FACULDADE DE TECNOLOGIA DE SOROCABA
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO FINAL

Jean Vitor Antunes Peixoto

PetAppy – Aplicativo para proteção dos animais

Sorocaba
Julho/2017

Fatec Sorocaba José Crespo Gonzales

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

PetAppy – Aplicativo para proteção dos animais

Jean Vitor Antunes Peixoto

Orientador: Prof. Dr. Paulo Edson Alves Filho

Sorocaba

Julho/2017

RESUMO

Esta pesquisa resultou em parte de uma aplicação de fácil acesso e alta disponibilidade que une a comunidade de pessoas interessadas na causa de animais abandonados e que tem como propósito principal ajudar a encontrar animais perdidos e disponibilizar uma plataforma capaz de unificar os anúncios e o resgate de animais que estão sem lar definitivo.

O projeto resultante baseia-se em padrões Web *Model-ViewController* (MVC) e alguns frameworks para o desenvolvimento Mobile tais como o de JavaScript com *React-Native* e servidor com *Ruby on Rails*. O trabalho ainda expõe conceitos úteis sobre configurações de arquivos, necessários para o entendimento da estrutura e do desenvolvimento do mesmo.

Resumo não tem parágrafos

Palavras-Chave*:* Android, React Native, usabilidade, MVC*.*

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

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

BD - Banco de Dados.

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

LISTA DE FIGURAS

1 INTRODUÇÃO E JUSTIFICATIVA

Atualmente existe um grande número de animais perdidos nas cidades do país, o que causa grande comoção social e o surgimento de diversas entidades de proteção animal.

Para suprir essa deficiência, propomos o aplicativo Pet Appy, que aborda anúncios de animais perdidos e sua relação com os anúncios de animais que foram achados. O aplicativo também oferece um sistema que facilita a adoção através de lares temporários.

Há uma grande comoção social com a causa animal percebida em várias comunidades engajadas com o movimento. Existem vários protetores espalhados pela nossa região e nenhum local onde se possa encontrar essas pessoas de forma organizada ou bem estruturada, normalmente um caminho para interação e localização entre elas são as redes sociais, utilizando algum grupo no Facebook como constatado durantes pesquisas feitas durante esta iniciação científica.

Após conversar com protetores de animais de nossa região detectamos necessidades ainda não solucionadas. Entre elas, o desencontro de informação que ocorre no anúncio de animais perdidos e avisos de animais encontrados. Existem muitas páginas nas redes sociais que tem como foco este assunto, entretanto muitas vezes as informações se desencontram, como o caso comum de animais que já foram encontrados e páginas que continuam replicando o anúncio como se ainda estivessem perdidos.

Outro problema é a grande quantidade de animais abandonados ou de rua que precisam de cuidados e nem sempre encontram ajuda. Há pessoas dispostas a cuidar deles e, por vezes, nunca viram uma oportunidade de adotar um animal. Apesar de não ser tratado nesta versão do aplicativo, uma funcionalidade como a descrita seria de grande potencial para uma versão futura.

Para fins de elaboração do aplicativo, inicialmente foram levantadas situações possíveis e relevantes a serem testadas, tais como os casos de

pessoas querendo adotar animais, pessoas que perderam algum animal ou pessoas que encontraram um animal e estão à procura de seu devido dono.

Sendo assim inicia-se a estruturação do projeto através do padrão modelo, visão e controle. O modelo de três camadas físicas divide um aplicativo de modo que a lógica de negócio resida no meio das três camadas. Isto é chamado de camada física intermediária ou camada física de negócios. A maior parte do código escrito reside na camada de apresentação e de negócio. Assim, nos baseamos em um padrão de projeto chamado MVC (Modelo Visão Controle). “O conceito de dividir uma aplicação em camadas que já está bastante difundido entre os profissionais que trabalham no desenvolvimento de software. Basicamente as camadas do MVC estão separadas em: apresentação, lógica da aplicação e gestão dos recursos” (ALUR: 2002).

2 REVISÃO BIBLIOGRÁFICA

Para a realização desta pesquisa, tomamos como referencial a engenharia de software, que “é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais" (BAUER: 1968). O autor lista itens essenciais que toda aplicação deveria conter em sua estrutura, assim como toda aplicação deve ser baseada em componentes. De acordo com PRESSMAN (2002),

> “a prática da tecnologia de software baseado em componentes, baseia-se no desenvolvimento de componentes reutilizáveis, levando a redução de tempo de desenvolvimento, facilitando as mudanças e a implantação de novas funcionalidades. Dessa forma, o processo de engenharia de software baseada em componentes tem mudado o modo pelo qual sistemas são desenvolvidos, pois desloca a ênfase da programação do software para a composição de sistema de software com base em componentes” (PRESSMAN, 2002, p. 257).

Sendo assim, o projeto é divido em vários componentes específicos de acordo com cada funcionalidade.

Outro item essencial destacado pelo autor é o reuso de código,

> “objetivo principal da engenharia de software baseada em componentes. Não se trata somente de reutilização de código, pois abrange também os artefatos envolvidos durante o todas as etapas de desenvolvimento. Com isso os riscos são menores ao usar um componente já existente em relação ao desenvolvimento de algo a partir do zero. Também ocorre o aumento da produtividade, tendo em vista a redução de esforços pela equipe de desenvolvimento. Seguindo a ideia, “Crie uma vez, use onde quiser. Dessa forma, a qualidade e confiabilidade do produto são maiores, pois o componente reutilizado já foi testado e aprovado”. (PRESSMAN: 2002, p. 316) [1]

Dessa forma, definimos que as funcionalidades repetidas possuirão uma ação equivalente assim o sistema não fica com uma quantidade grande de código repetido. Entre outras boas práticas descritas estão os padrões de projeto que são um grupo de práticas para definir o problema, a solução e em qual situação aplicar esta solução e suas consequências no desenvolvimento de software. Os padrões são desenvolvidos a partir de experiências práticas em projetos reais.

2 OBJETIVOS

O objetivo do projeto Pet Appy é realizar a união entre os grupos, entidades e apoiadores à proteção de animais da cidade, assim como possibilitar uma forma eficiente de encontrar seu animal perdido a partir de informações do aplicativo, desenvolvido para as plataformas mais utilizadas do mercado, tais como Android e IOS.

Logo o objetivo específico da pesquisa em questão foi a criação de uma sequência de passos necessários (Identificação das informações mais relevantes, Definição da estrutura analítica de projetos, Delimitação de papéis e responsabilidades, Criação um modelo que abranja a estrutura analítica, Simulação e validação do app) para a conclusão do projeto, através da aplicação do modelo MVC e desenvolvido a primeira versão do aplicativo.

3 MATERIAIS E MÉTODOS

Para iniciar esse projeto foram utilizados protótipos para o desenvolvimento para uma inicial simulação da aplicação final.

Levantamos as necessidades da comunidade que motivam o projeto através de entrevistas e diálogos, assim como problemas enfrentados pelas pessoas envolvidas na causa animal para termos uma avaliação inicial de como a aplicação proposta pode atendê-los.

Para estruturação do projeto foram seguidos passos assim detalhados:

a) Identificação das informações mais relevantes. Para criar um modelo de projeto é necessário definir quais são as informações importantes que precisam ser cadastradas a cada novo trabalho. Boa parte dessas informações é utilizada no termo de abertura do projeto.

As informações necessárias para iniciarmos o projeto foram: usuário alvo, alcance da solução, custo e beneficio. A validação dessas informações foi realizada com base em pesquisas de campo e conversas com pessoas envolvidas na causa.

b) Definição da estrutura analítica de projetos: Nessa etapa são definidas as macro-fases do projeto e as atividades que precisam ser executadas.

As macro-fases do petAppy foram: Criação da ideia, Pesquisa de campo, Primeiro esboço do app, FeedBack do primeiro esboço de alguns usuários, Desenvolvimento, Testes, Lançamento do APP.

Dentro da fase de desenvolvimento analisamos e escolhemos como padrão e estrutura MVC de projeto. Segundo esse padrão, na arquitetura MVC o modelo representa os dados da aplicação e as regras do negócio que governam o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo.

Um componente de visualização renderiza o conteúdo de uma parte particular do modelo e encaminha para o controlador as ações do usuário; acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados.

Um controlador define o comportamento da aplicação é ele que interpreta as ações do usuário e as mapeia para chamadas do modelo. Em um cliente de aplicações Web essas ações do usuário poderiam ser cliques de botões ou seleções de menus. As ações realizadas pelo modelo incluem ativar processos de negócio ou alterar o estado do modelo. Com base na ação do usuário e no resultado do processamento do modelo, o controlador seleciona uma visualização a ser exibida como parte da resposta a solicitação do usuário. Há normalmente um controlador para cada conjunto de funcionalidades relacionadas.

c) Delimitação de papéis e responsabilidades que envolvem cada atividade, recursos definidos a fim de concluir sua entrega adequadamente.

Assim nossas responsabilidades ficaram divididas em três partes: Estrutura do projeto, Experiência do usuário e o Desenvolvimento no servidor.

d) Criação um modelo que abranja a estrutura analítica e delimite a duração e a dependência envolvidas no processo. Um passo posterior e é salvar essa estrutura como modelo.

Foi criado um fluxo de funcionamento como base modelo onde abrange toda estrutura e suas dependências.

e) Simulação e validação, quando se testa a lógica utilizada na construção do projeto efetivamente. Simula-se um projeto real para que se verifique se todas as atividades e elementos envolvidos estão funcionando corretamente.

Após o termino do desenvolvimento é disponibilizado uma versão Beta na loja de aplicados do IOS e Android para um grupo de usuários escolhido.

3.1 Etapas

Conforme ajustado entre a equipe de desenvolvimento do projeto, as etapas da pesquisa são as seguintes:

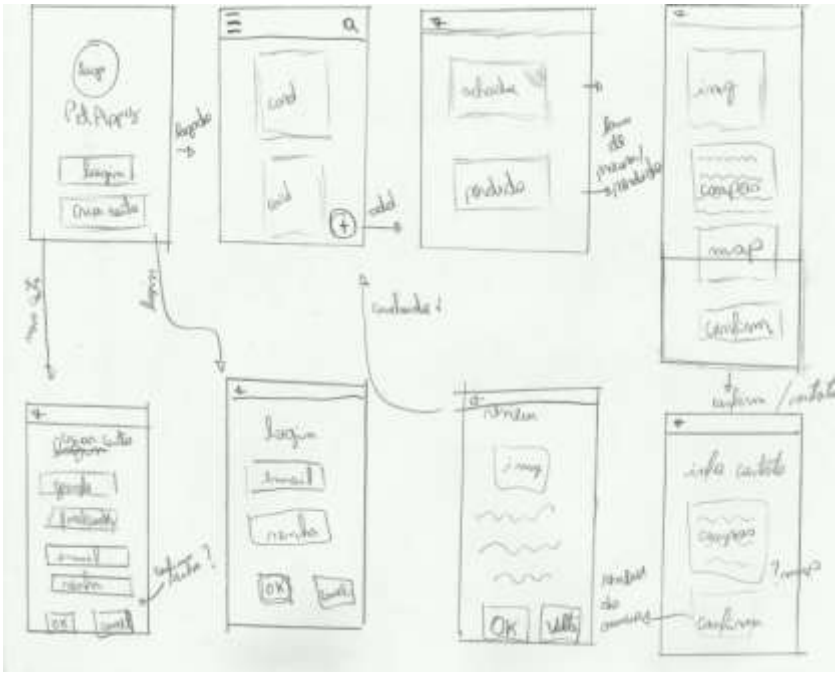
1. Primeira etapa: revisão bibliográfica, concluída;
2. Segunda etapa: pesquisa para entender a situação atual, suas principais falhas e como resolve-las, concluída;
3. Terceira etapa: Escolher uma estrutura e padrão de projeto a ser seguida, concluída.
4. Quarta etapa: criação do primeiro design do aplicativo logo em sua primeira versão, gerando assim uma nova versão de cada modificação realizada, concluída.
5. Arquitetura do front end da aplicação que consumirá a API desenvolvida em Rails.
6. Desenvolvimento da API desenvolvida em Rails, que conterá o sistema em si e armazenará os dados da aplicação.

4 RESULTADOS E DISCUSSÕES

4.1.1 Elaboração do modelo MVC do sistema

Para esta pesquisa, adotamos a metodologia Design Sprint, metodologia ágil focada na experiência do usuário, para criar o primeiro protótipo e assim definir o fluxo de funcionamento da aplicação, conforme a figura 1.

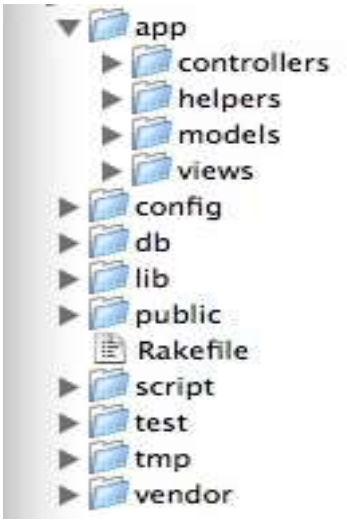
Figura 1: Modelo do aplicativo desenho no papel.



Fonte: PetAppy(2017)

Após a finalização do desse fluxo de funcionamento são criadas as regras de negócios que são as funcionalidades e regras especificas para a execução correta do fluxo definido na figura 1 começaram a ser desenvolvido através da linguagem Ruby, através do framework Ruby On Rails. Primeiramente foi criada a estrutura dos arquivos do projeto para que as regras fossem implementas, conforme a figura 2.

Figura 2: Estruturas de arquivo do projeto.



Fonte: PetAppy(2017)

Toda essa estrutura criada é equivalente ao C do MVC, ou seja a controladora. Ela acessa o modelo através do código e do postgresql.

O ultimo elemento do MVC, o V (View). é a parte visual do aplicativo ou seja, a interface na qual o usuário irá interagir. Foi desenvolvida através do React Native, framework feito com javascript, que possibilita o desenvolvimento mobile para ambas plataformas do mercado (Android e Iphone). Como resultado desse ciclo obtivemos a primeiro versão do aplicativo.

A primeira tela exibida ao iniciar a aplicação é a visualizada na figura 4, na qual o usuário tem somente dois caminhos possíveis: entrar na aplicação com sua conta já existente clicando no botão “Login” ou criar uma nova conta ao clicar no botão “Criar Conta”.

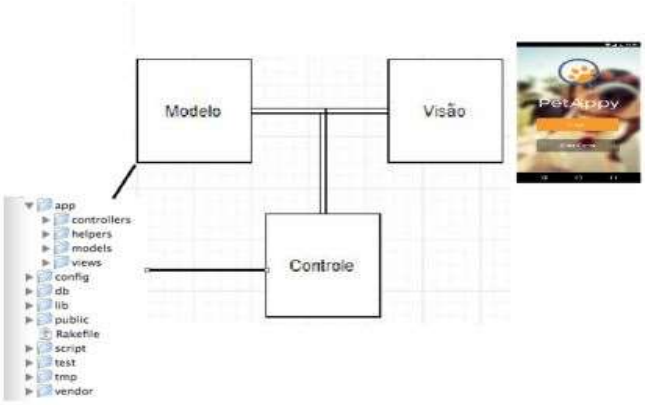
Figura 3: Tela inicial do aplicativo.

Fonte: PetAppy(2017)

Caso o usuário efetue o clique no botão “Entrar”, será encaminhado para está próxima tela na qual deve preencher os dois campos obrigatórios: e-mail e senha e em seguida clicar no botão entrar. Caso desejar voltar para a tela anterior deve clicar no botão cancelar.

Como modelo final do MVC obtivemos o resultado conforme a figura 4.

Figura 4: Modelo MVC.



Fonte: PetAppy(2017)

5. CONCLUSÃO

A partir de nossas percepções iniciais e levantamentos feitos nesta fase de pesquisa, a parte desenvolvida neste trabalho foram as telas da aplicação e o fluxo de funcionamento que o usuário deve seguir para concluir as ações do aplicativo.

Foi desenvolvida uma sequência de passos necessários para a conclusão do projeto, assim todo desenvolvimento foi baseado nos métodos de padronização do projeto, com a aplicação do modelo MVC.

Ao final, obtivemos um projeto totalmente padronizado na arquitetura MVC assim construímos um sistema para dispositivos móveis chamado PetAppy que foi desenvolvido para as plataformas Android e iPhone, com a principal função de anunciar um animal que foi perdido ou achado juntamente com as informações do animal, exibir os anúncios mais próximos do usuário de acordo com sua localização e prover uma plataforma unificada para protetores de animais de nossa região.

5.1 Problemas encontrados

Nosso projeto pretendeu realizar a integração e a troca de informações entre pessoas envolvidas com proteção animal na cidade. Entre as funcionalidades propostas em nosso projeto estão: unificar avisos, anúncios e pedidos de resgate de animais em perigo em redes sociais, além de ter a possibilidade de ser um utilitário que auxilie no mapeamento, dentro da comunidade, de lares temporários ou definitivos para animais que necessitem.

Quanto aos problemas específicos do Pet Appy, podemos citar a falta de um padrão nos projetos do desenvolvimento de software no mercado, o que torna a manutenção e expansão do projeto complexas e possivelmente inviáveis.

Dessa forma, o padrão de estrutura que seguimos para o desenvolvimento do projeto, dentre os existentes, foram os sete passos já descritos de acordo com o MVC (Modelo Visualização Controle).

Ele foi escolhido, pois forneceu uma maneira de dividir as funcionalidades envolvidas e sua apresentação com os dados para o usuário. Ou seja primeiramente o usuário terá o primeiro contato com a visão, uma tela de cadastro por exemplo. Após preencher os campos com os dados, esses dados serão enviados para o servidor e assim será executada uma funcionalidade de salvar esses novos dados e após a finalização dessa funcionalidade era para o banco de dados. Esse fluxo bem definido e separado em camadas é válida graças a utilização do padrão MVC de projeto.

REFERÊNCIAS

ALUR, D; CRUPI, J.; MALKS, D. “**Core j2ee patterns: as melhores práticas e estratégias de design**”. Rio de Janeiro: Campus, 2002.

BAUER. **The 1968/69 NATO Software Engineering Reports. Dagstuhl-Seminar 9635: "History of Software Engineering" Schloss Dagstuhl**, August 26 -30,1996.Disponivel em: <homepages.cs.ncl.ac.uk/brian.randell/NATO/

CSS, Mozilla Developer. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 16.Set.2016.

DUCKETT, Jon. HTML & CSS: **Design and Build Web Sites**, Wiley, Edição 1, 2011.

HTML, Mozilla Developer. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 14.Set.2016.

Javascript, Mozilla Developer. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 14.Set.2016.

10 Usability Heuristics for User Interface Design. Disponivel em <https://www.nngroup.com/articles/ten-usability-heuristics/>. Acesso em 18.Out.2016

Native, React Disponível em: <https://tableless.com.br/react-native-construa-aplicacoes-moveis-nativas-com-javascript/> Acesso em: 14.Set.2016.

NATO Reports. Acesso em: 21 jun. 2017.

NIELSEN, J.; LORANGER, H. **Web Usability**. 1a ed.: Elsevier, 2006.

NORMAN, DONALD. **The Design of Everyday Things**. 1a ed.: Basic Books; Rev Exp edition, 2013.

PRESSMAN, ROGER S. **Engenharia de Software**. 5ª Ed. Rio de Janeiro: McGraw- Hill, 2002.

Ruby. Disponível em https://www.ruby-lang.org/pt/ - Acesso em: 14.Set.2016.

Ruby On Rails. Disponível em http://rubyonrails.org/ - Acesso em: 14.Set.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".

REACT - Originalmente desenvolvido e, posteriormente, de código aberto pelo Facebook, React é um framework JavaScript do lado do cliente para construir interfaces de usuário. O React usa uma sintaxe declarativa e uma extensão JavaScript chamada JSX para descrever os layouts HTML. Cada componente React é suportado e configurado por propriedades e estado, as alterações desencadeiam atualizações por meio de um fluxo de dados unidirecional. Estas atualizações são otimizadas através de um DOM virtual, que compara os componentes para assegurar que somente aqueles alterados pela mudança de estado são atualizados.

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).

ECMA SCRIPT 6 - é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador,

realizando comunicação assíncrona e alterando o conteúdo do documento exibido.

PHOTOSHOP CS6 - Utilizado para desenhar as telas do protótipo inicial do aplicativo, Adobe Photoshop é um software caracterizado como editor de imagens bidimensionais do tipo raster (possuindo ainda algumas capacidades de edição típicas dos editores vectoriais) desenvolvido pela Adobe Systems. É considerado o líder no mercado dos editores de imagem profissionais, assim como o programa de facto para edição profissional de imagens digitais e trabalhos de pré-impressão.

EXPERIENCE DESIGN - Experience Design CC é o primeiro programa da Adobe voltado para designers de interface UI e user experience designers UX, sendo utilizado para o desenho de telas para diversos dispositivos facilitando a visualização de todas as formas e prototipação navegável entre as telas já criadas no mesmo aplicativo.