



Instituto Superior de  
Engenharia do Porto

P.PORTO

## Licenciatura em Engenharia de Sistemas

# DESENVOLVIMENTO DE UM SISTEMA DE INFORMAÇÃO DE FÁBRICA

Natural Life, Lda.

Jorge Gabriel Azevedo (1160929)

19 de julho de 2019

Orientador ISEP: Prof. Manuel Carlos Felgueiras

Supervisor Externo: Engº Telmo Azevedo



---

# Agradecimentos

---

A realização deste trabalho contou com importantes apoios e incentivos sem os quais não se teria sido possível desenvolver este projeto até ao fim, e pelos quais estarei eternamente grato.

Ao meu pai, à minha mãe e aos meus irmãos que me apoiaram de forma incondicional em todo este processo.

Aos administradores da Natural Life, Ida, em especial ao meu supervisor Engº Telmo Azevedo, por terem confiado em mim para fazer um trabalho tão importante para a empresa.

A todos os meus amigos que me apoiaram neste processo em especial à Maria Inês Almeida, à Bárbara Santos e ao Nuno Dinís por todo o apoio e força que me deram, mesmo nos momentos em que estive mais em baixo.

Não posso deixar de agradecer ao meu orientador de estágio, o Professor Carlos Felgueiras, pelo seu jeito tão particular de me motivar, pelo nível de qualidade de trabalho me exigiu ao longo de todo o tempo, pela dedicação que demonstrou e por sempre me fazer ver as situações como cristal.

Quero ainda agradecer, de uma forma muito especial, à Drª Tânia Almeida por me ter acompanhado ao longo de todos estes meses. Não tenho dúvidas de que sem o empenho e dedicação que demonstrou nada disto teria possível.

Por último, dedico este trabalho ao meu avô Américo Fontes por ter sido sempre um exemplo de coragem, persistência e dedicação. O seu maior sonho era ver o neto a concluir esta fase tão importante na vida de cada individuo e agora sinto que finalmente lhe posso pagar esta dívida.



---

# Resumo

---

O uso de velas de cera é uma prática muito comum em Portugal. Estas são usadas para as mais variadíssimas atividades, mas o tratamento dos resíduos provenientes da queima da vela não são devidamente tratados. Não existindo solução para estes resíduos, eles acabam em aterros poluindo ainda mais o nosso planeta. Neste sentido existem empresas como a Natural Life, Ida que procedem à recolha destes resíduos e posterior reciclagem. Sendo esta uma atividade bastante específica, o uso de um software ERP genérico pode trazer muita ineficiência ao processo. Neste documento é descrito o processo de desenvolvimento de um sistema de gestão dos recursos adequado ao processo da empresa Natural Life.

**Palavras-Chave:** Velas, Cera, NaturalLife, Reciclagem, ERP, Projeto/Estágio, Engenharia de Sistemas, ISEP.



---

# Índice

---

<b>Resumo</b>	<b>iii</b>
<b>Índice</b>	<b>v</b>
<b>Índice de figuras</b>	<b>vii</b>
<b>Índice de tabelas</b>	<b>ix</b>
<b>Abreviaturas</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Enquadramento . . . . .	1
1.2 Apresentação da empresa . . . . .	1
1.3 Objetivos . . . . .	2
1.4 Calendarização . . . . .	2
1.5 Organização do relatório . . . . .	2
<b>2 Contexto</b>	<b>3</b>
2.1 Processo de reciclagem . . . . .	3
2.2 O Problema . . . . .	4
2.3 Estado da arte . . . . .	4
2.3.1 SAP ERP . . . . .	4
2.3.2 PRIMAVERA Executive . . . . .	4
2.3.3 Microsoft Dynamics . . . . .	5
2.4 Comparação das opções disponíveis . . . . .	5
2.5 Analise das opções disponíveis . . . . .	7
2.6 Resumo final das opções tomadas para o desenvolvimento do projeto . . . . .	8
<b>3 Requisitos</b>	<b>9</b>
3.1 Primeiro levantamento de requisitos . . . . .	9
3.2 Estudo do sistema existente . . . . .	10
3.3 Modo de implementação do projeto . . . . .	11
3.4 Último levantamento de requisito . . . . .	11
3.5 Lista final de requisitos . . . . .	12
<b>4 Projeção do Sistema de Informação</b>	<b>15</b>
4.1 Planeamento do projeto . . . . .	15

4.2	Design da nova base de dados . . . . .	15
4.3	Aplicação . . . . .	16
4.4	Casos de utilização do sistema . . . . .	17
4.4.1	Aplicação Fábrica - Menu Inicial . . . . .	18
4.4.2	Aplicação Fábrica - Registo de Ponto . . . . .	19
4.4.3	Aplicação Fábrica - Registo de recolha . . . . .	20
4.4.4	Aplicação Fábrica - Produção . . . . .	21
4.4.5	Aplicação Fábrica - Registo de Produto Acabado . . . . .	22
4.4.6	Aplicação Fábrica - Registo de Saída de Produto Acabado . . . . .	24
4.4.7	Aplicação Fábrica - 2 <sup>a</sup> Via do código de barras . . . . .	26
4.4.8	Aplicação Fábrica - Completar recolha . . . . .	27
4.4.9	Aplicação Painel - Nota introdutória . . . . .	28
4.4.10	Aplicação Painel - Listagem . . . . .	28
4.4.11	Aplicação Painel - Inserir . . . . .	30
4.4.12	Aplicação Painel - Editar . . . . .	31
4.4.13	Aplicação Painel - Apagar . . . . .	32
4.4.14	Aplicação Painel - Desativar/Ativar . . . . .	33
4.4.15	Aplicação Painel - segunda via código de barras . . . . .	34
4.4.16	Aplicação Fábrica - Executar Analise . . . . .	35
4.5	<i>Script</i> de verificação da informação . . . . .	36
4.5.1	Princípio de funcionamento . . . . .	36
4.5.2	Objetivo de execução . . . . .	36
<b>5</b>	<b>Implementação e validação</b>	<b>37</b>
5.1	Desenvolvimento do projeto . . . . .	37
5.1.1	Implementação da primeira fase . . . . .	38
5.1.2	Configuração do servidor e migração de dados . . . . .	38
5.1.3	Primeiro dia de utilização . . . . .	40
5.1.4	Segunda fase de implementação . . . . .	40
5.1.5	Resultado do desenvolvimento . . . . .	40
5.2	Validação do trabalho . . . . .	44
5.3	Execução do projeto . . . . .	45
5.4	<i>Softwares</i> e serviços usados . . . . .	45
<b>6</b>	<b>Conclusões</b>	<b>47</b>
6.1	Resumo do relatório . . . . .	47
6.2	Objetivos realizados . . . . .	47
6.3	Limitações e trabalho futuro . . . . .	47
6.4	Apreciação final . . . . .	47
<b>Referências Bibliográficas</b>		<b>49</b>
<b>A</b>	<b>Documento com as alternativas para o desenvolvimento do projeto entregue à administração</b>	<b>51</b>
<b>B</b>	<b>Stored procedure fim de protocolo</b>	<b>59</b>
<b>C</b>	<b>Stored procedure erros de produção</b>	<b>63</b>
<b>D</b>	<b>Stored procedure erros no registo de ponto</b>	<b>67</b>

---

# Índice de figuras

---

1.1	Planamento - diagrama de Gantt . . . . .	2
2.1	Planta da fábrica . . . . .	3
3.1	Ecrãs da aplicação existe na empresa. . . . .	10
3.2	Exemplo dos códigos de barras gerados . . . . .	11
4.1	Estrutura final da base de dados . . . . .	16
4.2	Diagrama do modelo MVC . . . . .	17
4.3	Modelo do menu . . . . .	18
4.4	Modelo do formulário do registo do ponto . . . . .	19
4.5	Diagrama de sequência marcar ponto . . . . .	19
4.6	Modelo do formulário do registo de recolha . . . . .	20
4.7	Diagrama de sequência registo de recolha . . . . .	20
4.8	Modelo do formulário do registo de produção . . . . .	21
4.9	Diagrama de sequência registo de produção . . . . .	22
4.10	Modelo do formulário do registo de produto acabado . . . . .	22
4.11	Diagrama de sequência registo de produto acabado . . . . .	23
4.12	Modelo do menu . . . . .	24
4.13	Diagrama de sequência registo de produto acabado . . . . .	25
4.14	Modelo do formulário de pedido de 2 <sup>a</sup> via de código de barras . . . . .	26
4.15	Diagrama de sequência de pedido de 2 <sup>a</sup> via de código de barras . . . . .	26
4.16	Modelo do formulário para incrementar o peso de uma recolha . . . . .	27
4.17	Diagrama de sequência para incrementar o peso de uma recolha . . . . .	28
4.18	Modelo da página de listagem . . . . .	28
4.19	Diagrama de sequência da listagem . . . . .	29
4.20	Modelo da janela de inserção na página de listagem . . . . .	30
4.21	Diagrama de sequência de inserir registo . . . . .	30
4.22	Modelo da janela de edição na página de listagem . . . . .	31
4.23	Diagrama de sequência de editar registo . . . . .	31
4.24	Diagrama de sequência de apagar registo . . . . .	32
4.25	Diagrama de sequência de desativar/ativar registo . . . . .	33
4.26	Diagrama de sequência imprimir 2 <sup>a</sup> via do código de barras . . . . .	34
4.27	Modelo da página de resultado de uma análise . . . . .	35
4.28	Diagrama de sequência para executar uma recolha . . . . .	35
4.29	Fluxograma das <i>stored procedures</i> . . . . .	36

5.1	Camadas da página Aplicação Fábrica . . . . .	38
5.2	Menú da Aplicação Fábrica na primeira fase . . . . .	39
5.3	Modelo do menu . . . . .	40
5.4	Registo de ponto . . . . .	41
5.5	Registo de recolha . . . . .	41
5.6	Códigos de barras da recolha . . . . .	41
5.7	Registo de produção . . . . .	42
5.8	Registo de produto acabado . . . . .	42
5.9	Códigos de barras do produto acabado . . . . .	42
5.10	Registo de saída de produto acabado . . . . .	43
5.11	Funcionalidades inexistentes no sistema anterior . . . . .	43
5.12	Exemplo de <i>View</i> listagem . . . . .	43
5.13	<i>View</i> da Aplicação Painel . . . . .	44
5.14	<i>burndown char</i> da excussão do projeto . . . . .	45
A.1	Capa do documento . . . . .	52
A.2	Página 1 . . . . .	53
A.3	Página 2 . . . . .	54
A.4	Página 3 . . . . .	55
A.5	Página 4 . . . . .	56
A.6	Página 5 . . . . .	57

---

# **Índice de tabelas**

---

2.1	Tabela resumo das opções de sistemas já desenvolvidos . . . . .	5
2.2	Tabela resumo das opções para o desenvolvimento . . . . .	7



---

# Abreviaturas

---

Acrónimo	Descrição	Página
ISEP	Instituto Superior de Engenharia do Porto	
DBMG	Database Management System	7
ERP	Enterprise Resource Planning	4
ID	Identifier	10
LES	Licenciatura em Engenharia de Sistemas	2
MVC	Model View Controller	12
NIF	Número de Identificação Fiscal	16
PROES	Unidade Curricular Projeto / Estágio	2
SI	Sistema de Informação	2
UC	Unidade Curricular	2



# **Capítulo 1**

---

## **Introdução**

---

### **1.1 Enquadramento**

Em Portugal o uso de vela de cera é uma prática bastante recorrente. Estas são usadas para iluminação, como elementos decorativos e para o culto ao mortos. Todos os anos milhares de velas são utilizadas em território nacional, nomeadamente círios. Os círios são particularmente frequentes em cemitérios. Constituídos por um envolucro em plástico e com uma tampa em metal são vistos como elemento obrigatório na homenagem a um ente querido falecido. Pela quantidade de velas que são queimadas diariamente os resíduos destas amontoam-se e a solução que é vista para este problema é o colocar o que resta do círio no caixote do lixo indiferenciado. Esta situação acarreta vários problemas ambientais que a médio prazo podem ter consequências mais gravosas, como por exemplo, infestação de vespas e abelhas uma vez que estes animais usam a cera das velas para construir as suas estruturas de abrigo. A reciclagem de resíduos de velas ainda não é neste momento uma atividade com grande expressão no mercado Português. Existem apenas algumas empresas atualmente a fazer a reciclagem deste tipo de resíduos e as suas principais fontes de matéria-prima são os cemitérios e locais de culto religioso, onde o uso de velas é uma prática muito comum. Entre as entidades que operam neste setor podemos mencionar o Santuário de Fátima[1], o Centro Ambiental do Carvalho de Calvos[2] e a Natural Life[3].

### **1.2 Apresentação da empresa**

A Natural Life, Lda. é uma empresa fundada em 2013. Está sediada na rua de Terramonte nº 781, Armazém C20, na Maia. Tendo como atividade a recolha e reciclagem de resíduos de velas, a Natural Life surge como uma empresa ecológica, inteiramente ligada à área ambiental, mais concretamente à reciclagem[3]. A empresa cuida de todo o processo, desde a recolha da matéria-prima nos cemitérios com as quais tem parceria, a separação do corpo de plástico, da tampa de metal e da cera até à fundição da cera. Em alguns casos a recolha do material é substituída por material recolhido por terceiros, mantendo-se o resto do processo. A fábrica está dividida em duas áreas específicas: a zona de armazém, cargas e descargas e a zona de processamento dos círios. Os registos da fábrica são feitos numa base de dados muito simples, com vários tipos de limitações que inibem a empresa de poder expandir.

## 1.3 Objetivos

Implementar um sistema de informação (SI) para efetuar o registo do ponto dos colaboradores, recolhas, produção, acabamento e saída do produto. Esta solução teria de possuir duas áreas distintas, a primeira destinada ao uso na fábrica e a segunda destinada ao uso pela administração. Pretende-se ainda que esta seja uma solução bem integrada no processo da empresa e que traga o leve tempo possível de adaptação ao novo sistema por parte dos funcionários.

## 1.4 Calendarização

Para descrever a calendarização da Unidade Curricular (UC) PROjeto/EStágio (PROES) da Licenciatura em Engenharia de Sistemas (LES).



Figura 1.1: Planamento - diagrama de Gantt

## 1.5 Organização do relatório

No Capítulo 1 é feito o enquadramento do projeto dando uma visão alto nível do projeto. No capítulo 2, é apresentado o estado da arte. É neste capítulo que são descritas as diferentes opções para a realização do projeto e onde são descritas as escolhas feitas e o seu motivo. No capítulo 3 é apresentada a lista de requisitos e do projeto, bem como o processo para a sua obtenção. O capítulo 4, destina-se à projeção do sistema descrevendo o modo como os elementos do sistema se interligam entre si, para no capítulo 5 ser descrito o processo de implementação e a sua validação. No último capítulo, o 6º, são apresentadas as conclusões do projeto. É ainda neste capítulo que é feita a análise crítica do estudante face ao trabalho desenvolvido.

## Capítulo 2

---

# Contexto

---

Neste capítulo é explicado o processo de reciclagem de cera, o problema encontrado na empresa e as opções existentes para a resolução desse problema

### 2.1 Processo de reciclagem

O espaço físico da fábrica está dividido em duas partes, conforme apresentado na figura 2.1.

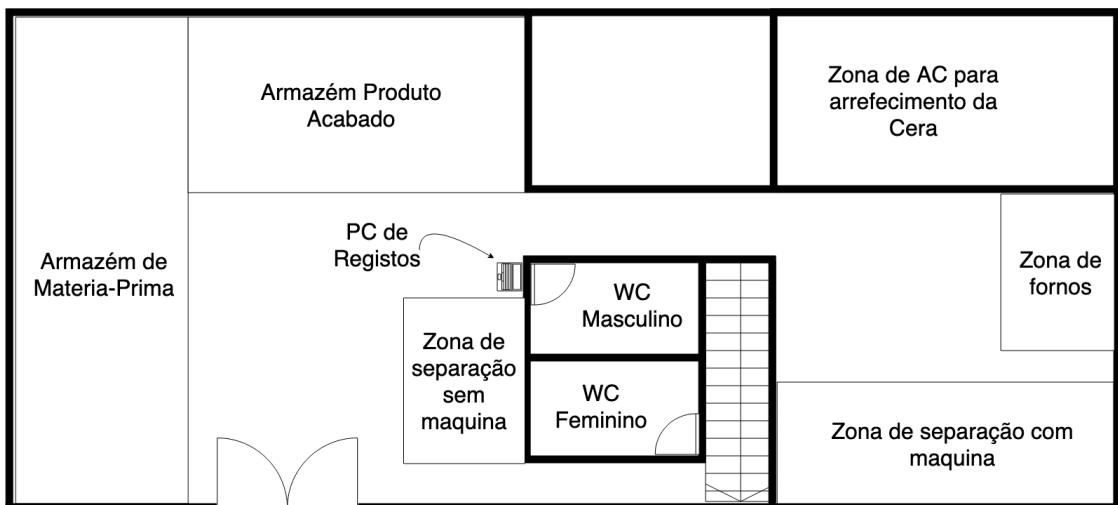


Figura 2.1: Planta da fábrica

Do lado esquerdo onde fica o armazém e onde é recepcionado a matéria-prima e do lado direito fica o espaço onde existe uma máquina que corta o corpo de plástico do círio. Após o corpo do círio, outro colaborador abre o corpo do círio e separa a cera do plástico. A alimentar a máquina está um operador que recebe o corpo do círio sem nenhum elemento metálico, tendo apenas que o colocar devidamente alinhado. Alguns círios, devido à sua forma, não podem ser processados nessa mesma máquina, tornando-se necessário que exista um ponto de separação manual, este fica no lado esquerdo da fábrica, junto à receção do material. Terminada a separação, a cera é levada para o forno. Quando esta termina de ser

derretida é enformada e colocada num espaço de ar condicionado para arrefecer e solidificar. Normalmente este processo é feito ao fim do dia e durante a noite para poder aproveitar a baixa de temperatura ambiental.

## 2.2 O Problema

A recolha da informação era feito por meio de uma base de dados desenvolvida com o software Microsoft Access com auxilio de alguns formulários embutidos na mesma. Aquilo que se previa ser uma solução temporária, acabou por se tornar definitiva pela simplicidade que a interface oferecia aos utilizadores, pela simples integração com outras aplicações Microsoft, como o Microsoft Excel e Microsoft PowerBI, e pela dificuldade de encontrar um sistema ERP comercial com as características da solução temporária, com um custo de aquisição que a empresa pudesse comportar e permitisse obter interfaces simplistas para os utilizadores que já se tinham acostumado com os formulários em Access. No entanto a solução desenvolvida no Microsoft Access é bastante limitada, nomeadamente em executar o mesmo ficheiro em dois computadores ao mesmo tempo. Este constrangimento limita as opções da administração nos seus planos expansão da fábrica ou até fazer gestão da própria empresa durante o período laboral da empresa. Este é ainda o motivo que obriga os colaboradores a deslocar-se vários metros até ao computador para aí fazerem registos. Além do que já foi referido, ainda existe a limitação da base de dados não suportar níveis de acesso, o que quer dizer que qualquer pessoa com acesso ao computador poderá não só ver todos os registos da empresa como modificá-los ou até mesmo eliminar-los. Assim, pretende-se com este projeto de estágio, a implementação de um novo sistema de recolha e consulta de informação, desenvolvido pelo estudante e consequente migração dos dados anteriormente registados.

## 2.3 Estado da arte

Existem varias ferramentas disponíveis para fazer a gestão dos recursos de uma empresa e cada uma com as suas características. Para analisar as opções disponíveis, selecionou-se alguns sistemas integrados de gestão empresarial (Enterprise Resource Planning - ERP) de forma a poder comparar os seus recursos com as necessidades da empresa. De seguida apresentam-se três das soluções estudadas.

### 2.3.1 SAP ERP

O SAP ERP é o produto principal da empresa alemã SAP AG, líder no segmento de software corporativos[4]. O SAP ERP é um sistema integrado de gestão empresarial transacional composto por vários tipos de módulos, cada um responsável por uma parte da atividade da empresa.

### 2.3.2 PRIMAVERA Executive

O PRIMAVERA Executive é um software ERP desenvolvido pela empresa PRIMAVERA Business Software Solutions com foco nas pequenas e médias empresas [5]. Trata-se de uma tecnológica portuguesa que se dedica ao desenvolvimento e comercialização de soluções de gestão e plataformas para integração de processos empresariais [6]. Esta empresa afirmou-se no mercado nacional de soluções informáticas de gestão por ser pioneira no desenvolvimento de aplicações para Windows. [6]. As soluções ERP desta empresa contam ainda integração com os softwares de faturação da própria.

### 2.3.3 Microsoft Dynamics

Microsoft Dynamics é um pacote *software* da Microsoft destinado a gestão corporativa ERP, para ajudar na tomada de decisão e melhoraria dos resultados administrativos e financeiros das empresas[7]. Dentro deste pacote de *software* existem as seguintes aplicações:

- Dynamics 365 for Sales
- Dynamics 365 for Retail
- Dynamics 365 for Finance and Operations
- Dynamics 365 for Talent

Este pacote está disponível em várias edições nas quais varia as funcionalidades disponíveis de modo a se ajustar melhor às necessidades de cada empresa.

## 2.4 Comparação das opções disponíveis

A informação coletada sobre as diferentes opções foi colocada na tabela 2.1

Característica	SAP	Priamvera Executive	Microsoft Dynamics	Solução Própria
Funções de gestão de recursos da empresa	✓	✓	✓	✓
Acesso remoto externo	✓	✓	✓	✗
Suporte técnico de terceiros	✓	✓	✓	✗
Controlo do desenvolvimento	✗	✗	✗	✓
Atualizações do sistema	✓	✓	✓	tem de as desenvolver
Adaptação total do sistema às necessidades da empresa	✗	✗	✗	✓
Implementação apenas dos recursos necessário	✗	✗	✗	✓
Compatível com os planos de investimento a curto e médio prazo	✗	✗	✗	✓
Curto período de adaptação dos utilizadores ao novo sistema	✗	✗	✗	✓

Tabela 2.1: Tabela resumo das opções de sistemas já desenvolvidos

No final desta análise, o caminho definido pela administração da Natural Life foi a criação de uma nova plataforma. Por esse motivo foi feita uma nova análise de opções à cerca do desenvolvimento da plataforma. Este documento foi entregue à administração para que o analisa-se e decidisse o modo como a aplicação deveria ser desenvolvida. O documento entregue esta presente no anexo A. Desse documento foi extraída a tabela comparativa das opções para o desenvolvimento da plataforma. Esta é apresentada na tabela 2.2.

	Opção	Vantagens	Desvantagens
Tipo de Aplicação	Desktop		Necessidade de configurar cada computador que receber a aplicação Parar o computador para atualizar a aplicação Necessidade de um servidor de base de dados
	Web	Acesso direto em qualquer computador  Maior familiaridade com a plataforma por parte do programador Menos pontos de falha	Necessidade de um servidor web e de base de dados
Sistema Operativo	Windows 10	Familiaridade com o sistema  Todas as máquinas já executam esta plataforma	Necessidade de comprar uma licença do Windows Server Necessidade de aprender ASP.NET ou usar o XAMPP Se não for usado só software Microsoft não há garantias de segurança/compatibilidade com o Windows
	Ubuntu 18.04	Plataforma líder de mercado  Milhares de pacotes no repositório que são revisados pela Canonical	Curva de aprendizagem para manutenção básica
Sistema de Gestão de Base de Dados	Microsoft SQL Server	Familiaridade com o software Versão Express gratuita	Versão Express limitada. Versões Enterprise e Standard pagas Versão Developer não utilizável em produção
	MariaDB	Gratuito (versão Open Source)  Totalmente compatível com MySQL da Oracle  Padrão no Ubuntu e no XAMPP Escalável	Não possui um ficheiro único para a base de dados. Tem de ser instalado manualmente no Windows ou usado com recurso ao XAMPP

		Fiável Recursos semelhante ao MS SQL Server Suporte nativo no Ubuntu	
Tipo de Máquina	Real	Não dependência da operação de outras máquinas	Hardware dedicado
	Virtual	<p>Não necessita de hardware dedicado.</p> <p>Várias instâncias da máquina ao mesmo tempo.</p> <p>Backup muito fácil.</p>	<p>Exige que o utilizador que esta a executar a VM nunca termine a sessão.</p> <p>Se a máquina real tiver de ser reiniciada, a VM tem de ser interrompida</p> <p>Partilha dos recursos da máquina hospedeira com a maquina virtual.</p>

Tabela 2.2: Tabela resumo das opções para o desenvolvimento

## 2.5 Analise das opções disponíveis

O documento entregue incide sobre os aspectos base para a criação do sistema de informação: tipologia de aplicação, servidor, linguagens de programação, sistema de gestão de base de dados.

Primeiro definiu-se a tipologia de aplicação. Decidir sobre uma aplicação *web* ou uma aplicação *desktop* iria condicionar o trabalho futuro. Aqui a sugestão passou pelo desenvolvimento de uma aplicação *web*, pois esta seria agnóstica de equipamento o que traria uma grande flexibilidade no futuro. Além deste fator, já se sabia que teria de existir uma servidor porque a base de dados seria centralizada, logo não havia nenhum gasto extra com esta solução. Ficou desde logo decidido que seria o servidor seria uma máquina física com o sistema operativo Ubuntu 18.04

Ultrapassada esta fase, foram analisadas as linguagens de programação e optou-se pela linguagem PHP para o *backend* e JavaScript para o *frontend*. Ficou ainda decidido que de modo a tornar o desenvolvimento mais ágil e dar robustez ao serviço, devia ser utilizado o *framework* Laravel. Esta escolha assentou no facto deste *framework* já ser bastante utilizado, inclusivamente é tido como escolha de referência para projetos de missão crítica em PHP pelas suas características de segurança[8].

Por fim foi definido o sistema de gestão de base de dados (DataBase Management System - DBMS). A opção opção recaiu sobre o Microsoft SQL Server apesar das vantagens enunciadas em relação ao MariaDB.

Dois pontos que não foi considerados neste documento foram o sistema e o serviço de *versionamento*. A administração optou por deixar ao estudante das LES desde que o houvesse a garantia do repositório ser privado, uma vez que tirando o facto de assim ser esta opção em nada influenciava o sistema a ser implementado. Assim o sistema de versionamento escolhido foi o Git no serviço GitHub.

## 2.6 Resumo final das opções tomadas para o desenvolvimento do projeto

No final de todo este processo ficou decidido construir uma plataforma própria em formato de aplicação Web. Esta aplicação web seria construída em PHP, com o framework Laravel, para o backend e JavaScript frontend. O sistema de gestão de base de dados escolhido foi o Microsoft SQL Server na edição Express e o servidor seria uma máquina física com o sistema operativo Ubuntu 18.04. Para fazer o versionamento do código foi utilizado um repositório privado no serviço GitHub.

## **Capítulo 3**

---

# **Requisitos**

---

Neste capítulo é apresentado todos os levantamentos de requisitos feitos, o estudo do sistema anterior e a lista de requisitos final.

### **3.1 Primeiro levantamento de requisitos**

Antes de iniciar qualquer tipo de desenvolvimento ou implementação, era fulcral também definir uma lista de requisitos, que descrevesse de uma forma muito exata e clara o que se esperava que a solução implementada fosse. Logo na primeira reunião foi entregue um primeira lista de requisitos dividida em dois tipos diferentes: os primeiros são designados por Requisitos Absoluto que são requisitos com um grande nível de detalhe sobre as suas características e perspetivas. Os segundos designamos por Requisitos Indexados, sobre os quais não se tem certeza sobre a totalidade das suas características ou dependem da informações externas. Os requisitos pertencentes a este último grupo devem passar por um processo de definição de modo a que se tornem também requisitos absolutos. No final da primeira reunião a lista de requisitos entregue foi a seguinte

Requisitos absolutos:

- A aplicação só poderia ser acessível da rede interna;
- Uso de uma base de dados relacional;
- Registo do horário de entrada e saída dos colaboradores;
- Registo do peso e ponto de recolha de onde vinha a matéria prima;
- Registo do peso de cera, metal e plástico de uma produção, bem como o colaborador associado;
- Registo do peso do produto final acabado;
- Registo da saída de produto acabado e cliente a quem foi vendido;
- Impressão de uma segunda via dos códigos de barras já impressos referente a uma recolha ou um produto acabado;

- Incremento do peso de uma recolha efetuada anteriormente;
- Zona protegida por IDentifier (ID) e Password para a consulta dos registos feitos;
- Possibilidade de editar e apagar registos já feitos;
- Script de analise da coerência dos dados registados.

Requisitos indexados:

- Deveria ocorrer uma melhoria do *design* da base de dados;
- A aplicação deveria ser acessível em dispositivos moveis e computadores;
- Desenvolvimento de uma funcionalidade que permitisse o incremento do peso de uma recolha.

Terminada esta reunião foi iniciado um processo de estudo do sistema atualmente implementado, o funcionamento da fábrica de forma a desindexar os requisitos indexados e determinar novos requisitos por forma a ter a melhor descrição possível do sistema a ser implementado.

### 3.2 Estudo do sistema existente

O sistema existente na empresa foi construído com recurso ao Microsoft Access. Era composto por 5 ecrãs destintos utilizados para recolher a informação gerada na fábrica. Um dos aspetos evidentes desde inicio é a não-coesão visual dos elementos, como demonstrado na figura 3.1.



Figura 3.1: Ecrãs da aplicação existente na empresa.

O fluxo dentro deste sistema era bastante simples. Ao executar o ficheiro Microsoft Access, era apresentado diretamente o menu ao utilizador. Este selecionaria a opção referente ao registo que pretendia fazer e uma nova janela surgia com o formulário correspondente.

O utilizador preenchia com os dados requeridos e pressionava o botão "Gravar". Em caso de sucesso não era apresentado nenhuma mensagem. Em caso de erro era apresentada uma mensagem com essa informação. Estas mensagens eram geradas diretamente pelo sistema de gestão de base de dados e não eram intuitivas. No caso de ser o registo de recolha ou de produto acabado, era ainda apresentado, numa nova janela um código de barras que seria impresso para identificar o elemento registado. Um exemplo do resultado impresso é demonstrado na figura 3.2.



(a) Recolha

(b) Produto Acabado

Figura 3.2: Exemplo dos códigos de barras gerados

Estes códigos de barras deveriam ter estruturas diferentes. O código de barras referente à recolha (figura 3.2 (a)) deveria ser constituído pelo código de barras correspondente ao ID da recolha, o ID do ponto de recolha e a data da recolha. Já no caso produto acabado era apenas apresentado o código de barras referente ao ID do mesmo (figura 3.2 (b)). A pedido da administração deveria passar a ser apresentado também o ID do produto acabado em formato numérico.

Após o estudo da aplicação foi iniciado o estudo da base de dados. Este estudo consistiu em compreender as tabelas que constituem base de dados, as suas relações e a informação que nelas era registada. Foi também solicitado pela empresa sugestões de melhoria do *design* da base de dados com foco na coerência da informação registada.

### 3.3 Modo de implementação do projeto

Numa das reuniões discutiu-se o modo como o novo sistema deveria ser implementado. No dia em que ocorresse a primeira implementação do novo sistema, este deveria substituir por completo a aplicação construída no Microsoft Access. Isto porque devido à natureza da informação registada e tendo em conta que a base de dados teria a sua estrutura modificada, não era possível existir um período híbrido onde as duas aplicações coexistiram. Assim ficou definido que enquanto o novo sistema não fosse capaz de registar o ponto dos colaboradores, as recolhas efetuadas, as produções feitas, o produtos finalizados e a sua saída, a implementação não iria ocorrer. Foi definido então que o sistema teria de passar por dois níveis de testes sendo o primeiro os testes de desenvolvimento e o segundo o teste com utilizadores reais de forma a identificar erros não previstos no período de desenvolvimento.

### 3.4 Último levantamento de requisito

Todos os requisitos da aplicação ficaram definidos no final da primeira semana, à excepção dos requisitos referentes a uma nova funcionalidade que permitisse o incremento do peso de uma recolha. Em algumas recolhas efetuadas, o peso total da recolha superava a capacidade

de ser pesado de uma só vez, seja pelo peso de matéria prima seja pela quantidade. Apesar de estarmos conscientes desta necessidade foi sugerido deixar a discussão sobre esta nova funcionalidade para o pós primeira implementação. Não se tratava de uma funcionalidade fundamental para o sistema e como tal havia a intenção de estudar a melhor forma de implementar esta solução e que apenas foi discutida na oitava semana do período de estágio. Consistiu na criação de uma nova tabela "Completar Recolha" que iria armazenar o histórico de incrementos feitos numa recolha, para assim poder identificar eventuais erros no registos da informação. Do ponto de vista da aplicação a recolha a ser incrementada seria identificada pelo código de barras impresso anteriormente e deveria ser apresentado ao utilizador o histórico de incrementos referente a essa recolha. Por fim definiu-se que uma recolha registada com o peso de zero quilos deveria redirecionar automaticamente o utilizador para o ecrã do incremento desta recolha após a impressão do primeiro código de barras.

### 3.5 Lista final de requisitos

Após as todas as reuniões de levantamento de requisitos, chegou-se à seguinte lista composta apenas por requisitos absolutos.

- A aplicação só poderia ser acessível da rede interna
- A aplicação alojada num servidor físico com Ubuntu Server 18.04
- A aplicação WEB desenvolvida em PHP, com o framework Laravel, e em JavaScript;
- A aplicação teria de seguir o modelo *Model View Controller* (MVC);
- A aplicação deve ser construída de forma modular de forma a facilitar a sua manutenção no futuro;
- Uso de uma base de dados relacional, construída com o sistema de gestão de base de dados Microsoft SQL Server e a estrutura aprovada pela administração;
- A implementação teria de ser feita em duas fases:
  - a primeira onde se substituía totalmente a aplicação existente na fabrica (Registo de ponto, de recolhas, de produção, de produto acabado e de saída de produto acabado);
  - a segunda fase durante a qual as novas funcionalidades seriam aplicadas incrementalmente;
- A aplicação teria de passar por uma fase de testes muito exigente para impedir erros que obrigassem ao uso da base de dados antiga.
- Os testes têm de ser divisos em dois níveis:
  - Nível 1: testes de desenvolvimento;
  - Nível 2: testes com utilizadores reais;
- O sistema tem de ser graficamente coeso;
- O sistema terá de ter um tempo de aprendizagem mínimo no modo de utilização dos funcionários;

- O sistema deverá emitir mensagens após cada ação;
- As mensagens têm de ter linguagem simples e direta;
- O registo do horário de entrada e saída dos colaboradores;
- O registo do peso e ponto de recolha de onde vinha a matéria prima;
- Impressão de um código de barras com o ID gerado para a recolha inserida, ID do ponto de recolha e data de recolha;
- O sistema deve permitir de incrementar o peso de uma recolha;
- Se a recolha fosse iniciada com 0 kg é iniciado automaticamente o processo de incremento do peso da recolha que acabou de ser registada;
- Após o incremento do peso deveria ser impresso uma segunda via do código de barras dessa mesma recolha;
- Registo do peso de cera, metal e plástico de uma produção, bem como o colaborador associado;
- Registo do peso do produto final acabado;
- Impressão de um código de barras com o ID gerado para a produto final acabado;
- Registo da saída de produto acabado e cliente a quem foi vendido;
- Impressão de uma segunda via dos códigos de barras já impressos referente a uma recolha ou um produto acabado;
- Zona protegida por ID e *Password* para a consulta dos registos feitos;
- As tabelas apresentadas na interface da aplicação teriam de ser capazes de refletir alterações na estrutura das tabelas da base de dados;
- O sistema deve permitir listar, inserir, editar, apagar e emitir uma segunda via de código de barras de uma *recolha*;
- O sistema deve permitir listar, inserir, editar e apagar uma *produção*;
- O sistema deve permitir listar, inserir, editar, apagar e emitir uma segunda via de código de barras de um *produto acabado*;
- O sistema deve permitir listar, inserir, editar, apagar e desativar um *colaborador*;
- O sistema deve permitir listar, inserir, editar e apagar um *registo de ponto*;
- O sistema deve permitir listar, inserir, e apagar um *utilizador*;
- O sistema deve permitir listar, inserir e editar uma *entidade*;
- O sistema deve permitir listar, inserir, editar, apagar e desativar um *ponto de recolha*;
- O sistema deve permitir listar, inserir, editar, apagar e desativar um *cliente*;
- O sistema deve permitir listar, inserir, editar, apagar e executar uma *análise personalizada*;
- *Script* de análise da coerência dos dados registados para Pontos de Recolha, Registos de Ponto e Produção.



## Capítulo 4

---

# Projeção do Sistema de Informação

---

Neste capítulo são abordados o planeamento do projeto, *design* da nova base de dados, a estrutura da aplicação e os casos de utilização do sistema.

### 4.1 Planeamento do projeto

A primeira semana do período do estágio foram destinadas ao levantamento de requisitos. Foi ainda durante esta semana que ocorreu o *redesign* da base de dados. As semanas 2 e 3 foram destinadas ao estudo dos *frameworks* a utilizar, do sistema de gestão de base de dados e do ambiente da empresa. Conforme indicado nos requisitos do projeto, o desenvolvimento do sistema de informação teria de ser dividido em duas fases. A data para a conclusão da primeira fase foi fixada no inicio da sexta semana, dando-se logo incio à fase dois.

### 4.2 Design da nova base de dados

A base de dados a ser utilizada neste projeto teve como ponto de partida aquela que já era utilizada pela aplicação antiga. De modo a tornar a estrutura o mais adequada possível, numa da reuniões, foi solicitado à administração que descrevesse em detalhe o fluxo de informação da estrutura até à data implementada. Após a reunião seguiu-se um estudo para encontrar formas de otimizar a estrutura da base de dados, não apenas na questão da normalização da estrutura mas também de forma a tornar o trabalho futuro mais simples procurando meios de otimizar o fluxo da informação. Após o estudo foi entregue de uma proposta da nova estrutura. Esta não foi aceite numa fase inicial porque, como resultado da normalização, foi criada uma tabela destinada a armazenar os cemitérios com os quais a empresa tem parceria, uma segunda tabela onde ficaram registadas as empresas a quem a Natural Life compra o matéria-prima e uma terceira tabela responsável por agregar os registos das primeiras duas de modo a estabelecer as relações com a restantes tabelas do sistema. A resposta da administração foi a de que preferia que existisse apenas uma única tabela, que agregasse os campos das três referidas anteriormente, com um campo extra para distinguir o tipo de registo. Apesar de compreenderem qual era o objetivo da mudança sugerida, esta iria tornar mais difícil a adaptação de ficheiros Excel e PowerBi já existentes. A mudança pedida foi efetuada e a administração aceitou as restantes alterações sugeridas. Dentre dela destaca-se a tabela *Entidades* criada para armazenar as informações de identificação (*Nome*,

*Morada, Código Postal e Número de Identificação Fiscal (NIF)*) dos pontos de recolha e clientes. Esta alteração permitiu não só reunir num único local a lista de entidades com as quais a Natural Life interage, eliminando eventuais incoerências de informação a respeito do NIF, por exemplo. O processo de reforma da base de dados manteve-se então congelado até à sétima semana, na qual se criou a tabela *completar\_recolha*, conforme descrito no capítulo anterior. No final de todo este processo a estrutura final da base de dados, aprovada pela administração, é a descrita na figura 4.1.

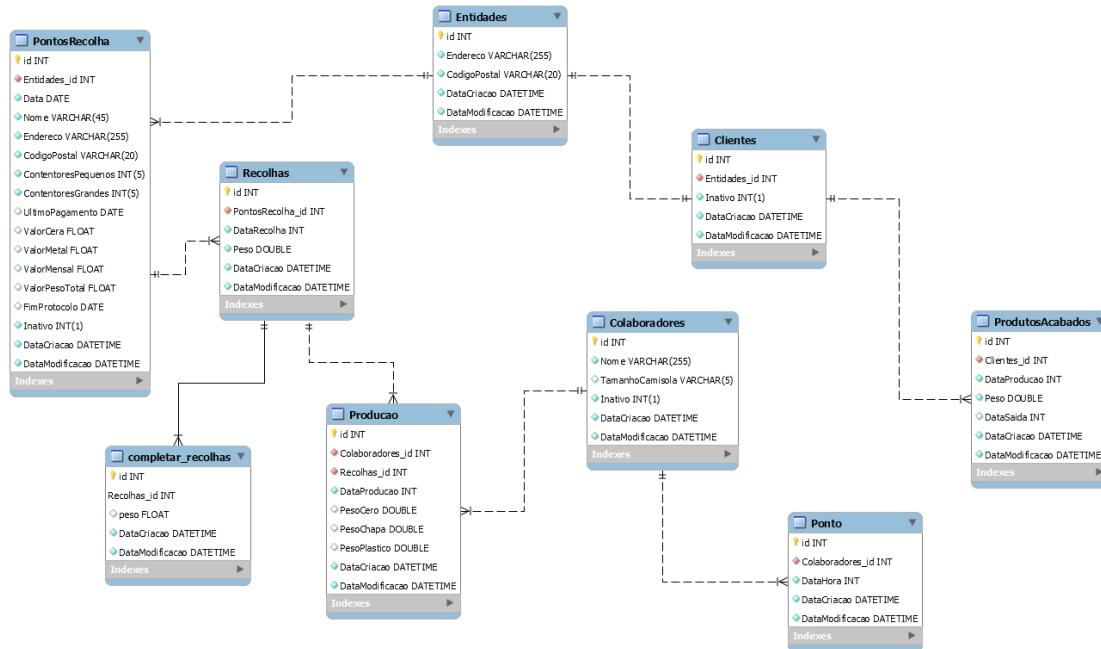


Figura 4.1: Estrutura final da base de dados

Finalizado este processo, iniciou-se o processo de *design* da aplicação.

### 4.3 Aplicação

Respondendo aos requisitos definidos no Capítulo 3, a aplicação a ser desenvolvida seria uma aplicação *web* construída em PHP, com o framework Laravel, e JavaScript segundo o padrão do modelo MVC. O modelo MVC, representado na figura 4.2, descreve a forma como uma aplicação deve ser construída separando-a em três camadas distintas: *Model View e Controller*.

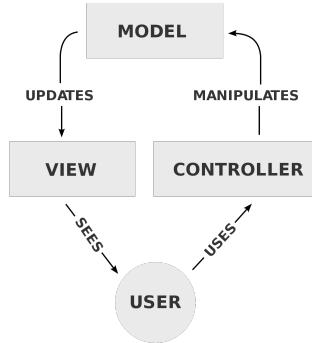


Figura 4.2: Diagrama do modelo MVC

Esta separação serve distinguir representações de informação internas dos modos como a informação é apresentada ao utilizador [9]. A abstração em camadas que o modelo oferece é particularmente vantajosa no que toca a fazer reutilização de código, pois cada classe tem apenas uma responsabilidade atribuída. Além de promover o baixo acoplamento do projeto, desta forma é muito fácil utilizar a mesma classe em partes distintas do projeto, facilitando a sua compreensão, manutenção e atualização. Estas características do modelo servem ainda de resposta a outros requisitos do projeto como modularidade do projeto. Ainda respondendo aos requisitos, deveria existir duas áreas distintas dentro da aplicação destinada aos colaboradores na fábrica e à administração. Estas suas sub-aplicações foram designadas de Aplicação Fábrica, acessível aos colaboradores da empresa e responsável pelos registos de informação na base de dados, e a Aplicação Painel, protegida por um sistema de autenticação na qual seria possível manipular toda a informação registada. Para fazer esta separação, os endereços dentro da aplicação foram divididos em dois grupos. Em primeiro lugar as páginas associadas à Aplicação de Fábrica estavam sob o endereço /Fabrica, i.e

`http://<ip.do.servidor>/Fabrica/<Nome_da_página_solicitada>`

enquanto as páginas relacionadas com a Aplicação Painel estavam sob o endereço /Painel, i.e.

`http://<ip.do.servidor>/Painel/<Nome_da_página_solicitada>`

A única exceção a esta regra está no diretório raiz do sistema, que direciona para a página index da Aplicação de Fábrica.

Esta organização em nada muda a estrutura de ficheiros do projeto. O framework Laravel, trás consigo um recurso de rotas, que permite facilmente que este tipo de regras seja incluída no projeto sem ter necessariamente que fazer modificações à árvore de diretórios da aplicação. Desta forma, manteve-se uma estrutura dos ficheiros da aplicação coesa, mantendo cada tipo de ficheiro na sua pasta definida e manter regras de padronização dos endereços da aplicação.

## 4.4 Casos de utilização do sistema

Definidos todos os detalhes supracitados, é possível dar inicio à definição de cada um dos casos de uso do projeto. Esta definição serve como guia de desenvolvimento, mantendo as informações necessárias para se conhecer o comportamento esperado de cada um dos casos de uso do sistema final e é explicada nas próximas páginas.

#### 4.4.1 Aplicação Fábrica - Menu Inicial

##### Descrição do caso de uso

No menu, espera-se que este seja muito simples e intuitivo. Deve conter botões que indiquem de forma inequívoca qual a qual funcionalidade dão acesso. As cores das opções já existentes na aplicação construída no Microsoft Access deverão ter cores similares.

Na primeira fase do projeto este menu deve apenas contemplar as opções existentes na aplicação anterior, tal como descrito na figura 4.3 a.

No final da segunda fase, espera-se que o menu inicial apenas acrescente dois botões no final, conforme demonstrado na figura 4.3 b

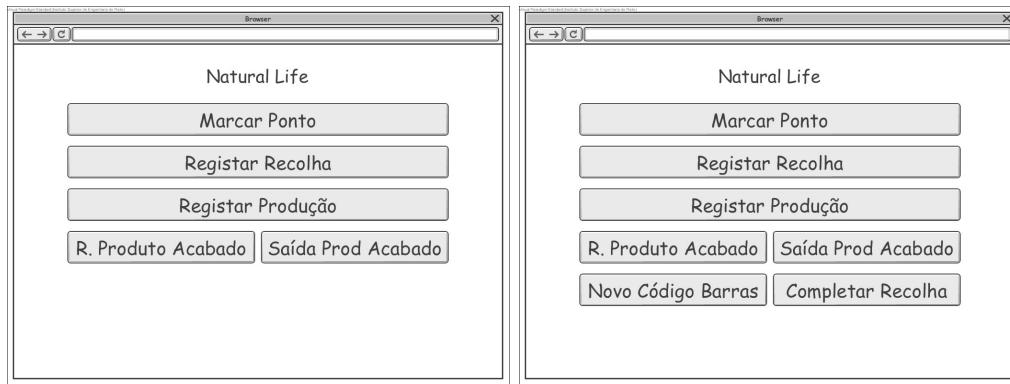


Figura 4.3: Modelo do menu

#### 4.4.2 Aplicação Fábrica - Registo de Ponto

##### Descrição do caso de uso

No registo do ponto, espera-se que utilizador entre na página e indique apenas o seu ID. A informação sobre a data e hora do registo deve ser automaticamente definida pelo sistema. A aparência da *view* deste caso de utilização será semelhante ao demonstrado na figura 4.4.



Figura 4.4: Modelo do formulário do registo do ponto

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do registo de ponto. É apresentado o formulário com a data e hora previamente preenchidas. O utilizador tem de indicar o seu ID numa lista de *dropdown*. Após indicar o seu ID precisa o botão *Entrada* ou *Saída* conforme se esta a indicar o horário de entrada ou saída, respetivamente. Ambos os botões executam a submissão do formulário. Após o registo é apresentada uma mensagem ao utilizador, tal como demonstrado na figura 4.5.

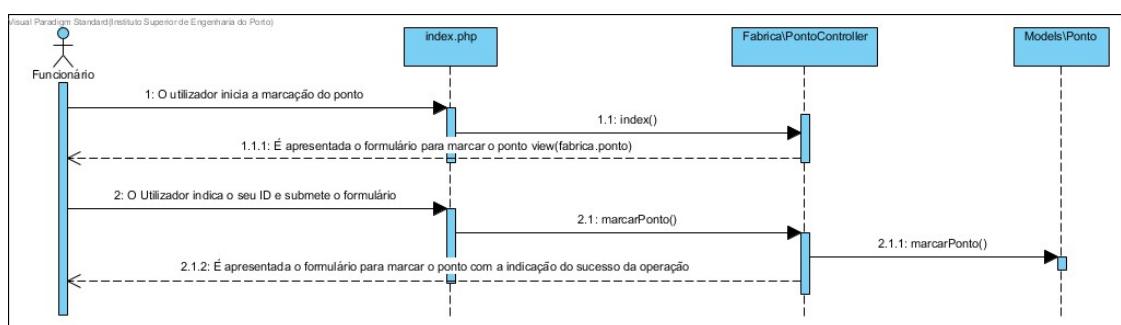


Figura 4.5: Diagrama de sequência marcar ponto

#### 4.4.3 Aplicação Fábrica - Registo de recolha

##### Descrição do caso de uso

No registo de recolha, espera-se que utilizador entre na página e indique o ID do ponto de recolha e o peso da recolha. A informação da data deve ser indicada automaticamente pelo sistema. A aparência da *view* deste caso de utilização será semelhante ao demonstrado na figura 4.6.

Figura 4.6: Modelo do formulário do registo de recolha

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do registo de recolha. É apresentado o formulário com a data previamente preenchida. O utilizador tem de indicar o ID do ponto de recolha numa lista de *dropdown* e o peso da recolha feita. Após indicar as informações solicitadas precisa o botão *Guardar*. No final do registo é apresentada uma mensagem ao utilizador. Caso o registo seja feito com sucesso um novo separador é aberto com o código de barras para ser impresso, tal como demonstrado na figura 4.7.

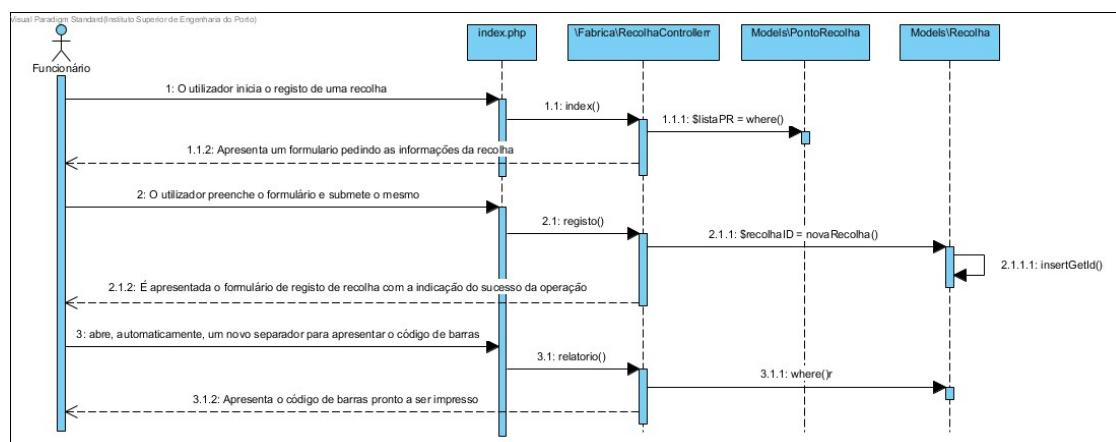


Figura 4.7: Diagrama de sequência registo de recolha

#### 4.4.4 Aplicação Fábrica - Produção

##### Descrição do caso de uso

No registo de produção, espera-se que utilizador entre na página e indique o código de barras da recolha, o peso de cera, metal e plástico e o seu ID. A informação da data de produção deve ser indicada automaticamente pelo sistema e a informação sobre a recolha (ponto de recolha e data da recolha) seja obtido, em *background*, após indicar o código de barras da recolha. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.8.

Figura 4.8: Modelo do formulário do registo de produção

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do registo de produção. É apresentado o formulário com a data de produção previamente preenchida. O utilizador tem de indicar o código de barras da recolha, peso de cera, metal e plástico e o seu ID numa lista de *dropdown*. Quando o utilizador termina de indicar o código da recolha é feito um *request* ao servidor para saber o ponto de recolha e data da recolha. Essa informação é apresentada automaticamente ao utilizador. Após indicar as informações solicitadas precisa-se o botão "Guardar". No final do registo é apresentada uma mensagem ao utilizador, tal como demonstrado na figura 4.9.

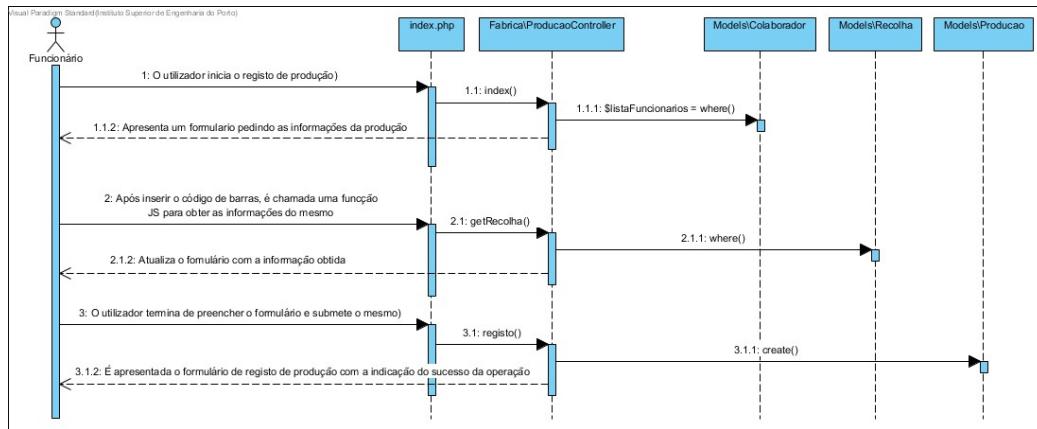


Figura 4.9: Diagrama de sequência registo de produção

#### 4.4.5 Aplicação Fábrica - Registo de Produto Acabado

##### Descrição do caso de uso

No registo de produto acabado, espera-se que utilizador entre na página e indique o peso do produto acabado. A informação da data deve ser indicada automaticamente pelo sistema. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.10.



Figura 4.10: Modelo do formulário do registo de produto acabado

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do registo de produto acabado. É apresentado o formulário com a data previamente preenchida. O utilizador tem de indicar o peso do produto acabado. Após indicar as informações solicitadas precisa o botão *Guardar*. No final do registo é apresentada uma mensagem ao utilizador. Caso o registo seja feito com sucesso um novo separador é aberto com o código de barras para ser impresso, tal como demonstrado na figura 4.11.

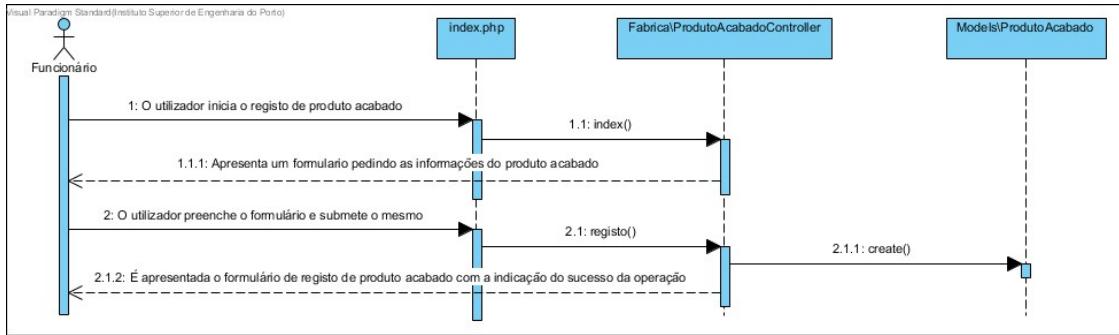


Figura 4.11: Diagrama de sequência registo de produto acabado

#### 4.4.6 Aplicação Fábrica - Registo de Saída de Produto Acabado

##### Descrição do caso de uso

No registo de produto acabado, espera-se que utilizador entre na página e indique o código de barras do produto acabado. A informação da data deve ser indicada automaticamente pelo sistema. A partir da segunda fase do projeto deverá ainda existir um campo para indicar o cliente. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.12. A figura 4.12 (a) descreve como será a interface após concluir a primeira fase do projeto e a figura 4.12 (b) descreve como será a interface após concluir a segunda fase.

(a) Após concluir primeira fase

(b) Após concluir segunda fase

Figura 4.12: Modelo do menu

##### Fluxo do caso de utilização (2<sup>a</sup> fase)

O caso de uso inicia-se com a abertura da página do registo de saída de produto acabado. É apresentado o formulário com a data previamente preenchida. O utilizador tem de indicar o código de barras do produto acabado e o ID do cliente de uma lista *dropdown*. Após indicar as informações solicitadas precisa o botão "Guardar". No final do registo é apresentada uma mensagem ao utilizador. Caso o registo seja feito com sucesso um novo separador é aberto com o código de barras para ser impresso, tal como demonstrado na figura 4.13.

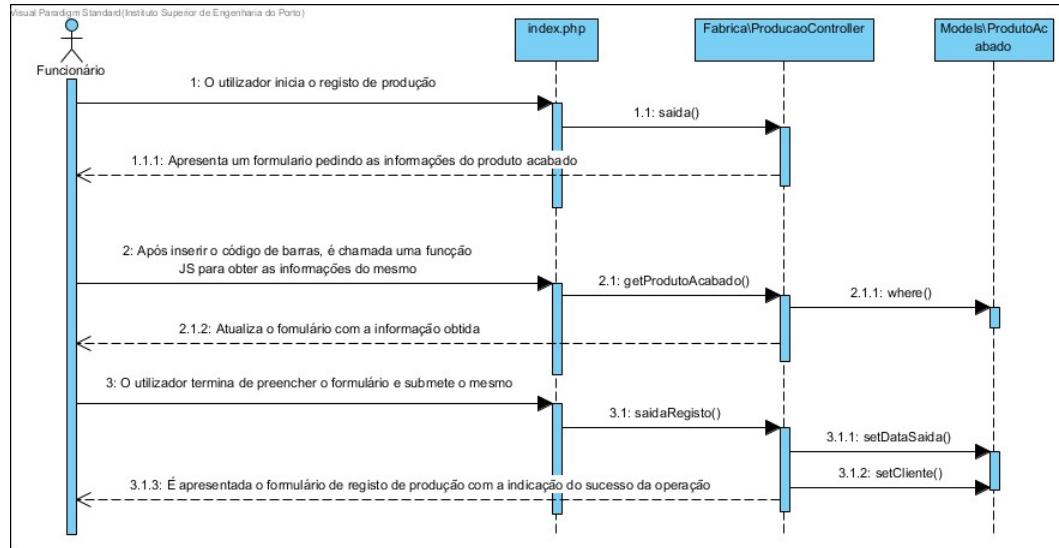


Figura 4.13: Diagrama de sequência registo de produto acabado

#### 4.4.7 Aplicação Fábrica - 2<sup>a</sup> Via do código de barras

##### Descrição do caso de uso

Na obtenção de uma segunda via de um código de barras, espera-se que utilizador entre na página e indique o tipo de código de barras e o código de barras que pretende. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.14.

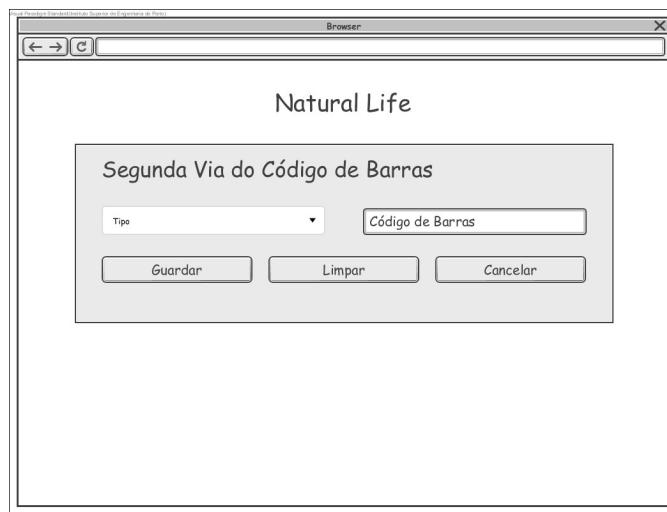


Figura 4.14: Modelo do formulário de pedido de 2<sup>a</sup> via de código de barras

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do pedido de segunda via de código de barras. É apresentado para o utilizador indicar o tipo de código de barras e o código de barras que pretende. Após indicar as informações solicitadas precisa o botão *Guardar*. É aberto um novo separador com o código de barras solicitado para imprimir, tal como demonstrado na figura 4.15.

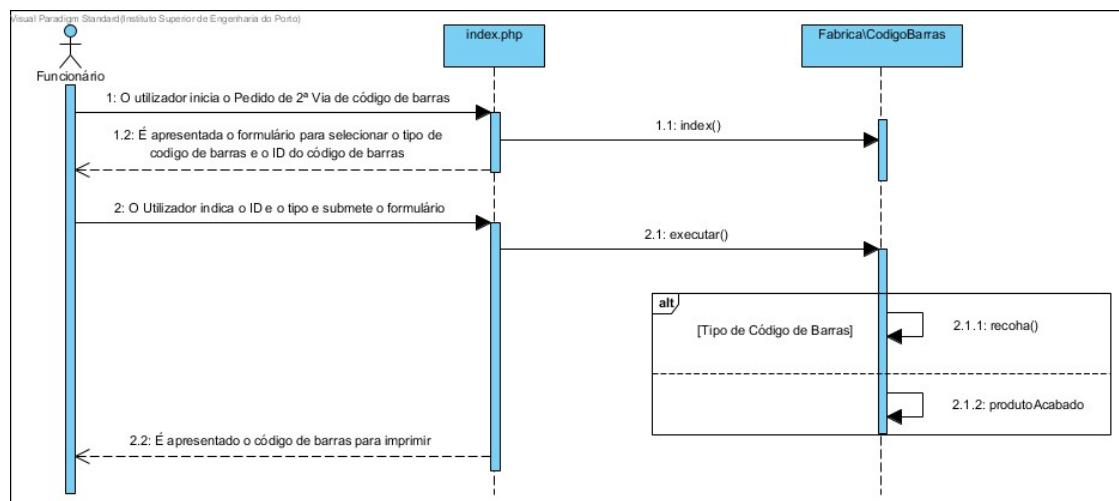


Figura 4.15: Diagrama de sequência de pedido de 2<sup>a</sup> via de código de barras

#### 4.4.8 Aplicação Fábrica - Completar recolha

##### Descrição do caso de uso

Para completar recolha, espera-se que utilizador entre na página e indique o ID do ponto de recolha e o peso que pretende acrescentar à recolha. A informação sobre o peso total já registado bem como o histórico de incrementos será apresentado automaticamente pelo sistema. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.17.

Figura 4.16: Modelo do formulário para incrementar o peso de uma recolha

##### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página do registo de recolha. É apresentado o formulário com a data previamente preenchida. O utilizador tem de indicar o ID do ponto de recolha numa lista de *dropdown* e o peso da recolha feita. Após indicar as informações solicitadas precisa o botão *Guardar*. No final do registo é apresentada uma mensagem ao utilizador. Caso o registo seja feito com sucesso um novo separador é aberto com o código de barras para ser impresso, tal como demonstrado na figura 4.17.

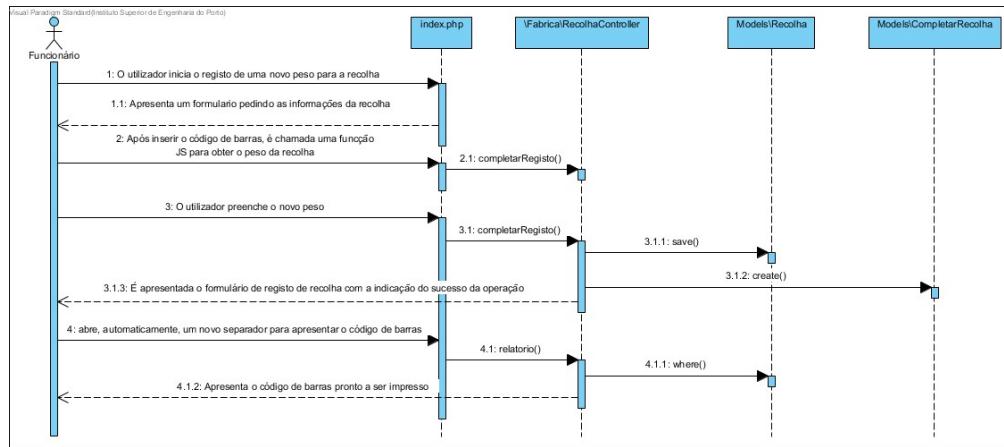


Figura 4.17: Diagrama de sequência para incrementar o peso de uma recolha

#### 4.4.9 Aplicação Painel - Nota introdutória

Os casos de utilização da Aplicação Painel têm comportamento semelhante ao descrito, independentemente do *model* ao qual está associado, pelo que nos abstemos de os apresentar individualmente.

#### 4.4.10 Aplicação Painel - Listagem

##### Descrição do caso de uso

Para fazer uma listagem, utilizador apenas necessita de entrar na página. A aparência da *view* deste caso de utilização será semelhante ao demonstrado na figura 4.18.

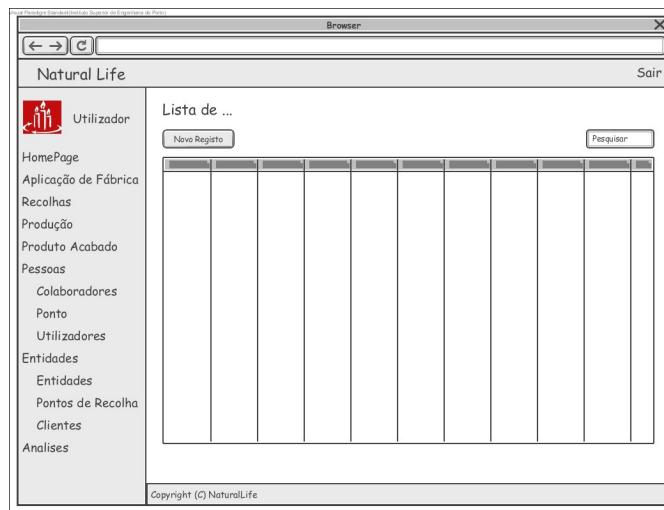


Figura 4.18: Modelo da página de listagem

##### Models compatíveis com o caso de uso

Todos os models são compatíveis com este caso de uso

### Fluxo do caso de utilização

O caso de uso inicia-se com a abertura da página de listagem. É apresentado uma tabela com os dados listado e algumas opções para cada linha, tal como demonstrado na figura 4.19.

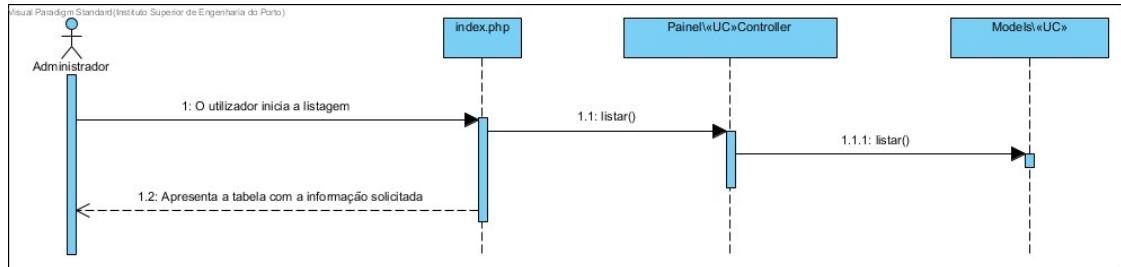


Figura 4.19: Diagrama de sequência da listagem

#### 4.4.11 Aplicação Painel - Inserir

##### Descrição do caso de uso

Para inserir um registo, o utilizador necessita pressionar o botão *Novo Registo*. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.20.

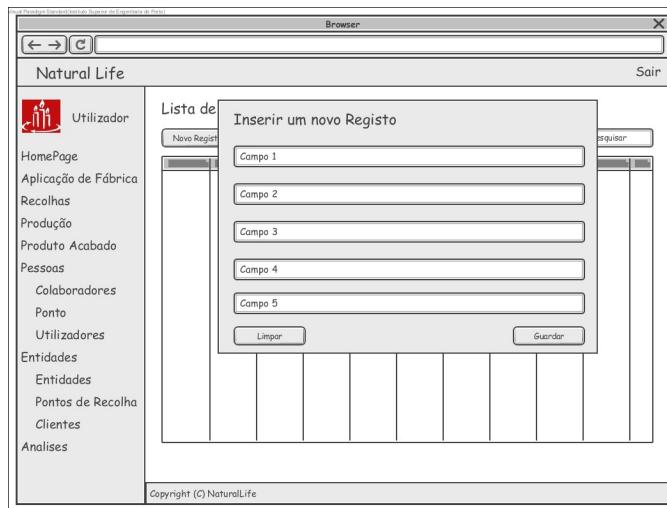


Figura 4.20: Modelo da janela de inserção na página de listagem

##### Models compatíveis com o caso de uso

Todos os *models* são compatíveis com este caso de uso

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão novo registo na página de listagem. É apresentado uma janela flutuante com os campos a serem preenchidos. O utilizador preenche os campos e pressiona o botão guardar. Finalizado o registo a janela fecha-se e é apresentado uma mensagem ao utilizador, tal como demonstrado na figura 4.21.

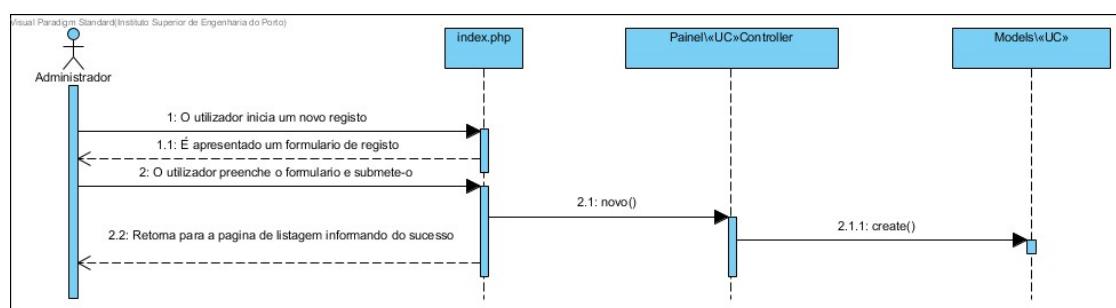


Figura 4.21: Diagrama de sequência de inserir registo

#### 4.4.12 Aplicação Painel - Editar

##### Descrição do caso de uso

Para editar um registo, o utilizador necessita pressionar o botão *editar* na linha referente ao registo que pretende atualizar. A aparência da *view* deste caso de utilização será semelhante ao demonstrado na figura 4.22.

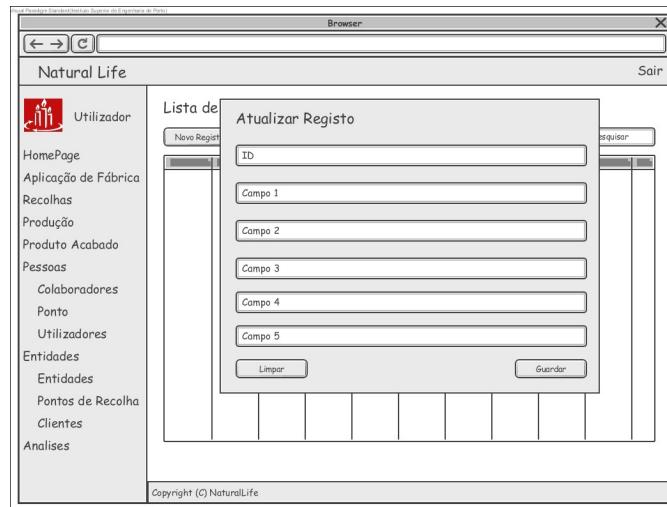


Figura 4.22: Modelo da janela de edição na página de listagem

##### Models compatíveis com o caso de uso

Este caso de uso é compatível com os *models* Recolha, Produção, Produto Acabado, Colaboradores, Registo de Ponto, Entidades, Pontos de Recolha, Clientes e Analises.

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão *editar* da linha do registo que pretende atualizar. O sistema vai fazer um request em background para obter as informações da base de dados e apresenta uma janela flutuante com os campos a serem pré-preenchidos com as informações recebidas. O utilizador faz as alterações que pretende e pressiona o botão *guardar*. Finalizado o registo a janela fecha-se e é apresentado uma mensagem ao utilizador.

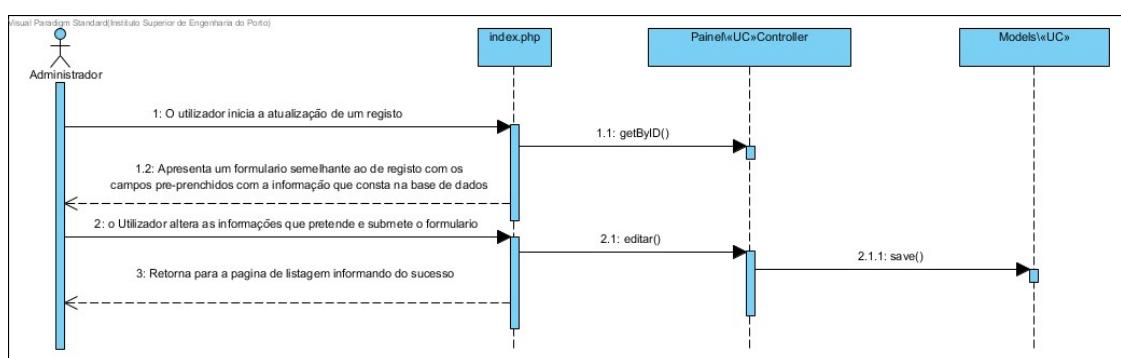


Figura 4.23: Diagrama de sequência de editar registo

#### 4.4.13 Aplicação Painel - Apagar

##### Descrição do caso de uso

Para apagar um registo, o utilizador necessita pressionar o botão *apagar* na linha referente ao registo que pretende atualizar. Pelo facto de ser apenas uma procedimento executado em *background*, não possui nenhuma *view*.

##### Models compatíveis com o caso de uso

Este caso de uso é compatível com os *models* Recolha, Produção, Produto Acabado, Colaboradores, Registo de Ponto, Utilizadores, Pontos de Recolha, Clientes e Analises.

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão apagar da linha do registo que pretende *apagar*. É apresentada uma janela de confirmação. Após executar a ação é apresentada uma mensagem ao utilizador, tal como demonstrado na figura 4.24

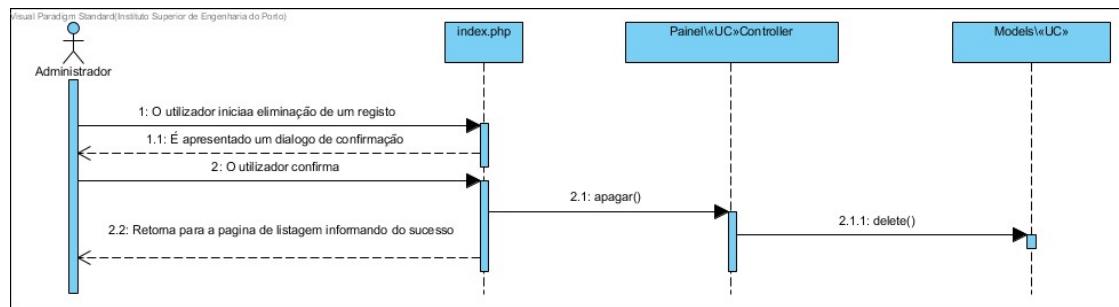


Figura 4.24: Diagrama de sequência de apagar registo

#### 4.4.14 Aplicação Painel - Desativar/Ativar

##### Descrição do caso de uso

Desativar um registo significa que este se vai manter na base de dados, mas não irá constar mais nas UI da aplicação de Fábrica. Para *desativar/ativar* um registo, o utilizador necessita pressionar o botão *desativar/ativar* na linha referente ao registo que pretende atualizar. Pelo facto de ser apenas uma procedimento executado em *background*, não possui nenhuma view.

##### Models compatíveis com o caso de uso

Este caso de uso é compatível com os *models* Colaboradores, Pontos de Recolha e Clientes.

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão desativar/ativar da linha do registo que pretende desativar/ativar. Após executar a ação é apresentada uma mensagem ao utilizador, tal como demonstrado na figura 4.25

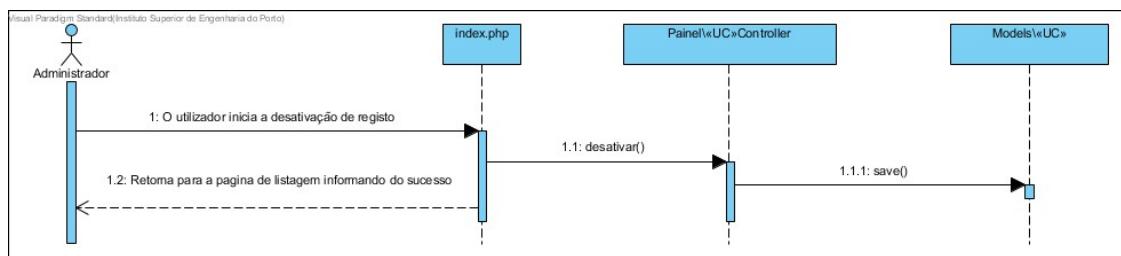


Figura 4.25: Diagrama de sequência de desativar/ativar registo

#### 4.4.15 Aplicação Painel - segunda via código de barras

##### Descrição do caso de uso

Caso pretenda, um utilizador do painel também pode re-imprimir um código de barras sem necessidade de abrir a aplicação Fábrica. Para isso o utilizador necessita pressionar o botão *segunda via* da linha referente ao registo que pretende atualizar. Pelo facto de ser apenas uma procedimento executado em *background*, não possui nenhuma *view*.

##### Models compatíveis com o caso de uso

Este caso de uso é compatível com os *models* Recolhas, Produto Acabado.

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão 2<sup>a</sup> via de código de barras da linha do registo. Um novo separador abre-se com o código de barras pronto a ser impresso, tal como demonstrado na figura 4.26.

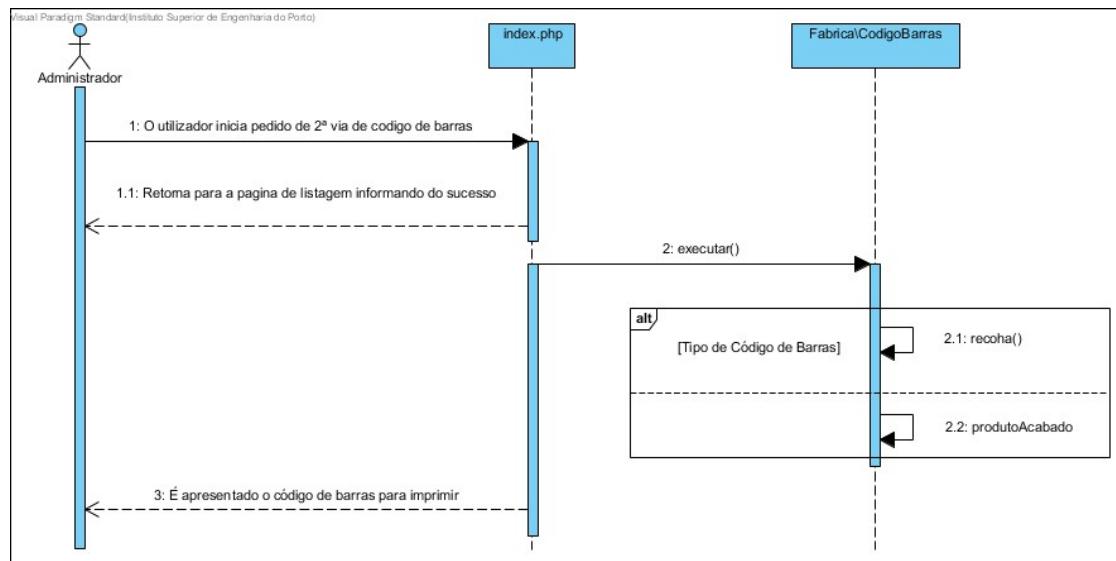


Figura 4.26: Diagrama de sequência imprimir 2<sup>a</sup> via do código de barras

#### 4.4.16 Aplicação Fábrica - Executar Analise

##### Descrição do caso de uso

Uma analise é um *script SQL* com uma *query* personalizada. Para executar uma analise basta pressionar o botão *executar* na linha referente à analise pretendida. A aparência da view deste caso de utilização será semelhante ao demonstrado na figura 4.27.

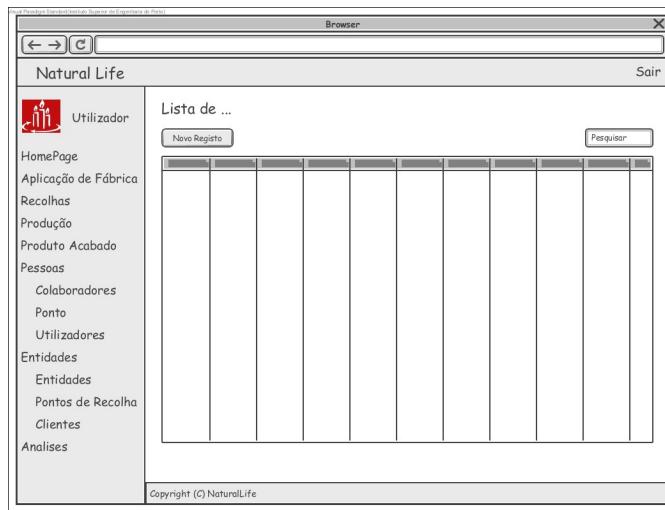


Figura 4.27: Modelo da página de resultado de uma analise

##### Models compatíveis com o caso de uso

Este caso de uso é apenas compatível com o *model* Analises.

##### Fluxo do caso de utilização

O caso de uso inicia-se quando o utilizador pressionar o botão executar da linha do registo que pretende executar. A página vai recarregar, fazendo aparecer o resultado da analise pedida, tal como demonstrado na figura 4.28.

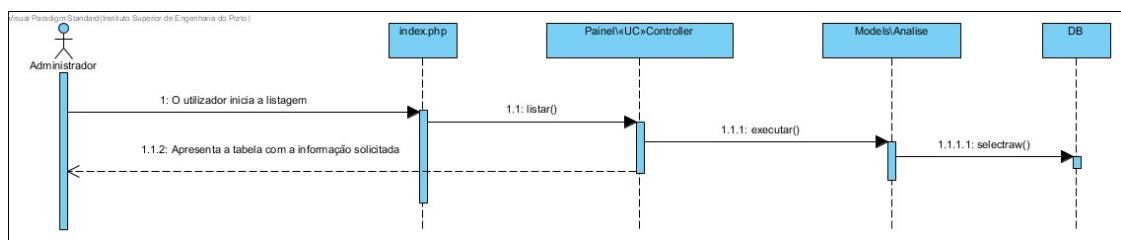


Figura 4.28: Diagrama de sequência para executar uma recolha

## 4.5 Script de verificação da informação

Concluído a projeção de cada caso de utilização, é necessário planejar o último componente do sistema, os *scripts* responsáveis por emitir alertas quando existem dados não coerentes nas tabelas da base de dados. Segundo a lista de requisitos são três os *scripts* a serem desenvolvidos: *Fim de Protocolo*, *Erros no Registo de Ponto* e *Erros de Produção*.

### 4.5.1 Princípio de funcionamento

Pretende-se que todos os *scripts* funcionem de um modo igual: uma SQL *stored procedure* no Microsoft SQL Server que é executada todos os dias. Essa *stored procedure* contém um *query* específico que visa identificar as situações que necessitam ser analisadas. Essa *query* é executada e caso haja retorno é enviado um *email* com essa informação (figura 4.29)

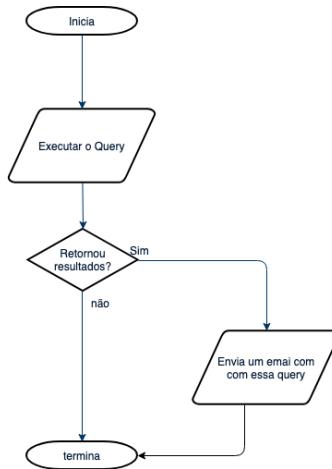


Figura 4.29: Fluxograma das *stored procedures*

### 4.5.2 Objetivo de execução

Cada uma das três *stored procedures* será criada para identificar determinados padrões e reportar-los via *email*. A *stored procedure* *Fim de Protocolo* serve para identificar Pontos de Recolha cujo o fim de protocolo esta-se a aproximar. O código desta *stored procedure* está disponível no anexo B. A *stored procedure* *Erros no Registo de Ponto* serve para identificar regístros de ponto que estão em falta ou duplicados. O código desta *stored procedure* está disponível no anexo C. A *stored procedure* *Erros de Produção* serve para identificar as produções cujo o somatório dos peso é superior ao peso da recolha. O código desta *stored procedure* está disponível no anexo D.

## Capítulo 5

---

# Implementação e validação

---

Neste capítulo são apresentadas as fases de desenvolvimento e a validação do projeto. No final é ainda apresentada uma lista das ferramentas utilizadas no projeto.

## 5.1 Desenvolvimento do projeto

A primeira fase do projeto foi desenvolvida entre as semanas 4 e 6. Este desenvolvimento consistiu criar os modelos (*models*), desenhar cada um dos ecrãs das aplicação (*views*) e os controladores (*controllers*) dos casos de uso Marcar Ponto, Registo de Recolha, Registo de Produção, Registo de Produto Acabado e Registo de Saída de Produto Acabado.

Utilizando os recursos do *framework* Laravel, os *models* foram gerados automaticamente com base na estrutura da base de dados, através do recurso de *migrations*. As *migrations* são um ficheiros que descrevem a estrutura de uma tabela da base de dados. Estes servem para poder criar ou alterar tabelas na base de dados sem que o programador tenha de se preocupar com o DBMS.

O Laravel inclui o Artisan, uma interface de linha de comandos que disponibiliza um conjunto de uteis para a construção da aplicação[10]. Para criar uma *migration* e o seu respetivo *model* basta executar no terminal o seguinte comando

---

```
1 $ php artisan make:migration NomeDaMigration --m
```

---

E serão gerados os ficheiros de *migration* e *model*. Após inserir a estrutura que se pretende para as tabelas da base de dados e suas relações nos ficheiros de *migrations*, usando o comando

---

```
1 $ php artisan migrate
```

---

As tabelas da base de dados seriam criadas exatamente da forma como foram descritas. Finalizada a criação dos *models*, iniciou-se o processo de desenho de cada uma das *views*. Nesta fase já se sabia que muitos elementos se iriam repetir ao longo da interface, uma consequência direta da coesão da interface. Por esse motivo procurou-se desde cedo isolar alguns elementos de cada um dos ecrãs, como descrito na figura 5.1, para que evitar reescrever código, além de simplificar futuro trabalho de manutenção. Definiu-se então que todas as páginas seriam construídas em cima de uma mesma base. Dependendo da página solicitada era incluída a *view* a ela referente. No caso dos formulários era ainda incluídos os botões de

ação (Guardar, Limpar e Cancelar), com a excepção do formulário de Registo de Ponto. Por fim, os elementos de *dropdown* de seleção de Colaborador e Ponto de Recolha eram ainda incluídos de outros dois ficheiros independentes.

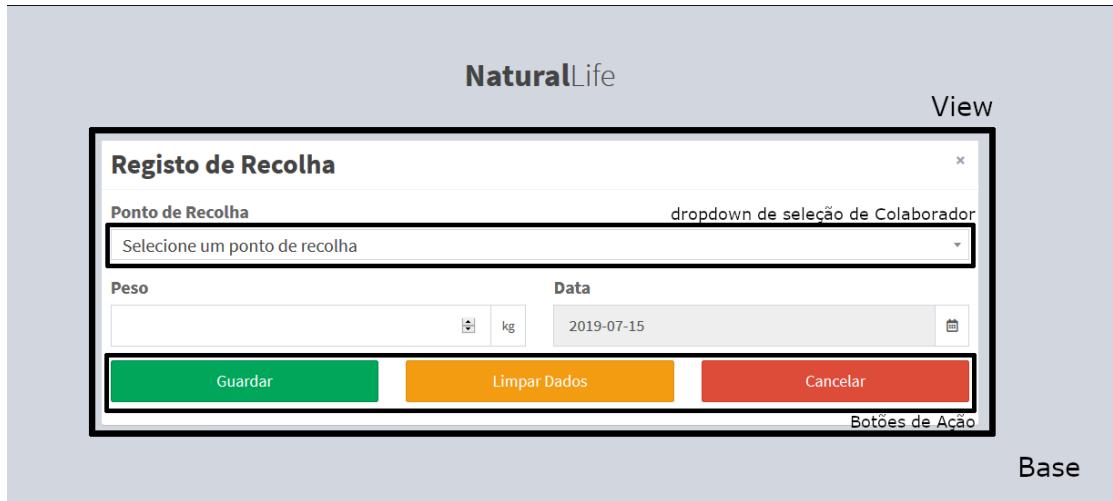


Figura 5.1: Camadas da página Aplicação Fábrica

Estas inclusões de ficheiros foram produzidas através do recurso Blade do Laravel. O Blade é um mecanismos de modelagem incluso no Laravel que ao contrario da maioria dos mecanismos de modelagem permite o uso de código PHP na própria *view*. São ficheiros com a extensão *.blade.php* e estão, normalmente, dentro do diretório *resources/views*[11]. Para utilizar este recurso do *framework* Laravel é necessário usar um template que tenha sido compatibilizado com este. O template utilizado foi o AdminLTE[12] implementado para o Blade pelo projeto Laravel-AdminLTE[13]. Faltava ainda definir o modo como os códigos de barras iriam ser gerados. Consultando os repositórios *online* chegou-se a solução de usar o projeto Laravel 5 Barcode Generator[14].

### 5.1.1 Implementação da primeira fase

Conforme os requisitos do projeto, a primeira fase só poderia ser implementada após as funcionalidades da aplicação existente na fabrica (Registo de ponto, de recolhas, de produção, de produto acabado e de saída de produto acabado). Só após estas funcionalidades bem testadas e a coesão do novo sistema se poderia implementar o novo sistema. No final da sexta a administração concordou que o sistema já tinha condições para ser implementado, mas como a semana seguinte coincidia com a ultima semana do mês de maio a administração solicitou que a implementação fosse adiada uma semana visto que não teria tempo de adaptar todas as ferramentas de analises de dados a tempo de fazer a produção dos relatórios mensais. Ficou então decidido que a implementação iria ocorrer na semana 8 e durante a semana 7 iria continuar o desenvolvimento do sistema além de continuar os testes à versão a ser implementada. De forma a manter a versão estável nessa condição criou-se no repositório uma segunda *branch* denominada *release*.

### 5.1.2 Configuração do servidor e migração de dados

A implementação do projeto teria de ocorrer durante o fim de semana pela necessidade de não haver regtos durante o processo de implementação e migração, ou seja, teria de ser

enquanto a fábrica está parada.

No dia da implementação o servidor já estava montado e sistema operativo instalado, restando apenas instalar o servidor web, interpretador PHP, DBMS, fazer a migração da base de dados e instalar o sistema de informação. Após a instalação dos softwares necessários passou-se para a migração da base de dados. A migração da base de dados consistiu na importação do ficheiro *.accdb* (Microsoft Access) para a base de dados nova. Concluída a importação, foi feito um conjunto de *querys* de inserção da informação que constava na base de dados antiga. Este não foi um processo sempre linear, nomeadamente nas tabelas que foram divididas porque envolveu *querys* mais complexas com *joins*. No final da migração foram efetuados vários *querys* para comparar a informação nas duas bases de dados.

O supervisor da empresa decidiu tornar este processo como um momento didático, optando por construir as *querys* à frente do estudante para assim poder passar alguns conhecimentos de sobre SQL, no lugar de simplesmente preparar as *querys* com antecedência e as executar no momento da migração, uma vez que a migração da base de dados não fazia parte dos requisitos do projeto.

Finalizada a migração da base de dados, ocorreu a instalação do sistema de informação. Para este processo bastou clonar a *branch release* do repositório uma vez que todo o desenvolvimento foi pensado para no momento da implementação as configurações do sistema informação serem as mesmas tanto no computador de desenvolvimento quanto no servidor de produção.

Ocorrem alguns erros no pós instalação pois a aplicação não estava a conseguir comunicar com a base dados, problema que foi resolvido através de uma configuração na *firewall* do servidor.

Para concluir o processo criou-se um utilizador destinado aos administradores, para que estes pudessem aceder ao Painel e adicionou-se atalhos no ambiente de trabalho de todos os computadores da empresa para ter um acesso fácil à aplicação. Após executar o atalho criado uma nova janela do *browser* era aberta, aparecendo o menu da aplicação de fábrica, conforme a figura 5.2.

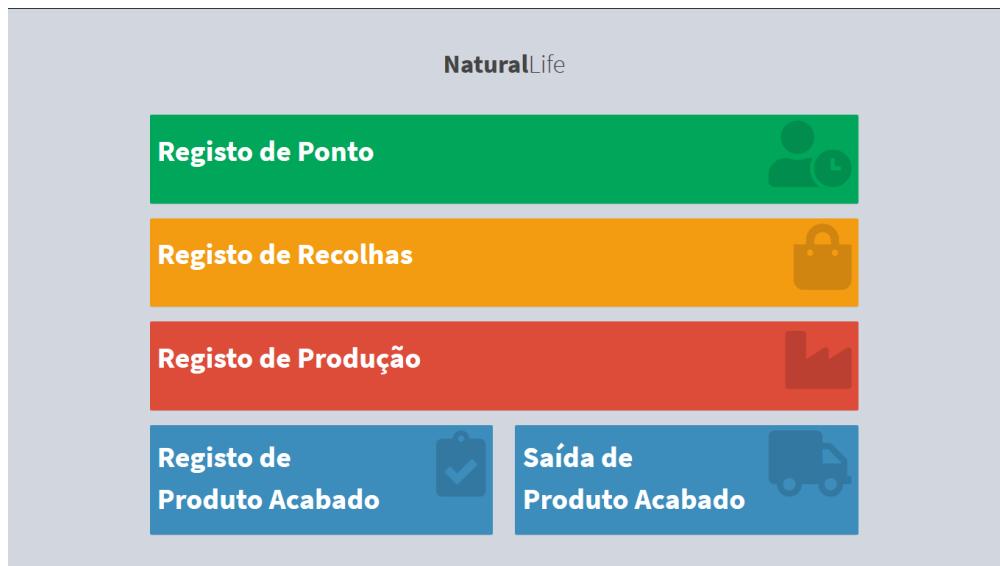


Figura 5.2: Menú da Aplicação Fábrica na primeira fase

### 5.1.3 Primeiro dia de utilização

O primeiro dia utilização foi a segunda-feira da semana oito, dia 3 de junho de 2019. Não foi detetado nenhum erro durante todo o dia, tendo sido apenas apontados detalhes de *user experience*, por exemplo os colaboradores esperavam poder usar a vírgula para separar as casas decimais no lugar do ponto. Apesar de não ser um requisito do sistema usar a vírgula como separador das casas decimais essas modificações foram implementadas pois o trabalho necessário para as fazer era mínimo se comparado com a melhoria da *user experience* dos colaboradores.

### 5.1.4 Segunda fase de implementação

Durante a segunda fase de implementação, as funcionalidades passaram a ser disponibilizadas após concluir o processo de teste. Este processo não era tão rigoroso como o da primeira fase, mas também não se fazia necessário porque deixou de existir a possibilidade da fábrica ser obrigada a parar por algum erro nas funcionalidades básicas da aplicação. No entanto as funcionalidades só passavam a estar disponíveis na versão em produção após o aval do supervisor. Foi nesta fase que além de se desenvolver toda a aplicação Painel, foram implementadas as funcionalidades segunda via de código de barras e Completar recolha.

### 5.1.5 Resultado do desenvolvimento

De forma a se poder comparar diferenças entre o novo sistema e o atual fez-se a seguinte comparação. Na figura 5.3 são apresentadas as diferenças entre o menu da aplicação anterior e o menu da nova aplicação.

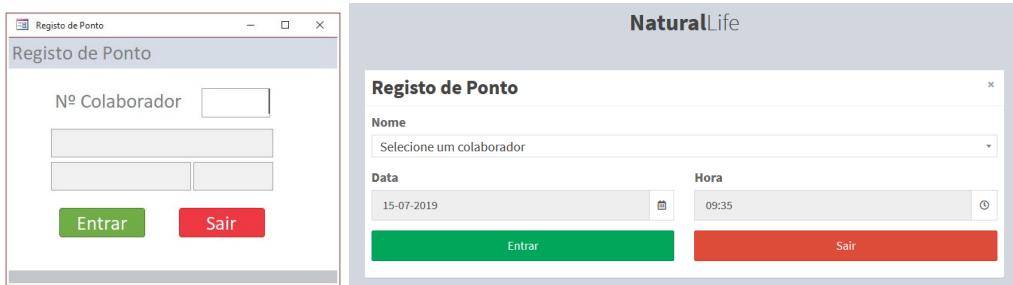


(a) Menu na aplicação anterior

(b) Menu na nova aplicação

Figura 5.3: Modelo do menu

Na figura 5.4 são apresentadas as diferenças entre a *view* de registo de ponto da aplicação anterior e o do novo sistema.

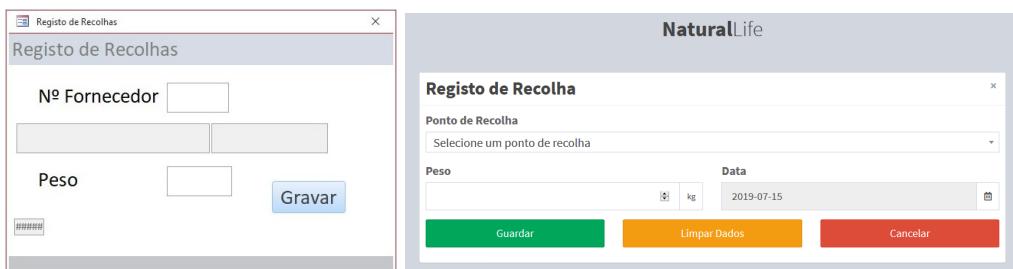


(a) *View* de registo de ponto na aplicação anterior

(b) *View* de registo de ponto na nova aplicação

Figura 5.4: Registo de ponto

Na figura 5.5 são apresentadas as diferenças entre a *view* de registo de recolha da aplicação anterior e o do novo sistema.

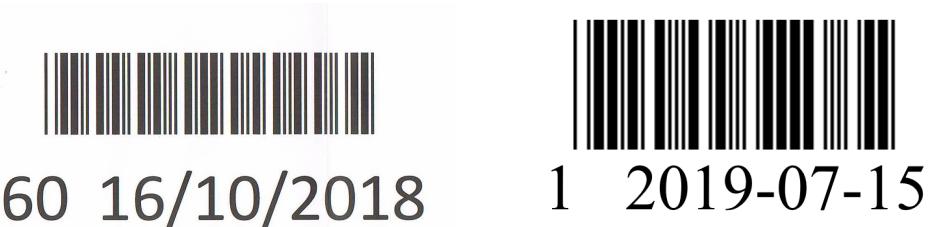


(a) *View* de registo de recolha na aplicação anterior

(b) *View* de registo de recolha na nova aplicação

Figura 5.5: Registo de recolha

Ainda relativamente ao registo de recolha, na figura 5.6 são apresentadas as diferenças entre o código de barras gerado pela aplicação anterior e pelo novo sistema.



(a) código de barras gerado pela aplicação anterior (b) código de barras gerado pela nova aplicação

Figura 5.6: Códigos de barras da recolha

Na figura 5.7 são apresentadas as diferenças entre a *view* de registo de produção da aplicação anterior e o do novo sistema.

- (a) *View* de registo de produção na aplicação anterior      (b) *View* de registo de produção na nova aplicação

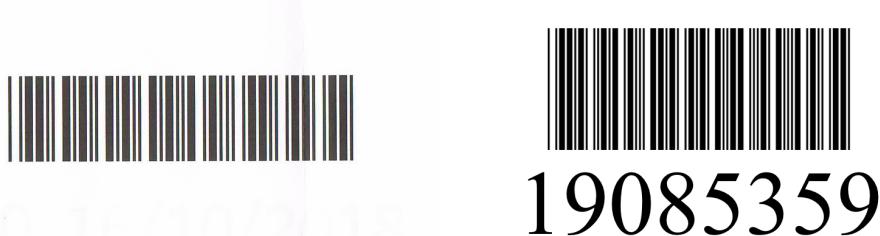
Figura 5.7: Registo de produção

Na figura 5.8 são apresentadas as diferenças entre a *view* de registo de produto acabado da aplicação anterior e o do novo sistema.

- (a) *View* de registo de produto acabado na aplicação anterior      (b) *View* de registo de produto acabado na nova aplicação

Figura 5.8: Registo de produto acabado

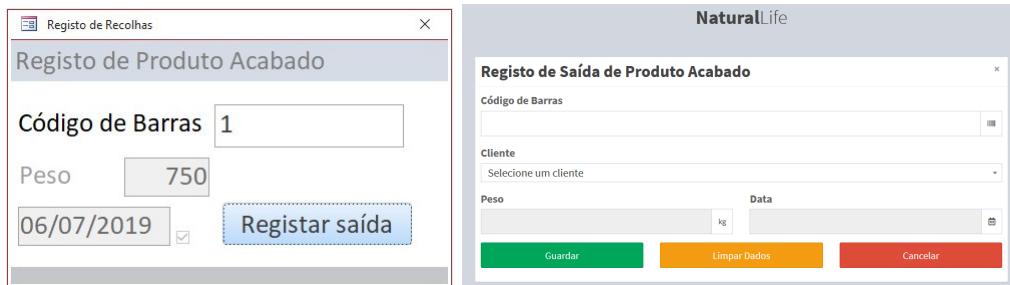
Ainda relativamente ao registo de produto acabado, na figura 5.9 são apresentadas as diferenças entre o código de barras gerado pela aplicação anterior e pelo novo sistema.



- (a) código de barras gerado pela aplicação anterior      (b) código de barras gerado pela nova aplicação

Figura 5.9: Códigos de barras do produto acabado

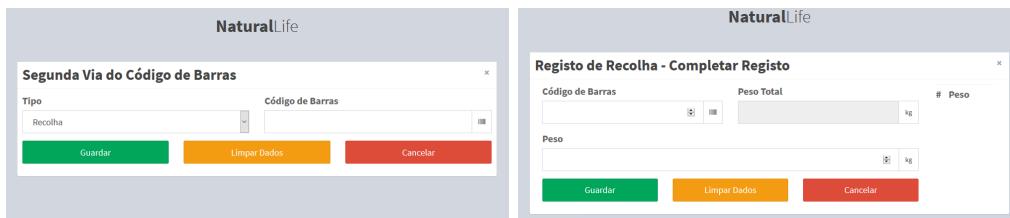
Na figura 5.10 são apresentadas as diferenças entre a *view* de registo de saída de produto acabado da aplicação anterior e o do novo sistema.



(a) *View* de registo de produto acabado na aplicação anterior (b) *View* de registo saída de produto acabado na nova aplicação

Figura 5.10: Registo de saída de produto acabado

Na figura 5.11 são apresentadas as funcionalidades segunda via e completar recolha que não existiam no sistema anterior.



(a) *View* do pedido de segunda via de código de barras (b) *View* do incremento de uma recolha

Figura 5.11: Funcionalidades inexistentes no sistema anterior

Outro elemento que não existia na aplicação anterior era o Painel. Nas figuras ?? e 5.13 são apresentadas as funcionalidades listagem, inserção e edição.

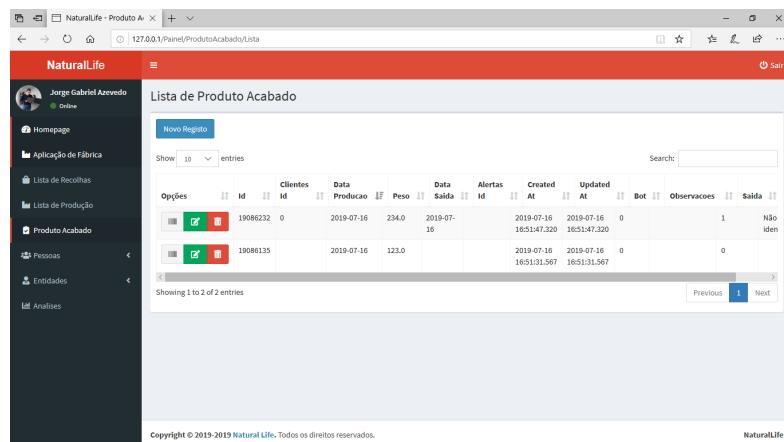
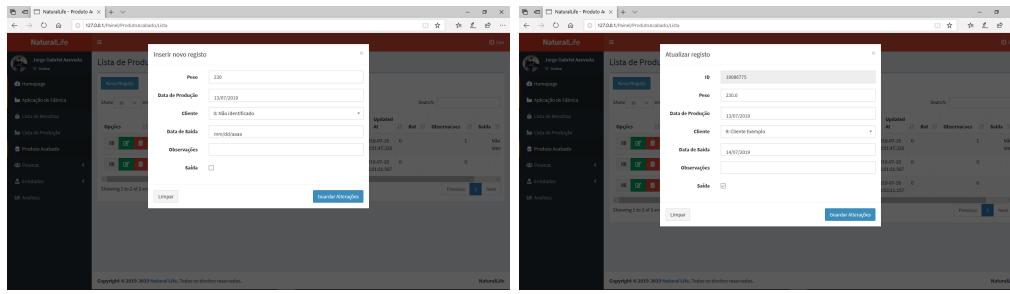


Figura 5.12: Exemplo de *View* listagem



(a) Exemplo de View inserção

(b) Exemplo de View edição

Figura 5.13: View da Aplicação Painel

## 5.2 Validação do trabalho

Logo desde a primeira fase de implementação notou-se uma grande satisfação por parte da administração da Natural Life. Este sentimento foi-se mantendo até ao final do estágio, momento no qual a empresa confirmou que a aplicação iria continuar a ser utilizada na empresa. Consultado o supervisor sobre a poupança de tempo semanal com a implementação deste sistema este referiu uma redução de 20% a 30% do tempo de trabalho semanal a efetuar confirmação e retificação de registos.

Da perspetiva dos colaboradores, o *feedback* também foi muito positivo. O maior receio da administração seria que estes não se adaptassem ao novo sistema e houvesse a necessidade de retornar ao anterior. Mas a verdade é que a reação foi bem melhor do que era esperado. A reação por parte dos colaboradores foi de enorme agrado, havendo inclusivamente lugar a sugestões de melhoria feitas à administração.

### 5.3 Execução do projeto

Como forma de avaliar o desenvolvimento do projeto, foi construído um gráfico da figura 5.14, denominado *burndown chart*, que demonstra a relação entre os objetivos a cumprir com o período do projeto.

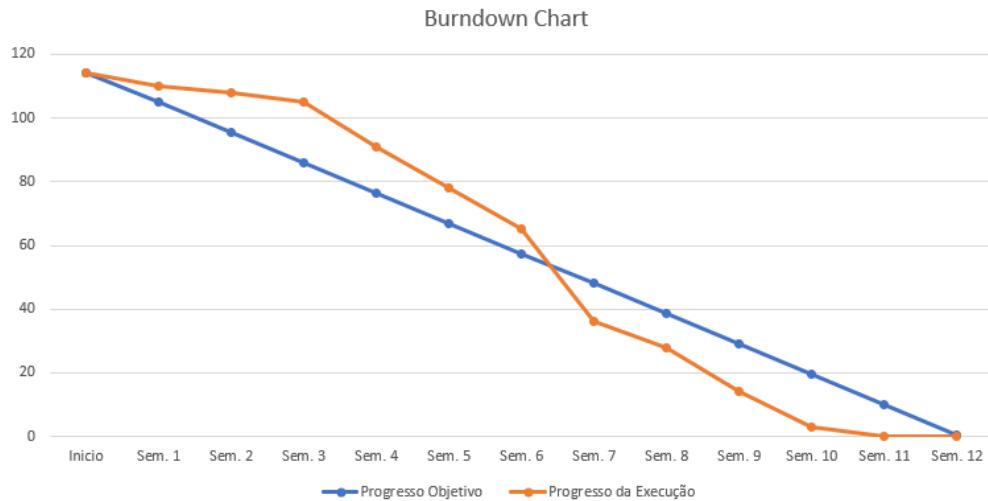


Figura 5.14: *burndown chart* da execução do projeto

A linha de cor azul refere-se ao que era esperável que acontecesse, um desenvolvimento constante no qual o projeto é entregue no último dia do período, com todos os objetivos concluídos. A laranja é possível observar a evolução real do trabalho. Desta linha é relevante analisar que durante as primeiras três semanas a evolução do projeto ficou à quem do esperado, mas após a terceira semana a linha de execução superou a previsão culminando com a conclusão do projeto uma semana antes do previsto. A razão para a diferença entre as duas linhas prende-se com o facto de que nas primeiras três semanas terem sido semanas de estudo e adaptação. Concluído esse processo já tinham sido construídos as ferramentas necessárias para a execução do projeto.

### 5.4 Softwares e serviços usados

O projeto possuía um conjunto tão alagado de requisitos que para proceder à sua elaboração foi necessário utilizar um grande conjunto de softwares e serviços. Desta lista fazem partes, sistemas operativos, editores de código, servidores, DBMSs, linguagens de programação e restantes utilitários.

- Microsoft Windows 10
- Ubuntu 18.04[15]
- Apache[16]
- Microsoft SQL Server[17]
- Microsoft Database Management Studio[18]
- Microsoft Visual Studio Code
- PHP[19]
- Laravel[20]
- AdminLTE[12]
- Laravel 5 Barcode Generator[14]
- JavaScript
- JQuery
- Microsoft Excel

- Microsoft Access
- Microsoft Excel
- Microsoft Word
- Visual Paradigm
- Draw.io
- Trello
- GitHub
- Git
- GitKraken
- L<sup>A</sup>T<sub>E</sub>X
- Overleaf.com[21]

Segundo dados da plataforma GitHub foram feitos 377 *commits*, contabilizando aproximadamente 115 500 linhas de código escritas.

## **Capítulo 6**

---

# **Conclusões**

---

### **6.1 Resumo do relatório**

Este trabalho foi desenvolvido na empresa Natural Life, Lda. No âmbito deste projeto foi feito o estudo completo do sistema anterior de forma a recolher o máximo de informação possível para a construção do novo sistema. O novo sistema satisfez os administradores que estes optaram por o manter em execução e segundo o supervisor permitiu uma poupança de 20% a 30% de tempo de trabalho semanal. Do ponto de vista dos colaboradores a reação foi também bastante positiva.

### **6.2 Objetivos realizados**

A lista de requisitos do trabalho era composta por 38 itens e todos estes requisitos foram cumpridos.

### **6.3 Limitações e trabalho futuro**

Apesar da aplicação estar muito bem integrada na fábrica neste momento é necessário ter a consciência de que com a evolução da fábrica poderá ser necessário fazer alguma atualização ao sistema. O sistema foi pensado para comportar algumas alterações em zonas mais sensíveis, mas é importante realçar que sendo uma aplicação própria da empresa, as funcionalidades que são implementadas são exatamente o que esta necessita, mas fica também a cargo desta criar todas as condições necessárias para manter a estabilidade e segurança deste sistema.

### **6.4 Apreciação final**

Finalizado o trabalho, o balanço que faço é muito positivo. Do ponto de vista técnico foi muito desafiante em dimensão e responsabilidade. Foram escritas aproximadamente 150 mil linhas de código além de ter contacto com 24 tecnologias diferentes, algumas destas como primeira experiência de utilização e foi necessário com agir perante problemas do mundo real e a cima de tudo saber resolvê-los de uma forma eficaz e eficiente. Do ponto de vista pessoal, sinto que também evolui muito tanto no modo como encarei as adversidades que foram surgindo, como também no modo como me empenhei em todas as tarefas realizadas.



---

# Referências Bibliográficas

---

- [1] Destak, "Dez toneladas de velas ardem em cada fim de semana de verão no Santuário de Fátima," 2010. [cited on p. 1]
- [2] Secundino Cunha, "Centro ambiental recicla velas de todas as igrejas do concelho," 2016. [cited on p. 1]
- [3] NaturalLife, "NaturalLife." [cited on p. 1]
- [4] Wikipedia, "SAP ERP." [cited on p. 4]
- [5] PRIMAVERA BSS, "PRIMAVERA BSS, Software de Gestão, Faturação, ERP e POS." [cited on p. 4]
- [6] Wikipedia, "Primavera Business Software Solutions." [cited on p. 4]
- [7] Wikipedia, "Microsoft Dynamics." [cited on p. 5]
- [8] S. Mansuri, "Why Laravel is the Recommended Framework for Secure, Mission-Critical Applications," 2018. [cited on p. 7]
- [9] Wikipedia, "MVC." [cited on p. 17]
- [10] Laravel, "Artisan Console - Laravel - The PHP Framework For Web Artisans." [cited on p. 37]
- [11] Laravel, "Blade Templates - Laravel - The PHP Framework For Web Artisans." [cited on p. 38]
- [12] Almsaeed Studio, "Free Bootstrap Admin Template | AdminLTE.IO." [cited on p. 38, 45]
- [13] J. Noten, "Laravel-AdminLTE." [cited on p. 38]
- [14] N. Milon, "Código de Barras Laravel." [cited on p. 38, 45]
- [15] Canonical, "The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu." [cited on p. 45]
- [16] Apache, "Welcome to The Apache Software Foundation!." [cited on p. 45]
- [17] Microsoft, "SQL Server | Microsoft." [cited on p. 45]
- [18] Microsoft, "Download SQL Server Management Studio (SSMS) - SQL Server | Microsoft Docs." [cited on p. 45]

- [19] PHP, "PHP: Hypertext Preprocessor." [cited on p. 45]
- [20] Laravel, "Laravel." [cited on p. 45]
- [21] Overleaf, "Documentation - Overleaf, Online LaTeX Editor." [cited on p. 46]

## **Anexo A**

---

# **Documento com as alternativas para o desenvolvimento do projeto entregue à administração**

---

O documento a seguir foi produzido para informar a administração das diferentes opções tecnológicas para o desenvolvimento da plataforma solicitada pela empresa.

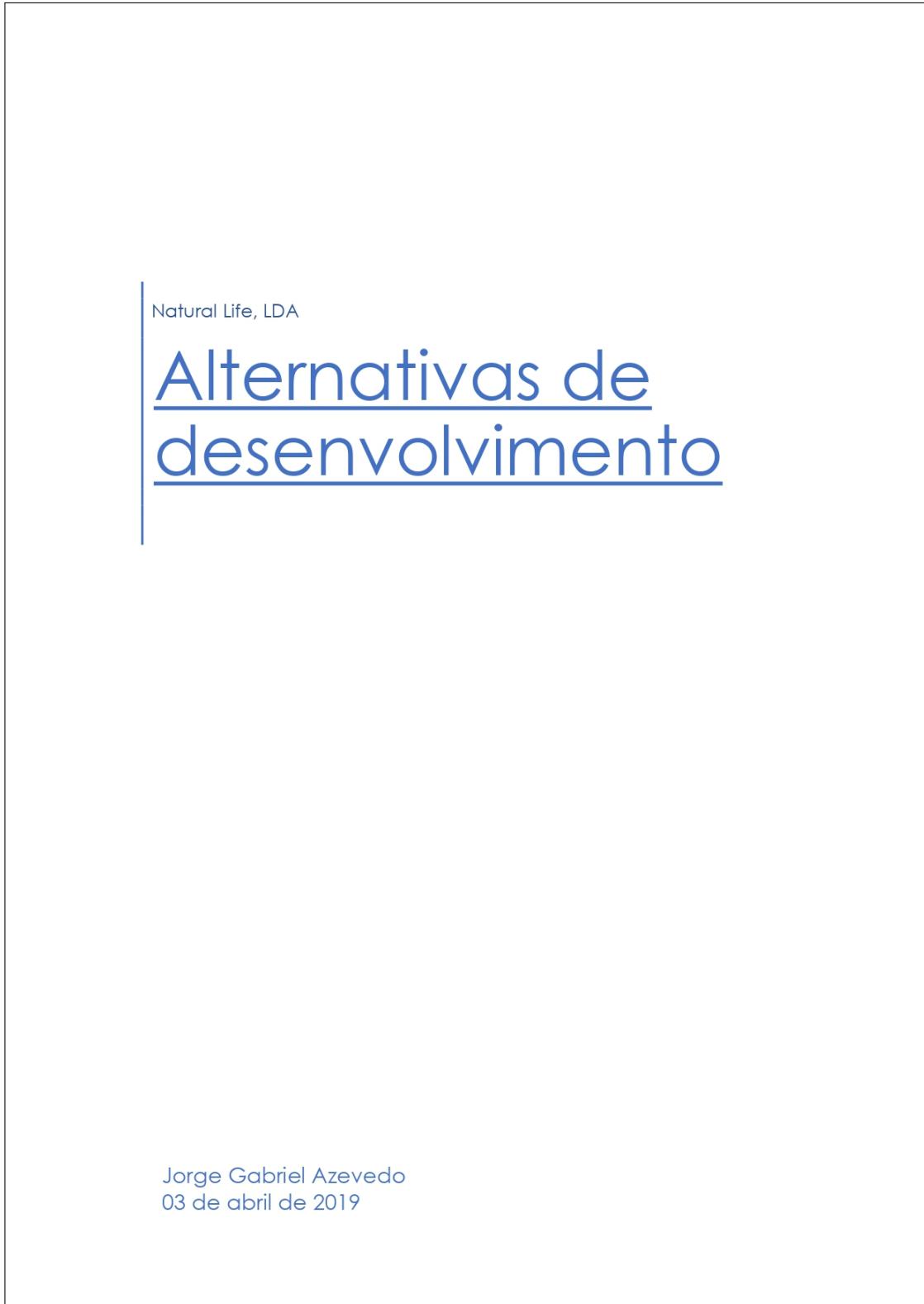


Figura A.1: Capa do documento

## Sobre este documento

Ao longo deste documento são apresentadas as vantagens e desvantagens das principais opções disponíveis para o desenvolvimento da aplicação. Esta comparação tem como objetivo definir a plataforma sobre a qual vai ser desenvolvida a aplicação destinada à Natural Life, LDA.

## Opção geral de desenvolvimento

Analizada o modo de operar atual da empresa determinou-se dois tipos diferentes de utilização, conforme demonstrado na Figura 1:

- Posto de Trabalho: destinado aos funcionários da empresa, é neste posto que são inseridas as informações da produção. Pretende-se que seja de fácil acesso, tenha um registo da informação facilitado e que esteja disponível para os colaboradores da empresa.
- Administração: destinado à administração da empresa, pretende-se apenas o acesso à informação para análise e eventuais correções de erros.

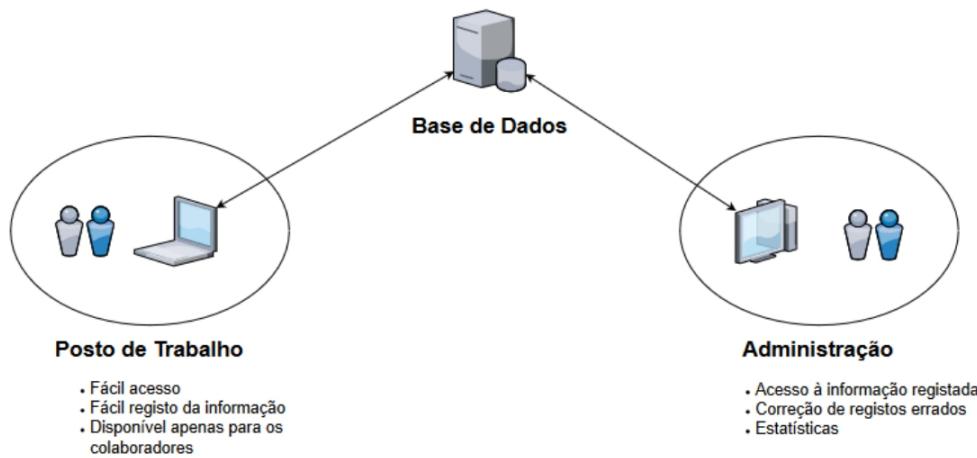


Figura 1 - Diagrama do sistema atual

Estudado o sistema atual, feito com recurso ao Microsoft Access, não se identifica nenhum tipo de impedimento no uso de uma solução web. Seria também possível a utilização de uma solução desktop, por exemplo em java. Esta última tem o inconveniente de ser necessário instalar todas as dependências da aplicação em todos os computadores cliente, além de ter de inutilizar cada máquina durante o tempo de atualização a aplicação. Uma solução web apenas exige que o cliente possua um web browser além de que se for necessário atualizar a aplicação, fazendo-o no servidor, esta fica de imediato disponibilizada aos clientes.

Tendo em conta que já há a necessidade de existir um servidor para a base de dados, faz todo o sentido que a aplicação seja também alojada neste servidor para evitar a dispersão de pontos de falha. Na Figura 2 é apresentada a proposta de estrutura do novo sistema.

Figura A.2: Página 1

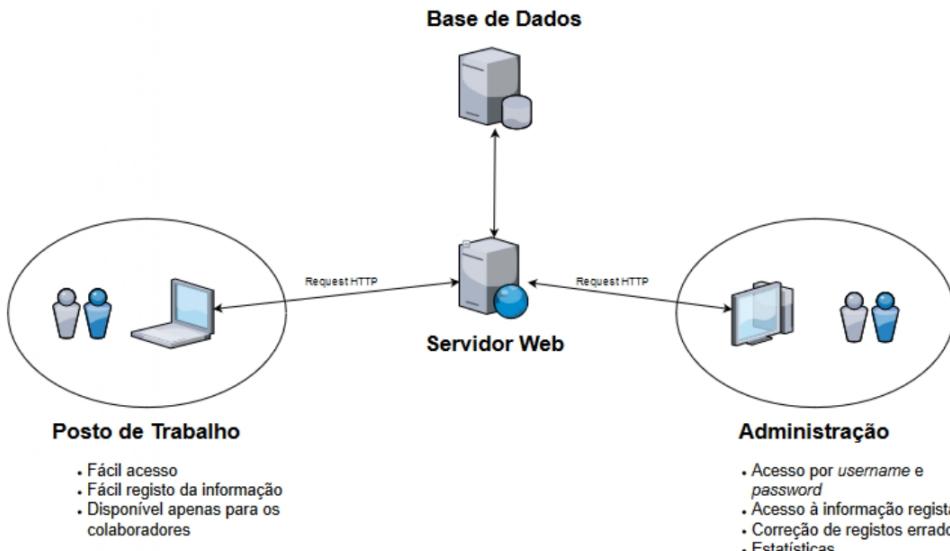


Figura 2-Diagrama do sistema proposto

Faz todo o sentido que se opte pelo servidor web em conjunto com a base de dados que além das vantagens anteriormente indicadas, facilita a expansão para outras funcionalidades que exijam a mobilidade do utilizador, p.e. pré-registo da matéria prima nos cemitérios.

### Sistema operativo do servidor

Como referido no ponto anterior, independentemente da plataforma onde a aplicação vai ser desenvolvida, existe sempre a necessidade de um servidor e foram identificadas duas opções bem distintas:

- Microsoft Windows
- Ubuntu

Todos os computadores da empresa executam uma versão do Windows, mas nenhuma delas é o Windows Server. Só este facto inviabiliza a utilização deste sistema devido à os seus termos de uso. Uma licença da versão Essentials custaria 719,00 € e não se tiraria partido das principais vantagens deste sistema. Esta plataforma iria ainda obrigar a aprender uma nova linguagem de programação (C# e .NET) ou instalação de uma versão adaptada do Apache (via XAMPP por exemplo). Por fim apenas é garantido o suporte do Windows a aplicações desenvolvidas pela própria Microsoft. O Ubuntu por sua vez não tem nenhum tipo de impedimento de utilização na sua licença além de possuir suporte nativo para o servidor Apache e PHP. É um sistema operativo muito utilizado no mercado de servidores pela sua fiabilidade e escalabilidade. Possui um recurso que permite atualizar todo o sistema sem necessidade de reiniciar a máquina o que além de manter a máquina atualizada e protegida, não afeta a disponibilidade da aplicação. O Ubuntu tem ainda a capacidade de ser executado sem interface gráfica o que evita o uso de recursos de processamento, memória e armazenamento da máquina com a interface gráfica. Todos os programas descarregados do repositório do Ubuntu têm suporte da empresa que faz o Ubuntu e são revisados por esta.

O Ubuntu apresenta-se assim como uma solução muito viável e é sobre esta plataforma que recai a sugestão de uso.

Figura A.3: Página 2

## Sistema de Gestão de Base de Dados (DBMS)

Aqui existem também várias opções disponíveis. Desde logo é necessário definir se seria uma base de dados SQL ou NoSQL. De um modo muito sucinto as bases de dados SQL são relacionais e as NoSQL são não relacionais. Tendo em conta o perfil de dados que a empresa regista a opção mais adequada seria SQL.

Definido o tipo de base de dados é necessário escolher o sistema de gestão de base de dados (DBMS). Serão apenas apresentadas aqui soluções que suportam tanto o sistema Windows como Ubuntu.

- Microsoft SQL Server: Disponível em várias versões (*Enterprise, Standard, Express e Developer*)
- MariaDB

Do lado da solução da Microsoft, as versões *Enterprise* e *Standard* são pagas e à partida a empresa não está interessada em despesas extra com este serviço. Sobram a versões *Express* e *Developer*, mas apenas a *Express* poderia ser usada porque os termos de uso da versão *Developer* proíbem o seu uso em produção. A versão *Express* que é bastante limitada e aparenta ser demasiado simplista para o trabalho que a empresa necessita.

O MariaDB é baseado no MySQL que por sua vez é desenvolvido pela Oracle. O MariaDB é um dos DBMS mais utilizados no mundo e herdou do MySQL as suas melhores características aperfeiçoando outros fatores, como por exemplo a performance e segurança. Este DBMS possui duas versões: uma comunitária (gratuita) e uma empresarial (paga). Pelos termos da licença do MariaDB, como esta será uma aplicação interna e com um conector open source (usado pelo PHP ou Java) é legal o uso da versão comunitária. O Ubuntu oferece suporte nativo para o MariaDB e no Windows este teria de ser instalado manualmente ou utilizando o software XAMPP que por padrão usa também o MariaDB.

Por fim, a opção pelo MariaDB parece ser a mais adequada. Além de ser uma plataforma utilizada mundialmente como um padrão de mercado, tem as mesmas características das versões pagas do Microsoft SQL Server, a custo zero.

## Máquina real ou virtual

A opção por uma máquina real ou virtual para alojar o servidor poderá ser a decisão mais complicada. Em ambos os casos, há a necessidade de existir uma máquina física ligada todo o tempo que for necessário usar a aplicação. Neste momento já existe uma máquina ligada 24 horas por dias 365 dias por ano. Além desta exigência, a máquina virtual exige ainda que o perfil de utilizador usado para executar a máquina virtual nunca termine a sessão nem que a máquina entre no modo de suspensão, pois isso vai interromper o processo responsável pela máquina virtual e indisponibilizar o serviço. A máquina real também tem vantagem de não precisar interromper o serviço com alguma atualização do Windows da máquina hospedeira. A máquina virtual, por sua vez, dispensa a necessidade de um hardware dedicado e permite ainda que várias instâncias do mesmo servidor sejam executadas ao mesmo tempo, o que pode ser interessante durante o processo de teste de novas funcionalidades no programa.

Figura A.4: Página 3

### Comparação final

	Opção	Vantagens	Desvantagens
Tipo de Aplicação	Desktop		<ul style="list-style-type: none"> <li>✗ Necessidade de configurar cada computador que receber a aplicação</li> <li>✗ Parar o computador para atualizar a aplicação</li> <li>✗ Necessidade de um servidor de base de dados</li> </ul>
	Web	<ul style="list-style-type: none"> <li>✓ Acesso direto em qualquer computador</li> <li>✓ Maior familiaridade com a plataforma por parte do programador</li> <li>✓ Menos pontos de falha</li> </ul>	<ul style="list-style-type: none"> <li>✗ Necessidade de um servidor web e de base de dados</li> </ul>
Sistema Operativo do Servidor	Windows 10	<ul style="list-style-type: none"> <li>✓ Familiaridade com o sistema</li> <li>✓ Todas as máquinas já executam esta plataforma</li> </ul>	<ul style="list-style-type: none"> <li>✗ Necessidade de comprar uma licença do Windows Server</li> <li>✗ Necessidade de aprender ASP.NET ou usar o XAMPP</li> <li>✗ Se não for usado só software Microsoft não há garantias de segurança/compatibilidade com o Windows</li> </ul>
	Ubuntu 18.04	<ul style="list-style-type: none"> <li>✓ Plataforma líder de mercado</li> <li>✓ Milhares de pacotes no repositório que são revisados pela Canonical</li> </ul>	<ul style="list-style-type: none"> <li>✗ Curva de aprendizagem para manutenção básica</li> </ul>
DBMS	Microsoft SQL Server	<ul style="list-style-type: none"> <li>✓ Familiaridade com o software</li> <li>✓ Versão Express gratuita</li> </ul>	<ul style="list-style-type: none"> <li>✗ Versão Express limitada.</li> <li>✗ Versões Enterprise e Standard pagas</li> <li>✗ Versão Developer não utilizável em produção</li> </ul>
	MariaDB	<ul style="list-style-type: none"> <li>✓ Gratuito (versão Open Source)</li> <li>✓ Totalmente compatível com MySQL da Oracle</li> <li>✓ Padrão no Ubuntu e no XAMPP</li> <li>✓ Escalável</li> <li>✓ Fiable</li> <li>✓ Recursos semelhantes ao MS SQL Server</li> <li>✓ Suporte nativo no Ubuntu</li> </ul>	<ul style="list-style-type: none"> <li>✗ Não possui um ficheiro único para a base de dados.</li> <li>✗ Tem de ser instalado manualmente no Windows ou usado com recurso ao XAMPP</li> </ul>
Máquina	Real	<ul style="list-style-type: none"> <li>✓ Não depende da operação de outras máquinas</li> </ul>	<ul style="list-style-type: none"> <li>✗ Hardware dedicado</li> </ul>
	Virtual	<ul style="list-style-type: none"> <li>✓ Não necessita de hardware dedicado.</li> <li>✓ Várias instâncias da máquina ao mesmo tempo.</li> <li>✓ Backup muito fácil.</li> </ul>	<ul style="list-style-type: none"> <li>✗ Exige que o utilizador que está a executar a VM nunca termine a sessão.</li> <li>✗ Se a máquina real tiver de ser reiniciada, a VM tem de ser interrompida</li> <li>✗ Partilha dos recursos da máquina hospedeira com a máquina virtual.</li> </ul>

Figura A.5: Página 4

### Diagrama do sistema sugerido

Analisadas as diferentes variáveis, definiu-se um seguinte modelo como aquele que melhor se adapta às necessidades e características da plataforma requisitada.

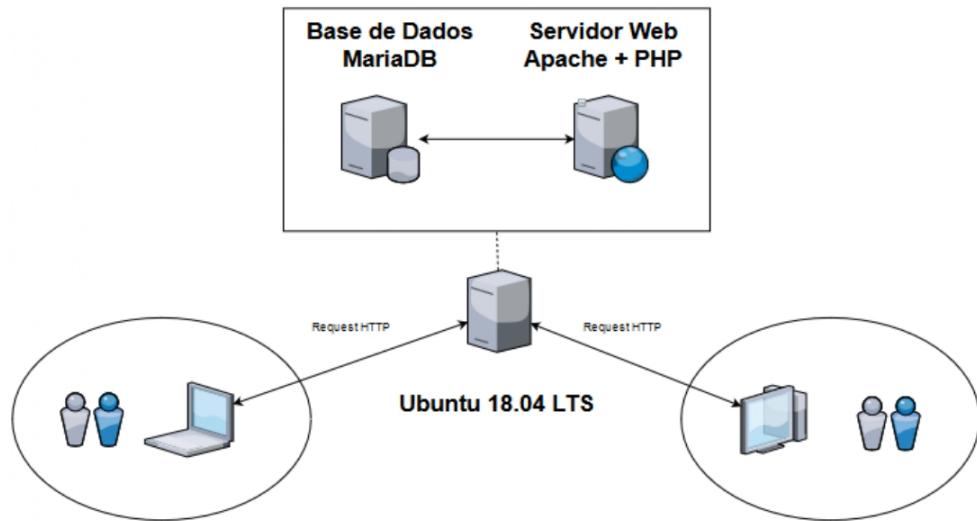


Figura 3 - Diagrama do sistema sugerido

Aqui apenas não se define se é utilizada uma máquina real ou virtual, pois essa é uma decisão que a própria empresa tem de tomar com base no conhecimento que tem do funcionamento das máquinas locais.

O custo final da implementação a cima indicada depende consoante se é escolhida a opção de máquina real ou virtual, mas o custo nunca passaria do custo de aquisição da máquina pois todas as plataformas indicadas são de código aberto e gratuitas.

Figura A.6: Página 5



## Anexo B

---

# Stored procedure fim de protocolo

---

Código utilizado na *stored procedure Fim de Protocolo* para notificar a administração da empresa sobre quais Pontos de Recolha cujo o protocolo está prestes terminar.

```
USE [NaturalLife]
GO
/***** Object: StoredProcedure [dbo].[FimProtocolos]
Script Date: 16/07/2019 08:38:38 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author: NaturalLife
-- Create date: 03-07-2019
-- Description: Alerta pontos de recolha
-- =====
ALTER PROCEDURE [dbo].[FimProtocolos]
AS
BEGIN

--params 1º alerta: 90 dias, 2º alerta: 30 dias, 3º alerta: 15 dias
DECLARE @Alerta1 AS INT = 90 -- 1º Alerta
DECLARE @Alerta2 AS INT = 30 -- 2º Alerta
--DECLARE @Alerta3 AS INT = 0 -- 3º Alerta

IF EXISTS(SELECT
pontos_recolha.id,
pontos_recolha.nome,
pontos_recolha.fimProtocolo,
DATEDIFF(day, GETDATE(), pontos_recolha.fimProtocolo)
FROM pontos_recolha
WHERE
```

```
DATEDIFF(day, GETDATE(), pontos_recolha.fimProtocolo)
in (@Alerta1, @Alerta2))
BEGIN

--Declara as variaveis
Declare @HTMLBody nvarchar(max),
@tableBody nvarchar(max)

--Cria a tabela HTML
SET @tableBody = CONVERT(NVARCHAR(MAX), (SELECT
(SELECT '' FOR XML PATH(''), TYPE) AS 'caption',
(SELECT
'ID' AS th,
'Nome' AS th,
'Fim do protocolo' AS th,
'Dias Restantes' AS th
FOR XML RAW('tr'), ELEMENTS, TYPE) AS 'thead',
(
--Inicio da Query
SELECT
pontos_recolha.id as td,
pontos_recolha.nome as td,
pontos_recolha.fimProtocolo as td,
DATEDIFF(day, GETDATE(), pontos_recolha.fimProtocolo) as td
FROM pontos_recolha
WHERE
DATEDIFF(day, GETDATE(), pontos_recolha.fimProtocolo)
in (@Alerta1, @Alerta2)
ORDER BY DATEDIFF(day, GETDATE(), pontos_recolha.fimProtocolo) ASC
--Fim da Query

FOR XML RAW('tr'), ELEMENTS, TYPE
) AS 'tbody'
FOR XML PATH(''), ROOT('table')));

--Corpo do HTML
SET @HTMLBody = '<html><head><style>
table, th, td {
border: 1px solid black;
}
table {
width: 100%;
border-collapse: collapse;
}
th {
width: 25%;
background-color: #99CCFF;
}
```

```
tr {
width: 25%;
background-color: #F1F1F1;
}
</style><title>Registo de Produção maior que Registo de Recolha</title>
</head><body>
--SET @HTMLBody = @HTMLBody + 'Aproxima-se a data do fim de protocolo
com os seguintes Pontos de Recolha<br/>'
SET @HTMLBody = @HTMLBody + @tableBody + '</body></html>

--envia o email
exec msdb.dbo.sp_send_dbmail
@profile_name = 'NaturalLife',
@recipients = '<Email de Destinatário>',
@subject = '[ALERTA] Fim de Protocolo',
@body = @HTMLBody,
@body_format = 'HTML'

END

END
```



## Anexo C

---

# Stored procedure erros de produção

---

Código utilizado na *stored procedure Erros de Produção* para notificar a administração da empresa sobre quais os registos de produção têm um peso superior ao peso da recolha associada.

```
USE [NaturalLife]
GO
/***** Object: StoredProcedure [dbo].[ErrosProducao]
Script Date: 16/07/2019 08:47:51 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author: NaturalLife
-- Create date: 03-07-2019
-- Description: Alerta erros de produção
-- =====
ALTER PROCEDURE [dbo].[ErrosProducao]
AS
BEGIN

IF EXISTS(SELECT
recolhas.id,
recolhas.data_recolha,
recolhas.peso,
SUM(producao.peso_cera + producao.peso_chapa + producao.peso_plastico)
FROM producao
INNER JOIN recolhas
ON recolhas.id = producao.recolhas_id
WHERE (producao.peso_cera + producao.peso_chapa + producao.peso_plastico) > recolhas.p
AND producao.created_at > GETDATE()-1
GROUP BY recolhas.id, recolhas.data_recolha, recolhas.peso)
BEGIN
```

```
--Declara as variaveis
DECLARE @HTMLBody NVARCHAR(MAX),
@tableBody NVARCHAR(MAX)

--Cria a tabela HTML
SET @tableBody = CONVERT(NVARCHAR(MAX), (SELECT
(SELECT '' FOR XML PATH(''), TYPE) AS 'caption',
(SELECT
'ID Recolha' AS th,
'Data de Recolha' AS th,
'Peso de Recolha' AS th,
'Peso de Produção' AS th
FOR XML RAW('tr'), ELEMENTS, TYPE) AS 'thead',
(
--Inicio da Query
SELECT
recolhas.id AS td,
recolhas.data_recolha AS td,
recolhas.peso AS td,
SUM(producao.peso_cera + producao.peso_chapa + producao.peso_plastico) AS td
FROM producao
INNER JOIN recolhas
ON recolhas.id = producao.recolhas_id
WHERE (producao.peso_cera + producao.peso_chapa +
producao.peso_plastico) > recolhas.peso
AND producao.created_at > GETDATE()-1
GROUP BY recolhas.id, recolhas.data_recolha, recolhas.peso
ORDER BY recolhas.id DESC
--Fim da Query

FOR XML RAW('tr'), ELEMENTS, TYPE
) AS 'tbody'
FOR XML PATH(''), ROOT('table')));

--Corpo do HTML
SET @HTMLBody = '<html><head><style>
table, th, td {
border: 1px solid black;
}
table {
width: 100%;
border-collapse: collapse;
}
th {
width: 25%;
background-color: #99CCFF;
}
tr {
```

```
width: 25%;  
background-color: #F1F1F1;  
}  
</style><title>Registo de Produção maior que Registo de Recolha</title>  
</head><body>'  
--SET @HTMLBody = @HTMLBody + 'Os seguintes registos de produção  
superaram o peso da recolha associada<br/>'  
SET @HTMLBody = @HTMLBody + @tableBody + '</body></html>'  
  
--envia o email  
EXEC msdb.dbo.sp_send_dbmail  
@profile_name = 'NaturalLife',  
@recipients = '<Email de Destinatário>',  
@subject = '[ALERTA] Registo de Produção maior que Registo de Recolha',  
@body = @HTMLBody,  
@body_format = 'HTML'  
END  
  
END
```



## Anexo D

---

# Stored procedure erros no negisto de ponto

---

Código utilizado na *stored procedure Erros no Registo de Ponto* para notificar a administração da empresa sobre quais os registos de ponto não são coerentes.

```
USE [NaturalLife]
GO
/***** Object: StoredProcedure [dbo].[ErroPonto]
Script Date: 16/07/2019 08:48:45 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author: NaturalLife
-- Create date: 03-07-2019
-- Description: Alerta erros nos registos de ponto
-- =====
ALTER PROCEDURE [dbo].[ErroPonto] AS
BEGIN
DECLARE @SQL table(id int NULL, nome nvarchar(max) NULL, data date NULL)

insert into @SQL
-- Número de registos ímpar
SELECT
colaboradores.id AS ID,
colaboradores.nome AS NOME,
cast(ponto.dataHora as date) AS DATA
--colaboradores.nome AS 'Nome',
--COUNT(*) AS 'Contador'
FROM ponto
INNER JOIN colaboradores
ON ponto.colaboradores_id = colaboradores.id
```

```

WHERE cast(ponto.dataHora as date)= cast(GETDATE()-1 as date)
GROUP BY colaboradores.id, colaboradores.nome,
cast(ponto.dataHora as date)
HAVING COUNT(*) % 2 <> 0

union

-- Registros com menos de 10 minutos de diferença
SELECT P1.colaboradores_id AS 'ID_Colab_RegProx',
colaboradores.nome AS 'Nome Colab',
cast(P1.dataHora as date) AS 'Data1'
-- MIN(P2.dataHora) AS 'Data2',
--DATEDIFF(minute, P1.dataHora, MIN(P2.dataHora)) AS 'Diferença'

FROM ponto as P1
LEFT JOIN ponto P2 ON P1.colaboradores_id = P2.colaboradores_id
AND P2.dataHora > P1.dataHora
INNER JOIN colaboradores
ON P1.colaboradores_id = colaboradores.id
WHERE cast(P1.dataHora as date) = cast(GETDATE()-1 as date)
AND cast(P2.dataHora as date) = cast(GETDATE()-1 as date)
AND DATEDIFF(minute, P1.dataHora, P2.dataHora) < 10
GROUP BY P1.colaboradores_id, P1.dataHora, colaboradores.nome

union

-- Número de regtos de entrada diferente de saída
SELECT
colaboradores.id AS ID,
colaboradores.nome AS NOME,
cast(ponto.dataHora as date) AS DATA
--colaboradores.nome AS 'Nome',
--COUNT(*) AS 'Contador'
FROM ponto
INNER JOIN colaboradores
ON ponto.colaboradores_id = colaboradores.id
WHERE cast(ponto.dataHora as date) = cast(GETDATE()-1 as date)
GROUP BY colaboradores.id, colaboradores.nome, cast(ponto.dataHora as date)
HAVING sum(case when entrada = 1 then 1 else 0 end) <>
sum(case when entrada = 0 then 1 else 0 end)

IF EXISTS(
SELECT * FROM @SQL
)
BEGIN

--Declara as variaveis
Declare @HTMLBody nvarchar(max),
@tableBody nvarchar(max)

```

```
--Cria a tabela HTML
SET @tableBody = CONVERT(NVARCHAR(MAX), (SELECT
(SELECT '' FOR XML PATH(''), TYPE) AS 'caption',
(SELECT
'ID' AS th,
'Nome' AS th,
'Data' AS th,
'Mais Detalhes' AS th
FOR XML RAW('tr'), ELEMENTS, TYPE) AS 'thead',
(
--Inicio da Query
SELECT
id as td,
nome as td,
data as td,
cast('<a href =
"http://192.168.1.67/Painel/Colaboradores/Ponto/Erro?SearchOnURL=' +
cast(id as nvarchar) + '_' + cast(cast(data as date) as nvarchar) +
'" target="_blank"> Ver no Painel </a>' as XML) as td
FROM @SQL
--Fim da Query

FOR XML RAW('tr'), ELEMENTS, TYPE
) AS 'tbody'
FOR XML PATH(''), ROOT('table')));

--Corpo do HTML
SET @HTMLBody = '<html><head><style>
table, th, td {
border: 1px solid black;
}
table {
width: 100%;
border-collapse: collapse;
}
th {
width: 25%;
background-color: #99CCFF;
}
tr {
width: 25%;
background-color: #F1F1F1;
}
</style><title>Registo de Produção maior que Registo de Recolha</title>
</head><body>
--SET @HTMLBody = @HTMLBody + 'Aproxima-se a data do fim de protocolo
com os seguintes Pontos de Recolha<br/>'
```

```
SET @HTMLBody = @HTMLBody + @tableBody + '</body></html>'

--envia o email
exec msdb.dbo.sp_send_dbmail
@profile_name = 'NaturalLife',
@recipients = '<Email de Destinatário>',
@subject = '[ALERTA] Erros de Ponto',
@body = @HTMLBody,
@body_format = 'HTML'
END
END
```