**IPV – Instituto Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu Departamento de Informática**

****

**Relatório de Trabalho Prático Licenciatura em Engenharia Informática**

**Realizado em Base de dados II por**

**Daniel Valpereiro –22894**

**Guilherme Félix – 25172**

**Cassamo Latif – 21373**

**Orientadores**

**Entidade: Instituto Politécnico de Viseu ESTGV: Paulo Tomé**

**João Henriques**

**Índice**

[1 Introdução 4](#_Toc188741256)

[1.1 Estrutura 4](#_Toc188741257)

[2 Fase 1 - Análise e planeamento 5](#_Toc188741258)

[2.1 Definição dos grupos de trabalho 5](#_Toc188741259)

[2.2 Identificação das responsabilidades individuais para cada tarefa 5](#_Toc188741260)

[2.3 Calendarização das atividades do projeto (Diagrama de Grantt) 5](#_Toc188741261)

[3 Fase 2 – Desenho 4](#_Toc188741262)

[3.1 Listagem das funcionalidades com a identificação dos requisitos 4](#_Toc188741263)

[3.1.1 Breve explicação das funcionalidades e requisitos 4](#_Toc188741264)

[3.2 Desenho da Arquitetura 6](#_Toc188741280)

[3.3 Desenho da base de dados 7](#_Toc188741281)

[3.4 Modelo conceptual de dados 8](#_Toc188741282)

[3.5 Modelo físico de dados 8](#_Toc188741283)

[3.6 Produção de um protótipo 9](#_Toc188741284)

[3.7 Planeamento dos testes sobre a base de dados 10](#_Toc188741285)

[4 Fase 3 – Desenvolvimento 11](#_Toc188741286)

[4.1 Implementação dos objetos a incluir na base de dados 11](#_Toc188741287)

[4.2 Implementação das funcionalidades a incluir na plataforma 11](#_Toc188741288)

[5 Scripts 12](#_Toc188741289)

[5.1 Script de Instalação 13](#_Toc188741290)

# Introdução

O trabalho prático tem como tema Processos na Indústria de Assemblagem de Computadores. O projeto irá ser dividido em 5 partes, proposta pelos professores orientadores, sendo estas cinco fases as seguintes:

* Fase 1 – Conceção
* Fase 2 – Desenho
* Fase 3 – Desenvolvimento
* Fase 4 – Avaliação
* Fase 5 – Instalação

Este relatório aborda a 5 fase.

## Estrutura

Este relatório encontra-se dividida em 3 capítulos. No primeiro e presente capítulo é realizada uma contextualização do estudo.

No segundo capítulo aborda a conceção e planeamento geral do projeto. No último capítulo aborda os scripts.

# Fase 1 - Análise e planeamento

## Definição dos grupos de trabalho

O grupo deste relatório é o nº 17, constituído por 3 elementos :

* Daniel Valpereiro
* Guilherme Felix
* Cassamo Latif

## Identificação das responsabilidades individuais para cada tarefa

Para o planeamento do projeto e identificação das atividades e responsabilidades individuais para cada tarefa, foi utilizado um *template* fornecido pelos professores orientadores, em formado *excel*, facilitando a gestão do projeto. Junto nesta pasta está o dito excel.

## Calendarização das atividades do projeto (Diagrama de Grantt)

Foi utilizado a plataforma *figma* para realizar o Diagrama de Gantt, tendo o seguinte link para o acesso ao mesmo: <https://www.figma.com/design/DFltDgf4ZQ306vmKoE5fGC/Untitled?node-id=0-1&p=f&t=DOTRuoix8jYt06WV-0>

# Fase 2 – Desenho

## Listagem das funcionalidades com a identificação dos requisitos

### Breve explicação das funcionalidades e requisitos

### 1 - Gestão de Componentes Requisitos: 1, 2, 9, 17, 20, 21, 22, 23 R1: Registo de dados sobre aquisição e armazenamento R2: Registo de informações de fornecedores R9: Registo da produção de equipamentos Requisitos Comuns: R17: Dados transacionais armazenados em PostgreSQL R20: Suporte a múltiplos tipos de usuários R21: Testes abrangentes para objetos do banco de dados. R22: Interfaces gráficas funcionais e atrativas. R23: Uso do Git para controle de versões.

### 5 - Exportação de Encomendas e Fornecedores Requisitos: 6, 17, 20, 21, 22, 23 R6: Exportação de encomendas em JSON/XML

### 6 - Importação de Componentes Requisitos: 7, 17, 20, 21, 22, 23 R7: Importação de componentes em JSON/XML

### 7 - Cadastro de Tipo de Equipamento Requisitos: 8, 18, 19, 17, 20, 21, 22, 23 R8: Cadastro de tipos de equipamento R18: Armazenamento de características no MongoDB R19: Atributos dinâmicos para apresentação ao cliente

### 8 - Registo de Produção de Equipamentos Requisitos: 9, 10, 19, 17, 20, 21, 22, 23 R10: Associação de custos de mão de obra

### 9 - Gestão de Venda Requisitos: 14, 18, 19, 17, 20, 21, 22, 23 R14: Registro de vendas de equipamentos

### 10 - Gestão de Compra Requisitos: 13, 17, 20, 21, 22, 23 R13: Registro de compra de componentes

### 11 - Cadastro de Informações de Clientes Requisitos: 15, 17, 20, 21, 22, 23 R15: Cadastro de clientes

### 11 - Listagem de Clientes Requisitos: 16, 17, 20, 21, 22, 23 R16: Listagem de clientes e vendas

### 12 - Listagem de Equipamentos Requisitos: 12, 18, 19, 17, 20, 21, 22, 23 R12: Listagem diversa

### 13 - Listagem de Stock Requisitos: 1, 3, 12, 17, 20, 21, 22, 23 R3: Controle de stock

### 14 - Listagem de Componentes Requisitos: 4, 5, 12, 17, 20, 21, 22, 23 R4: Consulta de dados de componentes.

### 15 - Listagem de Utilizadores Requisitos: 16, 17, 20, 21, 22, 23

### 16 - Filtragem de Dados Requisitos: 3, 4, 10, 12, 17, 20, 21, 22, 23

### 17 - Registo de Cliente Requisitos: 15, 17, 20, 21, 22, 23

### 3.1.2 Requisitos Funcionais:

### Desenvolvimento de uma plataforma de suporte aos processos da organização/sistema especificado.

### A plataforma deve basear-se em uma interface web para interação com o utilizador.

### Uso de pelo menos 6 entidades de dados no modelo conceptual.

### Implementação de fluxos de dados que envolvam pelo menos dois tipos diferentes de entidades externas, como fornecedores e clientes.

### Desenvolvimento de funcionalidades específicas relacionadas aos processos da organização.

### Implementação de diferentes tipos de utilizadores, com permissões distintas para acesso e interação com os dados.

### Seleção de uma entidade conceptual para armazenamento no SGBD MongoDB.

### Cobertura de todos os objetos da base de dados com testes funcionais, indicando resultados como "OK" ou "Not OK".

### Criação de interfaces gráficas funcionais e apelativas.

### Gestão de perfis de utilizadores na plataforma.

### Requisitos Não Funcionais:

### Armazenamento de todos os dados relativos a operações transacionais no SGBD PostgreSQL.

### Utilização de dois motores de base de dados (PostgreSQL e MongoDB), garantindo que cada dado é armazenado apenas em um deles.

### Uso da framework Django para o desenvolvimento das interfaces gráficas.

### Garantia de que as interfaces gráficas sejam responsivas e atraentes para os utilizadores.

### Implementação de testes sobre a base de dados para verificar a funcionalidade de todos os objetos.

### Uso do sistema de controlo de versões Git, com commits individuais, para avaliação do desempenho e das responsabilidades de cada membro do - -grupo.

### Respeito a prazos de entrega e fases do projeto, com a produção de artefactos específicos para cada etapa.

### Produção de documentação detalhada para todas as fases do projeto, incluindo relatórios, tabelas e diagramas.

### Garantia de qualidade nas entregas semanais e nas versões do projeto.

## Desenho da Arquitetura

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamenteO cliente tem interações com a aplicação web nas quais pode executar as operações CRUD, depois dependendo do tipo de objetos sobre os quais está a executar tais operações o servidor web irá comunicar com a base de dados relacional ou com a base de dados não relacional (PostgreSQL ou MongoDB). A seguir está um esquema representativo do fluxo de dados entre os componentes.

*Figura 3 - Desenho da Arquitetura*

## Desenho da base de dados

Como a aplicação irá ter opção CRUD sobre objetos, é necessário uma base de dados. O desenho da mesma apresenta-se a seguir. Como se pode observar, cada objeto possui cores diferentes no esquema, para maior facilidade de compreensão, como também existem ligações entre as entidades.

Em resumo, um componente possui tipo de componentes, armazenado em um armazém, e ao mesmo tempo tem ligação com fichas de detalhes, seja de produção ou encomenda. Na parte da encomenda de componentes, relaciona-se com fornecedores, que eventualmente poderá ter ligações com encomendas do cliente, neste caso com guias remessas. Já a produção, relaciona- se mais com os diversos atributos de produção, como mão de obra, componentes necessários ou os custos para a construção dos equipamentos, preparados para os clientes. Estes podem realizar encomendas de equipamentos que possui detalhes e um estado das encomendas. As fichas de produção e guias remessas podem ser acedidas pelos utilizadores, e estes têm um perfil para cada tipo.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

*Figura 4 - Desenho da BD*

## Modelo conceptual de dados

O modelo conceptual de dados, representa as ligações com a base de dados, com maior detalhe em cada entidade, como os campos, as chaves primarias, o tipo de valor no campo e a carnalidade das ligações. Descrevem o conceito e relacionamento das entidades no domínio.

As ligações neste modelo são semelhantes ao desenho da base de dados.

Uma imagem com texto, diagrama, Paralelo, Esquema

Descrição gerada automaticamente

*Figura 5 - Modelo conceptual de dados*

## Modelo físico de dados

De modo semelhante ao modelo conceptual, o modelo de dados físico mostra as ligações da base de dados. Diferente do modelo conceptual, o modelo físico mostra menos informações. Tem como informações a chave primaria, chaves estrangeiras e as ligações sem carnalidade. No geral, representa o modo de armazenamento fisico no sistema.

Uma imagem com texto, captura de ecrã, diagrama, Paralelo

Descrição gerada automaticamente

*Figura 6 - Modelo físico de dados*

## Produção de um protótipo

A fase de desenho, inclui um protótipo de alta fidelidade. Este representa os desenhos inicias das web quando o sistema for implementado.

De acordo com a lista de funcionalidades e de requisitos, as paginas que serão mostradas a seguir visam a corresponder aos requisitos e funcionalidades necessárias para a implementação do sistema. Foi utilizado a plataforma Figma para a construção do protótipo.

Poderá aceder ao mesmo através deste link: [https://www.figma.com/design/DFltDgf4ZQ306vmKoE5fGC/Untitled?node-id=0-1&p=f&t=L7pscxwPST040Vmd-0](mailto:https://www.figma.com/design/DFltDgf4ZQ306vmKoE5fGC/Untitled?node-id=0-1&p=f&t=L7pscxwPST040Vmd-0)

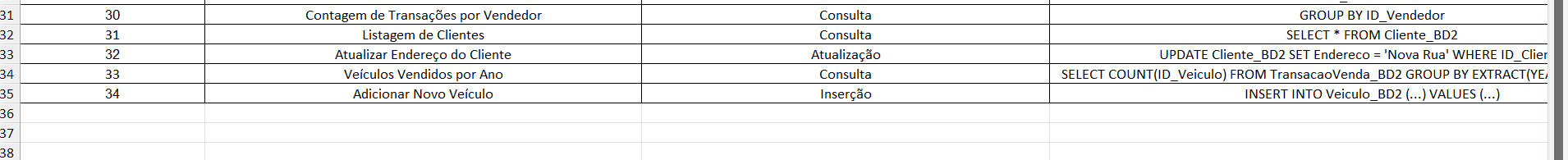
## Planeamento dos testes sobre a base de dados

A última parte da fase 2, desenho, aborda os testes realizados sobre a base de dados. Estes testes são: Procedimentos Armazenados, Views e Funções. Estes testes permitem verificar se existe algum erro para as operações, permitindo a confiabilidade e o bom funcionamento do sistema. Caso tudo resulte em sucesso, significa que não existe erros nos testes, preparado para a implementação da próxima fase do projeto.

Uma imagem com texto, captura de ecrã, file, número

Descrição gerada automaticamente

*Figura 7 - Planeamento de testes 1-29*

**

*Figura 8 - Planeamento de testes 29-34*

# Fase 3 – Desenvolvimento

## Implementação dos objetos a incluir na base de dados

Para a implementação criamos Procedimentos Armazenados, Funções e Triggers para serem utilizadas posteriormente na aplicação web para a comunicação, abaixo deixo o link do *Github* onde tem a os scripts para a criação dos objetos que estão dentro da pasta *CRUD*.

<https://github.com/Base-de-dados-2/Ficheiros>

## Implementação das funcionalidades a incluir na plataforma

Nesta parte começamos por criar o projeto *Django* e criamos uma app com o nome “app” e criamos um repositório no Github para facilitar o trabalho em grupo (<https://github.com/Base-de-dados-2/Aplicacao>) e decidimos utilizar a seguinte estrutura para organizar o código:

* A pasta “database” para guardar todas as funções que são utilizadas para conectar, ir buscar dados ou adicionar/atualizar dados da base de dados (exemplo, ficheiro utilizador.py:

<https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplicão/app/Database>

* A pasta “formulários” para separar o código dos formulários por tabela e não ter tudo em um só ficheiro (exemplo, ficheiro fichaproducao\_form.py: [https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplicão/app/Formularios](https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplica%C3%A7%C3%A3o/app/Formularios)
* A pasta “templates” organizada por tabela pelo mesmo motivo do ponto anterior (exemplo, pasta componentes:

<https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplicão/app/templates>

* A pasta “vistas” pelo motivo dos pontos anteriores (exemplo, ficheiro equipamento\_views.py:

[https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplicão/app/Vistas](https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplica%C3%A7%C3%A3o/app/Vistas)

E organizamos os URLs por secção, ou seja, no caso dos utilizadores tem /utilizadores/listar,

/atualizar/id e /registar optamos por neste caso não adicionar o apagar pois isso iria causa dados ambíguos para as tabelas que a referenciam (link Github )

<https://github.com/Base-de-dados-2/Aplicacao/blob/main/Main-aplica%C3%A7%C3%A3o/app/urls.py>

# Scripts

1. **Funções JSON :**

GetTransacaoVendaJson()

* + Retorna todas as transações de venda em JSON, incluindo detalhes do veículo, cliente e vendedor (filtra por ID\_Vendedor = 1).

GetReceitaTotalPorStandJson()

* + Calcula a receita total de cada stand (agrupando por nome) e retorna em JSON.

GetPromocoesAtuaisJson()

* + Lista promoções ativas (com desconto aplicado) em JSON, mostrando veículos, preços originais e com desconto.

GetTransacaoVendaDetalhesJson()

* + Similar à primeira função, mas inclui mais detalhes (como ID do cliente e vendedor) e ordena por data decrescente.

1. **Funções XML**

GetReceitaAnualXml()

* + Retorna a receita total por ano (agregando vendas) em formato XML.

GetVeiculosComPoucasVendasXml()

* + Lista veículos com 0 ou 1 venda em XML, incluindo preço médio de venda.

GetTransacaoVendaDetalhesXml()

* + Versão em XML da função GetTransacaoVendaDetalhesJson(), com estrutura hierárquica

1. **Triggers**

AtualizaResumoStand()

* Atualiza automaticamente a tabela ResumoStand\_BD2 (total de veículos, vendas e receita) após alterações em TransacaoVenda\_BD2.

Atualiza\_total\_vendas()

* Atualiza o campo Total\_Vendas do vendedor na tabela Vendedor\_BD2 sempre que uma nova venda é inserida.

Atualiza\_data\_criacao()

* Define a data atual (NOW()) no campo data\_criacao da tabela pedidos antes de inserir um novo registro.

1. **Procedimentos**

Relatorio\_receitas()

* Gera um relatório de receitas por stand, incluindo veículos disponíveis, vendas e receita total.

Relatorio\_clientes\_interessados()

* Lista clientes interessados em veículos que ainda não foram vendidos.

Relatorio\_total\_vendas\_por\_vendedor()

* Mostra o total de vendas por vendedor (com nome e ID).

Update\_veiculo\_preco\_desconto()

* Aplica 10% de desconto no preço de um veículo específico (via ID\_Veiculo).

Update\_fornecedor\_contato()

* Atualiza o contato de um fornecedor na tabela Fornecedor\_BD2.

1. **Views Materializadas**

Mv\_veiculos\_stand\_central

* Lista veículos disponíveis no stand "Stand Central" (cache para consultas frequentes).

Mv\_historico\_veiculo\_1

* Mostra histórico de manutenções e acidentes do veículo com ID\_Veiculo = 1.

Mv\_transacoes\_vendedor\_1

* Cache das transações do vendedor com ID\_Vendedor = 1.

Mv\_veiculo\_com\_desconto

* Exibe preço com 10% de desconto para o veículo ID\_Veiculo = 1.

Mv\_total\_vendas\_por\_vendedor

* Pré-calcula o total de vendas por vendedor (performance para relatórios).

Mv\_receita\_total\_stand

* Receita total por stand (usada para dashboards rápidos).

Mv\_veiculos\_sem\_fornecedor

* Lista veículos sem fornecedor associado (para auditoria).

Mv\_transacoes\_sem\_vendedor

* Transações sem vendedor associado (identifica inconsistências).

Mv\_resumo\_stand

* Resumo consolidado de veículos, vendas e receita por stand.

Mv\_clientes\_interessados\_nao\_vendidos

* Clientes interessados em veículos não vendidos (oportunidades de venda)

## Script de Instalação

Começamos por instalar o Python pelo seguinte url: <https://www.python.org/downloads/>

Depois instalamos os seguintes módulos:

* pip install django
* pip install django-fontawesome
* pip install psycopg2-binary
* pip install pymongo django-mongoengine

Após instalar os módulos clonamos o repositório do GitHub com o seguinte comando: git clone [https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplicão](https://github.com/Base-de-dados-2/Aplicacao/tree/main/Main-aplica%C3%A7%C3%A3o)

De seguida vamos para o repositório clonado anteriormente e executamos este ultimo comando:

python manage.py runserver