A B A K Ó S
PUC Minas Instituto de Ciências Exatas e Informática

Gerenciamento dos Processos de Controle de Contribuições de Paróquias Utilizando Sistema *Web*[*]

Web-based Control of Contributions to Parishes

Zamur Borges Vedovatto[1]
Cinthia Mikaela de Souza[2]
Magali Rezende Gouvêa Meireles[3]

Resumo

As paróquias são locais para reuniões de fiéis religiosos e, nos processos organizacionais, existem práticas de controle financeiro de custos e recebimentos. Em certas paróquias, o registro dos recebimentos é feito em folhas de papel durante as missas para, posteriormente, ser transferido para planilhas informatizadas, gerando o risco de perda de informação no processo. Dentre os principais recebimentos, encontram-se as contribuições, seja em forma de pagamento de dízimo ou de doação. O presente trabalho apresenta o desenvolvimento de um sistema *Web* para comunidades paroquiais aperfeiçoarem o controle do recebimento de contribuições. O objetivo do trabalho foi implantar o processo de transformação digital em paróquias que possuem os seus processos manuais. Foi desenvolvida uma solução que contemplasse as reais necessidades do público-alvo, que buscam por um processo mais seguro e eficaz para o registro das contribuições. O MEAN *Stack*, utilizado no processo, abrange todas as tecnologias utilizadas para o desenvolvimento. O *design* seguiu os conceitos de *User Experience* (UX), visando uma melhor experiência de uso da aplicação em termos de fluidez e facilidade em realizar as operações. Constatou-se, no processo de validação, que o sistema aperfeiçoa a organização dos processos envolvidos, trazendo melhorias ao controle das atividades financeiras.

Palavras-chave: Transformação digital. MEAN *Stack*. Sistema *Web*. Experiência de usuário. Usabilidade.

[*]Submetido em 21/06/2019 - Aceito em 10/03/2020

[1]Graduado em Sistemas de Informação pela PUC Minas, Brasil, vedovatto@gmail.com.
[2]Graduanda em Engenharia de Computação pela PUC Minas, Brasil, cinthia.mikaela@sga.pucminas.br.
[3]Doutora em Ciência da Informação, Docente do Instituto de Ciências Exatas e Informática da PUC Minas, Brasil, magali@pucminas.br.

Abstract

Parishes are places for meetings of religious believers and, in the organizational processes, there are practices of financial control of expenses and incomes. In certain parishes, incomes are recorded on paper during masses and, later transferred to computerized worksheets, generating the risk of loss of information during the process. Among the main incomes are the contributions, whether in the form of tithing or donation. In this work we present the development of a web system for parish communities to improve their control of contributions receipts. The objective is to implement the process of digital transformation in parishes, that are still not computerized. It is developed a solution to contemplate the real needs of the target audience, which aims for a more secure and effective process of contribution registration. The MEAN Stack, used in the process, covers all technologies used for the solution development. The design followed the concepts of User Experience (UX), aiming for a better experience in using the application regarding fluidity and facility when performing operations. It was verified, in the validation process, that the system improves the organization of the processes involved, bringing improvements to the control of financial activities.

Keywords: Digital Transformation. MEAN Stack. Web System. User Experience. Usability.

1 INTRODUÇÃO

Existem diversos locais para reuniões de fiéis religiosos, sejam eles católicos, espíritas ou de qualquer outra religião que os reúna em centros especializados. Um dos principais pontos de encontro são as paróquias, locais onde os fiéis são acolhidos, recebem orientação, ajuda espiritual e material. Uma comunidade paroquial é a congregação de pessoas que se relacionam visando a manutenção das atividades exercidas no dia a dia de uma paróquia, reunida com o intuito de manter todo o sistema que envolve a comunidade em pleno funcionamento.

Dentre os processos recorrentes de gestão de uma comunidade paroquial, encontram-se aqueles relativos à organização das finanças. E presentes tanto em custos quanto em recebimentos encontram-se os valores relacionados aos colaboradores, pessoas que contribuem financeiramente com a paróquia como modo de compensação ao esforço e aos trabalhos realizados para manutenção da comunidade. Atualmente, existem paróquias onde o controle do recebimento das contribuições é feito de forma desorganizada e descentralizada, realizado em cadernos de registro contábil ou em planilhas informatizadas. Desse modo, gera-se o risco de perda de informação quando registros são feitos de forma equivocada em folhas de papel durante as missas. Além disto, há um grande risco de se perder o histórico de registros ao manter os dados de controle em planilhas informatizadas em um computador com acesso local, caso o disco rígido dessa máquina se danifique.

O objetivo deste trabalho é desenvolver um sistema de informação para uso em processos organizacionais em uma comunidade paroquial, realizar a organização dos recebimentos de contribuições de comunidades paroquiais utilizando um sistema *Web* de controle de recebimento de contribuições, desenvolvido com tecnologias em destaque na atualidade, como o ambiente de execução de servidores Node.js, utilizado na criação do *web service* que servirá os dados ao *front end* da aplicação, desenvolvido com o *framework* Angular.

Os representantes da comunidade, pessoas de diferentes idades, níveis de escolaridade e níveis de inclusão digital, terão acesso ao sistema e, de forma simples, objetiva e prática, poderão realizar os lançamentos referentes às contribuições recebidas em sua paróquia.

As próximas seções estão organizadas como descrito a seguir. A Seção 2 apresenta o referencial teórico, contendo informações relacionadas à transformação digital, arquitetura de software e usabilidade. A Seção 3 apresenta a metodologia relacionada ao desenvolvimento do sistema, desde as ferramentas e tecnologias utilizadas, passando pelo levantamento dos requisitos, modelagem das entidades e relacionamentos, testes de usabilidade, até o *deploy* da aplicação para que esta seja colocada em produção, pronta para a utilização. Na Seção 4, são apresentados os resultados do sistema desenvolvido e, por fim, na Seção 5 são apresentadas as considerações finais e possíveis aperfeiçoamentos futuros do sistema.

2 REFERENCIAL TEÓRICO

As próximas subseções apresentam conceitos relacionados às arquiteturas de desenvolvimento de sistemas, transformação digital e heurísticas de usabilidade focadas no *design* da interface.

2.1 Arquiteturas de desenvolvimento de sistemas

Antes de desenvolver o código de uma aplicação, é necessário definir sua arquitetura. A arquitetura da aplicação estabelece como a aplicação é organizada nos sistemas finais. Em aplicações modernas, as arquiteturas mais utilizadas são a cliente-servidor e a *peer-to-peer* (P2P). Um sistema *Web*, por exemplo, é uma aplicação clássica da arquitetura cliente-servidor. Nessa arquitetura, tem-se um hospedeiro, denominado servidor, que atende as solicitações de vários hospedeiros denominados clientes (KUROSE; ROSS, 2012). No contexto dos sistemas *Web*, considera-se que:

> Em aplicações *Web*, o cliente é um dispositivo que, através de um navegador, requisita conteúdos a um endereço (URL) que referencia um servidor. Esse é responsável por responder ao cliente com conteúdo que é interpretado pelo navegador e mostrado ao usuário final (SCHNEIDER, 2016, p. 12).

Além dos navegadores, outros programas também podem ser clientes *Web*, como os *web services*. Os *web services* são uma solução que trabalha na integração e na comunicação de diferentes aplicações. Cerami (2002) define que um *web service* é qualquer serviço disponibilizado na Internet que utiliza um sistema de mensagens padronizado e não está vinculado a um sistema operacional ou a uma linguagem de programação. Dentre os padrões de arquitetura de *web services*, tem-se os denominados *web services* RESTful, que são aqueles que seguem o estilo arquitetural *Representational State Transfer* (REST). O REST é um estilo de arquitetura de *software* que utiliza a arquitetura cliente-servidor da *Web*. De acordo com Fielding (2002):

> *Representational State Transfer* destina-se a evocar uma imagem de como um aplicativo da *Web* bem projetado se comporta: uma rede de páginas da *Web* forma uma máquina de estado virtual, permitindo que um usuário progrida pelo aplicativo selecionando um *link* ou enviando um breve formulário de entrada de dados, em que cada ação resulta em uma transição para o próximo estado do aplicativo, transferindo uma representação desse estado para o usuário (FIELDING, 2002, p. 2, tradução nossa).

Assim sendo, os serviços que implementam o REST utilizam regras que permitem a criação de interfaces muito bem definidas, permitindo a fácil utilização de seus métodos. Além disso, os recursos REST são acessados por meio de uma interface uniforme e padrão, oferecendo vantagens como a familiaridade e a interoperabilidade (BELQASMI et al., 2011).

2.2 Transformação digital

De acordo com Solterman e Fors (2004), a transformação digital pode ser entendida como sendo as mudanças que a tecnologia digital causa ou influencia em todos os aspectos da vida humana. O termo transformação digital não possui uma definição compartilhada na literatura (MERGEL et al., 2019). Nesse trabalho, o termo é utilizado com base na definição de Solterman e Fors (2004). Sendo assim, entende-se que a transformação digital é um conceito que envolve a tecnologia e novas formas de pensar com o intuito de criar estratégias que remodelem o negócio das organizações.

Atualmente, a transformação digital vem sendo imposta como um meio para que as organizações possam tornar-se ou permanecer competitivas no mercado (KUTNJAK et al., 2019). A implementação de uma estratégia de transformação digital tornou-se uma preocupação essencial para muitas organizações, devido aos impactos que as tecnologias digitais possuem no ambiente interno e externo de uma organização (CHANIAS et al., 2019).

A transformação digital pode ser aplicada em diversas áreas, como, em sistemas de prestação de serviço do setor público (SANG-CHUL; RAKHMATULLAYEV, 2019), no varejo (REINARTZ et al., 2019), no setor produtivo (BORANGIU et al., 2019) e no setor de turismo cultural (SHEHADE; STYLIANOU-LAMBERT, 2020).

2.3 Heurísticas de Nielsen

Pode-se considerar uma boa prática no desenvolvimento de interfaces que desenvolvedores avaliem a usabilidade de seus sistemas, *sites* e aplicativos de acordo com dez princípios chamados de Heurísticas de Nielsen (MOLICH; NIELSEN, 1990). Realizar a análise da interface de um sistema a partir das heurísticas pode gerar resultados que aprimorem o seu uso do ponto de vista do usuário, deixando-a mais atrativa, intuitiva, limpa e de fácil utilização. Rocha e Sousa (2010, p. 24) caracterizam o método baseado nas heurísticas da seguinte maneira:

> Além de ser internacionalmente reconhecido, o método garante vantagens pelo seu baixo custo de aplicação e por ser facilmente aplicado para o teste de usabilidade em produtos ou serviços de informação disponibilizados na *Web* e, mormente, para a adequação de páginas e interfaces *Web* à interação homem-computador.

Esses princípios são citados e explicados a seguir:

a) Visibilidade do estado do sistema: realizado, principalmente, por meio de *feedbacks* pós interações ou carregamentos. O sistema deve manter o usuário informado sobre o seu estado atual, por exemplo, via mensagens de erro ou sucesso.

b) Equivalência entre o sistema e o mundo real: deve-se evitar o uso de linguagem extremamente técnica, preferindo utilizar linguagem próxima à do usuário do sistema. Além da

linguagem, o mesmo deve ser feito com ícones e imagens. Um exemplo é a utilização do famoso disquete em ícones e botões com ações de salvamento.

c) Liberdade e controle do usuário: a decisão da ação no sistema é do usuário. Portanto, ações devem ser sugeridas e não induzidas. Também é necessário dar ao usuário a possibilidade de desfazer alguma ação e retornar a pontos anteriores em fluxos do sistema.

d) Consistência e padrões: é necessário manter o padrão de interação em contextos que tratem ações similares, como, por exemplo, manter a seta de avanço do mesmo tamanho e posição entre diferentes páginas de uma mesma lista.

e) Prevenção de erro: sair de uma página por clique acidental no *mouse*, após ter preenchido diversos campos de um formulário, é algo capaz arruinar a experiência do usuário em certo sistema. Alertá-lo, com sinalizações ou mensagens de confirmação de ações, evita esse tipo de acidente.

f) Reconhecer ao invés de relembrar: tornar a utilização de um sistema simples para o usuário memorizar é o ponto principal deste princípio. Uma boa prática é tentar descobrir quais sistemas e aplicativos o usuário principal do seu sistema utiliza no dia a dia. Desenvolver um sistema similar ao que o usuário tem costume evita que ele tenha que aprender uma nova forma de utilização de sistemas.

g) Flexibilidade e eficiência de uso: o sistema deve estar apto à utilização de usuários avançados e novatos. A utilização de teclas de atalho, preenchimento automático de dados e máscaras de campos tornam o sistema eficiente e flexível, não gerando incômodos ao usuário avançado em termos de morosidade na utilização e, também, deixando o usuário novato apto à utilizá-lo sem grandes dificuldades de entendimento.

h) Estética e *design* minimalista: a interface do sistema deve possuir somente elementos chave ao cerne dos seus objetivos. Excesso de cores e elementos visuais podem confundir o usuário e levar a sua atenção para áreas menos importantes do sistema.

i) Auxiliar usuários a reconhecer, diagnosticar e recuperar ações erradas: junto à diretriz da prevenção de erros, deve-se conduzir o usuário à resolução de problemas inevitáveis do sistema, via mensagens de erro de fácil entendimento e dicas de entradas corretas no preenchimento de campos.

j) Ajuda e documentação: mesmo após aperfeiçoar a experiência e interação do usuário seguindo os nove princípios citados acima, sistemas podem ser naturalmente complexos, seja por lidarem com processos melindrosos ou novos ao entendimento do usuário. Nesse ponto, uma boa documentação e elementos de ajuda se tornam pontos de apoio úteis para a boa utilização do sistema.

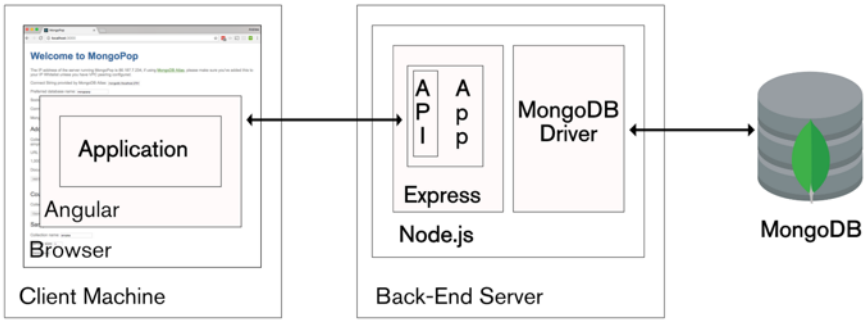
2.4 Tecnologias e Ferramentas

As próximas subseções apresentam os conceitos relacionados às tecnologias presentes no acrônimo MEAN *Stack*, que são as principais tecnologias utilizadas no desenvolvimento do sistema.

2.4.1 MEAN Stack

MEAN é o acrônimo das tecnologias MongoDB, Express, Angular e Node.js (BONFIM; LIANG, 2016). Juntas, formam uma pilha de tecnologias que propiciam o desenvolvimento de aplicações *Web* completas (HOLMES, 2015). A Figura 1 ilustra a arquitetura de uma aplicação desenvolvida com o MEAN *Stack* e apresenta como é realizada a troca de informações entre o cliente, o servidor e o banco de dados.

Figura 1 – Arquitetura de uma aplicação MEAN *Stack*



Fonte: Morgan (2017).

Trabalhar com as tecnologias do MEAN *Stack* traz ao desenvolvedor pontos positivos, tais como:

a) utilização do JavaScript no lado do cliente e no lado do servidor. Segundo Almeida (2014, p. 12), "a onipresença da linguagem ajuda na integração da equipe";

b) aumento da produtividade ao ponto em que os objetos no banco de dados e no cliente são semelhantes.

2.4.2 Node.js

Node.js é uma plataforma de código aberto utilizada para desenvolvimento de servidores *Web*. Foi desenvolvido por Ryan Dahl e teve o seu lançamento em 2009 (HERRON, 2013, p.20).

A maior motivação para a sua criação foi auxiliar os desenvolvedores na criação de aplicações escaláveis, capazes de manipular milhares de conexões simultâneas em uma única máquina. De Acordo com Rys (2011, p.4, tradução nossa), em termos de escalabilidade de aplicações, "a aplicação precisa ser capaz de escalonar para um grande número de usuários, potencialmente milhões".

O Node.js teve como estrutura do seu desenvolvimento a *engine* V8. Criada pelo Google e utilizada também no navegador Chrome, a *engine* V8, segundo definição do próprio *website* do projeto (V8, 2018, tradução nossa), "é uma sintaxe para código binário seguro de baixo nível, de alto desempenho e de código aberto", desenvolvida com o intuito de auxiliar no alto desempenho de aplicações ao compilar o código escrito em JavaScript para código de máquina antes de executá-lo.

O Node.js utiliza uma arquitetura voltada a eventos para lidar com a concorrência nas requisições e interpreta as requisições de forma assíncrona em uma única *thread*, ou seja, ao receber uma requisição e executar uma ação, o Node.js não aguarda a sua resposta para começar a processar a próxima ação. Quando a resposta de uma das ações é retornada, a função que deve ser executada como resposta é inserida em uma fila para ser executada assim que possível. Essa característica, chamada *loop* de eventos, torna o Node.js rápido e capaz de lidar com um alto número de requisições.

O Node.js utiliza o *Node Package Manager* (NPM) para gerenciamento de pacotes (pedaços de código com funções específicas) que podem ser utilizados em aplicações *front end*, aplicações *back end*, robôs, roteadores, entre outras. NPM é um serviço composto por um *website* onde os desenvolvedores podem buscar pelos pacotes desejados, e uma ferramenta de linha de comando, ou *Command Line Interface* (CLI), em que os desenvolvedores interagem com o NPM, instalando, atualizando e removendo pacotes do Node.js em suas aplicações.

2.4.3 Express

O Express é um *framework* para o Node.js de características minimalistas, voltado para a criação de aplicações *Web*. Segundo Almeida (2014, p.16), "o Express estende as capacidades do Node.js adicionando vários *middleware* e outras capacidades" ao efetuar abstrações de rotas e muitas outras funções. Este *framework* disponibiliza para o desenvolvedor um conjunto de recursos, tais como:

a) gerenciamento de requisições e rotas;

b) definições básicas para servidores *Web*, como a porta a ser usada em uma conexão e a localização dos modelos que serão usados para renderizar as respostas;

c) adição de *middleware* em qualquer ponto do tratamento da requisição, capazes de processá-la e dar novo direcionamento.

A utilização de outros *middleware* é um ponto importante no desenvolvimento de servidores *Web*. Segundo Moreno (2006, p.14), um *middleware* é "responsável por esconder das aplicações a complexidade dos mecanismos definidos pelos padrões, protocolos de comunicação e até mesmo sistema operacional do equipamento".

2.4.4 MongoDB

Nos bancos de dados relacionais, como o SQL Server e o MySQL, todos os registros estão contidos em estruturas de dados pré-definidas (tabelas com colunas fixas). Já os bancos de dados NoSQL (*Not only SQL*) possuem um esquema dinâmico com dados não estruturados, ou seja, o dado pode ser armazenado em diferentes formatos, como documentos, grafos e pares de chave-valor.

O MongoDB é um banco de dados NoSQL orientado a documentos que armazena os dados no formato *Binary Object Notation* (BSON). Esses documentos podem possuir vários atributos, inclusive documentos internos. Segundo Sirqueira e Lucena (2017, p.17), "BSON amplia o modelo *JavaScript Object Notation* (JSON) para fornecer tipos de dados adicionais, campos encomendados e ser eficiente para codificação e decodificação em diferentes idiomas". O JSON é um formato de transmissão de dados de fácil interpretação humana em que os dados são registrados em formato de pares chave-valor e em arranjos. Segundo Fonseca e Simoes (2007), "o JSON foi desenhado com o objetivo de ser simples, portável e textual". Segundo o próprio *website* do MongoDB (2018, tradução nossa), "o MongoDB armazena dados em documentos flexíveis semelhantes ao JSON, o que significa que os campos podem variar de documento para documento e a estrutura de dados pode ser alterada ao longo do tempo". Dentre as principais características do MongoDB, destaca-se a forma como ele lida com grandes volumes de dados mantendo a alta *performance*.

2.4.5 Angular

O Angular, um *framework* JavaScript, criado pelo Google, projetado para criação de aplicações *Web*, *Desktop* e *Mobile* seguindo, principalmente, o modelo de *Single Page Application* (SPA), é um dos principais *frameworks* para o desenvolvimento de aplicações *front end*, como ilustra a Tabela 1, que apresenta informações de novembro de 2018. Segundo Oliveira (2017, p.1), uma SPA possui "páginas que atualizam sua interface à medida que seus usuários interagem, sem a necessidade de recarregar todo o conteúdo". O conteúdo da aplicação é todo carregado no primeiro acesso e os componentes da aplicação são visualizados ou ocultados, sem necessidade de fazer uma nova requisição ao servidor à medida que o usuário interage com a aplicação. A interação com o servidor é necessária somente quando o usuário utiliza algum serviço da aplicação que faça novas requisições.

Tabela 1 – Popularidade dos *Frameworks Front End*

Classificação	*Framework Front End*
1	ASP.NET
2	Angular
3	Ruby on Rails e React
4	ASP.NET MVC
5	Django

Fonte: hotframeworks (2018).

Ao utilizar o Angular, os desenvolvedores criam a interface da aplicação e fazem uso de serviços capazes de trocar de arquivos JSON junto ao *web service*. Uma importante ferramenta para o desenvolvimento de aplicações com o Angular é a Angular CLI. Com ela, é disponibilizada, ao desenvolvedor, uma interface de linha de comando capaz de resolver todas as dependências de uma aplicação. O Angular CLI também simplifica a execução de todas as etapas que podem ser automatizadas, da criação do projeto ao seu *build*.

2.5 Trabalhos relacionados

Bonfim e Liang (2016), em seu trabalho, apresentaram as tecnologias envolvidas no acrônimo MEAN (MongoDB, Express, Angular e Node.js) e a maneira como é feita a integração entre estas tecnologias, enfatizando características e técnicas de *software* que contribuem para a escalabilidade, tais como o carregamento de *cache* e enfileiramento das entradas para o banco de dados. Esse trabalho teve como proposta principal desenvolver e analisar uma aplicação que realiza uma competição matemática em tempo real entre dois jogadores. Foram realizados testes de escalabilidade e analisados os resultados, utilizando parâmetros como o tempo de resposta do servidor dado um conjunto de requisições, o consumo de memória e a utilização do processador. Conclui-se que o MEAN *Stack* é uma ótima opção para o desenvolvimento de aplicações *Web* quando se tem como requisito a escalabilidade.

A partir do desenvolvimento de uma aplicação capaz de realizar buscas em um portal a partir da entrada de palavras-chave, Fonseca Junior et al. (2018) demonstraram a capacidade do *framework* Angular 2 no desenvolvimento *Web*, ressaltando as suas características principais, tais como a linguagem de programação utilizada, o *Typescript*, e o conceito de SPA. Foram apresentadas as ferramentas utilizadas no auxílio ao desenvolvimento da aplicação, como o gerenciador de pacotes do Node.js, o NPM. Outro ponto importante citado pelos autores é que o Angular é usado na base do *framework* Ionic, utilizado no desenvolvimento de aplicações híbridas que rodam em aparelhos de plataforma Android e IOS, além da possibilidade de gerar o aplicativo para navegadores *Web*.

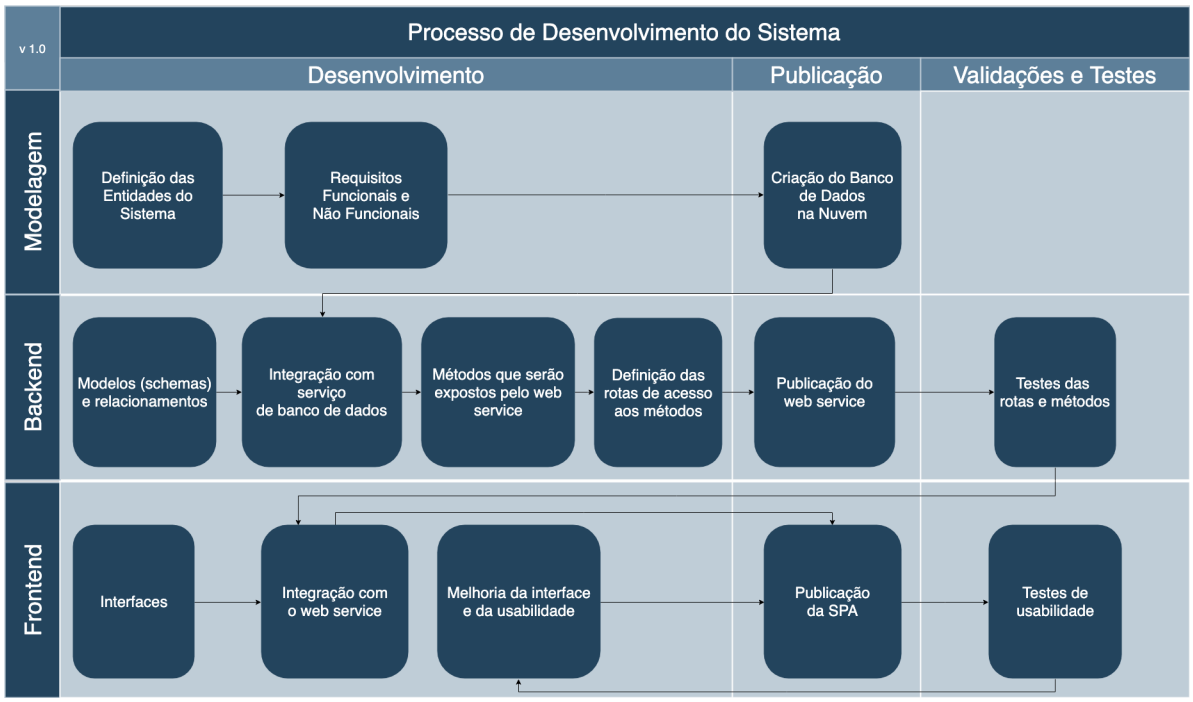
Na monografia de Alves (2017), foi desenvolvido um aplicativo *Web Mobile* com o Ionic, um *framework* que estende as funcionalidades do Angular. O foco deste aplicativo foi ajudar universitários a encontrar vagas em quartos de repúblicas. O banco de dados utilizado

foi o Firebase, escolhido por possuir serviços de fácil utilização e também uma interface considerada bastante intuitiva. Além disto, o autor desenvolveu a aplicação seguindo as boas práticas do *User Experience* (UX).

3 METODOLOGIA

O fluxograma apresentado, na Figura 2, ilustra as principais etapas do processo de desenvolvimento do sistema, estando divididas em modelagem, *back end* e *front end*. Cada etapa foi subdividida em fase de desenvolvimento, de publicação e de validação e testes.

Figura 2 – Fluxograma da Metodologia



Fonte: Elaborada pelos autores.

O processo de desenvolvimento do sistema iniciou-se a partir da definição das entidades que serão manipuladas pelos usuários finais, considerando também os relacionamentos destas entidades com outras do sistema, como aquele entre os contribuidores e as suas contribuições. Em seguida, foram definidos os requisitos do sistema para garantir que as funcionalidades e restrições exigidas por um sistema de controle de contribuições estejam presentes.

Com a definição das entidades, relacionamentos e requisitos, foi criada, em nuvem, a estrutura de armazenamento do banco de dados, seguido do desenvolvimento do *web service* da aplicação, de acordo com as seguintes etapas:

a) modelagem dos dados, relacionamentos e integração com o serviço hospedagem do banco de dados;

b) desenvolvimento dos métodos expostos pela API;

c) desenvolvimento das rotas de acesso aos métodos;

d) hospedagem;

e) validação de rotas e métodos.

Em seguida, foi produzida a SPA, utilizando o *framework* Angular em sua sétima versão. Esta SPA é o *front end* do sistema. O usuário final pode navegar entre os componentes da aplicação que acessam as rotas disponibilizadas pelo *web service* e também realizar todas as funções necessárias para o pleno funcionamento do sistema. Oferecer ao usuário uma melhor experiência de uso, similar à de um aplicativo *desktop*, foi a principal justificativa para a escolha da SPA para o *front end* da aplicação. A SPA foi colocada em produção na plataforma Heroku. A aplicação é acessada via navegador em qualquer sistema operacional com acesso a Internet.

Ao final do desenvolvimento da interface do sistema e de sua integração com os serviços disponibilizados pelo *web service*, foram realizados testes de usabilidade baseados nas Heurísticas de Nielsen (MOLICH; NIELSEN, 1990) e, posteriormente, o sistema passou por uma etapa de remodelagem, visando disponibilizar ao usuário uma melhor experiência de uso.

A aplicação foi hospedada em um serviço de Plataforma Como Serviço (PaaS) e, por fim, foram realizados testes de validação com possíveis usuários do sistema, visando obter análises do fluxo correto das operações realizadas no sistema e o alinhamento da aplicação com os processos de uma administradora de paróquia. As demais seções da metodologia apresentam com maior detalhamento cada uma das etapas de desenvolvimento.

3.1 *Web service*

Nessa seção, foram descritas as etapas de desenvolvimento do *web service* da aplicação. Inicialmente, foi realizada a modelagem das entidades e relacionamentos. Para o contribuidor, foram modelados, em formato de pares chave-valor, os dados pessoais de nome, data de nascimento, código do contribuidor, endereço de *e-mail*, gênero, número de telefone e endereço. As contribuições são documentos internos ao documento do contribuidor, sendo definidas por pares chave-valor de mês, ano, valor e data de pagamento. Mantê-las como parte integrante do documento do contribuidor é prática comum em bancos de dados NoSQL, visando facilitar o processo de construção da consulta e agilizar o seu retorno em termos de processamento. Para o usuário, foram mantidas as informações de usuário, endereço de *e-mail* e senha de acesso.

Posteriormente, foram definidas a utilização dos métodos expostos pela API e as rotas de acesso. O *web service* provê acesso aos métodos *Create, Read, Update e Delete* (CRUD) do contribuidor via métodos de requisição HTTP, que indicam o objetivo da rota em determinado recurso. Foram utilizados os métodos *GET* para realizar buscas de dados, *POST* para criação, *PUT* para edição e *DELETE* para remoção. Nesse sentido, foram mantidas as boas práticas de uma API REST, em que a semântica do método indica a ação que foi realizada sobre determinado recurso. A manipulação dos documentos das contribuições foi considerada como

uma edição do documento de um determinado contribuidor. Além destas rotas, o sistema possui a rota de autenticação do usuário, capaz de gerar um *token* que definirá se o usuário com determinado endereço de *e-mail* e senha de acesso é apto a utilizar o sistema.

Por fim, foram definidos os *middleware* utilizados no *web service*. As características dos *middleware* são apresentadas no Quadro 1.

Quadro 1 – Características dos *Middleware*

Middleware	Característica
Path	Módulo que provê ao Node.js uma forma de lidar com diretórios e caminhos para arquivos
Mongoose	Biblioteca do Node.js que auxilia o desenvolvedor na modelagem dos dados do banco de dados
Passport	Módulo para Node.js que auxilia a solucionar questões de autenticação de uma certa aplicação
CORS	Módulo capaz de lidar com a comunicação entre aplicações hospedadas em diferentes domínios
jsonwebtoken	Pacote que implementa o protocolo JSON *Web Token* ao *web service* desenvolvido com o Node.js

Fonte: Elaborado pelos autores.

3.2 Desenvolvimento do *front end* com Angular

Utilizando o *framework* Angular, foi desenvolvida a interface do sistema que é utilizada pelo usuário final, composta de módulos e componentes relacionados à autenticação do usuário e à manipulação dos documentos do contribuidor. No processo de desenvolvimento da SPA, foi realizada a integração do *front end* com o *web service*.

Cada componente da aplicação é composto por um arquivo no formato *Hypertext Markup Language* (HTML), responsável pela exposição de elementos da aplicação para o usuário final, um arquivo para formatação dos estilos dos elementos presentes no HTML no formato *Cascading Style Sheet* (CSS) e um arquivo no formato *Typescript* (TS), que lida com as ações do usuário junto ao HTML, além de relacionar o componente a serviços, módulos e outros componentes.

3.3 Avaliação e validação do sistema

Após o término do desenvolvimento da primeira versão do sistema, visando melhorar a usabilidade do sistema para o usuário em termos de navegação, interação e experiência, foram realizadas avaliações de usabilidade e de interface junto a *designers* e desenvolvedores. Com

o intuito de atender às expectativas reais do usuário final e validar o sistema, foi realizada uma demonstração da aplicação para um representante de uma comunidade paroquial. Os resultados das avaliações e da validação encontram-se na Seção 5.

4 APRESENTAÇÃO DOS RESULTADOS

Nessa seção, são apresentados os resultados obtidos com o sistema desenvolvido. O sistema proposto foi desenvolvido entre agosto de 2018 e abril de 2019. Para o desenvolvimento do sistema de controle de contribuições, inicialmente, foi realizada a etapa de modelagem do sistema, que consiste na definição das entidades e o levantamento dos requisitos funcionais e não funcionais. Nos Quadros 2 e 3, são apresentados, respectivamente, os requisitos funcionais e os requisitos não funcionais obtidos. Na primeira coluna da tabela são apresentados os requisitos, na segunda coluna, a descrição do requisito e, na terceira coluna, a classificação de prioridade.

Quadro 2 – Requisitos Funcionais

Requisitos	Descrição	Prioridade
RF1: Cadastro de novo usuário	Cadastro de usuários administradores do sistema	Alta
RF2: Edição de dados do usuário administrador	O usuário modifica o(s) campo(s) desejado(s) e salva a modificação	Média
RF3: Remoção de um usuário	Remover um usuário administrador do sistema	Média
RF4: Cadastro de novo contribuidor	Cadastro de contribuidores da paróquia	Alta
RF5: Remoção de um usuário	Remover um contribuidor do sistema	Média
RF6: Inserção de contribuição a um contribuidor	Os usuários cadastrados podem inserir o valor de uma contribuição referente a um ano, mês e contribuidor	Alta
RF7: Modificação de valor de contribuição	O usuário pode editar o valor de uma contribuição já realizada	Baixa

Fonte: Elaborado pelos autores.

Quadro 3 – Requisitos Não-Funcionais

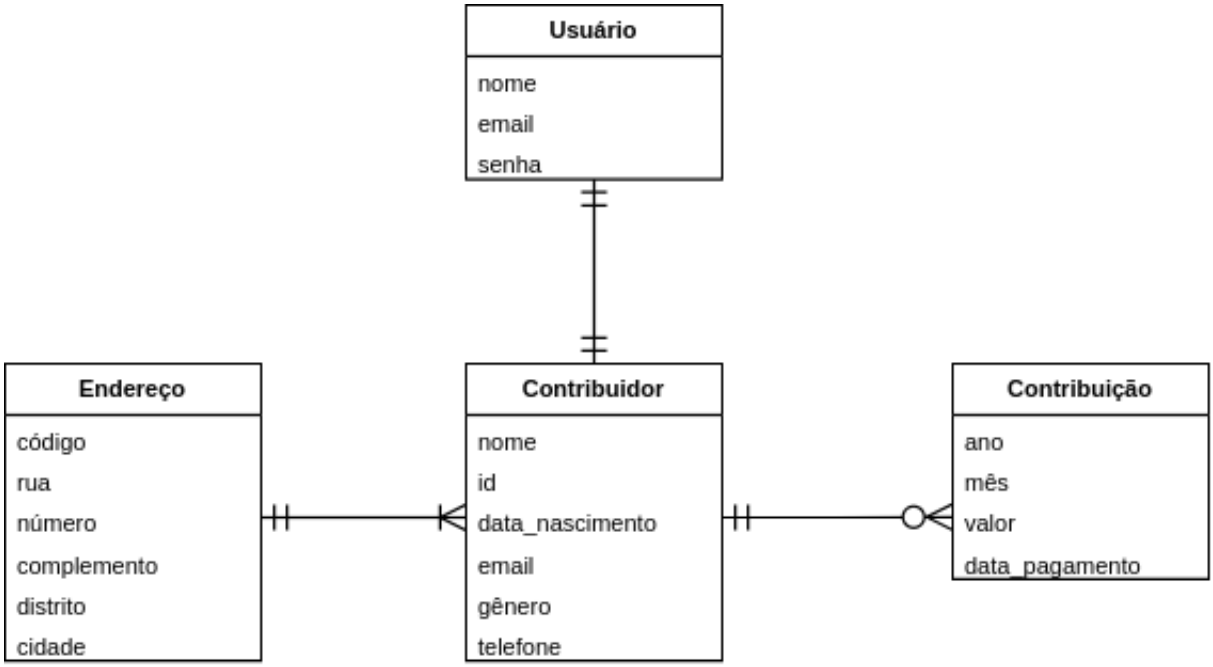
Requisitos	Descrição	Prioridade
RNF1: Usabilidade	O sistema deve ser simples e de fácil utilização, de forma que o usuário consiga, sem muito esforço, saber o estado atual em que ele se encontra dentro do sistema e efetuar as operações desejadas	Média
RNF2: Portabilidade	O sistema poderá ser utilizado em diferentes sistemas operacionais e navegadores	Média
RNF3: Segurança	Somente usuários autenticados poderão visualizar informações do sistema e modificar e excluir documentos nele registrados	Alta
RNF4: Disponibilidade	O sistema deverá estar sempre disponível aos usuários. Em caso de problema, este deve ser resolvido o mais rápido possível	Alta

Fonte: Elaborado pelos autores.

A base de dados do sistema contém documentos referentes ao contribuidor e aos documentos das contribuições, todos com seus conteúdos em formato de pares chave-valor e possíveis relacionamentos com outros documentos. Para cada novo documento inserido no sistema, é armazenado um *ObjectID* que, em termos de bancos de dados não relacionais, é um *string*

hexadecimal único. O sistema provê acesso restrito aos administradores da paróquia. Por isto, também persiste o documento que se relaciona a cada um deles junto ao seu *ObjectID*. Na Figura 3, é apresentado um diagrama que ilustra a modelagem do banco de dados.

Figura 3 – Modelagem dos Dados



Fonte: Elaborada pelos autores.

A subseção a seguir apresenta o sistema em sua versão final e são detalhadas as interfaces do sistema. Em seguida, são discutidas as análises realizadas com base nos resultados das avaliações do sistema e da validação junto ao seu público-alvo.

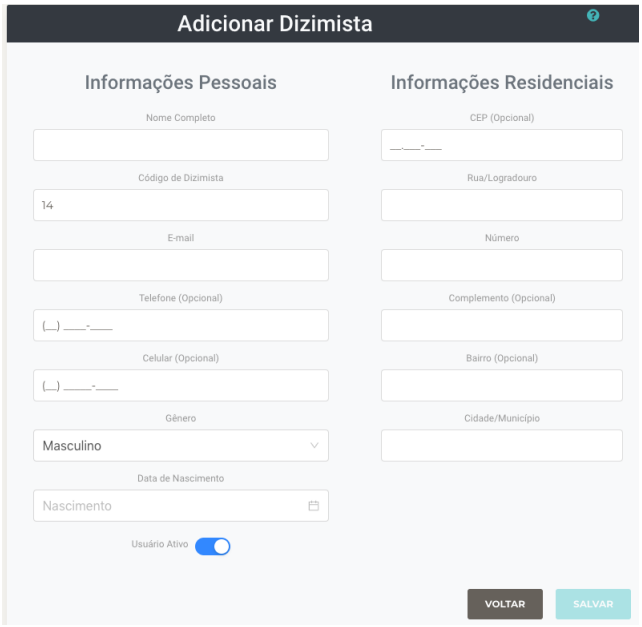
4.1 Detalhamento do sistema

Na aplicação, foram desenvolvidos componentes que lidam com todos os requisitos funcionais do sistema. O componente que disponibiliza o acesso do usuário ao sistema foi definido como a página inicial do sistema. Nesta interface, o usuário administrador, ao inserir os dados de nome de usuário e senha nos devidos campos e pressionar o botão acessar, é direcionado para o *dashboard* do sistema, podendo visualizar, modificar e excluir documentos nele registrados. Além disto, o usuário é direcionado para a interface de *login* quando requisitar acesso a rotas inexistentes ou quando o *token* de acesso se tornar inválido. Foram desenvolvidos diferentes componentes para lidar com a manipulação das informações do contribuidor e suas contribuições, ações relacionadas ao objetivo principal da construção do sistema.

O componente de listagem dos contribuidores registrados no sistema representa o elemento da interface onde são apresentados os dados principais dos contribuidores em formato de tabela. Essa interface oferece ao usuário administrador a possibilidade de, ao pressionar em ícones que representam as ações neles configuradas, visualizar todos os dados do contribuidor,

exclui-lo e, também, visualizar os componentes relacionados às suas contribuições. Essa interface também possui um botão que encaminha o usuário para a adição de novo contribuidor e outro que possui informações sobre como utilizar o sistema ilustrado na Figura 4.

Figura 4 – Tela para manipulação das informações



Fonte: Elaborada pelos autores.

Partindo da visualização da lista de contribuidores, o usuário tem a possibilidade de visualizar todos os dados registrados sobre um contribuidor em específico. As informações são exibidas na lateral direita do *template*, sem que haja recarregamento de página. Caso o usuário queira excluir um contribuidor registrado, deverá pressionar no ícone relacionado a essa ação dentro da coluna "Ações", na linha da tabela referente ao contribuidor. Em seguida, é exibido um modal de confirmação de exclusão ilustrado na Figura 5. Nesse sentido, o sistema evita que contribuidores sejam excluídos por clique acidental. Após a exclusão de um contribuidor, todas as suas contribuições também serão removidas do sistema.

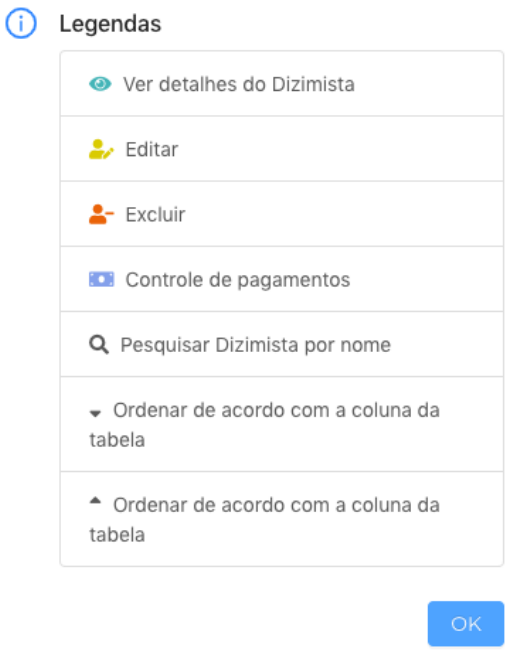
Figura 5 – Modal de confirmação de exclusão

Fonte: Elaborada pelos autores.

Por fim, tem-se os componentes de manipulação de contribuições. Nesse componente, encontra-se a listagem das contribuições realizadas em um determinado ano e o componente

de adição de uma nova contribuição. Ao realizar essa ação, a nova contribuição é carregada automaticamente para o componente de listagem de contribuições. Em caso de dúvidas sobre qual ação o ícone direcionará o usuário, há um componente que descreve cada uma das ações relacionadas aos ícones e botões do sistema. Esse componente é visto na Figura 6.

Figura 6 – Funcionalidades dos ícones



Fonte: Elaborada pelos autores.

4.2 Avaliação do sistema

O método de avaliação, em formato de *checklist*, foi criado com base nos dez princípios gerais do *design* de interface do usuário, desenvolvidos por Jakob Nielsen. Para melhor entendimento da avaliação, foram enviadas para os avaliadores informações sobre o contexto do sistema e um exemplo de *checklist* preenchido.

No *checklist* criado, foi avaliada a gravidade do problema, a partir da sua frequência e do seu impacto, considerando o grau de dificuldade do usuário em contornar o problema e a sua persistência. A gravidade do problema pode ser avaliada de 1 a 4. A avaliação foi realizada por cinco profissionais que atuam no desenvolvimento *Web*, visando resultados mais condizentes com os objetivos da avaliação.

Os avaliadores possuíam as seguintes características: gênero masculino, 80% formados em Sistemas de Informação ou cursando os últimos períodos, idades entre 23 e 32 anos, atuando como desenvolvedores *Web* ou profissões correlatas, tais como analistas de sistema e programador *Web*.

5 ANÁLISE DOS RESULTADOS

Em geral, os avaliadores conseguiram utilizar o sistema sem nenhum impedimento, realizando as ações necessárias para criar, editar, visualizar e excluir contribuidores e contribuições.

As seguintes heurísticas não tiveram problema destacado:

a) Flexibilidade e eficiência de uso;

b) Auxiliar usuários a reconhecer, diagnosticar e recuperar ações erradas;

c) Ajuda e documentação.

Foram encontrados problemas no sistema de acordo com as demais heurísticas e, após análise dos resultados das avaliações, foram definidos pontos passíveis de aperfeiçoamento, descritos a seguir:

a) citada em 60% das avaliações e com uma gravidade média nível 2 (solução de média prioridade), destacou-se a necessidade de mostrar *feedbacks* ao usuário na tela de *login*, em caso de erro nos dados inseridos ou sucesso na ação de acesso;

b) na tela de *login*, o sistema deve disponibilizar ao usuário a possibilidade de alterar ou recuperar a senha de acesso;

c) a tabela com todos os contribuidores deve organizá-los inicialmente por nome ou por código de identificação;

d) citado em 40% das avaliações e com uma gravidade média nível 2 (solução de média prioridade), destacou-se a necessidade de exibir um botão ou ícone para a ação de fechar a aba de controle de pagamentos;

e) deve ser alterada a posição do ícone de “Exibir mais informações”, que encontra-se na primeira coluna da tabela de exibição dos contribuidores. O ícone deve ser colocado junto aos outros ícones de ação, na última coluna da tabela;

f) alterar as cores dos botões: cor verde quando a ação for positiva ou de adição e vermelha quando for negativa ou de exclusão.

A fim de complementar a validação do sistema, foi realizada uma reunião com um representante de uma comunidade paroquial, em maio de 2019. Nessa reunião, foram apresentados o sistema e as suas funcionalidades. O representante ficou interessado na solução, que agilizaria o processo de registro das contribuições e dos contribuidores e diminuiria o uso de papéis no processo. Junto ao representante, foram identificados alguns pontos a serem aprimorados, tais como:

a) Para que o contribuidor possa visualizar as contribuições realizadas por ele, é necessária uma área onde este tenha acesso a essas informações. Desse modo, foi definido que, na tela de acesso ao sistema, existam dois campos onde o contribuidor insira o seu código de identificação e a sua data de nascimento e, caso as duas informações sejam condizentes, todas as contribuições realizadas por ele sejam exibidas em formato de lista.

b) Percebeu-se que o representante precisou de direcionamento para realizar algumas funções do sistema e, pensando em uma situação futura com um novo usuário utilizando o sistema sem a presença de auxílio direto, foi decidido criar uma seção de ajuda contendo vídeos onde são exemplificados todos os fluxos do sistema. Desse modo, um usuário administrador iniciará a sua imersão no sistema por esta seção, evitando que o aprendizado seja feito por tentativa e erro, além de sempre ter a possibilidade de assistir novamente os vídeos sobre um fluxo específico.

c) Foi solicitado que o elemento usado para os dados de data de nascimento do contribuidor na tela de adição de novo contribuidor seja alterado para outro componente mais simples, pois entende-se que o atual seja de utilização complexa para os administradores da paróquia.

Com base no exposto, é possível verificar que o sistema desenvolvido gerou um interesse do representante da paróquia visitada. A possibilidade de realizar uma gestão financeira dos recursos arrecadados pelas paróquias de modo simples e centralizado sem utilizar registros em cadernos ou planilhas se mostrou de grande importância. Algumas dificuldades foram encontradas durante a utilização da aplicação. Como o público-alvo tem o perfil caracterizado por diferentes idades, níveis de escolaridade e níveis de inclusão digital, produzir um sistema de usabilidade simples é questão fundamental para o desempenho esperado do projeto.

6 CONSIDERAÇÕES FINAIS

Foi desenvolvido neste trabalho um sistema *Web* para controle de contribuições e gerenciamento dos processos organizacionais de comunidades paroquiais. Foram definidos três objetivos específicos. O primeiro identificou o contexto de utilização da solução, o público-alvo e as suas reais necessidades. Com isso, percebeu-se a necessidade de alterar os processos realizados manualmente e em planilhas arquivadas em computadores com acesso somente local nos escritórios das paróquias. No segundo, buscou-se um conjunto de tecnologias que desenvolvesse uma aplicação completa, como o MEAN *Stack*, composto pelo banco de dados não-relacional MongoDB, o ambiente de execução de servidores Node.js e o seu *framework* ExpressJs e, para desenvolvimento do *front end*, o *framework* Angular em sua versão 7. Por fim, foram validadas as funções, o fluxo e a interface do sistema. Nessa etapa, o sistema passou por dois processos distintos de validação, o processo de avaliação da utilização do sistema junto a usuários finais e a avaliação do sistema por meio de análise baseada nas Heurísticas de Nielsen (MOLICH;

NIELSEN, 1990). Os resultados obtidos, quando implementados, aperfeiçoarão a experiência de uso da aplicação e o *design* de sua interface.

Alguns desafios foram enfrentados, na concepção do projeto, na tentativa de atender as reais necessidades dos usuários finais do sistema e de analisar a maneira em que o processo de obtenção de contribuições é realizado e registrado nas condições atuais. Posteriormente, as dificuldades para lidar com as tecnologias para desenvolvimento do sistema foram sanadas para aperfeiçoar o conhecimento das técnicas envolvidas e das ferramentas e *frameworks*.

Sabe-se que o controle de contribuições é somente uma pequena parte dentro do contexto de controle financeiro de uma paróquia. Como sugestão de continuidade deste trabalho, pode-se incluir, no sistema, novos módulos, tais como controle de contribuições avulsas, relatórios e acompanhamento de datas e eventos, visando, assim, ampliar o controle da administração da paróquia por meio do sistema *Web*. Outras possibilidades de aprimoramento podem ser implementadas utilizando-se os resultados das avaliações de usabilidade e da validação junto ao usuário final. A solução desenvolvida é útil às comunidades paroquiais e, desse modo, é viável dar continuidade ao projeto, adicionando novos módulos e aprimorando as funcionalidades do sistema, com o objetivo de manter as comunidades paroquiais em um processo contínuo e evolutivo de transformação digital.

Referências

ALMEIDA, F. **MEAN Full Stack JavaScript para aplicações web com MongoDB, Express, Angular e Node**. [S.l.]: Casa do Código, 2014.

ALVES, B. C. S. **Republic: Aplicação mobile para divulgar e procurar vagas em repúblicas universitárias**. 2017. Monografia (Graduação em Sistemas de Informação) – Universidade Federal de Uberlândia, Uberlândia, 2017.

BELQASMI, F.; GLITHO, R.; FU, C. Restful web services for service provisioning in next-generation networks: a survey. **IEEE Communications Magazine**, IEEE, v. 49, n. 12, p. 66–73, 2011. Disponível em: <https://ieeexplore-ieee-org.ez93.periodicos.capes.gov.br/stamp/stamp.jsp?tp=arnumber=6094008>. Acesso em: 28 de fev. 2020.

BONFIM, F. L.; LIANG, M. **Aplicações Escaláveis com MEAN Stack**. 2016, monografia (Graduação em Ciência da Computação) – Universidade Federal do Paraná, Paraná, 2016.

BORANGIU, T. et al. Digital transformation of manufacturing through cloud services and resource virtualization. **Computers in Industry**, Elsevier, v. 108, p. 150–162, 2019. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/0166361519300107>. Acesso em: 28 de fev. 2020.

CERAMI, E. **Web services essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL**. [S.l.]: O'Reill, 2002. ISBN 0-596-00224-6.

CHANIAS, S.; MYERS, M. D.; HESS, T. Digital transformation strategy making in pre-digital organizations: The case of a financial services provider. **The Journal of Strategic Information Systems**, Elsevier, v. 28, n. 1, p. 17–33, 2019. Disponível em: <https://www.wim.bwl.uni-muenchen.de/download/epub/pub_2018_01.pdf>. Acesso em: 28 de fev. 2020.

FIELDING, R.T. **Principle Design of the Modern Web Architecture**. [S.l.]: ACM Transactions on Internet Technology, 2002. Disponível em: <https://www.ics.uci.edu/ taylor/documents/2002-REST-TOIT.pdf/>. Acesso em: 27 de mar. 2019.

FONSECA JUNIOR, L.C.; FONSECA M.R.; RANGEL, H.A.L. Um estudo aplicado sobre framework angular 2. **Universidade Santa Cecília**, v. 7, n. 1, p. 18–25, 2018. Disponível em: <http://periodicos.unisanta.br/index.php/sat/article/download/801/1215/>. Acesso em: 13 de out. 2018.

FONSECA, R.; SIMOES, A. **Alternativas ao XML: YAML e JSON**. Braga: [s.n.], 2007. 14 f. Disponível em: <http://repositorium.sdum.uminho.pt/handle/1822/6230/>. Acesso em: 14 de out. 2018.

HERRON, D. **Node Web Development**. [S.l.]: Packt Publishing, 2013.

HOLMES, S. **Getting MEAN with Mongo, Express, Angular, and Node**. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2015. ISBN 1617292036, 9781617292033.

HOTFRAMEWORKS. **Find your new favorite web framework**. [S.l.], 2018. Disponível em: <http://hotframeworks.com/>. Acesso em: 14 de out. 2018.

KUROSE, J. F.; ROSS, K. W. **Computer networking: a top-down approach**. [S.l.]: Pearson Education, 2012. ISBN 0132856204.

KUTNJAK, A.; PIHIRI, I.; FURJAN, M.T. Digital transformation case studies across industries – literature review. In: IEEE. **2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.], 2019, p.1293–1298. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8756911>. Acesso em: 28 de fev. 2020.

MERGEL, I.; EDELMANN, N.; HAUG, N. Defining digital transformation: Results from expert interviews. **Government Information Quarterly, Elsevier,** v. 36, n. 4, p.101385, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0740624X18304131>. Acesso em: 28 de fev. 2020.

MOLICH, R.; NIELSEN, J. **Improving a Human-computer Dialogue**. New York, NY, USA: ACM, 1990, v. 33. p. 338–348. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/77481.77486>.

MONGODB. **What is MongoDB?** [S.l.], 2018. Disponível em: <https://www.mongodb.com/what-is-mongodb/>. Acesso em: 14 de out. 2018.

MORENO, M. F. **Um middleware declarativo para sistemas de tv digital interativa**. 2006. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

MORGAN, A. **Introducing the MEAN and MERN stacks**. [S.l.], 2017. Disponível em: <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack/>. Acesso em: 20 de jun. 2019.

OLIVEIRA, D. de J. **Uma proposta de arquitetura para Single-Page Applications**. 2017. Monografia (Graduação em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, 2017.

REINARTZ, W.; WIEGAND, N.; IMSCHLOSS, M. The impact of digital transformation on the retailing value chain. **International Journal of Research in Marketing**, Elsevier, v. 36, n. 3, p. 350–366, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167811618300739>. Acesso em: 28 de fev. 2020.

ROCHA, E. da C.; SOUSA, M. de F. E. de. **Metodologia para avaliação de produtos e serviços informação**. [S.l.]: Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict), 2010. Disponível em: <http://livroaberto.ibict.br/handle/1/785/>. Acesso em: 27 de mar. 2019.

RYS, M. **How do large-scale sites and applications remain SQL-based**. [S.l.: s.n.], 2011, v. 9, 8 p. Disponível em: <https://queue.acm.org/detail.cfm?id=1971597/>. Acesso em: 14 de out. 2018.

SANG-CHUL, S.; RAKHMATULLAYEV, Z. M. Digital transformation of the public service delivery system in Uzbekistan. In: IEEE. **2019 21st International Conference on Advanced Communication Technology (ICACT)**. [S.l.], 2019, p. 703–709. Disponível em: <https://ieeexplore.ieee.org/document/8702014>. Acesso em: 28 de fev. 2020.

SCHNEIDER, A. H. **Uma proposta de arquitetura para Single-Page Applications**. 2016. Monografia (Graduação em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2016.

SHEHADE, M.; STYLIANOU-LAMBERT, T. Revisiting authenticity in the age of the digital transformation of cultural tourism. In: SPRINGER. **Cultural and Tourism Innovation in the Digital Era**. [S.l.], 2020, p. 3–16. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-36342-0_1>. Acesso em: 28 de fev. 2020.

SIRQUEIRA, T. F. M.; LUCENA, C. J. P. de. **Proveniência de Dados no BDI4JADE: Um exemplo prático.** [S.l.: s.n.], 2017. 27 p. Disponível em: <http://wiki.les.inf.puc-rio.br/wiki/images/1/12/TassioFinal20172.pdf/>. Acesso em: 14 de out. 2018.

STOLTERMAN, E.; FORS, A. Information technology and the good life. **International Federation for Information Processing Digital Library; Information Systems Research;**, v. 143, 01 2004. Disponível em: <https://link.springer.com/content/pdf/10.1007/1-4020-8095-6_45.pdf>. Acesso em: 28 de fev. 2020.

V8. **Documentation**. [S.l.], 2018. Disponível em: <https://v8.dev/docs/>. Acesso em: 14 de out. 2018.