
Arquitecturas Móveis

# Lista de Compras

Trabalho iOS

Vasco Leite Gomes (21260509)

a21260509@alunos.isec.pt

# Índice

# 1. Introdução

No âmbito deste trabalho, desenvolveu-se uma aplicação que permite fazer a gestão de uma lista de compras.

A aplicação é constituída por uma lista de lista de compras, e cada lista é constituída por produtos que são definidos por: designação, marca, quantidade, unidades, preço, observações e data de inserção. O utilizador pode adicionar, apagar ou editar listas, bem como adicionar, editar ou apagar itens de cada lista. Tem também a possibilidade de copiar uma lista já criada. Dentro das listas, os produtos podem ser organizados e ordenados por ordem alfabética, data de inserção, marca ou categoria.

Finalmente, é possível também marcar um item da lista como já comprado, e no caso de todos os produtos de uma lista ficarem assinalados como comprados, a lista passa para um estado de lista completada, ficando a aplicação com listas completadas e listas atuais.

Toda a organização da aplicação, bem como algumas questões de desenvolvimento serão explicadas nos próximos capítulos deste relatório.

# 2. Organização Geral

Tentei usar o padrão MVC (*Model-View-Controller*) no âmbito deste trabalho, e isso é percetível na forma como os ficheiros estão organizados

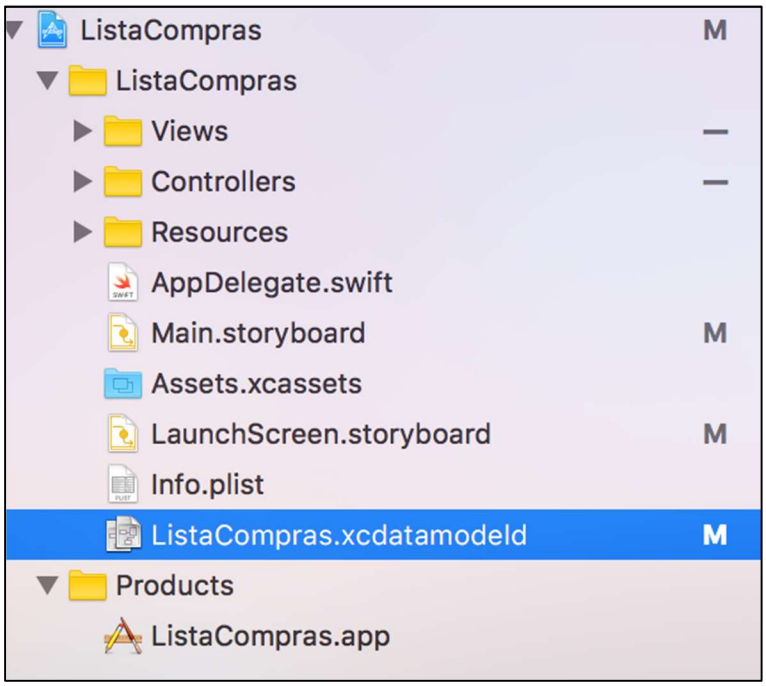


*Imagem 1 – Organização dos ficheiros do projeto*

## *Model*

No caso do *Model*, como a gestão da base dados e a criação dos objetos é feita pelo *Core Data*, com maior ênfase a partir da versão 3 do *Swift* (na versão 2 ainda era aconselhado criar objetos que representariam as nossas tabelas, e que seriam portanto os nossos objetos de negócio e integrariam o *Model*. No entanto, na versão 3 isso mudou e tudo é gerido pelo *Core Data*, tirando assim trabalho ao

programador. Por essa razão, não existe uma pasta Model que contenha objetos de negócio.

## *View*

As *views* que estou a utilizar no âmbito deste projeto são:

- *ListItemCell*
    - Esta classe representa a *TableViewCell* customizada da *TableView* que contém a lista de listas de compras.
- *CustomTextField*
    - Esta classe foi feita para customizar um pouco as textViews presentes neste programa, mudando alguns aspetos gráficos para ficar com uma visualização um pouco mais bonita.
- *ItemCellTableViewCell*
    - Esta classe representa a *TableViewCell* customizada da *TableView*, significando portanto que é a representação de um item.

## *Controller*

Os *controllers* que estou a utilizar no âmbito deste projeto são:

- *MainViewController*
    - Esta classe é uma extensão da *TableViewController* e representa a lista de lista de compras. É o 1º ecrã da nossa aplicação.
- *ListViewController*
    - Esta classe representa a lista de itens, com vários botões na barra de navegação que permitem editar a lista, apagar a lista, copiar a lista ou adicionar itens à mesma.
- *ItemDetailsViewController*
    - Esta classe representa a adição ou edição de um novo item a uma lista.

# 3. Modelo de Dados

Como já foi referido, utilizou-se o *Core Data* para fazer e gerir a ligação à base de dados. O modelo de dados que se implementou pode ser visto na imagem 5.

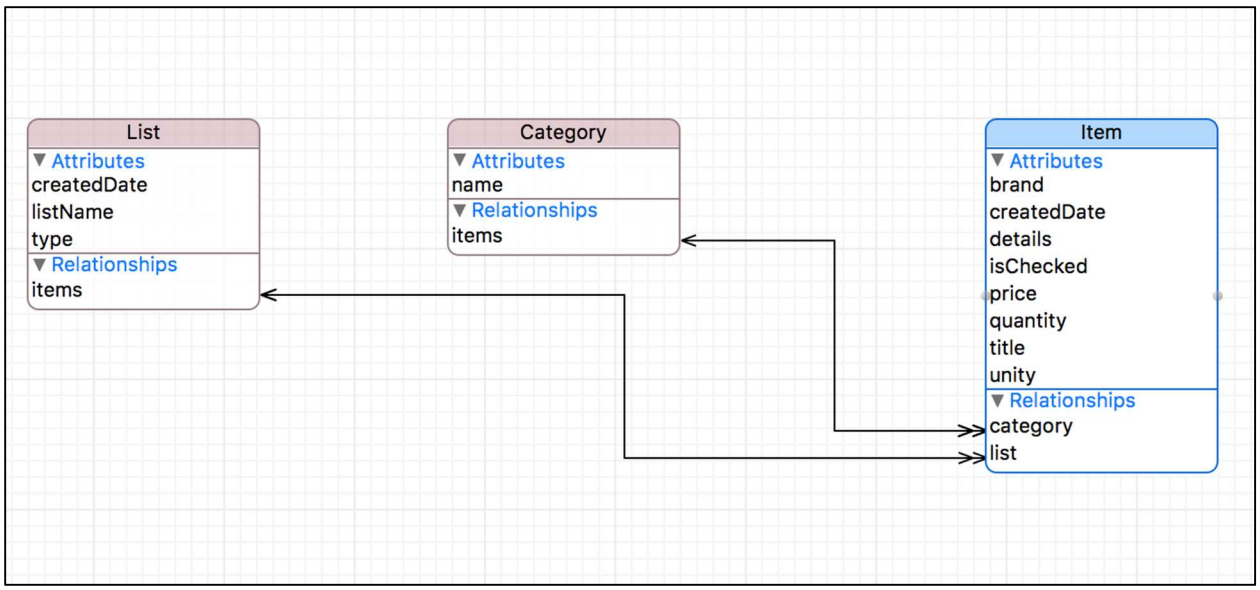


*Imagem 2 – Modelo de Dados*

Como podemos ver, a nossa aplicação necessita apenas de 3 entidades *Item*, *List*, e *Category*. Os nomes das entidades julgo que são autoexplicativos. É importante de referir as ligações entre elas, portanto, uma lista pode ter vários itens, uma categoria pode ter vários itens, mas um item tem apenas uma lista e uma categoria.

Depois de criados os objetos, toda a gestão de acessos à base de dados e gestão de concorrência é feita pelo Core Data, tirando assim trabalho ao programador de implementar essa gestão.

# 4. Interface

Para suportar correr em diferentes dispositivos com diferentes tamanhos de ecrã, tentou-se ao máximo usar partido de *constraints* para obrigar a representação dos objetos a ficarem ancorados uns nos outros. Fizeram-se alguns testes e o aspeto geral da aplicação manteve-se como esperado, pelo que considera-se que o objetivo foi atingido.

Para além disso, existem alguns tópicos que acho importante de referir e que serão abordados nos seguintes subcapítulos.

## Long Press

Para entrarmos no modo de edição de um item, o que se fez foi implementar um *Long Press*. Para o reconhecimento deste gesto, foi necessário adicionar o *Long Press Gesture Recognizer* à *view* da *ListViewController*. Este gesto de pressionar com um pouco mais de força na célula da *TableView* é depois capturada no método *longPressCheck* que pode ser visto no *ListViewController*. Quando o dispositivo deteta que o gesto terminou, chama a controlador *ItemDetialsViewController* e podemos entrar então no modo de edição do item. Em termos de usabilidade julgo que interessante a utilização deste tipo de interação, do que a colocação de um botão.

## Botões da *Navigation Bar*

Acho importante fazer referência aos botões da *Navigation Bar* da L*istViewController*, por ser o controlador que acabou por ter mais botões. Nos restantes controladores, existe apenas um botão de *Cancel* para voltar para trás (*ItemDetailsViewController*) e um botão de *Add* (*MainViewController*), estando a maioria dos botões concentrados na *ListViewController*, como se pode ver na imagem 6.

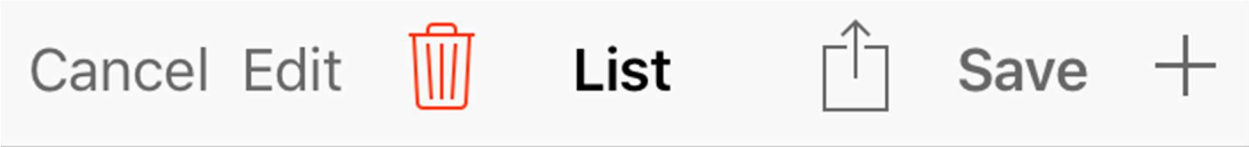


*Imagem 3 – Navigation Controller Buttons*

São então 6 botões :

- *Cancel*
  - Ao pressionar este botão, voltamos para a *MainViewController*.
- *Edit*
  - Ao pressionar este botão, abrimos o modo de edição e podemos apagar itens da lista de compras.
- *Garbage Can (Delete)*
  - Ao pressionar este botão, apagamos a presente lista das listas de compras.
- *Copy*
  - Este botão serve para copiar a presente lista, no entanto, de referir que qualquer item assinalado como comprado não manterá esse estado na nova lista criada.
- *Save*
  - Botão que permite salvar a lista de compras
- *Add*
  - Botão que permite abrir o *ItemDetailsViewController* no modo de edição.

Uma última nota, pois existem alguns botões que inicialmente estão desabilitados, e que apenas ficam habilitados a partir do momento em que temos uma lista de compras criada na base de dados.

## Custom Text Field

Explicando um pouco o que foi feito na customização do *Text Field*, podemos ver a imagem 4.


*Imagem 4 – Custom Text Field*

As alterações que foram feitas foi arredondar um pouco os cantos da Text Field, usando o seguinte comando :

- *self.layer.cornerRadius = 3.0*

Para além de definirmos o tipo de letra e a cor de fundo, acho importante de referir as funções *textRect* e *editingRect*, que servem simplesmente para dar alguma margem entre o início da *TextView* e o texto do *placeholder* ou o texto do que formos colocar na *TextView*.

# 5. Bónus

Foram feitos alguns desenvolvimentos que não estavam no enunciado do trabalho, no entanto considerou-se que se adequavam ao mesmo. Para além da implementação das categorias e de se poder agrupar e ordenar itens por esse atributo, implementou-se também um ordenamento e agrupamento por marca. Para além disto, podia-se ordenar os itens de uma lista por nome ou por data inserida. Utilizei o controlo *Segmented Control* para gerir a selecção dos diferentes ordenamentos e agrupamentos selecionados pelo utilizador.

Isso pode ser visto nas imagens 5,6, 7 e 8.

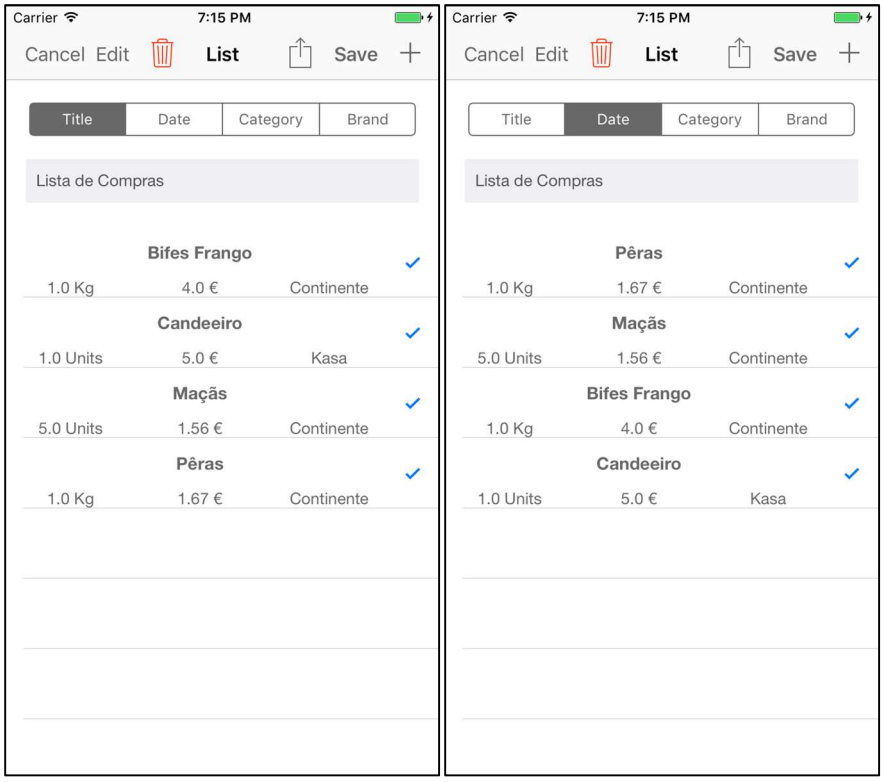


*Imagem 5 e 6 – Ordenamento por titulo e por data de inserção.*

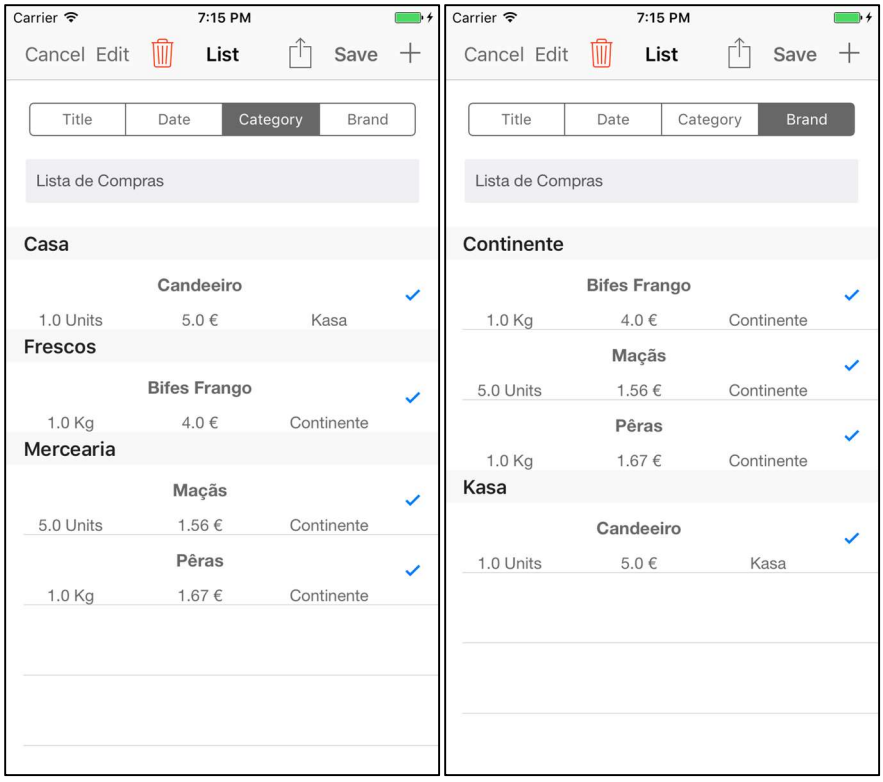


*Imagem 7 e 8 – Ordenamento e agrupamento por Categoria e Marca.*

Para além desta implementação e tendo em conta o requisito pedido de marcar os itens como "comprados" ou "*checked*", decidi implementar também secções na *listView* das listas de compras, e portanto quando todos os itens de uma lista estão marcadas, o atributo *type* passa a ter um valor diferente ("*Completed List*"), e será mostrada na *listView* das listas de compras numa secção diferente, que agrega todas as listas de compras já compradas guardadas na aplicação. Isso pode ser visto na imagem 9.

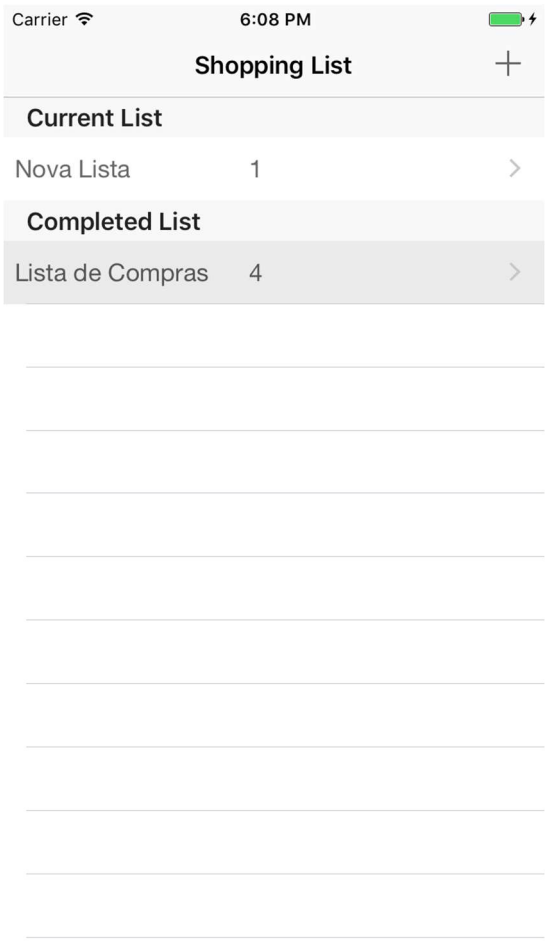


*Imagem 9 – Listas em Utilização e Listas Completadas*

**NOTA**: Caso uma destas listas seja copiada, os novos itens criados não irão estar marcados como comprados.