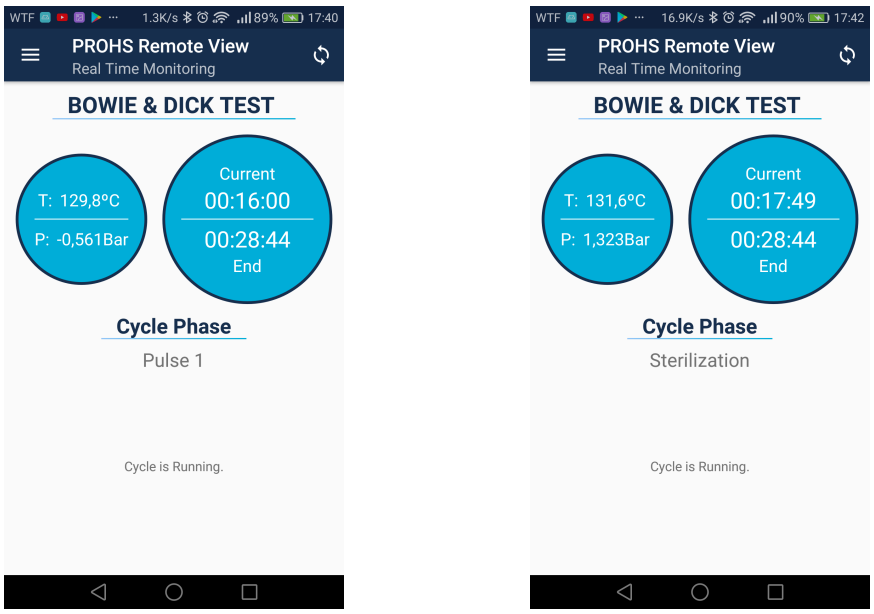
O processamento dos dados da Figura 5.27 difere consoante o tipo de aquisição:

- No caso de serem os dados adquiridos durante a execução do ciclo, esta etapa consiste na atualização dos valores na interface gráfica. Na Figura 5.28 estão representadas duas capturas de ecrã durante execução do ciclo em duas fases distintas;
- No caso em que ocorre um erro durante o ciclo, esta etapa consiste na indicação, no campo de mensagens, da ocorrência de um erro e que está a ser executado um ciclo de segurança, e na configuração do *pop-up* com os detalhes do erro[3], que surge de imediato após a ocorrência do erro e permanece até não ser dispensado. Também são colocados todos os campos relativos ao ciclo no seu estado por omissão;
- No caso de serem os dados do pré-ciclo, esta etapa consiste na atualização do valor do progresso de insuflação da pressão à camisa, numa barra de progresso (Figura 5.29). Esta barra é normalmente invisível e torna-se visível durante o pré-ciclo. No final, quando a barra se encontra toda preenchida, é indicado, no campo de mensagens,

[3]Este *pop-up* é igual àquele que aparece no histórico de ciclos quando é pressionado o indicador de validade nos ciclos inválidos (Figura 5.16).

que o ciclo está prestes a começar e a barra é dispensada;

- No caso de serem os *coils* do estado do esterilizador, esta etapa consiste no controlo da sequência de todo processo durante a monitorização em tempo real, sendo nesta parte que são programadas as restantes aquisições. Após a determinação do estado do esterilizador e consoante o mesmo, são inicializadas e/ou terminadas tarefas, em *background*, para as aquisições dos dados.



(a) Captura de ecrã durante a fase de pulsados.

(b) Captura de ecrã durante a fase de esterilização.

Figura 5.28: Monitorização em tempo real durante execução do ciclo em duas fases distintas.

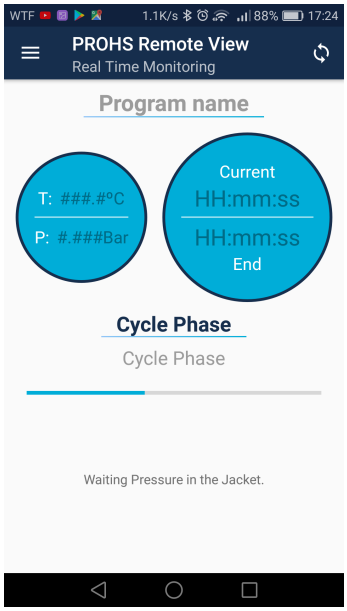


Figura 5.29: Captura de ecrã da monitorização em tempo real durante o pré-ciclo - insuflação da pressão à camisa.

A monitorização em tempo real começa com a inicialização da ligação Modbus e caso seja bem sucedida, é de seguida programada uma tarefa periódica, em *background*, para a leitura dos *coils* do estado do esterilizador. Esta tarefa é executada com periodicidade de um segundo, desde a sua inicialização até o utilizador sair da monitorização em tempo real ou a aplicação seja terminada. Caso não seja possível estabelecer a ligação, o utilizador deve verificar se está ligado à mesma rede que o esterilizador e que haja sinal e pressionar o *icon* de ligação para restabelecer a ligação (Figura 5.25). Os primeiros passos na monitorização em tempo real estão representados no esquema da Figura 5.30.

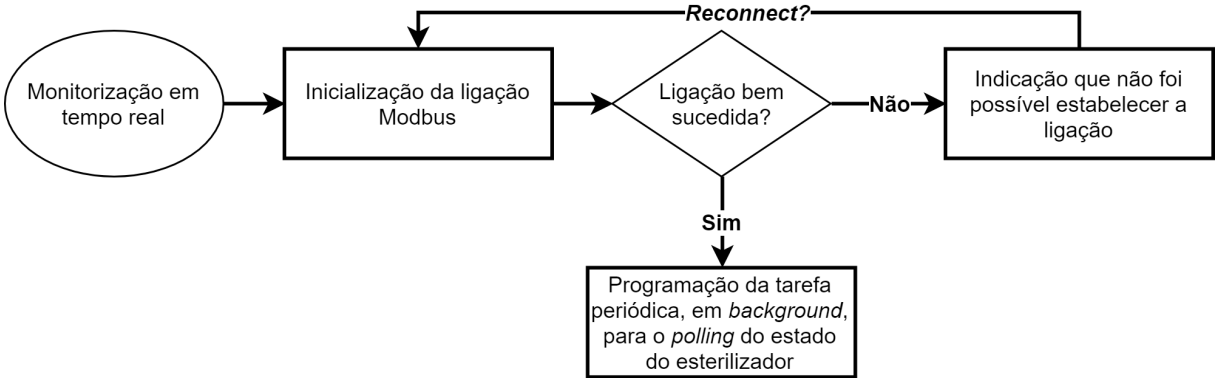


Figura 5.30: Esquema dos primeiros passos na monitorização em tempo real.

Na Figura 5.31 está representado o esquema de todo o processo relativo ao estado do esterilizador para sequenciação da monitorização em tempo real, que segue após a programação da tarefa periódica para o seu *polling* da Figura 5.30. Este processo começa pela leitura dos *coils* do estado do esterilizador e formulação da mensagem sobre esta aquisição, que é enviada para o método de interpretação dessas mensagens[4] (Figura 5.27). Este método serve de interface entre a aquisição dos dados via Modbus (*background*) e a restante aplicação (*foreground*). De seguida, é determinado o estado do esterilizador e são terminadas todas as tarefas de aquisição dos dados que não estejam de acordo com o estado (por exemplo quando ocorre um erro, é terminada a tarefa de aquisição dos dados do ciclo). Seguidamente, consoante o estado, são realizadas as seguintes ações:

- Para os casos do ciclo e do pré-ciclo é verificada a existência da tarefa de *polling* e, caso ainda não exista, é programada uma tarefa periódica, em *background*, para o *polling* dos dados (dados do ciclo e progresso de insuflação da pressão à camisa, respetivamente). O período do *polling* é de 200ms;

[4]O que se segue após o método de interpretação das mensagens, faz parte desse método, mas com uma representação mais detalhada do processamento, direcionado para o estado do esterilizador.

- Para o caso da ocorrência de um erro é programada uma tarefa de execução única, em *background*, para a leitura do número de erro. Desta forma, este será lido de forma periódica, durante o ciclo de segurança, com a mesma periodicidade que a leitura do estado do esterilizador;
- Para o caso do esterilizador parado são colocados todos os campos relativos ao ciclo nos seus valores por omissão e é indicado, no campo de mensagens, que no momento não está a ser executado nenhum ciclo e que se está a aguardar pela seleção do programa (Figura 5.23).

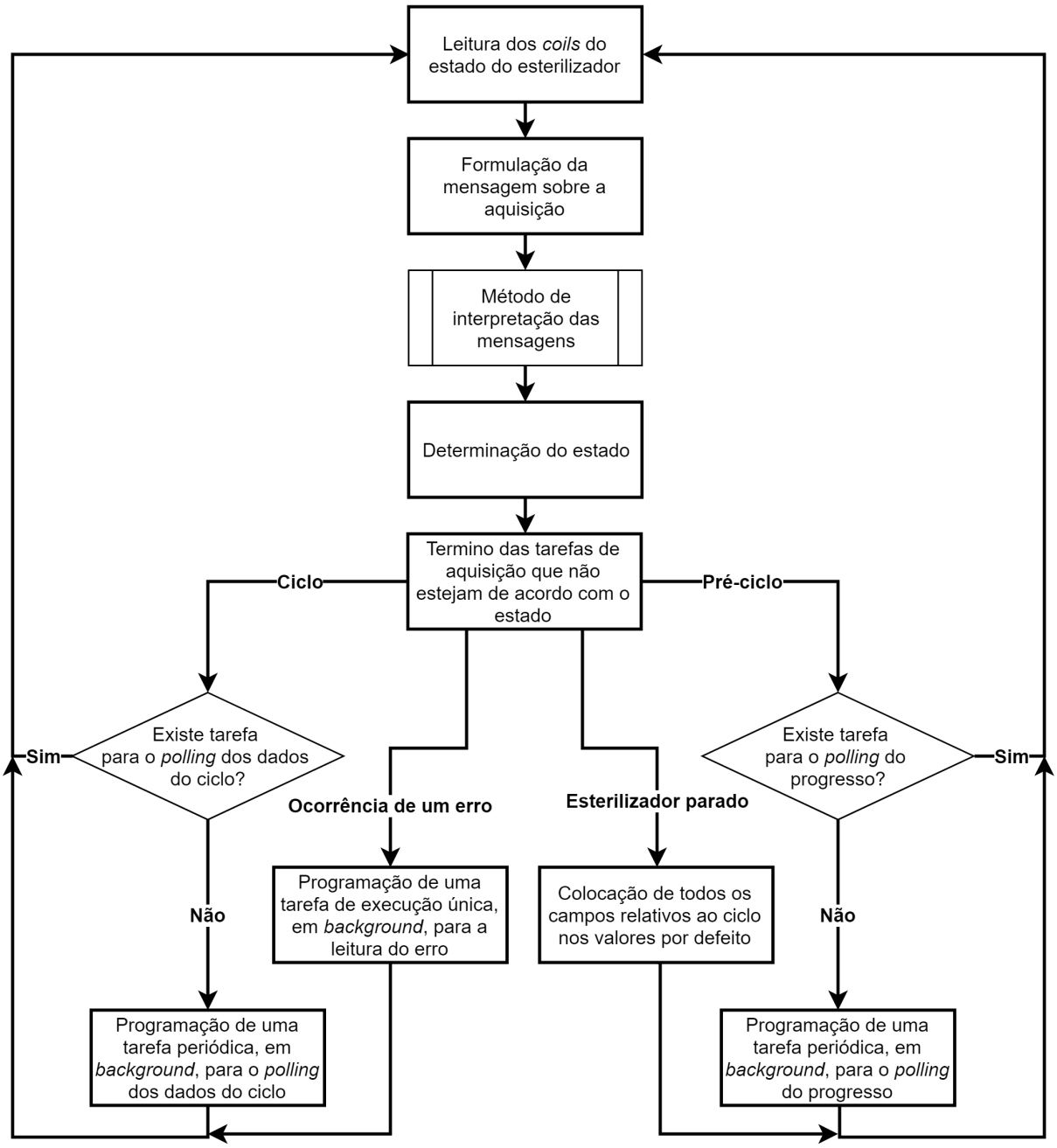


Figura 5.31: Esquema da leitura e processamento do estado do esterilizador.

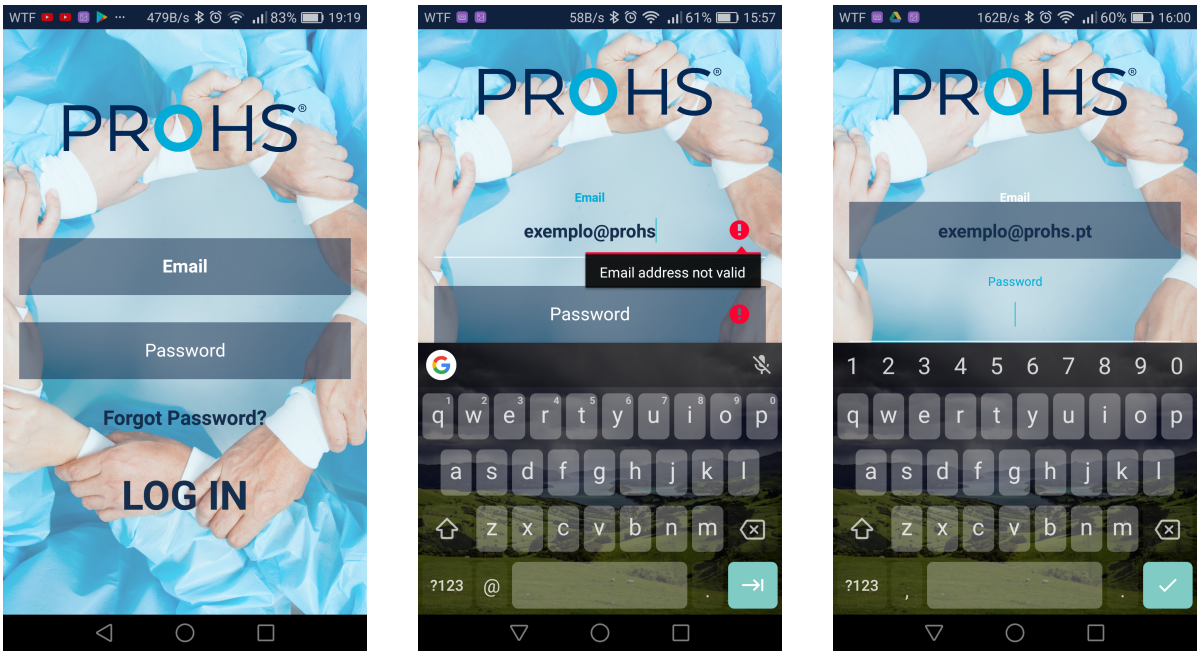
5.4 Autenticação e *Back-end* (Firebase)

Para o desenvolvimento da componente de autenticação na aplicação foram feitas várias pesquisas e estudos sobre técnicas e formas de a implementar. De uma forma geral, existem duas formas diferentes: uma que consiste no desenvolvimento de todos os componentes necessários, incluído o servidor, as API's de comunicação e de gestão da informação tanto no servidor como no Android, e uma segunda que consiste na utilização de um serviço *backend* em *cloud* desenvolvido e dedicado a esse propósito. Como existem muitos fatores a considerar no desenvolvimento integral desta componente e na gestão de infraestruturas (a implementação de um sistema de autenticação requer muita experiência e tempo), dos quais o mais importante é a segurança do utilizador e da informação, assim, optou-se em aprender e utilizar um serviço existente, onde a escolha recaiu sobre o Firebase. O Firebase é uma plataforma de desenvolvimento com muitas ferramentas e serviços que ajudam a desenvolver aplicações de qualidade, de forma rápida, simples e intuitiva. Um dos serviços que a Firebase disponibiliza é o de autenticação, que permite autenticar os utilizadores de várias formas recorrendo a: *email* e *password*, número de telefone, contas Google, Facebook e Twitter, entre outros. Para além disto, por ser uma plataforma da Google, o Firebase está completamente integrado no Android Studio, o que facilita a sua integração com o projeto e também é gratuito para projetos de pequeno porte, com algumas limitações[5].

Para esta aplicação optou-se por utilizar a autenticação com *email* e *password*, devido a ser a forma mais simples de autenticação e não traz limitações de quantidade (autenticação com número de telefone possui limitação de 10 mil autenticações/mês). Os primeiros passos consistiram na integração do Firebase no projeto e na ativação do serviço de autenticação. Após isto, foi criada uma atividade simples de autenticação (Figura 5.32(a)), composta por uma imagem de fundo, um título com o nome da empresa, dois campos de preenchimento (*email* e *password*), e dois botões, um para efetuar o *log in* e outro para alterar a *password* da conta. Geralmente, as aplicações também possuem a opção de se registar, mas neste caso é uma aplicação proprietária e é preferível efetuar registo dos utilizadores pelos técnicos da empresa de forma controlada. Os campos de preenchimento foram programados de forma a incorporar um mecanismo de verificação, ou seja, os campos possuem uma verificação básica antes de enviar o pedido de autenticação e desta forma não sobrecarregam a comunicação com o servidor com tentativas de autenticação inválidas. Para esse efeito recorreu-se a uma biblioteca externa, *Android EditText Validator*, desenvolvida por Andrea Baccega (2018). Esta biblioteca dispõe de um leque mais abrangente de testes de validade do que aqueles que estão incorporados, por omissão, no Android. Também permite uma configuração mais simples e direta das propriedades dos campos de preenchimento, como mensagens de indicação de preenchimento indevido Figura 5.32(b). Caso seja efetuada a tentativa de autenticação com os campos indevida-

[5]Para ver os detalhes do plano gratuito pode visitar a página *web* do Firebase (Google, 2018a).

mente preenchidos, surge um sinal vermelho do lado direito do campo de preenchimento e quando este é pressionado, é apresentada a mensagem. Para além da validação dos campos de preenchimento, estes foram configurados de forma a apresentar a sugestão de preenchimento, neste caso o campo de cima é destinado para *email* e o de baixo para o *password* (texto em branco na Figura 5.32(a)). Quando o campo é pressionado a sugestão não desaparece, mas sim transita para cima do campo (Figura 5.32(c)).



(a) *Layout* gráfico da atividade de autenticação.

(b) Captura com indicações de preenchimento indevido.

(c) Captura durante preenchimento dos campos.

Figura 5.32: Capturas de ecrã da atividade de autenticação.

Como a opção de registo na aplicação não foi implementada, o registo dos utilizadores é feito manualmente na plataforma do Firebase na secção de autenticação no separador de utilizadores (Figura 5.33). Este separador é composto por uma lista com todos os utilizadores registados e no topo da lista existe a opção de adicionar utilizador. Ao pressionar esta opção surge um *pop-up* com dois campos de preenchimento (*email* e *password*). Após adição do novo utilizador, é-lhe atribuído automaticamente um identificador único (UID - *Unique Identifier*) e não existe forma de saber a *password* do utilizador[6]. O Firebase possui um mecanismo de alteração da *password*, ou seja, caso o utilizador o pretenda, é-lhe enviado por *email* uma mensagem com o *link* (Figura 5.34) para a janela de alteração da *password* (Figura 5.35(a)). Para esse efeito, foi criado um *pop-up* na atividade de autenticação para enviar o pedido de alteração ao servidor da Firebase (Figura 5.35(b)) e para despoletar o mecanismo de alteração da *password*, o utilizador deve pressionar o botão "*Forgot Password*?" e seguir as instruções.

[6]É uma forma de segurança, ou seja, o proprietário da aplicação nunca irá saber as *passwords* dos seus utilizadores (caso seja modificado por parte do utilizador após registo).

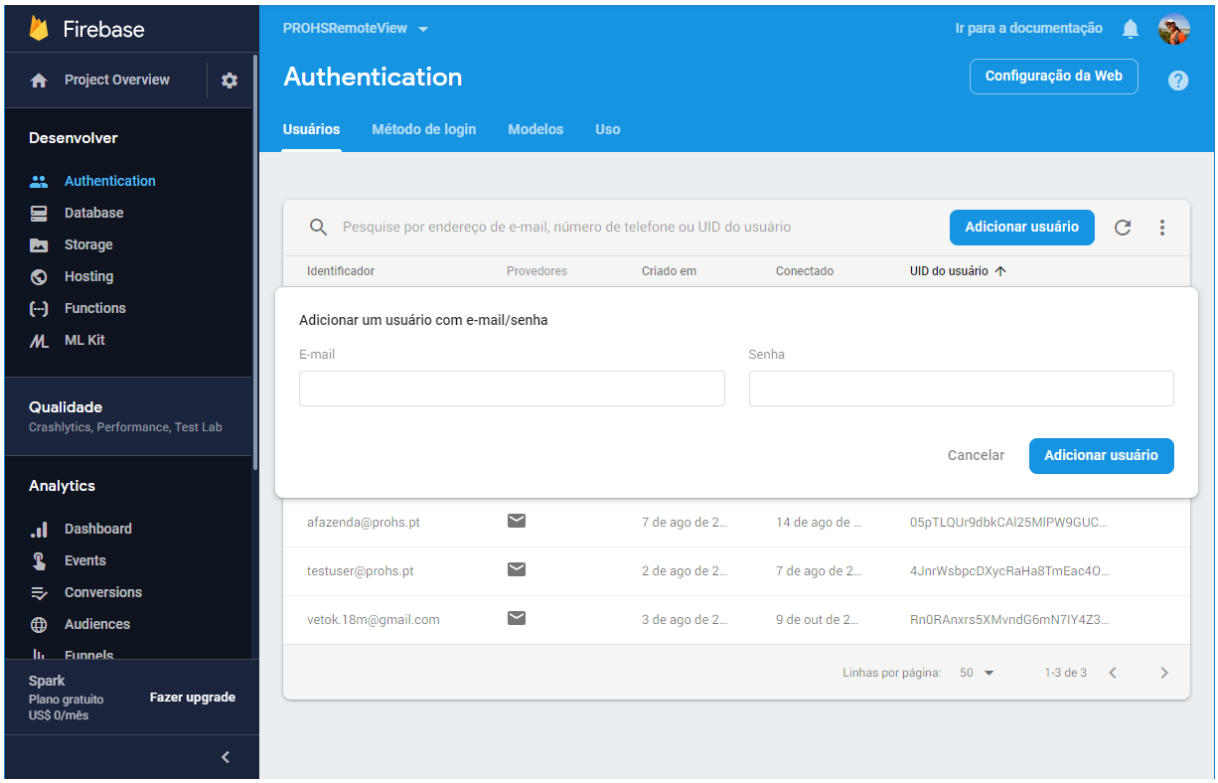


Figura 5.33: Janela de autenticação da consola do Firebase.

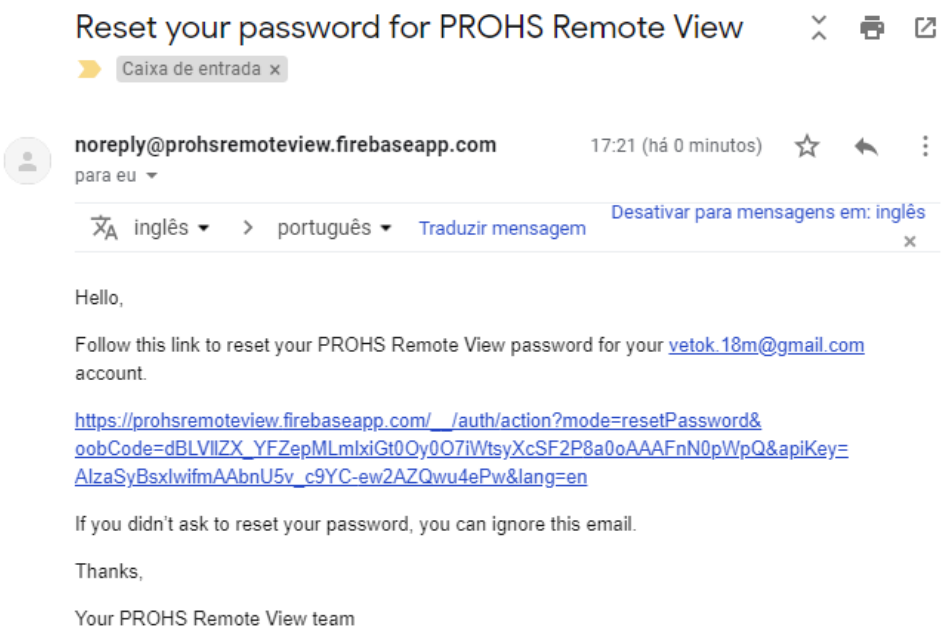
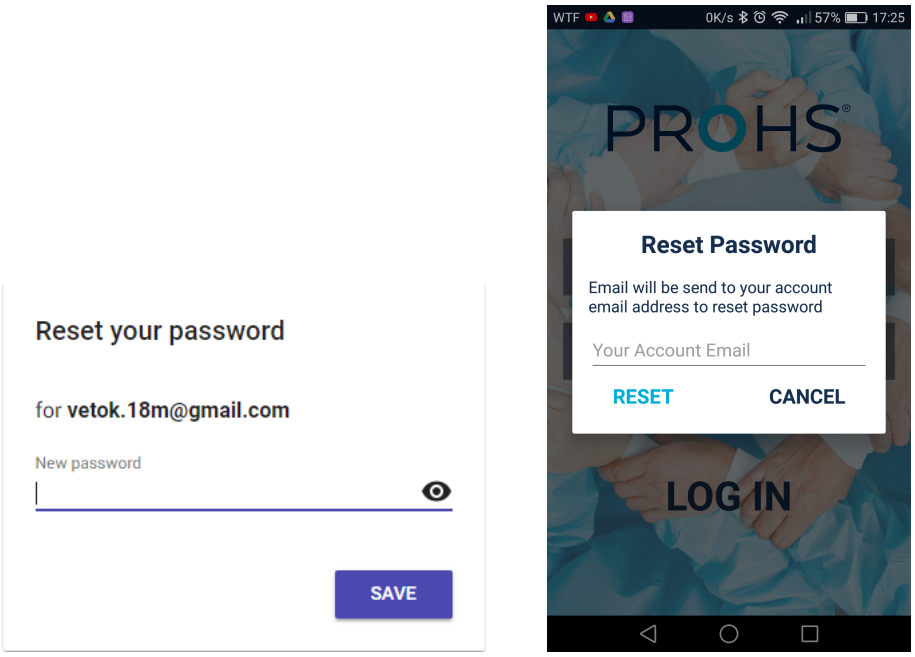


Figura 5.34: *Email* com as instruções para alteração da *password*.



(a) (b)

Figura 5.35: (a) Janela de alteração da *password*; (b) *Pop-up* de pedido de alteração da *password*.

Posto isto, o código da aplicação foi reorganizado de forma a que quando o utilizador ligar a aplicação, é efetuada a verificação se o utilizador está autenticado. Caso esteja autenticado, terá acesso imediato ao conteúdo da aplicação, caso contrário será apresentada a atividade de autenticação. A aplicação foi configurada para reter o estado de autenticação, para evitar constantes pedidos de autenticação ao utilizador. Caso este pretender terminar a sessão, pode fazê-lo a partir do *navigation drawer*, ao pressionar a opção "*Sign Out*". Na Figura 5.36 está representado o esquema simplificado do funcionamento da aplicação após a integração do sistema de autenticação à aplicação.

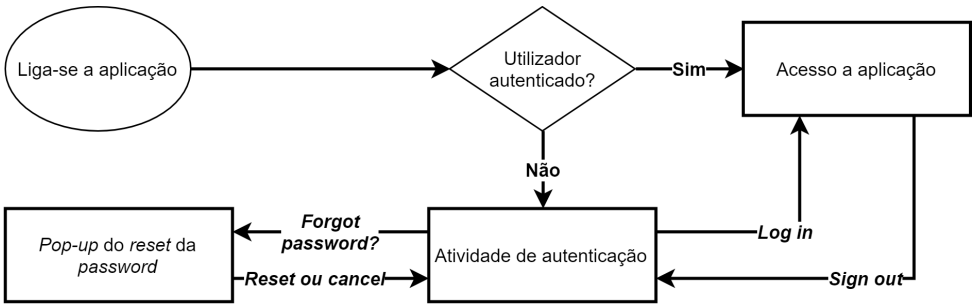


Figura 5.36: Esquema simplificado do funcionamento da aplicação com a adição da autenticação.

Após a implementação da autenticação, o objetivo consistia no desenvolvimento de um mecanismo de gestão e distribuição de informação para o estabelecimento da ligação com os esterilizadores. De uma forma temporária, os dados de ligação aos esterilizadores podiam ser configurados localmente no *smartphone* e consideravam-se que as credencias

de acesso a todos os esterilizadores eram as mesmas. Esta abordagem não era viável nem segura, pois desta forma cada *smartphone* dos clientes deveria ser configurado individualmente por um técnico ou dar total liberdade aos clientes para estabelecerem a ligação através do endereço IP. Por questões de segurança e privacidade, não era desejável, pois qualquer cliente poderia ligar-se a qualquer esterilizador da PROHS, desde que saiba o seu endereço IP. Para solucionar estas situações, recorreu-se ao outro serviço da Firebase, *Realtime Database*, uma base de dados *No Structured Query Language* (NoSQL) em *cloud*, que permite armazenar e sincronizar os dados dos utilizadores em tempo real. Esta base de dados consiste num objeto *JavaScript Object Notation* (JSON) único, que fica sincronizado com todos os utilizadores ligados. Isto é conseguido porque todos as aplicações cliente compartilham uma instância desta base de dados e através de *listeners* de alterações, atualiza a informação em tempo real.

Antes de avançar com a implementação da base de dados, foi necessário definir a estrutura dos dados de um modo semelhante àquele que se faz com as bases de dados SQL. A base de dados NoSQL possui as suas particularidades e pormenores de gestão eficiente, mas, de forma geral, nesta aplicação, a estrutura de dados consiste em três tabelas[7], com a informação minimalista (suficiente para o funcionamento correto da aplicação):

- Entidades - ficou decidido que não existe necessidade de criar credenciais de acesso ao servidor *Web* de forma individual para cada equipamento, mas sim organizar os equipamentos por entidade e atribuir as mesmas credencias para todos da mesma entidade. Como tal, cada membro desta tabela está constituído por:
  - Uma lista de equipamentos;
  - Uma lista de membros (utilizadores que pertencem a esta entidade);
  - Credencias de acesso (*login* e *password*).
- Equipamentos - os membros desta tabela são compostos por:
  - Nome da entidade a que pertence;
  - Identificador do equipamento;
  - Endereço IP;
  - Porta *Ethernet* do Modbus.
- Utilizadores - os membros desta tabela são compostos por:
  - *Email*;
  - Nome da entidade a que pertence.

Para além disto, o *Realtime Database* permite adicionar regras de acesso aos dados. Estas regras foram definidas de forma a que os utilizadores tenham acesso somente à informação

[7]Em JSON, ao contrário das bases de dados SQL, não existem tabelas nem registos. A informação no objeto JSON é organizada em forma de uma árvore com nódulos que representam os dados (Google, 2018b).

a que lhe diz respeito, da mesma forma foram adicionadas regras especiais no preenchimento e modificação dos dados, de forma a evitar ocorrência de erros. Esta parte é a única parte do lado do servidor que requer programação, tudo o resto é efetuado com a API da Firebase e código do lado do cliente, incluindo o preenchimento da base de dados. No entanto, no caso desta aplicação, uma parte do preenchimento deve ser realizado manualmente, nomeadamente a criação de entidades e adição de membros às mesmas. Para o registo dos esterilizadores foi desenvolvida a atividade de definições (Figura 5.37), que consiste: num *spinner* com a lista de esterilizadores registados, em três campos de preenchimento (identificação do equipamento, endereço IP e porta *Ethernet* do Modbus) e quatro botões, três de manipulação (registo, modificação e eliminação) e um para obter os dados do equipamento pretendido. Da mesma forma como na atividade de autenticação, os campos de preenchimento na atividade de definições possuem a validação (Figura 5.38), como por exemplo, os campos não podem ser vazios no caso de registo e modificação da informação.

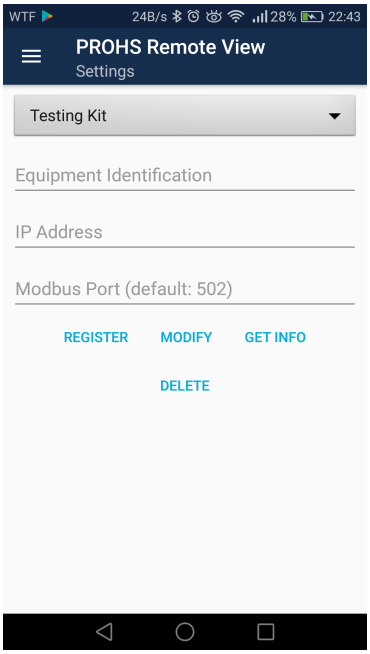


Figura 5.37: Captura de ecrã da atividade de definições.

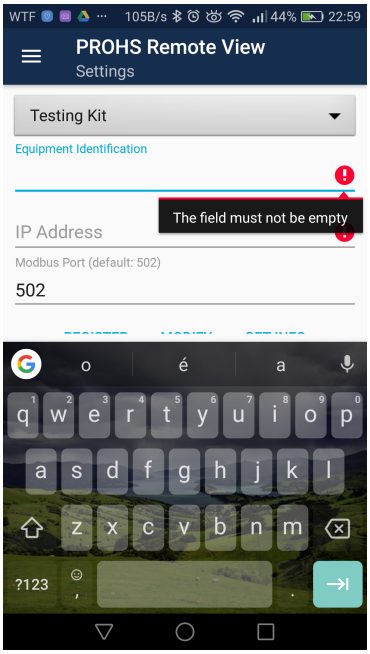


Figura 5.38: Captura de ecrã da atividade de definições com a validação dos campos.

O *Realtime Database*, no pacote gratuito, possui algumas limitações na quantidade de acessos simultâneos (100), de dados armazenada (1 *gigabyte*) e de informação transferida (10 *gigabytes*/mês). Estes limites pressupõem-se suficientes para esta aplicação, no entanto, para evitar possíveis sobrecargas, foram tomadas algumas precauções na quantidade de acessos à base de dados e na quantidade de dados descarregados, nomeadamente na estrutura da árvore do objeto JSON, para que somente sejam descarregadas as informações estritamente necessárias. A aplicação foi programada de forma a aceder à base de dados na *cloud* somente uma vez após a inicialização e autenticação, e sempre que hajam

alterações nos dados na Firebase, estes são atualizados.

Na aplicação, para além do *spinner* na atividade de definições, foi colocado um *spinner* também no *navigation drawer* (Figura 5.39), onde o utilizador pode escolher o esterilizador ao qual se pretende ligar. Para que isto tudo funcione, foi desenvolvido um mecanismo de gestão dos dados dos esterilizadores, que está baseado nos ficheiros de preferências compartilhadas. As preferências compartilhadas são uma das formas distintas de guardar os dados em Android, sendo a mais indicada para a questão em causa. Geralmente são utilizadas para guardar pouca quantidade de dados simples e não estruturados. As preferências compartilhas consistem em ficheiros XML, onde são escritos e lidos os dados primitivos, em forma de um par *key-value*, podem ser: booleanos, *floats*, inteiros, *longs* e *strings*. Estes ficheiros persistem ao longo das sessões do utilizador, mesmo após término da aplicação, o que torna viável a sua utilização para manter a informação sobre os esterilizadores para estabelecer as ligações. Para cada esterilizador é criado um ficheiro de preferências compartilhadas, com a sua informação (identificação, endereço IP e porta *Ethernet* do Modbus), numerados a começar com '0' para o primeiro esterilizador, '1' para o segundo, e assim sucessivamente. Pela mesma ordem é preenchido o *spinner*, em que as designações dos seus elementos são as identificações dos esterilizadores. Para que a aplicação saiba, de forma persistente ao longo das sessões, o esterilizador que o utilizador pretende monitorizar, foi criado um ficheiro de preferências compartilhadas geral/principal, que contém a informação acerca da entidade, mais especificamente a quantidade de esterilizadores registados e as credencias de acesso, e o número do esterilizador escolhido pelo utilizador.

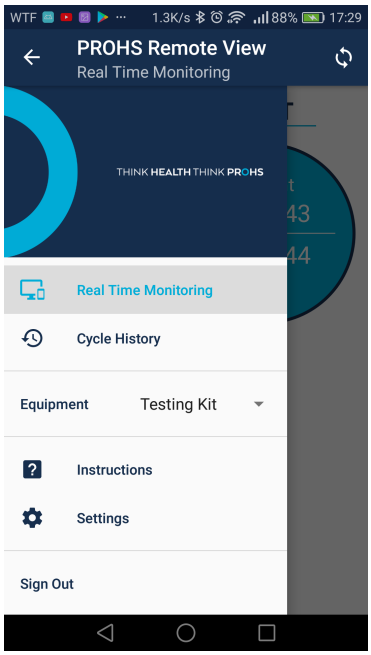


Figura 5.39: Captura de ecrã do *navigation drawer* final.

Inicialmente, o acesso à base de dados era efetuado somente uma única vez após a autenticação e os dados permaneciam guardados nos ficheiros de preferências compartilha-

das e somente eram apagados no *sign out*. Neste tipo de abordagem, para que o utilizador receba as atualizações efetuadas na base de dados, deve terminar a sessão e autenticar-se de novo, para recarregar os dados. Com o objetivo de aprendizagem e aprofundamento dos conhecimentos no *Realtime Database*, decidiu-se experimentar uma abordagem diferente da inicial, incorporando o conceito de atualização em tempo real deste serviço e o seu potencial. Para conseguir o pretendido, o comportamento da aplicação foi alterado, de forma a efetuar o acesso à base de dados sempre que a aplicação seja iniciada. Durante este acesso inicial, são adicionados *listeners* de alterações: à lista de equipamentos, para deteção de novos registos ou eliminações de equipamentos, e a todos equipamentos, individualmente, para capturar modificações nos dados. Assim, sempre que existe alguma alteração nestes dados, os *listeners* captam-na e a aplicação recebe a notificação em conjunto com um *data snapshot* dos dados correspondentes. De seguida são atualizados os ficheiros com preferências compartilhadas de acordo com as alterações. Na Figura 5.40 está representado o esquema simplificado do processo de registo, eliminação e modificação de um equipamento, recorrendo aos *listeners* mencionados anteriormente.

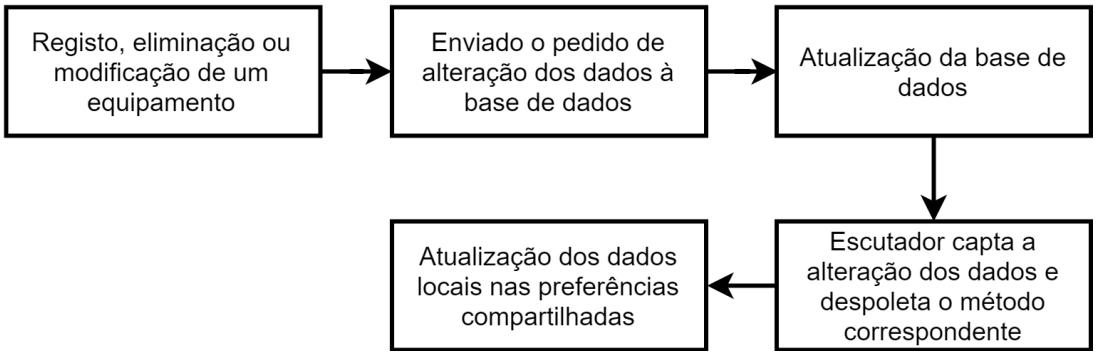


Figura 5.40: Esquema simplificado do processo de registo, eliminação e modificação de um equipamento.

Como os *listeners* são serviços em *background*, que funcionam mesmo após o termino da aplicação, é muito importante removê-los antes de terminar a aplicação. Para evitar complicações adicionais com as verificações da existência de alterações durante o tempo em que a aplicação esteve desligada, decidiu-se remover os dados guardados localmente em conjunto com os *listeners*, já que estes podem sempre ser recarregados durante a inicialização inicial. Desta forma, esta segunda abordagem transforma-se numa abordagem completamente *online*, pois é necessário a ligação à *Internet* para que o utilizador obtenha os dados sobre esterilizadores. Na Figura 5.41 está representado o esquema simplificado do funcionamento da aplicação com a adição do *Realtime Database*.

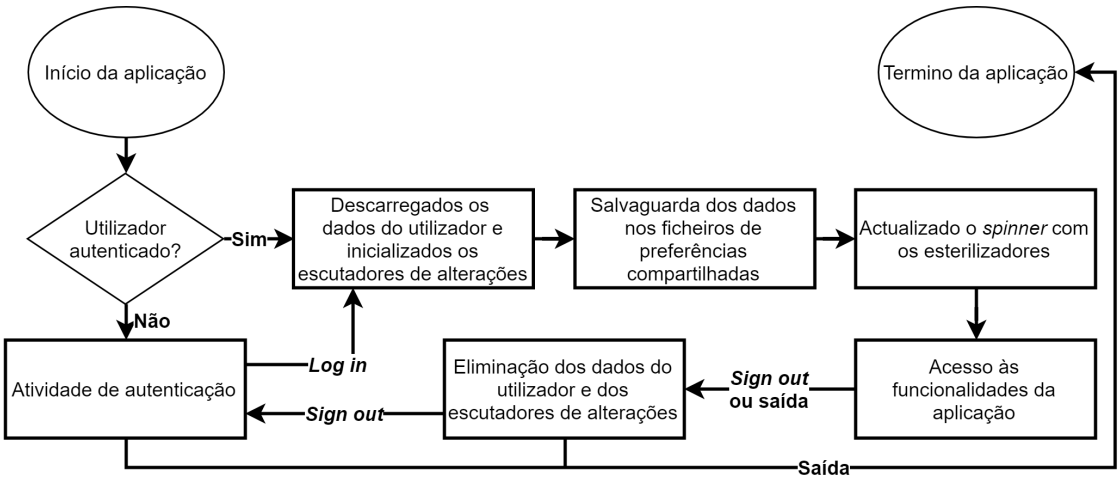
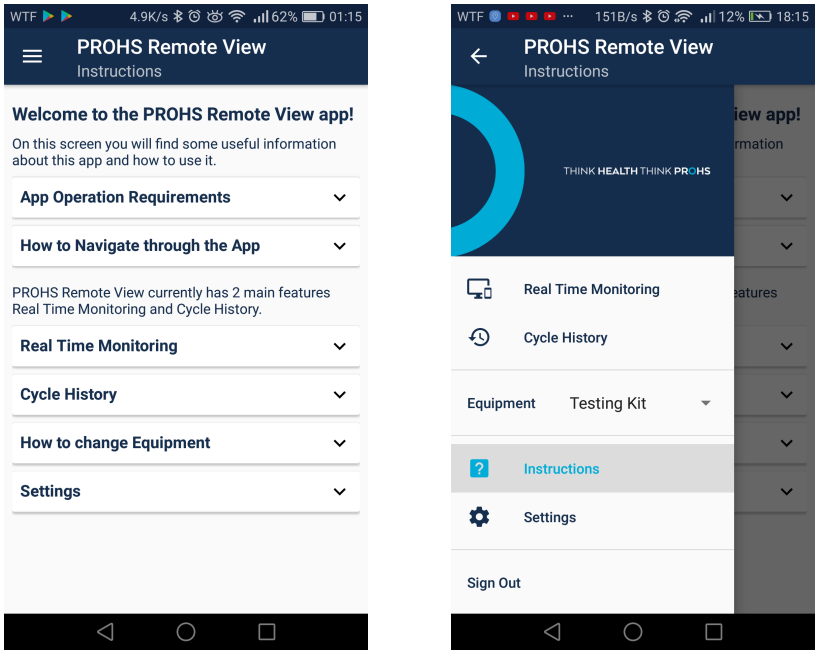


Figura 5.41: Esquema simplificado do funcionamento da aplicação com a adição do *Real-time Database*.

5.5 Instruções de Utilização

Foram adicionadas instruções básicas de utilização à aplicação, para facilitar a utilização inicial da aplicação e para esclarecer eventuais dúvidas que possam surgir durante a sua utilização. A atividade com as instruções (Figura 5.42(a)) é a atividade principal na aplicação, pelo que sempre que o utilizador iniciar a aplicação, verá esta atividade e a partir dela navega, utilizando *navigation drawer* (Figura 5.42(b)) para as funcionalidades que pretender.

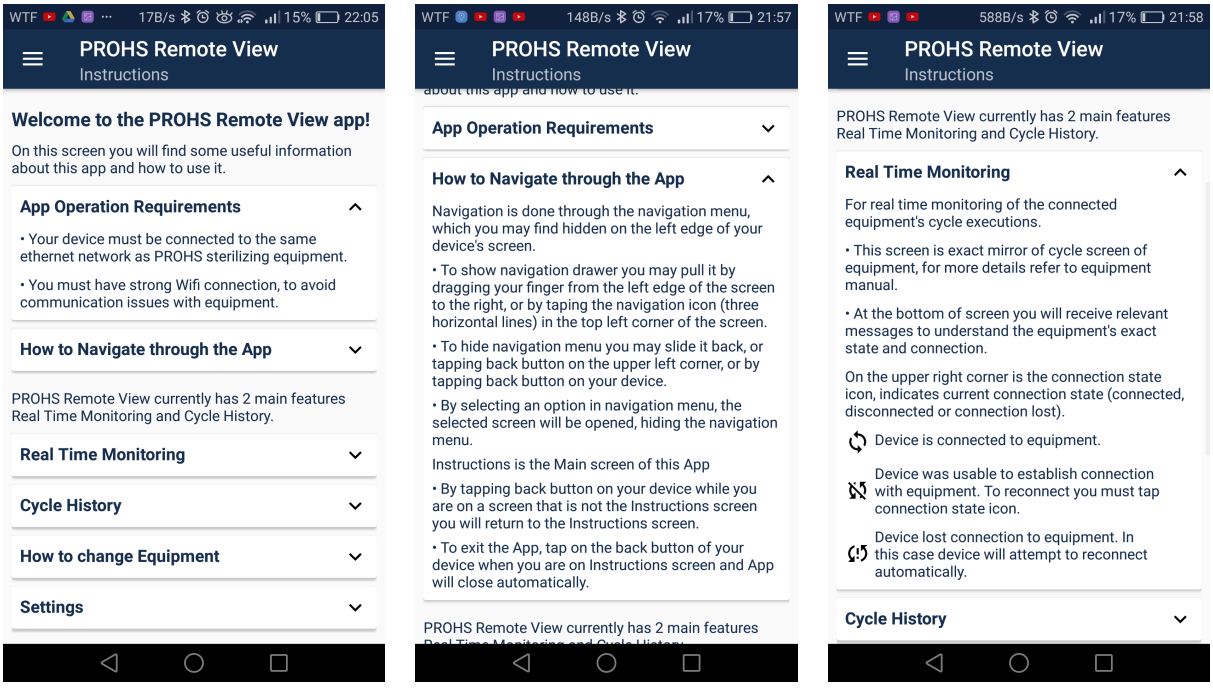


(a) (b)

Figura 5.42: (a) Captura de ecrã da atividade com as instruções de utilização; (b) Captura de ecrã do *navigation drawer* aberto a partir da atividade com as instruções de utilização.

Esta atividade é composta por: um texto de boas vindas; uma pequena introdução acerca da respetiva atividade e um conjunto de tópicos em forma de "gavetas" (no seu estado fechado) com a informação relevante. Para ver o conteúdo dos tópicos o utilizador deve pressionar na "gaveta", para que esta se expande e mostre o texto escondido. A informação está dividida por tópicos de modo a não aglomerar muita informação e ser fácil e rápido localizar informação pretendida. Os tópico são:

- **Requerimentos de operação da aplicação** (Figura 5.43(a)) - neste tópico estão descritos alguns requerimentos para o correto funcionamento da aplicação. Mais especificamente, a necessidade do dispositivo estar ligado à mesma rede *Ethernet* que o esterilizador ao qual se pretende estabelecer a ligação e a necessidade de uma ligação *Wi-Fi* forte para evitar problemas na ligação;
- **Navegação pela aplicação** (Figura 5.43(b)) - a navegação pela aplicação é conseguida via um menu de navegações (*navigation drawer*). Neste tópico é explicado como aceder e esconder este menu, trocar de funcionalidade, o comportamento da aplicação ao retroceder e como sair da aplicação;
- **Monitorização em tempo real** (Figura 5.43(c)) - neste tópico está descrita esta funcionalidade e as suas particularidades, como o campo de mensagens na parte inferior da atividade e diferentes estados de ligação;



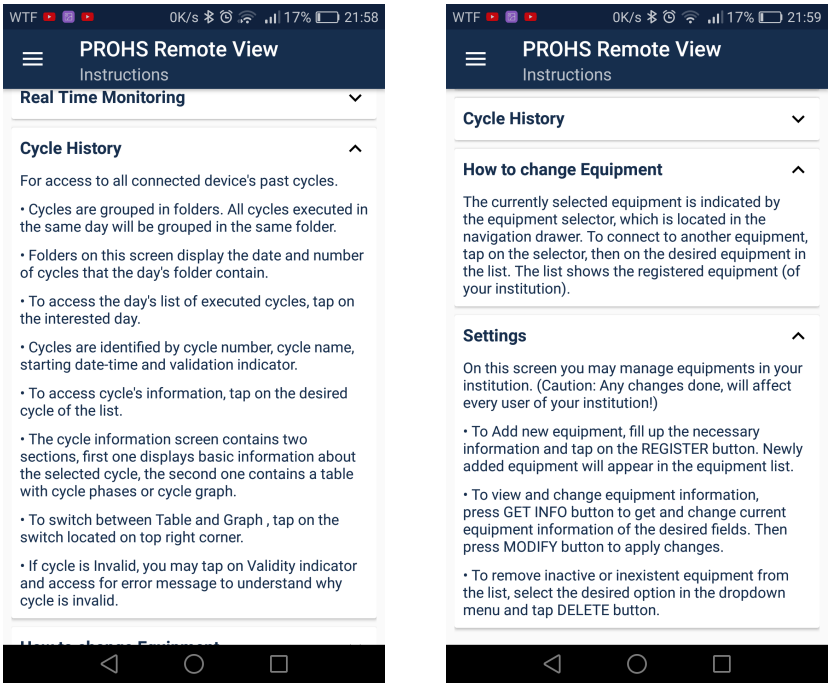
(a) (b) (c)

Figura 5.43: Capturas de ecrã das instruções: (a) dos requerimentos de operação da aplicação; (b) da navegação pela aplicação; (c) da monitorização em tempo real.

- **Histórico de ciclos** (Figura 5.44(a)) - neste tópico está descrita esta funcionalidade: a navegação pelo histórico, o *layout* das listas e da atividade que representa o ciclo. Também são explicados alguns pormenores, como a troca entre a tabela com as

fases do ciclo e o gráfico do ciclo e a visualização da mensagem de erro nos ciclos inválidos;

- **Forma de escolha do esterilizador** (Figura 5.44(b)) - o indicador do esterilizador selecionado está localizado no menu de navegação. Para selecionar outro esterilizador o utilizador deve pressionar no indicador para obter a lista dos esterilizadores acessíveis e escolher o esterilizador desejado;
- **Definições** (Figura 5.44(b)) - neste tópico são explicadas as funcionalidades desta atividade, como registo, modificação e eliminação dos equipamentos.



(a) (b)

Figura 5.44: Capturas de ecrã das instruções: (a) do histórico de ciclos; (b) da forma de escolha do esterilizador e das definições.

5.6 Considerações Finais

Neste capítulo foi descrito o desenvolvimento da aplicação Android para dispositivos móveis para a monitorização remota dos esterilizadores horizontais da PROHS. Foram desenvolvidas duas funcionalidades de monitorização:

- **Monitorização em tempo real** - recorrendo ao protocolo Modbus TCP/IP, esta funcionalidade consiste na monitorização do estado da execução dos ciclos do esterilizador. Permite distinguir os estados: esterilizador parado, execução do pré-ciclo, execução do ciclo e execução do ciclo de segurança (caso ocorra um erro ou falha durante execução do ciclo). Esta funcionalidade foi projetada de modo a possuir um *design* gráfico, semelhante ao que está utilizado nas consolas dos esterilizadores

da PROHS, de modo a facilitar a adaptação dos utilizadores à aplicação;

- **Histórico de ciclos** - consiste num conjunto de ficheiros CSV criados após a execução dos ciclos e organizados de forma racional na *pen* USB ligada à consola do esterilizador. A aplicação recorre ao servidor *Web* da consola para aceder ao histórico de ciclos e representa-o de forma intuitiva e agradável. A representação do ciclo na aplicação, é semelhante àquela que está no talão impresso após o ciclo, estando composta por um conjunto dos principais parâmetros do ciclo, uma tabela com as fases do ciclo e um gráfico do ciclo.

A navegação pela aplicação foi realizada através de um menu de navegação e as transições entre as funcionalidades foram conseguidas de forma suave, de modo a proporcional uma maior usabilidade e continuidade na utilização da aplicação.

No *design* da aplicação foram utilizadas imagens representativas da empresa com o tema da aplicação a recorrer às cores principais da PROHS: azul escuro, azul claro e branco.

Para proporcionar alguma segurança à aplicação e controlo dos utilizadores, foi adicionado um sistema de autenticação com os servidores da Firebase. Este sistema, para além de autorizar a utilização da aplicação, também distribui aos dispositivos dos utilizadores, de forma oculta e controlada, a informação necessária para estabelecimento das ligações com os esterilizadores.

Por fim, foram adicionadas à aplicação instruções básicas de utilização, para introduzir a aplicação a novos utilizadores e para esclarecer eventuais dúvidas durante a utilização.

Apesar de estarem implementadas as principais funcionalidades da aplicação (monitorização em tempo real, histórico de ciclos, autenticação e controlo dos utilizadores), estas ainda se encontram num estado imaturo. Por questões de tempo não foram realizados os testes exaustivos nem os chamados testes de *stress* à aplicação e ao sistema em geral, para a eliminação dos *bugs* e afinações. É de salientar que o *design* do histórico de ciclos não foi definido, pelo que se encontra num estado provisório. Outro aspeto a salientar é o sistema de distribuição dos dados aos utilizadores não estar acabado. O sistema está perfeitamente funcional, no entanto, as funcionalidades descritas (atualização em tempo real e gestão dos equipamentos) na abordagem final na página 90, não são particularmente úteis para os utilizadores normais da aplicação. No entanto, podem ser úteis para os técnicos da PROHS, para o registo de novos clientes e as configurações inicias. O que se pretendia, era interligar ambas as abordagens: a inicial, que está baseada na utilização mais *offline*, em que os dados do utilizador são descarregados após a sua autenticação e permanecem no dispositivo até o *sign out*, ou seja, os utilizadores não necessitariam da constante ligação à *Internet* para se poder ligar aos seus esterilizadores; e a final, que seria somente disponível para os técnicos para as configurações iniciais. Mas, por questões de tempo não foi possível concluir esta parte.

Estando concluída a descrição do desenvolvimento da aplicação, o próximo capítulo irá expor as conclusões do estágio e as propostas para trabalhos futuros.

Capítulo 6 - Conclusões

Este capítulo apresenta as conclusões que se podem retirar do estágio realizado e apresenta as sugestões para trabalhos futuros na aplicação desenvolvida.

6.1 Conclusões

O presente documento teve como objetivo descrever as tarefas realizadas durante o estágio efetuado na PROHS, com especial foco no desenvolvimento da aplicação Android para a monitorização remota dos esterilizadores horizontais da empresa. Nessa medida, foi efetuado um levantamento de requisitos para a aplicação de monitorização remota, a partir das soluções análogas dos principais concorrentes da PROHS, e tentou destacar-se esta solução das existentes. Um exemplo dos fatores diferenciadores da aplicação desenvolvida consiste em não requerer a instalação de infraestruturas adicionais (por exemplo um servidor dedicado), tanto do lado do cliente como do lado da empresa, o que por sua vez não traz custos adicionais.

A solução desenvolvida vai ao encontro dos objetivos propostos inicialmente. Desenvolveu-se um protótipo duma aplicação para dispositivos móveis Android, com capacidade de se ligar remotamente aos esterilizadores horizontais e monitorizar a execução dos ciclos em tempo real, reduzindo a exposição dos profissionais, que operam este tipo de equipamento, aos eventuais riscos que daí possam advir. Também foi desenvolvido e acrescentado à solução o histórico de ciclos, que traz uma mais valia à aplicação e aos esterilizadores da PROHS. Foi incorporado na aplicação um sistema de autenticação e uma base de dados da Firebase, para a gestão dos utilizadores, registo dos clientes e dos seus esterilizadores. A aplicação permite a ligação a vários esterilizadores, um de cada vez, com possibilidade de escolha do esterilizador a monitorizar.

Neste momento, a aplicação opera somente na rede *Ethernet* da instituição onde forem instalados os esterilizadores e requer a necessidade de ligação à Internet no momento da iniciação da aplicação, para a sincronização dos dados do utilizador da Firebase com a aplicação.

Outro dos pontos relevantes é que todas as ferramentas utilizadas no desenvolvimento da aplicação são ferramentas disponíveis sem custos e com licenças compatíveis com *Open Source*. Todas as bibliotecas desenvolvidas por várias pessoas/organismos incorporadas na aplicação possuem licenças abertas e permitem a sua inclusão em produtos comerciais.

Também é de salientar que a aplicação foi desenvolvida com um *design* agradável e com uma navegação intuitiva semelhante àquela das aplicações populares, aumentando a sua usabilidade.

Por último, mas não menos importante, foram atingidos os objetivos pessoais do estagiário com êxito, adquirindo competências profissionais em diversas áreas num ambiente empresarial.

6.2 Trabalhos Futuros

A aplicação desenvolvida ficou operacional, tendo sido assim cumprido o principal objetivo deste trabalho de estágio. No entanto, para que o trabalho desenvolvido possa ser integrado num esterilizador comercial, é necessário, antes, delinear e implementar uma fase de testes exaustivos, capaz de atestar o funcionamento e robustez da aplicação em contexto real, nomeadamente com diferentes esterilizadores, diferentes utilizadores e diferentes equipamentos móveis. Como tal, este é um dos trabalhos futuros que se sugere.

Sugere-se também, como trabalho futuro, dotar a aplicação de interface multi-língua, dado que a solução atual suporta apenas a língua inglesa. Adicionalmente, sugere-se a adaptação da aplicação para outros dispositivos móveis, nomeadamente com diferentes tamanhos de ecrã e densidade de pixeis, visto que a aplicação foi desenvolvida e testada num dispositivo particular.

Outras melhorias que podem ser feitas à aplicação são:

- Criar um *icon* da aplicação;
- Acrescentar notificações sonoras e/ou vibratórias no término do ciclo e na ocorrência duma falha/erro durante o ciclo;
- Acrescentar a possibilidade da aplicação operar *offline*, sem ligação à Internet, permanecendo os dados do utilizador de forma encriptada no dispositivo após o *log in* até o *sign out*, minimizando assim o tráfego de dados;
- Definir um tema ao histórico de ciclos, que de momento se encontra num estado provisório;
- Acrescentar uma forma de pesquisa pelo histórico de ciclos;
- Tratar as mensagens de erro no histórico de ciclos, quando existe algum problema na aquisição dos ficheiros (problemas na rede);
- Substituir o servidor *Web* pelo servidor FTP no acesso ao histórico de ciclos, aumentando assim a segurança do sistema e rapidez no funcionamento do histórico;
- Acrescentar uma forma de guardar o histórico de ciclos localmente no dispositivo, para ter acesso ao mesmo, sem estar ligado ao esterilizador.

Bibliografia

Acosta-Gnass, Silvia I. e Valeska de Andrade Stempliuk (2009). *Sterilization manual for health centers*. Pan American Health Organization. Cap. Sterilization, pp. 73–85. ISBN: 978-92-75-12926-5.

Android Developers (2018a). *Android Studio release notes. 3.2 (September 2018)*. URL: https://developer.android.com/studio/releases/ (acedido em 02/10/2018).

– (17 de abr. de 2018b). *App resources overview*. URL: https://developer.android.com/guide/topics/resources/providing-resources (acedido em 15/10/2018).

– (30 de abr. de 2018c). *Application Fundamentals*. URL: https://developer.android.com/guide/components/fundamentals (acedido em 14/09/2018).

– (2018d). *ART and Dalvik*. URL: https://source.android.com/devices/tech/dalvik (acedido em 20/09/2018).

– (24 de set. de 2018e). *Configure your build*. URL: https://developer.android.com/studio/build/ (acedido em 15/10/2018).

– (16 de abr. de 2018f). *Getting Started with the NDK*. URL: https://developer.android.com/ndk/guides/ (acedido em 02/10/2018).

– (11 de out. de 2018g). *Intents and Intent Filters*. URL: https://developer.android.com/guide/components/intents-filters (acedido em 12/10/2018).

– (7 de ago. de 2018h). *Kotlin and Android*. URL: https://developer.android.com/kotlin/ (acedido em 02/10/2018).

– (24 de set. de 2018i). *Meet Android Studio*. URL: https://developer.android.com/studio/intro/ (acedido em 03/10/2018).

– (3 de set. de 2018j). *Platform Architecture*. URL: https://developer.android.com/guide/platform/#art (acedido em 14/09/2018).

– (2018k). *Understand Tasks and Back Stack*. URL: https://developer.android.com/guide/components/activities/tasks-and-back-stack (acedido em 11/07/2018).

Baccega, Andrea (2018). *Android EditText Validator*. URL: https://github.com/vekexasia/android-edittext-validator (acedido em 16/07/2018).

Buckey, Caroline e Sarah Spikes (s.d). *How to Use Git and GitHub*. URL: https://eu.udacity.com/course/how-to-use-git-and-github--ud775 (acedido em 22/03/2018).

Catlin, Ben (29 de jul. de 2013). *Mobile Modbus*. URL: https://github.com/bigcat/Mobile-Modbus (acedido em 05/03/2018).

Clark, Michael (2015). "Optimizing a sterile processing quality system". Em: *Self-Study Series*. URL: https://www.hpnonline.com/ce/pdfs/1505cetest.pdf (acedido em 26/02/2018).

Dion, Marcel e Wayne Parker (2013). "Steam Sterilization Principles". Em: *PHARMACEUTICAL ENGINEERING* 33.6. Ed. por ISPE.

Forouzan, Behrouz A. (2009). *TCP/IP. Protocol Suite*. Fourth Edition. McGraw-Hill.

Francisco, António (2003). *Autómatos Programáveis*. Ed. por ETEP. Segunda Ed. Lidel.

Fujiwara, Lyla et al. (s.d). *Advanced Android App Development by Google*. URL: https://eu.udacity.com/course/advanced-android-app-development--ud855 (acedido em 11/04/2018).

Galpin, Dan et al. (s.d). *Developing Android Apps by Google*. URL: https://eu.udacity.com/course/new-android-fundamentals--ud851 (acedido em 04/04/2018).

Getinge (2013). *Geringe Online*. URL: http://ic.getinge.com/files/us-hc/product-documents/getingeonline/go-bc-overview.pdf (acedido em 26/02/2018).

– (2015). *Getinge Online*. URL: https://apkpure.com/br/getinge-online/com.getinge.customerportal (acedido em 19/03/2019).

– (2018). *Getinge Web Site*. URL: https://www.getinge.com/int/ (acedido em 26/02/2018).

Google (2018a). *Firebase Pricing*. URL: https://firebase.google.com/pricing/ (acedido em 26/07/2018).

– (2018b). *Structure Your Database*. URL: https://firebase.google.com/docs/database/ios/structure-data (acedido em 02/08/2018).

Haggar, Joel e Mloh Bihler (25 de mai. de 2016). *Modbus for Java*. URL: https://sourceforge.net/projects/modbus4j/ (acedido em 05/03/2018).

Hedley, Jonathan (2018). *jsoup: Java HTML Parser*. URL: https://jsoup.org/ (acedido em 11/04/2018).

Infinite Automation Systems (2018). *Modbus4J general discussion*. URL: https://forum.infiniteautomation.com/category/11/modbus4j-general-discussion (acedido em 05/03/2018).

Jahoda, Philipp (2018). *MPAndroidChart*. URL: https://github.com/PhilJay/MPAndroidChart (acedido em 20/04/2018).

Matachana (2018). *EasyLOOK Monitoring Software*. URL: https://www.matachana.com/en/healthcare-en/trazabilidad-en.html (acedido em 26/02/2018).

Matachana, Juan Antonio (2017). *Welcometo the Matachana experience*. URL: https://www.matachana.com/en/matachana-group-en.html (acedido em 26/02/2018).

MEDICA (2018). *MEDICA 2018. World Forum for Medicine*. URL: https://www.medica-tradefair.com/ (acedido em 26/02/2018).

MMM (2016). *Intelligent Service Advisor (ISA)*. URL: http://mmm.taimaz.com/brochure/isa_-_brochure_en.pdf (acedido em 22/02/2018).

– (2018). *Selectomat PL*. URL: https://www.mmmgroup.com/en/products/selectomat-pl (acedido em 22/02/2018).

Modbus.org (24 de out. de 2006). *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*.

– (26 de abr. de 2012). *MODBUS Application Protocol Specification V1.1b3*.

Nurik, Roman, James Williams e Nick Butcher (s.d). *Material Design for Android Developers by Google*. URL: https://eu.udacity.com/course/material-design-for-android-developers--ud862 (acedido em 05/06/2018).

OMRON (2014). *How To Use The NB Web Interface Functionality*. URL: https://www.myomron.com/index.php?action=kb&article=1660 (acedido em 03/09/2018).

– (11 de mar. de 2016). *How To Use USB Memory Sticks On NB HMI*. URL: https://www.myomron.com/index.php?article=1547&action=kb (acedido em 21/03/2018).

– (2017). *Programmable Controllers. CJ2 FAMILY*.

– (2018a). *NB-Designer OPERATION MANUAL*. Inglês. Versão V106-E1-17. OMRON. URL: https://industrial.omron.us/en/media/V106-E1-17_tcm849-114871.pdf.

– (2018b). *NB-series Programmable Terminals. Host Connection Manual*.

– (2018c). *Programmable Terminal. NB Series*.

– (2018d). *Remote Viewer for Human Machine Interfaces*. URL: https://industrial.omron.eu/en/products/sysmac-platform/application/sysmac-remote-viewer-for-hmi (acedido em 03/09/2018).

Omron (10 de mai. de 2018). *Consolas TÃ¡cteis Compactas. HMI1*. URL: https://industrial.omron.pt/pt/services-support/trainings/050-hmi1-nb (acedido em 29/11/2018).

Open Handset Alliance (2007). *FAQ*. URL: https://www.openhandsetalliance.com/oha_faq.html (acedido em 14/09/2018).

PROHS (2018). *Soluções de Serviço Central de Esterilização*. URL: https://www.prohs.pt/turnkey_capabilities.php (acedido em 10/10/2018).

Ramesh, Vivek (2015a). *Android Processes, Threads Slidenerd Style*. URL: https://www.udemy.com/master-android-zero-to-hero/ (acedido em 04/03/2018).

– (2015b). *Java Tutorial For Beginners - Only Object Oriented*. URL: https://www.youtube.com/playlist?list=PLzi8dDyr-I0n_E2uq3FGTjwMNbbkPfotj (acedido em 12/03/2018).

– (2016a). *Android Material Design Tutorial*. URL: https://www.youtube.com/playlist?list=PLonJJ3BVjZW6CtAMbJz1XD8ELUs1KXaTD (acedido em 05/06/2018).

– (2016b). *Android Tutorial for Beginners*. URL: https://www.youtube.com/playlist?list=PLonJJ3BVjZW6hYgvtkaWvwAVvOFB7fkLa (acedido em 04/03/2018).

Richardson, T. et al. (1998). "Virtual network computing". Em: *Richardson, T. and Stafford-Fraser, Q. and Wood, K. R. and Hopper, A.* 2 (1), pp. 33–38.

Sheng-Ju, Sang (2015). "Implementation of Cyclic Redundancy Check in Data Communication". Em: *2015 International Conference on CICN*.

Statista (2018). *Global market share held by smartphone operating systems from 2009 to 2017*. URL: https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/ (acedido em 14/09/2018).

Steelco (2018a). *Steelco Web Site*. URL: http://www.steelcospa.com/en/ (acedido em 26/02/2018).

Steelco (2018b). *SteelcoData ARES software*. URL: http://www.steelcospa.com/en/products-catalogue/medical-products/item/steelcodata-ares-software (acedido em 26/02/2018).

– (s.d[a]). *Steam sterilizing autoclaves*. URL: http://www.steelcospa.com/download_archive/DC-PD-02-VS%20EN%20Rev.11.pdf (acedido em 26/02/2018).

– (s.d[b]). *Washer disinfectors*. URL: http://www.steelcospa.com/download_archive/DC-PD-02-LC%20EN%20Rev.04.pdf (acedido em 26/02/2018).

Steris (2013). *Real Time Management for your CSSD Equipment*. URL: https://www.steris.com/onbDocs/V409/1263/670417.pdf (acedido em 26/02/2018).

– (2018). *Steris Healthcare Web Site*. URL: https://www.steris-healthcare.com/ (acedido em 26/02/2018).

Tamada, Ravi (2015). *Android Material Design working with Tabs*. URL: https://www.androidhive.info/2015/09/android-material-design-working-with-tabs/ (acedido em 14/11/2018).

The Git Project (2018). *PÃ¡gina oficial do git*. URL: https://git-scm.com/ (acedido em 29/11/2018).

Tuttnauer (17 de out. de 2015). *Hospital Autoclaves*. URL: https://tuttnauer.com/sites/default/files/brochures/medical-horizontal-hospital-autoclave-sterilizer-en-17-10-2015.pdf (acedido em 26/02/2018).

– (2018a). *R.PC.R Software*. URL: https://tuttnauer.com/laboratory-autoclaves/biohazard-autoclaves/bsl3-bsl4-autoclaves/55-laboratory-autoclave/documentation (acedido em 26/02/2018).

– (2018b). *Tuttnauer Web Site*. URL: https://tuttnauer.com/ (acedido em 26/02/2018).

zgkxzx (2018). *Modbus for Android*. URL: https://github.com/zgkxzx/Modbus4Android (acedido em 07/03/2018).