

PETVitae: Ferramenta de Auxílio ao Acompanhamento e Gestão de Grupos PET

Fernando H. F. Leite¹, Kelly S. Costa¹, Micael M. Gomes¹, Antonio M. Pinto¹,
Gabriel P. C. Freitas¹, Geraldo Braz Junior¹, Jádriel C. S. Junior¹, Jéssica C.
Mendes¹, Yandson de J. S. Costa¹

¹Programa de Educação Tutorial do curso de Ciência da Computação
Departamento de Informática - Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses, 1966, Campus do Bacanga – 65065-545 – São Luís – MA – Brasil

{fernandodojo, kellyberreca, micaellgoms, antoniomorara}@gmail.com

{gpcfreitas, jadielcsjr, jesscostaj4, yandson.jesus}@gmail.com,

gerald@nca.ufma.br

Abstract. *With the continuous development of the Tutorial Education Program in the course of Computing at UFMA, demands arose, resulting in the creation and use of third-party web systems with the intention of managing administrative, financial and organizational needs, such as division of tasks and responsibilities for each member, as well as the management of activities carried out by the group itself. In this context, this paper presents the proposal PETVitae, a tool that aims to unify the management of the main demands in the organization of a PET group.*

Resumo. *Com o contínuo desenvolvimento do Programa de Educação Tutorial no curso de Computação na UFMA, demandas foram surgindo, resultando na criação e utilização de sistemas web de terceiros com a intenção de gerenciar as necessidades administrativas, financeiras e organizacionais. Como exemplo, a divisão de tarefas e responsabilidades para cada integrante, e gerenciamento de atividades desempenhadas pelo próprio grupo. Neste contexto este trabalho apresenta a proposta PET Vitae, uma ferramenta que pretende unificar o gerenciamento das principais demandas na organização de um grupo PET.*

1. Introdução

O PET, Programa de Educação Tutorial, desenvolvido em grupos organizados a partir de cursos da graduação nas instituições de ensino superior é orientado pelo princípio da indissociabilidade entre ensino, pesquisa e extensão [MEC 2018], aplicando o mesmo na melhoria do seu curso. Como fruto desse comprometimento o grupo PET de Computação, PETComp, vem se desenvolvendo, crescendo e, para acompanhar essa expansão, diversas aplicações foram produzidas, tendo como objetivo suprir necessidades organizacionais, por exemplo, o *SafeBox* [Marcos-Costa et al. 2018], que é uma aplicação de controle financeiro. Além disso, serviços de terceiros também são adotados para controlar determinadas tarefas, como *Google Sheets* e *Trello*, onde é necessário um investimento de tempo por parte dos integrantes para utilizar as ferramentas.

Neste contexto de diversos sistemas e surgimentos de novas demandas, atrelados ao constante crescimento do grupo, cria-se a cultura de pensar em soluções para necessidades específicas, seja com desenvolvimento ou na busca de programas já consolidados. No entanto, esses sistemas geram uma demanda de tempo e organização, já que um petiano sempre precisará manter um controle de usuários e senhas, considerando a organização e estrutura de cada ferramenta. Desta forma, a aplicação PET Vitae tem como proposta unificar as funcionalidades de diversos sistemas produzidos pelo grupo, assim como possibilitar à criação e integração de novas funções.

No intuito de atingir o maior número de possíveis usuários, além de solucionar problemas como mobilidade e facilidade de uso conforme literatura [Deloitte 2018], optou-se pela adoção de dispositivos móveis como principal plataforma. Como a plataforma móvel disponibiliza diversos sistemas operacionais, buscou-se também uma tecnologia que viesse otimizar o processo de construção de aplicações multiplataformas, reduzindo o tempo de desenvolvimento sem perdas de usabilidade e desempenho significativas.

2. Fundamentação Teórica

Esta seção apresenta, alguns temas de significativa importância à elaboração deste trabalho. Primeiramente, serão abordadas as principais tecnologia e ferramentas para desenvolvimento de aplicações multiplataformas em contraste com o *React Native* que é a proposta presente neste trabalho. Posteriormente serão apresentados alguns conceitos e características sobre a arquitetura microsserviços.

2.1. Desenvolvimento Multiplataforma com *React Native*

Nos últimos anos, diferentes técnicas e *frameworks* surgiram como alternativa para a criação de aplicações multiplataformas, ou seja, aplicativo que em um único processo de desenvolvimento se consiga acesso de suas funcionalidades a partir de diversos sistemas operacionais. Sem o comprometimento do desempenho e usabilidade, ou a necessidade de desenvolver aplicações específicas para cada sistema operativo.

Uma das soluções mais comuns atualmente é a criação de *WebApps*, ou aplicações híbridas, que embora hajam pequenas diferenças no processo de desenvolvimento e até apresentem menor custo de desenvolvimento comparado com uma aplicação nativa, estes ainda não conseguem oferecer uma experiência equivalente devida à alta carga de renderização de componentes.

Já o *React Native* é baseado no *React*, um *frameworks JavaScript* desenvolvido pelo *Facebook* em 2013 [Udacity 2018], com foco na construção de *interfaces*. Tem como objetivo principal simplificar o processo de desenvolvimento de aplicações iOS e Android criando apenas uma aplicação em vez de duas específicas a cada plataforma. Sempre utilizando a mesma linguagem *JavaScript*, e o mesmo código-fonte, embora os componentes visuais possam divergir em um mesmo aplicativo por particularidades de cada plataforma.

O que difere completamente o desenvolvimento com *React Native* das outras técnicas e *frameworks* é não haver a necessidade de uma camada de renderização baseado em *Web*, utilizando os próprios componentes internos do sistema para renderização. O que garante menor custo de processamento ao dispositivo, melhor integração com o

sistema operacional e, principalmente, melhor desempenho comparada às outras tecnologias de desenvolvimento multiplataforma, alcançando semelhança com aplicações nativas [SIMPLY 2018].

2.2. Microserviços

A arquitetura microserviços é uma abordagem para sistemas distribuídos que promovem o uso de serviços finamente granulados com seus próprios ciclos de vida, que colaboram em conjunto [Newman 2015]. O objetivo principal da arquitetura de microserviços surgiu com a necessidade de otimização e descentralização de serviços e códigos, com uma menor interdependência, visando agilizar o processo de desenvolvimento, facilitando a manutenção e integração entre diferentes módulos.

Um microserviço é uma unidade de *software* autônoma que, junto de outros, compõem uma grande aplicação, cada parte podendo ser independentemente implantada e escalada. Pode ser escrita por diferentes equipes de desenvolvimento, e em diversas linguagens de programação e até testada individualmente [Stoiber 2017]. Diferente das aplicações monolíticas que são projetadas sem uma modularização externa, os microserviços permitem que uma aplicação ou serviços possa se integrar a outro serviço ainda de forma independente de modo a compor algo maior.

3. Metodologia

O aplicativo neste projeto tem como proposta a divisão em quatro etapas. A primeira etapa é denominada “Levantamento de Módulos”. Nesta etapa propõe-se a criação de uma documentação pela parte interessada ou *stakeholder*, descrevendo o problema que o aplicativo está se propondo a resolver, suas funcionalidades, objetivos e necessidades para seu funcionamento. Diversas versões desta documentação devem ser criadas e entre cada versão o *stakeholder* tem por obrigação realizar a validação (processo de verificação de resultados), assim como o registro das modificações a serem realizadas entre cada versão, com o intuito de garantir total entendimento do projeto. A partir desta documentação inicial, deve-se prosseguir para a segunda etapa denominada “Definição de Arquitetura”.

Na segunda etapa, propõe-se a extrair alguns tópicos importantes da documentação inicial com o intuito de auxiliar a equipe de desenvolvimento no levantamento de requisitos funcionais e não funcionais, e na diagramação, até ocorrer a validação conforme os critérios do *stakeholder*. O PetVitae tem como proposta utilizar a tecnologia *React Native* já que este se propõe a criar aplicações móveis multiplataforma com desempenho e usabilidade próximas a de aplicativo nativo, com baixo custo de desenvolvimento.

Além disso, a arquitetura de microserviços onde cada uma das funcionalidades/módulos do PETVitae será um serviço autônomo com comunicações e bancos de dados independentes entre si. Tornando a aplicação modular além de facilitar a criação e integração de novos módulos ao sistema, independente da tecnologia utilizada no desenvolvimento da nova funcionalidade, sem comprometer o funcionamento da aplicação.

Já o processo de desenvolvimento deverá ser pautado na abordagem de prototipação evolucionária conforme [Shaw and Garlan 1996], onde cada protótipo deve ser desenvolvido e evoluir cuidadosamente até que ocorra transformação na aplicação final. O *Scrum* como metodologia ágil para auxiliar e resolver problemas complexos e

adaptativos, enquanto entrega etapas do produto segundo [Wazlawick 2013]. O banco de dados idealizado é o *Firebase*, uma plataforma *web* que disponibiliza serviços e ferramentas para desenvolvimento de aplicativos *web* e *mobile* de acordo com [de Croos 2018].

O último ponto a ser tratado nesta etapa refere-se a comunicação seria realizada por *REST API* que farão requisições independentes ao banco de dados através de protocolos *HTTP*, que tem como resultado um arquivo de *JSON*.

A “Prototipação” consiste na terceira etapa, a qual tem como objetivo criar a primeira representação visual e funcional da aplicação. Nesta etapa o projeto se propõe a criar primeira identidade visual utilizando o *Figma*, uma ferramenta de prototipação *on-line* colaborativa, que possibilita interações entre *designers* e desenvolvedores através de comentários. Após criação do primeiro protótipo visual, este deverá ser submetido para validação pelo *stakeholder*, para confirmar o cumprimento dos requisitos. Em seguida, será desenvolvido o primeiro protótipo visual utilizando o *framework React Native*, contemplando todas as funcionalidades e módulos da aplicação, a prototipação compõe um ciclo junto com a validação, onde o desenvolvimento só seguirá para a próxima etapa caso a validação ocorra, caso contrário, a etapa da prototipação é refeita. Como mostrado na Figura 1.

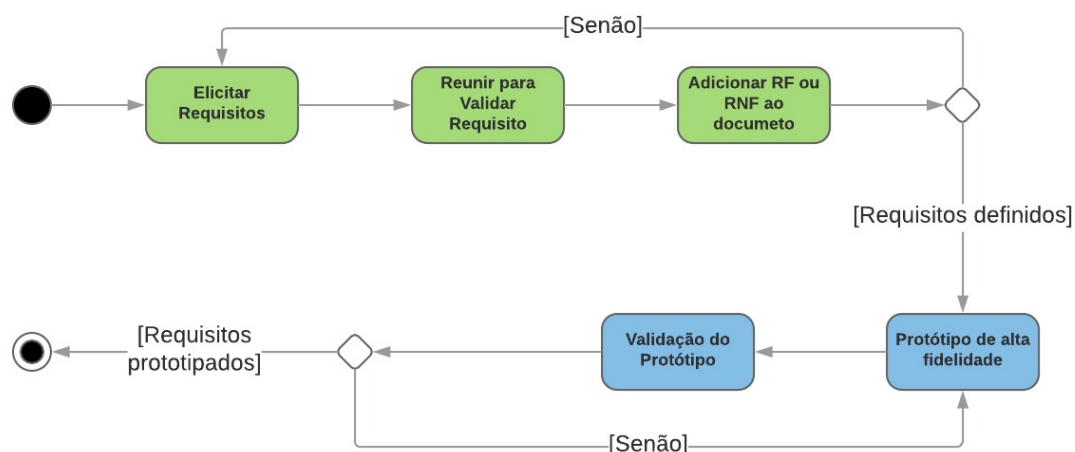


Figura 1. Processo de elicitação de requisitos seguido da prototipação com design participativo.

Na última etapa de “Implementação”, deverá ocorrer o desenvolvimento de fato da aplicação além de englobar a criação das *interfaces* de comunicação (APIs), e integração com o banco de dados fornecido pela *Firebase*. Entre cada pequena etapa da implementação serão realizadas baterias de testes assim como o processo de validação, para garantir total funcionamento e estabilidade da aplicação.

Esse processo foi aplicado para o desenvolvimento integrado dos módulos de gerenciamento de metas, financeiro, emissão de declarações e controle de atividades.

4. Resultado e Discussão

Após o desenvolvimento e aplicação em andamento do processo proposto, foi possível perceber a constante fragmentação do projeto, o que muitas das vezes não se permitia

entender exatamente o total do que se desejava realizar, ocasionando reestruturação, e constantes modificações, principalmente na primeira etapa de documentação e definição da arquitetura. Durante as etapas iniciais de desenvolvimento, verificaram-se problemas de incompatibilidade do *hardware* disponível com os emuladores utilizados para teste.

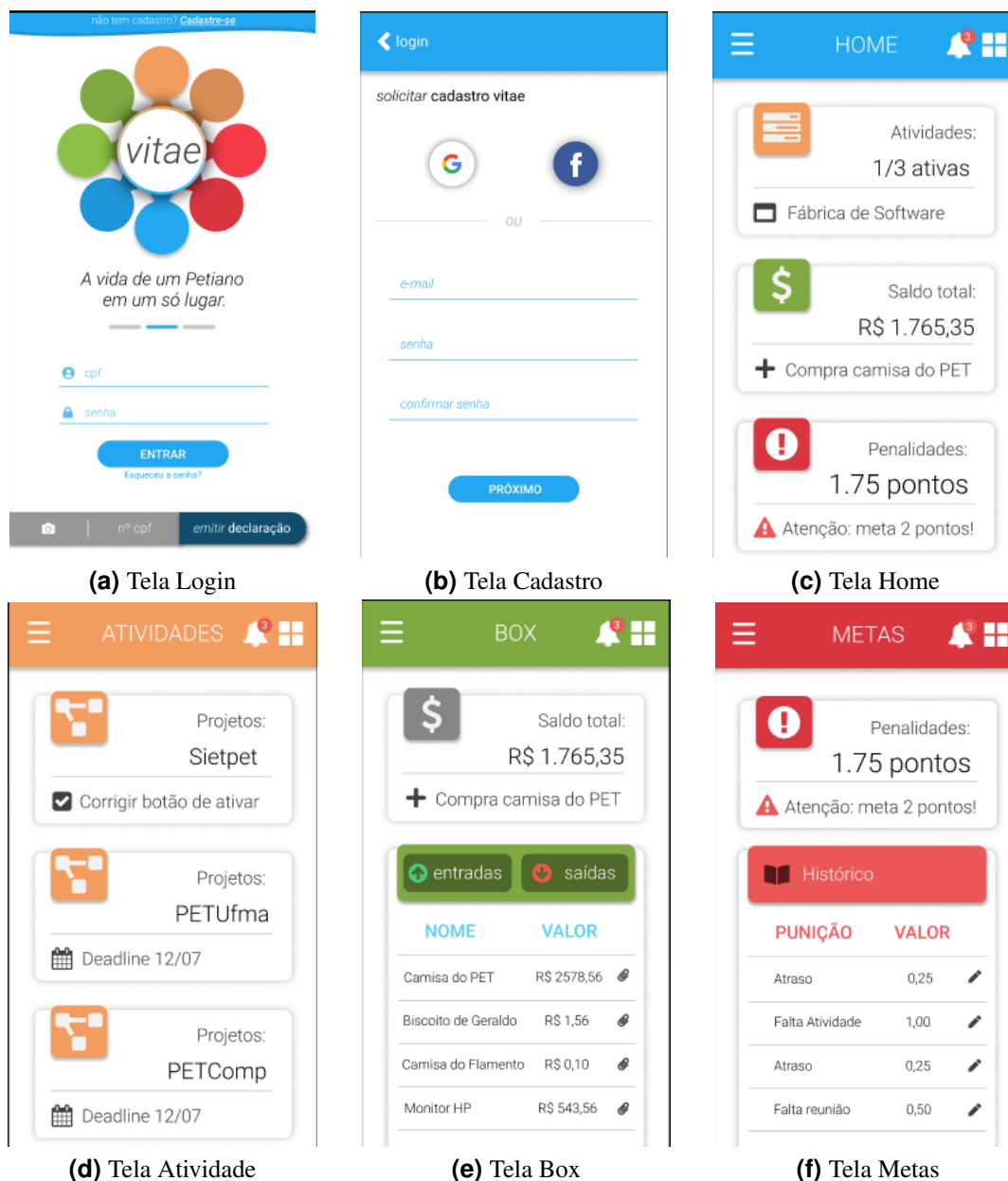


Figura 2. Protótipos de alta fidelidade gerados com Figma.

Na Figura 2, é possível observar os resultados da prototipação utilizando a ferramenta Figma, cada uma correspondente a uma das telas e suas respectivas funcionalidades. A Figura 2a e 2b correspondem a tela inicial ao abrir o aplicativo, onde é possível solicitar cadastro, realizar o *login* e emitir certificado de participação no PET, com informações referentes ao tempo de participação e atividades realizadas. A Figura 2c corresponde a tela principal do aplicativo após realizar o *login*. Nela é possível ter acesso a todos os outros módulos ou funcionalidades. Já as Figuras 2d, 2e, 2f são as princi-

país funcionalidades que o aplicativo se propõe. A “Tela Atividades” tem como objetivo gerenciar toda e qualquer atividade desempenhada por cada petiano, podendo designar tarefas, assim como acompanhar o progresso de cada uma. A “Tela Box” é responsável por gerenciar e manter histórico detalhado de todas as transações financeiras. Já a “Tela Metas” funcionará como um controle de rendimentos ao qual o petiano passará a ter pontos associados em casos de descumprimentos tarefas. Como foi observado, apensar de incompatibilidade entre *hardware* e emulador, ainda assim foi possível desenvolver protótipos de alta fidelidade.

5. Conclusão

O PetVitae é um aplicativo em desenvolvimento, seguindo as diretrizes do modelo ágil *Scrum*. O projeto continua em desenvolvimento seguindo as seguintes etapas: Levantamento de Módulos; Definição de Arquitetura; Prototipação e Implementação. Atualmente o processo de desenvolvimento encontra-se na etapa de prototipação com toda a *interface* gráfica concluída. Como *Sprints* a serem concluídas tem-se a etapa de prototipação e validação onde ocorrem os testes para certificação de atendimento dos requisitos assim como o pleno funcionamento das funcionalidades, que após validação será integrado à aplicação final. Ao término da implementação, a integração com servidor externo, onde será realizado a integração do aplicativo assim como testes, e por último a aplicação final e então encerramento do projeto.

Após a conclusão do projeto, o código será disponibilizado publicamente na plataforma de compartilhamento de *softwares* de código livre, o *GitHub*, possibilitando assim a utilização e modificação livremente por terceiros, permitindo a portabilidade e readaptação do projeto às necessidades dos diferentes grupos PETs. Espera-se que a aplicação atenda a todos os requisitos, com desempenho próximo ao de uma aplicação nativa *Android*, aplicando com sucesso a arquitetura de microsserviços.

Referências

- de Croos, P. (2018). When You Should (and Shouldn't) Use Firebase — Codementor.
- Deloitte (2018). Global Mobile Consumer Survey 2018.
- Marcos-Costa, Antoniompw, and cmanfeed (2018). GitHub - Marcos-Costa/safebox: Um sistema de gerenciamento ...
- MEC (2018). Pet - ministério da educação - portal mec.
- Newman, S. (2015). *Building Microservices, Designing Fine-Grained Systems*. O'Reilly Media, 1st edition.
- Shaw, M. and Garlan, D. (1996). Software architecture: perspectives on an emerging discipline. In *Advances in Computer Science*. Prentice Hall.
- SIMPLY (2018). LET'S GET CLEAR ABOUT REACT NATIVE.
- Stoiber (2017). Build your first Node.js microservice.
- Udacity (2018). React: o que é e como funciona essa ferramenta? - udacity.
- Wazlawick, R. (2013). Conceitos e práticas. In *Engenharia de Software*. Elsevier Brasil.