
**UNIVERSIDADE FEDERAL DO CEARÁ**
**CAMPOS DE QUIXADÁ**
**CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE**

**IURY MAIA DE ALMEIDA**

# HUNTAPI: UMA PLATAFORMA DE CRIAÇÃO DE JOGOS PERVASIVOS BASEADOS EM LOCALIZAÇÃO.

**QUIXADÁ**
**2016**

**IURY MAIA DE ALMEIDA**

# HUNTAPI: UMA PLATAFORMA DE CRIAÇÃO DE JOGOS PERVASIVOS BASEADOS EM LOCALIZAÇÃO.

Monografia apresentada ao Programa de Curso em Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia de Software. Área de concentração: Jogos eletrônicos, Arquitetura de software, Software livre.

Orientador: Profa. Dra. Paulyne Matthews Jucá.

QUIXADÁ

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

---

A447h Almeida, Iury Maia de.
HUNTAPI : Uma Plataforma de criação de jogos pervasivos baseados em localização / Iury Maia de Almeida. – 2016.
46 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Dr. Paulyne Matthews Jucá.

1. Jogos eletrônicos. 2. Arquitetura de software. 3. Software livre. I. Título.

CDD 005.1

2016

IURY MAIA DE ALMEIDA

# HUNTAPI: UMA PLATAFORMA DE CRIAÇÃO DE JOGOS PERVASIVOS BASEADOS EM LOCALIZAÇÃO.

Monografia apresentada ao Programa de Curso em Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia de Software. Área de concentração: Jogos eletrônicos, Arquitetura de software, Software livre.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Profa. Dra. Paulyne Matthews Jucá. (Orientadora)
Universidade Federal do Ceará (UFC)

_____
Prof. Mrs. Camilo Camilo Almendra
Universidade Federal do Ceará (UFC)

_____
Prof. Mrs. Bruno Góis Mateus
Universidade Federal do Ceará (UFC)

Aos meus pais, Maria de Fátima e Lino Alves.

## AGRADECIMENTOS

A Profa. Dra. Paulyne Matthews Jucá, pela excelente orientação.

Aos professores participantes da banca examinadora Camilo Camilo Almendra e Bruno Góis Mateus pelo tempo, pelas valiosas colaborações e sugestões.

Aos meus pais que sempre me ajudaram nessa jornada.

A todos que sonharam comigo e acreditaram que era possível.

## RESUMO

O trabalho apresenta a criação de uma plataforma de desenvolvimento de jogos pervasivos baseados em localização, com o objetivo de fornecer ao desenvolvedor um conjunto de ferramentas que envolvem os aspectos de geolocalização, persistência de dados, comunicação com o servidor para o sistema operacional Android. Essa plataforma oferece ferramentas ao desenvolvimento de tipos de jogos similares a caçadas ao tesouro. Para o desenvolvimento desta plataforma, uma arquitetura cliente-servidor foi escolhida utilizando diversas tecnologias como postgrees e o elasticsearch, bem como o google maps para geolocalização. Como resultados esse trabalho apresenta a implementação da plataforma e o modelo de dados bem como a definição da arquitetura. Além disso, a plataforma é validada através do desenvolvimento de um jogo utilizando a plataforma proposta e desenvolvida nesse trabalho.
**Palavras-chave**: Jogos Pervasivos. Arquitetura de Software. Jogo Geolocalizados.

## ABSTRACT

This paper proposes the creation of a pervasive game development platform based on location, in order to provide the developer with a set of pre -configured tools, involving tools aspects of geolocation, data persistence, exchange messages between client and server for the operation system Android. This platform provides tools to develop similar types of games to treasure hunts. For the development of this platform, we chose a client server architecture for the development of mobile games based on location, and will be used various tools such as Postgres and ElasticSearch as well the Google Maps for geolocation and soon as results have implementation and model platform data with the definition of its architecture, Furthermore it is shown the application developed on the platform.

**Keywords**: Pervasive Games. Software architecture. Geolocation games.

# LISTA DE FIGURAS

# LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| Http | Protocolo de Transferência de Hipertexto. |
| API | Interface de Programação de Aplicativos |
| URL | Localizador Padrão de Recursos |

# SUMÁRIO

# 1 INTRODUÇÃO

O surgimento de novas tecnologias como sensores que captam informação do ambiente e uso do GPS(“Global Positioning System”) capaz de identificar sua localização no globo, possibilitam a criação de novas mecânicas e de novas formas de interação com os jogos, ampliando assim o mercado de jogos. Um exemplo de novos jogos que nasceram da introdução dessas novas tecnologias são os jogos pervasivos. “Os jogos pervasivos são uma nova forma de jogo que amplia as experiências dos jogos de *video game* para o mundo físico”(Vázquez; sal, 2009).

As ferramentas utilizadas no desenvolvimento de jogos vem evoluindo, indo desde a modelagem de elementos ao uso de plataformas para auxiliar na criação e concepção de um novo game *design*. “Contudo a criação de um jogo ainda é um processo complexo e dispendioso”(NOLÊTO et al, 2014). Segundo Moore (2015),“jogos são desenvolvidos usando uma plataforma de desenvolvimento de jogos que possa facilitar a entrega do jogo”. Além disso o desenvolvimento de um jogo digital envolve profissionais de diferentes ramos do conhecimento como músicos, animadores, desenhistas, roteiristas, programadores.

Assim, quando o jogo a ser desenvolvido adiciona a todos esses fatores citados a colaboração de um sistema sensível a contexto, a complexidade do desenvolvimento do sistema aumenta. O uso de plataformas ajuda a abstrair essa complexidade. “Plataformas prometem alta produtividade e baixo *time-to-market* no desenvolvimento de aplicações através de design e reúso de código”(Riehle; 2000). Logo para ajudar no desenvolvimento de jogos pervasivos o presente trabalho desenvolveu uma plataforma que sustenta e agiliza a utilização de diversas funcionalidades, entregando para o desenvolvedor um conjunto de ferramentas pré-configuradas para que o mesmo se preocupe mais com as regras do seu próprio jogo e menos com os detalhes mais intrínsecos do desenvolvimento.

O objetivo desse trabalho é o desenvolvimento de uma plataforma, que possa satisfaça os requisitos evidenciados para auxiliar no desenvolvimento de jogos pervasivos. Para esse trabalho, uma plataforma é definida como “Uma plataforma de software é um conjunto de componentes que oferecem serviços similares para diferentes aplicações”(KUBOTA, Flávio Issao et al, 2013). A plataforma desenvolvida tem que ser capaz de persistir dados dos usuários, ser capaz de criar caçadas ou tesouro, oferecer ao desenvolvedor meios para consultar esses dados e uma forma de conseguir a geolocalização dos usuários. A plataforma tem como vantagens colocar a disposição do programador classes já configuradas e prontas

para serem usadas. Além disso, a plataforma é responsável por toda a troca de mensagens entre os clientes e o servidor do jogo desenvolvido.

A plataforma ainda dá suporte a geolocalização sendo capaz de atualizar automaticamente a posição do jogador, atualizar automaticamente novas *quest* no mapa, criar rodas para uma *quest* determinada, dentre outras funcionalidades.

As próximas páginas deste trabalho estão divididas em seções. Assim, a segunda seção apresenta os trabalhos relacionados que descreve trabalhos semelhantes. A terceira seção apresenta os conceitos chaves necessários para o embasamento teórico do trabalho. A quarta seção apresenta as tecnologias escolhidas nesse trabalho. A quinta seção apresenta os processos metodológicos que contém o passo-a-passo de como foi realizado do desenvolvimento bem como a definição da plataforma. A sexta seção apresenta os resultados encontrados do trabalho. Assim por fim a oitava é referente à conclusão do trabalho.

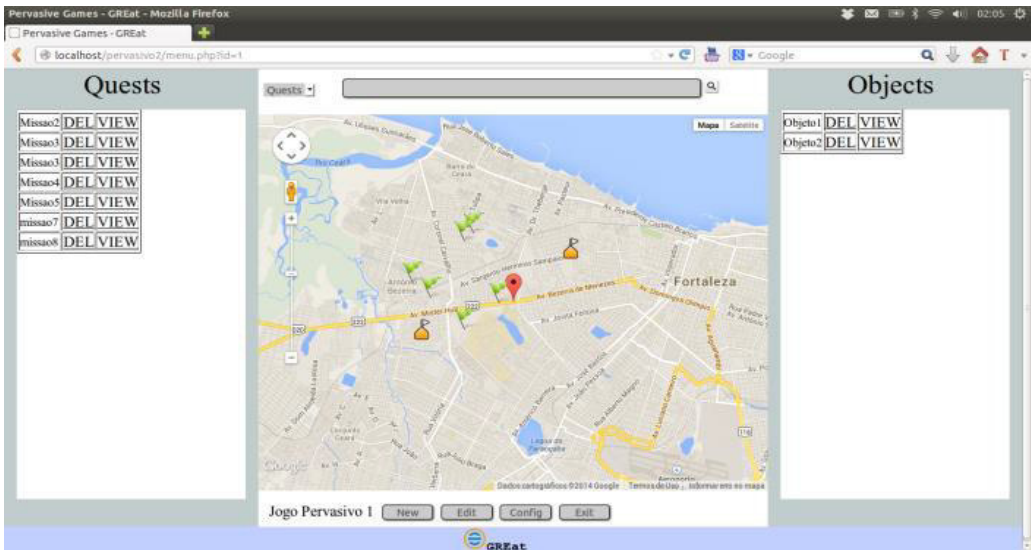
# 2 TRABALHOS RELACIONADOS

A seguir serão apresentados os trabalhos relacionados. Para cada trabalho serão detalhados e explicados os objetivos dos estudos, bem como suas semelhanças e diferenças com o trabalho proposto.

## 2.1 Uma ferramenta de autoria para desenvolvimento de jogos pervasivos baseados em realidade aumentada.

Nôleto et al (2014) propôs uma ferramenta de autoria para criação de jogos pervasivos. Segundo Nôleto et al (2014), “uma ferramenta de autoria deve permitir que seus usuários criem aplicações usando um conjunto de elementos pré-configurados (e.g., mecânicas, personagens)”. Os autores enfocam também que a ferramenta deve possuir uma manipulação intuitiva para que alguém sem nenhum conhecimento de programação possa utilizá-las. Essa característica é considerada muito importante pelos autores do trabalho relacionado, para atender ao intuito de criar jogos pervasivos de formas mais simples dando a possibilidade para uma nova parcela da população de fazer uso desta tecnologia.

Figura 1:Protótipo de tela da ferramenta autoria.




Fonte: Carleandro O.Nolêto (2014), p.2.

A ferramenta de autoria proposta engloba vários aspectos de desenvolvimento e utilização, incluindo a possibilidade de geolocalização, utilização de objetos através de realidade aumentada, definição de *quests* que são um conjunto de passos para alcançar um determinado objetivo ou superar um desafio como exemplificado na figura 1.

O presente trabalho possui uma proposta semelhante. Primeiro os dois trabalhos têm o intuito de desenvolver jogos pervasivos com tecnologia de geolocalização, contudo o público foco da ferramenta de Nolêto el al não requer conhecimento de programação enquanto o presente trabalho foca em desenvolvedores. O segundo ponto importante é a ferramenta de Nolêto el al possui um escopo muito maior que o abordado nesse trabalho. Ela dá a possibilidade de utilizar realidade aumentada, contudo o presente trabalho foca apenas no desenvolvimento de jogos baseados em localização com enfoque em caça ao tesouro, sem a utilizar realidade aumentada.

A plataforma proposta nesse trabalho pode atender às necessidades de um público-alvo, como desenvolvedores e usuários mais capacitados que necessitam de uma maior flexibilidade. Além disso a ferramenta proposta por Nôleto et al não se encontra disponível enquanto o presente trabalho é open source e disponibilizado no Git Hub onde qualquer um poderá ter acesso e desenvolver com a plataforma.

O trabalho desenvolvido tem o intuito de fornecer ferramentas para o desenvolvimento de um tipo de jogo pervasivo (jogo de "caça ao tesouro" geolocalizado), sem o intuito de ser genérico como o trabalho de Nôleto et al (2014).

## 2.2 Um jogo multi-jogador baseado em localização para encorajar comportamentos pró-ambientais.

Centieiro et al (2011) utiliza tecnologia pervasiva para criar um jogo chamado GAEA com o intuito de incentivar mudanças comportamentais destinadas a melhor consciência sobre reciclagem entre outros aspectos.

GAEA é um jogo pervasivo, baseado em localização, multiplayer e móvel em que os jogadores são incentivados a coletar materiais recicláveis em uma área específica. Ele foi desenvolvido para ser jogado em espaço aberto, onde os jogadores se movem pelo local coletando lixo virtual com seus celulares. Os objetos são relacionados em uma determinada área geográfica real onde os jogadores precisam estar fisicamente para poder coletá-los. O jogo possui também outro tipo de interação que são os pontos de coleta, áreas também definidas geograficamente onde existem um telão público passando e apresentado o atual estado do jogo, como é mostrado na figura 2. Além disso, nessas áreas poderá ser feita a reciclagem correta do seu lixo virtual. No final do jogo é apresentado um questionário sobre o objeto reciclado e perguntas ambientais para reforçar o conhecimento adquirido.

Figura 2: Reciclando objeto virtual.

Fonte: Pedro Centieiro 2011, p. 2.

A plataforma desenvolvida neste trabalho auxilia o desenvolvimento de jogos similares, também baseados em localização sendo essa uma das grandes semelhanças. Aliado ao desenvolvimento de um jogo semelhante, dar suporte à criação de objetivos para áreas específicas é outro ponto em comum dos trabalhos, já que para os jogadores poderem participar das atividades propostas eles têm de estar em um local físico real. Adicionalmente as perspectivas de design e interação dos jogadores com seus celulares e no mundo físico serão semelhantes e utilizadas nas formas de interação da plataforma proposta pelo trabalho. Porém a principal diferença é que o GAEA é um jogo pervasivo enquanto o presente trabalho apresenta uma plataforma para o desenvolvimento de jogos similares ao GAEA.

## 2.3 Criando jogos baseados em localização escaláveis: lições do GeoCaching.

Neustaedter, Tang e Judge (2011) mostra um dos maiores jogos geolocativos do mercado com mais de 2 mil caçadas ao tesouro pelo mundo. Afirma Kiefer (2006) “O GeoCaching é um dos mais antigos e maiores jogos baseados em localização”. O jogo é um *massively multiplayer online* em que pessoas do mundo todo criam suas caçadas aos tesouros chamados de *Cache* exemplificados na figura 3. Os próprios jogadores determinam uma área geográfica e uma quantidade de passos ou uma charada e o jogador que estiver disponível e

próximo da exata área onde a *Cache* foi criada pode participar dessa caçada ou tesouro.

O ponto mais forte desse estudo são os elementos de *design* para jogos baseados em localização que o GeoCache propõem. O trabalho faz uma análise detalhada sobre esses elementos, como acontece a ações realizadas entre os jogadores e o jogo e como a comunidade do jogo interage e se ajuda, evidencia elementos importantíssimos de diversão e satisfação que estão presentes na construção e ideologia do jogo. Assim, esse trabalho torna-se essencial para a plataforma do trabalho uma vez que o mesmo incluiu funcionalidades semelhantes às eliciadas sobre o GeoCaching, tornando mais simples o identificar os requisitos da plataforma pois a pesquisa por elementos e interações que ele possui será baseada no GeoCache.

Figura 3: Um tradicional GeoCache.


Fonte: Carman Neustaedter(2011) p. 339.

O presente trabalho, porém não se comprometeu com a demonstração do design dos jogos desenvolvidos pela plataforma. São apresentadas todas as decisões referentes ao desenvolvimento da arquitetura da plataforma, uma vez o objetivo deste trabalho é a disponibilização de uma plataforma que seja capaz de atender e recriar jogos semelhantes ao GeoCaching. Além disso, o GeoCache não é uma plataforma de criação de jogos baseados em localização, mas um jogo baseado em localização com suas próprias regras e desafios.

## 3 FUNDAMENTAÇÃO TEÓRICA

A computação ubíqua esta revolucionando nossa forma de interagir com novas tecnologias e, nesse contexto, surgem os jogos pervasivos. Jogos pervasivos englobam aspectos e ações reais para ampliar a experiência do jogador. Dentro dos jogos pervasivos,

estão os jogos baseados em localização que utilizam a real posição do jogador para se adaptar e interagir com o mundo virtual.

As seções a seguir apresentam os conceitos de computação ubíqua , jogos pervasivos e jogos móveis locativos.

## 3.1 Computação Ubíqua

Computação Ubíqua foi um termo cunhado por Mark Weiser (1991) no seu artigo *The Computer for the 21st Century.* Neste artigo Weiser demonstrou uma nova visão sobre como a informática poderia se relacionar com o homem, trazendo a ideia de integrar a informática com as ações comportamentais e naturais humanas.

Hoje a computação está espalhada nos mais diversos dispositivos e eletrodomésticos – geladeira, sensores, celulares, etc. Um *smartphone* hoje pode ter tanto poder de processamento quanto um computador pessoal. Com esse cenário a informática invade cada vez mais espaços, podendo ser levada a locais remotos e a quase todos os lugares urbanos. Com todos esses dispositivos podendo comunicar-se e integrados à internet, estamos cada vez mais próximos de visão de Weiser (1991) que declara que “a computação torna-se onipresente, embutida em diversos objetos do dia a dia”.

A computação ubíqua possui três pilares fundamentais: a integração natural com o ser humano, tecnologias inteligentes e comunicação. A integração natural com o ser humano ocorre através de interfaces intuitivas, as quais abstraem as interações com os aparelhos tecnológicos da forma mais natural possível. As tecnologias inteligentes se referem a dispositivos que são capazes de compreender onde estão e adaptar-se dependendo da situação. A comunicação então acontece dentre todos os dispositivos de uma área que tem possibilidade de troca de dados e status entre eles ou com a própria internet. “A computação ubíqua pode se apresentar por meio de diferentes dispositivos de várias formas e tamanhos (WEISER; 1991)”. Atualmente o foco da computação ubíqua é o contexto do usuário e como ele interage com o mesmo como sua localização física, temperatura, sons, etc.

A plataforma proposta por este trabalho tem aspectos de ubiquidade, uma vez que o mesmo é destinado a dispositivos móveis, que podem ser levados a quase todos os lugares, como smartphones. Além disso, ele necessita de informações do contexto do usuário, mais especificamente sua posição no globo, sendo necessária uma comunicação com as ferramentas de geolocalização e a plataforma a ser desenvolvida.

## 3.2 Jogos Pervasivos

Hoje com ampliação das tecnologias como sensores e o GPS, são responsáveis por novas configurações de jogos eletrônicos, que por sua vez criam novos tipos de experiências que seriam inviáveis de se concretizar através de meios tradicionais. Hinske et al (2007), define que “os jogos pervasivos são uma forma lúdica de entretenimento de realidade mista com objetos, regras, competições e ataques baseados na utilização de computação móvel e/ou tecnologias de computação pervasiva”.

Montola (2009) em seu livro intitulado “Jogos pervasivos: Teoria e design”, afirmou que “jogos pervasivos utilizam um ou mais recursos para ampliar o conceito de círculo mágico em qualquer âmbito seja ele espacial, temporal ou social”, ou seja, ampliar a experiência com o jogo e sua forma de interação fazendo com que ações no mundo real repercutam nas regras do jogo influenciando o mundo virtual. O círculo mágico foi um termo cunhado por HUIZINGA (1955) no qual ele se referia ao local físico ou virtual onde as regras do jogo fazem sentido e são aceitas por todos que fazem parte do jogo.

“De modo geral os jogos são considerados fenômenos transculturais que acompanham a humanidade desde os primórdios, sempre relacionando a algum tipo de aprendizado ou ganho cognitivo” (Andrade; Luiz Adolfo, 2013). Os jogos pervasivos ampliam a experiência para o mundo físico mudando radicalmente a forma de jogar. Como um exemplo bem simples de tal interação, “uma pessoa está andando pela cidade e de repente recebe uma mensagem através do seu smartphone, que informa sobre uma partida, na qual esta pessoa estava jogando anteriormente e seguiu decorrendo, e já ter um vencedor”(Vázquez; sal, 2009). Trazendo o exemplo para o trabalho o jogador possui a escolha de se juntar à caçada ao tesouro ou desistir. A aplicação ficaria responsável por misturar a caçada ao tesouro real(elementos presentes no mundo real) com e prêmios virtuais ou pontuação para o jogador dentro do jogo. A computação ubíqua que possibilita tal interação fazendo com que jogos ou qualquer outra aplicação sejam executados em dispositivos distintos.

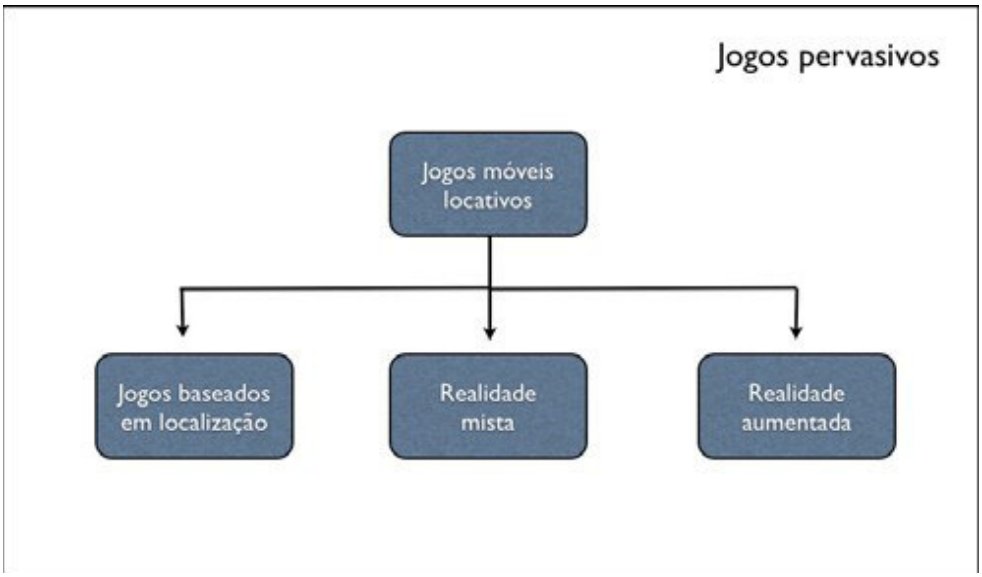
O conceito de jogos pervasivos é aplicado à plataforma proposta no momento que qualquer jogador para entrar em uma *quest* que esteja acontecendo na região próxima. O jogador tem de se deslocar ao início da *quest*, ou seja, sua exata posição definida pelo GPS tem de estar muito próxima do local da *quest*, assim sua posição influencia as regras do jogo, pois sem ela não seria possível entrar em nenhum tipo de jogo. Além disso, todos os jogos seguem com passos e ações a serem tomadas no mundo real e cada passo superado no desafio

escolhido influencia também no mundo virtual do jogo.

### 3.3 Jogos Móveis Locativos

“Jogos móveis, ou mobile games, são jogos baseados em dispositivos portáteis que oferecem uma experiência de entretenimento independentemente de locais determinados” (Mont'Alverne, 2012). Contudo os jogos móveis locativos estendem esse conceito adicionando ao jogo características do mundo físico. Mont'Alverne (2012) classificou os jogos móveis locativos em três formas, jogos baseados em localização, jogos de realidade mista e jogos de realidade aumentada, mostradas na figura 4.

Figura 4: Classificação de jogos móveis locativos.




Fonte: Mont'Alverne, 2012, p. 12.

Os jogos baseados em localização aliam tecnologias digitais móveis e sistemas de geolocalização, gerando uma interação entre os espaços eletrônicos e físicos, para desenvolvimento das regras do jogo. De acordo com a classificação proposta por Mont'Alverne (figura 4) para jogos baseados em localização “são jogos que utilizam tecnologias de localização e que integram a posição de um ou de muitos jogadores como elemento fundamental de suas regras” (Mont'Alverne, 2012).

Ele ainda define jogos de realidade mista e de realidade aumentada. Para ele esses conceitos são:

> Jogos de realidade mista: são jogos baseados em localização que acrescentam uma camada virtual ao mundo real do jogador. Os elementos dessa camada virtual não têm um equivalente no mundo físico. Em *Pac-Manhattan*6, jogadores representando os

> personagens do *videogame Pac-Man* correm pelas ruas de Manhattan enquanto são perseguidos por outros jogadores e recolhem *dots* virtuais nas ruas. Esses *dots* aparecem apenas no mapa virtual do jogo e não são representados por nenhum elemento no mundo real. Por isso, essa camada virtual sobre o mundo real do jogador exige certa dose de abstração e imaginação.
>
> Jogos de realidade aumentada: são jogos de realidade mista que criam uma camada virtual sobre o mundo real, possível de ser percebida visualmente pelos jogadores por meio de sensores especiais e capacetes adaptados para captar seus movimentos. Esse conjunto de dispositivos e sensores cria uma perspectiva em primeira pessoa do ambiente do jogador realçado por elementos virtuais, exigindo menos de sua imaginação Mont'Alverne (2012, p. 11).

O presente trabalho desenvolveu uma plataforma para desenvolvimento específico de jogos baseados em localização, como os jogos de caça ao tesouro apresentado pelo GeoCache. Contudo a plataforma não dará suporte aos aspectos de realidade mista nem de realidade aumentada, uma vez que adicionar tais aspectos de realidade mista e aumentada ampliaria a complexidade da plataforma, implicando em mais tempo de desenvolvimento. Por esta razão o presente trabalho focará apenas em jogos baseados em localização.

# 4 Tecnologias

Nessa seção, serão relatadas as tecnologias utilizadas na construção da plataforma, qual o objetivo delas e porque elas foram escolhidas.

## 4.1 Android

O Android é um sistema operacional desenvolvido pela Google, “Consiste em uma nova plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux” (LECHETA. Ricardo, 2013). A plataforma dará suporte em sua primeira versão para ao Android, pois ele é um dos sistemas operacionais móveis mais usados, de acordo com a IDC (2016) “o Android domina o mercado de smartphones com uma parcela de 87.6%”. Além disso, para o desenvolvimento ele é mais acessível do que os demais sistemas operacionais uma vez que possui uma boa quantidade de informação disponível.

## 4.2 postgresSQL

O postegresSQL é um banco de dados relacional, estando a mais de 15 anos no mercado

e possui uma grande reputação com a integridade dos dados. Como um sistema de banco de dados relacional o postregresSQL tem diversas funções como replicação assíncrona, *backups online*, tablespaces, recuperação de ponto a ponto, mas a principal para o servidor é o armazenamento de dados simples, estes relacionados as classes de persistência como *quest*s, etapa, dados do usuário entre outros.

O Postgres foi escolhido por ser um banco de dados relacional, com ele é possível de forma simples e eficiente mapear as relações entre os dados, persistir e realizar consultas. Além disso ele é um banco de dados open source e pode ser disponibilizado junto com a plataforma.

### 4.3 ElasticSearch

GORMLEY (2015) comenta que "o Elasticsearch é um motor de busca e análise, distribuído, escalável e em tempo real". Além disso, é uma ferramenta de código livre distribuído pela licença apache. O Elasticsearch possui várias características que levaram a sua escolha como, sendo uma ferramenta distribuída e escalável, possuir esquema livre, e funciona através de uma apiREST. Ele possui também variáveis nativas chamadas de geo-points, que são otimizadas para persistir e buscar arquivos com base em coordenadas geográficas. Com tal capacidade torna muito mais fácil manter e pesquisar objetos do jogo com base em suas posições, tornando mais eficiente à plataforma.

### 4.4 Jax.RS.

Rest é um padrão arquitetural que define uma gama de regras, BURKE (2009) comenta que "quando aplicado a sistemas distribuídos, essa arquitetura induz propriedades desejáveis como acoplamento frouxo e escalabilidade horizontal". O Jax.RS é uma API java para implementar *web services*.

Esta é uma biblioteca nativa java para o desenvolvimento de *web services* baseados na arquitetura Restful. "Essa nova API visa fazer o desenvolvimento de RESTful *web services* em java simples e intuitivo" (BURKE, Bill, 2009). Existem outras APIS que também implementam um servidor Rest contudo por esta ser nativa java e intuitiva foi a selecionada para este trabalho.

## 4.5 Google Maps.

O Google Maps é um serviço de pesquisa e visualização de mapas e imagens de satélite da terra gratuito e fornecido pela empresa Google. Hoje existem diversas ferramentas semelhantes no mercado como o yahoo maps, bing maps. SVENNERBERG (2010) em seu livro afirma que 43 por cento das aplicações web estavam usando o google maps, tornando ela a plataforma de geolocalização mais comum. Aliado a grande popularidade o Google maps e a escolha do Android como sistema operacional moveu da plataforma, os dois são desenvolvidos e distribuídos pela mesma empresa, assim provendo similaridades e uma melhor compatibilidade.

Dentre as diversas funções oferecidas estão as de geolocalização (que retorna a posição de algum usuário no globo), a criação de rotas para pontos específicos, além da customização dos mapas de acordo com o usuário.

Figura 5: Exemplo de criação de uma rota.


Fonte: Imagem feita pelo autor.

A utilização dessa ferramenta ajudará no desenvolvimento e na solução de diversos problemas como a manipulação e monitoramento da posição do usuário, a criação de uma rota específica para uma *quest* escolhido no mapa como demonstrado na figura 5. O Google maps é uma das mais completas ferramentas de geolocalização gratuitas, podendo ser

disponibilizada junto com a plataforma de forma livre e por isso foi escolhida nesse trabalho.

# 5 PROCEDIMENTOS METODOLÓGICOS

A seguir, estão detalhados os procedimentos metodológicos utilizados no presente trabalho.

## 5.1 Definindo escopo da plataforma

Nessa tarefa, com o intuito de gerar os requisitos para o sistema e novas ideias para a plataforma, foi evidenciado diversos requisitos baseado nos trabalhos relacionados. Como a possibilidade de criação de regras para criação dos jogos, sistema de pontuação, criação de diversos tipos de jogos, comunicação com servidor online para gerenciamento dos jogos e dos diversos usuários que possam usar os jogos desenvolvidos. Alguns desses elementos podem ser visto nos resultados deste trabalho.

## 5.2 Listar sensores utilizáveis.

Uma vez que a plataforma Android foi selecionada, foram escolhidos os sensores que podem oferecer funções úteis à plataforma de forma geral. Para escolher os sensores foi realizada uma pesquisa por quais sensores o Android poderia oferecer em sua documentação. Foi encontrado três tipos de sensores os de movimento, ambiente e de posição.

Sensores de movimento são responsáveis por medir forças de aceleração e rotação em todas as direções. Essa categoria inclui o acelerômetro, sensores de gravidade. Sensores de ambiente são responsáveis por medir vários parâmetros como a temperatura do ar e pressão, iluminação e umidade esta categoria inclui o barômetro, os fotômetros e termômetros.

Sensores de posição são responsáveis por medir a posição física do dispositivo. Essa categoria inclui sensores de orientação como o GPS e magnetômetros, para o desenvolvimento da plataforma foi concluído que para o escopo desse trabalho seria necessário utilizar apenas sensores de posição, especificamente os de orientação como o GPS.

## 5.3 Desenvolver a plataforma

Esta etapa é referente ao desenvolvimento da plataforma. Primeiramente ela foi desenvolvida para a plataforma móvel Android. Ainda é necessário definir os dispositivos que serão suportados pela plataforma. Logo depois, a linguagem escolhida foi Java, devido ao

bom suporte Android. Também foram decididas algumas plataformas de utilização como, a API do *google maps*, da qual foi extraída a geolocalização do personagem. Essa fase engloba o próprio desenvolvimento responsável pela codificação do sistema.

### 5.4 Desenvolver o jogo

Depois de ter desenvolvido da plataforma foi realizado o desenvolvimento de um caso de teste, um desenvolvimento de um jogo pervasivo baseado em localização utilizando a plataforma desenvolvida, disponíveis no GitHub o cliente[1] e o servidor[2].

### 5.5 Avaliar os resultados

Avaliar a plataforma com base no jogo desenvolvido. O objetivo é verificar, se todos os elementos propostos pela plataforma podem ser desenvolvidos no caso de teste realizado e se a mesma atende aos requisitos propostos no trabalho.

## 6 A PLATAFORMA DESENVOLVIDA

Nessa seção, serão apresentados os resultados do desenvolvimento. Ela está sendo subdividida em dois temas principais: a definição da arquitetura com as trocas de mensagens entre os módulos, além da API da plataforma.

### 6.1 Funcionalidades da plataforma.

Para dar suporte à criação de jogos baseados em localização a plataforma tem que dispor de algumas funcionalidades, primeiro é importante que a mesma seja capaz de transmitir os dados para diversos tipos de clientes, uma vez que um jogador pode estar usando o android ou outro sistema móvel, o servidor da plataforma deve ter meios de se comunicar com os dois.

A plataforma tem de ser capaz de persistir os principais dados do usuário, bem como disponibilizar ferramentas para conseguir sua posição geográfica. Para tornar a plataforma funcional ela deve dar suporte à criação dos jogos baseados em localização. Além disso deve dar a possibilidade de pesquisar os dados como a posição geográfica e as quests próximas.

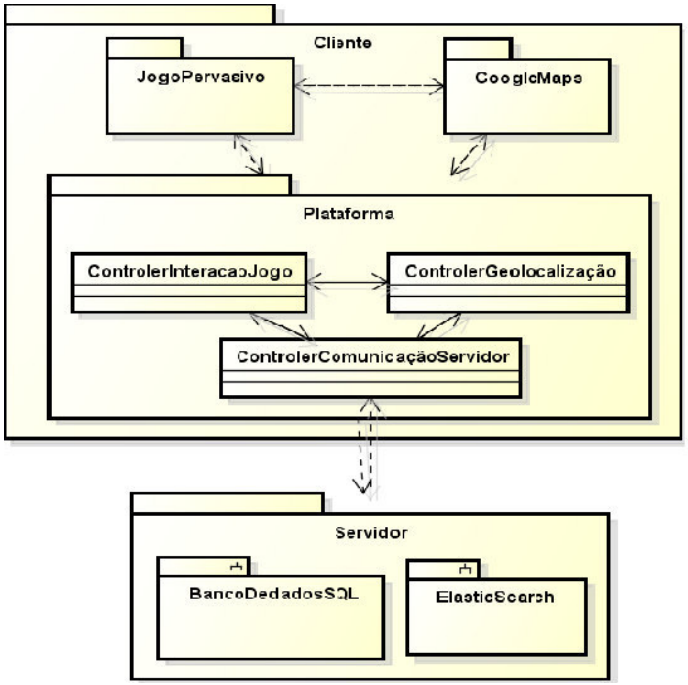
1 https://github.com/IuryMaiiaa/ClientApp
2 https://github.com/IuryMaiiaa/Servidor

## 6.2 Definição da arquitetura.

Um importante passo para o desenvolvimento da plataforma é a definição da arquitetura. Segue abaixo uma imagem ilustrando como a arquitetura do sistema funciona (figura 6) junto com a explicação de como os componentes irão se comunicar.

A figura demonstra como estão atualmente dispostas as componentes da plataforma, mostrando quais os pacotes e com quem eles interagem. A princípio, a arquitetura foi definida em um cliente-servidor, pois o sistema será online, por isso, tem de existir uma troca de informação entre os clientes online. A comunicação entre o cliente e o servidor ocorre através de requisições http. O servidor está implementado seguindo a arquitetura REST. Além disso, ele será responsável por gerenciar o banco de dados relacional e o Elasticsearch e garantir a consistência dos dados.

Figura 6: Definição da Arquitetura.




Fonte: Imagem feita pelo autor.

A parte do cliente está responsável por gerenciar e apresentar os dados ao jogo pervasivo. Assim, foram definidos três tipos de controladores para a plataforma. O controlador de comunicação com o servidor que é responsável por gerenciar as chamadas http para o servidor, além de receber os dados fornecidos pelo mesmo e repassar esses dados ao controlador de interação do jogo. O controlador de interação é responsável por avaliar os dados e as chamadas realizadas a plataforma sobre as classes relacionadas ao jogo, como os

objetos relacionados às *quests*, as etapas, objetivos dentre outros. Assim, por último o controlador de geolocalização está responsável por fornecer os métodos necessários para utilizar todos os aspectos relacionados à geolocalização fornecendo a posição do usuário e criando rotas para destino entre outros.

## 6.2 Classes modelo da plataforma.

A plataforma foi desenvolvida com base em quatro classes modelo para a persistência como ilustrado no diagrama da figura 7. A primeira decisão tomada foi de deixar toda a persistência dos dados do lado do servidor com essa base de dados unificada ficaria mais simples manter a consistência dos dados entre os diversos clientes.

Figura 7 : Diagrama de classes do modelo.


Fonte: Elaborada pelo autor.

A primeira classe do modelo é a coordenada geográfica. Essa classe foi desenvolvida visando englobar os aspectos de geolocalização. Nela estão armazenadas as informações de

latitude e longitude. Essa classe tem papel fundamental, pois é através dela que muitas mecânicas desenvolvidas nos jogos estarão relacionadas.

A *Quest* é a próxima classe modelo. Ela é a classe mais acoplada desse modelo porque ela representa a mecânica básica de uma caçada ao tesouro. Basicamente a *Quest* é um conjunto de passos necessários para se completar um determinado objetivo. Essa classe é composta de alguns atributos obrigatórios: o jogador que criou a *Quest*, a sua coordenada geográfica para definir onde a *Quest* deverá iniciar e, pelo menos, uma etapa para executar a *Quest*, tirando os atributos obrigatórios ainda existem mais dois atributos: um nome dado a *Quest* e uma descrição de como é a *Quest*.

A etapa é uma classe que possui uma relação de composição com a *Quest*, não pode existir uma etapa sem que este esteja ligado a uma *Quest* em específico. Logo como atributos obrigatórios para a criação de uma etapa foram definidos: a *Quest* a quem ela está ligada, a descrição do que é para ser realizado nessa etapa e, por fim, uma palavra-chave. Essa palavra-chave será responsável por validar se o passo foi cumprido com sucesso ou não. Ela é definida pelo usuário no momento em que ele cria a etapa. A classe tem um último atributo opcional que é a dica, um simples texto para ajudar a descrição da etapa para facilitar a descoberta da palavra-chave de forma mais rápida.

A última classe de persistência é o usuário. Essa classe é responsável por guardar as informações pertinentes aos usuários como seus nomes, e-mails, senha, cep, endereço, as *quests* criadas por usuário.

### 6.3 Banco de Dados

Para gerenciar a persistência dos dados foram escolhidos dois bancos de dados para dar suporte e agilidade a esta tarefa. O primeiro é o PostgreSQL que é um banco de dados SQL e responsável por manter a maioria dos dados. Nele é que estão sendo mantidas todas as quatro classes de persistência e suas relações. Contudo tendo em vista que a posição geográfica do meu cliente será constantemente modificada, serão realizadas várias requisições ao servidor para atualizar e localizar as *Quests* próximas do jogador. Apesar de existir uma extenção para tratar de dados geográficos, por motivos de familiaridade e experiência prévia o tratamento de geolocalização foi realizado utilizando o ElasticSearch.

O ElasticSearch é um banco de dados noSQL mais especificamente um motor de busca para grande quantidades de dados. Ele oferece nativamente suporte para criação e

gerenciamento de posições geográficas. Utilizando uma réplica do banco existe no PostgreSQL a plataforma gerencia e atualiza todas as informações relacionadas às coordenadas geográficas, dessa forma a plataforma é capaz de fazer pesquisas otimizadas e conseguir rapidamente todas as *Quest* próximas de uma dada posição geográfica.

Um grande ponto forte do Elasticsearch é sua forma de organizar os dados, o Elasticsearch trabalha com *index* onde é possível definir onde criar um *cluster,* nesse *cluster* é onde se encontra os dados para realizar a busca. Assim é possível fazer com que os *index* possam ser utilizados para limitar suas buscas. Um bom exemplo dessa possibilidade é utilizando um *index* Brasil posso adicionar nesse cluster apenas coordenadas relacionadas ao Brasil, logo quando uma pesquisa for realiza a margem de busca se torna muito menor através dessa simples modificação. Esse é só um exemplo de como pode ser simples otimizar trabalho do Elasticsearch com as coordenadas geográficas.

Apesar de o Elasticsearch ser muito eficiente em indexar dados e realizar buscas em dados não estruturados existem alguns problemas no mapeamento que prejudicariam muito a plataforma se ele fosse o único banco de dados. Mapear uma relação como Usuário possui várias *quests* e uma *quest* possui um Usuário é um exemplo de um problema que ele não consegue solucionar facilmente e para mapear essas relações que foi utilizado o PostgreSQL e seu banco de dados relacional.

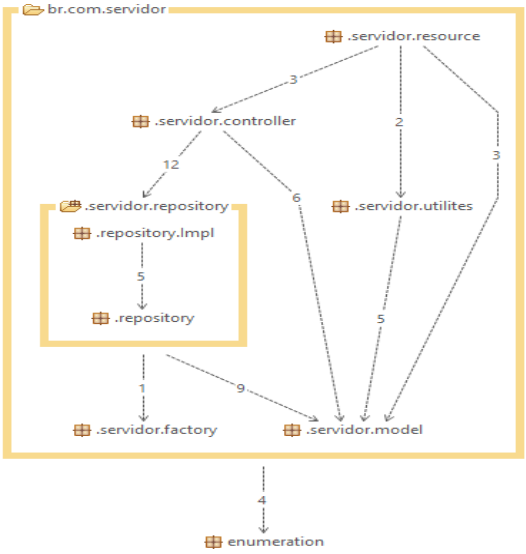
Para configurar os bancos de dados são necessários dois passos. Primeiro configurar o perssistence.XML com os dados do PostgreSQL e para configurar o Elasticsearch é necessário adicionar ao VM argument do tomcat a seguinte variável, -Des.path.home=(path para onde esta instalado o servidor do Elasticsearch). Com essas duas configurações os dois servidores devem funcionar normalmente.

## 6.4 Implementação do Servidor Rest.

O servidor foi desenvolvido usando a linguagem java e sua arquitetura é baseada em um padrão Rest. Ele foi dividido e organizado em cinco pacotes principais demonstrados na figura 7 com suas respectivas dependências. O pacote model é contém as classes do modelo como a coordenada geografia, a *quest,* a etapa e o usuário. O pacote repository é responsável por gerenciar a persistência das classes do modelo e é esse pacote que possui classes que se conectaram aos bancos de dados para realizar a persistência dos objetos. Ele também é responsável por realizar as pesquisas nos bancos de dados.

O pacote controller é responsável por gerenciar chamadas para o pacote repository. Esse pacote apenas conhece e endereça chamadas para realizar a persistência e retorna as buscas realizadas no repository. Um dos seus principais objetivos é manter o encapsulamento de como funciona o banco de dados e não ter de transmitir essa complexidade para outros pacotes do sistema.

Figura 8: Relacionamento dos Pacotes.




Fonte: Imagem feita pelo autor

O pacote resource é o pacote responsável por receber e tratar as chamadas http realizadas ao servidor. Ele está implementado com a biblioteca javax.RS que é a biblioteca java destinada a desenvolvimento de um servidor rest. Esse pacote possui três classes cada uma responsável por receber chamadas das respectivas classes de modelo as classes são CoordenadaResource, QuestResource e UsuarioResource. Toda a comunicação da plataforma para seus clientes são gerenciadas por essas três classes e feita através de um json contendo os dados dos objetos enviados.

O último pacote é o de utilities. Estão nesse pacote classes com propósitos diferentes dos mencionados anteriormente. Atualmente ele possui classes responsáveis por remover referências circulares no momento de converter um objeto para um json bem como adicionar as referências quando o json é recebido para criar um objeto.

Para realizar chamadas ao servidor são utilizadas URLs, com elas é possível acessar o servidor e seus métodos. Referente às coordenadas geográficas que são gerenciadas pela classe coordenadaResource.

Tabela 1 – cordenadaResource API.

| URL | Tipo | Parâmetro | Retorno |
|---|---|---|---|
| /Servidor/Coordenada/listarTodas | GET | Nenhum. | Lista de coordenadas Geográficas. |
| /Servidor/Coordenada/addCoordenada | POST | Coordenada Geográfica | Nenhum. |
| / Servidor/Coordenada/updateCoordenada | POST | Coordenada Geográfica | Nenhum. |
| / Servidor/Coordenada/removeCoordenada | POST | Coordenada Geográfica | Nenhum. |
| /Servidor/Coordenada/listarProximas | GET | Coordenada Geográfica, inteiro com o valor do raio de busca | Lista de coordenadas Geográficas. |

Fonte: elaborada pelo autor.

URLs referentes classes das quests são gerenciadas pela classe QuestResource.

Tabela 2 – questResource API.

| URL | Tipo | Parâmetro | Retorno |
|---|---|---|---|
| /Servidor/Quest/addQuest | POST | Quest Geolocalizada | Nenhum |
| /Servidor/Quest/updateQuest | POST | Quest Geolocalizada | Nenhum |
| /Servidor/Quest/removeQuest | POST | Quest Geolocalizada | Nenhum |
| /Servidor/Quest/listarTodos | GET | Nenhum | Lista de Quests Geolocalizadas |
| /Servidor/Quest/listarProximas | GET | Coordenada Geográfica, inteiro com o valor do raio de busca | Lista de QuestsGeolocalizadas |

Fonte: elaborada pelo autor.

URLs referentes às classes dos usuário são gerenciadas pela classe UsuarioResource**.**

Tabela 3 – usuarioResource API.

| URL | Tipo | Parâmetro | Retorno |
|---|---|---|---|
| /Servidor/Usuario/addUsuario | POST | Usuario | Nenhum |
| /Servidor/Usuario/updateUsuario | POST | Usuario | Nenhum |
| /Servidor/Usuario/removeUsuario | POST | Usuario | Nenhum |
| /Servidor/Usuario/getUsuario/{email}/{senha} | GET | String email, string senha | Usuario |

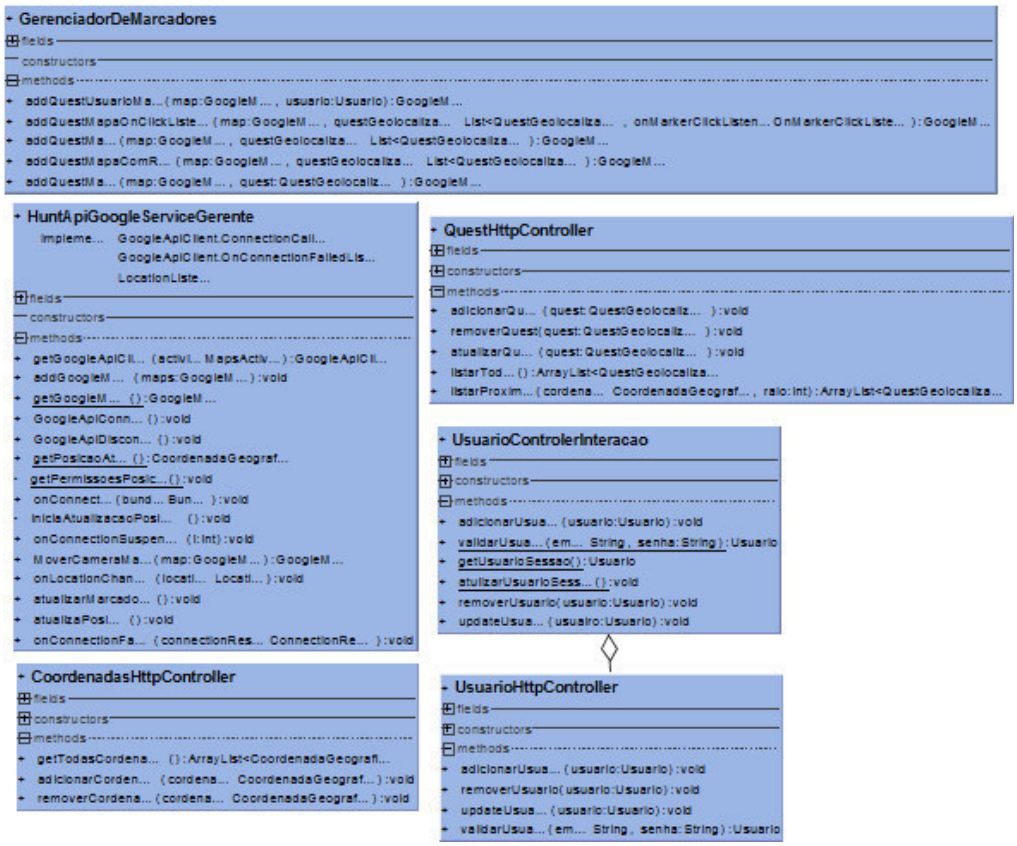
Fonte: elaborada pelo autor.

## 6.5 Cliente móvel para o sistema Android.

No cliente Android, a plataforma foi derivada em quatro pacotes principais. O primeiro é o model. Esse pacote é basicamente idêntico ao existente no servidor demonstrado no diagrama de classes da figura 9, ele possui réplicas das classes modelo para manuseio do cliente Android. O próximo pacote principal é o de controle das interações do jogo. Ele está responsável por mediar a comunicação entre o jogo e as principais funções da plataforma. É nesse pacote que estará o último usuário utilizado como seus dados. Também é através desse pacote que poderão ser feitas as chamadas para manusear o usuário logado.

Um pacote fundamental é o controle de geolocalização. Esse pacote é responsável por abstrair todos os aspectos de geolocalização relacionados ao google maps. Nesse pacote, é possível atualizar a movimentação do jogador em tempo real no mapa, receber a coordenada geográfica atual do jogador, adicionar ao mapa novos marcadores de *quest* em tempo real. Uma grande vantagem no desenvolvimento desse pacote foi tornar mais simples a utilização do google maps com os métodos das classes desse pacote com poucas configurações já é possível possuir o mapa do cliente atualizado e sempre atualizando as posições do jogador. É possível acessar as funcionalidades desse pacote através do HuntApiGoogleService, ele abstrair boa parte do gerenciamento das coordenadas geográficas como demonstrado no diagrama de classes da figura 9.

Figura 9 : Diagrama de Classes do Cliente Android.




Fonte: Elaborada pelo Autor.

O responsável por realizar todas as requisições ao servidor é o controle de comunicação com o servidor. Esse pacote é composto de mais quatro subpacotes: o controle de geolocalização responsável por se comunicar com o servidor sobre todos os aspectos referentes à persistência e buscas de coordenadas geográficas, o controle de *quests* também responsável pela comunicação com o servidor referente a todas as chamadas de persistência das *quests*, similar a estes dois, exite o controle de usuários. Por último vem o pacote de utilidades para dar suporte aos três controladores de comunicação. Esse pacote possui classes para adicionar e remover as referências circulares do modelo e uma classe para gerenciar as URLs do servidor. Estão detalhadas no apêndice A como cada uma das classes funciona.

As classes do controle de comunicação com o servidor tem outra grande funcionalidade. É da natureza do Android não poder para a *main threads* de execução, ou seja, no Android caso sua aplicação espere por uma chama assíncrona, como uma requisição http, caso essa chamada seja feia pela *main threads* a aplicação ira fechar automaticamente. Este é um importante ponto, pois todas as chamdas realizadas para a plataforma do ponto de vista do desenvolvedor são similares às chamadas síncronas. Uma chamada síncrona a comunicação

deve ocorer de forma que o retente recebe Isso deixa a cargo da plataforma criar *threads* para realizar as requisições e retornar os dados simplificando ainda mais o desenvolvimento, um bom exemplo é demonstrado na figura 10 no processo de listar todas as *quests* próximas.

Figura 10 : Exemplo de listar coordenadas próximas.

```
public void atualizarMarcadores() {
    QuestHttpController questHttpController = new QuestHttpController();
    GerenciadorDeMarcadores gerenciadorDeMarcadores = new GerenciadorDeMarcadores();
    CoordenadaGeografica cordena = HuntApiGoogleServiceGerente.getPosicaoAtual();
    List<QuestGeolocalizada> quests = questHttpController.listarProximas(cordena,10000);
    gerenciadorDeMarcadores.addQuestMapa( maps,quests);
}
```

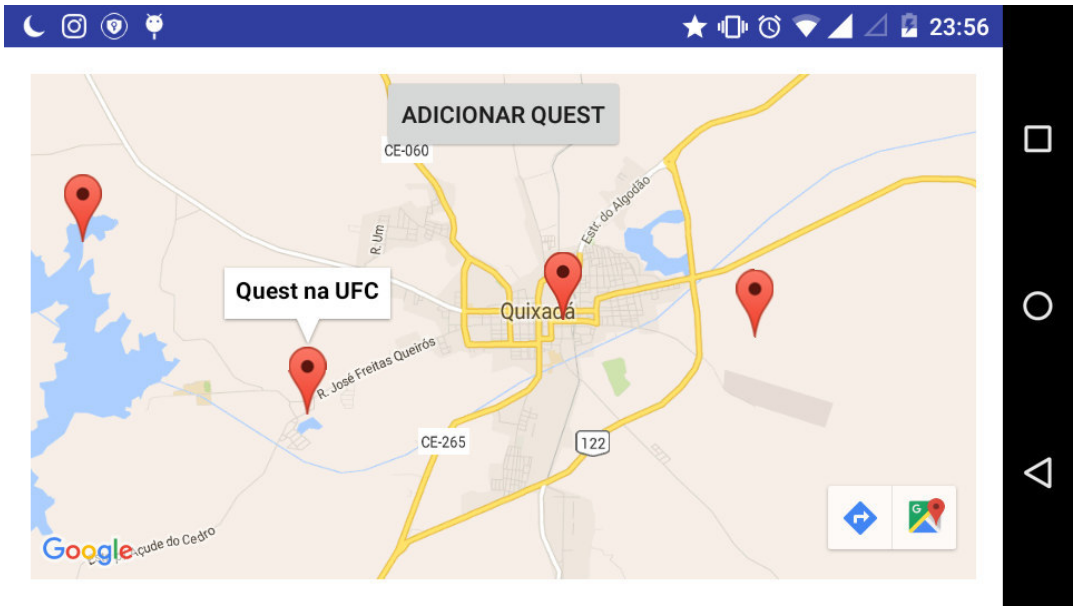
Fonte: elaborada pelo autor.

Como exemplificado na figura 10, basta requisitar a sua posição atual para a plataforma e com ela enviar a requisição através do questHttpController que por sua vez ira requisitar ao servidor, contudo como no trecho de código não é necessário criar nenhuma thread manualmente a plataforma é responsável por criar e gerenciar as threads, retornando o valor desejado como se fosse uma requisição a um método síncrono.

## 6.6 Jogo pervasivo baseado em localização desenvolvido.

A aplicação desenvolvida com a plataforma é um jogo simples com geolocalizado, primeiramente a aplicação possui algumas funções nela é necessário realizar o login para ter acesso ao jogo, depois na tela principal é possível visualizar as quest no mapa através dos marcadores. Também é possível criar quests e adicioná-las ao mapa para que outros jogadores possam joga-las. A Figura 11 exibe a tela principal do jogo com as quests.

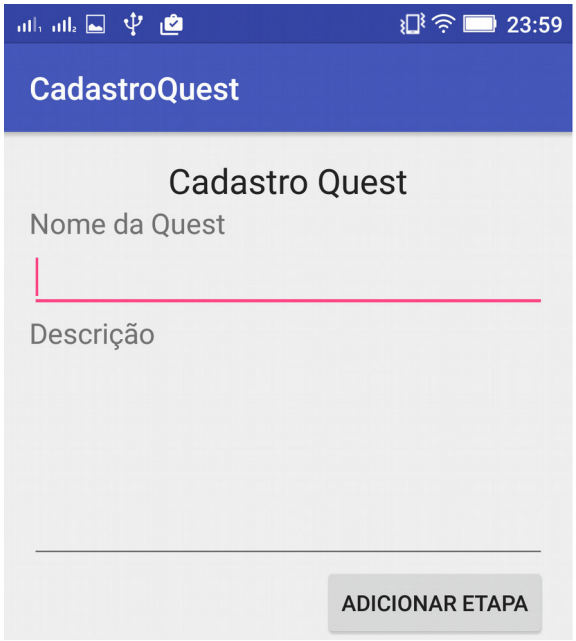
Figura 11: tela principal do jogo desenvolvido com a plataforma.




Fonte: Imagem feita pelo autor.

A primeira tela da aplicação desenvolvida é a tela de login. Nessa tela, serão apresentados dois campos a serem preenchidos o primeiro o e-mail cadastrado e por segundo a senha da conta. Para capturar esses dados foi implementado uma função no botão de login, logo quando pressionando os dados serão capturados e através do método validarUsuario da UsuarioControleInteração é feita validação do usuário pelo servidor. Após a validação do usuário. Caso o mesmo esteja cadastrado no servidor, a aplicação será redirecionada a tela principal demonstrada na figura 11.

Figura 12: Tela do cadastro de quests.

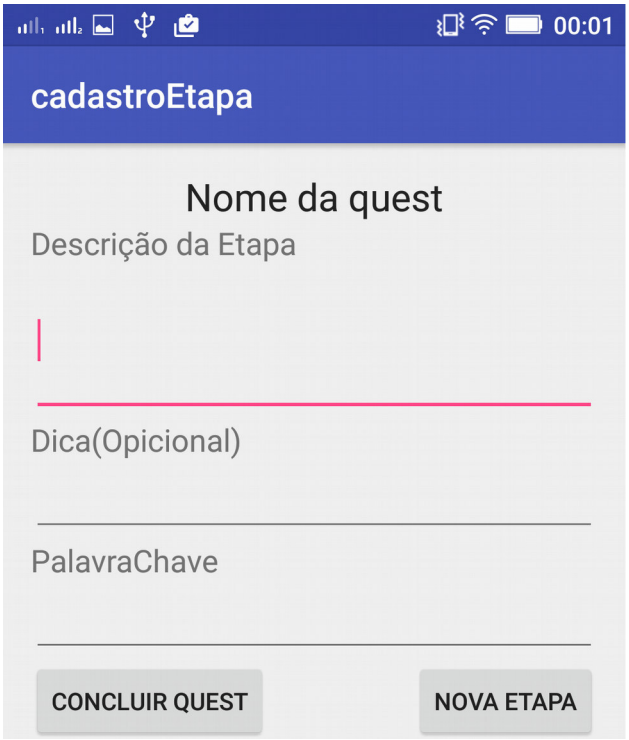


Fonte: Imagem feita pelo autor.

Para realizar o cadastro de um novo usuário existe um botão na tela de login chamado registrar, este botão redireciona para a tela de cadastro da aplicação. A tela de cadastro possui diversos campos a serem preenchidos, mas os principais para cadastrar a conta são o e-mail e a senha como demonstrado na figura. Uma vez preenchidos os campos principais basta pressionar o botão Registrar. Assim, a aplicação vai checar os campos necessários e enviar o cadastro ao servidor através da UsuarioControleInteração com o método adicionarUsuario.

Na tela principal do jogo é mostrado o mapa. Nele, serão apresentadas através de marcadores as *quests* próximas de acordo com o desejo do desenvolvedor do jogo, utilizando os métodos do gerenciador de marcadores. Na tela principal existe também o botão de Adicionar *Quest*. Através desse botão o jogo redirecionará o usuário para a tela de cadastro da quest demonstrada na figura 12.

Figura 13: Tela de Cadastro da Etapa.




Fonte: Feita pelo autor.

O cadastro de uma nova *quest* é feito em duas etapas. Primeiramente é definido o nome da *quest* que aparecera sobre o marcador quando adicionada ao mapa, depois a descrição do que se trata a quest. Após preencher os campos é necessário adicionar no mínimo uma etapa para completar o cadastro de quest. Assim basta pressionar o botão Adicionar Etapa

para ser redirecionado a tela de cadastro das etapas. Para cadastrar uma etapa é necessário dar uma descrição sobre o que é a etapa, depois uma dica sobre qual é a palavra-chave para superar a etapa e por fim a palavra-chave como demonstrado na figura 13. Caso seja necessário dividir a *quest* em múltiplas etapas basta pressionar o botão nova etapa a mesma quest. Caso contrário basta finalizar pressionando concluir *quest*, sendo assim o usuário redirecionado à tela principal.

### 6.7 Limitações da plataforma.

A plataforma foi pensada e desenvolvida visando em aplicações móveis, com foco em jogos que necessitam de geolocalização como descrito acima. Contudo, o uso de geolocalização pode se tornar um grande desafio, uma vez que para geolocalização estou usando uma api externa do google que não possui garantia de oferecer uma localização sem erros, ou seja, uma vez que a própria ferramenta não pode garantir uma posição completamente precisa, sua margem de erro na identificação da posição pode levar a plataforma a oferecer uma quantidade grande de possíveis localizações dentro do jogo. Isso implica que esses jogos podem sofrer com essa escolha, uma vez que uma geolocalização não tão precisa pode facilitar ou dificultar a experiência do jogador. Esse problema, ficou fora do escopo desse trabalho e é responsabilidade do desenvolvedor do jogo escolher localizações de quests que não sejam muito próximas.

## 7 CONCLUSÃO

Para auxiliar no desenvolvimento de jogos pervasivos baseados em localização, o presente trabalho desenvolveu uma plataforma. Esta que é capaz de gerenciar e adicionar novas coordenadas geográficas bem como associá-las as *quests,* que tem um papel muito importante para a criação dos jogos baseados em localização. Sendo assim a plataforma gerencia não somente toda a troca de dados entre o cliente e o servido, mas também a comunicação com a API do google maps e a geolocalização.

Apesar dos possíveis problemas que a plataforma poderá ocorrer, ela foi desenvolvida e pensada para desenvolvedores, uma vez que a mesma tem seu código livre e aberto sempre que algum desenvolvedor sentir a necessidade de aperfeiçoar ou modificá-la sempre existira essa possibilidade. Assim espero criar um ambiente onde todos podem ajudar a ampliar e aperfeiçoar a plataforma de forma comunitária.

# 8 TRABALHOS FUTUROS

Atualmente a plataforma cobre uma pequena parte dos jogos móveis locativos. No futuro seria muito interessante expandir essa perspectiva podendo adicionar interação com objetos virtuais através da realidade aumentada e mista. Além disso, a expansão para novos clientes da plataforma visando outros sistemas operacionais como Windows Phone e o Iphone poderão ser possíveis futuros trabalhos.

# REFERÊNCIAS

VÁZQUEZ, SAL. ***Jogos em ambientes pervasivos***. *2009. 82f.* Diss. Dissertação de Mestrado,Programa de Pós-Graduação em Computação, Universidade Federal do Riogrande do Sul, Porto Alegre, 2009.

L. A. ANDRADE. Jogos pervasivos: **Educação, cultura e cidade digital**. Revista Opara, 3(1), 2013.

P. CENTIEIRO, T. ROMÃO, e A. E. DIAS. **A location-based multiplayer mobile game to encourage pro-environmental behaviours**. In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, page 31. ACM, 2011.

S. HINSKE, M. LAMPE, C. MAGERKURTH, e C. ROCKER. **Classifying pervasive games: on pervasive computing and mixed reality**. Concepts and technologies for Pervasive Games-A Reader for Pervasive Gaming Research, 1:20, 2007.

P. KEIFER, S. MATYAS, e C. SCHLIEDER. **Systematically exploring the design space of location-based games**. In Pervasive 2006 Workshop Proceedings, Poster presented at PerGames2006, volume 7, pages 183–190, 2006.

A. MONT'ALVERNE. **Jogos móveis locativos: uma proposta de classificação**. Contemporânea, 10(1), 2012.

M. MONTOLA, J. STENROS, e A. WAERN. **Pervasive games: theory and design**. Morgan Kaufmann Publishers Inc., 2009.

C. NEUSTAEDTER, A. TANG, e T. K. JUDGE. **Creating scalable location-based games: lessons from geocaching**. Personal and Ubiquitous Computing, 17(2):335–349, 2013.

C. NOLÊTO, W. VIANA, e F. TRINTA. **Uma ferramenta de autoria para o desenvolvimento de jogos pervasivos baseados em realidade aumentada**.

D. RIEHLE. **Framework design**. PhD thesis, Diss. Technische Wissenschaften ETH Zurich, Nr. 13509, 2000, 2000.

M. WEISER. **The computer for the 21st century**. Scientific american, 265(3):94–104, 1991.

HUIZINGA, Johan. **Homo Ludens: A Study of the Play-element in Cult**. Beacon Press, 1955.

SVENNERBERG, Gabriel. **Beginning Google Maps API 3**. Apress, 2010.

LECHETA, Ricardo R. **Google Android-3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Novatec Editora, 2013.

IDC. **Smartphone OS Market Share, 2015 Q2**. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 10 de Novembro de 2016.

GORMLEY, Clinton; TONG, Zachary. **Elasticsearch: The Definitive Guide**. " O'Reilly Media, Inc.", 2015.

BURKE, Bill. **RESTful Java with JaX-RS**. " O'Reilly Media, Inc.", 2009.

KUBOTA, Flávio Issao et al. **Desenvolvimento de Plataforma de Produto e Modularidade: uma análise bibliométrica**. TPA-Teoria e Prática em Administração, v. 3, n. 2, p. 44-69, 2013.

## APÊNDICE A – API utilizadas pelo cliente Android.

Referente ao cliente Android as principais classes que podem ser usadas para interagir com o desenvolvedor serão detalhadas a seguir. O primeiro pacote a detalhar é o de comunicação com o servidor, nele existem duas classes principais que serão usadas, a questHttpController e a cordenadaHttpController. Para usar todos os métodos dessas classes basta apenas instanciar um objeto dessa classe.

Tabela 4 – Métodos da cordenadaHttpController

| Método | Parâmetro | Retorno | Descrição |
|---|---|---|---|
| getTodasCordenadas | Nenhum | Lista de Cordenadas Geograficas. | Retorna uma lista com todas as coordenadas do servidor |
| adicionarCordenada | Cordenada Geografica | Nenhum | Adicionar uma cordenada geografica. |
| removerCordenada | Cordenada Geografica | Nenhum | Remove uma cordenada geografica. |

Fonte: elaborado pelo autor.

Tabela 5 – Métodos da questHttpController.

| Método | Parâmetro | Retorno | Descrição |
|---|---|---|---|
| adicionarQuest | Quest Geolocalizada | Nenhum | Adicionar uma quest geolocalizada |
| removerQuest | Quest Geolocalizada | Nenhum | Remove uma quest geolocalizada |
| atualizarQuest | Quest Geolocalizada | Nenhum | Atualiza uma quest geolocalizada |
| listarTodas | Nenhum | Lista de QuestGeolocalizadas | Lista todas as quest geolocalizadas |
| listarProximas | Cordenada Geografica, inteiro com o valor do raio de busca | Lista de QuestGeolocalizadas | Lista apenas as quest existentes no raio de busca. |

Fonte: elaborado pelo autor.

O próximo pacote é Controle de geolocalização, este pacote sintetiza todos os métodos relacionados ao google maps. Para utilizar os métodos desse pacote é necessário possuir uma activity com o google maps e chamar o HuntApiGoogleServiceGerente em alguns locais

específicos. Primeiro é necessário instanciar um objeto da classe HuntApiGoogleServiceGerente no construtor da activity como desmontrado na figura 9.

Figura 14: instanciando HuntApiGoogleServiceGerente no construtor da activity.

```
public MapsActivity() {
    super();
    huntApiGoogleServiceGerente = new HuntApiGoogleServiceGerente();
}
```

Fonte: elaborado pelo autor.

Apos instanciar o objeto é necessário chamar outro método em um segundo ponto importante do código. No método *onCreate* de sua *activity* é necessário chamar novamente a HuntApiGoogleServiceGerente dessa vez para configurar o *googleApiServiceClient*, este cliente é importante pois é através dele que será feitas as chamadas a plataforma do *google maps,* a figura 10 demonstra como deve ser chamado o método.

Figura 15: Configurando *googleApiCliente* com a HuntApi.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    //setContentView(R.layout.fragment_maps);


    huntApiGoogleServiceGerente.getGoogleApiClient(this);

    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
```

Fonte: elaborado pelo autor.

Apos configurar o googleApiCliente é necessário fazer duas chamadas, uma para conectar a API e outra para desconectar, essas chamadas são feitas nos métodos *onStart* e *onStop* da activity. Estas chamadas estão descritas na figura 11.

Figura 16: Conectar e desconectar a API.

```
protected  void onStart() {
    huntApiGoogleServiceGerente.GoogleApiConnect();
    super.onStart();
}

protected void onStop() {
    huntApiGoogleServiceGerente.GoogleApiDisconnet();
    super.onStop();
}
```

Fonte: elaborado pelo autor.

A última parte da configuração tem de ser chamada no método *onMapReady* da activity que possui o google maps. Nesse método é necessário chamar a HuntApiGoogleServiceGerente enviando o mapa para quem ela possa sempre manter a atualização das posições, na figura 12 é demonstrado o método a ser chamado.

Figura 17: Adicionando o mapa a plataforma.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    huntApiGoogleServiceGerente.addGoogleMaps(mMap);
}
```

Fonte: elaborado pelo autor.

Apos essas configurações realizadas é possível utilizar o HuntApiGoogleServiceGerente e o gerenciador de marcadores normalmente, abaixo será mostrado os métodos que eles disponibilizam para o desenvolvedor nas tabelas 6 e 7.

Tabela 6 – Métodos do HuntApiGoogleServiceGerente.

| Método | Parâmetro | Retorno | Descrição |
|---|---|---|---|
| getGoogleMaps | Nenhum | GoogleMap | Retorna o mapa que está sendo utilizado na aplicação. |
| getPosicaoAtual | Nenhum | Cordenada Geografica | Retorna a posição atual do jogador. |
| moverCameraMapa | GoogleMap | GoogleMap | Movimenta a câmera do google maps e coloca ela sobre a atual posição do jogador. |
| atualizarPosicao | Nenhum | Nenhum | Atualiza a posição atual do jogador. |

Fonte: elaborado pelo autor.

Vale ressaltar que o HuntApiGoogleServiceGerente possui uma sub-rotina que atualiza automaticamente a posição atual do jogador tá vez que acontecer qualquer mudança em sua localização. Agora a tabela 7 detalhará os métodos que podem ser utilizador pelo gerenciador de marcadores.

Tabela 7 – Métodos do gerenciador de marcadores.

| Método | Parâmetro | Retorno | Descrição |
|---|---|---|---|
| addQuestUsuarioMapa | Google map, Usuario | Google map | Adiciona todas as quest do usuário ao mapa em forma de marcadores convencionais. |
| addQuestMapa | GoogleMap, Lista de quests | Google map | Adiciona uma lista de quest ao mapa em forma de marcadores. |
| addQuestMapaComRota | GoogleMap, Lista de quests | Google map | Adiciona uma lista de quest ao mapa em forma de marcadores. Mas ao se clicar em um marcador é criado no mapa uma rota ao destino. |
| addQuestMapa | GoogleMap, QuestGeolocalizada | Google map | Adiciona ao mapa apenas uma quest. |
| addQuestMapaOnClickListener | GoogleMap, lista de quests, GoogleMap.OnMakerClickListener | Google map | Adicionar ao mapa uma lista de quests, porém com o Click listener customizado pelo desenvolvedor. |

Fonte: elaborado pelo autor.

A última classe que é necessário interagir no cliente Android é o referente ao controle de informações do usuário. Essa classe é responsável por guardar o último usuário validado no pela plataforma e gerenciar os métodos básicos de manutenção dessa classe, para usar essa classe basta instanciar um objeto da mesma. A tabela 8 demonstra os métodos dessa classe.

Tabela 8 – Métodos do UsuarioControleInteracao.

| Método | Parâmetro | Retorno | Descrição |
|---|---|---|---|
| adicioanarUsuario | Usuario | Nenhum | Adicionar um Usuario ao servidor. |
| validarUsuario | String do email, String da senha | Usuario | Verifica no servidor se existe um Usuario com esse email e senha, caso verdadeiro ele retorna o Usuairo, caso contrário retorna null. |

| | | | |
|---|---|---|---|
| getUsuarioSessao | Nenhum | Usuairo | Retorna o último Usuario validado. |
| atualizaUsuario | Nenhum | Nenhum | Atualiza o usuario da sessão. |
| removeUsuario | Usuairo | Nenhum | Remove um Usuario do servidor |
| updateUsuairo | Usuairo | Nenhum | Atualiza um Usuario do servidor |

Fonte: elaborado pelo Autor.