



**Desenvolvimento da gestão de manutenção da empresa ACATEL, Acabamentos Têxteis S.A.**

José Carlos Rodrigues Arantes,

n.º 8636

Realizado sob orientação do Professor Michael Matias

Ano letivo 2019/2020

Licenciatura de Engenharia de Sistemas Informáticos.

Escola Superior de Tecnologia do Instituto Politécnico do Cávado e Ave.

**Identificação do aluno**

José Carlos Rodrigues Arantes

Número 8636, regime diurno

Ocupação: Estudante e trabalhador.

Licenciatura em Engenharia de Sistemas Informáticos.

**Orientação académica**

Professor Michael Matias

**Informação sobre a empresa de acolhimento**

ACATEL - Acabamentos Têxteis S.A.

Rua do Barreiro, 310 - Gilmonde

4754-909 Barcelos

Orientação: Engenheiro José Carlos Catalo

“*Os analfabetos do próximo século não são aqueles que não sabem ler ou escrever, mas aqueles que se recusam a aprender, reaprender e voltar a aprender*.” Alvin Tofler (1928-2016)

### RESUMO

Toda a nossa vida está em constante mudança e também evolução. Sem excepção disso está o mundo das engenharias e tecnologias, com novas ferramentas em constante mudança.

Reflete-se também esta evolução em empresas de diversos sectores. Parte desta evolução traz consigo a informatização de sistemas de sistemas obsoletos e que não cumprem normas ou regras de aceitação por parte das associações acreditadoras, sendo estas normas necessárias para manter os clientes, que neste caso é essencial à empresa ACATEL, a organização de acolhimento.

A área da programação WEB, é apontada como prática fundamental para a preparação de profissionais para o mercado de trabalho.

Nesse contexto, este projeto incide na criação de uma página WEB, disponível no dominio de rede da empresa, em tempo real, utilizada para facilitar a consulta, mas também a alertar a necessidade de reparações dos equipamentos (máquinas de laboração ou de ajuda à laboração).

Os objetivos estruturantes do presente relatório residem numa fase introdutória, onde é exposta a situação atual e obsoleta, que condiciona a aprovação de acreditações de futuras vistorias na empresa. De seguida, o objetivo detalha o processo de desenvolvilmento ao ambiente apresentado e a sua análise de requisitos.

(Nota: ver palavras chave-tema, e palavras chave-tecnologia)

### ABSTRACT

### AGRADECIMENTOS

A realização deste projeto final do curso de licenciatura, que tem como objetivo a obtenção do grau de Engenheiro de Sistemas Informáticos, contou como seria de esperar de grande apoio e suporte, sem os quais tornaria esta tarefa ainda mais difícil.

Em primeiro lugar, ao Professor Michael Matias por ter aceite o convite para me orientar nesta jornada, pelo seu conhecimento transmitido, por todas as horas que estivemos em video-conferência (em tempos de pandémicos), de modo a conseguir resolver alguns problemas que estavam a surgir. De um modo especial, à sua incasável disponibilidade de modo a que nunca me deixasse ficar mal. Ao Professor Michael, o meu grande apreço.

Ao amigo e colega de trabalho, José Carreira, trabalhador na empresa de acolhimento, pela ajuda no sentido de me fazer perceber como funcionam as regras da empresa, durante a laboração deste projeto.

Ao Engenheiro José Carlos Catalo, pessoa que me acolheu na empresa durante a realização deste projeto, e e ajuda ao mostrar o esquema de funcionamento.

Por fim, aos alicerces da minha vida, os meus pais, pelo grande apoio que me deram, por todo o sacríficio que também eles passaram para que eu possa ter concluído todo este percurso desde o início da licenciatura, até à fase final.

A todos, saúde e muito obrigado.

**Indíce**

[RESUMO 5](#_Toc59399746)

[ABSTRACT 6](#_Toc59399747)

[AGRADECIMENTOS 7](#_Toc59399748)

[ÍNDICE DE FIGURAS 10](#_Toc59399749)

[ÍNDICE DE TABELAS 11](#_Toc59399750)

[Glossário 12](#_Toc59399751)

[Siglas e Acrónimos 13](#_Toc59399752)

[**1.** **Introdução** 14](#_Toc59399753)

[1.1. Contexto e objetivos 14](#_Toc59399754)

[1.2. Estado da arte 15](#_Toc59399755)

[1.3. Estrutura do documento 16](#_Toc59399756)

[**2.** **Desenvolvimento** 17](#_Toc59399757)

[2.1 Análise ao problema 18](#_Toc59399758)

1. [2.1.1 Requisitos funcionais 19](#_Toc59399759)

[2.1.2 Requisitos não funcionais 21](#_Toc59399760)

[2.1.3 Diagrama de casos de uso 22](#_Toc59399761)

[2.2 Plataformas de desenvolvimento 24](#_Toc59399762)

[2.3 Introdução à base de dados 25](#_Toc59399763)

[2.4 Introdução à API (front-end) 28](#_Toc59399764)

[2.4.1 Controladores 28](#_Toc59399765)

### ÍNDICE DE FIGURAS

[Figura 1 - Diagrama de blocos geral 15](file:///C:\Users\José%20Carlos\Documents\GitHub\ProjetoGestaoManuntencao\Relatorio.docx#_Toc59386905)

[Figura 2 - Processo têxtil 17](#_Toc59386906)

[Figura 3 - Diagrama caso de uso 22](file:///C:\Users\José%20Carlos\Documents\GitHub\ProjetoGestaoManuntencao\Relatorio.docx#_Toc59386907)

### ÍNDICE DE TABELAS

[Tabela 1- Requisitos funcionais 20](#_Toc59385011)

[Tabela 2 - Requisitos não funcionais 22](#_Toc59385012)

### Glossário

**Software** -na gíria comum, traduz-se como sendo um programa que traz a experiência de interagir com equipamentos. Em termos técnicos, é usado para designar a parte lógica que fornece diretivas para hardwares presentes num equipamento.

**Hash** - ou função Hash, são usadas para fins de segurança para criptografar por exemplo palavras-passe.

**Framework** - é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Em termos mais simples, permite reutilizar código sem necessidade de ter de reescrever.

**Front-end** - parte da aplicação que interage diretamente com o utilizador.

**Back-end** - parte da aplicação que interage entre a base de dados e o front-end.

**Token** - é um sistema de transferência de dados que pode ser enviado via URL, POST ou num cabeçalho HTTP (header) de maneira "segura", essa informação é assinada digitalmente.

**Header** - traduzido em português como cabeçalho, é uma componente usada nos pedidos e respostas HTTP, onde definem os parâmetros destas mesmas transações.

**Authorize** - filtro de autorização no back-end. Caso implementado num método, só permite execução desse mesmo método caso o utilizador esteja autenticado.

**Role** – propriedade utilizada no back-end, de forma a definir que tipo de utilizador da aplicação terá acesso a determinado método/componente.

**Web Browser** - é um programa que habilita utilizadores a interagirem com documentos HTML hospedados num servidor de rede.

### Siglas e Acrónimos

**API** Application Programming Interface (Interface de Programação de Aplicação – que estabelece um conjunto rotinas e padrões de um software).

**HTTP** HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto).

**SQL** Structured Query Language (Linguagem de Consulta Estruturada).

**.NET** Plataforma única para desenvolvimento e execução de sistemas e aplicações.

**LINQ** Language Integrated Query (Consulta integrada à linguagem) - componente da framework .NET que adiciona funcionalidades de consulta em algumas linguagens de programação .NET.

**RF** Requisito funcional - representa o que o software faz, em termos de tarefas e serviços.

**RNF** Requisito não funcional - relacionado ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.

**EXE** é uma extensão de ficheiros que podem ser executados por computadores que têm por base sistema operativo Windows.

**VPN** Virtual Private Network (Rede privada virtual), é uma rede de comunicações privada construída sobre uma rede de comunicações pública.

**ERP** Enterprise Resource Planning (planeamento de recurso corporativo), é um sistema de gestão empresarial.

# **Introdução**

Este documento descreve o processo de implementação de uma aplicação desenvolvida no âmbito do estágio do Curso de Licenciatura em Engenharia de Sistemas Informáticos.

Este estágio e projeto, tem como objetivo a aplicação prática dos vários conhecimentos que foram adquiridos ao longo dos 3 anos de licenciatura num contexto de trabalho real.

O projeto foi desenvolvido com vista a resolver um problema de registo e consulta de parâmetros de avarias, e suporte de equipamentos de produção da organização de acolhimento, que até aqui era feito em suporte de papel.

O projeto aqui apresentado interliga várias àreas de programação, desde base de dados, API e WEB, sendo posto logo de parte o uso de uma aplicação do tipo .exe, por várias razões: um sistema WEB pode ser acessado em qualquer equipamento (tablet ou computador) conectado ao à rede interna da empresa. A segunda razão, prende-se pelo facto da programação WEB ser bastante importante no mercado de trabalho.

## **Contexto e objetivos**

A [Acatel - Acabamentos Têxteis S.A.](https://acatel.pt/pt/) (empresa de acolhimento), presente no mercado desde 1985, presta serviços de tinturaria, mercerização, tingimento à peça, estamparia digital e tradicional e acabamentos de malhas e tecidos.

Como trabalhador nesta empresa e ao mesmo tempo estudante no IPCA e sabendo a admnistração que me encontrava naquele momento a estudar, foi feita a proposta da realização deste projeto.

A existência de variadas soluções para este tipo de projeto que existem no mercado real, não significam que estas se adequem à empresa em questão, pois estamos perante um projeto criado de raíz, de modo a satisfazer os pormenores pedidos pela organização.

Assim, por ter sido criado de raíz, exigiu um esforço adicional para que o código desenvolvido seja validado e testado, de modo a não permitir falhas durante a sua produção em tempo real.

A seguinte figura representa o diagrama de blocos que apresenta uma visão geral sobre todo o sistema desenvolvido.

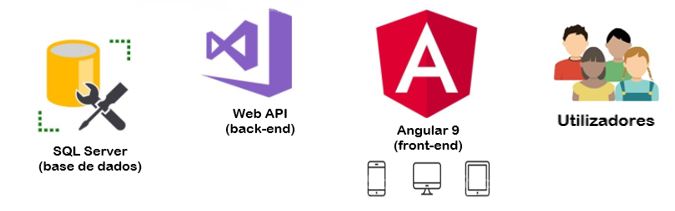
[](#_ÍNDICE_DE_FIGURAS)

Figura 1 - Diagrama de blocos geral

Os utilizadores interagem com uma aplicação WEB, podendo ser utilizada num equipamento que suporte um browser compatível com as funcionalidades de Angular, com recurso a HTTP, sendo que esta utiliza a API (.NET Framework), que por sua vez estabelece conexão com a base de dados (SQL Server).

## **Estado da arte**

Esclarecidos os objetivos para este projeto, foi hora de analisar o que seria necessário para criar este trabalho. Como dito no capítulo anterior, o objetivo seria criar um sistema adequado às especificações da admnistração, num ambiente fechado (servidor interno da empresa de acolhimento, no domínio interno, afastado da rede exterior), necessário estar disponível em todos os computadores ou equipamentos móveis da empresa (telemóveis ou tablets que suportem um browser atualizado).

A empresa em questão está dividida em várias secções. Somando todo o número equipamentos fixos (computadores), são várias dezenas. A isto, soma-se o número de equipamentos móveis (tablets, telemóveis).

As aplicações WEB (dependendo sempre da àrea a que se aplicam) estão em constante evolução, contudo, aplicações desktop e WEB continuarão a coexistir por muito tempo, cada uma com suas vantagens e desvantagens. Neste caso, era desavantajoso instalar uma aplicação .*exe* em todos os computadores da empresa, estes são também limitados pelo *hardware* no qual são executados, devem ser desenvolvidos num sistema operacional específico.

Por outro lado, uma aplicação WEB apenas necessita de um browser que suporte as frameworks usadas no desenvolvimento de código, podendo até mesmo, ser acessíveis fora do dominio interno da empresa, através de VPN.

***Ver a evolução entre .exe app vs web app e também os ERP disponíveis.***

## **Estrutura do documento**

Este documento está organizado em 3 (três) capítulos principais além do presente capitulo de introdução.

No capítulo inicial, é feita a análise ao problema, as ferramentas utilizadas, onde inclui uma análise dos requisitos, funcionalidades necessárias para a aplicação e apresentados os diagramas de caso de uso. Em seguida são expostas as tecnologias e metodologias utilizadas no processo de implementação do sistema.

No capítulo de desenvolvimento estão expostos todos os sistemas desenvolvidos. Este capitulo está dividido em subcapítulos que falam sobre cada um dos sistemas desenvolvidos durante o projeto, nomeadamente a base de dados, a API (back-end) e a interface com o utilizador (front-end), assim como os resultados conseguidos.

Por fim são apresentados possíveis melhorias na aplicação, e conclusões finais do projeto.

# **Desenvolvimento**

Para o desenvolvimento da aplicação é necessário identificar as necessidades da empresa, ou seja, conhecer o modus *operandi* da organização de acolhimento, para perceber a necessidade de que dados registar e como os registar.

A Acatel Acabamentos Têxteis S.A., é uma empresa têxtil que opera na área dos acabamentos e beneficiação têxtil. A empresa presta serviços a outras empresas na área de beneficiamento têxtil, mais concretamente em tinturaria, tinturaria á peça, estamparia digital, estamparia convencional e acabamentos de malhas e tecidos.

O desenvolvimento da sua atividade industrial está dependente de vários sistemas informáticos, contudo nenhum que seja específico para a manutenção dos equipamentos da empresa. A seguinte figura mostra o processo têxtil desde as fibras até á peça pronto e em loja.

Figura 2 - Processo têxtil

## **2.1 Análise ao problema**

A função da empresa de acolhimento contém diferentes secções. Diferentes secções, contêm diferentes grupos de máquinas, e para cada grupo de máquina, há diferentes tipos de máquinas.

A realização deste trabalho, tem por função organizar a manutenção de uma empresa que contém mais de 150 (cinto e cinquenta) máquinas, de diferentes tipo, tamanhos e funcionalidades.

O esquecimento, ou o não cumprimento da manutenção destes equipamentos pode comprometer a reputação desta empresa, visto que as encomendas têm prazos de entrega, que necessitam de ser cumpridos. O não cumprimento destes prazos devido à falta de organização da manutenção, ou da não realização de tarefas de manutenção preventiva por parte de um funcionário, de modo a prevenir avarias, pode comprometer a relação entre um cliente e a empresa de acolhimento.

As tarefas de um funcionário pertencente à equipa de manutenção (serralheiro ou eletricista), dividem-se em duas categorias: preventivas ou curativas.

As manutenções preventivas, são aquelas que são programadas pelo Eng.º da equipa de manutenção, necessárias para o bom funcionamento destes equipamentos, de forma a minimizar o desgaste dos mesmos.   
Por outro lado, as manutenções curativas não são programadas, e acontecem quando a manutenção preventiva pode ter falhado por algum motivo alheio, e é necessária a sua reparação, podendo esta ser notificada, avaliada e realizada.

Tornar este processo mais dinâmico através desta aplicação, melhora a classificação das normas que a empresa necessita de cumprir *(ISO 13372:2012,* normas de manutenção*),* requisitados pelos exigentes clientes.

## **Requisitos funcionais**

O levantamento de requisitos foi efetuado com atenção às funcionalidades e detalhes que seriam necessários/fundamentais a serem implementados na aplicação. Os mesmos foram separados por requisitos funcionais e requisitos não-funcionais.

Tabela 1- Requisitos funcionais

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisitos Funcionais** | | | |
| **ID** | **Tag** | **Descrição** | **Prioridade** |
| RF01.1 |  | O sistema deverá permitir registar novos utilizadores:   * Login; * Password; * Nome completo; * Número de trabalhador; * Secção de trabalho; * Tipo de perfil (administrador, técnico, funcionário comum); | **SHOULD** |
| RF01.2 | Gestão de utilizadores | O sistema poderá permitir inativar utilizadores. | **MUST** |
| RF01.3 |  | O sistema poderá permitir associar o utilizador a um perfil de acesso. | **SHOULD** |
| RF01.4 |  | O sistema deverá conter pelo menos estes três perfis:   * Admistrador; * Técnico de reparação; * Funcionário (chefe de secção de cada turno, para fazer relatar as anomalias); | **SHOULD** |
| RF02.1 | Gestão de Equipamentos | O sistema deverá permitir adicionar equipamentos, com os seguintes campos obrigatórios:   * Código interno e descrição; * Grupo de máquina pertencente; * Marca e número de série; * Tipo; * Data de compra e desativação; * Estado do equipamento (ativo/desativo); | **SHOULD** |
| RF02.2 |  | O sistema deverá permitir inativar equipamentos (equipamentos que deixam de operar, vendidos ou obsoletos). | **MUST** |
| RF03.1 |  | O sistema deverá permitir a criação de pedidos de assistência do tipo curativa ou preventiva. | **SHOULD** |
| RF03.3 |  | Os pedidos e intervenções deverão ter os seguintes estados:   * Sem intervenção; * Programado/(a); * Fechado; | **SHOULD** |
| RF03.3 | Gestão de pedidos de assistência | O sistema deverá para o tipo curativo conter os seguintes campos:   * Hora do pedido (introduzida automaticamente pelo servidor); * Hora da intervenção (início e fim, introduzida pelo utilizador); * Tempo de intervenção por técnico (num pedido podem ocorrer várias várias intervenções); * Utilizador que faz o pedido (introduzido automaticamente pelo servidor, baseado no login); * Equipamento; * Descrição da avaria; * Técnico que fez a intervenção; * Descrição da intervenção realizada; | **SHOULD** |
| RF03.4 |  | O sistema deverá para o tipo de assistência preventiva conter os seguintes campos:   * Hora do pedido (introduzida automaticamente pelo servidor); * Hora da intervenção (início e fim, introduzida pelo utilizador); * Horas de intervenção por técnico ( num pedido podem ocorrer várias vári * as intervenções ); * Utilizador que faz o pedido (introduzido automaticamente pelo servidor, baseado no login); * Equipamento; * Técnico que fez a intervenção; * Equipamento; * Descrição da intervenção realizada; * Limite de intervenção (caso seja preventiva, e programada pelo Eng.º de manutenção); | **SHOULD** |
| RF04.1 | Gestão de Planos | O sistema deverá permitir associar um técnico a um grupo de equipamentos para manutenções preventivas. | **SHOULD** |
| RF04.2 |  | O sistema deverá alertar o técnico de manutenções programadas a executar. | **MUST** |
| RF04.3 | Gestão de planos  (continuação) | O Eng.º de manutenção, emite um pedido de manutenção preventiva, com uma data limite de manutenção para grupo de máquinas (plano de prevenção). | **SHOULD** |
| RF04.4 |  | O sistema deverá permitir criar um plano de manutenção anual. | **MUST** |
| RF05.1 |  | O sistema deverá permitir a consulta de manutenção de um determinado grupo de máquinas, alojado, em formato *pdf*. | **SHOULD** |
| RF05.2 | Consulta de registos | O sistema deverá permitir consultar intervenções anteriormente realizadas. | **SHOULD** |
| RF05.3 |  | O sistema deverá permitir a consulta do relatório de manutenção anual. | **COULD** |

## **2.1.2 Requisitos não funcionais**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisitos Não Funcionais** | | | |
| **ID** | **Requisito** | **Descrição** | **Prioridade** |
| RNF01 | Desempenho | O sistema deverá apresentar bom desempenho: respostas rápidas após clique e a carregar conteúdos. | **MUST** |
| RNF02 | Usabilidade | O sistema deverá apresentar uma interface simples e intuitivo, facilitando a sua utilização. | **MUST** |
| RNF04 | Interoperabilidade | O sistema deverá comunicar com SQL Server. | **MUST** |
| RNF05 | Confiabilidade | O sistema deverá estar disponível durante todo o tempo de produção. | **SHOULD** |
| RNF06 | Idiomas | O sistema deverá suportar o idioma português | **SHOULD** |
| RNF07 | Segurança | O sistema deverá oferecer segurança, nem todas as pessoas com acesso a um computador, podem ter acesso à aplicação. | **SHOULD** |

Tabela 2 - Requisitos não funcionais

## **2.1.3 Diagrama de casos de uso**

A seguinte figura representa o diagrama de casos de uso de forma simplificada das principais tarefas do sistema da aplicação.

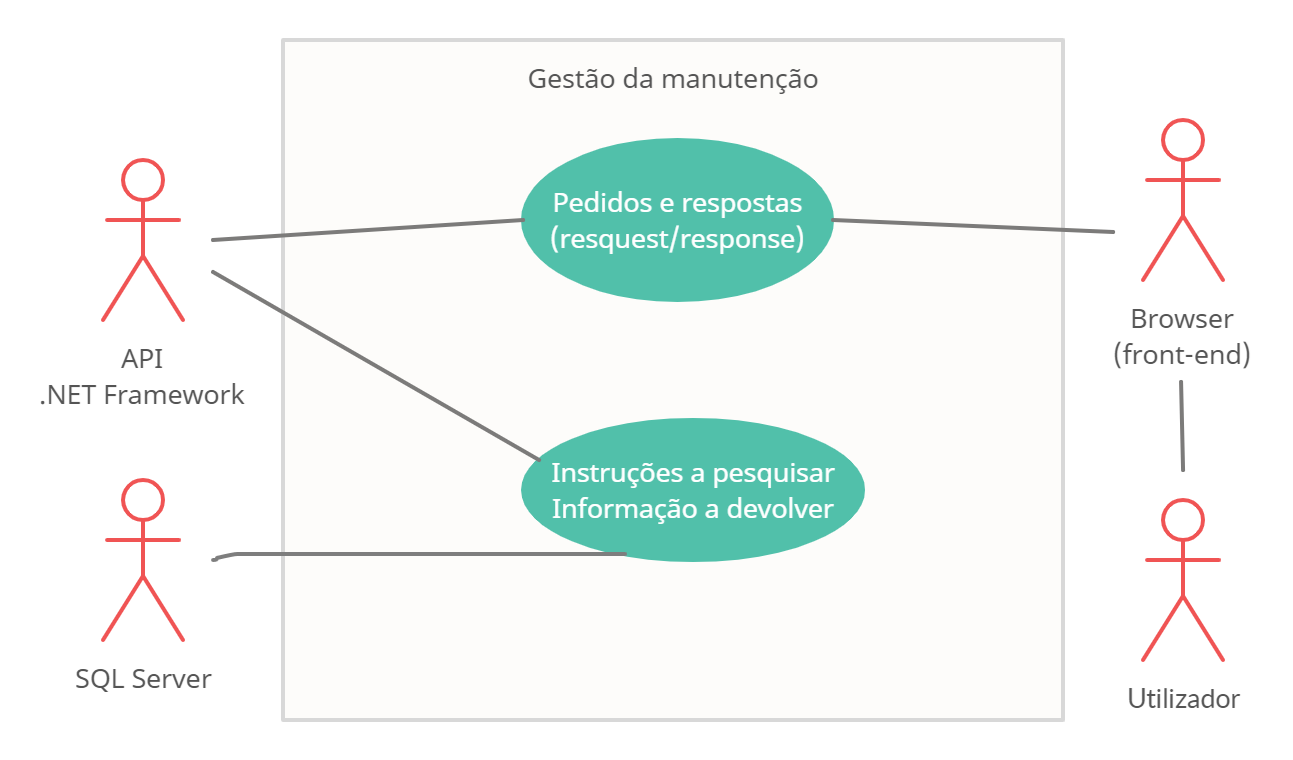


Figura 3 - Diagrama caso de uso

O diagrama demonstra as ações da aplicação inteira, sem referir para já que tipo de tecnologia é usada nas comunicações entre as diferentes partes, que também dependem das ações dos utilizadores, pois há ações do utilizador (front-end), que não necessitam de comunicar com a API (back-end).

Cada caso de uso tem uma descrição de funcionalidade que será construída no sistema proposto. De seguida são apresentadas as tabelas descritivas de cada caso de uso apresentado na figura anterior.

**PEDIDOS E RESPOSTAS**

|  |  |
| --- | --- |
| **PRÉ-CONDIÇÃO** | N/A |
| **DESCRIÇÃO** | O utilizador envia uma instrução. |
| **PÓS-CONDIÇÃO** | Primeiramente, a API verifica que tipo de permissão tem antes de executar a instrução. |

**INSTRUÇÕES A PESQUISAR, INFORMAÇÃO A DEVOLVER**

|  |  |
| --- | --- |
| **PRÉ-CONDIÇÃO** | N/A |
| **DESCRIÇÃO** | A API envia uma instrução à base de dados. |
| **PÓS-CONDIÇÃO** | A base de dados retorna informação, caso a instrução a validar tem resultados positivos. |

**Diagrama de atividades para receção e envio de informação**

**MELHORAR O DIAGRAMA DE CASOS DE USO E AS TABELAS   
(ESTOU CONFUSO NESTA PARTE)**

## **2.2 Plataformas de desenvolvimento**

* O Microsoft Visual Studio é um pacote de programas do Microsoft para desenvolvimento de software dedicado especialmente ao .NET Framework e às linguagens de programação Visual Basic, C, C++, C#, e J#.



Figura 4 - Logótipo Visual Studio

* O Microsoft SQL Server, é um sistema gerenciador de bancos de dados relacional, desenvolvido em parte pela Microsoft, compatível também com Visual Studio.



Figura 5 - Logótipo SQL Server

* O Visual Studio Code, é um editor de código criado pela Microsoft, para Windows, Linux e macOS, sendo ideal para trabalhar em linguagens tais como TypeScript (TS), JavaScript (JS), HTML e CSS.



Figura 6 - Logótipo Visual Studio Code

* Angular é uma framework para aplicações WEB e front-end, baseado em TypeScript, reescrita de AngularJS, criada pela Google.



Figura 7 - Logótipo Angular

## **2.3 Introdução à base de dados**

A base de dados disponível no servidor interno com licença, trata-se de SQL Server, tratando-se de um modelo de dados relacional. É também um bom candidato para usar em contínuo com a ferramenta Visual Studio, sem necessitar de drivers adicionais para estabelecer conexão entre a base de dados e a API a consumir.

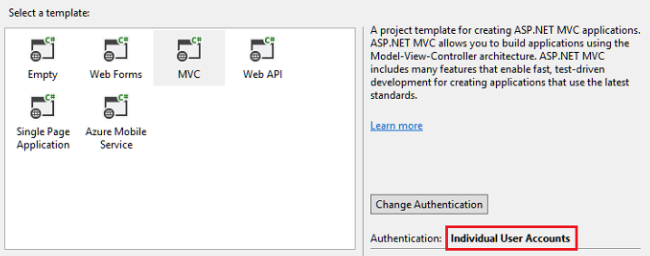
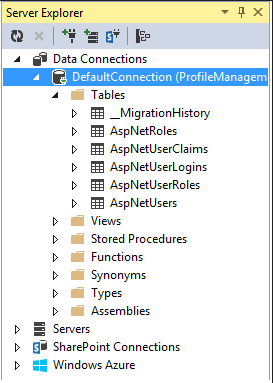
Por definição, um projeto .NET Framework escolhido com *individual user accounts authentication,* como o nome indica tem incorporado um sistema de autenticação de utilizadores. Assim, foi escolhido um projeto MVC (model, view, control). Nem tudo neste projeto criado por definição será usado, em substituição da *view,* o usado é a framework Angular (com recurso ao Visual Studio Code).

Figura 8 - Criação da API (Visual Studio)

Como referido anteriormente, por definição, este projeto acarta consigo o sistema de autenticação de contas de utilizador individuais, podendo guardar consigo dados de um utilizador, tais como email, password (encriptada sobre *hash)*.

Mais tarde neste relatório, é exposta a manipulação da classe *Register*, uma classe dedicada à criação de um novo utilizador na aplicação, para conter novos dados para além do *username* e da *password,* tais como o nome do utilizador, o tipo de permissão, entre outros.

Na figura abaixo, é mostrada a base de dados relacional para autenticação de utilizadores, contudo, não sendo necessárias todas as tabelas, esta foi adaptada de forma a coincidir com as necessidades da aplicação.

Nesta base de dados criada por definição, as tabelas que têm aqui interesse são as seguintes:

* + - * ***AspNetUsers***: tabela que contém as informações pessoais do utilizador.
      * ***AspNetRoles****:* tabela que contém o *ID* do tipo de permissão e a descrição dessa permissão.
      * ***AspNetUserRoles****:* tabela que serve de interface entre as duas tabelas descritas anteriomente, tendo a informação da *role* e o *ID* do utilizador.

Figura 9 - Base de dados do projeto

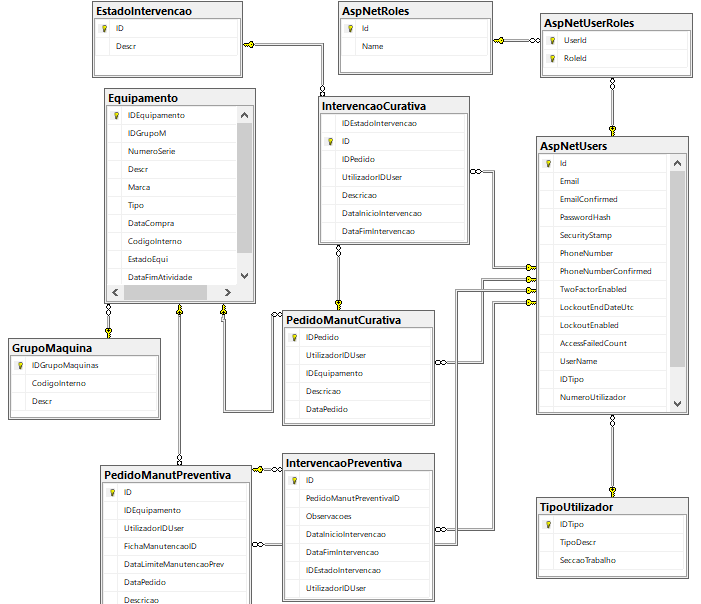
A figura seguinte, evidencia o transporte da base de dados anteriormente citada, para a base de dados do modelo de negócio.

Figura 10 - Base de dados da aplicação

A base de dados criada de forma a servir a aplicação, para além do que já foi citado no parágrafo anterior, contém essencialmente:

* Equipamentos: são todas as máquinas presentes na empresa que têm o propósito de servir o bom funcionamento da mesma.
* Grupo de máquina: destinada a conter a informação relativa aos diferentes grupos de equipamentos da empresa, podendo haver diferentes grupos de máquina na mesma secção da empresa.
* Pedidos de manutenção curativa: destinada a conter a informação dos pedidos curativos referentes aos equipamentos.
* Pedidos de manutenção preventiva: destinada a conter a informação dos pedidos preventivos referentes aos equipamentos.
* Intervenções curativas: destinada a conter a informação relativa a um pedido curativo. Um pedido curativo pode ter mais que uma intervenção.
* Intervenções preventivas: destinada a conter informação relativa a um pedido preventivo. Um pedido preventivo pode ter mais que uma intervençao.
* Estado de intervenção: destinada a conter informação relativa ao estado em que se encontra um pedido (curativo ou preventivo). Neste caso, o estado assenta na intervenção, contudo, no *front-end,* é feita uma transformação que, baseada na última intervenção de um pedido, é mostrado o estado em que se encontra o pedido (se contém alguma intervenção, se este já foi analisado mas ainda não reparado, ou se o problema já se encontra solucionado).
* Tipo de utilizador: destinado a conter informação relativa a um determinado utilizador, tal como a secção da empresa em que trabalha, e a sua função na empresa.

## **2.4 Introdução à API (front-end)**

O tipo de projeto aqui utilizado (já referido no capítulo anterior), MVC é um padrão de arquitetura de software. O seu princípio básico do é a divisão da aplicação em três camadas: a camada de interação do utilizador (***view,* esta não é utilizada, será substituida pela framework Angular, por ser por exemplo mais responsiva**). A camada de manipulação dos dados (***model***) e a camada de controlo (***controller***).

Assim sendo, o *model* tem a responsabilidade de representar o negócio (base de dados), e é também responsável pelo acesso e manipulação dos dados na aplicação, trata-se essencialmente da base de dados importada do SQL Server (*ADO.NET Entity Data Model).*

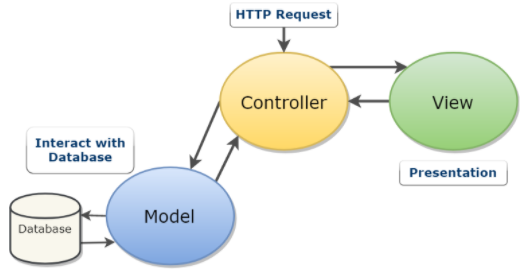
****Por outro lado, os *controllers*, têm a responsabilidade de interligar o modelo e o front-end, fazendo com que o modelo possa ser repassado para o front-end e vice versa.

Figura 11 - Arquitetura/Padrão MVC

## **2.4.1 Controladores**

## **2.4.2 DTO**

CORS; BASE DE DADOS CONNECTION; ETC