IPG Politécnico da Guarda
Polytechnic of Guarda

# RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Pedro Filipe de Melo Pinto

novembro | 2018

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

# RELATÓRIO DE PROJETO EM CONTEXTO DE ESTÁGIO

---

# GESTOR DE FINANÇAS PESSOAIS

PEDRO FILIPE DE MELO PINTO

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

11/2019

# Agradecimentos

Em primeiro lugar agradecer ao supervisor Luís Baptista, da entidade onde fiz o meu projeto em contexto de estágio que sempre se disponibilizou para tirar dúvidas e me encaminhar da melhor forma possível. Por sua vez agradecer à entidade de acolhimento, Techisobar que se disponibilizou para a realização do meu projeto do curso de Engenharia Informática.

Agradecer ao meu orientador, Celestino Gonçalves pelo tempo perdido, pelas reuniões, pela dedicação e por sempre estar disponível para tudo o que eu precisasse e pela ajuda na elaboração deste relatório.

Agradecer também ao IPG – Instituto Politécnico da Guarda por oferecer está oportunidade a todos os seus alunos.

Por último e mais importante, agradeço aos pais, irmãs e amigos por todo o apoio, não só no projeto, mas ao longo de todo o percurso académico.

# Ficha de Identificação

**Nome do Estudante:** Pedro Filipe de Melo Pinto
**Número de Aluno:** 1012590
**Curso:** Engenharia Informática

**Nome da Entidade de Acolhimento:** Techisobar
**Morada:** Rua Gen. Póvoas 30| 6300-714 Guarda
**Telefone:** 933260927

**Duração do Projeto em Contexto de Estágio:**
**Data de Início:**14/09/2019
**Data de Fim:**13/11/2019

**Nome do Supervisor da Entidade de Acolhimento:** Luís Baptista
**Grau académico:** Licenciatura em Engenharia Informática

**Nome do Orientador:** Celestino Gonçalves
**Grau académico:** Mestre em Sistemas e Automação

# Resumo

Este relatório descreve todo o trabalho realizado no projeto em contexto de estágio, expõe a empresa acolhedora, a análise do desenvolvimento do projeto, a metodologia, automatismos e tecnologias usadas.

O projeto consiste na criação de uma aplicação *Salesforce* para gestão de finanças pessoais, lidando com as despesas e receitas. Essa aplicação deve conter diversos tipos de relatórios e gráficos para o utilizador poder tirar conclusões e controlar as suas finanças. Deve existir um alerta de valores a pagamento no caso de transações agendadas. A aplicação deve permitir a importação de ficheiros CSV de serviços do banco, tais como a *app*, ou *website* do mesmo.

**Palavras-chave**

Aplicação *Salesforce*, Gestão de Finanças Pessoais, Relatórios, Serviços do Banco.

# Abstract

This report describes all the work done in the project in context of internship, it shows the welcoming company, the analysis prior to project development, the methodologie, automatisms and used technologies.

The project consists in the creation of a *Salesforce* application for personal finance management, dealing with expenses and incomes. This application should contain various types of reports and graphs for the user to draw conclusions and control their finances. There must be a payment alert for scheduled transactions. The application must allow the import of CSV files from bank services, such as the bank's app or website.

**Keywords**

Bank Services, Personal Finance Manager, Reports, *Salesforce* Application.

# Índice Geral

# Índice de Figuras

# Índice de Tabelas

# Lista de Siglas e Acrónimos

**API**- *Application Programming Interface*

**CSS**- *Cascading Style Sheets*

**CSV**- *Comma Separated Values*

**CRM**- *Customer Relationship Management*

**ER**- Entidade Relacionamento

**HTML**- *Hyper Text Markup Language*

**IDE**- Ambiente de Desenvolvimento Integrado

**NA**- Não Aplicável

**PDF**- *Portable Document Format*

**SOQL**- *Salesforce Object Query Language*

**YNAB**-You Need A Budget

# 1 Introdução

O presente documento descreve o projeto realizado em contexto de estágio desenvolvido pelo aluno Pedro Filipe de Melo Pinto, no âmbito da unidade curricular de Projeto de Informática do terceiro ano do curso de engenharia informática que visa à conclusão do curso.

## 1.1 Caraterização sumária da instituição

A empresa onde foi realizado o projeto em contexto de estágio foi a Techisobar, anteriormente conhecida por Blue-Inifinity, empresa Unipessoal, LDA, situada em Rua Gen. Póvoas 30. Dedicando-se à consultoria de negócios e atividades nas áreas de informática de negócios, escritório, organização, gestão, informação e marketing, bem como os serviços com as mesmas relacionados.

## 1.2 Motivação

Nos dias de hoje é de enorme utilidade e uma prioridade as pessoas terem controlo das suas despesas e dos seus ganhos. Esta aplicação visa ajudar nesse aspeto.

A nível pessoal surgiu o interesse pela tecnologia *Salesforce* [1], devido à opinião de colegas que realizaram estágios semelhantes, e como era uma tecnologia em que não havia conhecimento pessoal foi uma razão da escolha. A escolha do tema resulta da escolha de uma lista de projetos propostos em *Salesforce* na empresa Techisobar onde o projeto que despertou mais interesse foi o Gestor de Finanças Pessoal.

## 1.3 Descrição do problema

O problema, deve-se ao facto de as pessoas não terem um controlo financeiro e não terem uma maneira de saber como andam as suas finanças de forma simples. No ponto seguinte são referidos os objetivos da aplicação.

## 1.4 Objetivos

Os objetivos da aplicação Gestor de Finanças Pessoais a Desenvolver no *Salesforce* são os seguintes:

- Gestão financeira completa
- Programação de receitas/despesas recorrentes
- Alertas/Notificação de valores a pagamento
- Categorias multinível

- Possibilidade de associar ficheiros às transações
- Importação/Exportação de dados CSV (*Homebanking*)
- Relatórios diversos (dia/mês/ano, por tipo de despesas, por categoria, etc)
- Gerar PDF dos relatórios
- Sistemas de Pesquisas / Filtragem

## 1.5 Plano de estágio

As etapas do estágio foram as seguintes:

- Realização dos *Trails* [1]sugeridos pelo supervisor
- Análise dos requisitos do sistema
- Implementação da aplicação em *Salesforce*
- Publicação e testes

## 1.6 Estrutura do documento

O relatório é constituído por oito capítulos e está estruturado do modo seguinte: neste capítulo, foi feita uma breve introdução do contexto do projeto e os objetivos previstos. No capítulo dois são apresentadas plataformas que se enquadram no tema do projeto e foram comparadas com os objetivos da aplicação *Salesforce* a criar. No capítulo três é definida a metodologia a usar ao longo do projeto. No capítulo quatro é descrita toda a análise de requisitos. No capítulo cinco são apresentadas as tecnologias usadas na implementação do projeto. No capítulo seis é descrita a implementação da aplicação, sendo ilustrados alguns exemplos de *layouts* da mesma para melhor compreensão. No capítulo sete são expostos os testes para verificação de validações. Por fim, no último capítulo, encontram-se as conclusões do projeto, melhorias e trabalho futuro a desenvolver na aplicação.

[1] Trails: Módulos de aprendizagem guiados que ajudam a estudar uma tecnologia no menor tempo possível.

# 2 Estado da Arte

Depois de definidos os objetivos propostos para o desenvolvimento do projeto foi necessário realizar uma pesquisa para o levantamento de soluções existentes.

Após a realização dessa pesquisa e do fornecimento de alguns exemplos por parte do supervisor, realizou-se uma comparação das funcionalidades das diferentes plataformas.

A aplicação Personal Finance Manager é a desenvolvida pelo estudante. Tendo em conta os requisitos do projeto, sendo essa coluna uma representação dos objetivos do projeto, foi realizada uma comparação a outras aplicações: YNAB[2], Buxfer[3] e Controle Finance[4] que são três exemplos de aplicações de gestão de finanças. O Buxfer e o Controle Finance foram fornecidas pelo supervisor, e YNAB escolhida pelo estudante tendo algumas diferenças entre elas sendo essas comparações apresentadas na Tabela 1.

Feita uma análise às aplicações em comparação à aplicação *Salesforce* conseguimos perceber que a YNAB e o Buxfer não têm um sistema de alertas de despesas por pagar, nem podem gerar um relatório de PDF que são dois requisitos importantes para o projeto. A nível de associar um ficheiro a uma transação o YNAB e o Controle Finance não o permitem. O YNAB não tem um sistema de pesquisas, sendo a aplicação com mais falhas. No entanto, quase todas as aplicações cumprem a maioria dos requisitos propostos, sendo a aplicação que reúne todos os requisitos a Personal Finance Manager. Avançamos assim para o desenvolvimento do projeto com os requisitos necessários, com a configuração que se pretende usando a metodologia escolhida para o desenvolvimento deste projeto, que se encontra no capítulo seguinte.

*Tabela 1 Comparação de funcionalidades concorrentes*

| | **Aplicação** | | | |
|---|---|---|---|---|
| **Características** | Personal Finance Manager(Objetivos) | YNAB | Buxfer | Controle Finance |
| Gestão financeira completa | ✓ | ✓ | ✓ | ✓ |
| Alertas de receitas/despesas recorrentes | ✓ | X | X | ✓ |
| Agendar transações | ✓ | ✓ | ✓ | ✓ |
| Categorias multinível | ✓ | ✓ | ✓ | ✓ |
| Associar ficheiros às transações | ✓ | X | ✓ | X |
| Importar dados do homebanking | ✓ | ✓ | ✓ | ✓ |
| Importação de dados CSV | ✓ | ✓ | ✓ | ✓ |
| Relatórios diversos | ✓ | ✓ | ✓ | ✓ |
| Gerar PDF dos relatórios | ✓ | X | X | ✓ |
| Sistema de pesquisas/ filtragem | ✓ | X | ✓ | ✓ |
| Website responsivo | ✓ | ✓ | ✓ | ✓ |
| Plataformas | Cloud(web) | Cloud(web) | Cloud(web) | Cloud(web) |
| Plano gratuito | NA | 34 dias | Demo | Free |
| Preço Inicial | NA | 83.99 $ por ano ou 6.99$ por mês | 9.99$ por mês | NA |

# 3 Metodologia

## 3.1 Formação *Salesforce*

Para este projeto houve a necessidade de adquirir conhecimentos na tecnologia *salesforce*. Para tal, usou-se uma plataforma de aprendizagem online intitulada *Trailhead,* plataforma que foi sugerida pelo supervisor da empresa.

Esta plataforma é uma das melhores maneiras para aprender *salesforce*, pois a plataforma atribui um nível ao utilizador que é maior quanto mais módulos tiverem sido terminados. Um *Trail* contém vários módulos onde cada um tem como recompensa um emblema.

Os *trails* são caminhos de aprendizagem guiados e muito bem estruturados onde o utilizador aprende um tema e é testado no final, ganhando pontos, ou não, para a subida de nível.

Na Figura 1 encontra-se o perfil do aluno no *Trailhead*[5] onde mostra as conquistas, emblemas e competências adquiridas ao longo do percurso.

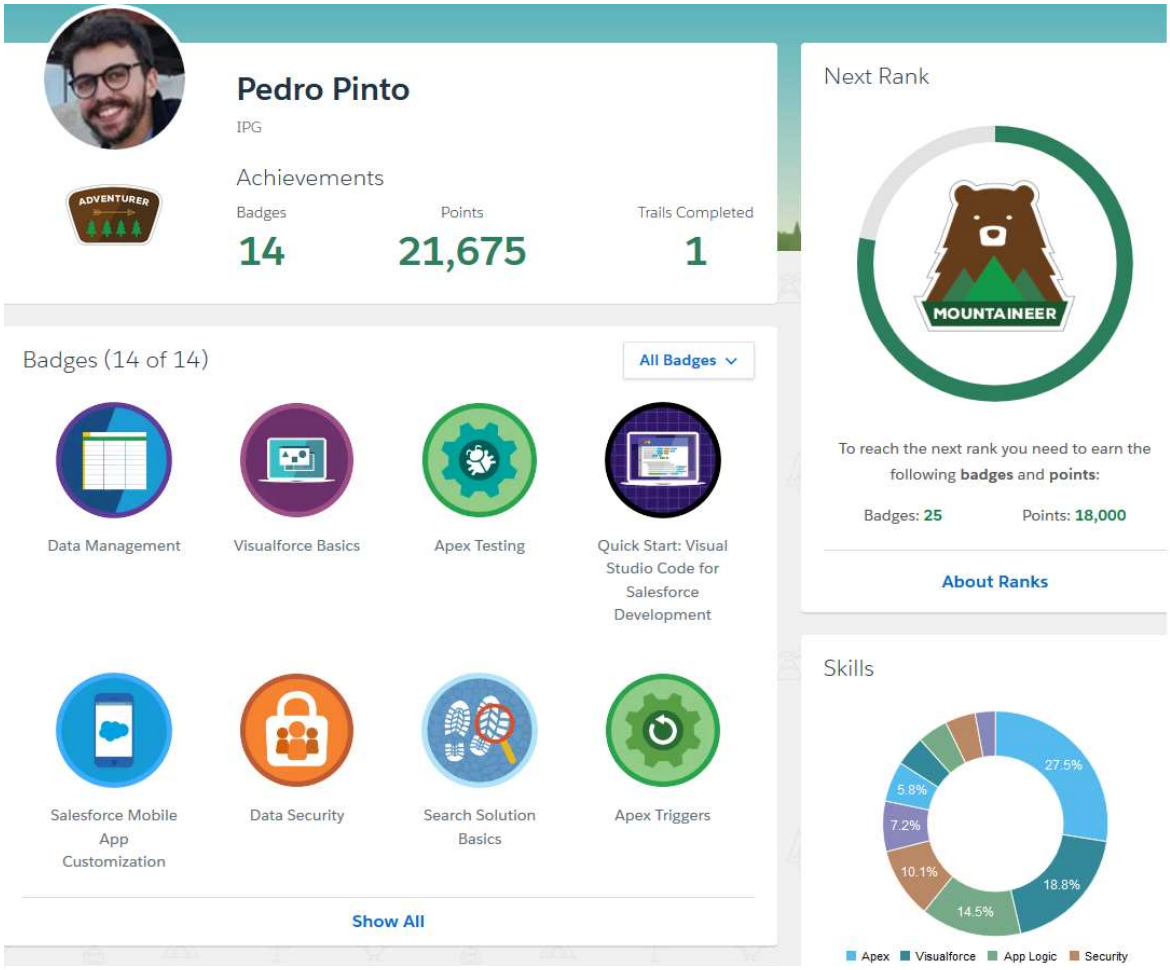



*Figura 1 – Perfil do aluno na plataforma Trailhead*

## 3.2 Metodologia *Extreme Programming*

Na criação da aplicação utilizou-se o desenvolvimento de *software* com metodologia ágil[6]. Esta metodologia tem alguns conceitos importantes tais como *software* funcional em vez de documentação e responder a mudanças em vez de seguir um plano ou seja ao longo do projeto melhorar e fazer da forma que se achar adequada na altura de maneira a obter o melhor resultado possível, o que se adequa ao *Salesforce,* e a este projeto especificamente.

Dentro desta metodologia escolheu-se o *Extreme Programming*[6] que é reservado a equipas pequenas e médias (neste caso o estudante e o supervisor são a equipa), onde se desenvolvem funcionalidades pequenas que rapidamente podem ser modificadas. No *Extreme Programming* existe f*eedback* constante, havendo comunicação entre as pessoas envolvidas, e uma abordagem incremental, ou seja, são criadas pequenas funcionalidades que são testadas e onde há comunicação com o supervisor, da qual surgem novas ideias e se testa a aplicação.

Houve pequenas reuniões, frequentes, com o supervisor ao longo do projeto onde foram mostradas as funcionalidades já implementadas e onde se comprovou se estava a funcionar, com testes, e demonstrações. Havendo um feedback e surgindo formas melhores de implementar e/ou melhorar, comprovando assim o uso desta metodologia.

Numa primeira reunião houve a discussão do modelo ER, tendo o estudante apresentado um modelo possível e onde o supervisor demonstrou a sua opinião e possíveis adições e mostrou alguns defeitos, levando à criação de um bom modelo.

Noutras reuniões o estudante apresentava já implementações de nível baixo onde o supervisor sugeriu novas alterações e ideias para as fases seguintes, havendo sempre uma melhoria no geral do trabalho desenvolvido.

Havia também a troca de emails a nível de dúvidas onde o supervisor fornecia documentação e opiniões acerca da melhor maneira de resolver os problemas, sendo úteis para o estudo e posterior implementação na aplicação.

# 4 Análise de Requisitos

Os requisitos de um sistema consistem numa descrição estruturada do procedimento que é esperado antes de ser desenhado, implementado e testado, sendo um aspeto comum e integrante de qualquer processo de desenvolvimento de produto.

O estudante depois da escolha do projeto reuniu-se com o supervisor, onde foram explicadas algumas bases do *salesforce* e fornecidas documentações sobre o mesmo para o estudo da plataforma. Nessa mesma reunião foram explicados os objetivos do projeto de forma mais pormenorizada onde ao mesmo tempo foram expostas ideias sobre como desenvolver esses objetivos.

Dessa reunião concluiu-se que a base da aplicação são as transações (movimentos na conta bancária), sendo essas transações associadas a uma conta bancária, e a uma categoria. Categorias são nomes de onde são efetuadas as despesas ou receitas, tal como transportes, restaurantes, casa de forma a se poder saber onde o utilizador gasta mais dinheiro ou ganha através dos *reports*. *Reports* são gráficos e tabelas onde se expõe os gastos e ganhos de uma pessoa, as categorias mais e menos usadas, tudo o que ajude o utilizador a tirar conclusões das suas finanças. Uma conta bancária tem informações preenchidas pelo utilizador e é associada a um banco e a um utilizador. As transações podem ser também associadas a um ficheiro pois existe a possibilidade de associar um ficheiro à transação tal como um recibo.

## 4.1 Tabela de classes/objetos

Para analisar os requisitos a serem utilizados no projeto, pensou-se primeiro nas entidades que iria utilizar.

As entidades, mais conhecidas por tabelas, na linguagem *Salesforce* denominam-se objectos. No *Salesforce* existem objetos standard, ou seja, objetos que podem ser usados e alterados de acordo com o projeto a ser realizado. No entanto, neste projeto optou-se por criar todos os objetos necessários.

Esses objetos são expostos na Tabela 2, de seguida.

*Tabela 2- Objetos criados e respetiva descrição*

| Objectos | Descrição |
|---|---|
| Users | Utilizador, o beneficiário de uma conta bancária. |
| Accounts | Contas bancárias existentes, tendo vários tipos disponíveis. |
| Banks | Todos os bancos existentes, que podem ou não estar ativos. |
| Transactions | Transações existentes com todo o tipo de informação relevante para os utilizadores e para o sistema. |
| Categories | Categorias a que podem ser associadas as transações, identificando o tipo de despesas/receitas. |
| Files | Ficheiros que podem ser adicionados relativamente às transações. |

## 4.2 Diagrama de contexto

No diagrama da Figura 2 é apresentado o que um administrador pode fazer no sistema e o que um utilizador pode fazer assim como o que o sistema mostra ao mesmo.

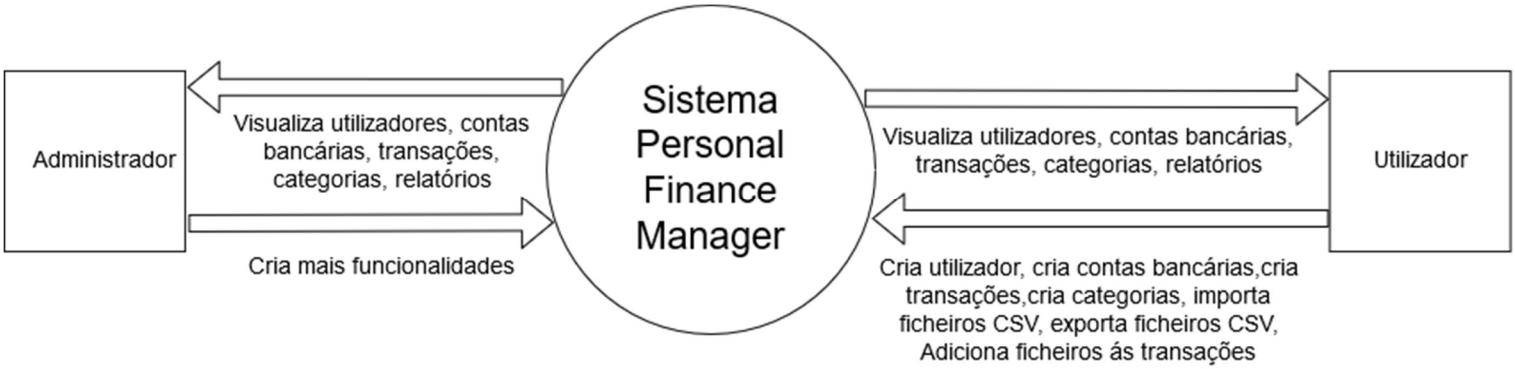



*Figura 2- Diagrama de Contexto*

## 4.3 Diagrama de casos de uso

Os diagramas de caso de uso permitem ver, de forma clara, todas as funcionalidades e interações que os atores vão ter com o sistema.

A Figura 3 mostra o diagrama de casos de uso, no qual é possível ver a fronteira que delimita os casos de uso e os seus respetivos atores.


*Figura 3- Diagrama de Casos de Uso*

## 4.4 Modelo Entidade Relacionamento (ER)

Sendo um modelo de dados, o modelo entidade relacionamento tem como objetivo descrever os dados e toda a informação necessária no projeto, sendo uma peça importante no que toca à construção da base de dados do mesmo. Nenhum projeto deve começar sem haver uma análise e criação do modelo entidade relacionamento. Como o nome indica, neste modelo existem duas componentes essenciais, as entidades que são as classes ou objetos como já foi referido e as relações que existem entre os mesmos. Os objetos têm também os atributos e as características referentes ao mesmo. Como exemplo, este projeto terá um objeto chamado "Accounts" com os atributos, nome, tipo de conta, saldo, entre outros, que por sua vez tem relações com outros objetos. O objeto "Accounts" tem uma relação de um para muitos com o objeto "Transactions" pois uma conta bancária pode ter um ou mais movimentos e um movimento só pode pertencer a uma e uma só conta.

Na Figura 4, nesta página encontra-se o modelo ER simplificado deste projeto.

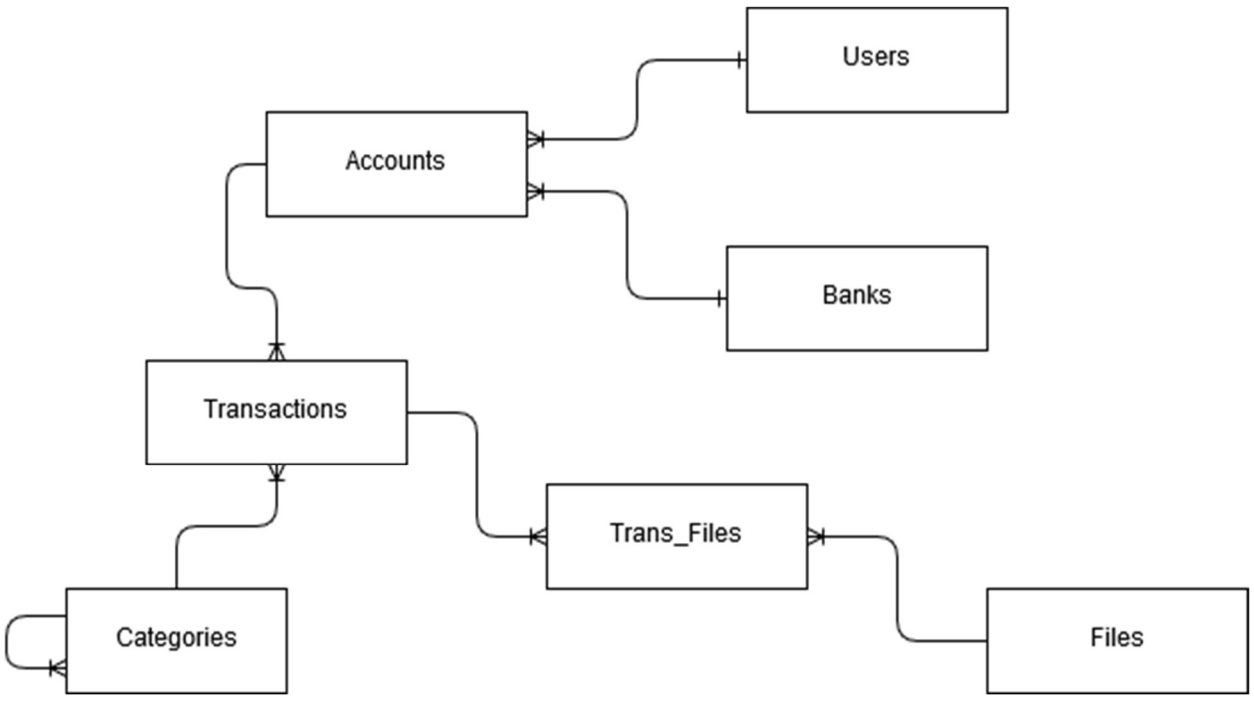



*Figura 4- Modelo ER simplificado*

Para além do modelo simplificado, foi criado um modelo mais complexo, onde estão identificados os atributos a serem usados em cada objeto. O modelo ER completo deste projeto pode ser consultado na Figura 5, nesta página.

Os campos "Category_Name, Bank_Name, Users_Name, Accounts_Name" são campos do tipo formula (*read only*) que vão buscar o nome à tabela respetiva, são usados para a construção dos *reports*.

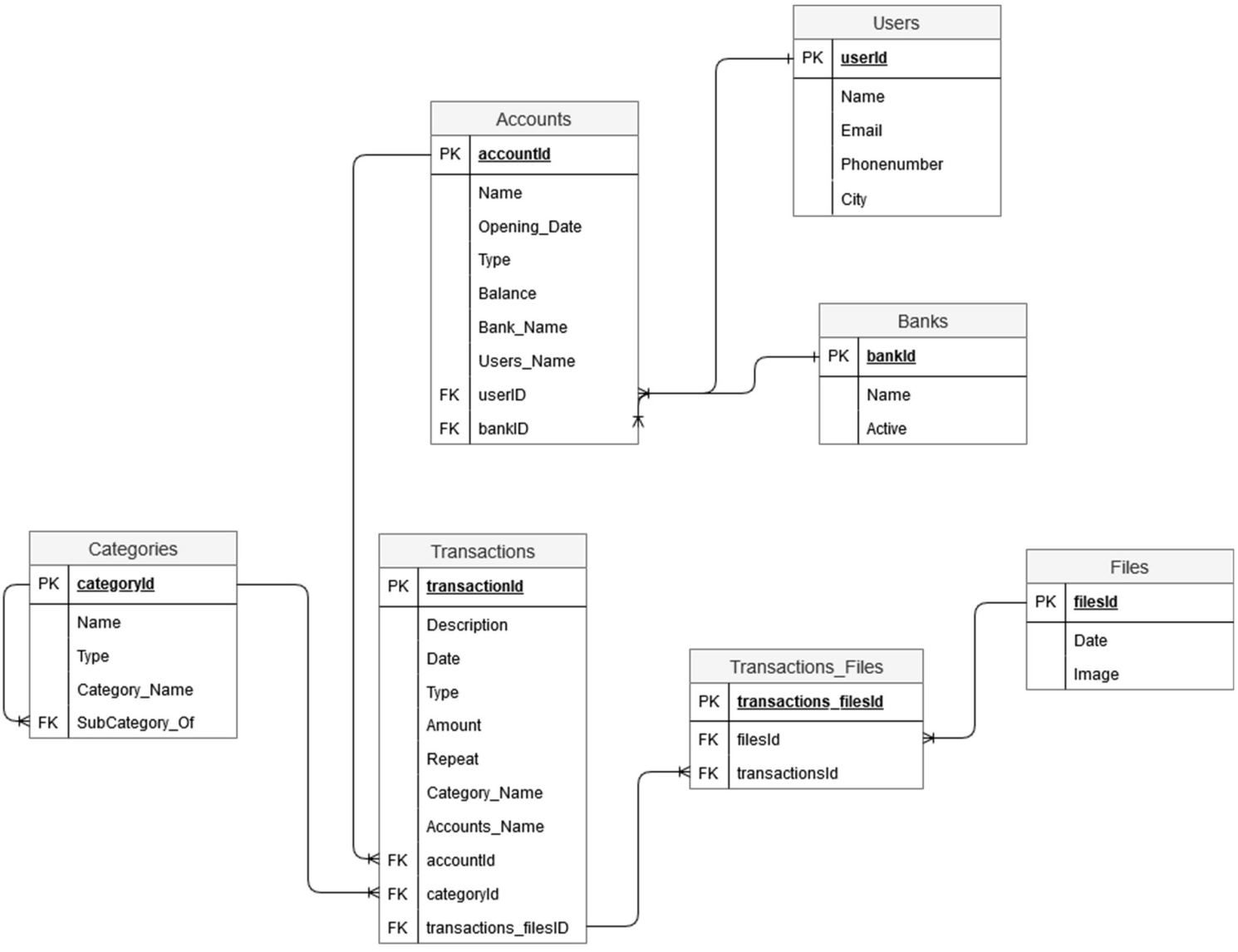



*Figura 5- Modelo ER completo*

## 4.5 Tabela de atores e respetivos casos de uso

A Tabela 3 seguinte apresenta os atores evolvidos e as respetivas ações na aplicação, de forma a entender-se a função de cada um dentro do projeto.

*Tabela 3- Atores e respetivos casos de uso*

| Ator | Função no projeto |
|---|---|
| Administrador<br>(Administrador do sistema) | Resolver erros/*bugs* da aplicação.<br>Fazer manutenção da aplicação.<br>Criar outras funcionalidades. |
| Utilizador | Criar utilizador.<br>Visualizar utilizadores.<br>Criar conta bancária.<br>Visualizar contas bancárias.<br>Criar transações.<br>Visualizar transações.<br>Criar categorias.<br>Visualizar categorias.<br>Importar ficheiro CSV (*homebanking*).<br>Exportar ficheiro CSV.<br>Adicionar ficheiros às transações. |

## 4.6 Casos de Uso

Foram selecionados dois casos de uso pela necessidade crítica dos mesmos. Estando o primeiro na Tabela 4 o caso de uso “Criar transações”, e na tabela 5 o caso de uso “Importar ficheiro CSV (homebanking)”.

*Tabela 4- Caso de Uso "Criar transações"*

| Criar transações | |
|---|---|
| **Descrição** | Permite aos utilizadores criar uma transação numa conta bancária. |
| **Pré-Condição** | Haver pelo menos uma conta bancária, e uma categoria. |
| **Caminho Principal** | 1.O utilizador escreve uma descrição da transação.<br>2.O utilizador seleciona uma conta bancária.<br>3.O utilizador escolhe uma data.<br>4.O utilizador escolhe o tipo de transação.<br>5.O utilizador escolhe a quantia de dinheiro.<br>6. O utilizador escolhe se quer repetir a transação futuramente. |
| **Caminho Alternativo** | 1a o número de caracteres é maior que o limite.<br>3a a data escolhida é anterior à data de abertura de conta. |
| **Suplementos** | Após a criação são mostrados os detalhes da transação. |

*Tabela 5- Caso de Uso "Importar ficheiro CSV (homebanking)"*

| Importar ficheiro CSV (homebanking) | |
|---|---|
| **Descrição** | Permite aos utilizadores carregar dados de ficheiros CSV exportados do homebanking. |
| **Pré-Condição** | Haver contas bancárias. |
| **Caminho Principal** | 1.O utilizador seleciona uma conta bancária.<br>2.O utilizador prepara o ficheiro csv de acordo com as especificações pedidas.<br>3.O utilizador escolhe o ficheiro.<br>4.O utilizador clica no botão import. |
| **Caminho Alternativo** | 2a o ficheiro não é bem configurado |
| **Suplementos** | Depois de importadas as transações aparecem organizadas no fundo da página |

## 4.7 Dicionário de dados

Após toda a análise feita anteriormente, o estudante focou-se na criação do dicionário de dados para todos os atributos e objetos existente no modelo ER.

O dicionário de dados tem por objetivo mostrar o tipo de dados, o significado de cada atributo e as restrições associadas. Com isto, futuros programadores deste projeto conseguem mais facilmente compreender o modelo da aplicação.

Apresenta-se nas secções seguintes os dicionários de dados de todas as classes do modelo ER do projeto.

### Objeto: Accounts

Na Tabela 6 encontra-se o dicionário de dados do objeto “Accounts”, que permite o acesso a toda a informação relevante sobre a conta bancária.

*Tabela 6- Dicionário de dados do objeto "Accounts"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **accountId** | Auto number | Números superiores a 1 | #N/A | Yes | Número sequencial que identifica univocamente cada conta bancária | Gerado pelo sistema/Não alterável Display Format: AC-{00000} |
| **Name__c** | Text(80) | #N/A | #N/A | Yes | Nome da conta | #N/A |
| **Opening_Date__c** | Date | #N/A | #N/A | Yes | Data em que foi aberta a conta | #N/A |
| **Type__c** | Picklist | Checking Account<br>Saving Account<br>Investment<br>Private Pension<br>Cash<br>Credit Card<br>Loans & Mortages<br>Asset<br>Other | #N/A | Yes | Tipos de conta disponíveis | #N/A |
| **Balance__c** | Number(16,2) | Número superiores a 0 | #N/A | Yes | Quantidade de dinheiro com que foi aberta a conta | #N/A |
| **Bank_Name__c** | Formula(text) | Banks__r.Name | #N/A | #N/A | Vai ao banco > nome buscar o nome do banco | Read Only |
| **Users_Name__c** | Formula(text) | Users__r.Name | #N/A | #N/A | Vai ao users > name buscar o nome do user | Read Only |
| **userId(FK)** | Lookup(Users) | #N/A | #N/A | #N/A | Ligação ao objeto Users | #N/A |
| **bankId(FK)** | Lookup(Banks) | #N/A | #N/A | #N/A | Ligação ao objeto Banks | #N/A |

### Objeto: Banks

Na Tabela 7 encontra-se o dicionário de dados do objeto “Banks”. O objeto “Banks” existe para ser associado a uma conta bancária, ajudando no processo de importação de dados do *homebanking*.

*Tabela 7- Dicionário de dados do objeto "Banks"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **bankId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada banco | Gerado pelo sistema/Não alterável Display Format: BK-{00000} |
| **Name__c** | Text(80) | #N/A | #N/A | Yes | Nome do banco | #N/A |
| **Active__c** | Checkbox | #N/A | #N/A | #N/A | Se está ativo | #N/A |

## Objeto: Categories

Na Tabela 8 encontra-se o dicionário de dados do objeto "Categories". As categorias servem para saber do que se trata as transações nas contas, fornecendo informações importantes para os *reports* da aplicação.

*Tabela 8- Dicionário de dados do objeto "Categories"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **categoryId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada categoria | Gerado pelo sistema/Não alterável Display Format: CG-{00000} |
| **Name__c** | Text(80= | #N/A | #N/A | Yes | Nome da Categoria | #N/A |
| **Type__c** | Picklist | Expenses Income | #N/A | Yes | Saber se a categoria é despesa ou receita | #N/A |
| **Subcategory_Name__c** | Formula | Subcategory_Of__r.Name | #N/A | #N/A | Nome da Sub categoria | #N/A |
| **SubCategory_OfId(FK)** | Lookup(Categories) | #N/A | #N/A | #N/A | Ligação à tabela Categories (self relation) | #N/A |

## Objeto: Files

Na Tabela 9 encontra-se o dicionário de dados do objeto “Files”. Os ficheiros são associados a transações se o utilizador adicionar um ficheiro.

*Tabela 9- Dicionário de dados do objeto "Files"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **filesId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada ficheiro | Gerado pelo sistema/Não alterável Display Format: FL-{00000] |
| **Date__c** | Date | #N/A | #N/A | #N/A | Data em que foi adicionado o ficheiro | #N/A |
| **Image__c** | Rich text Area(32768) | #N/A | #N/A | Yes | Imagem que pode ser um recibo ou confirmação de pagamento | #N/A |

## Objeto: Transactions

Na Tabela 10 encontra-se o dicionário de dados do objeto “Transactions”. Este objeto é o mais importante da aplicação sendo as transações importantes para os *reports* da aplicação.

*Tabela 10- Dicionário de dados do objeto "Transactions"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **transactionsId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada transação | Gerado pelo sistema/Não alterável Display Format: TR-{00000} |
| **Description__c** | Text(80) | #N/A | #N/A | Yes | Descrição da transação | #N/A |
| **Date__c** | Date | #N/A | #N/A | Yes | Data em que foi feita a transação | #N/A |
| **Transaction_Type__c** | Picklist | Expense Income | #N/A | Yes | Tipo de transação | Expense é um debito e Income é um crédito |
| **Amount__c** | Number(16,2) | #N/A | #N/A | Yes | Quantidade de dinheiro da transação | #N/At |
| **Balance__c** | Text(16,2) | #N/A | 0 | Yes | Quanto dinheiro tem na conta depois da transação | Este campo não precisa de ter o valor correto, esse valor vai ser calculado depois de ter as transações ordenadas por data |
| **Repeat__c** | Picklist | None Every Week Every Month Every Year | None | #N/A | Se o utilizador pretende que haja uma transação de pagamento semanal, mensal, ou anual | #N/A |
| **Category_Name__c** | Formula (Text) | Categories__r.Name | #N/A | #N/A | Vai buscar o nome da Categoria | #N/A |
| **Accounts_Name__c** | Formula (Text) | Accounts__r.Name | #N/A | #N/A | Vai buscar o nome da conta bancária | #N/A |
| **accountId(FK)** | Lookup(Accounts) | #N/A | #N/A | #N/A | Ligação às contas bancárias | #N/A |
| **categoryId(FK)** | Lookup(Caegories) | #N/A | #N/A | #N/A | Ligação às Categorias | #N/A |
| **Transactions_filesId(FK)** | Master-Detail(Transactions_files) | #N/A | #N/A | #N/A | Ligação à tabela intermédia que liga as transações aos ficheiros | #N/A |

### Objeto: Transactions_Files

Na Tabela 11 encontra-se o dicionário de dados do objeto “Transactions_Files”. Este é um objeto intermédio que funciona como ligação do objeto “Transactions” ao objeto “Files” devido a uma relação de muitos para muitos.

*Tabela 11- Dicionário de dados do objeto "Transations_Files"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **Transactions_FilesId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada ligação entre os ficheiros e as transações | Gerado pelo sistema /Não alterável Display Format: TF-{00000} |
| **filesId(FK)** | Master-Detail (Files) | #N/A | #N/A | #N/A | Ligação à tabela dos ficheiros | #N/A |
| **transactionsId(FK)** | Master-Detail(Transactions) | #N/A | #N/A | #N/A | Ligação à tabela das transações | #N/A |

### Objeto: Users

Na Tabela 12 encontra-se o dicionário de dados do objeto “Users”. Os utilizadores são associados a uma conta bancária.

*Tabela 12- Dicionário de dados do objeto "Users"*

| Name | Data type (Salesforce) | Values | Default Value | Mandatory | Descrição | Observações |
|---|---|---|---|---|---|---|
| **userId** | Autonumber | #N/A | #N/A | Yes | Número sequencial que identifica univocamente cada utilizador | Gerado pelo sistema/Não alterável Display Format: US-{00000} |
| **Name__c** | Text(80) | #N/A | #N/A | Yes | Nome do utilizador | #N/A |
| **Email__c** | Email | #N/A | #N/A | Yes | Email do utilizador | #N/A |
| **Phonenumber__c** | Phone | #N/A | #N/A | Yes | Número de contato do utilizador | #N/A |
| **City__c** | Text(40) | #N/A | #N/A | No | Cidade do utilizador | #N/A |

# 5 Tecnologias

Reunidos todos os requisitos e feita toda análise do projeto procedeu-se ao desenvolvimento da aplicação.

Uma vez que a aplicação é em *Salesforce* as tecnologias a usar não levantaram grandes questões, tendo-se usado tecnologias próprias do *Salesforce*. Contudo o trabalho foi enriquecido pelo o uso de algumas tecnologias fora do *Salesforce*.

## 5.1 *Salesforce*

A plataforma *Salesforce* é a número 1 em plataformas de CRM (*Customer Relashionship Management*) que junta as empresas e os utilizadores. A *Salesforce Platform*[1] é um conjunto de tecnologias que viabiliza e serve de base para o desenvolvimento de outras tecnologias. O que torna esta plataforma única é que dá suporte aos recursos personalizados criados por clientes e parceiros, podendo os mesmos ser *layouts* simples ou aplicações completas. O *Salesforce* também ajuda a automatizar com menos código, havendo muitos automatismos.

Esta plataforma é uma plataforma desenvolvida na *cloud* o que leva à existência de serviços variados. Alguns exemplos desses serviços são os seguintes:

- Vendas – vender rápido e de forma inteligente
- Serviços – gerir o suporte a clientes de todos os canais (mensagens, emails, redes sociais)
- Marketing – Enviar mensagens personalizadas em qualquer canal (mensagens, emails, redes sociais)
- *Commerce* – Tornar única a experiência do comprador
- *Quip* – Criar, editar, discutir e organizar o trabalho em equipa, tudo num só sítio
- *Plataform* – Criar, ligar e integrar aplicações

## 5.2 *Apex*

O *Apex*[7] é uma linguagem de programação orientada a objetos, usada pela plataforma *Salesforce*, que permite aos programadores executar instruções de controlo de transações de fluxo em servidores *Salesforce*, sendo possível fazer chamadas à API. Usa uma sintaxe muito semelhante à linguagem de programação Java. Possibilita os programadores adicionar botões ou mesmo atualizações de registos relacionados a páginas do *Visualforce*. O código *Apex* pode ser iniciado por pedidos ao serviço *Web* e por *triggers* em objetos. O *Apex* proporciona acesso direto aos *records* (registros), aos seus campos e à manipulação dos mesmos.

## 5.3 *Visualforce*

O *Visualforce*[8] é uma estrutura que permite aos programadores criar interfaces para os utilizadores de forma sofisticada e personalizada. Essas interfaces podem ser alojadas na plataforma *Lightning* do *Salesforce*.

A estrutura do Visualforce inclui uma linguagem de marcação baseada em *tags*, muito semelhante ao HTML. Cada *tag* corresponde a um componente de interface de utilizador, como por exemplo uma *related list* ou um campo.

O comportamento dos componentes do *Visualforce* pode ser controlado pela mesma lógica usada nas páginas padrão do *Salesforce* ou programadores podem associar a uma lógica personalizada a uma classe de controlador gravada no *Apex*.

## 5.4 *Visual Studio Code*

O *Visual Studio Code*[9] fornece um editor de código, ferramentas de automação de compilação, um depurador e o preenchimento de código inteligente, ou seja, um IDE (ambiente de desenvolvimento integrado). O Controlo das aplicações a desenvolver é feito através da CLI do *Salesforce*, conectando o mesmo ao IDE, neste caso ao *Visual Studio Code,* podendo assim sincronizar metadados entre organizações e fornecendo um sistema de controlo de versões.

O *Salesforce* oferece as melhores ferramentas para fazer o trabalho, e umas das ferramentas de programador são as extensões do *Salesforce* com o *Visual Studio Code*, facilitando o trabalho do mesmo.

## 5.5 *Developer Console*

O *Developer Console*[10] é um IDE que faz o mesmo trabalho que o *Visual Studio Code*, sendo este uma interface do utilizador do *Salesforce*, que atua no browser na plataforma *Salesforce*, ou seja não há necessidade de instalar o *Visual Studio Code* pois é possível fazer o mesmo no browser através do *Developer Console*.

## 5.6 *CSS*

O CSS (Cascading Style Sheets)[11] é uma linguagem de folhas de estilo usada para especificar de que forma alguns elementos da interface do utilizador são apresentados, em termos visuais. Estes documentos são escritos utilizando uma linguagem de marcação, a mais popular de todas, o HTML, sendo neste caso usada numa página *Visualforce*.

# 6 Implementação

Neste capítulo é descrita a implementação da aplicação *Salesforce*.

## 6.1 *Homepage* com recurso ao *Lightning App Builder*

Para a aplicação surgiu a necessidade de criar uma *homepage* e como um dos requisitos do projeto era existir vários tipos de *reports* com gráficos de vários tipos e mostrar transações por ano ou por mês optou-se por fazer uma *homepage* com esses gráficos. E para tal usou-se o *Lightning App Builder*[12] para criar uma *lightning page.* Essa *homepage* encontra-se na página seguinte, na Figura 6.

Nesta homepage existem 4 tabelas diferentes, e 6 gráficos diferentes para se poder tirar conclusões. Nestes gráficos encontram-se informações sobre as categorias mais usadas, onde se gastam mais dinheiro e uma lista de transações por mês e por ano: Existe sempre um gráfico/tabela para despesas e para receitas.

Foi criada também uma página onde ao abrir uma transação se o utilizador for às listas relacionadas pode adicionar um ficheiro à transação, um recibo, comprovativo do pagamento, em qualquer tipo de formato, pode-se ver isso na figura seguinte, Figura 7.

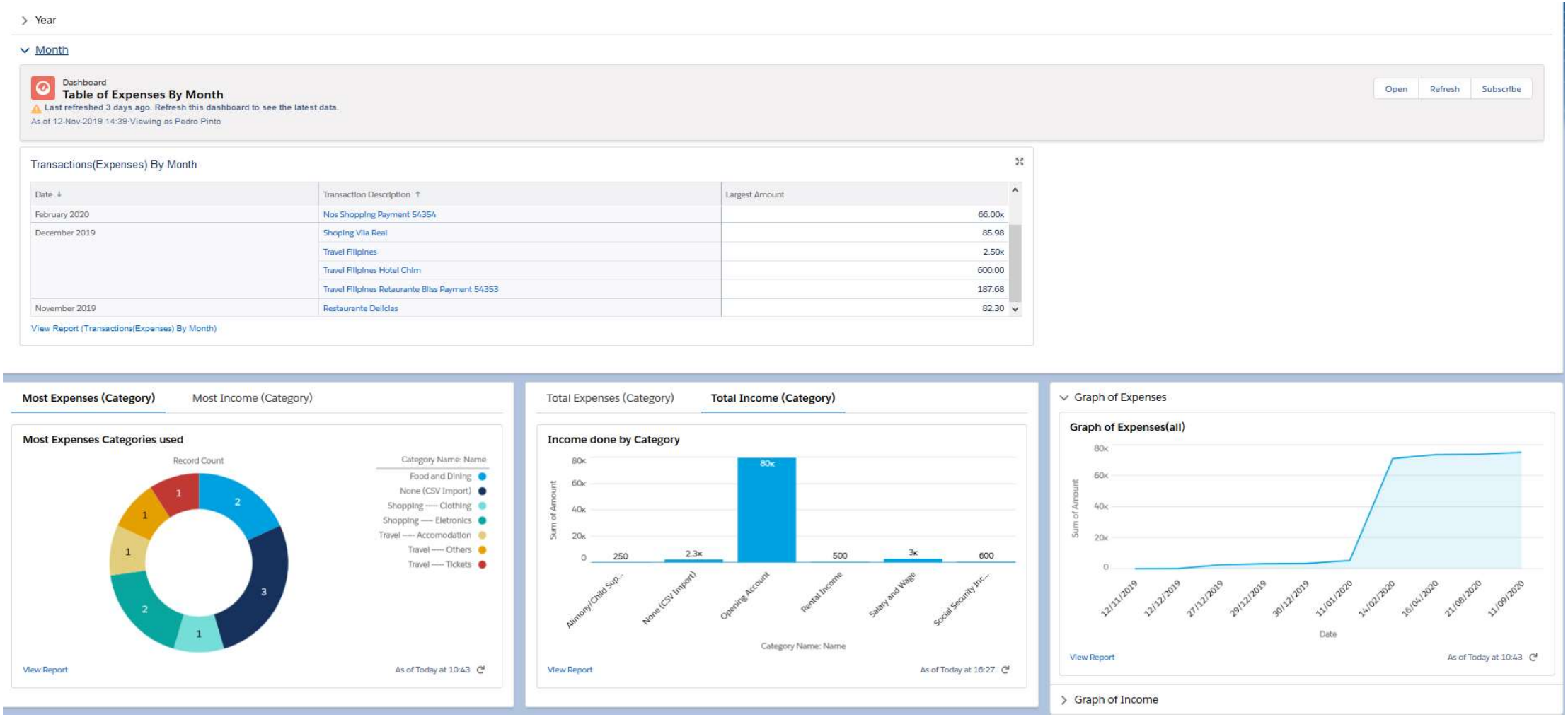



*Figura 6- Homepage na plataforma Salesforce*

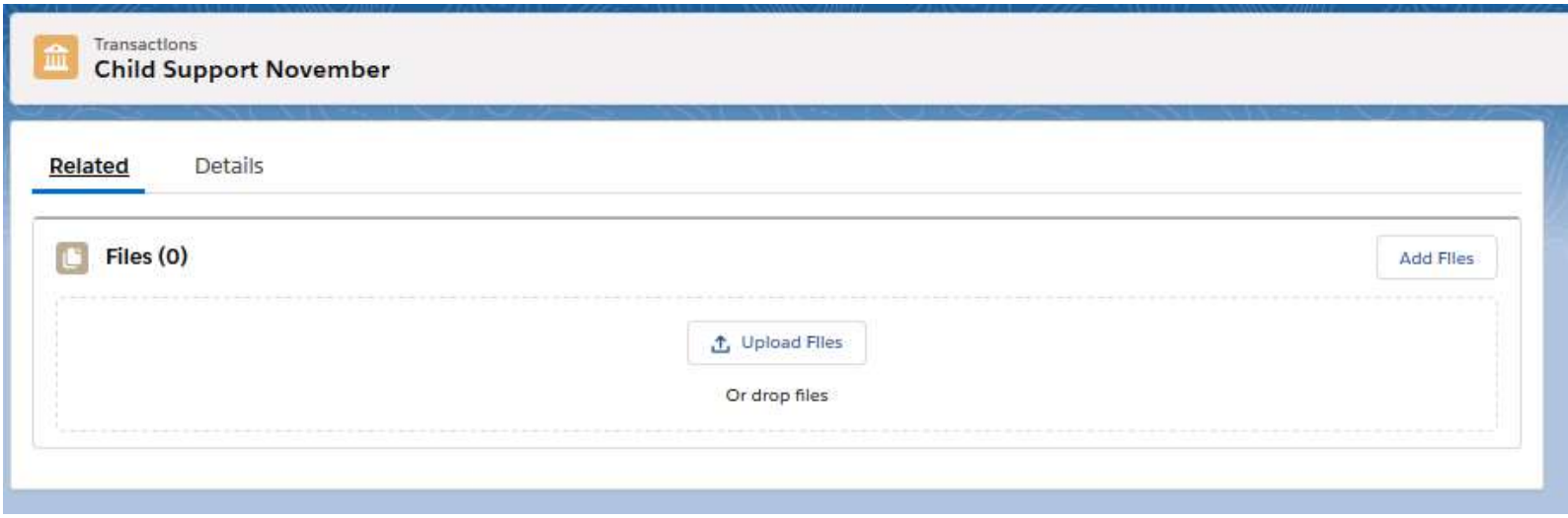



*Figura 7- Página para adicionar ficheiro à transação*

## 6.2 Regras de Validação

As regras de validação, permitem aprimorar a qualidade dos dados dentro de uma aplicação. Estas regras verificam se os dados introduzidos pelo utilizador estão dentro dos padrões para serem inseridos na base de dados, para guardar o registo.

Ao criar os objetos a opção de "*required*" é a primeira regra de validação usada pois se tentarmos inserir um registo sem esse campo preenchido recebemos uma mensagem de erro. Ao usarmos o tipo de campo por exemplo de Email estamos também a usar uma regra de validação pois se o campo não estiver preenchido com um email vai ser mostrada uma mensagem de erro como podemos ver na figura seguinte, Figura 8.

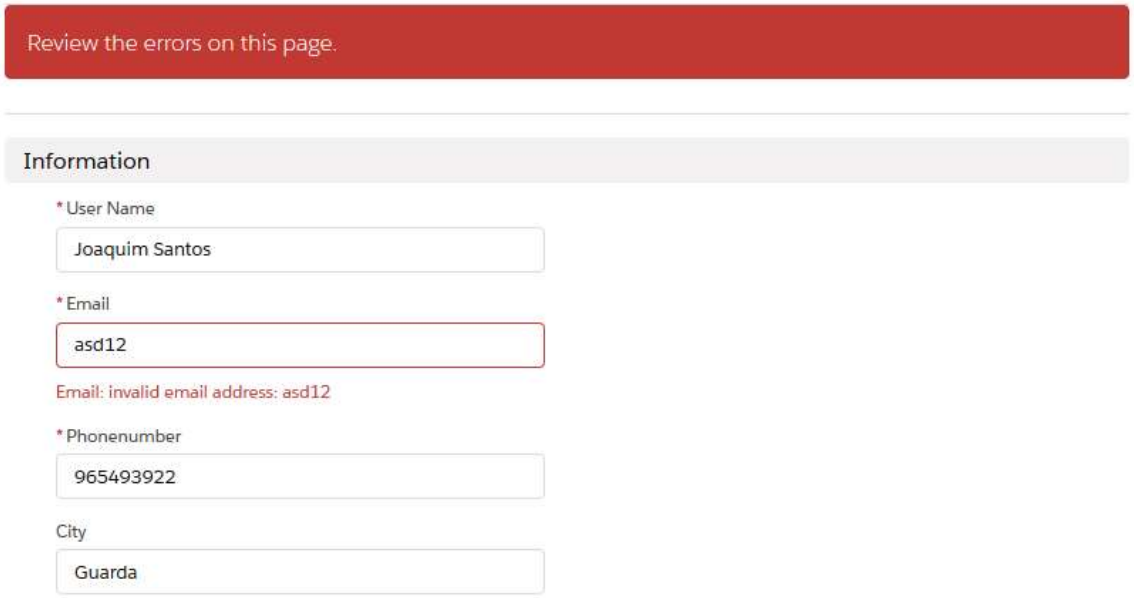



*Figura 8- Validação: Email incorreto na criação de um User*

As outras regras de validação podem conter fórmulas ou expressões que avaliam os dados em um ou mais campos e retornam o valor "True" ou "False". Se o valor retornado for "True", o valor é inválido logo vai haver uma mensagem de erro que é mostrada ao utilizador. Foram criadas algumas regras de validação em alguns objetos para melhorar a qualidade dos dados. Essas regras são explicadas de seguida.

### Objeto "Banks"

Neste objeto não podemos deixar criar objetos do tipo Banco com o mesmo nome, pois não faz sentido haver dois bancos com o mesmo nome, havendo necessidade de criar uma regra de validação com a seguinte fórmula:

Name = VLOOKUP($ObjectType.Banks__c.Fields.Name, $ObjectType.Banks__c.Fields.Name, Name)

Esta fórmula compara o nome inserido com todos os *records* existentes na base de dados se o valor retornado for falso a criação do banco prossegue, senão aparece uma mensagem de erro que o utilizador define.

Nestas fórmulas os espaços brancos são ignorados.

### Objeto "Categories"

Tal como foi feito no objeto anterior, também é necessário criar uma regra de validação semelhante, não permitindo ter duas categorias com o mesmo nome, uma vez que também não faria sentido.

### Objeto "Transactions"

Neste objeto é importante não permitir a criação de transações antes da data da abertura de conta, o que obriga a uma regra de validação com a seguinte fórmula:

Accounts__r.Opening_Date__c > Date__c

É comparada a data de abertura de conta com a data da transação, se esta for menor que a data de abertura aparece uma mensagem de erro definida pelo utilizador.

## 6.3 *Page Layouts*

Ao criar um *record* para cada objeto há uma uma *page layout standard*, mas há possibilidade de alterar/criar *page layouts* de acordo com o que se quer mostrar. Como o utilizador não pode ter acesso a certas funcionalidades é preciso personalizar os *page*

*layouts* de acordo com o que o utilizador precisa e pode ter acesso. A personalização vai de botões à disposição dos elementos na página, adição de vários tipos de elementos, tudo de acordo com o que o utilizador precisar e puder usar.

## 6.4 *Process Builders*

Existem muitas tarefas que normalmente são os utilizadores que têm de realizar podendo ir de tarefas simples a tarefas importantes e complicadas.

No *Salesforce* é possível configurar processos de modo a essas tarefas ficarem automáticas não havendo a necessidade de trabalho repetitivo manual. O Process Builder[12] é a ferramenta usada para configurar esses processos, havendo muitas possibilidades e configurações a fazer. Esta ferramenta suporta três tipos de automação de processo, que determina quando começa o processo. Estes tipos podem ser:

- Quando um registo é criado ou atualizado.
- Quando um evento da plataforma ocorre.
- Quando outro elemento, como por exemplo outro processo, o invoca.

Cada processo tem critérios que determinam quando executar uma ou mais ações, e contêm grupos de ações, que consistem em ações imediatas ou agendadas, sendo estas somente em processos de alteração de registos.

Sempre que possível cria-se *process builders* para facilitar o trabalho. Os process builders usados neste projeto encontram-se indicados abaixo.

### Objeto "Accounts"

Neste objeto surgiu a necessidade de ao criar uma conta bancária, criar uma transação chamada "Opening Balance" sendo o valor, "Amount" desta o valor inserido na criação da conta.

Na Figura 9 pode-se ver o fluxo do processo "Opening Balance Creation".

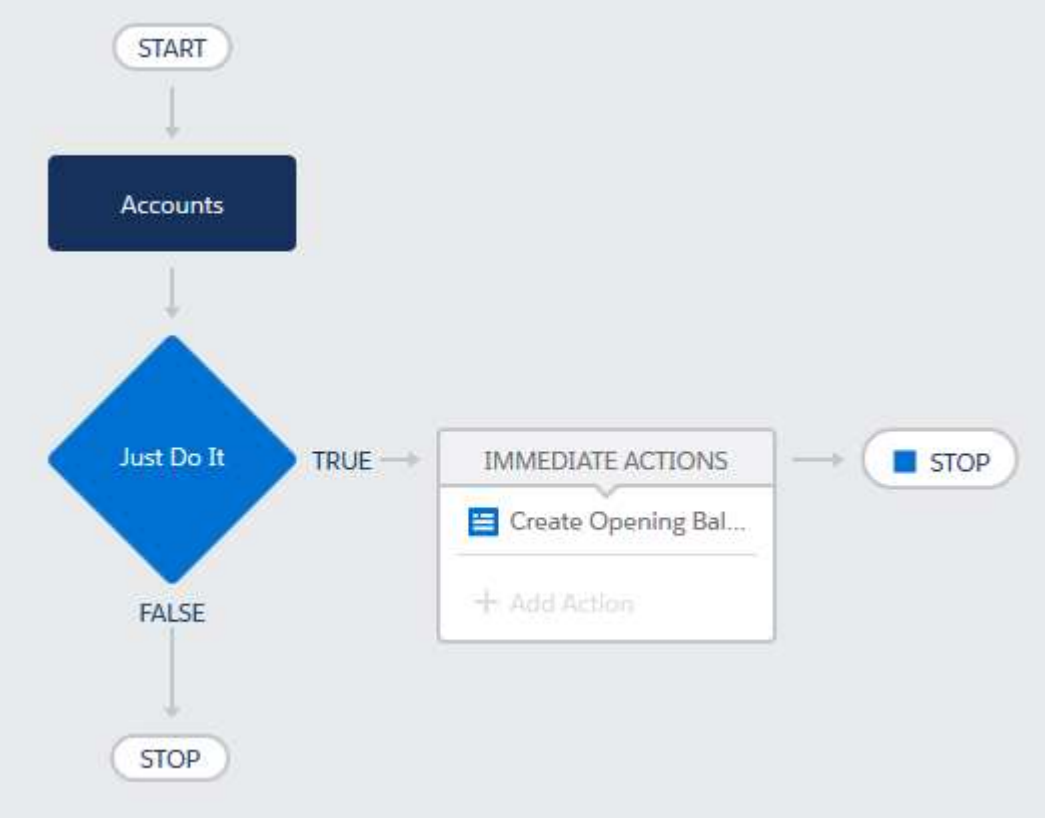



*Figura 9- Process Builder "Opening Balance Creation"*

Como podemos ver o processo está associado ao objeto “Accounts”. Quando criamos o processo temos de escolher o tipo de automação que pretendemos, escolhendo neste caso “Quando um registo é criado, ou atualizado”, desta forma ao criarmos/atualizarmos uma conta este processo vai ocorrer. Para tal não precisa de haver critério nenhum para a ação que se vai realizar, como podemos ver na figura seguinte, Figura 10 o critério de nome “Just Do It” em que só precisa de executar as ações.

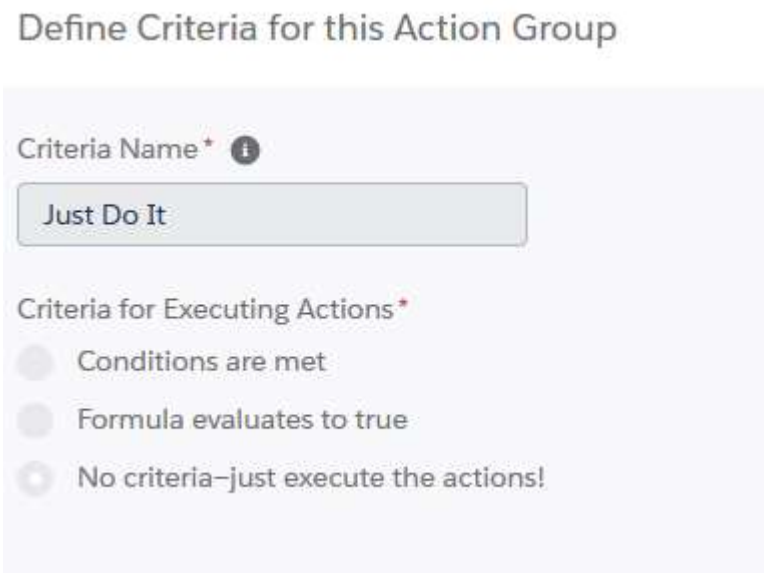



*Figura 10- Process Builder critério de decisão*

A ação a realizar é a criação de um novo *record*, do tipo "Transactions" como podemos ver na Figura 11.

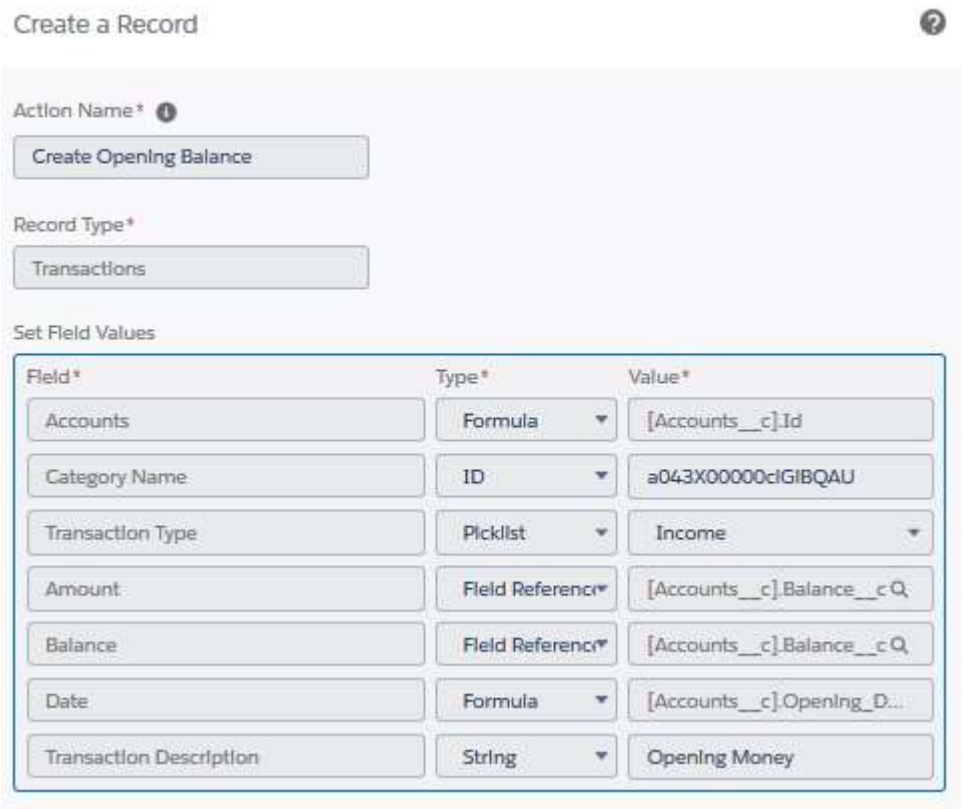



*Figura 11- Process Builder criação de um novo registro "Opening Money"*

Como podemos verificar os campos “Accounts, Amount, Balance, Date” são campos cujo valor vem do objeto “Accounts”, o tipo de transação é do tipo “income” pois é um acrescento de dinheiro na conta bancária, a descrição da transação é “Opening Money”, como foi definido, e a categoria é uma criada para este processo, chamada “Opening Account”

### Objeto “Transactions”

Neste objeto, na criação de um *record* novo existe um campo chamado “Repeat” do tipo *picklist*, em que dependendo da escolha do utilizador, poderia ou não ter de repetir a transação semanalmente, mensalmente ou anualmente. Para tal surgiu a necessidade de criar outro processo, com o nome de “Schedule Transactions”.

O processo pode ver-se na Figura 12.

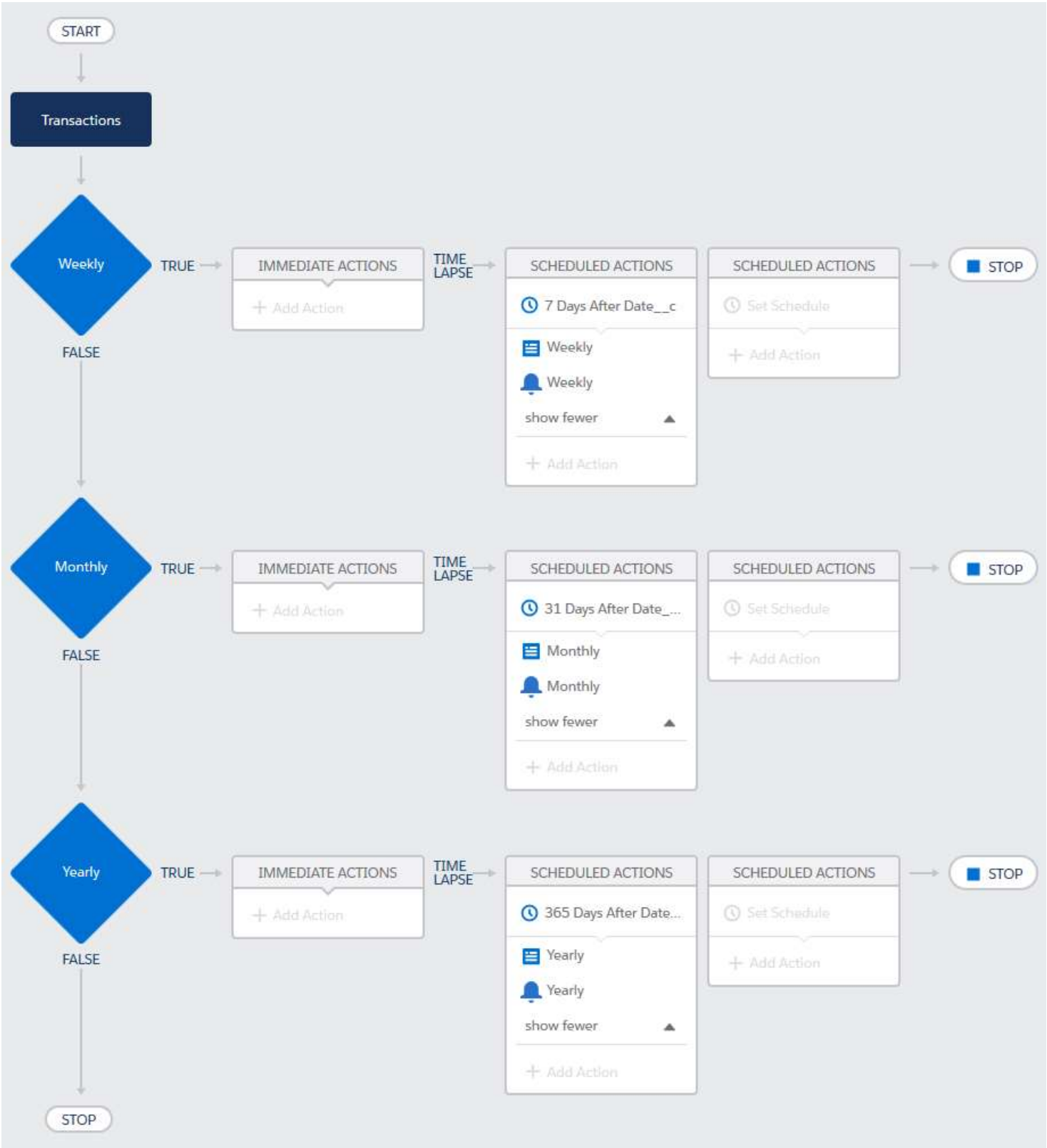



*Figura 12- Process Builder "Schedule Transaction"*

Como podemos ver este processo está associado ao objeto “Transactions” em que o critério de ação é se o valor da picklist for “Every Week”, “Every Month” ou “Every Year”, se um desses se verificar vai ser criada uma transação uma semana, um mês ou um ano depois da data da criação da transação, dependendo da escolha do utilizador com os mesmos valores dessa transação, em que o que difere é a data que vai ser uma semana depois ou um mês ou um ano como já foi referido. Pode-se ver na Figura 13 essa criação, em que a Data é uma fórmula que adiciona 7 dias à data da transação anterior:

[Transactions__c].Date__c + 7

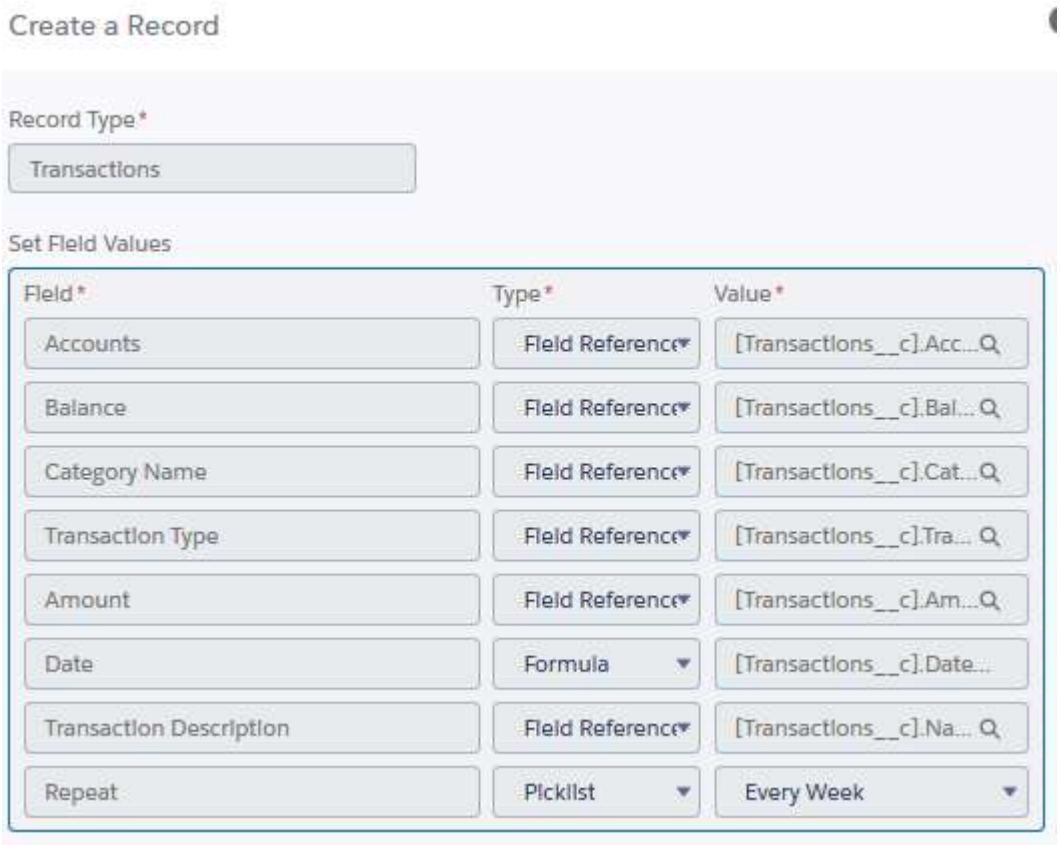



*Figura 13- Process Builder criação de um novo registo, uma transação*

Nas ações a executar depois desse tempo também foi adicionado a criação de uma notificação que é enviada para a interface do utilizador no *Salesforce,* como podemos ver na Figura 14, que é constituída por uma mensagem cuja configuração se ilustra na Figura 15.

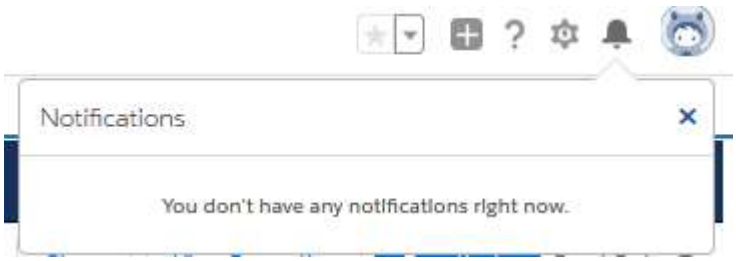



*Figura 14- Notificação na plataforma Salesforce*

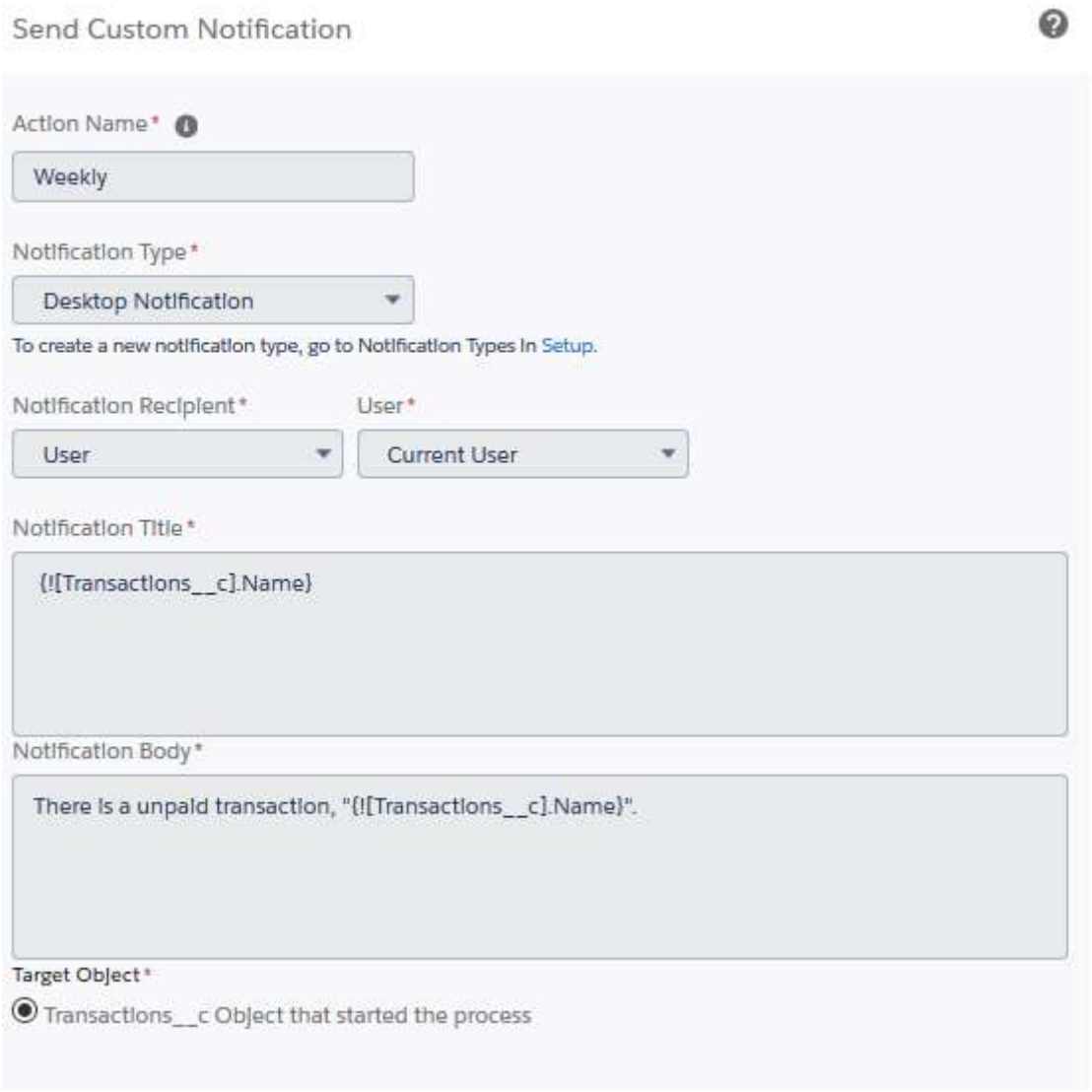



*Figura 15- Mensagem predefinida da notificação na plataforma Salesforce*

No título da mensagem e no corpo var ser passado o nome da transação em questão. Estas ações são repetidas no caso de ser semanalmente, mensalmente e anualmente.

## 6.5 *Triggers*

O *Salesforce* também permite criar *triggers*[14] como os conhecemos a nível de base de dados, através de código, pois apesar dos *process builders* serem uma ferramenta poderosa, não possibilitam a criação de tudo o que se necessita.

Como já foi referido a linguagem usada na plataforma *Salesforce* é o *Apex*. Aqui os *triggers* podem ser configurados para realizar ações personalizadas antes ou depois da alteração dos registos da aplicação, tal como *inserts*, *deletes* ou *updates*.

Neste projeto foi criado um *trigger* para alterar o nome de uma categoria antes de inserir ou de atualizar um registo no caso de haver uma subcategoria, o que permite a organização das categorias. Este caso é referido nos dois capítulos seguintes e o *trigger* pode ser consultado no anexo A.1.

## 6.6 Classes *Apex* (Controladores)

Tal como acontece na linguagem Java, no *apex* é permitido criar classes. Uma classe não é mais que um modelo onde os objetos são criados, ou seja, um objeto é uma instância dessa classe. Uma classe contém variáveis, métodos e pode conter outras classes, como exceção e código de inicialização. As variáveis devem ser usadas para determinar um estado do objeto, como o nome, data. Os métodos são utilizados para controlar o comportamento da classe.

Neste projeto foram criados controladores personalizados para podermos usar as páginas *Visualforce* com os elementos que este projeto requer.

O primeiro controlador disponível no Anexo 2, é um controlador simples onde usamos uma SOQL para retornar uma lista das categorias ordenadas por tipo e por nome.

No controlador "ListTransactions", disponível no Anexo 3, quando entramos na respetiva *Visualforce* page vai executar o *query* de listar as contas bancárias, depois de escolhida a conta é efetuado um outro *query* onde seleciona as transações associadas a essa conta, por ordem de data decrescente. É também calculado o saldo da conta a cada transação efetuada.

Um dos controladores mais complexo é o do Anexo 4 onde são importados dados do *homebanking* através de um ficheiro CSV. Primeiramente funciona como o controlador anterior, onde na *Visualforce* page o utilizador escolhe o banco passando o nome da conta e do banco para o controlador. Nessa página o utilizador escolhe também o ficheiro a importar de acordo com os requisitos definidos. Se o ficheiro CSV estiver em ordem, o controlador *apex* está programado para ir buscar a informação de cada coluna, acrescentando a informação do nome da conta e cria as transações. Se não houver erros é mostrada uma mensagem de sucesso na página *Visualforce* ou caso contrário uma mensagem de erro.

O controlador "ImportDataFromCSV", disponível no Anexo 5 é semelhante ao anterior sendo mais simples pois o utilizador tem de configurar as colunas do ficheiro CSV manualmente.

Por último, o controlador para exportar dados, disponível no Anexo 6, onde existe uma *Visualforce* page que lista as transações todas, havendo um botão para exportar. Esse botão através deste controlador é mandado para outra página *Visualforce* disponível no Anexo 12, que é onde se pode escolher para abrir ou guardar o ficheiro CSV no computador, como podemos ver na Figura 16.

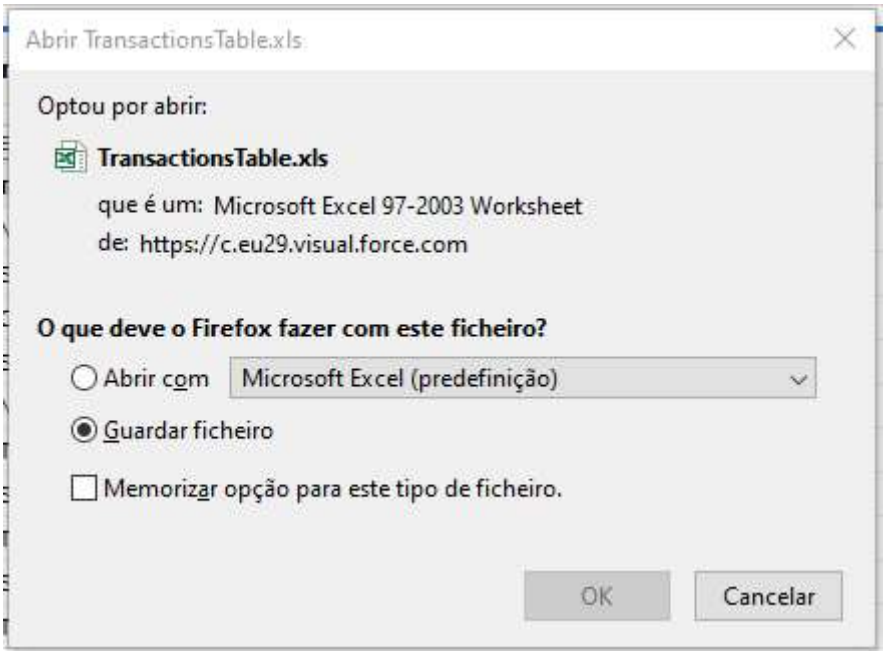



*Figura 16- Janela para guardar/abrir ficheiro*

## 6.7 *VisualForce* Pages

Através do *Visualforce* foi possível criar interfaces para o utilizador usar de forma mais fácil a aplicação e haver componentes que de outra forma não existiriam. As *Visualforce* pages têm dois elementos principais que são a marcação do *Visualforce* e o controlador do *Visualforce*.

No objeto "Categories" foi criada uma página *VisualForce* com o objetivo de ordenar as categorias por tipo de categoria e por nome (sendo este possível por causa do *trigger* "ChangeName"), e de alterar a cor das letras de acordo com o tipo, como podemos verificar na Figura 17.

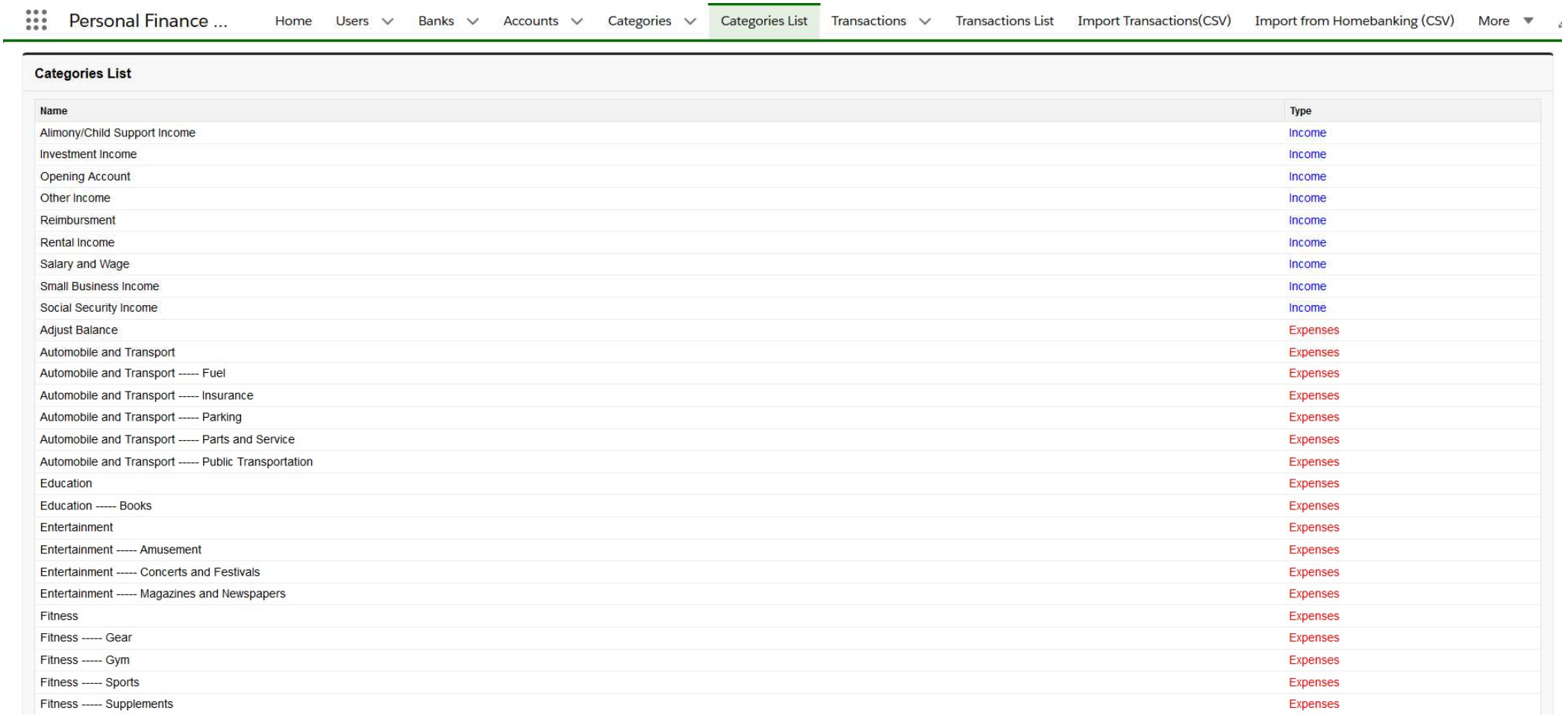



*Figura 17- Lista de Categorias*

Na primeira linha de código disponível no Anexo 7 constam-se alguns elementos importantes.

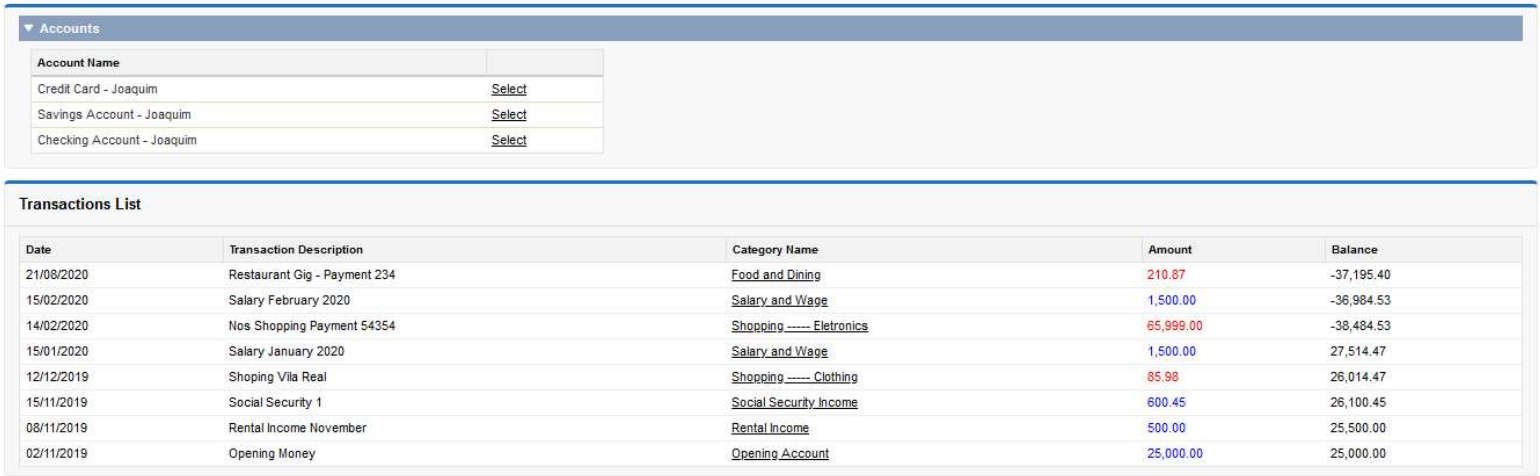
<apex:page controller="Categories" action="{!getCategories}">

Como podemos ver a Visualforce page está conectada ao controlador “Categories” e executa a ação “getCategories” que se encontra nesse controlador.

Para listar as transações também se criou uma página onde se mostra as transações de uma dada conta por ordem de data, e que mostra o saldo da conta consoante as transações efetuadas.

Essa conta é escolhida ao clicarmos no *select* em frente da conta desejada e através do “param” (*tag* na linguagem apex) passa a informação ao controlador que trata do resto.

Podemos ver essa página na Figura 18.



Accounts

| Account Name | |
|---|---|
| Credit Card - Joaquim | Select |
| Savings Account - Joaquim | Select |
| Checking Account - Joaquim | Select |

Transactions List

| Date | Transaction Description | Category Name | Amount | Balance |
|---|---|---|---|---|
| 21/08/2020 | Restaurant Gig - Payment 234 | Food and Dining | 210.87 | -37,195.40 |
| 15/02/2020 | Salary February 2020 | Salary and Wage | 1,500.00 | -36,984.53 |
| 14/02/2020 | Nos Shopping Payment 54354 | Shopping ----- Eletronics | 65,999.00 | -38,484.53 |
| 15/01/2020 | Salary January 2020 | Salary and Wage | 1,500.00 | 27,514.47 |
| 12/12/2019 | Shoping Vila Real | Shopping ----- Clothing | 85.98 | 26,014.47 |
| 15/11/2019 | Social Security 1 | Social Security Income | 600.45 | 26,100.45 |
| 08/11/2019 | Rental Income November | Rental Income | 500.00 | 25,500.00 |
| 02/11/2019 | Opening Money | Opening Account | 25,000.00 | 25,000.00 |

*Figura 18- Seleção de conta bancária na página das transações*

Uma das páginas mais importantes é a página onde importamos os ficheiros CSV que provém do *Homebanking.*

Na Figura 19 encontra-se essa página.

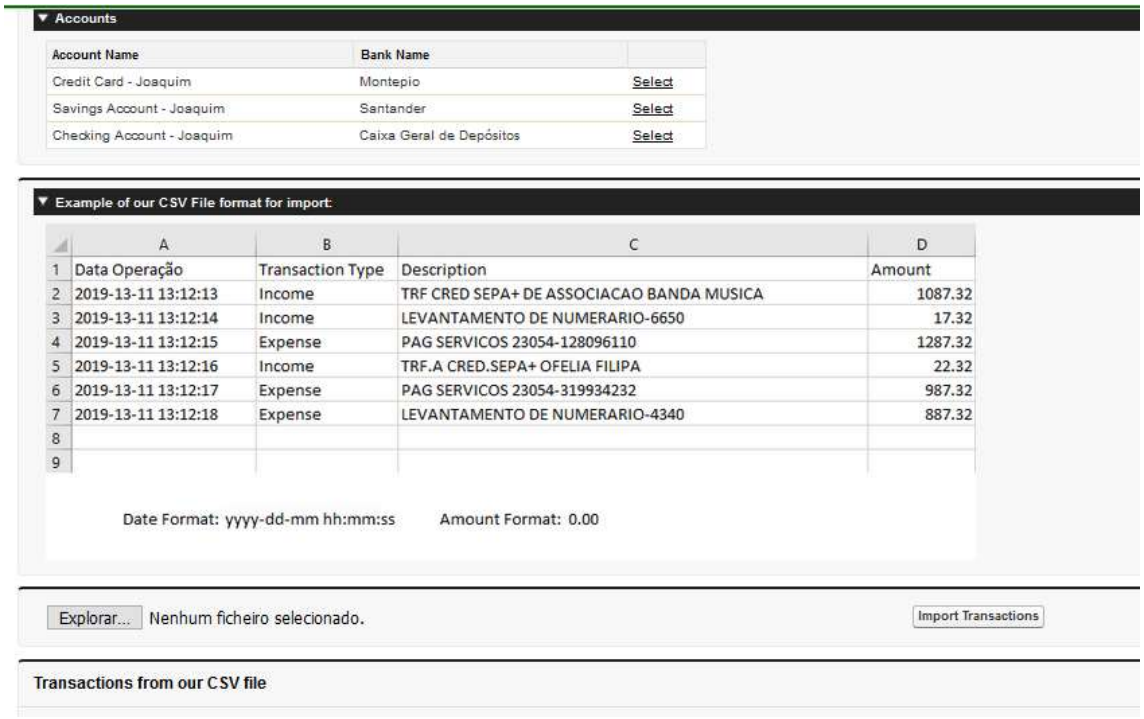



*Figura 19- Página de importação de ficheiro CSV*

Como se pode observar escolhe-se uma conta, sendo o nome da conta e do banco passados para o nosso controlador, preenchendo assim dois campos das transações a importar automaticamente.

Existe uma "BlockSection" (*tag* na linguagem apex) onde foi inserida uma imagem, que mostra como deve estar o ficheiro CSV, e o formato de dois dos campos para não haver problemas ao importar as transações.

Tendo essas configurações todas feitas adiciona-se o ficheiro CSV através do botão "explorar…" e clica-se em "Import Transactions", quando o resultado for bem-sucedido aparece uma mensagem de sucesso como na Figura 20, e as transações importadas como se pode ver na Figura 21.


*Figura 20- Mensagem de sucesso das transações importadas*

**Transactions from our CSV file**

| Transaction Description | Accounts | Date | Transaction Type | Category Name | Amount |
|---|---|---|---|---|---|
| TRF CRED SEPA+ DE ASSOCIACAO BANDA MUSICA | Checking Account - Joaquim | 11/01/2020 | Income | None (CSV Import) | 1,087.32 |
| LEVANTAMENTO DE NUMERARIO-6650 | Checking Account - Joaquim | 11/11/2020 | Income | None (CSV Import) | 17.32 |
| PAG SERVICOS 23054-128096110 | Checking Account - Joaquim | 11/09/2020 | Expense | None (CSV Import) | 1,287.32 |
| TRF.A CRED.SEPA+ OFELIA FILIPA | Checking Account - Joaquim | 11/09/2020 | Income | None (CSV Import) | 22.32 |
| PAG SERVICOS 23054-319934232 | Checking Account - Joaquim | 11/01/2020 | Expense | None (CSV Import) | 987.32 |
| LEVANTAMENTO DE NUMERARIO-4340 | Checking Account - Joaquim | 11/01/2020 | Expense | None (CSV Import) | 887.32 |

*Figura 21- Lista das transações importadas*

Por último, uma *Visualforce* page para exportar os dados das transações todas de uma conta. Esses dados podem ser usados como *backup* de dados ou para o que o utilizador entender. Nessa *Visualforce page* aparece a lista das transações e o utilizador só tem de clicar em "Export", Figura 22.

**Transactions Data Table**

Export

| Transaction Description | Accounts | Date | Transaction Type | Category Name | Amount |
|---|---|---|---|---|---|
| Nos Shopping Payment 54354 | Checking Account - Joaquim | 14/02/2020 | Expense | Shopping ----- Eletronics | 65,999.00 |
| Social Security 1 | Checking Account - Joaquim | 15/11/2019 | Income | Social Security Income | 600.45 |
| Child Support November | Savings Account - Joaquim | 20/11/2019 | Income | Alimony/Child Support Income | 250.00 |
| Salary February 2020 | Checking Account - Joaquim | 15/02/2020 | Income | Salary and Wage | 1,500.00 |
| Nos Shopping Payment Worten | Savings Account - Joaquim | 16/04/2020 | Expense | Shopping ----- Eletronics | 2,500.99 |
| Travel Filipines | Savings Account - Joaquim | 27/12/2019 | Expense | Travel ----- Tickets | 2,500.99 |
| Shoping Vila Real | Checking Account - Joaquim | 12/12/2019 | Expense | Shopping ----- Clothing | 85.98 |
| Travel Filipines Retaurante Bliss Payment 54353 | Savings Account - Joaquim | 30/12/2019 | Expense | Travel ----- Others | 187.68 |
| Salary January 2020 | Checking Account - Joaquim | 15/01/2020 | Income | Salary and Wage | 1,500.00 |
| Travel Filipines Hotel Chim | Savings Account - Joaquim | 29/12/2019 | Expense | Travel ----- Accomodation | 600.00 |
| Opening Money | Credit Card - Joaquim | 01/11/2019 | Income | Opening Account | 5,000.00 |
| Opening Money | Savings Account - Joaquim | 01/11/2019 | Income | Opening Account | 50,000.00 |
| Opening Money | Checking Account - Joaquim | 02/11/2019 | Income | Opening Account | 25,000.00 |
| Restaurante Delicias | Credit Card - Joaquim | 12/11/2019 | Expense | Food and Dining | 82.30 |
| Rental Income November | Checking Account - Joaquim | 08/11/2019 | Income | Rental Income | 500.00 |
| Restaurant Gig - Payment 234 | Checking Account - Joaquim | 21/08/2020 | Expense | Food and Dining | 210.87 |
| TRF CRED SEPA+ DE ASSOCIACAO BANDA MUSICA | Credit Card - Joaquim | 11/01/2020 | Income | None (CSV Import) | 1,087.32 |
| LEVANTAMENTO DE NUMERARIO-6650 | Credit Card - Joaquim | 11/11/2020 | Income | None (CSV Import) | 17.32 |
| PAG SERVICOS 23054-128096110 | Credit Card - Joaquim | 11/09/2020 | Expense | None (CSV Import) | 1,287.32 |
| TRF.A CRED.SEPA+ OFELIA FILIPA | Credit Card - Joaquim | 11/09/2020 | Income | None (CSV Import) | 22.32 |
| PAG SERVICOS 23054-319934232 | Credit Card - Joaquim | 11/01/2020 | Expense | None (CSV Import) | 987.32 |
| LEVANTAMENTO DE NUMERARIO-4340 | Credit Card - Joaquim | 11/01/2020 | Expense | None (CSV Import) | 887.32 |

*Figura 22- Tabela de todas as transações a exportar*

## 6.8 Recursos Estáticos

Os recursos estáticos servem para o carregamento de conteúdo externo que pode ser chamado numa página *visualforce*. Estes recursos podem ser ficheiros de *javascript*, ficheiros css, ficheiros comprimidos, tais como .zip, .jar ou imagens.

Neste projeto foram usadas duas imagens, uma para o ícone da aplicação na plataforma de *Salesforce*, Figura 23, e outra para mostrar a disposição das colunas do Excel na importação de transações de um ficheiro CSV, como referido no capítulo anterior, estando essa imagem na Figura 19.


*Figura 23- Ícone da aplicação na plataforma Salesforce*

# 7 Verificação e Validação

Ao longo de todo o projeto foram desenvolvidos vários tipos de testes visto que se optou por uma metodologia ágil, tal como foi explicado anteriormente.

Muitos desses testes foram úteis na criação de todos os automatismos e na escolha das melhores ferramentas para os fazer, o que ajudou a resolver os erros de forma rápida e a não complicar o projeto numa fase mais avançada.

Nas figuras seguintes são demonstrados alguns exemplos de teste, desde validações de campos (Figura 24 e 25) a impossibilidade de criação de, por exemplo, transações numa data anterior à da criação da conta bancária (Figura 26) e a importação de dados de forma incorreta (Figura 27).

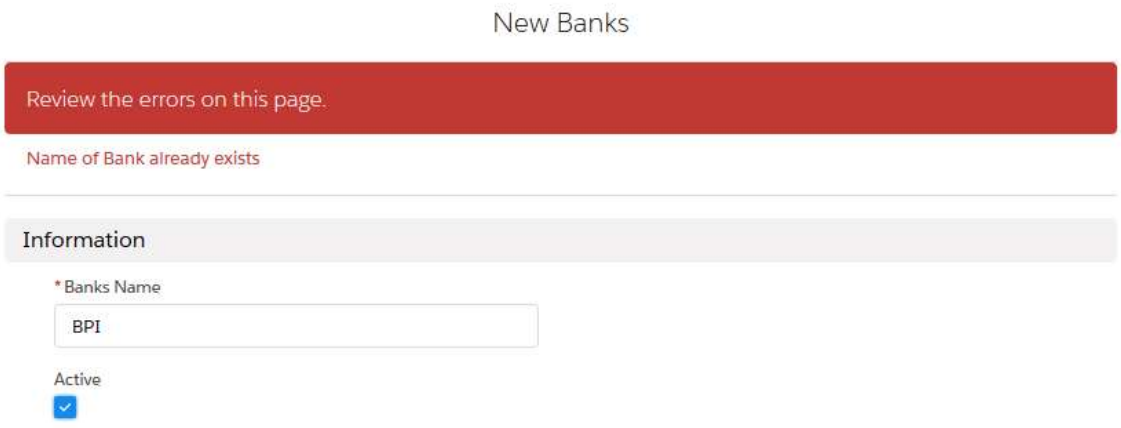



*Figura 24- Erro ao não preencher descrição e nome da conta numa transação*

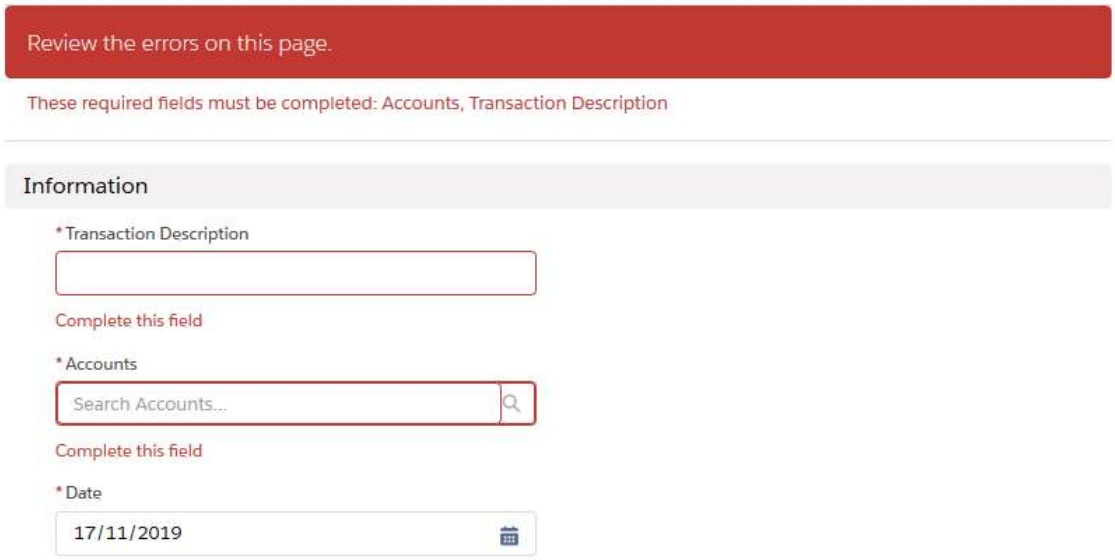



*Figura 25- Erro ao inserir nome do banco*

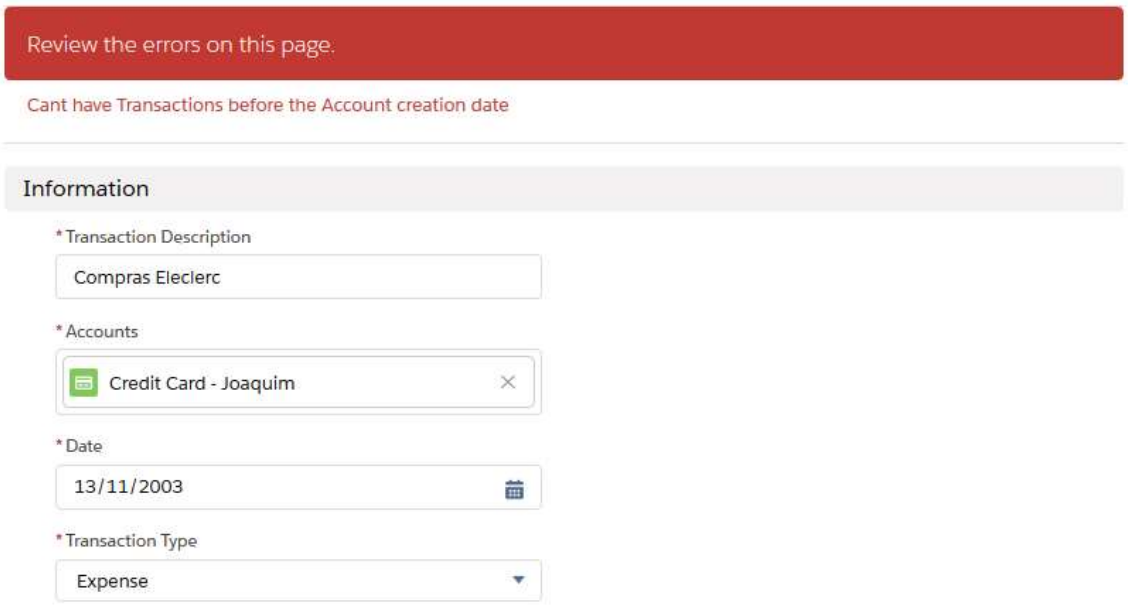



*Figura 26- Erro ao inserir uma data inferior ao da criação da conta*


*Figura 27-Erro na configuração do ficheiro CSV*

# 8 Conclusões

Nos tempos que correm gerir as finanças é um ponto importante na vida das pessoas. Com este projeto é possível ajudar nesse aspeto.

Neste projeto inicialmente encontraram-se algumas dificuldades que foram ultrapassadas através da análise de requisitos. Como em todos os projetos este é um ponto crucial.

As ferramentas e linguagens utilizadas no desenvolvimento da aplicação foram estudadas e entendidas, sendo que toda a programação realizada resolveu os problemas inicialmente propostos pela empresa onde foi realizado estágio.

Além desse estudo, algumas unidades curriculares do curso foram importantes na ajuda do desenvolvimento do projeto e da realização do relatório final, tais como Base de Dados I e II, Engenharia de Software I e II, bem como as unidades curriculares que envolviam programação.

Em relação à tecnologia *Salesforce*, que foi também estudada principalmente através dos *Trailheads*, foi adquirido um gosto pela plataforma ao longo da realização do projeto, onde foram cumpridos os requisitos inicialmente propostos para o mesmo.

Os testes foram feitos ao longo do projeto, tendo havido também *feedback* do supervisor, o que resultou na melhoria constante do resultado final.

O único requisito do projeto que não foi cumprido foi o "gerar PDF dos relatórios", pois o *Salesforce* ainda não permite gerar um PDF de uma página do tipo *reports,* estando esse requisito na lista de implementações futuras no *Salesforce*.

É de sublinhar que ainda há muitas outras funcionalidades e melhoramentos, como por exemplo a possibilidade de haver transferências entre contas internacionais, haver diferentes tipos de moeda, que possivelmente podem ser adicionados/feitos para além dos requisitos iniciais, havendo sempre espaço para mais.

O projeto em contexto de estágio é uma mais valia para os estudantes pois permite haver um primeiro contacto com empresas e o trabalho numa empresa, ajudando os estudantes que pretendam entrar no mercado de trabalho depois da licenciatura.

Para o futuro é esperada a melhoria e comercialização da aplicação, bem como a continuação do trabalho na plataforma *Salesforce*.

# Referências Bibliográficas

[1] “Plataform Development Basics” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/platform_dev_basics?trail_id=force_com_dev_beginner. [Acedido a 25 de Setembro de 2019].

[2] “YNAB” [Online]. Available: https://www.youneedabudget.com/. [Acedido a 20 de Setembro de 2019].

[3] “Buxfer” [Online]. Available: https://demo.buxfer.com/dashboard. [Acedido a 20 de Setembro de 2019].

[4] “Controle Finance” [Online]. Available: https://controle.finance/faces/safearea/dashboard.xhtml. [Acedido a 20 de Setembro de 2019].

[5] “Developer Beginner” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/trails/force_com_dev_beginner?lang=en. [Acedido a 27 de Setembro de 2019].

[6] Libardi, P., & Barbosa, V. (2010). Métodos Ágeis. Universidade Estadual de Campinas, Faculdade de Tecnologia

[7] “Apex Basics & Database” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_database?trail_id=force_com_dev_beginner. [Acedido a 28 de Setembro de 2019].

[8] “Visualforce Basics” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/visualforce_fundamentals?trail_id=force_com_dev_beginner. [Acedido a 4 de Outubro de 2019].

[9] “Quick Start: Visual Studio Code for Salesforce Development” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/projects/quickstart-vscode-salesforce?trail_id=force_com_dev_beginner. [Acedido a 15 de Outubro de 2019].

[10] “Developer Console Basics” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/developer_console?trail_id=force_com_dev_beginner. [Acedido a 15 de Outubro de 2019].

[11]. “CSS” [Online]. Available: https://www.w3schools.com/css/ . [Acedido a 6 de Outubro de 2019]

[12] “Lightning Flow” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/business_process_automation?trail_id=force_com_dev_beginner. [Acedido a 2 de Novembro de 2019].

[13] “Apex Triggers” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_triggers?trail_id=force_com_dev_beginner. [Acedido a 25 de Outubro de 2019].

# Anexos

# A 1. *Trigger* "ChangeName"

```
trigger ChangeName on Categories__c (before insert, before update) {

    List<Categories__c> ctg = new List<Categories__c>();
    for(Categories__c thisctg :trigger.new){
        String preCTG = ' ----- ';
        if(thisctg.Subcategory_Name__c!= null){
            thisctg.Name = thisctg.Subcategory_Name__c + preCTG + thisctg.Name;
            ctg.add(thisctg);
        }
    }
}
```

# A 2. Controlador “Categories.apxc”

```
public with sharing class Categories {

   public List<Categories__c> CategoriesRecords {get; set;}

   public void getCategories() {
      CategoriesRecords = [SELECT Name, Type__c, Subcategory_Name__c FROM Categories__c ORDER BY Type__c DESC, Name];
   }
}
```

# A 3. Controlador “ListTransactions.apxc”

```
public with sharing class ListTransactions {

   public List <Transactions__c> transactions {get; set;}

   public string accountName {get; set;}

   public List <Accounts__c> accountsList{get; set;}

   public List <Transactions__c> balancecalc = new List<Transactions__c>();

   public ListTransactions(){
      accountsList = new List<Accounts__c>();
      transactions = new List<Transactions__c>();
      accountsList = [SELECT Name FROM Accounts__c];
   }

   public void getRelatedTransactions() {
      Decimal balance;
      balancecalc = [SELECT Date__c, Name, Category_Name__c, Amount__c,
Balance__c, Transaction_Type__c FROM Transactions__c
                  WHERE Accounts_Name__c =: accountName ORDER BY Date__c
DESC];
      for(Integer i = balancecalc.size()-1; i > -1; i--){
```

```
            if(balancecalc[i].Category_Name__c == 'a043X00000ciGiBQAU'){
                balance = balancecalc[i].Amount__c;
            }else {
                if(balancecalc[i].Transaction_Type__c == 'Expense'){
                    balancecalc[i].Balance__c = balance - balancecalc[i].Amount__c;
                    }else {
                        balancecalc[i].Balance__c = balance + balancecalc[i].Amount__c;
                    }
                                                balance = balancecalc[i].Balance__c;
            }
        }
        transactions = balancecalc;
    }
}
```

# A 4. Controlador “ImportCSVHomebanking.apxc”

```
public with sharing class ImportCSVHomebanking {
        public Blob csvFileBody{get; set;}
   public string csvAsString{get; set;}
   public string accountName {get; set;}
   public string bankName{get; set;}
   public String[] csvFileLines{get; set;}
   public List<Transactions__c> trslist{get;set;}
   public List <Accounts__c> accountsList{get; set;}

   public ImportCSVHomebanking(){
        csvFileLines = new String[]{};
        trslist = new List<Transactions__c>();
        accountsList = new List<Accounts__c>();
        accountsList = [SELECT Id, Name, Banks__c, Bank_Name__c FROM
Accounts__c];

   }

   public void importCSVFile(){
      try{
                        csvAsString = csvFileBody.toString();
```

```
                    csvFileLines = csvAsString.split('\n');


        //if(bankName == 'Santander'){
            for(Integer i = 1; i <csvFileLines.size();i++){
            Transactions__c trsObj = new Transactions__c();
            String[] csvRecordData = csvFileLines[i].split(',');
            trsObj.Name = csvRecordData[2] ;
            trsObj.Accounts__c = accountName;
            trsObj.Date__c = Date.valueOf(csvRecordData[0]);
            trsObj.Transaction_Type__c =csvRecordData[1];
            trsObj.Category_Name__c = 'a043X00000cidxwQAA';
            trsObj.Amount__c = Decimal.valueOf(csvRecordData[3].trim());
            trslist.add(trsObj);
        //}
        if(trslist.size()>0){
        upsert trslist;
          ApexPages.Message successMessage = new ApexPages.Message(ApexPages.Severity.INFO,'The Transactions were imported correctly');
          ApexPages.addMessage(successMessage);
        }
       }
    }
    catch (Exception e){
```

```
            ApexPages.Message errorMessage = new
ApexPages.Message(ApexPages.Severity.ERROR,'An error has occured while
importing data please make sure input file is correct');
            ApexPages.addMessage(errorMessage);
        }
    }
}
```

# A 5. Controlador “ImportDataFromCSV.apxc”

```
public with sharing class ImportDataFromCSV {
    public Blob csvFileBody{get; set;}
    public string csvAsString{get; set;}
    public String[] csvFileLines{get; set;}
    public List<Transactions__c> trslist{get;set;}

    public importDataFromCSV(){
        csvFileLines = new String[]{};
        trslist = new List<Transactions__c>();
    }

    public void importCSVFile(){
       try{
                        csvAsString = csvFileBody.toString();
                        csvFileLines = csvAsString.split('\n');

         for(Integer i = 1; i <csvFileLines.size();i++){
            Transactions__c trsObj = new Transactions__c();
            String[] csvRecordData = csvFileLines[i].split(',');
            trsObj.Name = csvRecordData[0] ;
```

```
            trsObj.Accounts__c = csvRecordData[1];
            trsObj.Date__c = Date.valueOf(csvRecordData[2]);
            trsObj.Transaction_Type__c = csvRecordData[3];
            trsObj.Category_Name__c = csvRecordData[4];
            trsObj.Amount__c = Decimal.valueOf(csvRecordData[5].trim());
            trslist.add(trsObj);
        }
        if(trslist.size()>0){
        insert trslist;
        }
    }
    catch (Exception e){
        ApexPages.Message errorMessage = new ApexPages.Message(ApexPages.Severity.ERROR,'An error has occured while importing data please make sure input file is correct');
        ApexPages.addMessage(errorMessage);
    }
  }
}
```

# A 6. Controlador “ExportData.apxc”

```
public with sharing class ExportData {
  //constructor
  public ExportData() {
  }

  //list of accounts
  public List<Transactions__C> transactions {
    get {
      if(transactions != null) {
        return transactions;
      } else {
        transactions = [Select Name, Accounts__c, Date__c,
Transaction_Type__c,Category_Name__c,Amount__c from transactions__c];
        return transactions;
      }
    }
    set;
  }
  public Pagereference exportAll(){
    return new Pagereference('/apex/ExcelFilePage');
 }
}
```

# A 7. Página *Visualforce* "PageCategories.vfp"

```
<apex:page controller="Categories" action="{!getCategories}">
        <apex:pageBlock title="Categories List">
        <apex:pageBlockTable value="{!CategoriesRecords}" var="ctg" >
           <apex:column value="{!ctg.Name}"/>
           <apex:column value="{!ctg.Type__c}"
style="{!if(ctg.Type__c='Expenses','color:red','color:blue')}"/>
        </apex:pageBlockTable>
      </apex:pageBlock>
</apex:page>
```

# A 8. Página *Visualforce* "TransactionsDataTablePage.vfp"

```
<apex:page controller="ExportData" tabStyle="Account">
   <apex:form >
     <apex:pageBlock title="Transactions Data Table">
        <apex:commandButton value="Export"  action="{!exportAll}"/>
          <apex:pageBlockTable value="{!transactions}" var="trs">
            <apex:column value="{!trs.Name}"/>
            <apex:column value="{!trs.Accounts__c}"/>
            <apex:column value="{!trs.Date__c}"/>
            <apex:column value="{!trs.Transaction_Type__c}"/>
            <apex:column value="{!trs.Category_Name__c}"/>
            <apex:column value="{!trs.Amount__c}"/>
          </apex:pageBlockTable>
     </apex:pageBlock>
   </apex:form>
</apex:page>
```

# A 9. Página *Visualforce* "ListTransactionsPage.vfp"

```
<apex:page controller="ListTransactions" tabStyle="Account" showHeader="false" sidebar="false">
   <apex:form >
      <apex:pageBlock >
         <apex:pageBlockSection title="Accounts">
                  <apex:pageBlockTable value="{!accountsList}" var="account">
               <apex:column value="{!account.Name}"/>
                  <apex:column >
                  <apex:commandLink value="Select" action="{!getRelatedTransactions}">
                        <apex:param name="Name" assignTo="{!accountName}" value="{!account.Name}"/>
                  </apex:commandLink>
               </apex:column>
            </apex:pageBlockTable>
         </apex:pageBlockSection>
      </apex:pageBlock>
      <apex:pageBlock title="Transactions List">
         <apex:pageBlockTable value="{!transactions}" var="trs" >
            <apex:column value="{!trs.Date__c}"/>
            <apex:column value="{!trs.Name}"/>
```

```
            <apex:column value="{!trs.Category_Name__c}"/>
            <apex:column value="{!trs.Amount__c}"
style="{!if(trs.Transaction_Type__c='Expense','color:red','color:blue')}"/>
            <apex:column value="{!trs.Balance__c}"/>
          </apex:pageBlockTable>
       </apex:pageBlock>
   </apex:form>
</apex:page>
```

# A 10. Página *Visualforce* "ImportCSVPage.vfp"

```
<apex:page controller="ImportDataFromCSV">
   <apex:form >
      <apex:pageMessages />
      <apex:pageBlock >
                        <apex:pageBlockSection >
            <apex:inputFile value="{!csvFileBody}" filename="{!csvAsString}"/>
            <apex:commandButton value="Import Transactions"
action="{!importCSVFile}"/>
         </apex:pageBlockSection>
      </apex:pageBlock>
      <apex:pageBlock >
         <apex:pageBlockTable value="{!trslist}" var="trs">
               <apex:column value="{!trs.Name}"/>
            <apex:column value="{!trs.Accounts__c}"/>
            <apex:column value="{!trs.Date__c}"/>
            <apex:column value="{!trs.Transaction_Type__c}"/>
            <apex:column value="{!trs.Category_Name__c}"/>
            <apex:column value="{!trs.Amount__c}"/>
         </apex:pageBlockTable>
      </apex:pageBlock>
   </apex:form>
</apex:page>
```

# A 11. Página *Visualforce* "ImportCSVHomebankingPage.vfp"

```
<apex:page controller="ImportCSVHomebanking">
  <apex:form >
    <apex:pageMessages />
    <apex:pageBlock >
      <apex:pageBlockSection title="Accounts" >
          <apex:pageBlockTable value="{!accountsList}" var="account" >
        <apex:column value="{!account.Name}" />
        <apex:column value="{!account.Bank_Name__c}" />
          <apex:column >
          <apex:commandLink value="Select" >
              <apex:param name="Name" assignTo="{!accountName}"
value="{!account.Id}"/>
                <apex:param name="Bank" assignTo="{!bankName}"
value="{!account.Bank_Name__c}"/>
          </apex:commandLink>
        </apex:column>
      </apex:pageBlockTable>
    </apex:pageBlockSection>
  </apex:pageBlock>
  <apex:pageBlock >
    <apex:pageBlockSection title="Example of our CSV File format for import:">
```

```
                <apex:image
url="https://c.eu29.content.force.com/sfc/servlet.shepherd/version/renditionDownload?rendition=ORIGINAL_Png&versionId=0683X0000097b88&operationContext=CHATTER&contentId=05T3X00000UAsat"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
    <apex:pageBlock >
                    <apex:pageBlockSection >
        <apex:inputFile value="{!csvFileBody}" filename="{!csvAsString}"/>
        <apex:commandButton value="Import Transactions"
action="{!importCSVFile}"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
    <apex:pageBlock title="Transactions from our CSV file">
      <apex:pageBlockTable value="{!trslist}" var="trs">
            <apex:column value="{!trs.Name}"/>
        <apex:column value="{!trs.Accounts__c}"/>
        <apex:column value="{!trs.Date__c}"/>
        <apex:column value="{!trs.Transaction_Type__c}"
style="{!if(trs.Transaction_Type__c='Expense','color:red','color:blue')}"/>
        <apex:column value="{!trs.Category_Name__c}"/>
        <apex:column value="{!trs.Amount__c}"/>
      </apex:pageBlockTable>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

# A 12. Página *Visualforce* "ExcelFilePage.vfp"

```
<apex:page controller="ExportData" contentType="application/vnd.ms-excel#TransactionsTable.xls" showHeader="false" standardStylesheets="false">
  <apex:pageBlock >
    <apex:dataTable value="{!transactions}" var="trs">
      <apex:column value="{!trs.Name}"/>
      <apex:column value="{!trs.Accounts__c}"/>
      <apex:column value="{!trs.Date__c}"/>
      <apex:column value="{!trs.Transaction_Type__c}"/>
      <apex:column value="{!trs.Category_Name__c}"/>
      <apex:column value="{!trs.Amount__c}"/>
    </apex:dataTable>
  </apex:pageBlock>
</apex:page>
```