Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 25, Num. 2 (2018) 19-37

RESEARCH ARTICLE

IndoLoR – An Adaptable Platform for Indoor Location

IndoLoR – Uma Plataforma Adaptável para Localização em Ambientes Internos

Mário Andrade Vieira de Melo Neto[1]*, Gibeon Soares de Aquino Júnior[2]

Resumo: Location systems have become increasingly part of people's lives. For outdoor environments, GPS appears as standard technology, widely disseminated and used. However, people usually spend most of their daily time in indoor environments, such as: hospitals, universities, factories, buildings, etc. In these environments, GPS does not work properly causing an inaccurate positioning. Currently, to perform the location of people or objects in indoor environments no single technology could reproduce for indoors the same result achieved by GPS for outdoors environments. Due to this, it is necessary to consider use of information from multiple sources using different technologies. Thus, this work aims to build an Adaptable Platform for Indoor location. Based on this goal, the IndoLoR platform is proposed. This platform aims to allow information reception from different sources, data processing, data fusion, data storage and data retrieval for the indoor location context. The proposed platform is evaluated through a case study examining aspects related to easiness of adaptation and performance.

Keywords: Indoor — Environment — Location — Adaptability — IndoLoR

Resumo: Os sistemas de localização têm se tornado cada vez mais parte integrante da vida das pessoas. Em ambientes externos, o GPS se apresenta como tecnologia padrão, largamente difundida e utilizada. No entanto, as pessoas costumam passar a maior parte do seu tempo dentro de ambientes internos, como: hospitais, universidades, fábricas, edifícios, entre outros. Nesses ambientes, o GPS tem seu funcionamento comprometido não obtendo um posicionamento preciso. Atualmente, para realizar a localização de pessoas ou objetos em ambientes internos não existe nenhuma tecnologia que consiga atingir os mesmos resultados obtidos pelo GPS em ambientes externos. Devido a isso, é necessário considerar a utilização de informações provenientes de diversas fontes fazendo uso de diferentes tecnologias. Dessa forma, esse trabalho tem como objetivo geral construir uma plataforma adaptável para localização em ambientes internos. Baseado nesse objetivo, é proposta a plataforma IndoLoR. Essa plataforma tem como objetivos permitir o recebimento de informações provenientes de diferentes fontes, além de realizar o processamento, fusão, armazenamento e disponibilização dessas informações para o contexto da localização em ambientes internos. A plataforma proposta é avaliada através de um estudo de caso verificando aspectos relativos a facilidade de adaptação e performance.

Palavras-Chave: Localização — Ambientes Internos — Adaptabilidade — IndoLoR

[1] *Instituto Federal do Rio Grande do Norte (IFRN), Brazil*
[2] *Universidade Federal do Rio Grande do Norte (UFRN), Brazil*
***Corresponding author**: mario.melo@ifrn.edu.br
DOI: http://dx.doi.org/10.22456/2175-2745.78391 • **Received:** 30/11/2017 • **Accepted:** 16/05/2018

1. Introdução

Nos dias atuais, percebe-se que os sistemas de localização estão cada vez mais presentes na vida das pessoas. Esses sistemas podem ajudar as pessoas a resolver diversos tipos de problemas em uma grande variedade de situações [1]. Tais como: localizar a posição de pessoas ou objetos, calcular uma rota de um determinado ponto a outro, localizar os bombeiros em um prédio em chamas, entre outras [2]. O problema de localização é bastante relevante em diversas áreas da computação, em que essa localização, usuários e dispositivos, é bastante valiosa e importante para diversas aplicações, em especial as sensíveis ao contexto [3][4] [5].

As pessoas, em geral, costumam gastar de 80-90% do tempo delas dentro de ambientes internos [6]. Estes ambientes incluem shoppings, bibliotecas, aeroportos, universidades, escolas, escritórios, fábricas, hospitais, entre outros. Diante

disso, os serviços de localização em ambientes internos vêm obtendo uma atenção especial. Um dos motivos para o aumento da popularidade deste tipo de aplicação deve-se, em grande parte, a crescente popularização de dispositivos portáveis, como celulares, *smartphones*, tablets e notebooks, no qual em sua maioria já possuem diversos hardwares embutidos como WIFI, Bluetooth, GPS e diversos sensores.

Dada essa popularidade, existe uma necessidade crescente de buscar tecnologias que atendam aos requisitos de alta performance com um baixo custo, de forma a disponibilizar uma variedade de aplicações interessantes em espaços internos [7]. Além disso, de acordo com a OpusResearch, [8] até 2018 devem ser gastos cerca de 10 bilhões de dólares, de forma direta ou indireta, na área de localização *indoor*. Demonstrando assim, uma grande oportunidade de mercado. Dentre os tipos de aplicação presentes na área, as que realizam a propaganda direcionada emergem como as mais promissoras [8].

Além disso, diversos problemas podem ser resolvidos através de aplicações que utilizam a informação da localização de uma pessoa ou objeto em um determinado ambiente interno. Em que os principais tipos de aplicações que se beneficiam destas informações são: Realidade Aumentada [9], Campus Escolar [10] [11] [12], Tour em museus guiado [13], Shoppings [12], Navegação em Loja [14] [15],Estações de trem e ônibus, aeroportos e estações subterrâneas [16], Propaganda Direcionada [17].

A localização em ambientes externos, possui uma tecnologia bem difundida e padrão chamada Sistema de Posicionamento Global (GPS) [18] cujo funcionamento é baseado na utilização de satélites, desta forma oferece cobertura em todo o globo terrestre. No entanto quando se trata de ambientes internos, o mesmo não possui um funcionamento preciso pois as ondas rádio emitidas pelos satélites não conseguem penetrar pelas paredes dos ambientes, não permitindo assim a determinação do posicionamento de maneira eficaz, tipicamente em uma ordem superior a 10m [19][20]. A principal diferença entre a localização interna e externa está na possiblidade da localização externa poder possuir uma maior taxa de “erro"da acurácia de forma que a localização continue sendo precisa [2].

A localização de pessoas ou objetos em ambientes internos, vem sendo cada vez mais demandada para que possa ser confiável e precisa [21]. Nesse sentido, as grandes empresas de tecnologia atualmente como Google [22] e Apple [23] estão investindo nessa área de pesquisa para o desenvolvimento de soluções para a localização em ambientes internos [24] [25]. Isso demonstra que este problema permanece sem solução, ou seja, não existe uma tecnologia ou uma combinação de tecnologias que resolva o problema de maneira aceitável e a baixo custo [26]. E uma das principais razões para isto é a alta complexidade dos ambientes internos, em que existem uma série de impedimentos como paredes, equipamentos e até pessoas, diferente dos ambientes externos [27].

Dessa forma, é necessário que as soluções propostas para resolver o problema da localização em ambientes internos levem em consideração a complexidade desses ambientes. Em Melo e Aquino [28], é apresentado que existe uma tendência que as soluções para obter a localização em ambientes internos utilizem uma combinação de tecnologias com o objetivo de obter um posicionamento preciso e confiável. Nesse contexto, um *middleware* pode oferecer serviços comuns para aplicações e facilitar o desenvolvimento de aplicações [29]. Assim, são escondidos detalhes de implementação e configuração aumentando o nível de abstração possibilitando que os desenvolvedores foquem apenas na lógica de suas aplicações [30]. Dessa forma, com objetivo de propor uma solução que se adeque a qualquer tipo de ambiente e adapte suas funcionalidades as necessidades específicas impostas pelo meio onde estão incluídas, este artigo propõe uma plataforma de *middleware* adaptável que possibilite o uso de várias tecnologias para obter a localização de pessoas ou objetos em ambientes internos. Esta plataforma foi batizada de “IndoLoR”, acrônimo de *Indoor Location Radar*.

Este trabalho está organizado da seguinte maneira: Seção 3 será dedicada para a apresentação da plataforma proposta, detalhando seus requisitos, arquiteturas e componentes. A seção 4 apresenta um estudo de caso utilizando uma combinação de tecnologias com o intuito de avaliar a adaptabilidade e a performance da plataforma projetada. Na seção 2 apresenta os trabalhos diretamente relacionados com este estudo. Além disso, são apresentadas as principais características de cada um deles e suas diferenças em relação a plataforma proposta. Por fim, a seção 5 apresenta as considerações finais, assim como estudos futuros para este trabalho.

2. Trabalhos Relacionados

Nesta seção serão apresentados os trabalhos que estão diretamente relacionados a este, destacando as principais contribuições e apresentando diferenças com a abordagem proposta.

O modelo Arquitetural *Location Stack* introduzido por Hightower et. al. [31] apresenta um modelo padronizado para construção de sistemas de localização utilizando uma modelo arquitetural em camadas. Em que este modelo se refere a formas padronizadas de realizar a combinação de dados utilizando diferentes fontes através de uma arquitetura de software baseada no modelo de comunicação de rede *Open System Interconnect* (OSI).

A Figura 1 apresenta o modelo arquitetural *Location Stack*. É possível notar que para sua arquitetura foram definidas sete camadas, sendo elas: *Sensors* que possui os aspectos de hardware e software para o recebimento de informações; *Measurement* que tem como objetivo transformar os dados obtidos em tipos de medidas; *Fusion* cuja missão é realizar a combinação das medidas obtidas na camada anterior; *Arrangements* é a camada responsável pelo raciocínio através de probabilidades; *Contextual Fusion* tem como objetivo realizar a combinação de informações de contextuais com não contextuais; *Activities* é a camada na qual as aplicações ubíquas provêm a semântica sobre as informações previamente obtidas e por último *Intentions* que representa os desejos cognitivos que os usuários

desejam fazer com as informações ubíquas.

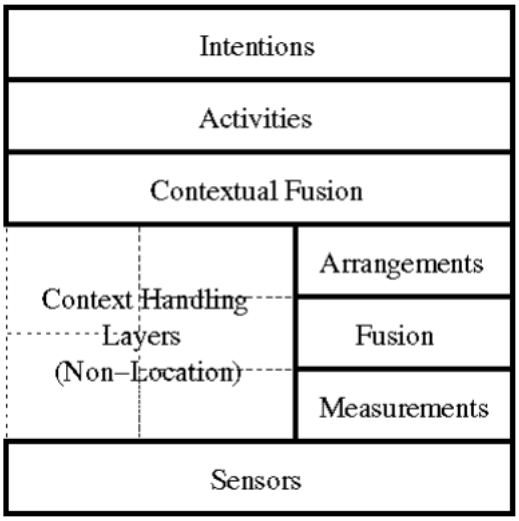


Figura 1. Modelo Arquitetural da *Location Stack* [31]

Como principais diferenças entre as arquiteturas do *Location Stack* e a plataforma proposta neste trabalho, a primeira de define camadas rígidas assim como no modelo OSI em que uma camada apenas recebe dados provenientes da camada imediatamente inferior e exporta os dados processados para a camada imediatamente superior torna a arquitetura pouco flexível, dificultando a extensão das funcionalidades. Na arquitetura da plataforma IndoLoR sua principal característica é permitir a adaptabilidade de todos os componentes e não definir todas as responsabilidades ou informações de entrada e saída que os mesmos devem obter/produzir, ao contrário da arquitetura *Location Stack*. Além disso, apesar de sua arquitetura também poder ser organizada em camadas, existe uma troca de informações entre camada em dois sentidos, ou seja, a comunicação é bidirecional entre os componentes da plataforma.

Outro fator diferencial é que a arquitetura *Location Stack* dificulta a utilização do processamento distribuído de componentes de sua arquitetura visto que há uma clara dependência de todas as camadas envolvidas no processo. A plataforma proposta permite que componentes individuais sejam executados em locais diferentes do núcleo principal, isso é obtido através da utilização do modelo arquitetural *Publish/Subscribe* em que cada componente é modelado como um serviço e existe um diretório de serviços que armazena a informação de como cada serviço será utilizado.

Em Najib et. Al. [32] é apresentado um *middleware* para localização em ambientes internos chamado de *MapUme*. O *MapUme* permite a fusão de dados provenientes de diversos tipos de sensores, em que esse processo acontece através da implementação de uma *engine* de fusão que é construída através do padrão de projeto fábrica abstrata. Além disso, possui a capacidade de processamento distribuído permitindo a escalabilidade e a alta performance. A arquitetura utilizada é baseada na proposta por Hightower et. al no qual a mesma é organizada em camadas. Além disso, utiliza o modelo arquitetural orientado a serviços.

O *MapUme* possui como requisitos principais: escalabilidade, usabilidade e extensibilidade. A escalabilidade será obtida a partir do processamento distribuído. O requisito de usabilidade refere-se à flexibilidade necessária para ser utilizada nos mais diversos tipos de aplicações, como prover localização físicas (sistema de coordenadas) ou simbólicas (andar, corredor, sala e etc.). A extensibilidade se dá através de interfaces definidas pelo middleware podendo adicionar novas funcionalidades, como novos tipos de tecnologias de sensoriamento e novos algoritmos para a fusão de informações.

Apesar de ser baseado em uma arquitetura modular e divididas em camadas, este middleware permite a extensibilidade de seus componentes. No entanto, a extensibilidade só é permitida para alguns dos componentes da arquitetura. No entanto, essa extensibilidade é obtida através da modificação /inserção de novas implementações na arquitetura do *MapUme* o que gera um alto grau de acoplamento entre os componentes estendidos. Essa característica a torna bem diferente da plataforma proposta que permite a inserção de novas características sem a necessidade realizar a interrupção da execução da plataforma, ou seja, em tempo de execução. A plataforma IndoLoR utiliza como base de sua arquitetura o padrão *Publish/Subscribe*, desta forma é possível garantir que os novos serviços ao serem instanciados e registrados estarão disponíveis para utilização dentro da plataforma sem a necessidade de parada da mesma, podendo inclusive ser escalados individualmente. Através do uso desse padrão também é possível desacoplar as adaptações do núcleo da plataforma, facilitando o desenvolvimento de adaptações visto que não é necessário conhecer detalhes de implementação da plataforma.

O LOC8 proposto por Stevenson et. al [33] é um framework desenvolvido em JAVA para localização em ambientes internos e externos suportando um modelo de contexto e espaço específico. Possui uma arquitetura baseada em camadas com uma robusta separação de preocupações. Além disso, suporta diferentes tipos de informações provenientes dos sensores, assim como, a extensão e customização dos algoritmos de fusão. Disponibiliza uma API para que os desenvolvedores possam utilizar as consultas para obter as informações de posicionamento sem que seja necessário conhecer a maneira como os dados dos sensores são obtidos. Assim como em Hightower et. al, a arquitetura proposta é baseada em camadas utilizando como base o modelo OSI. Possui como foco principal na realização de consultas aos dados presentes no modelo de localização proposto.

O framework LOC8 difere bastante da plataforma proposta pois tem como foco em criar um modelo para o espaço e as posições em ontologias, além de prover um mecanismo de consulta para suportar o uso de localização em sistemas pervasivos. A plataforma proposta possui a característica pervasiva pois possibilita o uso das informações presentes nos ambientes, porém não cria um modelo específico para espaços

Tabela 1. Lista de características presentes em cada uma trabalhos relacionados

Requisitos	Location Stack	MapUme	LOC8	MiddleWhere	IndoLoR
Utilizar uma arquitetura modular e adaptável	Não	Sim	Não	Não	Sim
Recepção e processamento de dados de fontes de informações heterogêneas	Sim	Sim	Sim	Sim	Sim
Prover mecanismos para a fusão de informações	Sim	Sim	Sim	Não	Sim
Garantir o suporte a diferentes repositórios para armazenamento de dados	Não	Não	Não	Não	Sim
Garantir o controle de acesso as informações	Não	Não	Não	Não	Sim

e não define como serão as consultas para obter a localização. Possui um modelo totalmente flexível, no qual cada tipo de aplicação poderá definir suas especificidades.

Em Ranganathan et Al. [34] é proposto um middleware sensível a localização *"MiddleWhere"*. Este middleware integra múltiplas tecnologias de localização, assim como, apresenta as aplicações clientes dados de localização consolidados. Para isso, utiliza uma arquitetura dividida em camadas para coletar as informações dos sensores, persistir as informações em um banco de dados espacial, uma engine de inteligência para realizar a fusão de dados para obter as informações de localização e uma camada de serviços para a disponibilização dessas informações. Além disso, possibilita a utilização do modo de interação pull e push para comunicação com as aplicações.

Diferente da abordagem proposta neste trabalho, esse middleware permite um pequeno grau de adaptabilidade pois permite que sejam incluídas e removidas as fontes de informações, esta adaptação pode ocorrer em tempo de execução. A arquitetura da plataforma proposta além de permitir a utilização de novos e diferentes fontes de informação também em tempo de execução, permite que seja modificado qualquer um dos componentes presentes na plataforma. A plataforma proposta busca tornar mais simplificado a utilização dos dados por ela obtidos, disponibilizando-os por meio de consultas simplificadas para que não só a plataforma possa ter suas funcionalidades adaptadas/estendidas mas também os dados que estiverem em sua posse. Dessa forma, os clientes podem solicitar os dados e atuarem como extensão da plataforma fazendo o uso das informações de localização e contexto de acordo com a sua necessidade.

Sumarizando a análise dos trabalhos, é apresentado na Tabela 1 os requisitos suportados pelos trabalhos avaliados, obtendo uma visão mais objetiva e direta das características presentes em cada um deles.

3. Plataforma Proposta

A plataforma define uma série de componentes que podem ter suas funcionalidades adaptadas com o intuito de permitir a recepção, tratamento e armazenamento dos diversos dados heterogêneos que compõem os ambientes internos. Além disso, a possibilidade da adaptação de seus componentes permite que a plataforma seja implantada nos mais diversos ambientes internos, podendo atender a especificidade de cada um deles. Para atender a essa heterogeneidade de informações, foi necessário definir um conjunto de requisitos para garantir que todas as especificações possam ser atendidas.

A definição dos requisitos se deu através de uma análise de trabalhos presentes na literatura diretamente relacionado a este trabalho, apresentados no seção 2. Durante essa análise, buscou-se identificar os requisitos não-funcionais contemplados em cada um destes trabalhos. No entanto, é importante ressaltar que os requisitos contemplados pela plataforma proposta não se limitaram aos encontrados na análise dos trabalhos, outros que foram julgados essenciais para um melhor funcionamento da plataforma foram incorporados.

Diante do exposto, os requisitos obtidos a partir dessa análise são apresentados em detalhes abaixo:

- Utilizar uma arquitetura modular e adaptável

  Com este requisito, tem-se o objetivo de garantir que a plataforma obtenha uma maior flexibilidade na definição de suas funcionalidades. Cada módulo da arquitetura é chamado de componente. A utilização dessa abordagem visa melhorar o processo de manutenção de funcionalidades da plataforma, garantindo assim que caso seja necessário realizar uma correção ou uma melhoria em um determinado componente da plataforma não haja impacto em outros componentes. Essa característica permite que seja possível definir responsabilidades específicas para cada um dos componentes aumentando a facilidade de entendimento e manutenção pelos desenvolvedores. Dessa forma, a modularização deve permitir o acoplamento ou desacoplamento de adaptações aos componentes de maneira *plug n' play*.

  A característica de adaptabilidade é a habilidade dos componentes de uma arquitetura de adaptar suas funcionalidades de maneira estática ou em tempo de execução

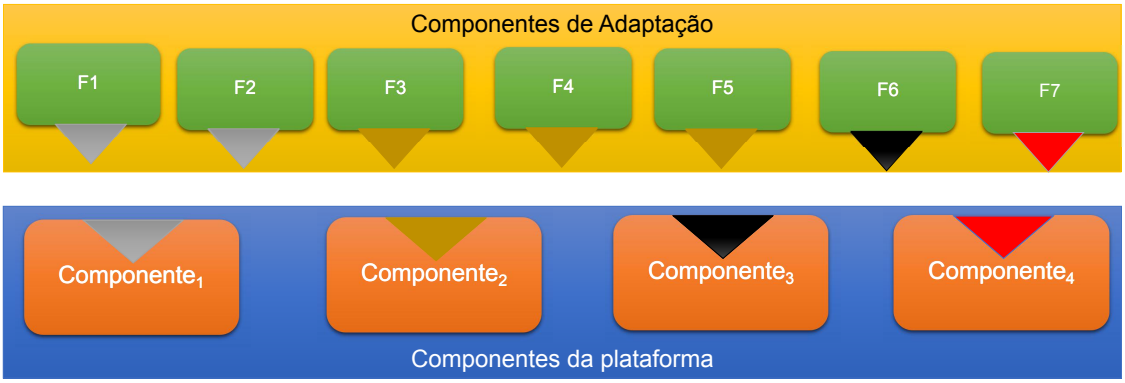


Figura 2. Esquema de adaptação dos componentes presentes na arquitetura

para atender à mudanças no ambiente ou nos requisitos definidos [35]. Cada componente da arquitetura possui uma interface de adaptação, no entanto nenhum componente adaptador a realiza pois podem existir diversos cenários de adaptação. Então, os componentes adaptadores especificam uma interface compatível com os componentes definidos na arquitetura plataforma. Dessa forma, forma-se uma ligação entre os dois componentes sempre que seja necessário, na qual é exemplificado na Figura 2. Além disso, dada essa divisão da arquitetura em componentes é possível realizar a adaptação de componentes específicos, não sendo obrigatória a adaptação de todos os componentes da plataforma para a inclusão ou alteração de capacidades.

- Recepção e processamento de dados de fontes de informações heterogêneas

  Este requisito visa garantir que a plataforma possibilite a manipulação de dados heterogêneos permitindo como entrada qualquer tipo de informação, independente da tecnologia ou unidade utilizada. Um dos objetivos deste requisito é permitir a combinação de diferentes tipos de informação para prover os vários tipos de serviços de localização.

- Prover mecanismos para a fusão de informações

  Deve prover mecanismos para permitir a fusão de informações de domínios iguais ou diferentes. Dessa forma, deve ser possível combinar dados brutos provenientes das diferentes fontes de informação ou dados derivados obtidos através da utilização de algoritmos de processamento.

- Garantir o suporte a diferentes repositórios para armazenamento de dados

  A plataforma deverá suportar diferentes formas de armazenamento de dados ou seja, ser adaptável no que tange a persistência de informações. Deve ser possível utilizar bancos de dados relacionais, não relacionais, memória ou ainda o sistema de arquivos. Além disso, cada componente poderá utilizar uma forma de armazenamento que atenda às suas necessidade sem que haja um repositório padrão definido.

- Garantir o controle de acesso as informações

  Deve garantir que apenas os componentes que possuem a permissão para realizar operações sobre um determinado tipo de dado tenham acesso ao mesmo.

- Prover diferentes formas de cálculo de posicionamento

  De acordo com Gressmann et al. [36] existem dois cenários para o cálculo de posicionamento. Em que estes são:

  – Auto posicionamento: Quando o objeto móvel é capaz de obter seu posicionamento através de informações coletadas a partir do ambiente em que está inserido. De posse desse posicionamento, essa informação deverá ser enviada para a plataforma para armazenamento.

  – Posicionamento distribuído: Quando o objeto móvel não é capaz de determinar seu próprio posicionamento por algum motivo, como baixo poder de processamento ou inexistência de hardware capaz. A plataforma deverá realizar a estimativa do posicionamento deste objeto através da utilização de informações provenientes de outras fontes ou de informações coletadas pelo próprio dispositivo.

  A plataforma proposta, suportará os dois cenários de cálculo do posicionamento.

3.1 Arquitetura Proposta

Na Figura 3 é apresentada a arquitetura geral da Plataforma IndoLoR. Pode-se perceber que a mesma foi projetada para garantir o suporte as diversas fontes de informações presentes nos mais diversos ambientes, assim como o suporte a toda essa heterogeneidade de informações. Nota-se que a arquitetura geral pode ser dividida em três grandes grupos, sendo estes: dispositivos móveis, sensores e a própria plataforma IndoLoR.

Os dispositivos móveis, além de grandes consumidores, também serão grandes provedores de informação, pois, em

sua grande maioria dispõem de *hardwares* capazes de obter informações relativas ao ambiente em que estão inseridos. As informações provenientes dos sensores de ambiente permitem que as abordagens e algoritmos utilizados para se obter a localização de coisas em ambientes internos utilizem o próprio meio em que o alvo está inserido e com isso seja possível diminuir a taxa de erro e aumentar a confiabilidade das soluções.

Por fim, o último grupo refere-se a própria plataforma cuja função é receber, processar, armazenar e disponibilizar os dados obtidos das diversas fontes de informações com o objetivo de obter a localização de pessoas ou objetos em ambientes internos. Esta plataforma dispõe de um conjunto de componentes pré-definidos, no qual todos permitem a adaptação de suas funcionalidades, sendo assim novas abordagens, algoritmos ou técnicas podem ser adicionadas sem que haja a necessidade da arquitetura base ser modificada. Além disso, percebe-se que é definido um fluxo de informações dentro da plataforma garantindo que os componentes mesmo que adaptados executem de acordo com os seus objetivos pré-definidos.

A plataforma IndoLoR é composta por 9 componentes padrões, em que cada um desses componentes pode ter suas funcionalidades adaptadas por componentes adaptadores. Cada um destes componentes padrões possui uma responsabilidade específica, sendo estas apresentadas em detalhes a seguir:

- ***IO Manager***

  É o componente de entrada para a plataforma, diferente dos outros componentes padrões, trata-se de um componente em que suas adaptações disponibilizam as interfaces de acesso com o meio externo. Quando esses componentes recebem as solicitações provenientes do meio externo, realizam seu processamento inicial e a repassam para este componente.

  Cada solicitação recebida pela plataforma segue um formato bem definido, conforme apresentado na Tabela 2.

- ***Request Manager***

  Este componente tem como função identificar o tipo de solicitação enviada à plataforma e a encaminhar para o componente padrão realizar seu processamento, diferente do componente *IO Manager* que apenas recebe a solicitação do meio externo e a repassa.

- ***Data Publication Manager***

  É o componente responsável por permitir que os dados produzidos na plataforma possam ser acessados pelas aplicações clientes. Este componente realizará a disponibilização das informações recebidas e geradas pela plataforma fazendo uso dos provedores de conteúdo providos pelo componente ***Data Storage Manager*** para disponibilizar essas informações.

- ***Map Manager***

  Caso o tipo da solicitação seja relativa ao domínio de mapas, este é o componente responsável por receber a solicitação, transformar a informação para este domínio e invocar o serviço que irá realizar o processamento da solicitação. Este componente é o responsável por prover os serviços para o cadastro de mapas, recuperá-los, cadastrar e recuperar pontos de interesse [37] associados aos mesmos, entre outros.

- ***Location Manager***

  Este é componente principal da plataforma sendo o responsável por realizar o processamento das informações de localicazação. As adaptações deste componente devem possuir três comportamentos a serem executados. O primeiro é o de realizar a transformação das informações recebidas em dados que possam ser processados pelo componente. A segunda é uma vez que a informação está um formato em que o componente consegue processá-la, o componente adaptador deverá identificar a melhor estratégia para o processamento dessas informações. Em que nesse processamento, pode-se utilizar diferentes algoritmos de localização, a combinação deles ou apenas algoritmos para fusão de informações gerando novos tipos de informações. Por último, as informações recebidas e processadas devem ser enviadas para armazenamento para que outras funcionalidades possam fazer uso delas.

- ***Event Manager***

Tabela 2. Formato dos dados das solicitações

Campo	Descrição
Requisição	Indicará qual o tipo de solicitação da requisição. Desta forma, uma vez definida o tipo da solicitação será possível definir o componente padrão da camada de processamento a que essa solicitação será repassada.
Versão	Indica qual a versão da funcionalidade que está sendo requisitada, pois as funcionalidades da plataforma podem necessitar de evolução,porém nem todas as aplicações clientes podem ou devem evoluir em sincronia com a plataforma.
Tipo de Dado	Este campo indicará ao componente de processamento,dentre os serviços registrados no seu contexto qual deverá processar a solicitação.
Operação	Indica qual operação deverá ser executada para realizar o processamento da solicitação em um componente adaptador
Dados	Conterá os dados que devem ser utilizados no processamento das operações da Plataforma.

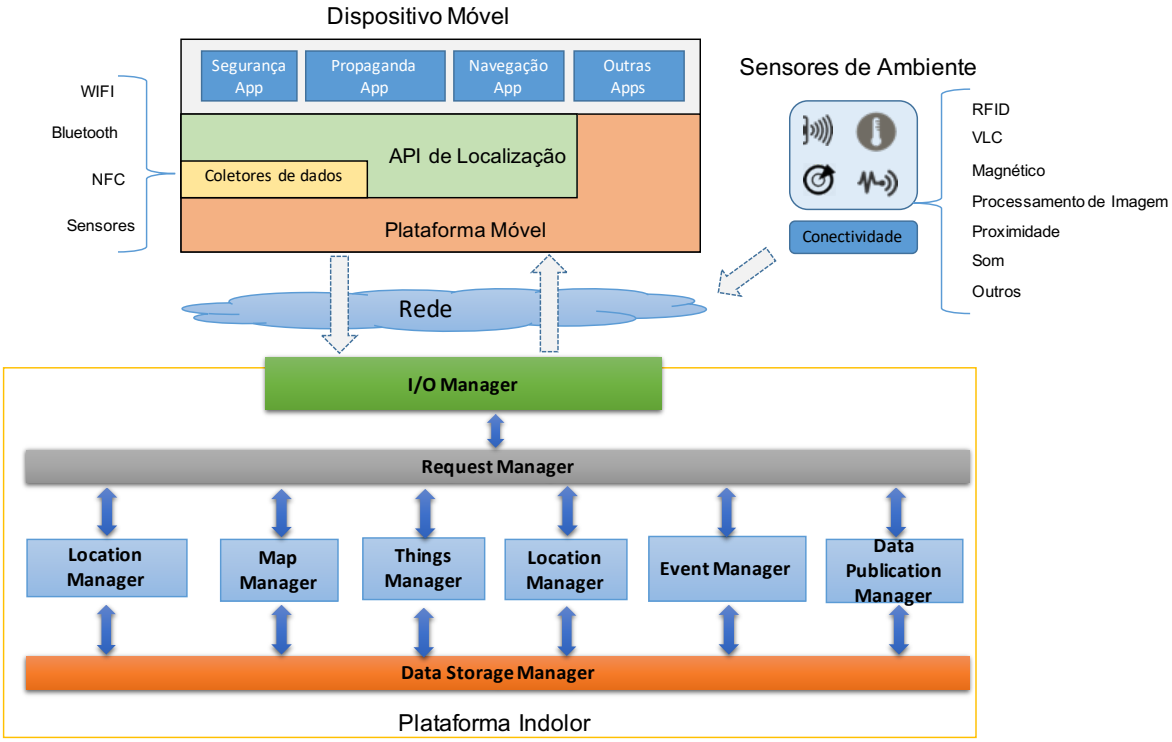


Figura 3. Visão Geral da Plataforma IndoLoR

Este componente permite que seja utilizado o modo de interação Push, para enviar informações para os dispositivos de maneira assíncrona, sendo possível realizar o registro de serviços que permitam o cadastro de eventos. Quando o evento cadastrado acontecer, é gerada uma notificação para os interessados. No intuito de otimizar a utilização de recursos como tráfego de rede e processamento, essas notificações são realizadas de maneira assíncrona.

- ***Things Manager***.

  Este componente é o responsável pela parte administrativa da plataforma sendo o responsável pelo cadastro, alteração e remoção das "coisas" necessárias para o funcionamento da plataforma. Essas coisas podem ser: Equipamentos dos ambientes, dispositivos, dados administrativos, entre outros.

- **Data Storage Manager**

  Este componente tem como função servir como um repositório de serviços de armazenamento. Os componentes padrões da plataforma informam o tipo de dado que desejam manipular e recebem o serviço que realiza todos os procedimentos de persistência. Dado o caráter adaptável da plataforma, é possível acoplar a cada serviço uma estratégia de armazenamento diferente, ou seja, pode ser utilizada a estratégia que melhor atenda o tipo de dado a ser consumido ou armazenado.

3.2 Descrição da Arquitetura

Com o objetivo de descrever a arquitetura proposta para a plataforma serão apresentadas as principais visões arquiteturais definidas por Clements et. al. [38], sendo elas: visão estrutural ou de módulos (do inglês, *Module View*) e a visão de componentes e conectores (do inglês, *Component-and-Connector View*).

3.2.1 Visão de Módulos

Esta visão apresenta a arquitetura em módulos, em que estes podem ser unidades de código que implementam alguma funcionalidade, da mais alta como pacotes até a mais baixa granularidade como classe, interfaces, entre outros. Essa visão tem como objetivo apresentar a organização das unidades de implementação e a relação de dependência que existe entre elas. Na Figura 4 é apresentada a visão de módulos da Plataforma utilizando a UML.

O estereótipo «usa» é utilizado com o intuito de indicar que um módulo A usa de um módulo B. Esse comportamento representa um relacionamento dependência em que A depende do funcionamento correto de B para satisfazer seus próprios requisitos [38]. Com essa visão percebe-se claramente que existe uma separação em camadas na arquitetura da plataforma, para representar essa visão Clements et. Al. [38] define o estilo arquitetural em camadas. A Figura 5 apresenta a arquitetura utilizando o estilo em camadas.

A plataforma é dividida em três camadas, em que a camada de entrada e saída possui a interface com o meio externo, obtendo e enviando informações. A camada de processamento

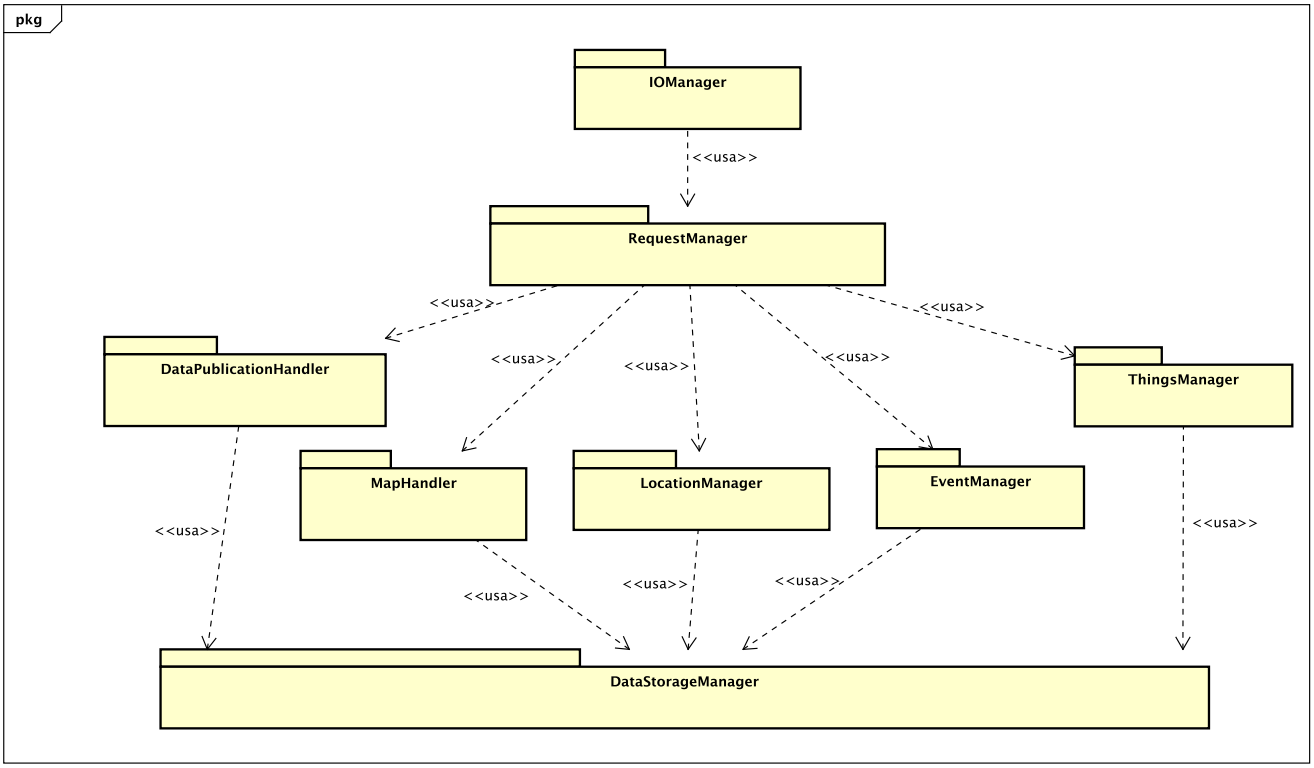


Figura 4. Visão de módulos da Arquitetura

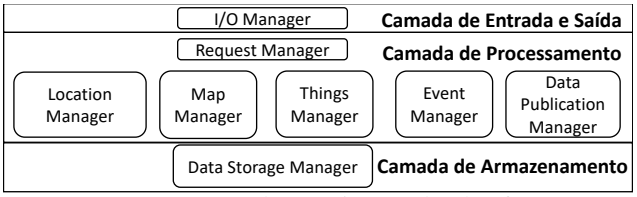


Figura 5. Apresentação da Arquitetura da Plataforma utilizando o estilo em camadas

que recebe os dados provinientes da camada anterior, analisa que componente será o responsável pelo seu processamento e o encaminha. Esta última, tem como função realizar todos os procedimentos referentes ao processamento das solicitações recebidas podendo necessitar dos componentes providos pela camada de armazenamento para recuperação ou inserção de informações.

3.2.2 Visão de Componentes e Conectores

A visão de componentes e conectores apresenta como o sistema está estruturado em seus conjuntos de elementos em relação ao comportamento e interações em tempo de execução [38]. Essa visão possibilita o entendimento do funcionamento da plataforma pois diversos componentes arquiteturais podem ser melhor entendidos quando tem seu comportamento analisado considerando a sua execução. Na Figura 6 é apresentado o diagrama UML que representa essa visão.

No momento em que a execução da plataforma é iniciada, cada componente é inicializado de maneira que existe apenas uma instância para cada componente em tempo de execução. Além disso, todos os componentes da plataforma não armazenam estado, ou seja, não armazenam informações entre uma execução e outra. Essa característica minimiza o seu tempo de utilização, além de aumentar a escalabilidade [39]. Cada um desses componentes executa na *thread* principal, porém a plataforma tem seu ambiente de execução *multithreading* pois para cada solicitação recebida a partir de um cliente externo a plataforma inicia uma thread que será a responsável por percorrer todo fluxo no interior da plataforma, disponibilizando ao final a resposta para o cliente.

3.3 Processo de Adaptação

Subramanian e Chung [40] definem adaptabilidade como a medida em que um sistema de software se adapta a mudanças em seu ambiente. Além disso, um sistema de software dito adaptável pode tolerar mudanças em seus ambientes sem a necessidade de intervenção externa. Segundo Fayad e Cline [41] existem quatro fatores que normalmente tornam um sistema adaptável: extensibilidade, flexibilidade, *tunability* e *fixability*. Os dois primeiros implicam em mudanças de mais alto nível e os dois últimos implicam em mudanças de baixo nível.

Extensibilidade é a capacidade de modificar facilmente a maior parte dos recursos de um sistema. A flexibilidade se refere a capacidade de modificar os recursos do sistema na forma como interagem. *Tunability* é característica da capacidade de melhorar aspectos relacionados a performance. E por fim, a *Fixability* é a como a capacidade de consertar uma

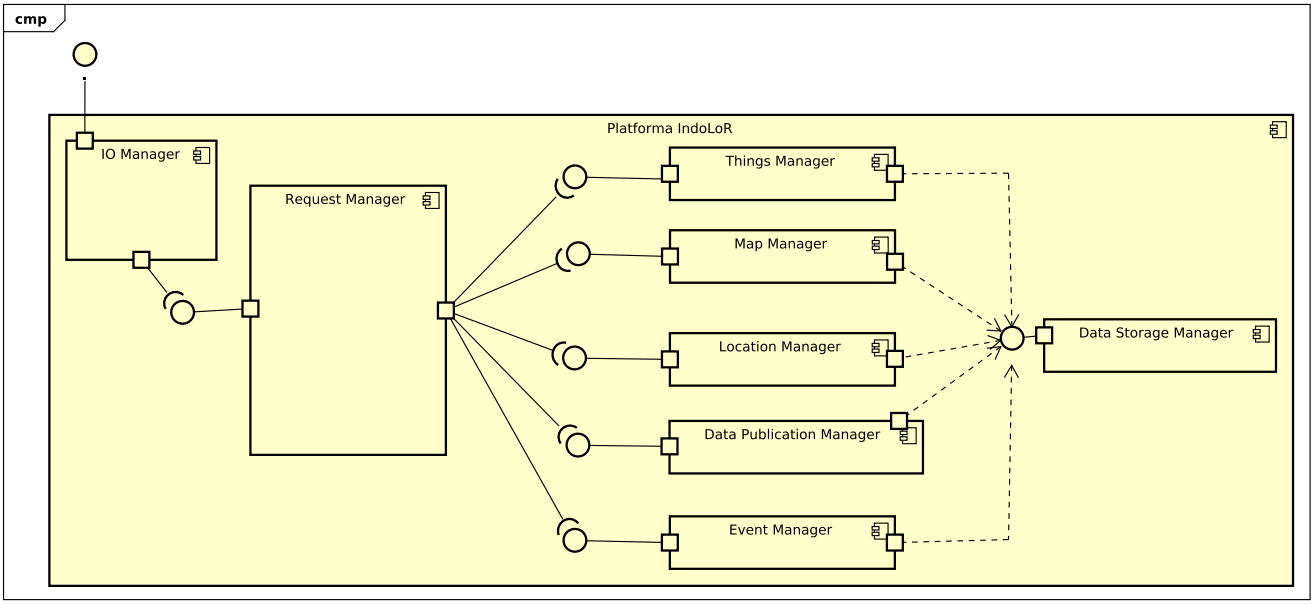


Figura 6. Visão Arquitetural de Componentes e Conectores

coisa sem quebrar outras duas outras coisas [41].

Para abarcar os fatores apresentados, o processo de adaptação das funcionalidades dos componentes da plataforma é apresentado na Figura 7. Cada componente permite a adaptação de suas funcionalidades utilizando o padrão de projeto conhecido como *Whiteboard Pattern* [42]. Em que ao invés do componente padrão buscar em um diretório o serviço que implementa a funcionalidade solicitada, o mesmo registra o interesse em serviços que implementem a adaptação de suas funcionalidades, garantindo a extensibilidade e flexibilidade. Assim, sempre que um componente adaptador se registrar no diretório de serviços, é verificado se existe algum componente padrão que possua interesse no mesmo. Caso exista, este componente é notificado. Ao receber essa notificação, o componente padrão garante o acesso a instância do serviço registrado pelo componente adaptador.

Pela utilização *Whiteboard Pattern*, há uma clara separação entre os componentes e a implementação de um componente, assim não havendo dependência da implementação entre componentes. Dessa forma, pode-se garantir que um ajuste em um componente não causa a quebra de um outro componente.

4. Estudo de caso

Para avaliar a plataforma proposta, foi realizado um estudo de caso com o objetivo de avaliar três facetas: o grau de adaptabilidade, a facilidade de criação adaptações para atender novas funcionalidades e a capacidade de processamento de solicitações. Tal estudo foi realizado utilizando a plataforma IndoLoR em um ambiente real realizando a localização interna de um usuário através do uso sinais de WIFI e tag/leitores de RFID. Para tal,o planejamento e descrição deste estudo seguiram as diretrizes estabelecidas por Yin[43], Kitchenham et al. [44] e Runeson e Höst [45].

4.1 Planejamento

Para alcançar o objetivo proposto, este estudo de caso deve responder às seguintes questões de pesquisa:

1. Qual o grau de adaptabilidade provida pela plataforma IndoLoR?

2. Criar uma adaptabilidade para plataforma IndoLoR é uma tarefa custosa?

3. O desempenho da plataforma é prejudicado quando os componentes padrões têm suas funcionalidades adaptadas?

O sujeito que envolvido no estudo de caso proposto foi um programador com vasta experiência no desenvolvimento de aplicações Java. O objeto utilizado para esse estudo de caso foi a adaptação da plataforma IndoLoR para obter a localização em ambientes internos utilizando dados WIFI e RFID, de maneira combinada. Assim, nesse cenário foi utilizado um dispositivo móvel para obter dados de WIFI e RFID presentes no ambiente, e os enviar para a plataforma a fim de realizar o processamento e calcular a posição estimada do dispositivo. Uma vez terminado o cálculo do novo posicionamento, o mesmo é enviado para o dispositivo móvel. A Figura 8 ilustra a descrição deste cenário.

As unidades de análise deste estudo são: a plataforma IndoLoR e a suas adaptações para obter a localização em ambientes internos. Para realizar o estudo proposto foi realizada a coleta de dados da quantidade de classes necessárias para realizar a adaptação de cada uma das funcionalidades utilizadas neste estudo, a quantidade de linhas de código totais

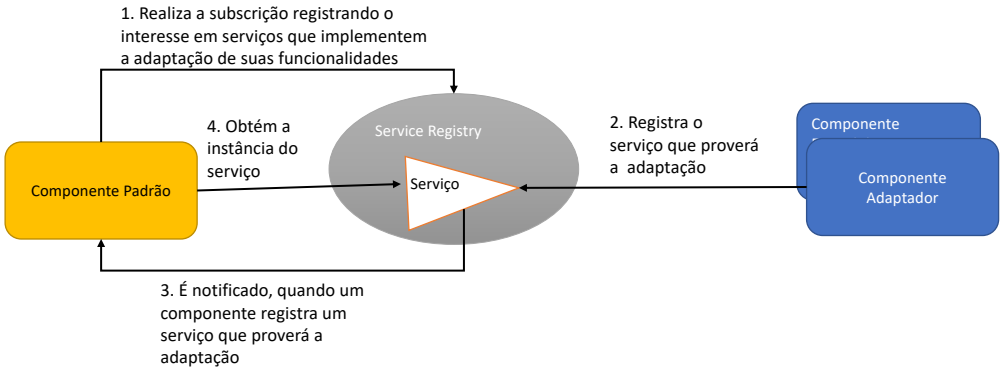


Figura 7. Processo de Adaptabilidade

e a quantidade de linhas que são necessárias para incorporar os componentes adaptadores à plataforma e os tempos de processamento das solicitações pela plataforma IndoLoR sem que haja nenhuma adaptação incorporada, com todas as adaptações incorporadas e o tempo gasto pelos componentes padrões quando há adaptações incorporadas.

4.2 Execução

Para obter a localização em um ambiente interno utilizando os dados WIFI e RFID foi necessária a construção de componentes adaptadores, de forma que os mesmos fossem incorporados à plataforma utilizando o processo de adaptabilidade descrito. Para a execução deste estudo foi necessária a criação de componentes adaptadores que implementassem as especificidades presentes no ambiente. Para tal, os componentes implementados foram:

- ***Location Data Publication Service***

  Este componente foi criado para prover a adaptação da funcionalidade de disponibilização os dados de localização calculados para um dispositivo. Sendo assim, será possível recuperar a posição atual de um determinado dispositivo, além do seu histórico.

- ***RFID Service***

  Este componente implementa a funcionalidade de processamento dos dados obtidos pelo dispositivo das tags RFID presente no ambiente e enviados para a plataforma. Ao receber essas informações, as mesmas são armazenadas e um novo posicionamento é atribuído para o dispositivo.

- ***WIFI Fusion Service***

  Este componente realiza a implementação da funcionalidade de localização utilizando dados de WIFI e RFID. A Figura 9, é apresentado como se dá o processo do cálculo do posicionamento do dispositivo utilizando dados de WIFI e RFID.

  O processo se inicia com o recebimento dos dados de WIFI pela plataforma, em seguida é verificado se não existe nenhum posicionamento obtido por RFID no último segundo, caso exista o cálculo não é realizado e o posicionamento atual será o último obtido. Caso não exista, é executado o algoritmo *Weighted Centroid* [46] [47], em que o posicionamento estimado utilizando coordenadas (x,y) é obtido a partir do cálculo do centroide do geométrico formado pelas coordenadas (x,y) de todos os *access points*. Com essas coordenadas inicias obtidas, e um novo posicionamento é atribuído para o dispositivo.

- ***WIFI Location Storage e RFID Location Storage***

  Estes componentes tem a função de realizar o armazenamento dos dados de WIFI e RFID recebidos pela plataforma. A forma de persistência escolhida para armazenamento das informações foi um utilizado o banco de dados não-relacional MongoDB [48].

O cálculo do posicionamento do usuário no ambiente realizado pelo componente *RFID Service* se inicia com o recebimento dos dados de WIFI pela plataforma, em seguida é verificado se não existe nenhum posicionamento obtido por RFID no último segundo, caso exista o cálculo não é realizado e o posicionamento atual será o último obtido. Caso não exista, é executado o algoritmo *Weighted Centroid* [46] [47], em que o posicionamento estimado utilizando coordenadas (x,y) é obtido a partir do cálculo do centroide do geométrico formado pelas coordenadas (x,y) de todos os *access points*. Com essas coordenadas inicias obtidas, e um novo posicionamento é atribuído para o dispositivo.

Além da implementação dos componentes adaptadores, foi construído um aplicativo para dispositivo móvel Android cuja função é de obter os dados de WIFI e RFID presentes no ambiente e enviá-los para a plataforma. Além disso, este aplicativo funciona como um aplicativo cliente para a utilização dos dados providos pela plataforma podendo dessa forma obter a sua localização provida pela mesma. A Figura 10 apresenta este aplicativo com o resultado da localização do dispositivo no ambiente escolhido. Em que este ambiente trata-se de um escritório com cerca de 115 m^2 com 4 *access points* previamente instalados e com configurações padrões.

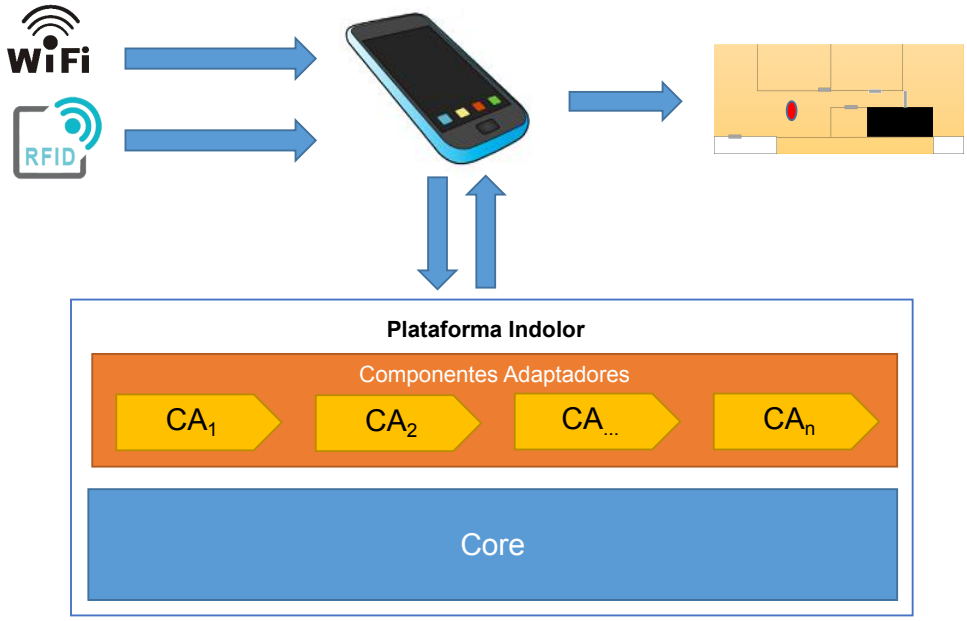


Figura 8. Cenário da Adaptação da Plataforma IndoLoR utilizando WIFI e RFID

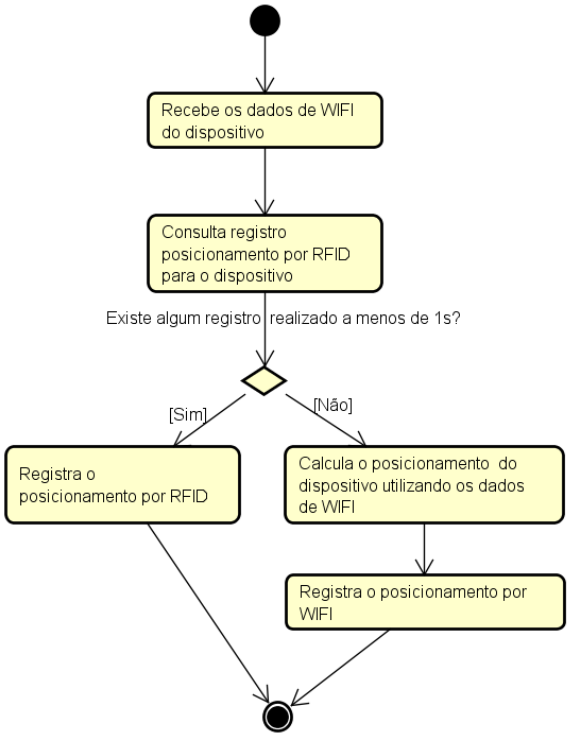


Figura 9. Processo do cálculo do posicionamento de um dispositivo utilizando uma abordagem dados heterogêneos

Além disso, em um dos cômodos foi colocada uma tag RFID presa a uma das paredes. A linha tracejada, esta linha representa a rota efetivamente executada pelo dispositivo no ambiente. Além disso, é possível perceber alguns pontos de destaque sendo representados pelas letras A,B,C,D e E. No qual estes pontos representam os locais calculados pelo algoritmo utilizado neste estudo.

Após a implementação dos módulos adaptadores e o aplicativo para dispositivo móvel responsável pela coleta dos dados do ambiente e consumo dos dados produzidos pela plataforma, iniciou-se a etapa de avaliação da adaptabilidade e performance da plataforma IndoLoR. Esta avaliação foi realizada em um computador com o sistema operacional Windows 10, processador Intel (R) Core (TM) i7-4500U CPU @ 1.80GHz e 8,00 GB de memória RAM.

A adaptabilidade foi avaliada sob duas facetas, em que a primeira é o grau de adaptabilidade provida pela plataforma IndoLoR, e a segunda leva em consideração a facilidade de realizar a adaptação e a acoplar na plataforma. Para avaliar a primeira faceta foram utilizadas duas métricas, em que a primeira é o Índice de Adaptabilidade da Arquitetura (*Architecture Adaptability Index* (AAI)) definido por Subramanian e Chung [40], em que os autores definem que adaptabilidade de uma arquitetura de software pode acontecer tanto para os componentes, quanto para os conectores, de maneira que estes elementos tem o mesmo peso ou significado na adaptabilidade.

Assim, para realizar o cálculo de tal métrica, é necessário definir o Índice de Adaptabilidade do Elemento (*Element Adaptability Index* (EAI)), em que este índice pode possuir dois valores: caso o elemento seja adaptável recebe uma unidade, caso não recebe o valor zero. A arquitetura da plataforma IndoLoR permite que tanto conectores, quanto componentes sejam adaptados pois é possível implementar novas interfaces de comunicação entre os componentes da plataforma através do processo de adaptação. Assim, a plataforma possui 15 conectores e 8 componentes. Sendo assim, o valor

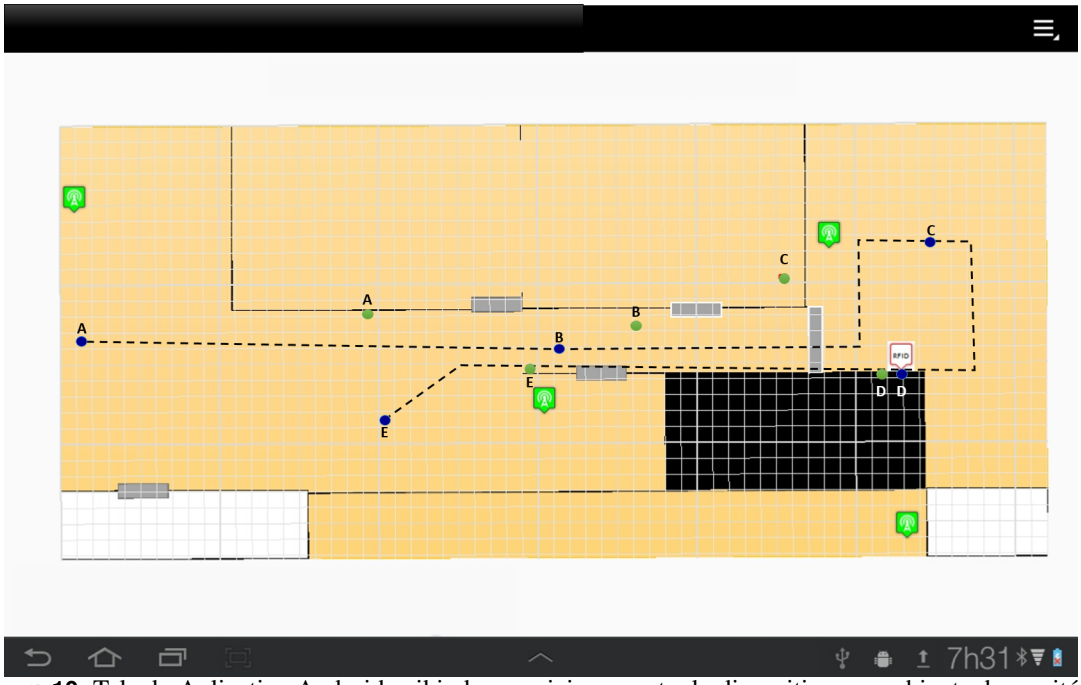


Figura 10. Tela do Aplicativo Android exibindo o posicionamento do dispositivo no ambiente de escritório

de AAI é dado pela Equação 1.

$$AAI = \frac{\sum_{i=1}^{n} EAI_i}{TotaldeElementos}$$
$$AAI = \frac{15+8}{15+8} = \frac{23}{23} = 1,0 = 100\% \quad (1)$$

A segunda métrica utilizada para avaliar o grau de adaptabilidade da plataforma é o Grau de Adaptabilidade da Arquitetura de Software (*Adaptability Degree of the Software Architecture* (ADSA)) definida por Liu e Wang [49], em que o autores definem que a definição do grau de adaptabilidade possui uma relação inversa com a métrica Impacto na Arquitetura de Software (*Impact On the Software Architecture* (IOSA)). O IOSA por sua vez é cálculado através soma dos Pontos de Classe (*Class Point*(CP)) [50] para cada cenário de adaptabilidade e sua probabilidade estimada (PC). Na Tabela 3 é apresentado o resultado do cálculo das métricas definidos por Liu e Wang [49] aplicado a plataforma IndoLoR. As equações utilizadas para o cálculo do IOSA e ADSA são descritas em 2. Em que |s| é quantidade de cenários de adaptação, PC é a probabilidade estimada do k-ésimo cenário e CP é o valor dos pontos de classe do k-ésimo cenário. O valor utilizado para N deve ser maior que 1, assim foi utilizado o valor 1,01 definido através de experimentos [49]. O cálculo completo e detalhado bem como os valores utilizados acessar podem ser obtidos em https://goo.gl/1FmwmW.

$$IOSA = \sum_{k=1}^{|s|} PC_k \times CP(C_k)$$
$$ADSA = N^{-IOSA}, N > 1 \quad (2)$$

Para avaliar a facilidade de adaptação de funcionalidades e incorporá-las na plataforma, foi coletado a quantidade de classes criadas em cada um dos componentes adaptadores utilizados neste estudo. Além disso, coletou-se a quantidade de linhas de códigos totais para cada um dos componentes e o total de linhas de código necessário para a incorporação da funcionalidade provida pelo componente adaptador na plataforma. A Tabela 4 apresenta esses dados.

Para avaliar a performance da plataforma IndoLoR, foi realizada a medição do tempo necessário para uma requisição ser totalmente processada em seu interior, sendo assim, é medido o tempo despendido entre o momento em que a solicitação é recebida até a sua resposta para o solicitante. Neste sentido, foi calculado o tempo médio que uma certa quantidade de solicitações enviadas para a plataforma tem o seu processamento realizado. Além disso, para cada quantidade de solicitações enviadas coletou-se dez amostras e, em seguida, gerou-se a média geral para cada uma dessas amostras obedecendo a Equação 3.

$$m = \sum_{i=1}^{10} \frac{\sum_{j=1}^{qs} t_j}{qs} \quad (3)$$

Onde:

- m – representa a média geral;
- qs – representa a quantidade de solicitações;
- t_j – representa o tempo de processamento para a solicitação j;

Tabela 3. Grau de Adaptabilidade da Arquitetura de Software da Plataforma IndoLoR

Cenários	PC	CP	IOSA	N	ADSA
Deve ser possível modificar a forma de comunicação entre o meio externo e a plataforma. Além disso, deve ser possível receber qualquer tipo de dado proveniente do meio externo.	0,1	18,26	1,826		
Deve ser possível incluir e remover novos componentes de processamento de requisições.	0,1	23,24	2,324		
Deve ser possível criar novos componentes para acessar coisas presentes nos ambientes monitorados.	0,1	6,64	0,664		
Possibilitar a inclusão de novas formas de tratamento de mapas.	0,1	6,64	0,664		
Possibilitar a inclusão e remoção de métdos de cálculo de localização em ambientes internos.	0,3	6,64	1,992		
Possibilitar a recuperação de qualquer tipo de dado armazenado pela plataforma para o meio externo.	0,1	6,64	0,664		
Possiblitar a inclusão de novas trativas de eventos para os dados gerenciados pela plataforma.	0,1	6,64	0,664		
Permitir a utilização de diversas formas de armazenamento, podendo incluir ou remover sempre que necessário.	0,1	6,64	0,664		
	1		9,462	1,01	0,91

Tabela 4. Tabela contedo as quantidades de classes e linhas de código dos componentes adaptadores

Componente	**Classes**	**Total de Linhas**	**Total para Adaptação**
WIFI Fusion Service	5	157	16
WIFI Location Storage	4	127	9
RFID Service	3	160	16
RFID Location Storage	3	143	9
Location Data Publication Service	4	117	14
Total	18	704	64

Desse modo, a Tabela 5 apresenta os valores do tempo médio (em milissegundos) coletados para o processamento das solicitações com apenas os componentes padrões da plataforma IndoLoR, ou seja, sem nenhum componente adaptador. Nas colunas são mostrados os tempos médios referentes a cada quantidade de solicitações (Q.S.), a qual varia de 1 até 1000000 solicitações. A linhas, por sua vez, indicam a amostra (Amost.) em que tais tempos de processamento foram mensurados.

De maneira análoga, foi realizada um outro experimento coletando o tempo médio apenas dos componentes padrões quando existem componentes adaptadores associados a plataforma, sendo apresentados na Tabela 6. Nas colunas são mostrados os tempos referentes a cada quantidade de solicitações (Q.S.), a qual varia de 1 até 1000000 solicitações. A linhas, por sua vez, indicam a amostra (Amost.) em que tais tempos de processamento foram mensurados.

Por fim, a Tabela 7 apresenta o total de processamento das solicitações pela plataforma. Em que este tempo inclui os componentes padrões e componentes adaptadores. Nas colunas são mostrados os tempos referentes a cada quantidade de solicitações (Q.S.), a qual varia de 1 até 1000000 solicitações. A linhas, por sua vez, indicam a amostra (Amost.) em que tais tempos de processamento foram mensurados.

4.3 Ameaças à validade

Para o estudo de caso realizado foram avaliados os quatro tipos de validade:

1. **Validade de construção**: a captura dos dados do tempo de processamento deste estudo de caso foi realizada de maneira automatizada por meio de software, o que mitiga o risco de falha na captura do mesmo. Além disso, esse experimento foi executado utilizando uma única máquina evitando que fatores de diferença de hardwares comprometessem os valores coletados no estudo.

2. **Validade interna**: o estudo de caso foi realizado por um programador com vasta experiência no desenvolvimento de aplicações Java, o que diminuiu o risco de que fatores relacionados a inexperiência dos programadores no desenvolvimento nas tecnologias utilizadas fugissem do controle.

Tabela 5. Tempo de processamento das solicitações na plataforma IndoLoR utilizando apenas os componentes padrões

Amost./Q.S.	1	10	100	1000	10000	100000	1000000
1	0,000	0,000	0,040	0,055	0,037	0,044	0,042
2	0,000	0,000	0,030	0,053	0,043	0,042	0,042
3	0,000	0,000	0,040	0,040	0,047	0,042	0,041
4	0,000	0,000	0,030	0,041	0,040	0,043	0,042
5	0,000	0,000	0,040	0,037	0,045	0,041	0,042
6	0,000	0,000	0,040	0,043	0,042	0,042	0,042
7	0,000	0,000	0,040	0,044	0,041	0,043	0,042
8	0,000	0,000	0,030	0,035	0,040	0,042	0,042
9	0,000	0,000	0,030	0,040	0,043	0,040	0,042
10	0,000	0,000	0,020	0,028	0,041	0,042	0,042
Média	0,000	0,000	0,034	0,042	0,042	0,042	0,042

Tabela 6. Tempo de processamento das solicitações na plataforma IndoLoR pelos componentes padrões utilizando os componentes adaptadores

Amost./Q.S.	1	10	100	1000	10000	100000	1000000
1	0,000	0,100	0,080	0,090	0,100	0,105	0,101
2	0,000	0,000	0,120	0,095	0,098	0,101	0,100
3	0,000	0,100	0,070	0,104	0,116	0,109	0,109
4	0,000	0,000	0,130	0,106	0,106	0,105	0,105
5	0,000	0,000	0,140	0,102	0,099	0,102	0,102
6	0,000	0,100	0,140	0,098	0,106	0,103	0,102
7	0,000	0,100	0,140	0,108	0,114	0,099	0,105
8	0,000	0,100	0,090	0,113	0,108	0,107	0,103
9	0,000	0,100	0,150	0,115	0,107	0,102	0,101
10	0,000	0,000	0,080	0,091	0,098	0,103	0,113
Média	0,000	0,060	0,114	0,102	0,105	0,104	0,104

3. **Validade externa**: a implementação dos componentes adaptadores utilizados neste estudo de caso foram realizadas por um programador experiente. Dessa forma, programadores iniciantes ou que ainda não dominem as tecnologias utilizadas na plataforma podem gerar componentes com uma maior quantidade de linhas de código ou de baixa qualidade. Além disso, os tempos de processamento das solicitações podem ser diretamente influenciados pelo computador em que a plataforma está sendo executada.

4. **Validade de conclusão**: neste estudo foram utilizados dados quantitativos que colaboraram para o processo de avaliação da plataforma. No que diz respeito aos dados dos tempos de processamento, os mesmos foram coletados em várias amostragens para a realização do cálculo de uma média, impedindo que desvios específicos influenciassem o resultado final do estudo.

4.4 Respostas às questões de pesquisa

Esta subseção responde as questões de pesquisa levantadas na Subseção 1.

4.4.1 QP1

Apesar da adaptabilidade ser uma propriedade qualitativa, foi possível analisar quantitavamente a arquitetura proposta através do uso das métricas definida por Subramanian e Chung [40] e Liu e Wang [49] para obter o grau de adaptabilidade de uma arquitetura de software. A métrica AAI representa o quanto, em termos de elementos, uma arquitetura de software pode ter as suas funcionalidades adaptadas. O resultado apresentado na Equação 1 demonstra que a arquitetura proposta possui um grau de adaptabilidade de 100%. Já métrica ADSA, apresentou o resultado de 0,91 conforme é demonstrado na Tabela 3. O intervalos de valores que a métrica IOSA, que é inversamente proporcional ao ADSA, é $[0,\infty]$ então os intervalos de valores que o ADSA pode assumir fica entre $[1,0]$. Dessa forma, significa que se o ADSA é igual a 1, a arquitetura é totalmente adaptável. Se for igual a 0, não é adaptável para nenhuma mudança de requisitos.

Assim, apesar de não existir outra arquitetura a título de comparação, pode-se afirmar que a arquitetura possui um alto grau de adaptabilidade.

4.4.2 QP2

Criar uma adaptabilidade para a plataforma IndoLoR demonstrou que há indícios de ser uma tarefa de baixo custo, pois é necessária uma baixa quantidade de linhas de código e de classes para tal. Conforme apresentado na Tabela 4, foi necessário um total de 704 linhas de código e 18 classes para a completa construção dos componentes necessários para este

Tabela 7. Tempo de processamento das solicitações pela plataforma IndoLoR utilizando componentes padrões e componentes adaptadores

Amost./Q.S.	1	10	100	1000	10000	100000	1000000
1	0,000	0,000	0,750	0,884	0,843	0,940	0,915
2	0,000	0,000	0,780	0,954	0,927	0,888	0,908
3	0,000	0,800	0,760	0,812	0,935	0,891	0,899
4	0,000	0,000	0,790	0,841	0,924	0,889	0,912
5	0,000	0,400	0,740	0,810	1,108	0,921	0,908
6	0,000	1,100	0,790	0,844	0,919	0,905	0,899
7	0,000	0,000	1,010	0,890	0,947	0,904	0,926
8	0,000	0,700	0,880	0,822	0,996	0,898	0,914
9	0,000	0,000	0,850	0,788	0,851	0,891	0,907
10	0,000	0,600	0,750	0,811	0,912	0,925	0,899
Média	0,000	0,360	0,810	0,846	0,936	0,905	0,909

estudo. Vale salientar que essa contagem considerou todas as linhas de código, incluindo as importações, declarações, etc.

Para avaliar o custo de incluir as adaptações na plataforma, conforme apresentado na Tabela 4 foram necessárias um total de 64 linhas de código para que os componentes adaptadores fossem acoplados a plataforma IndoLoR. Esse quantitativo levou em consideração apenas as linhas que estão diretamente ligadas ao processo de adaptação de funcionalidades. Dessa forma, tem-se que foi necessário apenas 9% do total de linhas para incorporar as adaptabilidades a plataforma.

Além da avaliação da facilidade de incorporação de adaptações na plataforma pela quantidade de código necessário realizar tal operação. Pode-se avaliar o crescimento estrutural das adaptações implementadas. Para tal, Raibulet e Masciadri [51] [52] definem algumas métricas para esta avaliação, sendo estas apresentadas na Equação 4.

$$OsAC = MsAC + \sum_{i=2}^{n} sACF_i$$
$$AvgGSE = \frac{OsAC}{n} \quad (4)$$

Onde:

- MsAC – representa o número mínimo de classes para uma adaptação.
- sACF – representa o número de classes necessárias para a adaptação da i-ésima funcionalidade.
- OsAC – representa o custo geral estrutural de todas as adaptações.
- AvgGSE – representa crescimento estrutural médio de classes necessárias para uma adaptação.

Utilizando os dados obtidos e apresentados na Tabela 4, a Figura 11 apresenta o resultado do cálculo da métrica OsAC para cada uma das funcionalidades adaptadas. Percebe-se que o gráfico obtido tende a uma função linear. Esse comportamento indica que as funcionalidades adaptadas são independentes uma das outras, dessa forma não é necessário que um programador que esteja desenvolvendo componentes adaptadores conheça ou estude a implementação de outro componente. Além disso, encontrou-se que o crescimento estrutural médio de classes para se obter uma adaptação é $AvgGSE = \frac{18}{5} = 3,6$. Portanto, são necessárias em média menos de 4 classes para se criar um componente adaptador.

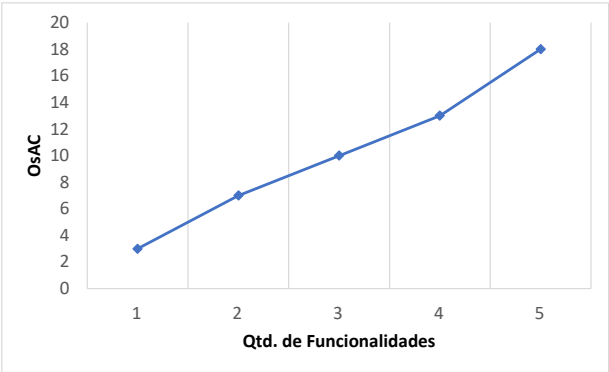
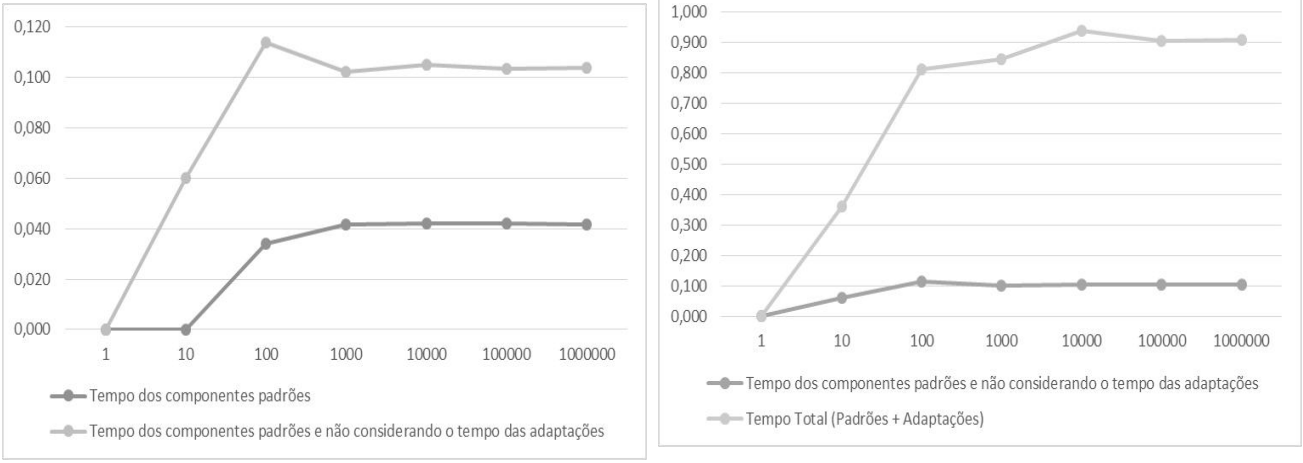


Figura 11. Gráfico da relação entre a métrica OsAC e as funcionalidades adaptadas

4.4.3 QP3

Com base nos dados obtidos na Tabela 5, que considera o tempo de processamento das solicitações pela plataforma em que apenas existem os componentes padrões, e na Tabela 6, que considera o tempo de processamento das solicitações pela plataforma em que existem as adaptações de suas funcionalidades considerando o tempo gasto apenas pelos componentes padrões é apresentada a Figura 12a. Este gráfico apresenta tempo médio de processamento para estas duas situações. Percebe-se que a inclusão de componentes adaptadores na plataforma tem influência direta no tempo de processamento dos módulos padrões, fazendo com que os mesmos aumentem o seu tempo de processamento total em cerca de 50% em relação ao tempo despendido quando não há adaptações



(a) Tempo médio de processamento das solicitações considerando apenas o componentes padrões e desconsiderando suas adaptações

(b) Tempo médio de processamento das solicitações considerando apenas padrões e desconsiderando as adaptações e o tempo médio total de processamento

Figura 12. Gráficos dos tempos médios de processamento das solicitações enviadas para a plataforma

presentes. No entanto, a arquitetura da plataforma se mostrou robusta e estável, pois mesmo com uma grande quantidade de pacotes, manteve o seu tempo médio de processamento aproximadamente estável, existindo ou não adaptações.

Se for realizada uma comparação entre os tempos de processamento gasto apenas com os componentes padrões e o tempo total de processamento considerando componentes padrões e componentes adaptadores, apresentados na Tabela 7, percebe-se que o tempo médio de processamento dos módulos padrões não ultrapassa 11% em relação ao tempo médio total de processamento, conforme apresenta o gráfico na Figura 12b. Diante disso, pode-se concluir que o tempo de processamento dos componentes padrões tem um impacto pouco significativo no tempo médio de processamento das solicitações enviadas para a plataforma IndoLoR. Isso é interessante porque demonstra que o tempo médio gasto pelos componentes de gestão de plataforma é pouco significativo. Desta forma, o tempo efetivamente dominante trata-se da lógica de negócio associada aos componentes adaptadores.

4.5 Considerações Finais sobre o estudo de caso

Com a execução deste estudo de caso, pode-se validar a utilização da plataforma com dados heterogêneos, WIFI e RFID. Além disso, pode-se validar a implementação do cálculo da localização em ambientes utilizando a fusão de dados provenientes das diferentes fontes. Com a construção e utilização do aplicativo para dispositivo móvel, comprovou-se que a plataforma cumpre o propósito definido em sua arquitetura geral, em que a mesma conseguiu obter informações de diferentes fontes, realizou o processamento transformando-a em dados de localização e as disponibilizando para as aplicações clientes.

Em relação à adaptação dos componentes padrões, foi realizada a implementação de cinco componentes adaptadores comprovando o funcionamento do processo de inclusão de adaptações às funcionalidades da plataforma. Além disso, comprovou-se que todos os componentes e conectores da arquitetura permitem a sua adaptabilidade. Ademais, foi possível traçar a rota percorrida por um dispositivo em um determinado ambiente, isso demonstra o funcionamento dos outros módulos padrões da plataforma IndoLoR e não apenas o de localização.

Por fim, este estudo de caso descreveu o processo de criação e inclusão de adaptações a plataforma que demonstrou apresentar indícios de ser um processo com um baixo custo. Além disso, dado essa característica, os componentes adaptadores funcionam de maneira independente, o que facilita o entendimento e utilização da plataforma pelos programadores pois não é necessário que se conheçam detalhes de implementação de outros componentes adaptadores. Outro importante aspecto avaliado foi a performance para realizar o processamento das solicitações enviadas à plataforma, em que o tempo de processamento despendido nos componentes padrões tem um impacto pouco significativo no processamento total das solicitações.

5. Considerações Finais e Trabalhos Futuros

Este trabalho apresentou o *design*, implementação e avaliação de uma plataforma adaptável para localização de pessoas ou objetos em ambientes internos, chamada de IndoLoR. Esta plataforma permite o uso de diferentes tecnologias, fontes de informações, técnicas, além de outras características, com o objetivo de se adaptar aos mais diferentes e complexos ambientes internos. Além disso, foi apresentado que a plataforma é dividida em componentes padrões e componentes adaptadores. Estes componentes padrões são providos pela arquitetura da plataforma e podem ter suas funcionalidades adaptadas pelos componentes adaptadores. Foi apresentado que todos os com-

ponentes padrões podem ter suas funcionalidades adaptadas, permitindo o uso nos mais diversos ambientes e tecnologias.

A plataforma proposta foi avaliada através da execução de um estudo de caso em que se utilizou dados de WIFI e RFID para localizar um dispositivo em um determinado ambiente. Esse estudo buscou avaliar a facilidade de adaptação dos componentes da plataforma e tempo médio de processamento de seus componentes principais. Dessa forma, foi possível comprovar que a plataforma IndoLoR permite a utilização de diversas fontes de informações e tecnologias. Além disso, os resultados se mostraram bastante satisfatórios, de maneira que a plataforma permite que todos os seus componentes padrões sejam adaptados e mostrou que o processo de incorporação de uma adaptação é uma tarefa simples e de baixo custo de implementação. Ademais, mostrou que mesmo incluindo adaptações, o tempo médio de processamento das solicitações permanece constante, demostrando a robustez da arquitetura proposta.

Como direções futuras, pretende-se avaliar a operação da plataforma utilizando outras tecnologias e técnicas a fim de verificar possíveis melhorias para a arquitetura proposta. Almeja-se que outros desenvolvedores construam suas adaptações de maneira a contribuir e melhorar a arquitetura proposta. Além disso, tem-se a intenção de avaliar outras características tais como: escalabilidade e processamento distribuído. Pretende-se evoluir a arquitetura da plataforma para incluir conceitos de sensibilidade ao contexto, *cloud computing* e *big data*.

6. Contribuição dos Autores

Mário foi o autor principal e responsável pela pesquisa científica. Desenvolveu a plataforma proposta e escreveu o artigo.

Gibeon, além de contribuir com o processo de revisão, foi a orientador do trabalho.

Referências

[1] HE, J. et al. A practical indoor toa ranging error model for localization algorithm. In: IEEE. *2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications*. Toronto, Canada: IEEE, 2011. v. 1.

[2] LIU, H. et al. Survey of wireless indoor positioning techniques and systems. *IEEE Trans Syst Man Cybern C.*, v. 37, n. 6, p. 1067–1080, 2007.

[3] HIGHTOWER, J.; BORRIELLO, G. Location systems for ubiquitous computing. *Comput.*, v. 34, n. 8, p. 57–66, 2001.

[4] SATYANARAYANAN, M. Pervasive computing: Vision and challenges. *Personal Commun.*, v. 8, n. 4, p. 10–17, 2001.

[5] WIN, M. Z. et al. Network localization and navigation via cooperation. *IEEE COMMUN MAG*, v. 49, n. 5, p. 56–62, 2011.

[6] SIMONI, M. et al. Indoor air pollution and respiratory health in the elderly. *Eur Respir J.*, v. 21, n. 40 suppl, p. 15–20, 2003.

[7] DODGE, D. *Why Indoor Location will be bigger than GPS or Maps, and how it works*. 2015. Online. Disponível em: <http://dondodge.typepad.com/the_next_big_thing/2013/04/why-indoor-location-will-be-bigger-than-gps-or-maps.html>.

[8] RESEARCH, O. *Mapping the Indoor Marketing Opportunity*. 2015. Online. Disponível em: <http://opusresearch.net/wordpress/pdfreports/OpusIndoorReport_24_Jan2014.pdf>.

[9] JUNAIO. *Junaio Augmented Reality Browser*. 2015. Online. Disponível em: <https://itunes.apple.com/br/app/junaio-augmented-reality-browser/id337415615?mt=8>.

[10] MAZEMAP. *Mazemap*. 2015. Online. Disponível em: <https://use.mazemap.com/?v=1>.

[11] BEESTAR. *Insight*. 2015. Online. Disponível em: <http://www.beestar.eu/>.

[12] ANYPLACE. *AnyPlace*. 2015. Online. Disponível em: <http://anyplace.cs.ucy.ac.cy/>.

[13] FRAUNHOFER. *awiloc*. 2015. Online. Disponível em: <http://www.iis.fraunhofer.de/en/ff/lok/tech/feldstaerke/rssi.html>.

[14] LIPS. *Local Indoor Positioning System*. 2015. Online. Disponível em: <http://lips.si/>.

[15] POINTINSIDE. *storemode*. 2015. Online. Disponível em: <http://www.pointinside.com/storemode/indoor-mapping/>.

[16] HERE. *Consumer*. 2015. Online. Disponível em: <https://company.here.com/consumer/>.

[17] GLOPOS. *GloPos*. 2015. Online. Disponível em: <http://glopos.com/indoor-positioning.html>.

[18] HOFFMANN-WELLENHOF, B.; LICHTENEGGER, H.; COLLINS, J. *GPS: theory and practice*. 5. ed. Vienna, Áustria: Springer-Verlag Wien, 1994. v. 1.

[19] KAPLAN, E.; HEGARTY, C. *Understanding GPS: principles and applications*. 2. ed. Boston, USA: Artech house, 2005. v. 1.

[20] CHOSA, T. et al. Evaluation of the dynamic accuracy of a gps receiver: Is dynamic accuracy the same as static. *EAEF*, v. 4, n. 2, p. 54 – 61, 2011.

[21] LEMIC, F. et al. Web-based platform for evaluation of rf-based indoor localization algorithms. In: ICCW. *IEEE International Conference on Communication Workshop*. London, UK: IEEE, 2015. v. 1.

[22] GOOGLE. *Indoor Maps*. 2014. Online. Disponível em: <http://www.google.com/intl/pt-BR/maps/about/partners/indoormaps/>.

[23] APPLE. *Footprint: Indoor Positioning with Core Location*. 2015. Online. Disponível em: <http://developer.apple.com/library/ios/samplecode/footprint/>.

[24] INSIDER, B. *Apple Is Launching A Vast Project To Map The Inside Of Every Large Building It Can*. 2015. Online. Disponível em: <http://www.businessinsider.com/apple-indoor-mapping-project-and-ibeacon-2014-6>.

[25] SUMMIT, D. *Indoor Positioning and Beacons:*. 2015. Online. Disponível em: <http://www.technologyreview.com/summit/14/digital/video/watch/indoor-positioning-and-beacons/>.

[26] LYMBEROPOULOS, D. et al. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. In: NATH, S. (Ed.). *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2015. v. 1.

[27] MELO, M.; AQUINO, G. A taxonomy of technologies for fingerprint-based indoor localization. In: ALMEIDA, H. (Ed.). *7o Simpósio Brasileiro de Computação Ubíqua e Pervasiva*. Recife, Brazil: SBC, 2015. (1, v. 1).

[28] MELO, M.; AQUINO, G. Categorization of technologies used for fingerprint-based indoor localization. In: BORCOCI, E. et al. (Ed.). *Tenth International Conference on Systems and Networks Communications*. Barcelona, Spain: IEEE, 2015. v. 1.

[29] RAZZAQUE, M. Middleware for internet of things: a survey. *IEEE IoT J.*, v. 3, n. 1, p. 70–95, 2016.

[30] CAPRA, L.; EMMERICH, W.; MASCOLO, C. Reflective middleware solutions for context-aware applications. In: YONEZAWA, A.; MATSUOKA, S. (Ed.). *Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*. London, UK, UK: Springer-Verlag, 2001. v. 1.

[31] HIGHTOWER, J.; BRUMITT, B.; BORRIELLO, G. The location stack: A layered model for location in ubiquitous computing. In: WMCSA. *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2002. v. 1.

[32] NAJIB, W.; KLEPAL, M.; WIBOWO, S. B. Mapume: Scalable middleware for location aware computing applications. In: MOREIRA, A.; MAUTZ, R. (Ed.). *2011 International Conference on Indoor Positioning and Indoor Navigation*. Guimaraes, Portugal: IEEE, 2011. v. 1.

[33] STEVENSON, G. et al. Loc8: A location model and extensible framework for programming with location. *IEEE Pervasive Comput.*, v. 9, n. 1, p. 28–37, 2010.

[34] RANGANATHAN, A. et al. Middlewhere: A middleware for location awareness in ubiquitous computing applications. In: JACOBSEN, H.-A. (Ed.). *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*. Berlin, Heidelberg: Springer-Verlag, 2004. v. 1.

[35] TARVAINEN, P. Adaptability evaluation at software architecture level. *TOCENGJ*, v. 2, n. 1, p. 1–30, 2008.

[36] GRESSMANN, B.; KLIMEK, H.; TURAU, V. Towards ubiquitous indoor location based services and indoor navigation. In: ZEISBERG, S.; KAISER, T. (Ed.). *2010 7th Workshop on Positioning, Navigation and Communication*. Dresden, Germany: IEEE, 2010. v. 1.

[37] SIMON, R.; FRöHLICH, P.; ANEGG, H. Beyond location based–the spatially aware mobile phone. In: CARSWELL, J. D.; TEZUKA, T. (Ed.). *Web and wireless geographical information systems*. Berlin, Germany: Springer, 2006, (Lecture Notes in Computer Science, v. 4295). p. 12–21.

[38] CLEMENTS, P. et al. *Documenting Software Architectures: Views and Beyond*. 2. ed. Boston, USA: Addison-Wesley Professional, 2003. v. 1.

[39] ERL, T. *SOA Principles of Service Design*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2016. v. 1.

[40] SUBRAMANIAN, N.; CHUNG, L. *Metrics for Software Adaptability*. [S.l.], 1999. v. 1, n. 1, 95–108 p.

[41] FAYAD, M.; CLINE, M. P. Aspects of software adaptability. *COMMUN. ACM*, v. 39, n. 10, p. 58–59, 1996.

[42] KRIENS, P.; HARGRAVE, B. *Listeners considered harmful: The whiteboard pattern*. San Ramon, USA, 2004.

[43] YIN, R. K. *Case study research: Design and methods*. 1. ed. Thousand Oaks, USA: Sage publications, 2013. v. 1.

[44] KITCHENHAM, B.; PICKARD, L.; PFLEEGER, S. L. Case studies for method and tool evaluation. *IEEE software*, v. 1, n. 4, p. 52–62, 1995.

[45] SIMON, R.; FRÖHLICH, P.; ANEGG, H. Guidelines for conducting and reporting case study research in software engineering. *EMPIR SOFTW ENG*, v. 14, n. 2, p. 131–164, 2009.

[46] KONRAD, P. W. T. *Wifi compass wifi access point localization with android devices*. Dissertação (Mestrado) — Information Security program at St. Polten University of Applied Sciences, St. Polten, Áustria, 2012.

[47] HAN, D. et al. Passive and active network measurement. In: HUTCHISON, D. et al. (Ed.). *Computer Communication Networks and Telecommunications*. Berlin, Heidelberg: Springer-Verlag, 2009, (Lecture Notes in Computer Science, v. 5448). p. 99–108.

[48] MONGODB. *MongoDB*. 2015. Online. Disponível em: <https://www.mongodb.org>.

[49] LIU, X.; WANG, Q. Study on application of a quantitative evaluation approach for software architecture adaptability. In: TSE, T. H. (Ed.). *Fifth International Conference on Quality Software*. Melbourne, Australy: IEEE, 2005. v. 1.

[50] COSTAGLIOLA, G. et al. Class point: an approach for the size estimation of object-oriented systems. *IEEE Trans. Softw. Eng.*, v. 31, n. 1, p. 52–74, 2005.

[51] RAIBULET, C.; MASCIADRI, L. Evaluation of dynamic adaptivity through metrics: an achievable. In: WICSA/ECSA. *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*. Cambridge, UK: IEEE, 2009. v. 1.

[52] RAIBULET, C.; MASCIADRI, L. Metrics for the evaluation of adaptivity aspects in software systems. *J Advances softw.*, v. 3, n. 1 & 2, p. 238–251, 2010.