Faculdade de Engenharia da Universidade do Porto

Interação remota através de manipulação direta para aplicações com multi-utilizadores em ecrãs públicos

Maria João Barreira

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Doutor Maria Teresa Galvão Dias

28 de julho de 2014

Interação remota através de manipulação direta para aplicações com multi-utilizadores em ecrãs públicos

Maria João Barreira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Prof. Doutor Jorge Manuel Gomes Barbosa

Arguente: Prof. Doutor José Benjamim Ribeiro da Fonseca

Vogal: Prof. Doutor Maria Teresa Galvão Dias

---

28 de julho de 2014

Resumo

Os ecrãs públicos digitais têm cada vez mais uma forte presença no nosso dia-a-dia, contudo poucos são os que permitem uma manipulação direta por parte do utilizador. Atualmente, a sua função é maioritariamente publicitar determinado serviço ou produto, e a maior parte das vezes até passam despercebidos aos transeuntes.

É importante reconhecer o valor de quem interage com o ecrã e permitir um desenvolvimento mais fácil de aplicações ricas, que propiciem uma interação eficaz por parte de um ou mais utilizadores.

Aliando o crescente uso de dispositivos móveis, como *smartphones* e *tablets*, aos recentes avanços tecnológicos existentes nos ecrãs públicos digitais, este projeto apresentava como principais objetivos a criação de uma *framework* que facilite o desenvolvimento de aplicações de cariz público e a implementação de alguns exemplos destas aplicações, que permitam a utilização da *framework* desenvolvida.

A *framework* desenvolvida encontra-se orientada a objetos, sendo composta por 4 classes distintas. As subclasses representam os tipos de controlos que o programador terá disponíveis e poderá implementar na sua aplicação. Neste caso, foram desenvolvidos 3 diferentes tipos de widgets, que permitirão ao utilizador final interagir com a aplicação.

Como exemplo de aplicação foi implementado o clássico jogo da *Snake*. Uma vez que não havia necessidade de desenvolver o jogo de raíz, foi necessário pesquisar por um exemplo em *html* e *javascript* ao qual se pudessem adaptar os controlos da *framework*. Apesar de existirem um grande número de opções, foi escolhido o jogo referido, uma vez que é um jogo bastante conhecido, que não necessita de grandes explicações e num modo multi-jogador torna-se competitivo.

Não só no final, mas também ao longo do desenvolvimento, ocorreu a realização de alguns testes. Os testes realizados ao longo do desenvolvimento permitiram saber se a funcionalidade implementada fazia realmente o que é desejado, obtendo de forma rápida *feedback* do trabalho acabado de realizar, prevenindo erros futuros semelhantes e trabalho desnecessário.

No final foi pedido a 3 estudantes do MIEIC que integrassem a *framework* desenvolvida com soluções *open source* recorrendo à mesma para a definição dos respetivos controlos.

Abstract

The presence of digital public displays in urban landscapes has increased, however only few of them allow a direct-manipulation to passersby. Nowadays, the main feature is to advertise a service or a product and often people ignore them.

It is important to be centered in the final user and allow to developers an easier development of applications providing an efficient interaction for one or more users.

Combining the growing use of mobile devices, such as smartphones and tablets, with the latest technological advances in existing public digital displays, this project had as main goals the creation of a framework that eases the development of public applications and the implementation of some examples allowing the use of the developed framework.

The developed object-oriented framework heavily relies in inheritance consisting of four distinct classes, one of them being an interface, that is the abstract representation of a control. The subclasses represent the types of controls that the developer will have available and can implement in an application. In this case, three different types of widgets, that allow the final user to interact with the application, have been developed.

As an example of application, it was implemented the classic game of Snake. Since there was no need to create the game, it was necessary to search for an example in HTML and JavaScript which it could apply the controls of framework. Although there were a large number of options, this game was chosen because it is well-known and, henceforth it does not need much explanation. Besides, in multiplayer mode the game becomes competitive.

Not only at the end, but also throughout development, some tests occurred. Tests during the development allowed whether the implemented functionality was the desired or not, quickly getting feedback from the most recent changes on the work, preventing similar future errors and unnecessary work.

At the end, three students from MIEIC were asked to incorporate the framework on open-source solutions using it for the respective controls implementation.

Agradecimentos

Em primeiro lugar, quero agradecer o suporte dado pela minha orientadora, a professora Teresa Galvão, pela disponibilidade e pensamento crítico ao longo de todo o projeto.

De seguida, agradeço também ao proponente desta dissertação, Dr. Jorge Cardoso, que sempre se mostrou disponível para esclarecer qualquer dúvida e acompanhar o desenvolvimento.

Um agradecimento especial, ao Bruno Maia e Renato Rodrigues, pela paciência que demonstraram, e a todos os amigos com quem tive a oportunidade de partilhar este percurso.

Por fim, agradeço aos meus pais e irmã pelo apoio incondicional.

Maria João Barreira

"If you can't explain it simply, you don't understand it well enough."

Albert Einstein

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Abreviaturas e Símbolos

API	*Application programming interface*
CSS	*Cascading Style Sheets*
DCU	Desenvolvimento Centrado no Utilizador
GPRS	*General Packet Radio Service*
HTML	*HyperText Markup Language*
IHP	Interação Humano-Computador
PDA	*Personal Digital Assistant*
SDG	*Single Display Groupware*
SMS	*Short Message Service*
TCP	*Transmission Control Protocol*

Capítulo 1

Introdução

Esta dissertação tem como tema o estudo de interação remota, baseada em manipulação direta para aplicações em ecrãs públicos, com multi-utilizadores, e foi proposta pelo CITAR[1] da Universidade Católica.

1.1 Contexto/Enquadramento

Na atualidade, é cada vez maior o número de ecrãs públicos existentes em diversos cenários urbanos, sejam eles paragens de transportes públicos, salas de espera ou outras zonas mais movimentadas. No entanto, a maioria apenas é utilizada como meio de divulgação de determinado produto ou serviço, não permitindo ao transeunte interagir com o mesmo.

Em [WEB+], Wilmsmann chega à conclusão que a população já está habituada à sua presença e falta de utilidade, que facilmente os ignoram, passando muitas vezes despercebidos.

Este cenário pode ser alterado, pois os recentes avanços da tecnologia permitem proporcionar aos utilizadores uma interação com estes ecrãs através da manipulação direta dos mesmos, usando para isso o seu dispositivo móvel.

Apesar de já existir algum desenvolvimento nesta área, alterar o estado atual dos ecrãs em algo completamente novo requer algum investimento tecnológico de inovação.

Uma pesquisa mais cuidada revela que já começa a ser comum, em áreas mais movimentadas, como estações de comboios ou praças públicas, a possibilidade de as pessoas interagirem com diversos ecrãs. Existem diferentes maneiras pelas quais esta interação é possível, como por exemplo através do toque, leitura de *QR code*, introdução de um código ou usando um *kinnect* inserido no próprio ecrã.

[1] Centro de Investigação em Ciência e Tecnologia das Artes, http://artes.ucp.pt/citar/

Nigel Davies [DLJS12] compara a situação dos ecrãs públicos com os *smartphones*, afirmando que a inovação ocorre quando são criados sistemas livres que encorajam a inovação e incentivam a um maior desenvolvimento de novos produtos.

1.2 Projeto e Objetivos

O tema proposto tem como objetivo principal desenvolver e validar uma arquitetura que permita uma interação baseada no paradigma da manipulação direta, ou seja, uma interação em que o utilizador se apercebe das alterações no exato momento em que se efetua.

No final, pretende-se obter uma *framework* que facilite a criação de aplicações para ecrãs públicos baseadas no paradigma acima descrito, bem como alguns exemplos de aplicações que sejam construídas com base na *Application programming interface (API)* desenvolvida.

1.3 Desafios

Como já foi referido anteriormente, cada vez mais, em locais públicos existem diversos ecrãs, contudo a sua maioria apenas serve para publicitar determinado produto ou serviço.

Esta é uma área nova, em constante desenvolvimento, em que há a possibilidade de estabelecer alguns conceitos de referência para o futuro.

Os ecrãs públicos estão situados em zonas estratégicas, encontrando-se maioritariamente localizados em zonas onde existe uma maior concentração de pessoas, sendo também importante estudar a possibilidade de uma interação por mais do que um utilizador.

O presente projeto apresenta um lado desafiante que leva à procura de soluções para o desenvolvimento de aplicações *web*, que suportem uma interação por múltiplos utilizadores. Estes deverão usar o seu próprio dispositivo móvel para poderem usufruir da aplicação. Será também relevante pensar nos tipos de controlo que deverão estar disponíveis.

Para além dos desafios acima mencionados, é também importante perceber de que modo a infra-estrutura da rede pode influenciar o correto funcionamento das aplicações, bem como se no mesmo ecrã poderá existir mais do que uma aplicação, ou ainda se para cada aplicação existe a necessidade de haver um servidor diferente.

Este trabalho irá também permitir o desenvolvimento de aplicações interativas mais ricas, abrindo o leque de possibilidades de configuração e interação com este tipo de aplicações.

1.4 Estrutura da Dissertação

Esta dissertação apresenta para além da introdução mais seis capítulos.

No capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados.

No capítulo 3, é apresentada a metodologia a usar no desenvolvimento do projeto. Segue-se o capítulo 4, onde se descreve pormenorizadamente a solução implementada e as tecnologias usadas para alcançar os objetivos propostos.

No capítulo 5 são descritos os testes realizados e as conclusões tiradas a partir dos mesmos, sendo realizada uma discussão e análise crítica do produto desenvolvido no capítulo 6.

Por último, no capítulo 7, é elaborada uma análise conclusiva e são referidos possíveis melhoramentos futuros.

Capítulo 2

Revisão Bibliográfica

A interação com ecrãs públicos não é um tema novo, existindo já diversas abordagens que podem ser usadas nas aplicações desenvolvidas para os mesmos.

Neste segundo capítulo são referidas soluções tecnológicas usadas nos diversos projetos que poderão ser utilizadas no corrente tema, bem como requisitos a considerar neste tipo de aplicações.

São também mencionados alguns trabalhos relacionados com o tema de forma a comparar o que existe desenvolvido na área referida com os problemas em aberto que ainda podem ser solucionados.

2.1 Ecrãs Públicos Interativos

A interação com os grandes ecrãs pode ser diferenciada em três diferentes domínios: pessoal, semi-público e público [BRSB04]. Os ecrãs pessoais permitem a um único utilizador visualizar e processar uma grande quantidade de informação ao mesmo tempo. Os ecrãs semi-públicos encontram-se situados num ambiente controlado, como por exemplo, salas de reuniões onde apenas um número limitado de pessoas interage, normalmente usando apenas um ecrã com aplicações de grupo. Ecrãs públicos encontram-se tipicamente localizados em áreas "abertas", usualmente em ambientes com grande movimento onde as pessoas passam e têm de esperar algum tempo, como estações de comboio, aeroportos ou parques.

Em [BRSB04] são identificadas algumas considerações específicas para técnicas de interação com ecrãs públicos, referindo:

- a vontade espontânea de o utilizador interagir com o ecrã;
- a facilidade de o utilizador "transportar" as ferramentas necessárias para que a interação seja possível, pois existem mecanismos que permitem ao utilizador interagir

sem qualquer dispositivo adicional, e outros que exigem que o utilizador possua determinado equipamento;

- as considerações de limpeza e saúde associadas à técnica de interação, como a condição física do ecrã;
- o número de mãos que serão necessárias para a operação, sendo um aspeto relevante uma vez que, a maior parte das vezes, o transeunte precisa das suas mãos para carregar os seus bens;
- a possibilidade de a técnica de interação suportar mais do que um utilizador ao mesmo tempo;
- o grau de segurança e privacidade esperados quando se interage com um ecrã em público;
- a necessidade de um serviço regular de modo a manter o sistema operacional e uma aparência que seja atrativa ao seu uso.

Os ecrãs públicos tendem a possuir monitores maiores, o que pode resultar em alguns benefícios, dando maior liberdade aos programadores, uma vez que a capacidade gráfica é superior à do ecrã do telemóvel. Além disso, tornam possível um maior número de movimentos para os participantes e criam uma atmosfera mais rica socialmente permitindo alguma interação social em diferentes cenários urbanos [VCBE08].

2.1.1 Requisitos para um interação pública

Tal como já foi referido anteriormente, no final do desenvolvimento deste projeto é desejado que seja mais fácil para os programadores, desenvolver aplicações que permitam a interação com ecrãs públicos. Segundo [CJ12], alguns dos requisitos necessários, são comuns a outros sistemas interativos, mas outros são específicos de ecrãs públicos, fazendo referência a:

Múltiplos mecanismos de interação - uma interação em ecrãs públicos pode usar múltiplos mecanismos de entrada, como *Short Message Service (SMS)*, *email*, *Bluetooth*, *Twitter*, gestos ou movimento do corpo. Nem todos os mecanismos fornecem os mesmos controlos de alto-nível, contudo os programadores devem ser capazes de especificar as necessidades da interação. Uma boa abstração deve ser aplicável a múltiplos mecanismos de entrada.

Interação Partilhada - uma interação partilhada consiste em dois níveis diferentes. O primeiro relaciona o comportamento do utilizador com os restantes utilizadores, adaptando o seu comportamento consoante o que os restantes estão a fazer. O segundo torna o sistema do ecrã capaz de aceitar não só a interação corrente, mas também conciliar as interações de resposta.

Interação Assíncrona - Quando se trata de um sistema de ecrãs públicos é impossível para os utilizadores, de um modo geral, controlar as aplicações. Estas aplicações devem, em primeiro lugar, estar disponíveis independentemente de o ecrã estar ou não a mostrar algum do seu conteúdo e, em segundo, o seu ecrã não ser o único ponto de interação com o mesmo. Um boa abstração deve suportar este ambiente de interação assíncrona e permitir a interação a qualquer momento.

Fácil Reconhecimento - Deve ser claro e percetível para os transeuntes, o reconhecimento da existência de funcionalidades interativas e as suas propriedades. Este requisito, apesar de ser comum a outros sistemas interativos, corresponde ao princípio básico de visibilidade de *design* de interface ajudando na avaliação do sistema. É especialmente importante em ecrãs públicos, porque quando as pessoas se aproximam de um ecrã não fazem ideia se o mesmo é interativo ou não.

Múltiplas Funcionalidades Específicas para Ecrãs Públicos - Uma boa abstração deve permitir aplicações com diferentes funcionalidades. Aplicações para ecrãs públicos precisam de um conjunto de controlos, a partir dos quais os programadores poderão escolher, que devem ser apropriados a uma interação deste tipo. Programadores devem ser capazes de especificar o número de funcionalidades que a aplicação precisa e os utilizadores ter a possibilidade de aceder às mesmas individualmente.

No âmbito das interações públicas é importante salientar o conceito de privacidade. O utilizador deve ter controlo sobre a sua informação e poder decidir que informação e quando esta deve ser mostrada em determinado ecrã [DLJS12]. Como prova de conceito, Nigel Davies et al. [DLJS12] desenvolveram e implementaram *Tacita*, representado na figura 2.1, um sistema de personalização *privacy-aware*, que suporta interações anónimas com ecrãs em locais abertos. Com *Tacita* em vez de ter de confiar num número ilimitado de ecrãs, participantes confiam em fornecedores de aplicações, como *Google* e *Facebook*.

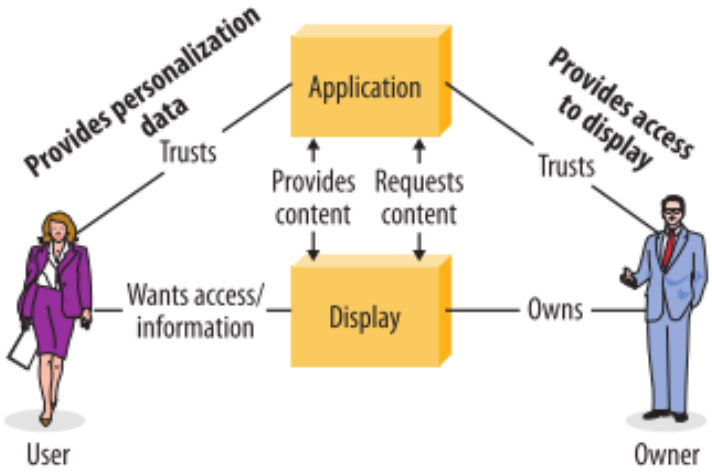


Figura 2.1: Sistema *Tacita* [DLJS12]

2.1.2 Tipos de ligação

Sempre que queremos interagir com determinado ecrã público, e o mesmo não possui um reconhecimento de movimento ou não é sensível ao toque, necessitamos de um dispositivo. Esse dispositivo permite-nos efetuar a ligação, esta pode ser realizada de variadas formas, seja através de SMS, QR code, Bluetooth, email, inserção de código ou acedendo a determinado link.

A interação através do toque transmite ao utilizador uma maior espontaneidade para este interagir com o respetivo ecrã, no entanto por norma é também sinónimo de piores condições de limpeza [BRSB04].

Uma conexão usando *Bluetooth* também pode ser intuitiva, contudo por vezes requer que o utilizador instale software específico no seu dispositivo e o computador do ecrã público necessita de uma configuração própria de modo a aceitar a ligação [BRSB04].

As ligações através de *QR code*, *email*, inserção de código ou *link* exigem que exista uma ligação à Internet para que o transeunte possa usufruir da aplicação. Gehring et al. [GDLC] fazem referência ao protocolo *TUIO*, visível na figura 2.2, para enviar a informação do dispositivo para o ecrã. Inicialmente o ecrã precisa de ser detetado e identificado, depois o dispositivo deteta automaticamente o ecrã em direto, se os dois estiverem conectados na mesma rede. Após isto, o reconhecimento do *QR code* existente no local permite emparelhar os dois dispositivos.

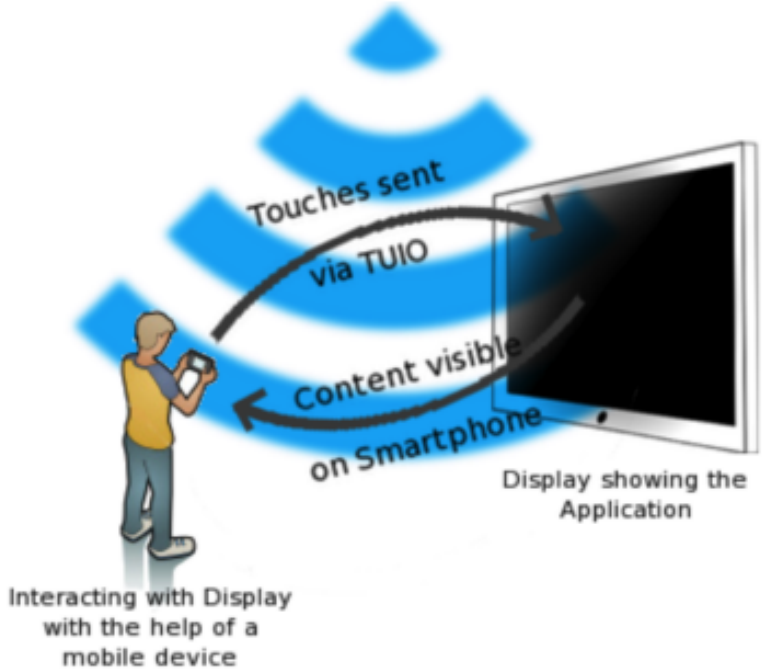


Figura 2.2: *TUIO* Protocolo

O uso de um dispositivo clarifica o paradigma da interação, mas providenciar dispositivos em locais públicos levanta alguns problemas relacionados com a segurança física,

condições sanitárias, manutenção e o uso simultâneo. O uso do seu próprio dispositivo permite resolver estes problemas [BRSB04].

2.1.3 Exemplo de Arquitetura

Como poderia uma aplicação integrar a rede de um ecrã público? Clinch et al. [CDKS12] apresentam uma arquitetura composta por 4 componentes, que é possível observar na figura 2.3:

- **aplicações móveis** - permitem que os transeuntes definam as preferências da aplicação e determinam a respetiva proximidade para com o ecrã;
- **ecrãs** - para renderizar o seu conteúdo;
- **conjunto de aplicações *web*** - especificamente desenvolvidas para ecrãs públicos;
- **diretório** - para fornecer informações sobre a localização geográfica de um ecrã, capacidades e aplicações disponíveis.

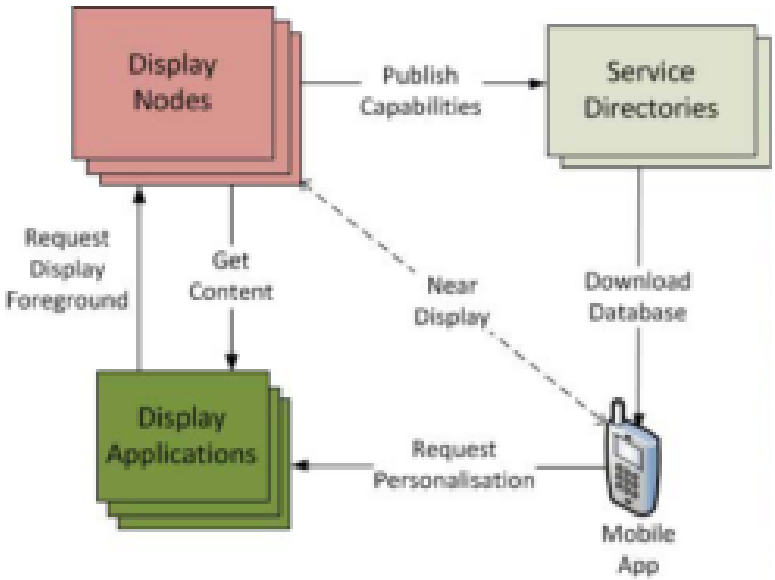


Figura 2.3: Componentes da Arquitetura [GDLC]

2.2 Aplicações para multi-utilizadores

As primeiras aplicações desenvolvidas para múltiplos utilizadores usavam *hardware* caro e desenvolvido para um propósito específico, pois eram necessários dispositivos específicos que permitissem a interação com determinado computador. Atualmente essa interação torna-se mais fácil, sendo possível através de um *tablet* ou *smartphone*. Stewart

et al. [Ste97] lançou o termo *Single Display Groupware (SDG)* com a aplicação KidPad, desenvolvida para crianças, permitindo-lhes desenharem ao mesmo tempo no mesmo computador.

PebblesDraw [MSG98], desenvolvido posteriormente, é um programa de desenho partilhado que permite a todos os seus utilizadores desenharem ao mesmo tempo e, tendo em conta que é um SDG, estes partilham também o mesmo ecrã e consequentemente as mesmas *widgets*. Esta aplicação data de 1998, ano em que o uso dos *smartphones* era diminuto ou mesmo inexistente, usando para a interação *personal digital assistants (PDAs)*, pois eram fáceis de programar, populares, e conectavam-se facilmente a um computador, independentemente do sistema operativo. A maneira convencional de identificar os diversos utilizadores passa por atribuir a cada um, uma cor diferente. No entanto, para esta aplicação foram desenvolvidas novas técnicas de interação para suportarem de forma mais eficiente o uso simultâneo, atribuindo a cada um deles uma forma. Tal como num jogo de tabuleiro, em que cada jogador escolhe o seu peão, também aqui podem escolher qual o objeto que os representa, como um quadrado, triângulo entre outros.

De modo a permitir o uso de vários utilizadores ao mesmo tempo, uma conexão através de *general packet radio service (GPRS)*, *WiFi* ou *Bluetooth*, será a melhor escolha em termos tecnológicos [BRSB04].

2.3 Trabalhos relacionados

- ***Poppet*** [VCBE08] - é uma *framework*, representada na figura 2.4, que utiliza os sensores do telemóvel, como câmaras, acelerómetros ou NFC, que podem ser ligados aos jogos, para correrem nos ecrãs públicos. Foi desenvolvido para um telemóvel em específico, Nokia 5500, contudo é suficientemente genérico para funcionar corretamente com uma larga gama de telemóveis usando o *Bluetooth* como meio de ligação. O reconhecimento dos gestos exige um estudo cuidado, pois a variação de como o utilizador manipula o dispositivo pode produzir anomalias nos *outputs* e não existia método de obter a posição física do mesmo no espaço real, como acontece, por exemplo, no comando da consola *Wii*.

- ***PUREWIDGETS SYSTEM [CJ12]*** - é composto por uma biblioteca de *widgets* e *web service* que permite lidar com eventos interativos. Representa um *toolkit* que suporta múltiplos mecanismos de interação, eventos assíncronos e interação concorrente. Fornece abstrações de alto nível que se adequam ao tipo de interação que normalmente se faz com aplicações públicas e permite que programadores se concentrem sobre o trabalho criativo de *design* de aplicações interessantes e experiências do utilizador. *PureWidgets* suporta diferentes tipos de ligação, como SMS, *email*, *Bluetooth naming*, *Bluetooth OBEX* and *QR codes*. Para a sua implementação foi usada a plataforma *Google's App Engine* e *Google's Web Toolkit*.

Figura 2.4: Sistema *Poppet* [VCBE08]

- ***Super Sync Sports***[1] - uma aplicação *web*, representada na figura 2.5, desenvolvida pela Google que permite ao utilizador jogar se este estabelecer uma ligação ao computador através do seu dispositivo móvel. O objetivo será assumir o papel de um desportista e terminar as diversas provas. São usados *websockets* para permitir a colaboração em tempo real, no seu desenvolvimento foi também usado *HyperText Markup Language (HTML)* e *Cascading Style Sheets (CSS)*

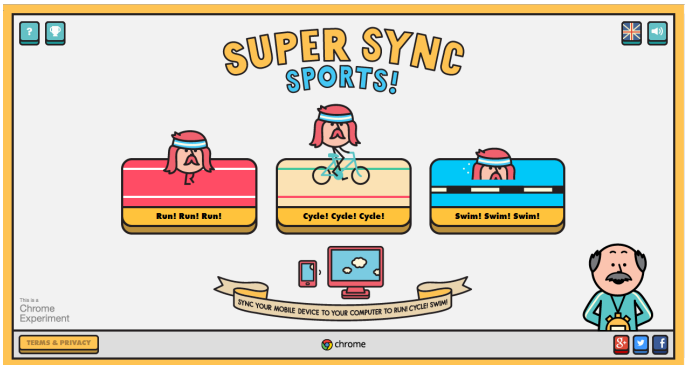


Figura 2.5: *Super Sync Sports*

- **Stripenight Racer**[2] - é também um pequeno jogo, ilustrado pela figura 2.6, que simula uma corrida de automóveis e o utilizador tem a autonomia de controlar o veículo com o movimento do seu dispositivo. Usa a mesma tecnologia para a ligação que a aplicação anterior.

[1] http://www.chrome.com/supersyncsports/#/en-GB/m/tfejb
[2] http://www.stripenight.com/racer/

Figura 2.6: *Stripenight Racer*

Os dois jogos apresentados, foram escolhidos pois demonstram uma interação baseada no paradigma *direct-manipulation*, uma vez que o utilizador vê em tempo-real as alterações conforme movimenta o seu dispositivo.

2.4 Conclusões

A elaboração deste capítulo permitiu a realização de uma pesquisa relacionada com o tema a desenvolver, facilitando o conhecimento e relação com os termos específicos mais usados na área. A criação de aplicações públicas não se centra apenas no desenvolvimento das aplicações enquanto programas de computador, existe todo um conjunto de fatores a ter em conta quando se pensa numa solução possível de ser implementada. Se a aplicação desenvolvida necessitar de um dispositivo para a interação ser possível, existem diversas formas de a conexão ocorrer, podendo ser através de SMS, QR codes, Bluetooth, email, link, etc.

É ainda possível, que estas aplicações sejam utilizadas por várias pessoas ao mesmo tempo, o que requer cuidados específicos quanto ao propósito da aplicação, diferenciação dos utilizadores e facilidade de adição dos mesmos. Nenhum dos trabalhos referidos, corresponde ao que se pretende com esta dissertação enquanto produto final, embora possam vir a ser úteis ao longo do seu desenvolvimento.

Capítulo 3

Metodologia

Neste capítulo é descrita de uma forma sucinta a metodologia usada para alcançar os objetivos propostos. São definidos alguns conceitos importantes para o correto desenvolvimento, bem como as diferentes etapas inicialmente definidas que permitirão atingir o desejado.

3.1 Definições e Terminologia

Neste projeto, de acordo com o que é pretendido, é possível distinguir dois utilizadores finais, o programador que usará a *framework* desenvolvida e o transeunte que utilizará a aplicação do ecrã público. Tendo em conta este último, será adequado considerar dois conceitos relacionados, sendo eles Interação Pessoa-Computador(IPC) e Desenvolvimento Centrado no Utilizador(DCU).

IPC é uma área focada no *design*, avaliação e implementação de sistemas interativos para uso antrópico, dando relevância à satisfação do utilizador final. Enquanto a engenharia de *software* se preocupa com a produção de soluções para aplicações de *software*, IPC foca-se em descobrir métodos e técnicas que melhorem a interação entre pessoa-computador.

Existem algumas razões que fazem com que IPC seja uma área de estudo com valor, muito ativa e em franca expansão [SA06]:

- Considera como principal no desenvolvimento de aplicações o utilizador final e preocupa-se com ele durante todas as fases de desenvolvimento.

- Fornece uma base sobre a qual é possível avaliar métodos de *design* para a sua eficiência e eficácia. O desenvolvimento de um sistema necessita de várias formas para avaliar os métodos usados, que podem ser obtidos através de IPC.

- Proporciona um ambiente baseado no mundo real, permitindo novas teorias baseadas na psicologia humana. E este campo é uma das áreas de crescimento mais rápido no campo da ciência da computação.

Desenvolvimento Centrado no Utilizador

DCU é um conceito para descrever os projetos nos quais o foco central é o utilizador final. É ao mesmo tempo uma filosofia ampla com vários métodos, pois há diversas maneiras através das quais os utilizadores são envolvidos. Uma metodologia baseada em DCU interessa-se pelas necessidades dos mesmos e frequentemente envolve-os durante o processo de *design*, normalmente durante o levantamento de requisitos e testes de usabilidade [AMkP04].

É uma abordagem que observa, regista e analisa as reações e o desempenho dos utilizadores em determinado cenário. É caracterizada por um desenvolvimento iterativo, ou seja, quando são encontrados problemas na fase de testes, são corrigidos e efetuados novos testes.

É importante desenvolver um sistema intuitivo e de fácil utilização aos seus utilizadores. DCU apresenta diversas abordagens que podem ser usadas na implementação de uma solução. Cada uma delas possui etapas diferentes, mostrando como as atividades se relacionam.

Para este projeto foi escolhida uma abordagem iterativa que pressupõe 4 fases, representadas na figura 3.1, sendo elas:

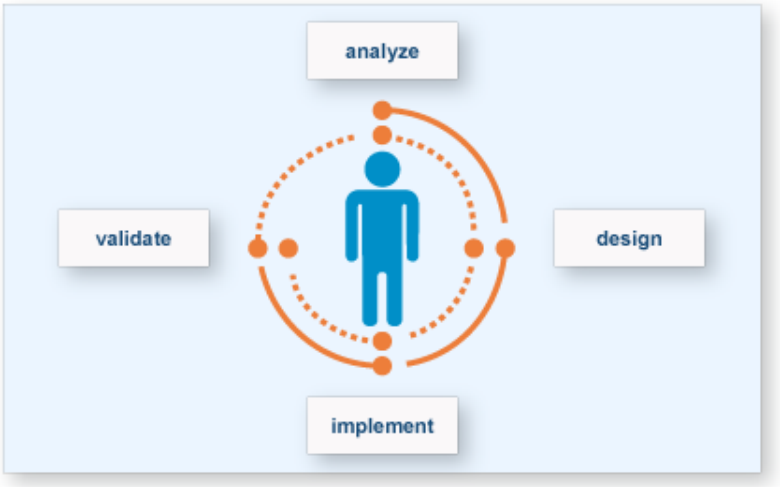


Figura 3.1: Desenvolvimento Centrado no Utilizador[1]

- **Analisar** - Engloba a recolha de requisitos, tendo em conta o contexto de uso e o propósito de desenvolvimento;

[1]http://www.medical-safety-design.de/en/medical-safety-design/user-centered-interface-design/

- **Design** - Projeta possíveis soluções que cumpram o que foi definido na fase anterior;
- **Implementar** - Desenvolvimento de protótipos que tornem percetíveis as soluções idealizadas, nos quais os conceitos têm de ser implementados;
- **Validar** - Avaliação, por parte de especialistas, da usabilidade e possíveis riscos tendo em conta os requisitos do utilizador final e os conceitos envolvidos.

As etapas acima referidas ocorrem de forma iterativa, existindo uma avaliação de desempenho no final de cada iteração. Inicialmente, na fase de análise são definidos os diversos elementos esperados e analisados em detalhe. Seguem-se as restantes fases, ocorrendo os respetivos testes na fase de validação, que fornece *feedback* em cada iteração. A solução pode ser redefinida numa próxima fase de análise de acordo com os resultados obtidos.

3.2 Fases de Desenvolvimento

Enquanto projeto de *software*, é importante uma fase inicial que permita especificar aquilo que se pretende implementar de modo a não divagar. Deste modo será mais fácil que o resultado obtido não fique aquém do que é pretendido.

Por conseguinte, são definidas as etapas previstas, na realização deste projeto, que levarão à solução final:

- definir as aplicações que são pretendidas como exemplo;
- estudar que tipo de controlos serão úteis para as aplicações definidas;
- definir a arquitetura do sistema;
- escolher as tecnologias usadas para a implementação;

Os pontos acima mencionados, funcionarão como guias iniciais antes do desenvolvimento começar, permitindo ter uma ideia do que será necessário ao longo de todo o projeto. Torna-se relevante definir que aplicações se pretende desenvolver, tal como os tipos de controlo, pois existem em grande número, não facilitando a especificação de objetivos. Perceber que arquitetura será usada, servirá para ajudar na posterior escolha de tecnologias.

É desejado que sempre que uma aplicação é desenvolvida, a mesma seja testada por utilizadores, facultando uma análise sobre o que pode ser melhorado, não se incorrendo nos mesmo erros na aplicação seguinte.

Ao longo de todo o processo poderão ocorrer pequenos testes que facilitem o desenvolvimento, evitando grandes alterações na fase final.

Capítulo 4

Solução Implementada

Neste capítulo é descrita de forma pormenorizada a solução implementada de modo a responder aos desafios colocados na introdução, cumprindo assim os objetivos propostos. Inicialmente é caracterizada a arquitetura definida, fazendo referência aos diferentes componentes que constituem o produto final. Posteriormente são referidas de uma forma breve as tecnologias envolvidas no seu desenvolvimento. Por último, é especificada a *framework* desenvolvida e o exemplo implementado.

4.1 Visão Geral

Tal como foi referido na definição dos objetivos, era esperado o desenvolvimento e validação de uma arquitetura que permitisse uma interação baseada na manipulação direta, através de um dispositivo móvel, facilitando a criação de aplicações para ecrãs públicos. Na solução encontrada, de uma maneira global, é possível diferenciar três diferentes componentes, sendo eles: o servidor, desenvolvido em *node.js*; a aplicação, que irá correr no servidor criado comunicando com este através de *web sockets* e ainda o utilizador final, aqui identificado como cliente.

A figura 4.1 apresenta, de uma forma simples, os componentes acima descritos.

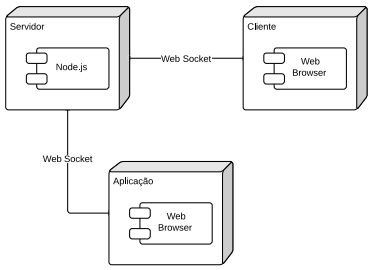


Figura 4.1: Componentes constituintes da solução desenvolvida

Na figura 4.2 é representado um diagrama de sequência mostrando as diferentes interações entre os componentes constituintes do sistema implementado. Inicialmente terá de existir um pedido por parte de um ecrã para aceder à aplicação. Neste momento, a aplicação comunica com o servidor e a partir daí está preparada para pedidos de possíveis utilizadores. Um utilizador, ao querer interagir com a aplicação, está a enviar um pedido para esta, que por sua vez avisa o servidor e este é responsável por mostrar no dispositivo o *widget* da aplicação. Nesse momento, a *widget* liga-se ao servidor e após ocorrer a ligação procede-se à troca de "mensagens" que permitem ao utilizador definir o seu nome. Quando o cliente tiver o nome definido pode usufruir dos *widgets* disponíveis, realizando-se a comunicação para o servidor, que por sua vez envia novamente para a aplicação.

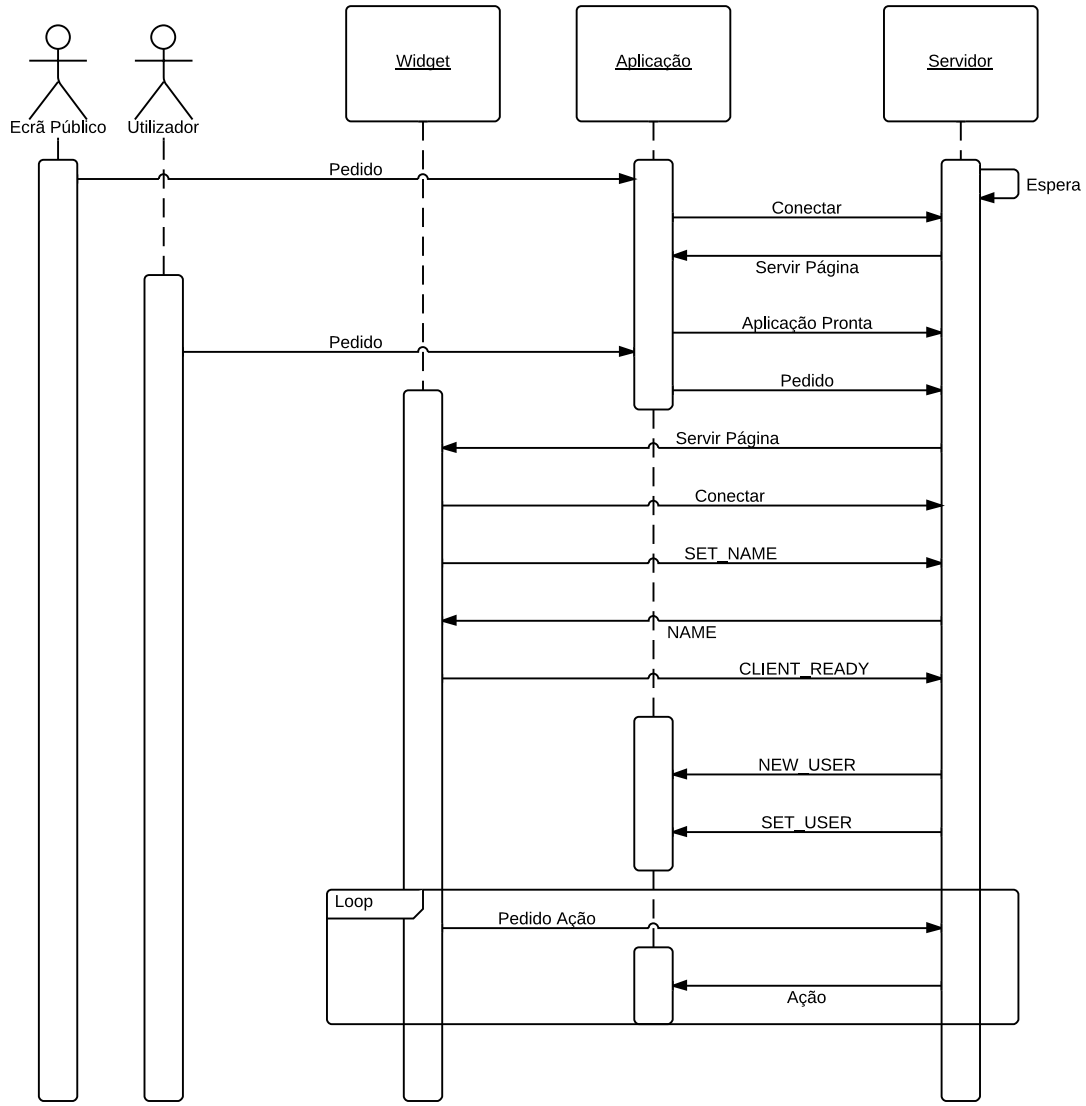


Figura 4.2: Diagrama de Sequência

4.2 Tecnologias Usadas

Ao longo da implementação houve necessidade de optar por diversas tecnologias para que fosse possível alcançar o objetivo desejado. *Node.js* foi usado para a implementação do servidor e *web sockets* e *socket.io* para facilitar a comunicação entre os diversos componentes. Foi também usada a *framework Prototype* que permite a manipulação de classes em *JavaScript*, e uma biblioteca, *Swipeable* que dá resposta a eventos *swipe* facilitando a sua utilização.

- ***Node.js***

  Node.js é uma plataforma construída para facilitar o desenvolvimento de aplicações de alta escalabilidade em tempo real, com base no interpretador *Javascript V8* da *Google*, que antes da execução compila *JavaScript* em código máquina, melhorando consideravelmente o tempo de execução. Deste modo, *Node* permite a construção de aplicações rápidas e altamente concorrentes.

  Segundo Michael Abernethy [Abe11], *node* altera a noção de como um servidor deve funcionar, referindo que o seu objetivo é permitir que um programador construa aplicações com grande escalabilidade e que o código desenvolvido suporte milhares de ligações simultâneas numa só máquina.

  Node.js opera apenas com uma *thread* e não bloqueia as chamadas de entrada e saída, permitindo o suporte das diversas ligações em simultâneo, tal como se pode ver na figura 4.3.

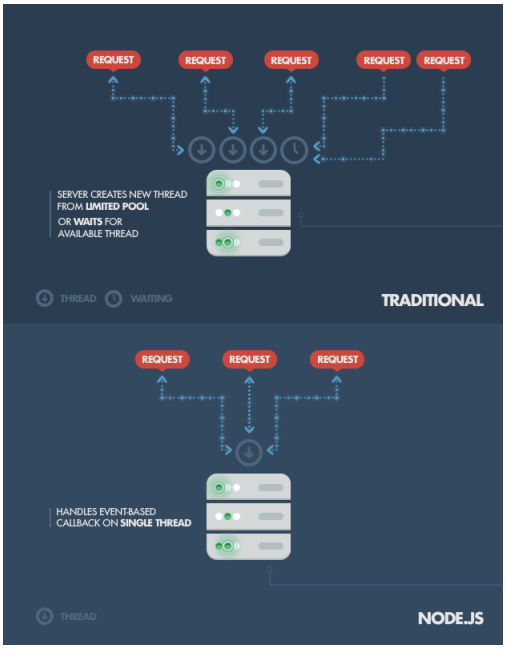


Figura 4.3: Diferença entre técnicas tradicionais de servidores e *Node.js*[1]

[1]http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

- ***Web Sockets***

  De forma a permitir a comunicação do utilizador (lado do cliente) com o servidor criado, foram usados *web sockets*. Estes foram desenvolvidos para serem implementados em aplicações ou servidores *web*, usando um protocolo independente baseado no *Transmission Control Protocol (TCP)*.

  O protocolo *websocket* encontra-se normalizado, o que significa que é seguida uma norma no envio de informação entre o servidor e o *browser* sem que haja uma solicitação por parte do cliente, o que possibilita uma maior interação entre estes, facilitando a criação de aplicações em tempo real. É deste modo criada uma ligação bi-direcional entre o *browser* e o servidor, pois a conexão é mantida aberta enquanto as mensagens são encaminhadas de um lado para o outro.

- ***Socket.io***

  Socket.io é descrita como uma biblioteca *JavaScript* usada no desenvolvimento de aplicações *web*. Esta é composta por duas partes: uma biblioteca para o lado do cliente, que é executada no *browser*, e outra para o lado do servidor, que para já terá de ser implementado em *node.js*, daí este estar acima referido como uma das tecnologias usadas. Quer o lado do cliente quer o do servidor apresentam *APIs* idênticas.

  Usa, principalmente como protocolo, *websockets*, também escolhido como tecnologia usada no desenvolvimento desta solução. Contudo, se necessário, podem ser utilizados outros, como por exemplo *Adobe Flash sockets, JSONP polling*, e *AJAX long polling*. A sua escolha aliada a *websockets* fornece bastante recursos, como a transmissão para múltiplos *sockets*, armazenamento de informação associada a cada cliente e ainda *inputs/outputs* assíncronos.

- **Prototype**

  Prototype é uma *framework* em *JavaScript* que fornece algumas funções para o desenvolvimento de aplicações em *JavaScript*. As suas funcionalidades variam entre pequenos atalhos de programação e principais funções para lidar com *XMLHttpRequest*.

  Esta *framework* fornece ainda uma biblioteca com funções que suporta classes e objetos baseados em classes, algo que não é possível em *JavaScript*.

- ***Swipeable***

  Swipeable trata-se de uma biblioteca que permite obter resposta a eventos *swipe* realizados num dispositivo sensível ao toque, sendo uma abstração do *touchstart*, *touchmove* e *touchend*. Foi incluída no ficheiro *widget.js* para possibilitar o desenvolvimento e o correto funcionamento do controlo através de *swipe*.

Para além das tecnologias acima definidas todo o projeto foi realizado recorrendo a *JavaScript*, HTML e CSS.

4.3 *Framework* Desenvolvida

Um dos principais objetivos centrava-se no desenvolvimento de uma *framework*, que facilitasse o desenvolvimento de aplicações para ecrãs públicos e permitisse a manipulação direta por parte dos utilizadores. Por conseguinte, surge a criação de uma API que permite a construção e utilização de três diferentes tipos de controlo. Na figura 4.4 é representada a estrutura usada, mostrando as classes constituintes, respetivos atributos e métodos e ainda a relação entre as classes.

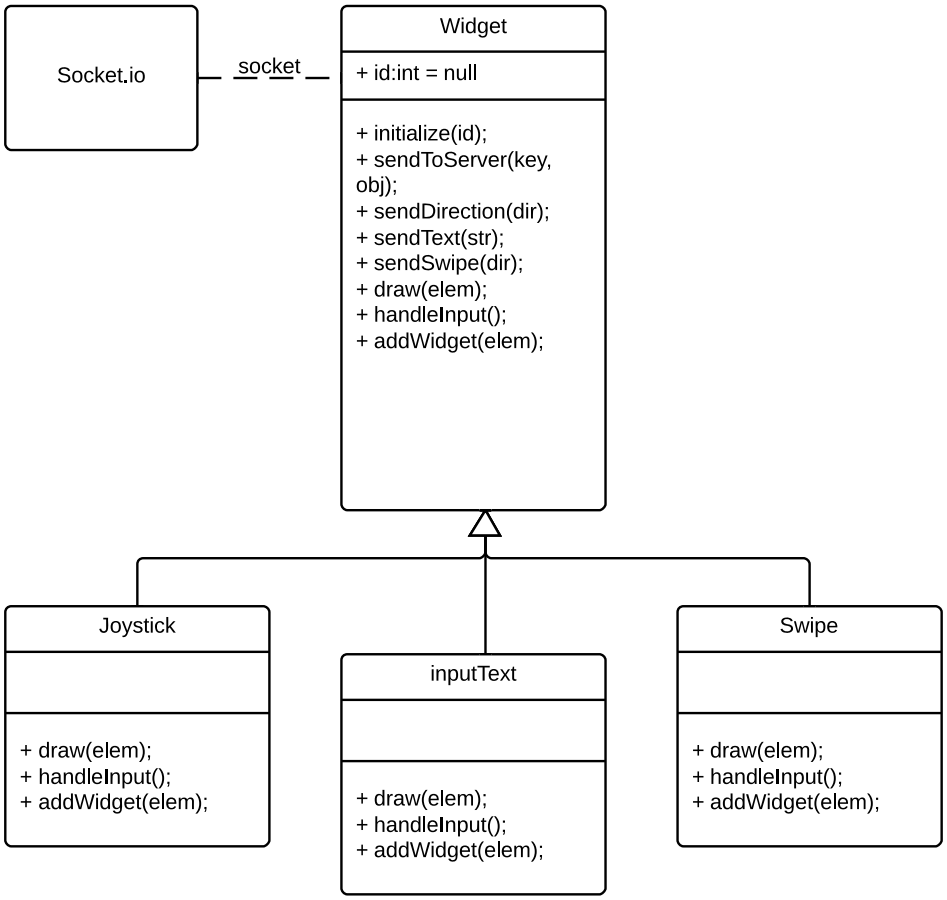


Figura 4.4: Diagrama de classes

A classe *Widget* representa uma abstração dos controlos possíveis, sendo a classe *Joystick*, *Swipe* e *inputText* instâncias desta. Uma vez que se trata de uma relação de herança, quer os atributos, quer os métodos da super classe são herdados pelas respetivas subclasses, permitindo a sua utilização.

4.3.1 Funcionalidades

Com o objetivo de facilitar a criação de aplicações para ecrãs públicos, foi desenvolvida uma API, com um conjunto de funcionalidades que permitem a um programador o seu

uso na criação de diferentes controlos de acordo com o requerido pela aplicação que irão servir.

A classe *Widget*, já definida, apresenta alguns métodos próprios que são usados nas subclasses, permitindo a comunicação com o servidor, como se pode ver tabela 4.1).

Tabela 4.1: Métodos da classe *Widget*

Métodos	**Parâmetros**	**Descrição**
initialize	id	Método responsável por inicializar o *widget*
sendToServer		Método responsável por realizar a comunicação com o servidor
sendDirection	dir	Método que usa o método *sendToServer* para enviar a direção, dir, obtida pela *widget do joystick*
sendText	str	Método que usa o método *sendToServer* para enviar texto, str, obtida pela *widget*
sendSwipe	dir	Método que usa o método *sendToServer* para enviar a direção, dir, obtida pela *widget swipe*

Os controlos referidos dizem respeito aos diferentes tipos de controlo que estão disponíveis para ser usados em qualquer aplicação, e na solução apresentada disponibilizam os métodos presentes na tabela 4.2:

Tabela 4.2: Métodos disponíveis em cada uma das *widgets*

Métodos	**Parâmetros**	**Descrição**
draw	elem	Método responsável por desenhar a widget correspondente no elemento (elem) da página html apresentada no dispositivo
handleInput		Método responsável por atribuir eventos às widgets
addWidget	elem	Método responsável por adicionar a widget à barra de widgets disponíveis no elemento, elem, da página html

Para além dos métodos acima descritos, que constituem as opções disponíveis para a instância de cada tipo de *widget*, o programador tem disponíveis outros dois, que podem ser vistos na tabela 4.3:

Tabela 4.3: Métodos Gerais

Métodos	Parâmetros	Descrição
start	url, options	Método para inicializar as widgets, responsável por efetuar a ligação com o servidor
drawBar	elem	Método responsável por desenhar, no elemento, elem, da página html, a barra que mostra ao utilizador as widgets que tem disponíveis
setOptions	options	Método que permite ao programador alterar o conteúdo do que é enviado da widget para a aplicação

4.3.2 Controlos Definidos

A API desenvolvida apresenta três diferentes tipos de controlos, apresentados na figura 4.5, que o programador terá à sua disposição para implementar, de acordo com aplicação desenvolvida. Neste momento existem como possíveis opções o *joystick*, o *swipe* e ainda outro, composto por uma caixa de introdução de texto.

- ***Joystick***

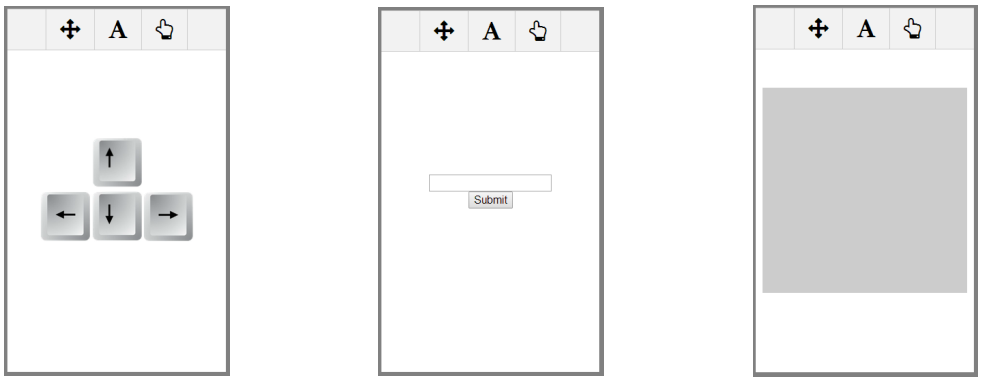
  Controlo composto pelas quatro setas tradicionais (esquerda, direita, cima e baixo) que pode ser utilizado em qualquer aplicação que exija uma movimentação.

- ***Swipe***

  Controlo que usa as propriedade *touch* do dispositivo para controlar a aplicação, definido apenas para reconhecer uma direção, embora isso possa ser alterado.

- ***InputText***

  Controlo que permite ao utilizador a interação com aplicações que exijam a introdução de texto.



(a) *Widget Joysitck* (b) *Widget InputText* (c) *Widget Swipe*

Figura 4.5: Diferentes Tipos de Controlo Disponíveis

4.3.3 Utilização

O desenvolvimento da *framework* apresentada tinha como objetivo auxiliar na criação de controlos para aplicações com as quais fosse possível interagir em ambiente público com um dispositivo móvel.

Um programador que pretenda usufruir das suas funcionalidades deverá:

1. Criar um documento *HTML* com o nome *widget.html* e a estrutura apresentada na figura 4.6
2. Incluir no ficheiro as seguintes bibliotecas:
   - prototype.js
   - swipeable.js
   - socket.io
3. Incluir no mesmo ficheiro um *script* onde são chamadas as funções da API.

```
1
2
3   <!DOCTYPE html>
4   <html>
5   <head>
6   </head>
7   <body>
8       <div id="widget_bar">
9       </div>
10      <div class="centered" id="widget">
11      </div>
12  </body>
13  </html>
14
15
```

Figura 4.6: Estrutura para criação de *widgets*

Apesar de haver liberdade nas escolha das widgets a implementar, o método *start(url,options)* tem obrigatoriamente de ser chamado antes da criação das *widgets* pretendidas.

4.4 Exemplo Implementado

No âmbito do presente projeto, foi criada uma aplicação exemplo, que permite ao utilizador o uso dos três tipos de controlo disponíveis.

Assim, foi desenvolvido o clássico jogo *Snake*, no qual o utilizador é obrigado a inserir o seu nome, usando para isso a opção de introdução de texto e só posteriormente poderá escolher entre o *joystick* ou o *swipe*, servindo qualquer um deles para controlar a direção da cobra durante o jogo. Cabe ao utilizador a escolha do controlo adequado de acordo com as suas preferências ou facilidades, necessitando apenas de carregar na barra superior no botão correspondente à *widget* desejada.

4.4.1 Ligação

Tendo em conta que a aplicação funcionará num ecrã público, e que é suposto o transeunte usufruir da mesma através do seu dispositivo móvel, é necessário que exista um método que permita a ligação entre os dois.

Na solução implementada, esta ligação ocorre através da leitura de um *QR code*, que se encontra sempre disponível no ecrã.

4.4.2 Feedback da aplicação

Uma vez que se trata de uma interação pública com a qual uma ou mais pessoas devem poder interagir, é importante que durante a sua utilização, a pessoa que se está a servir desta, seja capaz de perceber o fluxo de informação que ocorre entre ela e a aplicação. Por exemplo, deverá conseguir identificar-se quando existe mais do que um utilizador e ainda perceber em que momento pode começar a interagir.

No exemplo implementado, sempre que uma nova pessoa se conecta com o jogo, o servidor lança um alerta que faz aparecer uma notificação como mostra a figura 4.7, informando sobre a existência de um novo jogador em condições de usufruir da aplicação presente.

Figura 4.7: Notificação de Novo Utilizador

Quem desejar interagir com a solução implementada necessita de introduzir um nome de utilizador, antes de realmente poder jogar. No ecrã surgirá então o nome escolhido, que, neste caso concreto, surge acompanhado dos pontos conseguidos, e vai sendo atualizado à medida que são alterados.

4.4.3 Distinção de utilizadores

Neste tipo de aplicações é importante ter em conta de que modo é feita a distinção entre os diversos utilizadores, percebendo que cada um dos intervenientes terá de se reconhecer durante o momento da interação.

No jogo implementado, tal como foi referido no ponto anterior o jogador é identificado pelo nome que este escolhe quando se conecta com o ecrã. No entanto, há ainda a necessidade deste se reconhecer na aplicação que está no momento a utilizar. A solução encontrada, no contexto do exemplo usado, passou por colorir o nome do jogador com a cor da cobra que o representa durante o jogo, como é mostrado na figura 4.8, permitindo assim a distinção dos jogadores e transmitindo-lhes quem é quem no jogo que está a decorrer.

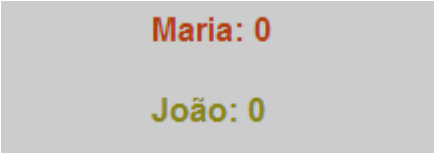


Figura 4.8: Distinção entre utilizadores

4.4.4 Como utilizar

Enquanto utilizador final, para poder utilizar a aplicação desenvolvida será necessário:

- Possuir um dispositivo móvel com acesso a Internet;
- Ter um *browser* instalado;
- Ter instalada no seu dispositivo uma aplicação que permita a leitura de *QR codes*.

O diagrama da figura 4.9, ilustra, de forma sequencial os passos existentes numa interação com a aplicação.

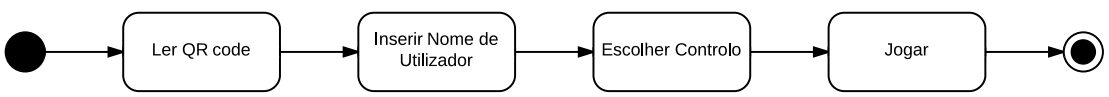


Figura 4.9: Interação com a aplicação

Ao aproximar-se de um ecrã que tenha a correr este exemplo, o transeunte deve:

1. Ler o *QR code* presente no ecrã;
2. Introduzir um nome pelo qual quer ser identificado;
3. Escolher o controlo desejado para jogar;
4. Jogar.

Capítulo 5

Testes

A realização de testes é uma fase importante no desenvolvimento de qualquer produto, permitindo descobrir alguma falhas no trabalho efetuado e dando sugestões de melhorias futuras. Neste capítulo são descritos os diversos tipos de testes executados, bem como as conclusões a que os mesmos permitiram chegar.

5.1 Testes Iterativos

Ao longo de todo o desenvolvimento, sempre que uma nova funcionalidade era implementada surgia a necessidade de a testar, de forma a perceber se esta efetuava a ação desejada.

Este tipo de testes, permite a quem está a desenvolver a aplicação, obter de forma rápida *feedback* do trabalho acabado de realizar, prevenindo erros futuros semelhantes e trabalho desnecessário.

Por exemplo, na solução implementada, aquando da criação dos tipos de controlo, primeiro foi definido e implementado apenas um, neste caso a *widget joystick*, que foi testado verificando se a comunicação com a aplicação existia, e se as setas executavam no jogo as ações supostas. Assim, qualquer erro detetado e corrigido contribuiu para uma maior facilidade na criação das restantes *widgets*, diminuindo a correção de erros no final da implementação.

5.2 Testes Finais

Os testes para verificar se as funcionalidades implementadas correspondiam ao especificado inicialmente são de fácil realização e apenas requerem que alguém experimente o que foi desenvolvido. Contudo neste projeto era também importante perceber de que

modo outros programadores poderiam usar a *framework* desenvolvida com o intuito de definirem alguns tipos de controlo para as sua próprias aplicações.

Para a realização destes testes foram escolhidos três estudantes do 5º ano do Mestrado Integrado em Engenharia Informática e Computação. Ao longo do seu percurso académico tiveram duas unidades curriculares onde fizeram *web development* e tiveram de interagir com *HTML5*, *CSS3* e *JavaScript*.

Um dos elementos, o sujeito A, apenas possuía a experiência adquirida durante o percurso académico. Os outros dois elementos, os sujeitos B e C, já tinham realizado alguns projetos extra-curriculares que os obrigou a explorar um pouco mais estas tecnologias, fornecendo-lhes uma maior experiência e conhecimento.

Inicialmente, foram explicados aos intervenientes os objetivos do projeto, para que eles estivessem enquadrados e pudessem perceber o seu papel nesta atividade. Também lhes foi dada, antes de iniciarem, uma breve explicação sobre o que teriam de fazer para usar a *framework*.

De seguida, foi-lhes dado todo o projeto desenvolvido, com a exceção de dois ficheiros. Um que continha o código do jogo implementado e o outro que dizia respeito à implementação dos controlos pretendidos.

O objetivo era que desenvolvessem uma pequena aplicação e que usassem para a definição dos controlos exigidos a *API* desenvolvida. Os participantes dispunham de 2 horas e 30 minutos para a realização desta atividade. O tempo dado foi utilizado de maneiras diferentes pelos participantes. Enquanto o utilizador A apenas se preocupou em implementar uma solução, os outros dois tiveram a preocupação de fazer perguntas pertinentes sobre a *framework* e sugerir alterações.

Das pessoas acima referidas, todas optaram pela implementação de jogos, contudo escolheram exemplos diferentes. Na tabela 5.1, é possível ver as aplicações escolhidas, bem como um pequeno resumo das soluções obtidas.

Tabela 5.1: Resultados Obtidos

Elemento	**Aplicação**	**Funcional**	**Multi-Utilizador**
A	Jogo da Forca	Sim	Não
B	Corrida de automóveis	Sim	Sim
C	Tetris	Sim	Não

Em todos os exemplos, os estudantes conseguiram com sucesso implementar os tipos de controlo existentes e ter uma aplicação funcional contudo, em todos eles foram referidas pequenas falhas ou sugestões relativas ao projeto.

O decorrer dos testes permitiu obter diferentes opiniões de acordo com o desenvolver da solução:

- **Jogo da forca:** Só foi usado o controlo que exigia introdução de texto. O utilizador inseria o seu nome de jogador e poderia começar a jogar.

Só foi conseguida a versão para um jogador.

- **Corrida de automóveis:** Foi implementada com sucesso, sendo possível usar os três diferentes tipos de controlo, começando por introduzir o nome de jogador, seguido da escolha da *widget* preferido para controlar o veículo.

  A solução apresentada suportava o modo multi-jogador, em que os jogadores eram distinguidos pela cor do automóvel e respetivo nome.

  Foi sugerida a criação de um novo tipo de controlo que usasse o acelerómetro do dispositivo, de modo a permitir uma interação através do movimento do mesmo.

- **Tetris:** À semelhança dos anteriores, também este exemplo se encontrava funcional no fim do tempo dado. Foram usados dois diferentes tipos de controlo, a caixa de texto para a escolha do nome do utilizador e o *joystick* para controlar as peças do jogo.

  Tal como aconteceu no exemplo do jogo da forca, este apenas podia ser jogado por um utilizador. A pessoa que o implementou possuía a solução para permitir um modo multi-jogador, no entanto, por falta de tempo não houve a possibilidade de implementar. A solução passaria pela existência de um número de "tabuleiros" de tetris igual ao número de jogadores.

Adicionalmente, foram também sugeridas pequenas alterações a nível de funções da *API* desenvolvida que foram corrigidas. O método *setOptions(options)*, que permite ao programador alterar a informação que é enviada através da *widget*, inicialmente estava definido para cada *widget* independentemente. Após a realização dos testes, foi sugerido pelo sujeito B, que faria sentido ser um método geral, uma vez que a informação enviada, independentemente do tipo de controlo seriam iguais para determinada aplicação. O sujeito C sugeriu que seria preferível o programador não ter de seguir um estrutura específica para o ficheiro *HTML*, pois condiciona a liberdade do mesmo.

Ainda durante a realização destes testes foi possível constatar que existem algumas alterações a efetuar no código original para a implementação de novas aplicações, o que dificulta o trabalho de futuras pessoas que não estejam familiarizadas com código fornecido. Por exemplo, é necessário alterar três vezes o endereço onde a aplicação está a correr, o que deveria ser evitado para que futuros utilizadores não necessitem de alterar ficheiros exteriores à *framework*. É também necessário que o programador preste especial atenção ao ficheiro *index.js*, local onde são chamadas funções específicas da aplicação implementada.

5.3 Testes Públicos

Aquando da definição e especificação do projeto foi referido que seria vantajoso testar a solução implementada em ecrãs públicos, no entanto, devido a algum atraso no desenvolvimento, o mesmo não foi possível.

A realização destes testes permitiria tirar algumas conclusões relativas à utilização por parte dos transeuntes, que poderiam ser resposta às seguintes questões:

- **Antes de efetuar a ligação com o ecrã:**
  - Que riscos estou a correr se me ligar a este ecrã?
  - O que tenho de fazer para me conseguir ligar?
- **Após estar conectado com o ecrã**:
  - O que tenho de fazer para usar a aplicação?
  - Como sei quem sou eu durante a interação?
  - O que faço quando desejar terminar?

5.4 Conclusões

Após a realização dos testes previamente descritos, houve a possibilidade de perceber realmente de que modo a aplicação era vista por alguém que não estava a par do seu desenvolvimento e que alterações deveriam ser efetuadas para obter uma melhor *performance*.

A sua realização permitiu descobrir falhas que puderam ser corrigidas ainda antes de se dar por concluída a solução final e outras que ficaram registadas como melhorias a efetuar caso exista a hipótese de dar continuidade ao trabalho realizado.

A realização dos últimos testes referidos poderá acontecer em qualquer altura sendo apenas necessário a existência de infra-estruturas adequadas.

Capítulo 6

Discussão Crítica

Ao longo de todo o processo de desenvolvimento, para além do cumprimento dos objetivos foi também importante perceber de que modo seria possível alcançá-los de forma eficaz e que tecnologias deveriam ser usadas.

O desenvolvimento de aplicações de cariz público levanta algumas questões, que no momento do desenvolvimento têm alguma importância nas escolhas efetuadas.

6.1 Escolhas Tecnológicas

Quando se fala de tecnologia, é do conhecimento geral que na maior parte das vezes, para a mesma finalidade existem diversas opções. Deste modo, é necessário ponderar e perceber qual delas se enquadra melhor na solução pretendida e se os resultados finais que serão obtidos correspondem ao que é desejado.

Uma vez que o projeto exigia que existisse uma comunicação entre o dispositivo do utilizador final e a aplicação, que o mesmo controlaria, e que no momento da interação, poderia existir mais do que um cliente, recorreu-se ao uso da biblioteca *Socket.io*, em conjunto com o protocolo *websocket*. Estes criam uma ligação bi-direcional, mantendo a conexão aberta enquanto as mensagens são encaminhadas de um lado para o outro, permitindo a transmissão de informação para múltiplos *sockets*, o armazenamento de informação associada a cada cliente e ainda *inputs/outputs* assíncronos.

Posteriormente, no desenvolvimento da API que daria origem aos *widgtes*, era pretendido o uso de classes com polimorfismo, de modo a que todos os tipos de controlo tivessem uma construção semelhante e que a sua implementação se tornasse mais intuitiva para futuras aplicações. No entanto *JavaScript* não suporta a manipulação de classes nem de objetos baseados nas mesmas. Esta característica de *JavaScript* leva à introdução da *framework Prototype.js* na solução.

Outra questão que exigiu uma reflexão mais cuidada e uma consequente tomada de decisão centrou-se no que o transeunte teria de fazer para que se conseguisse ligar a um ecrã e usufruir da aplicação. A escolha recaiu sobre a leitura de um *QR code*.

Apesar desta decisão ter como desvantagem a obrigatoriedade de o utilizador possuir no seu dispositivo um leitor de *QR codes* é por outro lado algo que suscita o interesse de quem passa por um ecrã, despertando a sua curiosidade e levando-o a querer experimentar. Segundo Wilson [WFSE14], os *QR codes*, são referidos como algo bastante simples de usar, recorrem a poucos recursos, como tempo e dinheiro, e mesmo com uma fraca utilização por parte do público em geral, trata-se de uma tecnologia demasiado simples para ignorar.

O uso das restantes tecnologias surge quase implicitamente ao longo do desenvolvimento, conforme os objetivos que se pretendiam alcançar.

6.2 Exemplo Implementado

Pretendia-se encontrar uma solução para os desafios relacionados com o tema proposto que são descritos na introdução. Foi solucionada a questão relacionada com a utilização da aplicação por um ou mais utilizadores, bem como os diferentes tipos de controlo que serão suportados.

O desenvolvimento da aplicação para a prova do conceito não fazia parte do contexto da dissertação, podendo integrar-se um exemplo já existente com a API desenvolvida. Consequentemente, a escolha efetuada surge com base numa pesquisa de jogos desenvolvidos em *JavaScript*, que permitissem o modo multi-jogador.

Optou-se pelo clássico jogo da *Snake*, por ser um jogo conhecido, que por não necessitar de instruções extensas, permite uma interação rápida, sendo desafiante e no caso de um modo multi-jogador torna-se competitivo. O exemplo encontrado só continha o modo de jogo individual, no entanto, foi fácil a sua alteração para que houvesse a possibilidade de mais do que uma pessoa jogar ao mesmo tempo.

No exemplo implementado, tal como já foi referido, os diferentes utilizadores são distinguidos com base na cor da cobra que os representa durante o jogo e ainda conseguem identificar-se inserindo logo no início o seu nome de utilizador.

Esta é apenas uma das possibilidades encontradas para dar resposta à distinção num sistema que suporta multi-utilizadores. No entanto, outras hipóteses poderiam ser usadas, estando esta característica bastante dependente do tipo de aplicação a implementar. No caso de se tratar de um sistema colaborativo, a distinção é mais fácil se for atribuída aos jogadores uma maneira de eles próprios escolherem a sua identificação, seja através de uma cor, de um nome ou até de um objeto que os identifique. Se, por outro lado, for um sistema independente, onde cada utilizador necessita do seu espaço será mais vantajosa a divisão do ecrã em partes iguais para cada um.

Na solução apresentada estão disponíveis três diferentes tipos de controlo representados por um *joystick*, uma caixa para introdução de texto e ainda por uma área de *swipe*. A seleção destes ocorreu na tentativa de abranger um grande número de aplicações, não sendo necessária a criação de novas *widgets* para que um programador consiga criar aplicações interativas.

Inicialmente foram apenas definidos o *joystick* e a *widget* de introdução de texto. O *swipe* apesar de ter a mesma funcionalidade que o *joystick*, surge de forma a fornecer outra alternativa ao utilizador recorrendo aos recursos disponíveis no seu dispositivo. Julie Rico [RB10] conclui que uma experiência positiva aumenta a recetividade a controlos gestuais. Neste caso concreto, os movimentos de *swipe*, levando à repetição da sua utilização.

Testes Realizados

Os testes realizados já quase no fim do desenvolvimento foram uma etapa importante que permitiu fazer um balanço do que tinha sido implementado.

Através destes, foi possível entender que, apesar dos controlos definidos serem bastante abrangentes existem outras possibilidades que seriam uma mais valia na *framework* desenvolvida, dando resposta a um maior número de aplicações. Uma vez que os testes apenas foram efetuados quase na reta final do projeto, não houve tempo para esta melhoria, ficando assim como possível trabalho futuro.

A sua realização permitiu ter uma visão externa, sugerindo alterações que para quem esteve no desenvolvimento da solução, muito provavelmente, passariam despercebidas. Algumas destas sugestões foram devidamente consideradas e procedeu-se à respetiva alteração ainda antes da entrega final. Outras ficaram apenas registadas para que, futuramente, se possa proceder à sua execução.

Destarte, concluí-se que a *framework* desenvolvida pode ainda ser melhorada, facilitando ainda mais o desenvolvimento de aplicações a ser utilizadas num ambiente público.

Capítulo 7

Conclusões e Trabalho Futuro

No presente capítulo, em modo de conclusão, é elaborado um balanço entre o definido inicialmente e o trabalho que foi desenvolvido. Na última secção são referidas as melhorias que poderão ser efetuadas e perspetivas futuras relativas ao projeto realizado.

7.1 Conclusão

É cada vez mais comum em áreas públicas a existência de ecrãs públicos com os quais os transeuntes podem interagir usando diferentes tecnologias.

Na elaboração deste projeto era pretendia encontrar-se uma solução que disponibilizasse uma interação multi-utilizador baseada no paradigma de manipulação direta através do dispositivo móvel do cliente. Também era esperado desenvolver uma *framework* que facilitasse o desenvolvimento de aplicações para esse conceito.

No início tinha sido planeada a implementação de diferentes tipos de aplicações que possibilitassem o uso de controlos diferentes para uma melhor avaliação da *framework* desenvolvida. Contudo com o avançar do tempo isso tornou-se impossível, existindo apenas um exemplo, o jogo da *Snake*, no qual foram testados todas as *widgets* criadas.

A *framework* apresentada foi testada, permitindo alterações de modo a melhorar a solução final. No entanto, ficaram por efetuar algumas das sugestões recebidas que seriam uma mais valia aquando da implementação de novas aplicações.

Em suma, apesar de nem tudo o que foi definido inicialmente ter sido alcançado, é possível afirmar que os objetivos gerais foram atingidos faltando apenas a implementação de mais exemplos específicos de modo a abranger um maior leque de aplicações e consequentes problemáticas.

7.2 Trabalho Futuro

Tendo em conta o trabalho realizado, no futuro, haverá facilidade em continuar o desenvolvimento da *framework* apresentada bem como a introdução de novas aplicações e de novos métodos a partir dos quais, o utilizador, se pode ligar a um ecrã.

Uma vez que apenas estão definidos três diferentes tipos de controlo, será plausível a introdução de novas *widgets*, como por exemplo, a criação de um que permita a seleção de objetos e consequente manipulação dos mesmos e outro que recorra ao uso do acelerómetro do dispositivo como forma controlo.

As sugestões recebidas durante a realização dos testes, que não foram realizadas, serão um possível ponto de partida se for desejada a continuidade do desenvolvimento da *API*, facilitando todo o trabalho posterior.

A continuidade das aplicações será da responsabilidade de alguém que pretenda implementar um sistema de interação. Apenas será necessário que a mesma se encontre desenvolvida em *JavaScript*.

Na solução apresentada, a leitura do *QR code* é a única forma disponibilizada para que o utilizador possa interagir com o jogo. Será vantajoso, numa abordagem futura, a criação de novos métodos que facilitem e incentivem o uso por parte dos transeuntes.

Uma das problemáticas que talvez ficou mais aquém das expectativas e, que no futuro deveria ser melhorada, centra-se na distinção dos utilizadores, para a qual existem múltiplas soluções e apenas uma foi explorada.

Referências

[Abe11] Michael Abernethy. Just what is Node . js ? pages 1–10, 2011.

[AMkP04] Chadia Abras, Diane Maloney-krichmar e Jenny Preece. User-Centered Design. pages 1–14, 2004.

[BRSB04] Rafael Ballagas, Michael Rohs, Jennifer G Sheridan e Jan Borchers. BYOD : Bring Your Own Device. 2004.

[CDKS12] Sarah Clinch, Nigel Davies, Thomas Kubitza e Albrecht Schmidt. Designing application stores for public display networks. In *Proceedings of the 2012 International Symposium on Pervasive Displays - PerDis '12*, pages 1–6, New York, New York, USA, June 2012. ACM Press. URL: http://dl.acm.org/citation.cfm?id=2307798.2307808, doi:10.1145/2307798.2307808.

[CJ12] Jorge C. S. Cardoso e Rui José. PuReWidgets: a programming toolkit for interactive public display applications. In Steve Reeves José Creissac Campos, Simone D. J. Barbosa, Philippe Palanque, Rick Kazman, Michael Harrison, editor, *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '12*, page 51, New York, NY, USA, June 2012. ACM Press. URL: http://dl.acm.org/citation.cfm?doid=2305484.2305496http://figshare.com/articles/PuReWidgets_presentation_slides_at_EICS_2012/92666, doi:10.1145/2305484.2305496.

[DLJS12] Nigel Davies, Marc Langheinrich, Rui Jose e Albrecht Schmidt. Open Display Networks: A Communications Medium for the 21st Century. *Computer*, 45(5):58–64, May 2012. URL: http://www.computer.org/csdl/mags/co/2012/05/mco2012050058-abs.htmlhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6174992, doi:10.1109/MC.2012.114.

[GDLC] Sven Gehring, Florian Daiber, Christian Lander e D Campus. Towards Universal , Direct Remote Interaction with Distant Public Displays. pages 3–6.

[MSG98] Brad a. Myers, Herb Stiel e Robert Gargiulo. Collaboration using multiple PDAs connected to a PC. *Proceedings of the 1998 ACM conference on Computer supported cooperative work - CSCW '98*, pages 285–294, 1998. URL: http://portal.acm.org/citation.cfm?doid=289444.289503, doi:10.1145/289444.289503.

[RB10] Julie Rico e Stephen Brewster. Usable Gestures for Mobile Interfaces : Evaluating Social Acceptability. pages 887–896, 2010.

[SA06] Serengul Smith-Atakan. *Human-computer interaction*. Cengage Learning EMEA, 2006.

[Ste97] Jason E Stewart. Single display groupware. In *CHI'97 Extended Abstracts on Human Factors in Computing Systems*, pages 71–72. ACM, 1997.

[VCBE08] Tamas Vajk, Paul Coulton, Will Bamford e Reuben Edwards. Using a Mobile Phone as a "Wii-like"Controller for Playing Games on a Large Public Display. *International Journal of Computer Games Technology*, 2008:1–6, 2008. URL: http://www.hindawi.com/journals/ijcgt/2008/539078/, doi:10.1155/2008/539078.

[WEB+] Dennis Wilmsmann, Juliane Exeler, Markus Buzeck, Tim Jay e Antonio Kr. Display Blindness : The Effect of Expectations on Attention towards Digital Signage. pages 1–8.

[WFSE14] Citation Wilson, Andrew M Forthcoming, Access Services e A M Edt. QR Codes in the Library : Are They Worth the Effort ? Analysis of a QR Code Pilot Project The Harvard community has made this article openly available . Please share how this access benefits you . Your story matters . Accessed. 2014.