**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**DEPARTAMENTO ACADÊMICO DE INFORMÁTICA**
**CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**GABRIEL JOSÉ LAZARINE**
**HENRIQUE LUIZ GRANATO DE QUADROS DO NASCIMENTO**

# DESENVOLVIMENTO DE UM SISTEMA MÓVEL PARA INTEGRAÇÃO DE USUÁRIOS A UM AMBIENTE DE CIDADE VIRTUAL

**TRABALHO DE CONCLUSÃO DE CURSO**

**CURITIBA**
**2017**

GABRIEL JOSÉ LAZARINE
HENRIQUE LUIZ GRANATO DE QUADROS DO NASCIMENTO

# DESENVOLVIMENTO DE UM SISTEMA MÓVEL PARA INTEGRAÇÃO DE USUÁRIOS A UM AMBIENTE DE CIDADE VIRTUAL

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná, Câmpus Curitiba, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Paulo Cezar Stadzisz

CURITIBA

2017

Ministério da Educação
**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**Câmpus Curitiba**
Diretoria de Graduação e Educação Profissional
***Departamento Acadêmico de Informática***
***Coordenação do Curso de Bacharelado em Sistemas de Informação***

## TERMO DE APROVAÇÃO

# "DESENVOLVIMENTO DE UM SISTEMA MÓVEL PARA INTEGRAÇÃO DE USUÁRIOS A UM AMBIENTE DE CIDADE VIRTUAL

por

## "Gabriel José Lazarine

## Henrique Luiz Granato de Quadros do Nascimento"

Este Trabalho de Conclusão de Curso foi apresentado no dia 06 de Julho de 2017 como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho _____.

| | |
|---|---|
| _____<br>**Prof. Paulo Cezar Stadizisz**<br>(Presidente - UTFPR/Curitiba) | _____<br>**Prof. João Alberto Fabro**<br>(Avaliador 1 – UTFPR/Curitiba) |
| _____<br>**Prof. Robson Ribeiro Linhares**<br>(Avaliador 2 – UTFPR/Curitiba) | _____<br>**Prof. Leyza Baldo Dorini**<br>(Professora Responsável pelo TCC – UTFPR/Curitiba) |
| _____<br>**Prof. Leonelo Dell Anhol Almeida**<br>(Coordenador(a) do curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba) | |

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso."

# RESUMO

LAZARINE, Gabriel, NASCIMENTO, Henrique, Desenvolvimento de um Sistema Móvel para Integração de Usuários em um Ambiente de Cidade Virtual. XX f. TCC (Curso de Sistemas de Informação), Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Este trabalho teve como objetivo o desenvolvimento de um sistema móvel, que possibilitasse a integração com um ambiente de cidade virtual já existente desenvolvido na UTFPR. A aplicação foi desenvolvida através do Unity 3D, e teve como foco principal de suas funcionalidades criar a possibilidade de interação entre usuários do ambiente previamente construído para desktop com usuários do ambiente mobile através de mensagens de texto em um chat. Assim como a possibilidade de movimentação dentro deste ambiente através da localização real do jogador, utilizando o GPS do *smartphone* para tal.

**Palavras-chave:** Cidades Inteligentes. Cidades Virtuais. Ciberespaço.

## ABSTRACT

LAZARINE, Gabriel, NASCIMENTO, Henrique, Mobile System Development for Integration of Users in an Virtual City Environment. XX f. TCC (Course of Information Systems) - Federal University of Technology - Paraná. Curitiba, 2017.

This work had the objective of developing a mobile system that could be integrated with an already existing virtual city environment developed at the UTFPR. The application was developed using Unity 3D, and had the main focus of its functionalities on creating the possibility of interaction between users of the desktop solution with users of the mobile app, thought text messages in a chat client. Just like the possibility of moving inside this virtual environment using the player's real position.



**Keywords:** Smart Cities. Virtual Cities. Cyberspace

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE SIGLAS E ACRÔNIMOS

| | |
|---|---|
| CWB-VP | Curitiba *Viewport* |
| CWB-VP-MOB | Curitiba *Viewport Mobile* |
| e-gov | Governo Eletrônico |
| GPS | Sistema de Posicionamento Global |
| IDC | *International Data Corporation* |
| IDE | Ambiente de Desenvolvimento Integrado |
| MR | *Mixed Reality* |
| ONG | Organização Não Governamental |
| PC | *Personal Computer* |
| PS3 | *Playstation* 3 |
| PS4 | *Playstation* 4 |
| RFA | Requisito Funcional da Aplicação |
| RFP | Requisito Funcional da Plataforma |
| RNFA | Requisito Não Funcional de Ambiente |
| RNFE | Requisito Não Funcional Econômico |
| RNFES | Requisito Não Funcional de Estima |
| RNFI | Requisito Não Funcional de Interface |
| RNFP | Requisito Não Funcional da Plataforma |
| URBS | Urbanização de Curitiba S/A |
| UTFPR | Universidade Tecnológica Federal do Paraná |

**SUMÁRIO**

# 1 INTRODUÇÃO

As cidades virtuais são implementações digitais de cidades reais ou fictícias, principalmente na forma de softwares de simulação (Miranda, 2012). Elas se aplicam tanto para entretenimento, como na tentativa de aproveitar o potencial das novas tecnologias de comunicação para apoiar a evolução e manutenção das cidades reais (Fernandes & Gama, 2006).

Muitos projetos e aplicações de cidades virtuais têm sido desenvolvidos e empregados para diferentes fins. Um exemplo disso são os Tour Virtuais que podem ser feitos por diversos países através da internet (Stockholm 360). O propósito, neste caso, é permitir o turismo virtual da cidade, visando motivar os viajantes a conhecerem e, eventualmente, irem visitar pessoalmente as cidades reais. Pode-se visitar construções, locais históricos e, até mesmo, fazer passeios por museus.

A modelagem de cidades virtuais, representando locais e estruturas reais, ainda está longe do realismo do mundo real, na maioria dos casos. Esta falta de realismo está associada com os propósitos de se construir uma cidade virtual. Cidades virtuais para entretenimento, por exemplo, não precisam de um realismo extremo, pois o foco está na aventura e jogabilidade. Cidades virtuais para simulações visando o gerenciamento da cidade, também não precisam de um grande realismo gráfico e sim de georreferenciamento e outros modelos numéricos de análise. Por outro lado, quando pretende-se que o usuário de uma cidade virtual possa observar e se envolver com a cidade, o grau de realismo torna-se muito relevante (Fernandes & Fernandes, 2006).

A construção de uma cidade virtual com alto grau de realismo implica um alto custo de desenvolvimento e de execução. Para desenhar e atualizar (i.e., *rendering* ou renderização) continuamente os cenários gráficos com alta qualidade, exige-se uma grande carga de processamento. No caso de desenho de uma cidade inteira, esta atividade pode exigir um processamento imenso.

Um dos modelos mais simples de cidade virtual envolve ambientes de simulação com imersão limitada do usuário. Neste caso, a cidade possui cenário padrão e previsível. O usuário imerge no ambiente por meio de um personagem e navega neste ambiente em busca de objetivos isoladamente como por exemplo a visita ao Louvre e jogos de campanha.

Outro modelo mais complexo de cidades virtuais são aquelas que envolvem múltiplos

usuários interagindo simultaneamente. Os cenários são dinâmicos e mutáveis, com o envolvimento de diferentes usuários. O usuário imerge no ambiente e interage com o ambiente e demais usuários. Há uma sensação de realismo, pois o comportamento dos personagens é realístico e os jogadores interagem tanto com o ambiente quanto com outros jogadores.

Finalmente, as cidades virtuais mais complexas são aquelas nas quais existe sincronização com o ambiente real. Dependendo de suas utilidades, o grau de sincronismo com o mundo real é maior, assim como a qualidade gráfica de seu ambiente. No caso de aprendizagem com realidade virtual imersiva do *Immerse Learning*, o grau de realismo gráfico é baixo, mas o nível de movimentação e interação com os procedimentos e com os instrutores é muito alto.

Com o grande avanço da tecnologia promovendo os computadores que são possíveis usar na forma de vestimenta, como o Google Glass, e o desenvolvimento da internet das coisas, relacionando os objetos e utensílios em uma rede unificada de comunicação, é possível começar a pensar em estender a experiência como a dos museus para a cidade como um todo em uma experimentação real de uso.

Para interligarmos toda uma cidade virtual, sendo essa a réplica da cidade verdadeira, com a cidade real, pode ser aplicado o conceito de *Mixed Reality* (MR – ou Realidade Misturada), no qual a realidade é replicada para o virtual, assim como o virtual é colocado com o mundo real (Ohta & Tamura, 2014). A MR trabalha com a interação do usuário de maneiras diferentes das tradicionais, de forma que o usuário pode visualizar os objetos do mundo virtual no mundo real e interagir sem distinção entre eles, como pode ser visto com as *HoloLens* da Microsoft, tecnologia que utiliza um óculos para projetar hologramas que interagem com o mundo real e podem ser manipulados da maneira que o usuário desejar (Ohta & Tamura, 2014).

Ainda assim, mesmo com a tecnologia necessária, é preciso pensar também em questões não tecnológicas. É necessário levar em conta a maneira com que essa tecnologia irá impactar nas questões sociais, financeiras, jurídicas, de segurança e todas as outras que existem na sociedade.

# 2 OBJETIVOS E JUSTIFICATIVAS

Este capítulo apresenta os objetivos geral e específicos deste trabalho e as justificativas que motivaram seu desenvolvimento.

## 2.1 OBJETIVO GERAL

O objetivo geral deste trabalho de conclusão de curso é especificar e desenvolver um sistema para integrar usuários a um ambiente de cidade virtual por meio de dispositivos móveis permitindo o rastreamento e comunicação por *chat* entre os usuários.

Este objetivo se insere em um projeto de pesquisa maior conduzido na UTFPR sob a coordenação do professor Dr. Paulo Cézar Stadzisz e que se propõe a conceber e desenvolver um sistema computacional para uma cidade virtual denominado CURITIBA-VIEWPORT. O trabalho de TCC proposto irá contribuir com este projeto de pesquisa no tocante à integração de usuários com a cidade virtual por meio de *smartphones*. Isso envolverá o desenvolvimento de software adicional ao código existe e a realização de mudanças em alguns de seus módulos.

## 2.2 OBJETIVOS ESPECÍFICOS

A partir do objetivo geral definido para este trabalho, foram estabelecidos os seguintes objetivos específicos:

- Adquirir conhecimentos sobre o estado da técnica na construção de cidades virtuais e realidade misturada (MR), para servir de fundamentação para o modelo de cidade virtual a ser proposto.
- Especificar e modelar um software experimental para dispositivos móveis que realiza a integração de usuários a um ambiente de cidade virtual
- Implementar o software proposto visando contemplar a navegação de uma entidade (i.e., um personagem) dentro de um contexto de cidade.
- Realizar experiências com usuários visando analisar a sua reação ao tipo de software desenvolvido.

## 2.3 JUSTIFICATIVAS DO TRABALHO

Cidades virtuais são instrumentos interessantes que possibilitam uma infinidade de utilizações distintas. Além de servir a propósitos de entretenimento e análises, cidades virtuais podem trazer uma contribuição às pessoas que moram ou transitam em suas contrapartes reais, ou seja, podem se tornar um instrumento de utilidade ou interesse público.

Uma cidade virtual sincronizada, pode ser utilizada para auxiliar pessoas, como turistas e moradores, em seus deslocamentos. Mais do que oferecem os sistemas de navegação GPS, em uma cidade virtual, o usuário pode observar com realismo o ambiente no qual se desloca ou deslocará no ambiente real. A telepresença seria outro motivo a ser considerado, ir virtualmente a lugares como passeios, visitas, encontros e ensino nos quais o usuário reconhece o ambiente real por meio do virtual. Dessa forma, pode-se diminuir o gasto e tempo com deslocamento.

A segurança por meio de uma cidade virtual ajudaria o mapeamento e controle de ruas, áreas de risco e, até mesmo, controle de localização de detentos em regime semi aberto, que usam a tornozeleira. Existe inclusive um ramo de negócios em que uma cidade virtual é colocada à disposição para ser alugada, assim que o pagamento é efetivado ela pode ser utilizada da maneira que o locatário desejar, seja para simular algo ou para qualquer outra utilização que seja. Campanhas publicitárias por exemplo, poderiam ser realizadas de acordo com as regras da cidade.

# 3 REFERENCIAL TEÓRICO

Neste capitulo será abordado o levantamento teórico referente a estudos similares que ajudam e suportam o desenvolvimento deste projeto.

## 3.1 CIDADES EM REDE E CIBERESPAÇO

Independentemente de sua magnitude, população ou características, as cidades são consideradas um produto social, construído a partir da relação do homem com o meio em que está inserido (Lencioni, 2008). Com o avanço da tecnologia de informação e comunicação, não é mais possível considerar as cidades como somente uma delimitação geográfica, mas deve-se, também, levar em consideração todas as inter-relações estabelecidas nesse meio (Moraes, 2012).

As transformações mais recentes pelas quais as cidades passaram, devem-se às transformações das tecnologias de informação e comunicação, presentes tanto no âmbito econômico, político e territorial, quanto no sociocultural (Moraes, 2012). As tecnologias tiveram influência tão grande para as cidades que redefiniram o conceito de espaço e território, tendo sido essenciais para a formação de uma rede interconectada para o compartilhamento de informações, ou seja, um ciberespaço (Moraes, 2012). Além ter influenciado a alteração do espaço e território, as tecnologias de informação e comunicação trouxeram alterações, também, nos serviços e produtos. Vários serviços que previamente só eram alcançáveis pessoalmente, se tornaram mais acessíveis e as cidades começaram a ser consideradas cidades de informação e conhecimento (Fernandes & Gama, 2006).

As redes de interconexão introduziram também a noção de um espaço reticulado, oriundo da ideia de que a rede é também política e social. O espaço reticulado pode ser considerado como um espaço de transação de informação permanente, precisa e rápida (Moraes, 2012). Para conseguir manter o espaço reticulado ativo e sem problemas, é necessário que as transmissões de informação pelas redes seja fluída. Segundo Milton Santos, “a fluidez contemporânea é baseada nas redes técnicas”. Assim, a busca por uma maior fluidez e maior qualidade na transmissão de informações, mensagens, produtos e dinheiro por um meio eletrônico, causa uma grande procura por técnicas mais efetivas e eficazes para a estabilidade das redes (Santos, 2004).

Uma das tecnologias mais importantes na atualidade é a rede mundial de computadores (i.e., Internet) que é entendida pela geografia como uma forma de organização espacial (Moraes, 2012). Isto se deve às características específicas de espacialidade como instabilidade, mobilidade nas redes e a complexidade de interações espaciais, incluindo a interação entre cidades (Moraes, 2012).

O maior foco de debate sobre a Internet na psicologia e sociologia, são os estudos sobre os impactos sociais e de interação humana formando uma "sociedade em rede", conforme descrito pelo sociólogo espanhol Manuel Castells. Essa rede no novo paradigma de tecnologia de informação tem possibilidade de fornecer a base necessária para a formação de uma estrutura social interconectada (Castells, 2008).

No contexto de sociedades em rede, um ponto importante é a criação de um ciberespaço, que "é originado a partir da interconexão de computadores por meio de uma rede" (Moraes 2012). No caso da Internet, esse conceito pode ser estendido para a relação de aparatos tecnológicos que possibilitam toda a interação da rede mundial de computadores. Essa extensão origina-se de uma releitura da definição do ciberespaço por Pierre Musso, o qual o define como "um conjunto de estruturas reticulares colocadas elas mesmas em rede" (Musso, 2009).

O ciberespaço da Internet trouxe a globalização da informação, alterando a economia. Com o crescimento do uso do ciberespaço da Internet, a proximidade territorial começou a perder importância sendo tomado seu lugar por novas formas de ações socioeconômicas, caracterizando, assim, a "nova economia" (Fernandes & Gama, 2006). Essa nova economia é apelidada "economia do conhecimento" e, segundo Gregersen, se reflete em quase todas as economias que têm como base o conhecimento, pois elas dependem de estruturas de conhecimento e de recursos humanos capacitados (Fernandes & Gama, 2006).

O ciberespaço permite a alteração no modo como os serviços podem ser ofertados em razão da praticidade e agilidade na utilização e tempo gasto para efetuar consultas e atendimentos. Os governos viram uma oportunidade de estreitar relações com seus cidadãos, podendo prover melhorias no relacionamento e na oferta de serviços públicos. Pesquisadores e estudiosos, como Bresser Pereira e Bovaird, defendem que a mudança de paradigma no atendimento e relacionamento com o cidadão é necessária e que "os Estados que não se modernizarem para acompanhar a nova ordem mundial certamente ficarão à margem do desenvolvimento da nova economia" (Simão, 2010).

Em âmbito nacional, pode-se citar alguns programas do Estado para o desenvolvimento de serviços e inovação tecnológica como o Programa de Modernização do Estado, o Programa da Sociedade da Informação e o Programa Governo Eletrônico. O Programa Governo Eletrônico, por exemplo, foi criado para melhorar e facilitar o atendimento e uso dos serviços públicos pelos cidadãos, por meio da internet (Simão, 2010).

Nas cidades e estados, os programas de desenvolvimento de prestação de serviços e portais são as ações comuns tomadas no uso das tecnologias de informação e comunicação (Simão, 2010). Um exemplo mais específico de uso de tecnologia para o atendimento ao cidadão é a "Prefs" da cidade de Curitiba que, por meio de rede social (i.e., Facebook), mantém contato direto com os cidadãos e mostra projetos que estão em execução na cidade. A partir da implantação de serviços de atendimento ao público e serviços digitais pelos governos, pode-se aplicar o termo "Cidade Digital". Segundo Ricardo Fernandes, uma cidade digital "se desenvolve perante uma tentativa de utilizar o potencial dos meios *on-line* ao serviço das regiões, das populações e do próprio marketing urbano" (Fernandes & Gama 2006).

## 3.2 CIDADES DIGITAIS

A interação do ciberespaço com a organização territorial da cidade é um assunto de total interesse do cidadão (Simão 2010). Assim como qualquer tecnologia, as cidades digitais trazem muitos benefícios, mas também, muitos desafios. Por exemplo, na construção civil, devem ser incluídas considerações sobre a influência exercida pelo ciberespaço no território geográfico.

Segundo Ricardo Fernandes e Rui Gama, "a cidade digital surge-nos como um conceito e política territorial inserida numa sociedade em rede". Ela é feita pela conexão de pessoas, cidadãos, e instituições, por meio de uma rede, podendo ser a internet, tendo suporte em uma infraestrutura de cidade real com objetivos diferentes para atender uma necessidade específica. Já Zancheti, afirma que o conceito de cidades digitais ainda não é consensual, podendo variar sua definição a partir do nível de desenvolvimento industrial de uma cidade, do modelo sócio-político implantado e das características culturais da população (Zancheti, 2001).

Em uma segunda definição, por Souto, Dall'Antonia e Holanda, a cidade para ser considerada digital, deve apresentar alguns requisitos, entre eles é apresentar infraestrutura de telecomunicações para acesso a rede de Internet, tanto para acesso particular como acesso

público e, também, fornecer por meio digital, serviços e informações públicas assim como serviços privados aos cidadãos. (Souto, Dall'Antonia e Holanda, 2001). Seguindo na mesma linha de pensamento, Zubieta e Woodley dizem que a cidade digital é a cidade que, oferecendo serviços de tecnologia da informação e comunicação, disponibiliza aos seus cidadãos serviços para melhorar o nível de desenvolvimento humano, econômico e cultural, tanto individualmente como em comunidade (Zubieta e Woodley, 2006).

Lemos (2006) sintetiza as ideias de entendimento sobre cidade digital como a representação de uma cidade no âmbito digital por meio de projetos governamentais ou privados, na forma de portais de instituições, provedores de serviços e informações de utilidade pública, comunidades ou fóruns virtuais e a representação política virtual sobre uma cidade. Englobando também a criação de infraestruturas, acesso aos cidadãos a serviços e novas tecnologias, objetivando fornecer acesso à internet por meio de estações como quiosques, modelar tridimensionalmente espaços geográficos da cidade com o intuito de simular o espaço urbano para ajudar no planejamento estratégico da cidade e criar cidades somente digitais, que não existem na realidade, focando na troca de informações e criar comunidades.

Na criação das cidades digitais o foco é a disponibilização da evolução tecnológica de informação e comunicação de maneira eficiente aos cidadãos, as quais são essenciais para a geração de conhecimento e transmissão de informações, ambos fonte principal de poder econômico e social da nova economia (Fernandes & Gama, 2006). A cidade se desenvolve a partir de meios digitais, promovendo marketing urbano, utilização de serviços regionais e territoriais e o potencial de uso dos cidadãos.

Nas cidades digitais, as novas tecnologias de informação e comunicação precisam ser consideradas, não como meios ou ferramentas, mas também como elementos culturais inseridos na vida das pessoas, elementos que se criam com a interação das pessoas, "novas formas econômicas e sociais, independentemente dos conceitos de distância, tempo e espaço." (Fernandes & Gama, 2006). Ricardo Fernandes e Rui Gama afirmam, ainda, que as tecnologias e mecanismos da virtualização de serviços, em conjunto com a realidade territorial e seguindo as políticas e legislações referentes às tecnologias de informação e transmissão de dados, resultando na criação de cidades digitais, são fatores necessários para a cidade se desenvolver na economia da informação e do conhecimento.

Para a implementação de cidades digitais, a partir dos conceitos apresentados

anteriormente, são necessários alguns pré-requisitos, a existência ou então a construção de uma infraestrutura de rede para conectar os cidadãos à Internet, sendo preferencial a rede ser banda larga. O segundo ponto a ser considerado é o apoio da administração pública da cidade que irá implantar o modelo de cidade digital e a participação do cidadão (Simão, 2010). Para que ocorra o interesse do cidadão a participar dos serviços promovidos pela cidade digital, é necessário que, primeiro, esteja envolvido com o processo para conhecer e saber dos benefícios e maneiras de utilizar o sistema. Além de utilizar os sistemas, o cidadão deve transformar e promover as tecnologias dispostas ao seu uso, "integrando novas utilidades às suas atividades, agregando valor aos seus produtos" (Simão, 2010).

Para o uso efetivo desses serviços e participação da população na gestão pública, é necessário que o cidadão tenha acesso às tecnologias de informação e comunicação (Baracho, 2012). Fernando Moraes defende em seu trabalho o incentivo a programas de inclusão digital para que o uso efetivo da cidade digital seja realizado. Moraes coloca em foco que o modelo de cidade digital que temos hoje no Brasil tem em primeiro plano somente a acessibilidade para quem já possui acesso à internet, sem ser levado em consideração uma avaliação qualitativa dos potenciais usos da mesma, causando assim a "segregação digital".

Como parte da definição de cidade digital, os governos deveriam prover à população acesso à rede de computadores para que todos conseguissem utilizar os serviços disponibilizados pela cidade digital. Programas de incentivo e de inclusão digital no Brasil, aliados com o aumento de poder aquisitivo das classes mais pobres e o barateamento dos equipamentos, estão fazendo com que o acesso à Internet e aos serviços disponibilizados digitalmente atinjam uma maior parcela da população (Moraes, 2012).

As cidades digitais tiveram seu começo no Brasil, no início dos anos 90, tendo como foco somente a implantação da infraestrutura de rede nas grandes cidades (Simão, 2010). Após uma grande idealização da digitalização das cidades, os programas começaram a diminuir, tendo alcançado seu mais baixo momento de interesse nos anos 90. Voltando a despertar o interesse dos governos ao final da década, os projetos de cidades digitais ficaram restritos às cidades metropolitanas (Simão, 2010). Atualmente, os programas de cidade digitais se estendem também para a administração pública, informações e programas de desenvolvimento do conhecimento da população.

Um dos projetos de inclusão e geração de conhecimento criado pelo governo brasileiro

foi a Escola Digital Integrada, o qual é um projeto de mediação da informação de desenvolvimento cognitivo em tempo integral utilizando as tecnologias de informação e comunicação como ferramentas de trabalho. A metodologia da Escola Digital Integrada foi desenvolvida como pesquisa de doutorado por Oliveira, em 2003, na Universidade de Brasília pelo Departamento da Ciência da informação e Documentação (Simão, 2010).

A escola digital integrada tem como objetivo principal possibilitar a inclusão digital de alunos da escola pública às novas tecnologias por meio de mediação baseando-se em interfaces tecnológicas, conteúdos e agentes sociais para ajudar na identificação da sua necessidade, na busca de informações, no uso da tecnologia para a construção do conhecimento se relacionando com o ambiente sociocultural e econômico em que o aluno está presente (Simão, 2010).

Oliveira 2003 definiu a metodologia da Escola Digital Integrada em etapas para serem seguidas para solucionar problemas e ajudar de maneira efetiva ao aluno aprender e a buscar informações. As etapas da metodologia de Oliveira 2003 são mostradas na tabela 1.

**Tabela 1 –** Etapas da metodologia da Escola Digital Integrada. Oliveira, 2003.

| **Etapa** | **Procedimentos** |
|---|---|
| Etapa Reflexivo-transformador | Gerar a reflexão referente a sociedade atual por meio da avaliação do trabalho, comunicação e conhecimento com o intuito de demonstrar a importância de autonomia no acesso à informação e ao conhecimento. Essa etapa provede também um aprimoramento na relação dos alunos com os mediadores e agentes sociais e explicar os objetivos do projeto para convencer os alunos da importância do projeto para o aprendizado e para o rendimento escolar cognitivo |
| Etapa Cognitivo-linguístico | A segunda etapa inicia com a explicação do que os alunos deverão seguir para alcançar um aprendizado efetivo e incentivá-los a executarem as tarefas de maneira positiva. Em seguida deve ser feita a conscientização dos alunos sobre seus pontos fortes e fracos mostrando como podem melhorar. Após ser feita a apresentação dos pontos fortes e fracos serão exercitadas as habilidades de observação, classificação e orientação do aluno assim como a compreensão da informação, assimilando as técnicas de aprendizagem nas diferentes fases do trabalho. |

| | |
|---|---|
| Etapa Tecnológica-Documental | Nessa etapa o aluno será introduzido ao mundo da informação baseados em seu desenvolvimento histórico e tecnológico com o contexto social. Serão desenvolvidas atividades com o aluno que ensine-o a buscar informação por meio de diferentes estratégias de trabalho cooperativo |
| Etapa Estratégico-Criativo | Se utilizar de desenhos para aplicações documentais a partir do desenvolvido previamente no projeto. |
| Etapa de Autoavaliação | O aluno fará uma auto avaliação a respeito do desenvolvimento de suas habilidades de busca por conhecimento e informação em conjunto com uma avaliação realizado pelo mediador. |

Na aplicação de governos eletrônicos para as cidades digitais, João Batista Simão (2010), fala sobre dois grupos de e-govs. O primeiro grupo foca somente em aplicar tecnologias de informação e comunicação para informatizar processos de administração pública. Segundo os autores Zweers e Plaqué (2001), o objetivo é disponibilizar informações, serviços e produtos para quem quiser quando quiser via meio digital, agregando valor aos *stakeholders* do âmbito público. O segundo grupo defende, segundo Mentaz, Apostolou, Young e Abecker (2001), "uma visão estratégica para a criação de um ambiente de transformação das atividades do governo pela aplicação de métodos de e-business no âmbito de setor público".

O Reino Unido foi um dos primeiros a criar um *site* para transmitir informações referentes ao governo em 1994. Já em 1998 foi feita uma releitura da proposta e foi avaliado que havia a necessidade de se criar um programa integrado para qualificar também os cidadãos, além de acesso aos serviços e informações. Nessa avaliação foi decidido criar os "programas de comunicação, conscientização e recursos para pequenas empresas" (Simão, 2010).

Um tempo depois, em 1999, o governo norte americano por intermédio do documento *Paperwork Elimination Act* garantiu que até o ano de 2003 proveria todos os serviços por meio digital. O método utilizado para alcançar esse objetivo foi autorizar a cada órgão do governo que implementasse sua própria iniciativa convergindo em um único *site* (Simão, 2010).

A grande base atualmente para se colocar um governo no meio digital (i.e., Internet) são os portais, os quais são caracterizados primariamente por 3 características: a centralização do acesso, a flexibilidade de acesso e a estruturação em canais. A centralização se dá por disponibilizar todas as informações e serviços necessários relacionados a uma instituição

através da internet (Moraes, 2012).

A principal atuação do governo brasileiro no meio digital é referente à transparência da informação, transmitir aos cidadãos informações sobre programas e projetos que o governo está realizando, dados sobre pagamentos de tributos, licitações, oferecendo também consultas e serviços públicos (Moraes, 2012). Esse modelo aplicado promove um portal com caráter consultivo, sem muita participação ativa da população em discussões referente ao portal da cidade de Curitiba, eleita em primeiro lugar como a melhor cidade digital do Brasil.

## 3.3 CIDADES INTELIGENTES

Com o grande crescimento desenfreado das cidades nas últimas décadas, os planos de desenvolvimento precisam ser cada vez mais elaborados para que a cidade cresça de maneira sustentável. Uma cidade inteligente segundo Stevenson e Wright (2006) é feita de um ambiente físico onde se encontram intrínsecos a objetos físicos tecnologias de informação e comunicação, além de sensores. Seguindo a linha de pensamento que a cidade inteligente precisa relacionar o ambiente físico com o virtual, Ricardo Fernandes e Rui Gama afirmam que as cidades inteligentes aparecem como regiões geográficas onde a influência da tecnologia na produção, uso, acessibilidade e disseminação são fluídas, promovendo a geração de conhecimento, inovação, em um patamar além do digital, gerando mão de obra qualificada, alto desenvolvimento tecnológico, econômico e social.

As atenções para cidades inteligentes estão voltadas, em sua maioria, para a estratégia territorial, tendo investimentos em pesquisa e desenvolvimento de planejamento tanto no setor privado como pelos governos. A competitividade em um contexto mundial está exigindo que cidades e regiões se tornem inteligentes (*learning regions*) se assim for possível, promovendo o constante desenvolvimento da população e de inovação (Serrano, 2005).

Para ser considerada efetivamente uma cidade digital, alguns parâmetros foram instaurados pelo projeto europeu de *smart cities.* Esses requisitos estão descritos conforme listagem a seguir:

- Sociedade: população com cultura cosmopolita, sendo necessário domínio não somente em sua língua nativa, participação em assuntos do âmbito governamental, educação continuada, bom índice de cultura geral e de leitura, alto nível de tolerância, atuação em voluntariados e fazer parte das eleições.

- Governo: sistema de gestão pública participativo com apoio ativo da população, gerar serviços públicos e sociais, informatizar de maneira transparente e planejar estrategicamente.
- Economia: gerar capacidade de inovação e empreendedorismo, diminuir as taxas de desemprego, aprimoração do sistema público de transporte e promover a flexibilidade no trabalho.
- Mobilidade: o transporte na cidade inteligente deve ser planejado para a melhor fluidez do trânsito, ser acessível a pessoas tanto locais como estrangeiras com um sistema de transporte público sustentável. É preciso apresentar, também, amplo acesso à Internet.
- Qualidade de vida: a cidade inteligente deve apresentar sistemas de educação, segurança e de saúde de qualidade, promover moradias sustentáveis e agradáveis à população, deve apresentar também atrações turísticas e alta percepção social.
- Meio Ambiente: é necessário que se apresentem espaços verdes na cidade, gestão sustentável dos recursos naturais, programas de incentivo ao consumo sustentável e apoio à reciclagem fazendo bom uso de seus espaços naturais.

Além do apoio de sistemas de informação no dia a dia da população e no planejamento estratégico governamental e privado, as cidades inteligentes apresentam alto nível de qualidade de vida promovendo o uso de novas tecnologias para a geração de conhecimento. Sobre essas características percebe-se que o foco da cidade inteligente não está nas partes integrantes do que a fazem ser inteligente, mas sim na conexão entre todos os elementos envolvidos, fazendo com que sejam transparentes para os cidadãos (Simão, 2010).

Alguns exemplos de implantação de projetos de cidades inteligentes inclui o de Boston para a estratégia de aprendizado comunitário. Esse projeto foi implantado em escolas, bibliotecas e centros comunitários, para que formassem uma espécie de *cluster* para que os adultos possam orientar os jovens em horários durante os quais não se encontram nas escolas.

João Simão cita que os serviços públicos estão cada vez mais desarticulados com agências do governo ficando mais inacessíveis à medida que as cidades crescem. Isso causa uma falta de sincronismo e convergência deixando o governo com mais dificuldade de atender e entender as reais necessidades da população à medida que a mesma aumenta.

Rosabeth Kanter defende o desenvolvimento do uso intensificado das tecnologias de

comunicação para gerar soluções de maneira integrada com a comunidade de forma sistêmica para interesses e projetos de alto impacto no crescimento de cidades. Incluindo ainda a integração da tecnologia em apoio ao crescimento e desenvolvimento, Kanter (2008) diz que a cidade inteligente é um complexo de comunidades de interesses, sendo que a tecnologia aplicada nessa cidade, deve fornecer serviços e prestar auxílio a essas comunidades para evitar a segregação das mesmas.

Kanter aponta alguns desafios para as *smart cities* focarem suas atenções, que somente elas conseguirão resolver a gestão e o controle demográfico, geográfico e jurisdicional, novas oportunidades de emprego, serviços públicos e do governo, atuação de ONG's de maneira fragmentada, lideranças cívicas, acordos com governos locais e isolamento social, realizar um planejamento estratégico de mobilidade urbana residencial. Com isso Kanter afirma que aliados a essas gestões e a geração de oportunidades devido à utilização das tecnologias de informação e comunicação, até mesmo as pequenas ou médias cidades se tornam mais atrativas e evitam a saída de seus cidadãos que esperam prosperar em áreas diversificadas.

Ricardo Fernandes e Rui Gama discorrem sobre a relação também do marketing territorial e a ajuda que a tecnologia oferecida pelas cidades inteligentes podem fornecer para seu melhor planejamento. Marketing territorial é definido como, atividades com o objetivo de otimizar características, funções e condições urbanas para cidadãos, turistas, empresas para fortalecer sua competitividade com as outras cidades dando suporte ao desenvolvimento da mesma. Para se desenvolver de maneira eficaz, o marketing territorial precisa ser planejado estrategicamente podendo assim promover e mobilizar recursos de maneira eficiente para tornar a cidade competitiva.

> "A relação entre a cidade inteligente e o marketing territorial depende de uma visão estratégica, das condições espaciais e econômicas, da liderança, do suporte político e social, da performance destas regiões do conhecimento, quer ao nível real quer na dimensão virtual, em diferentes esferas de ação (econômica, social, organizacional, entre outras) e da cooperação e organização." (Fernandes & Gama, 2006).

## 3.4. AMBIENTES VIRTUAIS

Os ambientes virtuais são implementações digitais de lugares, cidades ou paisagens, podendo ser reais ou fictícias, principalmente na forma de softwares de simulação (Miranda, 2012). Elas se aplicam tanto para entretenimento como em jogos eletrônicos somente para diversão, como na tentativa de aproveitar o potencial das novas tecnologias de comunicação para apoiar a evolução e manutenção de necessidades reais (Fernandes & Gama, 2006).

Muitos projetos e aplicações de ambientes virtuais têm sido desenvolvidos e empregados para diferentes fins. Exemplos disso, são os Tours Virtuais que podem ser feitos por diversos países, museus e lugares históricos por meio da Internet (Stockholm 360). O propósito, neste caso, é permitir o turismo virtual, motivando os viajantes a conhecerem e, eventualmente, visitar pessoalmente as cidades reais. Pode-se visitar construções, locais históricos e, até mesmo, realizar passeios por museus.

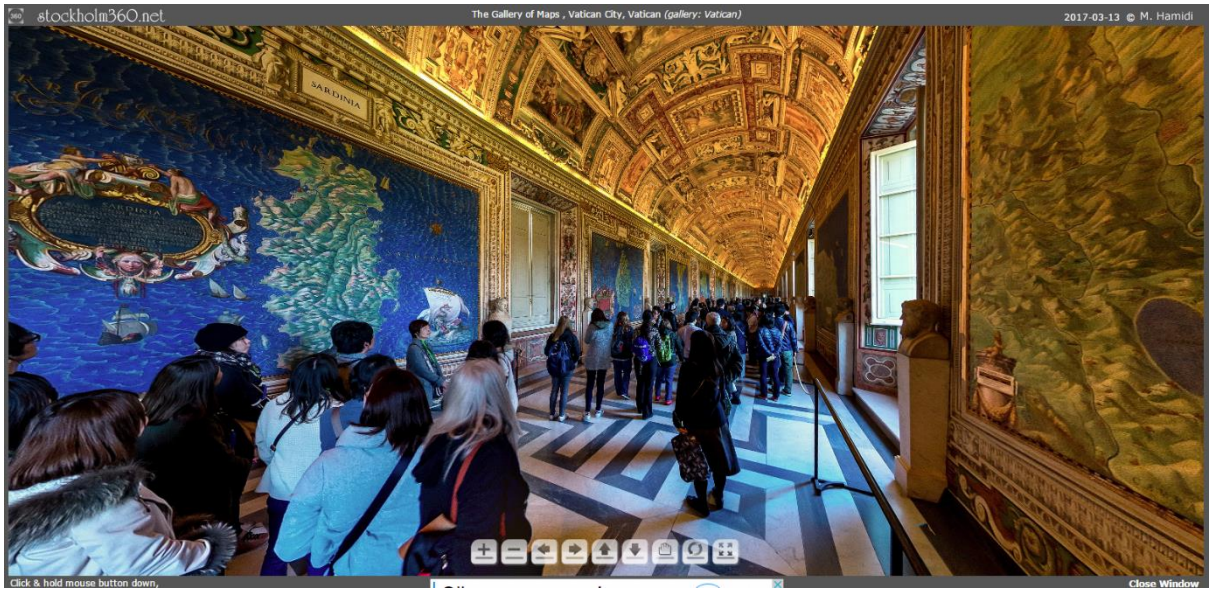



**Figura 1 -** Painel de controle para visitação ao Vaticano, stockholm.net

Esse exemplo de ambiente virtual utilizado anteriormente é somente para interação do usuário com o ambiente, sem nenhum tipo de contato com outros usuários. No aspecto multijogador conseguimos encontrar vários jogos com ambientes fantasiosos, sendo um dos mais famosos o *World of Warcraft*. O jogo produzido pela produtora *Blizzard*, em 2004, derivado de um jogo de estratégia anterior, é ambientado no mundo de fantasia de *Azeroth*,

onde os jogadores criam personagens para embarcar em aventuras junto com outros jogadores recebendo missões para proteger esse mundo e travar guerra contra a facção inimiga (World of Warcraft).


**Figura 2 -** Captura de tela realizada dentro do jogo World of Wacraft. Autoria pória.

Em um âmbito mais sério podemos encontrar ambientes para aprendizado e treinamento, como por exemplo o DOCTRAINING, um projeto conjunto da Universidade Federal Rural do Semi-Árido e a Universidade do Estado do Rio Grande do Norte para promover o aprendizado por estudantes de medicina em um ambiente de simulação virtual tridimensional. Nesse projeto o usuário pode escolher seu personagem e seu nome antes de entrar no ambiente. Quando o jogador faz suas escolhas iniciais ele é colocado junto com os outros jogadores que também estão no jogo. Dentro do jogo encontram-se personagens não controlados por humanos que quando interagem com jogadores, discutem informações referentes a assuntos médicos e podem requisitar consultas, onde os jogadores devem dizer qual é o diagnóstico do paciente (Lima, 2016).

Outro jogo sério extremamente utilizado é o X Plane-11, um jogo de simulação onde o jogador deve controlar uma aeronave. Nesse jogo não somente o jogador tem o controle do avião como também precisa controlar todos os equipamentos de voo dentro da cabine do piloto.

Esse jogo é utilizado para treinamento de novos pilotos e também treinar os pilotos em situações de risco, que no mundo real não seriam possíveis. Empresas aéreas também necessitam desses softwares de simulação para treinamento de novos pilotos em suas companhias (X Plane-11).

**Figura 3 -** Captura de tela da cabine de um avião em X Plane-11. Disponível em < http://www.x-plane.com/ >

Atualmente, os serviços providos às pessoas são difusos e em diferentes formatos, podem ser encontrados em cidades digitais, cidades inteligentes, mídias sociais, lojas eletrônicas, que ainda assim não estão limitadas à região e geralmente não são relacionadas entre si. O ambiente virtual em que este projeto é desenvolvido é o Curitiba *Viewport* (CWB-VP), o qual é uma plataforma única, geolocalizada e computadorizada da cidade de Curitiba com o objetivo principal de prover informações e serviços aos cidadãos da cidade.

Para que o CWB-VP promova um ambiente imersivo e apresente semelhanças ao ambiente real, a plataforma equaliza o tempo e clima do ambiente virtual com o real, assim como noção de dia e noite e a representação gráfica de transportes públicos em tempo real dentro do jogo, por meio do rastreamento dos automóveis. O pareamento das condições climáticas é realizado por meio de um webservice que se conecta com o site da Yahoo! e busca essas informações

para então atualizar no servidor do jogo e distribuir para os clientes. Dentro do motor de jogo Unity3D podemos configurar o comportamento do ambiente, conforme o buscado pelo webservice. Para o pareamento de transporte público, é utilizado outro serviço web que busca informações da central da URBS, onde está disponível a localidade do ônibus atualizado em tempo real.

O jogo sendo multijogador proporciona ao usuário promover a interação social com outras pessoas da cidade de forma anônima, enquanto fora de um grupo o personagem será identificado pelo nome escolhido pelo usuário, ou coletiva, quando dentro de um grupo o usuário poderá se identificar com seu nome real, dentro deste ambiente. Pessoas conectadas simultaneamente irão aparecer para outras na plataforma, caso estejam próximas, e poderão conversar por meio de chat interno do jogo assim como realizar atividades em conjunto.

Para atrair os usuários e promover interação entre os mesmos, será utilizada uma abordagem lúdica, permitindo exploração e aventura no ambiente, sendo por meio de missões ou eventos coletivos. Um exemplo que pode ser promovido, por exemplo, seria uma corrida entre parques da cidade em que seja necessário marcar a rota traçada entre os locais, sendo que o usuário possa escolher a categoria que deseja participar, como, bicicleta, a pé, skate ou como preferir, sem necessariamente ter um caminho pré-definido, deixando que o jogador tenha liberdade de escolha do caminho que realizará e por onde da cidade irá passar, somente com a premissa de precisar passar pelos parques.

Nessa mesma ideia pode-se utilizar a plataforma em grupo para outras atividades de entretenimento, não sendo necessariamente por meio de eventos disponibilizados organizados pelos servidores, e sim pelos próprios jogadores. Caso um jogador queira realizar uma atividade diferenciada, que possa impactar a cidade por meio de uma corrida, ou invasão de monstros, poderá ser disponibilizado para o usuário ou grupo de usuários uma cópia da cidade virtual para que seja possível a diversão sem afetar o ambiente colaborativo que provêm os serviços públicos e privados.

Para conseguir prover serviços tanto públicos como privados e com eficiência dentro desse ambiente virtual, a plataforma possuirá os prédios e estabelecimentos referentes a eles modelados dentro do jogo. Os serviços que poderão ser acessados são os mais variados possíveis, desde atendimentos na prefeitura, na polícia, bombeiros, assim como compras em lojas da região, atendimento ao consumidor.

Algumas das utilidades para serviço da plataforma e seus respectivos interessados são demonstrados a seguir:

**Tabela 2 –** Possíveis interesses, *stakeholders* e custos envolvidos com a plataforma.

| *Stakeholder* | Possíveis interesses | Custo |
|---|---|---|
| Estudantes | Assistir aulas a distância, estando virtualmente na Universidade junto de outros colegas e professores, requisitar serviços de secretaria, tirar dúvidas com professores, acompanhar monitoramentos. | Gratuito |
| Universidades | Aproximar os alunos a distância a um ambiente único para aulas onde os mesmos se sintam em um grupo. Promover a colaboração de alunos a distância, divulgar eventos, workshops e feiras de tecnologia, promover palestras online com profissionais especializados que não podem estar presencialmente. | Pago |
| Instituições públicas | Prefeituras, policiais, bombeiros, departamentos de trânsito podem oferecer serviços diretamente pela plataforma, da mesma maneira que fisicamente, aproximação da prefeitura com a população, promover ações culturais. | Pago |
| Instituições privadas | Provedores de serviços podem divulgar e interagir melhor com possíveis clientes por meio de marketing direcionado. Lojas online poderão demonstrar produtos de maneira mais natural do que somente por fotos, estando também virtualmente no local onde a loja realmente é. Clubes e academias poderão permitir aos cidadãos visitarem virtualmente o espaço e conhecer a infraestrutura que provêm. Imobiliárias poderão apresentar o apartamento ou casa o produto final de maneira virtual, antes mesmo do mesmo ser construído na realidade. | Pago |
| Cidadãos | Acessar todos os serviços previamente descritos sem precisar estar fisicamente no local, se divertir no ambiente da cidade, se encontrar com pessoas e interagir com maior facilidade. | Gratuito |
| Mídia | Promover propagandas em outdoor animados, assim como se utilizar do marketing direcionado, no qual cada pessoa verá uma oferta ou notícia referente ao seu interesse. | Pago |
| Investidores | Promoção interna na ferramenta assim como participação nos lucros provenientes da plataforma | Pago |
| Administradores do jogo | Prover melhora contínua à plataforma fazendo com que continue interessante aos *stakeholders* continuarem investindo nela. | Gratuito |

Claramente, desenvolver um jogo sério em um ambiente virtual dessa proporção não é nada simples, atualmente o projeto se encontra em desenvolvimento em que só temos a Universidade Tecnológica Federal do Paraná modelada tanto por dentro e por fora. Outras funcionalidades

como vídeos em outdoors, sincronização do clima e tempo, acesso multijogador e chat já foram desenvolvidas, mas ainda não está pronto para testes com serviços externos.

## 3.5 PLATAFORMAS MOBILE

Esta seção irá tratar, de maneira geral, das principais plataformas móveis existentes no mercado e suas utilizações mais comuns.

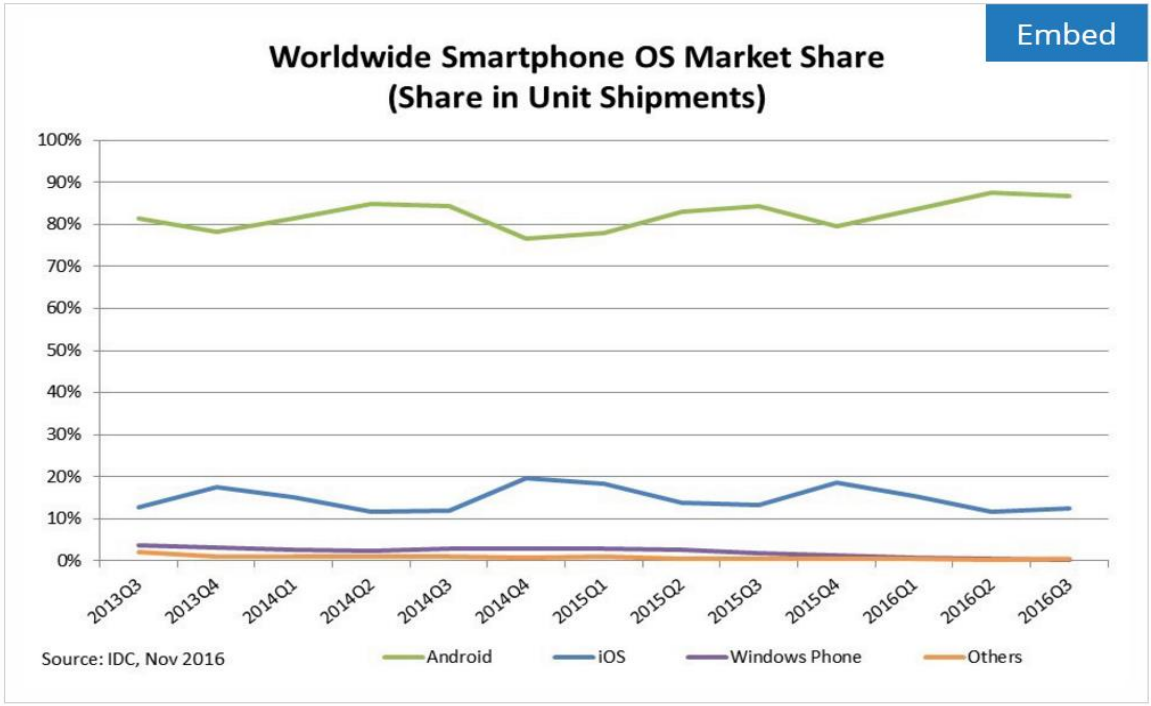
### 3.5.1 ANDROID

O Android é um sistema operacional criado para dispositivos móveis como *tablets* e smartphones com base em arquitetura Linux. Criado pela Android Inc em 2003, tinha como objetivo criar smartphones mais inteligentes, que percebem as preferências e localização de seu usuário. Em 2005 a empresa foi comprada pela Google, a qual era especulada para entrar no mercado mobile (Alian, Guang & Teng).

Dois anos após a aquisição, a Google lança o sistema operacional de maneira open source, sob licença Apache. Junto disso foi criado o Android Open Source Project (AOSP), o qual seria encarregado de prover suporte e melhorar continuamente o sistema operacional. Após ser lançado como *open source*, uma aliança de várias empresas chamada Open Handset Alliance veio à tona se mostrando favorável ao uso de tecnologias open source para dispositivos móveis e dispostas a utilizar o Android como sistema operacional. As empresas constituintes dessa aliança são: Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile, Texas Instruments, ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Huawei Technologies, PacketVideo, Softbank, Sony Ericsson, Toshiba Corp, and Vodafone Group Plc.

O Android possui uma das maiores comunidades de desenvolvimento de aplicações para mobile, contando em 2011 com mais de 500.000 aplicativos disponíveis para utilização nos dispositivos e mais de 10 bilhões de *downloads* dessas aplicações. O desenvolvimento dessas aplicações é facilitado, visto que a Google disponibiliza bibliotecas específicas para o Android e sua comunidade é extremamente ampla (desenvolvedores Android). As ferramentas de desenvolvimento utilizadas para programar para o Android são várias, pois a maioria dos IDE’s mais populares possuem extensões para programação móvel. A codificação é realizada

por meio de uma versão modificada de Java podendo também ser utilizada as linguagens C e C++, para programação de aplicativos diferenciados, como jogos, para os quais é necessário a interação com motores de jogos como Unity3D (Desenvolvedores Android). Com o tempo o Android se tornou o maior representante de sistema operacional de *smartphones* no mercado global (IDC, 2016).




| Period | Android | iOS | Windows Phone | Others |
|---|---|---|---|---|
| 2015Q4 | 79.6% | 18.7% | 1.2% | 0.5% |
| 2016Q1 | 83.5% | 15.4% | 0.8% | 0.4% |
| 2016Q2 | 87.6% | 11.7% | 0.4% | 0.3% |
| 2016Q3 | 86.8% | 12.5% | 0.3% | 0.4% |

**Figura 4 -** Porcentagem de utilização dos sistemas operacionais no mundo em 2016. Disponível em <http://www.idc.com/promo/smartphone-market-share/os>

## 3.6 ENGINES DE JOGOS

Nessa seção será descrito um pouco sobre o que são os motores de jogos e alguns dos motores mais famosos e utilizados.

### 3.6.1 VISÃO GERAL DOS ENGINES DE JOGOS

Os motores de jogo, *game engine* como foram definidas nos anos 1990, foram nomeadas a partir do jogo DOOM. O jogo possuía uma arquitetura muito bem definida que separava o núcleo de funcionamento do código como o sistema de renderização, sistema de colisão e sistema de som, das codificações de arte gráfica do jogo e suas regras de negócio que comprimiam o escopo de interação com o usuário (Gregory 2014). Essa separação se mostrou importante a partir do ponto que as produtoras começaram a utilizar códigos já feitos para outros jogos para criar novos, alterando a arte, os mundos, armas, personagens e regras de jogo sem ser necessário alterar a estrutura de funcionamento do mesmo, a *engine* (Gregory 2014).

Atualmente os motores de jogo são licenciados para quem quiser utiliza-las, trazendo uma segunda renda a produtora que a desenvolveu. Mesmo sendo necessário prática para conseguir desenvolver o jogo utilizando uma *engine* ainda assim é muito mais prático e rápido do que criar toda uma estrutura de validação, renderização e som do zero (Gregory 2014).

Geralmente os motores de jogo são específicos para um único gênero de jogo, por exemplo, um jogo de luta terá um motor diferente do que um jogo multijogador online em mundo aberto, pois as regras de base para os dois jogos são totalmente diferentes. Ainda que existam diferenças para entre os motores a serem utilizados, alguns conseguem se sobrepor a alguns gêneros devido sua semelhança (Gregory 2014).

O motivo de muitas vezes ser necessário um motor de jogo diferente é devido ao enfoque em que a *engine* foi desenvolvida, por exemplo, em jogos de tiro em primeira pessoa, o foco se concentra em renderização eficiente de ambientes 3D, controle de câmera e mira responsivo, animações extremamente trabalhadas em movimentos de braços e armas, sensação de movimento real do personagem para não parecer estar flutuando no mapa, animações bem produzidas de personagens controlados por inteligência artificial e baixa capacidade de jogadores simultâneos em ambiente multijogador, já para jogos de plataforma o foco é outro, o motor é feito para ambientes de quebra cabeça, acompanhamento de câmera em terceira pessoa controlado ou não pelo jogador, sistema de colisão eficiente da câmera com o ambiente

para não atrapalhar a imersão do jogador e plataformas que se movem, escadas, cordas e objetos que se movimentam para auxiliar o personagem (Gregory, 2014).

Atualmente, alguns dos motores de jogo mais conhecidos são: Unreal Engine (https://www.unrealengine.com), Frostbite (https://www.ea.com/frostbite), OGRE (http://www.ogre3d.org) e Unity3D (https://unity3d.com). A Unreal Engine foi criada em 1998 pela produtora Epic Games com o jogo Unreal. Desde seu início a Unreal foi a maior competidora da engine de Quake no gênero de FPS, *First Person Shooter*. Unreal Engine 2 foi criada para o jogo Unreal Tournament 2004 e abrangeu um maior espectro podendo ser utilizada para modificações e criações próprias fora do jogo Unreal Tournament (Tavakkoli, 2015). Na época era um dos motores de jogo mais utilizados por universidades e jogos comerciais. A última versão a ser lançada do motor é a Unreal Engine 4, melhorou os pontos fortes de suas versões anteriores e adicionou a ferramenta de desenvolvimento uma interface gráfica para facilitar criação de sombreamento e texturas. A Unreal Engine 4 é extremamente bem feita e pode ser utilizada para a criação de jogos de tiro em primeira pessoa e jogos de tiro e ação em terceira pessoa e possui uma extensa comunidade e documentação (Unreal Engine 4 Documentation).

A *engine* Frostbite foi criada pela produtora DICE para ser utilizada em 2006 no jogo Battlefield Bad Company. Desde então esse motor se tornou o mais popular em jogos criados pela Eletronic Arts, sendo usado nas franquias Mass Effect, Battlefield, Need for Speed e Dragon Age. O diferencial oferecido pela Frostbite é a ferramenta de criação de objetos dentro do jogo e possibilidade de destruição de cenário que não era englobada em outros motores (Frostbite Engine).

OGRE é outro motor de jogo fácil de ser utilizado, fácil de aprender a usar e bem arquitetado. Ele consegue renderizar ambientes e objetos tridimensionais incluindo iluminação avançada e sombreamento, produzir movimentos realistas de movimento do esqueleto, sistema de overlay em duas dimensões, interface gráfica e pós processamento da imagem como o efeito “borrão”. Seus produtores dizem que o OGRE não é um motor completo por si só, que possa produzir um jogo somente com ele, porém ele provê boa parte da base para conseguir produzir qualquer jogo em cima dele, além do motor ser *open-source* (OGRE Manual v1.8).

### 3.6.2 UNITY 3D

Unity3D é um motor de jogos e *run-time* para desenvolvimento de ambientes que consegue ser utilizado em várias plataformas diferentes como Windows, Linux, Mac, PS3, PS4, XBOX, Nintendo Wii e Wii U, Android, Apple iOS, Windows 10 Mobile. Além disso é capaz de executar programas nos *browsers* mais conhecidos. Os objetivos principais da *engine* Unity3D é o fácil desenvolvimento de aplicações e customização e o potencial de distribuição para várias plataformas, podendo interconecta-las sem problema (Unity Technologies). Para isso o Unity3D provê uma interface gráfica integrada que permite a criação e manipulação de personagens, ambientes, objetos, câmera e iluminação, permitindo ao passo que o jogo é construído, também ser pré-visualizado a partir da perspectiva do jogador. Para executar os jogos de maneira fluída em todas as plataformas, o Unity3D possui uma gama de ferramentas que ajudam o desenvolvedor encontrar possibilidades de melhorias e otimizações para a plataforma em que está sendo feita a distribuição (Unity User Manual).

A interface do Unity3D é distribuída basicamente em três seções: em parte da tela se pode manipular e criar os objetos e cenários, em outra parte fica a prévia do ambiente em que se está trabalhando no momento e a área na direita da tela onde se encontram todos os objetos e códigos de manipulação dos mesmos. Os *scripts* para controle e manipulação dos objetos podem ser desenvolvidos em JavaScript, C# ou Boo, um sistema de animação que suporta animações iguais para personagens totalmente diferentes (Olivenza, Puga, Martín & Calero 2013). Cada objetos e detalhe criado dentro do Unity3D é chamado de entidade. Cada entidade possui abaixo de si um ou mais *scripts* de comportamento e relação com outras entidades. Por exemplo uma entidade Árvore, tem o comportamento de não poder se mexer, balançar as folhas com o vento e colidir com outros objetos. O Vento seria outra entidade que interagiria com outros objetos como a árvore, causando o comportamento das folhas da mesma (Unity3D).

# 4 MATERIAL E MÉTODOS

Os recursos de hardware e *software* necessários para a execução deste projeto estão descritos na seção 4.1 Recursos de *Hardware* e 4.2 Recursos de *Software* mantendo o foco nos aspectos técnicos de cada recurso utilizado.

## 4.1 RECURSOS DE HARDWARE

Para o desenvolvimento do artefato final do projeto para execução e testes dos algoritmos, assim como as simulações, foram utilizados computadores pessoais com placa de vídeo (hardware compatível com os padrões atuais) e *smartphones* pessoais (Samsung S8 Plus e Asus Zenfone 3 Max). Sendo assim, foram utilizados os próprios equipamentos pertencentes aos membros da equipe, sem acarretar na necessidade de aquisição de novos equipamentos de *hardware*.

## 4.2 RECURSOS DE SOFTWARE

Em relação aos recursos de software que foram utilizados no desenvolvimento do projeto, foram utilizados o Visual Studio 2013, Unity3D, Eclipse, SQL Server 2014 e o Apache Tomcat 7. O Unity3D é uma *engine* para desenvolvimento de jogos provendo funcionalidades de física, iluminação, edição de terrenos, áudio e navegação. Esse software permite a importação dos modelos digitais. Os recursos do Unity3D possibilitam a criação de *scripts* para a implementação de comportamentos específicos aos objetos, como colisão, movimento, alterações de luz, interação com o ambiente, etc. Os desenvolvimentos dos códigos para o Unity3D foram feitos no Visual Studio 2017. Esse software é um IDE para desenvolvimento nas linguagens C#, C++ e Visual Basic. O Eclipse é um IDE para desenvolvimento de códigos na linguagem Java. E, por fim, o Apache Tomcat é um servidor que trabalha com Java, sendo utilizado mais especificamente para web. Tanto o Unity3D quanto o Visual Studio 2013, o Eclipse, o SQL Server e o Apache Tomcat são gratuitos, não tendo assim a necessidade de se pagar por uma licença de nenhum dos softwares.

## 4.3 VIABILIDADE

Sendo avaliado previamente nos recursos de *hardware* e software que serão gratuitos, e os computadores escolhidos para serem utilizados no desenvolvimento do artefato resultante desse projeto, não houve custo expressivo no projeto. Os custos do desenvolvimento foram apenas em termos de tempo de trabalho dos integrantes.

# 5. DESENVOLVIMENTO

Este capítulo apresenta as tarefas que foram realizadas durante todo o desenvolvimento do projeto.

## 5.1 PROJETO DA SOLUÇÃO

Neste capítulo será descrito como foi feita a modelagem da solução, incluindo os diagramas necessários.

### 5.1.1 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos é o processo de criação, onde cliente, analistas, engenheiros, gerentes e basicamente todos envolvidos no desenvolvimento do software criam, em conjunto, um grupo de requisitos baseados nas tarefas que compõem o escopo do projeto que o cliente deseja (Pressman & Maxim, 2016). Na engenharia de software, os requisitos devem ser claros, quantificáveis e livres de interpretação dúbia. Esses requisitos podem ser divididos em funcionais e não-funcionais. Os requisitos funcionais definem o que o software deverá fazer, descrevendo cálculos, detalhes técnicos ou manipulação e armazenamento de dados. Já os requisitos não-funcionais são os requisitos que definem a qualidade e o desempenho do software (Pressman & Maxim, 2016).

Requisitos Funcionais da Aplicação:

- RFA-001 – A CWB-VP-MOB deverá permitir ao usuário efetuar *login* no servidor da plataforma CWB-VP.
- RFA-002 – A CWB-VP-MOB deverá informar ao usuário o status de *login* no servidor da plataforma CWB-VP.
- RFA-004 - A CWB-VP-MOB deverá permitir a comunicação com jogadores conectados na CWB-VP.
- RFA-005 - A CWB-VP-MOB deverá capturar a localização do usuário e enviá-la ao servidor.
- RFA-006 - A CWB-VP-MOB deverá atualizar a localização do usuário quando o mesmo se movimentar.

- RFA-007 - A CWB-VP-MOB deverá requisitar ao usuário a permissão de acessar sua localidade.

Requisitos Funcionais da plataforma CWB-VP:

- RFP-001 – A plataforma CWB-VP deverá autenticar uma solicitação de *login*.
- RFP-002 - A plataforma CWB-VP deverá informar o status de *login* quando requisitado por algum CWB-VP-MOB.
- RFP-003 - A plataforma CWB-VP deverá receber os dados de localização do CWB-VP-MOB.
- RFP-004 - A plataforma CWB-VP deverá desenhar o personagem dentro do jogo no mesmo ponto geográfico que o recebido do CWB-VP-MOB.
- RFP-005 A plataforma CWB-VP não deverá autorizar a conexão na aplicação para PC caso receba uma tentativa login vindo do CWB-VP-MOB.
- RFP-006 - A plataforma CWB-VP deverá guardar a última localização do usuário para caso o mesmo acesse pelo PC.
- RFP-007 - A plataforma CWB-VIP deverá transmitir ao CWB-VP-MOB as mensagens do chat destinadas ao usuário.

Requisitos Não-Funcionais de Ambiente:

- RNFA-001 – O CWB-VP-MOB deverá ser escrito em linguagem Java.
- RNFA-002 – O CWB-VP-MOB deverá ser executar sobre plataforma Android.
- RNFA-003 – O CWB-VP-MOB deverá ser construído utilizando a ferramenta de desenvolvimento Unity 3D.
- RNFA-004 - O CWB-VP-MOB deverá se comunicar com a plataforma CWB-VP por meio de métodos próprios do Unity 3D.
- RNFA-005 - O CWB-VP-MOB deverá ter conectividade com a Internet para conexão com o CWB-VP.
- RNFA-006 – O CWB-VP-MOB deverá ter acesso à localização do usuário enquanto utiliza a aplicação.
- RNFA-007 - O CWB-VP-MOB deverá ser executado fora de sombra de satélite para não haver falha de GPS.

Requisitos Não-Funcionais de Interface:

- RNFI-001 – O CWB-VP-MOB deverá comunicar-se com o servidor da plataforma CWB-VIP usando TCP/IP sobre conexão de dados pela Internet.
- RNFA-002 – O CWB-VP-MOB deverá ter interface gráfica com o usuário construída nos padrões já utilizados no CWB-VP.
- RNFA-003 - O CWB-VP-MOB deverá ser limitado, com exceção de imagens, as cores preto, cinza, verde e branco.

Requisitos Não-Funcionais Econômicos:

- RNFE-001 – O CWB-VP-MOB deverá ser desenvolvido dentro do prazo de 12 meses.
- RNFE-002 – O CWB-VP-MOB deverá se limitar ao uso de ferramentas de desenvolvimento e tecnologias gratuitas.
- RNFE-003 – O CWB-VP-MOB deverá ser desenvolvido unicamente pelos dois desenvolvedores do projeto.

Requisitos Não-Funcionais de Estima:

- RNFES-001 – O CWB-VP-MOB deverá possuir interfaces que garantam alta usabilidade para usuário.

Requisitos Não-Funcionais da plataforma CWB-VP:

- RNFP-001 - O CWB-VP deverá ter um servidor web para receber dados do CWB-VP-MOB.
- RNFP-002 – O CWB-VP deverá ter um Banco de Dados para registro e validação de dados dos usuários.
- RNFP-003 - O CWB-VP deverá ser capaz de receber conexões externas à rede da UTFPR.
- RNFP-004 – O CWB-VP deverá ser capaz de processar chamadas de *WebService* em linguagem Java.
- RNFP-005 - O CWB-VP deverá ser capaz de desenhar mais de um usuário dentro do jogo.
- RNFP-006 - O CWB-VP deverá ter seu mapa geo-referenciado.

- RNFP-007 - O CWB-VP deverá ser capaz de processar dados geográficos para desenhar corretamente os personagens.

### 5.1.1 CASOS DE USO

Um caso de uso é uma descrição de um comportamento de um sistema, seja ele o sistema completo, ou apenas uma tarefa separada dentro do sistema como um todo, sendo que esta descrição é realizada relatando passo a passo a interação do usuário com o sistema, em um cenário ótimo (Melo, 2004). A seguir estão descritos os casos de uso que contemplam a solução apresentada, sendo definidos os atores como usuários do CWB-VP-MOB e CWB-VP, sendo o primeiro usuário da versão *mobile* e o segundo da versão *desktop* da aplicação.

**a) Realizar login no CWB-VP-MOB**

**Atores:**

Usuário do CWB-VP-MOB

**Pré-condição:**

O usuário deve possuir login e senha válidos.

**Fluxo de Eventos:**

1. O usuário irá abrir o aplicativo mobile
2. O usuário irá informar suas credenciais
3. O usuário irá submeter os dados ao sistema
4. O sistema irá enviar uma mensagem de erro caso o login seja inválido, ou redirecionar o usuário a tela principal caso o login seja válido

**Pós-condição:**

O usuário irá obter acesso ao sistema.

**b) Realizar login no CWB-VP**

**Atores:**

Usuário do CWB-VP

**Pré-condição:**

O usuário deve possuir login e senha válidos.

**Fluxo de Eventos:**

1. O usuário irá abrir o aplicativo desktop
2. O usuário irá informar suas credenciais
3. O usuário irá submeter os dados ao sistema
4. O sistema irá enviar uma mensagem de erro caso o login seja inválido, ou redirecionar o usuário a tela principal caso o login seja válido

**Pós-condição:**

O usuário irá obter acesso ao sistema.

**c) Deslocar-se no jogo através de sua posição real**

**Atores:**

Usuário do CWB-VP-MOB

**Pré-condição:**

O usuário deve ter realizado o login no aplicativo.

**Fluxo de Eventos:**

1. O usuário do CWB-VP-MOB irá se movimentar no mundo real e através da localização GPS o seu avatar irá se movimentar simultaneamente no mapa

**d) Visualizar outros jogadores no jogo**

**Atores:**

Usuário do CWB-VP-MOB e Usuário do CWB-VP

**Pré-condição:**

O usuário deve ter realizado o login no aplicativo.

**Fluxo de Eventos:**

1. O usuário irá se locomover pelo mapa, sendo possível visualizar outros usuários, tanto do CWB-VP quanto do CWB-VP-MOB assim como suas localização.

**e) Enviar uma mensagem ao chat**

**Atores:**

Usuário do CWB-VP-MOB e usuário do CWB-VP

**Pré-condição:**

O usuário deve ter realizado o login no aplicativo

**Fluxo de Eventos:**

1. O usuário irá clicar no ícone de mensagens
2. O usuário irá digitar a mensagem no campo de mensagem
3. O usuário irá clicar no botão de envio da mensagem

**Pós-condição:**

Tanto o usuário do CWB-VP-MOB quanto o usuário do CWB-VP irão poder visualizar as mensagens enviadas.

**f) Receber uma mensagem no chat**

**Atores:**

Usuário do CWB-VP-MOB e usuário do CWB-VP

**Pré-condição:**

O usuário deve ter realizado o login no aplicativo, devem haver outros usuários online

**Fluxo de Eventos:**

1. O usuário do CWB-VP irá enviar mensagens de chat através da plataforma desktop
2. O usuário do CWB-VP-MOB irá receber as mensagens enviadas no aplicativo mobile

### 5.1.3 DIAGRAMA DE CASOS DE USO

A Figura 5 apresenta o diagrama de caso de uso referentes aos casos de uso citados na sessão anterior.

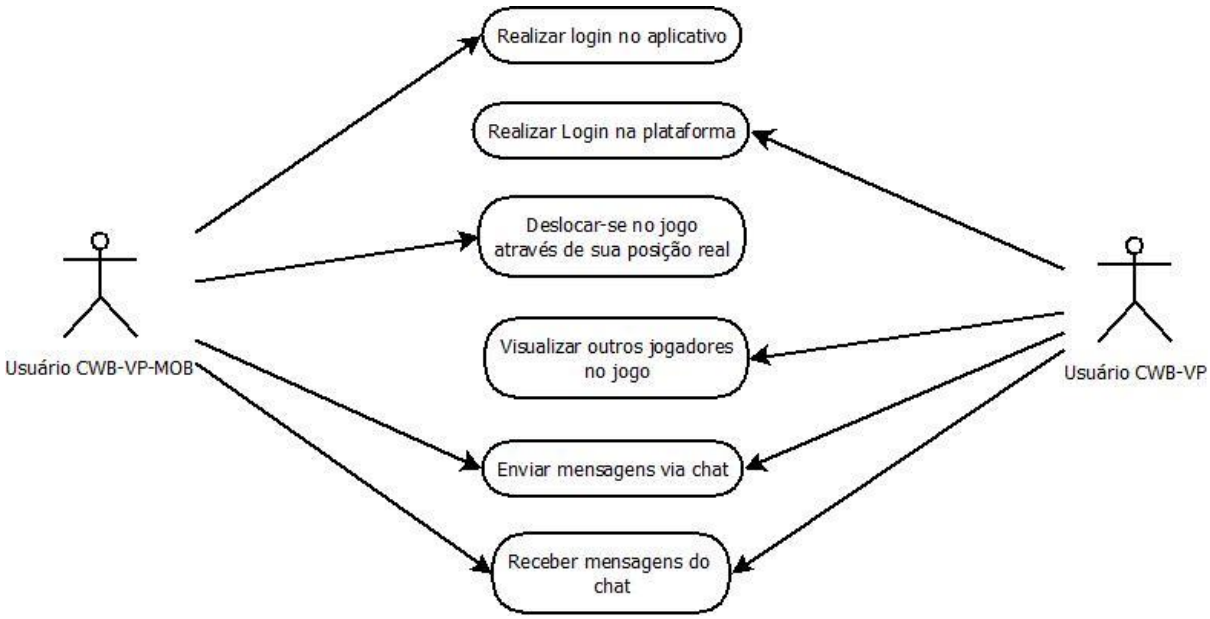



**Figura 5 –** Diagrama de casos de uso. Autoria própria

## 5.2 VISÃO ARQUITETURAL

O software desenvolvido se integra ao software CWB-VP adicionando funcionalidades para interação com usuários via *smartphones*. A Figura 6 apresenta uma visão arquitetural do software CWB-VP integrado com o software proposto neste trabalho de conclusão. Os blocos em preto indicam módulos novos que foram criados neste trabalho. Blocos cinza indicam módulos existentes no CBW-VP que sofreram modificações para operação na plataforma móvel.

O bloco “Cliente CWB-VP” indica a porção do software que é executada em dispositivos móveis dos usuários do CWB-VP. Já o bloco “Servidor-VP” indica a porção do software que executa em um computador servidor e tem o papel de gerenciamento das atividades do CWB-VP.

A cena de LoadSettings mostra uma tela de carregamento na qual os dados do servidor são carregados para o cliente. Informações que vêm do banco de dados são carregadas como,

por exemplo, clima, horário, notícias, últimos dados salvos e afins. Após realizar o carregamento, a cena de Login é chamada, onde o usuário insere suas credenciais para acesso. A tela de login dispõe a possibilidade também de alterar o idioma de português para inglês, essa funcionalidade ainda está em desenvolvimento e não está presente em todo o software.

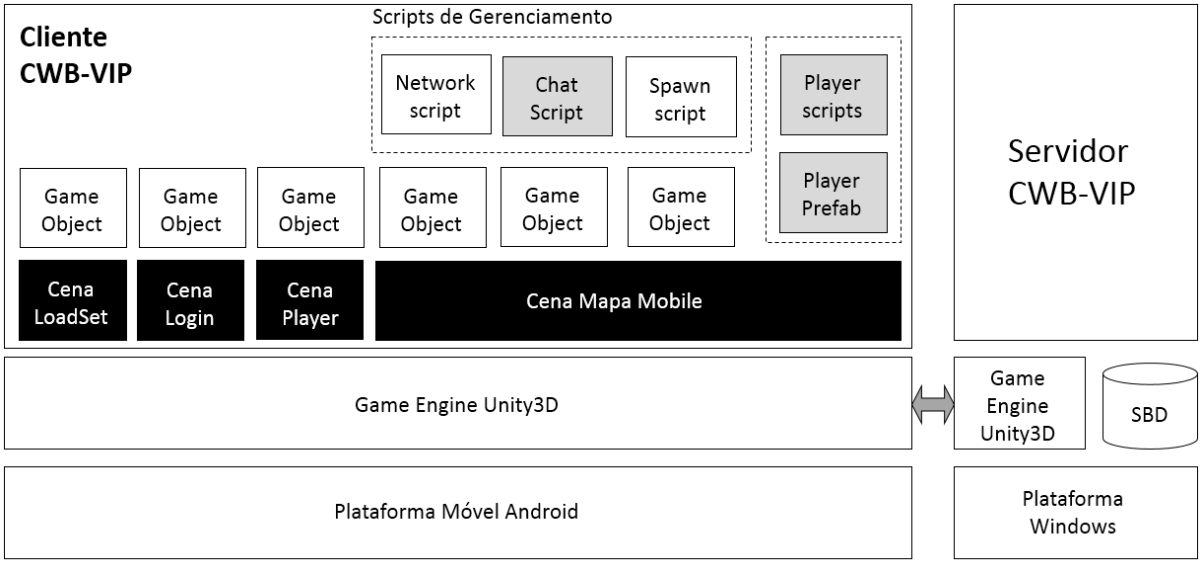



**Figura 6 –** Visão arquitetural do software desenvolvido. Autoria própria

A partir do login ser realizado, o software carrega a cena de Player, nessa cena o usuário poderá escolher entre três modelos de personagens para ser representado no software. Cada personagem é mostrado rotacionando o modelo para melhor visualização. Por fim após ser feita a escolha do personagem é carregada a cena do Mapa Mobile. Nesta cena podem ser visualizados o canvas de chat, a lista de amigos adicionados e a representação dos personagens conectados no jogo no mapa do aplicativo. O chat é funcional podendo ser utilizado a partir do teclado do Android e a posição do personagem é atualizada conforme se movimenta com o aparelho, por meio de captura de localização pelo GPS.

Nos *scripts* do Player foram realizadas alterações para adaptar o movimento, tamanho e física dos jogadores. Em um ambiente controlado como o PC é fácil controlar por física onde o personagem não pode ir em um mapa, porém se tratando de captura de localização por GPS está operação é mais complexa. Para não ter problema de um usuário ficar preso dentro de um prédio ou sair da área de mapa carregado e cair abaixo da área de colisão, os *scripts* de jogador para *mobile* ignoram a física do jogo, possibilitando atravessar paredes e outros jogadores caso

seja necessário.

As opções de modelo dos personagens são alterados na versão mobile, pois quando a cena de mapa é carregada a visão do personagem é inclinada de cima, para melhor visualização do mapa. Para ser possível ver o mapa e o personagem não ficar demasiadamente pequeno, quando o software verifica que a plataforma em que está executando é Android, o mesmo altera os modelos para que sejam aumentados em 40 vezes o tamanho que aparece na versão *desktop*.

A movimentação do personagem *mobile* não ocorre da mesma maneira que no computador. No Windows o usuário se movimenta pelas teclas WASD ou pelas flechas do teclado. No *mobile* o usuário não tem um controle ou liberdade de se movimentar como quiser, sua posição é definida somente pela localização do GPS e se altera somente da mesma forma.

Outras alterações realizadas são as verificações de funcionamento do GPS, foram implementadas verificações de permissão ao GPS do celular, validação se está conseguindo acessar com sucesso a rede de GPS, se não houve perda de comunicação com o satélite e se o usuário não desativou o GPS do aparelho.

Para o *chat* funcionar no software para dispositivos móveis foram necessárias alterações em como o usuário escreve no *chat* e como envia a mensagem. Para poder escrever no *chat* na versão de PC, o usuário clica na barra de digitação e simplesmente digita. Na versão *mobile* foi necessário alterar a barra de digitação para permitir ao Android abrir o teclado para digitação do usuário. Para o envio na versão *desktop* o usuário precisa apertar a tecla “Enter”, pois a partir do momento em que o *chat* entra em foco, uma validação contínua verifica se a tecla é pressionada. Na versão *mobile* a mensagem é enviada a partir de quando o usuário retira o foco do teclado pressionando do botão “OK” do teclado do celular.

## 5.3 MODELO ESTRUTURAL

O desenvolvimento do software envolveu a modificação de classes do software CWB-VP e a criação de novas classes. A Figura 7 apresenta o diagrama de classes do subsistema do CWB-VP que foi alvo deste projeto. Outros subsistemas do CWB-VP, que não dizem respeito a este projeto, não foram incluídos neste modelo.

O diagrama de classes tem por objetivo fornecer uma observação estática das classes que fazem parte do sistema, seus atributos e métodos, assim como todos os relacionamentos

existentes entre as mesmas, caso existam (Guedes, 2014). A Figura 7 apresenta o diagrama de classes da solução. Para uma análise melhor, o mesmo encontra-se em anexo.

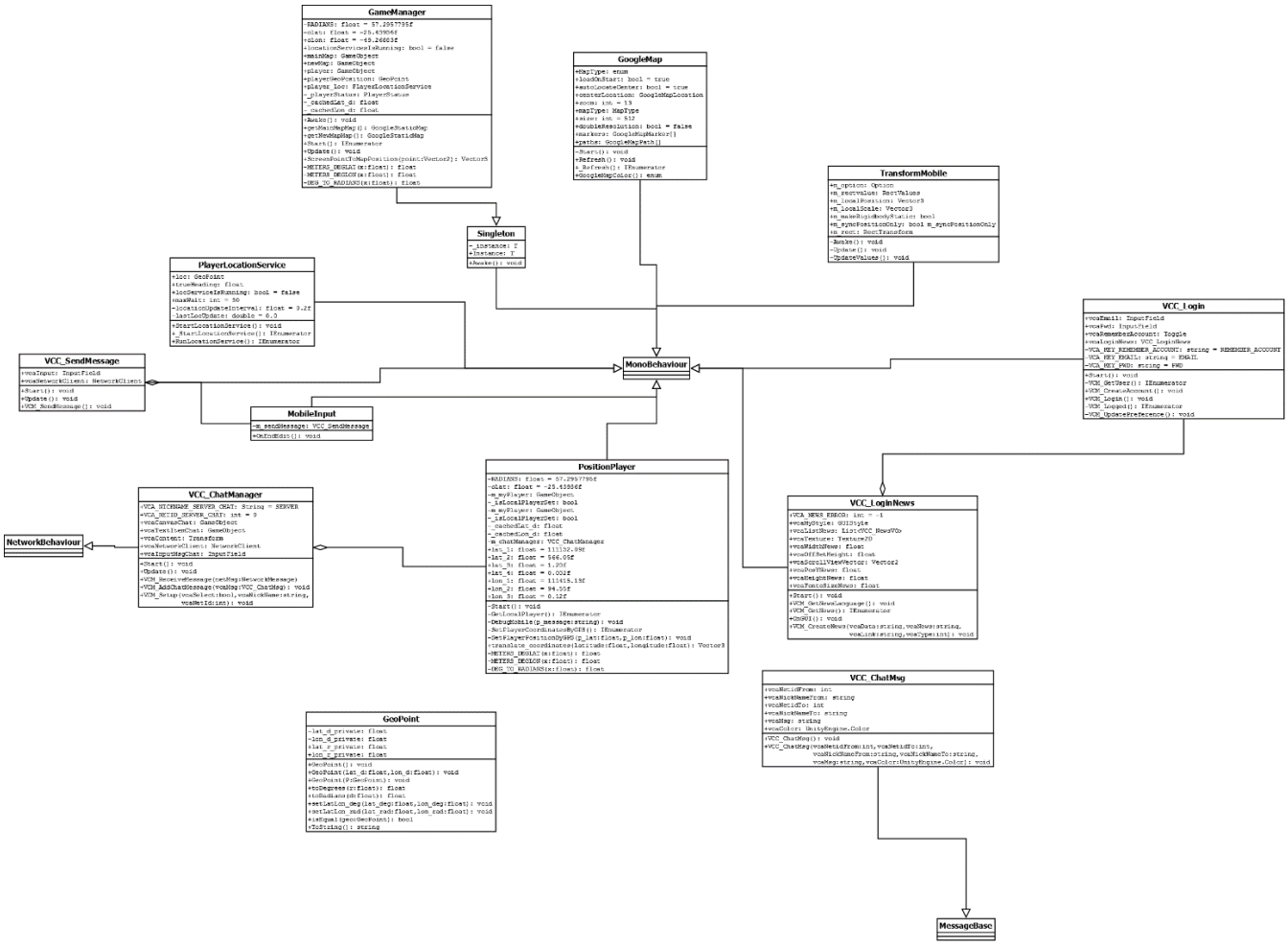



**Figura 7 –** Diagrama de classes. Autoria própria

Uma classe do CWB_VP precisou ser alterada, conforme descrito a seguir.

- **GameManager**

  Esta classe é responsável pelo controle do jogo na versão mobile e tem conhecimento dos objetos que estão ativos. Esta classe foi alterada adicionando os atributos e métodos a seguir:

  _chacheLat_d: atributo do tipo *float* para armazenar temporariamente o dado de latitude do personagem.

  _chacheLon_d: atributo do tipo f*loat* para armazenar temporariamente o dado de

longitude do personagem.

getMainMap: método para gerar o mapa a partir do ponto zero do jogo, que é a UTFPR.

getNewMapMap: método para atualizar o mapa a partir da posição do jogador.

As novas Classes adicionadas ao CWB-VP foram:

- **PlayerLocationService**

  Esta classe é responsável por iniciar o serviço de localização do GPS, validar se ele está ativo e atualizar a posição do jogador. Esta classe contém os seguintes atributos e métodos:

  loc: atributo do tipo *GeoPoint* para armazenar a localização.

  locServiceIsRunning: atributo do tipo b*oolean* para armazenar informação se o serviço de GPS está ativo.

  maxWait: atributo do tipo i*nt* para armazenar o tempo máximo de esperar para desconectar caso o serviço não esteja no ar.

  locationUpdateInterval: atributo do tipo *float* para definir o tempo para atualização da localização do usuário.

  lastLocUpdate: atributo do tipo *double* para armazenar informação de a quanto tempo foi atualizada a última vez a posição do usuário.

  StartLocationService: método responsável por ativar o serviço de localização.

  RunLocationService: método responsável por manter em execução o serviço de GPS.

- **PositionPlayer**

  Esta classe é responsável por atualizar a localização do personagem no jogo. Esta classe contém os seguintes atributos e métodos:

  RADIANS: atributo do tipo *float* para armazenar valor radiano para transformação de latitude e longitude para unidades do Unity 3D.

  olat: atributo do tipo *float* para marcar a latitude do ponto zero no mapa.

  m_myPlayer: atributo do tipo *GameObject* para armazenar informação de qual personagem é o do jogador.

  isLocalPlayerSet: atributo do tipo *boolean* para armazenar informação se o personagem local foi atribuído ou não.

CacheLat: atributo do tipo *float* para armazenar temporariamente a latitude do usuário.
cacheLon: atributo do tipo *float* para armazenar temporariamente a longitude do usuário.
M_chatManager: objeto do tipo *VCC_ChatManager* para criar um objeto de gerenciamento de chat para o personagem.
GetLocalPlayer: método para capturar informação de qual é o personagem do usuário.
DebugMobile: método para mostrar mensagem de erro para fins de debug.
SetPlayerCoordinatesByGps: método para capturar a informação de latitude e longitude a partir do GPS e atribuir ao personagem.
SetPlayerPositionByGps: método para definir a posição no mapa do jogo equivalente a posição de latitude e longitude
Translate_coordinates: método para transformar os dados de latitude e longitude para dados de posicionamento no Unity 3D.
Meters_deglat: método para transformar metros da latitude em graus.
Meters_deglon: método para transformar metros da longitude em graus.
DEG_TO_RADIANS: método para transformar os graus em radianos.

- **TransformMobile**

  Esta classe é responsável por atualizar os valores das *prefabs* dos personagens no jogo para mobile, caso seja móvel é necessário alterar tamanho, colisão e gravidade. Esta classe contém os seguintes atributos e métodos:
  m_option: atributo do tipo *Option* para armazenar se a plataforma é mobile e será necessário alterar as predefinições ou não.
  M_rectvalue: atributo do tipo *RectValues* para armazenar dados sobre o retângulo de colisão do personagem
  M_localPosition: atributo do tipo *Vector3* para armazenar a localização do personagem.
  M_localScale: atributo do tipo *Vector3* para armazenar a escala dos personagens que serão exibidos.
  M_makeRigidBodyStatic: atributo do tipo *boolean* para armazenar dado se o personagem deve ser rígido ou não.
  M_syncPositionOnly: atributo do tipo *boolean* para armazenar dado se o personagem é

atualizado somente via GPS ou não.

M_rect: atributo do tipo *RectTransform* para armazenar se a colisão do personagem precisará ser alterada ou não.

Awake: método para verificar a plataforma e chamar os outros métodos.

Update: método para verificar os valores e atualizar os parâmetros a partir da plataforma.

UpdateValues: método para atualizar os valores das predefinições dos personagens.

- **GoogleMap**

  Esta classe é responsável por gerar o mapa do jogo a partir da API do GoogleMaps para mobile partindo do ponto zero do software base. Esta classe contém os seguintes atributos e métodos:

  MapType: atributo do tipo *Enum* para armazenar se o mapa será satélite, ruas ou hibrido por número.

  loadOnStart: atributo do tipo *boolean* para armazenar dado se o mapa será carregado ou não.

  autoLocateCenter: atributo do tipo *boolean* para armazenar dado se o mapa estará centrado no ponto de criação do mesmo ou não.

  centerLocation: atributo do tipo *GoogleMapLocation* para armazenar onde estará centrado o mapa exibido.

  Zoom: atributo do tipo *int* para armazenar a aproximação no mapa.

  mapType: atributo do tipo *MapType* para armazenar informação do mapa para desenho do mesmo.

  Size: atributo do tipo *int* para armazenar o tamanho do mapa em pixels quadrados.

  doubleResolution: atributo do tipo *boolean* para armazenar se o mapa vai ser com melhor resolução ou não.

  Markers: atributo do tipo *GoogleMapMarker* para armazenar as posições de interesse no mapa.

  Paths: Atributo do tipo *GoogleMapPath* para armazenar as rotas do mapa.

  Start: método para inicializar o serviço do mapa.

  Refresh: método para atualizar o mapa.

GoogleMapColor: método para desenhar o mapa em cima do terreno do Unity.

- **MobileInput**

  Esta classe é responsável por gerar o teclado e receber o texto do chat a partir do Android. Esta classe contém os seguintes atributos e métodos:

  M_sendMessage: atributo do tipo *VCC_SendMessage* para armazenar os métodos de envio de mensagem

  OnEndEdit: método para enviar o texto a partir da perda do foco na caixa de digitação.

## 5.4 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência tem por objetivo documentar os eventos que fazem parte de um processo específico, documentando os métodos utilizados por cada um dos atores e os objetos envolvidos no processo, tendo sempre em vista que neste diagrama deve-se manter a ordem dos acontecimentos correta (Guedes, 2014).

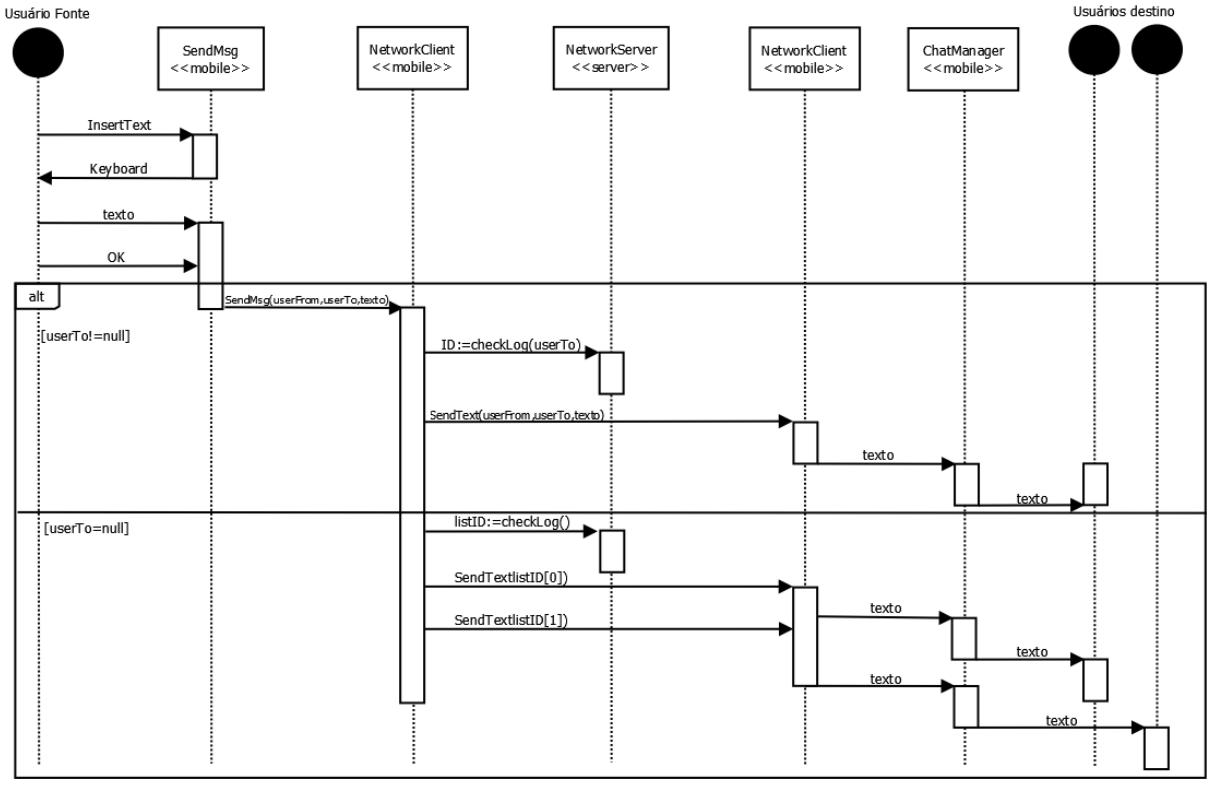



**Figura 8 –** Diagrama de sequência do funcionamento de envio de mensagem no chat. Autoria própria

A figura 8 apresenta um diagrama de sequência que descreve as interações de mensagens entre objetos e atores para envio de uma mensagem de chat. As mensagens iniciais entre o ator “Usuário Fonte” e o objeto da classe SendMsg representam a digitação da mensagem (com a escolha do destinatário) e sua confirmação de envio (“OK”). O objeto NetworkClient verifica se a mensagem tem um destinatário exclusivo ou será uma mensagem aberta a todos os personagens. Conforme a resultado desta verificação, um dos dois blocos de mensagens é seguido (bloco “alt”). É feita, a seguir, a checagem de login no objeto NetworkServer e são enviadas as mensagens com o texto do chat para o objeto NetworkClient do jogador destinatário que apresenta a mensagem através de seu objeto ChatManager.

A figura 9 apresenta um diagrama de sequência que descreve as interações de mensagens entre objetos e atores para atualização da posição do jogador conforme haja mudança de sua posição percebida pela variação de seu GPS.

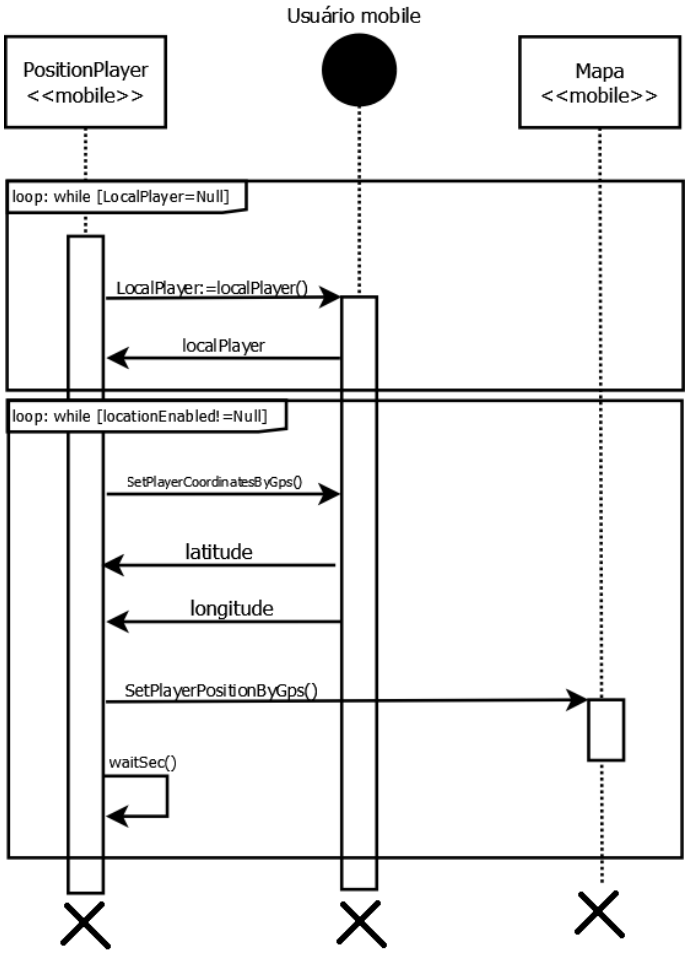



**Figura 9 –** Diagrama de sequência da captura de localização e desenho no Unity. Autoria própria

## 5.5 CONSTRUÇÃO DA SOLUÇÃO

Para a realização da implementação da solução foi utilizado o Unity3d, pois o mesmo já havia sido utilizado no desenvolvimento do CWB-VP para desktop, tornando-se assim, a opção mais adequada para resolver o problema. Foi utilizado também, o servidor e a base de dados que já haviam sido implementados na etapa em que foi desenvolvido o CWB-VP. O servidor foi escrito em Java e opera através de webservices que são chamados pelos *scripts* vinculados aos componentes dentro do Unity. Já a base de dados, foi construída com SQL Server 2014 Express. Na base de dados, atualmente, são salvos os dados de login dos usuários, a última posição em que um jogador estava antes de sair do jogo, notícias, informações meteorológicas, a lista de amigos do jogador, o modelo de personagem selecionado pelos usuários e as linguagens.

Para a construção geral do software como, por exemplo, a interface gráfica, mapas, personagens, física do mundo e ações de botões, foi utilizado o suporte próprio do Unity3D, que não exige programação nenhuma. De fato, é necessário apenas arrastar alguns componentes e selecionar suas configurações e características. A figura 10 ilustra a interface padrão e estrutura de um projeto no Unity.

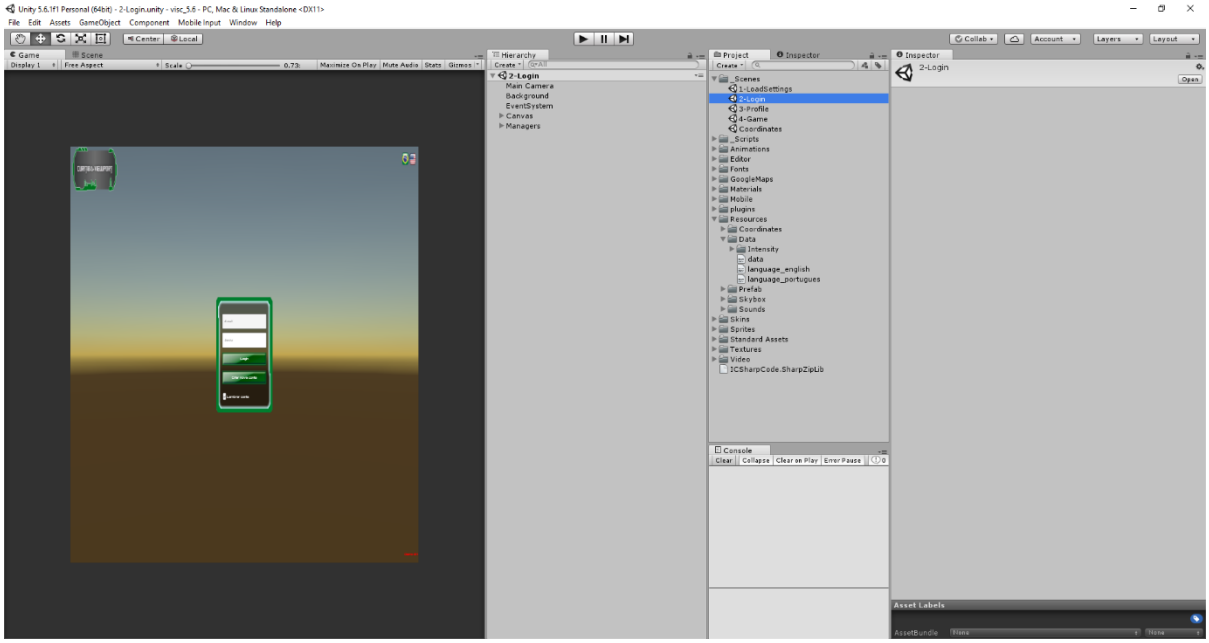


**Figura 10 –** Estrutura de arquivos de um projeto no Unity. Autoria própria.

Além desta configuração, também se fez necessário desenvolver *scripts* que foram utilizados para o funcionamento da solução, como por exemplo, fazer a tela de *login* se comunicar com o servidor para validação das credenciais, ou então enviar a localização atual para que o jogador possa ser modelado na versão para desktop do jogo, todos os scripts foram escritos em C#. A seguir, serão descritas as funcionalidades da solução mais detalhadamente.

### a) Cena de Login

Para poder manter a identidade visual já previamente estabelecida na solução, a tela de login da versão mobile foi construída com base na tela utilizada na versão desktop do CWB-VP. Nessa tela da solução, o usuário informa suas credenciais e envia ao servidor através de um script para validação. Aqui o usuário também pode escolher por manter seu login salvo

para as próximas vezes que ele for utilizar o software.


**Figura 11 -** Tela de Login. Autoria própria

**b) Cena de Seleção de Personagem**

Nesta parte o usuário pode selecionar um entre três modelos de personagens, para ser utilizado como seu avatar. Os modelos disponibilizados são os mesmos apresentados na solução construída para *desktop*.

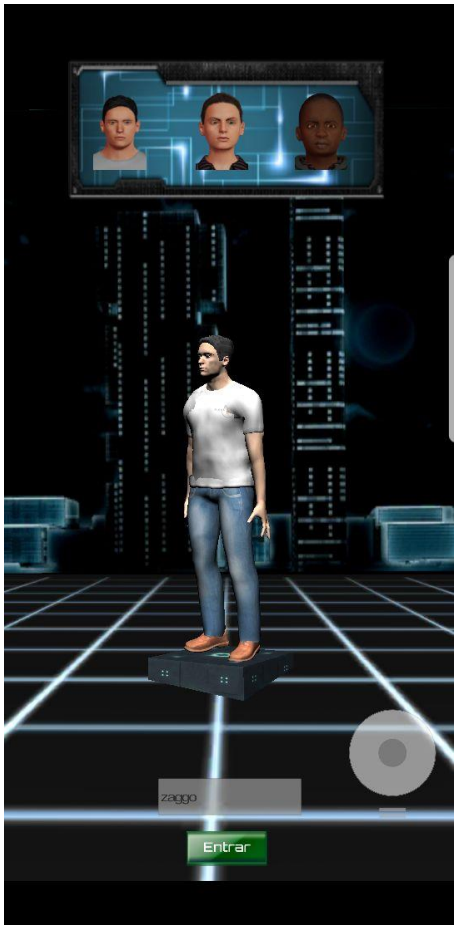



**Figura 12 –** Tela de Seleção de Personagem. Autoria própria.

**c) Cena de Mapa e Chat**

Esta é a tela com as funcionalidades principais do escopo. Nela, o avatar selecionado pelo usuário irá aparecer no mapa, localizado nas coordenadas atuais fornecidas pelo sistema de GPS do smartphone. Conforme o usuário se movimenta no mundo real, o avatar que estiver sendo utilizado por ele no mapa se move simultaneamente. Também é possível visualizar os avatares de outros jogadores se movimentando no mapa, assim como será possível visualizar os jogadores que estiverem utilizando a versão mobile através do desktop. Para criar o mapa,

foi utilizada a biblioteca Google Maps for Unity. Esta biblioteca carrega um mapa do Google para dentro do jogo e transforma este mapa em uma textura utilizável dentro do jogo (Google Maps for Unity).

A segunda parte desta tela é a parte de comunicação via mensagens instantâneas, na qual o jogador poderá receber e enviar mensagens tanto entre amigos através de mensagens privadas quanto para um chat geral. Para enviar ao chat geral, basta o usuário digitar a mensagem na região inferior da tela e clicar no botão de envio. Já para enviar uma mensagem para um amigo, o usuário irá clicar na parte superior da tela na lista de amigos e selecionar um amigo que estiver disponível, ao fazer isso todas as mensagens digitadas na tela de chat serão reencaminhadas para o amigo que foi selecionado.


**Figura 13 –** Tela de Mapa e Chat. Autoria própria

Para visualizar e interagir melhor no programa para dispositivos móveis foram criados alguns scripts para transformação dos objetos pré fabricados utilizados na versão base para PC. No script TransformMobile foram desenvolvidos procedimentos que realizam a alteração de tamanho, mudança na fisica e mudança de comportamento interativo entre os personagens no jogo. Por exemplo alguma das alterações realizadas foram: aumentar o tamanho dos personagens mobile em 40 vezes para mostrar acima do mapa no celular e para os usuários de PC não mostrar a diferença, personagens no mobile não possuem física, pois eles podem acabar ficando presos em prédios por saltos na localização do GPS e também não conseguem esbarrar nos outros personagens.

Foram alterados no projeto base os scripts de controle das cenas do jogo, pois as cenas para *desktop* não se adaptam automaticamente para a tela do celular. Para cada cena na versão PC foi criada uma para versão mobile, com os mesmos procedimentos porém com estruturas e comportamentos completamente diferentes. O Unity3D realiza a diferenciação da cena que precisa executar a partir de uma validação (conforme código apresentado a seguir), na qual tudo que for colocado nessa validação será executado somente em um ambiente Android mesmo que esteja dentro de uma *build* para outra plataforma.

```
#if  UNITY_ANDROID
#endif
```

O *script* responsável por capturar as informações de localização do usuário a partir do GPS é o PositionPlayer que se encontra em anexo ao documento. Este *script* está atrelado à cena do mapa no programa desenvolvido para dispositivos móveis. Nele estão contidos métodos responsáveis pela captura de dados, transformação dos dados de latitude e longitude em posições X, Y e Z no mapa do Unity 3D e redesenhar o personagem na posição correta.

A captura dos dados ocorre a cada segundo para não haver erro no posicionamento. Primeiramente é validado se o GPS do usuário está habilitado, caso esteja o *script* verifica se o usuário permitiu o programa a acessar o GPS do aparelho, e por fim antes de realizar a captura o script valida se não houve falha do GPS em encontrar o aparelho. Quando os dados são capturados, o *script* chama o método translate_coordinates o qual foi reutilizado a partir do projeto base para a conversão dos dados.

## 5.6 VERIFICAÇÃO DO SOFTWARE

Durante todas as etapas do desenvolvimento do projeto foram realizados diversos testes para garantir a qualidade do sistema desenvolvido. Foram feitos testes de mesa, com *fake GPS,* e testes em produção utilizando *smartphones* e também a versão *desktop* do jogo.

Primeiramente, ao fim do desenvolvimento de cada um dos *scripts* que foram utilizados na solução foi feito um teste de mesa para garantir que a lógica estava correta. Para testar o funcionamento do GPS em uma escala maior, foi utilizado o aplicativo *Fake GPS Location* versão 1.0.2.3, que simula uma localização no GPS do celular, e, então, através desta localização simulada foi possível verificar o funcionamento do aplicativo com movimentação do usuário em tempo real. Além disso, foram feitos testes com deslocamentos reais do dispositivo.

Após o término do desenvolvimento, foram feitos os testes do ambiente em produção, através da utilização de todas as funções do aplicativo com diferentes dispositivos em diferentes quantidades. Foram feitos três casos de testes:

- o primeiro caso de teste envolveu apenas um *smartphone* executando o jogo (Samsung S8 Plus). Durante este teste foram testadas as funções de *login* e de georreferenciamento.
- Oosegundo caso de teste foi feito com dois *smartphones* (Samsung S8 Plus e Asus Zenfone 3 Max). Nesta etapa de testes foram verificadas as funcionalidades de visualização de outros jogadores utilizando o aplicativo em plataforma *mobile* e o envio de mensagens entre jogadores mobile.
- por fim, no útimo caso de de teste, foram feitas verificações envolvendo os dois modelos de *smartphones* citados anteriormente, assim como uma instância do CWB-VP executando em *desktop*. Nesta última fase dos testes de produção, foram testadas as funcionalidades de visualizar um jogador *mobile* através do jogo em *desktop* e vice-versa, assim como a troca de mensagens entre jogadores operando em diferentes plataformas do jogo.

## 5.7 RESULTADOS E DISCUSSÕES

O resultado final alcançado pela realização deste trabalho foi um software que opera em uma plataforma *mobile* e será utilizado como complemento a uma plataforma já existente para cidades virtuais. As possibilidades de utilização do artefato alcançado, assim como as funcionalidades, previamente estabelecidas como objetivos foram completamente desenvolvidas e o aplicativo já se encontra completamente utilizável e está operando em sincronia adequada com a plataforma *desktop* para a qual este foi desenvolvido como complemento.

Durante o desenrolar do trabalho foram encontrados alguns impeditivos para a alcançar o funcionamento estável entre plataformas. O georreferenciamento foi complexo de resolver, pois quando criado para a plataforma Windows não havia uma prova real de utilização do GPS para confirmação do posicionamento. Havia somente cálculos de transformação de latitude e longitude para unidades do Unity. Após corrigir o posicionamento, a maior parte do trabalho foi relacionada a recriar os *canvas* e *prefabs* do projeto base feito para PC e adaptá-los para funcionamento em *smartphones*.

# 6 CONCLUSÕES

Durante a elaboração da proposta, este projeto demonstrou-se bastante ambicioso devido às várias concepções que são consideradas novas. Também devido à grandeza da elaboração do software como um todo, dificilmente este seria viável de se implementar. Porém, foi possível realizar um estudo aprofundado sobre o tópico, assim como um levantamento de requisitos e desenvolver um software experimental que permite avaliar os conceitos de cidade virtual.

Levando-se em consideração que a exploração dos conceitos de jogos em ambientes realísticos ainda não é muito aprofundada, especialmente quando se refere a cidades virtuais espelhadas, o projeto poderá servir como ferramenta de estudos a respeito dos impactos causados por uma ferramenta desta magnitude, social, política e economicamente.

Para a realização deste projeto, foram cumpridas algumas etapas de desenvolvimento. Inicialmente, foram realizados estudos referentes aos conceitos que poderiam ser aplicáveis no escopo, como cidades inteligentes, ciberespaço e cidades virtuais. Após estes conceitos terem sido esclarecidos e definidos apropriadamente, foram estudadas as possibilidades de desenvolver melhorias para o projeto já existente e iniciado, denominado CWB-VP (Curitiba ViewPort). Após decidir que seria desenvolvido um aplicativo mobile, foi iniciada a fase de especificação e desenvolvimento do projeto.

Para trabalhos futuros, poderão ser desenvolvidas novas funcionalidades tanto para a plataforma *mobile* como para a versão *desktop*. Existem diversas funcionalidades simples que podem ser implementadas ainda no projeto, como por exemplo:

- Criação de micro transações que possibilitem ao jogador comprar roupas e acessórios para personalizar seu avatar.
- Adaptar o sistema de dia e noite para o aplicativo mobile.
- Inserir anúncios patrocinados no aplicativo.

Também existem melhorias de uma complexidade mais elevada que podem ser incluídas nas aplicações, como por exemplo:

- Adicionar na plataforma *desktop* a possibilidade de utilizar meios de transporte, mais especialmente ônibus, sendo que este se moveria automaticamente e utilizaria a rota real e as paradas reais.

- Desenvolver a possibilidade de integração com outros periféricos, como microfone para chat via voz, *joysticks* ou até mesmo dispositivos de realidade virtual.
- Criar uma funcionalidade para traçar rotas no aplicativo, tendo em vista auxiliar no deslocamento dentro da cidade.
- Integrar o aplicativo com tecnologias de realidade aumentada, transformando os pontos turísticos da cidade em pontos de interesse dentro do jogo, com informações sobre eles sendo acessíveis a partir da interface do aplicativo, baseando-se na localização do jogador, por exemplo.

Este trabalho de conclusão de curso aplicou conhecimentos de várias disciplinas do curso de Bacharelado em Sistemas de Informação da UTFPR. Algumas disciplinas mais diretamente associadas são as de programação, análise e modelagem, engenharia de software e design e interação. Pode-se observar que o conhecimento adquirido nestas disciplinas forneceram os fundamentos necessários para a equipe cumprir todos os objetivos do projeto proposto.

Finalmente, os autores deste trabalho destacam o valor da experiência de desenvolvimento do projeto de conclusão. Este projeto permitiu integrar conhecimentos, colocar em prática os ensinamentos adquiridos no curso e produzir uma contribuição a um projeto de pesquisa relevante sobre cidades virtuais desenvolvido na UTFPR.

# REFERÊNCIAS


BARACHO, Ana Flávia de Oliveira ; GRIPP, Fernando Joaquim ; LIMA, Márcio Roberto de. *Os exergames e a educação física escolar na cultura digital.* Florianópolis: Rev. Bras. Ciênc. Esporte, v. 34, n. 1, p. 111-126, jan./mar, 2012.

CASTELLS, Manuel. *The New Public Sphere: Global Civil Society, Communication Networks and Global Governance.* Filadélfia: The ANNALS of the American Academy of Political and Social Science, 2008.

DESENVOLVEDORES ANDROID, *Site official do android para desenvolvedores.* Disponível em: <https://developer.android.com/> Acesso em: 10/06/2017

FERNANDES, Ricardo Jorge Lopes ; FERNANDES, Rui Jorge Gama. *A cidade digital vs a cidade inteligente: estratégias de desenvolvimento sócio-económico e/ou de marketing territorial.* Braga: Actas do 2º Congresso Luso-brasileiro para o Planeamento Urbano, Regional, Integrado e Sustentável, Universidade do Minho, 2006.

FERNANDES, Ricardo ; GAMA, Rui. *As cidades e territórios do conhecimento na óptica desenvolvimento e do marketing territorial.* Coimbra: FLUC - Universidade de Coimbra, 2006.

FLING, Brian. *Mobile Design and Development.* Sebastopol: O'Really Media, Inc., 2009.

FROSTBITE, *Site oficial do Frostbite.* Disponível em: <https://www.ea.com/frostbite> Acesso em 11/06/2017

GREGORY, Jason, *Game Engine Architecture (2 ed.),* 2014

GUEDES, Gilleanes TA. *UML 2–Guia Prático-2ª Edição*. Novatec Editora, 2014.

IDC. *Smartphone OS Market Share, 2016 Q3.* Disponível em: < http://www.idc.com/promo/smartphone-market-share/os > Acesso em: 10/06/2017

LEMOS, André. *O que é cidade digital?,* 2006. Disponível em <http://www.guiadascidadesdigitais.com.br/site/pagina/o-que-cidade-digital> Acesso em: 15/03/2017.

LIMA, Rodrigo Monteiro de. *Doctraining: Um ambiente 3D com jogo sério para o treinamento de estudantes de medicina em casos clínicos.* Mossoró: Dissertação (Mestrado em Ciência da Computação), Universidade Federal Rural do Semi-Árido – UFERSA, 2016.

LENCIONI, Sandra. *Observações sobre o conceito de cidade e urbano.* São Paulo: GEOUSP - Espaço e Tempo, Nº 24, pp. 109 - 123, 2008.

MELO, Ana Cristina. *Desenvolvendo Aplicações com UML 2.2*. Brasport, 2004.

MENTZAS, Gregoris ; APOSTOLOU, Dimitris ; ABECKER, Andreas. *Managing knowledge as a strategic resource for electronic government.* Workshop Electronic Government and Electronic Management, 2001.

MIRANDA, Maria João Pinto Luiz. *Jogo sério para reabilitação neurocognitiva: Cidade virtual.* Porto: Dissertação (Mestrado Integrado em Engenharia Informática e Computação). Faculdade de Engenharia da Universidade do Porto, 2012.

MORAES, Fernando Dreissing de. *A "cidade digital" de Porto Alegre(RS): Um estudo sobre espaço urbano e tecnologias de informação e comunicação a partir da apropriação do estado e de grupos (ciber)ativistas.* Porto Alegre: Dissertação (Mestrado em geografia), Universidade do Rio Grande do Sul, 2012.

MUSSO, Pierre. *Télé-politique: Le sarkoberlusconisme à l'écran,* 2009.

OGRE Manual v1.8('Byatis'). *Documentação oficial do OGRE*. Disponível em < http://www.ogre3d.org/docs/manual/> Acesso em 11/06/2017.

OHTA, Yuichi ; TAMURA, Hideyuki. *Mixed Reality: Merging Real and Virtual Worlds (1 ed.),* 2014

OLIVEIRA, Cecília Leite. *A revolução tecnológica e a dimensão humana da informação: a construção de um modelo de mediação.* Brasília: Tese (Doutorado em Ciência da Informação), Universidade de Brasília - UnB, 2003.

OLIVENZA, Ismael Sagrado; PUGA, Gonzalo Flórez; MARTÍN, Marco Antonio Gómez; CALERO, Pedro A.. *UHotDraw: a GUI Framework to Simplify Draw Application Development in Unity 3D.* Madrid: Universidad Conplutense de Madrid, 2013.

PRESSMAN, Roger S. ; MAXIM, Bruce R.. *Engenharia de Software: Uma Abordagem Profissional.* Nova Iorque: 8ª Edição, AMGH Editora Ltda, 2016.

SANTOS, Milton. *O Espaço dividido: os dois circuitos da economia urbana dos países subdesenvolvidos.* São Paulo: Edusp, 2. Ed., 2004.

SERRANO, António ; GONÇALVES, Fernando ; NETO, Paulo. *Cidades e Territórios do Conhecimento – Um novo referencial para a competitividade*. Lisboa: Edições Sílabo, 2005.

SIMÃO, João Batista. *A concepção de um modelo de cidade digital baseado nas necessidades informacionais do cidadão: o caso dos municípios brasileiros de pequeno porte.* Brasília: Dissertação (Doutorado em Ciência da Informação), Universidade de Brasília - UnB, 2010.

SOUTO, Átila Augusto ; DALL'ANTONIA, Juliano Castilho ; HOLANDA, Giovani de. *As cidades digitais no mapa do Brasil: uma rota para a inclusão digital.* Brasília: Ministério das comunicações, 2001.

STEVENTON, Alan ; WRIGHT, Steve. *Intelligent spaces: The application of pervasive ICT*. Londres: Springer Science & Business Media, 2010.

STOCKHOLM 360. *Virtual Tours, Fullscreen spherical panoramas from Stockholm, Sweden and other places.* Disponível em: <http://stockholm360.net/>. Acesso em 15/03/2017.

TAVAKKOLI, ALIREZA. *Game Development and Simulation with Unreal Technology.* CRC Press, 2015.

UNITY TECHNOLOGIES. *Site oficial do Unity*. Disponível em: <https://unity3d.com/>. Acesso em: 30/06/2015.

UNITY USER MANUAL (5.6). *Documentação oficial do Unity*. Disponível em: < https://docs.unity3d.com/Manual/> Acesso em: 12/06/2017

UNREAL ENGINE 4 DOCUMENTATION. *Documentação oficial da Unrial Engine 4.* Disponível em: <https://docs.unrealengine.com/> Acesso em 11/06/2017.

WORLD OF WARCRAFT. *Site oficial do jogo.* Disponível em: <https://worldofwarcraft.com/>. Acesso em 01/06/2017.

X-PLANE 11. *Site oficial do jogo.* Disponível em: <http://www.x-plane.com/>. Acesso em 01/06/2017.

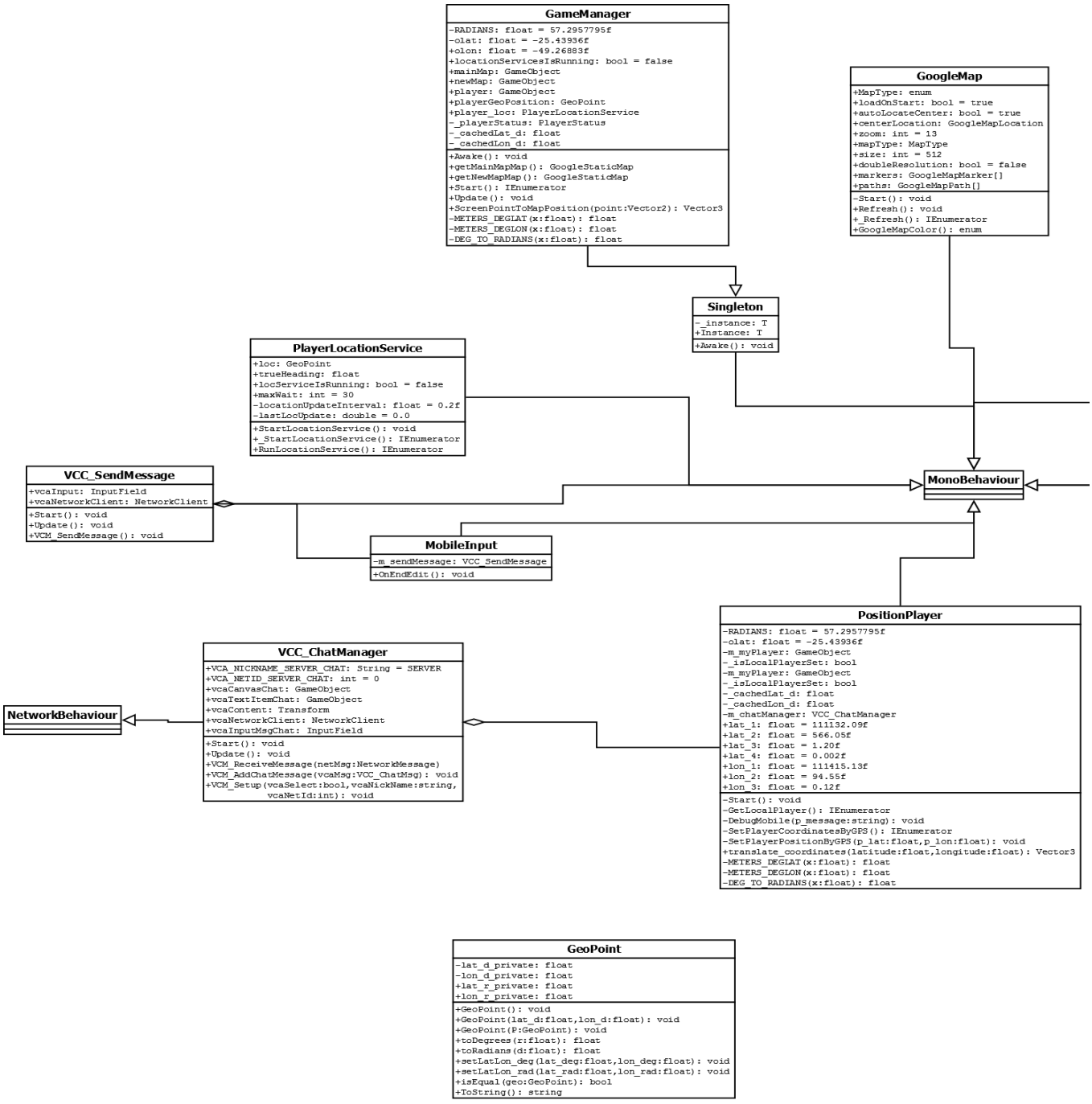
ZANCHETI, Sílvio Mendes. *Cidades Digitais e o desenvolvimento local*. Recife: RECITEC v.5, n.2, p.311-329, 2001.

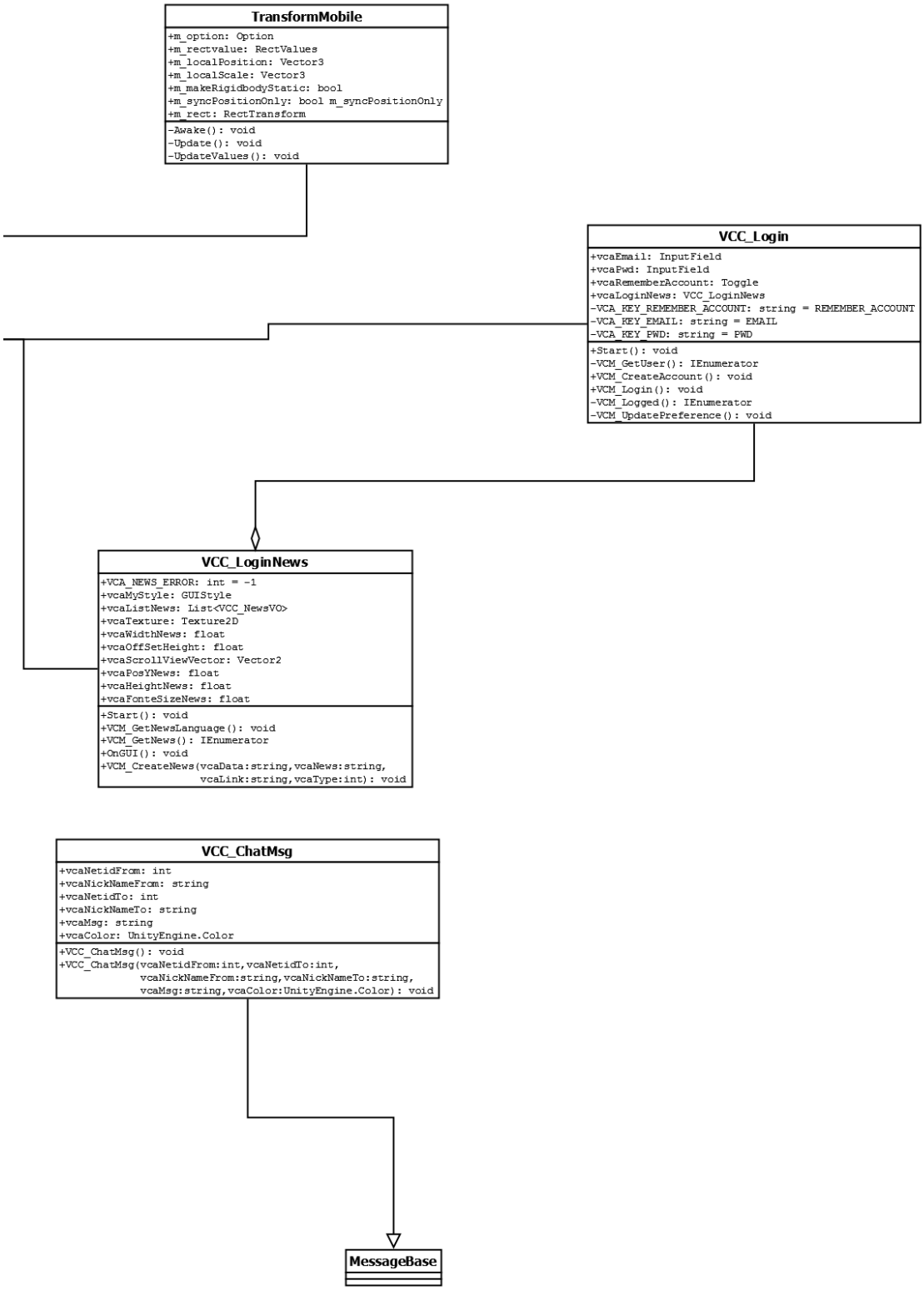
ZUBIETA, Roberto, WOODLEY, Tedy. *Manual Para el Desarrollo de Ciudades Digitales en Iberoamerica.* Madrid: Asociación Hispanoamericana de Centros de Investigación y Empresas de Telecomunicaciones - AHCIET, 2006.

ZWEERS, Koen ; PLANQUÉ, Kees. *Electronic Government. From a organizational base perspective towards a client oriented approach.* Boston: Kluwer Law International, 2001.

# ANEXOS

ANEXO A – “Diagrama-Classes.png” Diagrama de classes desenvolvido para melhor entendimento da aplicação *mobile.*




TransformMobile
+m_option: Option
+m_rectvalue: RectValues
+m_localPosition: Vector3
+m_localScale: Vector3
+m_makeRigidbodyStatic: bool
+m_syncPositionOnly: bool m_syncPositionOnly
+m_rect: RectTransform
-Awake(): void
-Update(): void
-UpdateValues(): void
VCC_Login
+vcaEmail: InputField
+vcaPwd: InputField
+vcaRememberAccount: Toggle
+vcaLoginNews: VCC_LoginNews
-VCA_KEY_REMEMBER_ACCOUNT: string = REMEMBER_ACCOUNT
-VCA_KEY_EMAIL: string = EMAIL
-VCA_KEY_PWD: string = PWD
+Start(): void
-VCM_GetUser(): IEnumerator
+VCM_CreateAccount(): void
+VCM_Login(): void
-VCM_Logged(): IEnumerator
-VCM_UpdatePreference(): void
VCC_LoginNews
+VCA_NEWS_ERROR: int = -1
+vcaMyStyle: GUIStyle
+vcaListNews: List<VCC_NewsVO>
+vcaTexture: Texture2D
+vcaWidthNews: float
+vcaOffSetHeight: float
+vcaScrollViewVector: Vector2
+vcaPosYNews: float
+vcaHeightNews: float
+vcaFonteSizeNews: float
+Start(): void
+VCM_GetNewsLanguage(): void
+VCM_GetNews(): IEnumerator
+OnGUI(): void
+VCM_CreateNews(vcaData:string,vcaNews:string,
vcaLink:string,vcaType:int): void
VCC_ChatMsg
+vcaNetidFrom: int
+vcaNickNameFrom: string
+vcaNetidTo: int
+vcaNickNameTo: string
+vcaMsg: string
+vcaColor: UnityEngine.Color
+VCC_ChatMsg(): void
+VCC_ChatMsg(vcaNetidFrom:int,vcaNetidTo:int,
vcaNickNameFrom:string,vcaNickNameTo:string,
vcaMsg:string,vcaColor:UnityEngine.Color): void
MessageBase

ANEXO B – “PlayerLocationService.cs” Script responsável por iniciar o serviço de validação do GPS.

```
using UnityEngine;
using System.Collections;

public class PlayerLocationService : MonoBehaviour {

	public GeoPoint loc = new GeoPoint();
	[HideInInspector]
	public float trueHeading;
	public bool locServiceIsRunning = false;
	public int maxWait = 30; // seconds
	private float locationUpdateInterval = 0.2f; // seconds
	private double lastLocUpdate = 0.0; //seconds



	public void StartLocationService() {
		Debug.Log ("Player Loc started.");
		StartCoroutine (_StartLocationService ());
	}

	public IEnumerator _StartLocationService()
	{

		// First, check if user has location service enabled
		if (!Input.location.isEnabledByUser) {
			Debug.Log ("Locations is not enabled.");

	    //NOTE: If location is not enabled, we initialize the postion of the
	    player to somewhere in Los Angeles, just for demonstration purposess
	    loc.setLatLon_deg (-25.439043f, -49.269868f);

		  GameManager.Instance.playerStatus =
		  GameManager.PlayerStatus.FreeFromDevice;
			// To get the game run on Editor without location services
			locServiceIsRunning = true;
			yield break;
		  }

		  // Start service before querying location
		  Input.location.Start();
		  // Wait until service initializes
		  while (Input.location.status == LocationServiceStatus.Initializing &&
		  maxWait > 0)
		  {
			yield return new WaitForSeconds(1);
			maxWait--;
		  }

		  // Service didn't initialize in maxWait seconds
		  if (maxWait < 1)
		  {
			print("Locations services timed out");
			yield break;
```

```
		}

		// Connection has failed
		if (Input.location.status == LocationServiceStatus.Failed)
		{
			print("Location services failed");
			yield break;
		} else if (Input.location.status == LocationServiceStatus.Running){
			GameManager.Instance.playerStatus =
GameManager.PlayerStatus.TiedToDevice;
			loc.setLatLon_deg (Input.location.lastData.latitude,
Input.location.lastData.longitude);
			Debug.Log ("Location: " +
Input.location.lastData.latitude.ToString ("R") + " " +
Input.location.lastData.longitude.ToString ("R") + " " +
Input.location.lastData.altitude + " " + Input.location.lastData.horizontalAccuracy +
" " + Input.location.lastData.timestamp);
			locServiceIsRunning = true;
			lastLocUpdate = Input.location.lastData.timestamp;
		} else {
			print ("Unknown Error!");
		}
		Debug.Log (loc.ToString());
	}

	public IEnumerator RunLocationService()
	{
		double lastLocUpdate = 0.0;
		while (true) {
			if (lastLocUpdate != Input.location.lastData.timestamp) {
				loc.setLatLon_deg (Input.location.lastData.latitude,
Input.location.lastData.longitude);
				trueHeading = Input.compass.trueHeading;
				Debug.Log ("Location: " +
Input.location.lastData.latitude.ToString ("R") + " " +
Input.location.lastData.longitude.ToString ("R") + " " +
Input.location.lastData.altitude + " " + Input.location.lastData.horizontalAccuracy +
" " + Input.location.lastData.timestamp);
				//locServiceIsRunning = true;
				lastLocUpdate = Input.location.lastData.timestamp;
			}
			yield return new WaitForSeconds(locationUpdateInterval);
		}
	}
}
```

ANEXO C – “PositionPlayer.cs” Script responsável por atualizar a localização do personagem no jogo.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityStandardAssets.Characters.ThirdPerson;

public class PositionPlayer : MonoBehaviour {
    private const float RADIANS = 57.2957795f;
    private const float olat = -25.43936f;
    private const float olon = -49.26883f;

    private GameObject m_myPlayer;
    private bool _isLocalPlayerSet;

    private float _cachedLat_d;
    private float _cachedLon_d;

    private void Start() {
        StartCoroutine(GetLocalPlayer());
    }

    private IEnumerator GetLocalPlayer() {
        while (ClientScene.localPlayers == null || ClientScene.localPlayers.Count <=
0) {
            yield return new WaitForSeconds(1f);
        }
        m_myPlayer = ClientScene.localPlayers[0].gameObject;
        _isLocalPlayerSet = true;
        StartCoroutine(SetPlayerCoordinatesByGPS());
    }

    //public float Lat = -25.437810f;
    //public float Lon =  -49.267277f;
    //private float _counter;
    //private void Update() {
    //    _counter += Time.deltaTime;
    //    if (_counter > 1f) {
    //        SetPlayerPositionByGPS(Lat, Lon);
    //        _counter = 0f;
    //    }
    //}

    private VCC_ChatManager m_chatManager;

    private void DebugMobile(string p_message) {
        Debug.Log(p_message);
        if (m_chatManager == null) {
            VCC_ChatMsg msg = new VCC_ChatMsg(0, 0, "Debug", "", p_message,
Color.white);
            FindObjectOfType<VCC_ChatManager>().VCM_AddChatMessage(msg);
        }
    }
```

```
    private IEnumerator SetPlayerCoordinatesByGPS() {
        if (!Input.location.isEnabledByUser) {
            DebugMobile("Location is not enabled by user.");
            yield break;
        }

        //Client has location enabled. Start Service then.
        Input.location.Start();
        DebugMobile("Starting location service.");

        //Give 20 seconds for Location Service to Start
        int maxWait = 20;
        while (Input.location.status == LocationServiceStatus.Initializing && maxWait
> 0) {
            yield return new WaitForSeconds(1);
            DebugMobile("Waiting for location service to start.");
            maxWait--;
        }

        // Service didn't initialize in 20 seconds
        if (maxWait < 1) {
            DebugMobile("Location Service time out. Shutting down.");
            yield break;
        }

        // Connection has failed
        if (Input.location.status == LocationServiceStatus.Failed) {
            DebugMobile("Location Service failed. Shutting down.");
            yield break;
        }

        while (Input.location.isEnabledByUser) {
            if (Input.location != null) {

                SetPlayerPositionByGPS(Input.location.lastData.latitude,
Input.location.lastData.longitude);
                yield return new WaitForSeconds(1f);
            }
        }

        Input.location.Stop();
    }

    private void SetPlayerPositionByGPS(float p_lat, float p_lon) {
        if (!_isLocalPlayerSet) {
            return;
        }
        Vector3 pos = translate_coordinates(p_lat, p_lon);
        pos = new Vector3(pos.x, m_myPlayer.transform.position.y, pos.z);

        DebugMobile(string.Format("Movendo jogador para {0}", pos.ToString()));
        m_myPlayer.transform.position = pos;
    }

    private Vector3 translate_coordinates(float latitude, float longitude) {

        float xx, yy, r, ct, st, angle;
```

```
        angle = DEG_TO_RADIANS(0);

        xx = (longitude - olon) * METERS_DEGLON(olat);
        yy = (latitude - olat) * METERS_DEGLAT(olat);

        r = Mathf.Sqrt(xx * xx + yy * yy);

        if (r > 0) {
            ct = xx / r;
            st = yy / r;
            xx = r * ((ct * Mathf.Cos(angle)) + (st * Mathf.Sin(angle)));
            yy = r * ((st * Mathf.Cos(angle)) - (ct * Mathf.Sin(angle)));
        }


        return new Vector3(xx, 0f, yy);// + new Vector3(24.4f, 0f, 37.93f);
    }

    public float lat_1 = 111132.09f;
    public float lat_2 = 566.05f;
    public float lat_3 = 1.20f;
    public float lat_4 = 0.002f;

    public float lon_1 = 111415.13f;
    public float lon_2 = 94.55f;
    public float lon_3 = 0.12f;

    private float METERS_DEGLAT(float x) {
        float d2r = DEG_TO_RADIANS(x);
        return (lat_1 - (lat_2 * Mathf.Cos(2.0f * d2r)) + (lat_3 * Mathf.Cos(4.0f *
d2r)) - (lat_4 * Mathf.Cos(6.0f * d2r)));
    }

    private float METERS_DEGLON(float x) {
        float d2r = DEG_TO_RADIANS(x);
        return ((lon_1 * Mathf.Cos(d2r)) - (lon_2 * Mathf.Cos(3.0f * d2r)) + (lon_3 *
Mathf.Cos(5.0f * d2r)));
    }

    private float DEG_TO_RADIANS(float x) {
        return (x / RADIANS);
    }
}
```

ANEXO D – “TransformMobile.cs” Script responsável por atualizar os valores das prefabs dos personagens no jogo para mobile.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

[ExecuteInEditMode]
public class TransformMobile : MonoBehaviour {
#if UNITY_ANDROID

    [SerializeField]
    private Option m_option;
    private enum Option {
        None = 0,
        RectTransform,
        LocalPosition,
        LocalScale,
        LocalRotation
    }

    [SerializeField]
    private RectValues m_rectvalue;

    [SerializeField]
    private Vector3 m_localPosition;

    [SerializeField]
    private Vector3 m_localScale;

    [SerializeField]
    private Vector3 m_localRotation;

    [SerializeField]
    private bool m_makeRigidbodyStatic;

    [SerializeField]
    private bool m_syncPositionOnly;

    private RectTransform m_rect;

    [System.Serializable]
    private class RectValues {
        public float Left;
        public float Top;
        public float Right;
        public float Bottom;
        public Vector2 AnchorMin;
        public Vector2 AnchorMax;
        public Vector2 Pivot;
    }

    private void Awake () {
        m_rect = GetComponent<RectTransform>();
        UpdateValues();
```

```
    }

       private void Update () {
        UpdateValues();
    }

    private void UpdateValues() {
        switch (m_option) {
            case Option.LocalScale:
                transform.localScale = m_localScale;
                break;
            case Option.LocalPosition:
                transform.localPosition = m_localPosition;
                break;
            case Option.RectTransform:
                //Debug.LogWarning("TODO: Setar esse cara");
                break;
            case Option.LocalRotation:
            transform.localRotation = Quaternion.Euler(m_localRotation.x,
m_localRotation.y, m_localRotation.z);
                break;
        }

        if (m_makeRigidbodyStatic) {
            //this.GetComponent<Rigidbody>().isKinematic = true;
            //this.GetComponent<Rigidbody>().useGravity = false;
            this.GetComponent<Rigidbody>().constraints =
RigidbodyConstraints.FreezePositionY;
            Physics.gravity = new Vector3(0f, 0f, 0f);
        }

        if (m_syncPositionOnly) {
            this.GetComponent<NetworkTransform>().transformSyncMode =
NetworkTransform.TransformSyncMode.SyncTransform;
        }
    }
#endif
}
```

ANEXO E – “GoogleMap.cs” Script responsável por gerar o mapa do jogo a partir da API do Google Maps para mobile.

```
using UnityEngine;
using System.Collections;

[ExecuteInEditMode]
public class GoogleMap : MonoBehaviour
{
	public enum MapType
	{
		RoadMap,
		Satellite,
		Terrain,
		Hybrid
	}
	public bool loadOnStart = true;
	public bool autoLocateCenter = true;
	public GoogleMapLocation centerLocation;
	public int zoom = 13;
	public MapType mapType;
	public int size = 512;
	public bool doubleResolution = false;
	public GoogleMapMarker[] markers;
	public GoogleMapPath[] paths;

	private void Start() {
		if(loadOnStart) Refresh();
	}

    //private float _counter;
    //private float _refreshTime = 1f;

    //private void Update() {
    //    _counter += Time.deltaTime;
    //    if (_counter > _refreshTime) {
    //        _counter = 0;
    //        Refresh();
    //    }
    //}

    public void Refresh() {
		if(autoLocateCenter && (markers.Length == 0 && paths.Length == 0)) {
			Debug.LogError("Auto Center will only work if paths or markers
are used.");
		}
		StartCoroutine(_Refresh());
	}

	private IEnumerator _Refresh ()
	{
		var url = "http://maps.googleapis.com/maps/api/staticmap";
		var qs = "";
		if (!autoLocateCenter) {
			if (centerLocation.address != "")
```

```
                        qs += "center=" + WWW.UnEscapeURL
(centerLocation.address);
                    else {
                        qs += "center=" + WWW.UnEscapeURL (string.Format
("{0},{1}", centerLocation.latitude, centerLocation.longitude));
                    }

                    qs += "&zoom=" + zoom.ToString ();
            }
            qs += "&size=" + WWW.UnEscapeURL (string.Format ("{0}x{0}", size));
            qs += "&scale=" + (doubleResolution ? "2" : "1");
            qs += "&maptype=" + mapType.ToString ().ToLower ();
            var usingSensor = false;
#if UNITY_IPHONE
            usingSensor = Input.location.isEnabledByUser && Input.location.status ==
LocationServiceStatus.Running;
#endif
            qs += "&sensor=" + (usingSensor ? "true" : "false");

            foreach (var i in markers) {
                qs += "&markers=" + string.Format
("size:{0}|color:{1}|label:{2}", i.size.ToString ().ToLower (), i.color, i.label);
                foreach (var loc in i.locations) {
                    if (loc.address != "")
                        qs += "|" + WWW.UnEscapeURL (loc.address);
                    else
                        qs += "|" + WWW.UnEscapeURL (string.Format
("{0},{1}", loc.latitude, loc.longitude));
                }
            }

            foreach (var i in paths) {
                qs += "&path=" + string.Format ("weight:{0}|color:{1}", i.weight,
i.color);
                if(i.fill) qs += "|fillcolor:" + i.fillColor;
                foreach (var loc in i.locations) {
                    if (loc.address != "")
                        qs += "|" + WWW.UnEscapeURL (loc.address);
                    else
                        qs += "|" + WWW.UnEscapeURL (string.Format
("{0},{1}", loc.latitude, loc.longitude));
                }
            }

        url += "?" + qs;
        url += "&key=AIzaSyCF83pgZddXnerlPQHxWh6dhersl_wDI1Y";

        Debug.Log(url);
        var req = new WWW(url);
        yield return req;
        this.GetComponent<MeshRenderer>().sharedMaterial.mainTexture = req.texture;
    }
}

public enum GoogleMapColor
{
      black,
```

```
        brown,
        green,
        purple,
        yellow,
        blue,
        gray,
        orange,
        red,
        white
}

[System.Serializable]
public class GoogleMapLocation
{
        public string address;
        public float latitude;
        public float longitude;
}

[System.Serializable]
public class GoogleMapMarker
{
        public enum GoogleMapMarkerSize
        {
                Tiny,
                Small,
                Mid
        }
        public GoogleMapMarkerSize size;
        public GoogleMapColor color;
        public string label;
        public GoogleMapLocation[] locations;

}

[System.Serializable]
public class GoogleMapPath
{
        public int weight = 5;
        public GoogleMapColor color;
        public bool fill = false;
        public GoogleMapColor fillColor;
        public GoogleMapLocation[] locations;
}
```

ANEXO F – “MobileInput.cs” Script responsável por gerar o teclado mobile e receber o texto do chat a partir do Android.

```
using System;
using UnityEngine;

namespace UnityStandardAssets.CrossPlatformInput.PlatformSpecific
{
    public class MobileInput : VirtualInput
    {
       [SerializeField] private VCC_SendMessage m_sendMessage;
        private void AddButton(string name)
        {
            // we have not registered this button yet so add it, happens in the
constructor
            CrossPlatformInputManager.RegisterVirtualButton(new
CrossPlatformInputManager.VirtualButton(name));
        }


        private void AddAxes(string name)
        {
            // we have not registered this button yet so add it, happens in the
constructor
            CrossPlatformInputManager.RegisterVirtualAxis(new
CrossPlatformInputManager.VirtualAxis(name));
        }


        public override float GetAxis(string name, bool raw)
        {
            if (!m_VirtualAxes.ContainsKey(name))
            {
                AddAxes(name);
            }
            return m_VirtualAxes[name].GetValue;
        }


        public override void SetButtonDown(string name)
        {
            if (!m_VirtualButtons.ContainsKey(name))
            {
                AddButton(name);
            }
            m_VirtualButtons[name].Pressed();
        }


        public override void SetButtonUp(string name)
        {
            if (!m_VirtualButtons.ContainsKey(name))
            {
                AddButton(name);
            }
            m_VirtualButtons[name].Released();
```

```
}


public override void SetAxisPositive(string name)
{
    if (!m_VirtualAxes.ContainsKey(name))
    {
        AddAxes(name);
    }
    m_VirtualAxes[name].Update(1f);
}


public override void SetAxisNegative(string name)
{
    if (!m_VirtualAxes.ContainsKey(name))
    {
        AddAxes(name);
    }
    m_VirtualAxes[name].Update(-1f);
}


public override void SetAxisZero(string name)
{
    if (!m_VirtualAxes.ContainsKey(name))
    {
        AddAxes(name);
    }
    m_VirtualAxes[name].Update(0f);
}


public override void SetAxis(string name, float value)
{
    if (!m_VirtualAxes.ContainsKey(name))
    {
        AddAxes(name);
    }
    m_VirtualAxes[name].Update(value);
}


public override bool GetButtonDown(string name)
{
    if (m_VirtualButtons.ContainsKey(name))
    {
        return m_VirtualButtons[name].GetButtonDown;
    }

    AddButton(name);
    return m_VirtualButtons[name].GetButtonDown;
}


public override bool GetButtonUp(string name)
{
```

```
            if (m_VirtualButtons.ContainsKey(name))
            {
                return m_VirtualButtons[name].GetButtonUp;
            }

            AddButton(name);
            return m_VirtualButtons[name].GetButtonUp;
        }


        public override bool GetButton(string name)
        {
            if (m_VirtualButtons.ContainsKey(name))
            {
                return m_VirtualButtons[name].GetButton;
            }

            AddButton(name);
            return m_VirtualButtons[name].GetButton;
        }


        public override Vector3 MousePosition()
        {
            return virtualMousePosition;
        }

       public void OnEndEdit() {
       #if UNITY_ANDROID
               m_sendMessage.VCM_SendMessage();
       #endif
    }
    }
}
```

Ministério da Educação
**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**Câmpus Curitiba**
Diretoria de Graduação e Educação Profissional
***Departamento Acadêmico de Informática***
***Coordenação do Curso de Bacharelado em Sistemas de Informação***

## TERMO DE APROVAÇÃO

# "DESENVOLVIMENTO DE UM SISTEMA MÓVEL PARA INTEGRAÇÃO DE USUÁRIOS A UM AMBIENTE DE CIDADE VIRTUAL

por

## "Gabriel José Lazarine
## Henrique Luiz Granato de Quadros do Nascimento"

Este Trabalho de Conclusão de Curso foi apresentado às _____**hs** do dia 06 de **Julho** de **2017** como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho _____.

| | |
|---|---|
| _____<br>**<Prof. Paulo Cezar Stadizisz>**<br>(Presidente - UTFPR/Curitiba) | _____<br>**<Prof. João Alberto Fabro>**<br>(Avaliador 1 – UTFPR/Curitiba) |
| _____<br><**Prof. Robson Ribeiro Linhares>**<br>(Avaliador 2 – UTFPR/Curitiba) | _____<br>**<Prof. Leyza Elmeri Baldo Dorini>**<br>(Professor Responsável pelo TCC – UTFPR/Curitiba) |
| _____<br>**<Prof. Leonelo Dell Anhol Almeida>**<br>(Coordenador(a) do curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba) | |

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso."