



UNIVERSIDADE
LUSÓFONA

Plataforma de gestão financeira para controlo de custos empresariais

Trabalho Final de curso

Relatório Final

Aluno: Duarte Cambra (21805799)

Orientador: Prof. Dr. Pedro Serra

Trabalho Final de Curso | LEI | 2021/2022

Direitos de cópia

Plataforma de gestão financeira para controlo de custos empresariais, Copyright de Duarte Cambra, ULHT.

A Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Resumo

Nas últimas décadas face ao avanço da tecnologia tem havido um crescimento considerável da quantidade de dados digitais geridos por empresas com a inovação da computação e digitalização. Os métodos de controlo e gestão financeira utilizados antigamente eram pouco eficientes tais como registos em papel ou tabelas simples em formato digital e já não são viáveis dado o imenso volume de dados e necessidade de constante alocação de recursos humanos para manter os registos atualizados e coerentes e conseguirmos extrair daí informação com valor.

O presente trabalho visa o desenvolvimento de um sistema *open source* de gestão financeira para pequenas e médias empresas para controlo de custos associados numa plataforma web online que venha resolver os problemas adjacentes a esta falta de informatização. Este sistema possibilita a organização dos dados, automatização de processos que sejam recorrentes e produção de relatórios e mapas de apoio à decisão com informação em tempo real, tudo isto numa forma simples e intuitiva de forma a esconder a complexidade inerente a estes sistemas, especialmente para aqueles com pouca literacia em tecnologia.

Tudo isto possibilita a alocação de recursos para outros aspetos importantes, o que resulta em menores gastos associados dado que o esforço para manter este sistema é muito mais reduzido do que os sistemas precários anteriormente utilizados e a possibilidade de maior alocação de recursos com foco em aumentar os rendimentos, usando também os relatórios e mapas de apoio à decisão gerados automaticamente pelo sistema e que podem sempre atender às necessidades específicas de cada empresa, dado que o software pode e deve se desenvolver à medida que as necessidades da empresa e do mercado também se adaptem.

Abstract

In the last few years there has been an exponential growth of the amount of data managed by companies face to the technology advancement and data digitalization. The methods of financial control and management used in the past were inefficient, such as paper records or simple tables in a digital format and no longer are viable due to the immense amount of data and constant need to allocate human resources to keep the records updated and consistent.

This current project aims on developing a financial management system to control outlays associated to small and medium enterprises on a web app that can solve all of the problems inherent to this lack of informatization. The data gets organized and categorized, recurrent processes are automated, and reports and decision support maps are produced with live data, all of this made in a simple and intuitive way so the complexity inherent to these systems is hidden so that anyone can be able to use it, especially those that have little technology literacy.

All of this allows to allocate more human resources to other important aspects since the effort required to maintain this system is minimum compared to the inefficient systems used before and allows to allocate more human resources focusing on improving income, using the reports and decision support maps generated automatically by the system, with live data, that can attend to specific needs of each company since the software can and should be improved as the company and market needs also adapt.

Índice

| | |
|--|----|
| Resumo..... | 3 |
| Abstract | 4 |
| Índice | 5 |
| Lista de Figuras..... | 6 |
| Lista de Tabelas | 7 |
| 1 Identificação do Problema | 8 |
| 2 Levantamento e análise dos Requisitos | 9 |
| 3 Viabilidade e Pertinência..... | 11 |
| 4 Solução desenvolvida | 12 |
| 4.1 Estrutura de dados | 12 |
| 4.2 Página Principal (Painel de Informações)..... | 14 |
| 4.3 Página Armazéns, Clientes, Encomendas, Faturas, Funcionários, Fornecedores, Itens, Pagamentos e Cargos | 15 |
| 4.4 Página Inventário | 15 |
| 4.5 Lista de e-mail de clientes | 16 |
| 4.6 Tarefas, Calendário e Mensagens Privadas..... | 17 |
| 5 Benchmarking..... | 19 |
| 6 Método e planeamento | 20 |
| 7 Resultados | 21 |
| 7.1 Testes | 21 |
| 7.2 Página Inicial..... | 22 |
| 7.3 Página Armazéns, Clientes, Encomendas, Faturas, Funcionários, Fornecedores, Itens, Pagamentos e Cargos | 22 |
| 7.4 Página Inventário | 23 |
| 7.5 Página Lista E-mail de Clientes..... | 24 |
| 7.6 Página Tarefas | 25 |
| 7.7 Página Calendário..... | 25 |
| 7.8 Página Mensagens Privadas | 26 |
| 8 Conclusão e trabalhos futuros | 27 |
| Bibliografia | 28 |
| Glossário..... | 29 |

Lista de Figuras

| | |
|--|----|
| Figura 1 - Diagrama de Entidades-Relações da Solução | 13 |
| Figura 2 - Excerto código models.py | 13 |

Lista de Tabelas

| | |
|---|----|
| Tabela 1 – Comparação do DEISI237 com outras soluções gratuitas existentes no mercado | 19 |
| Tabela 2 - Testes e validação da solução | 21 |

1 Identificação do Problema

Nos últimos anos face ao avanço da tecnologia tem havido um crescimento considerável da quantidade de dados digitais geridos por empresas com a inovação da computação e digitalização. Estas enormes quantidades de dados geridos por empresas são compostos por informações sobre inventário, trabalhadores, clientes, vendas, salários, etc., e são dados importantes em que alguns têm de ser arquivados e que contêm informações úteis que podem ser usadas para monitorização de gastos e lucros e para estudo de mercado e possivelmente usados para minimizar os gastos e aumentar os rendimentos.

Antes da grande evolução dos computadores todos os registos que existiam era em papel até que os computadores começaram a ser acessíveis a toda a gente, sendo que hoje em dia o formato digital é o que mais prevalece. Sendo assim, cada vez menos são viáveis os métodos usados antigamente pelas empresas para registos tais como papel ou tabelas básicas em formatos tipo Microsoft Excel devido ao grande volume de dados gerado constantemente que continua a aumentar com o tempo, devido também à quantidade de tempo necessário para manter todos os registos atualizados e também à quantidade de normas e procedimentos pela qual uma empresa tem que se reger tendo em conta as normas fiscais em vigor e que podem prejudicar gravemente a empresa caso não sejam cumpridas minuciosamente.

Existem ainda empresas, tipicamente pequenas, que nos dias que correm ainda não possuem qualquer tipo de sistema informático para gestão financeira como uma Instituição Privada de Serviço Social (IPSS) que tem particular interesse neste trabalho e planeia utilizar a solução final que vai ser implementada e configurada para resolver todos estes problemas associados à falta de organização de dados e facilitar assim o seu dia-a-dia.

2 Levantamento e análise dos Requisitos

De forma a ser entendida por todos os terceiros que que possam vir a ler este trabalho ou potenciais interessados na solução, dado que será *open source*, a descrição dos requisitos está definida em formato textual e separada em requisitos funcionais e não -funcionais.

Os requisitos funcionais são os seguintes:

- I. O sistema deve ter um sistema de utilizadores para login e recusar o acesso a utilizadores não existentes ou com credenciais de entrada erradas.
- II. O sistema deve ter uma página, acessível apenas a administradores do sistema, onde podem criar utilizadores e adicionar, editar ou remover permissões a eles atribuídas.
- III. O sistema deve permitir o *upload* de ficheiros SAF-T em formato “.xml”.
- IV. O sistema deve importar todos os produtos, clientes e faturas descritos nos ficheiros SAF-T carregados para a base de dados do sistema.
- V. O sistema deve enviar automaticamente um e-mail ao usuário que fizer o upload de um ficheiro SAF-T quando o processo estiver concluído.
- VI. O sistema deve permitir adicionar, editar e remover itens.
- VII. O sistema deve permitir visualizar, adicionar, editar e remover clientes através de uma tabela e pesquisar por nome ou Número de Identificação Fiscal.
- VIII. O sistema deve permitir visualizar, adicionar, editar e remover faturas através de uma tabela e pesquisar por número de fatura ou data.
- IX. O sistema deve permitir visualizar, adicionar, editar e remover armazéns através de uma tabela e pesquisar por nome, descrição ou morada.
- X. O sistema deve permitir visualizar, adicionar, editar e remover fornecedores através de uma tabela e pesquisar por nome ou Número de Identificação Fiscal.
- XI. O sistema deve permitir visualizar, adicionar, editar e remover encomendas através de uma tabela e pesquisar por item ou fornecedor.
- XII. O sistema deve permitir visualizar, adicionar, editar e remover pagamentos através de uma tabela e pesquisar por descrição ou data.
- XIII. O sistema deve permitir visualizar, adicionar, editar e remover funcionários através de uma tabela e pesquisar por nome ou Número de Identificação Fiscal.
- XIV. O sistema deve permitir visualizar, adicionar, editar e remover cargos através de uma tabela e pesquisar por nome.
- XV. O sistema deve permitir visualizar, adicionar, editar, remover e marcar como completadas tarefas e atribuí-las a um usuário com um prazo, através de uma tabela e pesquisar por título ou nome de usuário.
- XVI. O sistema deve permitir visualizar, adicionar, editar e remover eventos através do calendário.
- XVII. O sistema deve permitir trocar mensagens entre usuários, e visualizá-las.
- XVIII. O sistema deve permitir enviar um e-mail para todos os clientes, e apresentar o histórico numa tabela.

- XIX. O sistema deve possuir um sistema de inventário, atualizar automaticamente de acordo com vendas e encomendas no sistema e permitir visualizar através de uma tabela e pesquisar por código de produto, descrição ou fornecedor.
- XX. O sistema deve ter um painel de informações onde possa ser possível visualizar as vendas, os gastos e os ganhos no ano presente, de forma gráfica e textual.

Os requisitos não-funcionais são os seguintes:

- I. O sistema deve ser uma plataforma web, acedida através de um browser, possibilitando a sua utilização através de qualquer dispositivo Windows, Linux, Mac OS, Android e iOS.
- II. O sistema deve ter um layout responsivo de forma a poder ser visualizado em dispositivos com ecrãs com dimensões pequenas.
- III. O sistema deve ser desenvolvido usando a framework Django fazendo uso das linguagens Python, HTML, CSS e Javascript.
- IV. O sistema deve estar em língua portuguesa.

3 Viabilidade e Pertinência

Existem vários problemas que foram identificados anteriormente que podem facilmente ser resolvidos pela implementação de uma plataforma de gestão financeira para controlo de custos empresariais com a capacidade do sistema de arquivar registos, de automatizar processos, criar relatórios e fazer a análise e monitorização da faturação, sendo aqui as possibilidades infinitas sendo que o sistema pode e deve ser ajustado de acordo com as necessidades de cada utilizador, neste caso empresas.

Tendo em conta o avanço da tecnologia e constante mudança do mundo empresarial vai sempre haver uma necessidade de constante atualização dos sistemas usados, seja pelas necessidades da empresa, por alteração de legislações, pelo avanço da tecnologia que pode requerer que o sistema seja atualizado por questões de segurança por exemplo, havendo assim possibilidade para continuamento deste trabalho no futuro.

“90% das Pequenas e Médias Empresas (PMEs) portuguesas aumentaram o uso de ferramentas digitais durante a pandemia” – conclui um estudo elaborado pelo ‘Digitally Driven’ em conjunto com a Google com base em um inquérito de investigação de 500 PMEs portuguesas - realizado entre 16 e 21 de novembro de 2020.

Existem várias plataformas que incorporam algumas destas funcionalidades mencionadas acima e são tipicamente denominadas por aplicações *Customer Relationship Management* (CRM). No entanto, a maioria das funcionalidades mais interessantes destas plataformas estão apenas disponíveis em versões pagas e tipicamente em língua inglesa, como pode ser visto mais abaixo no capítulo [Benchmarking](#), e nem todas as PMEs portuguesas têm possibilidade de fazer esse investimento, entrando aqui o fator custo.

“Um estudo mundial da IDC revela que as pequenas e médias empresas que já adotaram o digital viram a sua receita aumentar. Os softwares de CRM e de análise surgem em segundo lugar na lista dos mais utilizados, com pelo menos 38 e 37 por cento das PME em todas as regiões a utilizar a respetiva tecnologia.” – conclui um estudo da IDC, que analisou os dados de 3.210 inquiridos de 11 países, que trabalham em empresas com 10 a 999 colaboradores, em 2016.

4 Solução desenvolvida

A solução desenvolvida consiste numa plataforma de gestão financeira para controlo de custos empresariais num servidor web com capacidade para vários utilizadores com diferentes permissões dentro da mesma empresa, sendo que o software está desenvolvido para ser usado por membros da direção, departamento financeiro, vendas, apoio ao cliente ou marketing.

Para implementar esta plataforma foi escolhida a tecnologia [Django](#) que é uma framework web server-side em Python que permite o desenvolvimento de websites que utiliza uma arquitetura MVT (Model View Template) que os alunos de LEI já se encontram familiarizados devido à UC de Programação Web, onde esta mesma tecnologia foi lecionada. Foi escolhida esta mesma tecnologia dado que é um *framework* em Python com altas capacidades de *backend* e é uma linguagem moderna de alto nível estável com inúmeros módulos já desenvolvidos disponíveis que possibilitam desenvolver websites seguros e de fácil manutenção.

4.1 Estrutura de dados

A estrutura de dados pensada para a implementação da solução usando os Modelos do Django que vise conseguir responder às necessidades impostas encontra-se representada abaixo em formato diagrama, extraída a partir da base de dados construída para responder aos problemas impostos no capítulo anterior, onde foram usados conhecimentos aprendidos na UC de Bases de Dados e Engenharia de Software. Vão ser usados esses mesmos dados para produzir dados acerca de gastos, vendas e ganhos ao longo dos meses que podem ser visualizados através de gráficos, fazendo uso do plugin Charts.js.

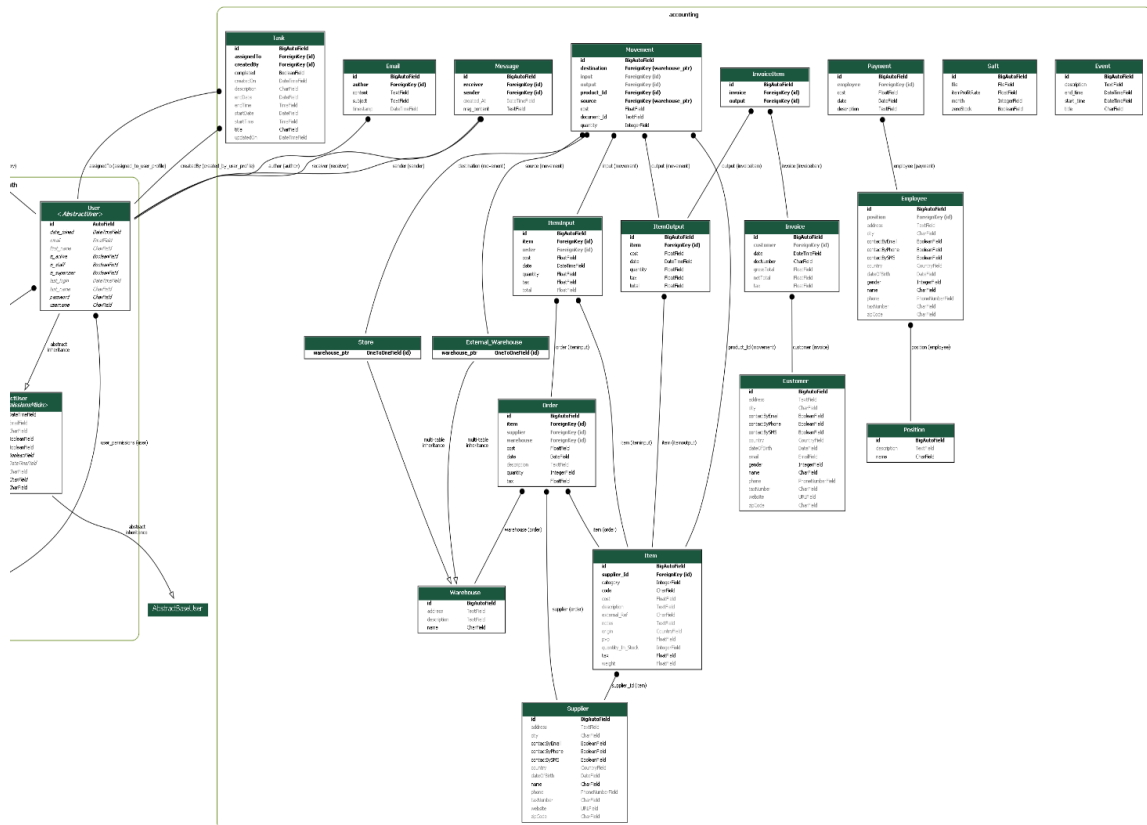


Figura 1 - Diagrama de Entidades-Relações da Solução

De forma a termos uma lista de inventário que fosse capaz de calcular o inventário dinamicamente ao invés de guardarmos esse valor numa variável o que é mais propenso a haver incongruências, criámos as duas classes representadas abaixo, sendo que ItemInput é referente a compras de itens a fornecedores, e ItemOutput é referente a vendas de itens a clientes. O cálculo do valor de unidades em stock fazendo um simples cálculo (n° de itens comprados – n° de itens vendidos). Estas classes estão descritas na figura abaixo, para uma melhor compreensão.

```
class ItemInput(models.Model):
    date = models.DateTimeField(blank=False, null=False)
    item = models.ForeignKey("Item",
                            null=False,
                            blank=False,
                            on_delete=models.CASCADE)
    tax = models.FloatField(null=False, blank=False)
    cost = models.FloatField(null=False, blank=False)
    quantity = models.FloatField(null=False, blank=False)
    total = models.FloatField(null=True, blank=True)
    order = models.ForeignKey("Order",
                             null=True,
                             blank=True,
                             on_delete=models.CASCADE)

class ItemOutput(models.Model):
    date = models.DateTimeField(blank=False, null=False)
    item = models.ForeignKey("Item",
                            null=False,
                            blank=False,
                            on_delete=models.CASCADE)
    tax = models.FloatField(null=False, blank=False)
    cost = models.FloatField(null=False, blank=False)
    quantity = models.FloatField(null=False, blank=False)
    total = models.FloatField(null=False, blank=False)
```

Figura 2 - Excerto código models.py

4.2 Página Principal (Painel de Informações)

Para podermos visualizar os ganhos, gastos e vendas a partir dos dados carregados no sistema de uma forma gráfica, e os apresentarmos aos utilizadores, calculámos os dados necessários para popular os gráficos fazendo uso de *queries* disponíveis na *framework* de *backend* utilizada (Django) e utilizámos o plugin Charts.js

Os três gráficos pertencentes ao painel principal do sistema, representam as listas calculadas no excerto de código abaixo, respetivamente allMonthsGains, allMonthsExpenses e allMonthsSales.

```
allMonthsSales = []
allMonthsStockExpenses = []
allMonthsPaymentExpenses = []
# monthsGains = Sales - Expenses
allMonthsGains = []
# monthsExpenses = Cost of stock + payments (like salaries, rent,...)
allMonthsExpenses = []

for i in range(1, 13):
    sales = Invoice.objects.filter(date__year=year,
                                   date__month=i).aggregate(
        Sum('netTotal'))
    stock = ItemInput.objects.filter(date__year=year,
                                      date__month=i).aggregate(
        Sum('total'))
    payments = Payment.objects.filter(date__year=year,
                                       date__month=i).aggregate(
        Sum('cost'))
    totalSales = sales['netTotal__sum']
    totalStock = stock['total__sum']
    totalPayments = payments['cost__sum']
    if totalSales == None:
        totalSales = 0
    if totalStock == None:
        totalStock = 0
    if totalPayments == None:
        totalPayments = 0
    allMonthsSales.append(totalSales)
    allMonthsStockExpenses.append(totalStock)
    allMonthsPaymentExpenses.append(totalPayments)
    allMonthsGains.append(totalSales - totalStock - totalPayments)
    allMonthsExpenses.append(totalStock + totalPayments)
```

Figura 3 – Cálculo de Valores para população de gráficos de Painel de Informações

De forma a apresentar os restantes valores apresentados no painel de informações, fizemos uso dos valores já calculados e criámos também uma *query* para calcular os valores de vendas e ganhos anuais, apresentada também no excerto de código mais abaixo.

```

yearSales = Invoice.objects.filter(date__year=year).aggregate(
    Sum('netTotal'))
yearStockExpenses = ItemInput.objects.filter(
    date__year=year).aggregate(Sum('total'))
yearPaymentExpenses = Payment.objects.filter(
    date__year=year).aggregate(Sum('cost'))
ySales = yearSales['netTotal__sum']
yStock = yearStockExpenses['total__sum']
yPayments = yearPaymentExpenses['cost__sum']

if ySales == None:
    ySales = 0
if yStock == None:
    yStock = 0
if yPayments == None:
    yPayments = 0

# We use [month-1] because months are 1-12 and the list goes from 0-11
monthSales = allMonthsSales[month - 1]
monthGains = allMonthsSales[month - 1] - allMonthsStockExpenses[
    month - 1] - allMonthsPaymentExpenses[month - 1]
yearGains = ySales - yStock - yPayments

```

Figura 4 – Cálculo de Valores para população de Pannel de Informações

4.3 Página Armazéns, Clientes, Encomendas, Faturas, Funcionários, Fornecedores, Itens, Pagamentos e Cargos

Para podermos visualizar, adicionar, editar e remover os objetos referenciados no título deste subcapítulo utilizámos um *template* base, apenas alterando o título da página, o texto descrito no botão de adicionar e as colunas da tabela. Tirámos proveito do plugin Datatables.js que permite criar tabelas como a apresentada em baixo facilmente e permite pesquisar entre os dados introduzidos através da caixa de pesquisa.

4.4 Página Inventário

Para podermos calcular o inventário e apresentá-lo ao cliente voltamos a tirar proveito do plugin Datatables.js. Para popular o valor de quantidade em stock necessitamos de fazer os cálculos a partir de encomendas e vendas feitas, sendo o valor em stock um simples cálculo, a diferença entre o valor de itens comprados e vendidos.

Calculando esse valor para todos os itens disponíveis na base de dados sempre que é acedida a página de inventário, temos sempre um valor correto e atualizado, sendo ainda possível apresentar o valor a reaver tendo em conta o preço da última fatura emitida em que o item foi vendido e a quantidade de itens em stock calculada.

```
@login_required
@permission_required('accounting.view_item')
def InventoryList_view(request):
    current_user = request.user
    items = Item.objects.all()
    inbox = Message.objects.filter(receiver=current_user)

    # quantity in stock = count(itemInputs) - count(itemOutputs)
    for item in items:
        quantityInput = 0
        quantityOutput = 0
        itemInputs = ItemInput.objects.filter(item=item)
        itemOutputs = ItemOutput.objects.filter(item=item)
        for itemInput in itemInputs:
            quantityInput += itemInput.quantity
        for itemOutput in itemOutputs:
            quantityOutput += itemOutput.quantity
        item.quantity = quantityInput - quantityOutput

    context = {'items': items, 'inbox': inbox, 'collapse': 'Items'}

    return render(request=request,
                  template_name="inventoryList.html",
                  context=context)
```

Figura 5 – Excerto de Código da View do Inventário

4.5 Lista de e-mail de clientes

Desenvolvemos também uma página onde podemos enviar um e-mail automaticamente para todos os clientes, desta forma podemos informá-los acerca de promoções, alterações de termos, etc.

Conseguimos alcançar isto facilmente iterando pela lista de clientes e acedendo à sua variável email, no entanto temos uma verificação por fazer, apenas enviamos o e-mail a clientes que queiram ser contactados por e-mail.


```
@login_required
@permission_required('accounting.add_email')
def EmailList_view(request):
    current_user = request.user
    inbox = Message.objects.filter(receiver=current_user)
    emails = Email.objects.all()

    form = EmailForm(request.POST or None,
                     request.FILES or None,
                     initial={'author': current_user})

    if form.is_valid():
        subject = form.cleaned_data['subject']
        content = form.cleaned_data['content']
        form.save()

        receivers = []
        for customer in Customer.objects.all():
            if customer.contactByEmail:
                receivers.append(customer.email)

        email = EmailMessage(subject,
                             content,
                             'deisi237@teste.pt',
                             receivers,
                             headers={'Reply-To': 'deisi237@teste.pt'})

        email.send(fail_silently=True)
        return redirect('EmailList')

    context = {'form': form, 'inbox': inbox, 'emails': emails}

    return render(request=request,
                  template_name="emailList.html",
                  context=context)
```

Figura 6 – Excerto de Código da View da Lista de E-mails

4.6 Tarefas, Calendário e Mensagens Privadas

Criámos também um sistema de tarefas de forma a utilizadores poderem atribuir tarefas entre eles. Os utilizadores também recebem um e-mail do sistema e uma mensagem dentro da plataforma, com as informações da tarefa de forma a serem informados que têm coisas a fazer.

Esta é a função que é chamada quando é criada uma tarefa nova, e que é responsável pela criação da mesma, da automatização do envio de um e-mail e mensagem para o utilizador ao qual a tarefa foi atribuída.

As tarefas são também filtradas, para apenas mostrarmos ao utilizador as tarefas que lhe interessam, as que criou ou que foi atribuído.

```

@login_required
@permission_required(['accounting.view_task'])
def TaskList_view(request):
    current_user = request.user
    inbox = Message.objects.filter(receiver=current_user)
    form = TaskForm(request.POST or None, request.FILES or None)
    tasks = Task.objects.filter(Q(createdBy=current_user) | Q(assignedTo=current_user))

    if form.is_valid() and current_user.has_perm('accounting.add_task'):
        post = form.save(commit=False)
        post.createdBy = request.user
        post.updatedOn = datetime.datetime.now()
        post.save()
        dateDeadline = form.cleaned_data['endDate']
        timeDeadline = form.cleaned_data['endTime']
        title = form.cleaned_data['title']
        sendTo = form.cleaned_data['assignedTo']
        email = EmailMessage(
            f'DEISI237 - {request.user} atribuiu-te uma tarefa: {title}',
            f'Olá, o utilizador {request.user} atribuiu-te uma tarefa com o prazo até {dateDeadline} às {timeDeadline} com o título: {title}',
            'deisi237@teste.pt', [sendTo.email],
            headers={'Reply-To': 'deisi237@teste.pt'})
        email.send(fail_silently=True)

        msg = Message()
        msg.sender = request.user
        msg.receiver = sendTo
        msg.msg_content = f'Mensagem Automática: Olá, o utilizador {request.user} atribuiu-te uma tarefa com o prazo até {dateDeadline} às {timeDeadline} com o título: {title}'
        msg.created_At = datetime.datetime.now()
        msg.save()

    return redirect('TaskList')

```

Figura 7 – Excerto de Código da View das Tarefas

Criámos também um calendário bastante simples para os utilizadores do sistema conseguirem manter-se a par de datas importantes, como feriados nacionais ou aniversário da empresa.

Foi também criado um sistema de mensagens interno de forma que os utilizadores do sistema consigam trocar mensagens entre si, e de forma a serem notificados quando têm tarefas atribuídas. O número de mensagens recebidas pode ser visto através do contador de mensagens, representado por um ícone de uma mensagem com o número por cima.

A implementação é bastante simples, criámos uma classe Mensagem, como representado abaixo:

```

class Message(models.Model):
    sender = models.ForeignKey(User,
                               null=False,
                               blank=False,
                               on_delete=models.CASCADE,
                               related_name="sender")
    receiver = models.ForeignKey(User,
                                 null=False,
                                 blank=False,
                                 on_delete=models.CASCADE,
                                 related_name="receiver")
    msg_content = models.TextField(null=False, blank=False)
    created_At = models.DateTimeField(editable=False, auto_now_add=True)

```

Figura 8 – Excerto de Código da Classe Mensagem

Para apresentar as mensagens ao utilizador apenas tivemos que fazer uma *query* simples:

```

@login_required
def MessageList_view(request):
    current_user = request.user
    inbox = Message.objects.filter(receiver=current_user)
    sentBox = Message.objects.filter(sender=current_user)

```

Figura 9 - Excerto de Código da View das Mensagens

5 Benchmarking

Existem várias aplicações e plataformas que incorporam algumas das funcionalidades mencionadas na Solução e são tipicamente denominadas por aplicações *Customer Relationship Management* (CRM), estando algumas das mais predominantes no mercado representadas na tabela abaixo com as suas funcionalidades comparativamente à nossa plataforma (DEISI237).

Note-se que o comparativo está a ser feito com as versões grátis do software sendo que algumas das funcionalidades referidas como não-presentes estão disponíveis em versões premium do software ou disponíveis à parte como extra, contrariamente ao nosso *software* que é *gratuito e open source*.

Tabela 1 – Comparação do DEISI237 com outras soluções gratuitas existentes no mercado

| | DEISI237 | HubSpot | Odoo | OroCRM | SugarCRM | SuiteCRM | Vtiger |
|--|----------|---------|------|--------|----------|----------|--------|
| Plataforma on-line | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Sistema de tarefas | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Calendário | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Visualização de ganhos e gastos num Painel | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ |
| Lista de envio de e-mails | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |
| Integração SAF-T (PT) | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Língua portuguesa | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| Open source | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |

<https://www.hubspot.com/products/crm>

<https://www.odoo.com/app/crm>

<https://suitecrm.com/>

<https://oroinc.com/orocrm/>

<https://www.sugarcrm.com/>

<https://www.vtiger.com/pt/all-in-one-crm/>

6 Método e planeamento

De seguida, ilustramos o calendário apresentado na entrega do 1º relatório, fazendo uma retrospeção as funcionalidades foram todas implementadas com sucesso.

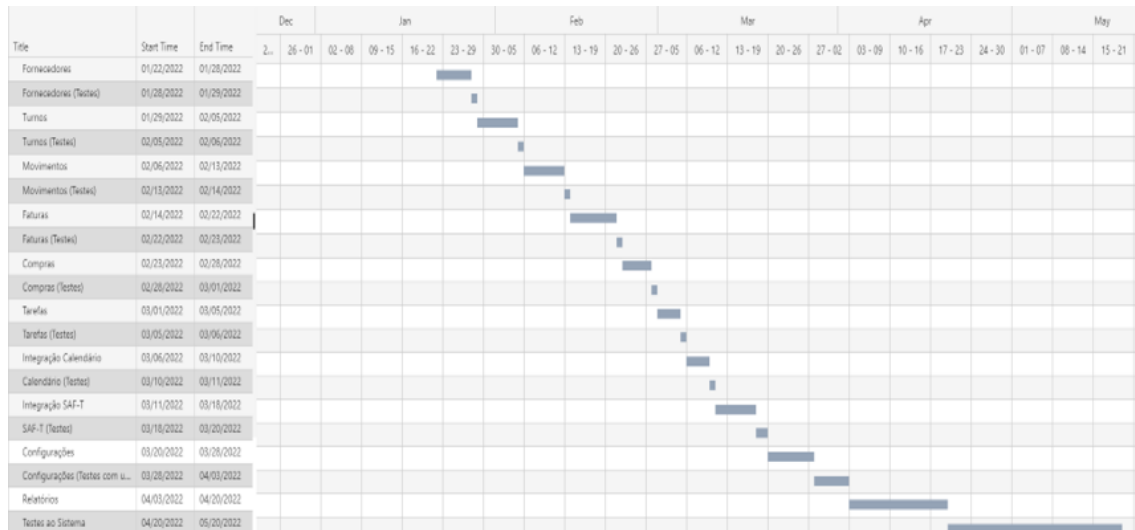


Figura 10 – Plano de Trabalho (Gantt) do 1º Relatório

De seguida, ilustramos o plano de trabalho apresentado no 2º Relatório, como foi referido anteriormente as funcionalidades foram todas implementadas com sucesso, tendo sido feita uma mudança no âmbito da visualização dos dados, tendo sido preferido um painel de informações ao invés de um relatório em formato PDF.

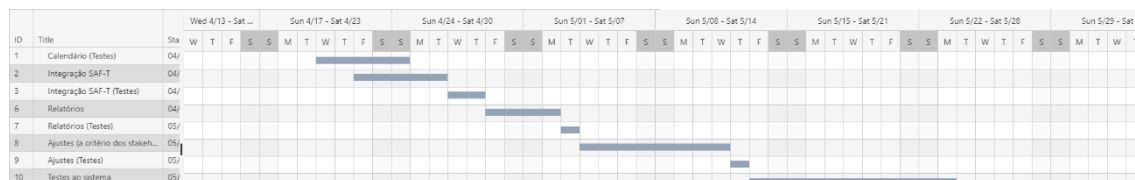


Figura 11 – Plano de Trabalho (Gantt) do 2º Relatório

7 Resultados

7.1 Testes

Abaixo estão descritos os testes definidos aquando da entrega do 2º relatório intercalar.

Tabela 2 - Testes e validação da solução

| ID | Descrição | Critério de validação |
|----|--|--|
| 1 | Iniciar sessão | O utilizador insere as suas credenciais de acesso e consegue aceder à aplicação. |
| 2 | Terminar sessão | O utilizador clica em 'Terminar sessão', confirma a sua escolha e é redirecionado para a página de login |
| 3 | Gerir utilizadores | O utilizador, caso seja administrador, acede à página de administrador da aplicação e adiciona/edita/remove utilizadores. |
| 4 | Gerir permissões | O utilizador, caso seja administrador, acede à página de administrador da aplicação e adiciona/edita/remove permissões de um utilizador ou de um grupo de utilizadores |
| 5 | Acesso a dados limitado por permissões | O utilizador não tem permissão para aceder à categoria ou ação que está a tentar aceder e recebe uma mensagem de erro |
| 6 | Listar objetos* | O utilizador clica numa das categorias do menu de navegação e caso tenha permissão para o fazer é mostrado uma tabela com os objetos todos dessa entidade |
| 7 | Adicionar novos objetos* | O utilizador clica em 'Adicionar', caso tenha permissão para o fazer, e adiciona novos objetos |
| 8 | Editar objetos* | O utilizador, clica numa linha da tabela que corresponde a um objeto e caso tenha permissão para o fazer edita esse objeto |
| 9 | Remover objetos* | O utilizador clica em 'Eliminar registo' ou 'Eliminar selecionados', caso tenha permissão para o fazer, confirma a sua escolha e remove os objetos selecionados |

| | | |
|----|---------------------------|---|
| 10 | Pesquisar objetos* | O utilizador clica na categoria correspondente ao tipo de objeto que quer encontrar, caso tenha permissão para o fazer, usa a barra de pesquisa com uma palavra-chave e são apresentados os registos encontrados |
| 11 | Importação ficheiro SAF-T | O utilizador clica em 'Importar SAF-T' e caso tenha permissão para o fazer, pode fazer o upload do ficheiro e os dados são inseridos na base de dados, atualizando o painel de informações e os gráficos nele contidos. |

* - Com objetos entende-se uma das entidades descritas no modelo de dados (ex: clientes, fornecedores, armazéns, etc)

7.2 Página Inicial

Tirando proveito do plugin Charts fizemos a representação dos ganhos, gastos e vendas calculados pelo servidor em gráficos separados, sendo possível visualizar o valor para cada mês passando o cursor por cima do gráfico.

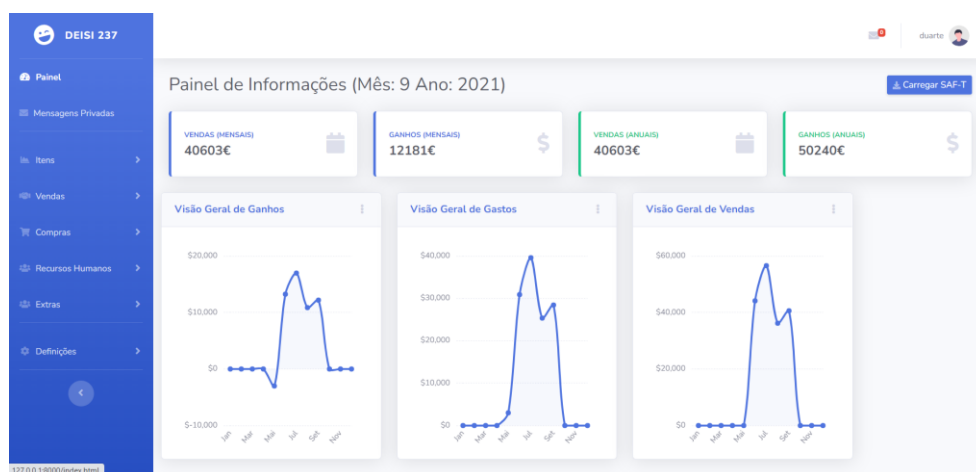
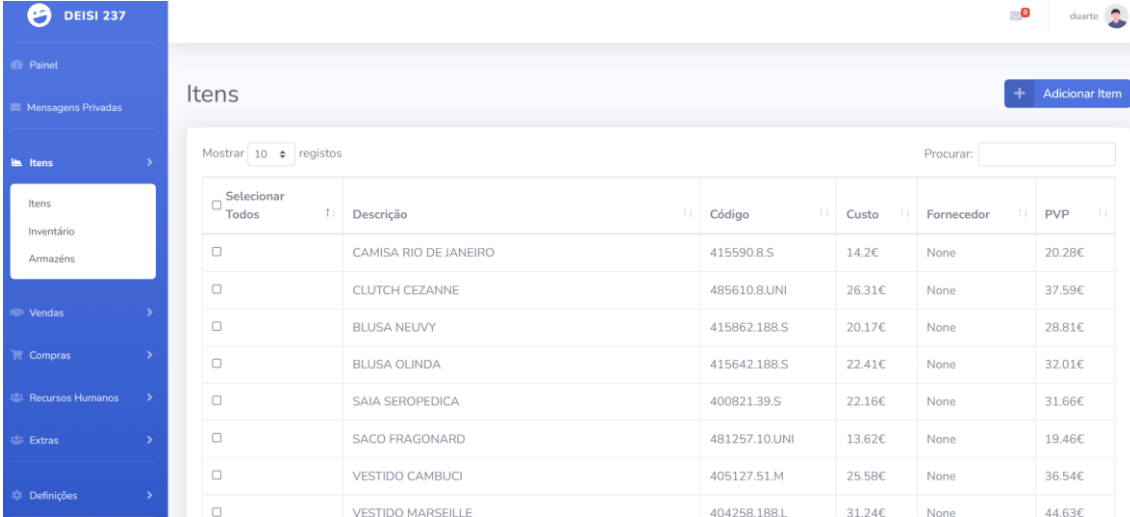


Figura 12 – Página Inicial do DEISI 237

7.3 Página Armazéns, Clientes, Encomendas, Faturas, Funcionários, Fornecedores, Itens, Pagamentos e Cargos

Acedendo a uma destas páginas temos todos os objetos respetivos disponíveis na base de dados listados numa tabela, fazendo uso do plugin Datatables.js possibilitando a sua ordenação e filtragem rápida. Podemos também entrar numa página que nos possibilita editar os objetos, clicando diretamente na linha do objeto pretendido da tabela. Temos também a

possibilidade de adicionar um novo objeto, através de um botão no canto superior direito da página, sendo então apresentado ao utilizador um formulário de forma a inserir os dados necessários.



DEISI 237

Painel

Mensagens Privadas

Itens

Itens

Inventário

Armazéns

Vendas

Compras

Recursos Humanos

Extras

Definições

Itens

Mostrar 10 registos

Procurar:

| <input type="checkbox"/> Selecionar Todos | Descrição | Código | Custo | Fornecedor | PVP |
|---|-----------------------|---------------|--------|------------|--------|
| <input type="checkbox"/> | CAMISA RIO DE JANEIRO | 415590.8.S | 14.2€ | None | 20.28€ |
| <input type="checkbox"/> | CLUTCH CEZANNE | 485610.8.UNI | 26.31€ | None | 37.59€ |
| <input type="checkbox"/> | BLUSA NEUVY | 415862.188.S | 20.17€ | None | 28.81€ |
| <input type="checkbox"/> | BLUSA OLINDA | 415642.188.S | 22.41€ | None | 32.01€ |
| <input type="checkbox"/> | SAIA SEROPEDICA | 400821.39.S | 22.16€ | None | 31.66€ |
| <input type="checkbox"/> | SACO FRAGONARD | 481257.10.UNI | 13.62€ | None | 19.46€ |
| <input type="checkbox"/> | VESTIDO CAMBUCI | 405127.51.M | 25.58€ | None | 36.54€ |
| <input type="checkbox"/> | VESTIDO MARSEILLE | 404258.188.L | 31.24€ | None | 44.63€ |

+ Adicionar Item

Figura 13 – Página Item

7.4 Página Inventário

Acedendo a esta página temos todos os itens disponíveis na base de dados listados numa tabela, fazendo uso do plugin Datatables.js, possibilitando a sua ordenação e filtragem rápida. O valor da quantidade em stock é calculado sempre que é acedida a esta página, evitando assim incongruências.

Inventário

Mostrar 10 registos

Procurar:

| Item | Código | Fornecedor | Quantidade em Stock | Valor a reaver(€) |
|-------------------|----------------|------------|---------------------|-------------------|
| ALMOFADA ANGERS | 467702.188.UNI | None | 30 | 240€ |
| ALMOFADA ANVERS | 467700.188.UNI | None | 0 | 0€ |
| ALMOFADA ARCACHON | 467703.188.UNI | None | 0 | 0€ |
| ALMOFADA BEAUMONT | 467711.188.UNI | None | 0 | 0€ |
| ALMOFADA CRUZEIRO | 467706.188.UNI | None | 0 | 0€ |
| ALMOFADA MARIGNY | 467708.188.UNI | None | 0 | 0€ |
| ALMOFADA NEVERS | 467704.188.UNI | None | 0 | 0€ |
| ALMOFADA ROISSY | 467709.188.UNI | None | 0 | 0€ |
| ALMOFADA SULLY | 467705.188.UNI | None | 0 | 0€ |

Figura 14 – Página Inventário

7.5 Página Lista E-mail de Clientes

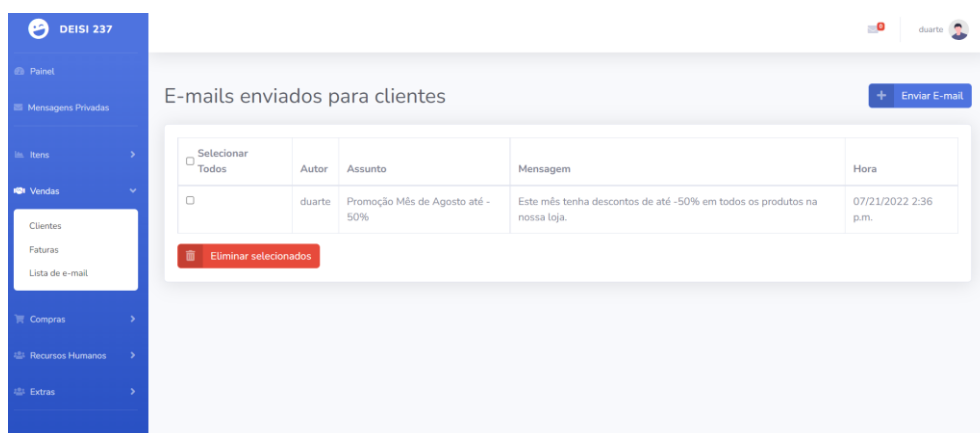


Figura 15 – Página Lista de E-mail de Clientes

Acedendo a esta página temos a funcionalidade de enviar um e-mail a todos os clientes que pretendam ser contactados através de endereço eletrónico, isto pode ser realizado inserindo um assunto e uma mensagem.

Enviar E-mail ×

Assunto:*

Conteúdo:*

Guardar

Figura 16 – Formulário de Envio de E-mail

7.6 Página Tarefas

Tarefas

Mostrar 10 registos

Procurar:

| <input type="checkbox"/> Selecionar Todos | Título | Atribuido por | Atribuido a | Data Prazo | Hora Prazo |
|---|--------------------------------|---------------|-------------|--------------|------------|
| <input type="checkbox"/> | Importar ficheiro SAF-T Agosto | duarte | duarte | Aug. 1, 2022 | 11:59 p.m. |

Mostrando os registos 1 a 1 num total de 1

[Anterior](#) **1** [Seguinte](#)

[+ Adicionar Tarefa](#)

[Eliminar seleccionados](#)

Figura 17 – Página Tarefas

7.7 Página Calendário

Calendário

[Anterior](#)

July 2022

| Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|-----|-----|-----|-----|--------------------------|-----|-----|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 • Feriado Nacional | 30 | 31 |

[Novo Evento](#) [Próximo](#)

Copyright © Universidade Lusófona de Humanidades e Tecnologia

Figura 18 – Página Calendário

7.8 Página Mensagens Privadas

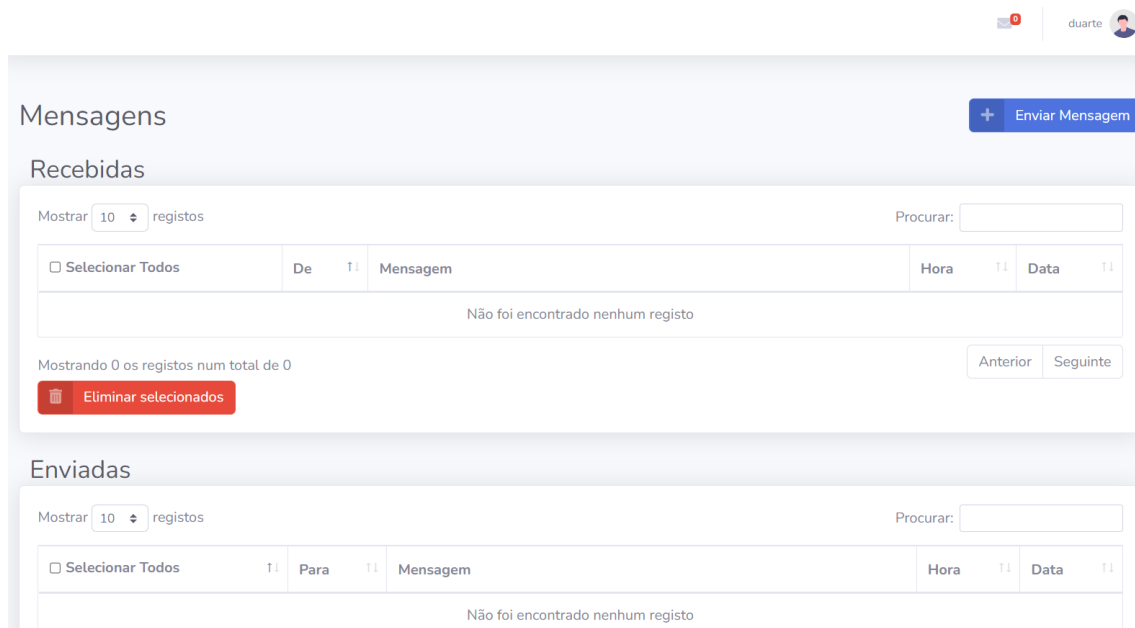


Figura 19 – Página Mensagens Privadas

Acedendo a esta página temos a funcionalidade de enviar uma mensagem privada interna para outro utilizador desta mesma plataforma, isto pode ser realizado inserindo um destinatário e uma mensagem.

8 Conclusão e trabalhos futuros

As plataformas de gestão financeira e controlo de custos empresariais podem incorporar inúmeras funcionalidades úteis e valiosas para os seus clientes, sendo as suas possibilidades infinitas. As ofertas disponíveis no mercado são bastantes e destacam-se da concorrência pelas suas funcionalidades e constante melhoramento. Esta plataforma apresenta bastante potencial comercial e tem bastante espaço para melhoramentos e continuação de desenvolvimento para trabalhos universitários, não se esgotando na data de conclusão deste trabalho.

Bibliografia

- [Django21] Django, <https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Django/Introduction>
- [DigDriv21] Digitally Driven, <https://digitallydriven.connectedcouncil.org/europe/wp-content/uploads/sites/2/2021/03/Digitally-Driven-Portugal.pdf>, acedido em Jan. 2022.
- [ItIns21] ITInsight, <https://www.itinsight.pt/news/digital/pmes-que-adotam-o-digital-crescem-mais>, acedido em Jan. 2022.
- [ULHT21] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em Out. 2021.
- [TaWe20] Tanenbaum,A. e Wetherall,D., *Computer Networks*, 6ª Edição, Prentice Hall, 2020.

Glossário

| | |
|-------|---|
| LEI | Licenciatura em Engenharia Informática |
| TFC | Trabalho Final de Curso |
| CRM | <i>Customer Relationship Management</i> |
| UC | Unidade Curricular |
| PMEs | Pequenas e Médias Empresas |
| SAF-T | <i>Standard Audit File for Tax purposes</i> |
| NIF | Número Identificação Fiscal |