FACULDADE DE TECNOLOGIA DE SOROCABA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO FINAL

Caique Hitoshi Mitsuoka

PetAppy – Aplicativo para proteção dos animais

Sorocaba

Março/2017

PetAppy – Aplicativo para proteção dos animais

Caique Hitoshi Mitsuoka

Orientadora: Prof.ª Dr.º Paulo Edson Alves Filho

Sorocaba

Março/2017

RESUMO

Esta pesquisa é parte integrante de um projeto maior que propôs um protótipo de aplicação de fácil acesso e alta disponibilidade para a união da comunidade de pessoas interessadas na causa de animais abandonados e animais perdidos. É uma plataforma capaz de unificar os anúncios e o resgate de animais que estão sem um lar definitivo, que utiliza tecnologias para automatizarem processos e aumentar o alcance dentro dessas comunidades. Neste relatório especificamente é tratado a criação um protótipo de aplicação web, que ficará hospedado em um servidor de aplicação e receberá as requisições do aplicativo móvel. Será usado a ferramenta Ruby on Rails para gerar uma aplicação de comunicação HTTP e utilizar um servidor de aplicação para reunir e distribuir as informações entre os dispositivos móveis dos usuários.

Palavras-Chave: dispositivo móvel, qualidade de software, Ruby on rails, Web Service

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface (Interface de Programação de Aplicativo).

CPU – Control Process Unit (Unidade de Controle de Processo)

GNU – GNU's not Unix (GNU não é Unix).

IP – Internet Protocol (Protocolo de Internet).

RVM - Ruby Version Manager (Gerenciador de Versões do Ruby).

SGDB – Sistema Gerenciador de Banco de Dados.

SQL – Structured Query Language (Linguagem de Consulta Estruturada).

REST - Representational state transfer (Transferência de Estado Representacional)

SUMÁRIO

1 INTRODUÇÃO E JUSTIFICATIVA

Todos os dias em Sorocaba, seja nas ruas ou nas redes sociais, é possível notar o número alto de animais desaparecidos ou abandonados. Em contato com pessoas envolvidas na causa animal, foi detectado principalmente problemas de comunicação, no sentido de todos terem a informação precisa e atualizada. Um caso frequentemente relatado foi o de animais que foram encontrados, mas continuaram anunciados em outros meios.

Procurando uma maneira de ajudar a comunidade de Sorocaba na luta contra este tipo de situação e apresentando soluções para os problemas decorrentes, este trabalho propõe um servidor de aplicação web de suporte para um aplicativo multiplataforma para dispositivos móveis.

Unificando a base de dados de animais, cuidadores e donos de animais é possível atualizar as informações mais rapidamente aos interessados através de notificações e assim encerrar buscas de animais já encontrados e cruzar essas informações para informar apenas ao público de interesse.

O projeto aqui desenvolvido utiliza dados dos sensores dos dispositivos, como, por exemplo, localização, de modo a notificar um desaparecimento apenas na área onde ocorreu e otimiza o conteúdo mostrado para o usuário dentro do aplicativo.

2 OBJETIVOS

O presente projeto propõe criar uma aplicação web de alta disponibilidade e escalabilidade, usando tecnologias atuais de desenvolvimento e compatível com um aplicativo móvel tais como Ruby, Rails e PostgreSQL Ele também utiliza geolocalização para proporcionar mais precisão às informações exibidas.

O projeto tem como objetivo prover a união entre diversos grupos, entidades e apoiadores à proteção de animais da cidade, assim como possibilitar uma forma mais eficiente de acolher ou adotar um animal com base nas informações captadas pela base de dados centralizada da aplicação e dos diversos grupos.

Para esse fim, a pesquisa apresenta uma aplicação de servidor para servir às requisições do aplicativo, através do protocolo HTTP.

3 REVISÃO BIBLIOGRÁFICA

Para desenvolvimento do protótipo foi escolhido Ruby on Rails como base para a aplicação. Ruby on Rails é um pacote de ferramentas que são comumente utilizadas para desenvolvimento de aplicações web.

Fabio Akita ao citar “Ruby on Rails”, observa que “a promessa era substituir toda a complexidade, toda a burocracia, por uma pilha sugerida, onde a maioria das decisões básicas já estariam feitas, e assim nós poderíamos focar no que realmente importa: o negócio. ” (AKITA: 2016).

Quando o autor fala sobre a pilha sugerida, ele está se referindo às opções existentes no mercado de desenvolvimento web de tecnologias, e todas as discussões sobre quais tecnologias utilizar. Diferentemente de outras ferramentas web, Ruby on Rails traz a proposta de abstrair essa pilha, e entregar tudo que é preciso em um único pacote, sem que os desenvolvedores precisem tomar decisões em um primeiro momento. Isso leva o desenvolvedor a desde o primeiro momento focar-se em desenvolver funcionalidades que impactam diretamente o usuário ao invés de desenvolver a aplicação antes de qualquer funcionalidade.

Para comunicação entre sistemas foi utilizado a arquitetura REST (acrônimo em inglês para Transferência de Estado Representacional), que é:

> “uma abstração sobre a arquitetura dos elementos de um sistema distribuído. REST ignora os detalhes de implementação, protocolo e sintaxe para focar no papel dos componentes e restrições sobre a sua interação com outros componentes.“ (FIELDING, 2000)

Fielding faz referência aos sistemas web, que para se comunicar utilizam um protocolo de comunicação chamado http. Este protocolo de comunicação é agnóstico ao seu conteúdo, ou seja, o protocolo é apropriado para qualquer tipo de mensagem dentro do seu conteúdo, e não é preso a nenhum tipo de conteúdo ou padrão especifico. Sendo assim Fielding propõe uma abstração sobre o HTTP para comunicação entre sistemas, onde a requisição representa o estado do componente, independentemente de sua implementação, assim é possível enviar uma representação para atualizar seu estado em outro sistema, ou apenar obter o estado atual do componente.

MATERIAIS E MÉTODOS

Foram levantadas as necessidades da comunidade que motivam o projeto através de entrevistas e diálogos para avaliar como a aplicação proposta pode atender a essas necessidades e mitigar esses problemas.

Os computadores utilizados no projeto são do laboratório da Fatec Sorocaba, computadores particulares, bem como contamos ocasionalmente com a colaboração dos estagiários ou auxiliares de docente para a instalação das ferramentas.

3.1 Etapas

Para o desenvolvimento do protótipo foi utilizado a metodologia SCRUM de desenvolvimento, onde são feitas pequenas interações e entregas, validando frequentemente a aplicação e respondendo a mudança de requisitos. A documentação foi feita utilizando testes automatizados de unidade e de comportamento, assim é possível validar requisitos e alterações.

Levantamento requisitos da aplicação

Primeiramente foi levantado os requisitos de software, e a conclusão foi a utilização de Ruby on Rails como ferramenta para desenvolvimento web. Como banco de dados foi escolhido PostgreSQL, devido a possibilidade de utilizar um plugin chama PostGIS para trabalhar com geolocalização.

Identificação de entidades fundamentais

Para identificar as entidades importantes para o sistema foram feitas Reuniões entre os integrantes e membros da comunidade. Assim foram modeladas de acordo com duas necessidades.

Geração da aplicação base

Ruby on Rails possui uma ferramenta de linha de comando que auxilia na criação de um projeto, gerando sua estrutura inicial com os padrões utilizados pela comunidade. Para isso é necessário selecionar as ferramentas que serão utilizadas para as mesmas serem configuradas**.**

O projeto foi iniciado utilizando as seguintes configurações:

```
$ rails new petappy --database=postgresql --skip-yarn --skip-
      action-cable --skip-coffee --skip-turbolinks --skip-test
      --api
```

Foram desabilitadas as dependências de front-end e as ferramentas não relacionadas a API, e por último foi passado a opção de API, assim foi gerada uma pilha otimizada para API REST.

Implementação

A implementação como citado a cima, foi feita utilizando a metodologia SCRUM. Assim foi possível identificar outras necessidades e se adequar a mudanças rapidamente. Portanto não há uma ‘receita’ de como o sistema foi implementado e sim uma descrição do processo.

3.1.1.1 Modelos

O modelo é a representação de uma entidade no sistema, exemplo: Um animal é uma entidade, e ela precisa ser representada sistematicamente.

Assim para a implementação foi levantado uma serie de atributos importantes para representar esta entidade através do ActiveRecord, para mapear uma classe a uma tabela no banco de dados, e do ActiveModel, ferramenta do Ruby on Rails para ajudar no gerenciamento de classes.

3.1.1.2 Testes de modelo

Nos modelos foi preciso ser testado quais atributos o mesmo possui, garantindo sua integridade em relação a outros componentes do sistema, como quando um cliente do sistema considera a existência do atributo ‘cor’ para o componente ‘animal’. Também foi necessário checar a implementação dos métodos do componente considerando seu comportamento esperado. Assim foi garantido que quando houvesse alterações, o componente não perderia suas características.

3.1.1.3 Controladores

Controlador é a modulo do sistema que lida com requisições web. Através deste, é possível determinar como a aplicação vai se comportar diante a aplicação móvel (cliente). O controlador é onde, a documentação tem extrema importância para que desenvolvedores de outros sistemas tenham material de consulta.

3.1.1.4 Teste de Controladores

O teste de controlador foi feito do fazendo requisições e verificando suas respostas. O código de resposta é importante para saber a natureza da resposta, e o corpo da resposta também foi verificado, pois caso houvesse alterações, seria necessário também alterar a implementação de clientes.

3.2 Ferramentas

Descrição das ferramentas

3.2.1.1 RVM

O RVM (Ruby Version Manager) é um gerenciador de versões do Ruby por linha de comando. É necessário para que quando usada mais de uma versão do Ruby, não haja conflitos no ambiente de desenvolvimento.

3.2.1.2 Ruby

Ruby é a linguagem de programação utilizada no Web Service. *Dinâmica, open source* com foco na simplicidade e na produtividade. Tem uma sintaxe elegante de leitura natural e fácil escrita. O Ruby será utilizado para codificar a aplicação, assim é possível aproveitar não só as características da linguagem, mas também sua vasta biblioteca de código aberto.

3.2.1.3 Ruby On Rails

Framework de desenvolvimento web na linguagem Ruby. É um framework livre focado na velocidade e facilidade no desenvolvimento de aplicações orientadas a banco de dados (database-driven web applications), uma vez que é possível criar aplicações com base em estruturas pré-definidas. Isso permite velocidades e que o foco de desenvolvimento seja nas funcionalidades.

3.2.1.4 Postgresql

Banco de dados relacional open-source. Indicado para o desenvolvimento de aplicações em ruby on rails. "O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. ” (OLIVEIRA, 2003).

3.2.1.5 Gem Bundler

Ferramenta integrada ao Ruby que faz o gerenciamento das bibliotecas das quais a aplicação depende.

3.2.1.6 NGIX

É um software para servidor HTTP, utilizado como interface entre a aplicação e o sistema operacional.

Ferramentas instaladas nos computadores de trabalho:

- GNU/Linux Ubuntu 16.04 x64
- RVM - Versão 1.26.11
- Ruby – Versão 2.3.4
- Rails – Versão 5.1.1
- Postgresql – Versão 9.6
- Gem Bundler – Versão 1.15

Ferramentas instaladas no servidor

- GNU/Linux Ubuntu 16.04 x64
- RVM - Versão 1.26.11
- Ruby – Versão 2.3.4
- Rails – Versão 5.1.1
- Postgresql – Versão 9.6
- Gem Bundler – Versão 1.15
- NGINX – Versão 1.13.4

4 RESULTADOS E DISCUSSÕES

4.1 Aplicação web

O resultado foi uma aplicação web de alta disponibilidade e performance, utilizando uma interface REST para comunicação com o Cliente via HTTP.

Com os requisitos foi possível modelar a aplicação de forma que o usuário não insira informações desnecessárias, utilizando os próprios dados do aparelho, como localização para poder exibir informações de maneira inteligente.

4.2 Anúncios

No protótipo foi gerado uma rota para gerar anúncios de animais dentro do aplicativos, e este é o diferencial do cliente em relação a concorrência que apenas exibe anúncios de forma sequencial em que foram enviados. Para isso foi necessária a análise de dados para poder gerar esses anúncios de maneira eficiente. Utilizando dados de geolocalização do dispositivo móvel do usuário, a aplicação calcula um raio de proximidade e exibe para o mesmo as oportunidades, seja de adoção ou de doação próximas a ele, e de acordo com o tempo desde a postagem do anuncio, o mesmo também ganha relevância. Sendo assim os usuários recebem sempre os anúncios relevantes.

4.3 Aplicação web em tempo real

Quando se trata de aplicações web, tempo de resposta é muito importante, pois afeta não apenas custo de operação, mas também a experiência do usuário. E nos anúncios, enquanto a aplicação analisa dados para gerar a melhor resposta, o usuário espera pelo conteúdo.

Para que tempo de resposta fosse diminuído, foi necessário utilizar consultas personalizadas em SQL. Convém observar que o tempo de resposta não é afetado apenas pelo processamento da requisição, mas também pela camada de transporte da rede.

CONCLUSÃO

Para desenvolvimento de aplicações web não foram necessárias apenas ferramentas adequadas, mas um cuidado no uso destas ferramentas. Ao escolher o que seria utilizado, também levou-se em conta a escolha da arquitetura e limitações da aplicação.

Ruby on Rails traz um conjunto de ferramentas modular para fazer aplicações web. Dentro do Ruby on Rails existe uma frase que diz: Convenção acima de configuração. E isso diz muito sobre a ferramenta, que diante a dilemas técnicos, escolhe o que é mais utilizado pela comunidade de desenvolvedores. E as ferramentas são criadas com o Ruby on Rails como base. Assim a ferramenta é sempre coesa, ou seja, integra muito bem seus diferentes módulos, e a cada atualização traz nova soluções para novos problemas. No protótipo, a maioria dos desafios técnicos possuem soluções abertas e de graça, assim foi possível utilizar essas soluções e para que o foco fosse a lógica de negócio, ou seja, os serviços que impactam os usuário, as funcionalidades, e o Ruby on Rails trouxe um pacote de soluções que se encaixou com a proposta e nos permitiu utilizar gratuitamente sua pilha de tecnologias abertas e de alto desempenho.

Para aplicações web também não basta apenas performance e uma boa arquitetura, mas também escalabilidade, isto é, a capacidade da aplicação de se adequar a quantidade maior de trafego. E o Ruby on Rails traz consigo a capacidade adaptar-se aos maiores servidores HTTP do mercado atualmente como o Puma (padrão), Apache2, NGIX, Unicorn e outros. Na aplicação, por conta de sua natureza, a escolha foi NGIX, pois ele traz facilidades de configuração, um padrão de alto desempenho e segurança.

O Rails por padrão utiliza o Puma, mas um ponto fraco deste são requisições longas. No caso dos dispositivos móveis, é muito comum, pois há lugares onde o sinal de telefonia não é estável o suficiente. Por isso a escolha na nossa aplicação foi NGIX receber as requisições, e quando completas, passar para o puma, assim é possível utilizar da velocidade e o gerenciamento de requisições do NGIX, e a paralelização de trabalhos do Puma.

O Ruby não traz uma grande solução para trabalhar com concorrência. A MRI, máquina virtual padrão do Ruby, lida com concorrência com uma trava global que para aplicações de alta performance é prejudicial, considerando que este tipo de aplicação depende de entrada e saído do dispositivo. Rails traz bibliotecas que resolvem este problema através de processamento paralelo. Hoje em dia, máquinas/servidores são multinúcleo e assim o processamento das requisições ocorrem em diferentes núcleos. Assim a aplicação estará preparada para escalar horizontalmente (utilizar a estratégia de réplicas para aumentar a capacidade de fluxo) e lidar com concorrência de um grande número de requisições em um curto espaço de tempo.

Portanto Ruby on Rails traz não somente um pacote completo de desenvolvimento, mas sim um conjunto de ferramentas amplamente utilizado e atualizado com as demandas atuais de performance e tecnologia.

REFERÊNCIAS

AKITA, Fabio. **The Ruby Community and Reputation**. Disponível em: http://www.akitaonrails.com/2016/08/19/the-ruby-community-and-reputation. Data de acesso: 13 de jul. de 2017.

FIELDING, Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. Dissertação - UNIVERSITY OF CALIFORNIA, IRVINE. 2000

GAMMA (1), Erich. HELM (1), Richard. JOHNSON (1), Ralph. VLISSIDES (1), John. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley. 1994. EUA.

Google Maps For Rails, Open-source project on GitHub. Disponível em: https://github.com/apneadiving/Google-Maps-for-Rails. Acesso em: 29 de ago de 2016.

HARTL, Michael. **RUBY ON RAILS TUTORIAL (RAILS 5)**. Addison-Wesley. 2017.

OLIVEIRA, Diogo de. **Introdução e Histórico do Postgresql**. Disponível em: https://wiki.postgresql.org/wiki/Introdu%C3%A7%C3%A3o_e_Hist%C3%B3rico. Data de acesso: 14 de set. de 2016.

PostgreSQL. Disponível em http://www.postgresql.org. Data de acesso: 14 de set. de 2016.

Rails Guides. Disponível em: http://guides.rubyonrails.org. Data de acesso: 14 de set. de 2016.

Rbenv. Disponível em http://rbenv.org. Data de acesso: 14 de set de 2016.

Ruby On Rails. Disponível em http://rubyonrails.org. Data de acesso: 14 de set de 2016.

Ruby. Disponível em https://www.ruby-lang.org/pt**.** Data de acesso: 14 de set de 2016.

RVM. Disponível em: https://rvm.io. Data de acesso: 14 de set de 2016.

SAUDATE, Alexandre. **REST: Construa APIs inteligentes de maneira simples**. São Paulo: Casa do Código, 2014.

GLOSSÁRIO

AUXILIAR DE DOCENTE – Técnico contratado pelo Centro Paula Souza (Fatec) para auxiliar professores e cuidar das redes e computadores no laboratório de informática.

API - API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos".

HTTP - protocolo HTTP (HyperText Transfer Protocol - Protocolo de Transferência de Hipertexto) data de 1996, época em que os trabalhos conjuntos de Tim Berners-Lee, Roy Fielding e Henrik Frystyk Nielsen levaram à publicação de uma RFC (Request for Comments) descrevendo este protocolo. Trata-se de um protocolo de camada de aplicação (segundo o modelo OSI) e, portanto, de relativa facilidade de manipulação em aplicações.

Este protocolo foi desenvolvido de maneira a ser o mais flexível possível para comportar diversas necessidades diferentes. Em linhas gerais, este protocolo segue o seguinte formato de requisições:

<método><URL> HTTP/<versão>

<Cabeçalhos - Sempre vários, um em cada linha>

<corpo da requisição>. SAUDATE (2014, p.14).

SQL - Structured Query Language, ou Linguagem de Consulta Estruturada ou SQL, é a linguagem de pesquisa declarativa padrão para banco de dados relacional (base de dados relacional).

URL - URL significa Universal Resource Locator e URI, Universal Resource Identifier . Uma URI, como diz o próprio nome, pode ser utilizada para identificar qualquer item – dar um caminho para um determinado conteúdo, dar nome a este, etc. Já uma URL pode ser utilizada apenas para fornecer caminhos – sendo que uma URL é, portanto, uma forma de uma URI. SAUDATE (2014, p. 5).

WEB SERVICE - Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes.

REACT NATIVE - O React Native é um projeto desenvolvido pelos engenheiros do Facebook e que consiste em uma série de ferramentas que viabilizam a criação de aplicações móveis nativas para a plataforma iOS e Android, utilizando o que há de mais moderno no desenvolvimento Front-end – mirando no futuro. É o estado da arte no que se refere ao desenvolvimento mobile baseado em JavaScript. O stack do React Native é poderoso, pois nos permite utilizar ECMAScript 6, CSS Flexbox, JSX, diversos pacotes do NPM e muito mais. Sem contar que nos permite fazer debug na mesma IDE utilizada para o desenvolvimento nativo com essas plataformas (além de tornar o processo extremamente divertido).