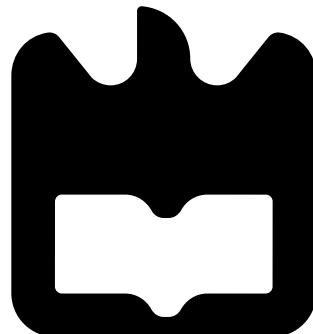




Bruno Manuel
de Moura Ramos

Sistema de Recolha e Armazenamento Remoto
de Informação Sensorial de um Processo
Industrial usando Bases de Dados Múltiplas

DOCUMENTO PROVISÓRIO



o juri/the jury

presidente/president

ABC

Professor Catedratico da Universidade de Aveiro (por delegacao da Reitora da Universidade de Aveiro)

vogais/examiners committee

DEF

Professor Catedratico da Universidade de Aveiro (orientador)

GHI

Professor associado da Universidade J (co-orientador)

KLM

Professor Catedratico da Universidade N

**agradecimentos /
acknowledgements**

Um obrigado aos que me ajudaram. (Adicionar agradecimentos)

Palavras-chave	Base de dados relacional; rede de bases de dados; múltiplas bases de dados; monitorização remota.
Resumo	<p>Os moldes de injeção são uma ferramenta essencial no mundo industrial. A fim de melhorar a qualidade do produto final e reduzir falhas surgiu a necessidade de instrumentar e monitorizar moldes remotamente, este projeto foca-se na parte da monitorização remota. Desenvolveu-se uma rede com múltiplas bases de dados relacionais e uma aplicação em ambiente <i>Web</i>. A primeira garante uma transferência remota de valores segura e permanente de forma a criar um histórico de registos, juntamente com um sistema de criação de <i>backups</i> para gerir e proteger estes históricos. A segunda permite ao utilizador interagir com as bases de dados desenvolvidas podendo editá-las e consulta-las de forma a gerar relatórios.</p> <p>No desenvolvimento deste projeto utilizou-se <i>MySQL</i> e <i>C</i> para criar e definir a rede de bases de dados resultando numa solução simples e funcional a baixo nível. Utilizou-se <i>Apache</i>, <i>PHP</i>, e <i>HTML</i> para desenvolver e implementar a aplicação de forma a que esta seja multiplataforma e garanta um acesso remoto ao utilizador.</p> <p>A solução proposta cumpre todos os objetivos definidos podendo ser já utilizada numa fase experimental. Existe ainda margem para desenvolvimentos futuros como melhoramentos de desempenho e novas aplicações que podem complementar a solução proposta.</p>

Abstract

Nowadays, it is usual to evaluate a work ...

Conteúdo

Conteúdo	i
Lista de Figuras	iii
Lista de Tabelas	vii
1 Introdução	1
2 Conceitos e Ferramentas de Desenvolvimento	3
2.1 Moldes de injeção	3
2.2 Base de dados relacional	4
2.2.1 Domínios, Atributos, Tuplos e Relações	6
2.2.2 Análise de requisitos	7
2.2.3 Diagrama Entidade Relação	7
2.2.4 Esquema Relacional	11
2.3 <i>Structured Query Language</i>	12
2.4 Software utilizado	13
2.4.1 Ferramentas auxiliares	13
2.4.2 <i>HTML, JS</i> e <i>PHP</i>	14
2.4.3 <i>MySQL</i>	15
3 Proposta de Solução	17
3.1 Infraestrutura de Dados	17
3.2 Base de dados	18
3.2.1 Análise de requisitos	18
3.2.2 Desenho conceptual e esquema lógico	24
3.2.3 Construção da base de dados	25
3.2.4 Base de dados regulação de procedimentos	27
3.3 Programa de transferência	28
3.4 Gestão de <i>backups</i>	32
3.5 Simulador	39
3.6 Utilizadores	40
4 Aplicação de Gestão do Sistema	43
4.1 Adaptação da infraestrutura	43
4.2 Interface gráfica	44
4.2.1 <i>Main</i>	47

4.2.2	<i>Login</i>	48
4.2.3	Consultas	49
4.2.4	Administração Local	52
Administração		53
Conectar Local		57
5	Testes de Usabilidade	61
5.1	Preparação dos testes	61
5.2	Plano dos testes	62
5.2.1	Primeira parte - Consultas	62
5.2.2	Segunda parte - Sistema Local	63
5.3	Resultados dos testes	64
5.3.1	Resultados da primeira parte	65
5.3.2	Resultados da segunda parte	67
6	Conclusões	73
6.1	Comentários	73
6.1.1	Infraestrutura de dados	73
6.1.2	Programa de transferência	74
6.1.3	Gestão de <i>Backups</i>	75
6.1.4	Aplicação de gestão do sistema	76
6.2	Trabalhos Futuros	79
Bibliografia		83
A	Queries para instalar a solução desenvolvida	85
A.1	Sistema Central	85
A.2	Sistema Local	89
B	Evolução dos Diagramas Entidade/Relação	93
C	Infraestrutura com todas as bases de dados	97
D	Dados iniciais dos testes de usabilidade	101

Listas de Figuras

1.1	Esquema de rede	2
2.1	Molde de injeção	3
2.2	Esquema de um sistema de bases de dados	5
2.3	Exemplo de atributos e tuplos	6
2.4	Representação de entidades, relações e atributos	8
2.5	Representação de entidades fortes e fracas e relação de identificação	8
2.6	Representação dos graus de relações	9
2.7	Exemplo de decomposição de uma relação ternária em relações binárias	9
2.8	Representação de múltiplas relações entre duas entidades	10
2.9	Representação dos tipos de cardinalidade	10
2.10	Notação da cardinalidade de Chen	10
2.11	Representação da obrigatoriedade de participação	11
2.12	Representação dos tipos de atributos	11
2.13	Representação esquema relacional	12
2.14	Logótipos dos <i>softwares</i> utilizados	13
2.15	Representação da ferramenta mais usada para desenvolver aplicações <i>Web</i>	14
3.1	Diagrama das bases de dados	17
3.2	Diagrama do fluxo de informação	18
3.3	Diagrama final da infraestrutura	19
3.4	Diagrama Entidade/Relação final do projeto	23
3.5	Esquema Lógico final do projeto	24
3.6	Diagrama da comunicação com a base de dados regulação de procedimentos	27
3.7	Fluxograma do programa principal de transferência	29
3.8	Fluxograma dos subprogramas de transferência	30
3.9	Fluxograma do programa principal de <i>backups</i>	33
3.10	Fluxograma da componente de gerar <i>backups</i>	34
3.11	Fluxograma da componente de concatenar <i>backups</i>	35
3.12	Representação da ligação entre os sistemas utilizados no projeto e respetivos endereços	42
4.1	Diagrama da ligação AGS-BDs	43
4.2	Fluxograma do processo de realizar consultas à base de dados	45
4.3	Fluxograma do processo de configurar e definir informação no sistema local do cliente	46
4.4	Funcionalidades da página <i>Main</i> com e sem <i>login</i>	47

4.5	Página de <i>Login</i> para iniciar sessão na base de dados central	48
4.6	Exemplos de erros retornados quando introduzidas credenciais não válidas na página <i>Login</i>	48
4.7	Página de Consultas e exemplo de resposta a uma consulta	49
4.8	Exemplos de erros retornados na página de Resposta da Consulta	51
4.9	Funcionalidades da página Administração Local com e sem conexão local	52
4.10	Área de Gestão de Clientes sem conexão local	53
4.11	Área de Gestão de Clientes com conexão local	54
4.12	Área de Gestão de Moldes	55
4.13	Área de Gestão de Sensores	55
4.14	Função do botão Validar	56
4.15	Área Conectar Local sem BDs locais	57
4.16	Área Instalar <i>MySQL</i> para instalar o <i>software</i> com os comandos em <i>Linux</i>	58
4.17	Área Criar Local	59
4.18	Área Criar Local com <i>queries geradas</i>	59
4.19	Área Conectar Local com BDs locais	60
5.1	Diagrama da montagem dos testes de usabilidade	61
5.2	Gráfico da ocupação profissional dos utilizadores	64
5.3	Gráfico do grau de acompanhamento do projeto dos utilizadores	64
5.4	Gráfico do grau de conhecimentos sobre <i>SQL</i> dos utilizadores	65
5.5	Gráfico dos dispositivos pessoais utilizados nos testes	65
5.6	Gráfico do resultado de iniciar sessão	66
5.7	Gráfico do resultado de gerar uma <i>query</i>	66
5.8	Gráfico do resultado de filtrar a resposta retornada	66
5.9	Gráfico do resultado de reabrir a reposta anterior pelo <i>link</i> após terminar sessão	67
5.10	Gráfico do resultado de verificar que o cliente não se encontra no sistema	68
5.11	Gráfico do resultado de adicionar o cliente	68
5.12	Gráfico do resultado de instalar e configurar o <i>MySQL</i>	68
5.13	Gráfico do resultado de criar base de dados temporária	69
5.14	Gráfico do resultado de criar base de dados local do cliente	69
5.15	Gráfico do resultado de conectar à base de dados local	69
5.16	Gráfico do resultado de alterar cliente	70
5.17	Gráfico do resultado de adicionar moldes	70
5.18	Gráfico do resultado de adicionar sensores	70
5.19	Gráfico do resultado de validar a informação	71
5.20	Gráfico do resultado de atualizar o programa de transferência	71
5.21	Gráfico do resultado de verificar registos a serem gerados e transferidos	71
6.1	Diagrama da infraestrutura com múltiplas BDs locais conectadas a uma BD central, com o programa de transferência de registos descentralizado	74
6.2	Exemplo de quando se clica em Conectar Local pela primeira vez	77
6.3	Exemplo de quando se clica em Conectar Local pela segunda vez	78
6.4	Exemplo de quando se clica em Conectar Local pela terceira vez	78
6.5	Exemplo da área Gestão de Clientes com a funcionalidade Eliminar Cliente	79
6.6	Exemplo de um cliente com moldes em pavilhões diferentes	80

6.7	Exemplo de área de Registar para criar utilizadores com base no ID de trabalhador e email	81
6.8	Exemplo molde 4 dinamometros a sair da cena	82
B.1	Primeira iteração do Diagrama Entidade/Relação	93
B.2	Segunda iteração do Diagrama Entidade/Relação	94
B.3	Terceira iteração do Diagrama Entidade/Relação	95
C.1	Diagrama da infraestrutura completa sem aplicação de gestão do sistema	97
C.2	Diagrama da infraestrutura completa com uma utilização geral da aplicação de gestão do sistema	98
C.3	Diagrama da infraestrutura completa com uma utilização local da aplicação de gestão do sistema	99

Lista de Tabelas

3.1	Tabelas da BD com os seus atributos e respetivos domínios e obrigatoriedades	25
3.2	Tabelas da base de dados regulação de procedimentos com os seus atributos e respetivos domínios e obrigatoriedades	28
3.3	Tabelas com as permissões das credenciais criadas para as tabelas das bases de dados	41
3.4	Tabelas com as limitações dos endereços de conexão das credenciais criadas nos sistemas do projeto	42
6.1	Tabela com os dados utilizados para testar a taxa de transferência de registos	75
D.1	Tabela com os valores iniciais da tabela tipo	101
D.2	Tabela com os valores iniciais da tabela fase	101
D.3	Tabela com os valores iniciais da tabela clientes	102
D.4	Tabela com os valores iniciais da tabela moldes	102
D.5	Parte 1 de 2 da tabela com os valores iniciais da tabela sensores	103
D.6	Parte 2 de 2 da tabela com os valores iniciais da tabela sensores	104

Capítulo 1

Introdução

Para satisfazer as necessidades de uma população cada vez maior, surge a necessidade de produzir artigos em grande escala. Uma das tecnologias que se salientou neste campo, especialmente na produção com plástico, foi a moldagem por injeção.

Os fabricantes de moldes de injeção procuram formas de melhorar as suas ferramentas, como a qualidade do processo de fabrico e a qualidade dos materiais usados na sua construção. No entanto, dada a complexidade do processo e a quantidade de variáveis envolvidas surge a necessidade de instrumentar e monitorizar o molde de forma a garantir um controlo de qualidade. Num esforço conjunto com outros projetos e dissertações procura-se dar mais um passo na monitorização dos moldes de injeção.

O comportamento destas ferramentas pode não se alterar de forma imediata mas alterar-se lentamente ao longo do seu tempo de vida. Para analisar esta alteração do comportamento procura-se uma forma de garantir a criação de um histórico da resposta térmica e integridade estrutural em pontos críticos de um molde de injeção. Dado que os moldes são produzidos para vários clientes pode tornar-se difícil coordenar os históricos desenvolvidos. Este projeto descreve uma solução capaz de coordenar e armazenar tais históricos mantendo a informação organizada e disponível a qualquer momento.

Propõe-se armazenar a informação gerada localmente nos clientes num sistema central situado na empresa que fabrica os moldes. Esta ligação entre os sistemas locais e central, representada na Figura 1.1, visa ser segura, atempada e sem interferência nos processos produtivos. Quanto ao método de armazenamento, propõem-se usar bases de dados relacionais que satisfaçao os requisitos propostos no projeto. Para consultar a informação nas bases de dados propõem-se uma aplicação de gestão do sistema em ambiente *Web*, disponível em qualquer momento, lugar e dispositivo. Esta deve permitir que utilizadores sem conhecimentos sobre bases de dados comuniquem com estas de forma a consultar a informação desejada. Assim sendo enumeram-se os objetivos principais do projeto:

1. Criar uma infraestrutura de bases de dados
2. Criar uma aplicação de gestão de sistema *Web* para consultar os históricos

Como requerimento definiu-se que deve ser dada prioridade à utilização de *software* gratuito. Além disto, dada a quantidade de opções disponíveis no mercado, definiu-se também que a solução desenvolvida deve ser simples de forma a promover a sua portabilidade para outras plataformas e linguagens.

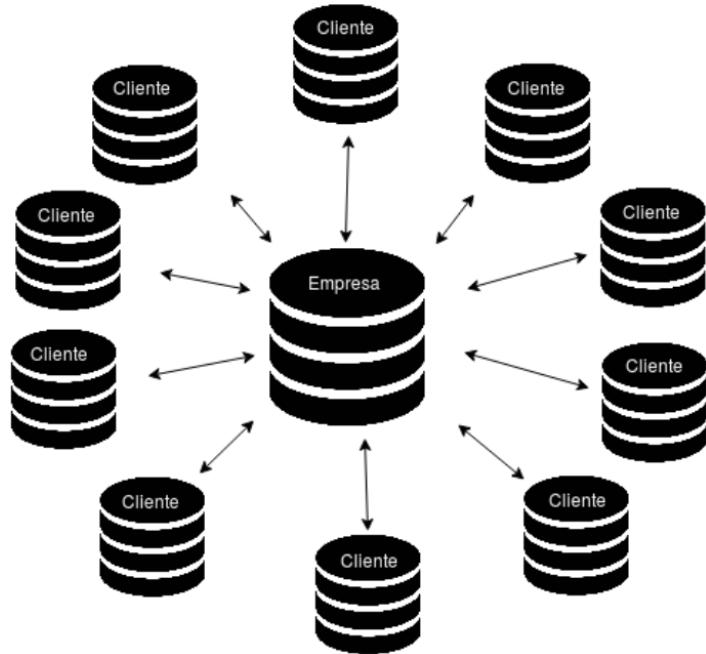


Figura 1.1: Esquema da rede criada com o sistema central na empresa promotora e os sistemas locais nos clientes

Este relatório está dividido em capítulos, divididos da seguinte maneira:

- Capítulo 1 descreveu-se os objetivos do projeto e apresentou-se a solução proposta
- Capítulo 2 consiste na explicação de alguns conceitos necessários importantes para a criação de bases de dados relacionais e na apresentação das ferramentas utilizadas para desenvolver este projeto
- Capítulo 3 explica a infraestrutura de bases de dados proposta, desenvolvida para uma utilização a baixo nível
- Capítulo 4 descreve algumas adições à infraestrutura proposta e as funcionalidades da aplicação de gestão do sistema
- Capítulo 5 apresenta resultados de testes realizados com utilizadores à aplicação de gestão do sistema
- Capítulo 6 contém comentários sobre a solução desenvolvida e ideias para trabalhos futuros que a possam complementar

Capítulo 2

Conceitos e Ferramentas de Desenvolvimento

2.1 Moldes de injeção

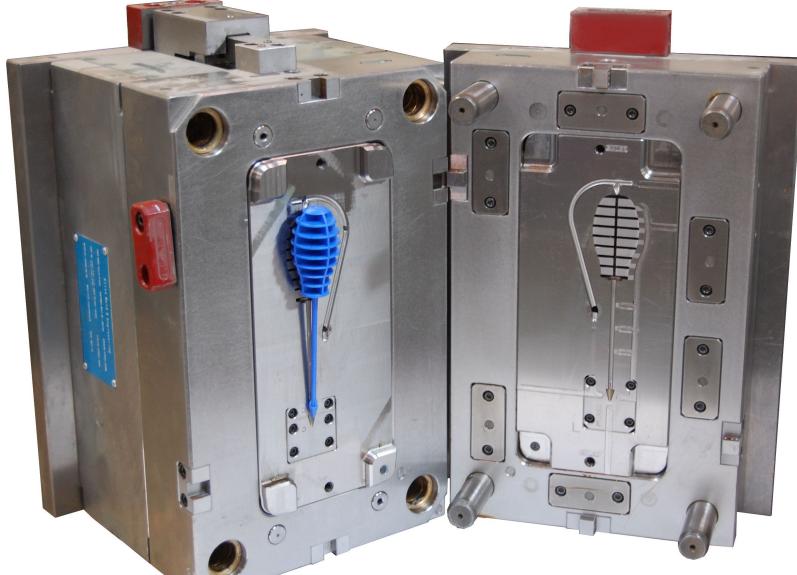


Figura 2.1: Molde de injeção aberto com a peça realizada [1]

Moldagem é o processo mecânico de dar forma a um material no estado líquido usando um molde [2, 3]. Um molde é uma ferramenta sólida oca desenvolvida para efeitos de fundição [4]. Esta é enchida com um material em estado líquido ou em pó como plástico, metal, cerâmica ou vidro [5, 6, 7, 8].

A moldagem por injeção é particularmente útil na produção em massa de peças de plástico com elevada complexidade geométrica [9, 10]. Estas podem ser encontradas em todas as áreas da industria como por exemplo empacotamento, aviação, construção e eletrónica [11]. Neste processo, plástico quente é forçado para dentro de um molde frio com a forma desejada. O

material cobre todas as feições do molde e vai endurecendo sobre o efeito de altas pressões. O processo de injeção pode ser dividido em fases distintas [9]:

- Fecho do molde
- Enchimento
- Compactação
- Abertura do molde
- Extração

Em 1868, John Wesley Hyatt inventou uma maneira de fazer bolas de bilhar injetando celuloide para dentro de um molde [12, 13].

Em 1872 John e o seu irmão Isaiah patentearam a primeira máquina de moldes de injeção. Esta era relativamente simples comparada às que são usadas hoje em dia na indústria. Consistia de um pequeno embolo para injetar plástico num molde através de um cilindro quente [12, 14].

A indústria cresceu lentamente produzindo artigos de plástico como botões e pentes. Nos anos 40 a utilização de moldes de injeção cresceu por causa da Segunda Guerra Mundial que tinha uma grande procura de produtos baratos e produzidos em massa [12].

Em 1946, James Hendry construiu a primeira máquina de moldes de injeção com um sem fim, revolucionando a indústria dos plásticos com um design para substituir o embolo de Hyatt. Este sem fim é colocado dentro do cilindro e mistura o material a ser moldado antes de ser injetado no molde. Isto permitiu que cor ou plástico reciclado fossem adicionados à mistura [12, 15].

A qualidade de fabrico dos moldes e das peças produzidas evoluiu com o passar do tempo. A industria investiu em técnicas sofisticadas no desenvolvimento de moldes, para que estes não contenham falhas, e no uso de materiais com qualidade elevada para evitar o desgaste da ferramenta. No entanto, dificilmente se atinge peças com a qualidade desejada unicamente através das ferramentas desenvolvidas. É necessário implementar também técnicas de monitorização de qualidade [16].

As técnicas de monitorização e controlo de qualidade dos moldes de injeção envolvem principalmente *Artificial Neural Networks (ANNs)* e algoritmos genéricos (em inglês *GA*), que analisam variáveis geradas no molde e adaptam o comportamento da ferramenta. Estes sistemas são aplicados diretamente durante a produção do molde e em circuito fechado [17, 16]. Neste projeto procura-se uma solução que crie um histórico das variáveis dos moldes para determinar alterações no comportamento da ferramenta devido ao desgaste. Apesar de terem objetivos diferentes, prevê-se que no futuro, a solução desenvolvida possa complementar, de alguma forma, as tecnologias de controlo de qualidade existentes na industria.

2.2 Base de dados relacional

Bases de dados (BDs) e sistemas de gestão de bases de dados (SGBDs) são uma componente essencial na vida da sociedade moderna: muitos de nós realizamos ações todos os dias que envolvem interações com bases de dados. Por exemplo, o ato de depositar e levantar dinheiro num banco, reservar estadias e voos ou comprar alguma coisa online podem envolver alguém ou um algum programa informático que acede a uma BD [18].

Esta tecnologia tem um impacto cada vez maior no uso dos computadores. É seguro afirmar que as BDs desempenham um papel crítico em quase todas as áreas onde são usados computadores [18].

Uma base de dados é uma coleção organizada de dados que estão relacionados e que podem ser partilhados por múltiplas aplicações [19]. São considerados dados factos reais que podem ser registados e têm um significado implícito, como por exemplo, nomes, moradas e números de telefone. Uma BD tem uma fonte de onde os dados são provenientes, algum grau de interação com eventos do mundo real e uma audiência que está ativamente interessada no seu conteúdo [18].

A implementação destas é garantida por parte de um SGBD. Este é um *software* de que facilita os processos de definir, construir, manipular e partilhar bases de dados entre vários utilizadores e aplicações. Definir uma BD envolve especificar o tipo de dados, estruturas e restrições dos dados a serem armazenados. Construir é o processo de guardar dados e armazena-los num local definido pelo SGBD. Manipular inclui funções como realizar *queries* à BD e receber informação específica, altera-la de acordo com mudanças nas variáveis e gerar relatórios a partir dos dados presentes. Partilhar permite que múltiplos utilizadores e programas acedam à BD simultaneamente [18].

Outras funções importantes fornecidas pelos SGBDs incluem proteger e manter a base de dados durante um longo período de tempo. Proteger inclui proteção contra falhas de *hardware* e *software* e segurança contra acessos maliciosos ou não autorizados. Tipicamente uma grande BD pode ter um tempo de vida de vários anos, então o SGBD tem de fornecer ferramentas para manter a BD de forma a permitir que o sistema evolua com novos requerimentos que surjam ao longo do tempo [18]. Para completar estas definições iniciais, chama-mos sistema de base de dados ao conjunto das BDs com o SGBD como representado na Figura 2.2.

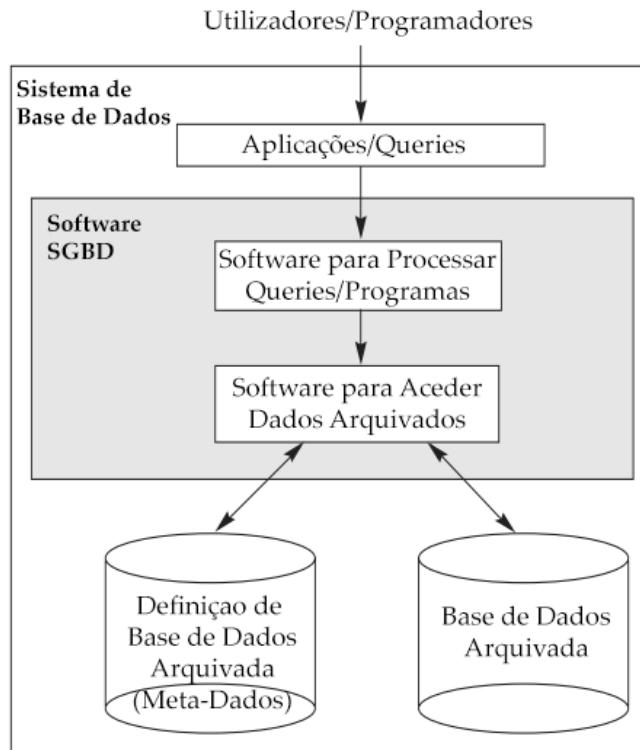


Figura 2.2: Esquema simplificado de um sistema de bases de dados [18]

O modelo relacional foi primeiramente introduzido por Ted Codd do *IBM Research* em 1970 [18, 20] e atraiu imediatamente atenção pela sua simplicidade e base matemática. O modelo utiliza o conceito matemático de "relação" representado por tabelas e baseia-se na teoria dos conjuntos [18].

As primeiras implementações comerciais do modelo relacional ficaram disponíveis no inicio dos anos 80, como o *SQL/DS* no sistema operativo *MVS* do *IBM* e o SGBD *Oracle*. Desde aí, o modelo tem sido implementado num grande número de sistemas comerciais. Os SGBDs relacionais populares atualmente incluem *DB2* e *Informix Dynamic Server* (do *IBM*), *Oracle* e *Rdb* (da *Oracle*), *Sybase* (da *Sybase*) e *SQLServer* e *Access* (da *Microsoft*). Existem também vários sistemas grátiis como *MySQL* e *PostgreSQL* [18].

Modelos de dados que procederam o relacional incluem os modelos hierárquicos e de rede. Estes foram propostos nos anos 60 e foram implementados nos primeiros SGBDs nos anos 60 e 70. Estes modelos tiveram bastante importância na história das bases de dados e são referidos atualmente como SGBDs *legacy* [18].

2.2.1 Domínios, Atributos, Tuplos e Relações

Um domínio é um conjunto atómico de valores. Atómico significa que cada valor num dado domínio é único e indivisível. Um método comum de especificar um domínio é especificar o tipo de dados do mesmo. Além disto, também é útil dar nomes aos domínios. Por exemplo, a informação de números de telemóvel pode ser guardada num domínio onde o seu tipo de dados é um inteiro de nove dígitos [18].

Um esquema de relação é constituído por um nome de uma relação e uma lista de atributos. Cada atributo é o nome do papel desempenhado por um domínio no esquema da relação. É possível múltiplos atributos terem o mesmo domínio [18].

O esquema de relação é um grupo de múltiplos tuplos. Cada tuplo é um conjunto de valores ordenados que estão associados diretamente a um atributo [18]. A Figura 2.3 representa um exemplo do que foi dito anteriormente.

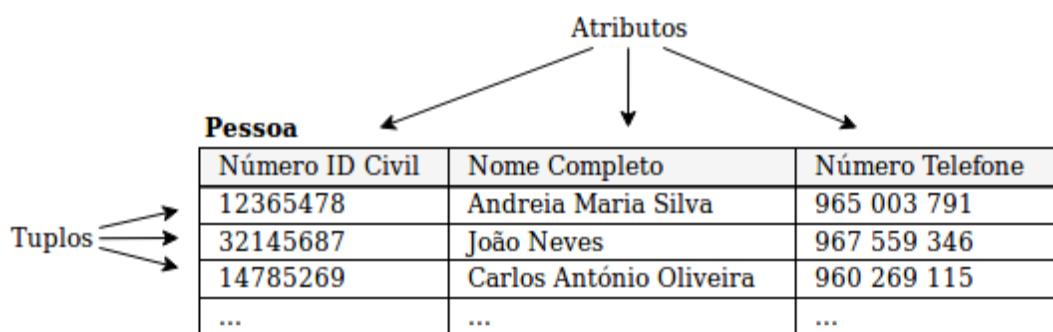


Figura 2.3: Exemplo onde é possível observar a relação com os seus atributos e alguns tuplos

2.2.2 Análise de requisitos

A análise de requisitos é o primeiro passo necessário para definir uma base de dados. Este processo envolve uma comunicação com a entidade que deseja adquirir a BD e pode ser sumariado nos seguintes passos:

1. Numa primeira fase realiza-se uma recolha detalhada de toda a informação referente ao problema do mundo real e retirar entidades, atributos, restrições, etc.
2. Filtrar a informação de forma a remover redundâncias e informação pouco relevante
3. Completar o problema com informação adicional necessária
4. Distinguir dados de operações

Este processo é transversal ao modelo de dados escolhido para construir a BD. Para uma BD relacional é necessário identificar as chaves de uma relação:

- Superchave - Conjunto de atributos que identificam de forma única dois tuplos distintos
- Chave candidata - são todas as superchaves que não podem ser mais simplificadas
- Chave primária - escolhida do conjunto de chaves candidatas e identifica de forma única o tuplo
- Chave única - restantes chaves candidatas que não foram escolhidas como chave primária
- Chave estrangeira - conjunto de atributos que é chave primária de outra relação

A escolha da chave primária pode ser feita de forma aleatória mas, priorizam-se chaves que identifiquem de forma natural um atributo. Por exemplo: uma pessoa pode ser identificada por um número de identificação, número de identificação fiscal ou pelo seu número de telefone dado que estes nunca se repetem. O número de identificação é mais natural para identificar uma pessoa do que o seu número de identificação fiscal e menos volátil do que o seu número de telefone, dado que este pode ser alterado.

Para realizar um Diagrama Entidade/Relação são necessárias informações sobre a relação entre as entidades, a cardinalidade e a obrigatoriedade de participação na relação. Estas são deduzidas durante a análise de requisitos mas serão explicadas na Subseção 2.2.3.

2.2.3 Diagrama Entidade Relação

A representação lógica dos dados é uma componente importante na área das bases de dados. Existem vários modelos para esta representação antes da criação do modelo entidade-relação apresentado por P. P. Chen em 1976, sendo os mais conhecidos os modelos em rede, relacional e entidade. Estes modelos têm as suas vantagens e desvantagens. O modelo em rede permite uma vista mais natural dos dados dividindo-os em entidades e relações (até um determinado ponto), mas a sua capacidade de garantir independência dos dados foi superada. O modelo relacional consegue ter um grande grau de independência dos dados, mas pode perder alguma informação semântica importante sobre o mundo real. O modelo de entidade também consegue um grande grau de independência dos dados, mas a forma visualizar os dados não é tão fácil para algumas pessoas [21].

O modelo entidade-relação adota uma vista mais natural em que o mundo real consiste de entidades e relações, incorpora alguma informação semântica importante sobre o mundo real e consegue ter um grande grau de independência de dados. O modelo entidade-relação é baseado na teoria das relações e foi desenvolvido com o objetivo de servir de base para um sistema de visualização de dados unificado [21].

O Diagrama Entidade/Relação (DER) é uma representação gráfica da análise de requisitos realizada anteriormente baseada no modelo entidade-relação. Este diagrama não é determinístico pois, para uma mesma análise, podem nascer diferentes diagramas que cumprem todos os requisitos.

Um diagrama é constituído elementos como entidades, atributos e relações. Uma entidade é algo que existe no mundo real como uma pessoa ou um carro. Um atributo é uma característica da entidade como uma pessoa tem um nome e um carro tem uma matrícula. Uma relação é como uma ou mais entidades interagem entre si, como uma pessoa tem um carro [21].

Passando à notação, as entidades são representadas por caixas retangulares, os atributos por caixas ovais e as relações por caixas em forma de losango como mostra a Figura 2.4. Os atributos sublinhados representam a chave primária da entidade.

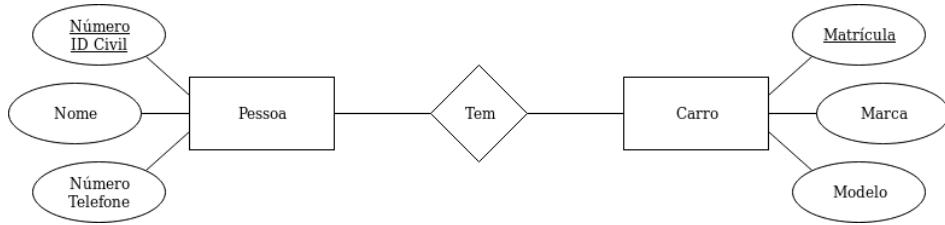


Figura 2.4: Representação das entidades Pessoa e Carro, uma pessoa identificada pelo número de ID civil tem um carro identificado pela sua matrícula

As entidades podem ser fortes e fracas. Uma entidade forte não depende de outras entidades, enquanto uma entidade fraca necessita de ser identificada em conjunto com uma entidade forte. As relações de identificação definem a ou as entidades fortes que identificam uma entidade fraca entre as quais esta se relaciona. As entidades fracas são representadas por caixas retangulares com linha dupla e as relações de identificação são identificadas por um losango de linha dupla como representado na Figura 2.5.

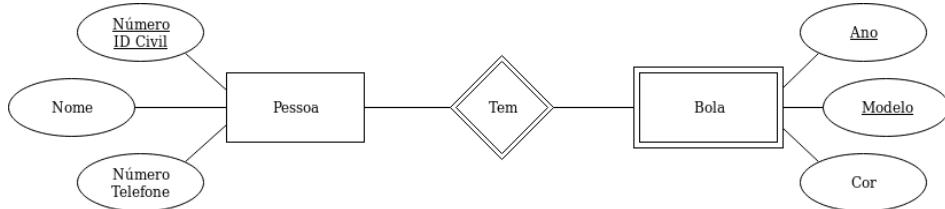


Figura 2.5: Representação das entidades Pessoa e Bola, uma pessoa identificada pelo número de ID civil tem uma bola de um determinado modelo e ano. A bola não tem um elemento caracterizador único pois podem existir bolas do mesmo modelo e ano, então identifica-se esta em conjunto com informação da pessoa que tem a bola

Para completar a relação entre entidades é necessário identificar também o grau, cardinalidade e obrigatoriedade de participação de uma relação. Quanto ao grau, uma relação pode ser unária, binária ou ternária como representado na Figura 2.6. As relações ternárias podem ser decompostas em relações binárias, como mostra a Figura 2.7.

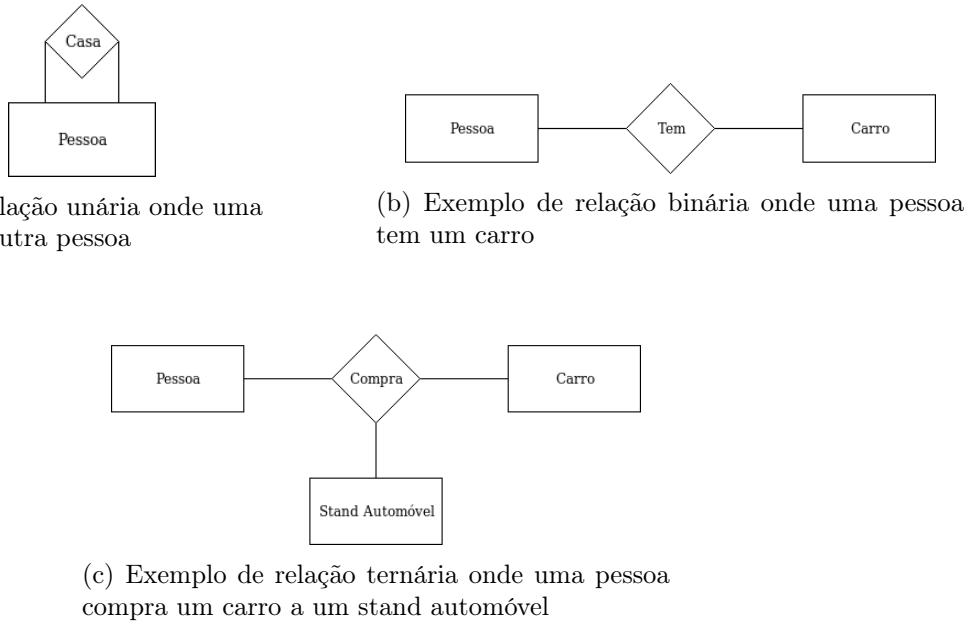


Figura 2.6: Representação dos graus de relações

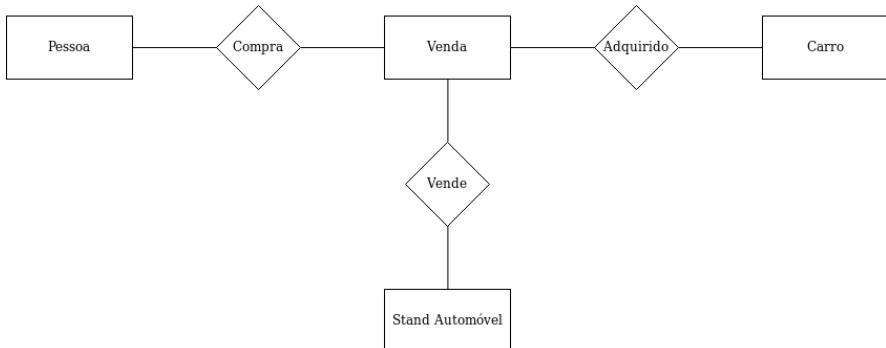


Figura 2.7: Exemplo de decomposição da relação ternária da Figura 2.6c em relações binárias. A entidade venda relaciona de forma mais simples a pessoa, o carro e o stand automóvel envolvidos na compra

As relações podem também ser múltiplas, ou seja, mais do que uma relação entre entidades como demonstrado na Figura 2.8.

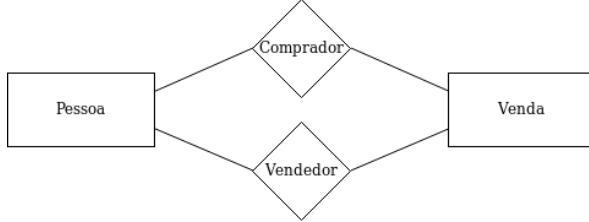


Figura 2.8: Representação de múltiplas relações entre duas entidades, no ato de venda uma pessoa é o vendedor e outra é o comprador

Quanto à cardinalidade uma relação pode ser de três tipos:

- 1 para 1
- 1 para N ou N para 1
- N para M

A Figura 2.9 apresenta uma representação visual destas cardinalidades e a Página 10 apresenta a notação da cardinalidade definida por Chen.

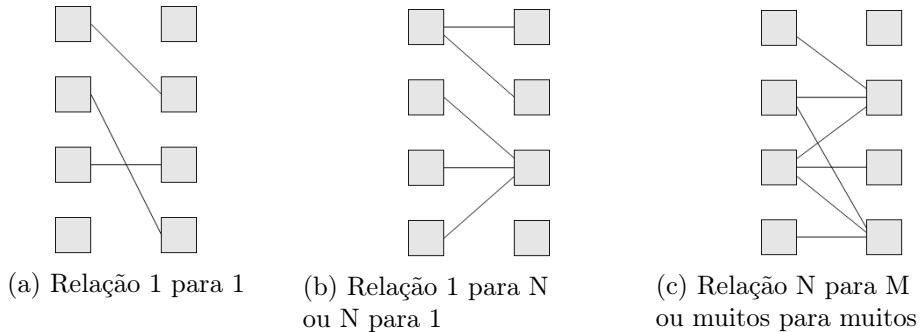


Figura 2.9: Representação dos tipos de cardinalidade

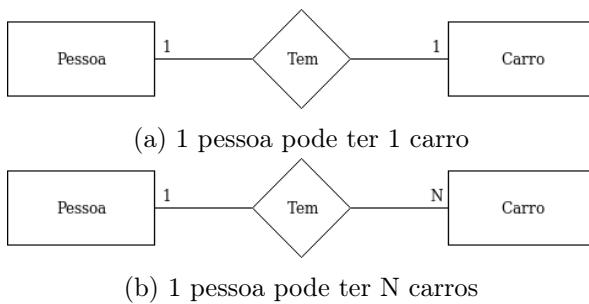


Figura 2.10: Notação da cardinalidade de Chen

A obrigatoriedade de participação numa relação define se uma entidade tem de participar obrigatoriamente na relação. Esta é representada por uma linha dupla na ligação com a relação como representado na Figura 2.11.

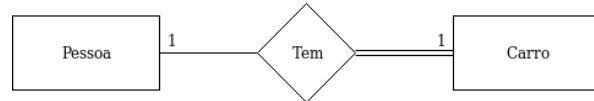


Figura 2.11: Representação da obrigatoriedade de participação. Neste caso uma pessoa pode ter ou não carro mas, um carro tem de pertencer sempre a uma pessoa

Os atributos também podem ser de tipos diferentes. Estes podem ser derivados, compostos ou multi-valor como mostra a Figura 2.12

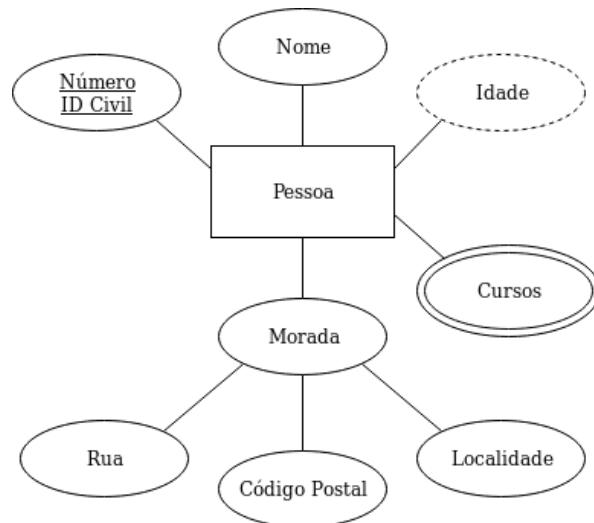


Figura 2.12: Representação dos tipos de atributos. Os atributos derivados são representados a tracejado, os compostos têm sub-atributos associados a si e os multi-valor são representados com linha dupla. A idade varia com a data, uma pessoa pode ter um ou mais cursos e a morada pode ser dividida em rua, código postal e localidade

2.2.4 Esquema Relacional

O esquema relacional inclui todas as relações que concretizam uma base de dados. Cada relação é representada pelos seus atributos, tendo os atributos chave sublinhados. Quando um ou mais atributos de relações diferentes representam o mesmo elemento do mundo real, estes são conectados. Neste caso o conjunto de atributos é chave primária de uma relação e, nas restantes relações, estes são considerados chaves estrangeiras. São representados com uma ligação da chave estrangeira para a chave primária como demonstrado na Figura 2.13 [21]. Este esquema pode ser deduzido a partir do DER onde as entidades se traduzem em relações e as relações do diagrama representam as chaves estrangeiras no esquema relacional.

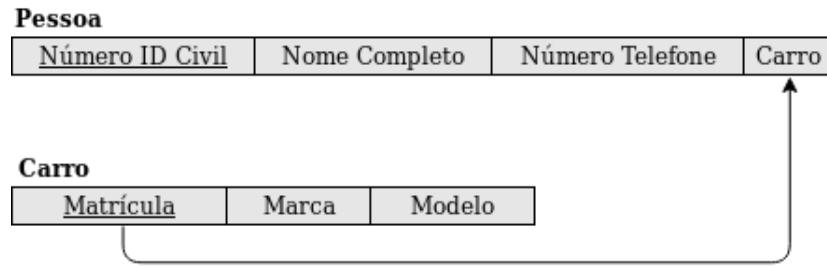


Figura 2.13: Representação do esquema relacional com as ligações das chaves estrangeiras deduzido do DER da Figura 2.10a

2.3 Structured Query Language

A linguagem *SQL* é considerada uma das maiores razões para o sucesso comercial das bases de dados relacionais. Porque se tornou normal a sua utilização em BDs relacionais, os utilizadores tiveram menos preocupações a transferir as suas aplicações de outros tipos de BDs - por exemplo, BDs hierárquicas e em rede - para uma BD relacional. Isto porque se um SGBD não cumprisse os requisitos definidos, não era problemático transferir a solução para outro SGBD dado que ambos utilizam a mesma linguagem. Na realidade existem várias sub-versões desta linguagem dependentes de cada SGBD. No entanto, se o utilizador se limitar apenas às funções standard desta linguagem a mudança de sistema é bastante simplificada. Outra vantagem de ter uma linguagem normalizada é que uma aplicação pode utilizar informação de BDs em SGBDs diferentes ao mesmo tempo [18].

O nome *SQL* atualmente significa *Structured Query Language*. Originalmente era chamado *SEQUEL* de *Structured English QUERy Language*, foi desenhado e implementado no *IBM Research* como uma interface experimental para o sistema de bases de dados relacionais *SYSTEM R*. *SQL*. Um esforço conjunto entre o *American National Standards Institute (ANSI)* e a *International Standards Organization (ISO)* conduziu à normalização da versão do *SQL* chamado *SQL-86* ou *SQL1*. Surgiu depois uma versão expandida chamada *SQL-92* ou *SQL2*. A seguinte versão reconhecida é o *SQL:1999* ou *SQL3*. Duas atualizações depois resultaram nas versões *SQL:2003* e *SQL:2006* [18]. Sendo a atualização mais recente em 2016.

O *SQL* é uma linguagem básica para realizar *queries* que permitem definir e construir uma BD, bem como interagir com os dados presentes nas BDs [18].

2.4 Software utilizado

Para desenvolver as redes de bases de dados e aplicação de gestão do sistema escolheram-se os *softwares* apresentados na Figura 2.14. Como a solução será futuramente implementada em ambiente empresarial, definiu-se que não se deve usar *softwares* que estejam em versão *beta*. Além disto definiu-se que estes devem ser gratuitos.

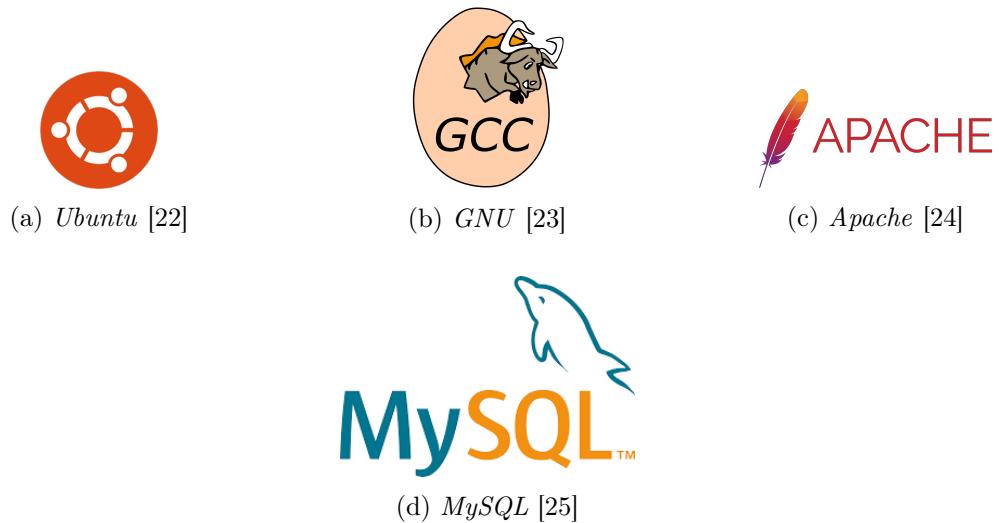


Figura 2.14: Logótipos dos *softwares* utilizados

2.4.1 Ferramentas auxiliares

Escolheu-se o *Ubuntu* 16.04LTS como sistema operativo para os sistemas usados neste projeto, representado na Figura 2.14a. Este é uma distribuição do *Linux* baseado no *Debian*. É um sistema operativo grátis e *open source* desenvolvido pela *Canonical*. A liberdade deste sistema operativo na área da programação criou uma comunidade de utilizadores que ajudam a pesquisar e desenvolver este sistema operativo [26].

A *Canonical* encarrega-se de garantir *updates* de segurança e performance de forma regular. Apesar de não ser infalível o *Linux* é um dos sistemas mais estáveis e menos provável de ser afetado por vírus, dado que a maior parte destes são desenhados para afetar sistemas operativos mais populares como o *Windows* [26].

Escolheu-se utilizar *C* no desenvolvimento deste projeto devido à afinidade com esta linguagem e com o facto do *MySQL* disponibilizar protocolos oficiais de comunicação com esta linguagem e documentação adequada [27].

O *C* é uma linguagem para todos os tipos de problemas e importante no mundo da programação. Originalmente desenvolvido por Dennis Ritchie entre 1969 e 1973 no *Bell Labs* e usado para re-implementar o sistema operativo *Unix*. Desde então esta tornou-se uma das linguagens de programação mais populares de sempre com uma oferta de vários compiladores existentes no mercado. O *C* foi então normalizado pelo *ANSI* em 1989 e consequentemente pelo *ISO*.

No projeto usa-se o *GNU Compiler Collection (GCC)*, representado na Figura 2.14b, como compilador dos programas em *C*. Este foi desenvolvido primeiramente para o sistema operativo *GNU* e focou-se em ser gratuito e garantir liberdade de programação ao utilizador. Conta com

uma comunidade ativa que desenvolvem constantemente novas soluções e *updates* regulares ao compilador de forma a que este não fique desatualizado [28].

Escolheu-se utilizar *Apache HTTP Server*, representado na Figura 2.14c como servidor *HTTP* para a aplicação *Web*. Este foca-se em ser um servidor gratuito e garantir uma solução segura, eficiente e extensível [29].

Lançado em 1995, tornou-se no *software* mais utilizado pelas empresas para correr os seus *sites*. Atualmente na versão 2.4, realizam *updates* constantes ao programa para o manter estável e atualizado [29].

2.4.2 HTML, JS e PHP



Figura 2.15: Representação da ferramenta mais usada para desenvolver aplicações em ambiente *Web* [30]

Escolheu-se *HTML*, *JS* e *PHP* para desenvolver a aplicação de gestão de sistema deste projeto. Junto com *SQL* e *CSS* formam o conjunto de ferramentas mais usadas no desenvolvimento de aplicações *Web* no mercado, representado na Figura 2.15.

HTML ou *Hyper Text Markup Language* descreve a estrutura da página *Web* desenvolvida. Possui elementos que funcionam como blocos para as páginas permitindo a criação de uma interface com um utilizador. *CSS* ou *Cascading Style Sheets* descreve como os elementos *HTML* devem ser dispostos na página, permitindo o desenvolvimento de uma interface mais apelativa. *JS* ou *JavaScript* permite interagir e alterar o código *HTML* e *CSS* criando uma interface mais interativa. Permite também desenvolver códigos e funções que são executadas no browser do cliente. *PHP* ou *PHP:Hypertext Preprocessor* permite desenvolver funções para aplicação que executam do lado do servidor. *JS* e *PHP* realizam ambos funções diferindo apenas no local onde são executadas [31].

Sem contar com a interação com o *HTML* e *CSS*, o *PHP* pode substituir o *JS* como linguagem para criação de funções do sistema. No entanto, isto pode causar problemas de desempenho dado que o servidor terá de correr funções de todos os utilizadores. Assim sendo define-se que o uso do *PHP* deve ser limitado a funções de sistema mais importantes, como conectar a uma base de dados em *SQL*, e o *JS* deve ser usado no desenvolvimento de funções da aplicação mais triviais.

2.4.3 MySQL

Escolheu-se o *MySQL*, representado na Figura 2.14d, como SGBD relacional para construir e utilizar as BDs relacionais usadas neste projeto, dada a sua simplicidade e velocidade de resposta. Existem outras ofertas gratuitas como o *SQLite* e o *PostgreSQL*. O *SQLite* funciona apenas localmente e não permite uma conexão remota que se procura como solução para o problema. O *PostgreSQL* é um sistema robusto que permite realizar tarefas que o *MySQL* não consegue realizar. No entanto, dada a simplicidade da BD prevista nesta fase do projeto, não é necessário usar uma ferramenta tão completa e robusta [32].

O *MySQL* é o mais popular e o SGBD relacional mais usado [32]. Garante um acesso múltiplo, rápido e robusto a uma BD no servidor. Foi desenvolvido para lidar com grandes quantidades de informação e *softwares* com utilização massiva [27].

O *MySQL* oferece soluções gratuitas e pagas, *Community* e *Enterprise*, respetivamente. A primeira recebe menos atualizações e correções que a solução paga. Tem também algumas funcionalidades bloqueadas e um limite na capacidade de sensivelmente 4Gb [27]. Estas limitações não afetam o projeto e como tal opta-se pela solução gratuita deste *software*.

Como referido na Seção 2.3, os vários SGBDs possuem as suas próprias sub-versões da linguagem *SQL* e, o *MySQL*, não é exceção. Segue-se a lista com os comandos principais utilizados neste projeto [27]:

- **SHOW DATABASES/TABLES** - mostra todas as BDs ou tabelas de uma BD (*MySQL*)
- **CREATE USER/DATABASE/TABLE** - criar utilizadores, BDs ou tabelas
- **DROP USER/DATABASE/TABLE** - eliminar utilizadores, BDs ou tabelas
- **SELECT FROM** - visualizar tuplos de uma tabela
- **INSERT** - inserir tuplos numa tabela
 - **INTO** - se forem inseridos múltiplos tuplos e um não respeitar as restrições de integridade definidas, nenhum tuplo é inserido
 - **IGNORE** - se forem inseridos múltiplos tuplos e um não respeitar as restrições de integridade, esse é descartado e os restantes são inseridos na tabela (*MySQL*)
- **DELETE** - elimina tuplos de uma tabela
- **UPDATE** - altera tuplos de uma tabela
- **GRANT** - dá privilégios a um utilizador

A *query* do tipo SELECT permite adicionar condições do tipo [27]:

- **WHERE** - definir parâmetros de busca de um atributo
- **GROUP BY** - permite agrupar informação permitindo operações matemáticas sobre valores, por exemplo: contagem, soma, média, etc.
- **ORDER BY** - escolher a ordem dos tuplos

As *queries* do tipo DELETE e UPDATE também permitem utilizar a condição WHERE.

Na criação das tabelas atribuem-se o nome dos atributos e o seu respetivo domínio. O MySQL oferece vários tipos de dados, dos quais foram usados [27]:

- **VARCHAR(n)** - string com tamanho máximo n
- **INT** - inteiro entre -2147483648 e +2147483647
- **TINYINT** - inteiro entre -128 e +127
- **FLOAT** - numérico com casa decimais
- **DATETIME** - data e hora no formato AAAA-MM-DD HH:MM:SS

Define-se também as restrições de integridade [27]:

- **PRIMARY KEY** - define chave primária
- **UNIQUE** - define chave única
- **FOREIGN KEY REFERENCES** - define a chave estrangeira e o atributo de referência
- **CHECK** - define restrições para os atributos

Estas restrições ou **CONSTRAINTS** podem e devem ser atribuídas um nome de forma a identifica-las em mensagens de erro. Além disto é possível definir o comportamento de uma restrição de chaves estrangeiras. Quando o atributo de referencia é alterado (**ON UPDATE**) ou eliminado (**ON DELETE**) [27]:

- **NO ACTION** - não deixa realizar a ação
- **CASCADE** - todas as chaves estrangeiras dependentes do atributo de referencia são alteradas ou eliminadas de acordo com este
- **SET NULL** - altera o valor da chave estrangeira para NULL
- **SET DEFAULT** - altera o valor da chave estrangeira para um valor predefinido

O valor NULL representa um valor de um atributo que não existe ou é desconhecido, adiciona-se **NOT NULL** a um atributo quando este não pode ser nulo [27].

Estes são os comandos e conceitos principais que surgem ao longo deste documento. Como referido na Seção 2.3, e se observa aqui, o *SQL* é uma linguagem simples e auto-explicativa.

Capítulo 3

Proposta de Solução

A infraestrutura de dados desenvolvida tem como objetivo ser simples, funcional a baixo nível e sem impor restrições na instrumentação dos moldes. Este capítulo descreve o processo de desenvolvimento das bases de dados relacionais, o método de transferência de registos, a gestão de *backups*, as permissões dos utilizadores e um simples simulador de registos. No Apêndice A encontram-se as *queries* desenvolvidas para construir a solução proposta, com base na informação deste capítulo.

3.1 Infraestrutura de Dados

A instrumentação dos moldes geram registos localmente nos clientes e estes devem ser guardados numa base de dados central. Realiza-se a transferência usando protocolos TCP/IP. A fim de minimizar a perda de registos em caso de falha de conexão, coloca-se uma base de dados local em cada cliente, como representado na Figura 3.1.

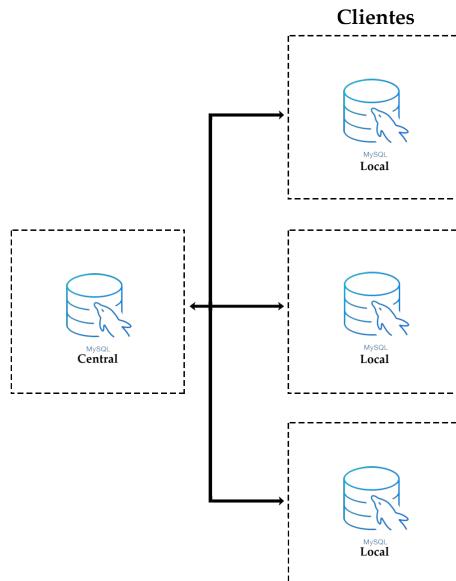


Figura 3.1: Diagrama da infraestrutura com múltiplas BDs locais conectadas a uma BD central

Para estabelecer o fluxo de informação desenvolveu-se um programa de transferência e um simulador, ambos desenvolvidos em *C*. O programa de transferência envia *queries* para as BDs locais e recebe respostas com registos que insere na BD central. O simulador gera registos para popular as BDs locais. Para simular múltiplos clientes usam-se simuladores independentes entre si, como representado na Figura 3.2.

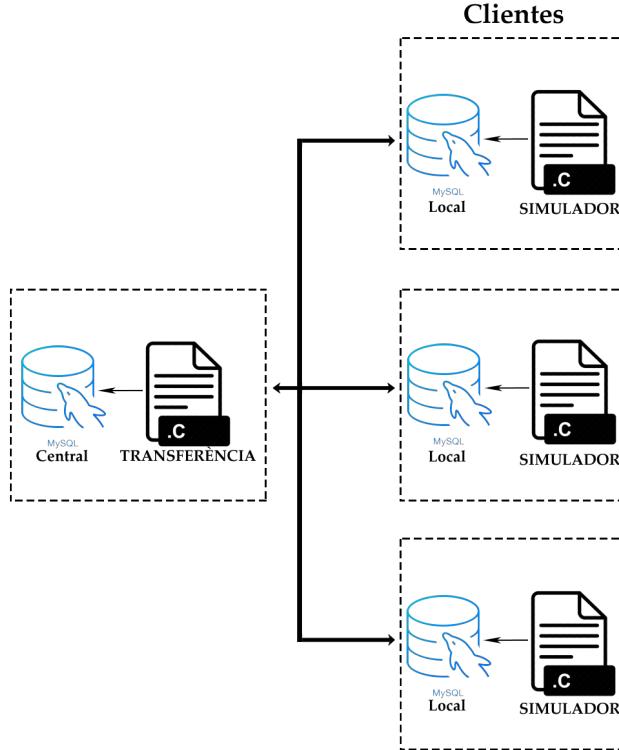


Figura 3.2: Diagrama do fluxo de informação. Os simuladores substituem provisoriamente sistemas de aquisição de dados

Para melhorar o desempenho da BD central e prevenir falhas críticas do sistema, desenvolveu-se um programa de gestão de *backups*, também em *C*. Este programa limpa e cria *backups* da BD e guarda-os num repositório *online*. De forma a não interferir diretamente com a informação da BD central criam-se duas BDs para a gestão dos *backups*. Com esta adição, o esquema da infraestrutura final está representado na Figura 3.3.

3.2 Base de dados

3.2.1 Análise de requisitos

Deseja-se, de uma forma geral, um sistema capaz de guardar remotamente medições realizadas em moldes instrumentados. Esta informação não é suficiente para se deduzir uma base de dados. Assim sendo, analisando e completando a informação inicial chega-se ao seguinte enunciado:

Um cliente, caracterizado por um ID único, nome, morada, IP e port, tem moldes. Estes moldes são identificados por um ID único, nome e descrição. Os moldes têm sensores com

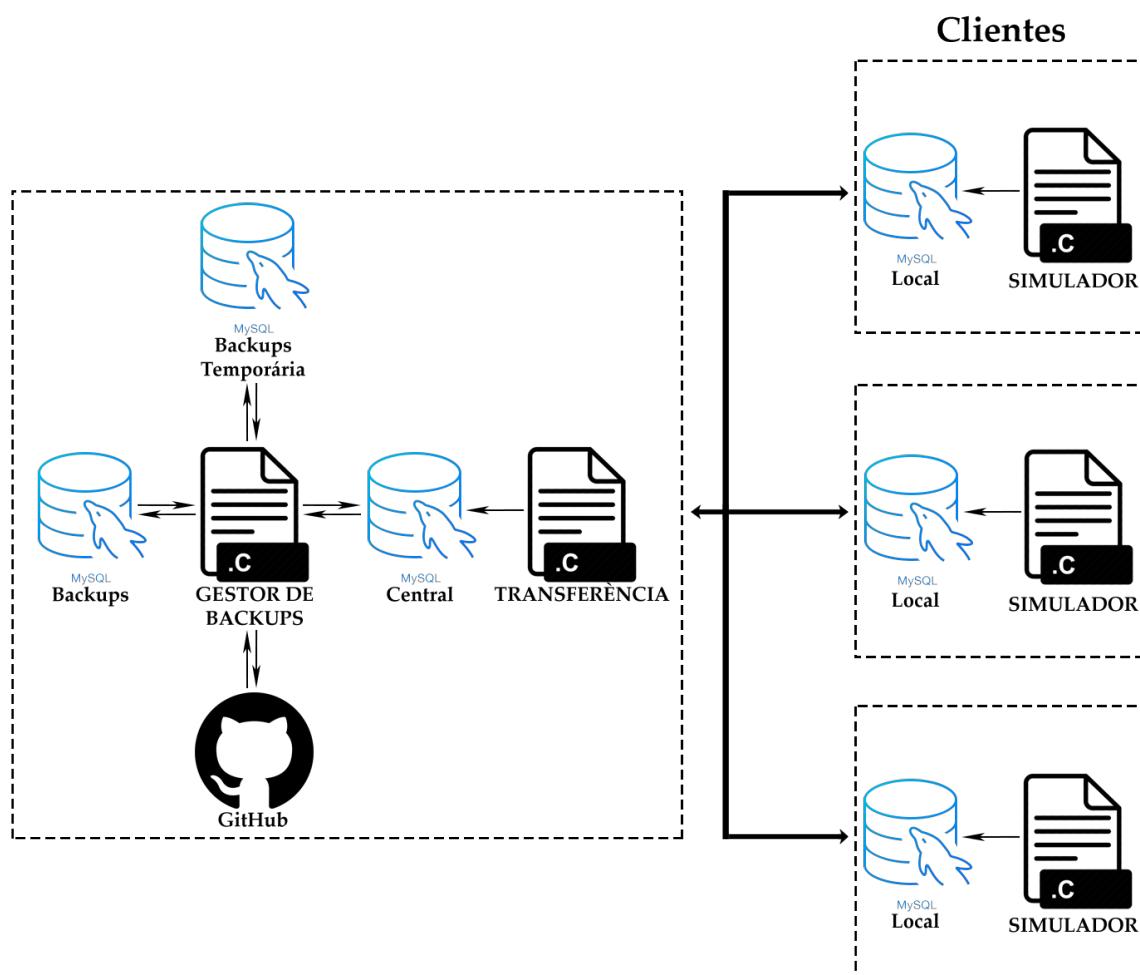


Figura 3.3: Diagrama final da infraestrutura de dados. Os registos gerados nos simuladores são transferidos para a BD central. Estes registos são então processados pelo gestor de *backups* que cria e guarda *backups* provisoriamente num repositório online no *GitHub*

um número referente ao molde, tipo (termómetro, dinamómetro, extensómetro, etc.), nome e descrição. Estes sensores geram registos onde guardam a fase do processo (fecho do molde, enchimento, compactação, abertura e extração), data, hora, milissegundos e o valor num determinado momento.

Analizando este enunciado identificam-se as seguintes entidades com os seus respetivos atributos:

- Tipo
 - ID
 - Nome
- Fase
 - ID
 - Nome
- Clientes
 - ID de cliente único
 - Nome
 - Morada
 - IP
 - Port
- Moldes
 - ID do cliente associado
 - ID de molde único
 - Nome
 - Descrição
- Sensores
 - ID do molde associado
 - Número do sensor
 - ID do tipo de sensor
 - Nome
 - Descrição
- Registos
 - ID do molde associado
 - Número do sensor associado
 - ID da fase do processo
 - Data
 - Hora
 - Milissegundos
 - Valor

Analizando os atributos identificam-se as seguintes chaves primárias, únicas e estrangeiras:

- Tipo
 - Chave primária: ID
 - Chaves únicas:
 1. Nome
- Fase
 - Chave primária: ID
 - Chaves únicas:
 1. Nome
- Clientes
 - Chave primária: ID de cliente
- Moldes
 - Chave primária: ID de molde
 - Chaves estrangeiras:
 1. ID do cliente associado
- Sensores
 - Chave primária: ID do molde associado, Número do sensor
 - Chaves estrangeiras:
 1. ID do molde associado
 2. ID do tipo de sensor
- Registros
 - Chave primária: ID do molde associado, Número do sensor associado, Data, Hora, Milissegundos
 - Chaves estrangeiras:
 1. ID do molde associado, Número do sensor associado
 2. ID da fase do processo

O tipo de sensor e a fase do processo são dicionários ou seja, são entidades que contêm informação predefinida de forma a evitar a adição de dados incorretos nas entidades relacionadas. Os seus dados iniciais são:

- Tipo
 - Termómetro
 - Dinamômetro
 - Extensómetro
 - Vibrômetro
 - Pressão
 - Acelerômetro X
 - Acelerômetro Y
 - Acelerômetro Z

- Fase
 - Fecho do molde
 - Enchimento
 - Compactação
 - Abertura de molde
 - Extração

Ligando as várias entidades entre si identificam-se as seguintes relações com a respetiva cardinalidade:

- 1 cliente tem N moldes
- 1 molde tem N sensores
- 1 sensor tem N registos
- 1 tipo define N sensores
- 1 fase define N registos

Quanto à obrigatoriedade de participação das entidades nas relações anteriores afirma-se:

- Não pode existir moldes sem clientes
- Não pode existir sensores sem moldes
- Não pode existir registos sem sensores
- Não pode existir sensores sem tipo
- Não pode existir registos sem fase

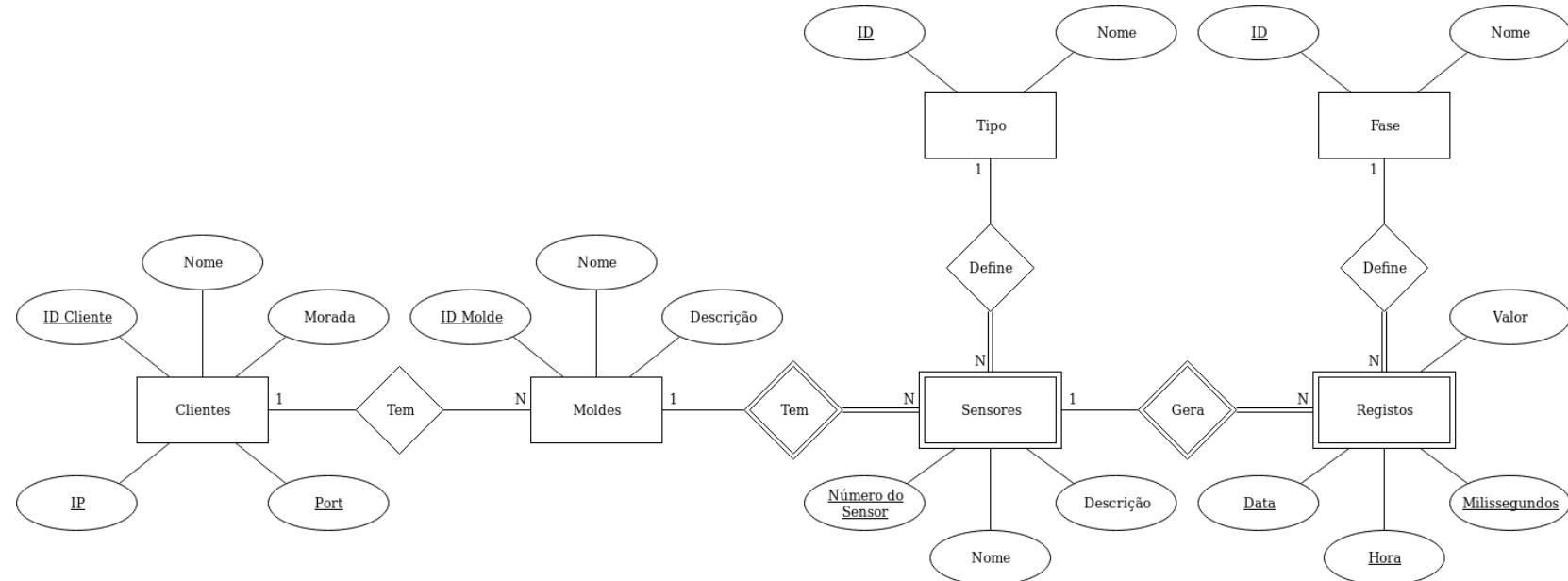


Figura 3.4: Diagrama Entidade/Relação final do projeto realizado a partir da análise de requisitos

3.2.2 Desenho conceptual e esquema lógico

Concatenando a informação descrita na análise de requisitos, obtém-se o DER representado na Figura 3.4. No Apêndice B encontra-se um histórico dos DERs realizados durante o desenvolvimento do projeto

Traduzindo as entidades do diagrama anterior para tabelas e as relações para chaves estrangeiras, obtém-se o esquema lógico da BD representado na Figura 3.5.

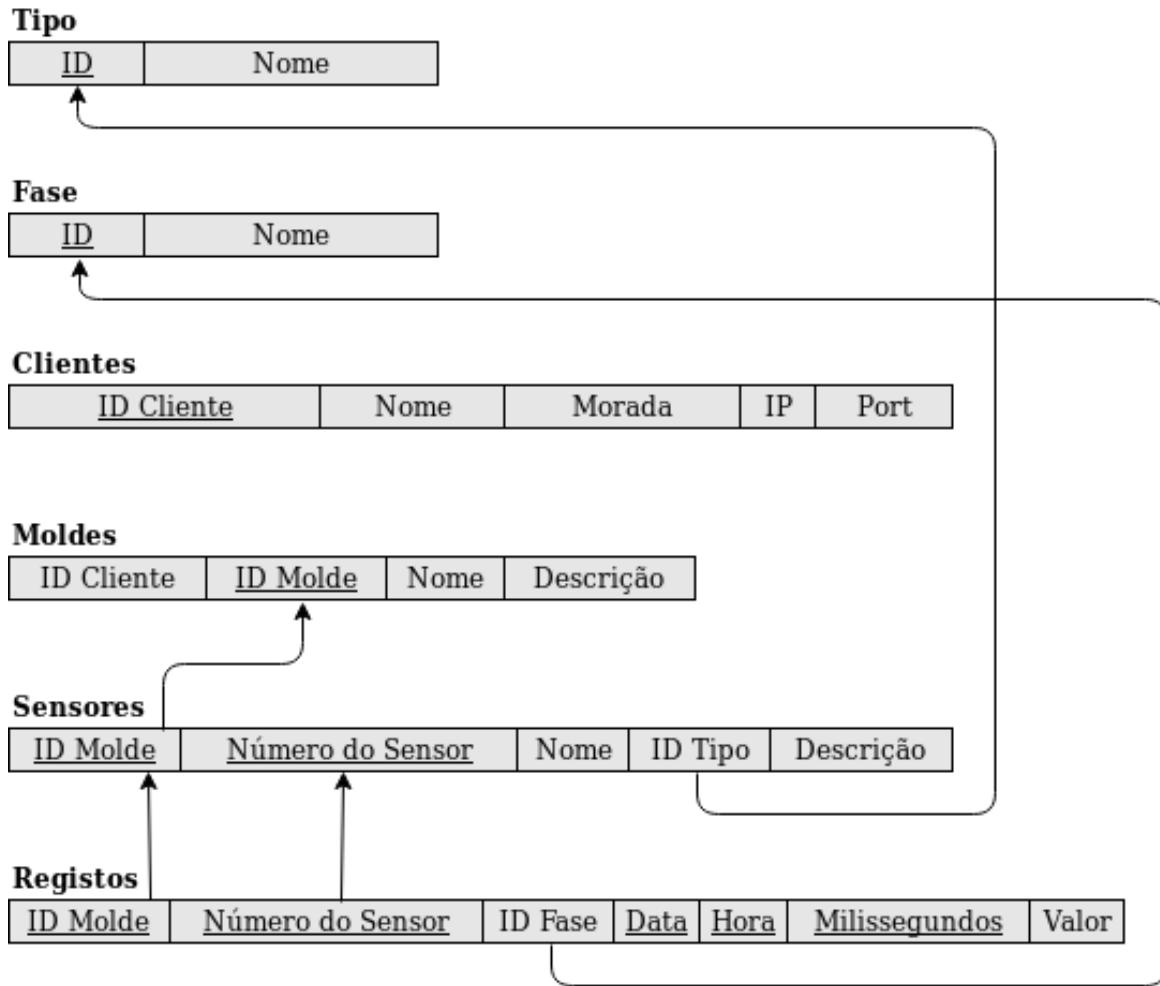


Figura 3.5: Esquema Lógico final do projeto obtido através da análise do DER na Figura 3.4 onde as entidades e os seus atributos se traduzem em tabelas e as relações nas chaves estrangeiras destas

3.2.3 Construção da base de dados

Avançando para a criação das tabelas em *MySQL*, a fim de evitar ambiguidades no nome dos atributos estes são alterados para campos semânticos específicos a cada entidade. Para cada atributo define-se o domínio e a obrigatoriedade como representado na Tabela 3.1.

tipo				
tipo_ID	tipo_nome			
int	varchar(50)			
not null	not null			
fase				
fase_ID	fase_nome			
int	varchar(50)			
not null	not null			
clientes				
cl_ID	cl_nome	cl_morada	cl_IP	cl_port
int	varchar(50)	varchar(100)	varchar(50)	int
not null	not null		not null	not null
moldes				
m_IDCliente	m_ID	m_nome	m_descricao	
int	int	varchar(30)	varchar(200)	
not null	not null			
sensores				
s_IDMolde	s_num	s_tipo	s_nome	s_descricao
int	int	int	varchar(30)	varchar(200)
not null	not null	not null		
registos				
r_IDMolde	r_numSensor	r_fase	r_data_hora	r_milissegundos
int	int	int	datetime	tinyint
not null	not null	not null	not null	not null

Tabela 3.1: Tabelas da BD com os seus atributos e respetivos domínios e obrigatoriedades. Os atributos data e hora da entidade registos são fundidos no atributo r_data_hora graças ao tipo de dados `datetime`

Define-se então nomes para as restrições de integridade definidas pelas chaves das entidades:

- Tipo
 - Restrição chave primária: REPETIDO_ID_TIPO
 - Restrição chaves únicas:
 1. REPETIDO_NOME_TIPO
- Fase
 - Restrição chave primária: REPETIDO_ID_FASE
 - Restrição chaves únicas:
 1. REPETIDO_NOME_FASE
- Clientes

- Restrição chave primária: REPETIDO_ID_CLIENTE
- Moldes
 - Restrição chave primária: REPETIDO_ID_MOLDE
 - Restrição chaves estrangeiras:
 1. MOLDE_MAU_ID_CLIENTE
- Sensores
 - Restrição chave primária: REPETIDO_NUM_SENSOR
 - Restrição chaves estrangeiras:
 1. SENSOR_MAU_ID_MOLDE
 2. SENSOR_MAU_ID_TIPO
- Registros
 - Restrição chave primária: REPETIDO_REGISTO
 - Restrição chaves estrangeiras:
 1. REGISTOS_MAU_ID_MOLDE_SENSOR
 2. REGISTOS_MAU_ID_FADE

Em relação ao comportamento das restrições das chaves estrangeiras identificam-se as seguintes opções:

- ON DELETE NO ACTION ON UPDATE NO ACTION
 - MOLDE_MAU_ID_CLIENTE
 - SENSOR_MAU_ID_TIPO
 - REGISTOS_MAU_ID_FADE
- ON DELETE CASCADE ON UPDATE NO ACTION
 - SENSOR_MAU_ID_MOLDE
 - REGISTOS_MAU_ID_MOLDE_SENSOR

Estes comportamentos definem que quando se tenta atualizar a chave primária de um tuplo (*UPDATE*), se esta tiver tuplos associados a ela, a operação não deve ser realizada pois pode comprometer os dados existentes na BD. No entanto, quando se tenta eliminar um tuplo são definidos dois comportamentos. Como existe muita informação associada a cada cliente, como segurança, não se permite a eliminação de um tuplo desta tabela, se este tiver informação associada a si noutras tabelas. O mesmo se aplica aos dicionários, se se tentar eliminar um tuplo, a informação associada não deve ser comprometida. No caso dos registos, dado que estão associados diretamente a um molde, se um tuplo da tabela moldes for eliminado, os dados dos seus sensores e respetivos regtos devem ser também eliminados para manter a BD limpa e sem informação desnecessária.

Revendo as restrições de integridade criadas pelas chaves estrangeiras, como não se pode criar uma tabela que dependa de outra que não tenha sido ainda criada, define-se uma ordem para a criação das tabelas:

1. Tipo
2. Fase
3. Clientes
4. Moldes
5. Sensores
6. Registos

Este modelo de dados aplica-se a todas as BDs usadas neste projeto com a exceção da BD regulação de procedimentos abordada na Subseção 3.2.4 e temporária local abordada na Seção 4.1.

3.2.4 Base de dados regulação de procedimentos

Com a introdução da aplicação de gestão do sistema surge a necessidade desta comunicar com os programas de transferência e gestão de *backups*. Para este efeito cria-se a BD regulação de procedimentos, com a qual a aplicação comunica de forma a ditar o modo de funcionamento dos programas, como representado na Figura 3.6.

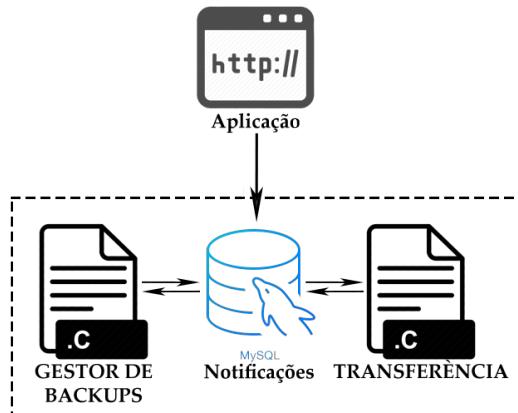


Figura 3.6: Diagrama da comunicação com a BD regulação de procedimentos, onde os programas de transferência e gestão de *backups* verificam parâmetros introduzidos pela aplicação de gestão do sistema

Esta BD contém variáveis globais importantes para a comunicação entre a aplicação e os programas. Define-se as entidades com os seus atributos e respetivos domínios e obrigatoriedade, como representado na Tabela 3.2. As entidades não se relacionam entre si e por isso esta BD não se classifica como uma BD relacional. No entanto, utiliza-se os mesmos termos para descrever o desenvolvimento desta.

atualizar		
a_indice	a_transferencia	a_backups
int	int	int
not null	not null	not null
backups		
b_IDMolde		
int		

Tabela 3.2: Tabela BD regulação de procedimentos com os seus atributos e respetivos domínios e obrigatoriedades

A tabela Atualizar só deve ter um tuplo, considera-se o atributo `a_indice` como chave primária que só pode ser igual a 1 e os atributos `a_transferencia` e `a_backups` só devem ser igual a 0 ou 1. Assim sendo definem-se as restrições:

- Atualizar
 - Restrição chave primária:
 1. REPETIDO_INDICE_ATUALIZAR
 - Restrição valor:
 1. ATUALIZAR_MAU_VALOR_INDICE
 2. ATUALIZAR_MAU_VALOR_TRANSFERENCIA
 3. ATUALIZAR_MAU_VALOR_BACKUPS
- Backups
 - Restrição chave primária:
 1. REPETIDO_ID_MOLDE

Define-se que tabela Atualizar deve ser iniciada com um tuplo com `a_indice` igual a 1, `a_transferencia` igual a 0 e `a_backups` igual a 0. As funcionalidades desta BD são exploradas nas Subsecções 3.3 e 3.4.

3.3 Programa de transferência

Este programa prioriza a conservação dos registos e a sua taxa de transferência. O código foi realizado com vista à portabilidade para outras linguagens de programação. Consiste em várias ideias simples que podem ser observadas nas Figuras 3.7 e 3.8.

Seguindo a estrutura do programa, primeiramente realiza-se uma conexão à BD central, seguida de uma *query* para se saber o número de clientes existentes:

```
SELECT *
FROM central.clientes;
```

Guarda-se o número de tuplos retornados numa variável e termina-se a conexão à BD central. De seguida replica-se o programa múltiplas vezes até se ter um subprograma para cada cliente com recurso à função:

```
fork();
```

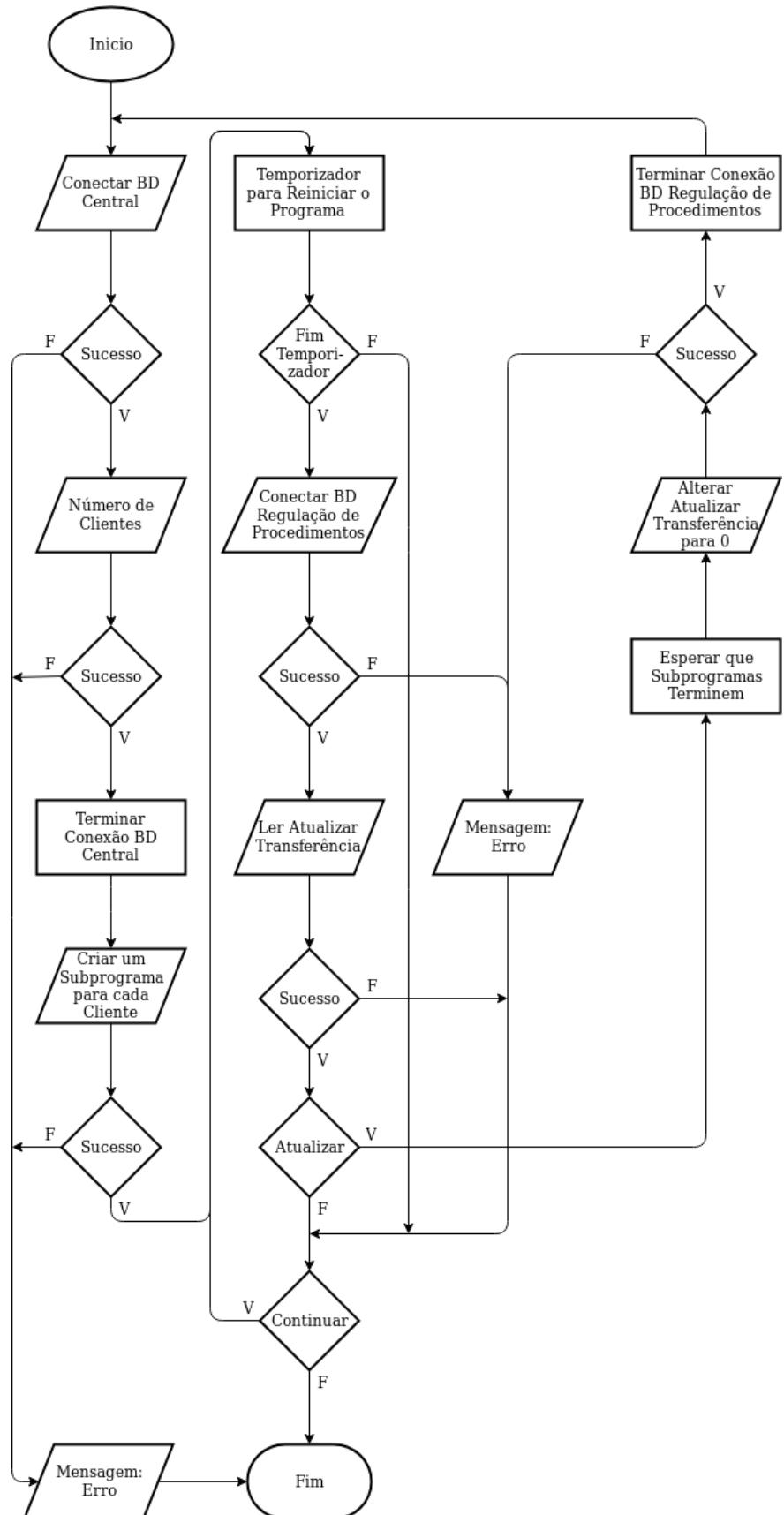


Figura 3.7: Fluxograma do programa principal de transferência, que cria subprogramas

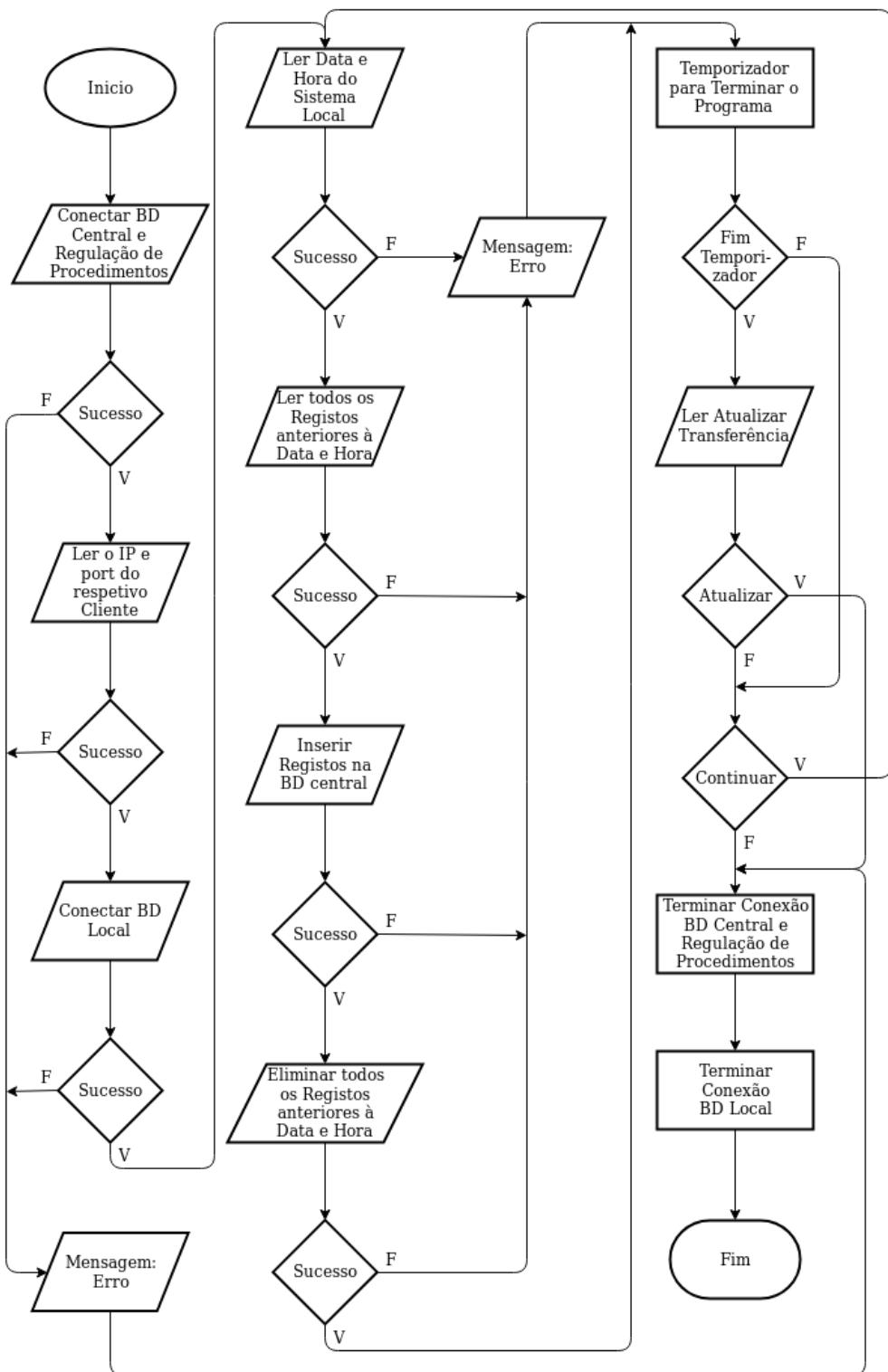


Figura 3.8: Fluxograma dos subprogramas de transferência, que transferirem os registos de cada cliente para a BD central

Atribui-se a cada subprograma o número do cliente a que está associado. Cada um realiza uma conexão à BD central e obtém os respetivos *IPs* e *ports*, necessários para realizar a conexão à BD local do cliente a que está associado:

```
SELECT cl_ID, cl_IP, cl_port
FROM central.clientes
ORDER BY cl_ID;
```

Com as conexões central e local estabelecidas inicia-se um ciclo infinito para a transferência de valores. Com a *query* à BD local:

```
SELECT CURRENT TIMESTAMP;
```

Recebe-se a data e hora atual do sistema local e guarda-se numa variável @datahora_lim. De seguida obtém-se todos os registo anteriores à data e hora:

```
SELECT *
FROM local.registros
WHERE r_data_hora < @datahora_lim
ORDER BY r_data_hora, r_milissegundos,
r_numSensor, r_IDMolde;
```

Esta organização pela data e hora garante uma transferência uniforme de valores. Por predefinição, o sistema retorna os registo um sensor de cada vez, tornando a transferência de valores menos eficiente para efeitos de análise. Por outras palavras, prefere-se saber os registo de todos os sensores num dado instante do que todos os registo de um sensor até ao momento.

Os tuplos retornados são guardados numa *string* @resposta e inseridos na BD central com uma *query* do tipo:

```
INSERT IGNORE central.registros VALUES
(tuplo1),
(tuplo2),
(tuplo3),
...,
(tuploN);
```

Inserir múltiplos tuplos permite uma maior taxa de transferência comparativamente a uma inserção individual. A *string* @resposta tem uma dimensão limitada, se a resposta proveniente da BD local for superior ao tamanho da variável, o programa divide-a em grupos de informação mais pequenos que respeitem a dimensão desta variável. Explicando melhor com um exemplo: se forem retornados 500 tuplos e a capacidade de @resposta for de 100 tuplos, divide-se a informação em 5 repostas.

A escolha da opção IGNORE garante que não são perdidos registo, no caso de serem gerados registo repetidos ou não válidos. Após a transferência ser concluída, os tuplos transferidos da BD local são eliminados:

```
DELETE FROM local.registros
WHERE r_data_hora < @datahora_lim;
```

Sempre que se desejar um subprograma para um cliente novo, o programa de transferência deve ser reiniciado. Isto pode ser realizado manualmente ou atualizando o valor de a_transferencia para 1. Com base num temporizador o programa principal e os subprogramas realizam a *query* à BD regulação de procedimentos:

```

SELECT a_transferencia
FROM reg_proc.atualizar
WHERE a_indice = 1;

```

Se @a_transferencia for igual a 1, os subprogramas terminam após completarem os seus ciclos e o programa principal recomeça do princípio e altera este parâmetro para 0, como demonstrado nas Figuras 3.7 e 3.8:

```

UPDATE reg_proc.atualizar
SET a_transferencia = 0
WHERE a_indice = 1;

```

3.4 Gestão de *backups*

À semelhança do programa anterior, o programa de gestão de *backups* foi realizado com vista à portabilidade para outras linguagens de programação. Este programa está dividido nas componentes de gerar e concatenar *backups*.

A primeira serve para manter a informação armazenada organizada. Gera *backups* segmentados dos históricos de registos da BD central e depois elimina a informação já salvaguardada. Isto aumenta a velocidade da consulta de registos mais recentes e previne que a base de dados central exceda o limite de 4Gb impostos pela versão gratuita do *MySQL*, como referido na Subseção 2.4.3. Em vez de se gerar um *backup* de toda a base de dados, gera-se um *backup* específico para cada molde, que inclui a informação dos seus sensores, registos e cliente a que está associado. Além destes, realiza-se um outro *backup* com a informação dos clientes, moldes e sensores mais atual. Assim, em caso de falha crítica, simplifica-se a recuperação do sistema.

A segunda serve para concatenar os *backups* segmentados, produzidos durante o tempo de funcionamento de um molde, num *backup* total do histórico de registos deste molde. Quando um molde sai de produção pode ser interessante arquivar o seu histórico e esta ação pode ser facilitada se, em vez de serem arquivados dezenas ou centenas de *backups*, for apenas arquivado um *backup* geral com todo o histórico do molde.

Ao contrário do programa anterior, este não executa de forma contínua. Define-se um temporizador para executar cada componente. Utilizam-se temporizadores diferentes: um maior e outro mais pequeno. O temporizador maior fica associado à primeira componente porque esta executa automaticamente e em intervalos espaçados. Por exemplo: podem ser gerados *backups* uma vez por dia ou por semana ou por mês dependendo da quantidade de informação produzida. O temporizador menor fica associado à segunda componente porque esta executa deve executar quando o utilizador deseja. Este deve garantir uma resposta do programa com o menor de latência possível. As Figuras 3.9, 3.10 e 3.11 representam a estrutura do programa.

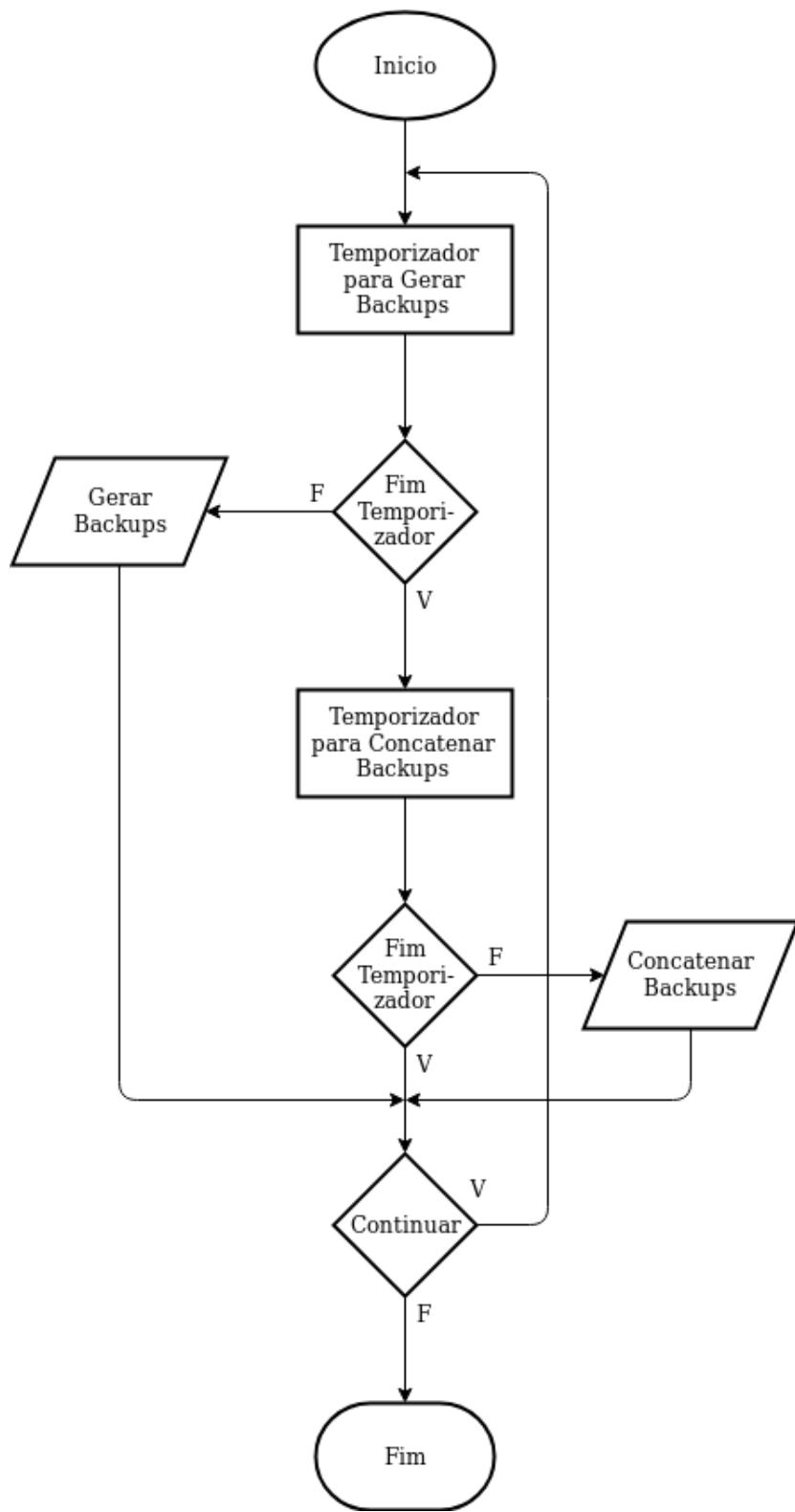


Figura 3.9: Fluxograma do programa principal de *backups*, com os temporizadores para gerar ou concatenar *backups*

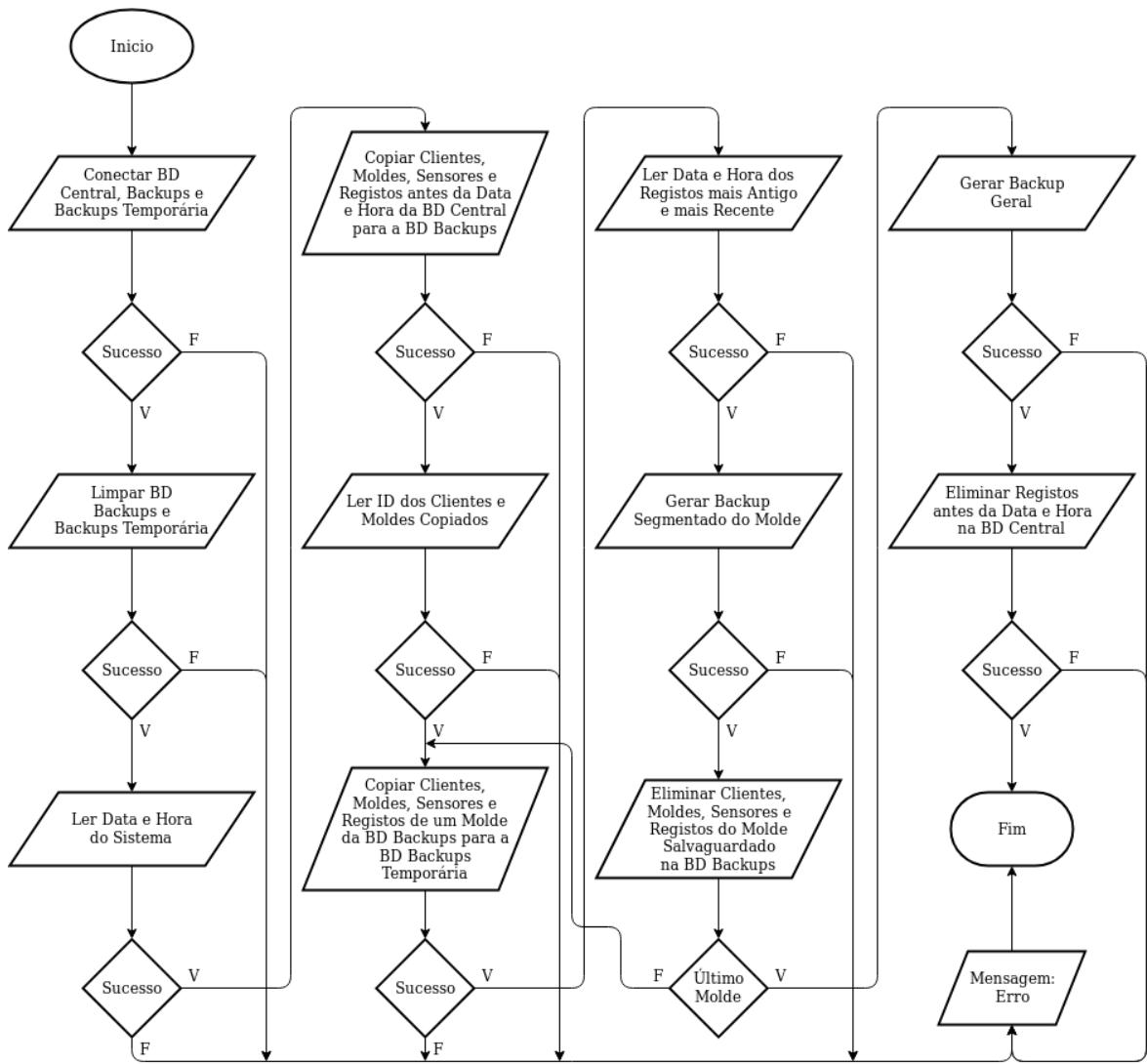


Figura 3.10: Fluxograma da componente de gerar *backups*. Quando esta chega ao fim retorna ao ciclo do programa principal representado na Figura 3.9

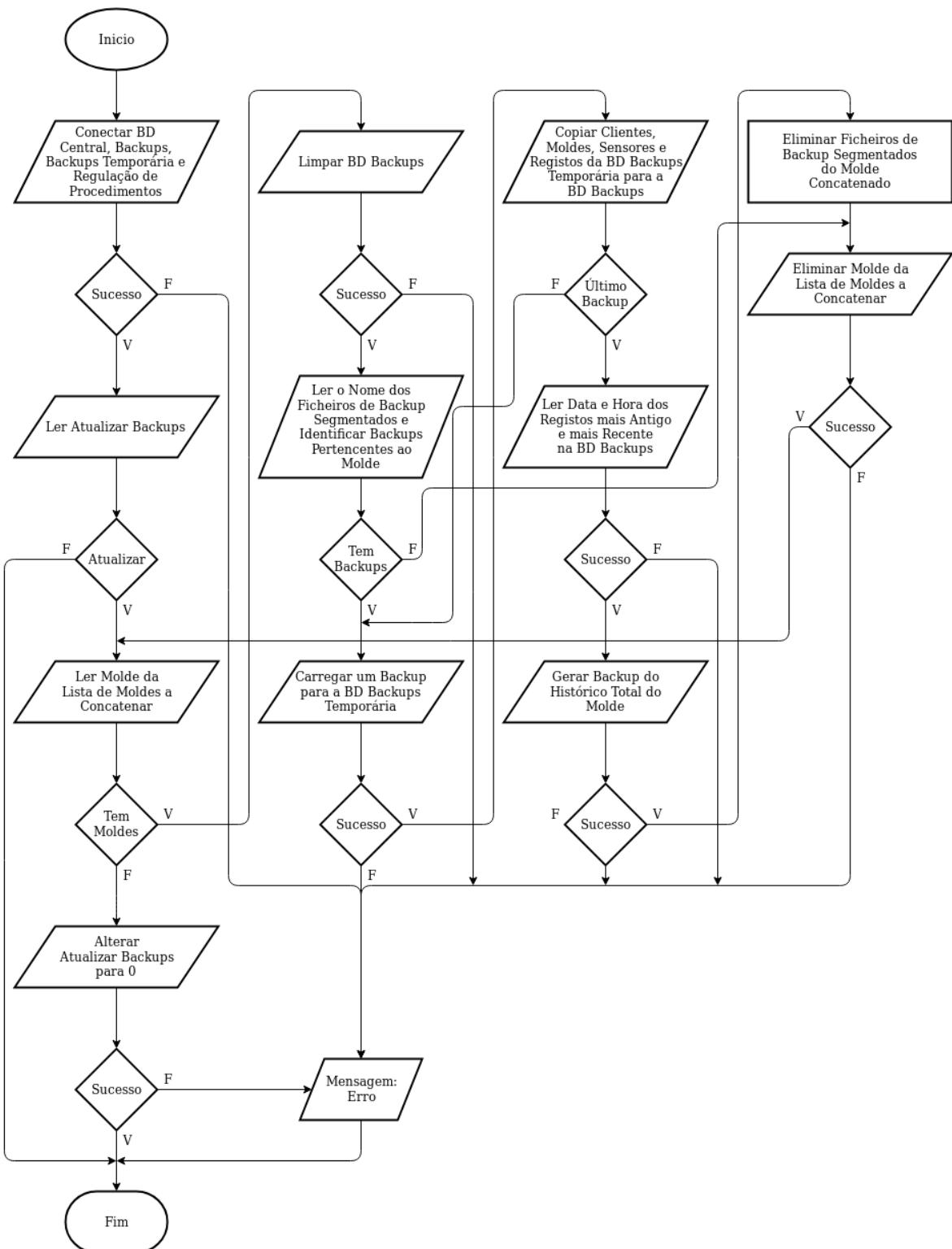


Figura 3.11: Fluxograma da componente de concatenar *backups*. Quando esta chega ao fim retorna ao ciclo do programa principal representado na Figura 3.9

Seguindo a estrutura da componente de geração de *backups* representada na Figura 3.10, realizam-se as conexões às BDs central, *backups* e *backups* temporária. Realiza-se então uma limpeza das BDs *backups* e *backups* temporária para certificar que não existem valores residuais que possam comprometer a informação dos *backups*:

```
DELETE FROM backups.moldes;
DELETE FROM backups.clientes;
DELETE FROM backups_temp.moldes;
DELETE FROM backups_temp.clientes;
```

Com a *query*:

```
SELECT CURRENT TIMESTAMP;
```

Recebe-se a data e hora atual do sistema e guarda-se numa variável @datahora_lim. Para proteger os dados presentes na BD central, realiza-se a cópia dos registo para a BD *backups*:

```
INSERT IGNORE backups.clientes
SELECT *
FROM central.clientes;
INSERT IGNORE backups.moldes
SELECT *
FROM central.moldes;
INSERT IGNORE backups.sensores
SELECT *
FROM central.sensores;
INSERT IGNORE backups.registros
SELECT *
FROM central.registros
WHERE r_data_hora < @datahora_lim;
```

De forma a gerar um *backup* para cada molde filtra-se a informação com base no seu ID:

```
SELECT m_IDCliente, m_ID
FROM backups.moldes;
```

As respostas são guardadas nas variáveis @IDCliente e @IDMolde. De forma cíclica copia-se a informação para a BD *backups* temporária:

```
INSERT IGNORE backups_temp.clientes
SELECT *
FROM backups.clientes
WHERE cl_ID = @IDCliente;
INSERT IGNORE backups_temp.moldes
SELECT *
FROM backups.moldes
WHERE m_ID = @IDMolde;
INSERT IGNORE backups_temp.sensores
SELECT *
FROM backups.sensores
WHERE m_ID = @IDMolde;
INSERT IGNORE backups_temp.registros
```

```

SELECT *
FROM backups.registros
WHERE m_ID = @IDMolde;

```

Após os valores copiados retiram-se informações sobre a data e hora mais antiga e mais recente:

```

SELECT r_data_hora
FROM backups_temp.registros
ORDER BY r_data_hora
LIMIT 1;
SELECT r_data_hora
FROM backups_temp.registros
ORDER BY r_data_hora DESC
LIMIT 1;

```

Os valores retornados são guardados nas variáveis @dataIni e @dataFim. Com estes valores e com a função:

```
FILE *popen(const char *command, const chat *mode);
```

Envia-se para o terminal o comando que gera os *backups* num local predefinido:

```
mysqldump -u backupmanager -pbackup1234 backups_temp >
~/Backups/backup_@IDCliente_@IDMolde_@dataIni_@dataFim.sql
```

Depois do ficheiro ser criado com sucesso elimina-se das BDs *backups* e *backups* temporária a informação relativa ao molde:

```

DELETE FROM backups_temp.moldes;
DELETE FROM backups_temp.clientes;
DELETE FROM backups.registros
WHERE r_IDMolde = @IDMolde;

```

Quando todos os moldes tiverem sido armazenados, a tabela registos da BD *backups* deve estar vazia. Realiza-se então o *backup* da restante informação dos clientes, moldes, sensores, tipo e fase para efeitos de recuperação do sistema:

```
mysqldump -u backupmanager -pbackup1234 backups_temp >
~/Backups/backup_geral.sql
```

Definiu-se um nome estático para este *backup*. Este será rescrito cada vez que o programa executa enquanto os *backups* individuais dos moldes vão acumulando com o tempo. Termina-se o processo apagando os valores armazenados na BD central:

```

DELETE FROM central.registros
WHERE r_data_hora < @datahora_lim;

```

Os *backups* são então transferidos para o repositório *online* no *GitHub*. Esta solução serve apenas para demonstrar a capacidade de mover os *backups* para um sistema diferente. Do ponto de vista prático, não se recomenda guardar informação potencialmente sensível num servidor público.

Na eventualidade de surgir uma falha crítica no sistema central, e este necessitar de ser formatado ou substituído, os registos cessarão de ser transferidos dos sistemas locais. Isto não significa que a informação esteja comprometida, apenas não está a ser transferida. Após re-instalar utiliza-se manualmente no terminal o comando:

```
mysql -u backupmanager -pbackup1234 central <
~/Backups/backup_geral.sql
```

Isto faz com que se recupere a informação dos clientes, moldes e sensores presentes no último *backup* geral mais rapidamente acelerando o processo de restauro, comparativamente a uma inserção manual dos dados.

Passando para a componente de concatenação de *backups*, os ficheiros criados com o comando mysqldump contêm toda a informação das tabelas e funcionam como ponto de restauro. Isto significa que quando se usa o comando de recuperação, a BD selecionada é substituída pela que está no ficheiro .sql. Este comportamento dificulta a concatenação de vários *backups* numa única BD assim sendo, à semelhança da primeira componente, utiliza-se a BD *backups* temporária como BD intermédia para a concatenação de *backups*.

Seguindo a estrutura do programa representada na Figura 3.11, realizam-se as conexões às BDs central, *backups*, *backups* temporária e regulação de procedimentos. Com a *query*:

```
SELECT a_backups
FROM reg_proc.atualizar
WHERE a_indice = 1;
```

Retira-se o parâmetro @a_backups, se este for igual a 1 o programa tem de concatenar *backups*. Identificam-se os moldes a ser arquivados:

```
SELECT b_IDMolde
FROM reg_proc.backups;
```

As respostas são guardadas numa variável @IDMolde. Realiza-se uma limpeza da BD *backups* para certificar que não existem valores que possam comprometer a informação dos *backups*:

```
DELETE FROM backups.moldes;
DELETE FROM backups.clientes;
```

Identificam-se os *backups* referentes ao molde e, de forma cíclica, carregam-se os ficheiros na BD *backups* temporária:

```
mysql -u backupmanager -pbackup1234 backups_temp <
~/Backups/backup_IDCliente_@IDMolde_dataIni_dataFim.sql
```

E copia-se a informação para a BD *backups*:

```
INSERT IGNORE backups.clientes
SELECT *
FROM backups_temp.clientes;
INSERT IGNORE backups.moldes
SELECT *
FROM backups_temp.moldes;
INSERT IGNORE backups.sensores
SELECT *
FROM backups_temp.sensores;
INSERT IGNORE backups.registros
SELECT *
FROM backups_temp.registros;
```

Após os valores carregados e copiados para a BD *backups* retiram-se informações sobre o cliente, a data e hora mais antiga e mais recente:

```

SELECT r_data_hora
FROM backups.registros
ORDER BY r_data_hora
LIMIT 1;
SELECT r_data_hora
FROM backups.registros
ORDER BY r_data_hora DESC
LIMIT 1;

```

Estes valores são guardados nas variáveis @IDCliente, @dataIni e @dataFim. Gera-se então o *backup* total do histórico completo do molde:

```

mysqldump -u backupmanager -pbackup1234 backups >
~/Backups/Arquivo/
backup_@IDCliente_@IDMolde_@dataIni_@dataFim.sql

```

Com este *backup* gerado deixa de ser necessário manter os *backups* segmentados e procede-se à eliminação destes. Estando o processo terminado para um molde remove-se este da lista de moldes a concatenar:

```

DELETE FROM reg_proc.backups
WHERE b_IDMolde = @IDMolde;

```

Estas operações executam de forma cíclica até todos os moldes selecionados serem arquivados. Quando todos tiverem sido arquivados altera-se o parâmetro a_backups de volta para 0:

```

UPDATE reg_proc.atualizar
SET a_backups = 0
WHERE a_indice = 1;

```

3.5 Simulador

Para popular as bases de dados locais foi desenvolvido um simples programa que gera registos. Estes seguem o perfil de uma onda sinusoidal com a expressão:

$$v = O + A \sin\left(\frac{2\pi x}{t} + \gamma\right) \quad (3.1)$$

Onde v é o valor calculado, O o offset da onda, A a amplitude, t o período em segundos, x a hora atual em segundos e γ o desfasamento. Inicia-se uma conexão à BD local e calcula-se a hora atual com recurso às funções:

```

struct tm *localtime(const time_t *timer);
int gettimeofday(struct timeval *tv, struct timezone *tz);

```

Os valores das horas, minutos, segundos e milissegundos são guardados nas variáveis @hora, @min, @seg e @mseg respetivamente. Aplicando a expressão:

$$x = @hora \times 3600 + @min \times 60 + @seg + @mseg \times 0.001 \quad (3.2)$$

Obtém-se a hora atual do sistema em segundos. Com as expressões 3.1 e 3.2 gera-se um valor para cada sensor. Para cada registo gerado atribui-se uma fase do processo. Esta atribuição realiza-se de forma cíclica entre as várias opções disponíveis. Com a informação de cada sensor estabelecida insere-se na BD local com uma *query* do estilo:

```

INSERT INTO registos VALUES
    (molde1, sensor1, fase, NOW(), @mseg, valor11),
    (molde1, sensor2, fase, NOW(), @mseg, valor12),
    (molde2, sensor1, fase, NOW(), @mseg, valor21),
    . . .
    (moldeI, sensorN, fase, NOW(), @mseg, valorIN);

```

3.6 Utilizadores

Para minimizar o risco de erros por parte dos utilizadores que interagem com o sistema, são definidas credenciais com permissões específicas para cada base de dados:

- user, password
- transferencia, transferencia1234
- sensores, sensores1234
- backupmanager, backup1234

As credenciais *user* representam o utilizador padrão, ou seja, aquele que interage com as BDs para gerir os clientes, moldes e sensores, bem como consultar a tabela registos. *transferencia* utilizam-se no programa de transferência de registos para realizar as conexões com as BDs central e locais. *sensores* utiliza-se no simulador para popular as BDs locais e, no futuro, pelo sistema de aquisição de dados que substituirá este simulador. *backupmanager* utiliza-se no programa de gestão de *backups*.

Cada conjunto de credenciais serve um objetivo, a Tabela 3.3 contém as permissões destas definidas em cada base de dados. Estas permissões estabelecem que nenhuma das credenciais estabelecidas pode fazer DROP às BDs o que eliminaria com um comando toda a estrutura e a informação nela contida. Além disto, nenhuma tem permissão INSERT e DELETE simultânea na tabela registos das BDs central e local protegendo o sistema de adulteração dos registos presentes nesta tabela.

De forma a proteger o sistema de acessos não autorizados, define-se também os endereços pelos quais as credenciais podem aceder aos sistemas central e locais. Existem três endereços principais neste projeto: o endereço do sistema central, dos sistemas locais e do servidor Apache que corre a aplicação de gestão do sistema, como representado na Figura 3.12. A Tabela 3.4 contém as limitações das credenciais em cada sistema. Estas definem que os sistemas só podem ser acedidos pelos endereços principais, protegendo a informação de acessos externos.

central

clientes/tipo/fase/ moldes/sensores	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user			X	X	X	X
transferencia						
sensores						
backupmanager			X			
registos	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user			X		X	
transferencia				X		
sensores						
backupmanager			X		X	

local

clientes/tipo/fase/ moldes/sensores	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user			X	X	X	X
transferencia						
sensores						
backupmanager						
registos	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user						
transferencia			X		X	
sensores				X		
backupmanager						

backups e backups temporária

clientes/tipo/ fase/moldes/ sensores/registos	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user						
transferencia						
sensores						
backupmanager			X	X	X	

regulação de procedimentos

atualizar	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user			X			X
transferencia			X			X
sensores						
backupmanager			X			X
backups	CREATE	DROP	SELECT	INSERT	DELETE	UPDATE
user			X	X	X	X
transferencia						
sensores						
backupmanager			X		X	

Tabela 3.3: Tabelas com as permissões das credenciais criadas para as tabelas das BDs

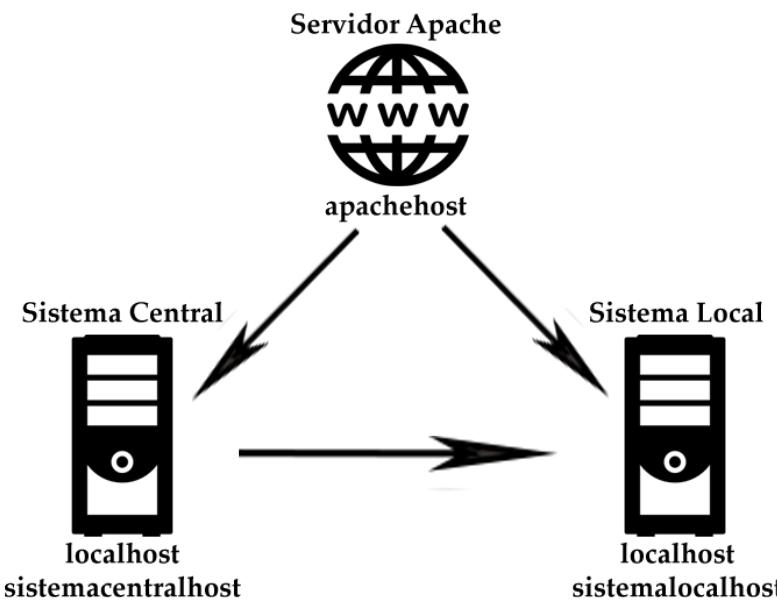


Figura 3.12: Representação da ligação entre os sistemas utilizados no projeto e respetivos endereços

sistema central			
	localhost	apachehost	sistemalocalhost
user	X	X	
transferencia	X		
sensores			
backupmanager	X		

sistemas locais			
	localhost	apachehost	sistematicentralhost
user	X	X	
transferencia			X
sensores	X		
backupmanager			

Tabela 3.4: Tabelas com as limitações dos endereços de conexão das credenciais criadas nos sistemas do projeto. localhost, apachehost, sistemalocalhost e sistematicentralhost representam os endereços do próprio sistema, do servidor *Apache*, dos sistemas locais e do sistema central, respectivamente

Capítulo 4

Aplicação de Gestão do Sistema

A aplicação de gestão do sistema (AGS) foi desenvolvida em ambiente *Web* com o objetivo de ser multiplataforma e permitir acesso remoto e sem recorrer a instalação de *softwares* nos dispositivos dos utilizadores. Corre num servidor *Apache* no sistema central e foi desenvolvida com *PHP*, *JS* e *HTML*. Este capítulo descreve a adaptação da infraestrutura desenvolvida e as várias funcionalidades da AGS.

4.1 Adaptação da infraestrutura

A infraestrutura desenvolvida no Capítulo 3 visa uma utilização a baixo nível. Ainda que funcional, pode haver margem para erros e incoerência dos dados introduzidos manualmente. A AGS minimiza estas incoerências através da instalação de uma nova BD temporária local no servidor local. Aqui os utilizadores têm a liberdade para adicionar, alterar e apagar informação sem consequências na infraestrutura de dados, antes desta ser introduzida nas BDs central e local como representado na Figura 4.1. No Apêndice C encontram-se os diagramas da infraestrutura com todas as bases de dados envolvidas. Como referido anteriormente, esta BD difere das restantes, não contendo em si as tabelas fase e registos.

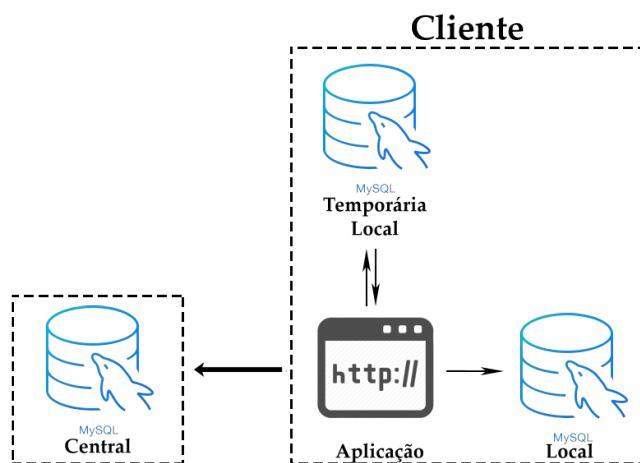


Figura 4.1: Diagrama da ligação AGS-BDs. A AGS comunica com a BD temporária local e depois regista os seus valores nas BDs central e local

4.2 Interface gráfica

A AGS divide-se em quatro partes distintas:

- *Main* - Página principal
- *Login* - Página de acesso
- Consultas
- Administração Local

As páginas *Main*, *Login* e Consultas realizadas para uma utilização geral. A página Administração Local foi realizada para uma utilização local. A primeira visa um uso a partir de qualquer dispositivo e acessível a qualquer momento e a segunda foca-se num acesso local com o objetivo de configurar e definir a informação no servidor local. Por outras palavras, para o utilizador usar as funcionalidades desta página tem de aceder à AGS num *browser* no sistema local, que se situa no cliente.

Instalar um molde é o culminar de um projeto de elevada responsabilidade, esta ideia junto com a criação da BD temporária local serve para melhorar a qualidade da informação introduzida na infraestrutura de dados e diminuir as falhas no processo de instalação deste.

As Figuras 4.2 e 4.3 representam os processos de realizar consultas e de configurar e definir a informação nos sistemas locais dos clientes, respetivamente.

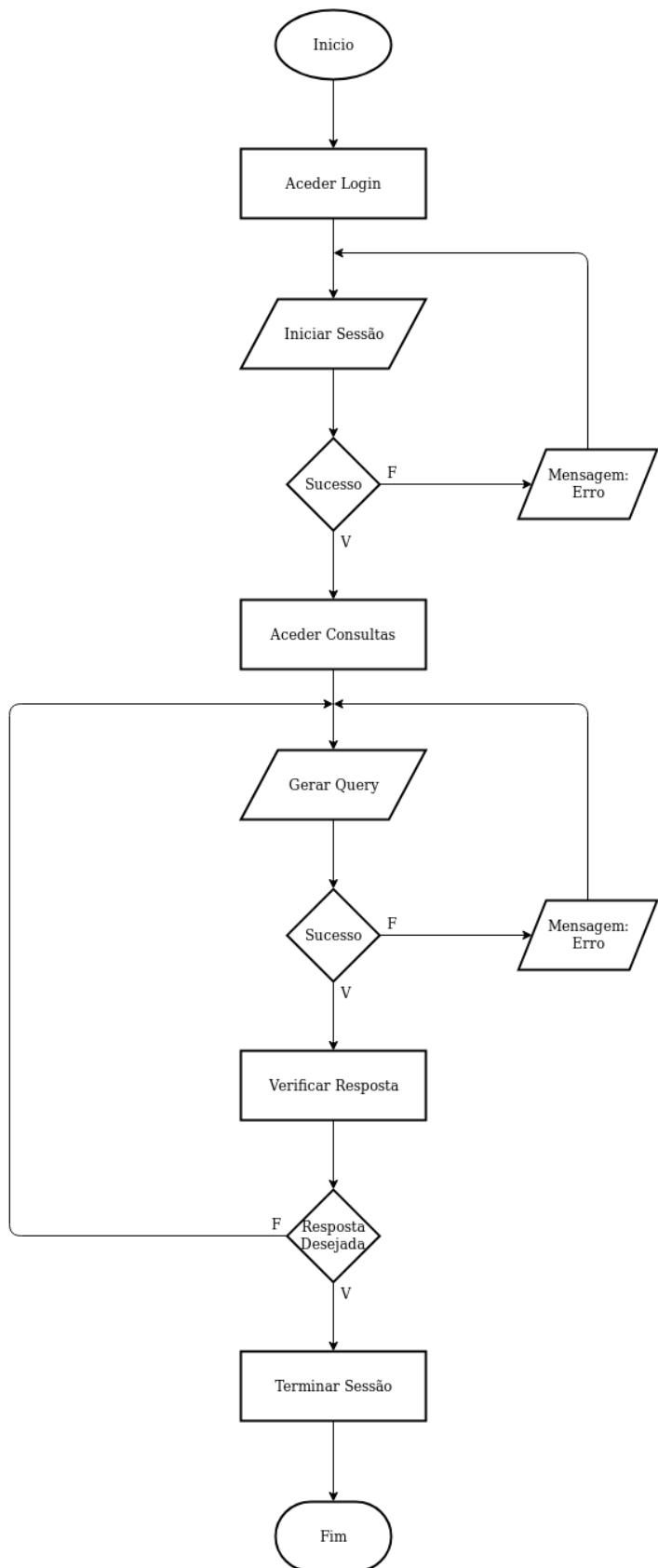


Figura 4.2: Fluxograma do processo de realizar consultas à base de dados e receber respostas

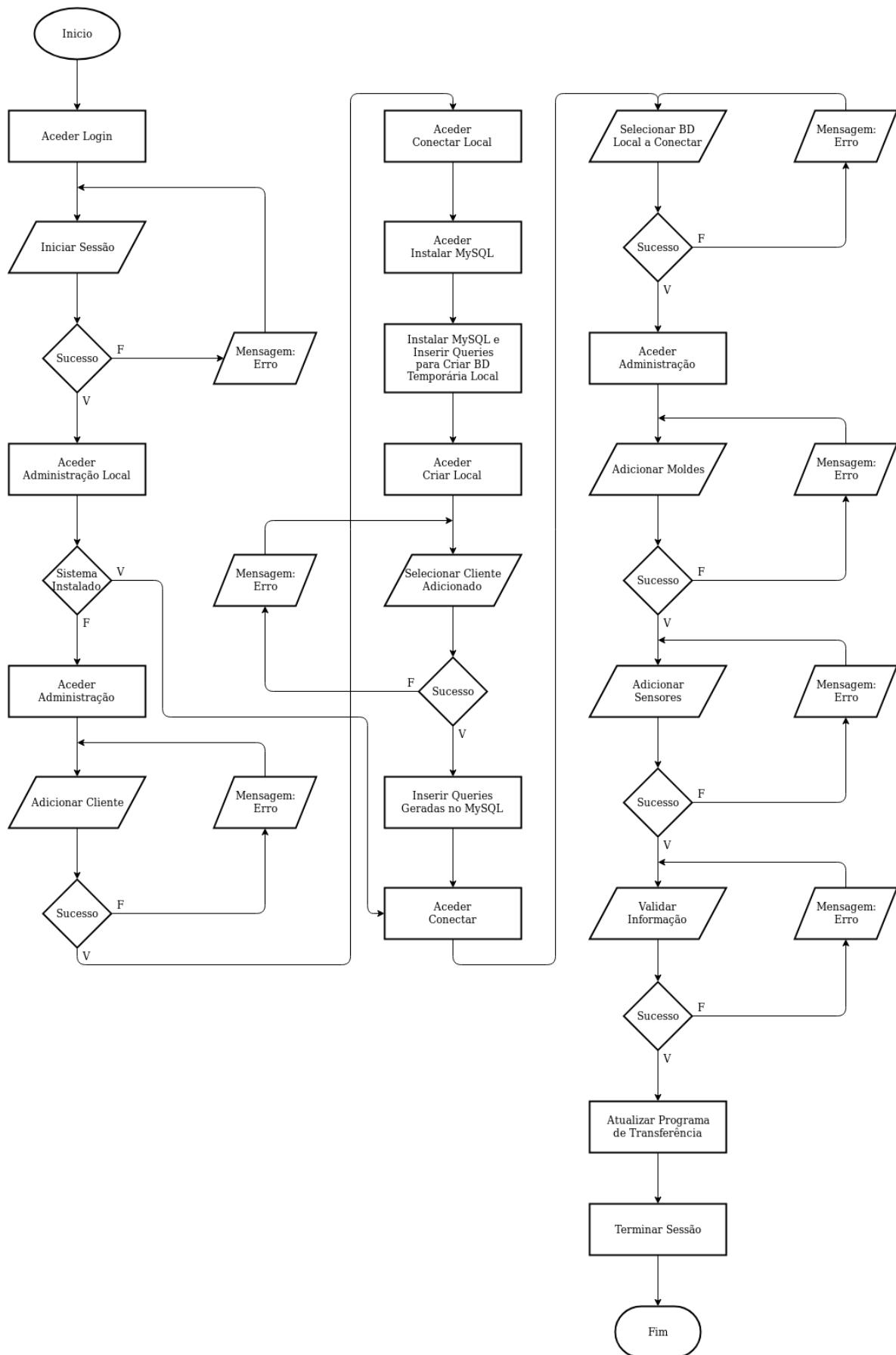


Figura 4.3: Fluxograma do processo de configurar e definir informação no sistema local do cliente

4.2.1 Main



Figura 4.4: Funcionalidades da página *Main* com e sem *login*

Main serve como página principal da AGS. Se não houver sessão iniciada, todas as restantes páginas redirecionam o utilizador para aqui. Contém apenas algumas informações gerais sobre a aplicação.

Iniciar sessão na página de *Login* desbloqueia funcionalidades na AGS, como demonstrado nas Figuras 4.4a e 4.4b. Depois de iniciada a sessão, navega-se com os botões para as páginas de Consultas e Administração Local.

4.2.2 Login

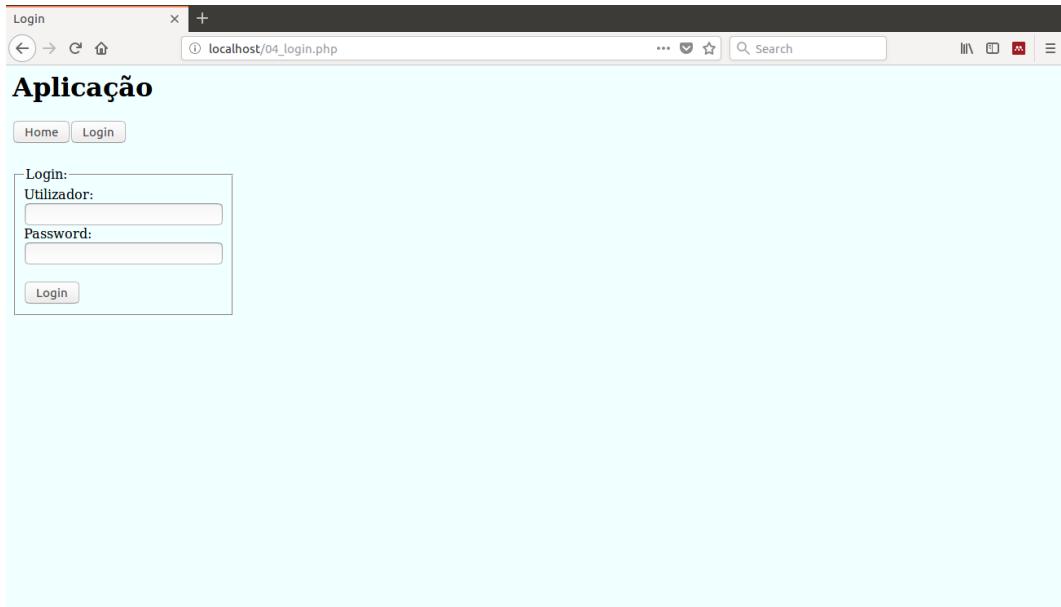
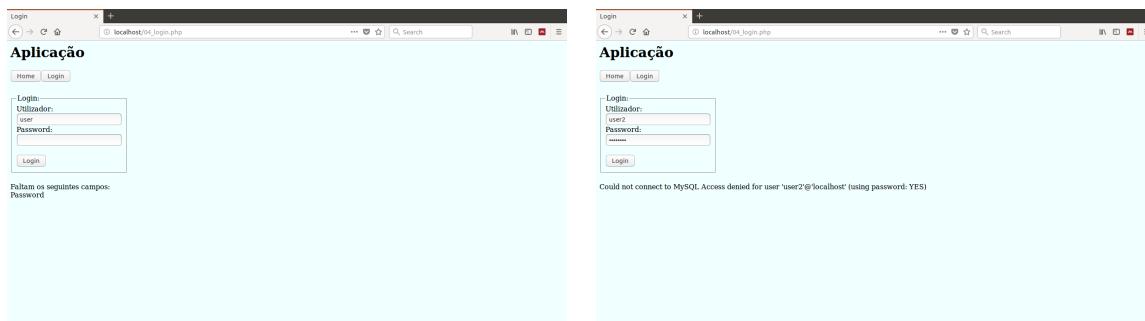


Figura 4.5: Página de *Login* para iniciar sessão na base de dados central

A página de *Login* consiste num simples formulário constituído por duas caixas de texto e um botão, como demonstrado na Figura 4.5. O botão *Login* lê as credenciais introduzidas e realiza uma conexão de teste à BD central validando-as diretamente com *MySQL*. Se as credenciais forem validadas com sucesso redireciona-se o utilizador para a página principal e altera-se o botão de *Login* para *Logout*. Se as credenciais introduzidas não forem suficientes ou válidas são retornados erros de forma a informar o utilizador, como demonstrado nas Figuras 4.6a e 4.6b.

Quando se acede à página com *Logout* termina-se a sessão e redireciona-se o utilizador para a página principal.



(a) Exemplo de erro de falta de informação

(b) Exemplo de erro *MySQL*

Figura 4.6: Exemplos de erros retornados quando introduzidas credenciais não válidas na página *Login*

4.2.3 Consultas

(a) Página de Consultas

10003	9007	1	Pressão	Fecho	2018-06-29 22:26:31	64	3351.4
10003	9008	1	Pressão	Fecho	2018-06-29 22:26:31	64	18579.2
10004	9009	1	Termômetro	Fecho	2018-06-29 22:26:31	64	492.26
10004	9009	2	Termômetro	Fecho	2018-06-29 22:26:31	64	492.26
10004	9009	3	Dinamômetro	Fecho	2018-06-29 22:26:31	64	1243
10004	9009	4	Dinamômetro	Fecho	2018-06-29 22:26:31	64	1243
10004	9009	5	Extensômetro	Fecho	2018-06-29 22:26:31	64	1090.5
10004	9009	6	Extensômetro	Fecho	2018-06-29 22:26:31	64	1011.88
10004	9009	7	Vibrômetro	Fecho	2018-06-29 22:26:31	64	14.29
10004	9009	8	Vibrômetro	Fecho	2018-06-29 22:26:31	64	14.29
10004	9009	9	Pressão	Fecho	2018-06-29 22:26:31	64	1657.34
10004	9009	10	Pressão	Fecho	2018-06-29 22:26:31	64	1657.34
10001	9000	1	Termômetro	Fecho	2018-06-29 22:26:31	65	991.81
10001	9000	2	Dinamômetro	Fecho	2018-06-29 22:26:31	65	1241.45

(b) Exemplo de resposta de consulta

Figura 4.7: Página de Consultas e exemplo de resposta a uma consulta

A página de Consultas assiste os utilizadores sem conhecimentos de *SQL* a criarem *queries* para consultar a BD central. Na Figura 4.7a observa-se várias *checkboxes* e três caixas de texto. As *checkboxes* permitem selecionar os atributos que se desejam consultar na BD; estes atributos são guardados numa variável `@atributos`.

Quando se prima o botão Gerar Query gera-se uma das seguintes quatro *queries*:

```
1-      SELECT @atributos  
        FROM clientes;  
  
2-      SELECT @atributos  
        FROM clientes  
        INNER JOIN moldes ON cl_ID = m_IDCliente;  
  
3-      SELECT @atributos  
        FROM clientes  
        INNER JOIN moldes ON cl_ID = m_IDCliente  
        INNER JOIN sensores ON m_ID = s_IDMolde  
        INNER JOIN tipo ON s_tipo = tipo_ID;  
  
4-      SELECT @atributos FROM clientes  
        INNER JOIN moldes ON cl_ID = m_IDCliente  
        INNER JOIN sensores ON m_ID = s_IDMolde  
        INNER JOIN tipo ON s_tipo = tipo_ID  
        INNER JOIN registos ON s_IDMolde = r_IDMolde  
        AND s_num = r_numSensor  
        INNER JOIN fase ON r_fase = fase_ID;
```

A seleção é feita com base na coluna mais à esquerda a que os atributos pertencem. Explicando melhor com um exemplo: se o utilizador desejar consultar o cl_ID e o cl_nome da coluna clientes gera-se a primeira *query* no entanto, se o utilizador desejar consultar os atributos cl_ID, m_ID e s_num gera-se a terceira *query*, dado que s_num pertence à terceira coluna.

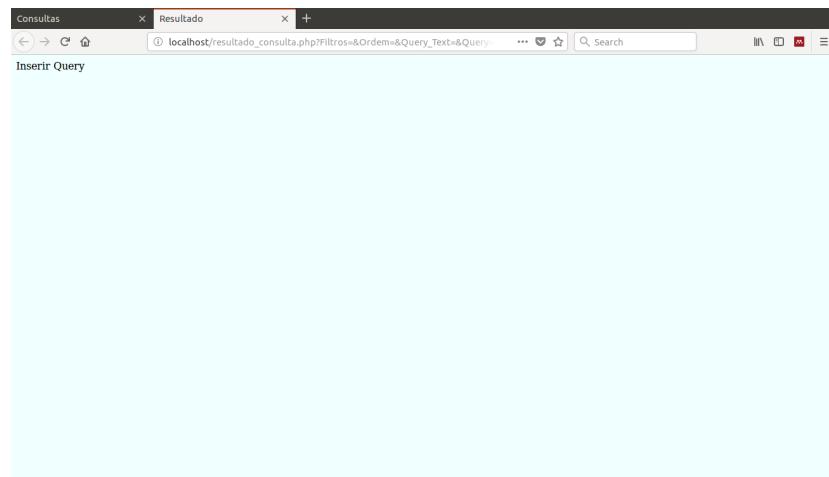
Além destas, foram adicionadas três *queries* específicas quando os atributos tipo_nome, fase_nome e r_data_hora são selecionados sozinhos. As primeiras duas permitem consultar as opções disponíveis nos dicionários e a terceira devolve as datas e horas entre o primeiro e último registos.

As caixas de texto Filtros e Ordem na Figura 4.7a permitem adicionar às *queries* geradas as cláusulas WHERE e ORDER BY, respetivamente. Assim o utilizador pode obter apenas a informação desejada e ordena-la como quiser. Passando com o cursor por cima destas caixa de texto, visualiza-se um exemplo de como se pode filtrar e ordenar a resposta.

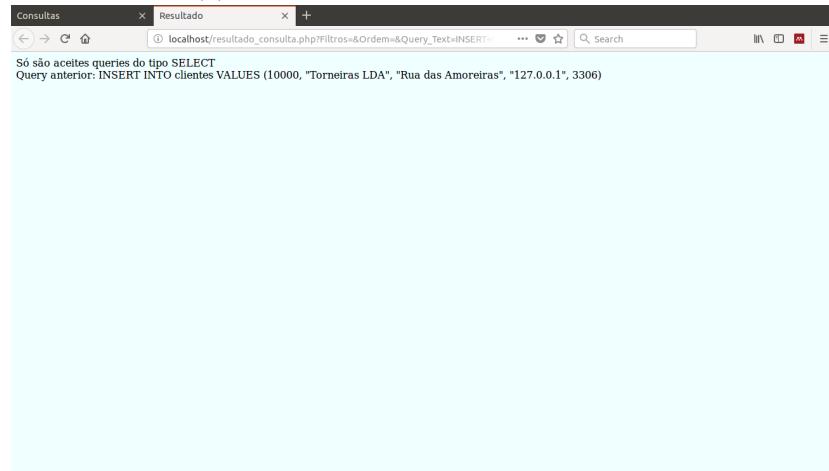
Para os utilizadores com conhecimentos em *SQL* está disponibilizada a caixa de texto *Query* que permite a criação direta de uma *query*. Este campo está limitado apenas para *queries* do tipo SELECT.

Depois da *query* ser gerada retorna-se uma resposta num novo separador como demonstrado na Figura 4.7b. O link desta resposta contém toda a informação da *query* gerada. Este pode ser arquivado ou enviado para outro utilizador sem ser necessário gerar a *query* novamente, isto é útil para *queries* com muitas cláusulas.

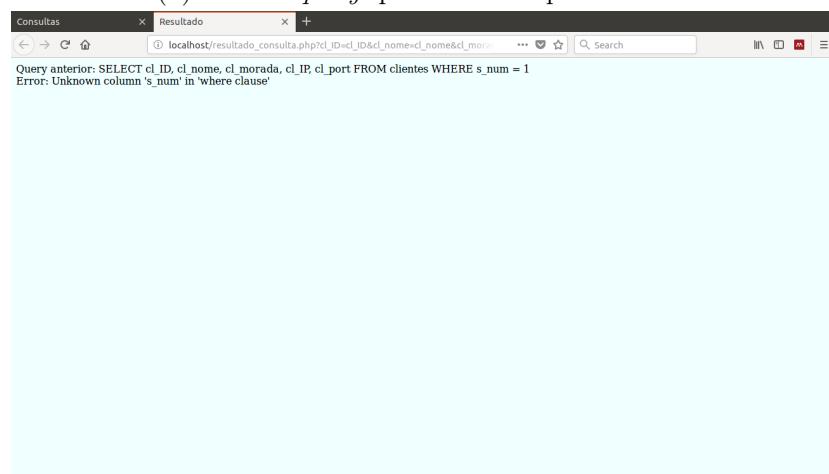
Se a *query* não for válida retorna-se um erro de forma a informar o utilizador, como demonstrado nas Figuras 4.8a, 4.8b e 4.8c.



(a) Erro de informação não introduzida



(b) Erro de *query* que não é do tipo SELECT



(c) Exemplo de erro *MySQL*

Figura 4.8: Exemplos de erros retornados na página de Resposta da Consulta

4.2.4 Administração Local

(a) Sem conexão local

ID	Nome	Nº de Moldes	Nº de Sensores
10000	Torneiras LDA	0	0

(b) Com conexão local

ID	Nome	Nº de Moldes	Nº de Sensores
10000	Torneiras LDA	0	0

Figura 4.9: Funcionalidades da página Administração Local com e sem conexão local

Esta área está dividida em duas componentes distintas. A componente de Administração permite ao utilizador alterar informações sobre os clientes, moldes e sensores. A componente de Conectar Local permite realizar uma conexão ao sistema local do cliente. Algumas funcionalidades da componente de Administração só podem ser acedidas quando for efetuada uma conexão à BD local. Após a conexão ser bem sucedida observa-se informação do cliente, como

demonstrado nas Figuras 4.9a e 4.9b:

```
SELECT cl_ID, cl_nome, COUNT(DISTINCT m_ID),
COUNT(DISTINCT s_IDMolde, s_num)
FROM clientes LEFT OUTER JOIN moldes ON cl_ID = m_IDCliente
LEFT OUTER JOIN sensores ON m_ID = s_IDMolde
GROUP BY cl_ID;
```

Administração

Esta componente está dividida em três áreas: Gestão de Clientes, Gestão de Moldes e Gestão de Sensores. No entanto, as áreas de Gestão de Moldes e Gestão de Sensores e as opções de Alterar Cliente, Atualizar, Ver Moldes e Ver Sensores só podem ser utilizadas quando for realizada uma conexão local, como demonstrado na Figura 4.10.

ID	Nome	Morada	IP	Port	Nº de Moldes	Nº de Sensores
10001	Autocolismos & Company	Rua das sanita, nº 2	127.0.0.1	3306	4	16
10002	Máscaras de Carnaval LDA	Rua de Elm street, nº 13	127.0.0.1	3306	6	30
10003	Aperture Science	Travessa de bolo, nº 404	127.0.0.1	3306	2	2
10004	Black Mesa	Área 51	127.0.0.1	3306	1	10

Figura 4.10: Área de Gestão de Clientes sem conexão local

Quando se accede a esta componente, redirecciona-se o utilizador diretamente para a área de Gestão de Clientes. Sem a conexão local realizada, o utilizador só pode adicionar clientes. O botão Adicionar Cliente executa uma *query* do tipo INSERT com a informação introduzida no formulário. Se as informações introduzidas não forem suficientes ou válidas são retornados erros de forma a informar o utilizador, como na Subseção 4.2.2. Pela AGS não se pode eliminar um cliente depois deste ter sido adicionado à infraestrutura de dados.

Após ser estabelecida uma conexão local, a opção de Adicionar Cliente não pode ser utilizada mas, podem-se utilizar as funcionalidades que estavam bloqueadas, como demonstrado na Figura 4.11.

Figura 4.11: Área de Gestão de Clientes com conexão local

O botão Alterar Cliente executa uma *query* do tipo UPDATE, dando ao utilizador a capacidade de alterar a informação dos clientes. Esta ação está limitada ao cliente presente no sistema local conectado. O botão Atualizar reinicia o programa de transferência de registos. Este realiza uma conexão à BD regulação de procedimentos e altera o parâmetro transferência da tabela atualizar de 0 para 1:

```
UPDATE reg_proc.atualizar
SET a_transferencia = 1
WHERE a_indice = 1;
```

As áreas de Gestão de Moldes e Gestão de Sensores demonstradas nas Figuras 4.12 e 4.13, permitem ao utilizador criar e apagar moldes e sensores, respetivamente. Estes dados são inseridos na BD temporária local, onde o utilizador pode adicionar e eliminar moldes e sensores sem afetar a infraestrutura de dados. Desta forma confirma-se a informação introduzida antes de a inserir permanentemente na infraestrutura. Os botões de Adicionar e Eliminar nestes formulários realizam *queries* do tipo INSERT e DELETE, respetivamente. Se as informações introduzidas não forem suficientes ou válidas são retornados erros de forma a informar o utilizador, como na Subseção 4.2.2.

Figura 4.12: Área de Gestão de Moldes

Figura 4.13: Área de Gestão de Sensores

O botão Arquivar Molde permite ao utilizador iniciar, para o molde selecionado, a componente de concatenar *backups* do programa de gestão de *backups* remotamente:

```
INSERT INTO reg_proc.backups VALUES
(@molde_selecionado);
UPDATE reg_proc.atualizar
SET a_backups = 1
WHERE a_indice = 1;
```

Quando a informação dos moldes e sensores estiver completa, o botão Validar tenta registar os valores presentes na BD temporária local nas BDs central e local. Se a ação não executar com sucesso retorna-se um erro MySQL de forma a informar o utilizador. Se a ação executar

com sucesso, limpa-se a BD temporária local e os valores são registados permanentemente nas BDs central e local, como representado nas Figuras 4.14a e 4.14b.

Depois de inseridos, moldes e sensores não podem ser eliminados via AGS. Esta opção foi removida da AGS para evitar erros, dado que apagar um molde em funcionamento pode fazer com que se percam novos registo.

The figure consists of two side-by-side screenshots of a web application titled "Administração Local".

(a) Dados antes de serem validados: This screenshot shows a temporary database table with one row of data. The columns are labeled: ID Cliente, ID Molde, Número do Sensor, Tipo de Sensor, Nome, and Descrição. The data is as follows:

ID Cliente	ID Molde	Número do Sensor	Tipo de Sensor	Nome	Descrição
10000	5000	1	Termômetro	NULL	Temperatura ponto critico 1

Below the table are input fields for adding new data: ID Molde, Número do Sensor, Tipo (dropdown menu with 'Termômetro' selected), Nome, and Descrição. At the bottom are buttons for 'Adicionar Sensor' (Add Sensor) and 'Validar' (Validate).

(b) Dados após serem validados: This screenshot shows the same temporary database table after validation. The row has been updated with the validation results:

ID Cliente	ID Molde	Número do Sensor	Tipo de Sensor	Nome	Descrição	Nº de Sensores
10000	5000	1	Termômetro	NULL	Temperatura ponto critico 1	2

Below the table are buttons for 'Adicionar Molde' (Add Mold) and 'Eliminar Molde' (Delete Mold). At the bottom are buttons for 'Ver Clientes', 'Ver Moldes', and 'Ver Sensores'.

Figura 4.14: Função do botão Validar, onde os valores da BD temporária local são transferidos de forma permanente para as BDs central e local

Nas várias áreas de gestão existem os botões Ver Clientes, Ver Moldes e Ver Sensores que executam respetivamente as *queries*:

- 1-

```
SELECT cl_ID, cl_nome, cl_morada, cl_IP, cl_port,
COUNT(DISTINCT m_ID), COUNT(DISTINCT s_IDMolde, s_num)
FROM clientes
LEFT OUTER JOIN moldes ON cl_ID = m_IDCliente
LEFT OUTER JOIN sensores ON m_ID = s_IDMolde
GROUP BY cl_ID
ORDER BY cl_ID
```
- 2-

```
SELECT m_IDCliente, m_ID, m_nome, m_descricao,
COUNT(DISTINCT s_IDMolde, s_num)
FROM clientes
INNER JOIN moldes ON cl_ID = m_IDCliente
LEFT OUTER JOIN sensores ON m_ID = s_IDMolde
GROUP BY m_ID
ORDER BY m_IDCliente, m_ID
```
- 3-

```
SELECT m_IDCliente, s_IDMolde, s_num, tipo_nome,
s_nome, s_descricao
FROM moldes
INNER JOIN sensores ON m_ID = s_IDMolde
INNER JOIN tipo ON s_tipo = tipo_id
ORDER BY m_IDCliente, s_IDMolde, s_num
```

Estas *queries* fornecem algumas informações contextuais para facilitar a navegação do utilizador. Sem conexão local, o botão Ver Clientes retorna a informação de todos os clientes

existentes na BD central. Com conexão local, o botão retorna apenas a informação do cliente a que a AGS está conectada.

Conectar Local

A área de Conectar Local na Figura 4.15 permite realizar uma conexão à BD local no servidor do cliente.

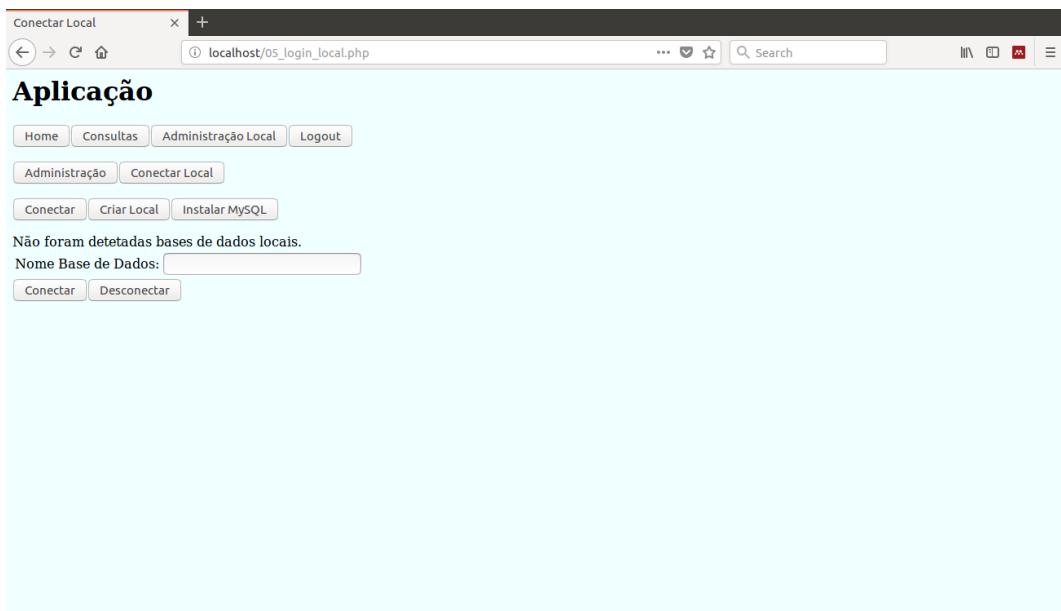


Figura 4.15: Área Conectar Local sem BDs locais instaladas no sistema

Inicialmente o sistema local do cliente não está preparado para fazer parte da infraestrutura de dados. Assim sendo, a área Instalar *MySQL* na Figura 4.16 descreve os passos para instalar o *MySQL* num sistema *Linux*. O botão Copiar Query contém as *queries* necessárias para a criação da BD temporária local e as respetivas permissões.

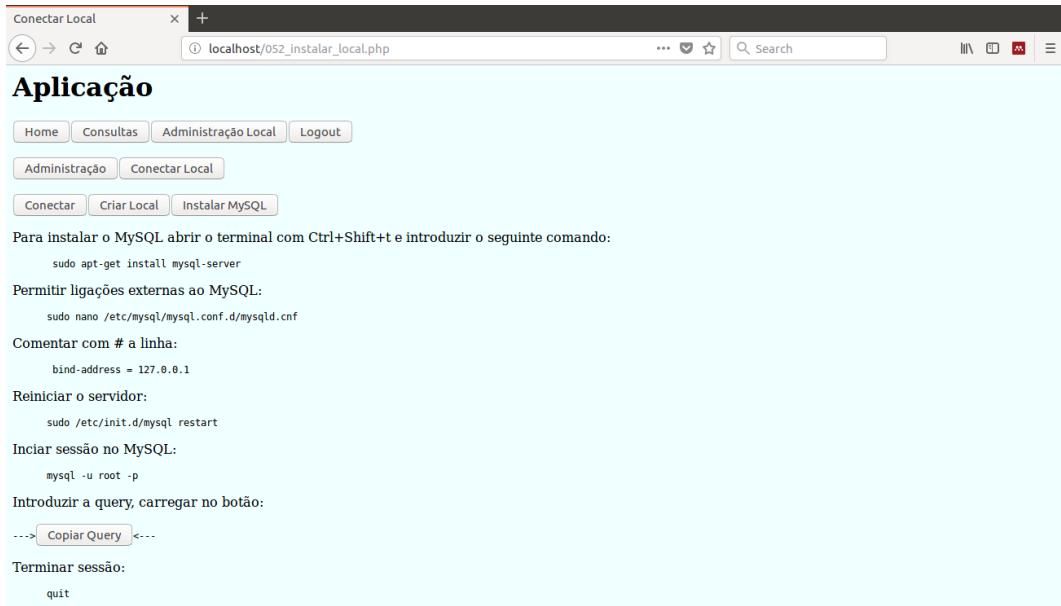


Figura 4.16: Área Instalar MySQL para instalar o *software* com os comandos em *Linux*

A área Criar Local na Figura 4.17 gera *queries* para criar uma BD para um novo cliente. São considerados novos clientes todos os que não tenham moldes associados a si, esta informação obtém-se com a *query*:

```

SELECT cl_ID, cl_nome, cl_morada, cl_IP, cl_port
FROM
  (SELECT cl_ID, cl_nome, cl_morada, cl_IP, cl_port,
COUNT(DISTINCT m_ID) AS n_moldes
FROM clientes
LEFT OUTER JOIN moldes ON cl_ID = m_IDCliente
GROUP BY cl_ID) AS contagem
WHERE n_moldes = 0
  
```

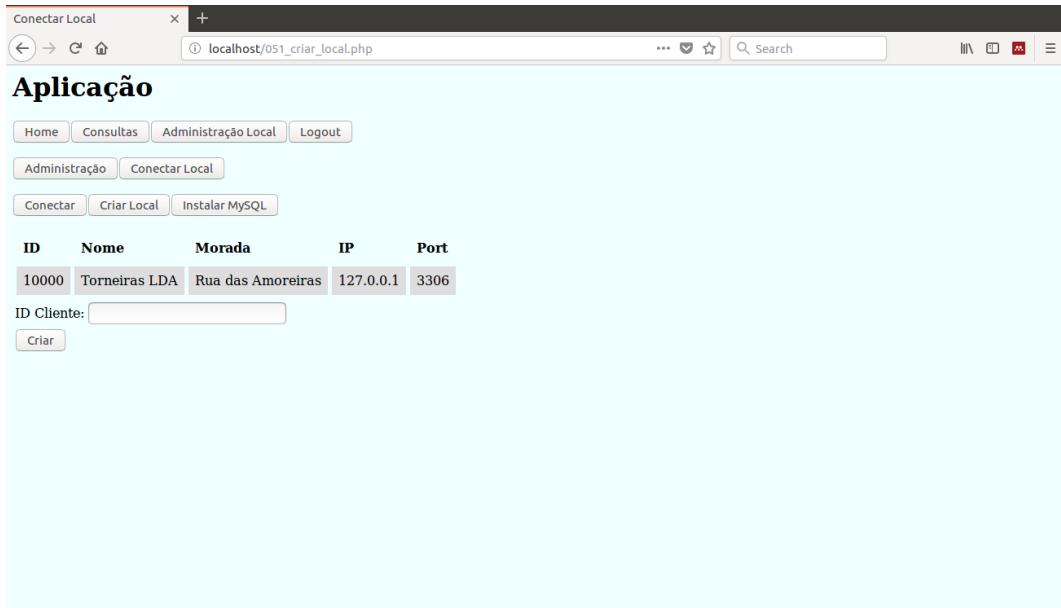


Figura 4.17: Área Criar Local para selecionar cliente adicionado a fim de se gerar *queries* para criar a BD local

Escolhendo um cliente válido o botão Criar gera *queries* para efetuar a instalação da BD local e definir as respetivas permissões. Estas *queries* não são visíveis ao utilizador, assim sendo, o botão Copiar Query copia-as para a área de transferência e estas devem ser introduzidas no MySQL com o comando colar, como representado na Figura 4.18.

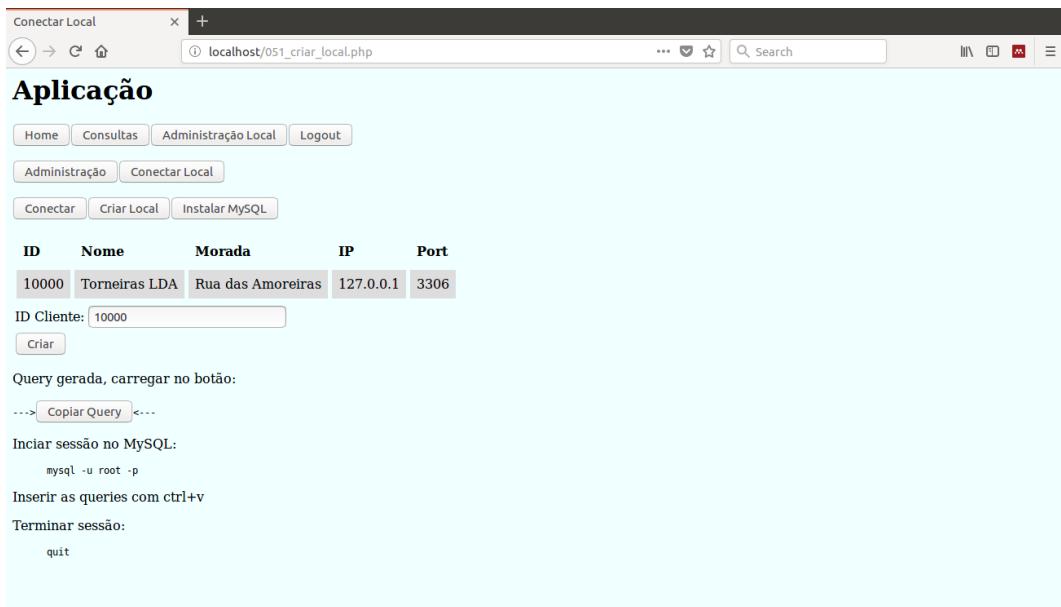


Figura 4.18: Área Criar Local com *queries* geradas para instalar a BD no sistema local. Contém também as permissões para os utilizadores, que só podem ser garantidas via *root*, e informação adicional para completar a infraestrutura de dados

Voltando a área de Conectar, com recurso à *query*:

```
SHOW DATABASES
```

Obtém-se todas as BDs instaladas no servidor local, como demonstrado na Figura 4.19. Do ponto de vista prático, cada cliente só terá uma BD local mas, para efeitos de desenvolvimento do projeto adotou-se esta vertente. O botão Conectar inicia sessão na BD local escolhida e redireciona o utilizador para a página de Administração Local na Figura 4.9b. O botão Desconectar termina esta sessão e redireciona o utilizador também para a página de Administração Local.

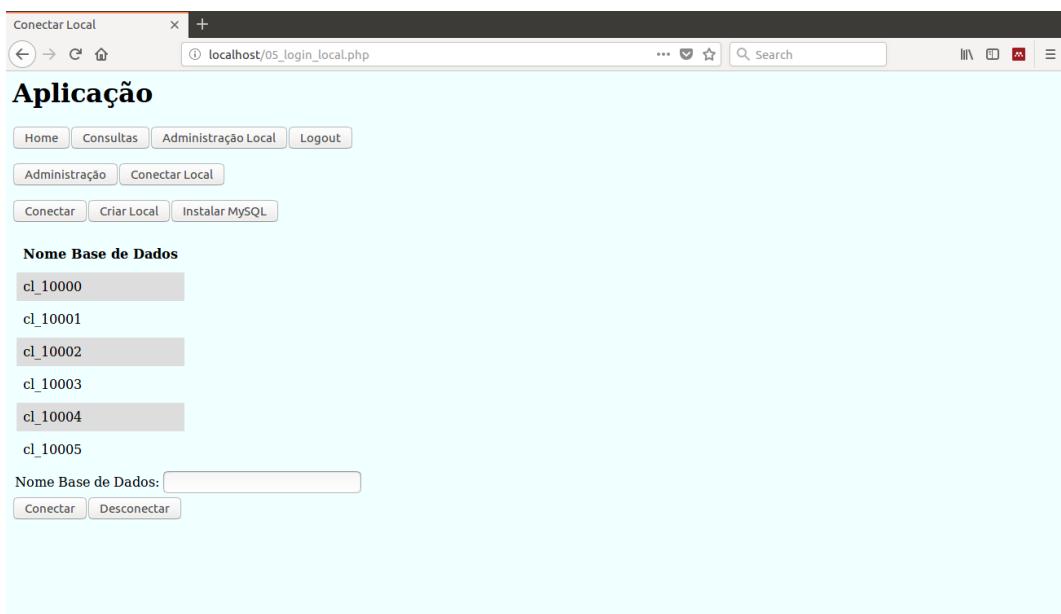


Figura 4.19: Área Conectar Local onde se visualiza as BDs instaladas no sistema

Capítulo 5

Testes de Usabilidade

Para testar as funcionalidades e robustez da AGS, convidaram-se pessoas para realizar testes de usabilidade. Assim, foi possível avaliar o comportamento da AGS, bem como recolher comentários e sugestões para futuros desenvolvimentos.

Foram realizados testes de usabilidade em várias alturas durante o desenvolvimento do projeto. As sugestões recolhidas levaram a AGS até ao seu aspeto atual. Este capítulo descreve a preparação, o plano e os resultados dos testes de usabilidade na versão atual da AGS.

5.1 Preparação dos testes

Na Figura 5.1 está representada a montagem dos testes de usabilidade, que simula a infraestrutura real com um sistema central e outro local. Como referido no Capítulo 4, a AGS está dividida em dois tipos de utilizações. A utilização geral tem de garantir um acesso a partir de qualquer lugar e dispositivo. Assim sendo, pede-se aos utilizadores que, quando possível, utilizem também os seus dispositivos pessoais nestes testes.

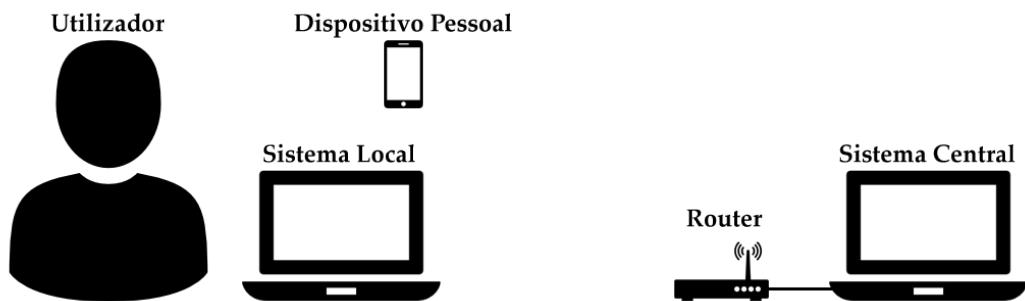


Figura 5.1: Diagrama da montagem dos testes de usabilidade. O sistema central está ligado por cabo ao *router* e os restantes dispositivos estão ligados via *Wi-Fi*. Para o sistema central usou-se um *Lenovo ideapad 110*, para o sistema local um *Lenovo Y50*, usou-se o *router Conceptronic C300BRS4A* e o dispositivo pessoal do utilizador pode ser um *smartphone*, *tablet*, portátil ou computador pessoal

Insere-se no sistema central os dados iniciais presentes no Apêndice D. Este gera registos para clientes, de forma ao utilizador ter informação para consultar a partir do seu dispositivo. O sistema local, no inicio do teste, tem apenas um simulador a gerar registos. O objetivo é simular moldes instalados no cliente, que se encontram a gerar registos e o utilizador deve preparar o sistema local de forma a que estes registos comecem a ser transferidos para o sistema central.

5.2 Plano dos testes

Inicialmente, contextualiza-se o utilizador sobre os objetivos e funcionalidades da AGS. Aproveita-se esta fase para recolher algumas informações sobre utilizador:

- Ocupação profissional
- Grau de acompanhamento do projeto
- Grau de conhecimentos sobre *SQL*

Esta informação permite traçar um perfil dos utilizadores que participaram nos testes.

Passando aos testes propriamente ditos, estes têm duas partes distintas que seguem sensivelmente os processos de realizar consultas e de configurar e definir os sistemas locais dos clientes, como representado nas Figuras 4.2 e 4.3, respetivamente. Em cada etapa do teste anota-se se o utilizador terminou a tarefa:

- Sem problemas - o utilizador não cometeu erros durante a realização da tarefa
- Após erro do utilizador - o utilizador cometeu erros durante a realização da tarefa e as mensagens de erro retornadas permitiram ao utilizador adaptar e terminar a tarefa
- Após erro da AGS - o utilizador encontrou um erro da AGS durante a tarefa que teve de ser resolvido pelo programador de forma a concluir a tarefa
- Não conseguiu terminar - o utilizador encontrou um erro fatal que não o permitiu terminar a tarefa

Terminando com uma avaliação por parte do utilizador se a AGS foi capaz de cumprir os objetivos definidos anteriormente.

5.2.1 Primeira parte - Consultas

Como referido no Capítulo 4, esta parte foi desenvolvida com vista a uma utilização geral. Assim sendo, pediu-se aos utilizadores que usassem, quando possível, um dispositivo pessoal para interagir com a AGS. Anota-se quantos utilizadores utilizaram dispositivos pessoais e qual o dispositivo usaram:

- *Smartphone*
- *Tablet*
- Portátil

Depois do utilizador aceder à AGS pelo dispositivo são pedidas as seguintes tarefas:

1. Iniciar sessão com as credências: user, password
2. Gerar uma *query* onde se observe: ID e nome dos clientes, ID dos moldes, número e tipo dos sensores e fase, data/hora, milissegundos e valor dos registos
3. Filtrar a resposta retornada para o cliente com o ID 10005
4. Copiar o *link* da resposta, terminar a sessão. O utilizador deve reabrir a resposta gerada pelo *link* sem ter de gerar a *query* novamente. Aqui espera-se que o utilizador seja redirecionado para a página principal e que perceba que tem de iniciar sessão e reintroduzir o *link*

Depois do teste ser realizado pede-se aos utilizadores que repitam o processo de forma livre para se obter mais informação sobre a experiência do utilizador. Quando o utilizador tiver terminado recolhem-se comentários e sugestões para implementar em trabalhos futuros.

5.2.2 Segunda parte - Sistema Local

Passando para o dispositivo que simula o sistema local, inicia-se o simulador do cliente 10003 para começar a gerar registos. O utilizador deve preparar o sistema local de forma a que os registos gerados passem a ser armazenados no sistema central.

Quando o utilizador aceder à AGS pelo dispositivo local são pedidas as seguintes tarefas:

1. Verificar que o cliente com ID 10003 não se encontra no sistema
2. Adicionar o cliente com ID 10003 e informação aleatória
3. Instalar e configurar o *MySQL*
4. Criar base de dados temporária local
5. Criar base de dados local do cliente adicionado
6. Conectar à base de dados do cliente adicionado
7. Alterar o cliente para adicionar o IP e port corretos
8. Adicionar os moldes 9007 e 9008
9. Para cada molde adicionar um sensor com o número 1
10. Validar a informação, registando-a permanentemente no sistema
11. Atualizar programa de transferência
12. Verificar que registos estão a ser gerados e transferidos

Mais uma vez , depois do teste ser realizado pede-se aos utilizadores que repitam o processo de forma livre para se obter mais informação sobre a experiência do utilizador. Quando o utilizador tiver terminado recolhem-se comentários e sugestões para implementar em trabalhos futuros.

5.3 Resultados dos testes

Para testar funcionalidades e robustez da AGS, foram convidadas 5 pessoas para participarem nos testes. Em relação aos utilizadores, 20% eram estudantes, 40% eram professores e 40% tinham outra ocupação, como representado na Figura 5.2. Quanto ao grau de acompanhamento do projeto, 20% não conheciam o projeto, 40% conheciam o projeto mas não o seu funcionamento e 40% conheciam o funcionamento do projeto, como representado na Figura 5.3. Uma auto-avaliação dos utilizadores do grau de conhecimentos sobre *SQL* mostrou que 80% não tinham conhecimentos, 20% tinham algum conhecimento e 0% tinham bons conhecimentos, como demonstrado na Figura 5.4.

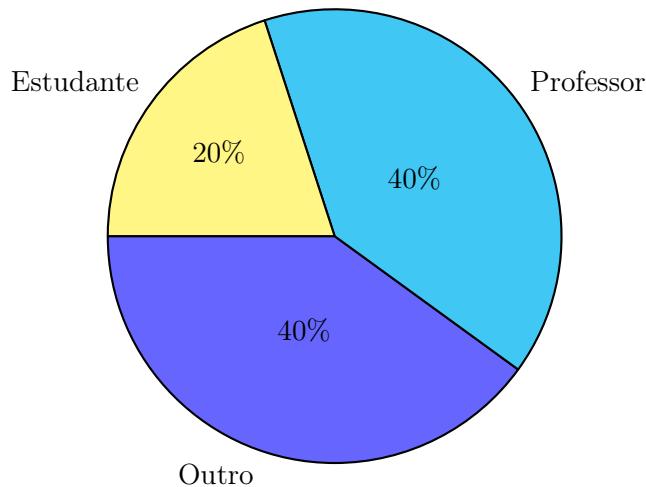


Figura 5.2: Gráfico da ocupação profissional dos utilizadores

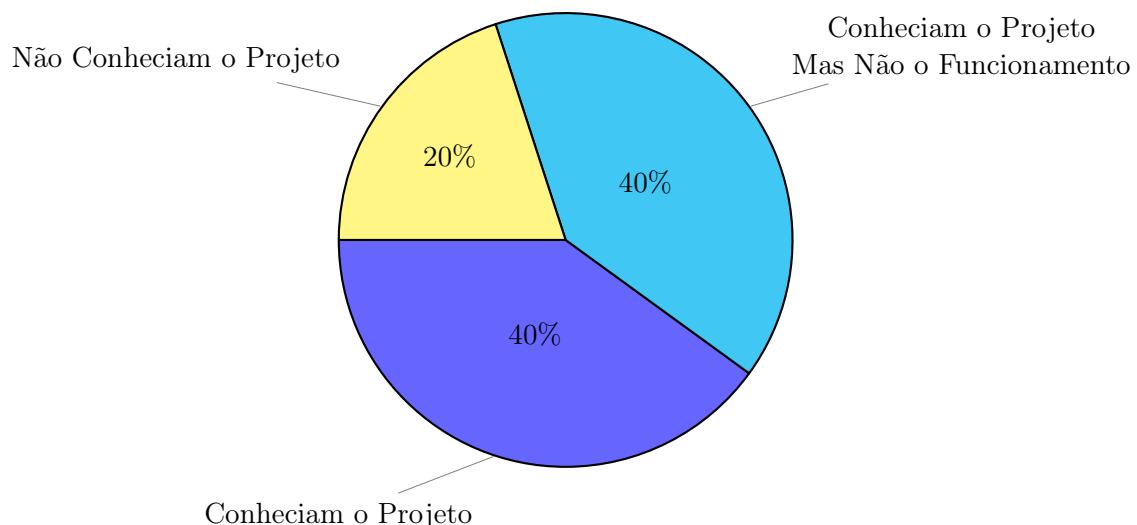


Figura 5.3: Gráfico do grau de acompanhamento do projeto dos utilizadores

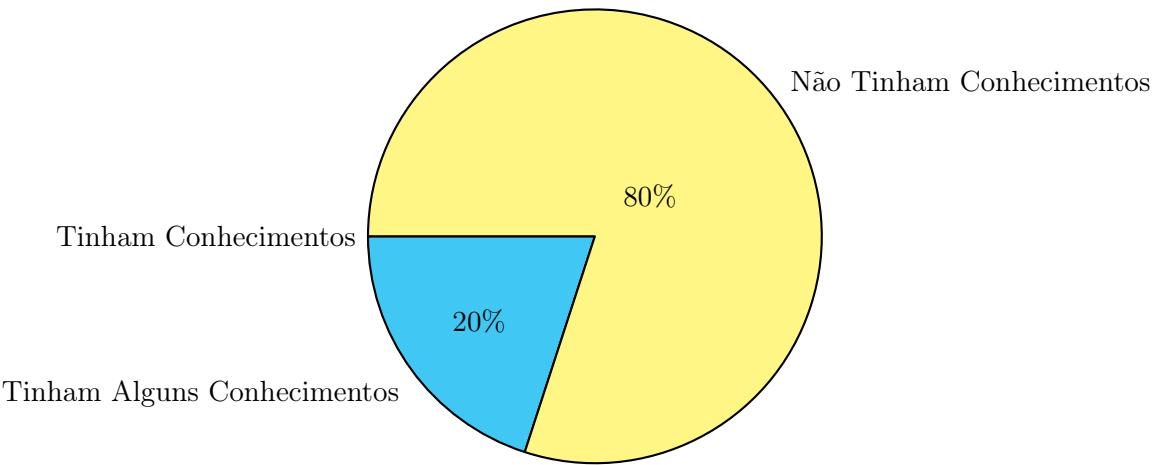


Figura 5.4: Gráfico do grau de conhecimentos sobre *SQL* dos utilizadores

5.3.1 Resultados da primeira parte

Na primeira parte, 100% dos utilizadores usaram um dispositivo pessoal, dos quais 40% eram *smartphones*, 20% *tablets* e 40% portáteis, como demonstrado na Figura 5.5.

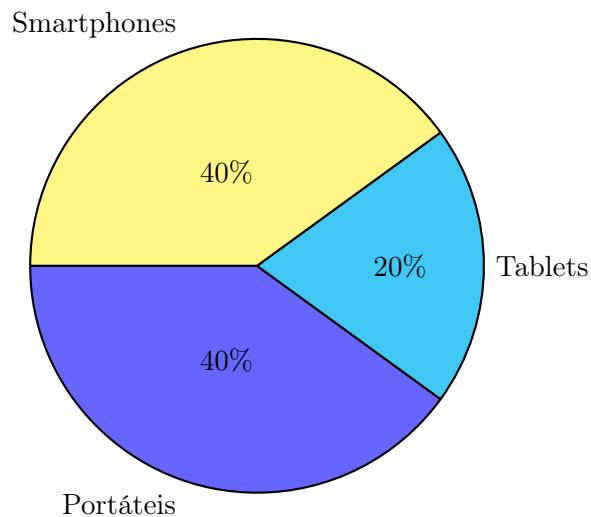


Figura 5.5: Gráfico dos dispositivos pessoais utilizados nos testes

Passando às etapas desta parte do teste, os resultados são apresentados da Figura 5.6 a Figura 5.9:

1. Iniciar sessão , 20% dos utilizadores introduziram credenciais erradas

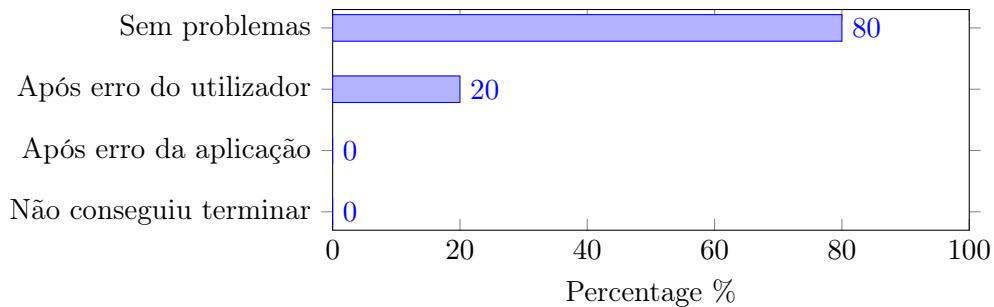


Figura 5.6: Gráfico do resultado de iniciar sessão

2. Gerar uma *query*, 100% dos utilizadores concluíram a tarefa sem problemas

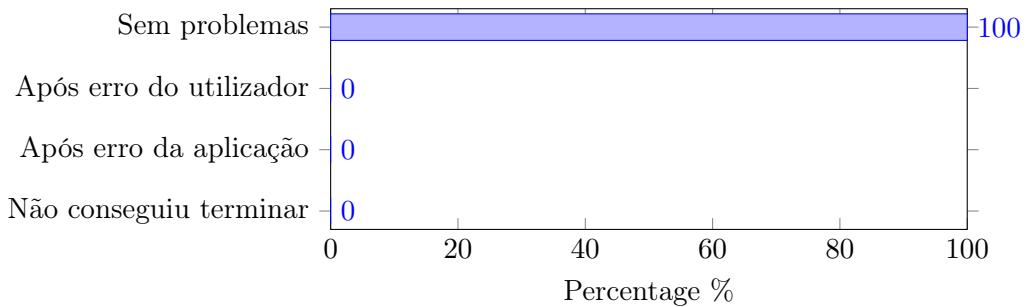


Figura 5.7: Gráfico do resultado de gerar uma *query*

3. Filtrar a resposta retornada, 20% dos utilizadores tiveram dificuldades em filtrar a *query*. Estes utilizadores tinham em comum o facto de estarem a utilizar os seus *smartphones* que não permitem visualizar o exemplo da caixa de texto Filtrar

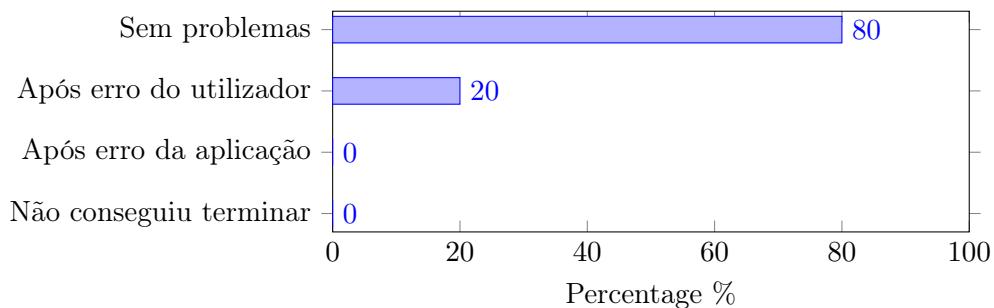


Figura 5.8: Gráfico do resultado de filtrar a resposta retornada

4. Reabrir a reposta anterior pelo *link* após terminar sessão, 100% dos utilizadores concluíram a tarefa sem problemas

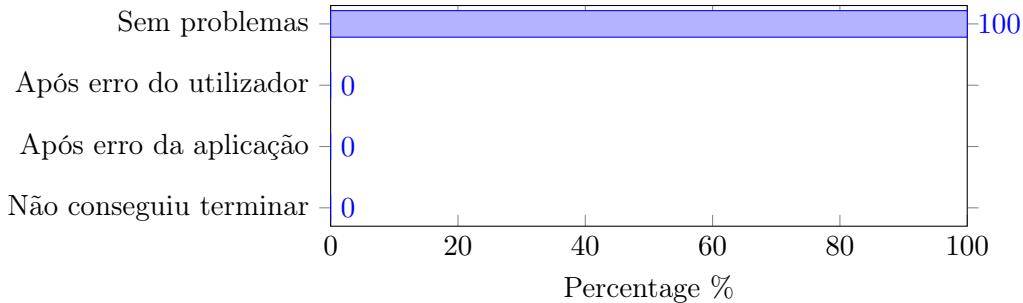


Figura 5.9: Gráfico do resultado de reabrir a resposta anterior pelo *link* após terminar sessão

No final do teste pediu-se aos utilizadores que repetissem o processo de forma livre. Durante toda esta parte do teste, notou-se que nenhum utilizador usou a componente de introduzir uma *query* diretamente. Isto seria de esperar dado que a maioria dos utilizadores não tem conhecimentos de *SQL* e confirma que aplicação cumpre o objetivo de assistir estes utilizadores na geração de *queries*. Para a componente das Consultas foram recolhidas as seguintes sugestões:

1. Tornar o exemplo de ajuda nas caixas de texto Filtrar e Ordenar mais visíveis, especialmente para os utilizadores dos *smartphones*
2. Criar uma *checkbox* para selecionar e desselecionar os atributos de uma coluna
3. Tornar a interface mais simples sem se utilizar `cl_`, `m_`, etc.
4. Quando se utiliza o *link* para reabrir a resposta gerada após terminar sessão, em vez do utilizador ser redirecionado para a página principal, redirecionar o utilizador para página de *Login* e depois abrir diretamente a resposta
5. Dividir a resposta por páginas em vez de tudo corrido
6. Complementar a resposta com gráficos

5.3.2 Resultados da segunda parte

Seguindo as etapas desta parte do teste, os resultados são apresentados da Figura 5.10 a Figura 5.9:

1. Verificar que o cliente não se encontra no sistema , 100% dos utilizadores concluíram a tarefa sem problemas

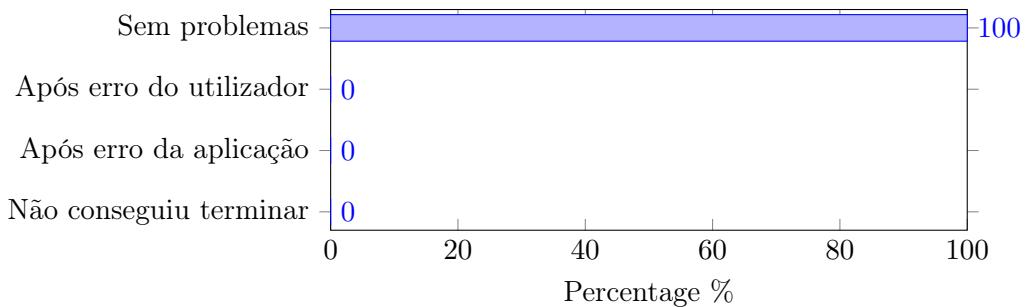


Figura 5.10: Gráfico do resultado de verificar que o cliente não se encontra no sistema

2. Adicionar o cliente, 100% dos utilizadores concluíram a tarefa sem problemas

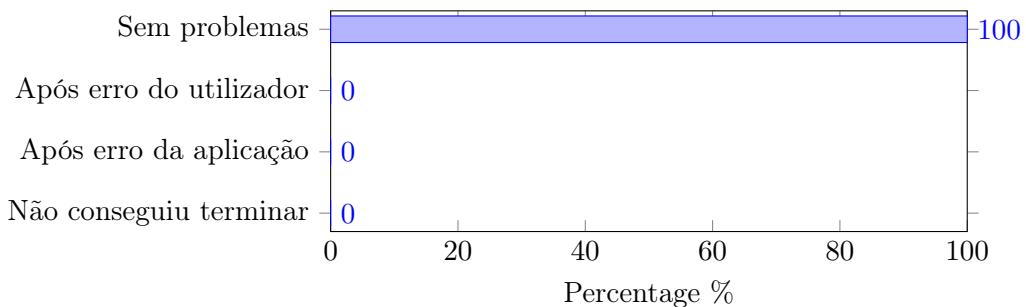


Figura 5.11: Gráfico do resultado de adicionar o cliente

3. Instalar e configurar o *MySQL*, 80% dos utilizadores não conseguiram realizar a tarefa porque, mesmo desinstalando o programa, os seus ficheiros de configuração não eram removidos do sistema

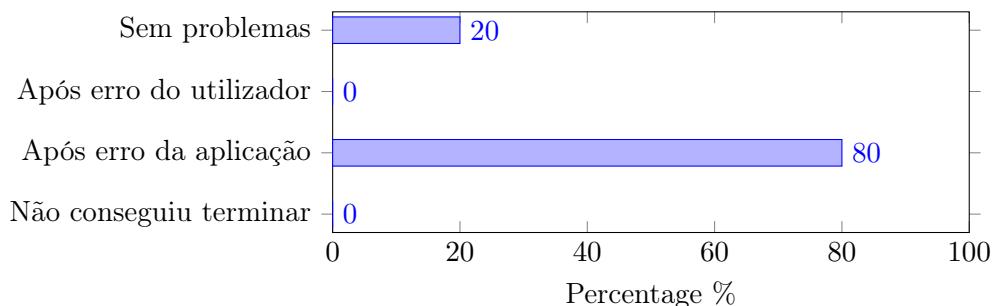


Figura 5.12: Gráfico do resultado de instalar e configurar o *MySQL*

4. Criar base de dados temporária, 100% dos utilizadores concluíram a tarefa sem problemas

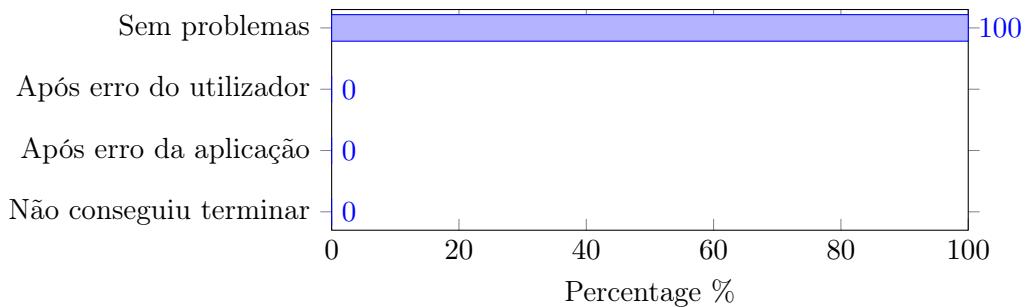


Figura 5.13: Gráfico do resultado de criar base de dados temporária

5. Criar base de dados local do cliente, 100% dos utilizadores concluíram a tarefa sem problemas

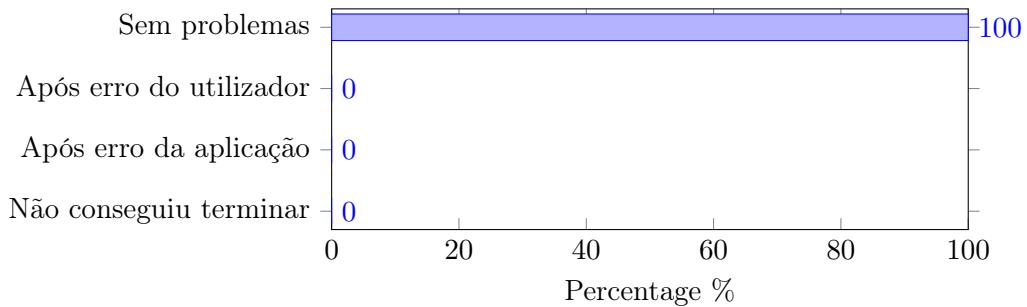


Figura 5.14: Gráfico do resultado de criar base de dados local do cliente

6. Conectar à base de dados local, 100% dos utilizadores concluíram a tarefa sem problemas

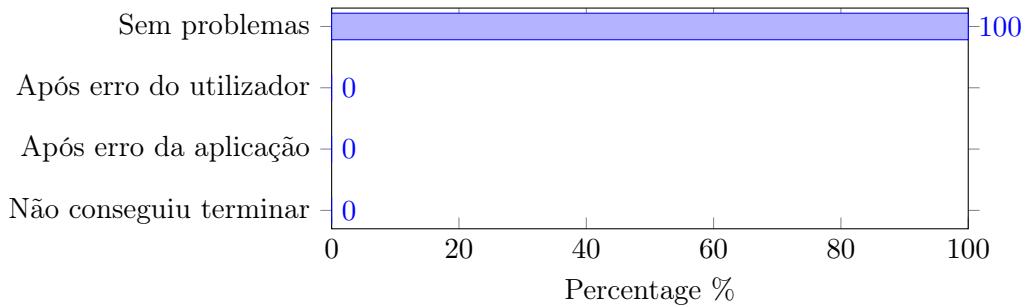


Figura 5.15: Gráfico do resultado de conectar à base de dados local

7. Alterar cliente, 20% dos utilizadores não inseriram toda a informação necessária para alterar um cliente e 20% dos utilizadores detetaram erros na aplicação que tiveram de ser corrigidos a fim de terminar a tarefa

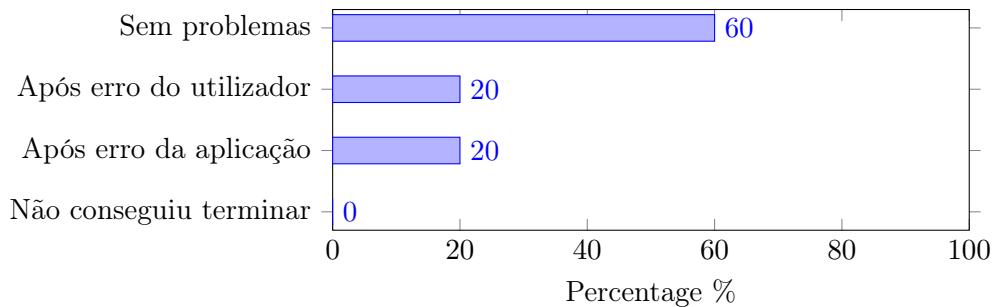


Figura 5.16: Gráfico do resultado de alterar cliente

8. Adicionar moldes, 100% dos utilizadores concluíram a tarefa sem problemas

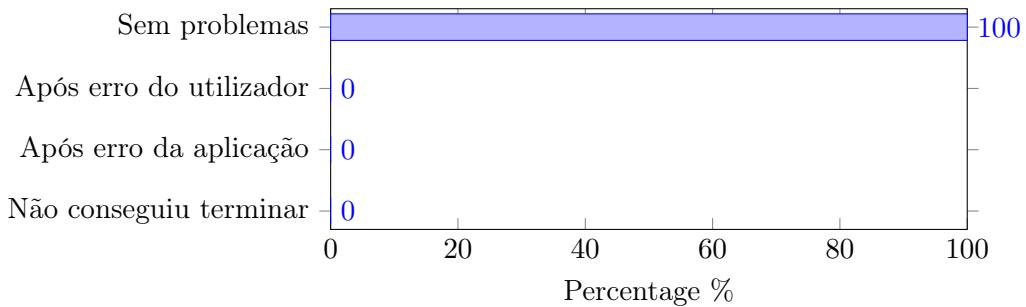


Figura 5.17: Gráfico do resultado de adicionar moldes

9. Adicionar sensores, 100% dos utilizadores concluíram a tarefa sem problemas

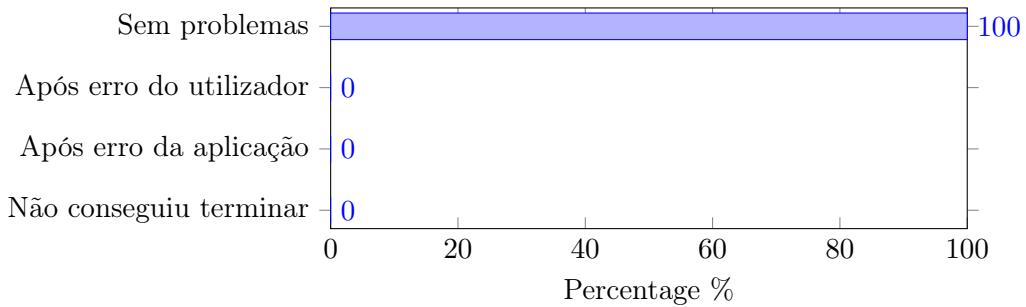


Figura 5.18: Gráfico do resultado de adicionar sensores

10. Validar a informação, 40% dos utilizadores detetaram erros na aplicação que tiveram de ser corrigidos a fim de terminar a tarefa. Um dos utilizadores teve um pico de latência do servidor que causou o fecho da aplicação, deduziu-se que este pico foi falha do *hardware* utilizado

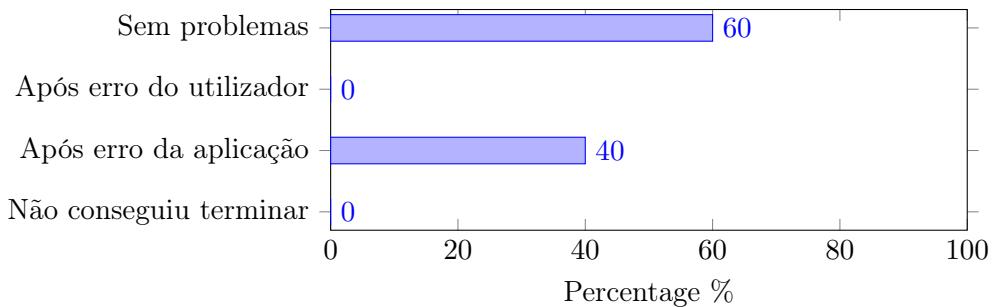


Figura 5.19: Gráfico do resultado de validar a informação

11. Atualizar o programa de transferência, 40% dos utilizadores não conseguiram terminar a tarefa porque foi detetado um conflito entre programas que não se pode resolver durante o teste e 20% dos utilizadores encontraram erros na aplicação que tiveram de ser corrigidos a fim de terminar a tarefa

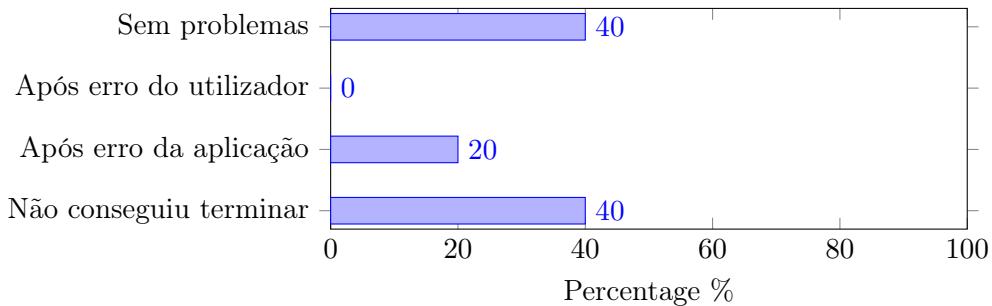


Figura 5.20: Gráfico do resultado de atualizar o programa de transferência

12. Verificar registos a serem gerados e transferidos, 40% dos utilizadores não conseguiram terminar a tarefa por não terem tido sucesso no passo anterior

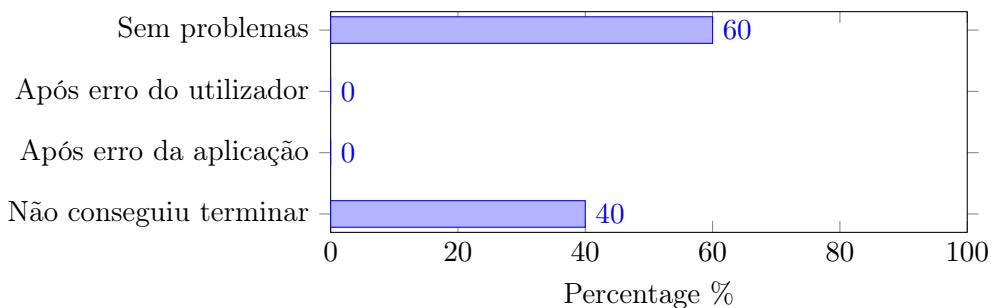


Figura 5.21: Gráfico do resultado de verificar registo a serem gerados e transferidos

Durante a segunda parte do teste, verificou-se que os utilizadores estavam mais à vontade para gerar *queries* comparativamente à primeira parte. Isto demonstra que a aplicação requer alguma experiência por parte dos utilizadores para poder ser usada eficazmente.

Para a componente da Administração Local foram recolhidas as seguintes sugestões:

1. Completar automaticamente a informação para alterar o cliente
2. Adicionar um tutorial na aplicação de como preparar o sistema local
3. Criar avisos nos botões Validar e Atualizar para que o utilizador não realize estas ações por engano

Terminado com a avaliação por parte dos utilizadores, 100% dos utilizadores concordam que a aplicação cumpre os objetivos de assistir na geração de *queries* e de configurar e definir o sistema local do cliente.

Capítulo 6

Conclusões

O desafio proposto consiste em monitorizar sensores remotamente. Para isto definiu-se como objetivos principais desenvolver uma rede de bases de dados relacionais e uma aplicação de gestão do sistema que permitisse interagir com elas. Estes objetivos foram concluídos com sucesso. Este capítulo contém comentários sobre o desempenho da solução desenvolvida e propostas para trabalhos futuros.

6.1 Comentários

6.1.1 Infraestrutura de dados

Em relação à infraestrutura proposta, esta cumpre todos os requisitos propostos de garantir uma transferência segura, confidencial e permanente de valores, criando um histórico dos moldes monitorizados. Consiste numa rede de BDs relacionais com uma central e uma local para cada cliente, o que minimiza a perda de registo em caso de falha de conexão. Num programa de transferência de registo que coordena a transferência dos registo para o sistema central que cria históricos das medições dos moldes. Noutro de gestão de *backups* que limpa a BD central dos registo mais antigos e arquiva-os em ficheiros num repositório *online*. Terminando com simuladores que substituem a aquisição de dados reais e permitem popular as BDs locais.

Neste projeto, para ajudar a distinguir as informações, manteve-se a opção de definir manualmente os IDs dos clientes, moldes e sensores. No entanto, numa solução final, estes IDs devem ser gerados automaticamente de forma incremental para manter as BDs organizadas.

Como definido nos objetivos, esta solução não impõe restrições na instrumentação dos moldes. Para introduzir dados nas BDs locais pode ser utilizado qualquer sistema operativo e linguagem de programação desde que esta tenha protocolos de comunicação com *MySQL* e gere *queries* do tipo:

```
INSERT INTO registros  
VALUES  
(molde, sensor, fase, data_hora, milissegundos, valor);
```

Na realidade esta infraestrutura pode ser utilizada em qualquer contexto de monitorização remota de sensores desde que seja desenvolvido um modelo de dados apropriado.

6.1.2 Programa de transferência

O programa cumpre os objetivos definidos de garantir a conservação e taxa de transferência dos registos gerados, bem como ser simples para permitir a sua portabilidade para outras linguagens de programação.

Este foi desenvolvido como sendo centralizado em vez de descentralizado, como demonstrado na Figura 6.1. Na versão centralizada, o sistema central tem de garantir a transferência de todos os clientes o que pode levar a um maior consumo de processador e consequente perda de desempenho. A versão descentralizada divide este consumo de processador pelos clientes dado que têm de ser os sistemas locais a garantir a transferência dos registos. Além disto, nesta versão, o programa de transferência pode ser mais simples dado que não existe necessidade do programa principal se dividir em subprogramas, podendo-se resumir aos passos representados na Figura 3.8.

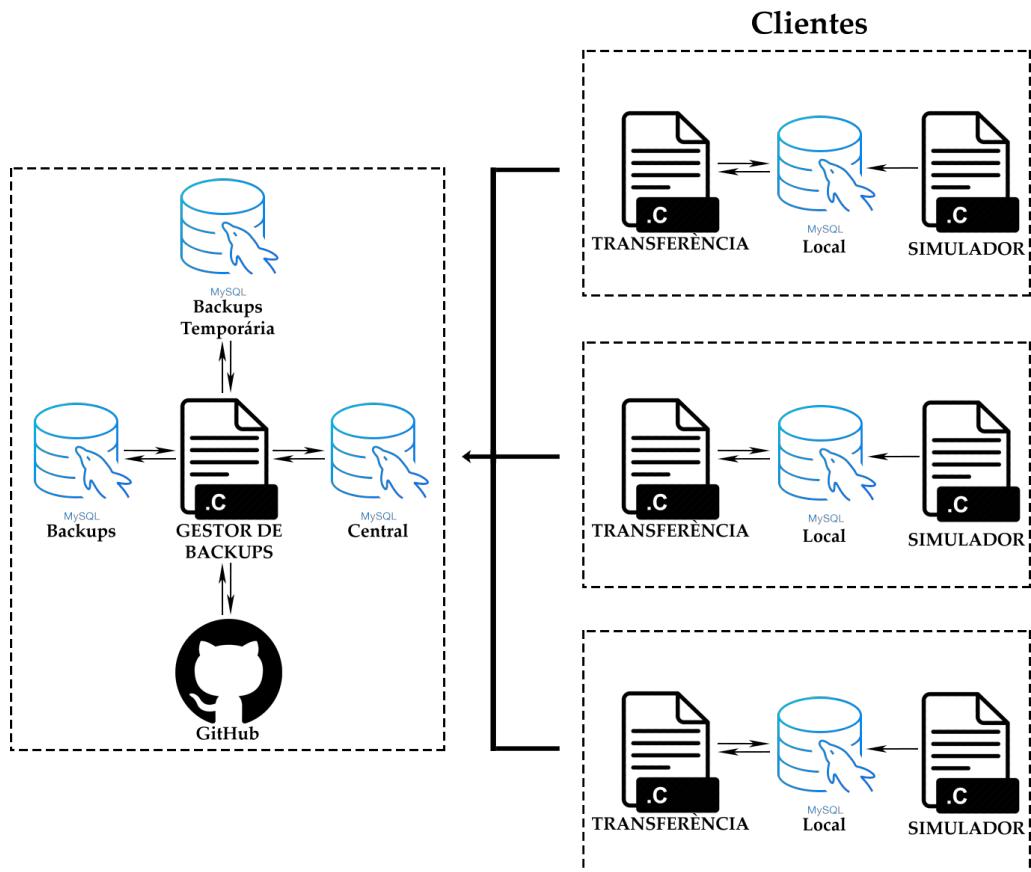


Figura 6.1: Diagrama da infraestrutura com múltiplas BDs locais conectadas a uma BD central, com o programa de transferência de registo descentralizado

No entanto, a versão descentralizada tem desvantagens, o aumento do consumo de processador dos sistemas locais, por causa do programa de transferência, pode comprometer o desempenho da aquisição de dados no caso desta necessitar de uma elevada cadêncie de regis- tos. Outra desvantagem, se for necessário alterar o programa de transferência para adicionar

novas funcionalidades tem de se aceder aos sistemas locais onde estes estão a ser executados. Esta ação de alterar o programa no cliente pode não ser trivial se, por exemplo, o sistema local do cliente estiver situado num país diferente. Terminando com a questão das permissões das credenciais do programa de transferência na Tabela 3.4. Na versão centralizada, o sistema central não tem de garantir permissões aos endereços dos sistemas locais, ao contrário da versão descentralizada, que tem de garantir permissões aos endereços de todos os sistemas locais. Esta versão requer um maior controlo dos acessos que complica o processo de criação e definição das credenciais do programa de transferência. Por causa destas desvantagens escolheu-se seguir pela versão centralizada.

O método para reiniciar o programa remotamente pela AGS pode ser melhorado. Atualmente, o programa verifica com base num temporizador a BD regulação de procedimentos. Dependendo deste temporizador pode existir alguma latência entre o ato de alterar o parâmetro `a_transferencia` e o reiniciar propriamente dito do programa. Seria interessante desenvolver outro método que garanti-se que este reinicia-se sem latência.

A taxa de transferência de registos é definida pela capacidade de processamento do sistema utilizado. Durante o desenvolvimento do projeto testou-se a taxa de transferência do programa com os dados presentes na Tabela 6.1.

clientes

ID	Nome	Nº de Moldes	Nº de Sensores	Registros/Segundo
10001	Autocolismos & Company	4	16	48
10002	Máscaras de Carnaval LDA	6	30	90
10003	Aperture Science	2	2	12
10004	Black Mesa	1	10	30
10005	Vault-Tec Corporation	1	1	3

Tabela 6.1: Tabela com os dados utilizados para testar a taxa de transferência de registos, com uma média dos registos gerados por segundo para cada cliente

A quantidade de registos gerada em cada cliente varia consoante o número de moldes e sensores, por este motivo não faz sentido que seja usada a mesma taxa de transferência para todos os clientes. Assim sendo sugere-se para futuros desenvolvimentos criar um método de ajustar automaticamente a taxa de transferência de registos para a quantidade de registos produzidos, diminuindo o consumo de processador do sistema central.

6.1.3 Gestão de *Backups*

Este cumpre os objetivos definidos de gerar *backups*, organizar e limpar a BD central, bem como ser simples para permitir a sua portabilidade para outras linguagens de programação.

Quanto à componente de gerar *backups*, existem algumas limitações no método utilizado. Atualmente, o programa cria os ficheiros de *backup* executando diretamente no terminal o comando `mysqldump` que permite definir a pasta de destino para o ficheiro criado:

```
mysqldump -u backupmanager -pbackup1234 backups_temp >
~/Backups/backup.sql
```

Este comando também pode ser usado via *query* para o *MySQL*. No entanto, este não tem permissões para criar ficheiros fora da sua pasta principal. Esta pasta está protegida com permissões de administrador do sistema operativo, o que torna difícil mover os *backups* para

o repositório *online*. Na verdade é possível permitir o *MySQL* criar ficheiros fora da sua pasta principal, só que isto pode constituir uma falha de segurança dado que se podem enviar comandos para o terminal via *query*.

Como referido na Seção 3.4 os ficheiros criados via `mysqldump` funcionam como ponto de restauro e requerem o uso de uma BD intermédia para proceder à sua concatenação. No entanto, foi a solução que permitiu interagir com o repositório *online*.

Uma solução explorada inicialmente para criar *backups* consistia em ficheiros `.csv` que continham apenas registo com uma *query* do estilo:

```
SELECT *
FROM backups.registros
WHERE r_IDMolde = @IDMolde
INTO OUTFILE
'backup_@IDCliente_@IDMolde_@dataIni_@dataFim.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
```

Esta solução simplifica o processo de criação e concatenação dos *backups* dado que esta não necessita de uma BD intermédia mas, tem algumas desvantagens. A desvantagem principal, como foi referido anteriormente, não se pode definir o diretório de criação do ficheiro fora da pasta principal do *MySQL*. Além disto, para os dados puderem ser carregados devidamente, as informações dos clientes, moldes, sensores e registo têm de ser armazenadas separadamente, o que vai gerar uma maior diversidade de ficheiros de *backup*. Isto pode constituir um problema se o planeamento dos ficheiros não for bem efetuado.

Em futuros desenvolvimentos, se for definido que o diretório de criação dos *backups* não é imperativo, deve-se dar ênfase a esta solução dado que o processo de gerar e concatenar *backups* é mais simples e consequentemente mais rápido do que a solução usada neste projeto.

Quanto à componente de concatenar *backups*, esta solução pode não se ter demonstrado útil. Se um molde tiver muitos registo, concatenar os seus *backups* pode ultrapassar a capacidade máxima da base de dados, referida na Subseção 2.4.3. Além disto, se se desejar consultar o histórico de um molde arquivado, pode ser interessante manter os *backups* segmentados. Carregar os *backups* necessários para a consulta é mais rápido do que ter de carregar um *backup* total.

Eventualmente, chegou-se à conclusão que esta solução para arquivar moldes não era útil, ainda assim escolheu-se manter as ideias do processo neste relatório. Concatenar *backups* pode ser útil noutras circunstâncias, como por exemplo, realizar uma consulta que envolvam registo mais antigos que já tenham sido guardados. Dada a relativa complexidade de realizar a concatenação decidiu-se que as ideias desenvolvidas podem servir de base para trabalhos futuros.

6.1.4 Aplicação de gestão do sistema

A AGS cumpre os objetivos propostos de ser multiplataforma e garantir um acesso remoto à BD central, bem como gerir as informações dos clientes, moldes e sensores.

As mensagens de erro retornadas diretamente pelo *MySQL*, apesar de conterem bastante informação, não são simples de se perceber. Deve-se melhorar as mensagens de erro retornadas para que o utilizador as perceba mais facilmente.

De momento a AGS está preparada para ter mais que um cliente no sistema local. Quando se transitar para a situação real em que cada sistema só tem um cliente pode-se concatenar o processo de Adicionar Cliente, Instalar *MySQL*, Criar Local e Conectar Local num processo só, como demonstrado nas Figuras 6.2, 6.3 e 6.4.

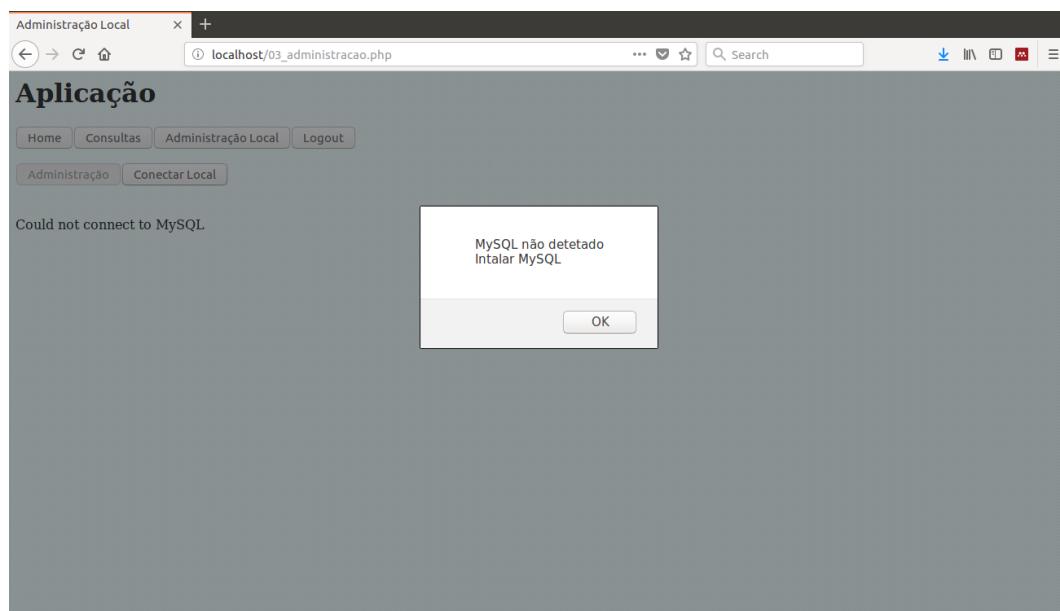


Figura 6.2: Exemplo de quando se clica em Conectar Local pela primeira vez e o sistema local não tem o *MySQL* instalado

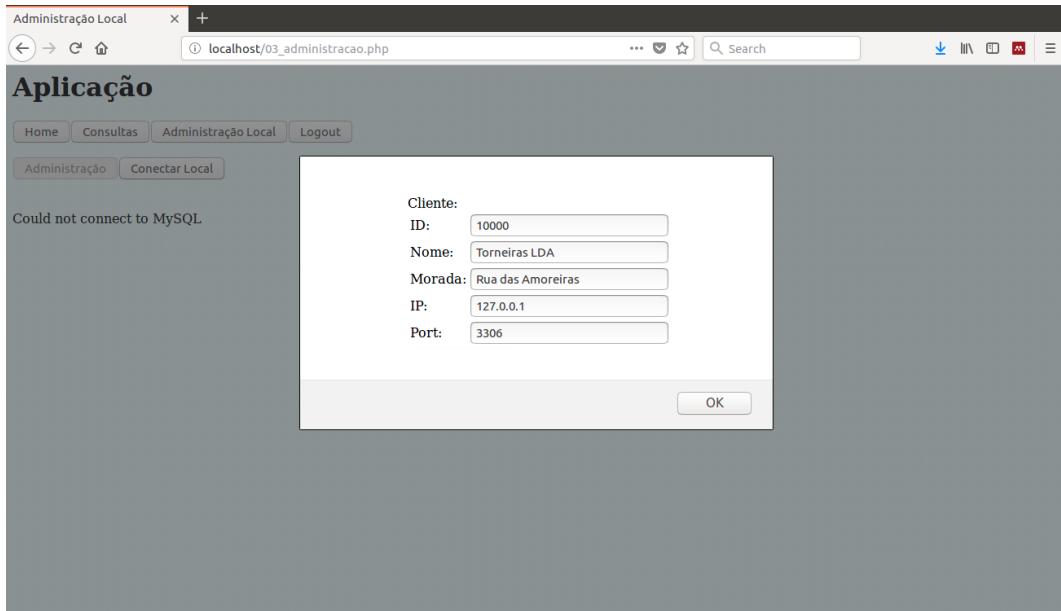


Figura 6.3: Exemplo de quando se clica em Conectar Local pela segunda vez para criar as BDs no sistema local

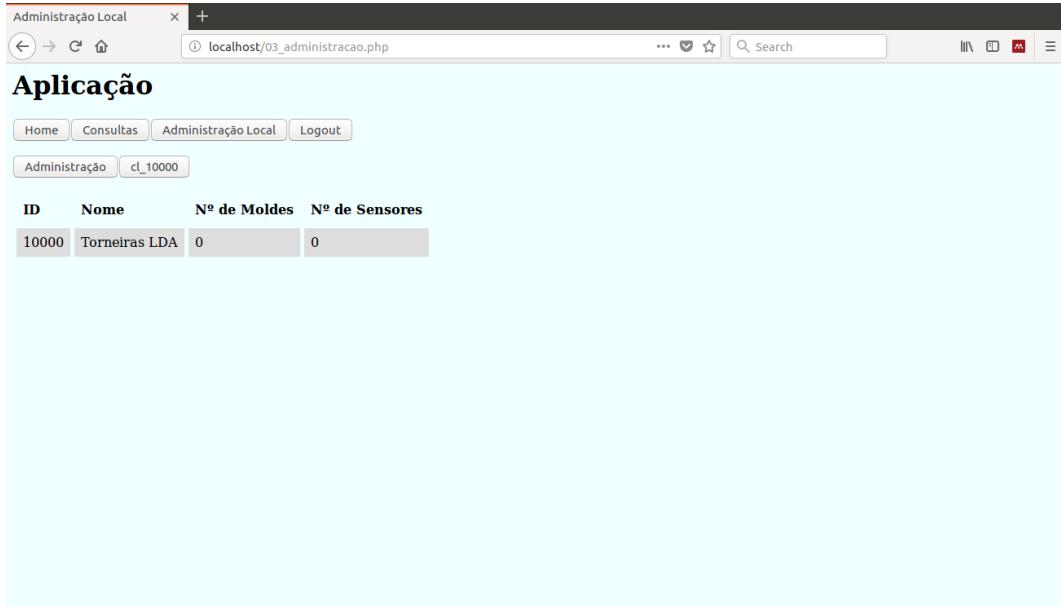


Figura 6.4: Exemplo de quando se clica em Conectar Local pela terceira vez em que a conexão é realizada diretamente, sem ser necessário escolher o cliente

A AGS só está preparada para adicionar informação à infraestrutura e instalar o sistema local. Sugere-se para futuros desenvolvimentos, preparar a aplicação para eliminar informação da infraestrutura. Na Figura 6.5 está um exemplo de um botão Eliminar Cliente que eliminaria

a informação relativa do cliente e desinstalaria automaticamente o sistema local.

ID	Nome	Morada	IP	Port	Nº de Moldes	Nº de Sensores
10000	Torneiras LDA	Rua das Amoreiras	127.0.0.1	3306	0	0

Figura 6.5: Exemplo da área Gestão de Clientes com a funcionalidade Eliminar Cliente, que irá remover toda a informação relativa do cliente e eliminar as BDs locais

Deve ser realizada uma revisão de segurança à AGS. Um programador com intenções maliciosas pode aceder ao sistema pela AGS e realizar comandos que podem comprometer o sistema.

Apesar de serem necessários alguns ajustes de forma a melhorar o desempenho, a solução proposta da infraestrutura de dados e AGS é completamente funcional e pode ser já implementada numa fase experimental.

6.2 Trabalhos Futuros

Quanto à infraestrutura dos dados não foi definido como os sensores dos moldes serão ligados ao servidor local. No desenvolvimento deste projeto assumiu-se que todos os moldes estão ligados diretamente ao sistema local. No entanto, isto pode não ser sempre possível de se realizar por motivos físicos. Na Figura 6.6 está representado um exemplo de um cliente que adquiriu quatro moldes que estão situados em pavilhões diferentes. Os moldes 1 e 2 podem ser facilmente ligados ao sistema local no entanto, dada a distância aos moldes 3 e 4 pode ser difícil liga-los ao sistema local.

Se se justificar, em vez de se ter um servidor local no cliente, ter um em cada molde. Esta adaptação irá criar uma maior quantidade de BDs locais mas isto não é problemático, se o modelo de dados e o programa de transferência forem adaptados para o efeito.

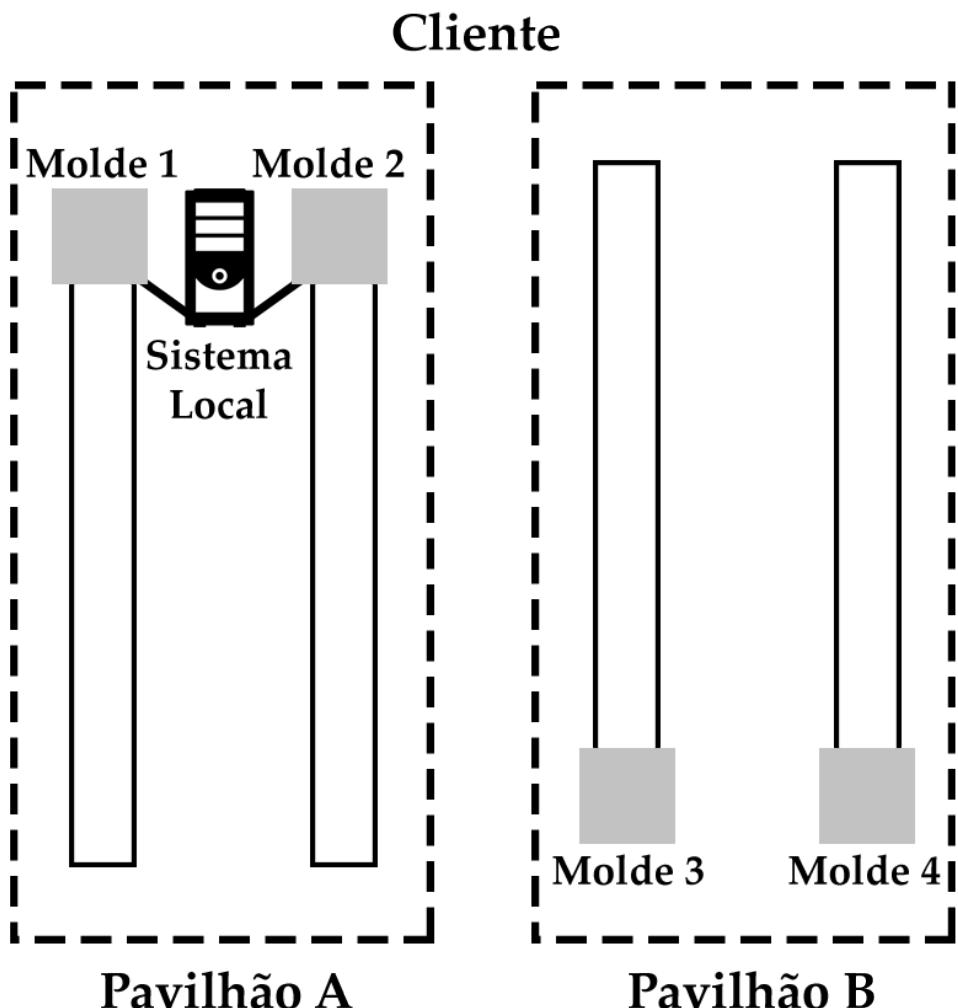


Figura 6.6: Exemplo de um cliente com moldes em pavilhões diferentes, que pode dificultar as ligações ao sistema local

Quanto à AGS, sugere-se que após uma apresentação inicial à empresa promotora, seja iniciado um processo iterativo de desenvolvimento para escolher e desenvolver novas funcionalidades que possam ser úteis e que não tenham sido abrangidas neste projeto, como por exemplo, a criação de utilizadores baseado no ID de trabalhador demonstrado na Figura 6.7, que pode depois servir para criar históricos das intervenções dos utilizadores nas BDs. Culminando numa estilização da AGS para que esta tenha um aspeto mais amigável ao utilizador.

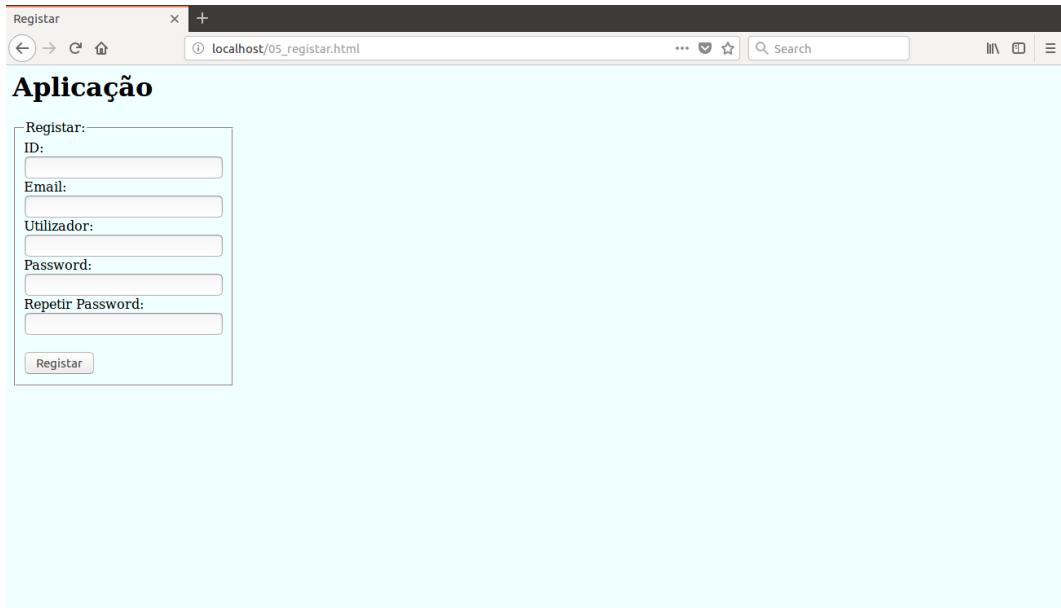


Figura 6.7: Exemplo de área de Registar para criar utilizadores com base no ID de trabalhador e email

Sugere-se a criação de um sistema de notificações e monitorização automática dos moldes. Durante o desenvolvimento do projeto chegou-se à conclusão de que é impossível para um utilizador analisar manualmente milhões de registo de um molde e concluir se este está a funcionar corretamente. Na Figura 6.8 está representado um exemplo de um possível comportamento de um molde. Cada ponto representa a média dos registo recolhidos em cada dinamómetro, durante um mês na fase de compactação. Estão representados 10 pontos simbolizando 10 meses de funcionamento de um molde. Observa-se um desvio do centróide do molde ao longo do tempo, até este sair fora da área de funcionamento.

Com uma representação visual parece fácil identificar o comportamento errático do molde no entanto, o utilizador teria de o identificar através de consultas e análise respostas do estilo da Figura 4.7a, todos os meses.

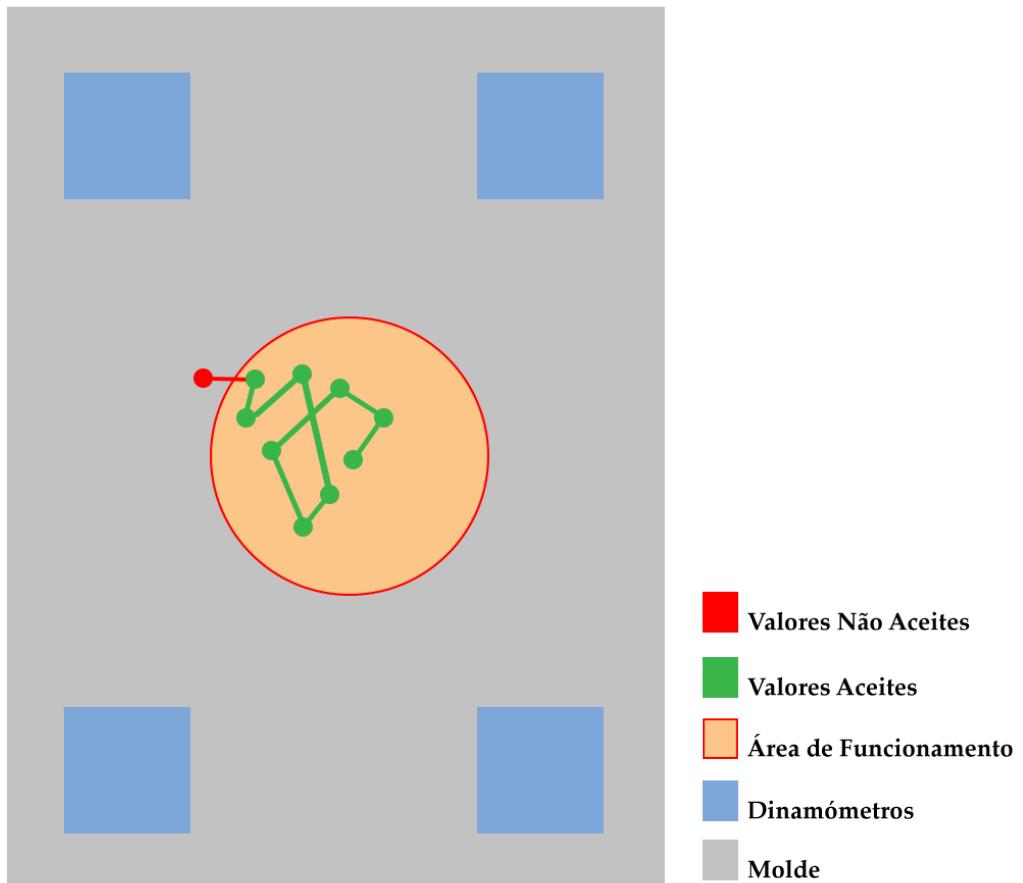


Figura 6.8: Exemplo molde 4 dinamômetros a sair da cena

Para auxiliar os utilizadores a identificar este comportamento, sugere-se criar um programa capaz de correr algoritmos que analisem constantemente o comportamento dos moldes. Este programa pode ser desenvolvido em *softwares* mais sofisticados, como por exemplo *MATLAB*, desde que estes tenham protocolos de comunicação com *MySQL*. Mantendo em mente o ambiente *Web*, sugere-se que este programa notifique os utilizadores via AGS ou via email quando um molde começa a demonstrar um comportamento errático.

Bibliografia

- [1] colored-part-large-mold-with-part.
- [2] moldagem in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2018. [consult. 2018-05-22 18:20:35]. Disponível na Internet: <https://www.infopedia.pt/dicionarios/lingua-portuguesa/moldagem>.
- [3] moldar in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2018. [consult. 2018-05-22 18:21:22]. Disponível na Internet: <https://www.infopedia.pt/dicionarios/lingua-portuguesa/moldar>.
- [4] molde in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2018. [consult. 2018-05-22 16:51:33]. Disponível na Internet: <https://www.infopedia.pt/dicionarios/lingua-portuguesa/molde>.
- [5] G. Williams and H. A. Lord. Mold-filling studies for the injection molding of thermoplastic materials. Part I: The flow of plastic materials in hot- and cold-walled circular channels. *Polymer Engineering and Science*, 15(8):553–568, aug 1975.
- [6] Michael Trovant and Stavros A Argyropoulos. The implementation of a mathematical model to characterize mold metal interface effects in metal casting. *Canadian Metallurgical Quarterly*, 37(3-4):185–196, jun 1998.
- [7] TN) Janney, Mark A. (Knoxville and NG) Omatete, Ogbemi O. (Lagos. Method for molding ceramic powders using a water-based gel casting, jul 1991.
- [8] Jiwang Yan, Takashi Oowada, Tianfeng Zhou, and Tsunemoto Kuriyagawa. Precision machining of microstructures on electroless-plated NiP surface for molding glass components. *Journal of Materials Processing Technology*, 209(10):4802–4808, jun 2009.
- [9] Changyu Shen, Lixia Wang, and Qian Li. Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method. *Journal of Materials Processing Technology*, 183(2-3):412–418, mar 2007.
- [10] K. Shelesh-Nezhad and E. Siores. An intelligent system for plastic injection molding process design. *Journal of Materials Processing Technology*, 63(1-3):458–462, jan 1997.
- [11] Babur Ozcelik and Ibrahim Sonat. Warpage and structural analysis of thin shell plastic in the plastic injection molding. *Materials & Design*, 30(2):367–375, feb 2009.
- [12] About Injection Molding | Xcentric Mold & Engineering.

- [13] *Polymer Process Engineering - R. Griskey - Google Livros.*
- [14] hyatt. nov 1872.
- [15] Injection molding machine. may 1949.
- [16] Suzanne L. B. Woll and Douglas J. Cooper. Pattern-based closed-loop quality control for the injection molding process. *Polymer Engineering & Science*, 37(5):801–812, may 1997.
- [17] D.F. Cook, C.T. Ragsdale, and R.L. Major. Combining a neural network with a genetic algorithm for process parameter optimization. *Engineering Applications of Artificial Intelligence*, 13(4):391–396, aug 2000.
- [18] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, USA, 6th edition, 2010.
- [19] base de dados in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2018. [consult. 2018-05-31 18:24:13]. Disponível na Internet: <https://www.infopedia.pt/dicionarios/lingua-portuguesa/base-de-dados>.
- [20] Edgar Frank Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, 1970.
- [21] Peter Pin-Shan Chen. The Entity-Relationship Unified View of Data Model-Toward a. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [22] ubuntu logo.
- [23] GNU Compiler Collection logo.
- [24] apache-logo.
- [25] mysql-logo.
- [26] Community | Ubuntu.
- [27] MySQL :: MySQL 8.0 Reference Manual.
- [28] GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF).
- [29] Welcome! - The Apache HTTP Server Project.
- [30] Web Development PHP HTML Arrows Royalty Free Vector Image.
- [31] W3Schools Online Web Tutorials.
- [32] SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean.

Apêndice A

Queries para instalar a solução desenvolvida

A informação contida no Capítulo 3 permite a realização das *queries* presentes neste apêndice.

A.1 Sistema Central

Para criar as BDs central, *backups* e *backups* temporária:

```
CREATE TABLE tipo
(
    tipo_ID INT NOT NULL,
    tipo_nome VARCHAR(50) NOT NULL,
    CONSTRAINT REPETIDO_ID_TIPO
        PRIMARY KEY(tipo_ID),
    CONSTRAINT REPETIDO_NOME_TIPO
        UNIQUE(tipo_nome)
);
CREATE TABLE fase
(
    fase_ID INT NOT NULL,
    fase_nome VARCHAR(50) NOT NULL,
    CONSTRAINT REPETIDO_ID_FASE
        PRIMARY KEY(FASE_ID),
    CONSTRAINT REPETIDO_NOME_FASE
        UNIQUE(fase_nome)
);
CREATE TABLE clientes
(
    cl_ID INT NOT NULL,
    cl_nome VARCHAR(50) NOT NULL,
    cl_morada VARCHAR(100),
    cl_IP VARCHAR(50) NOT NULL,
    cl_port INT NOT NULL,
```

```

CONSTRAINT REPETIDO_ID_CLIENTE
    PRIMARY KEY(cl_ID)
);
CREATE TABLE moldes
(
    m_IDCliente INT NOT NULL,
    m_ID INT NOT NULL,
    m_nome VARCHAR(30),
    m_descricao VARCHAR(200),
    CONSTRAINT REPETIDO_ID_MOLDE
        PRIMARY KEY(m_ID),
    CONSTRAINT MOLDE_MAU_ID_CLIENTE
        FOREIGN KEY(m_IDCliente)
            REFERENCES clientes(cl_ID)
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE TABLE sensores
(
    s_IDMolde INT NOT NULL,
    s_num INT NOT NULL,
    s_tipo INT NOT NULL,
    s_nome varchar(30),
    s_descricao varchar(200),
    CONSTRAINT REPETIDO_NUM_SENSOR
        PRIMARY KEY(s_IDMolde,s_num),
    CONSTRAINT SENSOR_MAU_ID_MOLDE
        FOREIGN KEY(s_IDMolde)
            REFERENCES moldes(m_ID)
                ON DELETE CASCADE ON UPDATE NO ACTION,
    CONSTRAINT SENSOR_MAU_ID_TIPO
        FOREIGN KEY(s_tipo)
            REFERENCES tipo(tipo_ID)
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE TABLE registros
(
    r_IDMolde INT NOT NULL,
    r_numSensor INT NOT NULL,
    r_fase INT NOT NULL,
    r_data_hora DATETIME NOT NULL,
    r_milissegundos TINYINT NOT NULL,
    r_valor FLOAT,
    CONSTRAINT REPETIDO_REGISTO
        PRIMARY KEY(r_IDMolde, r_numSensor,
                    r_data_hora, r_milissegundos),
    CONSTRAINT REGISTOS_MAU_ID_MOLDE_SENSOR
        FOREIGN KEY(r_IDMolde,r_numSensor)

```

```

        REFERENCES sensores(s_IDMolde,s_num)
                ON DELETE CASCADE ON UPDATE NO ACTION,
CONSTRAINT REGISTOS_MAU_ID_FADE
        FOREIGN KEY(r_fase)
        REFERENCES fase(fase_ID)
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
INSERT INTO tipo
VALUES
(1, "Term\'ometro"),
(2, "Dinam\'ometro"),
(3, "Extens\'metro"),
(4, "Vibr\'metro"),
(5, "Press\~ao"),
(6, "Aceler\'metro_X"),
(7, "Aceler\'metro_Y"),
(8, "Aceler\'metro_Z");
INSERT INTO fase
VALUES
(1, "Fecho"),
(2, "Enchimento"),
(3, "Compacta\231\~ao"),
(4, "Abertura"),
(5, "Extra\231\~ao");

```

A BD regula\~ao de procedimentos

```

CREATE TABLE atualizar
(
    a_indice INT NOT NULL,
    a_transferencia INT NOT NULL,
    a_backups INT NOT NULL,
CONSTRAINT REPETIDO_INDICE_ATUALIZAR
    PRIMARY KEY(a_indice),
CONSTRAINT ATUALIZAR_MAU_INDICE
    CHECK a_indice = 1,
CONSTRAINT ATUALIZAR_MAU_TRANSFERENCIA
    CHECK a_transferencia = 0 OR a_transferencia = 1,
CONSTRAINT ATUALIZAR_MAU_BACKUPS
    CHECK a_backups = 0 OR a_backups = 1
);
CREATE TABLE atualizar
(
    a_indice INT NOT NULL,
    a_transferencia INT NOT NULL,
    a_backups INT NOT NULL,
CONSTRAINT REPETIDO_INDICE_ATUALIZAR
    PRIMARY KEY(a_indice),

```

```

CONSTRAINT ATUALIZAR_MAU_INDICE
    CHECK a_indice = 1,
CONSTRAINT ATUALIZAR_MAU_TRANSFERENCIA
    CHECK a_transferencia = 0 OR a_transferencia = 1,
CONSTRAINT ATUALIZAR_MAU_BACKUPS
    CHECK a_backups = 0 OR a_backups = 1
);
CREATE TABLE backups
(
    b_IDMolde INT NOT NULL,
    CONSTRAINT REPETIDO_ID_MOLDE
        PRIMARY KEY(a_indice)
);
INSERT INTO atualizar
VALUES
    (1, 0, 0);

```

E definem-se as credenciais com as respetivas permissões. Como o servidor *Apache* está a correr no sistema central, o endereço do servidor é igual ao endereço do sistema central:

```

CREATE USER 'user'@'localhost'
    IDENTIFIED BY 'password';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON central.tipo TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON central.fase TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON central.clientes TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON central.moldes TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON central.sensores TO 'user'@'localhost';
GRANT SELECT, DELETE
    ON central.registros TO 'user'@'localhost';
GRANT SELECT, UPDATE
    ON reg_proc.atualizar TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON reg_proc.backups TO 'user'@'localhost';
CREATE USER 'transferencia'@'localhost'
    IDENTIFIED BY 'transferencia1234';
GRANT INSERT
    ON central.registros TO 'transferencia'@'localhost';
GRANT SELECT, UPDATE
    ON reg_proc.atualizar TO 'transferencia'@'localhost';
CREATE USER 'backupmanager'@'localhost'
    IDENTIFIED BY 'backup1234';
GRANT SELECT
    ON central.clientes TO 'backupmanager'@'localhost';

```

```

GRANT SELECT
    ON central.moldes TO 'backupmanager'@'localhost';
GRANT SELECT
    ON central.sensores TO 'backupmanager'@'localhost';
GRANT SELECT, DELETE
    ON central.registros TO 'backupmanager'@'localhost';
GRANT SELECT, INSERT, DELETE
    ON backups.* TO 'backupmanager'@'localhost';
GRANT SELECT, INSERT, DELETE
    ON backups_temp.* TO 'backupmanager'@'localhost';
GRANT SELECT, UPDATE
    ON reg_proc.atualizar TO 'backupmanager'@'localhost';
GRANT SELECT, DELETE
    ON reg_proc.backups TO 'backupmanager'@'localhost';
FLUSH PRIVILEGES;

```

A.2 Sistema Local

Para criar a BD local utiliza-se as mesmas *queries* para criar a BD central. A BD temporária local assume a forma:

```

CREATE TABLE tipo
(
    tipo_ID INT NOT NULL,
    tipo_nome VARCHAR(50) NOT NULL,
    CONSTRAINT REPETIDO_ID_TIPO
        PRIMARY KEY(tipo_ID),
    CONSTRAINT REPETIDO_NOME_TIPO
        UNIQUE(tipo_nome)
);
CREATE TABLE clientes
(
    cl_ID INT NOT NULL,
    cl_nome VARCHAR(50) NOT NULL,
    cl_morada VARCHAR(100),
    cl_IP VARCHAR(50) NOT NULL,
    cl_port INT NOT NULL,
    CONSTRAINT REPETIDO_ID_CLIENTE
        PRIMARY KEY(cl_ID)
);
CREATE TABLE moldes
(
    m_IDCliente INT NOT NULL,
    m_ID INT NOT NULL,
    m_nome VARCHAR(30),
    m_descricao VARCHAR(200),
    CONSTRAINT REPETIDO_ID_MOLDE

```

```

PRIMARY KEY(m_ID),
CONSTRAINT MOLDE_MAU_ID_CLIENTE
    FOREIGN KEY(m_IDCliente) REFERENCES clientes(cl_ID)
        ON DELETE NO ACTION ON UPDATE CASCADE
);
CREATE TABLE sensores
(
    s_IDMolde INT NOT NULL,
    s_num INT NOT NULL,
    s_tipo INT NOT NULL,
    s_nome varchar(30),
    s_descricao varchar(200),
    CONSTRAINT REPETIDO_NUM_SENSOR
        PRIMARY KEY(s_IDMolde,s_num),
    CONSTRAINT SENSOR_MAU_ID_MOLDE
        FOREIGN KEY(s_IDMolde) REFERENCES moldes(m_ID)
            ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT SENSOR_MAU_ID_TIPO
        FOREIGN KEY(s_tipo) REFERENCES tipo(tipo_ID)
            ON DELETE NO ACTION ON UPDATE CASCADE
);
INSERT INTO tipo
VALUES
(1, "Term\'ometro"),
(2, "Dinam\'ometro"),
(3, "Extens\'metro"),
(4, "Vibr\'metro"),
(5, "Press\~ao"),
(6, "Aceler\'metro_X"),
(7, "Aceler\'metro_Y"),
(8, "Aceler\'metro_Z");

```

Definem-se as credenciais com as respetivas permissões. Como o servidor *Apache* está a correr no sistema central, o endereço do servidor é igual ao endereço do sistema central:

```

CREATE USER 'user'@'localhost'
    IDENTIFIED BY 'password';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.tipo TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.fase TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.clientes TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.moldes TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.sensores TO 'user'@'localhost';
GRANT SELECT, INSERT, DELETE, UPDATE

```

```
    ON local.tipo TO 'user'@'centralhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.fase TO 'user'@'centralhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.clientes TO 'user'@'centralhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.moldes TO 'user'@'centralhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON local.sensores TO 'user'@'centralhost';
GRANT SELECT, INSERT, DELETE, UPDATE
    ON temp_local.* TO 'user'@'centralhost';
CREATE USER 'transferencia'@'centralhost'
    IDENTIFIED BY 'transferencia1234';
GRANT SELECT, DELETE
    ON local.registros TO 'transferencia'@'centralhost';
CREATE USER 'sensores'@'localhost'
    IDENTIFIED BY 'sensores1234';
GRANT INSERT
    ON local.registros TO 'sensores'@'localhost';
FLUSH PRIVILEGES;
```


Apêndice B

Evolução dos Diagramas Entidade/Relação

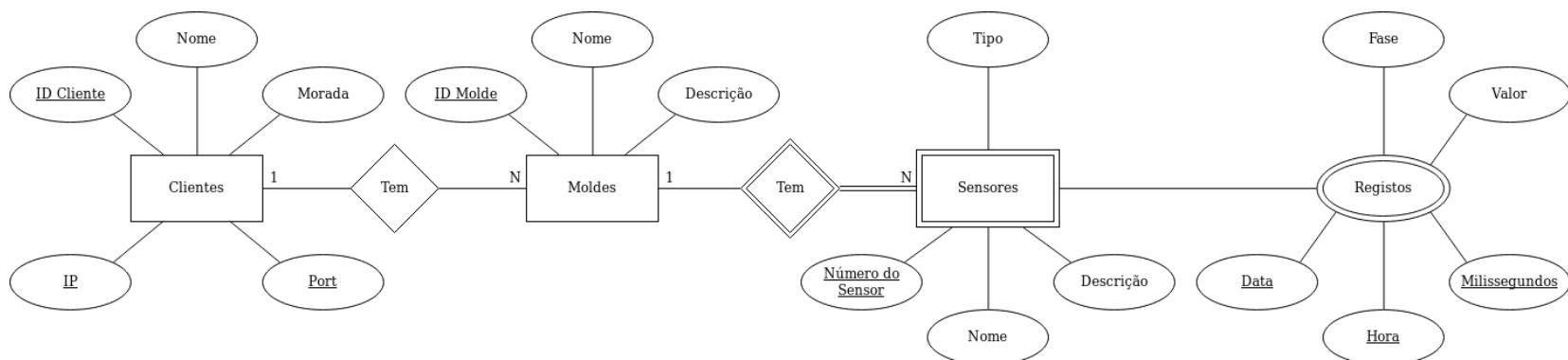


Figura B.1: Primeira iteração do DER, esta versão considerava os registos como um atributo multi-valor e composto da entidade sensores. Apesar das tabelas serem as mesmas que a versão presente na Figura B.2, não se consegue perceber a relação entre sensores e registos

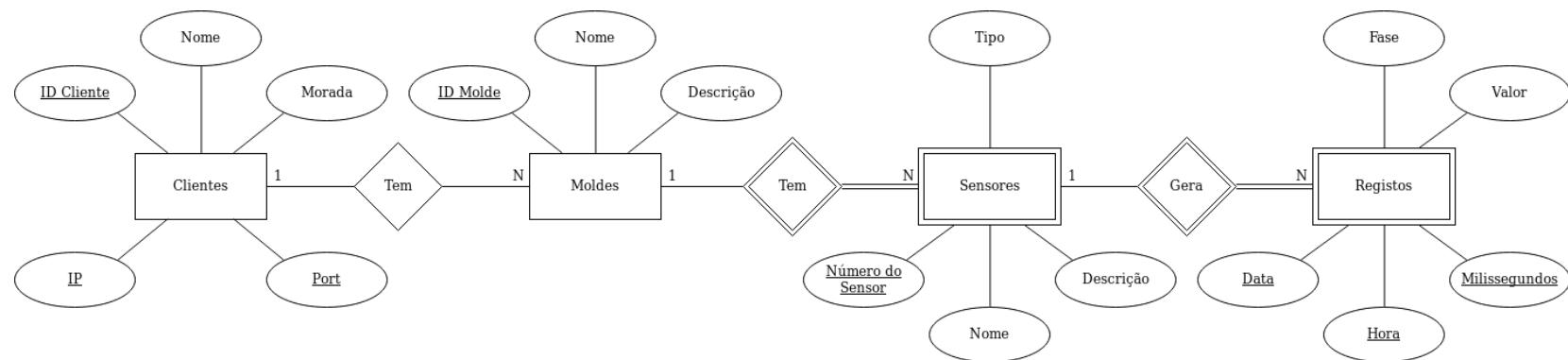


Figura B.2: Segunda iteração do DER, simplificação do DER representado na Figura B.1 onde se traduz o atributo multi-valor composto registos numa entidade, permitindo especificar a chave primária, relação com sensores, cardinalidade e obrigatoriedade

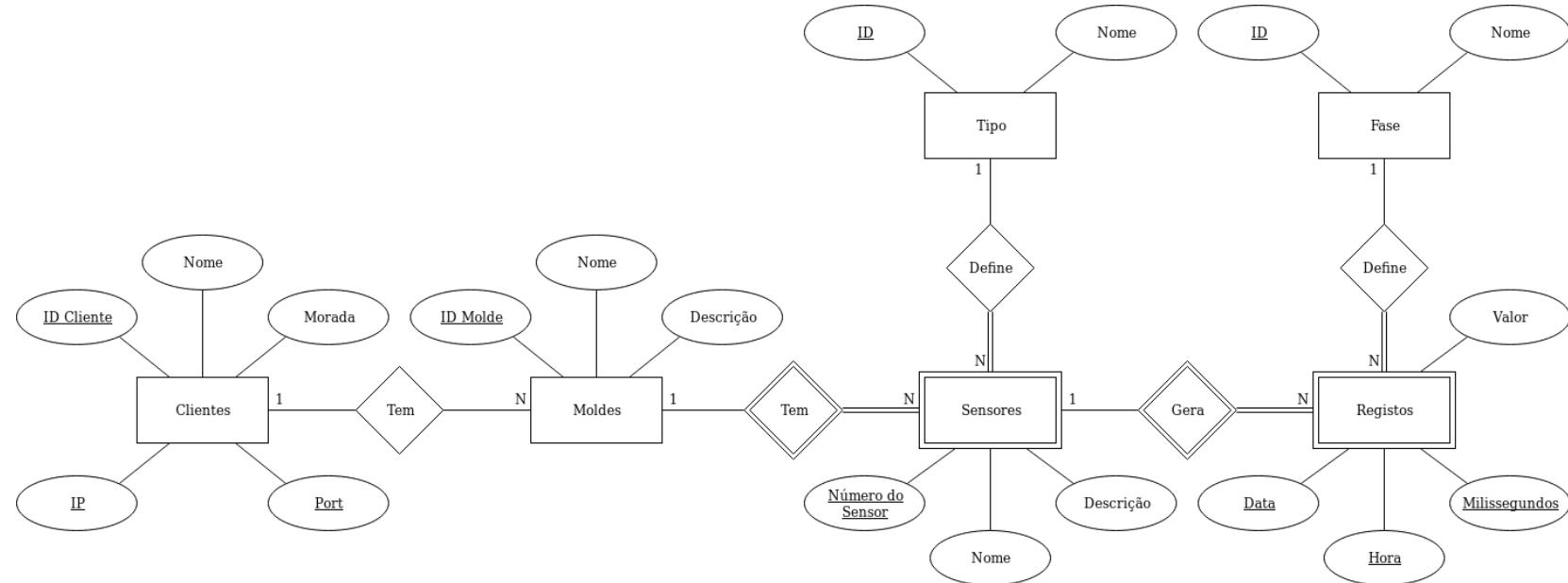


Figura B.3: Terceira iteração do DER, com a adição dos dicionários pelos motivos referidos na Subseção 3.2.1

Apêndice C

Infraestrutura com todas as bases de dados

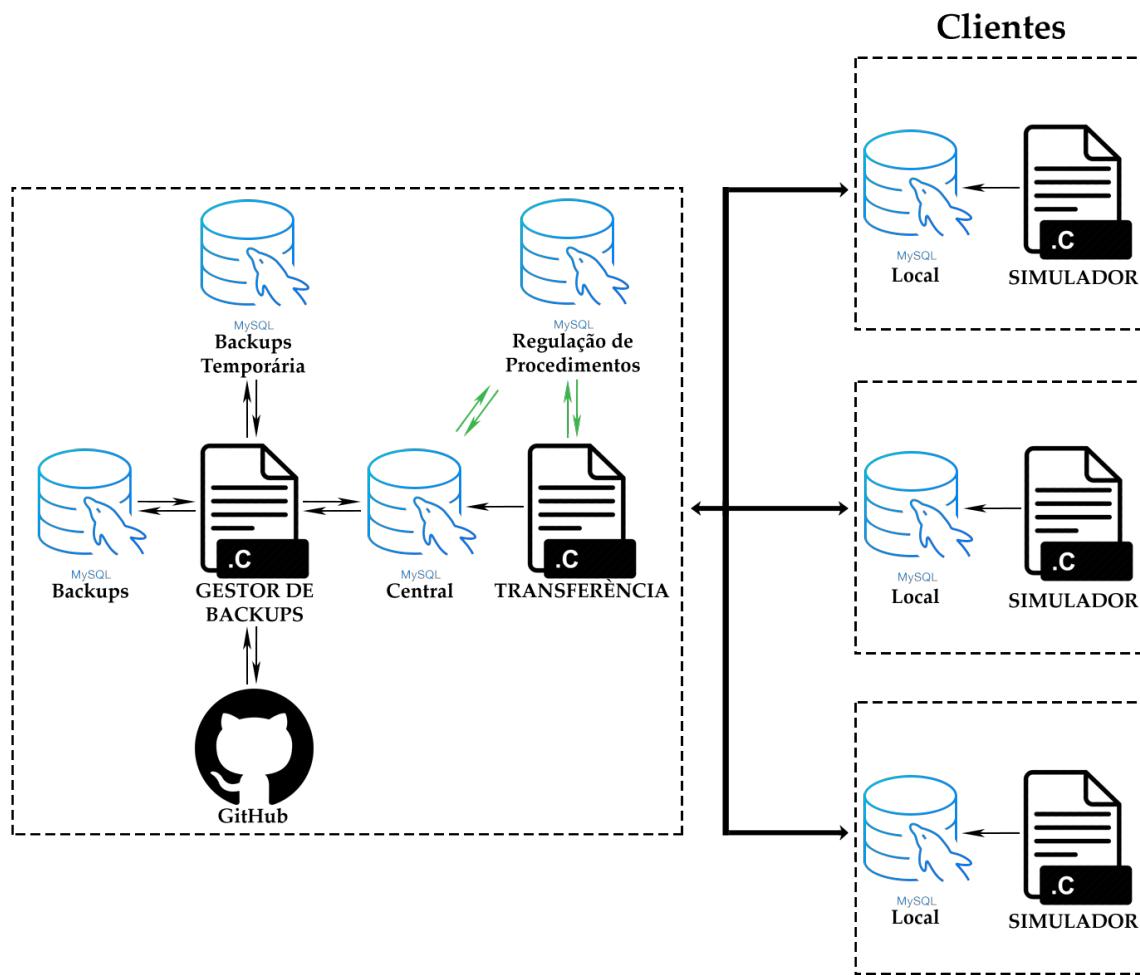


Figura C.1: Diagrama da infraestrutura completa, numa utilização de baixo nível, sem AGS. A preto está representado o fluxo de registos e a verde o fluxo de regulação de procedimentos

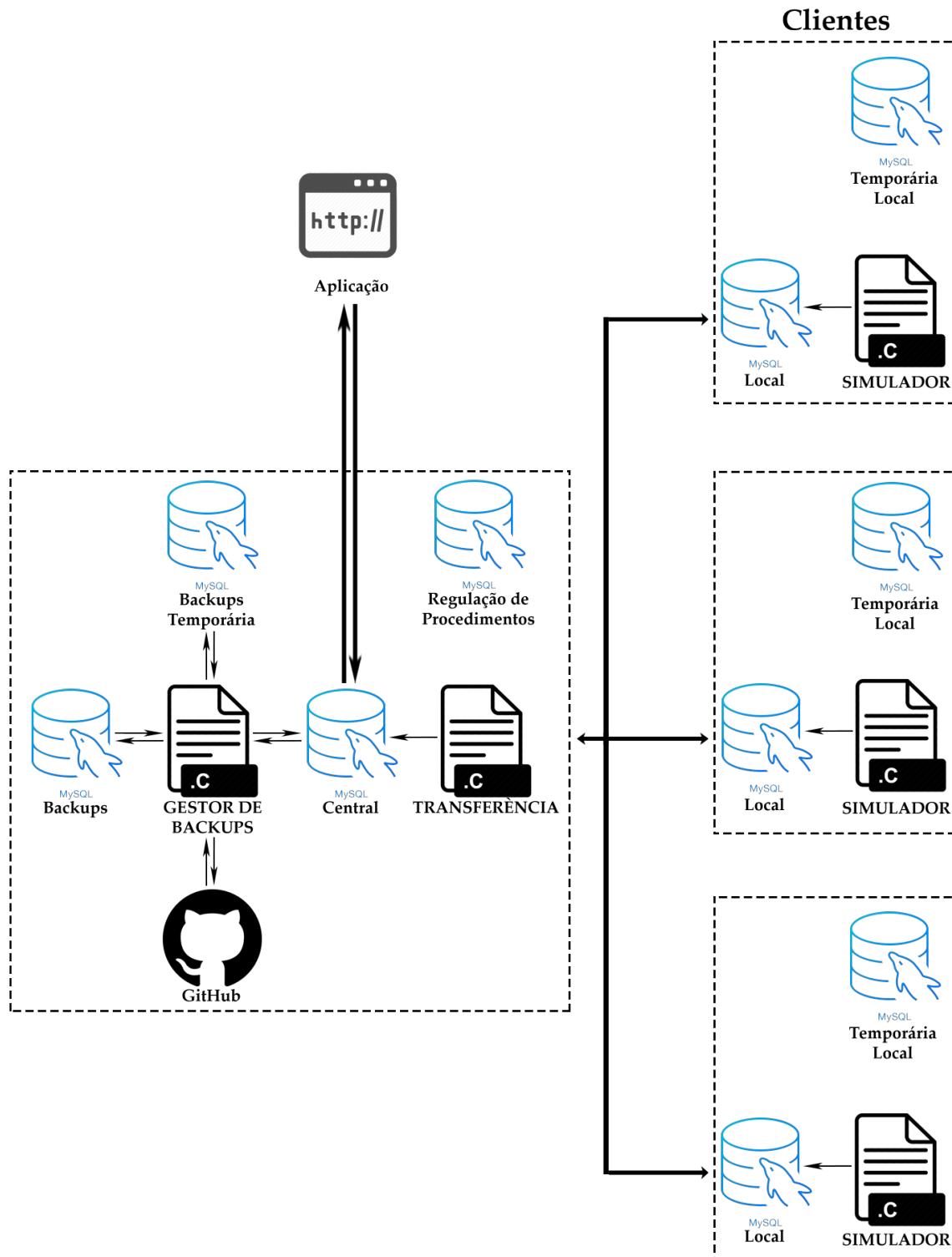


Figura C.2: Diagrama da infraestrutura completa com uma utilização geral da AGS. A preto está representado o fluxo de registos. Apesar de não estar representado, os programas de transferência e gestão de *backups* continuam a consultar a BD regulação de procedimentos. Escolheu-se esta representação para salientar que a AGS não interage com estes programas, dado que as funcionalidades Atualizar e Arquivar Molde só estão disponíveis numa utilização local

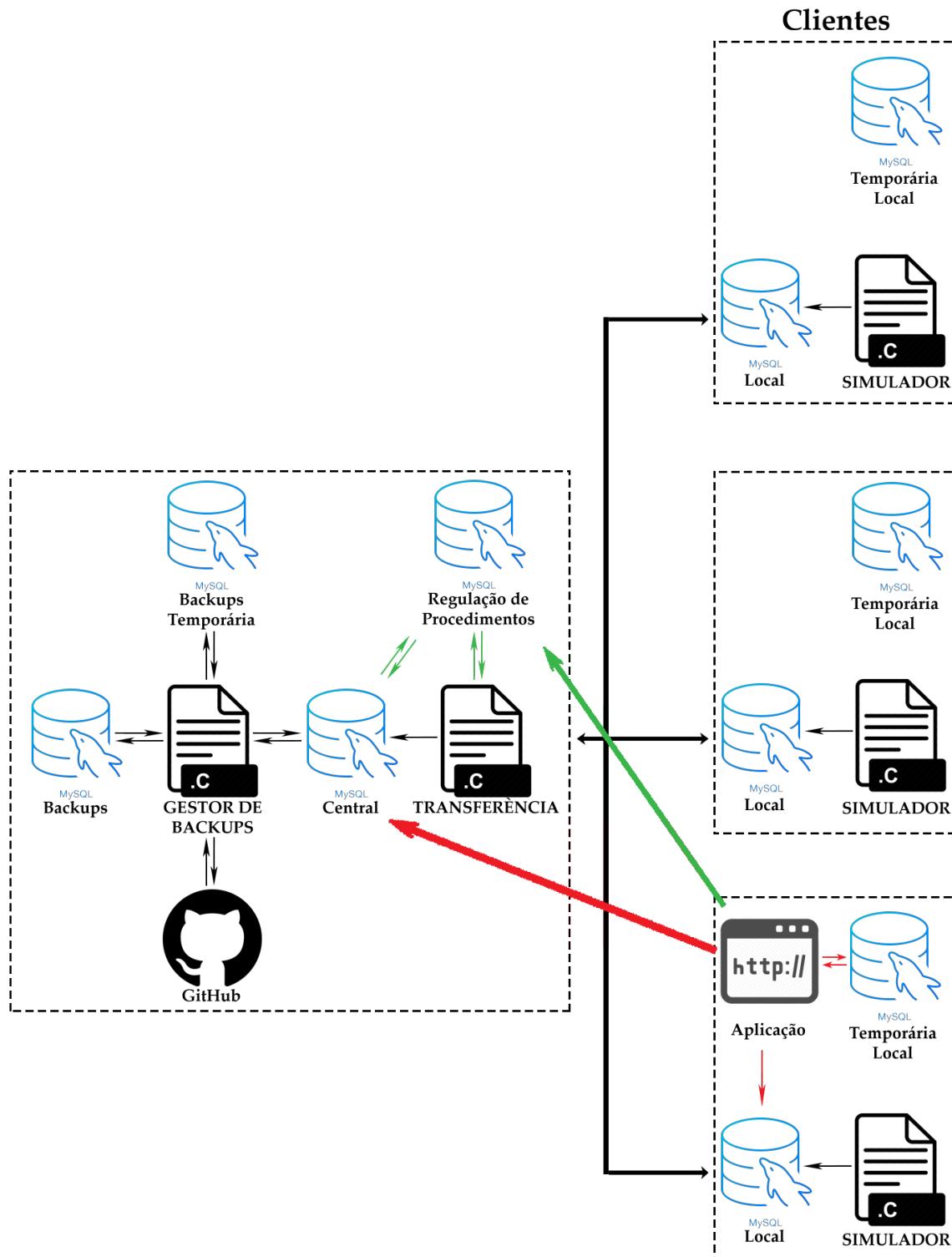


Figura C.3: Diagrama da infraestrutura completa com uma utilização local da AGS. A preto está representado o fluxo de registos, a verde o fluxo de regulação de procedimentos e a vermelho o fluxo da informação dos clientes, moldes e sensores. Apesar de não estar representado, a AGS continua a puder consultar informação diretamente à BD central como na Figura C.2. Escolheu-se não representar para evitar uma saturação da imagem

Apêndice D

Dados iniciais dos testes de usabilidade

tipo	
tipo_ID	tipo_nome
1	Termómetro
2	Dinamômetro
3	Extensómetro
4	Vibrômetro
5	Pressão
6	Acelerômetro X
7	Acelerômetro Y
8	Acelerômetro Z

Tabela D.1: Tabela com os valores iniciais da tabela tipo

fase	
fase_ID	fase_nome
1	Fecho
2	Enchimento
3	Compactação
4	Abertura
5	Extração

Tabela D.2: Tabela com os valores iniciais da tabela fase

clientes

cl_ID	cl_nome	cl_morada	cl_IP	cl_port
10001	Autocolismos & Company	Rua das sanitas, nº 2	127.0.0.1	3306
10002	Máscaras de Carnaval LDA	Rua de Elm street, nº 13	127.0.0.1	3306
10004	Black Mesa	Área 51	127.0.0.1	3306
10005	Vault-Tec Corporation	Rua da caixa-forte, nº 1234	127.0.0.1	3306

Tabela D.3: Tabela com os valores iniciais da tabela clientes

moldes

m_IDCliente	m_ID	m_nome	m_descricao
10001	9000	NULL	NULL
10001	9001	NULL	NULL
10001	9002	NULL	NULL
10001	9003	NULL	NULL
10002	9004	NULL	NULL
10002	9005	NULL	NULL
10002	9006	NULL	NULL
10004	9009	NULL	NULL
10002	9010	NULL	NULL
10002	9011	NULL	NULL
10002	9012	NULL	NULL
10005	9013	NULL	NULL

Tabela D.4: Tabela com os valores iniciais da tabela moldes

sensores

s_IDMolde	s_num	s_tipo	s_nome	s_descricao
9000	1	1	NULL	NULL
9000	2	2	NULL	NULL
9000	3	2	NULL	NULL
9000	4	2	NULL	NULL
9000	5	2	NULL	NULL
9001	1	1	NULL	NULL
9001	2	2	NULL	NULL
9001	3	2	NULL	NULL
9001	4	2	NULL	NULL
9001	5	2	NULL	NULL
9002	1	5	NULL	NULL
9002	2	3	NULL	NULL
9002	3	3	NULL	NULL
9003	1	5	NULL	NULL
9003	2	3	NULL	NULL
9003	3	3	NULL	NULL
9004	1	1	NULL	NULL
9004	2	2	NULL	NULL
9004	3	3	NULL	NULL
9004	4	4	NULL	NULL
9004	5	5	NULL	NULL

Tabela D.5: Parte 1 de 2 da tabela com os valores iniciais da tabela sensores

sensores

s_IDMolde	s_num	s_tipo	s_nome	s_descricao
9005	1	1	NULL	NULL
9005	2	2	NULL	NULL
9005	3	3	NULL	NULL
9005	4	4	NULL	NULL
9005	5	5	NULL	NULL
9006	1	1	NULL	NULL
9006	2	2	NULL	NULL
9006	3	3	NULL	NULL
9006	4	4	NULL	NULL
9006	5	5	NULL	NULL
9009	1	1	NULL	NULL
9009	2	1	NULL	NULL
9009	3	2	NULL	NULL
9009	4	2	NULL	NULL
9009	5	3	NULL	NULL
9009	6	3	NULL	NULL
9009	7	4	NULL	NULL
9009	8	4	NULL	NULL
9009	9	5	NULL	NULL
9009	10	5	NULL	
9010	1	1	NULL	NULL
9010	2	2	NULL	NULL
9010	3	3	NULL	NULL
9010	4	4	NULL	NULL
9010	5	5	NULL	NULL
9011	1	1	NULL	NULL
9011	2	2	NULL	NULL
9011	3	3	NULL	NULL
9011	4	4	NULL	NULL
9011	5	5	NULL	NULL
9012	1	1	NULL	NULL
9012	2	2	NULL	NULL
9012	3	3	NULL	NULL
9012	4	4	NULL	NULL
9012	5	5	NULL	NULL
9013	1	3	NULL	NULL

Tabela D.6: Parte 2 de 2 da tabela com os valores iniciais da tabela sensores