INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Estudo da Aplicação do Material Design em Computadores Pessoais: Desenvolvimento de um Media Player

GIL FILIPE MARTINS DE CASTRO
Outubro de 2016

POLITÉCNICO
DO PORTO

Estudo da Aplicação do Material Design em Computadores Pessoais
[Desenvolvimento de um Media Player]

Gil Castro

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Dr. João Pereira

Júri:
Presidente:
Dr. ..., DEI/ISEP

Vogais:
Dr. ..., DEI/ISEP
Dr. ..., DEI/ISEP

Porto, 23 de Outubro de 2016

Resumo

Na última década tem-se observado uma rápida evolução na forma como as pessoas interagem com interfaces gráficas de diversos tipos de dispositivo. Estes dispositivos têm chegado a pessoas que nunca ou raramente utilizaram os dispositivos mais habituais há uma década, aumentando a exigência dos utilizadores em geral. Assim surge a necessidade de desenvolver interfaces gráficas de uso eficaz e eficiente.

Nesta busca de interações humano-computador cada vez melhores também surgiu o desafio de criar um aspeto unificado entre todos os tipos de dispositivos. Empresas como a Google e a Microsoft lançaram as suas ideias para interfaces com boa usabilidade e que sejam consistentes. A importância dada por engenheiros de *software* às interfaces nem sempre foi suficiente, tendo nos últimos anos melhorado. A complexidade no desenvolvimento de *software* aumenta, a não ser que o sistema ou alguma camada intermédia abstraia algumas das preocupações necessárias a nível da interação com o utilizador.

O Material Design, criado pela Google, consiste numa linguagem de *design* com o objetivo de orientar equipas de desenvolvimento de *software* na criação da interface com o utilizador, seja qual for o dispositivo ou até mesmo o sistema operativo. A aplicação do Material Design em *software* para computador portáteis e de secretária é escassa, não se sabendo quão benéfica poderá ser. O projeto Papyros ambiciona desenvolver uma interface completa para computadores baseada em Material Design.

Neste projeto estudar-se-ão conceitos e trabalhos sobre usabilidade, o Material Design e tentar-se-ão obter conclusões acerca da sua aplicação em computadores. De forma a colaborar com a comunidade *open-source* e o projeto Papyros, desenvolver-se-á uma aplicação para este sistema. De forma a tirar conclusões, este mesmo *software* será testado por utilizadores com diferentes perfis, para além do confronto entre as orientações da documentação da Google com trabalhos desenvolvidos por peritos em usabilidade de sistemas.

Palavras-chave: interfaces gráficas, usabilidade, Material Design, *open-source*, multimédia, Linux

Abstract

Over the past decade we have seen the way people interact with graphical interfaces on many different types of devices evolve very quickly. These devices have reached those that barely interacted with devices that were commonly used a decade ago, increasing the demand for better interfaces. Consequently, developing effective and efficient graphical interfaces has become more important.

Throughout the search for better human-computer interactions, the challenges to develop a unified look-and-feel across devices became more apparent. Companies, like Google and Microsoft, launched their own ideas for consistent interfaces with good usability. While developers have not always given enough consideration for user interfaces, they have started showing improvements over the past few years. The complexity of developing software increases unless the system, or an intermediate layer, abstracts some of the concerns about designing the interaction with the user.

Material Design, created by Google, consists of a design language meant to guide software development teams on the creation of the user interface independently of the device or even the operating system. The application of Material Design in desktop and laptop software is scarce; we don't know how beneficial it can be. Papyros project aims to develop a complete interface for desktops and laptops based on Material Design.

In this project, we will study usability concepts and works, Material Design and try to reach conclusions about its application in desktops and laptops. In order to collaborate with the open-source community and the Papyros project, we will develop an application for this system. Conclusions will be reached by testing this software with users of different profiles beside the comparison between Material Design and the work made by experts in system usability.

Keywords: graphical interfaces, usability, Material Design, open-source, multimedia, Linux

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Acrónimos

dp	density-independent pixels.
FAB	Floating action button.
GUI	Graphical User Interface - Interface Gráfica com o Utilizador.
IANA	Internet Assigned Numbers Authority.
IDE	Integrated Development Environment.
IHC	Interação Humano-Computador.
JSON	JavaScript Object Notation.
MIME	Multipurpose Internet Mail Extensions.
QML	Qt Meta Language ou Qt Modeling Language.
sp	scaleable pixels.
SQL	Structured Query Language.
URI	Uniform Resource Identifier.
XML	Extensible Markup Language.

Capítulo 1

Introdução

1.1 Enquadramento

Este projeto faz parte da unidade curricular de Tese/Dissertação do Mestrado em Engenharia Informática, no ramo de Sistemas Gráficos e Multimédia do Instituto Superior de Engenharia do Porto (ISEP).

1.2 Contexto

Até recentemente, as equipas de desenvolvimento de software focavam-se principalmente no desenvolvimento de funcionalidades, preocupando-se pouco com a forma como o utilizador conseguiria chegar a essas funcionalidades. Esta negligência no desenvolvimento da interação do utilizador com o software resultaria em vários problemas, tais como:

- Falhas ligeiras como a necessidade de anulação/repetição de alguma tarefa;
- Falhas graves como a perda de dados ou até mesmo a morte;
- Dificuldades de compreensão;
- Esquecimento da sequência de uma tarefa;
- Funcionalidades dificilmente ou nunca encontradas.

No entanto, nos últimos anos têm-se visto várias mudanças e inovações nas interfaces com o utilizador e uma crescente exigência por parte destes. Para uma equipa de desenvolvimento de software, a criação de uma interface de qualidade enfrenta cada vez mais desafios. Geralmente estes desafios são facilmente superados com novos padrões de desenvolvimento e novas ferramentas mas não se garante que assim se desenham facilmente boas interfaces.

Uma das muitas iniciativas por várias empresas na melhoria das interfaces e do seu desenvolvimento foi a criação do Material Design pela Google (Equipa de Design da Google 2014). O Material Design consiste numa linguagem de design que orienta e aconselha durante o desenvolvimento de interfaces, dando alguma liberdade na estruturação e design da interface, permitindo assim que, mesmo duas aplicações que sigam esta linguagem, sejam distinguíveis. Apesar de ser feito com o objetivo de ser multiplataforma e suportar diferentes tamanhos de ecrã, ainda há poucas implementações deste design em computadores de secretária e portáteis e pouco se sabe sobre a sua usabilidade neste contexto.

1.3 Motivação

Existem vários projetos *open-source* com inúmeras finalidades, um projeto recente relevante neste assunto é o "Papyros". O Papyros é um novo ambiente gráfico para Linux que segue as linhas de orientação do Material Design (Spencer 2015).

Este projeto está aberto à colaboração e ainda se encontra numa fase inicial. Devido à relação entre ambos os assuntos, é possível estudar a aplicação do Material Design e, ao mesmo tempo, colaborar com o Papyros.

1.4 Objetivos

Este projeto tem os seguintes objetivos:

1. Pesquisar e analisar estudos existentes sobre usabilidade de interfaces com o utilizador

   Com esta análise pretende-se compreender melhor o que é a usabilidade de uma Graphical User Interface - Interface Gráfica com o Utilizador (GUI), que aspetos melhoram e pioram a sua usabilidade e de que forma poderão ser melhorados.

2. Estudar o conceito de Material Design da Google

   Ficar a conhecer melhor os conceitos mais importantes do Material Design com o objetivo de serem aplicados no trabalho a desenvolver.

3. Colaboração com o projeto *open-source* Papyros

   Esta contribuição poderá consistir na partilha de código que possa melhorar componentes do projeto ou mesmo com a criação do software desenvolvido, uma vez que tornará o ambiente gráfico mais completo.

4. Transpor o Material Design para um programa(s) desenvolvido(s) para computadores pessoais

   A interface gráfica do software desenvolvido deverá estar de acordo com a documentação oficial do Material Design, disponível em https://material.google.com/.

5. Criação de um leitor multimédia completo e integrado com o projeto Papyros, com suporte a diferentes plataformas

   O software desenvolvido deverá ter funcionalidades comuns em leitores multimédia completos, como por exemplo reproduzir músicas e vídeos e ter uma biblioteca multimédia. Este leitor deverá integrar-se corretamente com o Papyros e deverá ser desenvolvido com o objetivo de poder ser executado noutros sistemas, para tal, utilizar-se-á as ferramentas e bibliotecas do Qt, que também já são utilizadas pelo Papyros.

6. Avaliar a capacidade e facilidade de utilização da aplicação do Material Design em computadores pessoais

   Através das conclusões do ponto 1 e de análises de usabilidade ao trabalho desenvolvido, dever-se-á obter algumas conclusões relativamente à usabilidade do Material Design em computador pessoais.

1.5 Organização da Dissertação

Neste primeiro capítulo (1), o documento iniciar-se-á pela introdução ao mesmo e ao projeto.

No capítulo 2 será feito um levantamento de trabalhos relevantes ao projeto e será feita a introdução de alguns conceitos. Um dos assuntos mais importantes neste capítulo são os estudos a nível da interação com os computadores no qual se compreende melhor de que forma os seres humanos pensam e interagem com os computadores, tanto a nível do hardware como do software.

No capítulo 3 dar-se-á início ao projeto propriamente dito, planear-se-á as suas funcionalidades, arquitetura, design e de que forma(s) o produto deverá ser avaliado de forma a testar a usabilidade do Material Design.

Nos capítulos 4, 5 e 6 são documentadas as tarefas e assuntos mais relevantes do desenvolvimento do projeto, sendo o último composto pela análise da usabilidade do produto desenvolvido.

Por fim, o projeto encerra no capítulo 7 com as conclusões que visam contribuir para a concretização dos objetivos previamente apresentados, perspetivas de trabalho futuro.

Capítulo 2

Estado da Arte

Este projeto aborda vários assuntos distintos, portanto é importante conhecer melhor alguns trabalhos feitos nas áreas principais que compreendem este projeto. Também serão apresentados alguns leitores multimédia existentes, tanto *open-source* como *closed-source*, e a sua usabilidade será analisada. Neste capítulo serão apresentados alguns trabalhos feitos em algumas áreas tais como interfaces gráficas e multimédia.

2.1 Interação Humano-Computador

O estudo da Interação Humano-Computador (IHC), que emergiu em inícios dos anos 80 (Carroll 2013), envolve vários componentes tais como a ciência da computação, artes, design, ergonomia, psicologia, sociologia, semiótica, linguística, entre outros. Os estudos sobre a IHC focam-se em compreender de que formas as pessoas utilizam os computadores[1] com o objetivo de encontrar soluções para melhorar essa interação.

O utilizador comunica com o computador através dos dispositivos de entrada tais como o teclado, rato, ecrã tátil e microfone, essa comunicação será depois processada pelo computador e este pode responder ao utilizador através de dispositivos de saída tais como o monitor e as colunas de som. Tal como numa comunicação entre pessoas, poderá haver vários erros que levem o computador a fazer algo que o utilizador não estava à espera ou induzi-lo em erro.

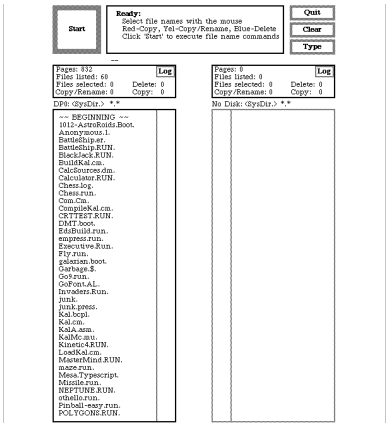
Existem várias dispositivos que possibilitam a comunicação entre o computador e o humano: luzes, colunas de som, ecrãs, etc. Nos dias de hoje, o ecrã tem um papel importante nessa comunicação apesar de não ter sido o primeiro dispositivo a ser utilizado. A forma como a comunicação é feita no ecrã com o utilizador sofreu várias alterações ao longo dos anos, sendo inicialmente bastante limitada, apresentando apenas texto.

Neste projeto, desenvolver-se-á um programa com interface gráfica. Este subcapítulo focar-se-á nas interfaces gráficas e irá apresentar como estas surgiram, alguns estudos que foram feitos nesta área e irá resumir alguns conceitos chave do Material Design, comparando-os a esses estudos.

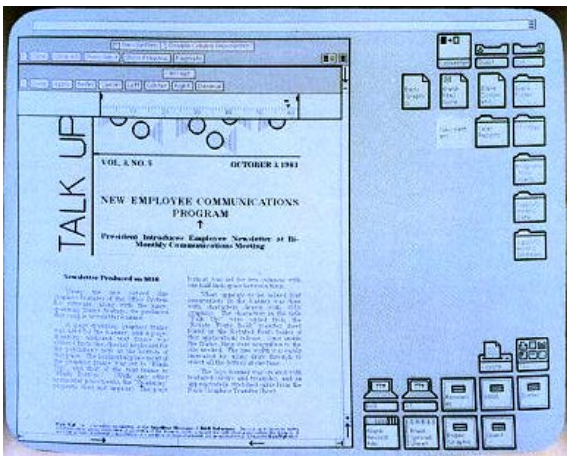
[1] Por "computador" entende-se vários dispositivos tais como computadores de secretária, computadores portáteis, telemóveis, sistemas embebidos, etc.

2.1.1 Interfaces Gráficas Iniciais

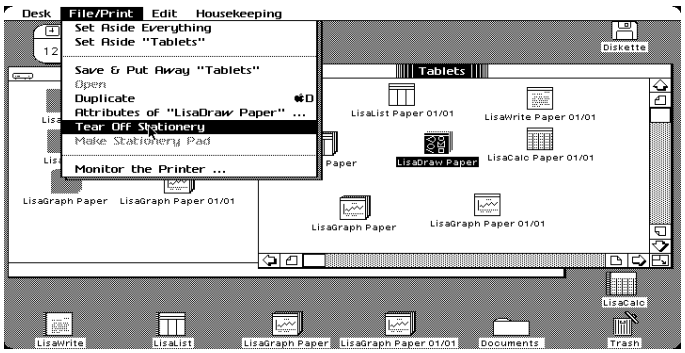
Hoje em dia, é habitual interagir-se com os computadores através de uma interface gráfica que recorre a diversos elementos gráficos para transmitir informação ao utilizador e também permitir ao mesmo comunicar com o computador. A primeira interface gráfica surgiu em 1973 pela Xerox (Lineback 2016), como é possível observar na figura 2.1a. Esta interface ainda se assemelha a uma interface de texto, comum na época, no entanto, já existiam botões e a interface era controlada com um rato. Em 1981, a Xerox lança o 8010 (Lineback 2016); na figura 2.1b, pode-se verificar que se assemelha às interfaces dos dias de hoje: existe um ambiente de trabalho e janelas. No ambiente de trabalho encontram-se ícones que transmitem mais informação para além de um nome, como por exemplo o seu conteúdo, algo que ressurgiu recentemente. Em 1983, a Apple lança o Apple Lisa (Lineback 2016) cuja interface gráfica resulta da continuação do trabalho desenvolvido pela Xerox (Akass 2001), na figura 2.1c pode-se verificar a adição da barra de menu e de menus drop-down, elementos também ainda usados nas interfaces atuais.



(a) Gestor de ficheiros do Xerox Alto (1973)



(b) GUI do Xerox 8010 (1981)



(c) GUI do Apple Lisa (1983)

Figura 2.1: Primeiras interfaces gráficas com o utilizador.

Estas interfaces gráficas introduziram vários componentes gráficos que ainda hoje fazem parte da IHC. No entanto a forma como são usados mudou ao longo dos tempos. Os nomes desses componentes podem variar dependendo da plataforma, de forma a introduzi-los no trabalho e a resolver possíveis ambiguidades, a lista seguinte descreve-os utilizando os nomes definidos pela especificação do Material Design.

1. **Button** Um botão deve comunicar claramente que ação ocorrerá quando o utilizador interagir com ele através de texto, uma imagem, ou ambos.
2. **Dialog** Uma caixa de diálogo contém texto e outros componentes focados numa tarefa específica informando o utilizador sobre algo importante ou pedir que o utilizador tome uma decisão.
3. **List** Uma lista apresenta múltiplos itens de forma contínua, geralmente dispostos verticalmente.
4. **Menu** Um menu permite ao utilizador executar uma ação, selecionando uma opção de uma lista.
5. **Progress bar/Activity circle** Informam ao utilizador o estado de uma operação.
6. **Selection control** Permitem ao utilizador selecionar opções. Há três tipos:
   (a) **Checkbox** Uma caixa de verificação permite escolher várias opções de um conjunto.
   (b) **Radio button** Um botão "de rádio" permite escolher uma única opção de um conjunto.
   (c) **Switch** Um interruptor que permite escolher entre dois estados (ligado/desligado). Apesar do seu funcionamento se assemelhar às checkboxes, este componente surgiu recentemente.
7. **Slider** Permite ao utilizador selecionar um valor de uma gama de valores. A natureza interativa do slider torna-o uma excelente opção para casos que reflitam níveis de intensidade tais como cores, brilho e volume.
8. **Tab**[2] Um separador facilita a exploração e a mudança entre diferentes vistas, aspetos funcionais ou a navegação em dados.
9. **Text field** Um campo de texto permite ao utilizador introduzir texto.

A metáfora da secretária, que surgiu no início de 1980 e consistia no uso de ficheiros e pastas apresentados como ícones e que podiam ser, e eram, espalhados pelo ecrã, tornou-se numa metáfora da "secretária desorganizada", tal como numa secretária no mundo real. Apesar de se poder dizer que a metáfora da secretária era superficial, ou talvez pouco explorada como um paradigma de design, captou a imaginação de designers. Surgiram novas possibilidades para a sociedade e a forma como se trabalha sofreu uma forte influência.

2.1.2 Shells

Uma **shell** é o sistema base da interface gráfica do utilizador e pode ser só de texto ou gráfica. Na perspetiva do utilizador, os programas executam dentro da *shell*. A *shell* contém elementos que permitem ao utilizador controlar o sistema e os programas que nele executem. Por exemplo, no caso da *shell* do Microsoft Windows, existe a barra de tarefas que permite ao utilizador gerir as janelas e iniciar programas. A *shell* também define um aspeto universal que os programas devem seguir para garantir a consistência.

[2] No Apple Lisa ainda não existia este conceito, sendo utilizados *radio buttons* em vez de *tabs*.

No caso do Linux, existem várias *shells* que podem ser instaladas e até mesmo partilhar componentes entre si. As *shells* mais habituais são o KDE[3], o Gnome, o Unity e várias baseadas no Gnome.

O **Papyros** é uma nova *shell open-source* para Linux com o objetivo de trazer o Material Design para computadores de secretária e portáteis[4] e desenhado para funcionar com as tecnologias mais recentes do Linux, contrariamente aos ambientes gráficos habitualmente utilizados.

Wayland vs. X

No Linux, o Wayland é a camada responsável por apresentar conteúdos no ecrã e enviar aos programas eventos de entrada tais como cliques, toques e o pressionar de teclas. O Wayland surgiu recentemente com o objetivo de substituir X, sistema responsável por executar essa mesma tarefa. (Høgsberg 2012)

O X tem um objetivo simples: os programas pedem-lhe para desenhar retângulos, texto e imagens. A sua arquitetura foi desenhada há mais de 30 anos, os casos de uso para uma interface gráfica eram menos exigentes que os dos dias de hoje: o suporte a tipos de letra era fraco; os programas não tinham grande controlo sobre os elementos desenhados pelo sistema; o gestor de janelas estava limitado. De forma a adaptar o X para os novos requisitos das interfaces gráficas, foram adicionadas cada vez mais camadas à sua arquitetura. Décadas depois, o X tem todos os componentes necessários para ser considerado um sistema operativo, apesar de não o ser.

De forma a resolver vários dos problemas do lado do gestor de janelas e das suas aplicações, estes passaram a desenhar toda a interface do seu lado numa imagem, enviando-a depois para o X, inutilizando dois terços dos casos de uso principais. O X também apresenta outros problemas em componentes bastantes comuns atualmente, tais como menus *popup*: cada *popup* pedia ao X para lhe enviar todos os eventos de entrada (para este poder fechar-se, por exemplo, quando o utilizador clica fora dele), impossibilitando que o resto do sistema os recebesse, tornando impossível ao utilizador controlar o resto do sistema (por exemplo, pausar música através de um atalho de teclado) e impedindo o protetor de ecrã de arrancar, visto que também precisaria de fazer o mesmo pedido ao X.

Outro problema com o X é o atraso gerado em todas as comunicações entre ele e os programas, para além de serem síncronas, bloqueando as interfaces de todos os programas quando qualquer um precisar de fazer qualquer alteração à sua interface.

Contrastando ao Wayland, o X tornou-se num sistema extremamente pesado e complexo apesar da sua funcionalidade base já nem ser utilizada nas interfaces atuais. O Wayland faz apenas o básico, delega as tarefas de gestão de janelas para o compositor e não bloqueia enquanto um programa envia o conteúdo da(s) sua(s) janela(s), permitindo que todos os programas executem as suas tarefas de desenho ao mesmo tempo. O compositor é o componente responsável por apresentar as interfaces de todas as aplicações - geralmente com um contorno à volta, representando uma janela - e receber e lhes retransmitir certos eventos como movimentos e cliques do rato e toques no ecrã.

[3] Recentemente, a *shell* do KDE passou-se a chamar Plasma e KDE refere-se à *shell* e ao conjunto de programas incluído com o mesmo.

[4] No entanto, o Papyros não tem quaisquer restrição para executar em outros dispositivos como telemóveis e *tablets*.

As limitações do X dificultariam a implementação do Material Design, para além dos vários problemas comuns a todas os ambientes gráficos atuais. Portanto a equipa de desenvolvimento do Papyros optou por utilizar o Wayland[5] em alternativa.

Qt

O Qt é uma *framework open-source* de aplicação multiplataforma que requer poucas ou mesmo nenhumas alterações para que o software funcione em diferentes sistemas operativos e de forma nativa (Meyer 2011). Tal como o KDE, o Papyros utiliza esta *framework* para toda a sua implementação.

O Qt fornece uma *toolkit*, ou seja, um conjunto de componentes base para as interfaces gráficas, como por exemplo os componentes listados na página 6 e seguinte. O Qt usa C++ com algumas funcionalidades extra que simplificam a implementação de software nesta linguagem de programação.

Recentemente, a equipa de desenvolvimento do Qt desenvolveu o Qt Meta Language ou Qt Modeling Language (QML), uma linguagem que permite facilmente implementar interfaces gráficas com pouca programação e com a possibilidade de utilizar JavaScript (Qt Company Ltd. 2016). O uso de C++ continua a ser possível, no entanto deixa de ser obrigatório para a implementação da interface. O Papyros utiliza QML para a implementação de grande parte da sua interface.

Para além das suas funcionalidades a nível gráfico, o Qt conta com outras classes para realizar outros tipos de tarefas tais como comunicação em rede, multimédia, interpretação de documentos Extensible Markup Language (XML) e JavaScript Object Notation (JSON), gestão de threads, etc.

2.1.3 Estudos

Foram realizados vários estudos que dão grandes contribuições para o desenho de interfaces eficientes, ou seja, interfaces que permitam executar as tarefas para as quais foram destinadas, exigindo o menor esforço possível do utilizador e que reduzam a probabilidade de erro. Uma boa interface é importante para que o software possa ser corretamente utilizado, até porque o utilizador geralmente apenas aprende a fazer o mínimo, não chegando a explorar todas as funcionalidades (Nielsen 2013). Neste subcapítulo, apresentam-se alguns desses estudos.

Donald Norman

Norman (2013), no seu livro "The Design of Everyday Things", cuja primeira versão foi publicada no ano de 1988, junta várias teorias, exemplos juntamente com as suas interpretações e várias dicas e princípios.

[5] No entanto, é possível executar os seus programas no X ou mesmo noutros sistemas operativos.

Norman diz que as duas características do bom design são:

Descoberta é possível descobrir que funções estão disponíveis e onde e como as executar,

Compreensão compreende-se o produto, a sua forma de utilização e o significado dos seus controlos.

Existem inúmeras deficiências na interação humano-máquina e Norman refere que isso se deve ao fato que a maioria dos designs são feitos por engenheiros com boas competências tecnológicas mas limitadas na compreensão das pessoas. Os engenheiros pensam "Nós mesmos somos pessoas, portanto compreendemos as pessoas." mas, na realidade, os humanos são incrivelmente complexos, os que não estudam o comportamento humano acreditam que este assunto é bastante simples. Os engenheiros cometem o erro de pensar que é suficiente a explicação que "se as pessoas lessem as instruções, tudo correria bem".

Engenheiros excelentes produzem experiências positivas. *Experiência*, palavra importante mas pouco apreciada pelos engenheiros devido à sua subjetividade. Quando a tecnologia se comporta de forma inesperada, ficamos confundidos, frustrados e até irritados, tudo emoções negativas. Em contraste, quando a tecnologia funciona como estamos à espera, cria-se uma sensação de controlo, mestria, satisfação ou até mesmo orgulho, tudo emoções positivas.

Quando se interage com um produto, é necessário compreender como trabalhar com ele. Isto significa **descobrir** o que faz, como funciona e que operações são possíveis. A **descoberta** resulta da aplicação correta dos cinco conceitos psicológicos fundamentais: *1*) *affordance*; *2*) *significador*; *3*) restrição; *4*) mapeamento; *5*) *feedback*. A esta lista também se deve incluir o **modelo concetual**.

Affordance *Affordances*[6] são as relações existentes entre as propriedades de um objeto e uma pessoa. As *affordances* podem depender de pessoa para pessoa ou de objeto para objeto, por exemplo: uma cadeira tem a *affordance* de (ou leia-se "serve para") sentar e também poderá ter a *affordance* de ser levantada, dependendo da força da pessoa ou do tipo de cadeira. Algumas *affordances* são percetíveis, outras não.

Significador Significadores são formas de indicar as *affordances*. Norman refere que os designers tendem a confundir estes dois termos. As *affordances* são as relações entre o objeto e a pessoa, os significadores são formas de indicar à pessoa essa relação. Há *affordances* que também são significadores. Os significadores devem ser sempre percetíveis, senão falham o seu propósito.

Tomando como exemplo uma porta que tem a *affordance* de puxar e empurrar, o próprio puxador poderá ser a *affordance* de puxar e também um significador. Já um texto a dizer "empurre" será apenas um significador.

Considerando um marcador de livros, objeto colocado num livro com o objetivo de significar o local em que uma pessoa ficou na leitura do mesmo. O marcador também tem, "acidentalmente" um outro significador: indicar quanto falta ler do livro.

[6] Palavra cuja tradução oficial para português não existe mas que poderá ser compreendida como "reconhecimento".

Restrição As restrições consistem em limitar as ações do utilizador. Quando feitas corretamente, podem guiar o utilizador a executar a tarefa pretendida e/ou ajudar a reduzir a taxa de erro. Há quatro classes de restrições: físicas, culturais, semânticas e lógicas.

Mapeamento O mapeamento refere-se ao posicionamento de controlos e a relação entre esse posicionamento e o(s) objeto(s) a ser(em) controlado(s). É um conceito importante no design e disposição dos componentes em interfaces gráficas. A relação entre um controlo e os seus resultados será mais fácil de aprender quanto mais compreensível for o mapeamento entre os controlos, as ações e o resultado pretendido.

Feedback *Feedback* é a comunicação de resultados de uma ação. Deve ser imediato, mesmo um atraso de um décimo de segundo pode causar transtornos ao utilizador.

O *feedback* deve ser planeado, todas as ações devem ser confirmadas mas de forma a não interromper o que o utilizador estiver a fazer. Deverá haver diferentes prioridades, dando mais ênfase ao *feedback* mais importante. Também se deve ter atenção à quantidade de *feedback*, visto que se pode tornar irritante para o utilizador. Todo o *feedback* deve ser claro, se o utilizador não o interpretar corretamente tornar-se-á inútil ou poderá induzir o utilizador em erro.

Se uma pessoa fizer algo na expetativa de ocorrer alguma reação (*feedback*) e nada acontecer, a pessoa interpretará essa falta de *feedback* como uma indicação de que fez mal o que queria fazer e consequentemente, a pessoa irá tentar de novo. Esta situação é bastante comum na interação com computadores.

Modelo Concetual O modelo concetual é uma explicação, geralmente bastante simplificada, sobre como algo funciona. Os ficheiros, pastas e ícones no ecrã de um computador ajudam as pessoas a criar o modelo concetual de documentos dentro de pastas que estão dentro do computador. Na realidade não existem pastas dentro do computador, no entanto este é um conceito eficaz que facilita o seu uso.

Também se podem criar conceitos errados, como por exemplo páginas *web* num computador, poder-se-á ter a ideia de que o *website* está no nosso computador, quando na realidade está "na nuvem". Se a ligação de rede for interrompida, os resultados poderão ser confusos para o utilizador.

Os modelos concetuais encontrados em manuais podem ser detalhados e complexos. Os modelos importantes no design são mais simples: eles residem na mente das pessoas que utilizarem o produto, chamam-se "modelos mentais". Cada pessoa poderá criar modelos mentais diferentes e até mesmo contraditórios.

Norman também apresenta os seguintes três níveis de processamento, fazendo todos parte das ações das pessoas (Norman 2013):

Refletivo Todo o conhecimento consciente reside neste nível. É neste nível que se desenvolvem compreensões aprofundadas, onde o raciocínio e a tomada de decisão consciente acontece. Os outros dois níveis funcionam de forma inconsciente e portanto, são rápidos mas passam por pouca ou nenhuma análise. Este nível é mais avançado e

lento. Muito útil no fim da tomada de ações pois aqui se avalia os resultados e se tiram conclusões. Os níveis mais elevados de emoção têm origem no nível refletivo, a emoção e o conhecimento estão fortemente interligados. As emoções a este nível são mais prolongadas e poderão influenciar a fase final da tomada de decisão, e consequentemente causar alterações positivas ou negativas da próxima vez que o utilizador decidir executar a mesma ação.

Comportamental Neste nível entram as competências adquiridas pela pessoa. Executam-se ações inconscientemente, no entanto, a pessoa sabe que as está a fazer e pode controlá-las parcialmente apesar de não conseguir saber facilmente os detalhes. Conversar normalmente é um bom exemplo: falamos por própria vontade, no entanto formamos as palavras sem pensar e, por vezes, só sabemos o que vamos dizer só depois de nos ouvirmos falar. Quando queremos executar uma ação bem aprendida, só temos de pensar no objetivo e este nível trata de todos os detalhes, escondendo ao nosso lado consciente grande parte do que se passa exceto a criação do objetivo. Uma boa forma de verificar o funcionamento a este nível consiste em pegar num copo com uma mão e usar a mesma mão para pegar noutros objetos ao mesmo tempo, inconscientemente todos os dedos se ajustam de forma a suportar todos os objetos.

Visceral Nível também existente nos outros animais. Este nível permite responder rapidamente e subconscientemente, é parte do mecanismo básico de proteção do ser humano e permite fazer julgamentos rápidos do ambiente: bom/mau, seguro/perigoso. A este nível de inconsciência, é possível afetar emoções básicas das pessoas e é aqui que o aspeto gráfico das interfaces poderá ter um fator determinante na criação de um sentimento de repulsa ou atração.

As interfaces devem ser desenhadas com atenção estes três níveis, especialmente ao nível refletivo. O nível comportamental é considerado o nível da interação e no qual se criam emoções baseadas em expetativas.

Os três níveis funcionam em conjunto: O nível alto, o refletivo, pode afetar emoções de baixo nível e as emoções de baixo nível podem afetar o nível refletivo. A tabela 2.1 apresenta uma comparação entre a forma como as ações são executadas ao nível subconsciente e consciente.

Tabela 2.1: Comparação entre ações subconscientes e conscientes

Subconsciente	**Consciente**
Rápido	Lento
Automático	Controlado
Vários recursos	Recursos limitados
Controla comportamentos já adquiridos	Utilizado em situações extraordinárias: durante aprendizagens, quando em perigo, quando alguma coisa não corre como esperado

A maioria dos produtos não causa medo nem vontade de fugir mas um mau design pode induzir frustração, raiva, desespero ou até mesmo ódio. Contrariamente, um bom design pode criar uma sensação de orgulho, bem-estar e controlo ou até mesmo afeto.

A Estrutura da Memória Norman diz que os psicólogos distinguem duas classes principais de memória (Norman 2013):

Memória de curto prazo / memória de trabalho Retém as experiências mais recentes ou o assunto atualmente a ser pensado. A informação é retida automaticamente e acedida sem esforço mas a quantidade de informação que pode ser retida é extremamente limitada. Esta memória é indispensável para a execução das tarefas do dia-a-dia, no entanto, qualquer distração poderá ser suficiente para se perder ou corromper a informação retida. Em média, esta memória é capaz de reter entre 5 a 7 itens, no entanto, deve-se assumir que o intervalo varia entre 3 e 5.

Memória de longo prazo É a memória do passado, a informação demora a ser registada nesta memória e acedê-la requer esforço. Acredita-se que dormir é fundamental para reforçar as memórias de cada dia. Não se sabe com alguma exatidão a quantidade de espaço disponível nesta memória mas pode-se assumir que praticamente não tem limite. No entanto, a informação nesta memória é menos fidedigna, uma vez que a interpretação usada aquando do registo poderá ser diferente daquela aquando do acesso à mesma.

O conhecimento necessário para uma pessoa executar uma dada ação poderá estar na sua memória ou no ambiente com o qual está a interagir. A tabela 2.2 compara ambas as possibilidades.

Tabela 2.2: Comparação entre a memória externa (ao utilizador) e a memória interna

Memória Interna	**Memória Externa**
Informação fácil de aceder e sempre acessível, desde que percetível	Informação na memória de curto prazo sempre acessível, caso esteja na de longo prazo será necessário esforço.
Interpretação substitui a aprendizagem. A facilidade de interpretação depende do design.	Requer aprendizagem que, apesar de ser um aspeto negativo, poderá ser facilitada se houver um bom modelo concetual.
Atrasado pela necessidade de encontrar e interpretar o conhecimento	Pode ser eficiente, especialmente se já estiver bem assimilada pelo utilizador, sendo as ações feitas a nível subconsciente.
Facilidade de utilização ao primeiro encontro	Dificuldade de utilização ao primeiro encontro
Pode ser pouco elegante, especialmente se for necessário manter muito conhecimento. Pode apresentar muita desarrumação.	Nada tem de estar visível, dando mais liberdade ao desenho da interface, resultando num aspeto mais organizado e agradável.

Erros A maioria dos acidentes industriais são causados por erro humano: entre 75 a 95% (Norman 2013). Como é que é possível haver tanta gente incompetente? A resposta é simples: não são. São falhas de design.

Quando acontece um erro, este pode levar a perdas financeiras, ou pior: ferimentos ou até mesmo a morte. E, na maioria das situações, as pessoas são sempre consideradas as

culpadas. O passo seguinte é puni-las com uma multa, despedi-las, ou pior. Quando há uma falha na qual se culpa, por exemplo, a arquitetura de um edifício que ruiu, analisa-se o incidente para encontrar as causas e não repetir o erro. Já quando a falha é humana, nada se faz exceto culpar e punir quem errou. Norman lista dois tipos de erro:

Lapso Os lapsos ocorrem quando o nosso objetivo está correto, mas enganamo-nos a executar as ações para atingi-lo. Exemplo: ter o objetivo de escrever *We* mas, em vez de carregar em $Shift + W$ para escrever o *W*, carrega-se em $Ctrl + W$ que pode fechar o separador onde se está a trabalhar.

Engano Os enganos acontecem quando se tem o objetivo ou plano errado. Nesta situação, mesmo se as ações forem executadas corretamente, elas serão parte do erro. Exemplo: ter o objetivo de fechar um separador mas, por exemplo, devido a uma grande quantidade de separadores, fechar o errado por engano.

Ambos os tipos de erro podem-se originar a partir de falhas de memória. Já os enganos podem ocorrer devido a falhas de conhecimento. Os enganos ocorrem a nível consciente, os lapsos a nível subconsciente.

Os seguintes pontos devem ser tomados em conta na implementação de interfaces:

- Compreender as causas do erro e planear a interface de forma a minimizar essas causas;
- Verificar se uma dada ação é executada de forma esperada pelas pessoas;
- Permitir reverter a ação - anulá-la - ou complicar a execução de tarefas que não possam ser revertidas;
- Facilitar a compreensão e descoberta de erros que ocorram e facilitar a sua correção;
- Não tratar a ação como um erro, em vez disso, ajudar o utilizador a completar a ação corretamente. Assumir que a ação está parcialmente correta.

Multitasking, ou seja, executar várias tarefas ao mesmo tempo, aparenta ser uma forma eficiente de trabalhar, no entanto está provado que isso não é verdade. O desempenho é mais baixo, aumenta-se a probabilidade de erros e a afeta-se a qualidade do resultado final.

Wickens, Gordon e Liu

Wickens, Gordon e Liu (2004) definiram 13 princípios de desenho de ecrãs no livro *An Introduction to Human Factors Engineering*. Estes princípios podem ser usados com o objetivo de reduzir erros e tempos de adaptação e aumentar a eficácia e a satisfação do utilizador. No entanto, estes princípios devem ser adaptados ao contexto em que forem utilizados. Estes são os princípios:

Perceção

1. **Legibilidade** Princípio crítico para o desenho de um ecrã que possa ser utilizado
2. **Evitar limites baseados em suposições** Não perguntar ao utilizador o nível de algo oferecendo-lhe opções limitadas.
3. **Processamento de cima para baixo (top-down)** A informação deve ser transmitida ao utilizador na ordem esperada por ele. Se for utilizada uma ordem contrária, então serão necessárias mais formas de lhe dar esse feedback.

4. **Redundância** Se uma informação é apresentada mais de uma vez, é mais provável que esta seja compreendida corretamente. Redundância não implica repetição, a mesma informação pode ser transmitida de várias formas diferentes.

5. **Similaridade cria confusão** Informações que pareçam semelhantes têm maior probabilidade de criar confusão. Deve-se eliminar semelhanças e as funções oposta devem estar corretamente diferenciadas.

Modelo mental

6. **Realismo pictorial** A apresentação deverá parecer-se com a variável que esta representa. Caso haja vários elementos, estes devem ser configurados de forma a assemelhar-se ao ambiente que representam.

7. **Princípio do elemento dinâmico** Elementos que alterem a sua posição ou aspeto devem corresponder àqueles que o utilizador espera.

Atenção

8. **Minimizar o custo de acesso a informação** Há um custo de tempo e esforço associado quando a atenção do utilizador é encaminhada de uma localização para outra para aceder a uma determinada informação. O ecrã deve minimizar este custo, dando acesso a informação frequentemente acedida colocando-a o mais perto possível[7].

9. **Compatibilidade de proximidade** Ter atenção a duas fontes de informação ao mesmo tempo pode ser necessário para concluir uma tarefa. Estas fontes devem estar mentalmente próximas e integradas, colocando-as próximas uma da outra, coloração comum entre ambas, padrões, formas, etc. No entanto, deve-se ter em atenção evitar o excesso de informação.

10. **Recursos múltiplos** O utilizador pode, por exemplo, recorrer a vários sentidos para processar informação mais facilmente em vez de sobrecarregar um dos sentidos.

Memória

11. **Substituir a memória por informação visual** O utilizador não deve precisar de reter informação temporariamente ou fazer o esforço de obter informação à sua memória de longo-prazo. A utilização de, por exemplo, listas de opções pode ajudar o utilizador sobrecarregando menos a sua memória. No entanto, há situações em que o uso da sua memória pode acelerar a execução de tarefas, o uso de teclas de atalhos é um exemplo disso.

12. **Ajuda baseada na previsão** Ações proativas são mais eficazes que ações reativas.

13. **Consistência** Hábitos vindos de outros ecrãs serão facilmente transferidos para suportar o processamento de novos ecrãs se estes forem desenhos de forma consistente com os anteriores.

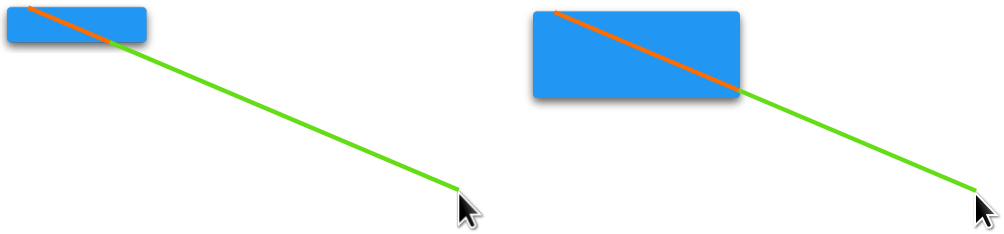
[7] Não se deve sacrificar o princípio 1 para reduzir este custo.

Lei de Fitts

Apesar de ter surgido ainda antes das primeiras interfaces gráficas, a lei de Fitts assume um papel importante no campo da IHC. Esta lei apresenta a seguinte fórmula (Fitts 1954):

$$T = a + b \cdot log_2 \left(\frac{D}{W} + 1 \right) \tag{2.1}$$

Onde T é o tempo necessário para chegar a um alvo que está a D unidades de distância e W é a largura desse mesmo alvo, medida no eixo do movimento, a e b são constantes empíricas. Interpretando a fórmula, deduz-se que o tempo para chegar a esse alvo se reduz aumentando a área da mesma e reduzindo a distância. Ao aumentar a área, reduz-se a probabilidade de erro que pode ocorrer caso o movimento termine antes ou depois do centro da mesma. A sua aplicação é útil na interação através do rato ou do toque, a figura 2.2 ilustra a sua aplicação na interação de um rato com um botão com áreas diferentes.



(a) Área pequena. (b) Área grande.

Figura 2.2: Ilustração variáveis da lei de Fitts: laranja = W, verde = D.

Quando o utilizador arrasta o rato para uma das bordas do ecrã, o mesmo não passará das bordas. Se o utilizador quiser apontar o rato para um objeto situado nas bordas terá menores probabilidades de errar porque o rato parará ao chegar à borda, permanecendo na área desejada. É possível assumir que W é infinito quando a área se encontra numa das bordas, reduzindo imenso o tempo de interação e, consequentemente, aumentando a eficiência. Nos cantos o efeito será maior visto que o rato fica limitado em ambos os eixos. (Atwood 2006)

Esta lei é mais famosa em comparações entre o macOS e o Windows (Tognazzini 1999) relativamente à barra de menus. O facto que a barra se encontra no topo do ecrã no macOS faz com que, segundo esta lei, a mesma tenha altura infinita porque os utilizadores apenas têm de "atirar" o cursor para o topo do ecrã e ele parará sempre na barra. Já no Windows, o utilizador terá de mover o rato com mais cuidado para conseguir pará-lo na barra de menus. Recentemente, a Microsoft lançou uma nova versão do Visual Studio que inclui a barra de menus na própria barra de título da janela, o que significa que esta fica no topo do ecrã quando a janela estiver maximizada, mas a mesma está espaçada um pixel em relação ao topo do ecrã, o que anula por completo a vantagem demonstrada pela lei de Fitts.

2.1.4 Testes de Usabilidade

Os testes de usabilidade são uma técnica usada no desenvolvimento de interações focadas no utilizador para avaliar um produto, testando-o com os próprios utilizadores. Esta técnica pode ser vista como insubstituível, uma vez que se recebe *feedback* direto sobre como os utilizadores usam o sistema. (Nielsen 1993)

Os testes de usabilidade focam-se na medição da capacidade dos utilizadores de um determinado produto usarem-no da forma para o qual este se destina. Estes testes são uma boa ferramenta de avaliação de interfaces.

Preparar um teste de usabilidade envolve criar um cenário realístico no qual a pessoa executará uma lista de tarefas com o produto a ser testado. A execução das tarefas deverá ser observada pelo avaliador. Mas também se podem realizar estes testes de outras formas:

Teste de corredor É um método rápido e barato no qual se seleciona pessoas ao acaso (por exemplo, como o nome indica: pessoas a passar num corredor) e pedir-lhes para experimentar o produto. Este método é bastante útil nas fases inicias do projeto, qualquer um pode testar exceto os engenheiros e designers.

Análise de perito Como o nome sugere, requer peritos na área para avaliar o produto. Geralmente, neste método faz-se uma avaliação heurística (Nielsen 1994) composta pelos seguintes fatores:

- Visibilidade do estado do sistema;
- Relação entre o sistema e o mundo real;
- Controlo e liberdade dos utilizadores;
- Consistência e *standards*;
- Prevenção de erros;
- Reconhecimento em vez de se recorrer à memória de longo prazo;
- Flexibilidade e uso eficiente;
- Design estético e minimalista;
- Ajuda o utilizador a reconhecer, diagnosticar e recuperar de erros;
- Ajuda e documentação.

Teste A/B Já habitual no desenvolvimento web, consiste na utilização de designs diferentes do mesmo produto, cada grupo testa um design diferentes. Os resultados são obtidos através do registo das ações dos utilizadores e não da observação direta ou da opinião do utilizador.

Segundo Jackob Nielsen, bastam cinco utilizadores para testar a usabilidade. Também afirma que se, durante os testes, houver dois ou três utilizadores completamente perdidos ao utilizar o produto, então é escusado fazer mais utilizadores passar pelo mesmo. (Nielsen e Landauer 1993)

2.1.5 Material Design

Nos últimos anos têm surgido computadores de tamanhos e formas bem variados, desde os habituais computadores de secretária e portáteis até aos relógios e óculos, passando pelos telemóveis e *tablets*, entre outros. Há cada vez mais utilizadores de mais de um deste tipo de dispositivos e esperam ver o software que utilizam funcionar corretamente em todos os seus dispositivos.

As equipas de desenvolvimento de software não poderão apenas ter mais código de funcionalidades para manter como também precisam de desenhar interfaces gráficas diferentes para os vários dispositivos. Nessa necessidade surgem vários requisitos, principalmente a consistência entre todos os ecrãs.

Várias equipas tentaram solucionar este problema com novos padrões e guias, como por exemplo o Modern UI (Microsoft 2012) e o Material Design (Equipa de Design da Google 2014). No entanto, o Material Design não foi desenhado com o único objetivo de definir a imagem de marca da Google, mas sim com o objetivo de uniformizar o design de interfaces gráficas em todas as plataformas recorrendo à utilização dos princípios clássicos do bom design juntando padrões modernos (tendo alguns surgido noutras plataformas recentemente) e incluindo o toque como uma das informações de entrada principais que inclui também o rato, teclado e voz. Apesar de fornecer regras e padrões para o desenvolvimento de interfaces gráficas, o Material Design dá espaço à inovação através de regras pouco restritas.

O Material Design apresenta os seguintes três princípios:

Material é a metáfora Baseado na realidade tátil e nas leis da física, inspirado no estudo do papel e da tinta mas tecnologicamente avançado e aberto à imaginação. As superfícies e arestas dos materiais fornecem pistas visuais de que estes fazem parte da realidade. A flexibilidade do uso desta metáfora em computadores permite que os componentes ultrapassem os limites do mundo real, sem contrariar as leis da física.

Forte, gráfico e intencional Os alicerces do design impresso (tipografia, grelhas, espaço, escala, cor e imagens) guiam o tratamento visual. Estes elementos fazem mais do que apenas agradar à vista, criam hierarquia, significado e realce. Cores corretamente escolhidas, imagens sem margens, letras grandes e espaçamentos criam uma interface forte e gráfica que imerge o utilizador na experiência. A ênfase nas ações do utilizador tornam a funcionalidade principal imediatamente visível e fornecer um ponto de início para o utilizador.

A cinética traz significado O movimento respeita e reforça o utilizador como o principal controlador do sistema. São as ações do utilizador que despoletam reações em cadeia de movimento, podendo transformar toda interface. Todas as ações passam-se num único ambiente, os objetos são apresentados sem quebrar a continuidade da experiência mesmo quando o ambiente se transforma e reorganiza. Os movimentos são apropriados e transmitem informação, servindo como forma de canalizar a atenção e manter a continuidade. O feedback é subtil mas claro, as transições são eficientes mas coerentes.

As interfaces gráficas encontram-se num espaço tridimensional com luzes pontuais e luz de ambiente. As sombras refletem a posição no eixo z, perpendicular ao plano do ecrã. A colocação de folhas de material em diferentes posições na coordenada z transmite informação extra e dá mais espaço ao movimento.

Uma folha de material apresenta-se como uma versão evoluída de uma folha de papel, tendo várias das suas propriedades mas aproveitando as capacidades do ambiente virtual ganhando novas capacidades físicas sem quebrar as mesmas leis. Todas as folhas têm a mesma espessura e, tal como as leis da física o exigem, múltiplas folhas não podem ocupar o mesmo espaço simultaneamente. As folhas podem ter diferentes tipos de conteúdo, tais como texto, imagens, vídeos ou uma combinação e podem alterar a sua forma, desde que não alterem a sua espessura. Esta regra impede que uma folha possa ser dobrada, no entanto podem-se separar em duas ou mais e mesmo reunirem-se posteriormente. As limitações relativamente ao eixo perpendicular ao ecrã (espessura e a impossibilidade de dobrar) devem-se ao facto de que todo o ambiente se passa dentro do ecrã e a interface deve respeitar essa limitação. Na maioria dos casos, o movimento neste eixo resulta da ação do utilizador. A figura 2.3 apresenta duas folhas de material.

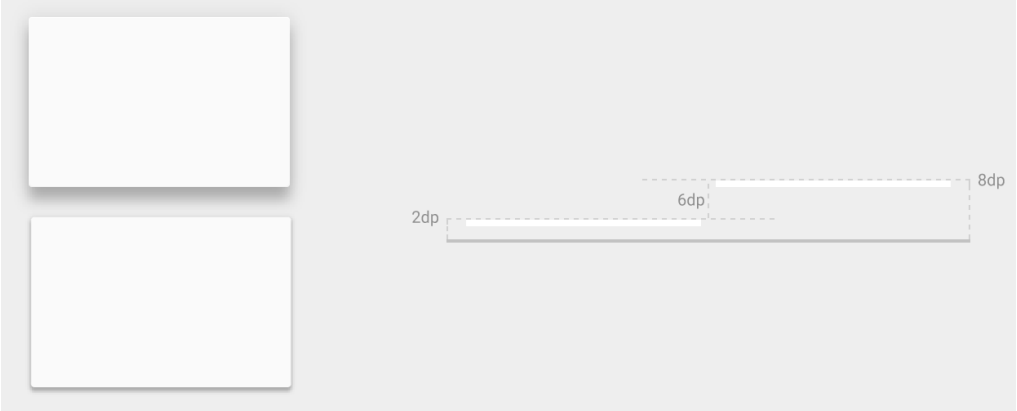


Figura 2.3: Folhas de material a alturas diferentes. O lado esquerdo apresenta a visualização real da interface, o lado direito representa as diferentes alturas.

Disposição dos componentes (Layout)

As linhas de orientação do Material Design também definem de que forma o conceito da folha de material pode ser utilizado para estruturar as interfaces gráficas. Existem dois princípios:

Costura Duas folhas podem ver as suas arestas juntas através de uma costura e, consequentemente, mover-se-ão em conjunto. (*Figura 2.4a*)

Degrau Uma folha que se sobreponha a outra (ou seja, posições no eixo z diferentes) forma um degrau, movimentando-se independentemente uma da outra. (*Figura 2.4b*)

(a) Costura. (b) Degrau.

Figura 2.4: Princípios da disposição de componentes.

O número de píxeis que constituem uma polegada é referido como "**densidade de píxeis**". Diferentes dispositivos têm diferentes densidades de píxeis, desenhar uma interface na qual os componentes têm medidas indicadas em número de píxeis resultaria em discrepâncias de dimensões enormes, incluindo em diferentes ecrãs do mesmo tipo de dispositivo. Para solucionar esse problema utilizam-se **density-independent pixels (dp)**, que significa "píxeis independentes da densidade", assim todos os componentes apresentam as mesmas dimensões independentemente da sua densidade de píxeis.

Outra unidade utilizada no Material Design é o **scaleable pixels (sp)** que significa "píxeis escaláveis" que funcionam de forma semelhante aos dp, no entanto destinam-se aos tipos de letra, esta unidade tem em conta a preferência do tamanho das letras, definido pelo utilizador. Assim, o mesmo texto terá o mesmo tamanho em diferentes ecrãs, a não ser que o utilizador tenha decidido aumentar ou diminuir o seu tamanho.

Todos os componentes são colocados numa grelha com células de 8dp de largura e altura podendo-se, em alguns casos, usar o valor intermédio de 4dp. Ou seja, todas as posições e dimensões serão múltiplos de 8dp. As linhas de orientação também especificam as dimensões e espaçamentos corretos para vários dos seus componentes.

Uma outra forma de atribuir dimensões corretas aos elementos baseia-se nas suas proporções sendo 16:9, 3:2, 4:3, 1:1, 3:4 e 2:3 as proporções recomendadas. Também é possível dimensionar os elementos tendo em conta a dimensão por incrementos que significa que a dimensão de um elemento será x vezes as dimensões de um outro elemento mais importante.

O Material Design define algumas regiões da interface principais, que não têm necessariamente de ser utilizadas em qualquer implementação mas que uniformizam e orientam a organização da interface:

Barra de navegação disponível do lado esquerdo, permite ao utilizador navegar no software

Barra da aplicação / barra de ferramentas principal dá acesso ações

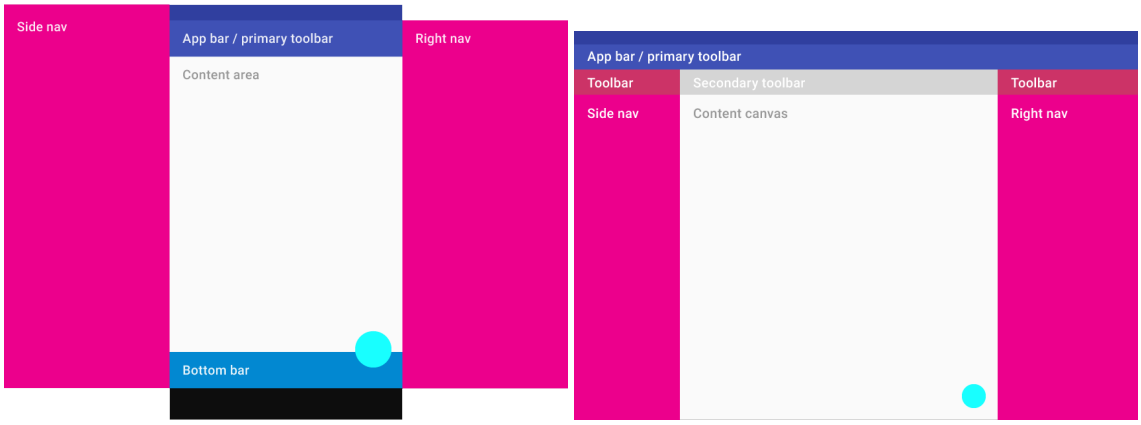
Área de conteúdo local onde a informação principal se encontrará

Barra de lateral extra raramente utilizada, disponível no lado oposto à barra de navegação

Barra inferior localizada no fundo do ecrã

A barra de navegação e a barra lateral extra encontram-se sempre acima das outras áreas e estão escondidas, exceto quando o ecrã é suficientemente grande. Nas culturas cuja a escrita é feita do lado direito para o esquerdo, estas duas barras invertem as suas posições. Todas as regiões são folhas de material distintas. A figura 2.5 ilustra essas regiões em dois dispositivos diferentes.

Poderá haver uma divisão horizontal ou vertical na área de conteúdo que poder-se-á estender à barra da aplicação. Esta divisão poderá ser feita através de uma costura ou um degrau. No entanto, não se pode criar hierarquias de divisões visto que podem complicar a disposição dos elementos e sobrecarregar a compreensão para o utilizador.



(a) Telemóvel. (b) Computador de secretária / portátil.

Figura 2.5: Regiões habituais de uma interface em Material Design.

Apesar de existirem estas regiões, é possível transpô-las, por exemplo com uma folha de material colocada no eixo z acima dos outros elementos. As linhas de orientação também descrevem formas de dispor corretamente os componentes dependendo das dimensões do ecrã de uma forma semelhante aos padrões surgidos recentemente no *design web*, conhecido como "responsive UI".

Componentes

Tal como outros designs, existem vários componentes comuns ao Material Design, no entanto alguns sofreram algumas alterações. Esta secção destina-se a apresentar apenas alguns desses componentes.

Card *Em Português: cartão.* Basicamente, consiste numa folha de material. Utilizado como contentor de outros elementos e geralmente em listas. Os cartões são usados como um ponto de entrada para informação mais detalhada e referem-se a uma informação específica. Geralmente, é possível deslizá-los para os lados de forma a removê-los da lista, executando a ação esperada pelo utilizador. A figura 2.6a apresenta um cartão com uma imagem, texto e ações.

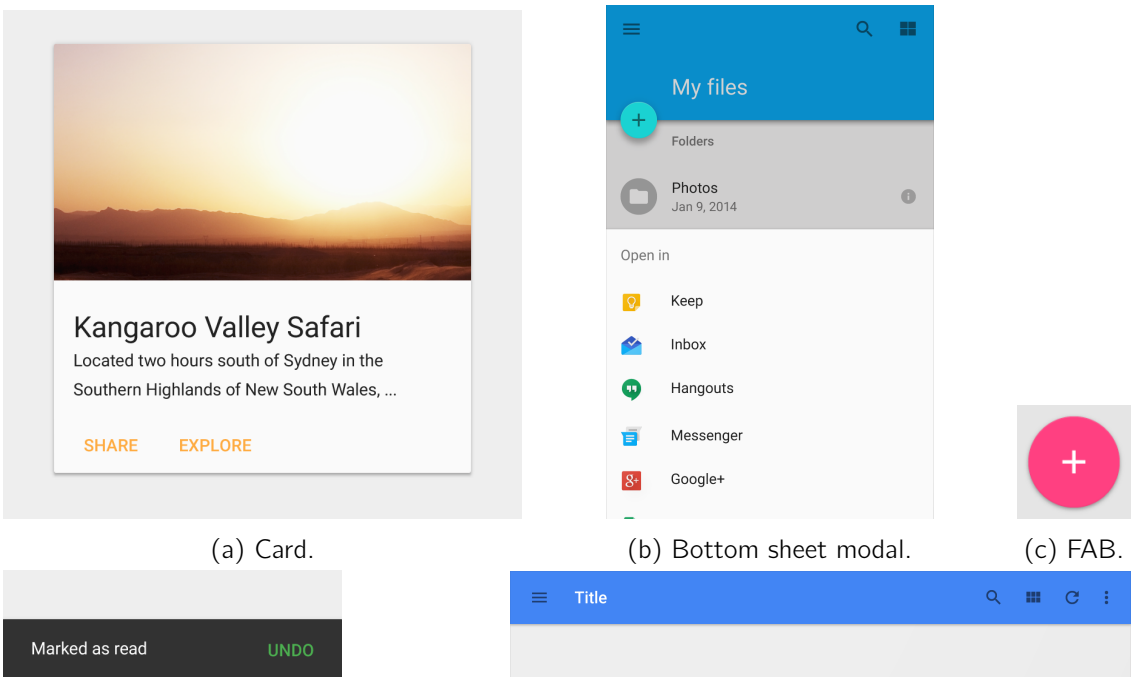
Bottom sheet *Em Português: folha inferior.* Uma folha de material que desliza do fundo do ecrã para cima, aparecem apenas como resultado de uma ação de um utilizador e podem ser deslizadas para cima para apresentar conteúdo adicional. Podem ser usadas como uma superfície temporária modal[8] ou um elemento persistente estrutural da interface. A figura 2.6b exemplifica o seu uso.

Floating action button (FAB) *Em Português: botão de ação flutuante.* Um botão redondo com um ícone e cor forte, facilmente distinguível de toda a interface. Apenas poderá existir um único FAB na interface do programa. O FAB dá acesso rápido à ação mais provável de ser executada, para além de dar uma pista a um utilizador novato devido à sua elevada visibilidade. Os FAB são colocados a uma altura superior ao conteúdo principal e não deslizam juntamente com o conteúdo de listas. A figura 2.6c apresenta um FAB, na figura 2.6b também é possível observar este componente.

[8] Apenas esta folha espera receber uma ação do utilizador, como um menu ou uma caixa de diálogo.

Snackbar Os snackbars (cujo nome provavelmente surge dos antigos *toasts* do Android) são barras que surgem no fundo do ecrã apresentando uma mensagem breve e, opcionalmente, uma única ação. Na figura 2.6d encontra-se uma snackbar com uma ação.

Toolbar *Em Português: barra de ferramentas.* Um componente bastante habitual desde os anos 80, no entanto o conceito da folha de material traz novas possibilidades. No Material Design, as toolbars assumem um papel estrutural da interface, chegando, geralmente, a alterar a sua posição e dimensões de acordo com o objetivo do utilizador. Na maioria dos casos, a toolbar estará separada do conteúdo principal através de um degrau ou de uma costura. A figura 2.6e apresenta uma barra de ferramentas simples, na figura 2.6b também é possível observar este componente.



(a) Card. (b) Bottom sheet modal. (c) FAB.

(d) Snackbar. (e) Toolbar.

Figura 2.6: Alguns componentes do Material Design.

Padrões

As linhas de orientação também fornecem algumas informações sobre como a interface deve reagir em certos casos e de que forma deve comunicar com o utilizador. Estes são alguns desses padrões:

Estados vazios Quando a interface não tem informação para apresentar, esta poderá reagir de uma das seguintes formas:

1. Apresentar uma imagem subtil que transmita ao utilizador a finalidade do *software*. Juntamente com a imagem, deverá surgir um texto que explique a razão de não haver nada a apresentar, sem transmitir a ideia de culpa do utilizador, mesmo caso seja.

2. Caso não seja possível recorrer a uma imagem para explicar qual a finalidade, poder-se-á apresentar um cartão simples que dê acesso a mais informações sobre o *software*.
3. Fornecer conteúdo pré-carregado. Por exemplo, uma aplicação de livros poderá já vir com algum conteúdo.
4. No caso de pesquisas em que nada é encontrado, deve-se tentar encontrar algo que se consiga aproximar o máximo possível daquilo que o utilizador procurava.

Ecrãs de arranque Quando um programa demora algum tempo a carregar, deve-se apresentar uma estrutura básica da interface ou, caso a imagem de marca do *software* seja importante, apresentar um ecrã de arranque com o logótipo.

Transições de navegação Quando o utilizador navega para um ecrã descendente do atual, essa navegação costuma surgir da seleção de um elemento pelo utilizador. Nesse caso, a folha desse elemento deve elevar-se e expandir, cobrindo todo o ecrã. Caso a folha contenha outros elementos, então a zona corresponder da folha deve separar-se do resto.

Notificações As notificações seguem algumas regras: devem priorizar aquelas que se refiram a interações sociais; ser oportunas; permitir aceder aonde o utilizador espera e juntar todos os eventos a notificar numa só notificação, quando possível.

Deslocamento Quando o utilizador pretende deslocar uma lista, o resto da interface pode simplificar-se, dando maior destaque aos itens da lista e mais espaço para estes serem vistos. A simplificação pode ocorrer ocultando as barras na parte superior ou inferior do ecrã. Quando o utilizador desejar voltar a aceder às barras, apenas terá de deslocar ligeiramente na direção oposta, mesmo sem se encontrar no topo da lista.

2.2 Multimédia

A história dos *media players* começou no início dos anos 90. O Windows Media Player e o QuickTime foram dos primeiros. Inicialmente o Windows Media Player (chamado apenas Media Player até ao Windows XP), lançado em 1991, suportava apenas a reprodução de som[9], sendo bastante útil para a reprodução de CD; já o QuickTime suportava a reprodução de vídeos. Em 1992, o Windows Media Player passou a suportar a reprodução de vídeos no formato AVI. Em 1995 surgiu o RealPlayer, um dos primeiros a suportar transmissão pela Web. Em 2000, o Windows Media Player passa a permitir o utilizador gerir a sua biblioteca multimédia.

Verifica-se que os casos de uso de um *media player* foram sofrendo alterações ao longo do tempo e que as interfaces se têm adaptado, em geral, a essas utilizações. Nas subsecções seguintes segue-se uma análise a alguns reprodutores multimédia recentes, confrontando-os com o que se reteve dos estudos anteriormente citados e, se adequado, dos conceitos do Material Design.

[9] No entanto, também suportava animações simples.

2.2.1 Google Play Music

Este é o *software* de reprodução de música da Google, oferecido em conjunto com um serviço de música ilimitado. Como seria expectável, a interface com o utilizador segue os padrões do Material Design. Este produto está disponível em dispositivos Android, iOS e em qualquer computador, desde que este tenha um navegador web suportado. Apesar de ter um ponto positivo no que toca à semelhança da GUI nos diferentes dispositivos, tem o inconveniente de forçar a utilização de um navegador web na versão para computador.

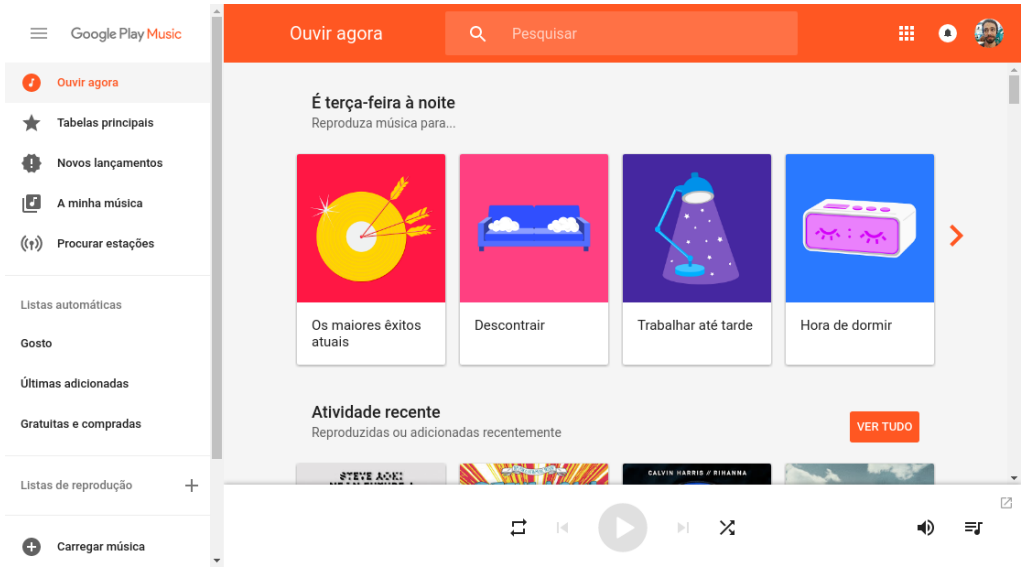


Figura 2.7: Ecrã inicial do Google Play Music (no computador).

No computador, quando o utilizador abre o Google Play Music, é apresentado um ecrã semelhante ao da figura 2.7 constituído por uma barra de navegação no lado esquerdo e, na área central, conteúdo pronto a consumir. À primeira vista, esta interface parece não só simplificar o trabalho do utilizador como ainda o incentiva a conhecer novas músicas[10], mas existem alguns problemas.

O FAB desta interface - que, relembrando, é um botão que executa a ação mais importante/provável - apresenta um problema logo ao primeiro contacto com o utilizador: está desativado. Este botão não só deveria estar ativado, como existem funcionalidades úteis para o atribuir, como por exemplo continuar onde a reprodução foi deixada na última utilização ou reproduzir aleatoriamente todas as músicas da biblioteca. A explicação para a existência desta falha torna-se ainda mais difícil de encontrar quando existe a funcionalidade "Sinto-me com sorte" que automaticamente reproduz músicas ao acaso, esta funcionalidade encontra-se escondida mais abaixo na área central e no "mini leitor" (figura 2.8), que não é tão fácil de encontrar.

Infelizmente, os problemas com este botão continuam, quando se atinge o fim da *playlist*, não há nenhuma forma de voltar a ouvir qualquer música carregando neste botão ou nos botões adjacentes uma vez que estão todos desativados. Mas existem ainda mais problemas de usabilidade que podem comprometer a experiência do utilizador: quando este está a construir uma *playlist*, ou simplesmente já está a ouvir algumas músicas para as quais espera

[10] Função dependente da subscrição do serviço de música ilimitado do Google Play Music. Caso o utilizador não seja subscritor, são apresentadas sugestões com a música existente na biblioteca multimédia do mesmo.

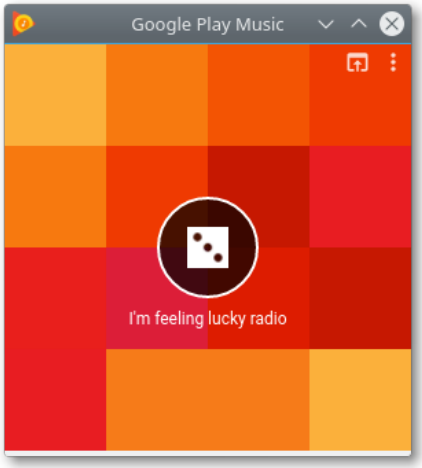
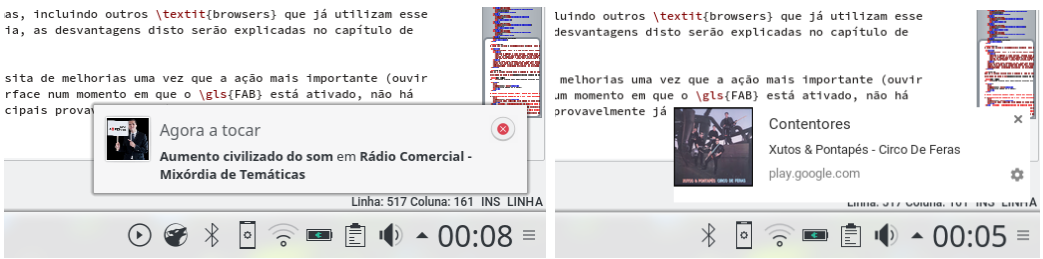


Figura 2.8: Mini leitor do Google Play Music quando a *playlist* está vazia.

poder facilmente recuar, e carrega no botão de reproduzir uma coleção de músicas (álbum, as mais populares de um artista, etc.), a sua *playlist* é substituída por essa coleção, sem a possibilidade de anular a ação. Este comportamento pode não ser o esperado pelo utilizador, originando um erro desnecessário (que tanto pode ser um lapso, como um engano) que nem sequer pode ser anulado. O *feedback* neste tipo de operação destrutiva também é escasso. O painel no qual é possível ver e gerir a *playlist* também apresenta um problema: é impossível permanecer aberto enquanto o utilizador interage com o resto da interface, havendo mesmo situações nas quais os cliques/toques do utilizador não fazem o esperado porque a interface utilizou-os para ocultar o painel primeiro.

No que toca à integração com o sistema, também se apontam algumas falhas: no macOS, as teclas multimédia não são reconhecidas pelo Play Music (Nield 2015); as notificações são geridas pelo Google Chrome que, no Linux não utiliza o sistema já padronizado há mais de uma década (Hearn, Hammond e McCann 2016), tendo um aspeto e comportamento bastante distintos das notificações de todas os outros programas, incluindo outros *browsers* que já utilizam esse sistema; apesar de o Linux também ter convenções para leitores multimédia, o Play Music não informa ao sistema que este é um leitor multimédia, as desvantagens disto serão explicadas no capítulo de desenvolvimento. A figura 2.9 apresenta a comparação das notificações do Google Play Music com o Amarok, que utiliza o sistema padronizado das notificações.



(a) Notificação padrão no Linux usando o tema "Brisa" no KDE 5.

(b) Notificação do Google Chrome no Linux.

Figura 2.9: Comparação entre uma notificação de sistema (à esquerda) e as notificações do Google Play Music (à direita).

Apesar do aspeto seguir os padrões do Material Design, as animações são escassas e não seguem os padrões. Mas isto poder-se-á dever a limitações do CSS para a implementação facilitada de alguns tipos de animação do Material Design.

Resumindo os principais problemas desta interface, pode-se dizer que a primeira característica (descoberta) do bom design de Norman necessita de melhorias uma vez que a ação mais importante (ouvir música) não pode ser executada através do botão mais importante. Relativamente à segunda característica (compreensão) e analisando a interface num momento em que o FAB está ativado, não há quaisquer problemas a apontar. Relativamente ao princípio 13 de Wickens, Gordon e Liu há aspetos positivos e negativos a salientar, sendo o positivo o fato que os controlos principais provavelmente já fazem parte do conhecimento do utilizador e o negativo o fato que a aplicação, ao não integrar-se com o sistema, não funciona como o utilizador espera. A versão do Google Play Music para computador não dá acesso à música armazenada localmente ao contrário da versão para Android.

2.2.2 Amarok

O Amarok é o leitor de música habitualmente encontrado em distribuições Linux com o KDE. Tem as funções básicas tais como a gestão da biblioteca de música e a gestão de *playlist* mas também conta com a integração com alguns serviços *online* e a sincronização com dispositivos multimédia, incluindo o Apple iPod, sendo das poucas formas de o fazer no Linux, uma vez que a Apple não fornece o seu software iTunes para este sistema.

A interface do Amarok é altamente personalizável, sendo a sua estrutura predefinida composta por: uma barra lateral esquerda na qual se navega pelos diversos conteúdos disponíveis; uma barra lateral direita com a *playlist*; uma barra no topo com os controlos principais; conteúdo central com *widgets*. Alguns pontos diferenciadores do Amarok são a barra com os controlos principais, com uma disposição diferente e menos botões que o habitual. O conteúdo central, sendo composto por widgets, apresenta conteúdo dependente das escolhas que o utilizador fizer, na configuração predefinida, apresenta várias informações acerca da música atual. A figura 2.10 apresenta a interface do Amarok, nesta figura é possível verificar uma má gestão de um problema: não foi possível obter a duração das músicas no momento em que foram adicionadas à lista, ficando todas marcadas com a duração "0:00", para não induzir o utilizador em erro, o melhor seria não apresentar a duração em caso de erro, para além disso, no momento em que a música é reproduzida já é possível saber a duração, nesse momento a lista poderia ser atualizada para apresentar a duração correta.

Algumas destas inovações parecem ter algumas vantagens, segundo os estudos referidos neste relatório. Por exemplo, a forma como se muda para a música anterior ou seguinte nem sequer requer o conhecimento prévio de qualquer símbolo uma vez que aparecem os nomes de ambas as músicas em cada extremo da barra, para além disso a mudança de música é acompanhada com uma animação na qual os nomes das músicas deslizam. A obtenção de informação extra sobre a música atual é mais um ponto positivo, no entanto, poderia haver uma melhor estruturação da mesma. Infelizmente, também há alguns problemas: ícones confusos e pouco consistentes, linguagem confusa, controlo de volume estranho, controlos mal alinhados, espaços desnecessariamente em branco, navegação pelos widgets confusa. Apesar destes pontos que discordam com os estudos previamente referidos, há um a favor que não se encontra no Google Play Music: a possibilidade de anular ações.

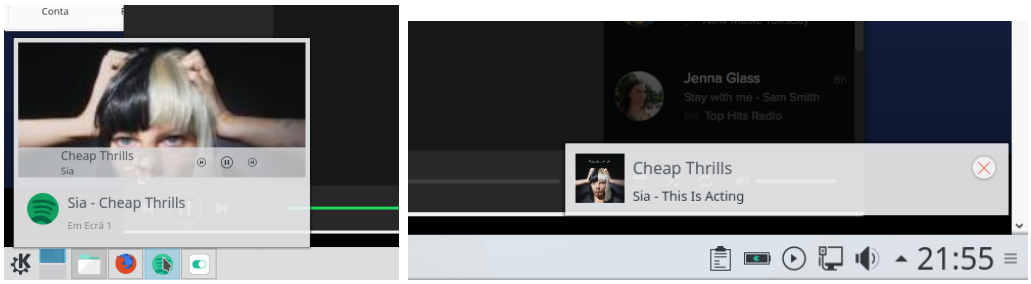
Figura 2.10: Interface do Amarok.

Contrariamente ao Play Music, o Amarok integra-se com o sistema. As notificações funcionam como esperado (figura 2.9a) e o sistema reconhece-o como leitor multimédia, sendo possível controlá-lo sem interagir diretamente com o Amarok.

2.2.3 Spotify

O Spotify oferece um serviço semelhante ao serviço por subscrição do Google Play Music. Contrariamente ao serviço da Google, o Spotify funciona numa aplicação separada do navegador web. O Spotify não segue oficialmente o Material Design mas alguns aspetos, principalmente a organização da interface, estão de acordo com a documentação do Material Design.

Tal como o Play Music, o Spotify inicia-se com um ecrã que oferece conteúdo pronto a consumir, o que permite ao utilizador escutar música em poucos passos. Na parte inferior também se encontra a barra de controlo de reprodução, que não apresenta nada que a maioria dos utilizadores já não estejam à espera.



(a) Controlos na janela (b) Notificação

Figura 2.11: Integração do Spotify com o sistema.

A barra lateral direita quase que força o utilizador a ligar as suas redes sociais ao Spotify para depois publicitar o que ouve neste serviço, esta barra dificilmente é removida e não só ocupa espaço desnecessário como também se torna um inconveniente para quem apenas quer ouvir música. Já no lado esquerdo, temos uma barra de navegação semelhante à do Play Music.

A navegação pela aplicação é fácil e, apesar do seu aspeto diferente, estrutura-se de forma também semelhante ao Google Play Music. Em termos de apresentação (e possível modelação) dos dados, o Spotify tem em conta que vários artistas podem fazer parte da mesma música e apresenta os artistas convidados corretamente em tabelas ao contrário do habitual "feat. *nome do artista*" na coluna do título da música em todos os outros programas.

Em termos de integração com o sistema, a equipa do Spotify fez um excelente trabalho, mesmo no Linux. Contrariamente ao Play Music, o Spotify informa ao sistema que é um leitor multimédia, o que implicitamente estende as suas funcionalidades, tais como a da figura 2.11a, assunto melhor descrito mais à frente do capítulo de desenvolvimento. Mais uma vez, ao contrário da aplicação da Google, o Spotify utiliza o sistema padrão de notificações o que lhe permite apresentar notificações consistentes com o resto do sistema, como se observa na figura 2.11b.

2.2.4 Windows Media Player e as Novas Aplicações da Microsoft

O Windows Media Player já faz parte do Windows há muito tempo, sendo inicialmente conhecido apenas como "Media Player". A partir do Windows 8 que surgiram duas novas aplicações: o "Groove Música" e o "Filmes e Programas de TV". O Windows Media Player parece ter parado no tempo, mantendo as funcionalidades e interface desde o Windows 7. Já as novas aplicações seguem os padrões de design do Windows 10.

O Windows Media Player, desde a versão incluída no Windows 7, que se inicia num ecrã que oferece acesso direto aos conteúdos multimédia do utilizador, facilitando a sua interação. Nesta versão também surgiu a opção de uma interface bastante simplificada, apenas com o essencial para controlar a reprodução das músicas e vídeos. No entanto, a gestão da lista de reprodução ou o carregamento de legendas não receberam tanta atenção como as funcionalidades básicas, sendo mesmo impossível, na maioria dos casos, carregar legendas.

O estado da aplicação não é mantida quando é reaberta, no entanto existe uma opção no menu de contexto da aplicação, o que muitos utilizadores poderão desconhecer. Como é de esperar, este leitor integra-se bastante bem com o sistema, à exceção do aspeto em geral da interface, que se diferencia bastante das outras aplicações.

Já com as novas aplicações, a Microsoft optou por separar a música dos vídeos. Ambas as interfaces se iniciam com um ecrã com o conteúdo do utilizador e novo conteúdo a comprar. A utilização do nome "Filmes e Programas de TV" tem a vantagem de ser óbvio para o utilizador quando este procurar exatamente isso, no entanto também pode causar confusão: caso o utilizador queira abrir um vídeo normal com outra aplicação verá a opção "Filmes e Programas de TV".

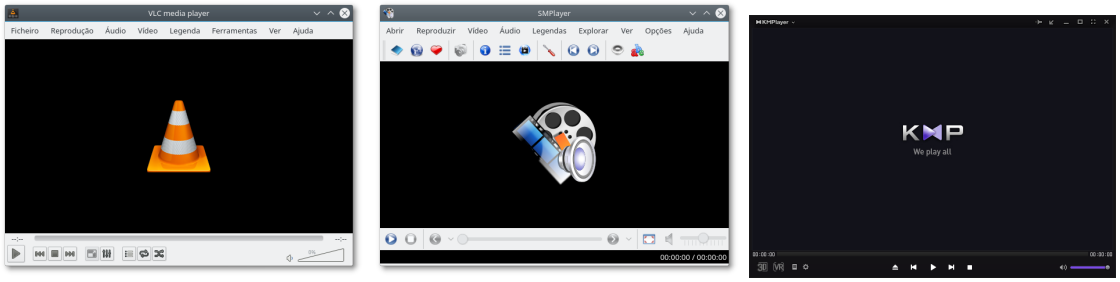
É possível carregar legendas, mas apenas manualmente, mesmo quando o ficheiro de legendas tem o mesmo nome que o ficheiro em reprodução. Como não é de esperar, a integração com o sistema fica-se apenas pela utilização do Modern UI da Microsoft, não é possível arrastar e largar conteúdos, a miniatura da janela não oferece quaisquer opções (exemplo:

figura 4.9a) e não há qualquer acesso direto através do menu de contexto da aplicação (exemplo: figura 4.9d). Estas aplicações também não oferecem qualquer personalização, para além das oferecidas a nível de sistema.

2.2.5 VLC, SMPlayer e KMPlayer

Estes três leitores também suportam a leitura de vídeos e têm todos uma estrutura semelhante, como apresentado na figura 2.12. Esta estrutura tem a vantagem de se assemelhar à de um programa tradicional, trazendo alguma familiaridade e consistência. No entanto, esta mesma estrutura pode não ser a mais adequada para os seus casos de uso, ou simplesmente as suas capacidades não foram tão bem aproveitadas.

Graças às suas opções avançadas, estas aplicações são mais apropriadas para um público que se sinta mais à vontade com a interação com computadores. Estas aplicações não suportam bibliotecas multimédia, sendo o carregamento de conteúdos multimédia feito através de ficheiros, pastas e endereços de rede. Estas interfaces suportam temas, podendo-se tornar mais apelativas, como no KMPlayer. Apenas o VLC e o SMPlayer estão disponíveis para Linux.



(a) VLC (b) SMPlayer (c) KMPlayer

Figura 2.12: Interfaces do VLC, SMPlayer e KMPlayer.

A interface visível após a abertura destas aplicações não oferece qualquer ponto de partida compreensível para quem não está à vontade com o conceito de ficheiros e pastas. Como consequência, esses utilizadores poderão não conseguir fazer nada caso abram a aplicação diretamente (em vez de abrirem um ficheiro). No entanto, o botão de reproduzir executará sempre alguma função, contrariamente ao Google Play Music.

O SMPlayer mantém o estado deixado na última utilização, facilitando a sua utilização, mas essa funcionalidade está praticamente invisível uma vez que a lista de reprodução encontra-se fechada por predefinição e não há qualquer outra indicação. O VLC possui menus mais sucintos que os outros mas algumas das funcionalidades não encontradas no menu são mais complicadas de aceder e configurar.

A interface do KMPlayer apresenta publicidade, o que reduz a qualidade da mesma e pode causar sérios problemas de usabilidade uma vez que poderão aparecer publicidades com mensagens ilusórias como por exemplo "Precisa da última versão do Flash Player, carregue aqui para o instalar!" ou "Foram encontrados 123 vírus no seu computador, deve removê-los imediatamente, carregue aqui para os remover!". Na figura 2.13 observa-se um dos tipos de publicidade existentes na aplicação, ironicamente a referir problemas no Windows quando se encontra em execução no Linux (através do Wine), também há publicidades que surgem na própria aplicação.

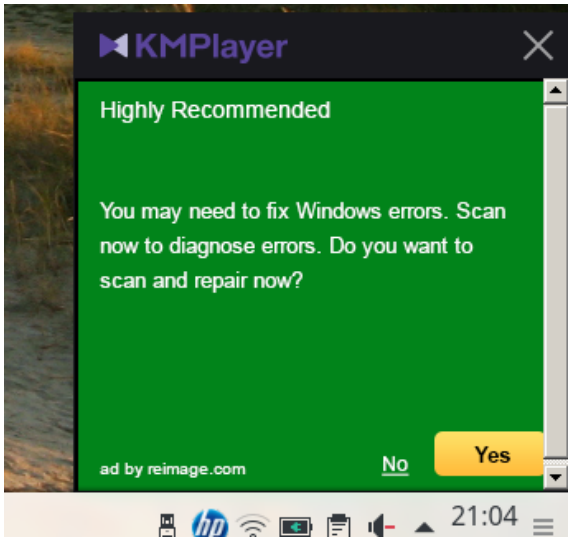


Figura 2.13: Publicidade enganosa.

Contrariamente às soluções da Microsoft, estes leitores costumam suportar uma grande quantidade de formatos de áudio e vídeo. Uma outra vantagem face ao Windows Media Player e ao "Filmes e Programas de TV" está no carregamento de legendas. Estes leitores não só carregam automaticamente as legendas se forem encontradas na pasta do ficheiro em reprodução mas também oferecem uma função para transferir as legendas (manualmente) da Internet. Já o controlo das legendas, apesar de bastante completo, é complicado, recorrendo mais uma vez a hierarquias de menus.

O VLC e o SMPlayer estão corretamente integrados com o sistema, proporcionando uma interface a nível do sistema exatamente igual a qualquer outro leitor. Esta integração também traz outras melhorias debatidas mais à frente, no capítulo de desenvolvimento.

Concluindo, estas aplicações têm a vantagem de oferecer um controlo muito mais avançado e permitir uma grande personalização das interfaces mas falham numa das características principais do bom design: descoberta. A função principal de um leitor multimédia - reproduzir músicas e vídeos - só pode ser acedida por utilizadores experientes a não ser que o ficheiro seja aberto externamente. Já algumas outras funcionalidades comuns em leitores (que requerem que algo já esteja em reprodução) são bastante fáceis de descobrir e vão de encontro ao já esperado pelos utilizadores.

Funcionalidades mais avançadas já não são tão fáceis de encontrar, como é de esperar, mas o controlo das legendas poderá ser bastante melhorado. A lei de Fitts indica-nos mais problemas: os botões mais utilizados são pequenos e não é aproveitado o facto que estes se encontram nas bordas do ecrã quando a aplicação está em ecrã inteiro, resultando num problema semelhante ao do Microsoft Visual Studio. Para utilizadores experientes, estas interfaces têm boa usabilidade mas não se pode dizer o mesmo em relação a utilizadores menos experientes.

2.3 Conclusão

Neste capítulo ficaram-se a conhecer melhor conceitos de psicologia, alguns erros a evitar e pequenas regras fáceis a ter em conta na criação de interfaces. Exploraram-se os conceitos mais importantes do Material Design, o projeto Papyros e as tecnologias relacionadas.

Também se analisaram as interfaces de vários leitores multimédia. Foram encontrados aspetos positivos e negativos em todos. A implementação do Material Design no Google Play Music fica-se mais pelo aspeto "à primeira vista", falhando em muitos aspetos durante uma utilização real da aplicação, estas falhas não só contrariam o Material Design, mas também as regras de bom design estudadas. O Spotify, apesar de não implementar oficialmente o Material Design, consegue aproximar-se mais das regras comportamentais que o Play Music, tendo uma interface, em geral, mais fácil e eficiente de utilizar. O Amarok ganha em termos de personalização e funcionalidades em relação aos anteriores mas a interface requer uma aprendizagem e/ou adaptação maiores. As soluções atuais da Microsoft pouco têm a apresentar devido às suas limitações funcionais, no entanto são fáceis de utilizar para as funções básicas. As outras três aplicações analisadas adequam-se mais a um público alvo mais experiente, principalmente o KMPlayer devido à utilização de publicidade que pode induzir os utilizadores em erro.

Capítulo 3

Análise

Neste capítulo apresenta-se a fase inicial do desenvolvimento do projeto na qual se analisa os diversos aspetos do mesmo e onde se tiram algumas conclusões iniciais sobre a sua estruturação. Serão tomadas várias decisões que determinarão de que forma a fase seguinte decorrerá, no entanto, estas poderão sofrer várias alterações mais à frente.

3.1 Valor

Este é um projeto *open-source* o que significa que todo o código do software é acessível também por pessoas que não façam parte da equipa. A comunidade *open-source* investe o seu tempo no desenvolvimento de *software* com o objetivo de o tornar livremente acessível a todos, esta liberdade permite que várias pessoas em todo o mundo trabalhem em conjunto para evoluir esse mesmo *software*. O *software open-source* por vezes também acaba por ser adotado mais tarde por empresas, como por exemplo o renderizador de HTML do navegador web Konqueror que foi adotado pela Apple para o seu navegador Safari e, mais tarde, pela Google para o Chrome (Barth 2013). Um outro exemplo é o sistema de impressão usado no Linux (CUPS), que em 2002 passou também a fazer parte do macOS e, em 2007, o código do *software* foi adquirido pela mesma empresa (Sweet 2007). Devido às licenças utilizadas, estas empresas não podem publicar software com versões modificadas desse mesmo código sem o manter *open-source*, desta forma, não só apenas indivíduos mas também empresas trabalham em conjunto na melhoria desse mesmo software e derivados do mesmo, sendo todos beneficiados. O desenvolvimento do Linux é feito com a contribuição de várias empresas, sendo as principais 11 a Intel, Red Hat, Linaro, Samsung, SUSE, IBM, Renesas, Google, AMD, Texas Instruments e a ARM (The Linux Foundation 2016). Esta contribuição é benéfica para ambos os lados: o Linux continua a evoluir e as empresas usufruem dos benefícios do mesmo.

O software utiliza componentes do ambiente gráfico do Papyros, projeto também ainda em desenvolvimento. Ao longo da implementação da interface gráfica do leitor multimédia, poder-se-ão verificar problemas ou funções em falta no projeto Papyros. O que estiver em falta poderá ser desenvolvido inicialmente para o leitor e mais tarde incluído no próprio projeto Papyros, contribuindo também para o desenvolvimento do Papyros e de outros projetos que dependam do mesmo. Com este software também é possível mostrar à comunidade as possibilidades do Papyros e dos seus componentes, podendo motivar mais pessoas a contribuir com o mesmo. Para os utilizadores do Papyros, este leitor multimédia irá melhorar a experiência de todo o sistema uma vez que há uma melhor integração.

A extensibilidade da aplicação permite que sejam fornecidos novos conteúdos multimédia de diversas fontes. Por exemplo, a Google poderia criar uma extensão do Google Play Music para a aplicação, sendo possível ter-se no computador uma experiência semelhante àquela fornecida pela versão para dispositivos móveis (acesso a conteúdo local mais o conteúdo na conta). Uma extensão para o Play Music traz benefícios para os utilizadores uma vez que não estariam dependentes do Google Chrome. Outros serviços também poderiam desenvolver as suas próprias extensões. Estas extensões poderão ajudar a divulgar o serviço ou a fornecer um melhor serviço aos utilizadores atuais.

Uma rede de valores é uma rede de relações que geram valor tangível e intangível através de trocas complexas entre dois ou mais indivíduos, grupos e/ou organizações. Qualquer organização envolvida em trocas tangíveis e intangíveis pode ser vista como uma rede de valores, quer pertença ao setor privado, público ou ao governo.(Nicola 2016)

Tal como muitos dos projetos *open-source*, este não tem em vista a geração de valor monetário, no entanto muitos outros valores são transferidos entre o projeto, os utilizadores, a comunidade *open-source*, o projeto Papyros e a sua equipa e até mesmo outros serviços externos. A figura 3.1 esquematiza os principais valores gerados e a rede formada na transferência dos mesmos:

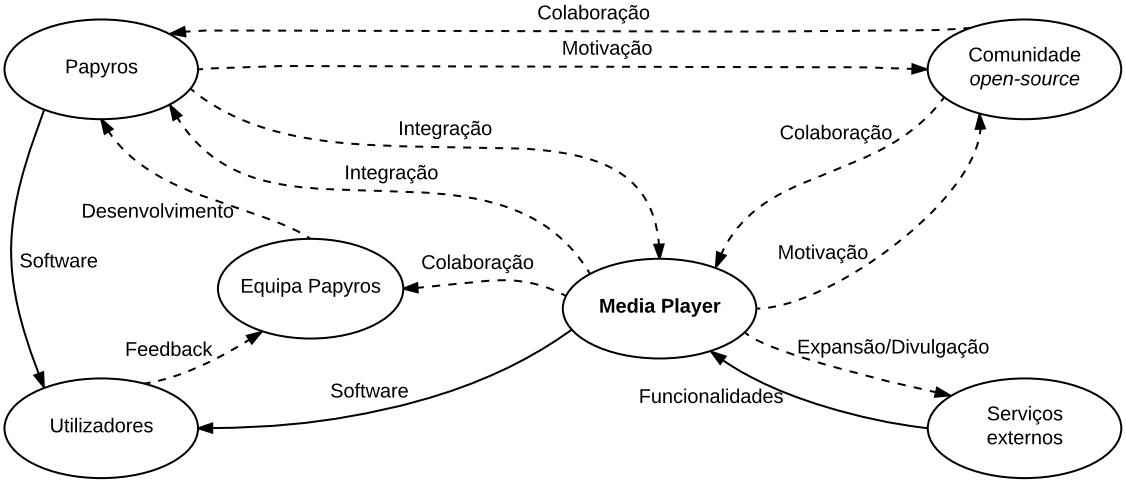


Figura 3.1: Rede de valores.

Cada elipse representa grupos e organizações, as setas completas e as tracejadas representam transferência de valor tangível e intangível, respetivamente.

3.2 Requisitos

Como já referido anteriormente, um dos objetivos deste projeto é testar a usabilidade do Material Design em computadores de secretária e portáteis e, para isso, desenvolver-se-á um *media player*. Para além de alguns aspetos do seu design, este produto terá de ir ao encontro das necessidades atuais dos utilizadores de hoje em dia, até porque, para além de servir para testar a utilização do Material Design neste contexto, também poderá ser um componente essencial a um ambiente gráfico.

Espera-se que a aplicação possa integrar serviços online. Estes serviços podem fornecer à aplicação informações que a auxiliem a organizar a biblioteca do utilizador, obter informações extra sobre artistas, programas de TV, etc. e também fornecer mais conteúdos multimédia, por exemplo através da integração do Google Play Music.

Apesar da utilização desses serviços na nuvem, espera-se que a aplicação consiga funcionar sem depender dos mesmos. Tal como nos leitores multimédia tradicionais, a aplicação também organiza músicas e vídeos armazenados localmente. A aplicação poderá funcionar como agregadora da biblioteca local e bibliotecas externas. Esta organização de conteúdos também poderá oferecer relações semânticas mais lógicas para o utilizador, como por exemplo reconhecer a participação de outros artistas em músicas ou em filmes.

A biblioteca multimédia é constituída inicialmente por músicas, programas de TV e filmes. No entanto, também é capaz de carregar vídeos e ficheiros de áudio que não se incluam nestes grupos. No caso dos vídeos, é possível carregar legendas e o seu carregamento será automático, se possível. O utilizador pode, manualmente, adicionar novos artistas e programas de TV, ajudando a aplicação a localizar possíveis ficheiros que não tenham metadados, esta funcionalidade também poderá ser útil quando utilizada com extensões de serviços de música ilimitada.

Outras funcionalidades também incluem o suporte a listas de reprodução, integração com o sistema e personalização. Em termos de requisitos não-funcionais, espera-se que a aplicação *1*) tenha uma boa usabilidade, de acordo com o que foi abordado na secção "Estudos" (página 9); *2*) siga as orientações do Material Design como explicado na secção "Material Design" (pagina 18); *3*) tal como os dois pontos anteriores sugerem, as operações que demorarem muito para o dispositivo concluir deverão executar num *thread* em separado, prevenindo que a interface bloqueie, também é necessário dar o *feedback* apropriado ao utilizador, sempre que possível também se deve encurtar a duração de operações que não sejam executadas em separado; *4*) seja multiplataforma.

A figura 3.2 apresenta um diagrama de casos de uso com um resumo das funcionalidades do sistema. O diagrama encontra-se na notação UML.

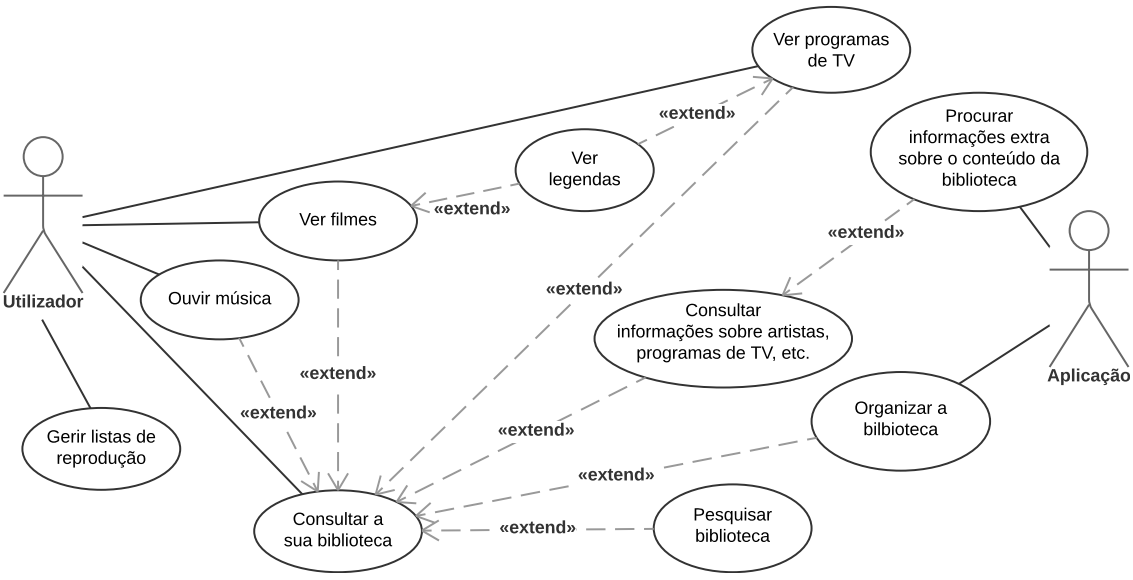


Figura 3.2: Diagrama de casos de uso (UML).

Outras funcionalidades extra, que poderão ser implementadas, dependendo do tempo disponível, são o suporte a *podcasts*, organização de outros tipos de conteúdo na biblioteca, reprodução e pesquisa de vídeos no YouTube e a pesquisa avançada de conteúdo. Algumas destas funcionalidades podem e/ou devem ser implementadas através de extensões.

3.3 Tecnologias a Utilizar

Apesar de ser multiplataforma, o *media player* tem o objetivo de se integrar de forma coerente com o **Papyros**. O Papyros está implementado, principalmente nas linguagens C++ e QML. A *framework* principal desta *shell*, como referido anteriormente, é o Qt, que fornece a *toolkit* necessária para a construção da interface gráfica, e não só.

Relativamente à gestão de hardware, atualmente o Papyros recorre ao componente "Solid" do KDE SC 5. O *media player* poderá precisar de recorrer a algum componente de gestão de hardware para tarefas como por exemplo identificação do estado da ligação à rede e/ou gestão de dispositivos multimédia.

O QML é uma nova linguagem de programação que faz parte do Qt, tem uma sintaxe semelhante à do JSON e permite a injeção de JavaScript e a inclusão de ficheiros .js, desenvolvida a pensar na experiência do utilizador e com o objetivo de funcionar de forma fluida (60 fotogramas por segundo). Os ficheiros em QML podem coexistir com ficheiros em C++, sendo possível manipular objetos declarados em QML através em C++.

No código 3.1 apresentado a seguir, é possível ficar a conhecer um pouco da sintaxe base do QML. Neste exemplo são criados dois `Rectangle`s dentro de um `Item`, o primeiro `Rectangle` tem o id/nome "`myRect`", o segundo e anónimo `Rectangle` terá sempre metade da largura (width) do `myRect`, mesmo se a largura desse `Rectangle` for alterada durante a execução do programa.

```
1  Item {
2    Rectangle {
3      id: myRect
4      width: 120
5      height: 100
6    }
7    Rectangle {
8      width: myRect.width/2
9      height: 200
10   }
11 }
```

Excerto de Código 3.1: Exemplo de código em QML. (*QML* 2016)

O QML tem várias vantagens que permitem acelerar o desenvolvimento de interfaces gráficas e, portanto, será a linguagem escolhida para a implementação da mesma. Já as camadas de dados e gestão multimédia serão implementadas em C++ devido à sua maior velocidade de execução e consumo de memória.

O Qt garante que o software seja independente da plataforma para a maioria das funcionalidades. No entanto poderá haver a necessidade de se aceder a algumas funcionalidades que não façam parte do Qt, talvez seja por isso que o Papyros atualmente depende de um componente de uma outra shell. De forma a evitar dependências a outras shells e de

plataforma, poder-se-á utilizar variáveis de compilação para compilar classes diferentes para os diferentes casos, algo semelhante ao padrão "strategy" mas parcialmente estático: cada sistema terá a sua própria classe para um determinado comportamento, havendo uma extra que servirá para os casos em que não haja suporte. Todas estas classes implementarão uma determinada interface de forma a encapsular todo o comportamento dependente da plataforma. O *media player* também poderá ter alguma integração específica com o Papyros, nesses casos a mesma solução, ou semelhante, será utilizada.

Em relação ao sistema de gestão de base de dados a utilizar, prefere-se um sistema leve e capaz de funcionar sem servidor (ou seja, dentro da própria aplicação). O primeiro passo tomado foi decidir entre uma base de dados Structured Query Language (SQL) e NoSQL. O NoSQL surgiu décadas após as base de dados SQL, e, apesar de ter algumas vantagens, não irá substituir por completo o SQL (Ibraheem 2014). Em qualquer dos casos, o NoSQL foca-se na escalabilidade horizontal (adicionar mais sistemas para processarem os dados em conjunto), algo útil para servidores mas que não traz qualquer benefício para bases de dados locais.

Portanto, escolheu-se utilizar SQL, dentro desta tecnologia há também várias opções que devem ser tomadas em conta. Uma vez que o objetivo é gerir a base de dados dentro da própria aplicação, devem-se excluir bases de dados que foram desenvolvidas com o propósito de serem executadas num servidor à parte tais como o MySQL. Duas boas opções são o SQLite e o Firebird. Ambas podem ser facilmente integradas na aplicação, ocupando, respetivamente, menos de meio MB e menos de 2 MBs. Relativamente ao desempenho de ambas, existem vários testes que se contrariam entre si, no entanto, as diferenças são mínimas. Visto que o SQLite é ligeiramente mais leve que o Firebird e que o Qt já oferece suporte completo, o SQLite será escolhido para a gestão da base de dados local da aplicação.

3.4 Arquitetura

Com o requisito da integração com serviços externos pode existir a possibilidade/necessidade de integrar a biblioteca multimédia do utilizador com, por exemplo, a biblioteca no Google Play Music. O *media player* pode então juntar todas as bibliotecas que o utilizador tiver, ajudando-o a organizar e consultar os conteúdos.

Dentro da sua biblioteca, contam-se, no mínimo, os grupos "Música", "Programas de televisão" e "Filmes". No entanto, o sistema será capaz de suportar outros grupos tais como *podcasts* e conteúdos pessoais.

A partir da biblioteca, o sistema poderá "aprender" que artistas, atores, etc. existem e também corrigir problemas nos metadados dos conteúdos, como por exemplo o uso de abreviaturas, diferenças entre maiúsculas e minúsculas, etc. Se possível, o programa poderá recorrer a algum componente (também extensível) para obter mais informações através da Internet. Ao reunir todos estes dados, é possível criar relações semânticas, como por exemplo, reconhecer um ator de um filme que também pertença a uma banda.

O *media player* também permite o suporte, através de extensões, à reprodução de outros conteúdos, como por exemplo vídeos do YouTube.

Apesar de o *media player* funcionar de forma completamente independente do acesso à Internet, é possível integrar o programa com serviços externos. A figura 3.3 representa,

de uma forma abstrata, de que forma os diversos componentes se integram, utilizando a notação UML.

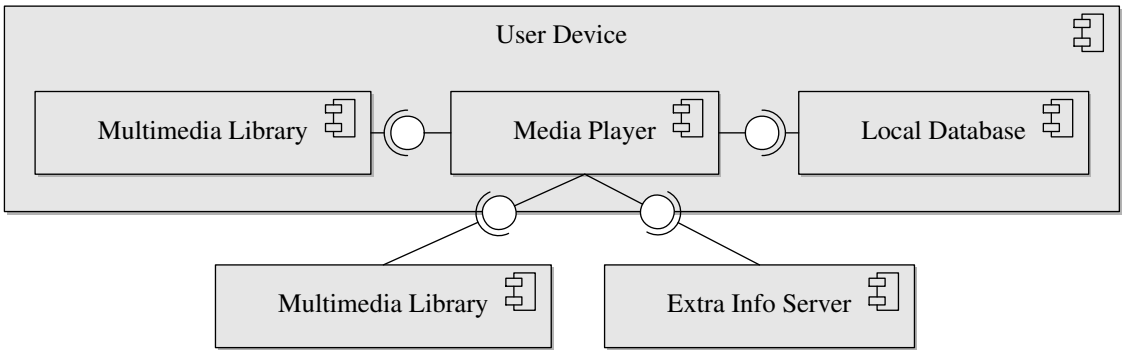


Figura 3.3: Diagrama de componentes do sistema.

Relativamente à arquitetura do sistema de gestão de dados e multimédia, existem os seguintes componentes principais:

Library Classe principal que permite o acesso à biblioteca completa do utilizador. Contém implementações de `MediaPropertyLibrary`'s.

LibraryProvider Interface que representa uma biblioteca do utilizador. O conteúdo multimédia destas implementações (como por exemplo uma extensão de *podcasts*) estarão acessíveis através da `Library`.

MediaProperty Interface que representa uma propriedade relevante a um elemento multimédia (artista, álbum, temporada, ator, etc.).

MediaPropertyLibrary Interface que representa diferentes bibliotecas de `MediaProperty`.

MediaContent Interface que representa qualquer conteúdo multimédia que possa ser reproduzido tais como músicas e episódios.

MediaProvider Interface cujas implementações acrescentam suporte a carregamento de conteúdos multimédia de diferentes fontes. A classe `GenericMediaProvider` já implementa o carregamento de conteúdos naturalmente suportados pela *framework* multimédia utilizada (neste caso, o Qt), uma outra implementação poderá, por exemplo, carregar vídeos do YouTube.

A figura 3.4 apresenta a relação entre estes componentes e algumas implementações.

Alguns `MediaProperty`'s (como por exemplo o `Artist`) podem registar quando foi a última vez que foi pesquisada mais informação pela aplicação e, num momento oportuno, atualizá-la caso já tenha passado algum tempo desde a última vez. A variável/propriedade "`alsoKnownAs`", presente em alguns `MediaProperty`'s é usada para auxiliar o utilizador quando este pesquisar por esses `MediaProperty`'s.

3.5 Interface com o Utilizador

A interface gráfica do programa dependerá das dimensões da sua janela (ou das dimensões do ecrã do dispositivo). Na maioria dos casos, existirá uma região central no qual serão

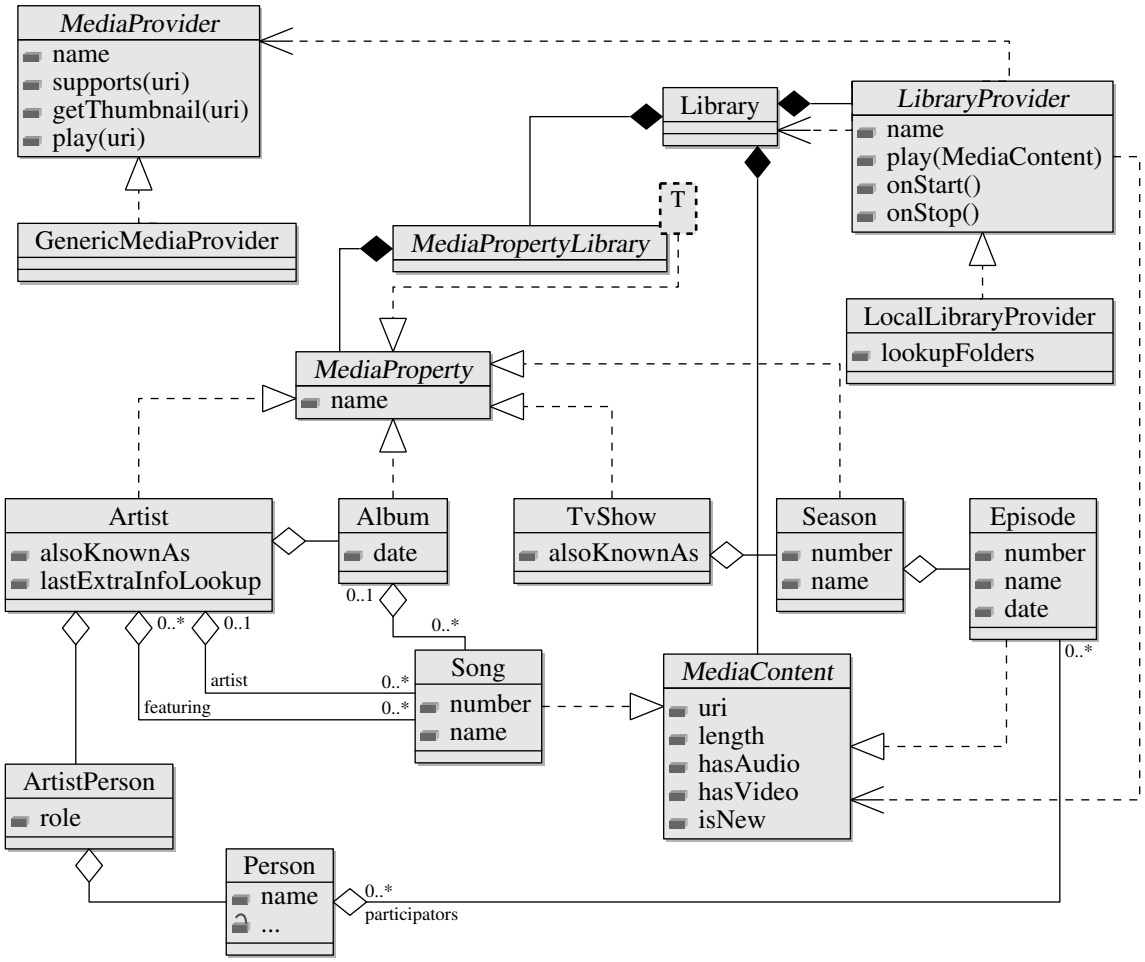


Figura 3.4: Diagrama de classes.

apresentados a biblioteca multimédia do utilizador, recomendações, dicas e outro conteúdo disponibilizado por extensões. Na barra de navegação lateral, será possível navegar pela biblioteca multimédia e na barra do lado oposto residirá a lista de reprodução atual. No fundo existirão os controlos multimédia tais como reproduzir, pausar, etc. A figura 3.5 apresenta as áreas principais da interface.

A área **A** apresenta a biblioteca do utilizador de forma simplificada e a área **C** apresenta a lista de reprodução atual. Ambas as áreas podem ser escondidas pelo utilizador de forma independente. Na área **B** será feita toda a navegação pelos conteúdos oferecidos pelo programa. No topo existe uma barra de ferramentas persistente composta por uma barra de pesquisa e opções avançadas à direita, se o sistema operativo do utilizador o suportar, a barra de título da janela integrar-se-á com esta barra. A área **A** tem a sua própria barra de ferramentas no topo.

Na área inferior existem os controlos principais do leitor, do lado esquerdo existem os controlos do áudio e vídeo (visíveis apenas quando necessário), no lado oposto existem os controlos associados à lista de reprodução, ao centro encontram-se os controlos principais. Em junção com as áreas **A**, **B** e **C**, cria-se uma estrutura simétrica mas personalizável pelo utilizador. A colocação dos botões da barra inferior do lado direito foi propositada, uma vez que se encontram do mesmo lado onde se situa a lista de reprodução. Na margem superior desta barra, existe uma barra que indica o progresso do ficheiro atualmente em reprodução.

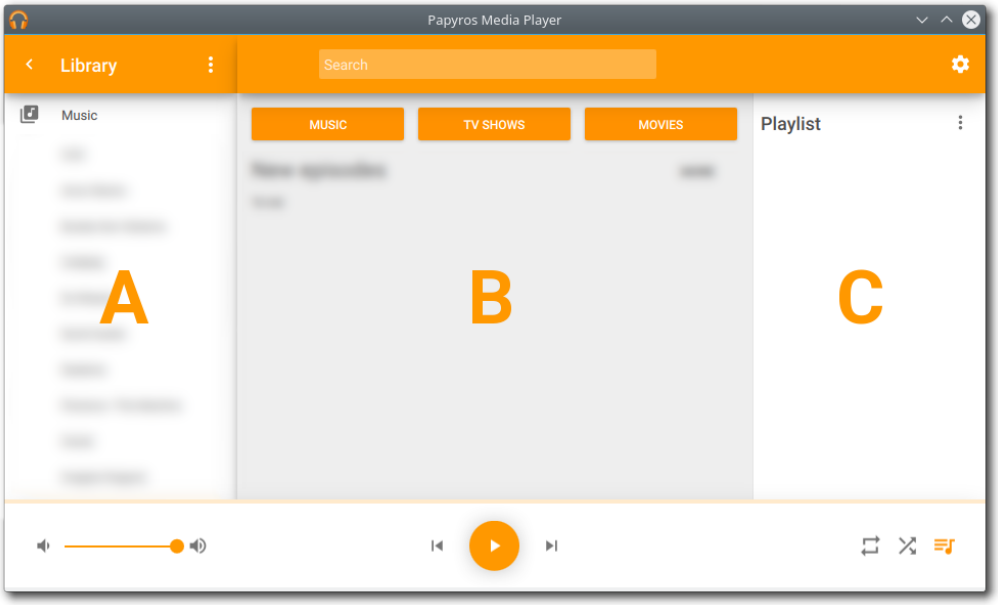


Figura 3.5: Estrutura principal da interface.

A navegação pelos diferentes conteúdos oferecidos pelo programa são feitos na área **B**, esta navegação assemelha-se à da maioria das interfaces em dispositivos móveis e páginas web. Sem quaisquer extensões, esta área permite a navegação pela biblioteca multimédia do utilizador. A redundância é propositada e defendida por Wickens, Gordon e Liu, assim não só se apresenta a biblioteca ao utilizador de duas maneiras diferentes, como é possível facilitar algumas tarefas de organização, como por exemplo associar uma música a um artista diferente arrastando-a da área **B** para um artista da área **A**. A área **A** também permite aceder rapidamente à biblioteca enquanto o utilizador consulta outra informação na área **B**.

A página inicial da área **B** assemelha-se ao Google Play Music, sugerindo conteúdos ao utilizador. No topo existem botões de acesso às categorias principais da biblioteca (que, sem extensões, são: músicas, programas de TV e filmes), a cada um destes botões corresponde uma página constituída principalmente por imagens, facilitando a procura do conteúdo pretendido.

A visualização de vídeos não deve ser feita na área **B** uma vez que tornaria a navegação confusa pelo utilizador e o forçaria a ocultar manualmente os painéis **A** e **C** sempre que quisesse ver o vídeo ocupando a maior área possível da janela. A solução é o vídeo ser apresentado por trás de todos os outros componentes. Quando o vídeo começa, os painéis, barra de ferramentas e barra inferior são ocultadas, assim o vídeo ocupa 100% da área da janela. Tal como na maioria das interfaces em ecrã inteiro, quando o utilizador mover o rato ou tocar no ecrã alguns dos componentes aparecerão, permitindo-o sair deste modo. No lado direito da barra inferior aparecerá um novo botão sempre que houver um vídeo em reprodução, este botão repõe todos os componentes ocultos, ficando ligeiramente transparentes de forma a ainda permitir ver o vídeo. Se este botão será suficientemente aparente ou não, só se poderá concluir depois dos testes de usabilidade.

3.5.1 Regras e Padrões

Nesta subsecção apresentam-se soluções para seguir corretamente as regras e padrões estudados no capítulo do Estado da Arte (página 5), possibilitando o melhor desenho e planeamento da interface.

Prevenção de Erros

Como já foi constatado na análise de leitores multimédia existentes, existe espaço para a melhoria da prevenção de erros neste tipo de interface. Um caso não mencionado nessa análise, mas que não foi coberto por quaisquer dos leitores analisados, é o lapso de se avançar ou recuar num ficheiro multimédia. Isto pode acontecer devido a cliques/toques acidentais ou ao pressionar acidentalmente o teclado. A solução pode passar por apresentar na barra de progresso um ponto representativo do tempo em que a reprodução estava no momento do lapso. Opcionalmente poderá ser oferecido um atalho para esta opção.

Relativamente a uma das possíveis fontes de lapsos em grande parte dos leitor analisados - os botões de avançar para o ficheiro anterior e seguinte - também é possível implementar-se uma solução. Quando a reprodução de um ficheiro se estiver a aproximar do fim, poderá aparecer um botão de repetir. Assim, um utilizador habituado ao comportamento dos outros leitores verá uma nova opção surgir com um ícone que representa melhor o seu objetivo, sendo convidado a explorar essa função, acabando por aprender o novo comportamento. Para o caso de um utilizador que desconheça o comportamento dos outros leitores, não há o risco de este utilizar o botão errado para repetir a reprodução do ficheiro.

Sempre que o utilizador carregar uma nova lista de reprodução, a lista atual será automaticamente substituída. Caso o utilizador pretenda na realidade juntar ambas as listas, poderá ser oferecida uma opção para tal após o carregamento da nova lista.

O utilizador poderá querer, por exemplo, ouvir uma música mas ter o volume muito baixo ou mesmo cortado. A aplicação poderá informar o utilizador desse fato, evitando que este pense que há alguma falha com a aplicação ou com o próprio sistema.

Também poderá ocorrer o lapso de se fechar a janela da aplicação acidentalmente. Deverá ser oferecida uma opção para repor o estado da aplicação antes do lapso ter acontecido.

De forma a prevenir outros erros, como por exemplo a eliminação de elementos, pode-se oferecer uma opção para anular essas opções. De acordo com o Material Design e Norman (2013), na maioria dos casos não se deve pedir confirmação ao utilizador e simplesmente oferecer uma opção de anulação.

Feedback e Animações

De acordo com o Material Design, as animações são uma excelente forma de transmitir informação ao utilizador. Uma utilização habitual de animações com este propósito é o *feedback* de alterações em listas. Quando um item é adicionado, removido ou movido, este deve surgir de uma forma fisicamente lógica e, consequentemente, os outros itens devem mover-se para as suas novas posições. A animação destas alterações, torna-as mais aparentes e mais fáceis de compreender. Este tipo de *feedback* deve ser incluído na aplicação, por exemplo, na lista de reprodução.

Sendo a interface da aplicação constituída por vários painéis recolhíveis, sempre que foram mostrados/ocultados, deverão ser animados. Caso o painel da lista de reprodução esteja ocultado e o utilizador execute alguma ação que a modifique, o ícone que permite mostrar/ocultar a lista deverá animar-se. Esta animação, não só indica a alteração, como também incentiva os utilizadores a clicarem-no caso ainda não compreendam ao certo para que serve ou sequer porque é que a lista não aparece.

Para além destas animações, vários dos elementos do Material Design (*tooltips*, snackbars, etc.) têm animações já definidas que deverão fazer parte da aplicação. A transição entre as páginas da área **B** deverá seguir as orientações do Material Design, no entanto, a sua implementação poderá ser bastante complicada, uma vez que o QML/Qt não oferecem suporte a este tipo de animação. Na barra de ferramentas estará visível a hierarquia das páginas visitadas pelo utilizador, dando *feedback* relativo ao "caminho" percorrido e permitindo-o voltar atrás.

Lei de Fitts

O Material Design, na maioria dos casos, defende que os componentes devem ter grandes dimensões, o que vai ao encontro da lei de Fitts (capítulo 2.1.3 da página 16), facilitando a interação. Por exemplo, a barra de ferramentas, tem a altura de $64dp$ em *tablets* e computadores e, apesar de os ícones terem o tamanho de $24dp \times 24dp$, a sua área de ação é de $56dp \times 56dp$.

Os itens em listas, por exemplo, podem ter dimensões elevadas. Isto facilita a interação mas pode reduzir demais a densidade de informação, portanto deve-se encontrar o equilíbrio entre ambas as desvantagens. O Material Design recomenda a altura mínima de $48dp$.

3.6 Conclusão

As ideias iniciais do produto a desenvolver foram descritas neste capítulo. Apesar de não existir valor monetário, geram-se e trocam-se muitos outros valores, principalmente a colaboração com comunidades *open-source*. Espera-se que este produto esteja preparado para integração com serviços na nuvem mas que não esteja dependente dos mesmos. A aplicação fará uso do Qt e do QML, já usados pela própria *shell* do Papyros. Também se delineou a forma como o sistema se deverá estruturar, no entanto poderão surgir várias alterações.

Capítulo 4

Desenvolvimento

A documentação do processo de desenvolvimento da aplicação e do planeamento dos testes de usabilidade encontram-se neste capítulo. Nem todos os passos estarão documentados devido à sua baixa relevância ou repetição de problemas já discutidos.

4.1 Ferramentas a Utilizar

Um Integrated Development Environment (IDE) é um programa no qual se escreve o código do *software* a desenvolver e auxilia esse processo, ajudando na escrita do código e nos testes da aplicação. No caso deste projeto, o Qt Creator será o IDE escolhido uma vez que foi desenvolvido pela equipa do Qt. A utilização do Qt Creator não é obrigatória, o KDevelop é um outro bom IDE que suporta projetos Qt e também é possível desenvolver aplicações Qt sem recorrer a IDEs. O Qt Linguist, também desenvolvido pela equipa do Qt será utilizado para a tradução da aplicação para português e também utilizado por outros tradutores que queiram contribuir, traduzindo a aplicação para outros idiomas.

Para o desenho de ícones, utilizar-se-á uma aplicação de desenho de gráficos vetoriais. O Inkscape é uma aplicação open-source e existem instruções para o desenho de ícones que sigam os padrões do Material Design, portanto será utilizado sempre que for necessário criar ícones para o projeto.

Para a gestão de tarefas, utilizar-se-á o Trello, um serviço gratuito de gestão de tarefas baseado em SCRUM.

4.2 Bibliotecas a Utilizar

Para além da *framework* do Qt, a aplicação dependerá de algumas outras bibliotecas:

Biblioteca do Material Design do Papyros Esta biblioteca implementa, em QML, os componentes do Material Design de acordo com o documento fornecido pela Google. Contrariamente à implementação no Android, e provavelmente devido a limitações do QML, esta não utiliza sombras processadas em tempo real, o que põe em causa a qualidade das animações no eixo z, como descrito na secção "Material Design" (página 18) e na figura 2.3.

Esta implementação encontra-se, de momento, incompleta: *1*) a barra de ferramentas não permite colocar ícones no menu flutuante manualmente (também conhecido

como "*overflow menu*"), sendo mesmo impossível colocar todas as ações nesse menu; *2*) apesar de a barra de ferramentas ter a implementação de menus flutuantes, não existe uma versão independente da barra, o que seria muito útil para a utilização em menus de contexto; *3*) os menus não suportam itens que tenham o estado "marcado/"desmarcado", também não há o suporte à indicação de teclas de atalho; *4*) o texto das *checkboxes* não segue as orientações do Material Design, apresentando um aspeto pouco consistente; *5*) não existe qualquer implementação de tabelas, quer nesta biblioteca, quer nos componentes base do QML. Estas limitações levarão à necessidade da implementação, no decorrer deste projeto, de alguns destes componentes e funcionalidades.

Esta biblioteca encontra-se disponível em https://github.com/papyros/qml-material e pode ser redistribuída e/ou modificada sob os termos da *GNU Lesser General Public License* versão 2.1 ou superior. A biblioteca contém ícones do Material Design, criados pela Google e lançados sob a licença *Creative Commons Attribution 4.0 International*, o repositório dos ícones está disponível em https://github.com/google/material-design-icons.

TagLib Um leitor multimédia, principalmente aqueles que giram a biblioteca multimédia do utilizador, precisam de carregar certas informações de ficheiros áudio e vídeo tais como a sua duração. Esta informação, para além do nome do artista, álbum, etc. provém de metadados incluídos nos próprios ficheiros. A TagLib é bastante leve, rápida e suporta metadados de diferentes formatos multimédia, incluindo as etiquetas ID3, comuns em ficheiros MP3. As bibliotecas de multimédia do Qt suportam o carregamento destes metadados mas requerem que o ficheiro seja carregado para reprodução, sendo um processo lento e complicado para a gestão de uma biblioteca multimédia.

Esta biblioteca encontra-se comummente instalada em sistemas Linux e é fácil de compilar e incluir juntamente com a aplicação em sistemas Windows. A sua inclusão em projetos do Qt Creator também é fácil.

Esta biblioteca encontra-se disponível em https://github.com/taglib/taglib e pode ser redistribuída e/ou modificada sob os termos da *GNU Lesser General Public License* versão 2.1 ou superior.

FFmpegthumbnailer Como recomendado pelo Material Design e comum na maioria dos leitores de vídeo e gestores de ficheiros, é necessário a apresentação de miniaturas (também conhecidas como *thumbnails*) dos vídeos. Esta biblioteca é leve e rápida a criar miniaturas, para além de permitir um grande controlo por parte da aplicação que a utiliza.

Esta biblioteca está dependente do libpng e do FFmpeg, que é habitual de se encontrar em sistemas Linux. O FFmpeg é um excelente descodificador e codificador de ficheiros multimédia, sendo utilizado por várias aplicações no Linux. A utilização do FFmpegthumbnailer no Linux é bastante simples, podendo ser incluída na própria aplicação devido ao seu pequeno tamanho.

No entanto, poderá não ser a melhor solução em versões para outras plataformas uma vez que está dependente do FFmpeg que está dependente de vários *codecs*. Para outros sistemas, o ideal poderá ser utilizar outro método, por exemplo, as bibliotecas de multimédia do Windows oferecem esta funcionalidade.

Esta biblioteca encontra-se disponível em https://github.com/dirkvdb/ffmpegthumbnailer e pode ser redistribuída e/ou modificada sob os termos da *GNU General Public License* versão 2 ou superior.

4.3 Interface Base

Como referido na secção "Tecnologias a Utilizar" (página 36), a interface gráfica é desenvolvida em QML. Uma janela em QML com o texto "Olá Mundo", que utiliza a biblioteca do Papyros tem o seguinte código:

```
1  import QtQuick 2.5
2  import QtQuick.Window 2.2
3
4  import Material 0.2
5
6  ApplicationWindow {
7    id: window
8
9    visible: true
10
11   title: qsTr("Window Title")
12
13   theme {
14     primaryColor: Palette.colors.orange["500"]
15     accentColor: Palette.colors.orange["A400"]
16   }
17
18   initialPage: Page {
19     id: root
20
21     Label {
22       anchors.fill: parent
23       text: qsTr("Hello World")
24     }
25   }
26 }
```

Excerto de Código 4.1: Janela de exemplo em QML com Material Design.

Repare-se que, tal como na documentação do Material Design, são definidas uma cor principal e outra de destaque. A classe `Palette` contém uma matriz `colors` com todas a paleta de cores do Material Design. O acesso a vetores e matrizes em JavaScript tanto pode ser feito com o operador [ ] ou com o ponto (.) o que torna o acesso à paleta mais intuitivo para o próprio programador.

Relativamente à internacionalização da aplicação, o Qt também fornece as ferramentas necessárias. A função `qsTr(sourceText, disambiguation, n)`[1] devolve o texto traduzido no idioma do sistema (ou definido manualmente pelo utilizador), esta função também gere corretamente plurais, de acordo com as regras do idioma. O segundo argumento desta função, aqui ignorado, indica ao tradutor o contexto atual, facilitando a compreensão ao tradutor.

Um comando do Qt permite gerar ficheiros de tradução *.tr* que serão utilizados pelos tradutores para traduzirem as diferentes `strings` da aplicação, estes ficheiros podem ser editados

[1] `tr(const char *sourceText, const char *disambiguation = Q_NULLPTR, int n = -1)` em C++.

pelo Qt Linguist ou com qualquer editor de texto simples. A versão compilada da aplicação incluí ficheiros binários com as traduções nos diversos idiomas.

A classe `ApplicationWindow`, da qual este exemplo se estende, faz parte da biblioteca do Papyros (sendo incluída no QML com `import Material 0.2`) e automaticamente coloca uma barra de ferramentas e uma pilha de páginas, estando inicialmente preenchida apenas com a `initialPage`.

Infelizmente, algumas das facilidades criadas com a `ApplicationWindow` não beneficiam a interface da aplicação deste projeto uma vez que esta classe não dá grande controlo da barra de ferramentas e não é possível colocar painéis livremente. Enquanto não houver uma versão da biblioteca com estas melhorias, toda a interface da aplicação terá de ficar dentro de uma página associada à propriedade `initialPage`.

4.3.1 Listas e Animações de Itens

Cada um dos painéis laterais é composto por uma `ListView` e cada uma delas apresenta itens provenientes de modelos implementados em C++. A implementação desses modelos será melhor explorada nas secções dedicadas à arquitetura do *backend*. Visto que podem ser adicionados e removidos itens dinamicamente na lista de reprodução e na biblioteca (por exemplo, o utilizador adiciona um novo artista ou a aplicação deteta uma nova música), o utilizador deverá ver essa alteração de forma animada. O código abaixo apresenta um excerto do código da `ListView` do painel da lista de reprodução:

```
add: Transition {
  id: lpaa
  SequentialAnimation {
    NumberAnimation { property:"opacity"; from:0; to:0; duration:0 }
    PauseAnimation {
      duration: (lpaa.ViewTransition.index - lpaa.ViewTransition.
    targetIndexes[0]) * 30
    }
    ParallelAnimation {
      NumberAnimation { property:"opacity"; from:0; to:1; easing.type:
    Easing.InOutCubic; duration:360 }
      NumberAnimation { property:"x"; from:-Units.dp(32); easing.type:
    Easing.InOutCubic; duration:380 }
    }
  }
}
remove: Transition {
  NumberAnimation { property:"opacity"; to:0; from:1; easing.type:Easing.
    InOutCubic; duration:260 }
  NumberAnimation { property:"x"; to:+Units.dp(32); easing.type:Easing.
    InOutCubic; duration:280 }
}
displaced: transitionItemsMoving
move: transitionItemsMoving
Transition {
  id: transitionItemsMoving
  NumberAnimation { property:"y"; duration:280; easing.type:Easing.
    InOutCubic }
}
```

Excerto de Código 4.2: Implementação das animações dos itens.

O `transitionItemsMoving` é um objeto do tipo `Transition` que apenas faz com que a coordenada *y* de cada item seja gradualmente alterada da posição inicial à final em 280

milissegundos (ou seja, $0,28s$) com uma interpolação `InOutCubic` sendo essa a mais aproximada à recomendada na documentação do Material Design. Os interpoladores mudam a velocidade da animação ao longo da mesma, evitando uma animação linear e "mecânica". Este objeto é associado à propriedade `displaced` que representa a animação a correr nos itens que terão de mudar de posição devido à adição e/ou remoção de outros itens e também associado à propriedade `move` que representa a animação a correr nos itens que forem movidos na lista.

A animação a utilizar nos itens a remover é definida na propriedade `remove`, neste caso o item ficará 0% opaco (ou seja, invisível) em 260 milissegundos e, ao mesmo tempo, será deslocado horizontalmente para a direita $32dp$ em 280 milissegundos, sempre com o interpolador `InOutCubic`. O deslocamento para a direita simboliza o item a sair da própria aplicação, uma vez que a lista já se encontra na direita da janela.

A animação mais complexa é a `add`, executada para cada item que for adicionado. Esta animação tem de raiz uma `SequentialAnimation` que, tal como o nome indica, executa cada uma das animações descendentes de forma sequencial. A primeira animação na realidade apenas serve para garantir que o item esteja invisível no início da animação. Depois é feita uma pausa, esta pausa depende da posição do item em relação ao conjunto de todos os itens que estejam a ser adicionados, quanto mais longe este item estiver dos primeiros do conjunto de novos itens, maior será a pausa. Esta pausa fará com que cada item entre na sua vez, tal como recomendado pelo Material Design. Por fim, são executadas duas animações ao mesmo tempo no `ParallelAnimation`, os itens ficam opacos em 360 milissegundos e deslizam na horizontal para a direita a partir de $32dp$ à esquerda da lista até à sua posição final. Esta animação simboliza o movimento dos itens a partir da área central ou do painel da biblioteca.

A figura 4.1 apresenta um momento da animação de adição de itens. Nesta figura é possível observar-se o efeito do interpolador: a velocidade é diminuída gradualmente, como é possível constatar pela maior proximidade dos itens que se encontram quase no fim da animação. No início da animação, a velocidade é aumentada gradualmente até a meio da mesma. Este interpolador confere uma animação mais natural e menos "brusca", sem obrigar o utilizador a esperar que a animação termine.

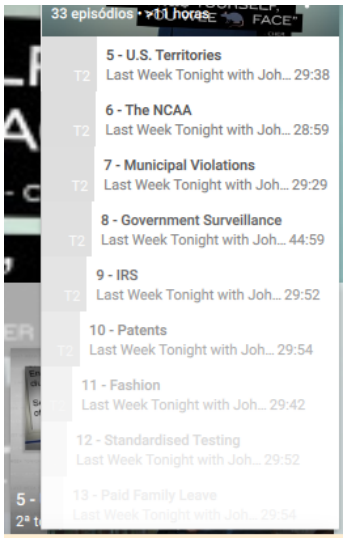


Figura 4.1: Animação de itens adicionados dinamicamente.

4.3.2 Controlo Multimédia através do QML

O controlo da reprodução multimédia será gerido em C++, portanto é necessário haver uma ligação entre ambas as camadas. A propriedade de contexto `context` é adicionada ao QML da seguinte forma, no ficheiro `main.cpp` durante a fase de arranque da aplicação:

```
1 MediaPlayerContext context;
2 QQmlApplicationEngine engine;
3 QQmlEngine::setObjectOwnership(&context, QQmlEngine::CppOwnership);
4 engine.rootContext()->setContextProperty("context", &context);
```

Excerto de Código 4.3: Introdução de uma variável no contexto do QML.

Esta variável é, na realidade, um objeto da classe `MediaPlayerContext` declarada e implementada em C++. Todas as propriedades e sinais declarados pela classe `MediaPlayerContext` podem agora ser usados em QML como se de um objeto em JavaScript tratasse. A linha 3 é importante neste exemplo concreto, ao indicar ao motor do QML que o C++ é o dono desta instância não haverá o risco do *garbage collector* do JavaScript destruir esta instância. O seguinte código apresenta parte do código do FAB da janela principal:

```
1  FloatingActionButton {
2    id: bPlay
3
4    z: 10
5    anchors.centerIn: parent
6
7    property real speed: context.playbackRate
8
9    opacity: barControls.anchors.bottomMargin < -barControls.height/2 ? 0 : 1
10   Behavior on opacity { NumberAnimation { duration: 120 } }
11
12   action: Action {
13     iconName: "av/play_arrow"
14     name: qsTr("Play") + " ("+qsTr("long press to change speed")+")"
15   }
16   alternateAction: Action {
17     iconName: "av/pause"
18     name: qsTr("Pause") + " ("+qsTr("long press to change speed")+")"
19   }
20   showAlternateAction: context.playbackState === MediaPlayer.PlayingState
21   animationDuration: 260 / context.playbackRate
22
23   onClicked: {
24     if(context.playbackState === MediaPlayer.PlayingState) {
25       context.pause();
26     } else if(context.hasMediaToPlay) {
27       context.play();
28     } else {
29       askUserToOpenMedia();
30     }
31   }
32 //(...)
33   onPressAndHold: popupSpeedSelector.open(bPlay, 0, 0);
34
35   onReleased: {
36     if(popupSpeedSelector.showing) {
37       popupSpeedSelector.close();
38       if(context.playbackRate !== speed) context.playbackRate = speed;
39       else clicked();
40     }
41   }
42
43   Popover {
44     id: popupSpeedSelector  // (...)
45   }
46 }
```

Excerto de Código 4.4: Código do botão de reproduzir/pausar.

Com este exemplo é possível compreender melhor o funcionamento do QML e ver vários exemplos de interação com a classe `MediaPlayerContext`. Esta implementação do FAB, desenvolvida para esta aplicação, suporta duas ações ao mesmo tempo, essas ações estão definidas na propriedade `action` e `alternateAction`. A propriedade `showAlternateAction`, *booleana*, indica se se deve apresentar a ação "*pause*" caso algum ficheiro esteja em reprodução (`context.playbackState === MediaPlayer.PlayingState`) ou então a ação "*play*". O valor desta propriedade será atualizado sempre que o estado de reprodução (ou seja, a propriedade `playbackState` do `context`) mudar.

Na linha 7 existe uma propriedade `speed` que terá o valor da propriedade `playbackRate` do `context`. Este valor é usado em algumas partes omitidas aqui e também na linha 38. Na linha 21 é definida a velocidade de animação com o valor `260/context.playbackRate`, esta animação é visível sempre que se alterna entre a ação principal e a alternativa, quanto mais lenta for a velocidade de reprodução, mais lenta será a animação.

Na linha 9 é definida a opacidade deste botão, mais à frente será explicada a razão pela qual este botão é ocultado. O importante aqui é o que acontece na linha 10, um `Behavior` define o que acontece quando o valor de uma propriedade muda, neste caso a propriedade `opacity` será animada com um `NumberAnimation`, desvanecendo o botão em 120 milissegundos sempre que a sua opacidade muda.

Na linha 23 é definido o que acontece quando o sinal `clicked` é emitido, ou seja, quando o botão for clicado/tocado. Ao contrário do que foi observado na análise do Google Play Music (página 24), este FAB executará sempre alguma ação. Caso haja algum ficheiro em reprodução, este será pausado. Senão, caso haja algum ficheiro na lista de reprodução, esse será reproduzido. Em último caso, é chamada a função `askUserToOpenMedia()` que abre uma caixa de diálogo para o utilizador escolher um ou mais ficheiros para abrir. Mais uma vez, há uma interação entre o QML e a classe `MediaPlayerContext` do C++.

4.3.3 Navegação

Outra parte importante da estrutura base da interface é a barra de ferramentas, no topo. Esta barra apresenta o caminho percorrido pelo utilizador, quando este sai da página inicial. A figura 4.2 apresenta um caminho possível.

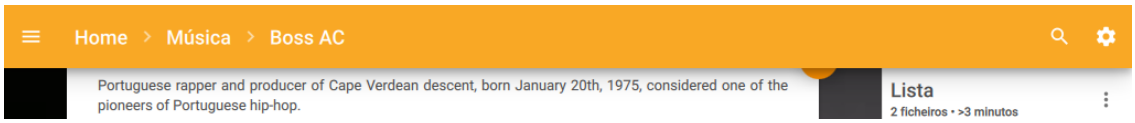


Figura 4.2: Exemplo de caminho e a sua representação na interface.

E o seguinte código apresenta as partes mais relevantes da implementação da barra de ferramentas[2]:

```
1 ActualToolbar {
2   id: toolbarMain
3
4   property bool bigPictureMode: navigator.currentItem ? navigator.currentItem.
      bigPictureMode : false;
5   backgroundOpacity: bigPictureMode ? 0 : 1
6   elevation: bigPictureMode ? 0 : 3
```

[2] Criou-se a classe `ActualToolbar` para ultrapassar algumas limitações na implementação original na biblioteca do Papyros.

```
7
8   customContent: Item {
9     id: navigationBarItem
10    TextField {
11      id: tSearchBox
12      readonly property bool open: activeFocus
13      opacity: open | navigator.depth <= 1 ? 1 : 0
14      Behavior on opacity { NumberAnimation { duration: 260 } }
15      placeholderText: qsTr("Search")
16      onTextChanged: {
17          var isSearch = navigator.__currentItem.isSearch;
18          if(isSearch) {
19            if(text.length == 0) {
20              navigator.pop();
21              tSearchBox.forceActiveFocus();
22            } else navigator.__currentItem.search(text);
23          } else {
24            if(text.length > 0) {
25              var pSearch = navigator.openPage("SearchPage");
26              pSearch.search(text);
27              tSearchBox.forceActiveFocus();
28            }
29          }
30      }
31      IconButton {
32        anchors.right: parent.right
33        action: Action {
34          text: qsTr("Close search")
35          iconName: "navigation/close"
36          onTriggered: tSearchBox.text = "";
37        }
38        opacity: parent.text.length > 0 ? 0.7 : 0
39        Behavior on opacity { NumberAnimation { duration: 200 } }
40      }
41    }
42    Row {
43      visible: navigator.depth > 1 && !tSearchBox.open
44      Repeater {
45        model: toolbarMain.bigPictureMode ? navigator.depth-1 : navigator.depth
46        delegate: Item {
47          width: label.implicitWidth + Units.dp(40)
48          height: navigationBarItem.height
49          Label {
50            id: label
51            anchors {
52              top: parent.top
53              bottom: parent.bottom
54              left: parent.left
55              leftMargin: Units.dp(8)
56            }
57            text: navigator.get(index).title
58            style: "title"
59          }
60          Icon {
61            id: arrow
62            visible: index+1 != navigator.depth
63            anchors {
64              verticalCenter: parent.verticalCenter
65              right: parent.right
66            }
67            name: "navigation/chevron_right"
68            opacity: 0.6
69          }
70          Ink {
71            id: mouseArea
72            enabled: index+1 != navigator.depth
73            anchors.fill: parent
74            onClicked: navigator.pop(navigator.get(index));
75          }
76        }
77      }
78    }
79  }
80
81  backAction: Action {
82    iconName: "navigation/menu"
83    name: qsTr("Library")
84    visible: !library.open
85    onTriggered: {
86      showLibrary();
87      library.wasOpen = true;
88      context.settings.librarySidebarOpen = true;
89    }
```

```
90    }
91
92    actions: [ Action {
93        iconName: "action/search"
94        name: qsTr("Search")
95        visible: tSearchBox.opacity == 0
96        onTriggered: {
97          tSearchBox.forceActiveFocus();
98          tSearchBox.selectAll();
99        }
100     }, Action {
101       iconName: "action/settings"
102       name: qsTr("Settings")
103       onTriggered: openSettings()
104     }]
105 }
```

Excerto de Código 4.5: Implementação da barra de ferramentas.

A propriedade `bigPictureMode` (linha 4) da `toolbarMain` terá um valor dependente da página atual, esta propriedade será melhor explorada mais à frente, no entanto é possível ver que, quando esta propriedade tem o valor `false`, a barra fica transparente e sem sombra (porque a elevação passa a ter o valor de 0). No fim do código são definidas as ações principais da barra: um botão de pesquisa, que só está visível quando a caixa de pesquisa `tSearchBox` estiver invisível e um botão de acesso ao ecrã de definições. Mais acima é definida a ação `backAction` que corresponde ao ícone da figura 4.2, no lado esquerdo. Esta ação só está visível quando o painel da biblioteca estiver oculto. Na linha 88 conserva-se a opção do utilizador em manter o painel aberto, mais uma vez, essa ação é feita em C++.

A propriedade `customContent` (linha 8) contém um item complexo que ocupa todo o espaço disponível na barra. Este item possui o campo de texto `tSearchBox` que só estará visível na página inicial ou quando tiver foco. A funcionalidade de pesquisa será melhor compreendida mais à frente.

Quando a `tSearchBox` não está visível, a `Row` estará. Este componente encarrega-se de dispor todos os seus elementos numa linha, horizontalmente. Dentro da `Row` existe um `Repeater`, os `Repeaters` criam *x* `delegates`, esse valor é definido pelo `model`, que tanto pode ser um número como pode ser um modelo complexo. Neste caso, o `Repeater` irá criar botões como os "Home", "Música" e "Boss AC", `navigator.depth` representa o número de páginas na pilha `navigator`. Cada botão apresenta o título da página (linha 57). O `Ink` é um componente específico da biblioteca do Papyros que implementa o efeito de gotas que faz parte do Material Design.

Na linha 25, é chamada a função `openPage()` do `navigator` para que seja colocada na pilha uma página de pesquisa na qual aparecerão os elementos que o utilizador estiver à procura. Esta função de abertura de página é a seguinte:

```
1  function openPage(pageName, replace, extraProperties) {
2    var component = Qt.createComponent("pages/" + pageName + ".qml");
3    if (component && component.status === Component.Ready) {
4      var properties = {"navigator": navigator,
5                        "toolbar": toolbarMain,
6                        "rightPanel": playlist,
7                        "uiState": uiState};
8      if(extraProperties) {
9        for(var name in extraProperties)
10         properties[name] = extraProperties[name];
11     }
12
13     var page = component.createObject(navigator, properties);
14
15     if(replace === true) while(navigator.depth > 1) navigator.pop();
16     push({item:page, destroyOnPop:true});
17
18     return page;
19   } else {
20     console.log("Couldn't load \"../pages/" + pageName + ".qml\": "
       + component.errorString());
21   }
22
23   return null;
24 }
```

Excerto de Código 4.6: Função de abertura de páginas.

Na linha 4 é criado um `array` com alguns componentes principais da interface, usados para que a página se ajuste de acordo com os componentes da interface e para também ter acesso a algumas funções da mesma. O uiState, também passado nesse `array`, para além de ter algumas propriedades só de leitura que refletem o estado da interface, evita a reutilização de código, oferecendo algumas funções que sejam comuns a várias páginas. As páginas são, na realidade, subclasses da classe QML `NavigatorPage` que já contém todas estas propriedades.

Caso a página a abrir esteja dependente de outros valores, estes serão passados para o mesmo `array` nas linhas 9 e 10. Finalmente, a página é carregada na linha 13, sendo-lhe passado o `array` com todos estes valores. Na linha 16 a página é adicionada à pilha, `destroyOnPop:true` garante que a página seja apagada da memória assim que for retirada, os objetos C++ associados a essa página também serão apagados caso o motor JavaScript seja o dono dos mesmos.

A navegação reflete uma hierarquia de páginas, no entanto, existem diferentes caminhos para aceder a certas páginas. Por exemplo, é possível aceder à página de um programa de TV diretamente da página inicial ("Home") mas também é possível acedê-la através da página de programas de TV ou da pesquisa. A funcionalidade de pesquisa, apesar de menos visível fora da página inicial, está disponível a qualquer momento, a pesquisa abrirá uma nova página sem destruir a navegação previamente realizada.

Devido a limitações na biblioteca do Papyros e/ou do estado atual do QML, a transição entre páginas não segue as recomendações do Material Design. A documentação sugere que as transições representem o surgimento da nova informação a partir dos elementos já existentes na página atual. Por exemplo, caso o utilizador escolha um programa de TV de uma lista, todos os outros itens deverão desaparecer exceto aquele que se deseja abrir, esse mesmo item poderá depois expandir-se de forma a ocupar todo o espaço da página.

4.3.4 Reprodução de Vídeos e Ecrã Inteiro

O `MediaPlayerContext` tem a propriedade `videoAvailable` que indica se, de momento, existe algum vídeo para ser apresentado ou não. Por exemplo, ao carregar um ficheiro MP3 não haverá vídeo. Sempre que este valor mudar a interface terá de se ajustar, para tal atualiza uma propriedade utilizada por vários componentes, incluindo as páginas, através do `uiState`.

A seguir, apresenta-se parte do código do botão que surgirá no lado direito da barra inferior sempre que o vídeo estiver disponível. É possível verificar que a variável `videoBehind` altera a opacidade e a margem da direita deste botão e que os mesmos serão animados. A ação associada ao botão também muda dependendo da mesma variável.

```
IconButton {
  color: barControls.iconColor
  enabled: videoBehind
  size: Units.dp(52)

  anchors {
    right: parent.right
    rightMargin: videoBehind ? Units.dp(24) : -width + Units.dp(24)
    verticalCenter: bPlay.verticalCenter
  }
  opacity: videoBehind ? 1 : 0

  Behavior on anchors.rightMargin {
    NumberAnimation { duration: 320; easing.type: Easing.InOutQuad }
  }
  Behavior on opacity {
    NumberAnimation { duration: 280; easing.type: Easing.InOutQuad }
  }

  action: Action {
    iconName: osdOpen ? "navigation/fullscreen_exit" : "navigation/
    fullscreen"
    name: osdOpen ? qsTr("Back to the video") : qsTr("Show navigator")
    onTriggered: osdOpen ? closeOsd() : openOsd()
  }
}
```

Excerto de Código 4.7: Botão de alternação do modo vídeo.

Quando estamos no modo de vídeo, todos os componentes escondem-se e o cursor do rato é ocultado. Infelizmente, não existe forma de controlar o rato em QML. A solução é fazer essa gestão em C++, por exemplo: adiciona-se um sinal no código QML da janela, acrescentado uma linha com "`signal setCursorVisible(bool visible)`", de seguida faz-se a ligação desse sinal a um *slot* em C++,

```
QQuickWindow * root = (QQuickWindow*) engine.rootObjects()[0];
connect(root, SIGNAL(setCursorVisible(bool)),
        this, SLOT(setCursorVisible(bool)));
```

Excerto de Código 4.8: Ligação de um sinal a um *slot*.

depois implementa-se o *slot*[3], também em C++,

[3]Nas versões mais recentes, o Qt também suporta expressões lambda, substituindo a necessidade de criar *slots*.

```
// no MainWindow.h :
class MainWindow : public QObject {
  Q_OBJECT
  // ...
  public slots:
    void setCursorVisible(bool);
  // ...
}
// no MainWindow.cpp :
void MainWindow::setCursorVisible(bool visible) {
  QGuiApplication::restoreOverrideCursor();
  if(!visible) QGuiApplication::setOverrideCursor(QCursor(Qt::BlankCursor));
}
```

Excerto de Código 4.9: Implementação de um *slot*.

e por fim, basta chamar no QML `setCursorVisible(true)` para mostrar o cursor ou `setCursorVisible(false)` para o ocultar.

Sempre que o utilizador mover o rato, a barra inferior deverá surgir, no entanto, a barra é bastante alta e um dos casos de uso será ver apenas o tempo atual da reprodução. Como solução, só uma pequena parte da barra deverá surgir e, quando o utilizador mover o rato para a barra, esta deverá surgir por completo. Visto que parte do FAB ficaria parcialmente visível, este deve ser ocultado, daí a linha 9 no código 4.4.

No topo também existe uma barra com o título do vídeo e botões de controlo, incluindo o de alternar entre o ecrã inteiro e o modo em janela. Numa versão futura, esta barra deverá aparecer de forma integrada com a moldura da janela, algo impossível de implementar corretamente num sistema X[4].

De forma a evitar que o utilizador tenha de esperar que passem *x* segundos para as barras se ocultarem, as mesmas desaparecem imediatamente após o utilizador mover o rato para fora da barra inferior. A implementação do controlo das barras, de acordo com os movimentos do rato, é a seguinte:

```
MouseArea {
  enabled: videoBehind
  property bool hideAsap: false;
  propagateComposedEvents: true
  hoverEnabled: true
  Timer {
    id: hidder
    interval: 1500
    onTriggered: {
        if(osdOpen || playlist.open) return;
        baseVideo.basicOsdVisible = false;
        if(parent.mouseY < root.height - barControls.height) {
          barControls.showing = false;
          barControls.peeking = false;
        }
        setCursorVisible(false);
    }
  }
  Timer {
    id: showUnlocker
    interval: 360
    onTriggered: parent.hideAsap = false;
  }
  onDoubleClicked: toggleFullscreen()
  onClicked: {
```

[4]Ver tópico Wayland vs. X (página 8).

```
26      barControls.peeking = true;
27      barControls.showing = true;
28    }
29    onMouseYChanged: {
30      hidder.stop();
31      if(osdOpen) return;
32      if(!hideAsap) {
33        baseVideo.basicOsdVisible = true;
34        barControls.showing = true;
35      }
36      if(mouse.y < root.height - barControls.height * 2) {
37        barControls.peeking = true;
38        if(hideAsap) {
39          baseVideo.basicOsdVisible = false;
40          barControls.showing = false;
41          barControls.peeking = false;
42          if(!playlist.open) setCursorVisible(false);
43          showUnlocker.start();
44        } else {
45          setCursorVisible(true);
46          hidder.start();
47        }
48      } else {
49        setCursorVisible(true);
50        barControls.peeking = false;
51        hideAsap = true;
52      }
53    }
54  }
```

Excerto de Código 4.10: Controlo das barras no modo de vídeo.

4.3.5 Feedback

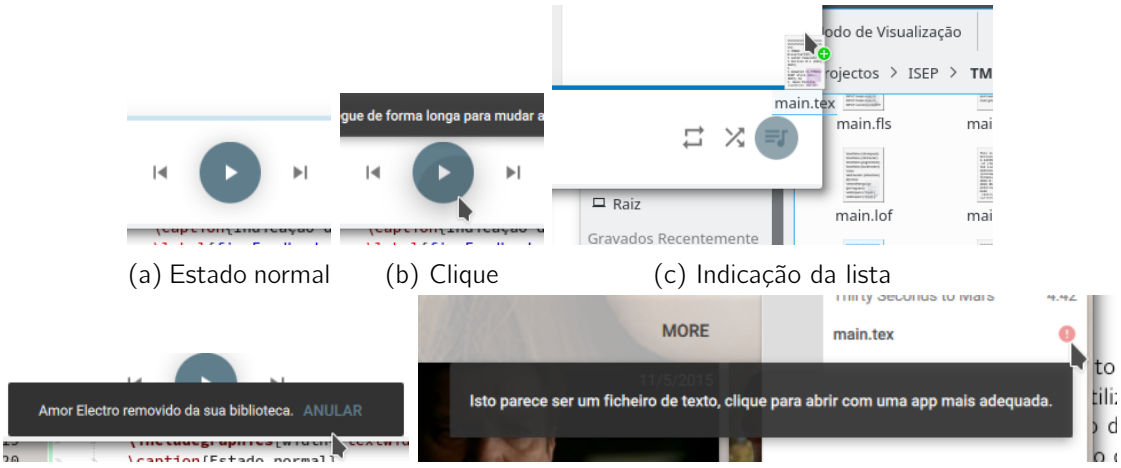
Para além do *feedback* já descrito aquando da adição e remoção de itens, existem outras situações em que as ações do utilizador lhe são confirmadas. O exemplo mais habitual no Material Design é a confirmação de cliques/toques em vários dos seus componentes, este *feedback* é feito através do efeito de uma gota que se expande, preenchendo gradualmente por completo o componente com o qual o utilizador interage. Outro exemplo do Material Design é a alteração da elevação de certos componentes: quando o utilizador interage com estes, a sua elevação aumenta, simulando uma atração magnética do botão ao dedo do utilizador, uma vez que existe um vidro entre o utilizador e a interface, não faz sentido simular o oposto. (Equipa de Design da Google 2015)

Estas duas reações à interação do utilizador estão exemplificadas mais abaixo, na figura 4.3a vê-se o botão no seu estado normal e na 4.3b um momento da animação para o seu estado clicado. Estes detalhes específicos do Material Design estão implementados na biblioteca do Papyros.

Em situações mais específicas da aplicação, existem diversas formas de confirmar ao utilizador as suas ações, sendo até possível permitir que as anule. Um exemplo é quando o utilizador remove um artista da sua biblioteca, para além da feedback em todas as listas nas quais o artista estava presente, um *snackbar* surgirá informando-o da remoção e oferecendo um botão para anular a ação (figura 4.3d). Quando o utilizador arrasta ficheiros para cima da lista de reprodução, o botão de mostrar/ocultar a lista é realçado, ajudando a confirmar que o ficheiro será adicionado à lista (figura 4.3c).

O *feedback* da adição de itens ao adicionar conteúdos multimédia à lista de reprodução com esta oculta torna-se irrelevante para o utilizador. Sem a lista visível, o utilizador não pode confirmar as suas ações através desse *feedback*. A solução está em recorrer ao mesmo efeito de realce referido no parágrafo anterior. Assim, não só verá a sua ação confirmada mas também é relembrado de que forma pode voltar a abrir a lista.

Quando ocorre algum problema a reproduzir um ficheiro, a aplicação informa o utilizador sem interromper o que estiver a fazer. O item que falhou é assinalado na lista de reprodução e a reprodução segue para o próximo item da lista. O utilizador poderá, quando quiser, verificar o que se passou passando o rato por sinal do sinalizador. Se possível, também poderá ser proposta uma solução para o problema, como é observado na figura 4.3e.



(a) Estado normal (b) Clique (c) Indicação da lista

(d) *Snackbar* (e) Erro

Figura 4.3: *Feedback*.

4.3.6 Escrita

O Material Design também tem as suas regras no que toca à escrita. O documento introduz o tema com a frase "O texto deve ser compreensível por qualquer pessoa, em qualquer lado, seja qual for a sua cultura ou idioma.". Apesar desta frase se aplicar mais concretamente à versão inglesa, pois é geralmente o idioma predefinido para os utilizadores que não possam utilizar uma determinada aplicação no seu idioma, não deixa de ser importante no desenvolvimento das traduções noutros idiomas.

A lista seguinte apresenta as recomendações do Material Design e exemplos adequados a este projeto, já na língua portuguesa:

Misturas de pessoas Deve-se evitar misturar a primeira com a segunda pessoas pois pode confundir o utilizador. *Mude as suas definições em A Minha Biblioteca.*

Segunda pessoa A segunda pessoa deverá ser utilizada em situações em que a aplicação está a transmitir uma informação diretamente ao utilizador. *O artista foi removido da sua biblioteca./O artista foi removido da biblioteca.*

Primeira pessoa do singular A primeira pessoa substitui a segunda quando se quer enfatizar o utilizador como dono de uma ação ou conteúdo. *Li e aceito os termos./Você leu e aceitou os termos.*

Primeira pessoa do plural A interface deve focar-se no utilizador e não em si mesma ou na sua empresa. *A sua biblioteca está a ser construída./Estamos a construir a sua biblioteca.*[5]

Concisão O texto deve ser curto e fácil de assimilar com uma rápida passagem dos olhos. *Procurar músicas e vídeos em/Pastas onde serão procuradas músicas e vídeos*

Presente Usar o tempo presente para descrever o comportamento do produto. Também evitar conjugações e modos verbais complexos. *Pode ouvir música./Poderá ouvir música.*

Simplicidade Ir direto ao assunto, evitar frases introdutórias desnecessárias. *Guardar alterações?/Deseja guardar as alterações?*

Vocabulário acessível Evitar termos que nem toda as pessoas poderão compreender corretamente. *Músicas e vídeos/Ficheiros multimédia*

Elementos da interface Nem sempre é necessário explicar se o objeto a interagir é um botão ou não. *Carregue no Ok./Carregue no botão Ok.*

Consistência Usar sempre o mesmo verbo. *A remover artista... Artista removido./A remover artista... Artista apagado.* Também, ao invés de um botão "Ok" ou "Sim", utilizar o mesmo verbo, neste caso "Remover", ajudando a confirmar a ação.

Objetivo primeiro Se a frase apresenta um objetivo e o meio para o atingir, começar sempre pelo objetivo. *Para remover o artista, carregue nos 3 pontos./Carregue nos 3 pontos para remover o artista.*

Respeito Se houver problemas, não chamar ainda mais à atenção. *Não é possível ler a música. Avançar para a próxima?/Desculpe! O codec não consegue carregar o ficheiro song.mp3 (mimetype /audio/mpeg).*

Humildade Não gabar a própria aplicação ou fazer promessas exageradas. Apresentar a função mas não dizer quão boa ela é. *Toda a sua música está aqui./Temos excelente música que vai adorar.*

Convidar o utilizador Incentivar o utilizador a experimentar funcionalidades em vez de entrar em detalhes técnicos. *Artista removido. [Anular]/Artista ocultado da sua biblioteca mas ainda armazenado na base de dados.*

Positivo Informar o utilizador sempre pela positiva. *Escreva o nome do artista./O nome do artista está em branco.*

Focar no essencial Para cada mensagem, a equipa de desenvolvimento deve-se perguntar "O utilizador precisa mesmo de saber isto?". *A sua biblioteca está a ser construída./A aplicação está a procurar ficheiros multimédia e a adicioná-los à sua biblioteca.*

Reduzir pontuações Nem sempre a pontuação é necessária e pode atrasar a leitura. *Tamanho dos textos e botões/Tamanho dos textos e botões:*

Números Mais uma vez, para facilitar a leitura, não se devem escrever os números por extenso. *1 episódio/Um episódio*

[5] Curiosamente, a Microsoft optou for fazer o contrário, por exemplo um dos primeiros textos apresentados ao executar o Windows 10 pela primeira vez é "*We're setting things up for you*" e o Outlook apresenta "*We didn't find anything to show here*" em listas vazias.

O capítulo sobre escrita do documento do Material Design termina lembrando a utilização de explicações contextuais para os tradutores (como já foi explicado para as funções `tr()` e `qsTr()` do Qt) e também com um lembrete de que nem todos os idiomas permitem uma escrita independente do género ("They are tall."/"Ele(a) é alto(a)").

Este tipo de escrita não só acelera a leitura como também facilita a comunicação com utilizadores menos experientes. Estas recomendações serão seguidas no desenvolvimento desta aplicação.

4.3.7 Problemas

Infelizmente, algumas ideias pensadas com o objetivo de melhorar a usabilidade tiveram de ser postas de parte. A classe `MouseArea` que faz parte da base do QML serve para executar determinadas ações dependendo do rato. Por exemplo, cada botão tem a sua própria `MouseArea`. O problema passa-se quando uma área dessas se sobrepõe a outra, a área que fica no topo absorve alguns dos eventos do rato.

Uma das ideias que melhoraria a usabilidade está no momento em que a interface muda automaticamente para o modo de vídeo. Nesse momento, o utilizador poderia estar, por exemplo, a utilizar um dos painéis laterais. Com um `MouseArea` por cima do painel, seria possível saber se o rato tinha sido movido nos últimos x milissegundos e impedir que apenas este painel fosse fechado até que o utilizador movesse o rato para fora do mesmo. Devido ao problema anteriormente indicado, ao colocar um `MouseArea` por cima do painel há problemas como por exemplo a falta de *feedback* quando o rato está por cima de itens da lista ou de botões ou pior: os *tooltips* não desaparecem depois de o rato sair de cima dos botões.

A `Toolbar` também apresenta uma falha de usabilidade que não é clarificada no documento do Material Design. Este problema tem a ver com a lei de Fitts. Quando a janela está em ecrã inteiro, existe uma barra no topo com opções. Numa boa implementação, seria possível arrastar o rato "infinitamente" até à borda superior do ecrã e depois escolher a opção, facilitando a coordenação motora necessária para a interação. Mas, todos os botões da `Toolbar` têm uma margem em relação às bordas dela mesma. Assim, o utilizador vê-se obrigado a mover o rato com mais cuidado para chegar à opção pretendida.

A utilização de C++ e JavaScript em conjunto também tem os seus aspetos negativos. O mecanismo de gestão de memória de ambas as linguagens é bastante diferente e a forma como o Qt funciona, apesar das suas intenções em facilitar a gestão de memória, pode trazer ainda mais problemas. A partilha de objetos entre C++ e QML é bastante comum em toda a aplicação. Em determinadas situações, os objetos passados do C++ para o QML podem passar a ser propriedade do motor de JavaScript. Isto significa que, quando esse objeto já não for necessário (no ponto de vista do QML), será eliminado da memória. Em várias situações, isto tira o peso do programador de ter de libertar objetos da memória mas também poderá resultar em falhas inesperadas. Sempre que se passar um objeto para o QML, deve-se compreender exatamente o ciclo de vida desse objeto e, caso este venha a ser utilizado no lado do C++ após deixar de ter utilidade no lado do JavaScript, indicar ao motor do QML que esse objeto será sempre propriedade do C++. Por falta de experiência e desconhecimento da transferência de propriedade, houve vários problemas que atrasaram o desenvolvimento da interface gráfica.

4.4 Camada de Dados

Apesar da decisão de se utilizar o SQLite para a gestão de dados, a aplicação foi desenvolvida de forma a ser independente dessa tecnologia. Para além dessa independência, deve ser possível estender a aplicação.

Todos as classes que representam algum conteúdo reproduzível, como por exemplo, músicas e programas de TV, são subclasses de MediaContent. Esta classe base fornece um número de identificação único para todos os MediaContents, um Uniform Resource Identifier (URI) com o caminho para o conteúdo a reproduzir, a duração do conteúdo, o seu nome e tipo. No caso geral, os URIs apontarão para ficheiros locais, no momento em que o conteúdo for carregado para reprodução, o URI poderá determinar se se deve usar o sistema multimédia do Qt ou outro específico que pode, por exemplo, ser fornecido com numa extensão da aplicação. O tipo é definido pela classe que implementar o MediaContent.

Atualmente existem as classes Song e Episode que representam músicas e episódios, respetivamente. Ambas estendem a classe MediaContent e têm alguns membros extra. Por exemplo, a Song possui um Artist artist e um List<Artist> featuring que representam o artista principal da música e a lista de artistas convidados, respetivamente. A classe Artist estende a classe MediaContentContainer que possui o método List<MediaContent>getMediaContents() que devolve todas as músicas associadas a esse MediaContentContainer. Tanto o MediaContentContainer como o MediaContent implementam a interface MediaProperty. Esta interface tem os métodos int id(), string name() e string type(). Tendo acesso a um ID e tipo é mais fácil saber como gerir os diferentes MediaProperty's, incluindo aqueles que vierem de extensões da aplicação.

A gestão de conteúdos multimédia é feito a partir da implementação da classe MediaContentLibrary, esta encarrega-se de carregar corretamente os MediaContents de acordo com o seu tipo. Cada implementação do MediaContent encarrega-se de carregar corretamente os MediaProperty's associados, por exemplo uma música trata de carregar o artista associado. No entanto, existe a classe Library que agrega a biblioteca de conteúdos multimédia mas também outras bibliotecas como por exemplo a de artistas e de programas de TV. O seu método MediaProperty getMediaProperty(string type, int id) trata de devolver o MediaProperty correto. Por exemplo, getMediaProperty("artist", 1) devolveria o Artist com o ID 1.

A declaração (simplificada) da classe Library é a seguinte:

```
1  class Library : public QObject, private LibraryItemRegistrationListener {
2    Q_OBJECT
3
4    public:
5      Library();
6
7      MediaContentLibrary * mediaContents();
8      MediaPropertyLibrary * mediaProperties(const QString & type);
9
10     virtual void onItemRegistered(const QString & type, MediaProperty * item) const;
11     virtual void onItemAdded(const QString & type, MediaProperty * item) const;
12     virtual void onItemRemoved(const QString & type, MediaProperty * item) const;
13     virtual void onItemRenamed(const QString & type, MediaProperty * item, const QString &
        newName) const;
14     virtual void onItemUnregistered(const QString & type, MediaProperty * item) const;
15
16     MediaProperty * getMediaProperty(const QString & type, int id);
17
18     Q_INVOKABLE
19     void deleteLibrary(bool onlyMediaContents, void * from);
20
21     virtual ~Library();
22
```

```
23    signals:
24      void onMediaPropertyRegistered(const QString & type, MediaProperty * item) const;
25      void onMediaPropertyAddedToLibrary(const QString & type, MediaProperty * item) const;
26      void onMediaPropertyRemovedFromLibrary(const QString & type, MediaProperty * item)
        const;
27      void onMediaPropertyRenamed(const QString & type, MediaProperty * item, const QString
        & newName) const;
28      void onMediaPropertyUnregistered(const QString & type, MediaProperty * item) const;
29
30    private:
31      static const short version;
32      QSqlDatabase db;
33
34      void createDatabase(QSqlQuery & q);
35
36    public:
37      static QString toString(const ImpreciseDate & date);
38      static ImpreciseDate toDate(const QString & date);
39  };
```

Excerto de Código 4.11: Declaração da classe Library.

Esta classe é um QObject, assim, para além de possuir certos metadados úteis, por exemplo para depuração, também é reconhecida pelo QML. A macro Q_INVOKABLE permite que o método void deleteLibrary () possa ser chamado a partir do QML. Estando a única instância desta classe agregada à classe MediaPlayerContext, é possível facilmente aceder à biblioteca a partir do QML. No entanto, as outras classes não partilham dos mesmos benefícios uma vez que os QObjects utilizam mais memória e ter, por exemplo, 200 artistas com várias músicas e 200 programas de TV com vários episódios poderia aumentar substancialmente o consumo de memória, toda essa gestão é feita de forma "tradicional" em C++.

Aqui é mais uma vez visível a utilização do tipo dos conteúdos/propriedades multimédia. Caso, por exemplo, um novo artista seja registado, a Library tratará de emitir o sinal com emit onMediaPropertyRegistered("artist", oNovoArtista). Tanto o C++ como o QML podem associar ações a executar no momento da emissão desse sinal, podendo assim atualizar todas as listas de artistas nesse momento. Há também a distinção entre registo e adição à biblioteca, a biblioteca pode ter conhecimento de uma música ou artista mas não os ter adicionados. Assim é possível manter dados associados a esses elementos sem estes terem de pertencer à coleção do utilizador.

O carregamento da base de dados é realizado da seguinte forma:

```
1  Library::Library() {
2    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
3    QStringList paths = QStandardPaths::standardLocations(QStandardPaths::AppConfigLocation)
       ;
4    QString dataPath;
5    if(paths.size() > 0) {
6      QDir configDir(paths[0]);
7      configDir.mkpath(".");
8      dataPath = configDir.filePath("library.sqlite");
9    } else {
10     dataPath = "library.sqlite";
11   }
12   db.setDatabaseName(dataPath);
13   if (!db.open()) {
14     qDebug() << "ERROR: Couldn't load the database!";
15   }
16
17   int version;
18   QSqlQuery q("SELECT value FROM metadata WHERE key='version'");
19   if (q.next()) {
20     version = q.value(0).toInt();
21   } else {
22     // the database isn't set up yet, let's set it up
```

```
23      createDatabase(q);
24    }
25
26    if(version < Library::version) {
27      // the database needs to be updated
28      qDebug() << "Updating the database from version" << version << "to" << Library::
        version << "...";
29      if(version <= 2) {
30        q.exec("ALTER TABLE mediaContents ADD COLUMN dateAdded INTEGER");
31        if(version <= 1) {
32          q.exec("CREATE TABLE artistsFeaturingMedia (id INTEGER PRIMARY KEY AUTOINCREMENT,
        artist INTEGER NOT NULL, media INTEGER NOT NULL)");
33        }
34      }
35      q.exec(QString("UPDATE metadata SET value=%1 WHERE key='version'").arg(Library::
        version));
36    }
37
38    this->db = db;
39  }
```

Excerto de Código 4.12: Carregamento da base de dados.

4.5 Multimédia

A reprodução de conteúdos multimédia é gerida, parcialmente, pelas classe de multimédia do Qt. A implementação dessas classes abstrai-se das diferenças entre os diferentes sistemas operativos. No caso do Linux, faz uso do GStreamer. Dependendo da distribuição do Linux do utilizador, é bastante provável que suporte os formatos multimédia mais conhecidos. Já nas versões mais recentes do Windows, é utilizado o Media Foundation que fornece uma gama mais limitada de codecs, impedindo o carregamento de alguns conteúdos multimédia. Segundo a documentação do Qt (*Qt 5.7 Multimedia Backends* 2016), a maioria das funcionalidades multimédia são suportadas em todos os sistemas nos quais o Qt está disponível.

Para casos de uso simples, a utilização desta biblioteca é bastante fácil: basta colocar o componente QML de vídeo e indicar o caminho para o conteúdo a reproduzir. Apesar de facilitar a implementação de funções de reprodução multimédia, a sua abstração impede a utilização de algumas funções. Sendo necessária a implementação manual para cada sistema, sendo por vezes impossível por não haver uma ligação direta entre a aplicação e o sistema multimédia utilizado.

No caso desta aplicação, o componente de reprodução multimédia encontra-se no lado do C++. Este componente (QMediaPlayer) suporta a reprodução através da indicação de um determinado caminho e também através de uma QMediaPlaylist. Esta classe representa uma lista de reprodução e cobre os casos de uso habituais em leitores multimédia. Apesar da simplificação para o programador, a aplicação utiliza uma classe própria (Playlist) que encapsula a gestão da QMediaPlaylist. Infelizmente, numa fase mais avançada no desenvolvimento da aplicação, notou-se que a alteração do conteúdo da lista da QMediaPlaylist enquanto esta está a ser utilizada pelo leitor faz com que a reprodução pare e se percam os valores necessários para retomar a reprodução (item atual e tempo). Atualmente, o problema está remediado, armazenando esses valores e redefinindo-os sempre que os conteúdos forem alterados, no entanto esta solução pode levar a cortes na reprodução. Caso os problemas não sejam resolvidos numa futura versão do Qt, poder-se-á substituir a utilização da QMediaPlaylist pela gestão completa da lista pela própria Playlist .

4.5.1 Metadados

Os metadados de um ficheiro multimédia são pequenos pedaços de informação que sobre esse mesmo ficheiro. No caso dos ficheiros de música MP3, é comum encontrar-se no ficheiros informações como o título da música, artista e álbum. Geralmente, todos os ficheiros têm a indicação de alguns detalhes mais técnicos, tais como a qualidade de imagem e som e duração do seu conteúdo.

A biblioteca multimédia do Qt permite a obtenção desses metadados, no entanto, apresenta uma grande limitação: o ficheiro tem de ser carregado num QMediaPlayer primeiro e depois, o processo que tem a intenção da obter os metadados terá de parar, à espera que o QMediaPlayer dê sinal de que o ficheiro está carregado e, só depois, poderá finalmente aceder, através desse QMediaPlayer, aos metadados. Esta solução é má por várias razões: do lado da equipa de desenvolvimento, o código tornar-se-ia confuso e mais difícil de gerir; em termos de hardware, a obtenção de metadados iria requerer, no mínimo, mais memória; para o utilizador, certas funções, como por exemplo a construção da biblioteca multimédia, demorariam mais tempo.

A solução foi utilizar uma outra biblioteca, a TagLib. A obtenção da duração de um ficheiro multimédia é efetuada de seguinte forma:

```
1 FileRef f(url.path().toStdString().c_str(), true, AudioProperties::Accurate);
2 const AudioProperties * ap = f.audioProperties();
3 qint64 length = -1;
4 if(ap) {
5   length = ap->lengthInMilliseconds();
6 }
```

Excerto de Código 4.13: Carregamento da duração de um ficheiro com o TagLib.

Na primeira linha, converteu-se o caminho de um QUrl para um **const char** * de forma a ser possível utilizá-lo com a TagLib. Para além do AudioProperties * audioProperties(), que devolve informações técnicas tais como a duração e qualidade de som e vídeo, também existe o Tag * tag() que devolve informações como o título da música, artista, etc.

A extração dos metadados é feita na classe MediaMetadataLoader que é atualmente utilizada em duas situações: construção da biblioteca multimédia; carregamento de informações de um ficheiro aberto na aplicação que não conste na biblioteca. O seu método MediaContent * load(**const** QUrl & url) instancia a implementação de MediaContent mais adequada. Estas implementações são diferentes daquelas devolvidas pelas classes de gestão da biblioteca, esta classe devolve MediaContents temporários. Apesar do seu caráter temporário, também poderão apontar para dados atuais da biblioteca, ou até mesmo registar novos.

O processo de carregamento de um novo MediaContent passa por diversas etapas, uma das primeiras é a determinação do seu tipo Multipurpose Internet Mail Extensions (MIME). Os tipos MIME (Freed et al. 2015), atualmente chamados de "tipos de média", identificam o formato de um ficheiro, a lista oficial de tipos é gerida pela Internet Assigned Numbers Authority (IANA). No caso de um leitor multimédia, os tipos que nos interessam são aqueles no grupo "*audio*" e "*video*". O Linux utiliza os tipos MIME para determinar o tipo de ficheiro, portanto o Qt dá acesso direto à base de dados de tipos de média do sistema. No Windows, a determinação é feita através da extensão do ficheiro, neste caso o Qt tem a sua própria implementação. A utilização dos tipos de média não só é mais prática, mas também mais precisa, uma vez que depende do conteúdo do ficheiro e não da sua extensão. Os tipos de média são utilizados da seguinte forma na aplicação:

```
const QString fileName = url.fileName();
const QMimeType mimeType = mimeDb.mimeTypeForUrl(url);

const bool audioOnly = mimeType.name().startsWith("audio/");
const bool hasVideo = mimeType.name().startsWith("video/");
```

Excerto de Código 4.14: Determinação do conteúdo de um ficheiro.

O valor das variáveis audioOnly e hasVideo é útil nas fases seguintes do carregamento. Assim, é possível, por exemplo, a possibilidade deste ficheiro ser um episódio ou um filme se o valor de audioOnly for verdadeiro.

Como nem todos os ficheiros indicam o título da música, o artista, etc. Este método também analisa o nome do ficheiro, procurando partes do nome na biblioteca. Caso encontre um artista e audioOnly for verdadeiro, assume-se que o ficheiro se trata de uma música e o título é o nome do ficheiro menos o nome do artista. Um raciocínio semelhante é aplicado a episódios de programas de TV, é utilizada a expressão regular `"[Ss](\\d+)[Ee](\\d+)"` para identificar padrões como "`NomeDaSerie.S03E10.mp4`". Depois, o nome da série é procurado na biblioteca, caso não seja encontrada, recorrer-se-á às extensões (como apresentado no capítulo Extensões, página 69) que poderão procurar a série na Internet. Depois, as extensões também são utilizadas para obter mais informações sobre o episódio, como por exemplo, o seu título. É possível que uma música tenha a participação de vários artistas, esses casos também são analisados no nome do ficheiro e nos metadados, sendo utilizada a expressão regular `"(.+)[ \\.,\\-'_]+[Ff]([Ee][Aa])?[Tt]([Uu][Rr][Ii][Nn])?[Gg]?[ \\.,\\-'_]*(.+)"`.

4.5.2 Listas de Reprodução

Guardar listas de reprodução é um caso de uso habitual nos leitores multimédia e também pode ser usado para melhorar a usabilidade: manter o estado da aplicação quando o utilizador a voltar a abrir mais tarde. As listas de reprodução da aplicação são guardadas no formato M3U, inicialmente desenvolvido pela sociedade Fraunhofer, para o seu software Winplay3 (Frankel, Sawyer e Greely 2002). Atualmente é suportado por vários leitores, incluindo os que foram analisados. O formato de um ficheiro M3U é bastante simples, composto por uma lista de ficheiros e/ou recursos online, separadas por parágrafos. Também é possível incluir comentários iniciando a linha com um cardinal (#). O método que guarda a lista de reprodução para um ficheiro neste formato é o seguinte:

```
bool Playlist::saveToFile(const QString & url) {
  QFile file(url);
  if (file.open(QIODevice::ReadWrite)) {
    QTextStream os(&file);
    os << "#EXTM3U" << endl;
    if(extraInfo) os << "#EXTPMPP:E:" << extraInfo->mediaPropertyId() << ":" <<
    extraInfo->mediaPropertyType() << endl;
    if(!title().isEmpty()) os << "#EXTPMPP:T:" << title() << endl;
    if(!description().isEmpty()) os << "#EXTPMPP:D:" << description() << endl;
    if(!imageUrl().isEmpty()) os << "#EXTPMPP:I:" << imageUrl() << endl;

    const int len = mediaCount();
    for(int i = 0; i < len; i++) {
      MediaContent * media = mediaAt(i);
      if(media == NULL) continue;
      os << endl << "#EXTPMPMC:" << media->id() << endl;
      os << (media->uri().scheme() == "file" ? media->uri().path() : media->uri().
    toString()) << endl;
    }
    file.close();
```

```
19
20      playlistUrl = url;
21      emit hasSourceUrlChanged();
22
23      return true;
24    } else {
25      qDebug() << "Failed to save playlist" << url;
26      return false;
27    }
28  }
```

Excerto de Código 4.15: Código simplificado da gravação de uma lista de reprodução.

As linhas de comentários são aproveitadas para estender as funcionalidades deste formato, como é visível das linhas 6 à 9, guardam-se o título e a descrição para além de outros dois elementos discutidos no final do capítulo Apresentação de Dados (página 66). Estas linhas não influenciam o carregamento do ficheiro por outros leitores. Na linha 15, armazena-se o ID associado ao ficheiro, caso este se encontre na biblioteca. Assim, caso o ficheiro tenha sido movido para outro local e a biblioteca tenha registado essa alteração, será possível localizar o ficheiro na mesma, evitando mais outro erro com o qual poderíamos interromper o utilizador desnecessariamente.

As listas são guardadas na pasta `playlists` na pasta de dados da aplicação. A lista de reprodução no momento em que a aplicação é fechada é guardada no ficheiro `state.m3u` na própria pasta de dados. Para além desse ficheiro, também é guardado mais outro com a posição na lista e tempo no momento em que a aplicação foi fechada.

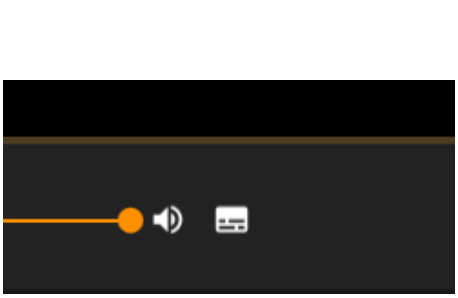
4.5.3 Legendas

Apesar do GStreamer suportar legendas, a biblioteca do Qt não suporta. A solução tomada foi implementar um sistema próprio na aplicação, colocando um elemento Text por cima do VideoOutput, no QML. Depois, esta etiqueta de texto será atualizada de acordo com o ficheiro das legendas. A aplicação suporta o formato SubRip (ficheiros `.srt`) e a implementação do carregamento deste formato encontra-se na classe SrtSubtitleLoader que devolve uma instância de Subtitles.

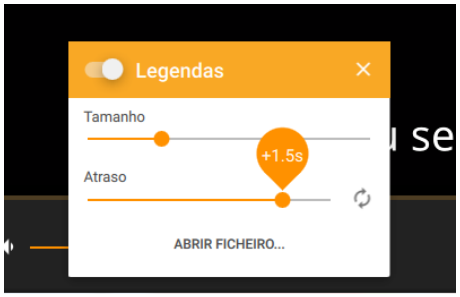
Aqui também se pensa na usabilidade da interface: O carregamento das legendas é feito assincronamente para não bloquear a interface. A aplicação procura automaticamente um ficheiro `.srt` com o mesmo nome do ficheiro em reprodução e carrega-o, se existir. O carregamento é feito da seguinte forma:

```
1  void SubtitlesAsyncLoader::onRun() {
2    if(fromUrl) {
3      SrtSubtitleLoader loader(url.path());
4      _subtitles = loader.loadSubtitles();
5    } else {
6      if(content->hasVideo()) {
7        QString path = content->uri().path();
8        int dot = path.lastIndexOf('.');
9        if(path.length() - dot < 5) path = path.replace(dot, path.length() - dot, ".
     srt");
10       SrtSubtitleLoader loader(path);
11       _subtitles = loader.loadSubtitles();
12     }
13   }
14 }
```

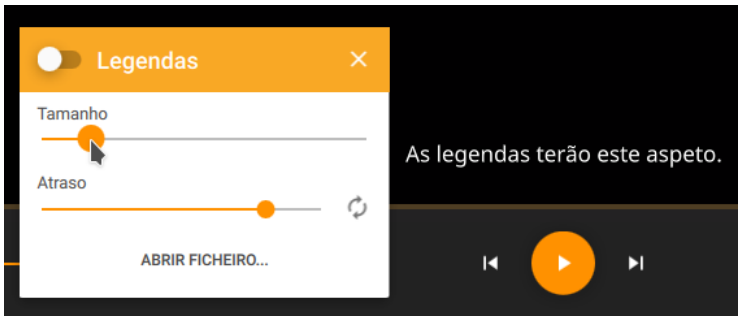
Excerto de Código 4.16: Código simplificado do carregamento assíncrono de legendas.



(a) Botão de legendas



(b) Caixa de controlo das legendas



(c) Alteração do tamanho das legendas

Figura 4.4: Legendas.

A interface que permite o utilizador controlar a legendagem também diferencia-se das que foram analisadas noutras aplicações. Na figura 4.4a é visível o botão que surge sempre que um vídeo esteja em reprodução, na maioria dos casos, não é preciso interagir com o botão, uma vez que as legendas são carregadas automaticamente.

Ao clicar neste botão, o utilizador pode carregar legendas manualmente mas também controlar o tamanho das letras e o atraso. Quando o atraso é ajustado, e de acordo com os padrões do Material Design, é apresentado o valor do atraso à medida que o utilizar o ajusta, como apresentado na figura 4.4b.

Ao alterar o tamanho de letra, o utilizador verá imediatamente o texto a ser ajustado, caso não haja nenhum texto no ecrã, será apresentado "As legendas terão este aspeto", como na figura 4.4c. A barra inferior nunca cobrirá as legendas, estas mover-se-ão para cima para que se mantenham legíveis.

De forma a apresentar as legendas, é necessário recorrer a um temporizador (QTimer) que chamará o seguinte método no intervalo de tempo que lhe for indicado:

```
1  void MainWindow::subtitleTimerTimeout() {
2    Subtitles * subtitles = context.subtitles();
3    if(subtitles == NULL) return;
4    qint64 pos = context.mediaPosition() + _subtitleDelay;
5    Subtitle * subtitle = subtitles->getAtCursor();
6    if(pos >= subtitle->start) {
7      if(pos >= subtitle->end) {
8        subtitles->moveToNext();
9        lSubtitle->setProperty("text", "");
10       if(subtitles->hasNext()) subtitleTimerTimeout();
11     } else {
12       lSubtitle->setProperty("text", subtitle->text);
13       if(context.playbackState() == QMediaPlayer::PlayingState) {
14         subtitleTimer.setInterval((subtitle->end - pos) / context.playbackRate());
15         subtitleTimer.start();
16       }
```

```
17      }
18    } else {
19      if(context.playbackState() == QMediaPlayer::PlayingState) {
20        subtitleTimer.setInterval((subtitle->start - pos) / context.playbackRate());
21        subtitleTimer.start();
22      }
23    }
24  }
```

Excerto de Código 4.17: Temporização e apresentação das legendas.

4.6 Apresentação de Dados

Como já foi falado no início deste relatório, o desenvolvimento das interfaces com o utilizador é várias vezes negligenciado. Mesmo havendo uma boa construção do *backend*, ou seja, a parte invisível para o utilizador, muitas vezes falha-se a chegar ao utilizador. Por vezes, a interface segue de forma rígida o modelo de dados, o que poderá não ser o mais natural para os utilizadores. Outras vezes, o modelo de dados (ou uma camada entre os dados e a interface) não contempla algumas das necessidades de uma boa interface. A modularização e reutilização de código, apesar de melhorarem todo o *backend*, podem baixar a qualidade da interface com o utilizador. No entanto, a culpa não é dessas práticas, mas sim do não desenvolvimento de classes específicas para a apresentação dos dados.

No caso desta aplicação, cada página terá, no mínimo, uma classe C++ associada que tratará de apresentar os dados de uma forma mais natural. Um componente muito utilizado nestas páginas é o QAbstractListModel que representa um modelo de dados e é utilizado, na maioria dos casos, por listas em QML ou nas interfaces Qt tradicionais. De forma a utilizar estes modelos em QML, devem-se implementar os seguintes métodos:

```
int rowCount(const QModelIndex& parent = QModelIndex()) const;
QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const;
QHash<int, QByteArray> roleNames() const;
```

O método rowCount() indica o número de elementos neste modelo, data() devolve a informação de um determinado campo para um determinado item, roleNames() indica o número de campos e o seu nome. O seguinte excerto exemplifica como implementar estes métodos:

```
QVariant Playlist::data(const QModelIndex& index, int role) const {
  int i = index.row();
  role -= Qt::UserRole;
  switch(role) {
    // ...
    case 2: {
      MediaContent * media = mediaContents.at(i);
      if(media) {
        const QString & type = media->type();
        if(type == "song") {
          Artist * artist = ((Song*)media)->artist();
          QList<Artist*> featuring = ((Song*)media)->featuring();
          if(artist) {
            QString result = artist->name();
            if(featuring.count() > 0) {
              result += " (" + tr("with", "Shown in the playlist for songs featuring
    someone else.") + " ";
              const int len = featuring.count();
              for(int i = 0; i < len; i++) {
                if(i == 0) result += featuring[i]->name();
```

```
20                else if(i+1 == len) result += " " + tr("and", "Shown in the playlist for
    songs featuring someone else.") + " " + featuring[i]->name();
21                else result += ", " + featuring[i]->name();
22              }
23              result += ")";
24            }
25            return result;
26          } else if(featuring.count() > 0) {
27            QString result;
28            const int len = featuring.count();
29            for(int i = 0; i < len; i++) {
30              if(i == 0) result += featuring[i]->name();
31              else if(i+1 == len) result += " " + tr("and", "Shown in the playlist for
    songs featuring someone else.") + " " + featuring[i]->name();
32              else result += ", " + featuring[i]->name();
33            }
34            return result;
35          } else {
36            return QVariant();
37          }
38        } else if(type == "episode") {
39          Season * s = ((Episode*)media)->season();
40          if(s) {
41            TvShow * ts = (TvShow*) s->parent();
42            if(ts) {
43              return ts->name();
44            } else {
45              return tr("Season %1").arg(s->number());
46            }
47          } else {
48            return "<unknown season and TV show>";
49          }
50        } else if(type == LoadingMedia::loadingMediaType) {
51          return "";
52        }
53      }
54    }
55    // ...
56  }
57}
58
59int Playlist::rowCount(const QModelIndex& parent) const {
60  return mediaContents.length();
61}
62
63QHash< int, QByteArray > Playlist::roleNames() const {
64  QHash<int, QByteArray> roles;
65  roles[Qt::UserRole] = "mediaName";
66  roles[Qt::UserRole + 1] = "title";
67  roles[Qt::UserRole + 2] = "description";
68  roles[Qt::UserRole + 3] = "duration";
69  roles[Qt::UserRole + 4] = "coverText";
70  roles[Qt::UserRole + 5] = "errorType";
71  roles[Qt::UserRole + 6] = "errorMessage";
72  roles[Qt::UserRole + 7] = "relatedProperties";
73  return roles;
74}
```

Excerto de Código 4.18: Implementação dos métodos necessários do QAbstractListModel para a apresentação da lista de reprodução.

Como apresentado na secção "Navegação" (página 49), a propriedade delegate indica o elemento gráfico a colocar para cada item do modelo. Em QML bastará escrever um dos nomes devolvidos no método roleNames() para aceder ao seu valor. Por exemplo delegate: **Label** { text: title }, apresentará, para cada item da lista de reprodução, uma etiqueta de texto com o título do ficheiro.

No método data(), é possível ver o que acontece quando o campo "*description*" é acedido. Numa futura versão, já com suporte a extensões para conteúdos multimédia, este código

deverá ser substituído com a utilização de um conjunto de classes (sendo algumas provenientes dessas mesmas extensões) encarregues de devolver uma descrição para um conteúdo multimédia de um determinado tipo. Neste exemplo também é possível ver a utilização do segundo argumento da função `tr`, indicando a razão pela qual se está a traduzir "*with*" e "*and*".

O método `data()` pode devolver qualquer valor desde que seja compatível com o `QVariant` no qual se incluem as subclasses de `QObject`. Assim, é possível devolver para o "relatedProperties" um mapa com diferentes elementos. Estes elementos são utilizados no menu de contexto do item na opção "Informação sobre" que lista os diferentes elementos relevantes do conteúdo multimédia, por exemplo, no caso de uma música aparecerá a própria música, o artista e os artistas convidados. A figura 4.5 apresenta dois valores devolvidos para o campo "description" e ainda a utilização do campo "relatedProperties" num menu de contexto.

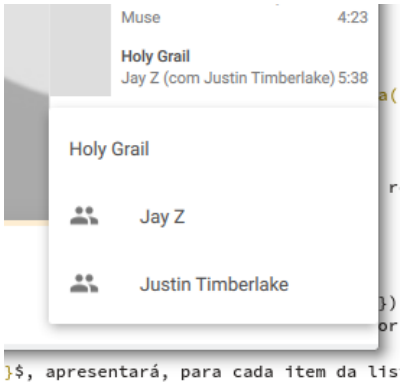
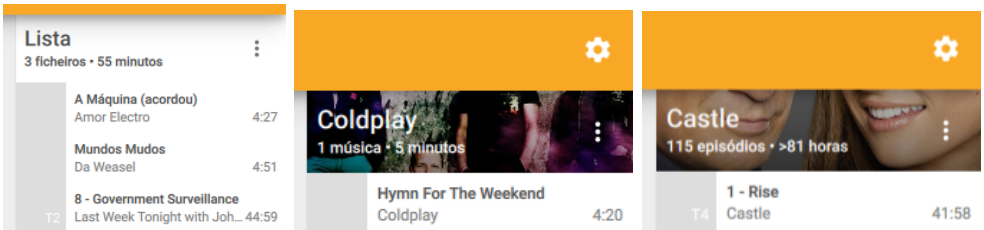


Figura 4.5: Suporte a diferentes associações de artistas e a sua representação na interface.

De forma a facilitar a compreensão dos conteúdos de uma lista de reprodução, o seu cabeçalho pode apresentar informação relevante sempre que possível. Por exemplo, ao adicionar músicas de um determinado artista, o título da lista mudará do seu nome predefinido para o nome do artista. Para além disso, a descrição e imagem também mudam de acordo com o conteúdo. Na figura 4.6 encontram-se três exemplo de cabeçalho, sendo o primeiro o predefinido. O título pode ser alterado pelo utilizador, no entanto, caso não o altere, o cabeçalho automaticamente decidirá o nome. Repare-se que, a descrição reflete o tipo de conteúdo multimédia incluído na lista.



(a) Conteúdo misto (b) Artista (c) Programa de TV

Figura 4.6: Diferentes cabeçalhos de lista automáticos.

Por vezes, e tal como foi observado no Amarok, não é possível obter a duração dos ficheiros no momento em que são adicionados à biblioteca ou à lista de reprodução. Nesses casos, a duração será omitida mas, como no momento em que o ficheiro é reproduzido já é possível

conhecer a duração do mesmo, a interface atualizará o item da lista com a duração correta e, se o ficheiro pertencer à biblioteca, esta nova informação será memorizada.

4.7 Extensões

A aplicação foi feita para funcionar de forma independente de qualquer serviço externo à aplicação e à comunidade open-source, no entanto, também foi feita a pensar na integração opcional com esses serviços. Devido às restrições de tempo, apenas se definiu dois tipos de extensão: Detetores de conteúdo; fontes de informação multimédia. O primeiro indica à aplicação se, por exemplo, uma determinada `string` é o nome de um artista ou não, o segundo fornece informações sobre artistas, programas de TV, etc. Apesar da sua distinção, uma única extensão pode implementar ambas as funcionalidades.

Foram desenvolvidas duas extensões que implementam ambas as funcionalidades: verificação da existência de e obtenção de informação sobre artistas através do serviço Discogs; verificação da existência de e obtenção de informação sobre programas de TV através do OMDb que já agrega informação de várias fontes. Como explicado no capítulo da Camada de Dados (página 59), existe a propriedade `type` que indica se o objeto em questão é, por exemplo, um artista ou um programa de TV. Esta informação é transmitida às extensões para que possam determinar se suportam ou não o tipo pedido.

O Qt facilita um pouco a criação de extensões ao abstrair-se das diferenças na utilização e implementação de bibliotecas dinâmicas entre os diferentes sistemas. O carregamento é feito após o arranque da aplicação e de uma forma assíncrona para não atrasar a interação do utilizador com a aplicação, o seguinte código procura e carrega as extensões e faz parte da classe `PluginManager` que trata de organizar e facilitar o acesso às diferentes interfaces para o acesso às funcionalidades das extensões:

```
1  qDebug() << "Looking for plugins...";
2  QStringList allPlugins;
3  QDir pluginsDir(QCoreApplication::applicationDirPath());
4  pluginsDir.cd("plugins");
5  lookForPlugins(allPlugins, pluginsDir);
6  lookForPlugins(allPlugins, QDir("/usr/lib64/papyrosmediaplayer/plugins"));
7  foreach(QString pluginFileName, allPlugins) {
8    QPluginLoader * loader = new QPluginLoader(pluginFileName);
9    QJsonObject metadata = loader->metaData();
10   if(metadata.count() > 0) {
11     loaders.append(loader);
12     qDebug() << " Loading" << metadata["IID"].toString() << ":" << pluginFileName;
13     foreach(QJsonValue imp, metadata["MetaData"].toObject()["implements"].toArray()) {
14       if(imp.toString() == "InformationDetector") informationDetectors.append(new Plugin
   <InformationDetectorPlugin>(loader));
15       else if(imp.toString() == "InformationSource") informationSources.append(new
   Plugin<MediaInformationSourcePlugin>(loader));
16     }
17   }
18 }
19 qDebug() << informationDetectorCount() << "information detectors loaded";
20 qDebug() << informationSourceCount() << "information sources loaded";
```

Excerto de Código 4.19: Carregamento de extensões.

Neste código é possível ver o carregamento de metadados de cada biblioteca e a agregação de extensões em listas separadas para cada funcionalidade. A classe `Plugin` encapsula o acesso a esses metadados e a instanciação das extensões. Fora da classe `PluginManager`, o acesso às extensões será sempre feita através de instâncias de `Plugins`.

A implementação de extensões tem alguns requerimentos. Um deles é o ficheiro de metadados, um simples ficheiro no formato JSON como o seguinte:

```
{
    "name": "Discogs",
    "description": "Looks for info about songs on Discogs",
    "implements": [ "InformationDetector", "InformationSource" ]
}
```

Excerto de Código 4.20: Ficheiro de metadados de uma extensão.

O outro requerimento está na definição da própria classe que serve de ponto de entrada para a extensão. Esta deve incluir algumas macros específicas do Qt. O seguinte excerto pertence à implementação do *plugin* que interage com os servidores da Discogs:

```
class DISCOGSINFORMATIONDETECTORPLUGINSHARED_EXPORT DiscogsInformationDetectorPlugin
  : public InformationDetectorPlugin, public MediaInformationSourcePlugin
{
  Q_OBJECT
  Q_PLUGIN_METADATA(IID "org.papyros.mediaplayer.plugins.discogs/1.0" FILE "manifest.json"
    )
  Q_INTERFACES(InformationDetectorPlugin)
  Q_INTERFACES(MediaInformationSourcePlugin)
```

Excerto de Código 4.21: Exemplo de declaração de uma extensão.

Em ambas as extensões, os servidores comunicam através de ficheiros JSON. O seguinte excerto ilustra a comunicação e processamento de uma resposta dada pelo servidor OMDb:

```
1  void OMDbInformationDetectorPlugin::getInformation(const QString & type, const QMap<
       QString, QVariant> & known, QMap<QString, QVariant> & info) {
2  const bool isEpisode = type == "episode";
3  const bool isTvShow = isEpisode ? false : type == "tvShow";
4  const bool isMovie = isEpisode || isTvShow ? false : type == "movie";
5
6  if(isEpisode || isTvShow || isMovie) {
7    const QString name = isEpisode ? known["tvShow"].toString() : known["name"].toString();
8    const QString typeToSearch = isEpisode ? "&type=episode" : (isTvShow ? "&type=series" :
       "&type=movie");
9    QJsonObject doc;
10   QNetworkAccessManager nam;
11   QNetworkReply * reply = isEpisode ? get(nam, "t="+QUrl::toPercentEncoding(name)+"&season
       ="+QString::number(known["season"].toInt())+"&episode="+QString::number(known["number"
       ].toInt())+"&plot=full") : get(nam, "t="+QUrl::toPercentEncoding(name)+typeToSearch+"&
       plot=full");
12   doc = QJsonDocument::fromJson(QString(reply->readAll()).toUtf8()).object();
13   reply->deleteLater();
14
15   if(!doc.isEmpty()) {
16     info["plot"] = doc["Plot"].toString();
17     const QString title = doc["Title"].toString();
18     if(!isEpisode || title != QString("Episode #") + known["season"].toString() + "." +
       known["number"].toString()) info["name"] = title;
```

Excerto de Código 4.22: Obtenção de informações sobre programas de TV e episódios.

O método `get()` comunica com o servidor e devolve a resposta do mesmo. Na linha 18 é visível o tratamento de uma situação específica deste servidor: quando o servidor desconhece o nome do episódio, este devolve um nome com o formato "`Episode #T.E`" sendo `T` a temporada e `E` o número do episódio. Visto que a aplicação gere por si mesmo situações nas quais o nome do episódio é desconhecido, a extensão ignorará esses casos.

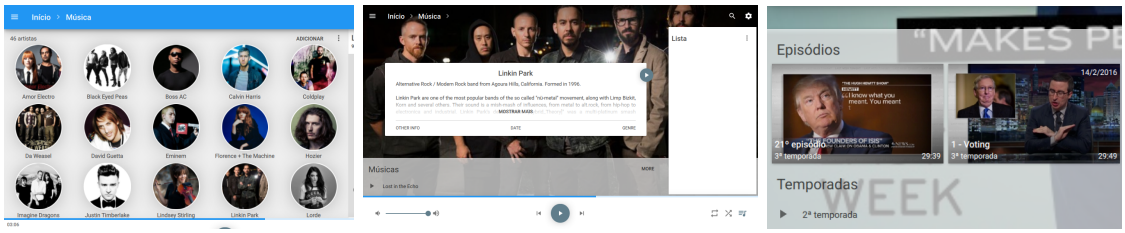
Ao longo de futuras versões, definir-se-ão novos tipos de extensão. O objetivo é permitir estender ainda mais funcionalidades da aplicação: leitores de conteúdos multimédia (para

o carregamento de conteúdos em formatos que o sistema base não saiba carregar), fornecedores de conteúdos multimédia (por exemplo, a integração com o Google Play Music e Spotify), fornecedores de legendas, fornecedores de letras de músicas, fornecedores de conteúdo para o ecrã inicial, fornecedores de soluções de erros de carregamento/reprodução de ficheiros, novos tipos de conteúdo multimédia (por exemplo, *podcasts*), fornecedores de grupos de pesquisa, entre outros.

Apesar de as extensões não terem acesso total à aplicação de forma direta, elas são executadas no mesmo processo onde a aplicação executa. Isto significa que as extensões têm acesso total à memória da aplicação, extensões desenvolvidas por pessoas mal intencionadas podem pôr em causa a segurança da aplicação. Caso se justifique, cada extensão deverá passar a ser executada num processo em separado. Para além disso, é possível impedir o acesso à memória secundária caso estes corram em processos separados.

4.8 Imagens

Tal como recomendado por Wickens, Gordon e Liu, o Material Design incentiva a utilização de imagens em várias situações. Na interface desta aplicação, é comum a utilização de imagens em vários locais, a figura 4.7 apresenta alguns desses casos.



(a) Lista de artistas (b) Página de artista (c) Miniaturas de vídeos

Figura 4.7: Diferentes utilizações de imagens.

A gestão de grandes quantidades de imagens é complicada. É preciso ter em conta as limitações do hardware, principalmente da memória principal (memória RAM), processador, memória gráfica e memória secundária (ex: disco rígido).

Na figura 4.7a vêm-se várias imagens em simultâneo, numa lista que contém ainda mais elementos do que aqueles visíveis. Estas imagens foram fornecidas pelas extensões, neste caso, vêm do Discogs. As imagens provenientes das extensões são armazenadas localmente (na memória secundária) durante um determinado período de tempo de aproximadamente um mês. Geralmente estas imagens estão no formato JPEG, as imagens desta figura ocupam, na memória secundária, aproximadamente 600 KiB, mas, ao serem carregadas para a memória principal ou gráfica, ocuparão muito mais espaço, uma vez que não haverá a compressão JPEG. Por exemplo, a primeira imagem nesta figura ocupa 28,6 KiB na memória secundária, mas, quando for carregada ocupará 4.036,032 KiB, já excluindo o canal *alpha*. Nesta figura existem imagens que ocupam mais de 8 mil KiB e há neste exemplo 34 artistas, poderemos precisar de 272 mil KiB, ou seja, mais de um quarto de um gigabyte. Para além dos problemas a nível da memória, o carregamento das imagens poderá demorar algum tempo, bloqueando a interação com o utilizador. A situação complica-se ainda mais uma vez que as imagens terão de ser transferidas da Internet para que possam ser armazenadas localmente.

Felizmente, o Qt ajuda parcialmente a resolver a situação. Cada imagem pode ser carregada já com um tamanho definido. Uma vez que todas estas imagens são apresentadas em pequenos itens, as imagens serão redimensionadas antes de serem carregadas para a memória. O Qt automaticamente gerirá a memória utilizada pelas imagens, eliminando aquelas que já não tenham sido utilizadas há algum tempo.

Ainda há espaço para mais melhorias, esta a um nível do desenvolvimento da aplicação e extensões. Indicar o caminho para as imagens requer não é tão trivial como parece, este está dependente de caminhos específicos ao sistema, utilizador e da própria imagem em questão. Para além disso, a imagem poderá estar ou não disponível no armazenamento local. Por exemplo, uma imagem de um artista poderá estar em "file:///home/gil/.cache/Papyros Media Player/mediaImages/artistsdAPpStj" num sistema Linux mas em "C:/Users/Gil/ AppData/Local/Papyros Media Player/cache/mediaImages/artistsdAPpStj" num sistema Windows. É possível reutilizar várias linhas de código e encapsular, não só a determinação do caminho mas também o carregamento das imagens caso tenham de vir de uma fonte externa. A solução é a utilização de QQuickImageProviders, assim, basta passar um caminho "image://mp/artist/6" e a implementação do fornecedor de imagens tratará de processar esse caminho e carregar a imagem correta. Um fornecedor de imagens é instalado da seguinte forma:

```
1 QQmlApplicationEngine engine;
2 engine.addImageProvider("mp", new MediaImageProvider(context));
```

Excerto de Código 4.23: Instação de um fornecedor de imagens.

No caso da implementação do MediaImageProvider, este estende o QQuickAsyncImageProvider que executa o carregamento de cada imagem assincronamente, não bloqueando a interface com o utilizador. Esta extensão terá de devolver uma implementação das interfaces QQuickImageResponse e QRunnable para cada caminho pedido. A seguir segue-se a implementação do método run() que é executado numa tarefa fora da *thread* da interface do utilizador e do método loadThumbnail() que carrega a miniatura de um vídeo:

```
1  void MediaImageResponse::run() {
2    if(provider->hasTask(id)) {
3      qDebug() << "Someone asked for the same image (" << id << ") more than once at the
       same time.";
4    }
5    provider->beginTask(id);
6    QMutexLocker lock(&waitBeforeDeleting);
7    QStringList path = id.split("/");
8    if(path.length() > 1) {
9      if(path[0].startsWith("thumb")) {
10       loadThumbnail(path[1].toInt());
11     } else {
12       loadMediaPropertyImage(path[0], path[1].toInt());
13     }
14   }
15   MIR_FINISH
16 }
17
18 bool MediaImageResponse::loadThumbnail(int mediaContentId) {
19   Library * library = this->library;
20   MediaContent * media = library->mediaContents()->get(mediaContentId);
21   if(media) {
22     QDir thumbsFolder(QStandardPaths::standardLocations(QStandardPaths::CacheLocation)[0])
       ;
23     thumbsFolder.mkdir("thumbnails");
24     thumbsFolder.cd("thumbnails");
25     QString thumbPath = thumbsFolder.filePath(QString::number(mediaContentId));
26     QImage image(thumbPath);
27     if(image.isNull()) {
28       try {
```

```
29 #ifdef MP_SUPPORT_FFMPEGTHUMBNAILER
30         qDebug() << "Generating new thumbnail for" << media->type() << media->name() << "(
       " << media->uri() << ")...";
31         ffmpegthumbnailer::VideoThumbnailer thumbnailer(360, false, true, 60, true);
32         thumbnailer.generateThumbnail(media->uri().path().toStdString(), Jpeg, thumbPath.
       toStdString());
33         image = QImage(thumbPath);
34 #endif
35       } catch (...) {
36         qDebug() << "Failed to load thumbnail for" << media->type() << media->name() << "(
       " << media->uri() << ")";
37       }
38     }
39     if(!image.isNull()) {
40       if(requestedSize.isValid()) image = image.scaled(requestedSize, Qt::
       KeepAspectRatioByExpanding, Qt::SmoothTransformation);
41       texture = QQuickTextureFactory::textureFactoryForImage(image);
42     }
43   }
44   delete media;
45   return true;
46 }
47
48 bool MediaImageProvider::hasTask(const QString & id) {
49   QMutexLocker locker(&tasksMutex);
50   if(tasks.contains(id)) {
51     tasks[id].waitCondition->wait(tasks[id].mutex);
52     return true;
53   } else {
54     return false;
55   }
56 }
```

Excerto de Código 4.24: Carregamento assíncrono de imagens.

Na linha 31 é utilizado o FFmpegthumbnailer para a criação da miniatura do vídeo. É criada e armazenada uma imagem JPEG com a qualidade a 60%, sendo suficiente para a utilização dada, evitando ocupar muito espaço na memória secundária. Por exemplo, 296 miniaturas armazenadas ocupam pouco mais de 13 mil KiB, ocupando cada uma, em média, 44 KiB apesar das elevadas dimensões. Na linha 40 a imagem é carregada nas dimensões pretendidas pelo componente da interface do utilizador.

Sendo o carregamento assíncrono, é possível haver situações em que uma imagem seja pedida para ser carregada várias vezes ao mesmo tempo. De forma a evitar essa situação, o MediaImageProvider tem o método hasTask() que verifica, de forma segura (*mutex* na linha 49), se o pedido já foi feito e ainda está a ser processado, bloqueando os pedidos repetidos até que possam continuar.

Infelizmente, o sistema de sinais e *slots* do Qt, que é obrigatório em comunicações em rede, complicando o código desnecessariamente, tem alguns problemas quando é utilizado entre vários processos em simultâneo, resultando em falhas na aplicação que a podem forçar a fechar. A implementação das extensões deverá ser revista numa futura versão.

4.9 Pesquisa

Na pesquisa, tal como em qualquer outra funcionalidade da aplicação, deve-se ter cuidado em evitar fazer o utilizador esperar e evitar qualquer situação que bloqueie a interface. A pesquisa é feita de forma imediata, mal o utilizador escreva qualquer caráter na caixa de pesquisa.

A pesquisa é constituída por diversos grupos, cada grupo efetua a pesquisa em artistas, programas de TV, músicas e episódios. Estes grupos não são fixos, extensões poderão fornecer novos grupos, por exemplo a possibilidade de pesquisar no YouTube. Relativamente à implementação da interface, na parte em QML da página existe o suporte para 4 diferentes tipos de apresentação: lista simples, pessoas, capas e miniaturas de vídeo. Cada grupo indica o tipo de apresentação adequado e também, tal como explicado no capítulo Apresentação de Dados, o modelo de dados. Cada vez que o utilizador escreve texto na caixa de pesquisa, todos os grupos serão notificados e irão, assincronamente, efetuar a pesquisa.

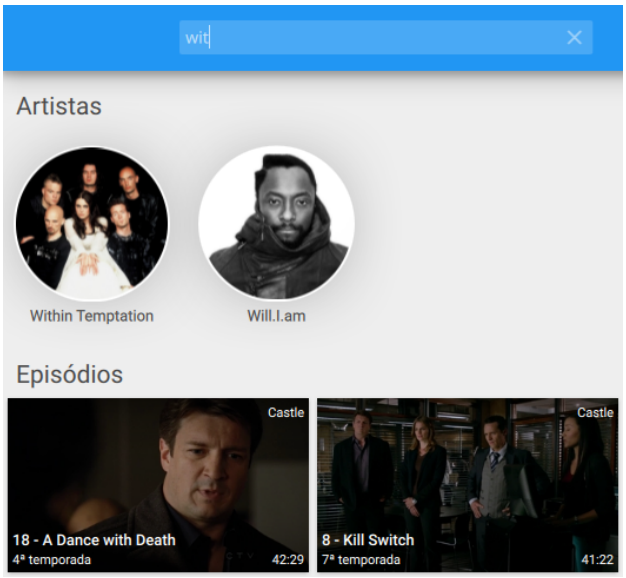


Figura 4.8: Exemplo de resultados de pesquisa para "wit".

É importante não esquecer que a introdução de texto pelo utilizador também é uma interação com a interface e, tal como qualquer outra interação, temos de contar com a possibilidade de erros por parte do utilizador. O utilizador poderá, por lapso, carregar numa letra errada ou não saber mesmo como se escreve corretamente o que estiver à procura. A solução é aceitar palavras semelhantes.

O algoritmo de distância de Levenshtein (1966) permite comparar dois textos, atribuindo um número que reflete a diferença entre ambos. Assim, é possível efetuar a pesquisa aceitando textos semelhantes. Os acentos também são ignorados, sendo as letras acentuadas substituídas pelas letras correspondentes sem acento. A figura 4.8 apresenta os resultados de pesquisa para "wit", os resultados surgiram instantaneamente, tais como os resultados para "wi" e "w", uma vez que a pesquisa é feita à medida que o utilizador escreve.

Na figura estão visíveis apenas os grupos que conseguiram devolver resultados na pesquisa. A implementação do primeiro grupo foi feita da seguinte forma:

```
1  class ArtistsGroup : public PeopleSearchGroup {
2    public:
3      ArtistsGroup(MediaPlayerContext * context) : PeopleSearchGroup(QObject::tr("Artists"))
       , artists(context->library()->artists()) { }
4      virtual QVariant data(int index, int field) const {
5        switch(field) {
6          case 0: return found.at(index)->id();
7          case 1: return found.at(index)->name();
8          case 2: return "image://mp/artist/" + QString::number(found.at(index)->id());
9        }
10       return QVariant();
11     }
```

```
12      virtual int count() const {
13        return found.length();
14      }
15      virtual void search(const QString & query) {
16        beginResetModel();
17        found = artists->lookForArtists(query, query.length()/3);
18        endResetModel();
19      }
20      virtual MediaProperty * getItem(int index) {
21        return found.at(index);
22      }
23    private:
24      ArtistsLibrary * artists;
25      QList<Artist*> found;
26  };
```

Excerto de Código 4.25: Implementação do grupo de artistas da página de pesquisa.

Sendo uma subclasse de PeopleSearchGroup, a sua implementação fica ainda mais simplificada. Parte da implementação do QAbstractListModel já é corretamente gerida pela superclasse SearchGroup, que é a superclasse comum a todos os grupos. Na linha 8 é visível a utilização do fornecedor de imagens, cujo seu funcionamento foi descrito na secção anterior.

4.10 Integração com o Sistema

Contrariamente ao Google Play Music, esta aplicação segue todos os padrões definidos pela organização freedesktop.org[6]. Graças a esses padrões, a aplicação integra-se corretamente com os diversos ambientes gráficos do Linux, não ficando apenas dependente do próprio Papyros.

Uma das partes mais importantes para a integração de uma aplicação com o sistema é o seu ficheiro `.desktop`. Este ficheiro descreve a aplicação e é utilizado para muitas das funcionalidades apresentadas mais à frente. O ficheiro `.desktop` da aplicação é o seguinte:

```
1  [Desktop Entry]
2  Name=Papyros Media Player
3  Exec=/usr/bin/papyrosmediaplayer %U
4  GenericName=Media Player
5  GenericName[pt]=Leitor Multimedia
6  Icon=papyrosmediaplayer
7  Categories=Qt;AudioVideo;Audio;Video;Player;
8  MimeType=application/ogg;application/ram;application/vnd.rn-realmedia;application/x-
       quicktime-media-link;application/x-shorten;application/xspf+xml;audio/ac3;audio/basic;
       audio/mp4;audio/mpeg;audio/ogg;audio/vnd.rn-realaudio;(...)
9  StartupNotify=true
10 Terminal=false
11 Type=Application
12 X-DBUS-ServiceName=org.papyros.PapyrosMediaPlayer
13 X-DBUS-StartupType=unique
14 Actions=RestoreState;
15
16 [Desktop Action RestoreState]
17 Name=Play where I left
18 Name[pt]=Continuar onde parei
19 Exec=papyrosmediaplayer --restore-state-and-play
```

Excerto de Código 4.26: Ficheiro .desktop da aplicação (excerto).

[6] A freedesktop.org não é uma organização formal de *standards* mas é através da qual as equipas de desenvolvimento dos ambientes gráficos (Plasma, Gnome, Papyros, etc.) formalizam padrões de acordo com

Este ficheiro é instalado na pasta de aplicações, geralmente em `/usr/share/applications/`. Os ambientes gráficos irão automaticamente adicionar a aplicação no menu na categoria adequada (no Linux, o menu de aplicações está agrupado por categorias: multimédia, jogos, escritório, etc.) e associar a aplicação aos tipos de ficheiro indicados no `.desktop`.

Mas a utilização deste ficheiro não fica por aqui. Muitas das informações nele contidas são utilizadas para melhorar a interação com o utilizador. Por exemplo, no Plasma, quando o utilizador abre uma aplicação, se esta tiver "`StartupNotify=true`", aparecerá o ícone dessa aplicação a saltar ao lado do rato até que esta abra. Assim, o utilizador recebe o *feedback* de confirmação da sua ação, mesmo se a aplicação demorar a abrir. A barra de tarefas também indicará o carregamento da aplicação. No lado da implementação do programador, este terá que sinalizar que o carregamento da aplicação está concluído.

Estando a aplicação identificada como um leitor multimédia, a integração pode ir ainda mais longe. O D-Bus é um sistema de transmissão de mensagens entre aplicações, permitindo que estas comuniquem entre si. Algumas dessas comunicações estão padronizadas com o objetivo de serem utilizadas de forma compatível em várias situações diferentes. Também pela freedesktop.org, existe a especificação da interface D-BUS MPRIS que é utilizada pelos ambientes gráficos para detetarem leitores multimédia em execução.

O Qt tem um conjunto de classes de suporte ao D-Bus, específicas para Linux, facilitando imenso a comunicação com este sistema. Tal como na utilização de classes C++ em QML, o Qt faz uso das "`Q_PROPERTY`" para indicar ao D-Bus que propriedades existem. O código seguinte apresenta alguns pequenos excertos da classe que implementa a interface `org.mpris.MediaPlayer2.Player`:

```
class MprisMediaPlayerController : public QDBusAbstractAdaptor {
  Q_OBJECT
  Q_CLASSINFO("D-Bus Interface", "org.mpris.MediaPlayer2.Player")
  Q_PROPERTY(QString PlaybackStatus READ playbackStatus)
  Q_PROPERTY(QVariantMap Metadata READ metadata)
  Q_PROPERTY(double Volume READ volume WRITE setVolume)
  Q_PROPERTY(qlonglong Position READ position)
  Q_PROPERTY(bool CanGoNext READ canGoNext)
  Q_PROPERTY(bool CanGoPrevious READ canGoPrevious)
  Q_PROPERTY(bool CanPlay READ canPlay)
  Q_PROPERTY(bool CanPause READ canPause)
  Q_PROPERTY(bool CanSeek READ canSeek)
  Q_PROPERTY(bool CanControl READ canControl)

  public:
    double volume() const;
    void setVolume(double volume) const;
  signals:
    void Seeked(qlonglong Position) const;
  public slots:
    void Next() const;
    void Previous() const;
    void Pause() const;
    void PlayPause() const;
    void Stop() const;
    void Play() const;
    void Seek(qlonglong Offset) const;
    void SetPosition(const QDBusObjectPath& TrackId, qlonglong Position) const;
    void OpenUri(QString uri) const;
  private:
    MediaPlayerContext * context;
};
```

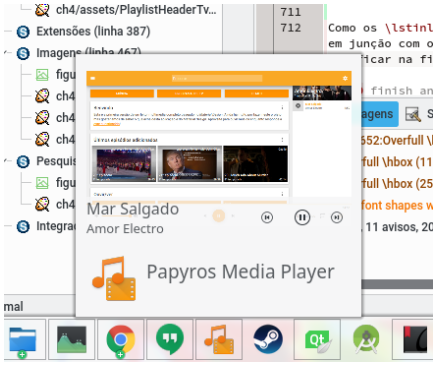
Excerto de Código 4.27: Excertos da implementação da interface org.mpris.MediaPlayer2.Player.

A especificação do MPRIS requer que sejam implementadas, pelo menos, as interfaces `org.mpris.MediaPlayer2` e `org.mpris.MediaPlayer2.Player`. Na linha 3 é indicado que esta classe implementa a interface `org.mpris.MediaPlayer2.Player`. Registar o serviço da aplicação é bastante simples, segue-se um excerto da classe responsável pelo carregamento desta funcionalidade:

```
1  QString name("org.mpris.MediaPlayer2.PapyrosMediaPlayer");
2  bool success = QDBusConnection::sessionBus().registerService(name);
3  if (!success)
4    success = QDBusConnection::sessionBus().registerService(name + ".instance" + QString::
       number(getpid()));
5
6  if (success) {
7    player = new MprisMediaPlayer(context, gui, this);
8    controller = new MprisMediaPlayerController(context, gui, this);
9    QDBusConnection::sessionBus().registerObject("/org/mpris/MediaPlayer2", this,
       QDBusConnection::ExportAdaptors);
10 }
```

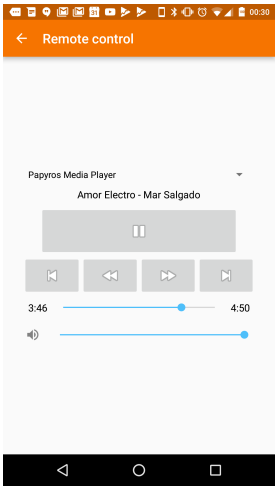
Excerto de Código 4.28: Registo de um serviço no D-Bus.

Como os QObjects suportam hierarquia por composição, o Qt automaticamente reconhece as duas implementações nos objetos player e controller. O ficheiro `.desktop`, em junção com o serviço D-Bus oferece ao utilizador um controlo mais fácil da aplicação, para além do suporte das teclas multimédia disponíveis em alguns teclados, também há outras vantagens como se pode verificar na figura 4.9.

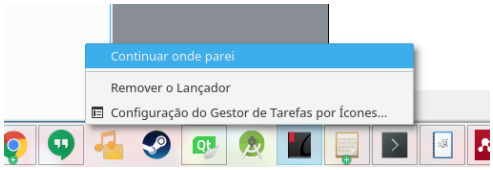


(a) *Tooltip* da janela.

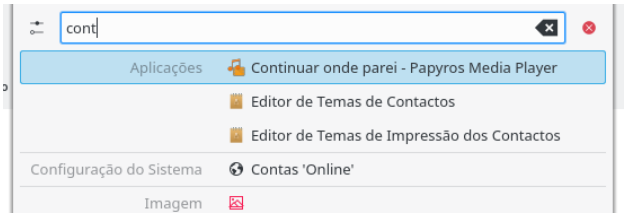
(b) *Widget* de controlo multimédia.



(c) Aplicação móvel "KDE Connect".



(d) *Desktop action*.



(e) Ação reconhecida no sistema de pesquisa.

Figura 4.9: Integração com o sistema.

as necessidades das interfaces gráficas. Esta organização contempla os *standards* que não façam parte do escopo de outras organizações formais, como por exemplo a Fundação Linux.

Graças ao registo do serviço MPRIS, a miniatura da janela (figura 4.9a), que aparece ao colocar o rato por cima do ícone da janela na barra de tarefas, automaticamente apresenta botões para controlar a reprodução, para além de informar qual música está de momento a ser ouvida. Junto ao relógio, aparece automaticamente um *widget* de controlo multimédia (figure 4.9b) sempre que um serviço MPRIS é detetado.

O KDE Connect (figura 4.9c) é uma aplicação móvel desenvolvida pela equipa do Plasma que melhora a integração entre os dispositivos móveis e o Plasma. Uma das suas funcionalidades é o controlo de qualquer leitor multimédia, desde que este registe um serviço MPRIS. Esta funcionalidade vai mais longe, melhorando ainda mais a interação entre o utilizador e ambos os dispositivos: caso o telemóvel receba uma chamada, a música será pausada e retomada automaticamente quando a chamada terminar. Algumas aplicações como o Amarok, VLC e Spotify também usufruem destas funcionalidades uma vez que registam no sistema uma implementação do serviço MPRIS.

Nas figuras 4.9d e 4.9e observa-se a utilização das linhas 14 à 19 no código 4.26. Esta ação permite o utilizador abrir a aplicação com a lista e estado de reprodução no momento em que a aplicação foi fechada. A lista também pode, opcionalmente, ser sempre recuperada no arranque.

4.11 Arquitetura

Uma vez compreendidos os aspetos mais importantes, é possível resumir a estrutura das diferentes camadas da aplicação no diagrama da figura 4.10. Nesta figura, cada camada e componente externo são representados por componentes na notação UML e cada classe, conjunto de classes, ou elemento abstrato é representado por um pacote também na notação UML. O acesso à camada de negócio através de camadas superiores é efetuado pela classe-fachada (padrão de software) MediaPlayerContext que organiza e gere as instâncias das classes desta camada e as de acesso à camada inferior.

Na camada de apresentação, ou seja a camada-interface com o utilizador, existe um conjunto de classes representado na "Camada de Apresentação de Dados". Estas classes são responsáveis, por exemplo, por formatar dados de acordo com os padrões do Material Design e de acordo com a informações esperada pelo utilizador. Nesta mesma camada também existe um conjunto de elementos QML específico desta aplicação que representam, por exemplo, episódios de séries. Uma vez que este elemento, entre outros, é utilizado em várias páginas, faz sentido disponibilizá-lo para todas as páginas, poupando código e mantendo a consistência, algo também importante para a usabilidade do sistema.

Apesar de não ser utilizado diretamente pelo utilizador, o serviço MPRIS compõe um outro ponto de interação com a aplicação para o utilizador. Este componente tem uma dependência opcional da janela principal uma vez que a especificação do serviço também contempla funcionalidades de controlo da janela do leitor.

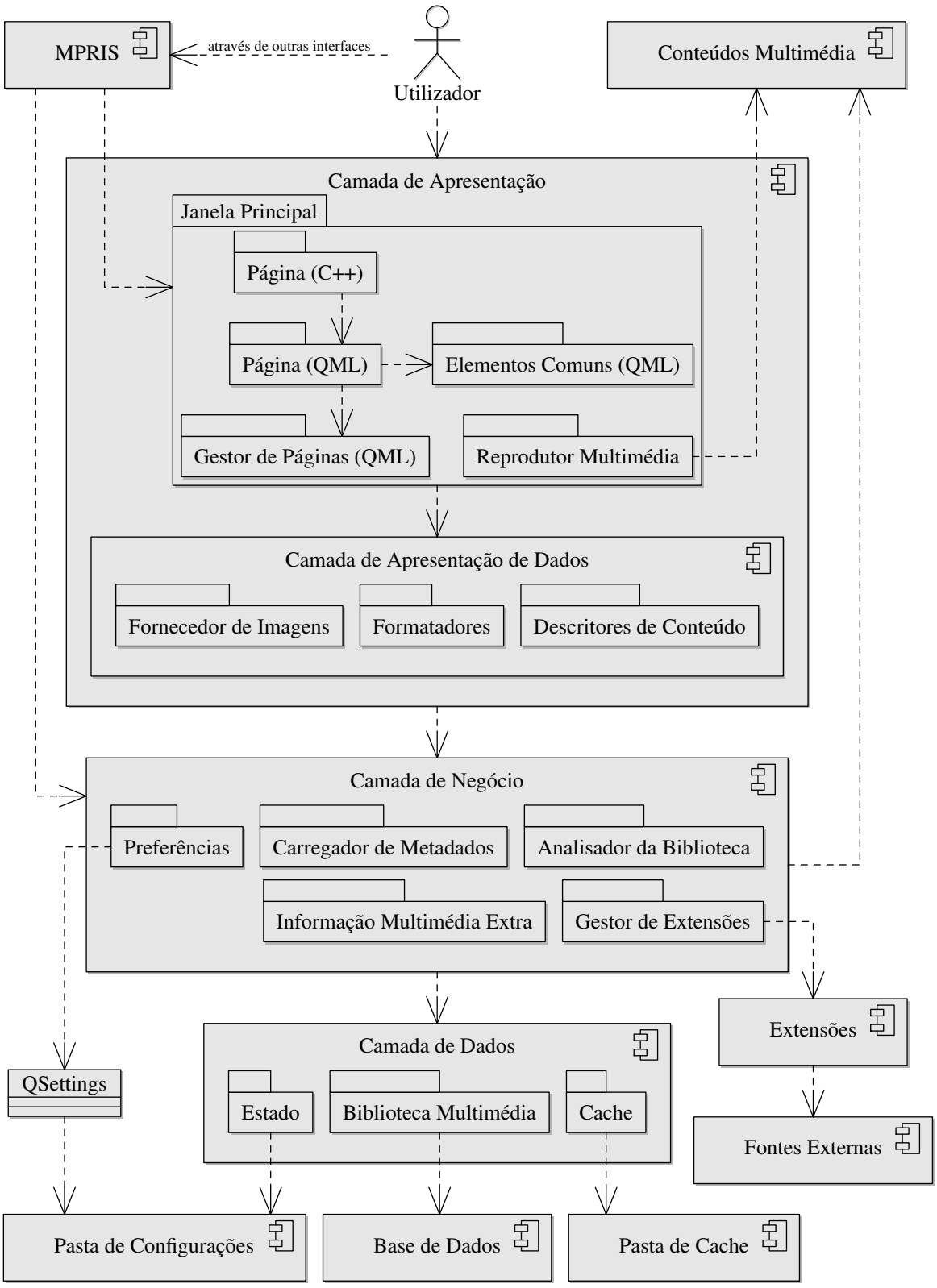


Figura 4.10: Diagrama das camadas do sistema.

4.12 Testes de Usabilidade

Estando a aplicação desenvolvida ao ponto de poder ser utilizada no dia-a-dia, é tempo de passar para os testes de usabilidade. O objetivo é realizar dois tipos de teste diferentes: inquéritos aos utilizadores; interação supervisionada.

4.12.1 Inquérito

Todos os utilizadores da aplicação verão uma ligação para o inquérito num cartão na página inicial, não sendo impedidos de remover esse cartão. O inquérito é composto por perguntas que ajudarão a determinar o perfil do inquirido. Assim, é possível agrupar as respostas de acordo com o conhecimento da pessoa em relação a interfaces gráficas e, mais concretamente, ao Material Design, sendo possível focar a análise mais nas pessoas que não estão tão à vontade com a interação com computadores.

Visto que a aplicação poderá ser utilizada por pessoas em todo o mundo, criou-se uma versão inglesa do formulário. Ambas as versões estão disponíveis no apêndice A. A aplicação irá automaticamente apresentar a ligação para o formulário adequado.

As opiniões e dificuldades de interação com a aplicação poderão variar bastante entre aqueles que estão habituados a utilizar computadores e os utilizadores esporádicos. Uma utilização rotineira da interface habitual de um computador poderá causar alguma resistência à mudança mas também pode reduzir possíveis dificuldades na interação. Portanto, as perguntas iniciais focam-se na obtenção de dados suficientes para agrupar diferentes perfis de utilizador. As perguntas seguintes focam-se nas impressões do utilizador ao primeiro contacto com a interface, seguindo depois para perguntas sobre a opinião de alguns pontos chave do Material Design. Já quase no fim, são colocadas questões sobre detalhes específicos da interface da aplicação, desenvolvidas de acordo com o que foi estudado no início deste relatório. O questionário finaliza com uma pergunta sobre a opinião da utilização do Material Design em computadores.

4.12.2 Interação Supervisionada

Na interação supervisionada, um pequeno grupo de pessoas irá, individualmente, utilizar a aplicação, executar determinadas tarefas e responder a algumas questões. Todo o processo será conduzido por mim, anotando as reações e respostas aos pedidos que forem feitos.

A sessão focar-se-á na avaliação da descoberta e compreensão das funcionalidades básicas da aplicação. Segundo Donald Norman, estas são as duas características do bom design. A aplicação é aberta e pergunta-se para que serve a aplicação. Apenas no caso de não ser dada uma resposta aceitável, explica-se para que a aplicação serve. De seguida, é pedido para que se ponha alguma música a tocar. Quaisquer passos dados serão aceites, desde que se consiga chegar ao objetivo. Caso sejam dados passos desnecessários, ou se demore a entender quais passos dar, o utilizador poderá estar perante uma falha de design.

Havendo uma música em reprodução, será pedido ao utilizador que execute tarefas habituais tais como pausar, retomar, recuar/avançar e também serão feitas perguntas que ajudarão a verificar se a interface é corretamente compreendida, por exemplo, perguntar-se-á em que minuto a música vai. De seguida verifica-se se o conceito de lista de reprodução é

compreendido. Uma vez que os passos dados para colocar uma música em reprodução podem ter já adicionado mais que uma música, esta parte estará dependente do conteúdo atual da lista. Nesta fase espera-se que o utilizador compreenda o que a lista representa e alguns conceitos básicos.

De seguida é pedido para o utilizador limpar a lista e colocar um episódio. Nesse momento, tal como explicado na secção "Reprodução de Vídeos e Ecrã Inteiro" (página 53), a interface altera-se, passando a estar apenas visível o vídeo. Aí, pergunta-se ao utilizador como poderá fazer para voltar a poder controlar a aplicação. Espera-se que o utilizador descubra, abanando o rato, que a barra inferior surgirá. Uma vez que é possível chegar a essa barra, pergunta-se como é possível mudar o tamanho das legendas. Apenas caso o utilizador não se aperceba que existem dois ícones novos na barra, dar-se-á essa mesma pista, se isto for necessário, então estaremos perante mais outra falha de design. Depois pergunta-se como é possível voltar ao ecrã anterior.

Todo o decorrer da sessão poderá estar dependente de vários imprevistos devido a dificuldades não reconhecidas no momento da construção e planeamento da interface ou de falhas informáticas ainda não corrigidas no software. Cada sessão poderá ter um guião diferente, no entanto, espera-se testar as interações acima mencionadas.

4.13 Conclusão

Neste capítulo desenvolveu-se o leitor multimédia e descreveu-se as partes mais relevantes do mesmo. Concluiu-se que era necessário despender-se bastante tempo na construção e implementação da camada de apresentação, aquela que fornece os componentes necessários para a interface com o utilizador. Nesta camada desenvolveram-se componentes visuais que representam, de uma forma mais fácil de compreender pelo utilizador, conteúdos multimédia, como por exemplo os cartões que representam vídeos, visíveis em várias partes da aplicação. Também nesta camada, foi necessário desenvolver interface gráficas que mantenham a consistência mas ao mesmo tempo consigam apresentar de forma adequada os diferentes tipos de conteúdo, como por exemplo as páginas de artista e programa de TV que, na realidade, são o mesmo componente mas apresentam grupos e tipos de conteúdo diferentes. Algumas classes responsabilizam-se por fazer pequenas tarefas, necessárias em quase todas as páginas da interface, como por exemplo os formatadores e o fornecedor de imagens. Concluindo, a camada de apresentação não pode ter menor importância do que as camadas inferiores, o utilizador não espera interagir diretamente com os objetos no seu estado "puro" mas sim de formas mais gráficas e que apresentem uma maior relação entre os diferentes objetos.

Relativamente ao desenvolvimento em geral, houve vários problemas, principalmente a nível da gestão de memória entre C++ e JavaScript, que atrasaram o desenvolvimento e levaram à exclusão de algumas funcionalidades. Para além desse atraso, alguns componentes devem ser revistos, como por exemplo os componentes de carregamento de imagens, uma vez que a aplicação por vezes falha provavelmente devido a um problema semelhante.

A nível da preparação para a avaliação de usabilidade, criaram-se os inquéritos e fez-se um guião informal para a interação supervisionada. O inquérito está acessível pela própria aplicação.

Capítulo 5

Distribuição

Este software está licenciado segunda a licença GNU Lesser General Public, versão 2.1 ou superior. Isto significa que o código fonte deste software é público e pode ser modificado, redistribuído, etc. No entanto, as liberdades associadas a este software não se perderão nas versões modificadas. Sendo o código público, também é possível que outros colaborem com o desenvolvimento do software.

O código fonte encontra-se num repositório GIT em https://github.com/gilfmc/MediaPlayer. O repositório contém as instruções gerais de compilação e instalação do software, no entanto, para outras plataformas poderão ser necessários alguns passos extra. Devido à enorme variedade de distribuições de Linux (openSUSE, Ubuntu, Arch, Debian, etc.), não é possível criar um pacote de instalação único que funcione em todas as distribuições. Estando o código público, outros utilizadores poder-se-ão voluntariar a criar os pacotes para as distribuições que utilizarem, no entanto, quem está à vontade a compilar o código e instalar diretamente não necessita de criar os pacotes.

5.1 Distribuição no openSUSE

O openSUSE utiliza pacotes no formato RPM. A criação destes pacotes requer um ficheiro spec (Streicher 2010) que indica os passos de compilação e instalação do software, este ficheiro é utilizado pelo software encarregue de criar o pacote e tem o seguinte conteúdo:

```
1  Summary: A media player using Material Design
2  Name: papyrosmediaplayer
3  Version: 0.1
4  Release: 1
5  License: LGPLv2
6  Group: Productivity/Multimedia/Video/Players
7  Source: papyrosmediaplayer.tar.gz
8  BuildRoot: /var/tmp/%{name}-buildroot
9  Requires: qml-material
10
11 %description
12 This is a complete media player that follow Material Design guidelines.
13
14 Under development.
15
16 %prep
17 %setup -q
18
19 %build
20 qmake-qt5
21 make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"
22 cd plugins/InformationDetectors
23 cd discogs
24 qmake-qt5
```

```
25 make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"
26 cd ..
27 cd OMDb
28 qmake-qt5
29 make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"
30
31 %install
32 rm -rf $RPM_BUILD_ROOT
33 make install INSTALL_ROOT=$RPM_BUILD_ROOT
34 cd plugins/InformationDetectors
35 cd discogs
36 make install INSTALL_ROOT=$RPM_BUILD_ROOT
37 cd ..
38 cd OMDb
39 make install INSTALL_ROOT=$RPM_BUILD_ROOT
40
41 %clean
42 rm -rf $RPM_BUILD_ROOT
43
44 %files
45 %defattr(-,root,root)
46 %doc LICENSE
47 /usr/bin/papyrosmediaplayer
48 /usr/share/applications/PapyrosMediaPlayer.desktop
49 /usr/share/icons/hicolor/192x192/apps/papyrosmediaplayer.png
50 /usr/share/papyrosmediaplayer/
51 %_libdir/papyrosmediaplayer/
52
53 %changelog
54 * Tue Aug 23 2016 Gil Castro <gilfmcastro@gmail.com>
55 - Initial release
```

Excerto de Código 5.1: Ficheiro spec de pacotes RPM.

Depois, o software de criação de pacotes RPM irá ler este ficheiro para então carregar o código fonte, compilá-lo, instalá-lo num sistema fictício e, finalmente, extrair o resultado da instalação nesse sistema. Os ficheiros instalados farão parte do pacote, em conjunto com metadados, como por exemplo as dependências do software. O ficheiro RPM estará pronto a ser instalado no openSUSE, no entanto, também poderá ser compatível com outras distribuições Linux que utilizem o gestor de pacotes RPM. Apesar de ser possível instalar o pacote diretamente, é mais habitual, e mais prático para o utilizador, a sua instalação através de um repositório de software. Estes repositórios fornecem todo o software do sistema e permitem que o processo de atualização do leitor multimédia seja gerido pelo próprio sistema operativo.

O openSUSE fornece um serviço de criação de pacotes na nuvem que recorre a máquinas virtuais que transferem o código diretamente do repositório GIT e seguem o ficheiro spec. Os pacotes gerados ficarão depois disponíveis num repositório de software, e assim é possível instalar o leitor multimédia com apenas uns cliques e sem quaisquer conhecimentos de programação ou de pacotes.

5.2 Distribuição no macOS X

A pedido de um utilizador deste sistema, e com a sua ajuda, foi possível criar uma versão para macOS X. Esta versão ainda não se integra corretamente com o sistema, por exemplo, as teclas multimédia não controlam o leitor e o sistema não abre músicas e vídeos com a aplicação automaticamente.

Sendo o projeto desenvolvido em Qt, em geral não é necessária qualquer alteração no código para que este funcione corretamente noutros sistemas. No entanto, as bibliotecas externas

poderão complicar a compilação do código. Portanto, devido às várias dependências do FFmpegthumbnailer, este não foi incluído na versão para macOS X, não apresentando as miniaturas dos vídeos. Já a TagLib foi incluída mas, contrariamente à maioria dos Linuxes, foi necessário compilar e instalar manualmente esta biblioteca.

Quando se compila a aplicação no Linux, o sistema de compilação do Qt (QMake) gera um ficheiro executável. Já no macOS X, gera uma pasta terminada em `.app`, mais conhecida por *bundle*. Dentro desta pasta deve existir o ficheiro executável e as suas dependências, o Qt fornece uma ferramenta que analisa o executável e tenta incluir todas as dependências automaticamente. Ao contrário do Linux, o *bundle* deve incluir todas as bibliotecas do Qt necessárias pois o Qt provavelmente não se encontrará instalado no sistema. A seguinte imagem apresenta o mesmo leitor em execução neste sistema, visto que está a ser utilizada a biblioteca do Material Design do Papyros, o aspeto é exatamente o mesmo.

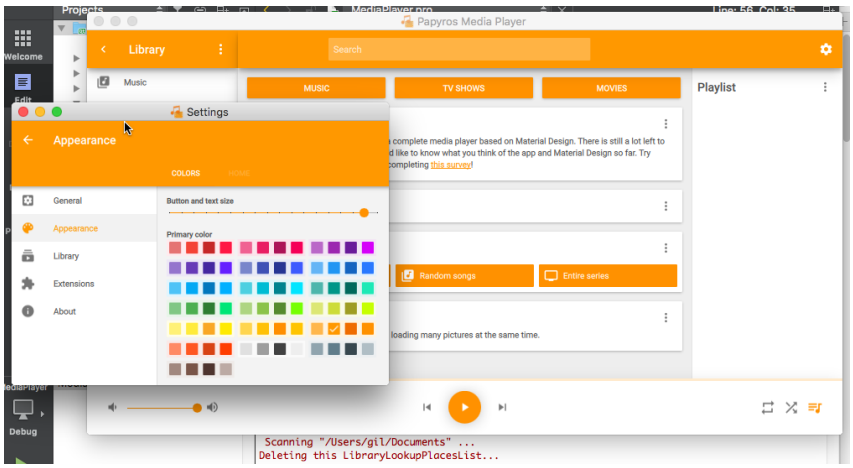


Figura 5.1: A aplicação em execução no macOS X.

Ainda há alguns problemas para corrigir, por exemplo, não é possível carregar os ficheiros de tradução (a não ser que se execute a aplicação através do Qt Creator) e a aplicação fecha inesperadamente em situações que não acontecem no Linux. Tanto estes problemas, como a má integração com o sistema poderão ser resolvidos numa futura versão.

5.3 Divulgação

Nas fases iniciais do projeto, contactou-se elementos da equipa do Papyros. Concluiu-se que o projeto se dividiu em dois: o próprio Papyros; o projeto Liri que consiste num conjunto de aplicações que utilizam a biblioteca do Papyros. Isto não está documentado no site do projeto nem na rede social na qual o projeto tem sido divulgado. Existem dois projetos muito simples de um leitor de vídeo e de um leitor de música associados a um dos elementos da equipa, no entanto, muito pouco conhecidos. Nestes contactos também se debateu algumas alterações propostas para a biblioteca do Papyros que serão mais tarde incluídas. Foi sugerido que este projeto fosse divulgado na comunidade do Papyros do Google+ e no reddit.

A divulgação do projeto apresenta vários obstáculos, não só as pessoas a par do projeto Papyros são poucas, como a maioria das pessoas não está habituada a compilar software

para depois o utilizar. A divulgação na comunidade do Papyros não parece ter tido qualquer efeito. Já no reddit, houve algum *feedback*. O projeto foi divulgado no *subreddit* de Linux e no de Material Design. Como era de esperar, principalmente neste último, muitas pessoas gostavam de testar a aplicação mas só está disponível o código fonte. Para além disso, parte dos leitores utilizam Windows e não foi possível compilar, a tempo, uma versão para Windows sem esta estar dependente do Qt Creator. Nesta mesma publicação, um leitor, apesar dos seus poucos conhecimentos de programação, ofereceu-se para ajudar a criar um pacote para macOS X.

Relativamente àqueles que não puderam testar a aplicação, a publicação tinha duas imagens da mesma em funcionamento e a opinião dos leitores foi positiva. No entanto, essa opinião pode não auxiliar na avaliação da usabilidade.

5.4 Conclusão

Neste capítulo passou-se à distribuição e divulgação do produto. Apesar de o Qt o permitir, não foi possível distribuir a aplicação no Windows, a aplicação corre corretamente dentro do IDE mas, quando executado fora, não abre e não dá qualquer *feedback*. No entanto, a aplicação funciona 100% corretamente em várias distribuições do Linux e é usável no macOS X. Para melhorar a compatibilidade com o macOS, será necessário *1*) compilar o FFmpegthumbnailer para macOS ou encontrar outra solução para a apresentação de miniaturas; *2*) melhorar a integração com o sistema (teclas multimédia, ícone da aplicação, associação a tipos de ficheiro); *3*) correção de falhas encontradas no macOS.

A divulgação do projeto não foi muito fácil, principalmente por não existir uma versão pronta a executar para Windows. No entanto, obteu-se bom *feedback* e curiosidade relativamente às imagens apresentadas da aplicação.

Capítulo 6

Avaliação de Usabilidade

Como referido no capítulo do desenvolvimento (mais concretamente na secção Testes de Usabilidade da página 80), criaram-se dois métodos para testar a usabilidade da interface: questionários e sessões de interação supervisionada. Agora, apresentam-se os resultados e a execução destes testes.

6.1 Questionários

Infelizmente, uma vez que não foi fácil a divulgação do projeto, não se obtiveram respostas suficientes para se chegar a conclusões sólidas ou sequer agrupar os resultados por diferentes perfis de utilizador. No entanto, apresentam-se os resultados obtidos, a população é de cinco utilizadores.

A primeira impressão em relação à interface pelos inquiridos não revela problemas de compreensão da mesma. Uns referiram que já estavam à espera e outros admitiram notar grandes diferenças em relação às interfaces a que estavam mais habituais mas que as diferenças eram positivas.

Os utilizadores não optaram todos por fazer o mesmo após a abertura da aplicação, por exemplo, alguns consultaram as definições e outros ficaram a ver a sua biblioteca multimédia a ser construída. Em todos os casos, todos concordaram que essas tarefas foram fáceis de executar. A navegação pela aplicação também recebeu boas críticas, sendo considerada fácil ou muito fácil.

Relativamente às animações, a opinião geral também é positiva. Um dos utilizadores não concordou que fossem interessantes mas concordou que faziam sentido e transmitiam informação extra sobre as ações tomadas pelo utilizador. Apenas um utilizador acredita que as animações nem sempre foram suficientemente rápidas para não atrasar a utilização da interface, tendo todos os outros referido que a duração das mesmas foi adequada. A opinião sobre a dimensão dos botões parece ser unânime: tinham as dimensões adequadas.

Apenas uma das respostas sobre a velocidade da aplicação e da funcionalidade de pesquisa não concorda que sejam rápidas. Isso, juntamente com a opinião diferente sobre a velocidade das animações pode indicar a possibilidade de existência de falhas técnicas na aplicação que não surgiram com mais nenhum utilizador.

Relativamente ao aspeto e organização da interface, todos tiveram opiniões positivas. Alguns detalhes diferenciadores, como a existência de um botão de repetir o conteúdo atual ao invés da utilização do botão de reproduzir o item anterior na lista, foram bem recebidos.

No entanto, há quem concorde que o botão deveria surgir mais cedo. Talvez seja boa ideia incluir uma opção nas definições para ajustar o comportamento deste botão segundo as preferências de cada utilizador. A minoria concorda que foi fácil voltar à interface completa durante a reprodução de um vídeo.

A pergunta final do questionário, que apresenta uma visão mais global sobre a utilização do Material Design em todo o sistema não recebeu respostas negativas, no entanto há quem refira que tem de ver primeiro para conseguir dar uma resposta mais concreta.

6.2 Interação Supervisionada

Relativamente à interação supervisionada, optou-se por um sistema iterativo. Dois utilizadores fazem parte da primeira iteração na qual se identificaram algumas falhas de usabilidade. Aquelas que podem ser resolvidas atempadamente e cuja solução não contradiga o Material Design, já não farão parte da segunda iteração. De forma a quantificar os resultados, para uma melhor análise, será utilizado um sistema de notas, tendo cada critério um diferente peso na avaliação final. Uma tabela com os resultados encontra-se mais abaixo, após a conclusão de todas as avaliações.

A interação supervisionada foi inicialmente realizada individualmente por dois utilizadores com alguns conhecimentos de informática mas culturas (Portugal e Estados Unidos, respetivamente) e sistemas (iniciado em Linux e iniciado em macOS, respetivamente) distintos. Durante ambas as sessões, foram testados principalmente as duas características do bom design de Donald Norman: descoberta e compreensão. Ao primeiro contacto, ambos reconheceram as funcionalidades da aplicação e onde executar as tarefas mais habituais.

Depois do primeiro contacto, foi-lhes pedido que colocassem alguma música em reprodução. A redundância criada pela barra lateral mostrou-se positiva uma vez que um dos utilizadores reconheceu os seus artistas nessa barra e interagiu com a mesma diretamente para ouvir música. Já no outro caso, o utilizador optou por navegar para um artista através da página inicial. Ambos conseguiram colocar músicas em reprodução rapidamente. No entanto, a apresentação de alguns elementos de grupos de conteúdos (músicas, álbuns, etc.), que podem ser vistos por completo carregando no botão "mais", não foi tão fácil de encontrar à primeira vez por um dos utilizadores.

Um dos utilizadores fez referência à imagem de fundo que surge nas páginas de artistas e programas de TV. Por vezes, estas imagens apresentam uma baixa resolução, reduzindo a qualidade da interface. A qualidade das imagens está dependente das extensões utilizadas, sendo de momento utilizado apenas o Discogs e o OMDb.

Outro problema deve-se à forma como a imagem se desloca à medida que o utilizador navega na página, apresentando partes menos relevantes da imagem. Também, devido ao Discogs, algumas descrições de artista contêm excertos que o utilizador não irá compreender, estes excertos são pequenos pedaços de código que representam referências a outros artistas, álbuns, etc.

A opção de repetir a música em vez de utilizar o botão de voltar para a música anterior foi benvinda por ambos os utilizadores, já um defende que o botão poderá estar sempre visível. As opções que permitem saltar diretamente para outro item da lista de reprodução quando o item atual termina e de repetir uma vez o item atual também foram bem recebidos, no entanto, o utilizador dos Estados Unidos sugere uma melhoria no texto de uma destas opções. A opção de anular o avanço e recuo no conteúdo multimédia em reprodução não foi percetível, no entanto um dos utilizadores considerou-a útil após a sua descoberta.

Relativamente à reprodução de vídeos, ambos também optaram por caminhos diferentes: um arrastou um vídeo; outro pausou a música atual, entrou na página de um programa de TV e depois escolheu um episódio. Antes da execução desta tarefa, a aplicação não foi reiniciada, o que significa que a interface para ambos os utilizadores não se encontra no mesmo estado.

Um dos utilizadores preferiu voltar ao início antes de executar a tarefa, para isso carregou no botão de fechar a barra lateral cujo ícone representa uma seta para a esquerda, criando uma possível confusão. As versões mais recentes da documentação do Material Design, lançadas após o início deste projeto, já descrevem diferenças na implementação de barras laterais que poderão solucionar esta confusão. Após o reconhecimento da falha pelo próprio utilizador, este executou os passos corretos para fazer o que pretendia.

Durante a reprodução do vídeo, ambos conseguiram instantaneamente aceder à barra inferior sempre que precisassem, também conseguiram facilmente controlar as legendas. Relativamente à opção de voltar à interface completa da aplicação, ambos revelaram enormes dificuldades no entanto, segundo as suas opiniões, o problema está apenas no ícone utilizado e não no mecanismo de utilização ou localização do mesmo.

A funcionalidade de pesquisa recebeu boas críticas por ambos os utilizadores, conseguiram encontrar o que procuravam e instantaneamente. Para finalizar, são colocadas algumas questões sobre a personalização da interface. Concluiu-se que ambos sabiam alterar as cores, no entanto um dos utilizadores esperava uma maior simetria no que toca à barra lateral e ao painel da lista de reprodução, tendo procurado ocultar o painel da lista de reprodução no seu próprio menu.

Após estes dois testes, é possível concluir que existem algumas falhas independentes da documentação do Material Design, pelo que, aquelas que não levarão muito tempo, serão corrigidas antes de avançar com novos testes. Estas correções são: *1*) Adição de uma opção para ocultar o painel da lista de reprodução através do seu próprio menu. Com este ajuste, utilizadores que se baseiem mais na simetria, encontrarão a opção onde acharão mais lógico. De forma a prevenir casos em que o utilizador não saiba como voltar a abrir a lista, o ícone no canto inferior direito piscará de forma semelhante àquela apresentada na figura 4.3c (página 56); *2*) Os ícones de alternar entre a interface de visualização de vídeo e a interface principal foram substituídos por um ícone representativo de vídeo e pelo ícone de "todas as aplicações" respetivamente. O ícone de "todas as aplicações" é constituído por vários quadrados, o que pode assemelhar-se aos conteúdos das páginas, para além disso, é a este ícone que utilizadores de interfaces em Material Design recorrem para procurarem aplicações quando estas não se encontram no ecrã principal. Anteriormente, estes ícones tinham dimensões superiores aos outros com o objetivo de chamarem à atenção, o que se provou ineficaz, portanto, de forma a melhorar a consistência, estes ícones agora apresentam-se num tamanho igual aos outros; *3*) O texto "Play after this one" foi substituído por "Play next". A tradução portuguesa "Tocar depois desta" foi substituída por "Tocar depois da atual" uma

vez que "esta" pode criar confusão entre o item selecionado na lista e o item atualmente em reprodução. *4*) A opção "mais", visível nos grupos de conteúdos foi substituída por "ver tudo" o que pode facilitar a sua compreensão.

Já com as alterações indicadas acima, passou-se à segunda ronda de testes. Desta vez, iniciou-se com um desafio maior, com alguém com mais de 70 anos que utilizou computadores há mais de 20 anos para tarefas simples com apenas o teclado em contexto profissional. Esta utilizadora possui um telemóvel Android com a versão 6, já baseada no Material Design, no entanto nunca fez utilização do leitor de música ou de vídeos.

Tal como nos testes anteriores, começou-se por se pedir para colocar uma música em reprodução, optou-se por carregar no botão "Música" no ecrã inicial e depois clicou-se num artista. Chegado à pagina de artista, notaram-se dificuldades em concluir a tarefa, clicando no título do artista e em algumas ocorrências do nome do artista na sua própria descrição. Depois dessas tentativas falhadas, a utilizadora verificou que se encontrava, mais abaixo, o nome de uma música, na qual clicou. A utilizadora não carregou no botão de reproduzir ao lado do nome da música, carregar apenas no nome adiciona-a à lista. Apesar de ainda não ter conseguido colocar a música em reprodução, reparou que a lista de reprodução havia mudado, aqui verifica-se a vantagem de pelo menos um dos detalhes implementados na lista: *1*) animações: o facto que o item surgiu deslocando-se do centro da interface para a lista poderá ter chamado o utilizador à atenção; *2*) cabeçalhos de lista automáticos: como apresentado na figura 4.6 (página 68), surgiu um cabeçalho representativo do artista adicionado à lista, isto poderá ter chamado à atenção também devido a uma enorme diferença em termos de luminosidade no lado lateral da interface. A descoberta da adição do item à lista levou o utilizador a concluir que devia carregar nesse item, e assim o carregou e conseguiu ouvir música. O facto que inicialmente a utilizadora não notou a existência da música na página do artista poderá dever-se a limitações no campo visual da própria utilizadora, algo que possivelmente apenas se melhora com a experiência. A utilização da interface de um telemóvel com um ecrã de cinco polegadas não requer observar uma área tão grande como a de um computador com um ecrã de mais de quinze polegadas.

De forma a confirmar a compreensão da interface por parte da utilizadora, optou-se por repetir o pedido mas com outro artista. Desta vez, optou por carregar no item "Música" no painel lateral da biblioteca, ignorando o facto que já se encontravam vários artistas por baixo desse item. De seguida, rapidamente clicou num outro artista sem qualquer hesitação mas, o problema observado inicialmente na página de artista surgiu novamente. Neste momento, verificou-se que a utilizadora não conhecia algo bastante importante num leitor multimédia: o ícone de reprodução de música (também conhecido como *play*). Felizmente, a disposição do ícone de reprodução na página do artista forneceu uma semântica que a levou a concluir o que o botão faria. Como se observa na figura 6.1, o botão encontra-se alinhado verticalmente com o nome do artista e aponta na direção da lista. Assim a utilizadora concluiu que iria juntar o artista à lista.

De seguida, pediu-se que se pausasse a música. Mais uma vez, notou-se limitações na exploração visual pela utilizadora, tendo percebido só neste momento que existia uma barra no fundo. Curiosamente, reconheceu o botão de pausa, movendo o rato até ao botão, neste momento surgiu um *tooltip* que a ajudou a confirmar que esse botão pausaria a música e assim conseguiu executar a tarefa corretamente. De seguida pediu-se para avançar para a próxima música na lista sem carregar na lista, a utilizadora conseguiu imediatamente clicar no ícone apropriado sem quaisquer dúvidas.

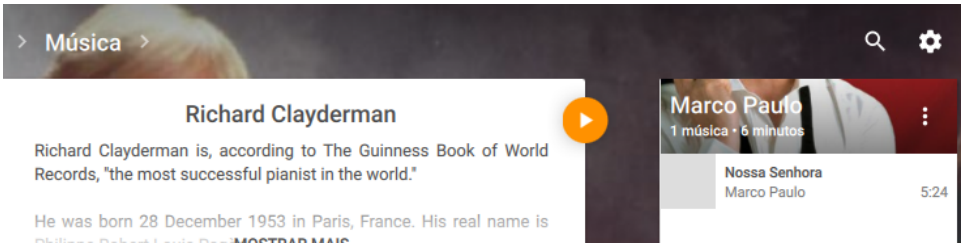


Figura 6.1: Botão de reprodução na página do artista.

Concluída a parte de testes na utilização de músicas, pediu-se que se limpasse a lista de reprodução[1]. A ideia inicial da utilizadora foi carregar no X da própria moldura da janela, para evitar fechar desnecessariamente a janela, indicou-se que esse não seria o passo correto. Compreendendo que essa era a opção errada, a utilizadora questionou "é aqui nestas três coisinhas?" referindo-se aos três pontos no cabeçalho da lista (também visível na figura 6.1), foi apenas respondido "não sei, experimenta". A utilizadora carregou então nesses três pontos e conseguiu limpar a lista. Passou-se então à segunda parte: reprodução de vídeos.

Após o pedido, a utilizadora voltou a socorrer-se da barra lateral para aceder à lista de programas de TV, de seguida clicou num programa, sendo levada para a página de programa de TV que, em termos técnicos é a mesma página que a página de artista e também se assemelha visualmente. Ou devido à aprendizagem através dos erros anteriores, ou devido ao facto que um item de um episódio é visualmente diferente do item de uma música, desta vez a utilizadora compreendeu rapidamente que devia clicar num desses itens para assistir a um episódio.

Já com o vídeo em reprodução, ou seja, com grande parte da interface oculta, pediu-se à utilizadora para "voltar a pôr todos aqueles botões e etc. que estavam aqui antes". A utilizadora reparou que a barra inferior e uma barra nova no topo surgiam mas demorou algum tempo a perceber que isso acontecia quando movia o rato. Quando compreendeu isso, começou por clicar no título do episódio no canto superior esquerdo, sem surtir efeito. Prosseguiu para os botões no conta superior direito, tendo aqui optado por ler todas os *tooltips*. Seguiu depois para o canto inferior direito no qual conseguiu apresentar a lista de reprodução, neste momento foi-lhe dito que "conseguiste fazer metade do que pedi, é possível mostrar mesmo tudo" e então foi finalmente clicado o botão ao lado que apresentou a interface completa. Infelizmente, neste momento a aplicação fechou inesperadamente, possivelmente devido ao recarregamento das miniaturas que mudam de escala na transição entre a interface completa e de vídeo, problema já discutido anteriormente. Reiniciou-se a aplicação e colocou-se no estado do momento da falha. Relembra-se que nesta iteração, mudou-se a forma de voltar à interface completa e que, comparativamente às sessões anteriores, e tendo em conta as suas dificuldades por falta de experiência de interação com computador e leitores multimédia, conseguiu rapidamente voltar à interface completa.

Depois, pediu-se que colocasse o vídeo em ecrã inteiro. Apesar de localizar o botão correto, notou-se que o tempo durante o qual os botões se encontram visíveis requer uma destreza bastante elevada para se conseguir carregar no botão. Passadas várias tentativas, foi possível colocar o vídeo em ecrã inteiro. De seguida, pediu-se para mudar o tamanho das legendas. Talvez por já ter compreendido que ações existiam nos cantos superiores e inferiores direitos, optou por começar pelo canto inferior esquerdo. Primeiro experimentou carregar na barra

[1] Note-se que durante estas sessões, tal como na escrita na própria interface da aplicação, a lista de reprodução é sempre referida simplesmente como "lista".

de volume sem qualquer efeito, depois tirou o som e repôs e, por fim, clicou no botão das legendas, tendo confirmado através do *tooltip*.

A utilizadora terminou a sessão dizendo que "isto é fácil, só é preciso usar algumas vezes". Visto que esta pessoa não teve qualquer experiência nas últimas duas décadas de utilização de computadores com ratos nem de ecrãs com mais de quinze polegadas e alta resolução, os resultados foram bastante positivos.

Já noutro teste, este realizado por um jovem já habituado a utilizar regularmente computadores e telemóveis, o guião foi complicado de seguir. Este utilizador optou por começar imediatamente a explorar a aplicação, tendo muito facilmente executado várias das tarefas ainda antes de serem pedidas. No entanto, na interface de vídeo, o facto que a barra inferior não está completamente visível fez o utilizador assumir que seria impossível regressar à interface completa quando um vídeo se encontra em reprodução.

A última sessão foi realizada por uma pessoa que raramente utiliza computadores e apenas em contexto profissional, tendo tido formação para utilizar um software específico. Esta utilizadora também possuí um telemóvel com Android 5, no entanto as modificações da Samsung ocultam muito do Material Design existente neste sistema operativo. Esta foi a primeira vez que utilizou um rato tátil, complicando a interação. A utilizadora compreendeu corretamente quais as funções desta aplicação e conseguiu facilmente ouvir música. A maioria dos controlos foram corretamente utilizados, no entanto a barra de progresso não era aparente. A utilizadora recorreu aos *tooltips* para localizar algumas da funcionalidades. Os três pontos na lista também foram facilmente compreendidos, tornando a tarefa de limpar a lista fácil de executar. Mais uma vez, as animações ajudaram a compreender a adição de itens à lista. Colocar a interface em ecrã inteiro demorou algum tempo a ser executado. Já voltar à interface completa foi bastante fácil, possivelmente tendo reconhecido o ícone. A personalização foi fácil de executar exceto encontrar o ícone das definições. Para finalizar, foi perguntado o que fazer para procurar um determinado artista caso tivesse uma lista bastante extensa de artistas, a utilizadora imediatamente explicou como o fazer escrevendo na barra de pesquisa o nome do artista a procurar (foi dado um artista de exemplo), logo ao introduzir a inicial surgiu esse artista tendo a utilizadora elogiado a facilidade.

A tabela 6.1 apresenta os resultados obtidos. Os critérios de avaliação foram agrupados e foram-lhes atribuídos pesos indicados na coluna P. De forma a simplificar a atribuição de pontos a cada critério, cada um tem uma escala de 0 ao valor indicado na coluna E. Na linha do total, na coluna de cada utilizador (identificados por U.), encontra-se o resultado final em valor percentual no qual 100% corresponde à nota máxima. Esta nota é calculada de acordo com a fórmula 6.1. Cada iteração apresenta a sua média, na última linha encontra-se a média do resultado de todos os utilizadores. A nota atribuída a cada critério é 0 quando representa tarefas que não foram executadas com sucesso, a velocidade com que as tarefas são executadas baixam a nota, podendo chegar ao 0.

$$R = \sum_{i}^{criterios} \frac{\frac{nota_i}{escala_i} \cdot peso_i}{\sum^{pesos}} \tag{6.1}$$

Tabela 6.1: Resultados da interação supervisionada

				1ª Iteração		2ª Iteração		
Grupo	**Critério**	**P**	**E**	**U. 1**	**U. 2**	**U. 3**	**U. 4**	**U. 5**
Geral	Compreensão básica da estrutura da interface	5	2	2	2	1	2	2
	Compreensão dos controlos básicos	5	2	2	2	1	2	1
	Capacidade de navegação	3	3	2	3	3	3	3
	Qualidade dos resultados de pesquisa	2	1	1	1	1	1	1
	Rapidez de pesquisa	1	1	1	1	1	1	1
	Qualidade do texto	3	2	2	1	2	2	2
Reprodução	Consegue reproduzir conteúdo	4	1	1	1	1	1	1
	Compreende a lista de reprodução	3	2	2	2	2	2	2
	Compreende funções extra da lista	1	1	1	1	1	1	1
	Compreensão e preferência do botão de repetir	1	1	1	1	1	1	1
Vídeos	Consegue aceder à barra inferior	4	2	2	2	1	1	2
	Consegue controlar as legendas	1	1	1	1	1	1	1
	Consegue sair do modo de vídeo na janela inteira	5	1	0	0	1	1	1
	Consegue entrar em ecrã inteiro	3	1	1	1	1	1	0
Imagens	Boa utilização de imagens	2	1	1	1	1	1	1
	Boa apresentação das imagens	1	2	2	1	2	2	2
Conteúdo externo	Boa qualidade das imagens	1	2	2	1	1	2	2
	Boa qualidade do texto	1	2	2	1	0	2	1
	Reconhecimento correto da biblioteca do utilizador	1	10	8	10	10	9	10
Personalização	Controlo das barras laterais	2	2	1	2	2	1	2
	Alteração das cores	1	1	1	1	1	1	1
Total	21 critérios	50	41	86%	84%	83%	94%	88%
				85%		**88%**		
				87%				

6.3 Conclusão

Apesar da escassez de resultados ao inquérito, foi possível obter boas conclusões através das sessões de interação supervisionada. Foram identificadas falhas de usabilidade independentes do Material Design e, as que puderam ser resolvidas atempadamente, foram resolvidas de forma a evitar que o resultado final seja influenciado.

Não é possível tirar conclusões sólidas com os resultados dos inquéritos, mas os inquiridos tiveram boas opiniões sobre este *design*. A organização e aspeto sugeridos pela documentação do Material Design foram bem recebidos, no entanto há quem precise de ver primeiro um sistema completamente baseado em Material Design para decidir se concorda ou não com essa aplicação.

As sessões de interação supervisionada foram bastante interessantes de se executar, principalmente devido à colaboração por pessoas com dificuldades e conhecimentos bem distintos. Utilizadores experientes tiveram dificuldades em voltar à interface completa, o que mostrou a necessidade imediata de encontrar uma solução diferente. Já na segunda iteração, uma das utilizadoras sem qualquer experiência conseguiu voltar à interface completa.

Na segunda iteração, as tarefas pedidas foram executadas corretamente e a velocidade com que certas tarefas puderam ser executadas mostraram uma grande eficiência da interface. A eficiência de uma interface indica quantos recursos (geralmente o tempo) são necessários para o utilizador executar uma dada tarefa. Assim conclui-se que as tarefas principais são feitas num curto espaço de tempo, facilitando a vida do utilizador. No entanto, a interface de vídeo parece precisar de algumas melhorias.

É importante relembrar que todas estas pessoas utilizaram a aplicação sem qualquer ajuda ou explicação dada previamente, nem a própria interface continha instruções, havendo mesmo utilizadores que nunca tinham utilizado leitores multimédia. Caso seja necessário, as versões mais recentes do Material Design sugerem métodos para dar algumas dicas ao utilizador com o objetivo de o dar a conhecer novas funcionalidades gradualmente ou quando necessário, estes métodos poderão ser implementados para auxiliar os utilizadores a voltar à interface, encontrar o ícone das definições, saber que a barra inferior está "escondida" no fundo durante a reprodução de vídeos, etc.

Capítulo 7

Conclusão e Trabalho Futuro

Finalizado o projeto, agora tiram-se conclusões. Os objetivos propostos foram alcançados, estudaram-se e ficaram-se a conhecer melhor algumas regras para o desenvolvimento de interfaces com boa usabilidade e o Material Design. Desenvolveu-se uma aplicação baseada no Material Design que faz uso da biblioteca do Papyros na qual se encontraram alguns problemas e se desenvolveram alguns componentes que foram discutidos com um elemento da equipa do Papyros e poderão ser integrados na biblioteca.

A divulgação do *software* não foi tão fácil como se esperava, *shells* como o KDE têm sites dedicados à divulgação de conteúdos de personalização e *software* mas o Papyros ainda é um projeto muito pouco conhecido. A existência de uma versão para Windows poderia ter ajudado imenso a divulgar o projeto no *subreddit* do Material Design. Como a aplicação chegou a poucos utilizadores, obtiveram-se poucas respostas ao questionário.

Felizmente, já estava planeada uma outra forma de avaliação da usabilidade. As sessões de interação supervisionada permitiram identificar falhas e compreender melhor como os utilizadores interagem com a aplicação. Algo importante nestes testes seria identificar situações em que o Material Design auxiliava ou complicava a interação com o utilizador. Conclui-se que a estrutura da aplicação, que segue o Material Design, fornecia redundância que permitiu diferentes utilizadores percorrer diferentes caminhos para chegar aonde queriam, havendo assim vários caminhos disponíveis caso o utilizador não saiba como encontrar o que procura. As animações e o uso mais habitual de imagens, também recomendados pelo Material Design, parecem ter ajudado os utilizadores a direcionar o olhar para áreas da aplicação onde acontecia algo relevante, facilitando a compreensão das ações efetuadas pelos próprios utilizadores. Apesar de o FAB principal já ser bem conhecido pela maioria das pessoas, o segundo FAB, visível na página de artista/programa de TV, ajudou a compreender o funcionamento do ícone de reprodução (*play*) para quem não estava à vontade com esse conceito. Os FAB's são mais outro conceito do Material Design. Em geral, mesmo para quem não está habituado a usar o rato, a dimensão dos botões parece ter sido adequada, concordando com a lei de Fitts.

Estes resultados não permitem assumir que uma *shell* completamente desenhada em Material Design seja uma evolução em termos de usabilidade mas pode-se concluir que este *design* trouxe melhorias para a interface da aplicação, tendo mesmo acabado por ensinar conceitos à medida que foi utilizado. Possivelmente, uma *shell* como o Papyros poderá aumentar a eficiência de utilização dos computadores para os utilizadores que não recorram a interações avançadas como por exemplo teclas de atalho uma vez que o Material Design foca-se em colocar de forma mais acessível possível as funcionalidades principais do sistema a interagir.

Num trabalho futuro, seria interessante já ter uma versão completa e estável do Papyros com o qual se fariam testes semelhantes mas mais pormenorizados. Uma forma de tornar os resultados deste trabalho mais relevantes poderia passar por testes semelhantes com alguns dos leitores apresentados inicialmente. Também se poderia efetuar trabalhos semelhantes para outros tipos de aplicação, poder-se-á concluir que o Material Design se adeque mais para uns tipos do que outros.

Numa nota mais pessoal, gostaria de sugerir que o ISEP desse alguma atenção extra ao bom *design* de interfaces com o utilizador. Os computadores são ferramentas essenciais em muitas profissões e tenho observado trabalhadores a "atrapalharem-se" ao utilizar esses computadores. Uns pareciam estar a seguir instruções 100% à risca, como se não soubessem o que estavam a fazer e mesmo assim falhavam, voltando ao início do processo. Outros pareciam dar voltas desnecessárias para fazer algo que se podia fazer instantaneamente. Também observei pessoas a escrever tudo em maiúsculas, como se o sistema não fosse encontrar o que procuravam independentemente desse detalhe. Algumas destas interfaces nem sequer faziam bom uso do ecrã, ocupando menos de um terço do mesmo, estando o resto preenchido com o branco da página *web* onde essa interface residia. A maioria destas interfaces quase que se assemelham a consultas SQL diretas à base de dados quando poderiam ser feitas imensas melhorias que não só facilitavam a vida desses trabalhadores, como aceleraria as filas de espera. Concluindo esta nota, acredito que incentivar os alunos do ISEP a tornar o seu *software* mais fácil de usar para todos será uma mais valia num país onde este assunto tem sido descuidado (assumindo que o problema não é assim tão mau fora de Portugal) e quem sofre com isso são os trabalhadores e quem está dependente desses trabalhos: todos nós.

Bibliografia

Akass, Clive (2001). *The men who really invented the GUI*. url: http://www.computeractive.co.uk/pcw/pc-help/1925325/the-invented-gui.

Atwood, Jeff (2006). *Fitts' Law and Infinite Width*. url: http://blog.codinghorror.com/fitts-law-and-infinite-width/ (acedido em 25/01/2016).

Barth, Adam (2013). *Blink: A rendering engine for the Chromium project*. url: https://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html%20http://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html.

Carroll, JM (2013). «Human computer interaction (HCI)». Em: *The Encyclopedia of Human-Computer Interaction*, pp. 1–28. url: http://www.interaction-design.org/encyclopedia/human%7B%5C_%7Dcomputer%7B%5C_%7Dinteraction%7B%5C_%7Dhci.html.

Equipa de Design da Google (2014). *Material Design*. url: https://www.google.com/design/spec/material-design/introduction.html%7B%5C#%7D (acedido em 20/01/2016).

– (2015). *Making Material Design*. url: https://www.youtube.com/watch?v=rrT6v5sOwJg.

Fitts, Paul M. (1954). «The information capacity of the human motor system in controlling the amplitude of movement.» Em: *Journal of Experimental Psychology* 47.6, pp. 381–391. issn: 0022-1015. doi: 10.1037/h0055392.

Frankel, Justin, Ben Sawyer e Dave Greely (2002). *Winamp Documentation - The Playlist and Playlist Editor*. url: https://web.archive.org/web/20020606035458/http://www.winamp.com/download/docs/index.jhtml?filenumber=120%7B%5C&%7Dlanguage=english%7B%5C&%7Dlayout=normal%7B%5C&%7Dprevlayout=normal (acedido em 01/06/2002).

Freed, Ned et al. (2015). *Media Types*. url: https://www.iana.org/assignments/media-types/media-types.xhtml%20http://www.iana.org/assignments/media-types/media-types.xhtml.

Hearn, Mike, Christian Hammond e William Jon McCann (2016). *Desktop Notifications Specification*. url: https://developer.gnome.org/notification-spec/ (acedido em 14/04/2016).

Høgsberg, Kristian (2012). *Wayland*. url: https://wayland.freedesktop.org/.

Ibraheem, Bello (2014). *Making Sense of NoSQL*. url: http://hebrayeem.blogspot.pt/2014/01/making-sense-of-nosql.html (acedido em 12/02/2016).

Levenshtein, Vladimir I. (1966). «Binary codes capable of correcting deletions, insertions, and reversals». English. Em: *Soviet Physics Doklady* 10.8, pp. 707–710. issn: 00385689. doi: citeulike-article-id:311174. arXiv: arXiv:1011.1669v3.

Lineback, Nathan (2016). *Graphical User Interface Timeline*. url: http://toastytech.com/guis/guitimeline.html (acedido em 19/01/2016).

Meyer, David (2011). *Nokia gives Qt open-source governance*. url: http://www.zdnet.com/article/nokia-gives-qt-open-source-governance/.

Microsoft (2012). *Design UWP apps*. url: https://dev.windows.com/en-us/design.

Nicola, Susana (2016). *Análise de Valor de Negócio*. Porto.

Nield, David (2015). *Here's How To Use Google Play Music From a Desktop App*. url: http://fieldguide.gizmodo.com/play-google-play-music-from-your-computer-desktop-1745221033.

Nielsen, Jakob (1993). *Usability Engineering*. Academic Press, Inc., p. 362. isbn: 0125184069. doi: 10.1145/1508044.1508050.

– (1994). «Enhancing the explanatory power of usability heuristics». Em: *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 152–158. issn: 0897916506. doi: 10.1145/191666.191729. url: http://portal.acm.org/citation.cfm?doid=191666.191729%7B%5C$%7D%5Cbackslash%7B%5C$%7Dnpapers3://publication/doi/10.1145/191666.191729%20http://portal.acm.org/citation.cfm?doid=191666.191729$%5Cbackslash$npapers3://publication/doi/10.1145/191666.191729.

– (2013). «User Expertise Stagnates at Low Levels». url: https://www.nngroup.com/articles/stagnating-expertise/.

Nielsen, Jakob e Thomas K. Landauer (1993). «A mathematical model of the finding of usability problems». Em: *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pp. 206–213. issn: 0-89791-575-5. doi: 10.1145/169059.169166. url: http://peres.rihmlab.org/Classes/PSYC6419seminar/p206-Five%20Users%20nielsen.pdf%20http://portal.acm.org/citation.cfm?doid=169059.169166.

Norman, Donald A (2013). *The Design of Everyday Things*. Vol. 16. 4, p. 272. isbn: 0465067107. doi: 10.1002/hfm.20127. url: http://ucdwiki.chuank.com/uploads/Main/UCDReading%7B%5C_%7Dwk5.pdf%20http://ucdwiki.chuank.com/uploads/Main/UCDReading%7B%5C%7B%7D%7B%5C_%7D%7B%5C%7D%7Dwk5.pdf.

QML (2016). url: https://en.wikipedia.org/wiki/QML.

Qt 5.7 Multimedia Backends (2016). url: http://wiki.qt.io/Qt%7B%5C_%7D5.7%7B%5C_%7DMultimedia%7B%5C_%7DBackends (acedido em 12/02/2016).

Qt Company Ltd. (2016). *QML 5.7*. url: http://doc.qt.io/qt-5/qtqml-index.html.

Spencer, Michael (2015). *About*. url: http://papyros.io/about/index.html.

Streicher, Martin (2010). *Packaging software with RPM, Part 1: Building and distributing packages*. url: http://www.ibm.com/developerworks/library/l-rpm1/.

Sweet, Michael (2007). *CUPS Purchased by Apple Inc.* url: https://www.cups.org/blog/2007-07-11-cups-purchased-by-apple-inc..html.

The Linux Foundation (2016). *Linux Kernel Development: How Fast It is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It*. Rel. téc. url: https://www.linux.com/publications/linux-kernel-development-how-fast-it-going-who-doing-it-what-they-are-doing-and-who-5.

Tognazzini, Bruce "Tog" (1999). *A Quiz Designed to Give You Fitts*. url: http://www.asktog.com/columns/022DesignedToGiveFitts.html (acedido em 25/01/2016).

Wickens, Christopher D., Sallie E Gordon e Yili Liu (2004). *An introduction to human factors engineering*. 2nd. Pearson, p. 608. isbn: 0131837362.

Apêndice A

Questionários de Usabilidade

Nas páginas seguintes seguem-se as duas versões do questionário, num formato impresso exportado do Google Forms.

A.1 Versão Portuguesa

Papyros Media Player

Olá! Este inquérito ajudará a compreender se as regras de design e comportamento utilizadas no Papyros Media Player trazem benefícios para utilizadores de computadores.
Não és obrigado(a) a responder a todas as perguntas.

*Obrigatório

1. **Idade:**
   Marcar apenas uma oval.

   - ◯ Menos de 20
   - ◯ Entre os 20 e os 40
   - ◯ Entre os 40 e os 60
   - ◯ 60 ou mais

Perfil

Primeiro temos de saber quanto sabes sobre tecnologia e o Material Design.

2. **Com que frequência usas computadores?**
   não confundir com telemóveis, tablets, etc.
   Marcar apenas uma oval.

   - ◯ Nunca, só usei agora para testar a aplicação *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ Já usei algumas vezes na minha vida *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ Uma vez por mês *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ Uma vez por semana
   - ◯ Algumas vezes por semana
   - ◯ Todos os dias ou quase

3. **Quais destes dispositivos tens?**
   Marcar tudo o que for aplicável.

   - ☐ Telemóvel sem ecrã tátil
   - ☐ Telemóvel com ecrã tátil
   - ☐ Tablet
   - ☐ Relógio inteligente (smartwatch)
   - ☐ Consola de jogos (Playstation, Xbox, Wii, etc.)

4. **Já ouviste falar em "Material Design"?**
   Marcar apenas uma oval.

   - ◯ Não.
   - ◯ Já ouvi falar mas não faço ideia o que seja.
   - ◯ É qualquer coisa da Google ou do Android mas não sei o quê.
   - ◯ É qualquer coisa da Apple ou do iOS mas não sei o quê.
   - ◯ Sim.
   - ◯ Sim e tenho noção das suas linhas de orientação.

5. **Que leitores de música/vídeo já usaste?**
   Marcar tudo o que for aplicável.

   - ☐ Apple iTunes
   - ☐ Google Play Music
   - ☐ Microsoft Groove
   - ☐ SMPlayer
   - ☐ Spotify
   - ☐ Microsoft Windows Media Player
   - ☐ VLC
   - ☐ Amarok
   - ☐ Outra: .....

Perfil

Primeiro temos de saber quanto sabes sobre tecnologia e o Material Design.

6. **Que sistema usas em computadores regularmente?**
   não confundir com telemóveis e tablets, etc.
   Marcar tudo o que for aplicável.

   - ☐ Apple Mac OS
   - ☐ Linux (Ubuntu, openSUSE, Arch, Mint, etc.)
   - ☐ Microsoft Windows (entre o XP e o 7)
   - ☐ Microsoft Windows (8 e mais recentes)
   - ☐ Não sei
   - ☐ Outra:

Perfil

Primeiro temos de saber quanto sabes sobre tecnologia e o Material Design.

7. **Que sistemas usas em telemóveis e/ou tablets regularmente?**
Marcar tudo o que for aplicável.

- [ ] Android 1 ou 2 (Androids muito antigos)
- [ ] Android 3 ou 4 (também conhecidos como Honeycomb, Ice Cream Sandwich, Jelly Bean e KitKat)
- [ ] Android 5 ou mais recentes (também conhecidos como Lollipop, Marshmallow e Nougat)
- [ ] iOS (iPhone, iPad, iPod Touch)
- [ ] Windows
- [ ] Symbian (maioritariamente Nokias antigos)
- [ ] Blackberry
- [ ] WebOS
- [ ] Não uso um telemóvel com ecrã tátil ou tablet
- [ ] Não sei
- [ ] Outra: .....

Opinião sobre o Papyros Media Player

Agora que sabemos se estavas à vontade com a aplicação ou não, podemos fazer perguntas sobre ela.

8. **Foi um choque quando abri a aplicação pela primeira vez.** *
Marcar apenas uma oval.

- ◯ Sim, nem percebi o que era.
- ◯ Sim, percebi o que a aplicação fazia mas não percebi como.
- ◯ Sim, mas no bom sentido.
- ◯ Foi diferente do habitual mas não me chocou.
- ◯ Não, já estava à espera.

9. **Depois do primeiro contato, decidi...**
Marcar apenas uma oval.

- ◯ esperar que a aplicação encontrasse toda a minha música
- ◯ pôr uma música/vídeo manualmente
- ◯ adicionar artistas e/ou programas de TV manualmente
- ◯ percorrer a lista de artistas/programas de TV porque a aplicação já tinha encontrado alguma coisa
- ◯ ir às definições
- ◯ Outra: .....

10. **...e isso que fiz foi...**
Marcar apenas uma oval.

- ◯ fácil e interessante.
- ◯ fácil.
- ◯ confuso inicialmente mas depois percebi bem como funcionava.
- ◯ difícil.
- ◯ impossível.

Opinião sobre o Papyros Media Player

E agora perguntas um pouco mais específicas.

11. **A navegação foi...** *
(exemplo: ir à lista de artistas, entrar na página do artista, entrar na lista completa de músicas do artista)
Marcar apenas uma oval.

- ◯ muito fácil, era só dar uns cliques/toques e já está!
- ◯ fácil.
- ◯ fácil mas não sabia voltar atrás.
- ◯ inicialmente a navegação é confusa mas depois torna-se fácil.
- ◯ confusa.
- ◯ nem consegui navegar.

12. **As animações...** *
Marcar apenas uma oval por linha.

	Não	Às vezes	Sim
faziam sentido	◯	◯	◯
acabavam por explicar melhor e/ou confirmar ações	◯	◯	◯
eram interessantes	◯	◯	◯
eram suficientemente rápidas para não atrasar a utilização	◯	◯	◯

13. **Os botões e textos...** *
Marcar apenas uma oval.

- ◯ podiam ser mais pequenos.
- ◯ eram grandes demais mas nada de grave.
- ◯ tinham as dimensões adequadas.
- ◯ eram pequenos demais.

14. **A aplicação...**
Marcar apenas uma oval.

- ◯ bloqueou várias vezes, tive de a fechar à força e abrir de novo.
- ◯ era muito lenta.
- ◯ podia ser mais rápida.
- ◯ tudo o que fazia acontecia instantaneamente ou quase.

15. **O aspeto da aplicação...** *
Marcar apenas uma oval por linha.

	Concordo a 100%	Concordo um pouco	Indeciso	Discordo um pouco	Discordo a 100%
é inovador	◯	◯	◯	◯	◯
é inovador nesta área	◯	◯	◯	◯	◯
bonito (esquecendo a subjetividade da cor)	◯	◯	◯	◯	◯
tem uma estrutura bem organizada	◯	◯	◯	◯	◯

16. **A funcionalidade de pesquisa**
Marcar apenas uma oval.

- ◯ Foi muito rápida e encontrei o que queria
- ◯ Foi muito rápida mas não foi fácil encontrar o que queria
- ◯ Foi lenta
- ◯ Não encontrei o que queria

17. **A linguagem da aplicação... ***
Marcar apenas uma oval por linha.

	Não	Mais ou menos	Sim	Não sei
foi fácil de entender.	◯	◯	◯	◯
foi natural (ex: "tocar" em vez de "reproduzir").	◯	◯	◯	◯
prefiro uma tradução mais "tradicional".	◯	◯	◯	◯

Opinião sobre o Papyros Media Player

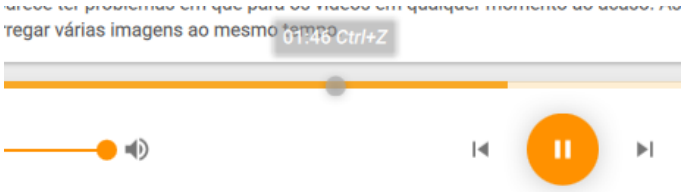
Há algumas coisas que podem distinguir esta aplicação dos outros leitores de música/vídeo, vamos agora perceber se as diferenças são benéficas ou não.

18. **Quando vejo um vídeo...**
Se ainda não o fizeste, experimenta :)
Marcar tudo o que for aplicável.

- ☐ gostei do facto que o vídeo ocupa toda a janela.
- ☐ foi fácil voltar às outras partes da aplicação.
- ☐ prefiro ter o vídeo sempre num espaço mais pequeno para poder ver sempre o resto da aplicação.
- ☐ é fácil pôr em ecrã inteiro.
- ☐ dá jeito e/ou é prático ter a lista (playlist) à frente.
- ☐ Outra: .....

19. **Quando vejo um vídeo e abano o rato e/ou toco no ecrã...**
Marcar apenas uma oval.

- ◯ gosto do fato que posso ver quanto tempo falta/já passou sem aparecer a barra toda.
- ◯ prefiro que apareça sempre a barra toda.
- ◯ não consigo ver a barra do fundo toda.



20. **Quando avanço ou recuo numa música/vídeo, aparece uma opção para voltar a pôr onde estava e...**
Marcar apenas uma oval.

- ◯ deu jeito.
- ◯ pode vir a dar jeito se avançar/recuar acidentalmente.
- ◯ é desnecessário
- ◯ qual opção? Não vi nada...

21. **Quando carreguei no botão de ir para a música/vídeo anterior, o botão...**
Marcar apenas uma oval.

- ◯ fez exatamente isso.
- ◯ estava à espera que pusesse a música/vídeo do início.
- ◯ Outra:

22. **O botão de repetir a música/vídeo atual...**
Marcar tudo o que for aplicável.

- ☐ é melhor que usar o botão de ir para a música/vídeo anterior para fazer isso.
- ☐ devia aparecer mais cedo.
- ☐ devia estar sempre visível.
- ☐ prefiro carregar no botão de ir para a música/vídeo anterior para fazer isso.
- ☐ qual botão? Não vi nada...

23. **Relativamente às opções "repetir uma vez depois" e "tocar depois desta":**
Estas opções aparecem ao clicar com o botão direito nos itens da lista (playlist).
Marcar apenas uma oval.

- ◯ Dão jeito.
- ◯ Talvez venham a dar jeito.
- ◯ São desnecessárias.

24. **A aplicação compreende que nem todas as músicas são cantadas por um único artista, com isso em mente:**
Marcar tudo o que for aplicável.

- ☐ O grupo "Participou em" que aparece nos artistas ajuda-me a encontrar alguma música que não saiba ao certo de quem é.
- ☐ O facto que posso aceder a informações de todos os artistas carregando com o botão direito na lista (playlist) é interessante.
- ☐ Gostei da forma como essas músicas aparecem na lista (playlist).
- ☐ Essa funcionalidade é irrelevante para mim.
- ☐ Não tenho músicas que abranjam esse caso para poder testar.

Material Design

O aspeto e comportamentos da aplicação são baseados no Material Design, um conjunto de regras e linhas de orientação desenvolvido pela Google com o objetivo de ser utilizado em diferentes dispositivos e sistemas.

25. **E se o aspeto, organização e comportamentos da aplicação fossem aplicados a todo o sistema do computador?** *

Marcar tudo o que for aplicável.

- [ ] O meu computador seria mais fácil de usar.
- [ ] O meu computador teria um aspeto mais bonito.
- [ ] Teria de ver primeiro.
- [ ] Parece só fazer sentido em algumas aplicações e não em todo o sistema.
- [ ] Já vi isto no meu telemóvel/tablet.
- [ ] Prefiro o meu computador como está porque não gosto de muitas mudanças.
- [ ] Prefiro o meu computador como está porque não gosto do Material Design.

26. **Se tiveres algum comentário que queiras fazer, podes escrever em baixo:**

.....

.....

.....

.....

A.2 Versão Inglesa

Papyros Media Player

Hi! This survey will help us understand whether the design and behavioral rules used in Papyros Media Player are beneficial for desktop and laptop users.
You don't have to answer every question.

*Obrigatório

1. **Age:**
   Marcar apenas uma oval.
   - ◯ Under 20
   - ◯ Between 20 and 40
   - ◯ Between 40 and 60
   - ◯ 60 or over

Profile

First we need to know how much you know about technology and Material Design.

2. **How often do you use desktops and/or laptops?**
   Marcar apenas uma oval.
   - ◯ Never, I just used it now to test the app *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ I've used it a few times before *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ Once a month *Após a última pergunta desta secção, passe para a pergunta 7.*
   - ◯ Once a week
   - ◯ A few times a week
   - ◯ Daily or almost

3. **Which of these devices do you have?**
   Marcar tudo o que for aplicável.
   - ☐ Phone without touchscreen
   - ☐ Phone with touchscreen
   - ☐ Tablet
   - ☐ Smartwatch
   - ☐ Game console (Playstation, Xbox, Wii, etc.)

4. **Have you ever heard of "Material Design"?**
   Marcar apenas uma oval.
   - ◯ No.
   - ◯ Yes but I have no idea what it is.
   - ◯ It's something from Google or Android but I don't know what.
   - ◯ It's something from Apple or iOS but I don't know what.
   - ◯ Yes.
   - ◯ Yes and I know its guidelines.

5. **What music/video players have you used before?**
   Marcar tudo o que for aplicável.
   - ☐ Apple iTunes
   - ☐ Google Play Music
   - ☐ Microsoft Groove
   - ☐ SMPlayer
   - ☐ Spotify
   - ☐ Microsoft Windows Media Player
   - ☐ VLC
   - ☐ Amarok
   - ☐ Outra: .....

Profile

First we need to know how much you know about technology and Material Design.

6. **What systems do you regularly use on desktops/laptops?**
   Marcar tudo o que for aplicável.
   - ☐ Apple Mac OS
   - ☐ Linux (Ubuntu, openSUSE, Arch, Mint, etc.)
   - ☐ Microsoft Windows (between XP and 7)
   - ☐ Microsoft Windows (8 and up)
   - ☐ I don't know
   - ☐ Outra: .....

Profile

First we need to know how much you know about technology and Material Design.

7. **What systems do you regularly use on phones and/or tablets?**
Marcar tudo o que for aplicável.

- [ ] Android 1 or 2 (very old versions of Android)
- [ ] Android 3 or 4 (Honeycomb, Ice Cream Sandwich, Jelly Bean and KitKat)
- [ ] Android 5 or up (Lollipop, Marshmallow and Nougat)
- [ ] iOS (iPhone, iPad, iPod Touch)
- [ ] Windows
- [ ] Symbian (mostly old Nokias)
- [ ] Blackberry
- [ ] WebOS
- [ ] I don't use a phone with touchscreen or tablet
- [ ] I don't know
- [ ] Outra: .....

Opinion about Papyros Media Player

Now that we know whether you were comfortable with the app or not, we can ask some questions about it.

8. **Were you shocked when you opened the app for the first time?** *
Marcar apenas uma oval.

- ◯ Yes, I didn't even understand what it was.
- ◯ Yes, I understood what the app is for but not how it works.
- ◯ Yes but in a good way.
- ◯ It was different from what I'm used to but it didn't shock me.
- ◯ No, I was already expecting this.

9. **After the app opened for the first time, I decided to...**
Marcar apenas uma oval.

- ◯ wait for the app to find all my music
- ◯ play some music or a video manually
- ◯ add artists and/or TV shows manually
- ◯ check the list of artists/TV shows because the app had already found something
- ◯ go to settings
- ◯ Outra: .....

10. **...and it was...**
Marcar apenas uma oval.

- ◯ easy and interesting.
- ◯ easy.
- ◯ initially confusing but then I understood well how it worked.
- ◯ complicated.
- ◯ impossible.

Opinion about Papyros Media Player

And now a little more specific questions.

11. **Navigating the app was...** *
(example: go to the artists list, open an artist's page, open the full list of songs of that artist)
Marcar apenas uma oval.

- ◯ very easy, just a few clicks/taps and that's it!
- ◯ easy.
- ◯ easy but I didn't know how to go back.
- ◯ the navigation was confusing initially but then it gets easy.
- ◯ confusing.
- ◯ I couldn't even navigate.

12. **The animations...** *
Marcar apenas uma oval por linha.

	No	Sometimes	Yes
make sense	◯	◯	◯
end up explaining better/confirming actions	◯	◯	◯
were interesting	◯	◯	◯
they were fast enough so I didn't have to wait for them to finish	◯	◯	◯

13. **Text and buttons...** *
Marcar apenas uma oval.

- ◯ could be smaller.
- ◯ were too big but that's ok.
- ◯ had the right dimensions.
- ◯ were too small.

14. **The app...**
Marcar apenas uma oval.

- ◯ froze many times, I had to force close it and open it again.
- ◯ was very slow.
- ◯ could be faster.
- ◯ was fast, everything I did was fast, fluid, or close to it.

15. **The appearance of the app...** *
Marcar apenas uma oval por linha.

	Fully agree	Agree a little	Undecided	Disagree a little	Fully disagree
is innovative	◯	◯	◯	◯	◯
is innovative in this field	◯	◯	◯	◯	◯
pretty (not including your color preference)	◯	◯	◯	◯	◯
its structure was well organized	◯	◯	◯	◯	◯

16. **The search feature**
Marcar apenas uma oval.

- ◯ Was very fast and I found what I wanted
- ◯ Was very fast but it wasn't easy to find what I wanted
- ◯ Was slow
- ◯ I didn't find what I wanted

Opinion about Papyros Media Player

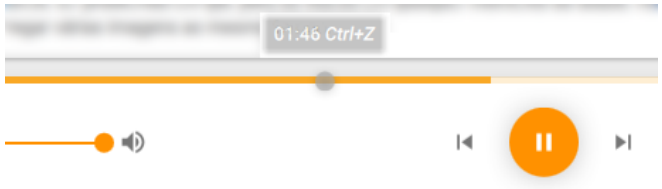
There are some things that may distinguish the app from the other music/video players, now let's understand whether those differences are beneficial or not.

17. **When I watch a video...**
If you haven't done so, try it now :)
Marcar tudo o que for aplicável.

- ☐ I liked that the video takes the whole window.
- ☐ it was easy to go back to the other parts of the app.
- ☐ I prefer to always have the video in a smaller space so I can see the rest of the app.
- ☐ it's easy to go full screen.
- ☐ it's handy and/or it's easy to have the playlist in front.
- ☐ Outra: .....

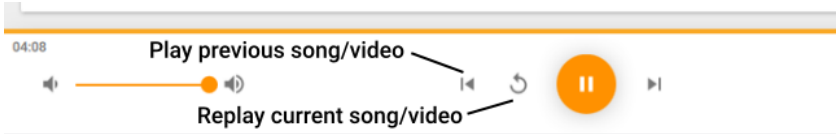
18. **When I watch a video and move the mouse and/or touch the screen...**
Marcar apenas uma oval.

- ◯ I like that I can see the movie progress without showing the entire bar.
- ◯ I prefer to see the entire bar.
- ◯ I don't know how to reveal the entire lower bar.



19. **When I jump forward or backward in a song/video, there is an option to go back to the previous time (see picture above) and...**
Marcar apenas uma oval.

- ◯ that was convenient.
- ◯ it may be convenient if I jump forward/backward accidentally.
- ◯ it's unnecessary.
- ◯ what option? I didn't see anything...



20. **When I pressed the button to play the previous song/video, the button...**
Marcar apenas uma oval.

- ◯ did just that.
- ◯ I was expecting it to replay the current song/video.
- ◯ Outra: .....

21. **The button to replay the current song/video...**
Marcar tudo o que for aplicável.

- ☐ is better than using the button that plays the previous song/video to do that.
- ☐ should show up earlier.
- ☐ should be visible all the time.
- ☐ I prefer using the button that plays the previous song/video to do that.
- ☐ what button? I didn't see anything...

22. **The options "repeat once after" and "play after this one"**
These options show up when right clicking the items in the playlist.
Marcar apenas uma oval.

- ◯ are convenient.
- ◯ may be convenient one day.
- ◯ are unnecessary.

23. **The app understands that not all songs are by a single artist, with that in mind:**
Marcar tudo o que for aplicável.

- ☐ The group "Participated in" (visible in the artist page) helps me find some song that I don't know exactly who sings it.
- ☐ I like that I can access info about any other artists of the song by right clicking it in the playlist.
- ☐ I like how those songs show up in the playlist.
- ☐ That's irrelevant to me.
- ☐ I don't have any songs that apply to this case so I can test it.

Material Design

The appearance and behaviors of the app are based on Material Design, a set of rules and guidelines developed by Google with the goal of being applied to any device and system.

24. **What if the appearance, organization and behavior of the app were applied to the entire system of your computer?** *
Marcar tudo o que for aplicável.

- ☐ My computer would be easier to use.
- ☐ My computer would look better.
- ☐ I'd have to see it first.
- ☐ It seems to make sense in some apps but not in the whole system.
- ☐ I've seen this in my phone/tablet.
- ☐ I prefer my computer as it is because I don't like changes.
- ☐ I prefer my computer as it is because I don't like Material Design.

25. **If you have any comments, you may leave them below:**

.....

.....

.....

.....
