



Uma Arquitetura para um Servidor de Contexto

Rafael Alves Feliciano

JUIZ DE FORA
FEVEREIRO, 2014

Uma Arquitetura para um Servidor de Contexto

RAFAEL ALVES FELICIANO

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Eduardo Barrere

JUIZ DE FORA
FEVEREIRO, 2014

UMA ARQUITETURA PARA UM SERVIDOR DE CONTEXTO

Rafael Alves Feliciano

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Barrere
Dr. em Engenharia de Sistemas e Computação, UFRJ

Marcelo Ferreira Moreno
Dr. em Informática, PUC-Rio

Victor Stroele de Andrade Menezes
Dr. em Engenharia de Sistemas e Computação, UFRJ

JUIZ DE FORA
11 DE FEVEREIRO, 2014

A Deus acima de todas as coisas.

A minha mãe, pelo apoio e sustento.

Ao meu irmão e amigos.

Resumo

Quando se tira uma foto de um celular, queremos armazenar informações sobre o contexto da foto: local (gps), quem (redes sociais), onde (mecanismos de busca de objetos), quando, etc. Sistemas multimídias e sensíveis ao contexto, tem emergido como um novo domínio de aplicativos para smartphones, tablets e outros dispositivos móveis graças à adição de sensores, dados do usuário e funcionalidades multimídias para esses dispositivos. Neste trabalho, deseja-se desenvolver uma arquitetura para um servidor de contexto que disponibilize as informações contextuais de mídias e propor um serviço que armazene as informações de contexto.

Palavras-chave: Sistemas multimídias, dispositivos móveis, sensível ao contexto.

Abstract

As you take a picture of a cell, we store information about the context of the photo: location (gps), who (social networks), where (search engines objects), when, etc.. Multimedia systems and context-sensitive, has emerged as a new field of applications for smartphones, tablets and other mobile devices thanks to the addition of sensors, user data and multimedia capabilities for these devices. In this work, we want to develop an architecture for a server context that provides the contextual information media and propose a service that stores the context information.

Keywords: Multimedia systems, mobile, context-sensitive.

Agradecimentos

Agradeço primeiramente a Deus por toda graça alcançada na minha vida e em todo meu curso.

Agradeço ao meu pai (*in memoriam*), pelo carinho e amor me concedido em vida.

A minha mãe, pelo apoio incondicional e preocupação das noites mau dormidas.

Ao meu irmão pelo carinho e companheirismo.

A todos os meus parentes e amigos, pelo encorajamento e apoio.

Aos amigos da faculdade que de algum modo contribuíram pela minha formação.

Ao professor Eduardo Barrere pela contribuição, orientação e paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

*”O único lugar onde ’sucesso’ vem antes
do ’trabalho’ é no dicionário.”*

Albert Einstein

Sumário

Lista de Figuras	7
Lista de Tabelas	8
1 Introdução	9
1.1 Objetivos	11
1.2 Estrutura do Trabalho	11
2 Pressupostos Teóricos	12
3 Servidor de contexto	15
3.1 Arquitetura geral	15
3.2 Media producer	16
3.3 Media context server	16
3.4 Multimedia or context applications	16
3.5 Context server	16
3.5.1 Context Storage Service	17
3.5.2 Context Consumer Service	17
3.5.3 Security	17
3.5.4 Context Transaction	19
3.5.5 Persistence	19
4 Context Server: Serviços e arquitetura orientada a eventos	20
4.1 Serviços implementados	20
4.1.1 Storage	20
4.1.2 Consumer	21
4.2 Arquitetura orientada a eventos	21
4.2.1 <i>EventBus</i>	22
5 Protótipo	24
5.1 Estudo do Armazenamento em um SGBD	24
5.2 Acesso aos serviços do servidor	25
5.3 Envio de xml para Storage	27
5.4 Envio de xml para Consumer	28
6 Conclusão	31
Referências Bibliográficas	32

Lista de Figuras

3.1	Arquitetura geral: produtor, consumidor, servidor de conteúdo e servidor de contextos	15
3.2	Arquitetura do Media producer	16
3.3	Arquitetura das aplicações que consomem contexto	17
3.4	Arquitetura do Context server	18
3.5	Exemplo de xml do contexto para persistência	18
3.6	Exemplo de xml do contexto para consumo	19
4.1	Exmplo de xml com as mídias pesquisadas	22
4.2	Interação entre o bus e os módulos da arquitetura.	23
5.1	Página do server funcionando.	26
5.2	Advanced REST client	26
5.3	Envio de xml para Storage	27
5.4	Resposta do servidor	28
5.5	Contexto persistido no banco	28
5.6	Envio de xml para Consumer	29
5.7	Resposta do servidor	30
5.8	Xml de reposta da consulta feita	30

Lista de Tabelas

5.1	Tabela com campos persistidos no banco	25
5.2	Tabela com exmplo do dados fictícios persistidos no banco	25

1 Introdução

Com o advento das máquinas fotográficas digitais, celulares com câmera embutida e mais recentemente os tablets, é cada vez mais comum a geração de fotos digitais. Isso tem gerado uma grande quantidade de imagens digitais compartilhadas em sites, blogs e redes sociais.

Essa grande quantidade de imagens gera um desafio: associá-las aos contextos em que foram geradas para uso futuro. Neste sentido, existem diversos softwares para catalogação de imagens, nos quais o usuário deve informar a qual contexto a imagem pertence e se essa informação está vinculada somente aos software em questão, sem fornecer um padrão que permita o intercâmbio dessa informação com outros softwares e aplicações multimídia de forma geral.

Para resolver esse intercâmbio, uma das soluções interessantes são as informações de contexto (Feng et al, 2008), que permitem associar (automaticamente ou não) informações complementares à foto. Ao visualizar uma foto, se obtém informações de acordo com o contexto em que ela foi adquirida, desde o momento, o local e quem está presente, podem ser informações de uma foto.

Geralmente, a noção de contexto refere-se ao estado do usuário e do ambiente em certo momento. O termo "sensível ao contexto" é associado a sistemas que guiam o seu comportamento de acordo com o seu contexto de uso. Em sistemas multimídias, a noção de contexto e o seu uso são um pouco distintos. Braga et al (2012), concluíram que a maioria das informações referidas pelas pessoas sobre suas fotos estão relacionadas ao contexto no momento em que a foto é tirada ("quando", "onde", "com quem", etc.). Muitas informações referidas pelas pessoas sobre as fotos remetem ao momento em que foi criada. Essas informações auxiliam a busca por uma foto específica.

Atualmente, os dispositivos móveis, equipados com câmeras digitais e sensores, fornecem as tecnologias capazes de reunir essas informações de contexto que serão armazenados numa base de dados em um servidor conforme padrões e posteriormente essas informações poderão ser utilizadas por aplicações.

A geração de conteúdo digital está tornando muito popular devido a várias vantagens que incluem fácil processamento, cópia, partilha e armazenamento barato que não estão mais limitados às câmeras digitais, mas incluem celulares, PDAs e outros.

A popularização de dispositivos móveis com câmeras digitais e o baixo custo para o armazenamento de conteúdo (fotos e vídeos) estão gerando uma grande quantidade de informação, o que torna difícil o seu gerenciamento, (Viana et al, 2011). Como consequência, há um novo desafio de anotar contextualmente, armazenar e recuperar estas e muitas outras coleções de conteúdo.

Uma das maneiras que os usuários usam para recordarem das suas fotos é através da identificação das pessoas presentes nestas. No padrão adotado pela maioria dos fabricantes de câmeras digitais, (Lacerda et al, 2008), há um campo que armazena uma descrição de uma determinada fotografia que pode ser editada manualmente pelo usuário. Deste modo, um dono de uma determinada coleção pode nomear as pessoas presentes nas fotografias. No entanto, para uma a grande coleção essa solução se torna inviável pois demanda muito tempo para o usuário descrever as fotografias.

A evolução dos dispositivos móveis está permitindo que haja um aumento dos celulares com GPS embutido, surgindo assim sistemas multimídias capazes de reunir informações de contexto. Com isso, os serviços baseados em localização estão se tornando soluções mais viáveis. Por exemplo, descobrir a localização fornecidas por esses serviços são muito úteis para anotação de fotografias, pois os amigos que estiverem próximos no momento da captura têm uma grande chance de estarem presentes nela.

Esses sistemas multimídias sensíveis ao contexto podem ser classificados em três grupos, (Braga et al, 2011):

- (i) O primeiro define sistemas que fornecem anotação de fotos usando metadados contextuais;
- (ii) O segundo grupo contém sistemas que usam o contexto para auxiliar a experiência de compartilhar conteúdo;
- (iii) O terceiro grupo é a geração de novos conteúdos com base nos metadados contextuais e conteúdos multimídias produzidas pelos usuário.

Entretanto, com muitos dados de contextos sendo gerados, é necessário um serviço que os armazene e disponibilize.

Neste trabalho, deseja-se desenvolver uma arquitetura que utilize os conceitos de anotação contextual, tanto na parte de armazenamento, quanto na parte de recuperação e utilização das informações obtidas.

1.1 Objetivos

Propor uma arquitetura para armazenar as informações contextuais de mídias. Definir a melhor alternativa de armazenamento dos dados contextuais e desenvolver um serviço para armazenar e fornecer as informações de contexto de dados multimídias para aplicações.

1.2 Estrutura do Trabalho

No Capítulo 2 é apresentado a fundamentação teórica, bem como o conceito de contexto, evento e anotação automática de imagens. O Capítulo 3 apresenta a arquitetura proposta no trabalho bem como os módulos que a representa. No Capítulo 4 é apresentado os serviços para acesso a arquitetura. No Capítulo 5 é apresentado um protótipo de implementação da arquitetura. No sexto e último capítulo descreve as conclusões tiradas do trabalho e apresenta as perspectivas de trabalhos futuros relacionados.

2 Pressupostos Teóricos

Com o crescente aumento do número de coleções de imagens, devido ao advento da fotografia, se faz necessário uma forma de pesquisá-las e procurá-las. Atualmente, testemunhamos um progresso nos dispositivos que geram essas coleções (**dispositivos móveis**), bem como seu desempenho. Procurar e encontrar um grande número de imagens e vídeos a partir de uma coleção é um problema desafiador, que pode ser resolvido anotando informações sobre as imagens.

A captura de fotografias é uma das funções mais comuns em dispositivos móveis atuais. Com a facilidade de sempre estar com a câmera, usuários tem capturado uma quantidade cada vez maior de fotografias digitais. Uma informação de contexto refere-se a um determinado acontecimento acerca de **quem, onde e quando**.

Historicamente, essas anotações eram informadas para cada imagem separadamente, (Lavrenko et al, 2003). Um processo exaustivo e caro nos tempos atuais. Essas anotações são usadas para recuperar imagens apropriadas, pressupondo que o texto pode capturar a semântica da imagem requerida. Como é custoso anotar cada imagem separadamente, tem havido um grande interesse em formas de anotar as imagens a partir do seu conteúdo. Conceitos com base em conteúdo e imagens, como cor, textura têm sido propostos para a recuperação de conteúdo de imagem, mas a maioria das pessoas prefere representar as consultas com texto a achar imagens relevantes para essa consulta em um determinado evento.

Um **evento** é caracterizado por um acontecimento que reúne pessoas com um objetivo em comum, em um determinado lugar e durante um determinado período de tempo. Exemplos de eventos incluem festa de aniversário, concertos, casamentos, etc (Wagenaar et al, 1986).

A captura de várias fotografias por câmeras distintas, em um mesmo evento, é uma situação corriqueira. Entretanto, cada pessoa possuidora de uma câmera fotográfica, possui uma organização das fotografias diferente. Pelo fato de câmeras distintas possuírem funções distintas, é possível que fotografias capturadas num mesmo evento possuam tipos

de informações de contexto diferentes. Por exemplo, as fotografias capturadas a partir da câmera de um smartphone equipado com GPS possuem a informação de localização geográfica nos seus metadados.

Um problema para generalizar as consultas com sistemas de recuperação de imagens a base de texto, é obter as palavras-chave que descrevem as imagens. A lógica é que as palavras-chave capturem o conteúdo semântico da imagem e ajudem a recuperar as imagens, (Lavrenko et al, 2003). Essa técnica também é utilizada por organizações de notícias de televisão para recuperar imagens de arquivo a partir de seus vídeos (Hironobu et al, 1999). Formalmente, uma imagem I com características visuais $V_i = \{v_1, v_2, \dots, v_n\}$ e um conjunto de palavras-chave $W = \{w_1, w_2, \dots, w_n\}$, a tarefa consiste em encontrar um subconjunto automaticamente da palavra-chave $W_i \subset W$, que pode apropriadamente descrever a imagem I , (Feng et al, 2008).

Um dos problemas atuais abordados na literatura com relação a anotação de fotografias é a detecção automática de eventos. A maioria das abordagens propostas utiliza informações visuais, temporais e espaciais.

As abordagens visuais são baseadas no conteúdo da fotografia. As características visuais extraídas utilizam técnicas de recuperação de informação baseada no conteúdo (**content-based image retrieval**, (Smeulders et al, 2000)). Fotografias capturadas no mesmo evento, geralmente contêm características visuais similares, com as cores, textura, formato e estrutura, Vailaya et al (2001). Por exemplo, fotografias capturadas durante uma caminhada na praia em um dia ensolarado possuem similaridades visuais significantes.

As abordagens baseadas em metadados temporais utilizam a data e a hora extraídas dos metadados das fotografias. A utilização da informação temporal é baseada na teoria da explosão de fotografias, na qual fotografias são capturadas em curtos intervalos de tempo durante um evento, (Naaman et al, 2004).

As abordagens que utilizam os metadados espaciais e temporais, baseiam-se no fato de que eventos ocorrem em algum lugar, em um determinado momento. Dessa forma, um evento contém fotografias de lugares próximo.

Métodos atuais de anotação de imagem funcionam bem quando grandes quantida-

des de imagens rotuladas estão disponíveis, mas podem passar por dificuldades, quando o número de imagens e palavra-chave para um determinado tema é relativamente pequeno.

Anotação automática de imagens é uma tarefa popular em Visão Computacional. As primeiras abordagens estão intimamente relacionadas com a classificação da imagem, (Feng et al, 2008), onde as imagens são atribuídas a um conjunto de descrições simples, como a paisagem interior, exterior, pessoas, animais, etc.

Algumas arquiteturas para servidor de contexto foram desenvolvidas (Braga et al, 2011) (Freitas, 2002), porém, elas abordam casos específicos o que dificulta uma reutilização das suas funcionalidades.

3 Servidor de contexto

O servidor de contexto apresentado neste capítulo faz parte do servidor de contexto em desenvolvimento pelo grupo de pesquisa LApIC (Laboratório de Aplicações e Inovação em Computação) do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora. No desenvolvimento deste servidor, outros alunos de graduação e pós-graduação colaboram desenvolvendo partes específicas como MediaProducer e um módulo para segmentação de vídeo (parte integrante do MediaAnalyser).

3.1 Arquitetura geral

A arquitetura geral, apresentada na figura 3.1, demonstra a interação entre o servidor de contexto (context server) que é o módulo a ser desenvolvido neste trabalho, servidor de conteúdo (media content server), produtor de mídias (media producer), módulo desenvolvido em outro trabalho de conclusão de curso (Barbosa, 2013) e aplicativos de contexto (multimedia or context apps). Estes módulos transferem entre si informações de contexto adquiridas pelo *Media Producer* e/ou produzidas pelo *Context Server*.

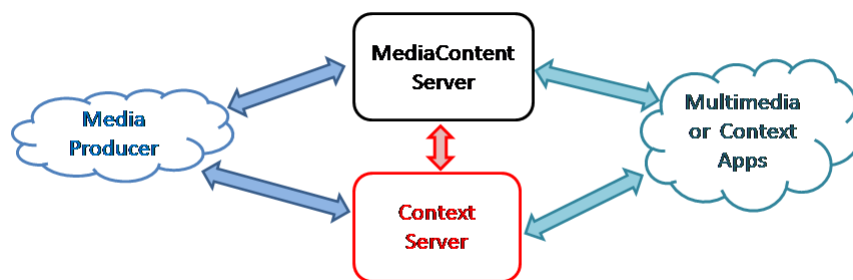


Figura 3.1: Arquitetura geral: produtor, consumidor, servidor de conteúdo e servidor de contextos

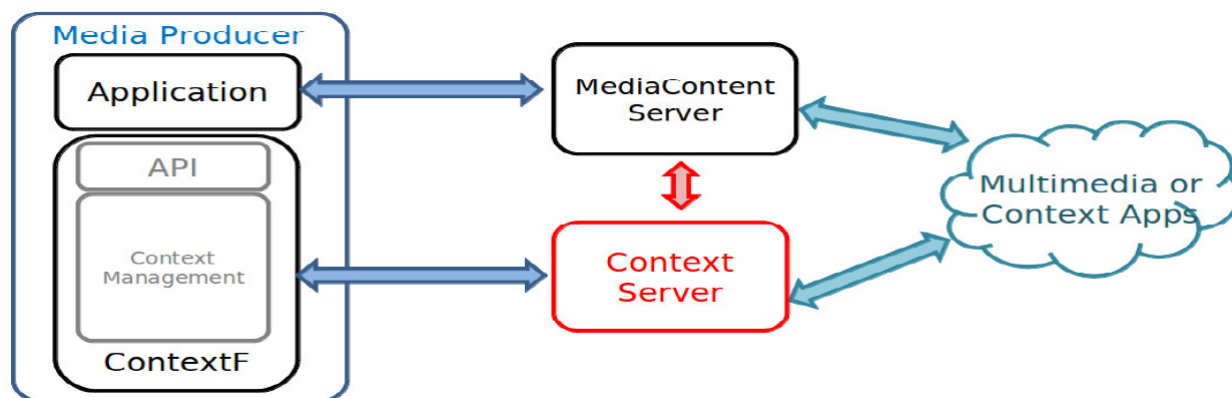


Figura 3.2: Arquitetura do Media producer

3.2 Media producer

Responsável por capturar recursos multimídias e obter informações de contexto básicas. A figura 3.2 exibe uma visão da arquitetura do Media producer.

3.3 Media context server

Servidor que contém as mídias enviadas pelo produtor. Pode ser qualquer repositório de mídias.

3.4 Multimedia or context applications

São aplicações que consomem informações de contexto. Podem ser outras aplicações que geram contextos; aplicações que exibem informações do contexto; aplicações multimídias para tv digital. A figura 3.3 exibe a arquitetura dessas aplicações.

3.5 Context server

A arquitetura proposta é um modelo cliente-servidor onde os clientes são produtores e consumidores de mídias com o conteúdo inserido em um arquivo xml e o servidor é um web service, a ser implementado em uma máquina física e conectada a internet.

Neste trabalho, é apresentado uma arquitetura para o servidor de contexto e as

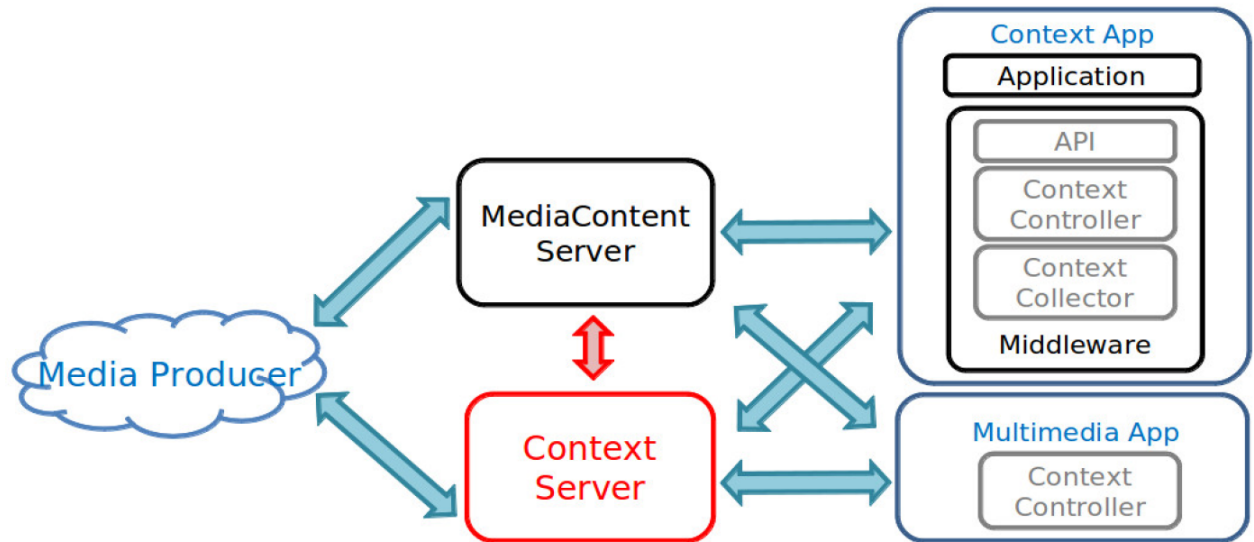


Figura 3.3: Arquitetura das aplicações que consomem contexto

interações entre seus módulos. O servidor recebe, armazena, processa e disponibiliza as informações de contexto enviadas pelo produtor e consumidor. A figura 3.4 exibe as camadas que compõe o servidor.

3.5.1 Context Storage Service

Módulo responsável por receber uma requisição de produtor de mídia (media producer), contendo um arquivo xml (figura 3.5) com informações de contexto para ser persistido. Ele monta um objeto contendo os dados do xml e repassa ao módulo de segurança.

3.5.2 Context Consumer Service

Módulo responsável por receber uma requisição para consumir algum dado persistido no sistema. Ele monta um objeto com os dados do xml (figura 3.6) com informações de contexto a ser pesquisados e repassa para o módulo de segurança.

3.5.3 Security

Módulo responsável por implementar a segurança com regras de autenticação e privacidade das informações. Este módulo não será implementado neste trabalho.

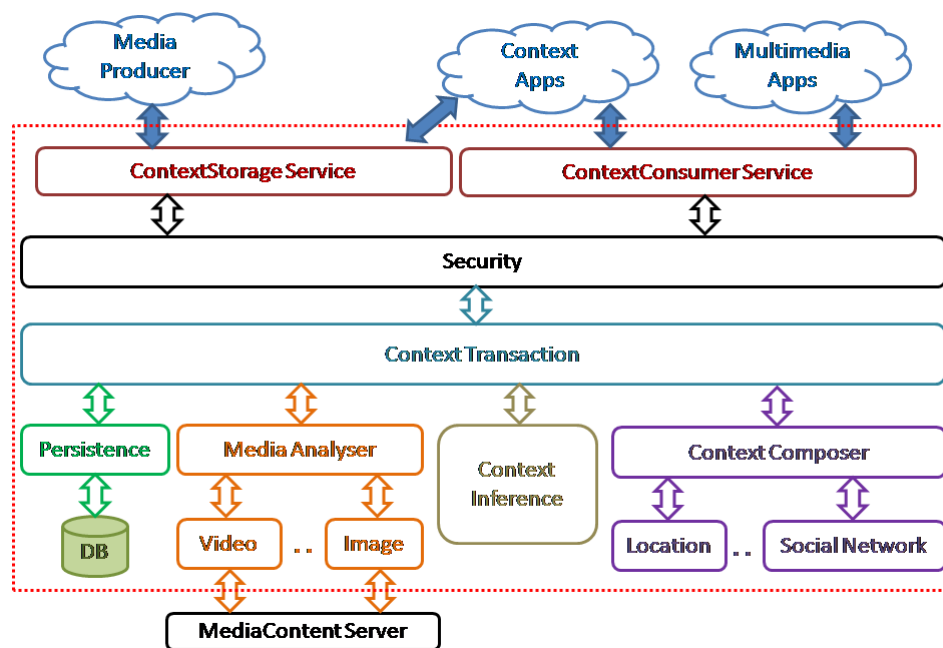


Figura 3.4: Arquitetura do Context server

```

<resource>
  <media-content-url></media-content-url>
  <resource-type></resource-type>
  <user>
    <user-type></user-type>
    <user-id></user-id>
  </user>
  <role></role>
  <context>
    <temporal>
      <created-at></created-at>
      <synced-at></synced-at>
    </temporal>
    <spatial>
      <latitude></latitude>
      <longitude></longitude>
    </spatial>
    <device>
      <resolution></resolution>
    </device>
  </context>
</resource>

```

Figura 3.5: Exemplo de xml do contexto para persistência



Figura 3.6: Exemplo de xml do contexto para consumo

3.5.4 Context Transaction

Módulo responsável por gerenciar as requisições feitas pelos módulos internos ao servidor. Ele é responsável por direcionar as interações entre os módulos *ContextStorage Service*, *Context Consumer Service* com os módulos de *Persistence*, *Media Analyser*, *Context Inference* e *Context Composer*.

3.5.5 Persistence

Módulo responsável por armazenar de forma persistente as informações de contexto.

4 Context Server: Serviços e arquitetura orientada a eventos

A comunicação do *ContextStorage Service* e *ContextConsumer Service* com o meio externo a arquitetura (*Media Producer*, *Context App* e *Multimedia Apps*) se dá a partir de serviços implementados em um web service.

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web services* são componentes que permitem às aplicações enviar e receber dados em formato XML ou JSON. Cada linguagem pode ter sua própria "sintaxe", que é traduzida para uma linguagem universal, o formato XML ou JSON.

Neste trabalho, o Web service pode ser acessado a partir de dois serviços implementados. Esses serviços são acessados pela internet no protocolo HTTP ao qual é transmitido um arquivo XML por ele.

A seguir segue os dois serviços implementados.

4.1 Serviços implementados

4.1.1 Storage

Envio do XML para persistir no banco.

O recurso implementado utiliza do comando POST e pode ser acessado a partir da seguinte URL: `http://www.URL_BASE:8080/contextserver/storage`, onde URL_BASE é a url do servidor onde o web service está hospedado. Juntamente com o acesso seguem alguns parâmetros:

- Parâmetros:
 - Cabeçalho: Content-Type: application-xml

- Body: No corpo é passado o xml da figura 3.5.

- Retorno

- 200 OK: resposta do web service em que o xml foi entregue.
- 500 INTERNAL_ERRO_SERVER: resposta do web service alegando um erro interno. Provavelmente proveniente da falta de parâmetros contido no xml.

4.1.2 Consumer

Envio do xml para consultar o banco e resposta em um arquivo xml contendo as mídias pesquisadas.

O recurso implementado utiliza do comando POST e pode ser acessado a partir da seguinte URL: `http://www.URL_BASE:8080/contextserver/consumer`, onde URL_BASE é a url do servidor onde o web service está hospedado. Juntamente com o acesso seguem alguns parâmetros:

- Parâmetros:

- Cabeçalho: Content-Type: application-xml
- Body: No corpo é passado o xml da figura 3.6.

- Retorno

- 200 OK: resposta do web service em que o xml foi entregue e no corpo da resposta o xml com as mídias pesquisadas conforme a figura 4.1.
- 500 INTERNAL_ERRO_SERVER: resposta do web service alegando um erro interno. Provavelmente proveniente da falta de parâmetros contido no xml.

4.2 Arquitetura orientada a eventos

A programação orientada a eventos é um paradigma de programação em que a estrutura e implementação de programas são determinadas pelos sucessos que ocorrem no sistema, definido pelo usuário ou que eles mesmos causam. Para entender a programação orientada

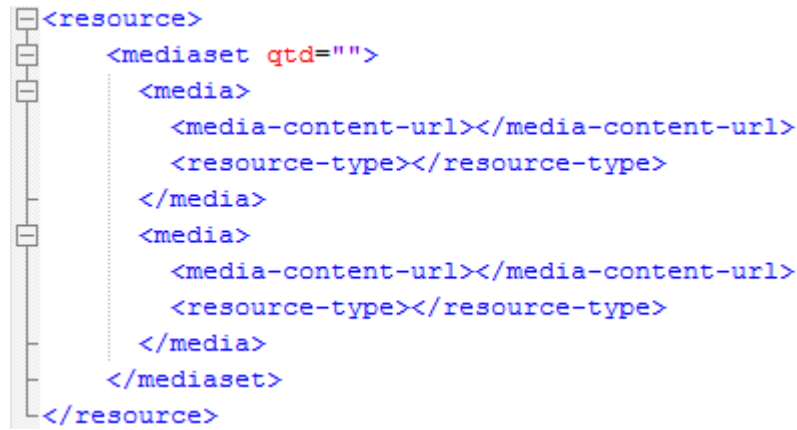


Figura 4.1: Exmplo de xml com as mídias pesquisadas

a eventos, nós colocamos a ela o que ela não é: enquanto na programação sequencial ou estruturada é o programador que define o que será o fluxo do programa, na programação orientada a eventos é o usuário ou o que está acionando o programa o que dirige o fluxo do programa. Embora a programação sequencial possa ter um programa de intervenção externa, estas intervenções ocorrerão quando o programador tenha determinado, e não, em qualquer momento, como no caso da programação orientada a eventos. A programação orientada a eventos é a base do que chamamos de interface do usuário, mas também pode ser usado para desenvolver interfaces entre os componentes do software ou do núcleo.

A arquitetura baseada em eventos é implementada pelo *EventBus*. Ele gerencia toda a comunicação entre componentes, controla as transações e sincronizações entre os módulos.

4.2.1 *EventBus*

EventBus permite a comunicação no estilo *publish-subscribers*. Neste padrão o bus atua como meio de transporte sem a necessidade de que os componentes conheçam entre si. Ele transporta eventos postados a seus respectivos *subscribers*. A figura 4.2 mostra as interações dos módulos e do bus.

Na figura 4.2, o módulo *Security* não utiliza o *Bus*, pois ele não foi implementado, mas é considerado que ele exista, sendo uma possível implementação futura.

O módulo *Outros* representa os módulos *Media Analyser*, *Context Inference* e *Context Composer* que não foram implementados neste trabalho mas podem ser facilmente

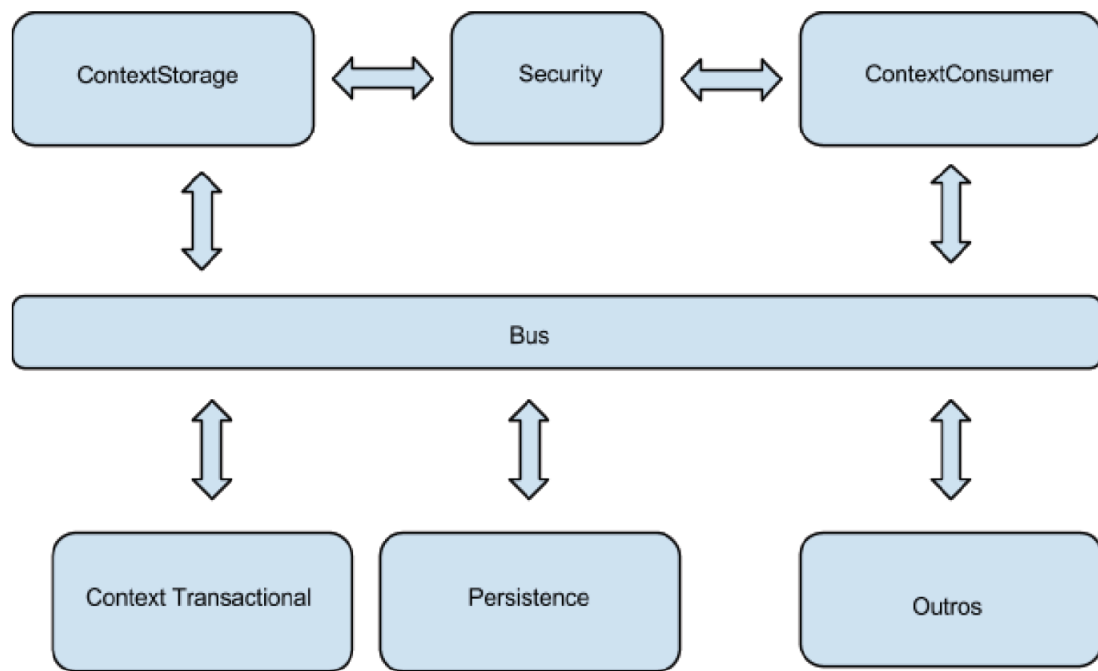


Figura 4.2: Interação entre o bus e os módulos da arquitetura.

inseridos na arquitetura.

5 Protótipo

O estudo de caso desta monografia é a construção de um webservice que implemente a arquitetura proposta para armazenar e disponibilizar os contextos.

O webservice foi implementado utilizando *Spring Framework*. O Spring é um framework *open source* para a plataforma Java. Trata-se de um *framework* não intrusivo, baseado nos padrões de projeto **inversão de controle** (IoC) e **injeção de dependência**. No Spring o *container* se encarrega de "instanciar" **classes** de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework, o que é chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Dessa forma o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

O Spring possui uma arquitetura baseada em **interfaces** e **POJOs** (*Plain Old Java Objects*), oferecendo aos POJOs características como mecanismos de segurança e controle de transações. Com Spring, pode-se ter um alto desempenho da aplicação.

Também foi utilizado um servidor de web Java, o Tomcat que é um container de *servlets*.

A tecnologia a ser utilizada na troca de informações por evento foi a biblioteca *guava-libraires*. É uma API do google que fornece comunicação baseada em eventos, o *EventBus*.

5.1 Estudo do Armazenamento em um SGBD

Uma solução inicial e mais simples é o armazenamento em sistemas de arquivos. Essa opção é bastante suscetível a falhas, pois não existe controle de acesso e dos processos que têm permissão para manipular dados. Uma outra solução é o armazenamento de imagens em um SGBD - Sistema de Gerenciamento de Banco de Dados, que já tem seu uso consolidado no armazenamento de dados não complexos, os dados textuais.

O SGBD é o responsável pela manutenção e gerenciamento de uma base de dados.

Ele é o responsável pela integridade, pelo controle de acesso e a manipulação de dados.

Atualmente, no mercado existem diversas soluções na área de gerenciamento de banco de dados, e o escolhido para este trabalho foi o MySQL.

Para simplificar o desenvolvimento do protótipo, foi definida uma tabela única para armazenar os dados. A estrutura escolhida conforme tabela 5.1 e exemplo de dados conforme tabela 5.2:

Tabela 5.1: Tabela com campos persistidos no banco

Campo	Descrição
media_content_url	Url do local onde a mídia está armazenada.
resource_type	Tipo do recurso.
user_type	Tipo de id do usuário.
user_id	Id de login do usuário.
role	Modificador de acesso do recurso.
created_at	Valor em timestamp do momento em que o recurso foi capturado.
synced_at	Valor em timestamp do momento em que foi sincronizado com o servidor de contexto.
latitude	Valor do atributo latitude do GPS.
longitude	Valor do atributo longitude do GPS.
resolution	Valor em string da resolução da tela do dispositivo.

Tabela 5.2: Tabela com exemplo de dados fictícios persistidos no banco

Campo	Valor
media_content_url	www.instagram.com/usuariofoto1
resource_type	png
user_type	instagram
user_id	instagram/usuario
role	privado
created_at	1230192381
synced_at	1260490984
latitude	33.0034234
longitude	34.5634533
resolution	800x600

5.2 Acesso aos serviços do servidor

Em testes locais, ao acessar o endereço `http://localhost:8080/contextserver` é exibido uma página HTML indicando que o servidor está em funcionamento, conforme figura 5.1.

Para os testes de envio do xml foi utilizado um aplicativo do google, o **Advanced REST client** conforme figura 5.2.

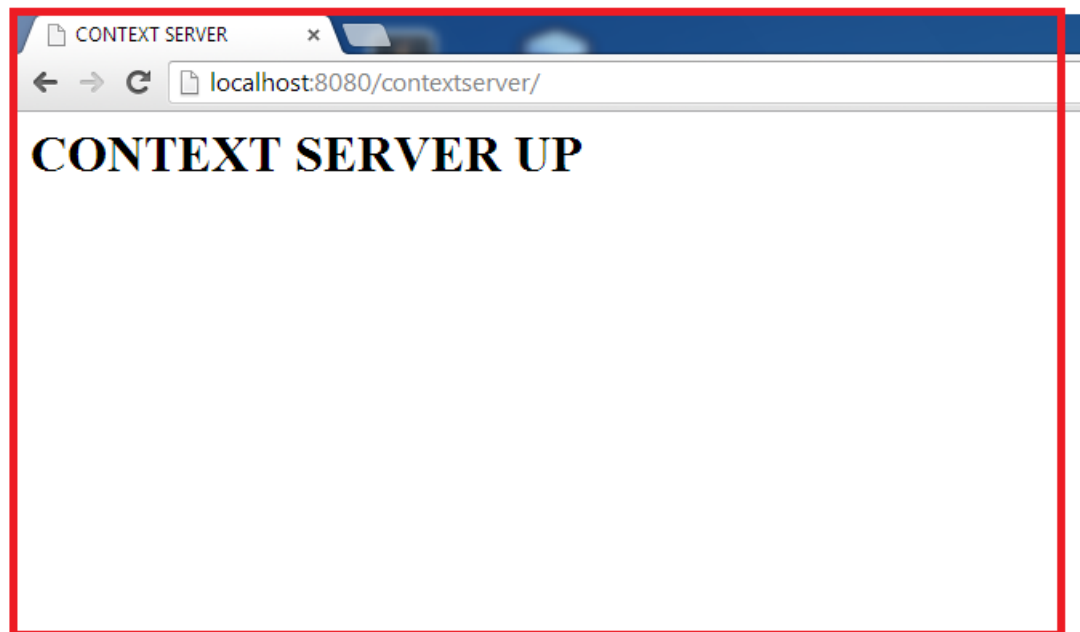


Figura 5.1: Página do server funcionando.

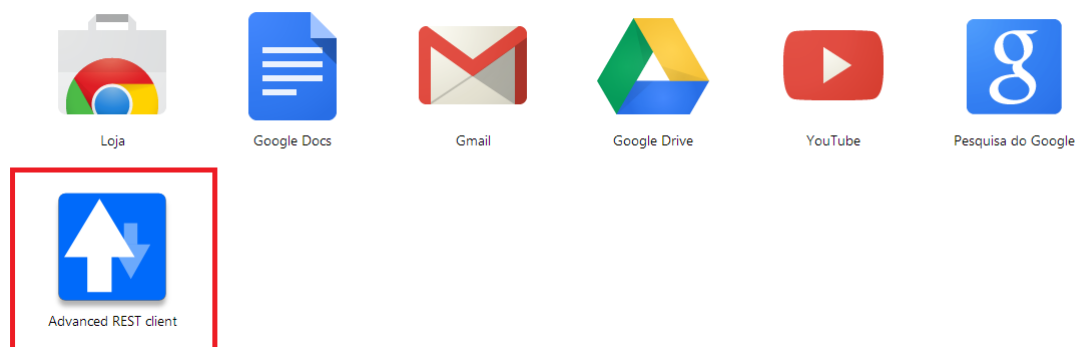


Figura 5.2: Advanced REST client

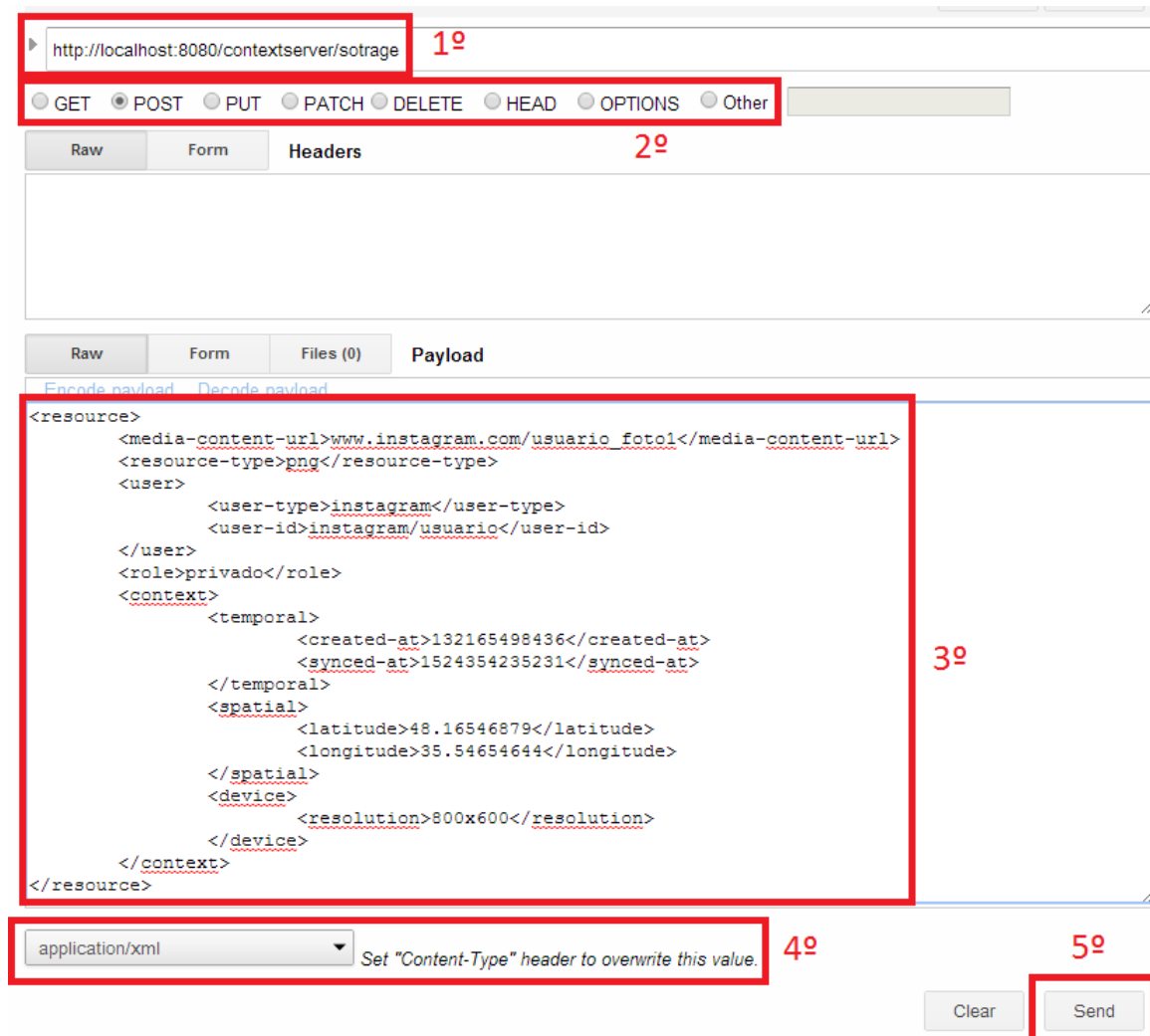


Figura 5.3: Envio de xml para Storage

5.3 Envio de xml para Storage

Enviando o xml para Storage, foi configurado o Advanced Rest client conforme figura 5.3 nos seguintes passos:

1. Insere o caminho da localização do servidor bem como o serviço a ser utilizado.
2. Escolhe envio como POST.
3. No corpo do documento enviar o XML com os dados de contexto.
4. Selecionar Content-Type como `application/xml`.
5. Enviar XML.

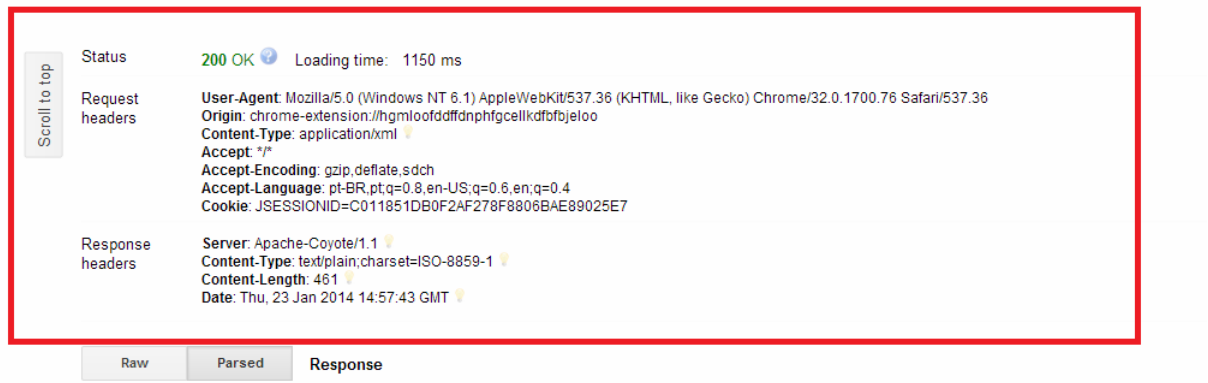


Figura 5.4: Resposta do servidor

SELECT *
FROM `resource`
LIMIT 0, 30

Mostrar : Linha inicial: 0 Número de linhas: 30 Cabeçalho a cada 100 linhas

Ordenar por chave: Nenhum

	id	media_context_url	resource_type	user_type	user_id	role	created_at	synced_at	latitude	longitude	resolution
☐ Editar ☐ Copiar ☐ Apagar	1	www.facebook.com/usuario	png	facebook	facebook/usuario	publico	1970-01-16 22:49:10	1970-01-16 22:49:10	30.648	45.987	800x400
☐ Editar ☐ Copiar ☐ Apagar	2	www.facebook.com/usuario	png	facebook	facebook/usuario	publico	1970-01-16 22:49:10	1970-01-16 22:49:10	30.648	45.987	800x400
☐ Editar ☐ Copiar ☐ Apagar	3	www.facebook.com/usuario	png	facebook	facebook/usuario	publico	1970-01-16 22:49:13	1970-01-16 22:49:13	0.648	1.987	800x400
☐ Editar ☐ Copiar ☐ Apagar	4	www.facebook.com/usuario	png	facebook	facebook/usuario	publico	0000-00-00 00:00:00	1970-01-16 22:49:06	0.648	1.987	800x400
☐ Editar ☐ Copiar ☐ Apagar	5	www.twitter.com/usuario	jpg	twitter	@usuario	privado	1970-01-16 22:49:36	1970-01-16 23:05:46	55.6548	35.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	6	www.instagram.com/usuario	jpg	instagram	instagram/usuario	privado	1970-01-18 02:36:16	1970-01-18 02:52:26	51.6548	40.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	7	www.instagram.com/usuario	jpg	instagram	instagram/usuario	privado	1970-01-18 02:36:16	1970-01-18 02:52:26	51.6548	40.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	8	www.instagram.com/usuario	jpg	instagram	instagram/usuario	privado	1970-01-18 02:36:16	1970-01-18 02:52:26	51.6548	40.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	9	www.instagram.com/usuario	jpg	instagram	instagram/usuario	privado	1970-01-18 02:36:16	1970-01-18 02:52:26	51.6548	40.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	10	www.instagram.com/usuario	jpg	instagram	instagram/usuario	privado	1970-01-18 02:36:16	1970-01-18 02:52:26	51.6548	40.3525	1024x980
☐ Editar ☐ Copiar ☐ Apagar	11	www.instagram.com/usuario	png	instagram	instagram/usuario	privado	1970-01-16 23:14:48	1970-01-16 23:14:48	21.7517	43.3528	1024x768

Figura 5.5: Contexto persistido no banco

Teremos como resposta vindo do servird conforme figura 5.4 e persistência no banco conforme figura 5.5.

5.4 Envio de xml para Consumer

Enviando o xml para Consumer, foi configurado o Advanced Rest client conforme figura 5.6 nos seguintes passos:

1. Insere o caminho da localização do servidor bem como o serviço a ser utilizado.
2. Escolhe envio como POST.
3. No corpo do documento enviar o XML com os dados de contexto.

The image shows a REST client interface with the following components:

- 1º** URL bar: `http://localhost:8080/contextserver/consumer`
- 2º** Method selection: `POST` is selected.
- 3º** Payload area: Contains an XML document.

```
<resource>
  <user>
    <user-type></user-type>
    <user-id></user-id>
  </user>
  <role>publico</role>
  <context>
    <temporal>
      <created-at_min></created-at_min>
      <created-at_max></created-at_max>
    </temporal>
    <spatial>
      <latitude_min></latitude_min>
      <latitude_max></latitude_max>
      <longitude_min></longitude_min>
      <longitude_max></longitude_max>
    </spatial>
    <device>
      <resolution_min></resolution_min>
      <resolution_max></resolution_max>
    </device>
  </context>
</resource>
```
- 4º** Content-Type dropdown: `application/xml`
- 5º** `Send` button

Figura 5.6: Envío de xml para Consumer

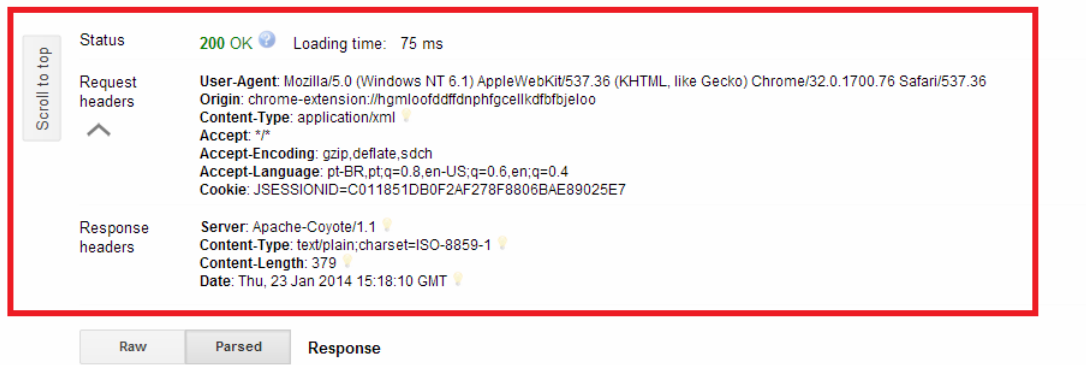


Figura 5.7: Resposta do servidor



Figura 5.8: Xml de reposta da consulta feita

4. Selecionar Content-Type como application/xml.
5. Enviar XML.

Teremos como resposta vindo do servir conforme figura 5.7 e um xml de reposta conforme figura 5.8.

6 Conclusão

Com a constante evolução dos dispositivos móveis, os usuários têm gerado várias mídias a partir deles. Essas mídias contém contextos que trazem informações referentes a elas. Armazenar esses contextos e disponibilizados de maneira correta é um novo desafio a ser feito.

Este trabalho tem o desafio de apresentar e desenvolver uma arquitetura para que esses contextos possam ser armazenados e possam estar disponíveis para diversas aplicações. A arquitetura desenvolvida é baseada em eventos que coordena a comunicação entre os módulos internos que a compõe. Uma das vantagens do uso de arquiteturas orientadas a eventos é o desacoplamento permitido por esta arquitetura, permitindo que seus módulos seja altamente customizáveis.

Apesar do desenvolvimento da arquitetura cumprir o simples objetivo da mesma, existem outros objetivos em aberto que podem ser abordados em trabalhos futuros como por exemplo o desenvolvimento de um módulo de segurança que implemente alguns recursos como autenticação, obrigando assim que as aplicações que utilizarem da arquitetura estejam obrigatoriamente autenticadas no servidor para envio ou requisição de qualquer informação. Um exemplo disto é o OAuth2 que é um mecanismo de autenticação cliente/servidor utilizado em redes sociais como Facebook e Google Plus. Com relação a transferência de dados podem ser adicionados o uso de protocolos que criptografam a conexão, como por exemplo o HTTPS que utiliza o SSL/TLS.

Também como trabalhos futuros, temos o desenvolvimento dos restantes módulos que compõe a arquitetura proposta, como o *Media analyser* que é responsável por extrair informações de contextos dos recursos, o *Context inference* que é responsável por descobrir outros contextos e o *Context composer*, responsável por transformar informações de contextos básicas em informações que agregam valor ao recurso.

Referências Bibliográficas

- Barbosa, M. H. Contextf: framework de apoio a construção de aplicações sensíveis ao contexto. 2013.
- Braga, R. B.; Martin, H.; Viana, Windson, W. ; Andrade, R. M. C. Captain: geração de diários de bordo digitais usando anotação contextual e conteúdo multimídia. 2011.
- Braga, R.; Moraes Medeiros da Costa, S.; Carvalho, W.; Castro Andrade, R. ; Martin, H. **A context-aware web content generator based on personal tracking**. In: Martino, S.; Peron, A. ; Tezuka, T., editors, Web and Wireless Geographical Information Systems, volume 7236 de **Lecture Notes in Computer Science**, p. 134–150. Springer Berlin Heidelberg, 2012.
- Feng, Y.; Lapata, M. **Automatic image annotation using auxiliary text information**. In: in ACL HLT, 2008.
- Freitas, C. d. Uma arquitetura baseada em padrões abertos para visualização científica via internet aplicada à medicina. 2002.
- Hironobu, Y. M.; Takahashi, H. ; Oka, R. **Image-to-word transformation based on dividing and vector quantizing images with words**. In: in Boltzmann machines, Neural Networks, p. 405409, 1999.
- Lacerda, Y. A.; de Figueirêdo, H. F.; de Souza Baptista, C. ; de Paiva, A. C. **Expandido e utilizando informações de contexto para a sugestão de anotações de fotografias digitais**. In: Proceedings of the 14th Brazilian Symposium on Multimedia and the Web, WebMedia '08, p. 162–169, New York, NY, USA, 2008. ACM.
- Lavrenko, V.; Manmatha, R. ; Jeon, J. **A model for learning the semantics of pictures**. In: IN NIPS. MIT Press, 2003.
- Lavrenko, V.; Manmatha, R. ; Jeon, J. **A model for learning the semantics of pictures**. In: IN NIPS. MIT Press, 2003.
- Naaman, M.; Song, Y. J.; Paepcke, A. ; Molina, H. G. **Automatic organization for digital photographs with geographic coordinates**. In: Fourth ACM/IEEE Joint Conference on Digital Libraries (JCDL 2004), June 2004.
- Smeulders, A. W. M.; Member, S.; Worring, M.; Santini, S.; Gupta, A. ; Jain, R. Content-based image retrieval at the end of the early years. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v.22, p. 1349–1380, 2000.
- Vailaya, A.; Figueiredo, M. A. T.; Jain, A. K. ; Zhang, H. Image classification for content-based indexing. **IEEE Transactions on Image Processing**, p. 117–130, 2001.
- Viana, W.; Miron, A. D.; Moisuc, B.; Gensel, J.; Villanova-Oliver, M. ; Martin, H. Towards the semantic and context-aware management of mobile multimedia. **Multimedia Tools Appl.**, v.53, n.2, p. 391–429, jun 2011.
- Wagenaar, W. A. My memory: A study of autobiographical memory over six years. **Cognitive Psychology**, v.18, n.2, p. 225 – 252, 1986.