Ministério do Planejamento, Orçamento e Gestão
Secretaria de Logística e Tecnologia da Informação

Arquitetura Técnica Referencial para Abertura de Dados

Piloto de Dados Abertos SICAF

Departamento de Integração de Sistemas e Informação

2011

Brasília, Dezembro de 2011
versão 1.0

Sumário

Histórico de Alterações

Data	Versão	Histórico	Responsável(is)
10/08/2011	0.1	Versão inicial	Nitai Silva, Augusto Herrmann e Christian Miranda - SLTI / Time de Dados Abertos

1 Introdução

1.1 Objetivo do documento

A Secretaria de Logística e Tecnologia da Informação do Ministério do Planejamento – SLTI, tem dentro de suas atribuições a competência de planejar, coordenar, supervisionar e orientar, normativamente, as atividades do Sistema de Administração de Recursos de Informação e Informática – SISP, propondo políticas e diretrizes de Tecnologia da Informação, no âmbito da Administração Pública Federal direta, autárquica e fundacional.

Este documento tem por objetivo servir como referência para implementação de soluções para disponibilização de dados na Internet seguindo os princípios de dados abertos.

Em março de 2011 a SLTI realizou a implementação de um projeto piloto de Dados Abertos com os dados do Sistema de Cadastramento Unificado de Fornecedores – SICAF. Este documento fornece detalhes técnicos quanto ao desenvolvimento da aplicação. Resumidamente são respondidos questionamentos sobre a metodologia de desenvolvimento utilizada e o processo de abertura numa visão de gerência de projeto.

O conjunto de ferramentas e bibliotecas utilizadas no projeto constitui uma solução derivada das tecnologias escolhidas no início, porém é fundamental esclarecer que outras combinações de linguagens e bibliotecas podem proporcionar o mesmo, ou melhor, resultado. Portanto, o caráter arquitetural, ou seja, padrões de projetos, padrões e formatos interoperáveis e as boas práticas de desenvolvimento, têm maior valor orientativo para as fábricas de software que desejam implementar soluções semelhantes de abertura de dados. Utilize este documento como referência para sua solução de abertura de dados. É importante salientar que o conceito de dados abertos está fundamentado em princípios simples, e que para certos casos, dados estáticos por exemplo, uma solução bem mais simplificada pode alcançar um ótimo nível de qualidade.

Entendemos que dados abertos não é apenas transparência e controle dos gastos públicos, mas também uma plataforma onde a sociedade desenvolve e compartilha serviços. Queremos fugir do usual, aguardamos o retorno da sociedade e temos a certeza que seremos surpreendidos.

1.2 Objetivo do projeto piloto

A escolha de abrir os dados do SICAF se deve a dois motivos: primeiramente, é dever do Ministério do Planejamento dar o exemplo e implementar dados abertos em seus sistemas, o SICAF é gerido pela Diretoria de Logística e Serviços Gerais - DLSG que também faz parte da SLTI. Segundo, enxergamos que o cadastro de fornecedores é um conjunto muito rico de dados e queremos potencializar seu uso e agregar mais valor disponibilizando-o para sociedade.

A aplicação é *RESTful*, ou seja, constitui uma API disponível na Internet para acesso aos dados em vários formatos abertos através de requisições HTTP seguindo os princípios REST.

Usamos o termo piloto para esclarecermos que é um projeto experimental, no sentido de que estamos testando as tecnologias, desenhando processos, acumulando boas práticas, ou seja, estamos aprendendo como abrir dados. Toda essa experiência fundamenta várias de nossas decisões na Infraestrutura Nacional de Dados Abertos (INDA).

1.3 Realização

O desenvolvimento do presente projeto foi realizado pela SLTI. A gestão do projeto e a implementação da aplicação foi feita pelo Time de Dados Abertos do Departamento de Integração de Sistemas e Informação (DSI). A logística para acesso aos dados em produção e o entendimento aprofundado sobre os conceitos em torno do SICAF foram possibilitados pelo Departamento de Logística e Serviços Gerais, o qual é responsável pelo SICAF.

1.4 Escopo do projeto

Numa perspectiva de dados abertos, o escopo se resume à disponibilização dos dados públicos do SICAF na Internet sob uma licença pertinente. Dos vários tipos de dados que constituem o SICAF, definiu-se dois conjuntos mais importantes, dados das Unidades Cadastradoras e dados dos Fornecedores. Dados geográficos relacionados a esses dois conjuntos também foram disponibilizados.

No caráter técnico, os dados estão disponíveis na Internet seguindo os princípios da *Representational State Transfer* - REST, ou seja, documentos em hipermídia acessíveis através de requisições HTTP. A arquitetura possibilita a combinação de diversos filtros nas requisições e a serialização em vários formatos. A serialização é feita em tempo real, pois os dados são atualizados com boa frequência. Primando pela qualidade máxima de abertura de dados, foi definido um rascunho de vocabulário, ou ontologia mínima, cobrindo os principais conceitos do SICAF.

A título de exemplificação, também foi desenvolvida uma aplicação (*mashup*) fazendo uso dos dados disponibilizados pela aplicação.

1.4.1 O SICAF em produção (Serpro)

O SICAF, assim como outros subsistemas do Comprasnet, é desenvolvido e mantido pelo Serpro. No capítulo 3 é apresentada a arquitetura do sistema desenvolvido neste projeto, claramente é perceptível a integração com a aplicação em produção a nível de banco de dados. O aplicação de produção do SICAF utiliza uma base de dados PostgreSQL.

O SICAF sofreu diversas melhorias no início de 2011. O processo de cadastramento e habilitação de fornecedores foi simplificado, e faz maior uso dos meios eletrônicos. O sistema pode ser acessado através da URL https://www3.comprasnet.gov.br/SICAFWeb.

2 Metodologia de desenvolvimento

2.1 Recursos

A SLTI possui um time dedicado ao desenvolvimento de ações focadas em Dados Abertos, que foi responsável pela gestão e execução deste projeto. Com caráter experimental, um dos principais objetivos para realização deste projeto foi absorver e relatar o aprendizado no desenvolvimento do piloto. Para isso o desenvolvimento foi todo realizado nas dependências do Ministério do Planejamento.

A infraestrutura para o desenvolvimento e manutenção atual da ferramenta é uma instância do servidor virtualizado no Labcluster, como parte integrante do projeto Software Público. Todos os componentes utilizados são softwares livres. O sucesso do projeto confirma o potencial de desenvolvimento de soluções de dados abertos utilizando estritamente software livre.

2.2 Desenvolvimento Ágil

O conjunto de requisitos e tecnologias envolvidas caracteriza o projeto como ambicioso e inovador. Juntando um cronograma com prazos apertados à falta de modelos específicos de implementação de dados abertos, decidimos fazer uso de uma metodologia adequada afim de minimizar o risco e maximizar o valor agregado ao software, decidimos utilizar SCRUM. A Ilustração 1 resume as etapas da utilização de SCRUM.

Ilustração 1: Iterações e produtos no desenvolvimento ágil SCRUM

Um dos princípios norteadores da metodologia ágil é a de que o cliente faz parte do time, ele está comprometido com o sucesso do projeto, o que demanda fortes interações em ciclos curtos. No paradigma de dados abertos, a plataforma de dados provida pelo governo é para uso da sociedade, mas defendemos que o próprio governo deve fazer uso desses dados, enxergando esta como uma plataforma de interoperabilidade intragoverno. Nesse sentido gestores do Departamento de Logística e Serviços Gerais – DLSG – participaram intensamente no refinamento da aplicação de acordo com as cobranças por dados percebidas.

É importante pontuar que a configuração destes recursos com a metodologia ágil possibilitou o desenvolvimento e implantação da aplicação em pouco mais de 2 meses.

2.3 Gestão do processo de abertura

Apesar de não ser objetivo principal do projeto piloto, é atribuição fundamental da SLTI orientar e apoiar os órgãos do SISP na gestão, com caráter estratégico, do processo de abertura dos dados.

Um dos primeiros produtos desenvolvidos pelo time de dados abertos da SLTI, foi um Guia de Abertura de Dados, o qual está em consulta pública no portal de Governo Eletrônico [https://www.consultas.governoeletronico.gov.br/ConsultasPublicas/consultas.do?acao=exibir&id=93]. O guia é um manual destinado aos gestores que querem iniciar o processo de abertura de dados em suas instituições. Nele são elencados diversos atores envolvidos no processo de abertura, desde o responsável máximo da instituição, o qual tem poder de autorização legal de abertura dos dados, até o envolvimento com a sociedade, considerado no paradigma de dados abertos o principal beneficiado. O manual também define um macroprocesso para orquestração da abertura, sempre focando no que é mais importante e agregador ao objetivo. Também são citadas etapas importantes como processo de anonimização e de classificação das informações.

O manual passou por um processo de amadurecimento, com o objetivo de aprofundar os detalhes de planejamento e operacionalização da abertura de dados, fornecendo orientação para que as áreas de TI nos órgãos possam promover essa ação como estratégica em suas instituições. Quando do encaminhamento para consulta pública, em 1º de setembro de 2011, ele passou a se chamar Guia de Abertura de Dados.

O Guia de Abertura de Dados é um produto previsto do GT1 da INDA. Para aqueles que necessitam de apoio e orientação mais dedicada o time de dados abertos da SLTI pode ser solicitada pelo endereço http://c3s.sisp.gov.br/ ou pelo e-mail sisp@planejamento.gov.br.

2.4 Licença dos dados

O projeto busca um alinhamento com todos os princípios de dados abertos, onde um deles sugere que o dado deve ser livre de regulação, patente, marca, e tudo que restrinja seu reuso. Reconhecemos que ainda não foram desenvolvidas licenças específicas de dados abertos ideais, adaptadas ao contexto do governo brasileiro. Dessa forma, decidimos utilizar as Licenças de Banco de Dados Aberto (ODbL) e de Conteúdo de Banco de Dados Aberto (DbCL). Essas são licenças livres, escritas especificamente para dados. Nessa direção, dentro da Infraestrutura Nacional de Dados Abertos, que está sendo construída, está prevista a implementação de uma licença de dados abertos governamentais brasileira.

A ODbL é uma licença aberta, segundo os critérios da definição de conhecimento aberto - http://opendefinition.org. Isto é importante para garantir a interoperabilidade de licenças com outros dados abertos, quando o usuário dos dados, seja a sociedade ou o próprio governo, for fazer cruzamento entre esses dados e outros.

Bases de dados e bancos de dados têm algumas características específicas que as diferenciam de conteúdo, e por isso é necessário utilizar licenças que são específicas para utilização em bases de dados. Assim como, há alguns anos, considerando as diferenças de características entre software e conteúdo, desenvolveu-se a licença Creative Commons especificamente para licenciar conteúdo de forma livre (considerando-se o sentido de "livre" em espírito semelhante ao que a licença GPL faz para tornar o software livre).

A DBCL foi utilizada para o conteúdo dos dados pois, segundo o OpenDataCommons FAQ [2], direitos diferentes podem ser aplicáveis à base/conjunto de dados como um todo e ao conteúdo dos dados. (Trecho da nossa reportagem para Wireless Mundi).

2.5 Definição da Interface de dados

O principal objetivo do projeto é disponibilizar o acesso aos dados através da Internet. Isso fica muito claro ao analisar a arquitetura da aplicação no Capítulo 3, onde a camada superior é uma interface programável de acesso à dados abertos. Essa característica, inerente às arquiteturas de softwares baseados na nuvem, provoca um desenvolvimento orientado à dados, mais conhecido como *Data Driven Development* (DDA). Em outras palavras, os recursos disponíveis na interface de dados foram desenhados pensando-se nas dependências de dados na perspectiva de quem irá consumir os dados.

Os métodos para acesso aos dados estão distribuídos em 3 grupos, como na Ilustração 2. O grupo principal é o Consulta ao Cadastro, onde é possível combinar filtros e acessar dados das Unidades Cadastradoras e dados dos Fornecedores. O grupo de Informações Básicas possui métodos para consulta dos dados necessários para possibilitar a construção dos filtros nas Consultas ao Cadastro. Por exemplo, nas informações básicas estão os códigos para filtrar os fornecedores de um município específico ou de uma linha de fornecimento específica. No grupo de Informações Detalhadas do Fornecedor, da Unidade Cadastradora ou de Município.

Os detalhes de cada método, como os campos de retorno, foram definidos de acordo com o propósito do método. Vários dados de Fornecedor só podem ser acessados pelo método no grupo de Informações Detalhadas, assim nas Consultas ao Cadastro estão apenas dados básicos, reduzindo-se o tamanho dos arquivos transmitidos. A diversidade de formatos para cada consulta possibilita que o usuário utilize o de menor tamanho, o mais expressivo semanticamente ou o que se adapta melhor ao seu sistema. Todos os métodos disponíveis estão listados na URL http://api.comprasnet.gov.br/sicaf/doc/metodos.html.

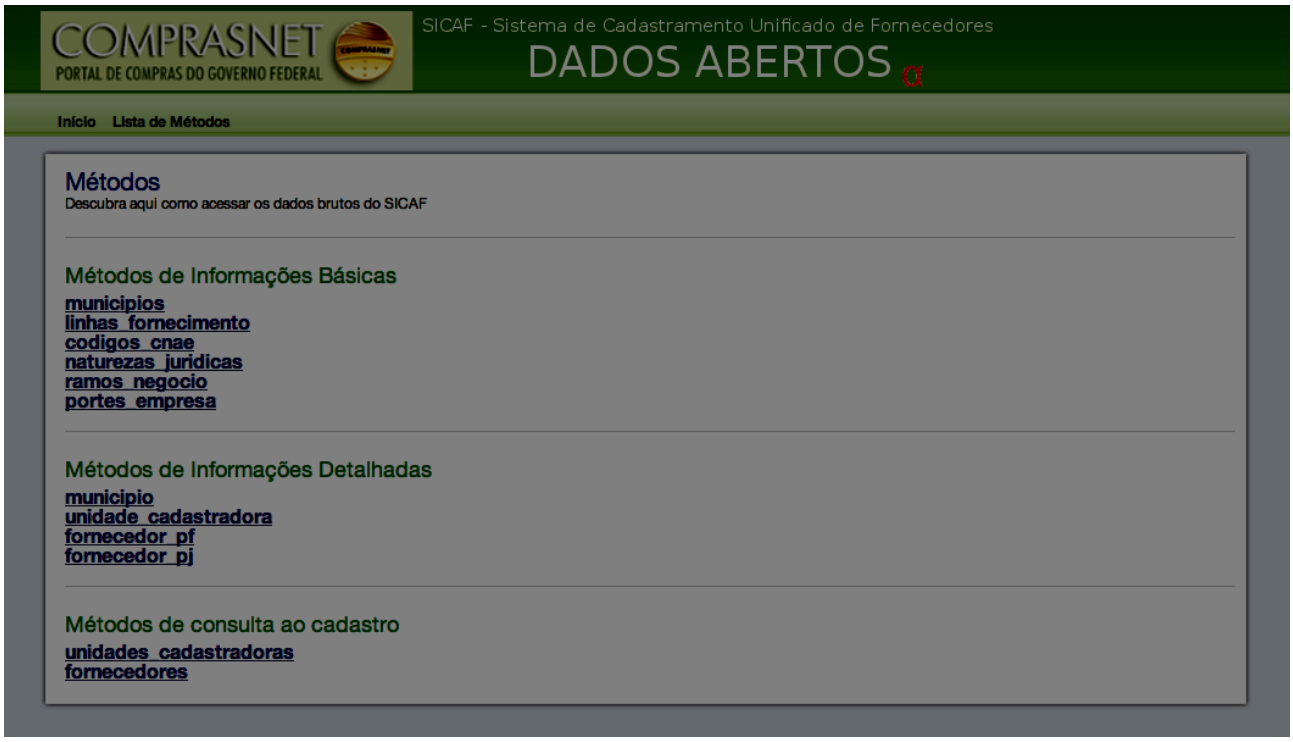


Ilustração 2: Documentação da API listando todos os métodos disponíveis para consulta.

2.6 Documentação

Uma documentação de qualidade e completa é extremamente importante. É na documentação

que o usuário da API encontrará os detalhes técnicos sobre os meios possíveis de acessarem os dados. Mais detalhes garantem maior adesão à plataforma de dados. Na documentação é preciso que se esclareça toda e qualquer informação que possa influenciar a implementação da solução que fará uso dos dados. Sugerimos que a documentação forneça detalhes sobre:

- Lista de métodos, recursos, tipos ou classes dos grupos de dados.
- Descrição textual esclarecendo o significado de cada método, recurso, tipo ou classe de dados.
- Lista de campos que compõe o resultado de consulta de cada método, recurso, tipo ou classe de dado.
- Lista de parâmetros e valores, se houver, que possibilitam realizar filtros sobre o método, recurso, tipo ou classe de dados.
- Lista dos diferentes formatos para acesso aos dados.
- Informações sobre frequência de atualização dos dados.
- Exemplo de consulta para cada método, recurso, tipo ou classe de dados.
- Licença, se houver, sob a qual os dados estão publicados.

A documentação da API do SICAF está disponível em http://api.comprasnet.gov.br/sicaf/doc/ trazendo informações gerais da plataforma. Para cada método ou recurso da API existe uma documentação específica, por exemplo em http://api.comprasnet.gov.br/sicaf/doc/fornecedores.html.

2.7 Construção de metadados (RDF)

Foram realizadas uma série de reuniões com pessoas ligadas à área de negócio correspondente na DLSG, nas quais foram coletadas informações que nos levaram a esboçar um vocabulário mínimo para o SICAF, que representasse os dados que seriam disponibilizados. Nesse processo, foi tomado o cuidado de ligar as classes e propriedades modeladas a outros conceitos externos definidos em vocabulários amplamente utilizados na comunidade internacional da web semântica.

2.8 ???Sequência de desenvolvimento????

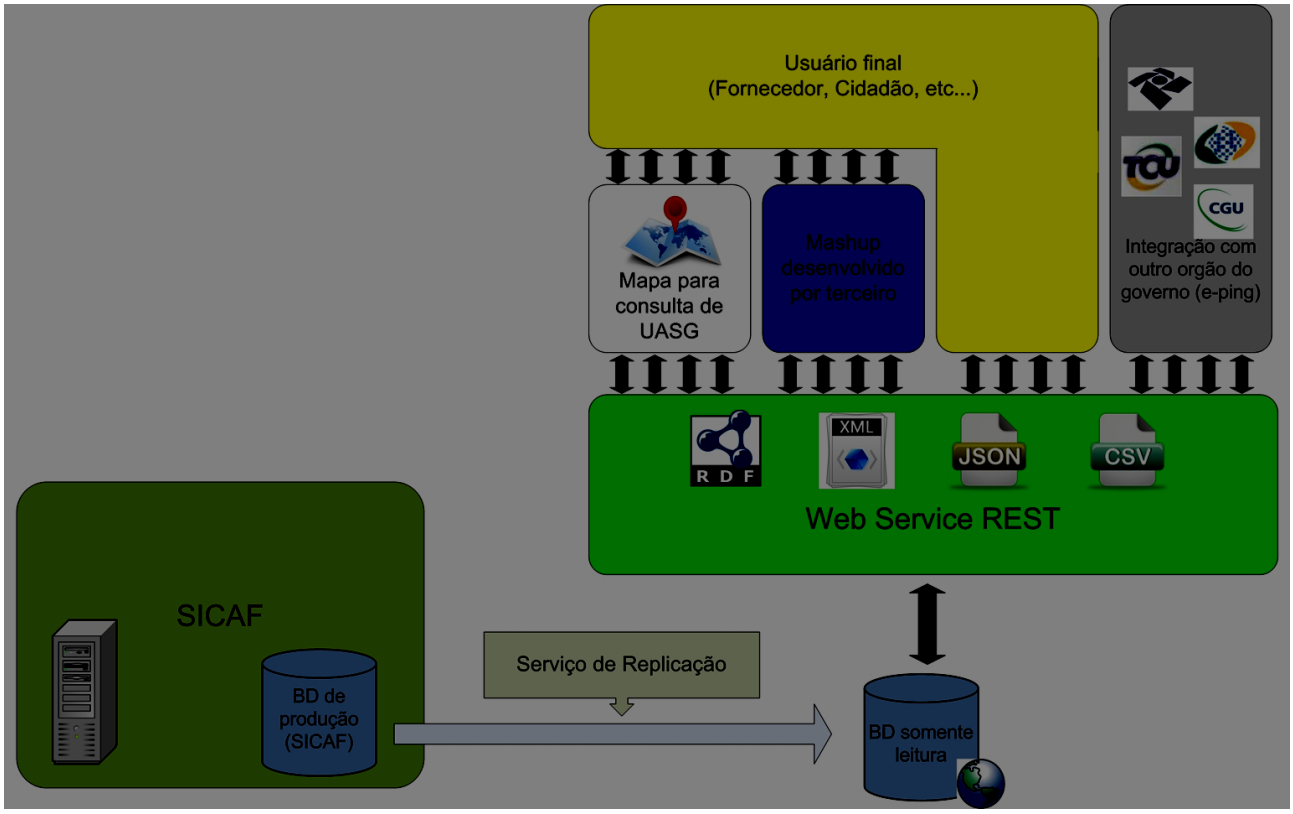
[O que deveria estar aqui?]

3 O Sistema

3.1 Arquitetura do sistema

O sistema Dados Abertos SICAF é um serviço de acesso a dados disponível na Internet. A imagem a seguir descreve o sistema interagindo com outros componentes, o SICAF em produção que mantém os dados originais e vários aplicativos que, por sua vez, fazem uso dos dados abertos, cruzando-os com outros dados e gerando novas visualizações e serviços.

O projeto do sistema segue os princípios REST, fazendo uso extenso da arquitetura Web, o que possibilita a incorporação de diversas tecnologias beneficiadas do amadurecimento dos servidores web de documentos e conteúdos. Exemplos desses benefícios são cache no servidor web, compactação dos arquivos trocados de forma transparante, dentre outros.



3.1.1 Funcionamento independente do sistema de produção

O sistema Dados Abertos SICAF foi projetado para funcionamento independente do sistema de produção SICAFWeb, como visto na Ilustração 3. Apesar de ambos estarem referenciados pelo mesmo domínio principal, comprasnet.gov.br, cada um é mantido em ambiente isolado e independente. Este desacoplamento traz uma série de benefícios: o Dados Abertos SICAF possui base de dados própria, o que possibilita estratégias de *tunning* específicas para as consultas disponibilizadas, a incorporação de dados geográficos persistentes, e evita que o excesso de consultas aos dados abertos afete o SICAFWeb em produção. A integração entre as bases para atualização dos dados abertos é feita por um módulo de replicação que atua diretamente no banco de dados, onde é executada a seleção dos dados, detalhado na seção 17.

A separação possibilita que os serviços executem em ambientes de infraestrutura independentes. Com essa separação regras de segurança podem ser simplificadas no sistema de dados

abertos, uma vez que a proposta é fornecer acesso irrestrito aos dados desta base.

3.1.2 Arquitetura da Web

No desenvolvimento da API, foram seguidos os princípios de Representational State Transfer – REST, defendidos na tese de doutorado de Roy Fielding, compreendem as características essenciais da web que a tornaram uma plataforma de sucesso. O estilo REST de arquitetura para serviços web utiliza os mesmos protocolos da web (HTTP) e visa uma maior escalabilidade do sistema por uma série de medidas.

Uma delas é armazenar o estado da aplicação no cliente, e não no servidor. Isso permite que mais nós seja acrescentados à aplicação, caso seja necessário, para comportar um aumento no volume de solicitações recebidas, sem que haja a necessidade de se manter a comunicação entre os nós servidores acerca do estado da aplicação para cada cliente. No caso de uma aplicação implantada em nós distribuídos, a falta de estado combinada à distribuição dos nós em infraestruturas independente proporciona, adicionalmente, a vantagem de tolerância a falhas.

Outro benefício desse estilo de arquitetura é possibilitar o gerenciamento de cache, no servidor, em *proxies* intermediários e nos clientes, visando para otimizar o tráfego de dados redundantes na rede. O próprio protocolo HTTP provê os mecanismos para que sejam informadas as situações em que o *cache* pode ser utilizado com segurança, bem como as situações em que ele deve ser renovado com novos dados a partir da origem.

Outra característica é que as informações necessárias para as próximas requisições encontram-se representadas dentro do documento recebido em cada solicitação. Este princípio, chamado de *Hypermedia as the Engine of Application State* (Hipermídia como o Motor do Estado da Aplicação), preconiza que, seja qual for o formato do documento (ex.: html, xml, json), ele deve prover links para as próximas requisições possíveis que possam interessar ao cliente. Esta característica traz a vantagem de enfraquecer o acomplamento entre o cliente e a versão do serviço, uma vez que, se o cliente consome os endereços a ele fornecidos pelo servidor, ele não seria quebrado pela possível alteração desses endereços.

Há, ainda, a possibilidade de se utilizar a compactação transparente de dados durante a transmissão, nos casos em que for verificado pelo servidor que essa funcionalidade é suportada nos clientes.

O uso de uma arquitetura REST para serviços web visa aproveitar neles os benefícios proporcionados pela arquitetura Web e pelo protocolo HTTP.

3.2 Estrutura de persistência dos dados (MER)

Esta seção foi intencionalmente descrita separadamente da arquitetura do sistema. O objetivo é separar a arquitetura modelo da implementação específica do caso Dados Abertos SICAF.

O modelo entidade relacional abaixo é uma simplificação do modelo entidade relacional do SICAF em produção. Do modelo inicial foram removidos dados que não são públicos, dados de controle interno das transações do sistema em produção, e dados irrelevantes. Nesta etapa do projeto a simplicidade foi tomada como norma, e a incorporação de novas tabelas foi analisada uma a uma. Em alguns casos houve desnormalização, considerando a simplicidade sem onerar o desempenho.

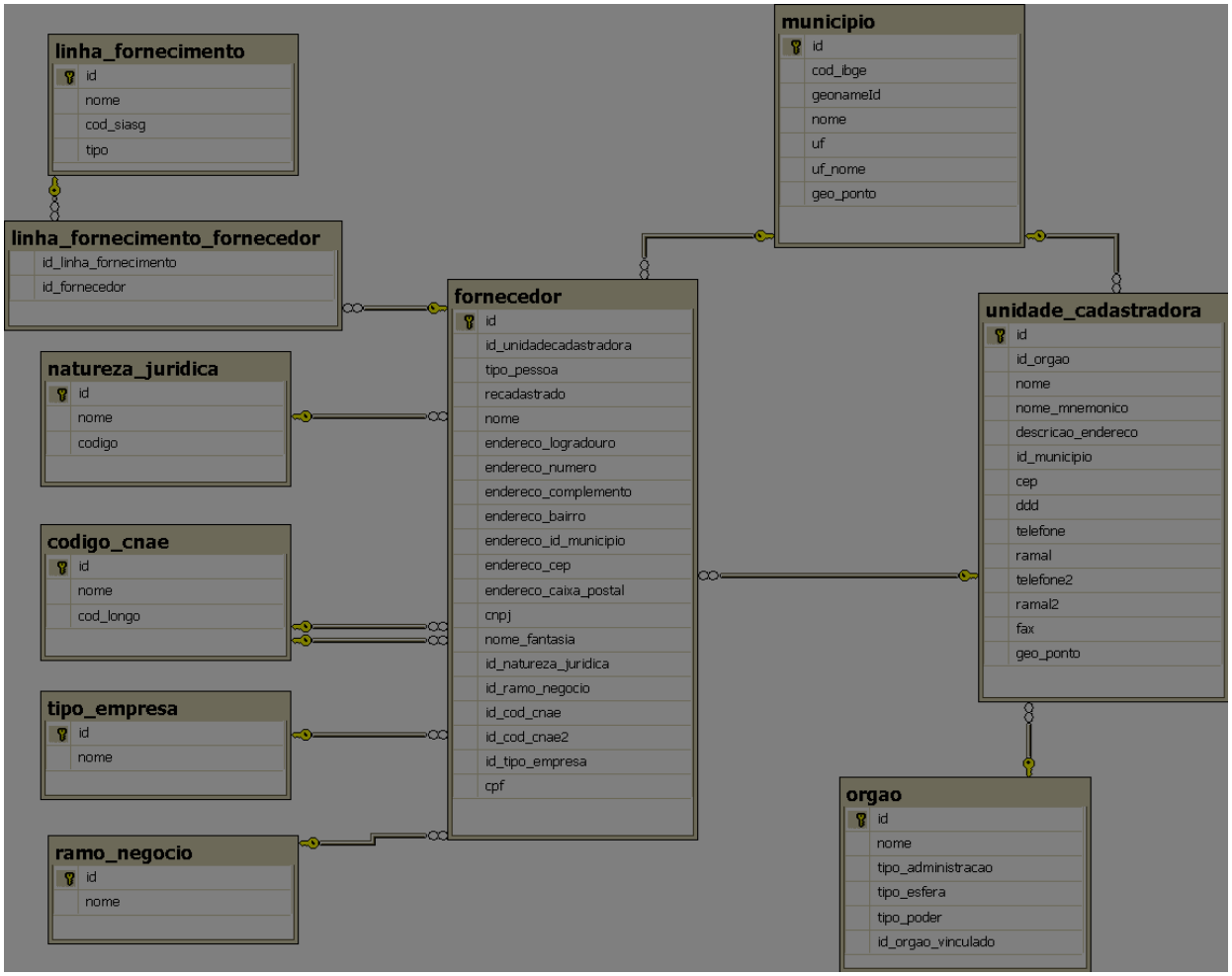


Ilustração 4: Modelo Entidade Relacional do banco de dados da API.

3.3 Módulos

3.3.1 Interface de acesso aos dados

O acesso é feito pela internet, utilizando o protocolo HTTP, a partir de uma requisição a URLs específicas para o tipo de consulta desejado, conforme especificadas na documentação do sistema.

Como o acesso utiliza protocolos da web e formatos e padrões abertos e amplamente conhecidos, há uma ampla gama de ferramentas e bibliotecas em muitas linguagens de programação que permitem a utilização dos dados de forma simples.

3.3.1.1 Padrão de construção de URL de consulta

As URLs do serviço se dividem em dois grupos:

- aquelas que retornam informações sobre algum tipo individual de objeto, seguindo os princípios de Linked Data, detalhadas na próxima seção, e
- aquelas que chamam uma API que retorna informações sobre um conjunto de objetos do mesmo tipo, filtrados opcionalmente por uma série de parâmetros.

A prática em que se utiliza uma URI para identificar um objeto físico ou abstrato que redireciona para uma URL que retorna um documento com informações sobre objeto, bem como o uso da

negociação de conteúdo HTTP, estão condizentes com o que preconiza o W3C no documento "*Cool URIs for the Semantic Web*". Cabe ainda observar que esse princípio é também utilizado na iniciativa de dados abertos ligados do Reino Unido, a qual está descrita no documento "*Designing URIs for the Public Sector*".

As URLs das consultas à API seguem a forma `http://api.comprasnet.gov.br/sicaf/{versao}/consulta/{metodo}.{formato}?{filtro-1}={valor-1}&{filtro-2}={valor-2}&…&{filtro-n}={valor-n}`, sendo o formato e os filtros opcionais. Se o formato não for especificado, o servidor realizará negociação de conteúdo HTTP para escolher um formato. Se nenhum filtro for especificado, serão retornados todos os registros, respeitado o limite máximo na quantidade de registros por requisição. A lista de métodos disponíveis na API pode ser consultada na documentação do sistema, disponível no endereço `http://api.comprasnet.gov.br/sicaf/doc/`, ou ainda, em forma processável por máquina, a partir do endereço `http://api.comprasnet.gov.br/sicaf/v1/consulta/`.

3.3.1.2 Exemplos de consulta

Abaixo segue uma lista do conteúdo resultante de algumas consultas:

- Consulta dos municípios do estado de Pernambuco que contenha "ados" como parte do nome em formato xml:

http://api.comprasnet.gov.br/sicaf/v1/consulta/municipios.xml?uf=PE&nome=ados

```
<municipios>
    < municipio href="http://api.comprasnet.gov.br/sicaf/id/municipio/23019" id="23019">
        <cod_ibge>2600104</cod_ibge>
        <nome>Afogados da Ingazeira</nome>
        <geonameId>3408274</geonameId>
        <geo_ponto>
            <lat>-7.736764</lat>
            <lon>-37.619728</lon>
        </geo_ponto>
        <uf_nome>PERNAMBUCO</uf_nome>
        <uf>PE</uf>
    </municipio>
    < municipio href="http://api.comprasnet.gov.br/sicaf/id/municipio/24813" id="24813">
        <cod_ibge>2609105</cod_ibge>
        <nome>Machados</nome>
        <geo_ponto>
            <lat>-7.703749</lat>
            <lon>-35.500524</lon>
        </geo_ponto>
        <uf_nome>PERNAMBUCO</uf_nome>
        <uf>PE</uf>
    </municipio>
</municipios>
```

- Consulta das Unidades Cadastradoras do Distrito Federal que contenha "apoio" como parte do nome e em formato csv:

http://api.comprasnet.gov.br/sicaf/v1/consulta/unidades_cadastradoras.csv?uf=DF&nome=apoio

id,uri,cep,ddd,descricao_endereco,fax,geo_ponto/lat,geo_ponto/lon,municipio,nome,nome_mnemonico,orgao,ramal,ramal2,telefone,telefone2,total_forn,total_recad
330086,http://api.comprasnet.gov.br/sicaf/id/unidade_cadastradora/330086,70059900,,"ESPLANDA DOS MINISTERIOS BLOCO F 2º ANDAR SALA 249, NEXO ALA A",,-15.8081634,-47.8813175,<objeto: http://api.comprasnet.gov.br/sicaf/id/municipio/97012>,PROJETO DE APOIO A REESTRUT. DO SIST. PREVIV.,PROASP/MPS,<objeto: http://api.comprasnet.gov.br/sicaf/id/orgao/33000>,,,,,0,0
120006,http://api.comprasnet.gov.br/sicaf/id/unidade_cadastradora/120006,70045900,61,"ESPLANADA DOS MINISTERIOS BL. ""M"" - ED. ANEXO - TERREO",,-15.8083615287,-47.8813775145,<objeto: http://api.comprasnet.gov.br/sicaf/id/municipio/97012>,MAER-GAPBR-GRUPAMENTO DE APOIO DE BRASILIA/DF,GAPBR,<objeto: http://api.comprasnet.gov.br/sicaf/id/orgao/52111>,,,20232611,20232611,32,5

- Consulta de Fornecedores do município Juina – MT (id=98310) e que forneça equipamentos de cemitérios (id=702), em formato JSON

http://api.comprasnet.gov.br/sicaf/v1/consulta/fornecedores.json?id_municipio=98310&id_linha_fornecimento=702


```
{
  "metadados":{
    "total_registros":1
  },
  "fornecedores":[
    {
      "cnpj":"8562676000180",
      "unidade_cadastradora":{
        "UnidadeCadastradora":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/unidade_cadastradora/255022",
          "id":255022
        }
      },
      "nome":"C. L. PAGNUSSATT - ME",
      "cod_cnae":null,
      "porte_empresa":{
        "PorteEmpresa":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/porte_empresa/5",
          "id":5
        }
      },
      "ramo_negocio":{
        "RamoNegocio":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/ramo_negocio/49",
          "id":49
        }
      },
```

```
    "href":"http://api.comprasnet.gov.br/sicaf/id/fornecedor_pj/119776",
    "natureza_juridica":{
      "NaturezaJuridica":{
        "href":"http://api.comprasnet.gov.br/sicaf/id/natureza_juridica/1",
        "id":1
      }
    },
    "linhas_fornecimento":[
      {
        "LinhaFornecimento":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/linha_fornecimento/1121",
          "id":1121
        }
      },
      {
        "LinhaFornecimento":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/linha_fornecimento/645",
          "id":645
        }
      },
      {
        "LinhaFornecimento":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/linha_fornecimento/702",
          "id":702
        }
      },
      {
        "LinhaFornecimento":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/linha_fornecimento/2950",
          "id":2950
        }
      }
    ],
    "endereco":{
      "bairro":"CENTRO",
      "logradouro":"AV MATO GROSSO, 148",
      "municipio":{
        "Municipio":{
          "href":"http://api.comprasnet.gov.br/sicaf/id/municipio/98310",
          "id":98310
        }
      },
      "cep":"78320000"
    },
    "cod_cnae2":null,
    "id":119776,
    "recadastrado":false
  }
]
```

}

3.3.1.3 Padrão de URI para recursos

As URIs que identificam recursos individuais seguem o padrão `http://api.comprasnet.gov.br/sicaf/id/{classe}/{identificador}` para identificarr objetos específicos. Esses são recursos não-informacionais que, quando acessados, redirecionam a requisição pelo código HTTP 303 para uma URL de documento (um recurso informacional), da forma `http://api.comprasnet.gov.br/sicaf/doc/{classe}/{identificador}`, o qual representa um documento que contém informações sobre o objeto identificado. Quando esse recurso for solicitado por um cliente, será realizada pelo servidor uma negociação de conteúdo HTTP (ver RFC 2616, que define o protocolo HTTP 1.1). Dessa negociação de conteúdo, baseada no cabeçalho `Accept` da requisição, resultará no formato preferido pelo cliente dentre os suportados pelo servidor. Haverá, então, mais um redirecionamento, acrescentando o trecho `.{formato}` ao final da URL, que representa o referido documento em um formato específico.

3.3.1.4 Implementação RESTful

A Interface de Programação de Aplicações (API) implementada no piloto de Dados Abertos do SICAF segue o estilo REST (*REpresentational State Transfer*) de *webservices*, ou seja, uma arquitetura alinhada à Web, como descrito na Seção 11.

Procurou-se, seguindo as boas práticas de serviços web RESTful, construir as URIs de consultas e de identificação dos conceitos a partir de segmentos “amigáveis”. Isto é, visou-se fazer com que um ser humano, ao visualizar a estrutura da URI, pudesse tirar conclusões antecipadas acerca da natureza dos dados que seriam retornados.

Naturalmente, como todos os métodos da API são somente para a consulta e não alteram estado, seguindo as recomendações do protocolo HTTP, e ainda, para possibilitar o uso de cache na aplicação, usou-se somente o verbo GET desse protocolo para realizar toda e qualquer consulta.

Seguiu-se, também, o princípio da Hipermídia como o Estado da Aplicação do estilo de arquitetura REST ao disponibilizar, no corpo dos documentos enviados como resposta às requisições HTTP, *hyperlinks* para outros documentos que possam ser relevantes no contexto. Por exemplo, na consulta às informações de um dado município, são fornecidos os links para as consultas de fornecedores naquele município e de unidades cadastradoras presentes naquele município. Outros links que podem estar presentes são aqueles que apontam para a consulta à próxima página, no caso do resultado ser paginado. Esses exemplos podem ser conferidos nos códigos exibidos na Seção 13.

3.3.1.5 Funcionamento em poucas palavras

A aplicação tem seu funcionamento semelhante a qualquer outra aplicação web. As consultas aos dados são feitas através de requisições GET com o protocolo HTTP.

A URL submetida carrega informações sobre qual o dado solicitado, quais filtros a serem aplicados no banco de dados e qual o formato do arquivo que será retornado. Por exemplo http://api.comprasnet.gov.br/sicaf/v1/consulta/municipios.xml?uf=PE retorna um arquivo xml contendo dados de todos os municípios do estado de Pernambuco.

O servidor web, Apache neste caso, avalia em sua estratégia de cache se há o recurso pronto para retornar, se não, a requisição é transferia para a aplicação.

A aplicação processa a URL de consulta e extrai os parâmetros, formato de retorno, tipo de dado e conjunto de filtros.

A camada superior executa o método específico submetendo os parâmetros na camada de negócio, responsável por extrair o conjunto de dados especificado.

O acesso aos dados é abstraído por uma biblioteca de mapeamento objeto relacional.

O conjunto de dados é então serializado no formato solicitado.

Sobre esse resultado é calculado um resumo criptográfico (*hash*), o qual é denominado e-Tag no protocolo HTTP. A aplicação verifica se a requisição passada foi um GET condicional, e se foi informado um valor de e-Tag presente no cache do cliente. Caso exista esse valor e ele coincida com o e-Tag calculado, o servidor economiza banda de transmissão enviando uma resposta, sem corpo, com status 304 (Não modificado). Essa resposta serve para indicar ao cliente que a sua cópia de cache ainda é válida e atual.

A aplicação verifica se o cliente indicou, pelos cabeçalhos da requisição HTTP, suportar a compactação da informação em formato gzip. Em caso afirmativo, o corpo da resposta é compactado antes do envio.

A resposta é repassada para o servidor web.

Caso a resposta não apresente qualquer anormalidade, isto é, possua, o status 200 (OK), o servidor web a armazena em cache para consultas posteriores.

Por último, o cliente recebe o arquivo como retorno da requisição HTTP. Se necessário, o servidor dispõe dos códigos de retorno HTTP para expressar comportamentos adversos.

3.3.2 Replicação dos dados

O módulo de replicação é responsável por popular o banco de dados da aplicação com os dados devidamente selecionados. Este processo é executado diretamente no banco de dados, independendo do serviço de consulta aos dados. A implementação foi feita estritamente com scripts SQL, específicos para o PostgreSQL.

Diferentemente do módulo de acesso aos dados, este processo é extremamente acoplado à tecnologia utilizada, ou seja a linguagem de script do PostgreSQL. Essa abordagem não é recomendada, pois enrijece a arquitetura, dificultado a extensibilidade e portabilidade.

O procedimento de replicação é semelhante à tarefas ETL, ou seja, extração, transformação e carga dos dados. Recomendamos a utilização de ferramentas de alto nível para este tipo de integração de base de dados. Um exemplo de ferramenta útil para esse tipo de tarefa é o Pentaho Kettle, software livre e compatível com praticamente todos os tipos de dados.

No caso do SICAF em produção, diversas políticas de segurança do Serpro tornam impossível conexões diretas ao banco de dados. A solução padrão é a disponibilização de um arquivo DUMP, extraído do banco de dados, através de uma fila *Message Queue* (MQ).

Enfatizamos que a alternativa de replicação que implementamos no Dados Abertos SICAF demonstrou ser a solução mais rápida naquele momento, porém não recomendamos como parte da arquitetura extensível e portável. Para fins de conhecimento detalhamos abaixo a solução implementada. Apesar do serviço estar funcionando perfeitamente para consulta, o módulo de replicação não foi finalizado, ou seja, sua execução é semiautomatizada.

3.3.2.1 Funcionamento em poucas palavras

Nesta seção são explicadas as etapas do processo semiautomatizado de replicação dos dados. Estas etapas também estão descritas no arquivo *Roteiro para migração.txt* localizado na pasta *SICAF scripts migracao* que acompanha os arquivos fonte do projeto. Todas as etapas são executáveis utilizando uma aplicação cliente do banco de dados PostgreSQL. No nosso caso utilizamos o pgAdmin III.

Etapas:

1. Copiar o arquivo DUMP extraído do SICAF em produção para o sistema de arquivo do servidor de banco de dados. Restaurar este backup em uma nova instância no banco de dados chamada SICAF_dump.
2. Criar uma nova instância no banco de dados chama sicaf_novo utilizando o sicaf_template como template. Neste momento o sicaf_novo não possui dados.

3. A sequência de execução a seguir deve ser executada na órdem especificada. Cada script é responsável por copiar um conjunto de dados do SICAF_dump e carregá-los no sicaf_novo. Cada um corresponde a uma sentença SQL combinando um comando SELECT com um INSERT em seguida.
    1. Executar o script *migra municipio.sql*. Popula a tabela municipio no sicaf_novo com dados da tabela sicaf_municipio do SICAF_dump.
    2. Executa o script *migra orgao.sql*.
    3. Executa o script *migra linha_fornecimento.sql*.
    4. Executa o script *migra unidade_cadastradora.sql*.
    5. Executa o script *migra tipo_empresa.sql*.
    6. Executa o script *migra ramo_negocio.sql*.
    7. Executa o script *migra codigo_cnae.sql*.
    8. Executa o script *migra natureza_juridica.sql*.
    9. Executa o script *migra fornecedor fisico.sql*.
    10. Executa o script *migra fornecedor juridico.sql*.
    11. Executa o script *migra linha_fornecimento_fornecedor.sql*.
    12. Executa o script *update migração municipio.sql*. Atualiza as colunas geo_ponto, cod_ibge e geonameID a partir da instâncis de produção de dados abertos, *sicaf*. A coluna geo_ponto foi gerada a partir de uma base de dados do IBGE contendo o polígono de fronteira de todos os municípios. geo_ponto é o baricentro desse polígono. O cod_ibge foi mapeado através de um batimento semiautomatizado com uma tabela do IBGE. O geonameID foi conseguido através de consultas ao web service www.geonames.org. Estes procedimentos devem ser executados novamente apenas nos casos de alteração na base de municípios do IBGE.
    13. Executa script de geração de índices *cria indices.sql*.
    14. A instância sicaf_novo está pronta para entrar em produção, para isso basta renomeá-la para sicaf, necessariamente a instância que estava sendo utilizada precisa ser renomeada para não conflitar. Sugerimos que ela seja guardada por algum tempo, possibilitando sua restauração em caso de erros.

3.3.2.2 Melhorias futuras

Diversas melhorias estão planejadas para o módulo de replicação. O principal objetivo é torná-lo auto mático, possibilitando um agendamento da atividade de replicação, assim os dados podem ser atualizados com frequência diária ou semanal. Esta melhoria será implementada utilizando python, a linguagem utilizada no módulo de acesso aos dados. Através de uma biblioteca de integração com o postgreSQL (psycopg2) todos os scripts podem ser executados através dos comandos específicos do banco.

Outras alternativas podem ser utilizadas, como por exemplo, a transcrição de todos os scripts em procedimentos python, fazendo uso do SQLAlchemy para abstração do banco de dados relacional. Outra alternativa é a implementação dos scripts com uma ferramenta ETL, como por exemplo, o Pentaho Kettle, já citado anteriormente.

3.4 Tecnologias e frameworks

Esta seção agrega todos frameworks e bibliotecas utilizadas na implementação da solução Dados Abertos SICAF. Porém, gostaríamos de esclarecer que esta combinação não é mandatória para implementação de dados abertos governamentais. Este conjunto de ferramentas demonstrou ser a melhor combinação considerando as linguagens de programação dominadas pelo time e as alternativas existentes. Várias outras combinações de linguagens e frameworks podem alcançar o

mesmo, ou melhor, resultado. O objetivo principal desta documentação é fornecer um conjunto de padrões e arquiteturas de referência para implementação de soluções de abertura de dados.

3.4.1 Linguagem de programação

Python, versão 2.5. A linguagem foi escolhida pelo amplo suporte de bibliotecas para realizar as diversas tarefas necessárias durante o projeto (mapeamento objeto-relacional, serialização em vários formatos, etc.), pela possibilidade de implementação ágil e pelo conhecimento da equipe de desenvolvimento.

Lembramos que outras linguagens também podem atender a esses requisitos, portanto, a escolha da linguagem deve ser realizada considerando-se a realidade e o contexto presentes no processo de desenvolvimento de software adotado.

3.4.2 Framework web

Pyramid, versão 1.0. Esse framework para aplicações web possui rápido desempenho e facilidade para implementar as funcionalidades requeridas para HTTP (ex.: negociação de conteúdo, compactação). Além disso, tem ampla documentação e uma comunidade ativa de desenvolvedores.

3.4.3 Serialização

A serialização é o processo que transforma os dados estruturados, presentes na memória, em uma sequência de valores (*bytes*), seguindo um determinado formato, a serem gravados em um arquivo ou transmitidos pela rede. Para a serialização dos dados, foram desenvolvidas classes para abstrair o formato de serialização. Essas classes serializam apenas os atributos que estiverem na presentes na lista de atributos expostos para o webservice, a qual é definida no código que mapeia o modelo de dados (isto é, nas definições das classes relacionadas ao negócio). Por não depender dos dados em si, esse componente pode ser facilmente reutilizado em outros projetos de aberturas de dados que utilizem linguagem Python.

Para cada formato disponível, foi utilizada uma biblioteca Python específica para a serialização. Classes para serializar em formatos JSON e CSV estão presentes na biblioteca padrão do Python. Para o formato XML, foi usada a biblioteca Amara, versão 2.0, em vez da ElementTree ou Minidom presentes na biblioteca padrão, pois a forma de serialização por geradores, suportada pelo primeiro, facilita a geração de elementos e atributos XML aproveitando automaticamente estruturas hierárquicas que podem presentes nos objetos (i.e. dicionários e listas Python aninhadas).

Já os quatro formatos de arquivos baseados em RDF – RDF/XML, N-Triplas, Turtle e Notação 3, foram tratados com a biblioteca RDFLib, versão 3.0. O código de serialização transpõe os dados expostos do objeto para o modelo de grafo RDF e, somente no momento da serialização, o formato específico de arquivo é repassado ao método de serialização de grafo da RDFlib.

3.4.4 Banco de dados – PostgreSQL

O banco de dados utilizado foi o PostgreSQL [http://www.postgresql.org/] versão 8.4. Ele foi escolhido por ser um dos bancos de dados em software livre mais utilizados. Todos componentes do time de dados abertos têm experiência com banco de dados relacionais e não tiveram barreiras para aprender as especificidades do PostgreSQL. Outro fator que contribuiu para a escolha é o suporte aos tipos geográficos de maneira transparente. Para isso é preciso instalar o módulo PostGIS. Isso adiciona uma série de recursos de manipulação dos dados geográficos.

Consideramos a alternativa de utilizar algum banco de dados não relacional, conhecidos pela sigla NoSQL. Exemplos são o CouchDB, Riak e MongoDB. Esse novo paradigma de banco de dados incorpora uma série de sofisticações que supre necessidades específicas dos sistemas atuais, como acesso por chave-valor, processamento e replicação distribuídos, orientação a consultas, persistência e serialização nativa em JSON e criação de *views* para o algoritmo *mapreduce* em linguagem

Javascript.

3.4.5 Mapeamento objeto relacional – SQLAlchemy

Para integrar a aplicação ao banco de dados PostgreSQL foi utilizado o SQLAlchemy [http://www.sqlalchemy.org/]. Esta ferramenta realiza a integração através de uma camada de abstração do modelo relacional do banco de dados, permitindo ao programador permanecer no paradigma orientado a objetos. SQLAlchemy realiza todas manipulações no banco de dados de forma transparente, inclusive cria as tabelas e o esquema de dados automaticamente de acordo com as classes modeladas em Python. Esse desacoplamento possibilita a troca posterior do banco de dados, se necessário. O SQLAlchemy é compatível com vários outros bancos de dados relacionais, tanto softwares livres quanto proprietários.

3.4.6 Dados geográficos – GeoAlchemy + PostGIS

Para manipulação dos tipos de dados geográficos foi utilizado o GeoAlchemy [http://www.geoalchemy.org/], uma extensão do SQLAlchemy para esse fim. Ele fornece abstração aos tipos geográficos como se fossem tipos nativos da linguagem. Complementando, o banco de dados deve suportar tipos geográficos, nesse caso foi instalado o módulo PostGIS [http://postgis.refractions.net/] no banco de dados PostgreSQL. Assim como o SQLAlchemy suporta diversos bancos de dados, o GeoAlchemy suporta várias das respectivas extensões geográficas existentes para o banco. Isso vem corroborar a possibilidade de se substituir, com pouco esforço, o banco por outra solução de banco de dados relacional.

Inicialmente foi planejada a possibilidade de uso dos campos geográficos para filtrar consultas de Unidades Cadastradoras. Por exemplo, consultar todas as Unidades Cadastradoras num raio de 3000m em torno de um ponto geográfico passado como parâmetro. GeoAlchemy possui operadores para esse propósito, porém esse filtro não se mostrou essencial para a versão alpha do projeto e foi implementada apenas na consulta a unidades cadastradoras. Nela, o parâmetro *geo_ponto* define um ponto central para busca de unidades cadastradoras e o parâmetro raio define o *raio* máximo de pesquisa.

Além disso, foram desenvolvidos scripts para fazer o batimento dos códigos internos de municípios com os códigos do IBGE e com os códigos do Geonames.

3.4.7 Scripts SQL

Na Seção 20é esclarecida a utilização de uma ferramenta de mapeamento objeto relacional, uma estratégia de engenharia de software para desacoplar a aplicação do banco de dados, que dentre outras facilidades dispensa a utilização da linguagem nativa de consulta ao banco de dados, o SQL, sendo substituída por operadores do mundo orientado a objetos. Em outras palavras, não foram utilizadas sentenças SQL na interface de consulta aos dados.

Na Seção 17[continua aí, papai... tem uma exceção: os scripts de replicação da base]

3.4.8 Modelagem do vocabulário de ontologia – Protégé

Um dos principais requisitos definidos para o projeto, é a utilização de ontologias, que enriquece semanticamente os dados disponibilizados. Para a atividade de modelagem da ontologia foi utilizado o editor Protégé [http://protege.stanford.edu/]. A ferramenta é de código livre e aberto, e é mantida pela Universidade de Stanford. Provê um ambiente gráfico de edição de ontologias OWL integrado à Internet. Nela, foram esboçadas as classes e propriedades necessárias para descrever os dados de domínio da aplicação do SICAF. Alguns conceitos são de domínio externo ao SICAF (e.g. modalidade de licitação) e foram modelados em arquivos separados.

Para as classes e propriedades foram definidos identificadores – URIs – com o esquema HTTP e, seguindo o princípio de Linked Data, os arquivos dos esboços de vocabulários foram hospedados

em um servidor de tal forma que o acesso via HTTP à URI de uma classe ou propriedade retorne o vocabulário que a define. Para mais informações sobre esse vocabulário, veja a Seção 24.

Houve, ainda, a preocupação de fazer referência a vocabulários e ontologias externos, que são de ampla utilização internacional pela comunidade Linked Data. Os vocabulários externos referenciados são os seguintes:

- Friend of a Friend (FOAF): a classe correspondente a fornecedor pessoa física é definida como sendo subclasse de foaf:Person;
- VCARD: os endereços de pessoas jurídicas e unidades cadastradoras são descritos com o vocabulário VCARD (classe vcard:Address, propriedades vcard:adr, vcard:postal-code, vcard:street-address, vcard:tel);
- WGS64 Geo do W3C: para expressar coordenadas geográficas (latitude e longitude) em RDF (propriedades geo:lat e geo:long);
- DBPedia Ontology e DBPedia properties: a propriedade dbont:state liga o município ao estado, as propriedades dbont:abbreviation e dbprop:isocode descrevem atributos do estado (o qual não está modelado como um objeto e só é identificado como uma entidade no momento da serialização em RDF); os estados brasileiros e o Distrito Federal também estão mapeados para as respectivas instâncias presentes na DBPedia; um município é descrito como uma instância da classe dbont:Settlement;
- Vocabulary of Interlinked Datasets (VOID): usado para descrever o grafo RDF retornado por uma consulta como um void:Dataset;
- Upper Mapping and Binding Exchange Layer (Umbel): baseada no projeto OpenCyc, essa ontologia contém uma classificação conceitos gerais, alguns dos quais são referenciados na modelagem dos esboços vocabulários do SICAF (e.g. a classe correspondente a fornecedor pessoa jurídica é subclasse de umbel:CommercialOrganization, e órgão é subclasse de umbel:GovernmentalOrganization).

3.4.9 Servidor web – Apache

O piloto Dados Abertos SICAF funciona similarmente a qualquer outra aplicação web, ou seja, é um serviço que retorna, se existir, um recurso a cada requisição HTTP. Em nosso ambiente está sendo utilizado o servidor web Apache v.2.2.9.

A implementação de uma arquitetura web para acesso a dados possibilita a utilização de uma série de tecnologias sofisticadas comuns em aplicações web clássicas. O Apache possibilita o acoplamento de diversos módulos de maneira transparente à aplicação web que utilizará. Na nossa API estamos utilizando módulos de cache em disco para melhorar o desempenho e não repetir consultas desnecessariamente.

Usamos também, o módulo mod_wsgi, que implementa uma interface WSGI (Web Server Gateway Interface), padrão para a linguagem Python, pela qual a nossa aplicação Pyramid se conecta ao servidor web Apache.

3.4.9.1 Compactação Gzip

Para minimizar o fluxo de dados enviados pelo servidor, a API possibilita a compactação dos arquivos de forma transparente utilizando o protocolo HTTP. A compressão é feita seguindo a especificação GZIP do HTTP. É necessário que o cabeçalho da requisição HTTP informe que a aplicação cliente suporta a compactação, e conseguirá manipular a resposta.

3.4.9.2 Compartilhamento de Recursos de Origem Cruzada – CORS

Por questões de segurança, um navegador web (*browser*) normalmente não aceita que uma página web receba e processe por Javascript informações dinâmicas (requisições assíncronas, também conhecida como AJAX, apesar de esta denominação se restringir àquelas que utilizem o formato de

dados XML) recebidas a partir de uma URL situada em domínio distinto do da página. Essa medida visa evitar que scripts maliciosos se aproveitem de vulnerabilidades do tipo CSRF (*Cross-Site Request Forgery*), que podem estar presentes certas em aplicações web, para obter acesso indevido a dados do usuário contidos nessas aplicações enquanto o usuário estiver autenticado nas mesmas. Essa restrição é conhecida como a "política da mesma origem".

Ela, todavia, afeta aplicações legítimas que queiram misturar dados de diferentes fontes para, por exemplo, apresentar uma visualização que interesse ao usuário – os chamados *mashups*. Para contorná-la, os desenvolvedores web vinham utilizando uma técnica chamada JSONP ("*JSON with padding*") que possibilita receber documentos JSON modificados de qualquer domínio. Dentre outras limitações, essa técnica não permite o uso de outros formatos de transmissão dos dados, tais como XML ou CSV.

A alternativa moderna ao JSONP é a técnica CORS (*Cross-Origin Resource Sharing*), que está sendo padronizada pelo W3C, e é suportada por versões recentes dos principais navegadores. Ela possibilita que a fonte de dados especifique, por uso de cabeçalhos específicos na resposta HTTP, quais domínios estão autorizados a ler os dados. Isso possibilita que as aplicações *mashup* façam cruzamento dos dados diretamente no navegador, utilizando apenas JavaScript, dispensando uma aplicação de servidor para fazer esse cruzamento.

No caso da API de Dados Abertos do SICAF, pela natureza pública dos dados, usamos o CORS para definir que qualquer domínio poderá utilizar os dados, o que esperamos facilitar a criação de *mashups* e aplicações de visualização que utilizem os mesmos.

3.4.9.3 Cache

Foi configurado no servidor Apache um módulo de cache, o mod_disk_cache. Esse módulo persiste no disco rígido uma cópia do resultado para cada requisição HTTP que retorne dados. Até que o tamanho limite do cache seja atingido, a API Dados Abertos SICAF só é requisitada para gerar um arquivo para cada URL uma única vez, a cada alteração que os dados sofrerem. Quando houver uma requisição repetida o servidor Apache faz a interceptação da requisição e retorna o mesmo arquivo gerado na primeira consulta. A API foi desenhada para ser utilizada apenas para consulta, o que permitiu que a cache fosse configurada com um tempo muito grande, várias semanas. Cada vez que os dados são replicados do sistema de produção do SICAF, é necessário apagar todos os arquivos de cache, forçando a API a gerar novos resultados para cada requisição.

3.4.10 Sistema operacional

Todas as ferramentas descritas nesta seção estão instaladas em um único servidor virtualizado rodando uma instância do sistema operacional Debian Lenny 5.0.

[Mais alguma coisa essencial? Acho que seria bom colocar o tamanho da memória RAM, disco e talvez alguma outra característica da VM]

3.5 Padrões

A grande inovação na proposta de Dados Abertos é a possibilidade de acesso a dados de forma automatizável, ou seja, interação máquina-máquina. Para que isso aconteça, o conjunto de dados disponibilizado deve ser fornecido em algum formato estruturado. Essa e várias outras definições passam pela instituição de diversos padrões.

No piloto Dados Abertos SICAF foram implementados vários formatos de serialização para cada consulta aos dados. Diversificar os formatos de serialização facilita para o usuário, que pode escolher o que melhor supre sua necessidade.

Para potencializar o cruzamento dos dados do SICAF, os dados referentes aos municípios carregam vários códigos de outras bases geográficas.

3.5.1 Formatos, qual o melhor?

O funcionamento da API é muito simples, ela retorna um arquivo para cada requisição realizada. Além de filtros, a API possibilita que o usuário indique qual o formato que ele deseja receber, tudo através da URL. Todos os formatos disponíveis são de especificações abertas. É importante enfatizar que essa parte da aplicação segue vários padrões de projeto que possibilita o reuso do serializador.

Abaixo segue uma lista dos formatos disponíveis e uma breve explicação:

1. JSON:

JSON é um acrônimo para JavaScript Object Notation. É um padrão aberto de estruturação de dados baseado em texto e legível por humano. A especificação é a RFC 4627. JSON ganhou maior utilização com o advento do Ajax. A serialização em JSON é muito simples e resulta em uma estrutura pouco verbosa o que se mostra uma ótima alternativa para o XML. JSON possibilita serialização de estrutura de objetos complexos, como listas e subpropriedades. JSON está se tornando o padrão mais utilizado para integração de dados entre repositórios e frameworks. Assim como relatado na seção 19, JSON também está se tornando o padrão nativo de armazenamento em alguns bancos de dados modernos.

2. XML:

XML significa Extensible Markup Language, e é um conjunto de regras para codificar documentos em um formato legível por máquina. É baseado em texto e tem como principais objetivos [fontes?] simplicidade, extensibilidade e usabilidade. XML é largamente utilizado como formato de troca de dados nos clássicos Web Services SOAP. Apesar da larga utilização, é cada vez menos encorajada a utilização desse formato para integração de aplicações. Em substituição, recomenda-se utilizar JSON, por economizar banda e ser de processamento mais leve.

3. CSV:

CSV significa Comma-Separated Values, ou valores separados por vírgula, e é um formato para armazenamento de dados tabulares em texto. A codificação é muito simples: cada linha do arquivo representa uma linha na tabela, e as colunas são separadas por vírgula. CSV é recomendado para representação de estrutura de dados mais simples, de natureza tabular, onde não existem subpropriedades ou listas, gerando um arquivo menor e mais leve para processamento. Arquivos CSV são processáveis diretamente por editores de planilhas, como o OpenOffice e o Excel.

4. RDF/XML:

Alguns dos formatos disponíveis para a consulta na API baseiam-se no padrão Resource Description Framework (RDF), do W3C. Um deles, chamado RDF/XML, baseia-se na sintaxe do padrão XML, e utiliza, em geral, a extensão .rdf.

Os formatos baseados em RDF têm seus dados descritos em vocabulários de uso comum na web semântica e também em vocabulários próprios do domínio de dados do SICAF, que foram esboçados com esta finalidade. Para informações sobre os vocabulários utilizados e referenciados, veja as Seções 20e 24.

5. Notação 3:

Formato proposto por Tim Berners-Lee, o criador da Web e da Web Semântica, com o objetivo de permitir uma leitura facilitada por seres humanos e ao mesmo tempo ser igualmente fácil de processar por máquina. O conteúdo apresentado representa exatamente o mesmo grafo que o formato RDF/XML, porém, serializado nessa sintaxe. A extensão utilizada nesse formato é, em geral, `.n3`.

6. Turtle:

Subformato do Notation 3, porém, com algumas restriçẽs. Para uso desta aplicação, a serialização é exatamente a mesma do formato Notation 3. A extensão utilizada é `.ttl`.

7. N-Triplas:

Subformato simples do Turtle, todavia bastante prolixo, com o objetivo de facilitar o processamento à custa do espaço de armazenamento e transmissão. As triplas são representadas uma por linha, com sujeito, predicado e objeto expressos por extenso. A extensão utilizada é `.nt`.

3.5.2 Esboço de Ontologia do SICAF – RDF

Um vocabulário para o SICAF foi esboçado em OWL Lite a partir de informações coletadas com pessoas ligadas à área de negócio correspondente na DLSG. Ele foi modelado com as seguintes classes, e encontra-se disponível no endereço http://vocab.e.gov.br/sicaf#

- `Fornecedor`
- `OrganizacaoComercial` (equivalente a `umbel:CommercialOrganization`)
- `UnidadeCadastrador`
- `RamoNegocio`

O vocabulário define, ainda, as seguintes propriedades:

- `linhaFornecimento`
- `registradoEm` (e a inversa `responsavelPeloRegistroDe`)
- `recadastrado2011`

O objetivo desta modelagem não foi representar toda a semântica do negócio, e sim, possibilitar uma melhor recuperação da informação e viabilizar o cruzamento dos dados com outras fontes. Para esse propoísito, a expressividade da linguagem OWL Lite mostrou-se ser suficiente.

Em julho de 2011, foi realizada uma capacitação em engenharia de ontologias, com participantes, entre outros, da DSLG. Espera-se, subsequentemente, que esse esboço de vocabulário seja evoluído com as melhorias de qualidade potencializadas por esse ferramental, resultando em uma nova versão do vocabulário do SICAF que possa ser considerada estável.

3.5.3 Dados geográficos

Por diversos motivos decidimos dar uma atenção especial para dados geográficos. Nos últimos anos a utilização de informação geolocalizada ganhou maior relevância pela popularização de dispositivos com GPS, smartphones e navegadores, e o crescente número de serviços web que agregam camadas de informações sobre mapas.

Dado geográfico carrega nativamente grande poder de reutilização e cruzamento, isso acontece porque todo dado geográfico depende de um referencial comum, o globo terrestre. As ferramentas para cruzamento de dados geográficos potencializam a agregação da informação. Por exemplo, em um mapa, duas informações não precisam possuir exatamente o mesmo ponto geográfico, mas apenas estarem próximos, para se relacionarem.

Para incentivar o uso dos dados do SICAF num caráter geográfico a tabela de municípios foi trabalhada de maneira diferenciada. À tabela de municípios estão relacionadas as duas tabelas principais, a de Fornecedores e a das Unidades Cadastradoras. À tabela de municípios foi adicionado o campo geoponto, contendo a posição geográfica daquele município. Esse valor foi calculado como sendo o baricentro do polígono que delimita o município, informação disponível no portal do IBGE. À tabela de municípios também foram incorporados dados do código do IBGE, do código Geonames e da relação com a DBPedia. Abaixo segue explicação de cada um deles.

3.5.3.1 IBGE

O IBGE é o órgão que tem a competência legal para catalogar os municípios. Em outras

palavras, as informações mais fiéis sobre identificação, localização geográfica, históricos de mudanças e fronteiras dos municípios estão em sistema de posse do IBGE. Considerando a importância de se seguir uma nomenclatura comum, realizamos uma atividade de cruzamento da tabela de municípios do SICAF com a tabela do IBGE, disponível em ftp://geoftp.ibge.gov.br/. Assim incorporamos ao nosso piloto informações do código do IBGE, único para cada município, que segue uma lógica de composição deste identificador no IBGE. Já explicado na seção anterior, a posição geográfica foi calculada com dados das fronteiras do município no IBGE.

É importante lembrar que na nossa implementação da ontologia do SICAF, não foi criado o conceito para o campo cod_ibge, uma vez que enxergamos essa competência exclusiva ao IBGE.

Esta URL exemplifica o cruzamento de todas esses dados: http://api.comprasnet.gov.br/sicaf/dados/municipio/62138.xml.

3.5.3.2 GeoNames

GeoNames é um banco de dados geográfico disponível e acessível através de diversos web services. Além da possibilidade ampla de integração através de APIs cada propriedade dos dados está definida em uma ontologia. Esses diferencias fazem com que o GeoNames se torne referência internacional para interoperabilidade de dados geográficos. Semelhantemente ao código IBGE, incorporamos à tabela de municípios o código do geonames. Isso possibilita uma integração mais rica, podemos assim dizer que os dados do SICAF fazem parte da grande malha da web semântica. Esta URL mostra o poder dos relacionamentos das propriedades como recursos web: http://api.comprasnet.gov.br/sicaf/dados/municipio/62138.rdf.

3.5.3.3 DBPedia

DBPedia é um projeto com o objetivo de extrair conteúdo estruturado das informações contidas nas páginas da Wikipedia. A disponibilização do conteúdo num formato mais estruturado possibilita que usuários façam consultas mais complexas relacionando dados de outras fontes. Neste caso apenas informações em nível de unidade federativa foram agregadas ao resultado. Como existem apenas 27 unidades federativas, o mapeamento UF ==> URI na DBPedia foi feito diretamente no código, em Python. Assim, esses dados não estão disponíveis no banco de dados, apenas no código. Além disso, esse cruzamento está disponível apenas nas consultas dos municípios em RDF, como no exemplo a seguir: http://api.comprasnet.gov.br/sicaf/dados/municipio/62138.rdf, que contém informações do município de Barueri, e uma delas aponta para um recurso sobre o estado de São Paulo http://dbpedia.org/resource/S%C3%A3o_Paulo_(state).

4 Case - Painel de recadastramento de fornecedores no SICAF

O paradigma de dados abertos faz uma divisão clara entre uma arquitetura de disponibilização de dados e aplicativos que consomem esses dados. Ou seja, dados abertos se refere às tecnologias e meios sofisticados de disponibilização de dados na Internet. Nesse paradigma, o governo empodera o cidadão para desenvolver as aplicações, dedicando-se a prover uma infraestrutura aos dados mais genérica e padronizada.

Apesar de não ser o objetivo do governo, nesse paradigma de dados abertos, a implementação dos aplicativos que cada usuário pode solicitar, decidimos desenvolver um *mashup* para exemplificar o potencial de utilização dos dados e provocar a criatividade nos outros.

Recentemente o SICAF passou por uma reestruturação. Os cadastros de fornecedores foram incrementados e receberam novos campos. Para levantar essas novas informações todos os fornecedores estão sendo recadastrados. Para apoiar esse recadastramento, uma aplicação web foi desenvolvida utilizando os dados abertos disponibilizados. A aplicação foi desenvolvida utilizando apenas HTML e JavaScript.

O nome da aplicação é Painel de Acompanhamento do Recadastramento no novo SICAF e está disponível na URL: http://api.comprasnet.gov.br/sicaf/app/painel.htm

O painel é um conjunto de ferramentas integradas ao mapa do Google para entender como andam as taxas de recadastramento em cada região do Brasil. Gráficos de pizza foram incorporados ao mapa agrupando as Unidades Cadastradoras evitando que a imagem fique poluída. Também é gerado um gráfico de barras e um ranking classificando as Unidades Cadastradoras pela quantidade de recadastramento.

No desenvolvimento deste aplicativo tivemos o apoio técnico da Analista em TI Patricia Lustosa [patricia.ribeiro@planalto.gov.br].

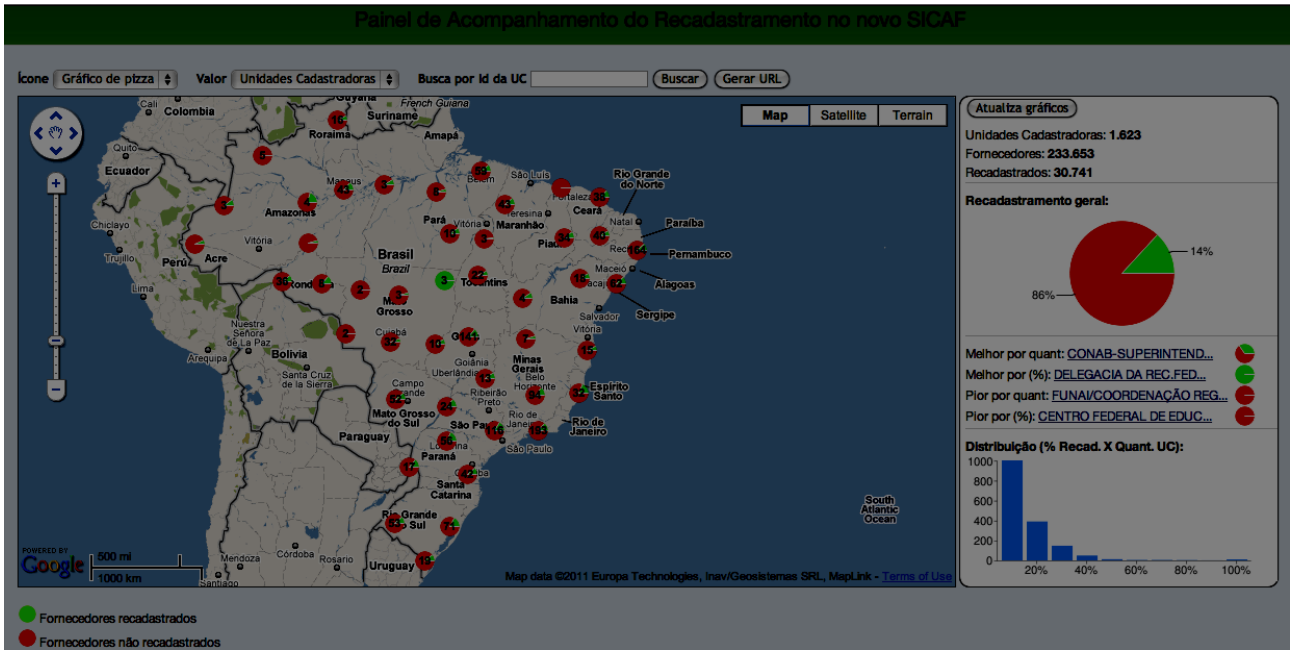


Ilustração 5: Painel de Acompanhamento do Recadastramento no novo SICAF.

Abaixo segue uma lista das tecnologias utilizadas no desenvolvimento deste *mashup*.

4.1 Tecnologias do Mashup

Por se tratar de uma simples ferramenta de visualização de dados, a aplicação é muito simples, funcionando apenas no navegador do cliente. Isso quer dizer que dispensa a utilização de algum sistema como servidor desta aplicação. O arquivo HTML é fornecido como qualquer outro recurso web (html estático), executando apenas no cliente, quando renderizado pelo navegador.

4.1.1 Página HTML

A aplicação é uma pagina web comum com alguns diferenciais de visualização de dados. No servidor web, a aplicação é apenas mais um arquivo estático no sistema de arquivo, ou seja, não existem scripts de geração desse HTML.

4.1.2 JavaScript

Para a utilização do mapa, impressão das pizzas, do gráfico de barras e calculo das estatísticas, foi utilizado programação em JavaScript. É uma linguagem muito simples e de fácil programação. Apesar de ser interpretada pelo navegador, torna as aplicações web mais rápidas, uma vez que evita comunicação constante com o servidor para tarefas que não envolvem outros dados. Note que, ao pressionar o botão “Atualizar gráficos”, as informações dentro do retângulo à direita são recalculadas considerando apenas as Unidades Cadastradoras visíveis no mapa naquele momento. Esse é um exemplo de logica de negocio mais complexa que foi implementada em JavaScript.

4.1.3 Ajax - jQuery

Já foi explicado anteriormente que o aplicativo executa exclusivamente no lado cliente, no navegador. É muito comum nas aplicações atuais, a realização de consulta a dados em resposta a algum evento do usuário, e que não acarretam o carregamento de uma nova página HTML. Essa prática é conhecida como Ajax, chamadas assíncronas em JavaScript.

Para realização das consultas dos dados disponibilizados em nossa API e desserialização do mesmo, fizemos uso da biblioteca jQuery, disponível em http://jquery.com/. Para esse fim, realizamos requisições em formato JSON.

Em nossa aplicação essas consultas acontecem exclusivamente em dois momentos. Ao iniciar o carregamento da página automaticamente é feito uma consulta com dados de todas as unidades cadastradoras, através da URL http://api.comprasnet.gov.br/sicaf/v1/consulta/unidades_cadastradoras.json. Esses dados são então mantidos em uma variável durante todo o funcionamento da aplicação, utilizada em vários casos. O segundo caso, acontece ao clicar em algum marcador das Unidades Cadastradoras, nesse momento a aplicação envia uma solicitação para acessar dados detalhados, como esta http://api.comprasnet.gov.br/sicaf/dados/unidade_cadastradora/120041.json.

4.1.4 Google Maps

A aplicação Google Maps possui uma biblioteca em JavaScript desenvolvida e mantida pelo Google para criação de aplicativos web que interajam com mapas. A utilização é muito simples, e toda a complexidade para administrar as imagens do mapa, ferramentas de zoom, e eventos de clique do usuário nos marcadores, é encapsulada pela biblioteca.

Em nosso aplicativo utilizamos poucas funcionalidades do mapa. Basicamente pintamos marcadores em posições específicas do mapa, além disso, para cada marcador tratamos o evento de clique, exibindo o balão com dados detalhados da Unidade Cadastradora. Por fim, acessando propriedades do mapa, podemos filtrar a lista de Unidades Cadastradoras mantendo aquelas que estão visíveis no mapa em um determinado momento. Com isso são calculadas várias estatísticas e gráficos, exibidos no retângulo à direita.

Documentação e tutoriais para utilização do Google maps estão disponíveis em http://code.google.com/apis/maps/index.html.

4.1.5 Google Charts

Google Charts é um serviço do Google para facilitar desenvolvedores na geração visualizações de dados. Nós fizemos uso do serviço através da API disponível em http://code.google.com/intl/pt-BR/apis/chart/image/docs/gallery/pie_charts.html. O serviço gera imagens customizadas em tempo real com os parâmetros passados na URL. Utilizamos o serviço para criar as imagens de pizzas e os gráficos de barras. Um exemplo é a URL:

```
https://chart.googleapis.com/chart?chxt=x,y&cht=bvs&chd=t:1003,
390,147,49,12,4,4,3,1,10&chco=76A4FB&chls=2.0&chs=300x120&chbh=20&chxl=0:||20%||
40%||60%||80%||100%&chxr=1,0,1003&chds=0,1003.
```

Todos os parâmetros necessários para compor a imagem estão na URL e o resultante é:

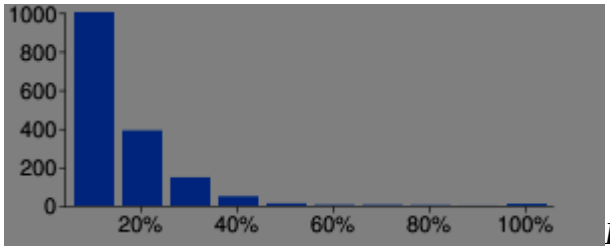


Ilustração 6: Imagem gerada em tempo real pelo Google Charts.

4.1.6 MarkerCluster

MarkerCluster é uma biblioteca que gerencia agrupamentos de marcadores para cada nível de zoom. O principal objetivo é tornar a navegação mais leve, mais legível e rápida. É um projeto mantido no Google Code no seguinte endereço: http://code.google.com/p/gmaps-utility-library-dev/. A utilização é simples, ele cria uma camada de abstração à biblioteca nativa do Google Maps, fazendo a gerência desta internamente.

No nosso caso, fizemos uma leve customização para possibilitar a utilização de marcadores customizados nos casos de agrupamento. No caso prático, em um agrupamento de Unidades Cadastradoras, o marcador é uma pizza considerando a soma das informações dos marcadores ali agrupados. Lembrando, o marcador de pizza é do Google Charts.

4.2 Recursos – 25h

Destacamos esta seção para que os desenvolvedores de *mashups* tenham uma noção de complexidade e custo no desenvolvimento de aplicativos simples e impactantes como este. O que queremos concluir é que a complexidade é muito baixa, para quem já tem alguma experiência com aplicativos web.

No papel de desenvolvedor, deve-se considerar que toda a plataforma para disponibilização dos dados, a API, já existe e é acessível por meio de requisições HTTP.

Considerando todo o tempo de desenvolvimento e melhoria da aplicativo, foram envolvidas duas pessoas, num total de 25 horas.

5 Considerações finais

5.1 Onde está o código fonte?

5.1.1 Portal do SPB

5.2 Licença de uso

5.2.1 GPL, versão 2 (SPB)?

5.3 Grupo consultor em Dados Abertos do SISP – C3S.