UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
(MDCC)
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

WILLIAN ALMEIDA RODRIGUES

TARDIGRADE - UM FRAMEWORK ANDROID PARA DESENVOLVIMENTO DE JOGOS DE CARTAS UBÍQUOS

FORTALEZA
2016

WILLIAN ALMEIDA RODRIGUES

TARDIGRADE - UM FRAMEWORK ANDROID PARA DESENVOLVIMENTO DE JOGOS DE CARTAS UBÍQUOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-graduação em Ciência da Computação (MDCC) do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação

Orientador: Prof. Dr. Windson Viana de Carvalho

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R617t Rodrigues, Willian Almeida.
Tardigrade - Um Framework Android para Desenvolvimento de Jogos de cartas Ubíquos / Willian Almeida Rodrigues. – 2016.
110 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2016.
Orientação: Prof. Dr. Windson Viana de Carvalho.

1. Computação Ubíqua. 2. Jogos Ubíquos. 3. Framework. 4. Jogos de Cartas Ubíquos. I. Título.
CDD 005

WILLIAN ALMEIDA RODRIGUES

TARDIGRADE - UM FRAMEWORK ANDROID PARA DESENVOLVIMENTO DE JOGOS DE CARTAS UBÍQUOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-graduação em Ciência da Computação (MDCC) do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação

Aprovada em: 28 de novembro de 2016

BANCA EXAMINADORA

Prof. Dr. Windson Viana de Carvalho (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. Fernando Antonio Mota Trinta
Universidade Federal do Ceará - UFC

Prof. Dr. Lincoln Souza Rocha
Universidade Federal do Ceará - UFC

Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão - UFMA

Ao sol, que sempre volta a brilhar.

AGRADECIMENTOS

Durante os 30 meses de mestrado enfrentei muitos desafios, desde a mudança para um estado novo até a vida longe de familiares e amigos. Mas ao longo dessa jornada encontrei quem tornasse o caminho menos difícil.

Se existe alguma entidade maior a zelar por nós, ela foi quem mais esteve presente durante os piores e melhores momentos, me fazendo lembrar dos meus objetivos e me dando força pra continuar (e não foi pouca!). Obrigado.

Agradeço aos meus pais que me apoiaram e se esforçaram para que eu pudesse estar em Fortaleza e concluir essa etapa.

Agradeço a família que me acolheu em Fortaleza no meu primeiro mês e que sempre me deu suporte e me amparou enquanto eu ainda perdido tentava me situar na cidade. Obrigado Tio Elione, Tia Flaubenia, seus filhos e principalmente meu amigo Leontino que quis me acolher junto a sua família, serei sempre grato a todos vocês.

Agradeço aos velhos amigos, que acreditaram em mim e que me deram apoio mesmo de longe. Rafael, João Victor, Rayssa, Daniela, Lorrayne Lima e Lorrayne Rodrigues e muitos outros. Aos amigos que fiz em Fortaleza: Pedro, Fábio, Camila, Priscila, Queiroz, Paula, Lorena, obrigado por todas as madrugadas jogando lobo, elas muita das vezes era minha válvula de escape. E a Paula é lobo.

Agradeço ao meu orientador Windson Viana que teve uma paciência enorme comigo e que dividia numa mesma conversa o papel de amigo e de orientador, me incentivando e me dando esperanças de que eu iria conseguir terminar tudo (e conseguimos!). Obrigado por todos os momentos em que você me apoiou e acreditou em mim me dando conselhos e dando puxões de orelha, sem você eu não teria chegado nem perto disso.

Agradeço ao meu co-orientador Glaudiney Mendonça, que não pode ser devidamente citado como co-orientador mas teve um papel importante para esse trabalho. Obrigado por me permitir tomar parte do seu tempo, pela atenção data e pela prestatividade em muitos momentos para me ajudar nesse trabalho.

À banca da dissertação. Ao professor Carlos Salles, por participar da banca de minha defesa. Ao professor Fernando e Professor Lincoln, por todos os conselhos dados e por ter contribuído para o aperfeiçoamento do trabalho com críticas e sugestões nas bancas de pré-proposta e proposta no decorrer do mestrado. Aos meus colegas Paulo Arthur e Luís Fernando, que contribuíram imensamente durante o desenvolvimento desse trabalho,

valeu pela força.

Agradeço também aqueles que passaram pelo meu caminho durante esses anos. Algo de cada um deles me ajudou em algum momento. E a todos aqueles que, embora não nomeados, presentearam-me com seus inestimáveis apoios em distintos momentos. A todos o meu reconhecido muito obrigado!

"Chase your dreams, and remember me, sweet bravery. 'Cause after all those wings will take you, up so high"

(Adam Young)

RESUMO

Pesquisas em jogos ubíquos buscam descobrir como dispositivos computacionais e sensores podem ser utilizados para promover entretenimento e criar jogos que borrem a linha entre o real e o virtual. Esses jogos herdam ao mesmo tempo desafios tradicionais do desenvolvimento de jogos e problemas intrínsecos aos sistemas ubíquos, tais como: a heterogeneidade dos dispositivos, a descoberta de dispositivos, a comunicação volátil e a limitação dos recursos disponíveis. Neste trabalho de mestrado propõe-se um modelo para o desenvolvimento de jogos de cartas ubíquos chamado Tardigrade. Ele busca minimizar os desafios de implementação desse tipo de jogo ubíquo. O Tardigrade abrange três tipos de cartas: carta tangível-identificável, carta tangível-interativa e carta tangível e sensível ao contexto. Um *framework* também foi implementado seguindo o modelo Tardigrade. Ele fornece componentes para o reconhecimento e gerenciamento de cartas, o gerenciamento do estado de jogo e para a descoberta e a comunicação entre dispositivos. Uma prova de conceito foi implementada a partir do *framework*: o jogo Hyper Try. Esse jogo móvel ilustra como o *framework* pode ser instanciado para a criação de jogos de cartas ubíquos. Quatorze voluntários avaliaram o HyperTry por meio da metodologia PLEXQ (*Playful Experience framework Questionnaire*). Os resultados da avaliação indicam que o jogo desenvolvido é divertido e desperta o interesse, principalmente, devido às suas mecânicas inovadoras. Um outro experimento de avaliação foi conduzido para o *framework* com propósito de verificar as suas funcionalidades. Este experimento foi realizado com um grupo de seis desenvolvedores Android que executaram quatro atividades de codificação. Cada atividade representava a instanciação de um módulo do *framework*. Os resultados indicam que o *framework* foi bem aceito. Apesar de distintos perfis de desenvolvedores, nenhum deles teve grandes dificuldades durante as tarefas de instanciação.

Palavras-chave: Computação Ubíqua. Jogos de Cartas Ubíquos. Framework

ABSTRACT

Researches in Ubiquitous Games aim at discovering how computing devices and sensors can be used to promote entertainment. They try to extend game experience beyond the virtual world by extrapolating its boundaries to the player's physical world. These games inherit traditional challenges of both domains (i.e., game development and ubiquitous systems), such as, device heterogeneity, resource discovery, communication, and limitation of available resources. In this master thesis, we propose a model for the development of ubiquitous card games, called Tardigrade. It seeks to minimize the challenges of implementing this kind of ubiquitous game. Tardigrade supports three card types: identifiable cards; interactive cards; and context-aware cards. An Android *framework* that follows Tardigrade model was also implemented. It provides components for card recognition and management; game state management; and device discovery and communication. A proof of concept has been implemented by using the *framework*: the Hyper Try game. This game illustrates how the mobile game *framework* can be instantiated to create ubiquitous card games. Fourteen volunteers evaluated the HyperTry through PLEXQ methodology (Playful Experience Framework Questionnaire). The evaluation results indicate that the game developed is fun and arouses interest, mainly because of its innovative mechanics. Another experiment evaluation was implemented in order to evaluate the *framework's* functionalities. This experiment was conducted with a group of six Android developers who performed four coding activities. Each activity represented the instantiation of a *framework* module. The results indicate that the *framework* was well accepted. Although distinct profiles of developers, none of them had great difficulties during the instantiation tasks.

Keywords: Ubiquitous Computing. Ubiquitous Card Games. Framework.

LISTA DE ILUSTRAÇÕES

Figura 1 – Pokémon Go, jogo que mistura o real e o virtual por meio de realidade aumentada. 25
Figura 2 – Definição de contexto - Reprodução (VIANA, 2010). 28
Figura 3 – Jogo XCOM, um jogo de tabuleiro guiado por uma aplicação disponível para dispositivos móveis e PC. 29
Figura 4 – Jogo móvel locativo LAGARTO Scout e a tela para escolha das missões disponíveis no jogo. 30
Figura 5 – *Framework Bootstrap* para desenvolvimento de interfaces web responsivas em ação em três dispositivos: desktop, tablet e smartphone. 34
Figura 6 – Arquitetura do Whist. 38
Figura 7 – Visão geral do *Augmented Trading Card Games (Augmented TCG).* . 39
Figura 8 – Pirates! o jogo. 40
Figura 9 – Visão do jogo Alquimistas após uma combinação identificada pelo *smartphone*. No topo, as duas cartas reais e na aplicação móvel o resultado de suas combinações. 41
Figura 10 – Jogo Jo-Ken-Po implementado usando o Wanda. 42
Figura 11 – Situação de jogo com jogadores compartilham um único dispositivo. . 48
Figura 12 – Situação de jogo com múltiplos dispositivos com mesmo estado. 49
Figura 13 – Situação de jogo com múltiplos dispositivos conectados. 49
Figura 14 – Diagrama de *Features* do Modelo de Jogos do Tardigrade. 50
Figura 15 – Organização estrutural dos módulos principais do Tardigrade. 54
Figura 16 – Fluxo do Hub para integração de serviços. 55
Figura 17 – Funcionamento do reconhecimento de cartas. 56
Figura 18 – Resumo do diagrama de classe do módulo de cartas. 57
Figura 19 – Módulo de comunicação. 59
Figura 20 – Cartas do jogo Hyper Try. Guardião Vermelho como carta tangível-identificável; Mercenário como carta tangível-interativa; Gaia como carta tangível sensível ao contexto. 63

Figura 21 – Janelas para interação ou captação de contexto. a) Janela de interação para números aleatórios b) janela de interação para detectar o dispositivo sendo sacudido c) janela de interação para detectar luminosidade do ambiente d) janela de detecção de clima. 64
Figura 22 – Telas para escolha de poder de força e tela com poder escolhido. . . . 65
Figura 23 – Carta “Guardião verde” e o fato de multiplicação para cada uma das forças baseado na escolha do adversário. 66
Figura 24 – Base de dados de cartas como arquivo de texto CSV. 67
Figura 25 – Fluxo de troca de informações de estado durante uma partida. 67
Figura 26 – Distribuição das médias obtidas para as categorias de descoberta, exploração, desafio e companheirismo. 69
Figura 27 – Distribuição das médias obtidas para as categorias de controle, competição, sensação e emoção. 70
Figura 28 – Distribuição das médias relaxamento, humor e captação. 71
Figura 29 – Distribuição das médias obtidas para as categorias de crueldade e sofrimento. 71
Figura 30 – Perfil dos voluntários. 74
Figura 31 – Tarefas realizadas no experimento de avaliação. 75
Figura 32 – Questões do formulário de acompanhamento. 77
Figura 33 – Instrução da atividade 2. 77
Figura 34 – Tempo para capacitação dos módulos por voluntário. 78
Figura 35 – Tempo para implementação das atividades por voluntário. 79
Figura 36 – Resultado do questionário pós-experimento. 80

LISTA DE TABELAS

Tabela 1 – Análise comparativa dos jogos relatados 43
Tabela 2 – Características vantajosas dos tipos de jogos 45

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Chamada do Hub para ler uma carta por QR Code 55
Código-fonte 2 – Implementação da carta de baralho Ás de Paus 57
Código-fonte 3 – Implementação do comando de audiodescrição para uma carta 57
Código-fonte 4 – Exemplo de instanciação de um estado 58
Código-fonte 5 – Implementação do evento que recebe as mensagens 60
Código-fonte 6 – Implementação do serviço de descoberta 60
Código-fonte 7 – Implementação da integração entre as classes `Manager` e `State` . 61

LISTA DE ABREVIATURAS E SIGLAS

Augmented TCG	*Augmented Trading Card Games*
UC Games	*Ubiquitous Computing Games*
CAC	Componente de Aquisição de Contexto
LoCCAM	*Loosely Coupled Context Acquisition Middleware*
NFC	*Near Field Comunication*
PLEXQ	*Playful Experience Framework Questionnaire*
RFID	Identificação por Rádio Frequência
SDK	Kit de Desenvolvimento de Software
SUS	*System Usability Scale*
THAR Games	*Tabletop Handheld Augmented Reality Games*
WLAN	Wireless Local Area Network

SUMÁRIO

1 INTRODUÇÃO 18
1.1 Contextualização 18
1.2 Motivação 19
1.3 Objetivo e Contribuições 20
1.4 Metodologia 21
1.5 Organização da Dissertação 22
2 FUNDAMENTAÇÃO TEÓRICA 24
2.1 Computação Ubíqua 24
2.2 Sensibilidade ao Contexto 27
2.3 Jogos Ubíquos 28
2.3.1 Definição de Jogos 28
2.3.2 Definição de Jogos Ubíquos 28
2.3.3 Princípios de design de Jogos Ubíquos 31
2.4 Jogos de Cartas Ubíquos 33
2.5 *Framework* 33
2.6 Conclusão 35
3 TRABALHOS RELACIONADOS 37
3.1 Jogos Ubíquos 37
3.1.1 Whist 37
3.1.2 Augmented TCG 39
3.1.3 Pirates! 39
3.1.4 Alquimistas 40
3.2 Ferramentas para Construção de Jogos Ubíquos 41
3.2.1 Middleware uOS 41
3.2.2 Wanda 42
3.3 Conclusões 42
4 TARDIGRADE 45
4.1 Elicitação dos requisitos 45
4.1.1 Cenário motivador 46
4.1.2 Daylife Ubiquitous Card Game 46

4.2 Modelo . . . 48
4.3 Tipos de Carta Tardigrade . . . 51
4.4 Módulos e estrutura do *Framework* . . . 53
4.5 Aquisição de contexto . . . 53
4.6 Módulos do Tardigrade . . . 54
4.6.1 ***Hub*** . . . 54
4.6.2 ***Módulo de Cartas*** . . . 56
4.6.3 ***Módulo de Estado de Jogo*** . . . 58
4.6.4 ***Módulo de Comunicação*** . . . 59
4.7 Conclusões . . . 61
5 PROVA DE CONCEITO . . . 63
5.1 *Hyper Try* . . . 63
5.1.1 ***Regras*** . . . 64
5.1.2 ***Mecânica do jogo*** . . . 65
5.2 Implementação . . . 66
5.3 Avaliação de *Playful* . . . 68
5.3.1 ***Perfil dos jogadores*** . . . 68
5.3.2 ***Procedimento*** . . . 68
5.3.3 ***Resultados e Discussão*** . . . 69
5.4 Conclusões . . . 71
6 AVALIAÇÃO DO *FRAMEWORK* . . . 73
6.1 Objetivos do Experimento . . . 73
6.2 Perfil dos Usuários . . . 74
6.3 Ameaças a Validade . . . 74
6.4 Procedimento . . . 75
6.5 Materiais e Instrumentos . . . 76
6.6 Resultados e Discussão . . . 78
6.7 Problemas encontrados e limitações da avaliação . . . 81
6.8 Conclusões . . . 81
7 CONCLUSÃO . . . 83
7.1 Resultados Alcançados . . . 83
7.2 Produção Bibliográfica . . . 84

7.3 **Limitações** 85
7.4 **Trabalhos Futuros** 85
REFERÊNCIAS 87
APÊNDICES 90
APÊNDICE A – Documentos de treinamento 90
APÊNDICE B – Questionário Pré Experimento 106
APÊNDICE C – Questionário de Atividade 108
APÊNDICE D – Questionário Pós Experimento 109

1 INTRODUÇÃO

Este trabalho propõe um novo conceito de jogo de cartas pautado em um mecanismo que auxilia sua utilização por meio de dispositivos móveis. Este capítulo contextualiza esta pesquisa na Seção 1.1. A Seção 1.2 apresenta a motivação para o desenvolvimento deste trabalho. Na Seção 1.3 estão listados os objetivos e contribuições deste trabalho. A Seção 1.4 descreve a metodologia utilizada. Por fim, na Seção 1.5 é exposta a organização dos capítulos desta dissertação.

1.1 Contextualização

Jogos digitais destacaram-se nas últimas décadas, em parte, por permear diversas plataformas, desde os já tradicionais consoles de videogame até os dispositivos móveis. Apesar do forte sucesso dos jogos digitais, os jogos analógicos (e.g., jogos de cartas, jogos de tabuleiro) ainda conquistam seguidores, principalmente pelo fator da tangibilidade física que eles proporcionam em sua atividade lúdica e interativa. Recentemente, pesquisas e, inclusive, alguns jogos comerciais (e.g., *Alchemist, Dream Phone*) buscaram formas de realizar a integração entre essas modalidades de jogos (KOTRY, 2014)(GRAY, 2015).

A mescla de interações digitais com elementos tangíveis por meio de tecnologias móveis é a abordagem mais proeminente devido à popularidade dos *smartphones*. Aumentar a experiência de um jogo analógico com aplicativos móveis para reconhecer cartas, apresentar o estado do tabuleiro ou mostrar animações digitais podem ser vistos como exemplos da materialização dos princípios de Computação Ubíqua no domínio dos jogos de cartas. Mark Weiser (WEISER, 1991), que cunhou o termo Computação Ubíqua, descreve tal paradigma como a tecnologia computacional que auxilia as atividades diárias das pessoas e muda completamente a experiência de uso dos computadores em decorrência da naturalidade das interações e do desaparecimento da tecnologia. Jogos ubíquos são, portanto, jogos que transcendem, de forma natural, o mundo virtual, borrando a linha entre o que é real e o que é virtual.

Em 2002, Björk et al. (BJöRK *et al.*, 2002) relatou o crescimento no interesse dos pesquisadores por novas técnicas de design e de uso de Computação Ubíqua para a criação de jogos. Os desafios apontados pelos autores permanecem os mesmos até hoje, como por exemplo, a necessidade de descobrir como dispositivos computacionais comuns e

sensores podem ser utilizados para promover esses jogos ubíquos (CHEUNG *et al.*, 2013). Durante os últimos dez anos, uma série de iniciativas foi proposta, em especial, jogos híbridos que usam mesas interativas para expandir a experiência das mecânicas analógicas (YAN *et al.*, 2011)(ZHOU *et al.*, 2004). Entretanto, poucos são os trabalhos voltados para o design e desenvolvimento de jogos de cartas que mantém a mobilidade inerente a esses jogos.

1.2 Motivação

Os dispositivos móveis apresentam-se como elemento fundamental para promover e expandir a experiência dos jogos de cartas analógicos com o mínimo de interferência na forma como esses jogos são utilizados. Isso decorre do fato de que os *smartphones* são hoje dispositivos onipresentes, de fácil mobilidade e aptos a oferecer uma gama de funcionalidades multimídia (e.g., áudio, imagens, vídeo, realidade aumentada). Essas funcionalidades podem ser aplicadas para expandir a experiência dos jogos de cartas analógicos.

Os jogos de cartas ubíquos herdam, ao mesmo tempo, desafios tradicionais do desenvolvimento de jogos e problemas intrínsecos aos sistemas ubíquos, tais como: a heterogeneidade dos dispositivos, a descoberta de dispositivos, a comunicação volátil e a limitação dos recursos disponíveis.

Além dos desafios herdados da computação ubíqua, os jogos de cartas ubíquos possuem ainda seus próprios desafios, tais como: a tangibilização das cartas (CHEUNG *et al.*, 2013) (i.e., fazer o mapeamento entre as cartas reais e seus espelhos virtuais), a utilização do contexto dos jogadores (e.g., localização, atividade) como componente do jogo e o desenvolvimento de aplicações sensíveis ao contexto que incorporem essas informações. É na procura por uma solução a esses desafios que se pauta a motivação desta pesquisa de mestrado. Existe uma lacuna de bibliotecas de software, de *frameworks* e de plataformas de *middleware* que auxiliem no desenvolvimento de jogos de cartas ubíquos com essas características supracitadas.

1.3 Objetivo e Contribuições

O objetivo principal deste trabalho é **prover um *framework* para auxiliar o desenvolvimento de jogos de cartas ubíquos que minimize o esforço de implementação deste tipo de jogo**.

Além disso, outros objetivos secundários foram elencados, a saber:

- Criar um Modelo para jogos de cartas ubíquos;
- Propor novos tipos cartas que se beneficiem das tecnologias móveis e ubíquas;
- Implementar um conjunto de módulos chaves para jogos de cartas ubíquos;
- Verificar se o *framework* desenvolvido facilita o desenvolvimento de jogos de cartas ubíquo como se propõe;
- Realizar testes para verificar se um jogo de carta ubíquo desenvolvido com *framework* é bem aceito por jogadores.

A principal contribuição deste trabalho de mestrado é um modelo para o desenvolvimento de jogos de cartas ubíquos chamado Tardigrade [1]. Ele busca minimizar os desafios de implementação desse tipo de jogo ubíquo. O Tardigrade abrange três tipos de cartas: carta tangível-identificável; carta tangível-interativa e carta tangível sensível ao contexto.

Um *framework* para plataforma Android foi desenvolvido implementando o modelo Tardigrade e fornecendo componentes para o reconhecimento e gerenciamento de cartas, o gerenciamento do estado de jogo e a descoberta e a comunicação entre dispositivos; e permitindo a captação de contexto para criar jogos sensíveis a ele. A partir do *framework* é possível criar jogos em três diferentes situações: na primeira, um único dispositivo gerencia todo o estado de uma partida do jogo; na segunda, um dispositivo gerencia o estado do jogo, mas esse estado pode ser reproduzível para outros dispositivos; e na terceira, vários dispositivos comunicam-se entre si, gerenciando e trocando informações sobre o estado do jogo.

Uma segunda contribuição desta pesquisa foi o jogo *Hyper Try*. Esse jogo móvel ilustra como o *framework* Tardigarde pode ser instanciado para a criação de jogos de cartas ubíquos. O jogo também funciona como uma prova de conceito dos diferentes tipos de cartas apresentados no modelo criado. O jogo tem como contexto a batalha entre

[1] Tardígrado (do inglês, *Tardigrade*) é um animal microscópico com grande resiliência, sobrevivendo sob condições extremas de calor e frio. Vivendo de forma ubíqua na natureza, eles estão presentes em todos os lugares da terra. Estima-se que existe pelo menos 1 bilhão deles para cada humano vivo.

titãs da natureza que usam os poderes do fogo, água e terra em batalhas. Cada titã é representado por uma carta e possui um poder diferente, sendo alguns deles interativos e sensíveis ao contexto. Os quatorze voluntários que avaliaram o *Hyper Try* indicaram que o jogo desenvolvido é divertido e desperta o interesse, principalmente, devido às suas mecânicas inovadoras.

1.4 Metodologia

Este trabalho foi desenvolvido em seis etapas de um ciclo iterativo. As etapas consistiram de: i) a revisão bibliográfica dos conceitos teóricos e jogos com características ubíquas; ii) aprofundamento dos trabalhos relacionados e de jogos para definição do modelo Tardigrade para jogos de cartas ubíquas; iii) realização de grupos focais para elicitação de requisitos; iv) definição do modelo Tardigrade a partir dos resultados das elicitação dos requisitos; v) a implementação do modelo Tardigrade como um *framework* para plataforma Android, e a instanciação de um jogo como prova de conceito; e por fim, vi) os experimentos de avaliação do jogo desenvolvido e do *framework* Tardigrade. As seis etapas são detalhadas a seguir:

- **Revisão de literatura.** Para este trabalho foi realizada uma revisão bibliográfica reunindo conceitos de computação ubíqua, jogos ubíquos, computação sensível ao contexto e *frameworks*.
- **Estudo aprofundado dos trabalhos relacionados.** O escopo do trabalho foi reduzido a abordagem de jogos de cartas ubíquos, *frameworks* e bibliotecas para desenvolvimento de jogos com características ubíquas. Para esses trabalhos, foi realizada uma revisão de literatura e os trabalhos encontrados foram estudados e comparados. Jogos comerciais também foram estudados para abordar as características e mecânicas apresentadas por eles.
- **Elicitação dos requisitos.** Foi realizado um grupo focal para elicitação dos resultados explorando a expectativa de jogadores de jogos digitais e de jogos de tabuleiro quando questionados sobre jogos com características híbridas (digitais e analógica).
- **Definição do Modelo Tardigrade.** A partir das características dos jogos encontrados e do resultado dos grupos-focais, foi criado um cenário fictício do qual foram extraídas as características desejadas dos jogos de cartas ubíquos. Em seguida, o

modelo do Tardigrade foi definido.

- **Implementação do *framework*.** Esta etapa consistiu em implementar o modelo criado como um *framework* para plataforma móvel Android. Baseado no cenário construído, o *framework* foi desenvolvido em quatro grandes módulos, a saber: o módulo de gerenciamento de cartas; o módulo de reconhecimento de cartas; o módulo de estado de jogo; e o módulo de comunicação. Uma vez com o *framework* implementado, um jogo foi instanciado para que apresentasse as funcionalidades abordadas por ele.
- **Avaliações**. A partir do jogo implementado, foi utilizada a metodologia *Playful Experience Framework Questionnaire* (PLEXQ) (BOBERG *et al.*, 2015) para avaliar a experiência de um jogo instanciado a partir do *framework*. O *framework* também foi avaliado com intuito de verificar se era bem aceito por desenvolvedores e se apresentava dificuldades de utilização.

1.5 Organização da Dissertação

O restante desta dissertação está organizada em sete capítulos. Este, o primeiro, contextualiza o assunto abordado neste trabalho e apresenta a motivação e objetivos da dissertação. Os demais capítulos são:

Capítulo 2 – Fundamentação Teórica: Aborda os conceitos teóricos que compõem esta dissertação. Dentre eles estão: Computação Ubíqua, Computação Sensível ao Contexto e Jogos Ubíquos.

Capítulo 3 – Trabalhos Relacionados: Apresenta jogos com característica ubíquas e ferramentas para o desenvolvimento de jogos ubíquos.

Capítulo 4 – Tardigrade: Apresenta a proposta desta dissertação que inclui um modelo de jogos de cartas ubíquos e um *framework* implementado em Android para criação de jogos de cartas ubíquos.

Capítulo 5 – Prova de Conceito: Este capítulo apresenta a instanciação do *framework* por meio da criação do jogo *Hyper Try* e a avaliação de jogabilidade realizada com 14 voluntários sobre a experiência.

Capítulo 6 – Avaliação do *framework*: Apresenta as avaliações realizadas com o *framework* Tardigrade, a metodologia de aplicação do experimento, bem como os procedimentos, materiais e resultados obtidos.

Capítulo 7 – Conclusão: Descreve os resultados alcançados e as conclusões deste trabalho. Também são apresentadas possíveis melhorias que podem ser realizadas em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais conceitos teóricos relacionados a este trabalho. A Seção 2.1 apresenta o conceito de Computação Ubíqua abrangendo sua definição e utilização em sistemas ubíquos. A Seção 2.2 aborda conceitos de Sensibilidade ao Contexto e como ela pode estar presente nos jogos. A Seção 2.3 apresenta a definição de jogo, as diferentes visões sobre Jogos Ubíquos, suas características fundamentais e os desafios encontrados na implementação desses jogos e princípios de design de jogos ubíquos. Nesse âmbito, os jogos Ubíquos Sensíveis ao Contexto ganham destaque nesta pesquisa. A Seção 2.4 apresenta em específico os Jogos de Cartas Ubíquos, as suas características e seus diferenciais em relação aos jogos analógicos de cartas e jogos eletrônicos de cartas. A Seção 2.5 disserta sobre *frameworks*, explicando seus princípios, vantagens, desvantagens, estruturas, conceitos principais e suas diferentes abordagens de desenvolvimento. Por fim na Seção 2.6 são apresentadas as conclusões sobre os principais conceitos apresentados e como estão relacionados a este trabalho.

2.1 Computação Ubíqua

Mark Weiser (WEISER, 1991), em seu artigo seminal, idealizou a Computação Ubíqua como um novo paradigma no qual dispositivos computacionais incorporados naturalmente ao cotidiano das pessoas de tal forma que estes se tornariam "invisíveis" devido à indissociabilidade entre o digital e o tangível. Nieuwdorp (NIEUWDORP, 2007) descreve que, por desaparecer, Weiser propõe que a computação será tão embutida no cotidiano dos usuários, que suas interfaces naturais e fáceis de usar se tornariam metaforicamente invisíveis para os usuários. Por exemplo, objetos de uso diário (cafeteira, geladeira, TV) comunicariam uns com os outros e seriam capazes de adaptar suas execuções de acordo com os diversos perfis de usuários, locais em que estivessem instalados e mudanças nos seus ambientes de execução.

Spínola (SPÍNOLA; TRAVASSOS, 2012) descreve nove aspectos dos sistemas ubíquos mais abordados em pesquisas acadêmicas e sistemas conhecidos:

- **Serviço onipresente**, que permite que o usuário se mova com a sensação de que o serviço se move com ele;
- **Invisibilidade**, que é a habilidade de estar presente nos objetos do dia a dia de

Figura 1 – Pokémon Go, jogo que mistura o real e o virtual por meio de realidade aumentada.

Fonte: Divulgação/Niantic

forma que o usuário não tenha a sensação explícita de um computador. Assim, é aumentada a percepção de que esses objetos são "inteligentes", tornando a interação entre o homem e a interface desses objetos mais natural.

- **Sensibilidade ao contexto**, que está relacionada à capacidade de um sistema ter ciência do contexto atual do usuário e do sistema e, em consequência, reagir aos eventos alterando sua atuação;
- **Adaptabilidade**, que é a capacidade de um sistema de se adaptar oferecendo serviços de acordo com o ambiente de execução ou ao contexto de utilização no qual o sistema se encontra de forma a respeitar as suas limitações;
- **Captura de experiência**, que indica a capacidade de registrar o que já aconteceu com o usuário e com o sistema no passado para usar em futuras utilizações;
- **Descoberta de serviço**, que é a propriedade do sistema ser capaz de descobrir, de forma autônoma, os serviços, recursos e dispositivos no ambiente em que se encontra, permitindo que o usuário interaja com eles;

- **Interoperabilidade instantânea**, que define que o sistema ubíquo é capaz de mudar os serviços aos quais está associado durante o seu funcionamento enquanto o usuário se move.
- **Heterogeneidade de dispositivos**, que define um desafio que o sistema ubíquo deve lidar ao executar em diversos dispositivos, ajustando-se a cada um em função de suas características de hardware e de software; e
- **Tolerância a falhas**, que representa uma característica desejada ao sistema ubíquo que deve se adaptar sempre que se deparar com uma falha. Como o caso da indisponibilidade temporária ou não dos serviços de comunicação.

De acordo com o autor, mesmo com essa abstração dos sistemas ubíquos, é difícil estabelecer o impacto dessas características em projetos de aplicações ubíquas, sendo que a ubiquidade de um sistema está associada com atender, total ou parcialmente, as essas características.

Em complemento as características dos sistemas ubíquos apresentadas, Santos (SANTOS, 2016) sugere três princípios desejáveis para aplicações ubíquas:

- Elas devem ser *proativas* e monitorar o ambiente de forma constante, predizendo ações do usuário e tomando decisões que melhorem a sua experiência;
- Elas devem ser *adaptativas*, reagindo em diferentes contextos e mudanças que acontecem de forma flexível; e
- Elas devem ser *transparentes* de forma que a atenção do usuário esteja apenas focada em seus objetivos e nas ações que deseja realizar sem que se preocupe com a tecnologia.

Entretanto, desenvolver aplicações reais com características de sistemas ubíquos implica em um conjunto de desafios (COSTA *et al.*, 2008) atrelados às mesmas características apresentadas por Spínola (SPÍNOLA; TRAVASSOS, 2012), tais como resolver os problemas da heterogeneidade de dispositivos e protocolos de comunicação de forma a garantir que informações possam ser trocadas de forma eficiente entre eles. Outro desafio a ser considerado é a invisibilidade do sistema, de forma que o computador desapareça no plano de fundo minimizando a distração do usuário.

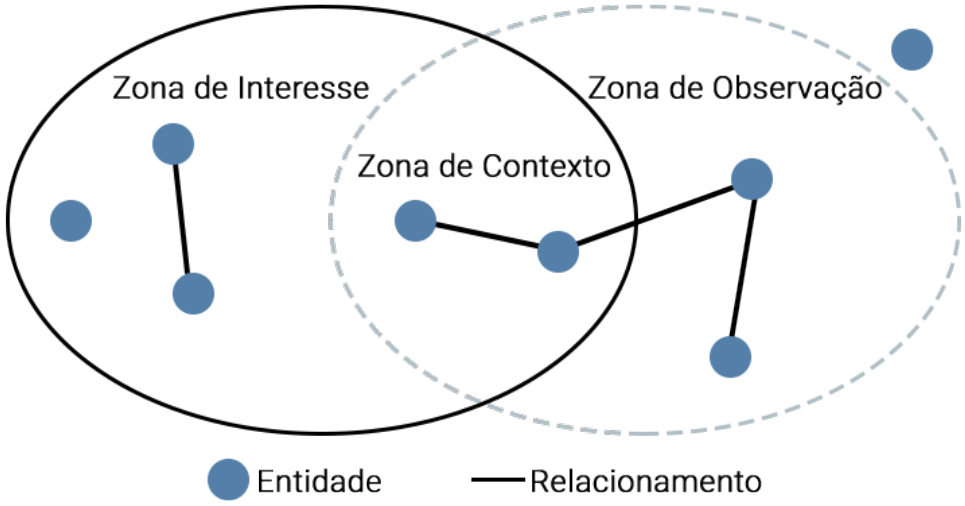
2.2 Sensibilidade ao Contexto

Dentre as diversas características da Computação Ubíqua, encontra-se a Sensibilidade ao Contexto. Dey (ABOWD *et al.*, 1999) define contexto como qualquer informação usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar ou um sistema que seja considerado relevante para a interação entre o usuário e a aplicação, incluindo a interação entre eles. Já um sistema sensível ao contexto, na definição de Dey (ABOWD *et al.*, 1999) é aquele que usa o contexto para prover informações e/ou serviços ao usuário, onde a relevância depende da tarefa realizada. Na definição de Dey (ABOWD *et al.*, 1999), contexto pode ser de quatro tipos: Atividade; Identidade; Localização; e Tempo. Esses contextos podem servir para prover três tipos de funcionalidades em sistemas sensíveis ao contexto: adaptação da apresentação (interface); execução automática e marcação.

Em complemento à definição de Dey (ABOWD *et al.*, 1999), outra definição é a de Viana (VIANA, 2010) que apresenta Contexto como a interseção entre a zona de observação e zona de interesse de um sistema ubíquo (como pode ser visto na Figura 2). A zona de observação é tudo que pode ser observável em um instante t (e.g., sensores disponíveis no *smartphone*, sensores do ambiente) e a zona de interesse é o conjunto de entidades relevantes para o sistema (e.g., expressão facial do jogador, localização). Dessa forma, o contexto é a intersecção entre o que pode ser observado e o que é relevante para o sistema. Esses conjuntos são dinâmicos e evolutivos, uma vez que o sistema pode modificar seu interesse ao longo da execução e a capacidade de observá-lo dependerá de elementos de captura acessíveis em um determinado instante para caracterizá-lo.

Para este trabalho, por exemplo, considera-se que o *smartphone* do jogador dispõe de um sensor de luminosidade na parte superior do dispositivo. Essa informação poderá ser observada pelo sistema e, caso ele tenha interesse, ela fará parte do contexto do jogador. Porém, caso o sensor funcione de forma incorreta ou não esteja disponível, essa informação não fará mais parte do contexto, embora a aplicação/jogo tenha sido projetada para se adaptar a essa informação. Esse detalhe influencia diretamente a forma de se pensar um jogo ubíquo e sensível ao contexto uma vez que seu *gameplay* deve ser suficientemente flexível (tanto em termos de design como em implementação) para suportar a dinamicidade e evolução do contexto.

Figura 2 – Definição de contexto - Reprodução (VIANA, 2010).



Fonte: o autor.

2.3 Jogos Ubíquos

2.3.1 Definição de Jogos

Salen e Zimmerman (SALEN, 2012) definem jogo como “um sistema no qual os jogadores se envolvem em conflito artificial, definido por regra". Sendo esta uma definição genérica, o sistema no qual os autores se referem são peças que se inter-relacionam. E a atividade na qual os jogadores se submetem interage com o tal sistema. Envoltos num conflito, os jogadores têm suas ações limitadas por regras que resultam na experiência de jogo.

2.3.2 Definição de Jogos Ubíquos

Defini como jogos ubíquos, ou *Ubiquitous Computing Games* (*UC Games*), os jogos digitais que se misturam com eventos do mundo real transcendendo a experiência do jogo (SAKAMOTO *et al.*, 2014). Este conceito de jogo ubíquo possui uma contradição entre o objetivo de um jogo, prender a atenção de um jogador, e o objetivo da computação ubíqua, que é tornar as interações mais transparentes. Nieuwdorp (NIEUWDORP, 2007) propõe uma definição mais simples na qual jogos ubíquos são os jogos diretamente associados ao uso da tecnologia ubíqua como recurso no desenrolar do jogo (Figura 3).

Os jogos ubíquos englobam uma variedade de tipos de jogos que podem utilizar diversas dominações, tais como: jogos pervasivos (NIEUWDORP, 2007) jogos móveis

Figura 3 – Jogo XCOM, um jogo de tabuleiro guiado por uma aplicação disponível para dispositivos móveis e PC.

Fonte: http://boardgamegeek.com/boardgame/163602/xcom-board-game/

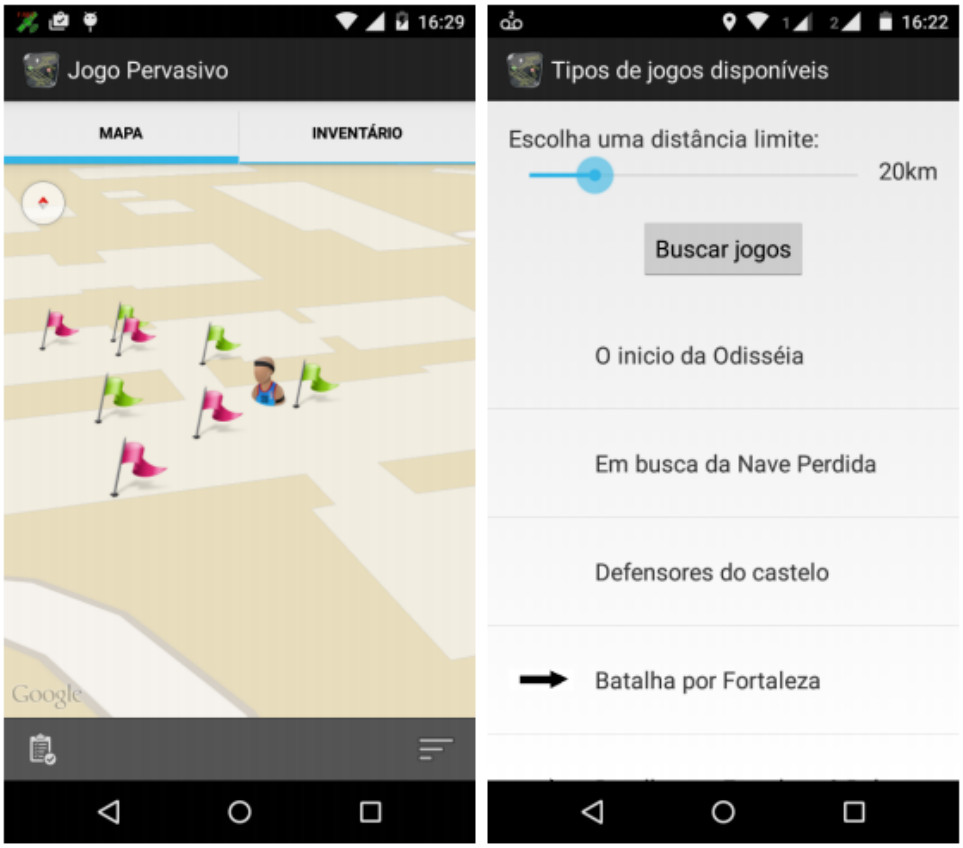
locativos (NOLETO *et al.*, 2015)(ZENDER *et al.*, 2014), jogos de realidade aumentada (SAKAMOTO *et al.*, 2014)(ZENDER *et al.*, 2014), *computer-based board games* (RöMER; DOMNITCHEVA, 2002) e *Tabletop Handheld Augmented Reality Games* (THAR Games) (MAHMUD *et al.*, 2007).

Embora a taxonomia encontrada na literatura seja diferente, as suas características individuais possuem pontos em comum. Como exemplo, tem-se os *Augmented TCG* (SAKAMOTO *et al.*, 2014) que possui as mesmas características dos THAR Games (MAHMUD *et al.*, 2007), que são jogos com cartas que possuem integração com o virtual por meio de realidade aumentada usando monitores ou projeção. Essas variações podem ser compreendidas como subcategorias de jogos pervasivos e/ou ubíquos.

A discussão que tenta separar o conceito de pervasivo e ubíquo já existe há algum tempo. Para Nieuwdorp (NIEUWDORP, 2007) uma vem da influência na tecnologia - computação ubíqua - e a outra que influência na cultura - computação pervasiva. A conclusão, em sua análise no que tange à computação, é que os termos "pervasivo" e "ubíquo" são ambíguos. Já no tocante a jogos, o pervasivo seriam aqueles que dependem primariamente de tecnologia pervasiva já existente, isto é, a cultura de jogos se apropria da tecnologia e a insere como um recurso de jogo. Já os jogos ubíquos seriam aqueles

onde o jogo é como uma funcionalidade da tecnologia sendo ela o foco. Em uma outra visão, jogos pervasivos são descritos como uma camada virtual por cima do mundo real, o que muitas vezes é implementado por meio de tecnologias de Realidade Aumentada. Um exemplo são os Jogos Móveis Locativos, como LAGARTO Scout (NOLETO *et al.*, 2015) (Figura 4), que propõe aos jogadores a exploração do mundo real de forma a experimentar objetos gráficos 3D interativos sobrepostos à sua realidade guiados por mapas e missões.

Figura 4 – Jogo móvel locativo LAGARTO Scout e a tela para escolha das missões disponíveis no jogo.



Fonte: (NOLETO *et al.*, 2015).

]

Nesta pesquisa, considera-se os termos Jogos Pervasivos e Jogos Ubíquos como sinônimos, uma vez que muitos exemplos da literatura extrapolam essas definições propostas. Devido à popularidade do termo "ubíquo" em detrimento de "pervasivo" para definição de sistemas optou-se pelo termo jogo ubíquo devido a maior popularidade do termo. Além disso, o termo jogo ubíquo engloba um conjunto maior de jogos do que o termo jogos pervasivos é usado para indicar. Assim, o termo *ubíquo*, para jogos, é um

adjetivo que permite a interação entre o meio virtual e físico de forma natural e relevante para aquele jogo herdando as características e desafios dos sistemas ubíquos.

Jogos de tabuleiro aumentados (GRAY, 2015)(KOTRY, 2014) com o uso da computação ainda não são tão difundidos devido ao custo de componentes tecnológicos e desenvolvimento de aplicações específicas. Mas essa nova categoria vem ganhando espaço tanto pelo contexto inovador quanto pela interação de componentes físicos e virtuais de forma conjunta. Tanto os jogos de tabuleiro que utilizam de Realidade Aumentada (ZHOU *et al.*, 2004) como os jogos virtuais que usam de componentes tangíveis (MAHMUD *et al.*, 2007)(RöMER; DOMNITCHEVA, 2002) são exemplos de jogos ubíquos que provocam interesse nas pessoas pelo contexto inovador.

2.3.3 Princípios de design de Jogos Ubíquos

Cheung (CHEUNG *et al.*, 2013) propõem um conjunto de princípios de design como propriedades desejáveis em um jogo ubíquo, são eles: dispensabilidade, autoajuste instantâneo, tangibilidade, mobilidade; e valor.

Dispensabilidade - É a capacidade do sistema remover componentes de hardware ou de software que os jogadores julguem não essenciais ao jogo, sendo possível desacoplar esses componentes sem tornar o jogo inviável. O oposto disso é quando o componente não é dispensável e está fortemente ligado ao jogo, como no caso de remover o sistema de pontuação (*score*), que determina o vencedor e o perdedor, tornaria impossível determinar como o jogo terminou.

Autoajuste instantâneo - É a capacidade de realizar mudanças durante o jogo de forma rápida e prática, sem que seja necessário esperar uma nova partida ou parar o jogo. Um exemplo disso seria um temporizador (*timer*) que pode ser reajustado durante o jogo, dando mais tempo aos jogadores, sem que seja necessário pausar ou recomeçar.

Tangibilidade - Essa propriedade refere-se a sistemas que enfatizam o tangível como uma técnica de interação, usando objetos físicos cotidianos como controle ou para apresentar informações digitais.

Mobilidade - É o suporte para "jogar em todos os lugares". São soluções não dependentes dos locais. Como exemplo, pode-se citar aplicações para dispositivos móveis.

Valor - As aplicações de Jogos Ubíquos devem ter o objetivo de aumentar a experiência de jogo. Esse princípio foi incorporado para manter a experiência do jogo

como um fator importante nesses sistemas híbridos.

A visão de Cheung (CHEUNG *et al.*, 2013) pode ser relacionada às características dos sistemas ubíquos de Spínola (SPÍNOLA; TRAVASSOS, 2012). Por exemplo, a característica *Sensibilidade ao Contexto* de Spínola pode estar presente em alguns dos princípios de Cheung. Por exemplo, no *Auto ajuste instantâneo* com a detecção do contexto para ajustes em tempo real, como posição do dispositivo ou sensores disponíveis no momento. Também podemos observar a relação dos conceitos na *Tangibilização* de informações físicas, sejam estes objetos - como cartas - ou dados do ambiente - como o clima e iluminação e no *Valor*, onde o contexto que altera a experiência de jogo pode ser única e tornar cada partida também única.

Relacionando as nove propriedades de Spínola (SPÍNOLA; TRAVASSOS, 2012) e os 5 princípios de Cheung (CHEUNG *et al.*, 2013), a única característica que não é apontada nos princípios de design de Cheung é a *Descoberta de Serviço*. Tal característica diz respeito ao sistema ubíquo oferecer, de forma autônoma, serviços de que os usuários possam utilizar e interagir. Um exemplo dessa característica são os Jogos Ubíquos Multijogadores (*multiplayer*), no qual o jogo pode encontrar jogadores no ambiente de forma autônoma, onde ocorre a descoberta do "Serviço" de jogo em outros dispositivos. Embora Cheung não tenha abordado essa característica, ela pode fornecer uma gama de possibilidades aos jogos ubíquos, como a interação entre jogadores que ainda não se conhecem (o jogo pode descobrir que existe um outro jogador próximo ao usuário).

Algumas dessas características podem estar mais presentes que outras em um jogo ubíquo, por exemplo: um jogo pode fazer a utilização de contexto para autoajuste instantâneo da interface do jogo, enquanto outro pode ignorar completamente o contexto e trabalhar o aspecto tangibilidade. Os princípios de design de Cheung (CHEUNG *et al.*, 2013) não são um modelo que precisa ser seguido rigidamente, assim como as características de Spínola (SPÍNOLA; TRAVASSOS, 2012). Entretanto, segui-los garante que parte das características dos jogos ubíquos estarão presentes de forma harmônica.

Para este trabalho de mestrado, são utilizadas as propriedades propostas por Cheung (CHEUNG *et al.*, 2013) para o design de jogos ubíquos. As características propostas por Spínola (SPÍNOLA; TRAVASSOS, 2012) complementam esses princípios de Cheung para definir quão ubíquo pode ser um jogo criado com o Tardigrade.

2.4 Jogos de Cartas Ubíquos

O foco deste trabalho são os jogos ubíquos cujo componentes elementares do jogo sejam cartas. Chamado, então, de Jogos de Carta Ubíquos, esses jogos utilizam das propriedades dos sistemas ubíquos como mecânica ou recurso de jogo interagindo diretamente com as cartas disponíveis.

Desta forma, a tangibilização, uma das propriedades desejáveis nos jogos ubíquos apresentadas por Cheung (CHEUNG *et al.*, 2013), passa a ser mandatória nesse tipo de jogo. Isto é, o mínimo de interação desejada em um jogo de cartas ubíquo é o reconhecimento pelo meio digital das cartas reais.

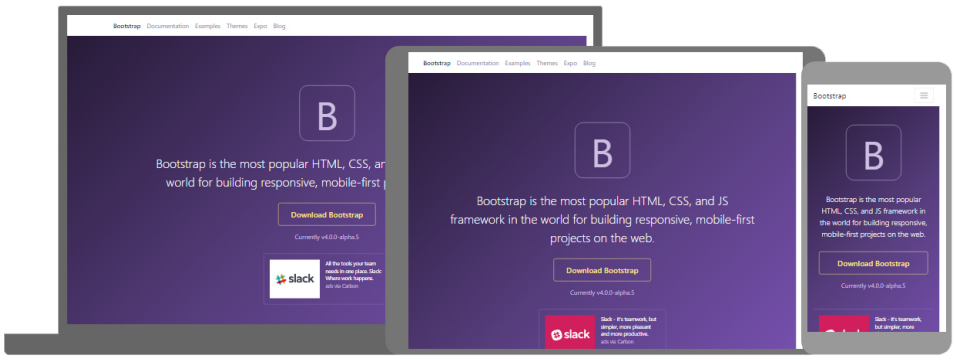
A sensibilidade ao contexto é uma outra característica interessante que pode estar presente nesses jogos. Fatores como o clima, a localização, sensores lógicos e físicos dos dispositivos móveis, e até informações digitais disponíveis na web (e.g., índices da bolsa de valores, ranking de palavras mais buscadas e *hashtags* mais utilizadas nas últimas 24h) podem ser utilizadas como variáveis relevantes no jogo tornando cada partida única e proporcionando um fator de re-jogabilidade incontável.

2.5 *Framework*

Uma definição geral de *framework* é apresentada por Johnson(JOHNSON; FOOTE, 1988) como sendo "um projeto abstrato para um tipo de aplicação". *Frameworks* podem ser descritos ainda como um conjunto de objetos reutilizáveis englobando diversas áreas aplicadas a um domínio específico. Um *framework* também pode ser definido como "um projeto de reutilização que descreve como o sistema está decomposto em um conjunto de objetos que interagem entre si, podendo ser uma aplicação inteira ou apenas um subsistema" (FILHO, 2004).

O objetivo de um *framework* é facilitar para o desenvolvedor a construção de aplicações com a adição da menor quantidade de código escrito possível e oferecer soluções reutilizáveis dentro de um mesmo domínio de aplicação. A Figura 5 ilustra o uso do *framework Bootstrap* em interfaces web responsivas construidas a partir de elementos reutilizáveis. *Framework* também são construídas para permitir a adição de novas funcionalidades. Taligent Inc. (INC., 1993) classifica *frameworks* de acordo com o domínio que ela abrange: *frameworks* de aplicação, *frameworks* de domínio e *frameworks*

Figura 5 – *Framework Bootstrap* para desenvolvimento de interfaces web responsivas em ação em três dispositivos: desktop, tablet e smartphone.



Fonte: https://v4-alpha.getbootstrap.com.

de suporte.

- *Frameworks de aplicação* encapsulam uma variedade de conhecimento aplicáveis a uma grande variedade de programas. Um exemplo são as interfaces gráficas (GUI) para aplicações comerciais;
- *Frameworks de domínio* encapsulam um conjunto de soluções para um problema em particular, como os *frameworks* para desenvolvimento de jogos de computador; e
- *Frameworks de suporte* proveem serviço de sistema, como acesso à arquivos, suporte à computação distribuída ou *driver* de dispositivos.

De uma forma geral, o processo de desenvolvimento de um *framework* é realizado em três fases (MAGALHãES, 2005). A fase de **Análise do domínio** consiste em identificar e organizar o conhecimento sobre o domínio proposto para o *framework*. O objetivo dessa fase é capturar os requisitos e encontrar pontos comuns para compor o *framework*. A fase seguinte é o **projeto do *framework***, que consiste em criar as abstrações do *framework* e identificar os relacionamentos entre as classes do framework. A última fase é a **instanciação do *framework***, onde o desenvolvedor constrói classes a partir das interfaces disponíveis ou escolhe classes fornecidas na estrutura para criar aplicações.

Comumente, como *frameworks* são focados na reutilização de boas soluções, o conceito de padrões de software são utilizados durante a fase de projeto de um *framework*. Segundo Coplien (COPLIEN; ALEXANDER, 1996), um padrão é um pedaço de literatura que descreve um problema de projeto e uma solução geral para este problema num determinado contexto. Dito isso, um padrão é uma entidade que documenta um problema, uma solução e um contexto em que se deve aplicar tal solução. Utilizados em diferentes

estágios do desenvolvimento de um software, os padrões são classificados de acordo com a fase que são aplicados (e.g., padrões de análise, padrões de projeto) (SANTOS, 2004).

Neste trabalho, considera-se um *framework* como um arcabouço de componentes reutilizáveis de arquiteturas orientadas a objetos, sendo que de um *framework* podem ser especializados parte ou total de uma aplicação. Dessa forma, o *framework* determina a arquitetura da aplicação e predefine padrões no projeto, permitindo ao desenvolvedor se concentrar nos detalhes específicos da aplicação. Foram também utilizados diversos padrões de projetos na construção do *framework* proposto. A utilização de padrões, além de facilitar a construção de *frameworks*, são adequados para documenta-los, já que o uso de padrões permite o entendimento do *framework* sem a necessidade de conhecer seu código fonte.

2.6 Conclusão

Neste capítulo, foram apresentados os principais conceitos relacionados a jogos ubíquos, suas características e seus principais desafios.

A respeito de Computação Ubíqua, foram abordadas suas características e quais os aspectos encontrados nos sistemas ubíquos. Também foram elencados três princípios desejáveis em aplicações ubíquas: proatividade, adaptatividade e transparência. Entretanto, existem desafios no desenvolvimento dessas aplicações como a heterogeneidade de dispositivos. Sobre a sensibilidade ao contexto, uma das mais importantes propriedades dos sistemas ubíquos, é apresentada sua definição e sua relevância para construir sistemas que se adaptam ou se modificam de acordo com o contexto de entidades, do dispositivo e do usuário.

No tocante à definição dos jogos, foi apresentada uma visão sobre os jogos ubíquos e como eles são definidos a partir da presença da característica dos sistemas ubíquos com o objetivo de borrar a linha entre a realidade e a virtualidade aumentando a imersão do jogador. Também foi apresentado o conceito de jogos sensíveis ao contexto, que reagem às alterações e informações de contexto do usuário, do ambiente e do dispositivo. Além disso, foi abordado o conceito de jogos de cartas ubíquos e como este está relacionado aos outros conceitos de jogos e ubiquidade apresentados.

Por fim, foram descritos os conceitos e as características presentes nos *frameworks* e alguns dos seus benefícios para o desenvolvimento de software.

A proposta de um *framework* neste trabalho tem como objetivo integrar os principais conceitos apresentados nessa fundamentação teórica em um arcabouço de recursos reutilizáveis com foco na construção de jogos de cartas ubíquos.

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os principais trabalhos encontrados sobre Jogos ubíquos ou jogos com características ubíquas, destacando aqueles com componentes de cartas como parte do jogo.

A Seção 3.1 apresenta quatro jogos que podem ser classificados como ubíquos de acordo com as características apresentadas na Seção 2.3 desta dissertação. São apresentadas as características desses jogos e os aspectos positivos e negativos encontrados. A Seção 3.2 apresenta um *framework* e um *Middleware* desenvolvidos para dar suporte ao desenvolvimento de jogos ubíquos e quais são suas especificidades, suas vantagens e desvantagens. Por fim, na Seção 3.3 é realizado uma síntese comparando os trabalhos a proposta desta dissertação.

3.1 Jogos Ubíquos

Diversos trabalhos relatam aplicações e jogos com características ubíquas (RöMER; DOMNITCHEVA, 2002)(SAKAMOTO *et al.*, 2014)(CHEUNG *et al.*, 2013)(DRUMOND *et al.*, 2014). Os diferentes trabalhos apontados nesta dissertação apresentam os desafios encontrados ao aplicar elementos da computação ubíqua em jogos. A construção de um ambiente de jogo e a integração de diferentes dispositivos são dois dos maiores desafios abordados na construção desse tipo de jogo.

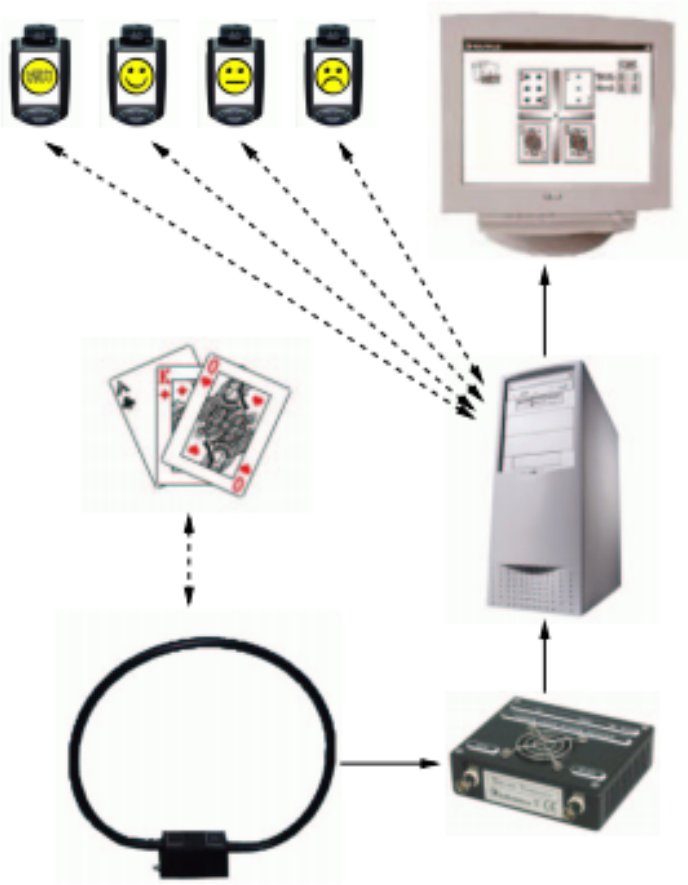
Como apresentado no capítulo 2, existem outras classificações que podem ser entendidas como jogos ubíquos. Santos (SANTOS, 2016) entende como jogo ubíquo todo jogo que "aplica os princípios da computação ubíqua para criar novos tipos de mecânica e modelos de interação". Os trabalhos apresentados a seguir foram selecionados por apresentarem o uso desses princípios da computação ubíqua, como sensibilidade ao contexto e por serem jogos de cartas.

3.1.1 Whist

O jogo *Whist*(RöMER; DOMNITCHEVA, 2002) é um jogo de cartas de baralho onde os jogadores devem seguir um padrão de número ou simbolo das cartas. A proposta em Whist é acompanhar a performance dos jogadores durante a partida por meio de dispositivos móveis, mostrando estatísticas sobre as cartas já jogadas. A estrutura do jogo

consiste em um baralho com etiquetas de Identificação por Rádio Frequência (RFID), uma antena para detecção das cartas e um computador que recebe e processa as informações do jogo, e distribui para o PDA[1] de cada jogador (Figura 6).

Figura 6 – Arquitetura do Whist.



Fonte: (RöMER; DOMNITCHEVA, 2002)

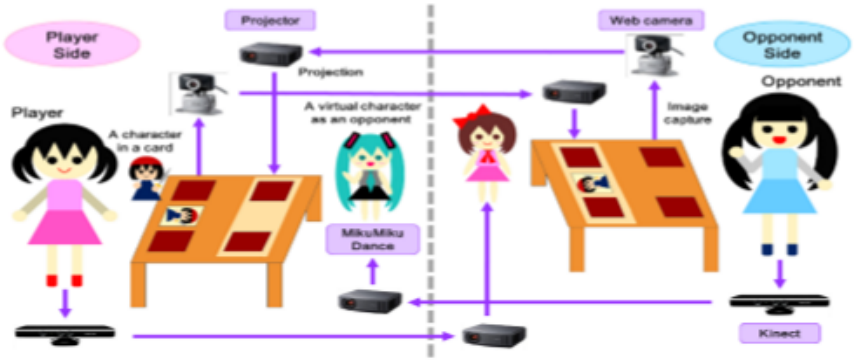
O reconhecimento das cartas de forma automática com uma antena trouxe ao jogo limitações da tecnologia como um dos seus problemas. A sobreposição de etiquetas RFID também fez com que o jogo não reconhecesse algumas cartas. A detecção acidental de cartas que ainda não tinham sido jogadas, mas entraram no campo de detecção de carta, foi outro inconveniente relatado no trabalho. Outro problema do ambiente desse jogo é a difícil reprodução em outros contextos, por o mesmo exige dispositivos embarcados e de baixa mobilidade.

[1] Assistente Pessoal Digital antecessor aos *smartphones*

3.1.2 Augmented TCG

O trabalho *Augmented TCG* (SAKAMOTO *et al.*, 2014) consiste em uma plataforma onde dois jogadores geograficamente separados disputem uma partida do jogo de cartas Yu-Gi-Oh!. O resultado é um ambiente dedicado à aplicação envolvendo câmeras e projeção, além de um monitor que apresenta as reações do adversário, que são captadas pela câmera e representadas por um personagem (Figura 7).

Figura 7 – Visão geral do *Augmented TCG*.



Fonte: (SAKAMOTO *et al.*, 2014)

O foco do estudo é a utilização de personagens virtuais que reproduzam as reações dos jogadores em um jogo entre dois jogadores distantes geograficamente um do outro, permitindo assim uma experiência de interação social semelhante a que vivenciaria em uma partida com os dois jogadores no mesmo local. Embora a percepção do ambiente de jogo e dos jogadores, os equipamentos utilizados tornam o trabalho difícil de ser replicado, além de não serem de uso tão comum (kinect, câmera, projetor e monitor), dificultando a distribuição da aplicação. O jogo também não tangibiliza as cartas de fato e nem utiliza elas de forma digital, sendo ela uma proposta inteiramente social.

3.1.3 Pirates!

O jogo *Pirates!* (CHEUNG *et al.*, 2013) tem como objetivo os jogadores acertarem o próximo digito correto do número PI. Um celular com um sensor *Near Field Comunication* (NFC) identifica a carta utilizada pelos jogadores e apresenta o resultado. A aplicação permite aos jogadores receberem dicas e marca a pontuação de cada um dos jogadores (Figura 8).

Figura 8 – Pirates! o jogo.

Fonte: (CHEUNG *et al.*, 2013)

Diferente dos demais trabalhos encontrados, *Pirates!* segue os princípios de design propostos pelo autor para desenvolvimento de jogos de cartas ubíquos (CHEUNG *et al.*, 2013).

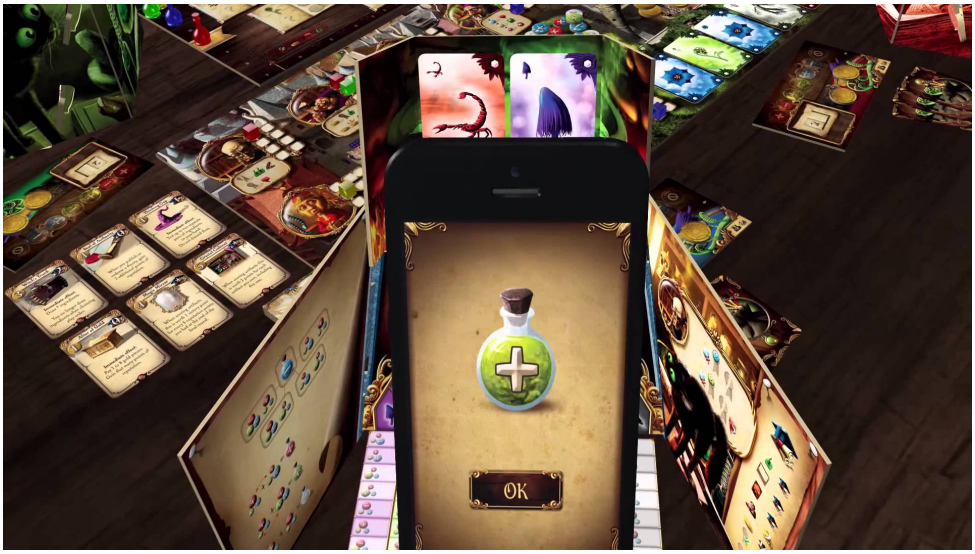
3.1.4 Alquimistas

O jogo Alquimistas[2] é um exemplo de jogo de tabuleiro ubíquo com múltiplas mecânicas sendo uma delas a de *Card Drafting*, onde os jogadores escolhem cartas dentro de um conjunto de cartas limitado de uso comum entre todos. O jogo consiste em criar combinações usando as 8 diferentes cartas do jogo para obter informações e assim conquistar pontos de vitória no jogo. O estado do jogo é representado por uma matriz de combinações das cartas e uma aplicação da acesso às ações principais do jogo, que consiste em reconhecer o padrão das cartas e apresentar o resultado da combinação (Figura 9) que um dos jogadores teria que fazer manualmente.

Esse é um bom exemplo de jogo que faz o uso da tecnologia como aporte aos jogadores permitindo que todos eles possam participar ativamente do jogo sem que seja necessário um deles ser o que o jogo chama de Mestre do Jogo. Um jogador que dá o resultado das combinações baseado em um dos tabuleiro do jogo. O aplicativo também elimina as chances de equívocos do Mestre do Jogo na resolução das combinações.

[2] Alquimistas - http://devir.com.br/alquimistas/

Figura 9 – Visão do jogo Alquimistas após uma combinação identificada pelo *smartphone*. No topo, as duas cartas reais e na aplicação móvel o resultado de suas combinações.



Fonte: Google/Imagens

3.2 Ferramentas para Construção de Jogos Ubíquos

O mesmo fator de inovação presente nos considerados jogos ubíquos encontrados em pesquisas e relatos é o que motiva pesquisadores a desenvolverem mecanismos que tornem esse processo de concepção de jogos ubíquos mais rápido e de maior qualidade. Esses mecanismos geralmente focam em um pequeno domínio de jogos com o objetivo de explorar e apresentar alguns dos conceitos dos sistemas ubíquos.

3.2.1 Middleware uOS

O *middleware* uOS (BUZETO, 2015)(SANTOS, 2016) é um exemplo de especialidade nos trabalhos desenvolvidos, com o foco na comunicação entre dispositivos ele segue um conjunto de conceitos e requisitos relacionados à criação de ambientes inteligentes. A solução abrange os principais desafios envolvidos na construção de jogos ubíquos: heterogeneidade, mobilidade, integração instantânea e sensibilidade ao contexto.

O motor *uImpala* (BUZETO, 2015) é uma solução que usa o *middleware* uOS como apoio para desenvolvimento de jogos ubíquos. O motor mantém uma arquitetura de software familiar aos desenvolvedores de jogos digitais e permite que a comunicação entre diferentes dispositivos seja tratada de forma simplificada e unificada, que simplifica

o esforço do desenvolvimento de novos títulos.

3.2.2 Wanda

Outra proposta também especializada, o Wanda (DRUMOND *et al.*, 2014) é descrito como "um simples *framework* criado para construir especificamente jogos baseado em cartas que podem ser utilizados como ferramentas educacionais". A proposta consiste em fornecer um arcabouço de código reutilizável para instrutores e professores especificarem novos jogos de cartas com cartas digitais. O objetivo do jogo consiste na implementação de dois agentes com estratégias de jogo (por dois grupos de alunos diferentes, por exemplo). Depois, os agentes disputaram a partida do jogo seguindo a estratégia de jogo implementada neles.

Figura 10 – Jogo Jo-Ken-Po implementado usando o Wanda.

Fonte: (DRUMOND *et al.*, 2014)

3.3 Conclusões

Existem diferenças na taxonomia dos jogos que se enquadram na definição de jogos ubíquos e jogos de cartas ubíquos e os trabalhos relacionados apresentam características positivas e negativas para a abordagem dos jogos dessa categoria. A Tabela 1 apresenta as principais propriedades dos trabalhos aqui relatados.

Entre as vantagens encontradas nos trabalhos relatados estão o uso de tecnologia

Tabela 1 – Análise comparativa dos jogos relatados

Jogo	Taxonomia	Vantagens	Desvantagens
Whist	*Computer Based Boardgame*	- Uso de RFID para tangibilização das cartas de forma transparente - Sistema distribuído de estado de jogo	- Não possui mobilidade - Área de reconhecimento da cartas é abstrata - Inviável comercialmente
Yu-Gi-Oh! Aumentado	*Augmented TCG*	—	- Não possui mobilidade - Equipamentos de alto custo - Inviável comercialmente
Pirates!	Jogo Ubíquo	- Uso de NFC para tangibilização das cartas de forma calma - Projetado sobre princípios de jogos ubíquos	- Limitação tecnológica (NFC) no baralho - Custo tecnológico alto (NFC) inviabiliza comercialmente
Alchemists	Jogo de Tabuleiro	- Reconhecimento de cartas por padrão - Custo de produção não depende de tecnologia (e.g., NFC ou RFID) - Estado do jogo reproduzível - Aplicação aumenta a experiência do jogo	—

como forma de tornar o reconhecimento transparente[3] (RFID) ou calmo[4] (NFC).

O acoplamento das mecânicas de jogo com a tecnologia também é um fator a ser considerado. O uso de tecnologia de reconhecimento a partir de padrões utiliza a câmera do dispositivo, que é mais comum que a presença de tecnologia NFC nos dispositivos. Além disso, usar o reconhecimento por padrões torna o custo do jogo menor, uma vez que essas cartas não dependem de um recurso tecnológico que pode ter um custo alto e que também limita o jogo à dispositivos com tecnologia compatível.

Os jogos categorizados como "sem mobilidade" são aquelas que têm como mandatório o uso de tecnologias embarcadas no ambiente de jogo, tais como projetores, computadores e antenas. Isso prejudica não só a reprodução do jogo como a distribuição do mesmo.

Das ferramentas de desenvolvimento de jogos ubíquos relacionadas a este trabalho foi apresentado o *uOS* com foco em comunicação e um motor que apoia o uso desse *middleware* em um ambiente simplificado familiar aos desenvolvedores de jogos. Outra ferramenta apresentada tem foco especializado na construção de jogos de cartas ubíquos porem apenas cartas digitais.

Neste trabalho, o fator mobilidade[5] será um dos pontos fortes. O objetivo é fornecer um recurso de desenvolvimento de jogos que possam ser distribuídos de forma livre sem necessidade de preparação de um cenário de jogo com dispositivos embarcados.

[3] Transparente, pois ele é feito automático quando a carta entra no campo de reconhecimento de jogo.
[4] Calmo porque o jogador precisa apenas aproxima a carta do dispositivo, sem necessidade de sair da aplicação.
[5] Jogar em qualquer lugar a qualquer momento.

Diferente do *framework* Wanda que implementa apenas cartas digitais, a proposta desta dissertação visa cartas que misturam o digital e o tangível. Além do processo de comunicação entre dispositivos, o *framework* apresentado neste trabalho também abrange o reconhecimento das cartas, o gerenciamento delas e do estado do jogo. O *framework* apresentado nessa dissertação também deve fornecer janelas de abertura para implementação de outros recursos como a sensibilidade ao contexto.

4 TARDIGRADE

O elemento central proposto nesta dissertação é o Tardigrade, um *framework* para desenvolvimento de Jogos de Cartas Ubíquos para plataforma Android. O *framework* oferece os componentes básicos para o desenvolvimento de jogos de cartas que mesclam componentes reais e virtuais. O seu propósito é estimular o desenvolvimento de jogos com características ubíquas a partir de um modelo pré-estabelecido de jogo.

Este capítulo apresenta na Seção 4.1 como foi o processo de elicitação dos requisitos e a apresentação de um cenário motivador criado. A Seção 4.2 apresenta o modelo de jogo de carta ubíquo criado, na Seção 4.3 diferentes situações de jogo e os conceitos de cartas criados. A Seção 4.4 apresenta o *framework* implementado a partir do modelo criado. A Seção 4.5 apresenta como o *framework* pode fazer uso de contexto. A Seção 4.6 apresenta os cinco módulos do *framework* implementado: o Hub; o Módulo de Cartas; o Módulo de Estado de Jogo; e o Módulo de Comunicação. Por fim, a Seção 4.7 as conclusões a respeito da proposta desenvolvida apresentada neste capítulo.

4.1 Elicitação dos requisitos

Com objetivo de elicitar requisitos para o framework, foi realizado um grupo focal para explorar as expectativas de jogadores de jogos digitais e analógicos quando apresentados à proposta de jogos de tabuleiro baseado em computação.

O grupo focal composto por 9 participantes onde metade do grupo se auto declarou jogador assíduo de jogos de tabuleiro, e outra metade de jogos digitais. Apenas um dos nove não se declarou como desenvolvedor de jogos digitais.

A principal questão levantada durante o grupo foi *quais elementos dos jogos analógicos eles gostariam de encontrar nos jogos eletrônicos ou vice-versa.* A Tabela 2 seguinte apresenta essas características.

Tabela 2 – Características vantajosas dos tipos de jogos

Jogos digitais	**Jogos analógicos**
Facilidade com cálculos	O contato com o tangível
Setup automático	A socialização física
	Regras flexíveis

Foram consideradas que ambos tipos de jogos possuem suas vantagens e desvan-

tagens. Os jogos digitais poupam o jogador de efetuar cálculos e do processo de montagem do tabuleiro e preparação do jogo. Em contra partida, os jogadores relataram que o contato com peças tangíveis é um ponto que atrai as pessoas que jogam os jogos de tabuleiro, assim como a flexibilização das regras do jogo. Além disso, a socialização física e presencial é um fator importante nesse tipo de jogo.

Dentre os resultado obtidos, elencou-se que a adaptação das mecânicas já existentes em jogos de cartas virtuais ou jogos de cartas analógicas não seria a técnica mais adequada para jogos híbridos entre o virtual e o real pois isso poderia descaracterizar ou complicar as mecânicas já utilizadas. Sendo a solução a criação de novas mecânicas híbridas entre os meios físico e o digital.

A partir disso, foram elencadas as possibilidades que utilizam desde os sensores de dispositivos eletrônicos, como a transformação do tangível em digital ou vice-versa criando um cenário motivador ideal.

4.1.1 Cenário motivador

Para ilustrar o potencial de jogos que podem ser construídos com Tardigrade, foi criado um cenário motivador apresentado a seguir. Como visto na Seção 2.3, jogos de cartas que se qualificam como ubíquos podem possuir as mesmas propriedades dos sistemas ubíquos, tais como, serviço onipresente, invisibilidade, sensibilidade ao contexto, adaptabilidade, captura de experiência de uso, descoberta de serviço, interoperabilidade instantânea, heterogeneidade e tolerância a falhas. Dessa forma, o cenário apresentado descreve um possível jogo que possui algumas dessas propriedades dos sistemas ubíquos.

4.1.2 Daylife Ubiquitous Card Game

Num sábado de sol, com seu baralho de cartas colecionáveis no bolso, David caminha no shopping fazendo compras numa loja de departamento em Fortaleza. Seu iPhone vibra anunciando que existe um oponente nas proximidades dele. Imediatamente, David escolhe chamar o oponente para um combate. Jack também foi notificado de que alguém está nas proximidades e tem seu *smartphone* Android em mãos quando a solicitação de combate chega, aceitando-a imediatamente. Agora, os *smartphones* de David e Jack os guiam para um ponto comum por meio de navegação GPS, iniciando o combate automaticamente no momento em que os dois oponentes se reconhecem encostando um

celular no outro.

O combate entre David e Jack vai além das telas dos seus dispositivos. Numa experiência mista de criaturas virtuais visíveis pelas pequenas telas dos celulares, e criaturas e ataques só vistos na imaginação dos jogadores. Essa imaginação é despertada pelas ilustrações do baralho de cartas colecionáveis que agora cada um utiliza na esperança de derrotar seu adversário.

David começa o combate juntando uma combinação de cartas na parte de trás do seu aparelho, fazendo Jack contra-atacar colocando uma carta atrás e começando a balançar seu celular com afinco. David na verdade utilizou uma série de cartas de ataque, e Jack utilizou uma carta de defesa que tem seu resultado aprimorado com o movimento de chacoalhar o aparelho. Os ruídos do shopping permitiram Jack contra-atacar utilizando uma das suas cartas que aumenta o ataque do jogador de acordo com o ruído de áudio do local. David defende-se protegendo o seu *smartphone* cobrindo os sensores de proximidade e iluminação.

No meio da batalha, Jack recebe uma ligação que precisa atender, mas seu relógio inteligente continua mostrando o progresso do combate e permite a ele escolher sua próxima ação. Uma carta é colocada por cima do seu relógio ativando um ataque mais forte que irá ser potencializado pelo fato de ser um dia ensolarado fora do shopping. David nota que a partida está perdida, porém decide continuar defendendo-se do combate cobrindo os sensores de iluminação e proximidade com o intuito de impedir os danos do próximo ataque.

Jack termina a ligação no mesmo momento em que o combate é finalizado, deparando-se com seu celular informando que o combate foi vencido. Como recompensa, David deveria entregar uma de suas cartas usadas em combate a Jack. David entrega a carta que já não o pertence mais e parabeniza Jack pela vitória, mas deixando claro que voltará para reconquistar a carta perdida. Durante a batalha, informações de contato dos jogadores foram trocadas, permitindo com que David envie mensagens para Jack e, assim, possa ter a chance de uma revanche.

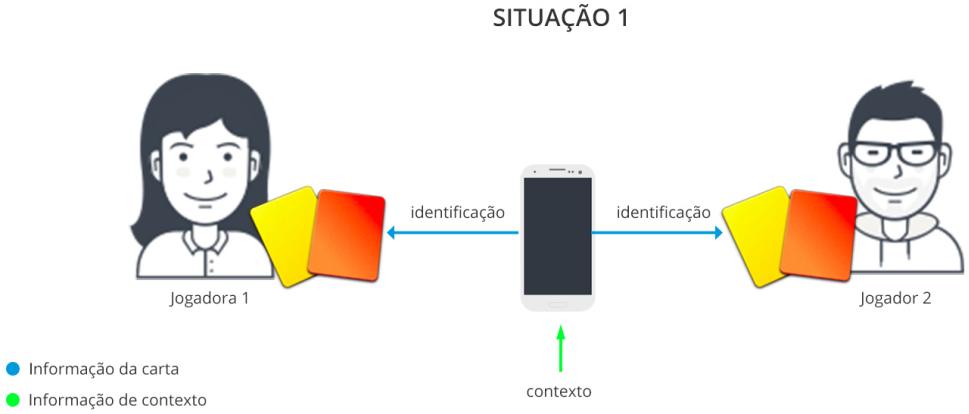
O cenário motivador apresenta mecânicas de combate baseadas em cartas tangíveis e um aplicativo de combates virtuais. Cartas são reconhecidas multi ou unicamente por proximidade pelos dispositivos, que reagem de acordo com os efeitos de cada carta. Algumas cartas necessitam de informações contextuais (i.e., que descrevem a situação do

jogador) para serem bem-sucedidas, como aumentar o dano baseado no clima do local ou no nível de ruído onde o combate está sendo realizado. Já outras cartas utilizam sensores, como o de proximidade, iluminação e acelerômetro, para terem seus efeitos maximizados ou minimizados. E existem ainda as cartas que são apenas identificadas pela aplicação para que seus efeitos sejam aplicados.

4.2 Modelo

O modelo base do Tardigrade foi projetado para três possíveis situações de jogo. Para a primeira situação (Figura 11), tem-se um único dispositivo móvel sendo compartilhado por todos os jogadores. Uma variação dessa situação seria mais de um dispositivo com uma aplicação que permite acessar um mesmo estado de jogo a partir de alguma referência, como ilustrado na Figura 12.

Figura 11 – Situação de jogo com jogadores compartilham um único dispositivo.

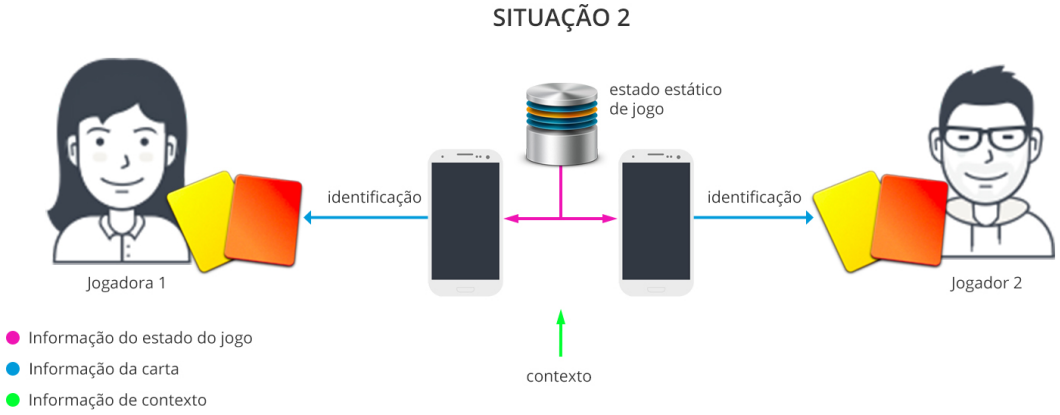


Fonte: o autor.

Na primeira situação, o jogo pode processar dados de entrada para chegar em um final inesperado. Na segunda, os jogadores compartilham de um mesmo conjunto de constantes que representam o estado do jogo, limitando as possibilidades de resultado final.

Ampliando o conceito de compartilhar da segunda situação, tem-se uma terceira visão de jogos suportados pelo modelo. A situação que está ilustrada na Figura 13 é composta por múltiplos dispositivos conectados em rede (e.g., *WiFi*, *Bluetooth*) de forma que seja possível comunicação entre os pares. Nessa situação os dispositivos

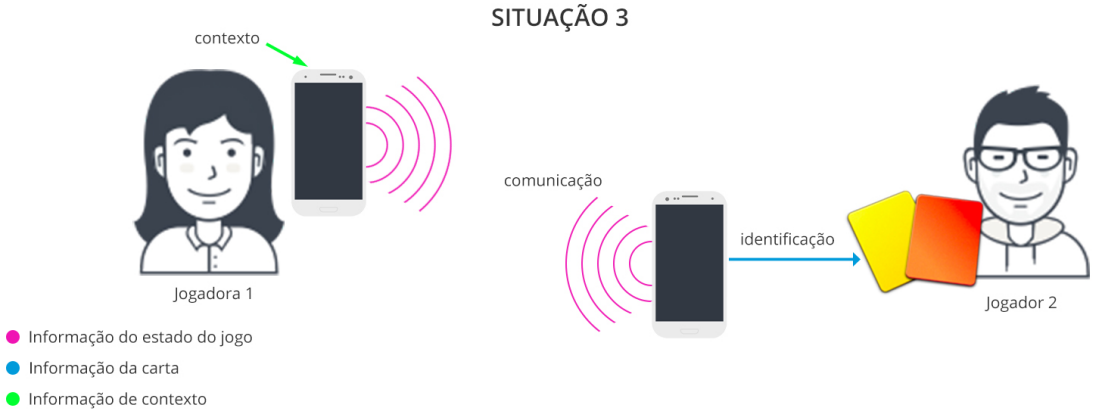
Figura 12 – Situação de jogo com múltiplos dispositivos com mesmo estado.



Fonte: o autor.

podem compartilhar e trocar o estado de jogo e compartilhar eventos detectados em cada dispositivo de forma individual. Ou seja, cada um tem uma visão distinta do jogo.

Figura 13 – Situação de jogo com múltiplos dispositivos conectados.



Fonte: o autor.

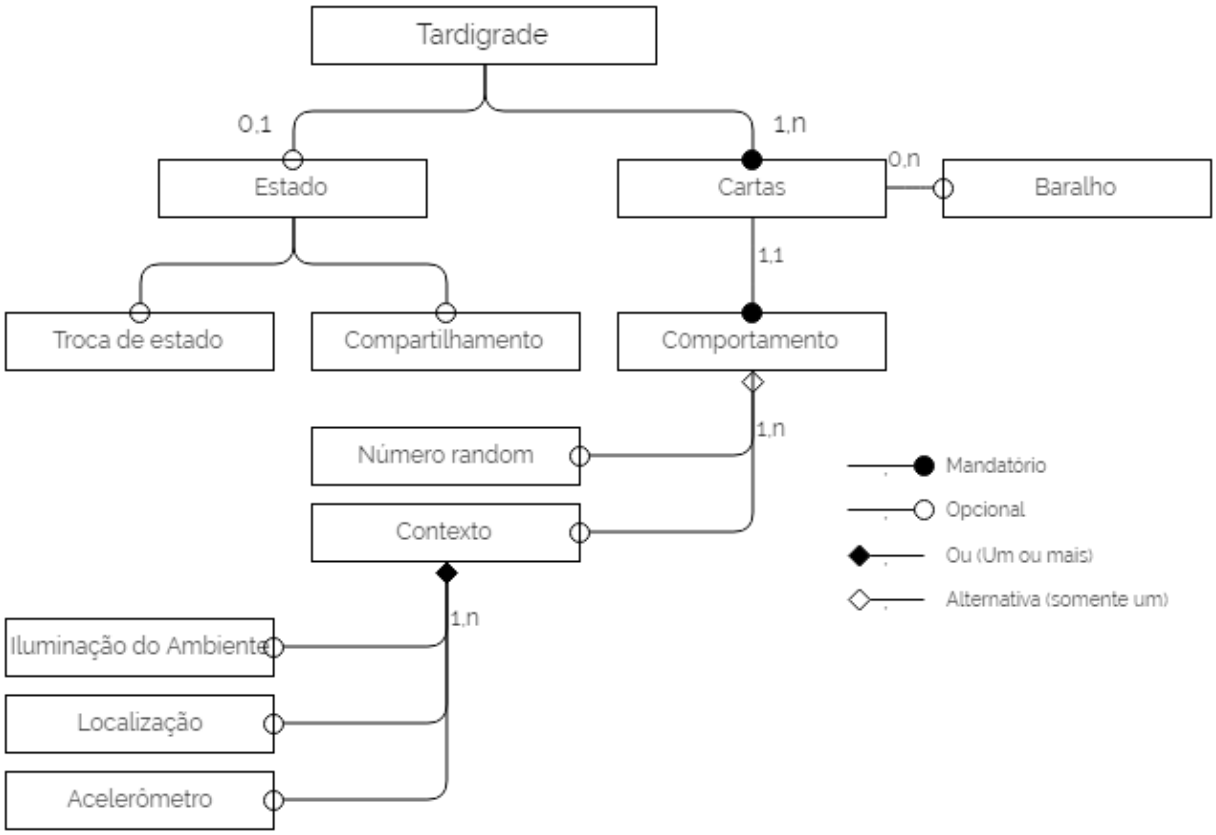
Para as três situações citadas, é possível utilizar os três tipos de cartas do Tardigrade: cartas tangíveis-identificáveis, cartas tangíveis-interativas e cartas sensíveis ao contexto. O conceito de cada um desses tipos de cartas é explicado na Seção 4.3.

A utilização das cartas suportadas pelo *framework* exige a interação por meio do uso de dispositivos móveis. Essas interações ampliam as experiências do jogo analógico com a utilização de recursos digitais. Um possível exemplo de jogo analógico usando o Tardigrade é uma aplicação que explica o jogo de Poker identificando as cartas, mostrando possíveis jogadas e mostrando estratégias usando as cartas em jogo como informações de

entrada.

Outro uso que pode ser implementado com o *framework* é tornar um jogo de cartas acessível para pessoas com deficiência visual por meio de audiodescrição das cartas reconhecidas. Essa aplicação não seria de uso mandatório, mas seu uso mudaria a experiência dos jogadores, em especial, àqueles com algum problema visual.

Figura 14 – Diagrama de *Features* do Modelo de Jogos do Tardigrade.



Fonte: o autor.

A Figura 14 apresenta um diagrama de características do modelo desenvolvido para o *framework* baseado no cenário motivador apresentado. No diagrama, as ligações terminadas em círculos preenchidos representam relações mandatórias, dessa forma pode-se observar que um jogo de cartas ubíquo baseado no modelo do Tardigrade pode ter: **a)** estados que descrevem o jogo, **b)** esses estados podem ser compartilhados e/ou trocados entre diferentes instâncias (jogadores); **c)** todos os jogos devem ter cartas que **d)** podem ser gerenciadas e armazenadas em um baralho; e **e)** essas cartas possuem comportamento de um ou mais tipos, como por exemplo, cartas com comportamento sensível ao contexto fazem uso do contexto adquiridos a partir de sensores ou uma carta identificável numa

aplicação acessível vai realizar a audiodescrição das informações contidas nesta carta.

As visões de sistemas ubíquos (SPÍNOLA; TRAVASSOS, 2012) e de jogos ubíquos (CHEUNG *et al.*, 2013) foram utilizadas para guiar as características do modelo. Por exemplo, o reconhecimento de cartas tangíveis é uma característica adaptável que pode levar em consideração a disponibilidade de recursos do dispositivo, podendo elas serem reconhecidas usando distintas formas (e.g., NFC, QR Code). Os jogos derivados do modelo devem ser desenhados pensando na interação com dispositivos móveis e a heterogeneidade de sensores que eles possuem. A tangibilidade pode ser incorporada a partir das cartas físicas do jogo e a propriedade de autoajuste instantâneo está associada ao comportamento dessas cartas. Por exemplo, ao projetar o comportamento de uma carta que usa como base o sensor de proximidade, o game designer deve pensar num comportamento alternativo caso o dispositivo não disponha do sensor desejado.

4.3 Tipos de Carta Tardigrade

O Tardigrade suportar três tipos de cartas com comportamentos e utilidades distintas, a saber: as tangíveis-identificáveis; as tangíveis-interativas; e as sensíveis ao contexto. Todas essas cartas podem ter efeitos de aumento de realidade, revelando sons, animações e interações gráficas que tornem a experiência de jogo mais imersiva.

As **cartas tangíveis-identificáveis** possuem um ou mais elementos de identificação que permitem a passagem de suas propriedades da realidade para a virtualidade. O elemento de identificação pode ser uma etiqueta NFC, QR Code, um padrão visual ou qualquer outra coisa que permita sua identificação por meio de um dispositivo móvel. Uma aplicação que reconhece as cartas do baralho de UNO e faz sua transcrição em áudio é um exemplo de uso de cartas tangíveis-identificáveis que pode criar interação de jogadores videntes e não videntes em um jogo já conhecido sem que este precise de modificações.

As **cartas tangíveis-interativas** são aquelas, que além da identificação, possuem propriedades que variam ou são ativadas de acordo com elementos de interação (e.g., o jogador deve agitar o celular para que ela funcione) ou elementos do mundo digital (e.g., valor inteiro randômico dentro de um intervalo). Essas cartas permitem jogos com características híbridas, isto é, jogos analógicos que se misturam com o virtual. Como em um jogo que abre uma janela de interação onde o jogador precisa sacudir o dispositivo para obter um resultado baseado nessa interação.

Uma **carta sensível ao contexto** está relacionada à ideia de que elementos do contexto de um ou mais jogadores podem ser utilizados para aumentar, diminuir, ativar ou desativar efeitos de cartas em um jogo. Por exemplo, em um jogo com estilo inspirado no *Yu-gi-oh!*, cartas de polimerização só poderiam ser utilizadas se o jogador estivesse em um contexto específico (e.g., o jogador deve estar sorrindo em uma manhã chuvosa). O dispositivo digital funciona como verificador dessa condição contextual (detecta o sorriso do jogador e checa a meteorologia daquela localização).

Ao utilizar cartas sensíveis ao contexto, o jogo pode ser qualificado como jogo ubíquo já que faz o uso de contexto de entidades diversas, sejam elas usuários, outros sistemas, dispositivos ou a própria aplicação do jogo. Este conceito está associado às definições de contexto e de sensibilidade ao contexto apresentadas (VIANA, 2010)(ABOWD *et al.*, 1999). Ao reconhecer uma carta sensível ao contexto, as entidades relevantes são adicionadas à zona de interesse da aplicação (e.g., rosto do jogador, meteorologia). O efeito da carta acontece quando as entidades na zona de interesse são também entidades observáveis e, no caso de um condicional, a sua condição de ativação deve ser atendida. Essa condição pode ser referente a uma atividade, identidade, localização ou tempo, e seu efeito é baseado, não exclusivamente, em execução automática de algo.

O tipo de contexto apresenta a entidade de observação que é necessária. Uma carta que, por exemplo, precise que o jogador esteja em movimento para ser ativada e precisa monitorar a atividade do jogador (e.g., se está em movimento, se esta parado ou se está dirigindo) tem como entidade a atividade do usuário.

Esses três tipos de cartas apresentadas permitem o desenvolvimento de diversos jogos de cartas que podem ser entendidos como ubíquos. Uma das características que permite isso é a tangibilização de componentes reais para o meio virtual, que podem ou não fazer o uso do contexto dos jogadores como componentes do jogo. Por exemplo, um jogo no qual os poderes do jogador são calculados através de dados precisos do sensor que mede a luminosidade do ambiente, ou usa como parâmetro uma determinada palavra-chave (*hashtags*) utilizadas em redes sociais. Cartas que permitem o uso de itens especiais ou criam um vínculo entre o jogo digital e o contexto de um ambiente externo ao do jogo.

4.4 Módulos e estrutura do *Framework*

O *framework* foi desenvolvido para plataforma móvel Android, utilizando seu Kit de Desenvolvimento de Software (SDK) como fonte de recursos para as funcionalidades básicas implementadas pelo *framework*. A Figura 15 ilustra os módulos principais do Tardigrade e suas relações. Esses módulos são:

1. Módulo de comunicação: responsável pela descoberta, conexão e troca de informações entre dispositivos;
2. Módulo de gerenciamento de cartas: responsável por gerenciar, carregar e executar as cartas e seus diferentes comportamentos e o reconhecimento das cartas físicas usando o Módulo Hub;
3. Módulo de estado do jogo: que descreve o cenário do jogo em um dado momento dele; e
4. Módulo Hub: que é responsável por agrupar e intermediar a comunicação entre os demais módulos do *framework* e a camada de aquisição de contexto, seja essa nativa ou uma plataforma de *middleware* de terceiros.

4.5 Aquisição de contexto

Para adquirir contexto, inicialmente o *framework* foi acoplado à plataforma de *middleware Loosely Coupled Context Acquisition Middleware* (LoCCAM) (MAIA *et al.*, 2013) que oferece acesso a informações de sensores, processamento de contexto e regras de ativação contextual utilizando os Componente de Aquisição de Contextos (CACs). A plataforma de *middleware* LoCCAM funciona como uma aplicação que gerencia diferentes contextos numa instância única que pode ser acessada por diferentes aplicações daquele dispositivo.

Em outra variação, foi utilizada a *Awareness API do Google*[1]. A proposta do serviço da Google é que seja possível criar aplicações que reajam inteligentemente à situação atual do usuário. São sete diferentes tipos de contexto que podem ser explorados pelo serviço, incluindo localização, clima, atividade do usuário e dispositivos próximos.

A troca da plataforma de *middleware* LoCCAM pela *Awareness API* foi realizada pela facilidade na integração, acurácia, melhor gerenciamento dos recursos de

[1] Google API Awareness https://developers.google.com/awareness/

Figura 15 – Organização estrutural dos módulos principais do Tardigrade.

Fonte: o autor.

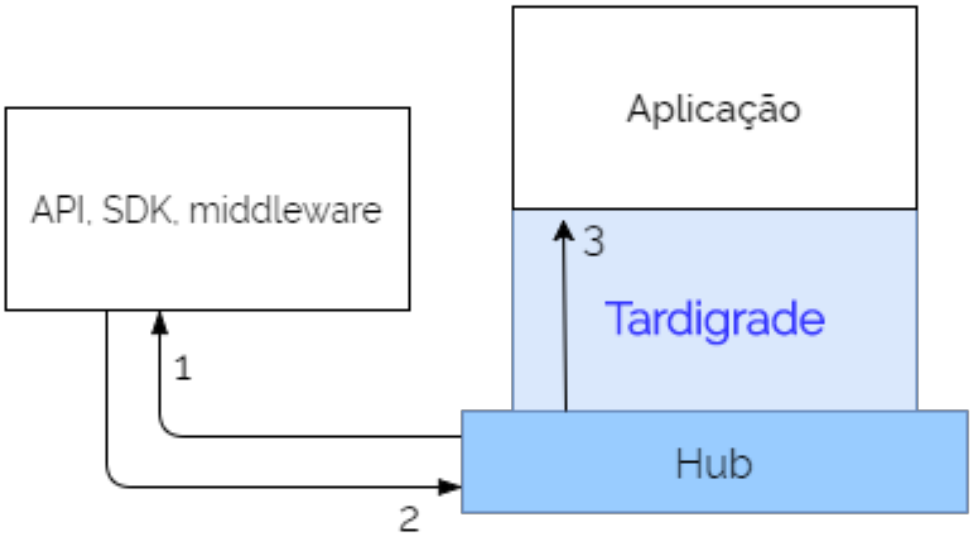
processamento e bateria do dispositivo, e, principalmente, por funcionar sem a necessidade de uma aplicação própria instalada no dispositivo. Contudo, vale ressaltar que o *framework* suporta as duas abordagens de aquisição de contexto.

4.6 Módulos do Tardigrade

4.6.1 Hub

O `Hub` funciona como um agregador de informações e funções externas ao *framework*, uma espécie de padrão *Façade* para o acesso a essas informações (e.g., dados contextuais capturados pela *Context API*). Sua responsabilidade é agrupar métodos de integração, sejam com sensores nativos, plataformas de *middleware* ou bibliotecas externas. Foi implementado seguindo o padrão *Singleton*, de forma que, com sua instância, é possível acessar o mesmo serviço de diferentes partes da aplicação. A Figura 16 ilustra o fluxo do `Hub` utilizando uma API para prover uma funcionalidade específica de um jogo.

Figura 16 – Fluxo do Hub para integração de serviços.



Fonte: o autor.

Na Figura apresentada, o fluxo começa com o `Hub` que implementa a chamada do serviço e recebe as respostas das requisições realizadas. O `Hub` interage com a aplicação passando pelo *framework*. A interação com aplicações de terceiros possui um fluxo diferenciado. Esse tipo interação do `Hub` faz uso do modelo de comunicação baseado em `Intents` do Android. `Intents` podem ser vistas como objetos de mensagens usadas para solicitar uma ação de outra aplicação ou de um serviço. O resultado dessa solicitação é recebido pela aplicação como um novo objeto de mensagem no *callback* `onActivityResult()` (Figura 17).

Um dos recursos mais importantes do modelo de jogo de cartas ubíquo é a tangibilização das cartas físicas para o meio digital. O *Hub* é quem intermedia a interação de leitura de carta. Dentre as implementações já disponíveis está a utilização de leitura de QR Code para identificar cartas que possuam um código de barras bidimensional. O trecho de código 1 apresenta como é a chamada para a leitura de uma nova carta.

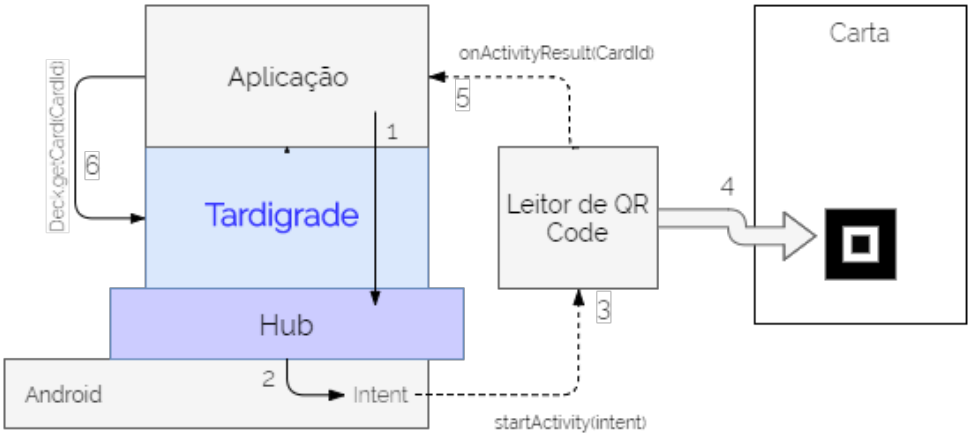
Código-fonte 1 – Chamada do Hub para ler uma carta por QR Code

```
1 Hub.getInstance(MainActivity.this).readQR();
```

Quando uma carta é identificada, a informação lida pelo *Hub* pode ser um identificador associado às informações da carta numa base de dados da aplicação, ou a própria informação lida permite a "construção" da versão digital da carta em tempo de execução. A Figura 17 apresenta o fluxo realizado na leitura de uma carta por QR Code.

O `Hub` inicia uma intenção de leitura de QR Code e a solicitação abre a aplicação que faz a leitura e retorna o resultado para o `Hub` que a repassa para a aplicação. O último passo consiste em tratar o resultado obtido para recuperar as informações da carta identificada.

Figura 17 – Funcionamento do reconhecimento de cartas.



Fonte: o autor.

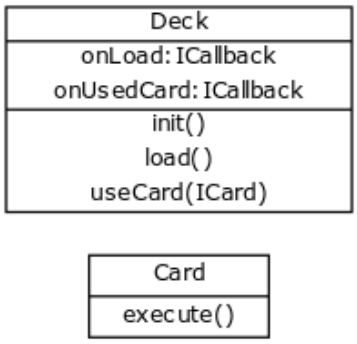
Uma outra implementação permite a leitura de cartas de forma mais simples, utilizando etiquetas NFC, permitindo que as cartas sejam identificadas apenas aproximando-as de dispositivos compatíveis com a tecnologia. As diferentes formas de identificar uma carta podem e devem funcionar em paralelo, permitindo ao usuário escolher qual delas é mais eficiente para ele.

4.6.2 Módulo de Cartas

O módulo de cartas é composto pelas classes `Deck` e `Card`. A classe `Deck` representa o baralho e é responsável por agregar e gerenciar um conjunto de objetos `Card`. A classe `Deck` pode armazenar e fornecer informações de objetos `Card`.

Apresentado na Figura 18, o método `init()` é quem realiza a inicialização do `Deck`. O método `load()` é utilizado para carregar um conjunto de cartas, sincronizando essas informações com uma base de dados da aplicação. Uma implementação auxiliar permite que um arquivo *.csv* (com dados separados por virgulas) forneça informações de um conjunto de cartas a serem sincronizadas com a base de dados. Para controlar e fornecer informações à aplicação, o atributo `onLoad` do `Deck` executa um evento que é executado ao fim de uma sincronização e o atributo `onUseCard` pode executar um evento quando uma carta é executada.

Figura 18 – Resumo do diagrama de classe do módulo de cartas.



Fonte: o autor.

Código-fonte 2 – Implementação da carta de baralho Ás de Paus

```
1 /** Atributes:
2  *  Id, Name, Description, Deck color, Card value, Card
      Symbol
3  */
4
5 Card.Create("AP1", "As", "Paus", "Azul", "1", "P").
```

A classe `Card` implementa uma representação virtual das cartas físicas e seus atributos (e.g, identificador, nome, descrição, um vetor de valores que representam os atributos da carta) e possui métodos para recuperar e gerenciar essas informações. O código-fonte 2 exemplifica como seria feita a construção de uma carta de baralho com a ajuda do *framework*.

O comportamento dessas cartas é encapsulado na classe `Card` seguindo o padrão de projeto *Command*. A implementação desse comportamento é feita no método `execute()` do objeto `Card`. O comportamento ubíquo e/ou interativo tem o `Hub` como fonte, onde também acontece a interação com uma plataforma de *middleware*, biblioteca externa, sensores, ou com o serviço responsável por adquirir o contexto. O trecho de código 3 apresenta como seria a interação para realizar uma audiodescrição de uma carta de baralho.

Código-fonte 3 – Implementação do comando de audiodescrição para uma carta

```
1 @Override
2 public void execute(){
3     Hub.getInstance().AudioDescription(Card.this);
4 }
```

Cartas identificáveis, interativas e sensíveis ao contexto são implementadas a

partir desse mesmo método, invocando os métodos da classe `Hub` conforme seu comportamento. Desta forma, o método `execute()` é uma interface que permite a execução de uma carta, mesmo que esta seja serializada e enviada pela rede a um outro dispositivo. Toda carta, quando é utilizada, dispara no `Deck` o evento do atributo `onUseCard`. Cartas também podem implementar o método `revert()` que desfaz a ação realizada no comando `execute()`.

4.6.3 Módulo de Estado de Jogo

O estado de um jogo no Tardigrade representa a memória do andamento e do progresso do mesmo. Esse estado descreve informações do cenário, dos objetos envolvidos e dos jogadores em um dado momento do jogo, e possui uma estrutura única para as regras de cada jogo, ficando a cargo do desenvolvedor do jogo explicitar suas peculiaridades ao usar o *framework*.

No Tardigrade, uma classe `State` é quem cumpre esse papel de gerenciamento do estado de jogo. O código 4 exemplifica a declaração da classe e seus eventos. Os eventos representam três momentos: **1)** quando o estado do jogo é modificado pelo próprio jogador, `onChange` na linha 2; **2)** sincronizado com outro dispositivo, `onUpdate` na linha 8, e **3)** o último quando qualquer alteração no estado foi realizada, seja ela local ou oriunda de uma sincronização, `onMake` na linha 14.

Código-fonte 4 – Exemplo de instanciação de um estado

```
1  State state = State.getInstance();
2  state.setOnChange(new ICallback(){
3      @Override
4      public void doit(IPack pack){
5          //Action to modify game state
6      }
7  });
8  state.setOnUpdate(new ICallback(){
9      @Override
10     public void doit(IPack pack){
11         //Action when game state is received from another
               player
12     }
13 });
14 state.setOnMake(new ICallback(){
15     @Override
16     public void doit(IPack pack){
17         //Change pack.getValue() independent from the
```

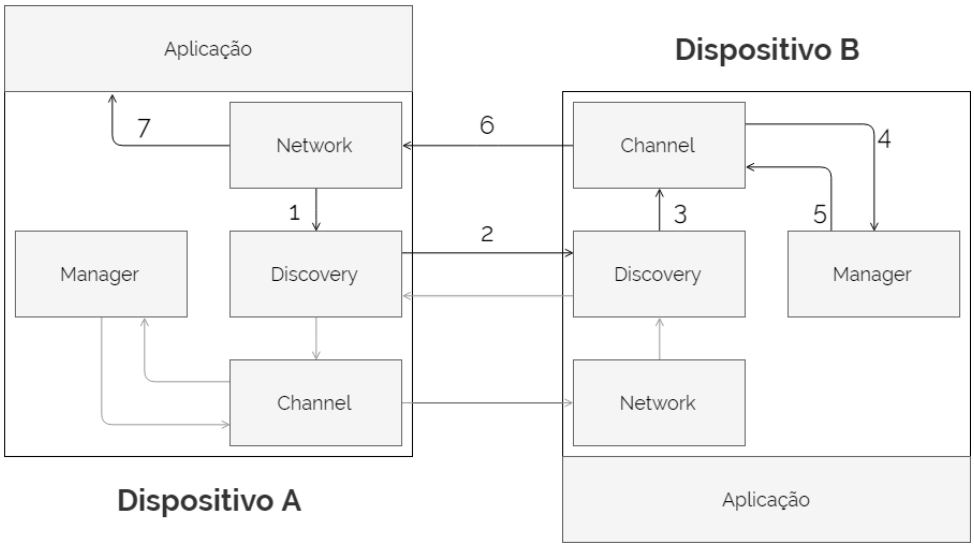
```
                origin of game state change
18      }
19  });
```

As alterações de estado de jogo recebidas por mensagem disparam o `onReceiverUpdate` do módulo de comunicação (Seção 4.6.4). No fim de cada etapa de alteração de estado de jogo, o evento `onMake` realiza as alterações na interface e sinaliza as alterações realizadas.

4.6.4 Módulo de Comunicação

O módulo de comunicação é responsável por descobrir, conectar e gerenciar a troca de mensagens entre dispositivos conectados. A Figura 19 representa a integração das classes desse módulo que é composto por quatro componentes: a classe `Network` que ativa um canal de recebimento de mensagens; a classe `Discovery` que implementa um serviço de rede e a descoberta de outros dispositivos que possuam o mesmo serviço ativado (integrando-se com a classe `Network` para criar um laço entre dispositivos); cada conexão é representada por um objeto `Channel` que envia mensagens para o canal de rede correspondente; e iv) a classe `Manager` que implementa o padrão *Observer* enviando mensagens a um grupo de canais observadores deste objeto.

Figura 19 – Módulo de comunicação.



Fonte: o autor.

A Figura 19 apresenta o fluxo que é seguido no processo de comunicação. 1) O serviço de rede é iniciado e associado ao serviço de descoberta; 2) o Serviço de descoberta

descobre outros serviços semelhantes na rede; 3) ao descobrir e se conectar a um outro dispositivo, é criado um canal de comunicação; 4) o canal de comunicação é armazenado pelo `Manager`, 5) que gerencia o envio de mensagens para os canais; 6) o canal faz a entrega da mensagem 7) que é recebida pelo canal de rede.

O trecho de código no algoritmo 5 apresenta a instanciação da classe `Network` e do seu evento que recebe os pacotes com mensagens. O método `setOnReceivePack` recebe um objeto do tipo `Pack`. A estrutura de um objeto `Pack` contém um par de *chave* - do tipo `String` - e um *valor* - do tipo `Object`.

Código-fonte 5 – Implementação do evento que recebe as mensagens

```
Network.getInstance().setOnReceivePack(new ICallback() {
    @Override
    public void doit(IPack pack) {
        //Fazer uso do pacote recebido
    }
});
```

A classe `Discovery` possui quatro métodos principais. Dois deles dizem respeito ao inicio e ao fim do serviço de descoberta - `onStart` e `onStop`. Já os outros dois são acionados, respectivamente, quando um dispositivo com o serviço de interesse é encontrado (`onFound`) e quando ele é perdido (`onLost`). Nos eventos em que um dispositivo é descoberto ou perdido, o conteúdo do objeto de mensagem é um objeto `Channel` que pode ser utilizado para manter comunicação com o dispositivo encontrado. Na implementação disponível no *framework*, o serviço de descoberta limita-se a encontrar dispositivos que estejam conectados numa mesma Wireless Local Area Network (WLAN). O trecho de código 6 apresenta a declaração da classe `Discovery` e a implementação da descoberta de um dispositivo.

Código-fonte 6 – Implementação do serviço de descoberta

```
// Declaracao do Discovery
Discovery mDiscovery = Discovery.getInstance();
// Implementacao do evento OnFound
mDiscovery.setOnFoundDevice(new ICallback(){
    @Override
    public void doit(IPack data){
        IChannel device = (IChannel) data.getValue();

        // Registra o dispositivo encontrado no Manager
        Manager.getInstance().registerObserver(device);
```

```
11      }
12  });
13  // Inicia o servico de descoberta
14  mDiscovery.start();
```

O `Manager` integra-se ao canal de comunicação distribuindo mensagens via *broadcast*[2]. O método `registerObserver` da classe compõem uma lista de objetos `Channel` como observadores. Quando uma mensagem é enviada utilizando o método `notifyObservers`, ela será endereçada a cada dispositivo da lista. O trecho de código 7 apresenta a integração da classe `Manager` com a `State` para notificar todos os observadores no caso de uma alteração do estado do jogo.

Código-fonte 7 – Implementação da integração entre as classes `Manager` e `State`

```
1  // Declaracao do manager
2  Manager mManager = Manager.getInstance();
3  // Implementacao do evento onChange na classe State
4  State.getInstance().setOnChange(new ICallback(){
5      @Override
6      public void doit(Pack data){
7          //Notificar todos os observadores sobre a mudanca
               na classe State
8          mManager.notifyObservers(data);
9      }
10 });
```

4.7 Conclusões

Este capítulo apresentou a proposta principal deste trabalho, o *framework* Tardigrade para desenvolvimento de jogos de cartas Ubíquos para a plataforma Android.

Um cenário motivador foi apresentado para ilustrar as propriedades e comportamentos desejados em jogos de cartas ubíquos. O cenário também apresenta o uso de contexto do usuário e do ambiente na dinâmica do jogo desde a descoberta de outros jogadores até a influência do funcionamento dos atributos de uma carta tangível. A partir desse cenário, um modelo base do *framework* foi descrito e seus componentes procuram fornecer o suporte necessário para o desenvolvimento de jogos, assim como o do cenário motivador, que mixem cartas tangíveis com elementos digitais por meio de um aplicativo móvel. E apresenta três modalidades distintas de jogos : um dispositivo para vários

[2] Emissão e transmissão em massa.

jogadores; múltiplos dispositivos, sem conexão entre eles, para vários jogadores; e cada jogador com um dispositivo e conexão entre eles.

No processo de criação do *framework*, buscou-se o uso de padrões de projeto de software com o intuito de melhor organizar a estrutura do código e facilitar tanto a manutenção quanto a compreensão dos desenvolvedores que irão utilizá-lo. Embora o *framework* não tenha sido concebido especificamente para plataforma Android, alguns dos relacionamentos entre as classes (em especial a classe `Discovery`) tem forte acoplamento com a plataforma.

O *framework* suporta três tipos de cartas, são elas: cartas tangíveis-identificáveis, cartas tangíveis-interativas e cartas tangíveis sensíveis ao contexto. As cartas tangíveis-identificáveis são cartas com comportamento previsível. As cartas tangíveis-interativas já possuem comportamento dependente de interação entre componentes virtuais ou físicos. E as cartas sensíveis ao contexto fazem o uso do contexto de entidades (e.g. usuários, dispositivos, sistemas e ambiente) para definir o seu comportamento.

O capitulo seguinte apresenta a implementação de uma instância do *framework* para o desenvolvimento do jogo *Hyper Try* como prova de conceito do Tardigrade.

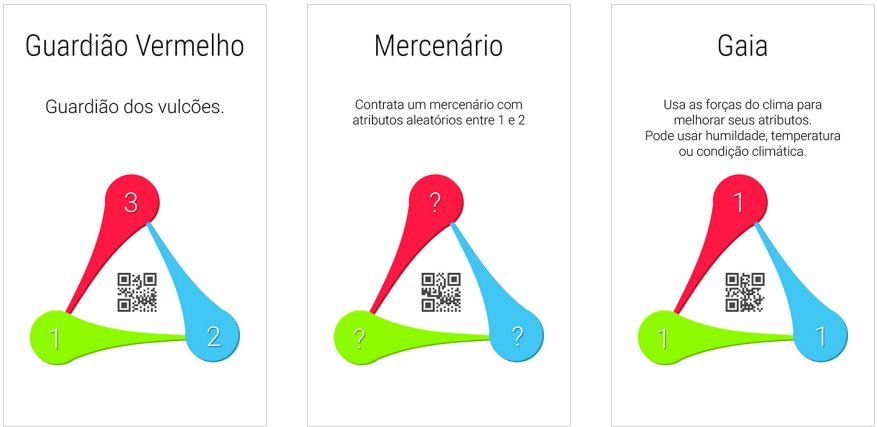
5 PROVA DE CONCEITO

Neste capítulo, é apresentado o Jogo *Hyper Try*, desenvolvido com a utilização o *framework* Tardigrade. Na Seção 5.1, o jogo e suas mecânicas são apresentadas. Na Seção 5.2, são abordadas as especifidades da implementação do *Hyper Try* e sua relação com as classes e os módulos do *framework* Tardigrade. A Seção 5.3 apresenta a avaliação do jogo baseada na metodologia PLEXQ (BOBERG *et al.*, 2015), que foi realizada com 14 voluntários a fim de verificar se este cumpre seu papel como objeto de interação e experiência lúdica. Na Seção 5.4 são apresentadas as conclusões do capítulo.

5.1 *Hyper Try*

Hyper Try é um jogo de cartas ubíquo para dois jogadores que disputam um duelo entre titãs com poderes de fogo, água e terra. A dinâmica do jogo consiste em um baralho com oito personagens com características distintas e em dispositivos dos jogadores conectados numa mesma rede sem fio. As batalhas acontecem no meio virtual através da tangibilização das cartas e comunicação entre os dispositivos. A Figura 20 apresenta três das oito cartas do jogo.

Figura 20 – Cartas do jogo Hyper Try. Guardião Vermelho como carta tangível-identificável; Mercenário como carta tangível-interativa; Gaia como carta tangível sensível ao contexto.



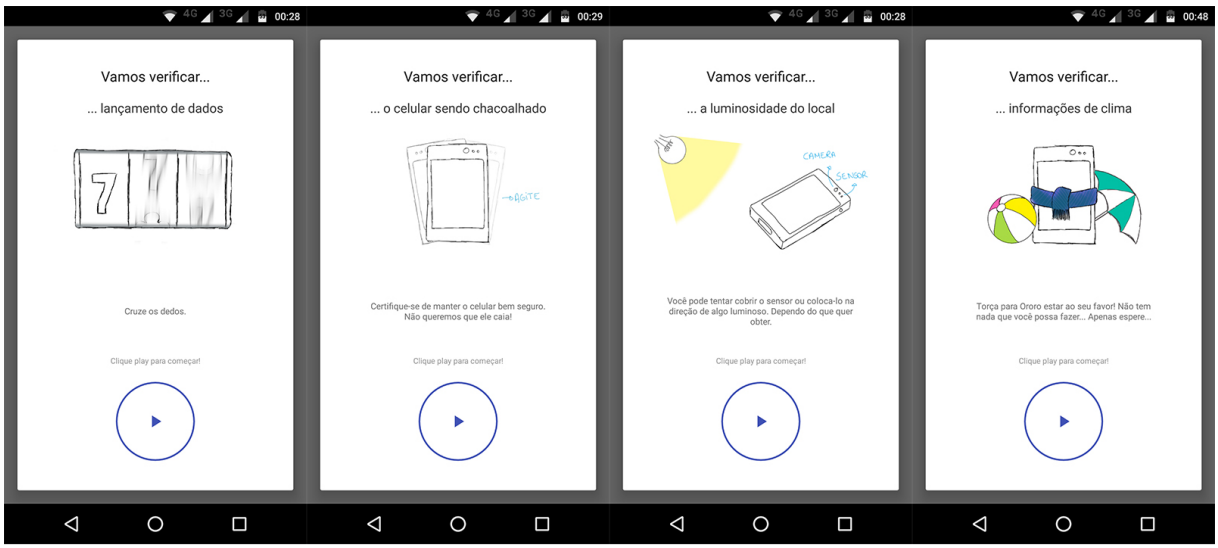
Fonte: o autor.

Das informações apresentadas em cada carta estão: o nome e a descrição da

carta, os seus atributos, representados pelas cores vermelho, azul e verde, e o QR Code usado para identificação.

Em cada *round*, os jogadores devem escolher um dos seus titãs e reconhecê-lo com o aplicativo. Os poderes de alguns titãs dependem de interações como: geração de valores aleatórios, chacoalhar o dispositivo ou analisar a luminosidade do ambiente e informações de clima como temperatura, condição climática e umidade (Figura 21).

Figura 21 – Janelas para interação ou captação de contexto. a) Janela de interação para números aleatórios b) janela de interação para detectar o dispositivo sendo sacudido c) janela de interação para detectar luminosidade do ambiente d) janela de detecção de clima.



a) b) c) d)

Fonte: o autor.

5.1.1 Regras

O jogo consiste em dividir as oito cartas entre os jogadores para que eles disputem 4 batalhas, escolhendo um titã para cada batalha sem repetir.

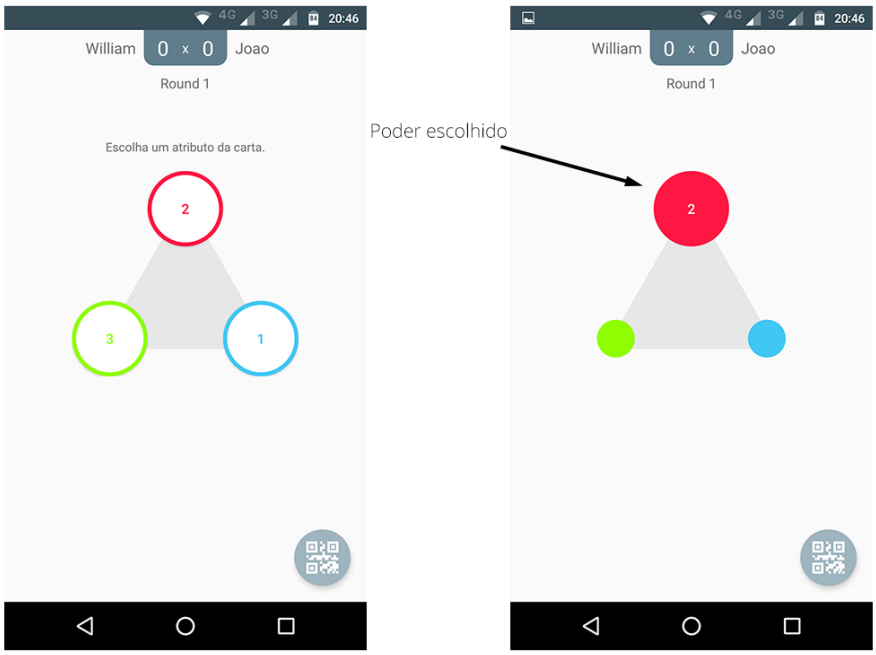
Cada batalha é representada por um *round*, e para cada *round* **1)** os jogadores devem escolher uma carta de titã; **2a)** se o titã depende de alguma interação ou contexto, deve-se seguir as instruções apresentadas no dispositivo para interação ou captação do contexto solicitado; **2b)** os jogadores devem escolher uma das três forças de poderes de batalha apresentado no dispositivo (água, fogo ou terra); **3)** o jogador que teve o titã com

maior poder de força ganha o *round*. O jogo acaba depois de quatro *rounds* e o jogador que tiver mais pontos de vitória ganha o jogo.

5.1.2 Mecânica do jogo

Dois valores são importantes durante uma batalha: o poder de força de um titã, que é o valor base da carta apresentado na tela de escolha, e o poder do ataque, que é calculado com base na escolha de força de ataque do adversário. A Figura 22 apresenta a tela de escolha de atributo e como o poder pode ser escolhido pelo jogador.

Figura 22 – Telas para escolha de poder de força e tela com poder escolhido.

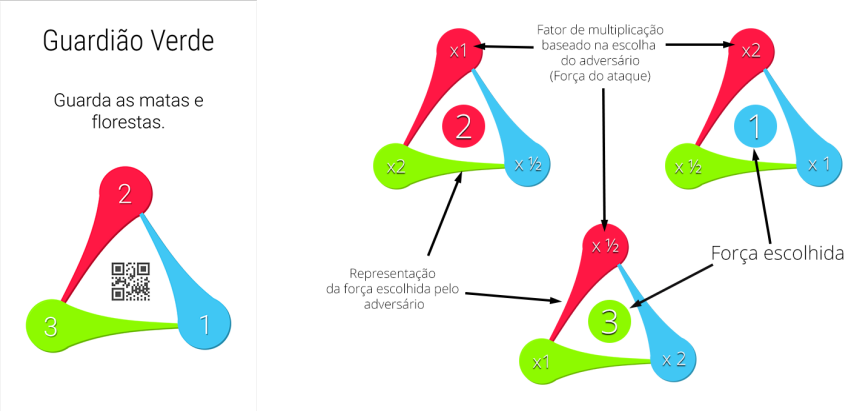


Fonte: o autor.

Foi criado um sistema de vantagens e desvantagens de força de atributos, assim cada um possui vantagem contra um outro atributo e desvantagem contra o terceiro. O poder do ataque realizado é calculado de acordo com a vantagem e a desvantagem do atributo escolhido. Contra uma vantagem o poder de ataque do titã dobra, e contra uma desvantagem o poder do ataque cai pela metade. A Figura 23 ilustra as escolhas do poder de força da carta “Guardião Verde” e o fator de multiplicação para cada uma delas. Seguindo o exemplo de escolha da Figura 22, se o jogador escolhe a força de poder fogo de valor dois, seu fator multiplicador será de 1/2 quando o adversário escolher o poder

de água, de 1 quando o adversário escolher o poder de fogo e de 2 quando o adversário escolher o poder de terra.

Figura 23 – Carta “Guardião verde” e o fato de multiplicação para cada uma das forças baseado na escolha do adversário.



Fonte: o autor.

Os atributos das cartas, ou seja, os poderes dos titãs, podem ser obtidos de seis formas distintas. A forma mais simples ocorre nas cartas tangíveis-identificáveis, na qual a carta reconhecida possui seus valores de atributos fixos apresentados na tela para escolha. Para as cartas tangíveis-interativas, existem duas mecânicas que geram os valores dos atributos das cartas identificadas: a primeira forma calcula os atributos com base em quantas vezes o jogador consegue chacoalhar o dispositivo, e a segunda utiliza o sensor de luminosidade inferindo os valores dos atributos da carta identificada de acordo a claridade do ambiente. Nas outras três formas, os atributos das cartas são gerados a partir de informações contextuais do clima, baseado na localização do jogador. Entre as informações contextuais, são utilizadas: a condição climática (e.g. nublado, chuvoso, ensolarado), a temperatura (C ou F) e a umidade do ar (em %).

5.2 Implementação

A implementação do jogo utilizou como base a versão inicial do *framework* Tardigrade. Inicialmente, foi criada uma base de dados em um arquivo de texto com as cartas do jogo e as suas informações. A Figura 24 apresenta essa base de dados na qual cada linha representa um registro com suas propriedades separadas por vírgula.

Figura 24 – Base de dados de cartas como arquivo de texto CSV.

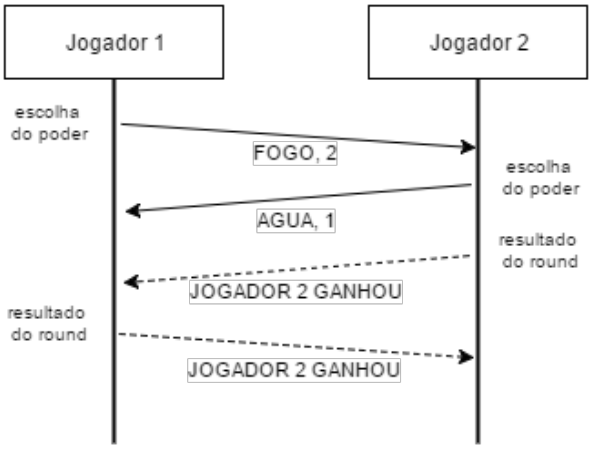
```
uid;name;description;ATTR_RED;ATTR_BLUE;ATTR_GREEN
1;  Guardião Vermelho;Guardião dos vulcães.;3;2;1
2;  Guardião Azul;Guarda os oceanos, mares e rios.;1;3;2
3;  Guardião Verde;Guarda as matas e florestas.;2;1;3
4;  Guardião do Destino;As vezes bom. As vezes mau. Gera atributos aleatórios entre 0 e 3.;0;0;0
5;  Mercenário;Gera atributos aleatórios entre 1 e 2;0;0;0
6;  Guardiã da Justiça;Você escolhe o quão justa ela será. Quanto mais ela enxerga, maior o risco.;0;0;0
7;  Guardião Adormecido;Ele precisa ser acordado.;0;0;0
8;  Gaia;Usa as forças do clima para melhorar seus atributos;1;1;1
```

Fonte: o autor.

O comportamento de cada carta foi implementado utilizando a integração já disponível no *framework* com o *API Google Awareness* (clima) e as interações do *framework* com sensores lógicos e físicos (acelerômetro, sensor de luminosidade e gerador de números aleatórios) do dispositivo. O método de execução da carta usa o **id** da carta para executar o seu comportamento correspondente. O comportamento de todas as cartas é dado como encerrado quando seus atributos são apresentados na tela do dispositivo que a reconheceu.

Ao escolher um dos valores apresentados na tela, o estado do jogo envia para o adversário duas informações: qual das forças (água, fogo ou terra) foi escolhida e qual o valor dessa força. O resultado é calculado duas vezes, uma vez em cada dispositivo, e o resultado que o dispositivo irá usar para si é o resultado que seu adversário enviar. Isso foi feito como forma de verificação, diminuindo a chance de *cheating*. A Figura 25 apresenta esse fluxo de troca de informações dos cálculos realizados.

Figura 25 – Fluxo de troca de informações de estado durante uma partida.



Fonte: o autor.

O estado do jogo foi implementado em formato de pilha e um novo nó é adicio-

nado todas as vezes em que um jogador escolhe o poder de uma carta e o adversário envia informações da escolha dele. O processamento consiste em verificar o fator multiplicador de cada escolha (Figura 23) e calcular qual poder de ataque resultante é maior, enviando as informações do vencedor para o adversário.

5.3 Avaliação de *Playful*

O objetivo da avaliação foi mensurar a experiência de jogo proporcionada pelo *Hyper Try* e, para isso, utilizou-se a metodologia PLEXQ (BOBERG *et al.*, 2015) que consiste num formulário que mensura 17 facetas da experiência de jogo. Quatro facetas foram removidas do questionário por não pertencerem ao escopo do jogo *Hyper Try*, são elas: *Completion* (Realização), *Expression* (Expressão), *Nurture* (Cuidado) e *Subversion* (Subversão).

O PLEXQ consiste em três itens do questionário para cada categoria/faceta. Numa versão adaptada, o formulário criado para esta avaliação consistiu em apenas dois itens por categoria, somando um total de 26 itens no questionário de avaliação. O objetivo da alteração foi reduzir o tempo de resposta ao formulário, para assim tornar o processo de avaliação mais viável. Os voluntários responderam cada item numa escala Likert com as seguintes opções: “discordo totalmente”, “discordo parcialmente”, “indiferente” ou “neutro”, “concordo parcialmente” e “concordo totalmente”.

5.3.1 Perfil dos jogadores

Os voluntários foram escolhidos por conveniência: 64.3% deles são pós-graduados ou pós-graduandos, 14.3% são graduados, 14.3% são graduandos e 7.1% restante possuem apenas o ensino médio completo. A faixa etária dos voluntários está entre 21 e 30 anos de idade.

5.3.2 Procedimento

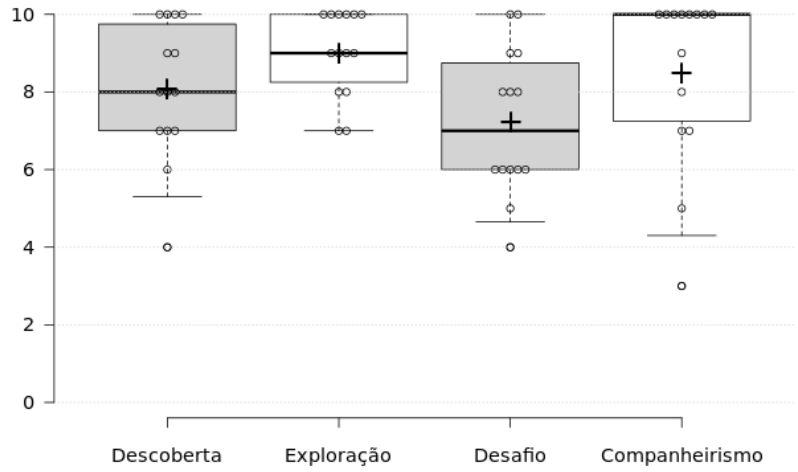
A avaliação foi realizada em dois momentos: num encontro informal, com jogadores de jogos de tabuleiro (quatro participantes) e, durante 3 dias, com voluntários que frequentaram a sala de pesquisa do Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREAT) localizado na Universidade Federal do Ceará.

A aplicação do jogo foi instalada no dispositivo do voluntário por meio do Google Play[1]. Com os dispositivos prontos para o jogo e com as cartas do jogo em mãos, todas as regras do jogo foram explicadas junto com o funcionamento do aplicativo. Os voluntários foram monitorados durante apenas uma partida de quatro *rounds*. Ao fim dessa partida, cada voluntário respondeu o questionário de avaliação.

5.3.3 Resultados e Discussão

Para a análise dos dados, foi calculada a média para cada par de perguntas referentes às categorias, obtendo 13 itens com 13 médias de avaliação cada. (BOBERG *et al.*, 2015) realiza a análise dos resultados do PLEXQ(BOBERG *et al.*, 2015) a partir de quatro dimensões da experiência de jogo, são elas: **1)** *stimulation* (Estimulação), **2)** *pragmatic* (Pragmatismo), **3)** *momentary* (Momentâneo) e **4)** *negative* (Negativo).

Figura 26 – Distribuição das médias obtidas para as categorias de descoberta, exploração, desafio e companheirismo.



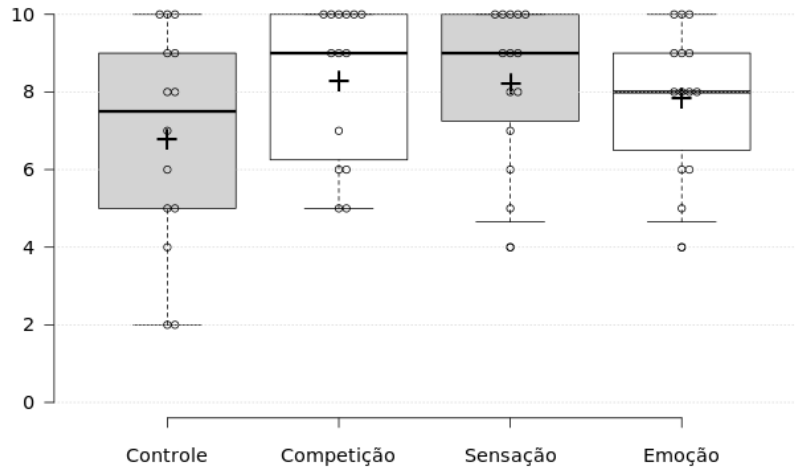
Fonte: o autor.

Na primeira dimensão, observa-se a experiência de estimulação do jogador. Pelo gráfico da Figura 26, destacam-se como pontos fortes do jogo: que os jogadores sentiram-se estimulados com os desafios impostos e com as novidades propostas no jogo (relacionados ao uso de contexto como informações de jogo). Os voluntários também sentiram-se estimulados a descobrir e explorar os componentes do jogo. E como ponto

[1] *Hyper Try* - https://play.google.com/store/apps/details?id=com.well.hypersupertrunfo

mais forte da categoria, eles sentiram-se estimulados a compartilhar a experiência de jogo com outras pessoas.

Figura 27 – Distribuição das médias obtidas para as categorias de controle, competição, sensação e emoção.



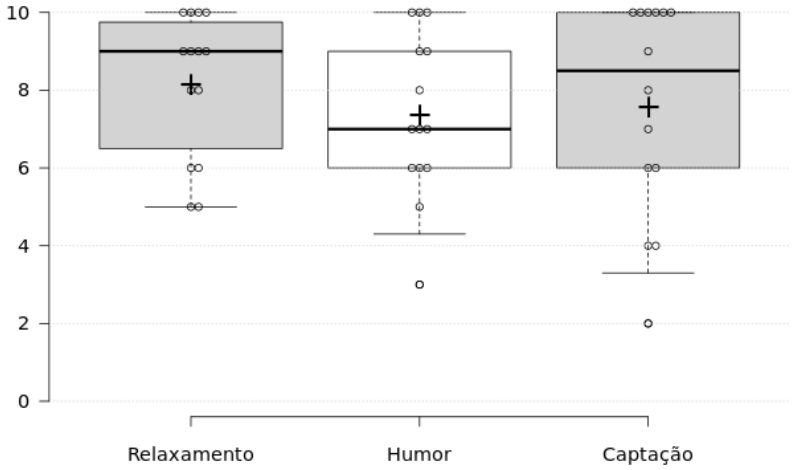
Fonte: o autor.

Na segunda dimensão de experiência, tem-se o pragmatismo. Esta dimensão é sobre a obtenção de resultados e manutenção do controle sobre eles. Isso pode ser relacionado a utilização de novas formas de interação disponíveis no jogo, envolvendo a captura de contexto e o controle parcial do jogador sobre os resultados dessa interação. Os resultados da avaliação obtidos para as categorias de controle e competição, apresentados na Figura 27, ditam a natureza competitiva do jogo que faz com que os jogadores foquem em obter resultados melhores do que o dos seus adversários. Os resultados obtidos nas categorias sensação e emoção mostram que o jogo possui uma boa aparência e gera excitação no jogador.

A terceira dimensão está relacionada às experiências momentâneas do jogo. Conforme apresentado na Figura 28, os jogadores relatam ter uma experiência de jogo que os distrai, captando a sua atenção e ao mesmo tempo que diverte.

A quarta e última dimensão está relacionada as experiências negativas de jogo. Se deparar com situações frustrantes ou de malícia podem causar uma experiência negativa para alguns jogadores, bem como a situação de tempo perdido ou sofrimento causado. A Figura 29 mostra que o jogo não proporciona uma experiência negativa de jogo considerável. Podendo-se concluir ainda que a distribuição do fator crueldade pode ser interpretada

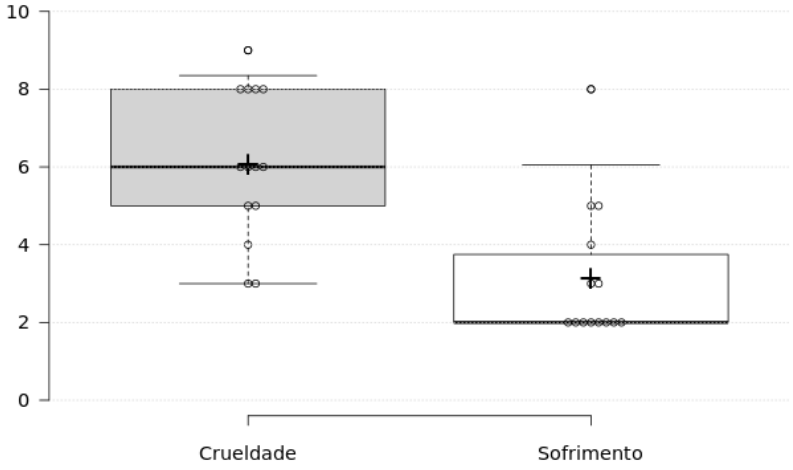
Figura 28 – Distribuição das médias relaxamento, humor e captação.



Fonte: o autor.

como: os jogadores que ganharam tiverem uma experiência de malícia ao liderar o jogo, enquanto a outra parte teve uma experiência negativa em decorrência dessa malícia.

Figura 29 – Distribuição das médias obtidas para as categorias de crueldade e sofrimento.



Fonte: o autor.

5.4 Conclusões

Neste capítulo foi apresentado o jogo Hyper Try e sua construção utilizando o *framework* Tardigrade. Foram apresentadas as especifidades do jogo, como as regras do jogo e as mecânicas envolvidas e a implementação a partir de uma instância do *framework*

Tardigrade.

O jogo construído utiliza os principais recursos disponíveis no *framework* Tardigrade como a implementação dos três tipos de cartas disponíveis, a comunicação entre dispositivos, o reconhecimento e gerenciamento de cartas e o gerenciamento do estado do jogo.

A avaliação de jogabilidade aplicada mostrou que o jogo desenvolvido cumpre seu papel como ferramenta de entretenimento e desperta o interesse dos jogadores devido ao seu caráter inovador de interação entre o real e o digital.

Em uma análise geral, é possível apontar que as mecânicas do jogo de tangibilização de cartas físicas e de interação com sensores e o contexto do ambiente são fatores relevantes nos resultados obtidos, desde o estímulo de descoberta e exploração até o controle que o jogador assume sobre as interações realizadas. Foram principalmente esses aspectos que captaram a atenção e tornaram a experiência do jogo divertida e positiva.

6 AVALIAÇÃO DO *FRAMEWORK*

O *framework* Tardigrade tem como propósito disponibilizar uma forma de implementar jogos de cartas ubíquos que apresentem interação do real com o virtual, podendo ou não fazer uso de contexto como componente de jogo. Dessa forma, neste capítulo é descrito o processo de avaliação inicial do *framework* como ferramenta de desenvolvimento que tem como foco a validação das funcionalidades do *framework* para reconhecimento e gerenciamento de cartas, estado de jogo e comunicação entre dispositivos.

As seções deste capítulo descrevem a avaliação realizada com seis voluntários que possuíam algum nível de experiência no desenvolvimento de aplicações Android, detalhando as atividades desenvolvidas durante o processo de experimento, como os questionários aplicados e o processo de capacitação dos voluntários para utilizarem o *framework*, os materiais utilizados e os resultados obtidos e as discussões pertinentes.

A Seção 6.1 apresenta os objetivos do experimento com o *framework*. Seguidos pela Seção 6.2 que apresenta o perfil dos usuários, a Seção 6.3 que apresenta ameaças a validade do experimento realizado, a Seção 6.4 descreve o procedimento realizado durante a avaliação e a Seção 6.5 apresenta os materiais e instrumentos do experimento. Na Seção 6.6 são apresentados os resultados e discussões do experimento. A Seção 6.7 apresenta os problemas e limitações encontrados durante o experimento. Por fim, são apresentadas as conclusões do experimento realizado na Seção 6.8.

6.1 Objetivos do Experimento

O objetivo dessa avaliação é validar o *framework* Tardigrade e suas funcionalidades para o desenvolvimento de jogos de cartas ubíquos tendo como foco verificar se desenvolvedores com alguma experiência no desenvolvimento de aplicações Android conseguem utilizar o *framework* e os componentes disponíveis com facilidade.

Foi escolhido de forma não probabilística, e por conveniência, um grupo de 6 voluntários com experiência prévia em desenvolvimento Android. Para cada participante, foi solicitado realizar um conjunto de atividades sequenciais com intuito de coletar dados sobre o uso do *framework* e inferir o grau de facilidade de uso e aprendizado.

6.2 Perfil dos Usuários

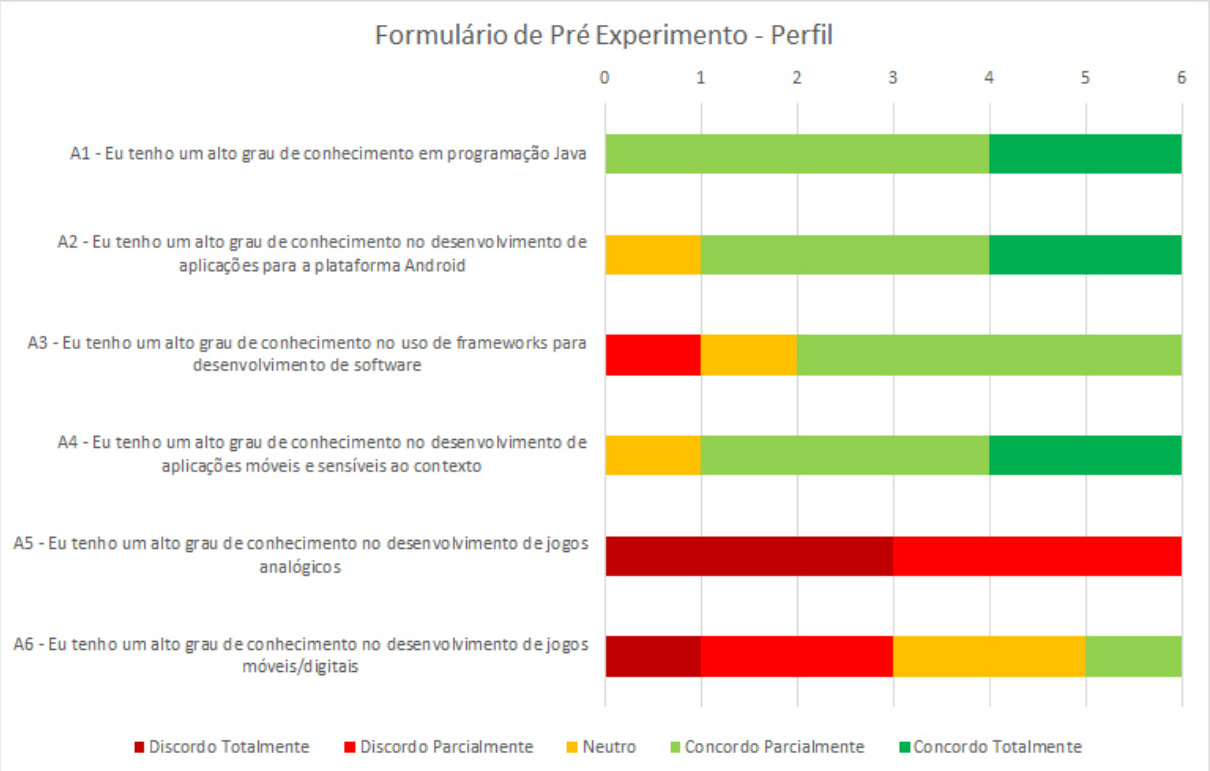
Do perfil dos avaliados, quatro alunos mestrandos e dois doutorandos. Apenas um deles tinha algum conhecimento no desenvolvimento de jogos digitais. A Figura 30 apresenta mais informações sobre o perfil do grupo de avaliados.

Sobre o perfil de programador dos avaliados, todos reportaram ter conhecimento relevante em programação Java e desenvolvimento de aplicações Android. E apenas um deles declarou não ter conhecimento relevante no desenvolvimento de aplicações sensíveis ao contexto.

6.3 Ameaças a Validade

Um das ameaças do experimento é a subjetividade na respostas dos voluntários. Mitigando parte do problema, foi calculado o tempo que cada avaliado levou para realizar as atividades do experimento. Dessa forma é possível comparar o tempo que cada um gastou com as respostas sobre as impressões na utilização do *framework* e se elas estão de acordo com o resultado prático.

Figura 30 – Perfil dos voluntários.

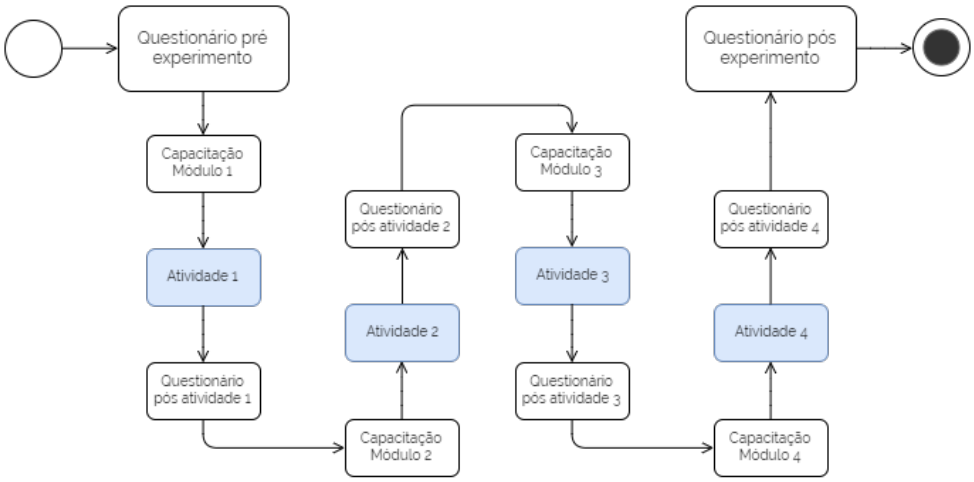


Fonte: o autor.

6.4 Procedimento

Os voluntários realizaram a avaliação de forma individual. Para cada um deles, foi solicitado executar uma sequência de atividades e o preenchimento de formulários de avaliação disponibilizados a partir do *Google Forms*. O tempo em que cada um dos voluntários levou para realizar as tarefas era coletado durante as avaliações.

Figura 31 – Tarefas realizadas no experimento de avaliação.



Fonte: o autor.

Conforme apresentado na Figura 31, cada avaliação consistiu em:

1. Aplicar um formulário pré-experimento
2. Aplicar o experimento dividido em quatro módulos, e cada um com quatro sub-etapas
   a) Capacitação do avaliado sobre uma determinada parte do *framework*
   b) Execução da atividade relacionada
   c) Aplicação de um formulário de acompanhamento da Atividade
3. Aplicar um formulário pós-experimento

A aplicação do experimento foi dividida em quatro módulos. Cada um deles consistia na capacitação sobre as principais funcionalidades do *framework* seguido por uma atividade relacionada às características elencadas no módulo em avaliação. As atividades consistiam em codificar os módulos conforme as instruções. Uma tarefa era considerada como concluída quando o voluntário autodeclarava que as instruções dadas tinham sido codificadas por completo.

O conjunto realizado tinha como objetivo criar uma aplicação simples que fizesse uso dos componentes do *framework*. Após cada atividade, uma parte dessa aplicação era

construída, as instruções eram compiladas e era esperado que o jogo fosse capaz de realizar algumas de suas funcionalidades. Uma vez que a execução fosse correta, o voluntário preenchia o formulário de acompanhamento da atividade antes que fosse dada sequência ao próximo módulo. Em caso de erros, eles eram anotados como relato do experimento.

Ao fim de cada atividade, o comportamento esperado do jogo era: a **atividade 1** deveria reconhecer uma carta e apresentar seu valor na interface; a **atividade 2)** deveria reconhecer uma carta identificável, uma carta interativa e uma carta sensível ao contexto apresentando o resultado baseado na interação realizada na interface; a **atividade 3)** deveria somar o valor das cartas reconhecidas ao estado do jogo e apresentar a soma na tela; e a **atividade 4)** deveria realizar a descoberta de um segundo dispositivo, conectar-se a ele e trocar com ele informações de cartas reconhecidas, devendo eles apresentarem a soma das cartas reconhecidas em ambos os dispositivos.

6.5 Materiais e Instrumentos

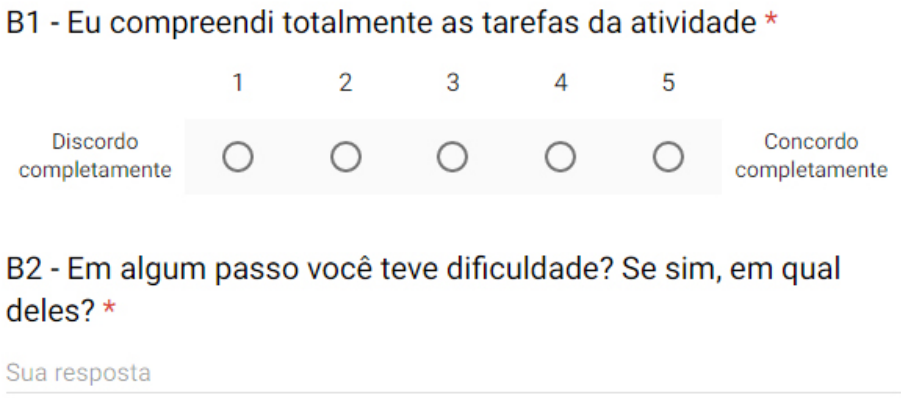
Durante o experimento, foram utilizados seis formulários digitais criados no *Google Forms* para realizar a coleta das informações durante as etapas da avaliação. Os formulários foram: de pré-experimento (Apêndice B), de acompanhamento para cada atividade (totalizando quatro) (Apêndice C) e, por fim, de pós-experimento (Apêndice D).

O formulário de pré-experimento tinha como objetivo coletar informações sobre o perfil dos voluntários. Ao fim de cada atividade, era verificado se o funcionamento ocorreu conforme esperado e em seguida era aplicado um formulário de acompanhamento com o objetivo de saber quanto o voluntário compreendeu da atividade e das dificuldades encontradas. A Figura 32 apresenta uma parte desse formulário.

O formulário pós-experimento tinha como objetivo coletar informações sobre a utilização do *framework* e foi inspirado no *System Usability Scale* (SUS) (BROOKE, 2013). A metodologia já validada permite uma forma de obter resultados relevantes mesmo em uma amostra reduzida. Foram utilizadas sete dos dez itens do questionário. O resultado final foi um formulário com um conjunto de sete afirmações usando a escala de Likert, que o voluntário deve responder “discordo totalmente”, “discordo parcialmente”, “indiferente” ou “neutro”, “concordo parcialmente” e “concordo totalmente”. O objetivo desse formulário é obter uma avaliação preliminar a cerca do uso do *framework*.

Para o experimento, foi criado um documento de capacitação (Apêndice A)

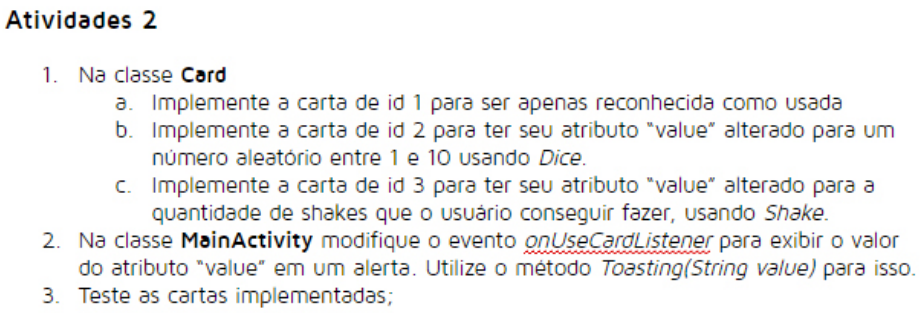
Figura 32 – Questões do formulário de acompanhamento.



Fonte: o autor.

dos voluntários para o uso do *framework*. O documento teve como objetivo apresentar o conceito de jogos de cartas ubíquos e a utilização das funcionalidades do *framework*. O documento divide-se em quatro partes, cada uma representando um módulo da ferramenta. Ao final de cada módulo, uma atividade dava instruções para uma a realização de uma implementação. A Figura 33 apresenta um exemplo de atividade.

Figura 33 – Instrução da atividade 2.



Fonte: o autor.

As atividades foram: instanciar o gerenciamento de cartas e reconhecer uma cartas; implementar o comportamento de cartas identificáveis, interativas e sensíveis ao contexto (Figura 33); implementar e instanciar um estado de jogo; e, por último, instanciar o módulo de descoberta para realizar a comunicação entre dispositivos.

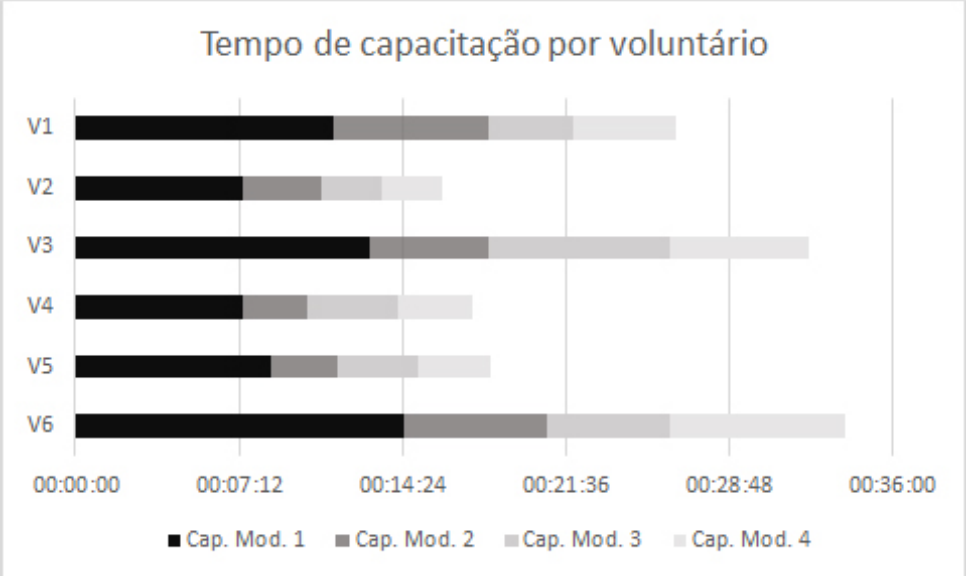
Para implementar as atividades, um ambiente de desenvolvimento no *Android Studio* foi preparado com um projeto base já configurado. Esse projeto consistia no *framework* Tardigrade já implantado com um projeto de interface já pronta. Essa escolha foi feita para focar o experimento apenas no que tangia as funcionalidades do *framework*.

Os voluntários contavam também com um dispositivo *Nexus 5 (D820)* para testar a aplicação ao fim de cada atividade. Na última atividade, um segundo dispositivo era exigido para completar o teste e, para tal, foi utilizado um outro dispositivo idêntico, com uma versão completa e funcional da aplicação já previamente instalada. O dispositivo com a aplicação construída pelo voluntário deveria conectar-se e comunicar-se com este segundo dispositivo para que a atividade fosse concluída com sucesso.

6.6 Resultados e Discussão

O tempo que cada voluntário demorou para se preparar e executar as tarefas deste experimento foi coletado e estão apresentados nas Figuras 34 e 35. A capacitação aconteceu em partes, somando uma média de 24 minutos. Os voluntários tiveram o maior gasto de tempo na capacitação do primeiro módulo (média de 10 minutos), que incluía também uma introdução aos jogos ubíquos, uma visão geral do *framework* para então abordar o módulo de reconhecimento de cartas de forma detalhada. Na capacitação dos demais módulos, a média foi de 4 minutos e meio para cada um dos três módulos finais. Já para implementar as instruções das atividades, os voluntários gastaram em média 11 minutos e meio.

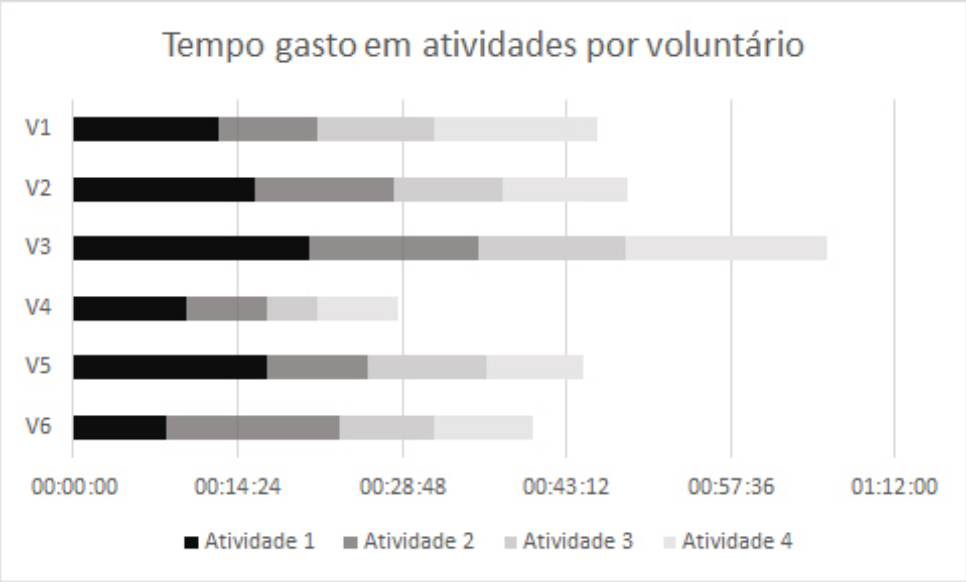
Figura 34 – Tempo para capacitação dos módulos por voluntário.



Fonte: o autor.

O primeiro módulo, que avaliou o funcionamento do reconhecimento de cartas,

Figura 35 – Tempo para implementação das atividades por voluntário.



Fonte: o autor.

foi o único que apresentou problemas durante a execução do experimento. Apesar de cinco dos seis voluntários afirmarem terem compreendido a atividade, o mesmo número apresentou problemas na implementação deste módulo, principalmente na implementação de um atributo na classe `Card`. Já o voluntário 4 não teve dificuldades em implementar a atividade e não apresentou problemas na compilação, apresentando uma marca de menos de 10 minutos na atividade 1. O tempo médio desta atividade foi de 14 minutos.

O segundo módulo tinha como foco os diferentes comportamentos das cartas reconhecidas. A média foi de 11 minutos para que os voluntários codificassem as instruções da atividade 2. Diferente da primeira atividade, os voluntários não tiveram problemas para executar as instruções que implementavam o comportamento das cartas, entretanto os voluntário 2, 3 e 6 tiveram mais dificuldades na abstração das instruções e precisaram de orientações durante esta etapa do experimento, necessitando de mais tempo para concluí-la. A outra metade do grupo levou aproximadamente metade do tempo para finalizar as instruções. Apesar da variação de tempo ter sido alta, todos os voluntários finalizaram a execução das instruções sem nenhuma falha na implementação.

O módulo 3 implementou um estado de jogo e a interação de cartas com esse estado. Com uma média de 9 minutos, os voluntários tiveram sucesso ao implementar todas as instruções sem dificuldades ou problemas de compilação. Nessa atividade, quatro dos voluntários tiveram tempo superior a média de tempo gasto, sendo que a diferença de

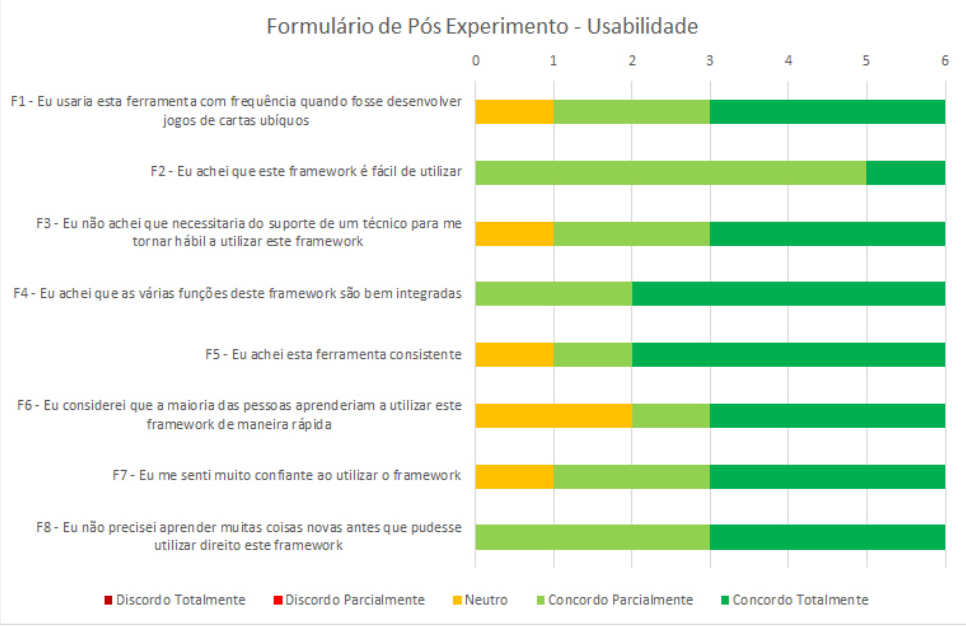
tempo entre esse grupo varia em 3 minutos (entre 9 minutos e 31 segundos e 12 minutos e 49 segundos), com o voluntário 4 apresentando novamente um tempo consideravelmente inferior aos demais (apenas 4 minutos e 28 segundos).

O último e quarto módulo tinha como propósito apresentar a implementação do módulo de comunicação. Neste módulo, os voluntários demonstraram menos dificuldade na compreensão do seu funcionamento e nas instruções dadas. Com uma média de 11 minutos e 13 segundos, todos foram capazes de finalizar a atividade com eficiência.

No geral, os voluntários tiveram um nível razoável de compreensão das atividades. A dificuldade encontrada na atividade 1 foi causada pela grande quantidade de passos para se implementar a estrutura de uma carta (atributos + arquivo de importação) e pela integração da aplicação externa para leitura de QR Code usando o *callback* da `Activity`, `onActivityResult()`.

A Figura 36 apresenta os resultados obtidos através do formulário de avaliação pós-experimento.

Figura 36 – Resultado do questionário pós-experimento.



Fonte: o autor.

As respostas foram, na totalidade, positivas. Os voluntários (50% concordam totalmente, 33% concordam parcialmente) afirmaram que utilizariam o *framework* caso fossem desenvolver jogos de cartas ubíquos. Todos os voluntários afirmaram que o *framework* é de fácil utilização e concordam que a maioria das pessoas poderia aprender

a utilizá-lo de maneira rápida. A maior parte (83.3%) acredita que não necessitaria de um suporte técnico para se tornar hábil a utilizar o *framework*. O *framework* foi dado como consistente pela maior parte dos voluntários (66,7% concordam totalmente e 16,7% concordam parcialmente) e suas várias funções foram julgadas como bem integradas (66.7% concordam totalmente e 33.3% concordam parcialmente).

Dado o feedback recebido e considerando o perfil dos voluntários como desenvolvedores com experiência prévia, pode-se concluir que as funções do *framework* estão bem integradas e o *framework* apresenta uma consistência considerável, podendo ser uma forma válida de desenvolvimento de jogos de cartas ubíquos. Indo além da proposta da avaliação, ainda é possível concluir que existe uma facilidade na utilização das funções do *framework* e que desenvolvedores poderiam utilizar do *framework* para criarem jogos nessa nova categoria.

6.7 Problemas encontrados e limitações da avaliação

O experimento foi conduzido sob situações tendenciosas, como voluntários com vínculos com o aplicador do experimento, o que pode alterar o resultado obtido no formulário pós-experimento, porém, ainda assim, a coleta de informações durante a aplicação foi utilizada para minimizar a contradição das informações fornecidas nesse formulário. O tempo e as dificuldades de cada voluntário tiveram um peso considerável sob a análise dos resultados obtidos.

A quantidade de experimentos realizada é outro fator que limita os resultados obtidos. Com o tempo médio de experimento sendo maior do que 60 minutos, a captação de voluntários para este processo torna-se uma tarefa difícil.

6.8 Conclusões

Neste capítulo foi apresentada a validação das funcionalidades do *framework* Tardigrade. Sobre o experimento, foram apresentados os critérios de escolha dos voluntários e o procedimento de experimentação com o fluxo das atividades realizadas por eles. Também foram apresentados os materiais e instrumentos utilizados na avaliação. Por fim, os resultados foram sumarizados e discutiu-se sobre as conclusões inferidas a partir dos dados obtidos.

Chegou-se a conclusão de que o *framework* Tardigrade foi de fácil utilização pelos voluntários, sem problemas relevantes na instanciação dos recursos disponíveis mesmo por desenvolvedores com menor experiência. Entretanto, o número de voluntários que realizaram o experimento limitam a qualidade dos resultados obtidos.

7 CONCLUSÃO

Neste trabalho, foi apresentado o modelo e o *framework* Tardigrade para o desenvolvimento de jogos de cartas ubíquos. Neste capítulo, estão sumarizados os principais resultados, contribuições e limitações do trabalho desenvolvido nesta dissertação. Também são apresentadas propostas de trabalhos futuros que dão continuidade a esta pesquisa. Na Seção 7.1, estão listados os resultados alcançados. Na Seção 7.2, é apresentada a produção bibliográfica do autor deste trabalho ao longo do mestrado. A Seção 7.3 indica as principais limitações do trabalho. E, por fim, a Seção 7.4 apresenta propostas de trabalhos futuros derivados da pesquisa.

7.1 Resultados Alcançados

O conceito de integrar jogos de cartas com características de sistemas ubíquos (como a sensibilidade ao contexto) abre novas perspectivas para a criação de um universo de jogos de cartas no qual o virtual e o real não tem fronteiras bem definidas e a jogabilidade pode ser expandida. Esta pesquisa de mestrado é um embrião desse conceito e o modelo Tardigrade visa delimitar e identificar alguns elementos chaves para a criação deste tipo de jogos.

O modelo desenvolvido pode ser implementado para diferentes plataformas, sendo possível utilizar os três tipos de cartas especificados nele. Esses tipos de cartas podem ser utilizados para construções de diversos jogos e dão suporte ao *game designer* na fase de concepção dos jogos de cartas ubíquos.

Do ponto de vista do desenvolvedor de jogos, a principal contribuição deste trabalho é o *framework* implementado na plataforma Android. O *framework* Tardigrade implementa o modelo e busca diminuir o esforço de desenvolvimento de jogos de cartas ubíquos. Três tipos de cartas e seus códigos de reconhecimento e execução são fornecidos pelo *framework*, a saber: cartas tangíveis-identificáveis, cartas tangíveis-interativas e cartas tangíveis sensíveis ao contexto. Entre os recursos disponíveis para agilizar o desenvolvimento estão: o gerenciamento em memória das cartas, o reconhecimento das cartas reais, o gerenciamento do estado do jogo, a descoberta e comunicação entre dispositivos. Outros serviços e aplicações podem ser integrados ao *framework* para aumentar as mecânicas dos jogos, como a utilização de serviços de aquisição de contexto e realidade aumentada.

A avaliação do *framework* mostrou que todos os desenvolvedores foram capazes de implementar as funcionalidades do *framework* sem grandes dificuldades, embora tivessem perfis distintos de expertise em Android. O *framework* foi bem aceito por todos como uma forma de agilizar o desenvolvimento jogos de cartas ubíquos. Vale ressaltar, no entanto, que as avaliações não foram aplicadas a desenvolvedores com experiência no desenvolvimento de jogos eletrônicos cuja pouca familiaridade com programação nativa em Android poderia dificultar o uso do *framework*.

A prova de conceito, ou seja, o jogo *Hyper Try*, representa uma contribuição no cenário escasso de jogos de cartas ubíquos. O jogo, publicado na Google Play, atesta que o *framework* pode ser utilizado para desenvolvimento de aplicações autocontidas que não dependem de bibliotecas, aplicações ou preparação no ambiente de execução. A avaliação de jogabilidade conduzida a partir do PLEXQ mostrou que o jogo cumpre seu propósito como ferramenta de entretenimento e proporciona uma experiência de jogo relevante.

7.2 Produção Bibliográfica

Durante o período que se estendeu este trabalho de mestrado, quatro publicações foram aceitas em eventos nacionais e internacionais. Três delas foram publicadas:

- *Card Project Pro: A Serious Card Game to Motivate Project Management.* Aceito inicialmente como *fast abstract* na COMPSAC 2015, não foi publicado por falta de recursos financeiros. O trabalho foi adaptado e publicado como artigo completo no Fórum de Educação e Engenharia de Software (FEES) 2015, apresentando um jogo de cartas de gerenciamento de projetos. Autores: Willian Almeida Rodrigues, Luís Fernando Silva, Fernando Antonio Mota Trinta, Jackson Gomes de Souza e Windson Viana.
- *Cartas Sensíveis ao Contexto: Aumentando a experiência em jogos de cartas ubíquos.* Publicado como artigo completo no Simpósio Brasileiro de Games (SBGames) 2016 na trilha de Arte Design. Com foco no *framework* desenvolvido e suas funcionalidades. Autores: Willian Almeida Rodrigues, Glaudiney Mendonça Júnior e Windson Viana.
- *Auragame: A Case Study of a Zero Programming Augmented Reality Game.* Publicado como artigo completo no Simpósio Brasileiro de Games (SBGames) 2016 na trilha de Arte Design. Apresentando um jogo de realidade aumentada utilizando a ferramenta Aurasma. Autores: Luis Fernando M S Silva, Windson Viana, Fernando

Antonio Mota Trinta, Willian Almeida Rodrigues.

7.3 Limitações

Como limitações, este trabalho apresenta as seguintes questões ainda não resolvidas:

- As avaliações com seis desenvolvedores são ainda insuficientes para indicar quão fácil é a utilização do *framework*. Além disso, o *framework* foi instanciado apenas uma vez para a criação do Jogo *Hiper Try*, sendo necessárias novas implementações de jogos para permitir uma constatação mais adequada da completude do *framework* e também para que mais possibilidades de jogos sejam identificadas;
- O *framework* foi implementado apenas em Android e necessita de uma refatoração de forma a organizar melhor seu código fonte, o intuito é de ter uma versão estável e simplificada para distribuição. Sua documentação ainda é escassa o que pode tornar sua utilização complicada para uma distribuição maior;
- O *framework* não garante a produção de jogos com características ubíquas, sendo seu papel fornecer os recursos aos desenvolvedores para que jogos com tais características sejam desenvolvidos. Por exemplo, o desenvolvedor pode criar um jogo que apenas identifica as cartas por QR Code e toca um áudio associado.

7.4 Trabalhos Futuros

Como exposto anteriormente, esta pesquisa de mestrado é apenas um ponto de partida na investigação do conceito de jogos de cartas ubíquos. Os trabalhos futuros podem ser divididos em duas perspectivas: a do design de jogos e a da implementação de bases tecnológicas de jogos ubíquos.

Na perspectiva do design de jogos de cartas ubíquos, pode-se listar como trabalhos futuros:

- O estudo com designers de jogos de tabuleiro sobre as perspectivas e possíveis usos dos três tipos cartas propostos nesta dissertação.
- O design, implementação e avaliação de novos jogos de cartas ubíquos com intuito de averiguar com mais precisão como o modelo Tardigrade pode ser utilizado para a criação desse tipo de jogos. Uma das possibilidades é tornar jogos de cartas

tradicionais acessíveis à pessoas com deficiência visual;

Na perspectiva tecnológica, pode-se listar:

- Realizar melhorias na estrutura do código-fonte do *framework* e da sua disponibilização em um repositório online. Para isso, será necessário desenvolver uma documentação mais adequada do *framework* e também realizar novos experimentos de avaliação do *framework* com desenvolvedores;
- Criar mais módulos de integração com serviços e APIs disponíveis para aumentar as opções de jogos a serem criados com Tardigrade. Como exemplos, pode-se citar a integração com o Vuforia[1] para permitir o uso de Realidade Aumentada associada às cartas reais e o reconhecimento de cartas com técnicas de processamento de imagens disponível em bibliotecas como a Open CV[2].

[1] Vuforia - https://vuforia.com/
[2] Open CV - https://opencv.org/

REFERÊNCIAS

ABOWD, G. D.; DEY, A. K.; BROWN, P. J.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a better understanding of context and context-awareness. In: **Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing**. London, UK, UK: Springer-Verlag, 1999. (HUC '99), p. 304–307. ISBN 3-540-66550-1. Disponível em: <http://dl.acm.org/citation.cfm?id=647985.743843>.

BJöRK, S.; HOLOPAINEN, J.; LJUNGSTRAND, P.; MANDRYK, R. Special issue on ubiquitous games. **Personal Ubiquitous Comput.**, Springer-Verlag, London, UK, UK, v. 6, n. 5-6, p. 358–361, jan. 2002. ISSN 1617-4909. Disponível em: <http://dx.doi.org/10.1007/s007790200040>.

BOBERG, M.; KARAPANOS, E.; HOLOPAINEN, J.; LUCERO, A. Plexq: Towards a playful experiences questionnaire. In: **Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play**. New York, NY, USA: ACM, 2015. (CHI PLAY '15), p. 381–391. ISBN 978-1-4503-3466-2. Disponível em: <http://doi.acm.org/10.1145/2793107.2793124>.

BROOKE, J. Sus: A retrospective. **J. Usability Studies**, Usability Professionals' Association, Bloomingdale, IL, v. 8, n. 2, p. 29–40, fev. 2013. ISSN 1931-3357. Disponível em: <http://dl.acm.org/citation.cfm?id=2817912.2817913>.

BUZETO, F. N. **Jogos Ubíquos Reconfiguráveis: Da Concepção à Construção**. Tese (Doutorado), 2015.

CHEUNG, G.; LEE, A.; CHENG, K.; LEE, H. J. Dispensable, tweakable, and tangible components: Supporting socially negotiated gameplay. **Games and Culture**, 2013. Disponível em: <http://gac.sagepub.com/content/early/2013/02/25/1555412013496893.abstract>.

COPLIEN, J. O.; ALEXANDER, A. W. O. **Software Patterns**. 1996.

COSTA, C. A. da; YAMIN, A. C.; GEYER, C. F. R. Toward a general software infrastructure for ubiquitous computing. **IEEE Pervasive Computing**, v. 7, n. 1, p. 64–73, Jan 2008. ISSN 1536-1268.

DRUMOND, R. R.; DAMASCENO, A. L. B. d.; NETO, C. d. S. S. Wanda: A framework to develop card based games to help motivate programming students. In: **2014 Brazilian Symposium on Computer Games and Digital Entertainment**. [S.l.: s.n.], 2014. p. 158–164. ISSN 2159-6654.

FILHO, J. de R. M. B. **FRAMESEC: Um framework para a provisão de segurança fim-a-fim para aplicações no ambiente de computação móvel**. Dissertação (Mestrado) — Universidade Federal do Ceará, 2004.

GRAY, M. **Dream Phone**. Grow, 2015. Disponível em: <http://www.lojagrow.com.br/jogo-dream-phone-03100/p>.

INC., T. **Building Object-Oriented Frameworks**. 1993. A Taligent White Paper.

JOHNSON, R. E.; FOOTE, B. Designing reusable classes. **Journal of Object-Oriented Programming**, v. 1, n. 2, p. 22–35, June/July 1988. Disponível em: <http://www.laputan.org/drc.html>.

KOTRY, M. **Alchemists**. Devir, 2014. Disponível em: <https://boardgamegeek.com/boardgame/161970/alchemists/credits>.

MAGALHãES, K. C. P. de. **FRAMEPERSIST: Um Framework de Persistência de Objetos para o Desenvolvimento de Aplicações para Dispositivos Móveis**. Dissertação (Mestrado) — Universidade Federal do Ceará, 2005.

MAHMUD, A. A.; MUBIN, O.; OCTAVIA, J. R.; SHAHID, S.; YEO, L.; MARKOPOULOS, P.; MARTENS, J. B.; ALIAKSEYEU, D. Affective tabletop game: A new gaming experience for children. In: **Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on**. [S.l.: s.n.], 2007. p. 44–51.

MAIA, M. E. F.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. M. C. Loccam - loosely coupled context acquisition middleware. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. (SAC '13), p. 534–541. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480465>.

NIEUWDORP, E. The pervasive discourse: An analysis. **Comput. Entertain.**, ACM, New York, NY, USA, v. 5, n. 2, abr. 2007. ISSN 1544-3574. Disponível em: <http://doi.acm.org/10.1145/1279540.1279553>.

NOLETO, C.; LIMA, M.; MAIA, L. F.; VIANA, W.; TRINTA, F. An authoring tool for location-based mobile games with augmented reality features. In: **SBC – Proceedings of SBGames**. [s.n.], 2015. p. 140–148. Disponível em: <http://www.sbgames.org/sbgames2015/anaispdf/computacao-full/147513.pdf>.

RöMER, K.; DOMNITCHEVA, S. Smart playing cards: A ubiquitous computing game. **Personal Ubiquitous Comput.**, Springer-Verlag, London, UK, UK, v. 6, n. 5-6, p. 371–377, jan. 2002. ISSN 1617-4909. Disponível em: <http://dx.doi.org/10.1007/s007790200042>.

SAKAMOTO, M.; ALEXANDROVA, T.; NAKAJIMA, T. Introducing virtuality to enhance game-related physical artifacts. **International Journal of Smart Home**, v. 8, n. 2, p. 137–152, 2014. ISSN 1975-4094.

SALEN, E. Z. K. **Regras do Jogo - Fundamentos do Design de Jogos**. first. [S.l.]: Blucher, 2012. v. 1-4.

SANTOS, L. H. d. O. **Arcabouço para Construção de Jogos Ubíquos com Foco em Reabilitação**. Dissertação (Mestrado), 2016.

SANTOS, M. da S. **Uma Proposta para a Integração de Modelos de Padrões de Software com Ferramentas de Apoio ao Desenvolvimento de Sistemas**. Dissertação (Mestrado) — Universidade Federal do Ceará, 2004.

SPÍNOLA, R. O.; TRAVASSOS, G. H. Towards a framework to characterize ubiquitous software projects. **Inf. Softw. Technol.**, Butterworth-Heinemann, Newton, MA, USA, v. 54, n. 7, p. 759–785, jul. 2012. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2012.01.009>.

VIANA, W. **Mobility and Context-awareness for Personal Multimedia Management: CoMMediA**. Tese (Theses) — Université Joseph-Fourier - Grenoble I, fev. 2010. Financé par CAPES-Brésil. Disponível em: <https://tel.archives-ouvertes.fr/tel-00499550>.

WEISER, M. The computer for the 21st century. **Scientific American**, v. 265, n. 3, p. 66–75, set. 1991.

YAN, X.; EVAN, B.; IULIAN, R.; MARIBETH, G.; BLAIR, M. Chores are fun: Understanding social play in board games for digital tabletop game design. In: **DiGRA &#3911 - Proceedings of the 2011 DiGRA International Conference: Think Design Play**. DiGRA/Utrecht School of the Arts, 2011. ISBN ISSN 2342-9666. Disponível em: <http://www.digra.org/wp-content/uploads/digital-library/11307.16031.pdf>.

ZENDER, R.; METZLER, R.; LUCKE, U. Freshup—a pervasive educational game for freshmen. **Pervasive and Mobile Computing**, v. 14, p. 47 – 56, 2014. ISSN 1574-1192. Special Issue on Pervasive EducationSpecial Issue on The Social Car: Socially-inspired Mechanisms for Future Mobility Services. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1574119213001168>.

ZHOU, Z.; CHEOK, A. D.; CHAN, T.; LI, Y. Jumanji singapore: An interactive 3d board game turning hollywood fantasy into reality. In: **Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology**. New York, NY, USA: ACM, 2004. (ACE '04), p. 362–363. ISBN 1-58113-882-2. Disponível em: <http://doi.acm.org/10.1145/1067343.1067403>.

APÊNDICE A – DOCUMENTOS DE TREINAMENTO

Avaliação Tardigrade V1 - Página 1

Tardigrade

O Tardigrade é um *framework* Android para desenvolvimento de jogos de cartas ubíquos. Mas afinal o que é um jogo cartas ubíquo?

Jogos ubíquos

Pode-se definir parte dos jogos ubíquos, ou *Ubiquitous Computing Games* (*UC Games*), como aplicações digitais que se misturam com eventos do mundo real transcendendo a experiência do jogo.

Jogos de cartas ubíquos

Definimos como sendo jogos de cartas ubíquo todo jogo com cartas tangíveis que possua interação entre componentes físicos e digitais, seja a parte digital complementar ou mandatória. De tal definição, podemos separar os jogos de cartas ubíquos em dois grupos de aplicações: as aplicações discretas e as aplicações não discretas.

Aplicações discretas

Jogos de cartas podem ser criados ou adaptados para possuírem uma aplicação de jogo cujo o objetivo seja prover acessibilidade. Um bom exemplo é uma aplicação que consegue identificar as cartas e fornecer uma áudio descrição da mesma, tornando o jogo acessível para pessoas com deficiência visual.

Aplicações não discretas

Ao contrário do caso do exemplo anterior, jogos podem depender diretamente da aplicação para funcionarem, normalmente isso porque o dispositivo móvel permite interações que não seriam possíveis sem o seu uso. Um jogo de colheita pode, por exemplo, usar o clima (temperatura e/ou condições climáticas) para modificar o rendimento da safra do jogo

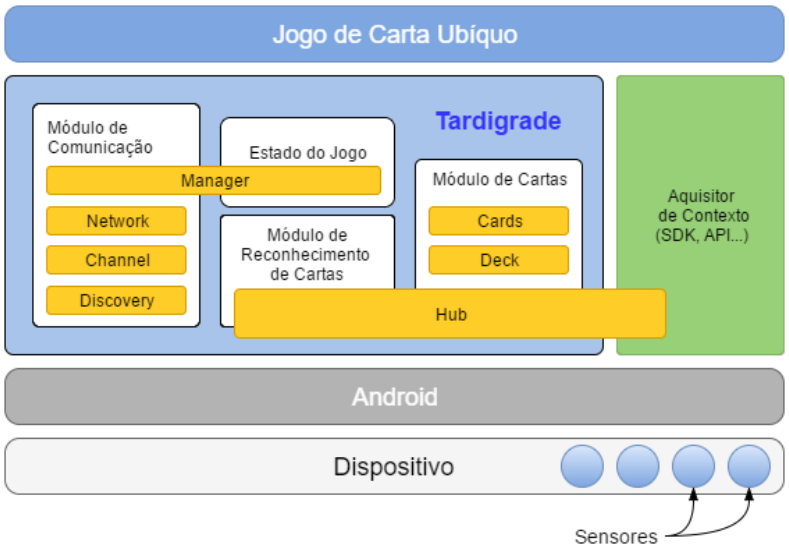
Jogos de cartas ubíquos e Tardigrade

O Tardigrade permite construção de aplicações de jogos para os dois tipos: discretos e não discretos. Para isso, ele implementa três tipos de cartas. 1) A **carta tangível-identificável** é apenas reconhecida pela aplicação, como o caso da aplicação que oferece áudio descrição das cartas. 2) A **carta tangível-interativa** quando reconhecida interage com componentes digitais que podem alterar o comportamento da carta. Um exemplo de carta tangível-interativa são cartas com valores ou comportamentos randômicos. 3) A **carta sensível ao contexto** se assemelha a carta tangível-interativa por ter interações digitais e comportamento não padrão, porém ela se diferencia, principalmente, pela dependência de contexto para obter seu resultado final.

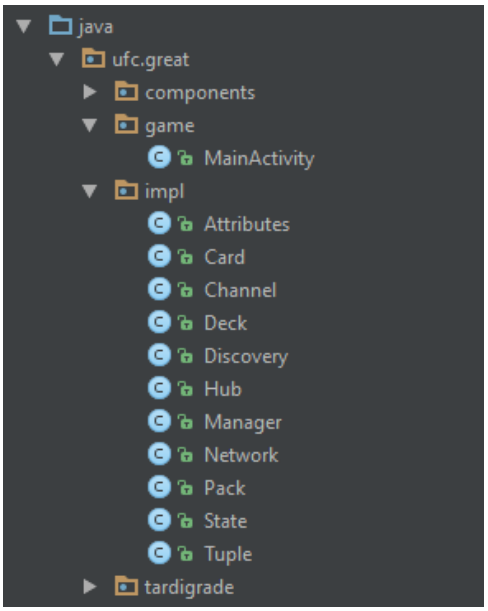
O *Framework*

O *framework* está organizado em 4 módulos:gerenciador de cartas; reconhecimento de cartas; gerenciador de estado de jogo; e comunicação.

Avaliação Tardigrade V1 - Página 2



Estrutura de pastas



A estrutura de pastas de um projeto Android que utiliza o framework está organizado da seguinte forma:

A pasta **components** contém componentes de interface e interação para auxiliar no processo de desenvolvimento de jogo, porém esses não estão diretamente ligados ao framework.

A pasta **game** é a pasta raiz do projeto android e deve conter as atividades e outras classes.

Na pasta **impl** está implementado as funcionalidades do framework, são essas as classes que você deve modificar para criar seu jogo.

A pasta **tardigrade** contém as interfaces do framework, arquivos que não devem ser alterados.

Módulo 1

Gerenciador de cartas

O módulo de gerenciamento de cartas tem a função de gerir todas as cartas do jogo e seus comportamentos. Esse módulo é composto por duas classes: Card e Deck. A classe Card é onde temos os atributos das cartas e o comportamento dela. Já a classe Deck gerencia um conjunto de cartas e o baralho base com as informações de todas as cartas do jogo.

O Hub

O **Hub** é onde todos os métodos de interação estão concentrados. Entre esses métodos de interação, estão disponíveis:

- Ler uma carta por QR Code
- Iniciar um serviço de contexto
- Para um serviço de contexto
- Verificar se o serviço de contexto está funcionando
- E métodos sensíveis ao contexto (Shake, Clima, Iluminação e Random Number)

Utilizando esses o módulo e o Hub vamos para as implementações.

Implementar os atributos da carta do jogo

Para criar novos atributos numa carta devemos realizar 5 passos:

Passo 1. Na classe **Card** declaramos cada atributo como uma constante estica pública como variável global;

```
public final static String ATTR_RED = "redAttr";
```

Passo 2. No construtor da classe **Card** iniciamos a instância *attributes,* que é do tipo Attributes, com as constantes de atributos;

```
attributes = new Attributes(new String []{ ATTR_RED } );
```

Passo 3. Ainda na classe **Card** alteramos o método *Create(...)* para iniciar os atributos baseado nos parâmetros recebidos.

```
card.setAttribute(ATTR_RED, value);
```

Passo 4. Na classe **Deck** referenciamos as constantes criadas na variável *CARD_ATTR*;

```
private static String[] CARD_ATTR = {Card.ATTR_RED};
```

Passo 5. Agora precisamos declarar o novo atributo no arquivo de base de dados de cartas localizado em */assets/cardbase.csv*. vamos modificar para possuir uma nova coluna, o atributo criado (ATTR_RED = redAttr), e o valor desse atributo para cada carta da base.

Avaliação Tardigrade V1 - Página 4

```
uid;name;description;redAttr
1;Carta 1;Sem descrição;0
2;Carta 2;Sem descrição;0
3;Carta 3;Sem descrição;0
```

Usar a classe Deck

Para utilizar o deck de cartas precisamos pegar sua instância e inicializar as informações das cartas. Para isso, seguimos os passos seguintes.

Passo 1. Primeiro declaramos uma variável global do tipo **Deck**;

```
private Deck deck = null;
```

Passo 2. Antes de utilizar a classe precisamos pegar a instância de **Deck** que é sempre única, precisando passar como parâmetro a *Activity* para primeira vez que está sendo utilizado.

```
deck = Deck.getInstance(MainActivity.this);
```

Usar o Hub

O Hub é semelhante ao Deck para ser utilizado.

Passo 1. Primeiro declaramos o **Hub**;

```
private Hub hub = null;
```

Passo 2. Agora podemos pegar a instância do **Hub** passando como parâmetro a *Activity* para a primeira vez que está sendo utilizado.

```
hub = Hub.getInstance(MainActivity.this);
```

Passo 3. O **Hub** pode ser declarado internamente em um método ou como variável Global. É recomendado criar uma instância global dele na ***MainActivity*** para que ele possa ser utilizado facilmente posteriormente.

```
Hub.getInstance(null).<chamada>;
```

Implementar o reconhecimento de uma carta

Para reconhecer a carta usamos o método **Hub**.*readQR*() esse método abre o leitor de QR-Code e retorna o resultado da leitura. Implementamos então um método na *Activity* para recuperar esse valor, em seguida recuperar o objeto da carta e executar a carta recuperada.

Passo 1. Num evento de click de um botão chamamos o método a partir de um Hub já instanciado

```
hub.readQRCard();
```

Passo 2. Implementamos então método onActivityResult para receber o resultado da leitura do QR Code.

```
protected void onActivityResult(
      int requestCode, int resultCode, Intent intent) {

      String result = intent.getStringExtra("SCAN_RESULT");
}
```

Passo 3. Pegar as informações da carta pelo Id

```
iCard card = deck.getCard(result);
```

Passo 4. Executar a carta

```
card.execute();
```

Resultado final:

```
protected void onActivityResult(
      int requestCode, int resultCode, Intent intent) {

      if (requestCode == 0) {
            if (resultCode == RESULT_OK) {
                  String result = intent.getStringExtra("SCAN_RESULT");
                  iCard card = deck.getCard(result);
                  card.execute();
            }
      }
}
```

Apresentar informações da carta lida

Para apresentar as informações das cartas lidas precisamos implementar o evento *onUseCardListener(...)* na classe Deck. Esse evento é chamado sempre ao fim da execução de uma carta para avisar que ela foi utilizada. Em um passo-a-passo devemos implementar um **iCallback** como parâmetro para esse evento onde ele recebe um objeto do tipo **Pack** que contem uma **iCard** como valor.

Passo 1. Primeiro implementamos o **iCallback** no evento *onUseCardListener*() no **Deck**

```
deck.setOnUseCardListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
      }
});
```

Passo 2. Agora podemos pegar o valor recebido no evento

```
            iCard card = (iCard) pack.getValue();
```

Passo 3. E usar a variável do tipo **iCard** e suas propriedades e usa-las no jogo

```
            String name = card.getName();
```

Resultado final:

```
deck.setOnUseCardListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            iCard card = (iCard) pack.getValue();
            String name = card.getName();
      }
});
```

Tarefa A

1. Implemente a classe **Card** para possuir o atributo "value"
2. Na **MainActivity** declare uma variável global das classes
    a. Deck
    b. Hub
3. No *onCreate* da **MainActivity**, pegue as intâncias de Deck e Hub
4. No *onCreate* da **MainActivity**, na ação do botão chame a função da instância do Hub para ler a carta por QR Code;
5. Na **MainActivity** implemente o reconhecimento da carta lida usando QRCode.
6. No *onCreate* da **MainActivity**, implemente na instância da classe **Deck** o evento *onUseCardListener* para mostrar um alerta com o nome da carta reconhecida. Utilize o método *Toasting(String value)* para isso.
7. Teste a o reconhecimento da carta com ID 1.

Avaliação Tardigrade V1 - Página 7

Módulo 2

Implementar e usar uma carta

Para implementar a função de um carta.

Passo 1. Precisamos apenas criar a função da carta modificando o método *execute()* na classe **Card**. A função da carta depende do id dela.

```
if(getId().equals("a"){
      Deck.getInstance(null).useCard(this);
}
```

Implementar e usar uma carta interativa ou sensível ao contexto

Cartas interativas podem ter seus atributos gerados de forma virtual, como pelo método que gera valores aleatórios. Para implementar uma carta assim seguimos o passo a passo a baixo.

Passo 1. Primeiro passo é declarar uma instância do Hub dentro do *execute* da classe **Card**. O Hub precisa já ter sido iniciado previamente na *MainActivity*, dessa forma não precisamos passar a *Activity* como parametro.

```
Hub hub = Hub.getInstance(null);
```

Passo 2. Agora vamos identificar a carta pelo id

```
if(getId().equals("b"){ }
```

Passo 3. Vamos agora implementar, dentro do if, um evento de captura de dados do tipo dice (que simula lançamento de dados)

```
hub.Dice(1, 10, 1, new StatusListener() {
      @Override
      public void onStart(Object object) {    }
      @Override
      public void onUpdate(Object object) {
            int [] values = (int[]) object; // valores gerados
      }
      @Override
      public void onEnd(Object object) {
            Deck.getInstance(null).useCard(Card.this);
      }
});
```

Passo 4. O ultimo passo é alterar o atributo baseado no valor gerado, então acrescentamos no método onUpdate

```
setAttribute(Card.ATTR_RED, values[0]+"");
```

Resultado final:

Avaliação Tardigrade V1 - Página 8

```
public void execute(){
      Hub hub = Hub.getInstance(null);
      if(getId().equals("b"){
            hub.Dice(1, 10, 1, new StatusListener() {
                  @Override
                  public void onStart(Object object) {    }
                  @Override
                  public void onUpdate(Object object) {
                        int [] values = (int[]) object; // valores gerados
                        setAttribute(Card.ATTR_RED, values[0]+"");
                  }
                  @Override
                  public void onEnd(Object object) {
                        Deck.getInstance(null).useCard(Card.this);
                  }
            });
      }
}
```

Pegando um atributo de uma carta

Para pegar o atributo de uma carta e utilizar em seu jogo basta seguir os passos:

Passo 1. No evento que reconhece que uma carta foi usada recupere o valor do atributo a partir do objeto da carta.

```
card.getAttribute(Card.ATTR_RED);
```

Exemplo de utilização:

```
deck.setOnUseCardListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            iCard card = (iCard) pack.getValue();
            String redattr = card.getAttribute(Card.ATTR_RED);
      }
});
```

Detalhamento das funções interativas e sensíveis ao contexto

Shake

O shake detecta quantas vezes o dispositivo é chacoalhado. Seu callback vai receber os seguintes valores:

- `onStart(Object data)` - `data = null`
- `onUpdate(Object data)` - `data = numero de shakes (int)`
- `onEnd(Object result)` - `data = null`

Utilização:

Avaliação Tardigrade V1 - Página 9

```
hub.Shake(new StatusListener() {
      @Override public void onStart(Object object) { }
      @Override
      public void onUpdate(Object object) {
            int value = (int) object;
            setAttribute(Card.ATTR_RED, value + "");
            }
      @Override
      public void onEnd(Object object) {
            Deck.getInstance(null).useCard(Card.this);
      }
});
```

Dice

O dice gera números aleatórios. Seus parâmetros são:

- *min* (int) - valor mínimo a ser gerado
- *max* (int) - valor máximo a ser gerado
- *times* (int) - quantos valores serão gerados
- callback (StatusListener) - implementação de um Status Listener

Seu callback vai receber os seguintes valores:

- onStart(Object data) - data = *null*
- onUpdate(Object data) - data = lista de valores (*int [times]*)
- onEnd(Object result) - data = *null*

Utilização:

```
hub.Dice(0, 10, 1, new StatusListener() {
      @Override public void onStart(Object object) {    }
      @Override
      public void onUpdate(Object object) {
            int [] values = (int[]) object;
            setAttribute(Card.ATTR_RED, values[0]+"");
      }
      @Override
      public void onEnd(Object object) {
            Deck.getInstance(null).useCard(Card.this);
      }
});
```

Atividades 2

1. Na classe **Card**
    a. Implemente a carta de id 1 para ser apenas reconhecida como usada
    b. Implemente a carta de id 2 para ter seu atributo "value" alterado para um número aleatório entre 1 e 10 usando *Dice*.
    c. Implemente a carta de id 3 para ter seu atributo "value" alterado para a quantidade de shakes que o usuário conseguir fazer, usando *Shake*.
2. Na classe **MainActivity** modifique o evento *onUseCardListener* para exibir o valor do atributo "value" em um alerta. Utilize o método *Toasting(String value)* para isso.
3. Teste as cartas implementadas;

Avaliação Tardigrade V1 - Página 10

Módulo 3

Estado do jogo

O gerenciador de estado de jogo contem a classe de estado do jogo, **State**, e esta associada à classe **Manager** do módulo de comunicação. Em resumo, o estado contem o conjunto de operações que determinam as regras e ações do jogo criado.

A classe de estado possui 3 eventos:

- `OnChangeListener - quando o estado é alterado pelo jogador`
- `OnUpdateListener - quando o estado recebe alteração de outro jogador`
- `OnMakeListener - quando alguma mudança foi processada no estado`

Criando estado do jogo

Como exemplo vamos criar um estado de jogo que armazena um valor inteiro.

Passo 1. Na classe **State** instanciamos uma variável global do tipo inteiro chamada *soma*.

```
int sum = 0;
```

Modificando o estado do jogo

Vamos criar um estado do jogo que soma o valor de todas as cartas utilizadas.

Passo 1. Criar um método que soma um valor recebido na variável *soma* com um boleano para sabermos se é uma atualização (true) ou uma alteração (false)

```
public void sum(int value, boolean update){
      sum += value;
}
```

Passo 2. Nesse método precisamos declarar que houve uma mudança no estado do jogo

Passo a. Quando a atualização é feita pelo próprio jogador

```
onChangeListener.callingBack(Pack.create(Flag.UPDATE, value));
```

Passo b. Quando a atualização foi feita por outro jogador e está sendo aplicada ao estado

```
onUpdateListener.callingBack(Pack.create(Flag.UPDATE, value));
```

Passo 3. E também avisar que algo foi processado pelo estado do jogo (para alterar na interface)

```
onMakeListener.callingBack(Pack.create(Flag.RESULT, sum));
```

Resultado final:

Avaliação Tardigrade V1 - Página 11

```
public void sum(int value, boolean update){
	sum += value;
	if(update){
		// Foi uma atualização vinda de outro jogador
		onUpdateListener.callingBack(Pack.create(Flag.UPDATE, value));
	}else{
		// Atualização feita pelo próprio jogador
		onChangeListener.callingBack(Pack.create(Flag.UPDATE, value));
	}
	// Alteração do estado de forma geral
	onMakeListener.callingBack(Pack.create(Flag.RESULT, sum));
}
```

Juntando a carta com o estado do jogo

Para fazer com que os valores da carta utilizada altere o estado do jogo:

Passo 1. Pegue a instância da classe **State**

```
State state = State.getInstance();
```

Passo 2. Altere o evento de reconhecimento de cartas para alterar chamar o método de soma

```
deck.setOnUseCardListener(new iCallback() {
	@Override
	public void callingBack(iPack pack) {
		iCard card = (iCard) pack.getValue();
		String redattr = card.getAttribute(Card.ATTR_RED);

		// Somando atributo ATTR_RED da carta
		state.sum(Integer.parseInt(redattr), false);
	}
});
```

Passo 3. O último passo é implementar o evento *onMakeListener* na instância de **State** para interagir com a interface sempre que o estado do jogo for alterado usando o valor da soma que o evento recebe do callback

```
state.setOnMakeListener(new iCallback() {
	@Override
	public void callingBack(iPack pack) {
		int sum = (int) pack.getValue();
	}
});
```

Atividades C

1. Na classe **State** crie uma variável do tipo inteiro chamada *sum*
2. Implemente na classe **State** um método chamado *Sum* que receba como parâmetro um inteiro e um boleano
   a. O método deve somar o valor inteiro recebido a variável *sum*
   b. O método deve usar o boleano para disparar o evento de mudança no estado:

Avaliação Tardigrade V1 - Página 12

   i. se *verdadeiro* deve ser chamado o *OnUpdateListener* e passar o valor inteiro que foi somado como parâmetro
   ii. se *falso* deve ser chamado o *OnChangeListener* e passar o valor valor inteiro que foi somado como parâmetro
c. O método deve chamar o evento OnMakeListener no final passando o valor da variável *sum* como parâmetro.

3. Na **MainActivity** declare uma variável global da classe **State**
4. No *onCreate* da **MainActivity** pegue a instância da classe **State**
5. No *onCreate* da **MainActivity**, implemente o evento *OnMakeListener* na classe **State** para exibir o valor alterado no estado. Utilize o método *changeData(String value)* para exibir o valor.
6. Modifique o evento *onUseCardListener* para chamar o método *Sum* da classe **State** passando o atributo "*value*" da carta como parâmetro e *false* no segundo parâmetro (atualização feita pelo próprio jogador)
7. Teste com as cartas agora

Avaliação Tardigrade V1 - Página 13

Módulo 4

O módulo de comunicação é composto por: classe **Network**, que é a sua rede de comunicação que recebe informações de outros dispositivos; classe **Channel** que é o canal de comunicação com a **Network** de outros dispositivos; classe **Discovery** que encontra outros dispositivos na mesma rede; e a classe **Manager** que gerencia a comunicação para troca de estado de jogo. Vamos supor que no nosso jogo, os jogadores disputam um cabo de guerra, ou seja: toda carta que o jogador A jogar, vai aumentar o valor do seu estado do jogo, e toda carta que o jogador B jogar vai subtrair do estado do jogo do jogador A. Para isso vamos precisar trocar informações entre os jogadores.

Um Pack

Um objeto do tipo **Pack** é serializar informações para enviar internamente no framework e externamente via rede pelo canal de comunicação. Um objeto desse tipo possui uma *Key* e um *Value*. A *Key* é um descritor sobre como tratar a informação do pacote. O *Value* são as informações e dados transmitidas no pacote.

Passo 1. Para criar um objeto do tipo **Pack** você precisa chamar o criador da classe e passar dois parâmetros: A Key é uma String e o *value* é um *Object*.
```
iPack pack = Pack.create(key, value);
```
Passo 2. Para pegar a *Key* de um objeto do tipo **Pack** basta usar o get
```
pack.getKey();
```
Passo 3. Para pegar o *Value* de um objeto do tipo **Pack** basta usar o get e fazer o cast para o tipo do objeto esperado.
```
(int) pack.getValue();
```

Usando a classe Network

Para usarmos a classe **Network** para receber pacotes de outros dispositivos precisamos instanciar e implementar o evento de recebimento de pacotes.

Passo 1. Pegar a instância da classe **Network**
```
Network network = Network.getInstance();
```

Passo 2. Implementar o evento de recebimento
```
network.setOnReceiveListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            Flag flag = pack.getKey(); // Descrição do pacote
            Object value = pack.getValue; // Pacote
      }
});
```

Enviando um pacote

Para enviar um pacote precisamos ter um objeto do tipo **Channel** já instanciado. Para enviar o pacote basta usar o método de envio de pacotes.

Passo 1. Na instância de **Channel** invoque o método send

```
channel.Send(pack);
```

Usando o manager

O manager permite que dois ou mais dispositivos fiquem conectados e recebem pacotes de dados de forma simultânea. Para usá-lo fazemos:

Passo 1. Pegar a instância do Manager

```
Manager manager = Manager.getInstance();
```

Passo 2. Registrar objetos do tipo **Channel** como observador, supondo que já temos um objeto do típo **Channel** na variável *channel*

```
manager.registerObserver(channel);
```

Passo 3. Notificar todos os observadores quando houver alguma alteração no estado atraves de um objeto do tipo **Pack**, supondo que já temos um objeto do típo **Pack** na variável *pack*

```
manager.notifyObserver(pack);
```

Exemplo de notificação de observadores quando o estado do jogo muda:

```
state.setOnChangeListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            manager.notifyObserver(pack);
      }
});
```

Serviço de descoberta

O framework conta com um serviço de descoberta já implementado, para utilizar suas funcionalidades basta implementar seus eventos, são esses:

- OnStartListener - quando inicia a descoberta
- OnStopListener - quando para a descoberta
- OnFoundListener - quando encontra um dispositivo na rede
    - Retorna um **Channel** com informações do dispositivo encontrado
- OnLostListener - quando perde contato com um dispositivo
    - Retorna um **Channel** com informações do dispositivo perdido

Para usar o serviço de descoberta precisamos primeiro iniciá-lo.

Passo 1. Pegar a instância do serviço de descoberta

```
Discovery discovery = Discovery.getInstance();
```

Passo 2. Inicializar o serviço de descoberta passando como parâmetro a **Activity** e a instância de **Network**

```
discovery.init(MainActivity.this, network);
```

Passo 3. Startar o serviço de descoberta

```
discovery.start();
```

Passo 4. Parar o serviço de descoberta

```
discovery.stop();
```

Implementando o serviço de descoberta

Para usar o serviço de descoberta precisamos implementar os seus eventos:

Passo 1. Implementar o evento disparado quando a descoberta é startada

```
discovery.setOnStartListener(new iCallback() {
	@Override public void callingBack(iPack pack) {
		// O que acontece quando iniciar a descoberta?
	}
});
```

Passo 2. Implementar o evento disparado quando a descoberta é parada

```
discovery.setOnStopListener(new iCallback() {
	@Override public void callingBack(iPack pack) {
		// O que acontece quando parar a descoberta?
	}
});
```

Passo 3. Implementar o evento disparado quando um dispositivo é encontrado

```
discovery.setOnFoundListener(new iCallback() {
	@Override public void callingBack(iPack pack) {
		iChannel channel = (iChannel) pack.getValue();
		// O que acontece quando encontrar um dispositivo?
	}
});
```

Passo 4. Implementar o evento disparado quando um dispositivo é perdido

```
discovery.setOnLostListener(new iCallback() {
	@Override public void callingBack(iPack pack) {
		iChannel channel = (iChannel) pack.getValue();
		// O que acontece quando perder um dispositivo?
	}
});
```

Exemplo ligando a descoberta ao manager:

```
discovery.setOnFoundListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            iChannel channel = (iChannel) pack.getValue();
            Manager.getInstance().registerObserver(channel);
      }
});
discovery.setOnLostListener(new iCallback() {
      @Override
      public void callingBack(iPack pack) {
            iChannel channel = (iChannel) pack.getValue();
            Manager.getInstance().removeObserver(channel);
      }
});
```

Atividades D

1. Na **MainActivity** declare como variável global:
    a. Um objeto do tipo **Network**
    b. Um objeto do tipo **Discovery**
    c. Um objeto do tipo **Manager**
2. No *onCreate* da classe **MainActivity**
    a. Pegue a instância de **Network**
    b. Pegue a instância de **Discovery** e faça a sua inicialização
    c. Pegue a instância do **Manager**
    d. Implemente o evento de *onFound* para registrar dispositivos encontrados como observadores no **Manager**
    e. Implemente o evento *onLost* para remover dispositivos perdidos do **Manager**
    f. Modifique a ação do Switch para ativar o serviço de descoberta quando ele estiver com o *checked = true* e para parar o serviço quando *checked = false*
    g. Implemente o evento *onReceiver* da **Network** para somar o valor recebido ao estado, usando o mesmo método *Sum* porém passando um *true* no segundo parâmetro (atualização vinda de outro jogador)
    h. Modifique o evento *OnChange* da classe **State** para notificar todos os observadores registados no **Manager**
3. Usando dois dispositivos, teste as cartas

APÊNDICE B – QUESTIONÁRIO PRÉ EXPERIMENTO

Fase 1 - Pré-experimento

Questionário pré-experimento do estudo de caso para avaliação do Tardigrade.

*Obrigatório

1. **Nome** *

2. **Grau de escolaridade** *
   Marcar apenas uma oval.
   - ◯ Graduando
   - ◯ Graduado
   - ◯ Cursando Especialização
   - ◯ Especialista
   - ◯ Mestrando
   - ◯ Mestre
   - ◯ Doutorando
   - ◯ Doutor

Sobre desenvolvimento de aplicações móveis

3. **A1 - Eu tenho um alto grau de conhecimento em programação Java** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

4. **A2 - Eu tenho um alto grau de conhecimento no desenvolvimento de aplicações para a plataforma Android** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

5. **A3 - Eu tenho um alto grau de conhecimento no uso de frameworks para desenvolvimento de software** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

6. **A4 - Eu tenho um alto grau de conhecimento no desenvolvimento de aplicações móveis e sensíveis ao contexto** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

Sobre desenvolvimento de Jogos

7. **A5 - Eu tenho um alto grau de conhecimento no desenvolvimento de jogos analógicos** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

8. **A6 - Eu tenho um alto grau de conhecimento no desenvolvimento de jogos móveis/digitais** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

Powered by
Google Forms

APÊNDICE C – QUESTIONÁRIO DE ATIVIDADE

Atividade

*Obrigatório

1. **Nome** *

.....

2. **Eu compreendi totalmente as tarefas da atividade** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

3. **Em algum passo você teve dificuldade? Se sim, em qual deles?** *

.....

.....

.....

.....

4. **Alguma opinião/crítica/comentário a fazer sobre a atividade?**

.....

.....

.....

.....

Powered by
Google Forms

APÊNDICE D – QUESTIONÁRIO PÓS EXPERIMENTO

Fase 6 - Pós-experimento

Questionário pós-experimento do estudo de caso para avaliação do Tardigrade.

*Obrigatório

1. **Nome** *

Sobre a utilização do framework

2. **F1 - Eu usaria esta ferramenta com frequência quando fosse desenvolver jogos de cartas ubíquos** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

3. **F2 - Eu achei que este framework é fácil de utilizar** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

4. **F3 - Eu não achei que necessitaria do suporte de um técnico para me tornar hábil a utilizar este framework** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

5. **F4 - Eu achei que as várias funções deste framework são bem integradas** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

6. **F5 - Eu achei esta ferramenta consistente** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

7. **F6 - Eu considerei que a maioria das pessoas aprenderiam a utilizar este framework de maneira rápida** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

8. **F7 - Eu me senti muito confiante ao utilizar o framework** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

9. **F8 - Eu não precisei aprender muitas coisas novas antes que pudesse utilizar direito este framework** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Concordo completamente

10. **F9 - Você tem alguma sugestão, elogio ou crítica em relação ao framework como um todo?** *

.....

.....

.....

.....

Powered by
Google Forms