UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE LICENCIATURA EM COMPUTAÇÃO

JÚLIA FERNANDES ALVES

DESENVOLVIMENTO DE UM SISTEMA PARA EDUCAÇÃO E CULTURA

CAMPINA GRANDE
2017

JÚLIA FERNANDES ALVES

DESENVOLVIMENTO DE UM SISTEMA PARA EDUCAÇÃO E CULTURA

Trabalho de Conclusão de Curso apresentada em Licenciatura em Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Licenciado em Computação, Campus I.
Área de concentração: Engenharia de *Software*.

Orientador: Prof. Me. Edson Holanda C. Junior

CAMPINA GRANDE
2017

A474d Alves, Júlia Fernandes.
Desenvolvimento de um Sistema para educação e cultura [manuscrito] : / Julia Fernandes Alves. - 2017.
63 p. : il. colorido.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2017.
"Orientação : Prof. Me. Edson Holanda Cavalcante Júnior, Coordenação do Curso de Computação - CCT."

1. Engenharia de Software. 2. Modelo cascata. 3. Desenvolvimento de sistemas. 4. Sistema mais cultura Zélia Braz.

21. ed. CDD 005.1

JÚLIA FERNANDES ALVES

DESENVOLVIMENTO DE UM SISTEMA PARA EDUCAÇÃO E CULTURA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Licenciatura plena em Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Computação.

Aprovada em 12 de Dezembro de 2017.

Prof. Me. Edson Holanda Cavalcante Júnior (UEPB)
Orientador(a)

Prof. Dr. Frederico Moreira Bublitz (UEPB)
Examinador(a)

Prof. Me. Antônio Carlos Albuquerque (UEPB)
Examinador(a)

Dedico este trabalho a minha esposa, minha amiga e companheira de todas as horas, Isabelle Maria Lima de Sousa, e que está sempre presente em todas as batalhas que enfrento, sem o seu apoio não as teria vencido.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me confortar durante as dificuldades enfrentadas no trajeto da minha vida acadêmica.

A minha mãe, Ivana Fernandes Lacerda, por ter me ensinado que sou capaz de ser quem eu desejar, e por sempre ter me apoiado em todas as minhas decisões. Agradeço por ter me criado com muito amor, sempre fazendo o melhor diante de suas limitações. Sou grata por compreender minhas ausências, para que eu possa crescer mais um degrau na minha vida.

Agradeço a Isabelle Maria Lima de Souza por estar presente na minha vida, facilitando minha caminhada graças ao seu apoio, sem você este trabalho não seria possível. Sei que posso fazer qualquer coisa se você estiver presente na minha vida.

Aos meus grandes amigos Pablo Herivelton, Valdecio Maximino, Marcondes Jorge, Júlia Cibele e Ianne Maria por me oferecerem apoio e por me proporcionarem momentos felizes de amizade.

Ao professor Edson Holanda por ter aceitado o desafio de auxiliar na construção deste trabalho. Através de sua orientação, passei a conhecer uma pessoa humana, ética, dedicada e disposta a ajudar; características essenciais para um bom docente. Parabéns pelo profissional que és, saiba que por você carregarei comigo, a partir daqui, admiração e gratidão.

A Prefeitura Municipal de Sumé, na pessoa de Inaldo Lourenço por ter confiado em meu trabalho me dando a incumbência de desenvolver o Sistema Mais Cultura Zélia Bráz, instrumento base deste trabalho.

Aos professores Antônio Carlos e Frederico Moreira por ter aceitado em fazer parte da banca examinadora, além de terem feito parte de minha jornada acadêmica como meus professores.

Muito obrigada!

“Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.” Alan Turing

SUMÁRIO

1 INTRODUÇÃO 7
2 SISTEMA MAIS CULTURA ZÉLIA BRAZ 8
2.1 Processo de Desenvolvimento 10
2.1.1 Modelo de Processo de Desenvolvimento 11
2.1.2 Etapas do Desenvolvimento 13
2.1.2.1 Definição de Requisitos 14
2.1.2.1.1 Funcionalidades e Requisitos Funcionais (RF) da Área Administrativa 14
2.1.2.1.2 Funcionalidades e Requisitos Funcionais (RF) do Site do Projeto 19
2.1.2.1.3 Requisitos Não-Funcionais (RNF) Gerais 22
2.1.2.1.4 Proposta de Prototipação Estática 24
2.1.2.2 Projeto de Sistema e Software 26
2.1.2.3 Implementação 28
2.1.2.3.1 *Front-End* 28
2.1.2.3.2 *Back-End* 31
2.1.2.4 Integração e Testes 35
2.1.3 Cronograma 38
2.1.4 Sistema Final - Resultados e Discursões 39
CONCLUSÃO 43
REFERÊNCIAS 45
ANEXO A – PROTOTIPAÇÃO DO PROJETO 48
ANEXO B – TELAS DO PRODUTO FINAL – ÁREA ADMINISTRATIVA 57
ANEXO C – TELAS DO PRODUTO FINAL – SITE DO PROJETO 61

DESENVOLVIMENTO DE UM SISTEMA PARA EDUCAÇÃO E CULTURA

Júlia Fernandes Alves[1]

RESUMO

O presente trabalho apresenta o processo de desenvolvimento do Sistema Mais Cultura Zélia Braz (SMCZB), iniciativa de uma Unidade Municipal de Ensino da cidade de Sumé-PB, realizada com o apoio do Governo Federal. O sistema tem como objetivo permitir que alunos da Unidade construam um acervo digital multimídia dos artistas do cariri paraibano. Para o desenvolvimento do sistema, foi empregado o modelo em cascata com as fases de "Definição de requisitos", "Projeto de sistema e *software*", "Implementação", "Integração e testes" e "Sistema final". Durante o processo, foram aplicados conceitos da Engenharia de *Software* (ES) de maneira equilibrada, considerando as necessidade do cliente; as técnicas e tecnologias mais indicadas, sendo demonstrado, através das boas práticas e lições aprendidas, que a boa aplicação da ES é fundamentada no equilíbrio entre o propósito do produto a ser desenvolvido e a metodologia aplicada. Os resultados obtidos foram extraídos a partir das atividades realizadas ao longo do projeto e dos artefatos produzidos que demonstraram o potencial do modelo de desenvolvimento em cascata. Além disso, favoreceu uma discussão com base no produto final acerca do potencial de um Licenciado em Computação para o mercado de desenvolvimento, tanto no tocante às questões técnicas, quanto às educacionais. Pretende-se com o presente trabalho auxiliar desenvolvedores de *softwares* nas tomadas de decisões de escolha do modelo de desenvolvimento, tecnologias e arquiteturas.

Palavras-Chave: Engenharia de *Software*, Modelo Cascata, Desenvolvimento.

1 INTRODUÇÃO

A Engenharia de *Software* é um ramo da Ciência da Computação que surgiu em 1969 em uma conferência destinada a abordar a crise do *software*. Naquela época, buscava-se resolver os problemas inerentes ao desenvolvimento de *software* em detrimento ao aumento da demanda por produtos de *softwares*, do escopo complexo dos problemas a serem resolvidos e, sobretudo, da falta de diretrizes constituídas para o processo de desenvolvimento.

Sommerville (2007) elucida que a ES "tem por objetivo apoiar o desenvolvimento profissional de *software*, mais do que a programação individual. Ela inclui técnicas que apoiam especificação, projeto e evolução de programas, que normalmente não são relevantes para o desenvolvimento de *software* pessoal." Entretanto, o desenvolvimento de *softwares*

[1] Aluna de Graduação em Licenciatura em Computação na Universidade Estadual da Paraíba – Campus I. Email: juliafealves@gmail.com

para a educação exige esforços adicionais que se integram com as questões técnicas da ES, a fim de obter um ambiente propício para atender um objetivo educativo.

O Sistema Mais Cultura Zélia Braz foi uma iniciativa de uma Unidade Municipal de Ensino da cidade de Sumé-PB, realizada com o apoio do Governo Federal através do "Programa Mais Cultura nas Escolas". O SMCZB objetiva oferecer um acervo digital multimídia dos artistas da terra construído por alunos da escola, além de despertar nos alunos novas possibilidades, promovendo a inclusão digital, construindo o senso crítico e permitindo que atuem de maneira interdisciplinar.

Para o desenvolvimento do SMCZB, foram consideradas as técnicas da ES e de desenvolvimento de *softwares* educacionais no tocante às formas de interação com o usuário, uma vez que o produto desenvolvido é caracterizado como sob encomenda, ou seja, sua construção foi baseada conforme a necessidade de negócio do cliente.

Isto posto, o presente trabalho tem como objetivo apresentar de maneira detalhada o processo de desenvolvimento do SMCZB baseado no Modelo em Cascata, destacando que a aplicação de conceitos da Engenharia de *Software* perpassa o uso de técnicas e diretrizes complexas. O bom emprego da ES é fundamentado no equilíbrio entre o propósito do produto a ser desenvolvido e a metodologia aplicada, pois a finalidade é construír o produto de qualidade com custos apropriados.

Os resultados apresentam o produto finalizado através de sua interface gráfica ilustrada por meio de *printscreens* capturados do sistema real. Além disso, através de uma discussão em torno do processo de desenvolvimento, são levantados os problemas e soluções encontrados durante o desenvolvimento, assim como os proveitos e lições aprendidas.

As principais contribuições do presente trabalho são: (i) apresentar uma aplicação do modelo de desenvolvimento em cascata; (ii) apresentar o processo de desenvolvimento de um *software* destinado à educação considerando direcionamentos da Engenharia de *Software* e de *Softwares* Educacionais; (iii) demonstrar através do produto final o potencial de um Licenciado em Computação para o mercado de desenvolvimento, tanto no tocante às questões técnicas, quanto às educacionais.

2 SISTEMA MAIS CULTURA ZÉLIA BRAZ

Intitulado de "Mais Cultura Zélia Braz", o sistema *web* foi desenvolvido para a Unidade Municipal de Ensino Infantil e Ensino Fundamental Professora Zélia Braz da Cidade de Sumé-PB, no âmbito do "Programa Mais Cultura nas Escolas" do Governo Federal.

Embasado na resolução CD/FNDE nº 4 de 31/03/2014, o Programa Mais Cultura nas Escolas, através de uma aliança entre o Ministério da Educação e o Ministério da Cultura, destina-se promover ações que aliem os fatores culturais e educacionais em escolas públicas brasileiras. Para isso, através de um edital anual, as escolas podem submeter projetos e serem fomentadas com investimentos para: contratação de serviços relacionados às atividades artísticas e pedagógicas; aquisição de materiais de consumo; contratação de serviços; aquisição de materiais permanentes e equipamentos; entre outros. (Manual de Desenvolvimento das Atividades, 2015).

> O Programa consiste em uma iniciativa interministerial, firmada entre os Ministérios da Cultura (MinC) e da Educação (MEC), com a finalidade de fomentar ações que promovam o encontro entre o projeto pedagógico de escolas públicas contempladas com os Programas Mais Educação e Ensino Médio Inovador e experiências culturais em curso nas comunidades locais e nos múltiplos territórios. (Fonte: Manual de Desenvolvimento das Atividades, 2015. < https://goo.gl/MpmkFP > Acesso em: 27/10/2017)

Isto posto, o sistema foi pensado e fundamentado nesses dois universos: Cultura e Educação. Com ele, alunos do Ensino Fundamental da escola acima citada, poderão administrar e construir um acervo cultural de artistas da região do Cariri Paraibano, com a possibilidade de integrar diversas mídias, como por exemplo, vídeos, fotos e *hiperlinks* servindo como um meio para divulgação e consulta das obras dos mesmos.

O objetivo do SMCZB é, em primeiro lugar, construir um acervo digital multimídia dos artistas da terra, considerando que a região é conhecida por seus talentos, mas que por vezes os próprios caririzeiros não conhecem suas raízes e histórias. Em segundo lugar, é despertar nos alunos inseridos no projeto novas possibilidades, promovendo a inclusão digital, construindo o senso crítico e permitindo que atuem de maneira interdisciplinar. Gomes (2015) destaca que a discussão relacionada à entrada da tecnologia na educação não deve mais ser uma questão utópica de uso técnico, mas sim de desenvolvimento de novas práticas, sobretudo ao se colocar em frente aos atuais discentes, nativos digitais que não aprenderam usar tecnologia, mas que nasceram sabendo usar.

O SMCZB vem como uma resposta à essas discussões, sua essência de aplicação é mais prática do que tecnológica, no entanto, para que o sistema seja técnico e pedagogicamente capaz de suportar as práticas, é necessário incluir preocupações especiais durante seu processo de desenvolvimento. Para Ferreira e Rodrigues (2015) o desenvolvimento de *software* para educação exige cuidados especiais, além dos conteúdos e práticas aplicadas que devem ser embasadas em teorias de aprendizagem, é necessário

ressaltar as perspectivas técnicas, especificamente a interface. Os conteúdos e as práticas serão de responsabilidade dos alunos e da gestão da escola na aplicação do sistema, mas no desenvolvimento foram considerados principalmente critérios de usabilidade e navegabilidade da interface, pois os usuários principais serão crianças do Ensino Fundamental.

Atualmente com a era do conhecimento, a colaboração é um diferencial na construção do conhecimento, e a tecnologia com seu avanço vem facilitando as relações e práticas em pró do desenvolvimento social e educacional. Segundo Emydio et al. (2012) em um ambiente educativo, a comunicação deve fluir livremente de maneira colaborativa, e considerar a inserção de maneira correta das tecnologia nesse cenário é um fator que pode auxiliar no processo de ensino e aprendizagem.

O SMCZB proposto neste trabalho, permitirá que alunos e professores trabalhem colaborativamente para o desenvolvimento de habilidades relacionadas às diversas ciências. Com isso, será possível efetivar a difusão do conhecimento através de atividades práticas relacionadas à educação e cultura.

2.1 Processo de Desenvolvimento

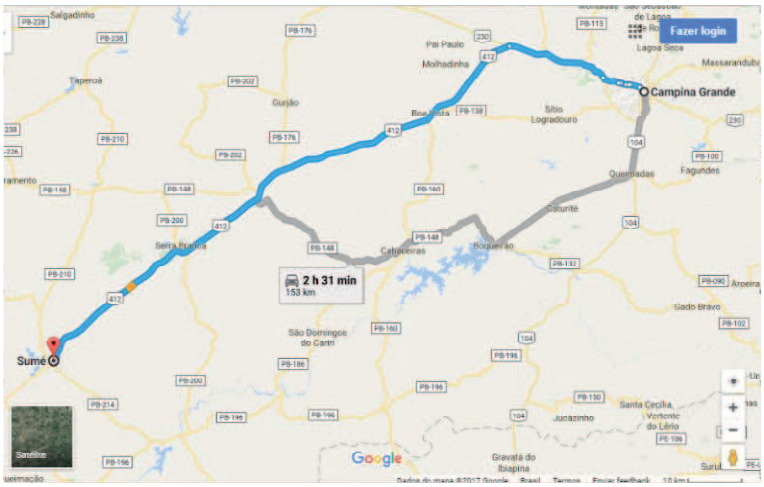
O desenvolvimento do SMCZB, ocorreu no período de junho a outubro de 2016 e seguiu os principais fundamentos da ES necessários para o "desenvolvimento dentro de custos adequados de sistemas de *software* de alta qualidade" (SOMMERVILLE, 2007). Por se tratar de uma ferramenta para a educação, a preocupação com a qualidade foi amplificada, sendo incluídas no contexto preocupações relacionadas às competências não técnicas, assim como àquelas que se aproximam das habilidades e sensibilidade social, "habilidades que são igualmente importantes para o sucesso de qualquer projeto de software quanto às competências" Rodrigues et al. (2016).

A construção de produtos *Softwares* com carácter educacional de qualidade exige da equipe técnica de desenvolvimento habilidades e competências que vão além dos aspectos interpessoais. Com isso em mente, foi possível iniciar o planejamento do projeto definindo inicialmente os profissionais necessários; a metodologia de desenvolvimento adotada; as etapas e seus marcos de entrega; até as tecnologias ideais para dá suporte à construção do produto. Nos subtópicos que se seguem, serão apresentados de maneira detalhada cada item, seja ele relacionado ao fator pedagógico ou técnico da Engenharia de *Software* aplicado no SMCZB.

2.1.1 Modelo de Processo de Desenvolvimento

No contexto real do desenvolvimento do SMCZB, o cliente residia na cidade de Sumé e a equipe do projeto por sua vez na cidade de Campina Grande, ficando separados geograficamente por uma distância de 136 km e a aproximadamente 2h30min de viagem segundo o *Google Maps*, fatores que inviabilizariam a escolha de métodos dependente de muitas reuniões presenciais para validar os marcos.

Imagem 01: Distância entre Cliente e Equipe de Desenvolvimento



Fonte: Elaboração própria a partir do *Google Maps*

Além do fator distância, por se tratar de um projeto advindo de um edital do Programa Mais Cultura, o escopo estava bem definido e uma vez que a análise de requisito estivesse aprovada, a falta de contato direto com o cliente não acarretaria grandes impactos no andamento do projeto. Fernández et al. (2015) em seu artigo "*Naming the Pain in Requirements Engineering Comparing Practices in Brazil and Germany*" realizou um estudo sobre o a Engenharia de Requisitos em dois cenários: Brasil e Alemanha. No estudo, foi verificado que as fábricas de *software* alemãs tendem a utilizar menos os métodos ágeis e passam a adotar padrões como o Cascata; já no Brasil, o cenário foi o inverso. Essa realidade é verificada em razão de, na Alemanha a maior parte das desenvolvedoras de *software* serem contratadas por um cliente com uma demanda bem definida. Assim, após o levantamento de requisitos ser aprovado, as demais etapas podem seguir em cascatas, pois o cliente sabe o que quer e não será necessário realizar mudanças drásticas no escopo durante seu ciclo de vida. Outro fator importante do estudo, foi o alto índice de requisitos incompletos ou ocultos em projetos brasileiros que utilizavam metodologias ágeis, e a justificativa dada pelos autores foi

relacionada à grande extensão territorial do país que pode dificultar os encontros contínuos da equipe com o cliente, pois nesses métodos o cliente participa diretamente do desenvolvimento do produto.

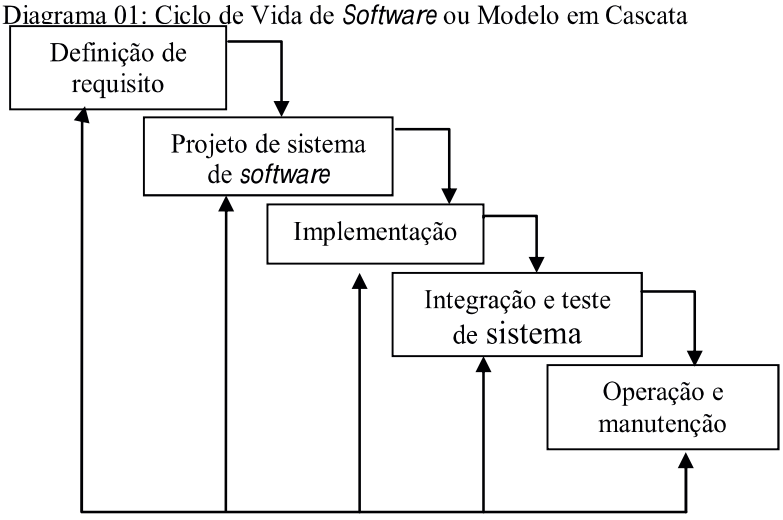
Dito isto, o modelo escolhido para guiar o desenvolvimento do SMCZB foi o cascata, originado dos conceitos mais simples e gerais da Engenharia de *Software* (Royce, 1970), no entanto, diante do contexto de escopo definido e separação geográfica entre cliente e equipe, se mostrou muito promissor.

Também conhecido como ciclo de vida do *software*, o modelo carrega o nome de cascata devido ao encadeamento das etapas (fases) de desenvolvimento sendo um sequenciamento, ao término de uma, a seguinte passa a ser executada.

> Este modelo sugere uma abordagem sequencial e sistemática para o desenvolvimento de software. Dessa forma, começamos com o levantamento de requisitos ou necessidades junto ao cliente, depois vamos para a fase de planejamento onde definimos estimativas, cronograma e acompanhamento, após isso partimos para a modelagem onde fazemos a análise e projeto, seguindo da construção onde codificamos e testamos, passamos para a implantação ou emprego onde efetuamos a entrega, suporte e feedback do software concluído. (Fonte: Medeiro, H. Introdução ao Modelo Cascata. DEVMEDIA < https://goo.gl/zasMvQ>. Acesso em: 28/10/2017)

O modelo cascata é usado sobretudo quando os requisitos de um determinado problema são bem definidos e compreendidos, fato que se encaixa diretamente com o projeto em desenvolvimento. Sommerville (2007) defende 5 (cinco) etapas para o ciclo de vida, são elas: definição de requisitos; projeto de sistema de *software*; implementação e teste de unidade; integração e teste de sistema; operação e manutenção. O processo está ilustrado no diagrama 01:

Diagrama 01: Ciclo de Vida de *Software* ou Modelo em Cascata



Fonte: Sommerville, Engenharia de *Software*, 8ª Edição

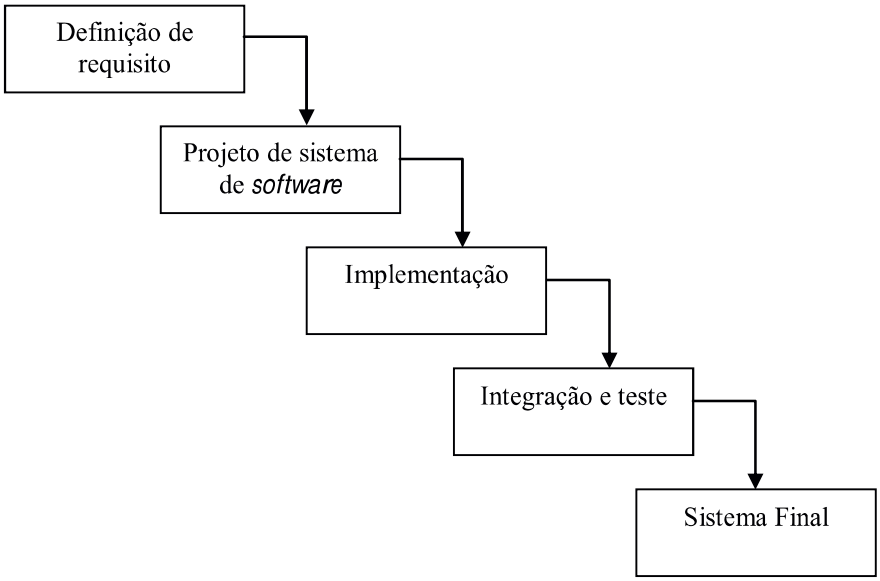
No entanto, para ser aplicável à realidade do SMCZB, foi necessário realizar adaptações nas fases do método, considerando alguns fatores: equipe de desenvolvimento reduzida; escopo do projeto; entrada em operação com data não definida. Ao enfatizar o primeiro e o segundo fatores, observou-se a importância de unir algumas fases em uma, eliminando algumas ações que não causaria grandes impactos, mas que exigiria grandes esforços e como o objetivo inicial é garantir a qualidade com o custo adequado, é imprescindível ponderar para se ter o equilíbrio.

Por fim, como a data de entrada em produção era desconhecida, não havia como prever o acompanhamento da operação e possíveis evoluções; essas ações, se necessárias, fariam parte de um aditivo do projeto. Dessa forma, a metodologia de aplicação do modelo foi seguindo, uma após a outra sequencialmente, mas as fases contempladas, e que serão tratadas de maneira detalhadas no próximo subtópicos foram: definição de requisitos; projeto de sistema de *software*; implementação; integração e teste; Sistema Final.

2.1.2 Etapas do Desenvolvimento

Baseadas na metodologia cascata de desenvolvimento de *software*, as etapas de desenvolvimento do SMCZB está ilustrada no Diagrama 02.

Diagrama 02: Etapas de desenvolvimento do sistema MCZB



Fonte: Elaboração Própria

A partir da leitura do diagrama é possível observar que, diferentemente do ciclo de vida de *software* apresentado por Sommerville (2007), o planejamento não contemplou a fase de operação e manutenção do produto, sendo dessa forma eliminado do diagrama o ciclo que retoma o início do processo para a realização da evolução do sistema. Essa realidade se fez necessária devido ao tipo de contrato estabelecido junto ao cliente, em detrimento a verba para o desenvolvimento do projeto advinda de um edital com data e orçamentos definidos, desta forma, o objetivo foi o produto final acabado.

2.1.2.1 Definição de Requisitos

O sistema desenvolvido é constituído por duas áreas: Área Administrativa (AA) e Site do Projeto (SP). A primeira, é o espaço do sistema responsável por autorizar cadastro de artistas pelo público e, efetivar o gerenciamento geral dos módulos do sistema. O acesso a AA é restrito exigindo uma credencial previamente cadastrada e pode ser categorizado como Administrador ou Redator. Por sua vez, o SP trata-se de um área de acesso público, com livre acesso para consulta dos conteúdos disponibilizados no sistema.

2.1.2.1.1 Funcionalidades e Requisitos Funcionais (RF) da Área Administrativa

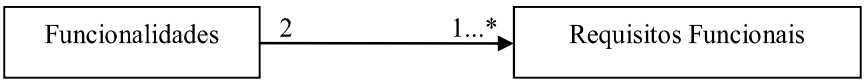
A Área Administrativa possui as funcionalidades de Controle de Acesso e Gerenciamento de Artista, identificadas como F01 e F02 respectivamente. A seguir é apresentado o descritivo de cada uma:

- **Controle de Acesso - F01:** Permitirá que os usuários devidamente cadastrados no sistema se autentiquem através da área de autenticação (e-mail e senha), efetuem *logout*, recuperarem e mudem a senha. Inicialmente, o projeto contém com um módulo de autenticação simples, com dois usuários previamente cadastrados.
- **Gerenciamento de Artista - F02**: Possibilitará ao usuário com perfil de Redator cadastrar os dados gerais, artístico e multimídia do artista; Permitirá ao usuário administrador administrar a solicitação de cadastro de artista advindos externamente através do SP, podendo ele autorizar ou excluir o cadastro.

Segundo Plínio Ventura (2016), uma funcionalidade pode processar diversos requisitos funcionais, implicando que ao desenvolver uma funcionalidade é possível que um ou mais requisitos funcionais possam ser atendidos. Assim, a partir das funcionalidades do sistema, foram definidos os requisitos funcionais necessário para que todos os objetivos de

F01 e F02 fossem cumpridos. Dessa forma, a multiplicidade entre funcionalidade RF da AA está representada no Diagrama 03.

Diagrama 03: Multiplicidade entre Funcionalidade e Requisitos Funcionais



Fonte: Elaboração Própria

Os requisitos funcionais, que representarão as tarefas específicas realizadas pelo sistema, conforme especificações das funcionalidades, estão dispostos na Tabela 01.

Tabela 01: Requisitos Funcionais da Área Administrativa

Funcionalidade	ID	Requisito Funcional	Descrição
F01	RF01	Autenticar Usuário	• Através da entrada *(e-mail* e senha) do usuário correta, realizará a checagem e permitirá seu acesso ao sistema; • Através da entrada (*e-mail* e senha) do usuário incorreta, realizará a checagem e lançará um aviso de dados incorretos não permitindo o acesso ao sistema.
	RF02	Recuperar Senha	• Através da entrada (*e-mail*) do usuário que esteja cadastrado na base de dados, realizará a checagem e enviará para o e-mail informado uma nova senha gerada aleatoriamente; • Através da entrada (*e-mail*) do usuário que não esteja cadastrado na base de dados, realizará a checagem e lançará um aviso de usuário não cadastrado e não fará o envio da nova senha.

	RF03	Alterar Senha	• O usuário logado no sistema, através da entrada (senha atual e nova senha digitada duas vezes) fará: o Checagem da senha atual, caso a entrada esteja conforme o cadastro do banco, passa-se para o próximo passo. Caso a entrada não esteja conforme o cadastro do banco, lançará um aviso de inconsistência de dados; o Checagem da digitação dupla da nova senha, caso as duas entradas estejam iguais, o sistema lançará um aviso de senha alterada com sucesso. Caso as entradas não esteja em conformidade, o sistema lançará um aviso de inconsistência de dados.
	RF04	Efetuar *Logout*	• O usuário autenticado, ao clicar no botão "Sair" o sistema desconectará o usuário e o redirecionará à tela de Autenticação de Usuário.
F02	RF05	Cadastrar Artistas	• Permitirá que usuários do tipo Redator ou Administrador, autenticados na AA, cadastrem artistas inserindo os dados gerais, dados artísticos e dados multimídias. É uma espécie de *wiki* com editor de texto e campos para inserir cada informação. Com todos os campos preenchidos o usuário deve clicar no botão "Confirmar". Caso esteja tudo preenchido corretamente, o sistema armazenará a publicação no banco de dados e publicará no Site do Projeto. Caso haja algum erro, o sistema lançará um aviso de erro de preenchimento de informações.

	RF06	Editar artistas	• Permitirá que usuários do tipo Redator ou Administrador, autenticados na AA, editem todos os dados de um artista cadastrado. Ao realizar as devidas alterações, caso esteja tudo preenchido corretamente, o sistema armazenará a publicação no banco de dados e publicará no Site do Projeto. Caso haja algum erro, o sistema lançará um aviso de erro de preenchimento de informações.
	RF07	Listar artistas	• Permitirá que usuários do tipo Redator ou Administrador, autenticados na AA, visualizem todos os artistas cadastrados no sistema.
	RF08	Excluir artistas	• Permitirá que usuários do tipo Redator ou Administrador, autenticados na AA, excluam artistas cadastrados no sistema. Ao clicar no botão “Excluir”, todos os dados do artista serão apagados do sistema, sendo consequentemente apagado do Site do Projeto.

	RF09	Autorizar cadastro de artistas	• Permitirá que o usuário Administrador, autenticado na AA, autorize ou exclua o cadastro de artista advindo de usuários público através do Site do Projeto; • Através do botão "Aprovar", o sistema lançará um aviso de certificação de aprovação, clicando em "Sim" o artista será cadastrado na base de dado. Clicando em "Não", o aviso sairá da tela, e o usuário ficará na tela de autorização de cadastro de artista; • Através do botão "Reprovar", o sistema lançará um aviso de certificação de aprovação, clicando em "Sim" a solicitação de cadastro de artista será deletada da lista e o usuário não será cadastrado na base de dados. Clicando em "Não", o aviso sairá da tela, e o usuário ficará na tela de autorização com a solicitação de cadastro de artista mantida na lista e com o usuário não cadastrado na base de dados.

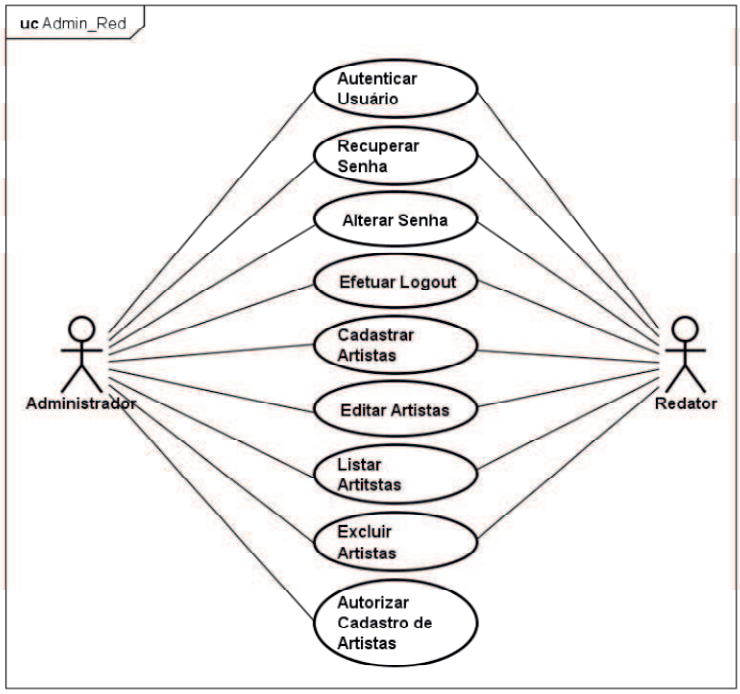
Fonte: Elaboração Própria

As funcionalidades e os requisitos funcionais dependem da existência de usuários. Na Área Administrativa existem o Administrador e o Redator. As descrições de permissões de cada perfil estão dispostas a seguir:

- **Redator**: possui acesso ao Módulo de Artistas, tem total liberdade para gerenciar os dados gerais, artísticos e multimídia do artista, podendo cadastrar, editar, excluir e listar;
- **Administrador**: possui todos os privilégios de acesso do perfil de Redator, além de deter a liberdade de validar os dados do artista externo pelo Módulo de Autorização de Cadastro de Artistas.

Através do Diagrama de Caso de Uso ilustrado no Diagrama 04 é possível observar a relação entre funcionalidades e os tipos de usuários da Área Administrativa.

Diagrama 04: Diagrama de Caso da Área Administrativa



Fonte: Elaboração Própria

2.1.2.1.2 Funcionalidades e Requisitos Funcionais (RF) do Site do Projeto

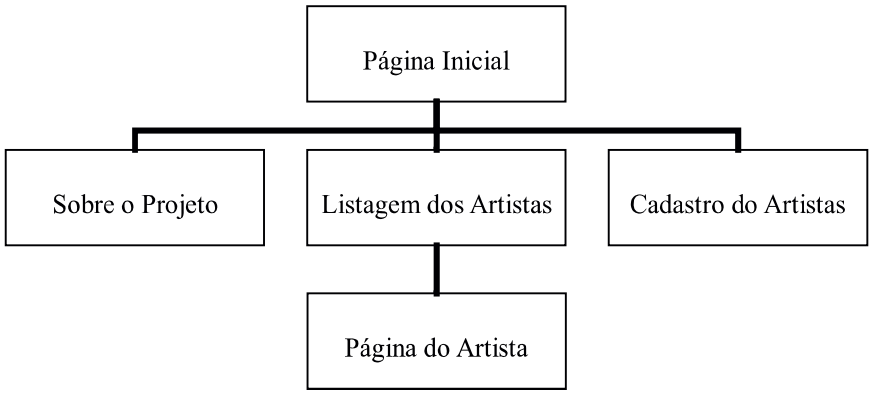
O Site do Projeto, por ser uma área pública, tem a principal função de apresentar informações para o público, no entanto, como a proposta considerou que usuários externos enviassem solicitações para o cadastro de novos artistas, foi necessário adicionar funcionalidades também a essa área.

Assim sendo, o SP possui as funcionalidades de Apresentar Informações, Enviar Solicitações de Cadastro de Artistas e Buscar Artistas, identificadas como F03, F04 e F05 respectivamente. A seguir é apresentado o descritivo de cada uma:

- **Apresentar Informações – F03:** Irá apresentar ao acesso público para consulta os conteúdos cadastrados ou aprovados através da Área Administrativa. As informações poderão está no formato de textos, imagens vídeos, *links* e, distribuídas nas páginas do SP conforme o propósito de cada uma.
- **Enviar Solicitações de Cadastro de Artistas – F04:** Possibilitará que o usuário externo do tipo Público envie uma solicitação de cadastro com dados gerais, artístico e multimídia do artista;
- **Buscar Artistas – F05:** Apresentará uma lista com todos os artistas ou filtrado pelo usuário, através da barra de busca. Esta lista apresentará os dados essenciais do artista

Para definir os requisitos funcionais das funcionalidades, primordialmente a F03, fez-se necessário definir as páginas do SP com os tipos de conteúdo e a dinâmica de ligação entre elas. Através do Diagrama 05 é possível observar a dinâmica entre as cinco páginas que compõem o SP e como o fluxo de acesso do usuário poderá ser realizado.

Diagrama 05: Mapa do Site do Projeto



Fonte: Elaboração Própria

A descrição geral das páginas está listada a seguir:

- **Página Inicial:** Página raiz que possuirá o menu de acesso às demais páginas, além de informações expositivas com textos e/ou imagens;
- **Sobre o Projeto:** Acessada através da "Página Inicial", possuirá informações expositivas com textos e/ou imagens acerca do Projeto Mais Cultura Zélia Braz;
- **Listagem dos Artistas:** Acessada através da "Página Inicial", possuirá a funcionalidade Buscar Artistas e apresentará os dados essenciais sobre o artista pesquisado, através de textos e *link* de acesso a "Página do Artista";
- **Página do Artista:** Acessada através da "Listagem dos Artistas", possuirá todos os dados previamente cadastrados na AA referente ao artista selecionado.

Baseado nos propósitos das páginas, foram definidos os requisitos funcionais, que representarão as tarefas específicas realizadas pelo SP, e que estão dispostos na Tabela 02.

Tabela 02: Requisitos Funcionais do Site do Projeto

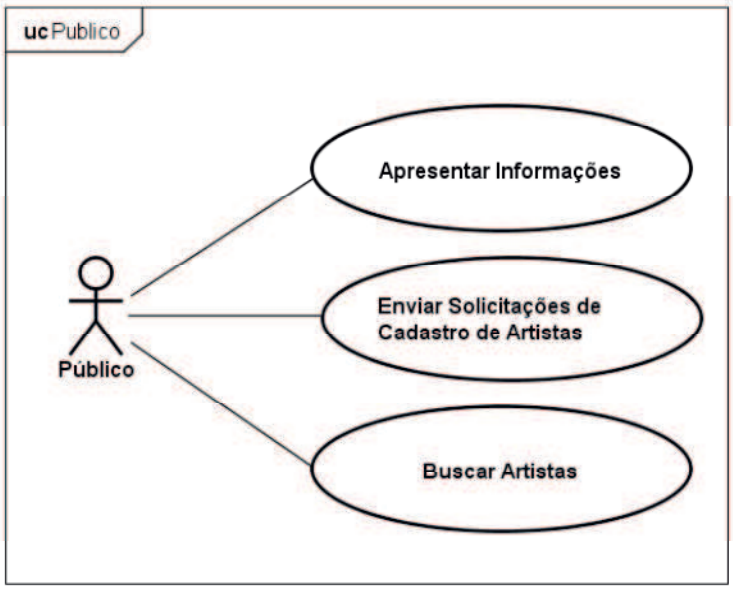
Funcionalidade	ID	Requisito Funcional	Descrição
F03	RF10	Apresentar Informaçõ es na Página Inicial	• Carregar na área delimitada para o conteúdo da "Página Inicial" as informações cadastradas na Área Administrativa, sejam elas textos ou imagens.
	RF11	Apresentar Informaçõ es na página Sobre o Projeto	• Carregar na área delimitada para o conteúdo da página "Sobre o Projeto" as informações cadastradas na Área Administrativa, sejam elas textos ou imagens.
	RF12	Apresentar Informaçõ es na Página do Artista	• Carregar na área delimitada para o conteúdo da "Página do Artista" as informações de dados gerais, dados artísticos e dados multimídias cadastrados na Área Administrativa.
F04	RF13	Enviar Solicitaçã o de Cadastro de Artista	• Permitir que usuários do tipo Público, através da página "Cadastro de Artista", cadastrem artistas inserindo os dado gerais, dados artísticos e dados multimídias. É uma espécie de *wiki* com editor de texto e campos para inserir cada informação. Com todos os campos preenchidos o usuário deve clicar no botão "Confirmar". Caso esteja tudo preenchido corretamente o sistema enviará para a aprovação do Administrador da AA. Caso haja algum erro, o sistema lançará um aviso de erro de preenchimento de informações.
F05	RF14	Buscar Artista	• Permitirá, a partir de um botão de "Busca" que o usuário do tipo Público, insira o nome do artista que deseja visualizar as informações e solicite que o sistema apresente os dados na tela. O

			mecanismo de busca irá verificar a existência dos dados inseridos na base de dados. Podendo ocorrer: o Caso os dados não sejam identificados na base de dados, o sistema lançará um aviso de artista não cadastrado na base de dados; o Caso os dados estejam presente da base de dados, o sistema irá carregar na tela de "Busca de Artista" os dados essenciais do(s) artista(s) encontrado.

Fonte: Elaboração Própria

Através do diagrama de caso de uso ilustrado no Diagrama 06 é possível observar a relação entre funcionalidades e usuário do tipo Público do Site do Projeto.

Diagrama 06: Diagrama de Caso de USP do Site do Projeto



Fonte: Elaboração Própria

2.1.2.1.3 Requisitos Não-Funcionais (RNF) Gerais

Para garantir que o produto final do projeto atenda as funcionalidades especificadas, foram definidos os Requisitos Não-Funcionais gerais para o projeto como um todo. Os RNFs especificados foram pensados tanto para o SP quanto para a AA e alguns foram considerados

mediante às necessidades de um *software* educacional, a exemplo da usabilidade, tempo de resposta (*feedback*), entre outros.

Tabela 03: Requisitos não-funcionais gerais

Segurança	
RNF01	**Autenticação/Alteração de senha** • Autenticação do usuário com criptografia de senha; • Recuperação de senha de usuários com geração aleatória e criptografada. **Aplicação** • Proteção de acesso às *API*s através de autenticação.
Desempenho	
RNF02	**Tempo de resposta** • As consultas de dados devem durar menos de 1 minutos; • As mensagens de erros devem ser lançadas no máximo 15 segundos após sua ocorrência; • *Upload* de arquivos com limitação de 10MB.
RNF03	**Acesso simultaneamente** O sistema não limitar acessos simultâneos.
Confiabilidade	
RNF04	**Política de *backup*** • O servidor de hospedagem deve realizar um *backup* automático a cada 72 horas.
Portabilidade	
RNF05	**Sistema *Web*** • Sistema deve ser web e compatível com todos os navegadores que tenham compatibilidade com *html5*; • Sistema *web* deve ser responsivo se adaptando às telas de dispositivos móveis.
Interoperabilidade	
RNF06	**Integração com *API*** • O sistema deve se concetar com a *API* da aplicação.
Usabilidade	

RNF07	• As consultas de dados devem ser realizadas com no máximo 3 cliques; • Evitar uso de múltiplas barras de rolagens; • Bom constrate entre fundo, formas e fontes; • Uso de ícones/texto em botões que deem significado a sua ação; • Equilíbrio entre tamanho das formas, objetos e fontes; • Linguagem usual para as mensagens de comunicação com o usuário; • Nas telas de cadastro, dispor de editor de texto similar aos que são comumentes encontrados no mercado; • Oferecer ao usuário controle de suas ações.

Fonte: Elaboração Própria

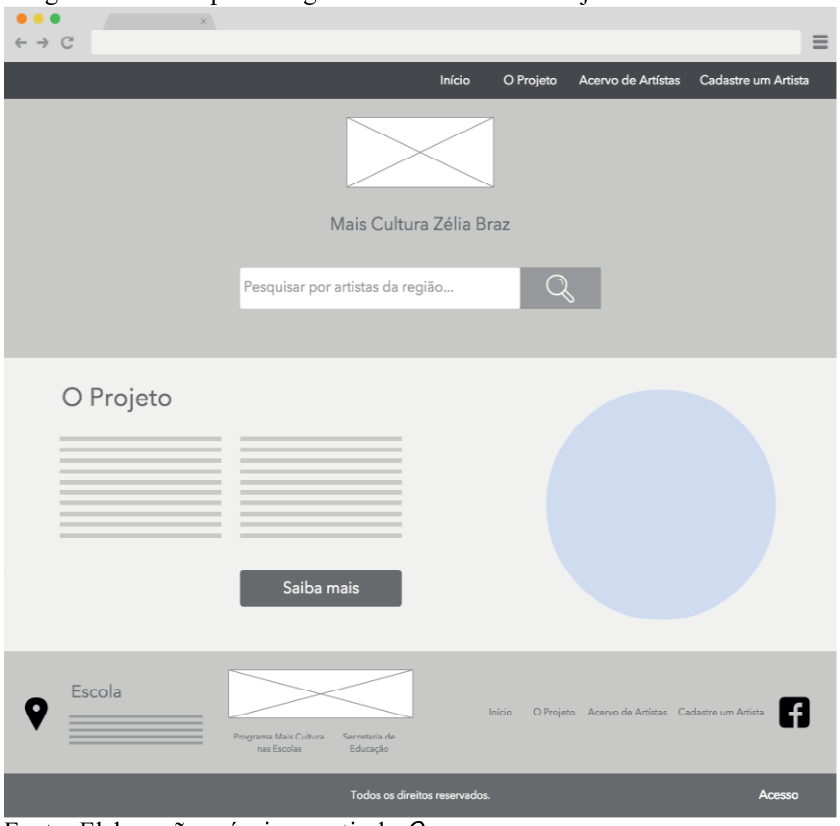
2.1.2.1.4 Proposta de Prototipação Estática

Objetivando ilustrar as características fundamentais da proposta de *design* da interface do SMCZB, e obter *feedbacks* advindos do cliente, foi construída a prototipação estática do projeto, contemplando as páginas do Site do Projeto e da Área Administrativa.

Para ilustrar a prototipação dinâmica, àquela que apresenta a interação entre páginas propostas, além de representar a distribuição das informações das páginas como áreas de textos, imagens vídeos, botões de *input* e *output*, foi utilizado a ferramenta de edição de diagramas nas nuvens *Cacoo* (Cacco, 2017).

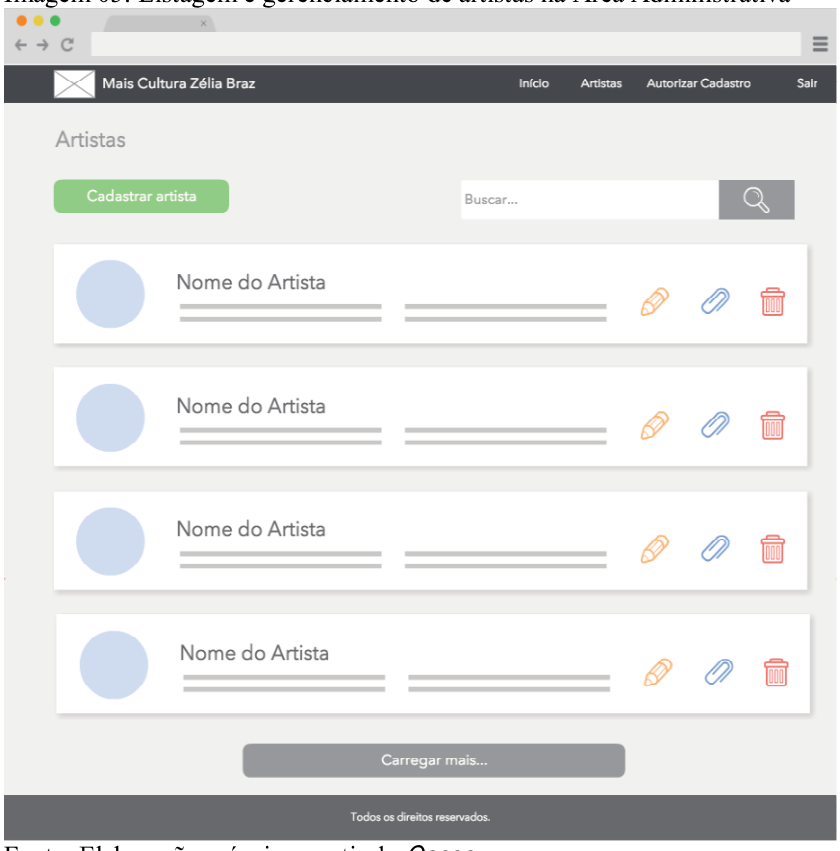
As Imagens 02 e 03 ilustram o detalhamento da prototipação realizada. O projeto completo se encontra no Anexo A deste estudo, no entanto, é possível visualizar as transições de telas com *hiperlinks* através do endereço eletrônico: https://goo.gl/1apI8V, clicando no menu "Ação de Diagrama" > "*Presentation*

Imagem 02: Protótipo da Página Inicial do Site do Projeto



Fonte: Elaboração própria a partir do *Cacoo*

Imagem 03: Listagem e gerenciamento de artistas na Área Administrativa



Fonte: Elaboração própria a partir do *Cacoo*

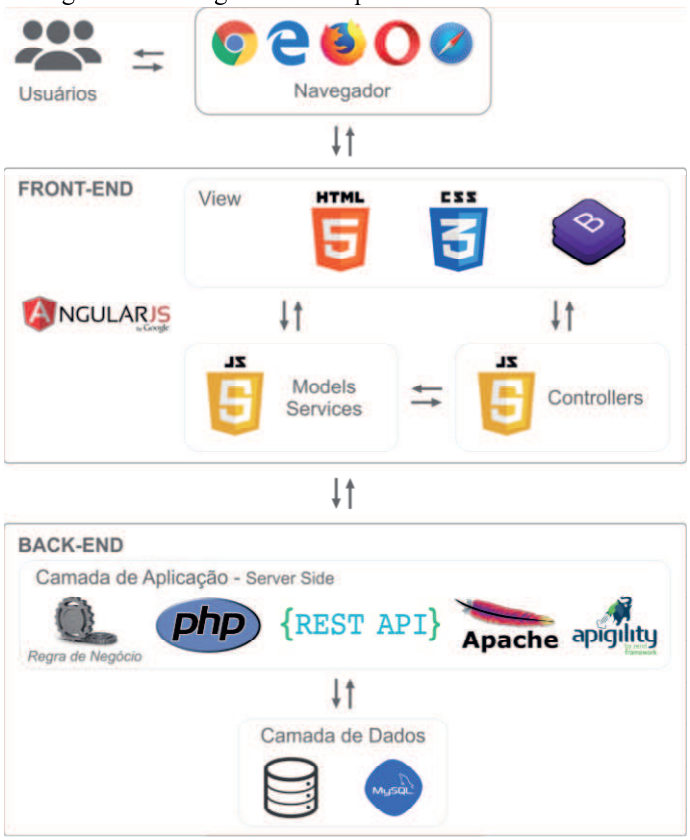
2.1.2.2 Projeto de Sistema e Software

O escopo do projeto contemplou a implementação de duas áreas, cada uma com funcionalidades específicas, mas que convergiam para o resultado final do funcionamento do sistema. O Site do Projeto possui a função de expor informações, mesmo tendo alguns procedimentos de buscas e listagens, mas o objetivo final é a visualização dos dados de artistas. A Área Administrativa tem em sua essência funcionalidades ativas que são as responsáveis pelo cadastramento das informações visualizadas do SP e a administração do projeto como um todo.

Desse modo, para atender a variedade das áreas do sistema, o desenvolvimento foi organizado em *Front-End* e *Back-End*. Segundo Viana (2017) "o desenvolvedor *front-end* é responsável por 'dar vida' à interface. Trabalha com a parte da aplicação que interage diretamente com o usuário." Dessa forma, é importante levar em consideração critérios de interação do usuário com a aplicação como por exemplo, usabilidade e navegabilidade. Já ao abordar o contexto de *Back-End,* Vianna sugere que "o desenvolvedor *back-end* trabalha na parte de 'trás' da aplicação. Ele é o responsável, em termos gerais, pela implementação da regra de negócio." Desse modo, é direcionado à parte não visual sendo responsável pela implementação das regras de negócio do sistema.

Para cada área acima citada, foram definidas tecnologias específicas, considerando as mais atuais e que poderiam dá suporte a implementação dos requisitos funcionais e não funcionais definidos. A partir do Diagrama 07 é possível observar de maneira geral a arquitetura do projeto de *software*, com a organização do *Front-End* e *Back-End*, suas tecnologias e a forma de atuação de cada uma.

Diagrama 07: Diagrama de Arquitetura de *Software*



Fonte: Elaboração própria

No *Front-End*, para construir a estrutura das páginas, tanto do Site do Projeto, quanto da Área Administrativa, foi utilizada a linguagem de marcação e hipertexto, tradução do inglês *Hypertext Markup Language (HTML5)* por oferecer suporte nas áreas de compatibilidade, utilidade, interoperabilidade e acesso universal; requisitos primordiais para uma aplicação educacional de qualidade. A customização das páginas foi realizada através da versão 3 da folha de estilo em cascata, expressão traduzida do inglês *Cascading Style Sheets* (*CSS*). Segundo o site *W3C* o *CSS* é "um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) aos documentos *web*", e pela riqueza multimídia do projeto SMCZB pretendeu-se dispor da melhor ferramenta de estilização vigente.

Primando pela responsividade das páginas do sistema, foi empregado o *Bootstrap* (*Bootstrap*, 2016) em sua terceira versão. O recurso foi escolhido por ser o maior e mais popular conjunto de ferramentas de código aberto para o desenvolvimento com *HTML*, *CSS* e *JavaScript*. Com o objetivo de criar um *Front-End* melhor estruturado, com desenvolvimento de interface *web* dinâmica, sem necessidade de manipular o *Document Object Model* (DOM), foi empregado o *framework JavaScript AngularJS*. Criado pelo *Google*, esse *framework*

implementa o padrão *Model View Whatever* (MVW) e tem como principal característica estender o *HTML* possibilitando um desenvolvimento descomplicado.

Para construir a estrutura do *Back-End* foi necessário fazer uso de tecnologias que implementassem as camadas de aplicação e de dados, conforme ilustrado no Diagrama 07. Na camada de aplicação, foi desenvolvido um *web services* provendo *Application Programming Interface* (*API*) para acesso aos serviços que implementam as regras de negócios do projeto que são de sua responsabilidade. A *API* foi arquitetada com o apoio do construtor *APIgility* da *Zend Framework*, idealizado para simplificar a criação e manutenção de *APIs*, além de facilitar o consumo das mesmas e gerar um produto bem estruturado. A programação da *API* foi implementada em *Hypertext Preprocessor* (*PHP*) juntamente com *o Zend Framework* 2, e em detrimento ao uso do *PHP* fez-se necessário fazer uso do servidor *web Apache e* na camada de dados, o banco *MySQL*.

2.1.2.3 Implementação

A etapa de implementação é destinada a codificação de todas as especificações levantadas até então. Nesse momento, é desenvolvido a compilação e construção de códigos que serão executados pela aplicação, e assim, oferecer um produto de *software.*

Para descrever a metodologia utilizada na implementação do projeto SMCZB, foi feito uso do paradigma de programa Orientado a Objeto (OO). Para exemplificar no presente trabalho a codificação realizada no desenvolvimento do projeto, foi utilizado como referência o requisito funcional RF07 que ilustra a implementação da funcionalidade "Listar artistas" com maior profundidade. Além disso, serão apresentadas as duas faces da RF07, no âmbito do *Front-End* e *Back-End.*

2.1.2.3.1 *Front-End*

A implementação da *view* foi realizada utilizando *HTML5*, *CSS3*, juntamente com *Bootstrap*. Através do fragmento de código ilustrado na Imagem 04, é possível observar as classes de estilo "*col-sm-2*" e "*row*" características do *Bootstrap*, indicando sua aplicabilidade. Além disso, com o uso do *framework Angu*larJS foi possível assegurar maior organização e simplicidade do código conforme observado no fragmento de código.

Imagem 04: Fragmento de código da *view* artista.listar.html

```
<section id="main-content">
    <section class="wrapper">
        <h3>
            <i class="fa fa-users"></i> Artistas
        </h3>

        <button class="btn btn-success btn-lg m-t-10 m-b-15" ui-sref="artista-cadastrar">
            <i class="fa fa-user-plus"></i> Novo Artista
        </button>

        <div class="row">
            <div class="col-sm-2 mb" ng-repeat="artista in ArtistaListarCtrl.artistas">
                <div class="content-panel pn">
                    <div class="box-artista" ui-sref="artista-editar({artista: artista})" role="button"
                         ng-style="{'background-image': 'url(http://api.maisculturazeliabras.com.br/foto/' + artista.foto_perfil)}">
                        <div class="user">
                            <h4>{{artista.nome_artistico}}</h4>
                            {{artista.estilo}}
                            <br>
                            <span class="label label-success" ng-if="artista.ativo == 1">Ativo</span>
                            <span class="label label-danger" ng-if="artista.ativo == 0">Inativo</span>
                        </div>
                    </div>
                    <div class="pr2-social centered">
                        <a ui-sref="artista-editar({artista: artista})" title="Editar" role="button">
                            <i class="fa fa-edit text-theme03"></i>
                        </a>
                        <a title="Dados Multimidias" role="button" ui-sref="multimidia({artista: artista})">
                            <i class="fa fa-photo text-theme04"></i>
                        </a>
                        <a ng-click="ArtistaListarCtrl.excluir(artista.id)" title="Excluir" role="button">
                            <i class="fa fa-trash text-danger"></i>
                        </a>
                        <a href="#" title="Visualizar" role="button">
                            <i class="fa fa-eye text-theme02"></i>
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </section>
</section>
```

Fonte: Elaboração própria

O fragmento de código da Imagem 04 tem como objetivo estruturar as informações advindas do *webservices*, para apresentação para o usuário. O trecho de código que inicia na linha 12 até e finaliza na linha 39, será repetido *n* vezes até que todos os artistas cadastrados na base de dados tenham sido apresentados. Isso é possível através da diretiva do *AngularJS* "*ng-repeat*" que permite iterar a lista de artistas de forma simples e intuitiva, e assim exibir cada dado na *view* utilizando a sintaxe {{valor}}, proporcionando a comunicação entre o v*iew* e *model* do *AngularJS*. Por fim, é importante destacar que os atributos das *tags* prefixadas com "*ng-*" são diretivas que permitem estender o *DOM* e acrescentar novos artifícios para simplificar a apresentação dos dados na *view*.

O vínculo entre *view* com o *model* foi realizado através do componente *controllers* do *AngularJS*, que disponibiliza os comportamentos e os dados para a *view*, consequentemente favorece organizar a aplicação conforme o fragmento de código da Imagem 05.

Imagem 05: Fragmento de código do *controller* artista.listar.controller.js

```
(function (angular)
{
    'use strict';

    angular
        .module('MaisCultura')
        .controller('ArtistaListarController', ArtistaListarController);

    ArtistaListarController.$inject = ['$state', '$ngConfirm', 'CONFIG', 'ArtistaFactory', 'NotificacaoService'];

    function ArtistaListarController($state, $ngConfirm, CONFIG, ArtistaFactory, NotificacaoService)
    {
        var vm = this;
        vm.artistas = [];
        vm.excluir = excluir;

        listarArtistas();

        /**
         * Lista todos os artistas cadastrados no base de dados.
         */
        function listarArtistas()
        {
            ArtistaFactory
                .listar()
                .then(function (retorno)
                {
                    vm.artistas = retorno.data._embedded.artista;
                }, NotificacaoService.tratarErro);
        }

        /**
         * Abre um modal solicitando a exclusão do artista.
         */
        function excluir(id)
        {
            $ngConfirm({
                title: 'Confirmar excluisão!',
                content: "Tem certeza que deseja excluir este artista?!",
                buttons: {
                    excluir: {
                        text: 'Excluir',
                        btnClass: 'btn-red',
                        action: function ()
```

Fonte: Elaboração própria

Por padrão, na inicialização do *controller* é feita a chamada da função "listarArtistas()", que objetiva invocar o *service* "Artista*Factory*" (Imagem 06) recebido como argumento do *controller*, fato que caracteriza a injeção de dependência no *AngularJS* que é realizada automaticamente. No "Artista*Factory*" encontra-se a função "listar()" que irá fazer a requisição *HTTP* via método *GET* ao endereço da *API* Artista[2] com uso da biblioteca "padrão(*$http*)" do *AngularJS*. Após, será retornada a lista dos artistas cadastrados na base de dados e, consequentemente será repassada para o *controller* que por sua vez, preenche o *model* "vm.artistas" e envia para a *view*.

[2] URL de acesso da API: http://api.maisculturazeliabras.com.br/artista

Imagem 06: Fragmento de código do *service* artista.*factory*.js

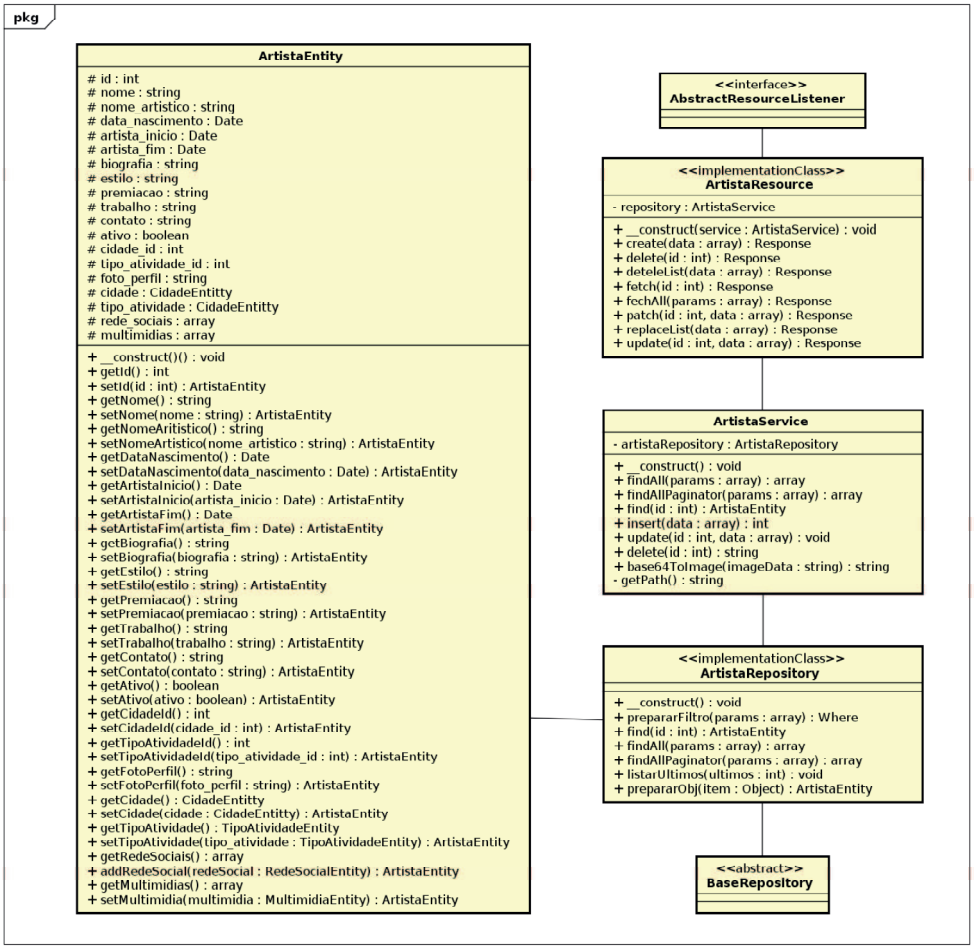
```
(function (angular)
{
    'use strict';

    angular
        .module('MaisCultura')
        .service('ArtistaFactory', ArtistaFactory);

    ArtistaFactory.$inject = ['$http', 'CONFIG'];

    function ArtistaFactory($http, CONFIG)
    {
        var url = CONFIG.API_URL + 'artista';

        return {
            consultar: consultar,
            editar: editar,
            excluir: excluir,
            listar:listar,
            salvar: salvar
        };

        /**
         * Consulta os dados do artista.
         */
        function consultar(id)
        {
            return $http.get(url + '/' + id);
        }

        /**
         * Edita a informação do artista.
         */
        function editar(artista)
        {
            return $http.put(url + '/' + artista.id, artista);
        }

        /**
         * Lista os artistas com paginação.
         */
        function listar()
        {
            return $http.get(url);
        }

        /**
         * Lista todas as cidades de um UF.
```

Fonte: Elaboração própria

2.1.2.3.2 *Back-End*

Para implementar a camada de aplicação do requisito funcional RF07, primeiramente foi construído o diagrama de classe (Diagrama 08) que retrata a interação entre as classes dentro do *APIgility*.

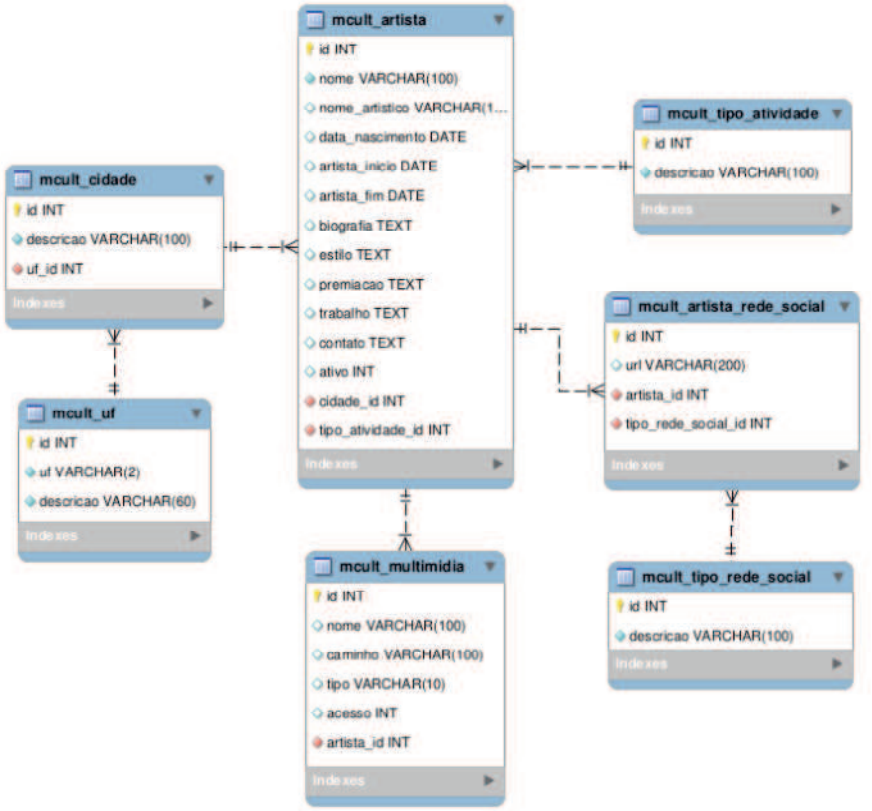
Diagrama 08: Diagrama de classe do requisito funcional RF07



Fonte: Elaboração própria

Em complemento, foi arquitetado o diagrama de entidade relacionamento (Diagrama 09) com intuito de vislumbrar as tabelas que integram a camada de dados. Ambos os diagramas foram seguidos como recurso para a implementação geral do RF07.

Diagrama 09: Diagrama de entidade relacionamento do requisito funcional RF07



Fonte: Elaboração própria

Todas as requisições *HTTP* para a *API* Artistas são direcionadas para a classe Artista*Resource* (Imagem 07), responsável por processar e retornar o "*Response* válido", ou seja, a resposta do servidor que pode ser a lista de artista ou uma mensagem de erro. No fragmento de código da Imagem 07, ao requisitar a *URL* via método *GET* é executado a função "*fetchAll*()" que recebe como parâmetro um *array* de *string* contendo a palavra-chave "listar". Havendo a existência da palavra-chave, a lista dos artistas será apresentada sem paginação, caso contrário será empregado o recurso de paginação. Por fim, o método repassa a solicitação de listagem para a classe "Artista*Service"* (Imagem 08).

Imagem 07: Fragmento de código do Artista*Resource*.php

```
80      /**
81       * Fetch all or a subset of resources
82       *
83       * @param  array $params
84       * @return ApiProblem|mixed
85       */
86      public function fetchAll($params = array())
87      {
88          if (isset($params['listar'])) {
89                  $res = $this->repository->findAll($params);
90          } else {
91                  $res = $this->repository->findAllPaginator($params);
92          }
93
94          return $res;
95      }
```

Fonte: Elaboração própria

Considerando o princípio da responsabilidade única de OO, foi criado o arquivo "Artista*Service*" responsável pela implementação das regras de negócios da *API* e quando necessário, realizar chamadas ao banco de dados através da classe "Artista*Repository*" (Imagem 09).

Imagem 08: Fragmento de código do Artista*Service.php*

```php
<?php
namespace ApiModule\V1\Rest\Artista;

class ArtistaService {

        private $artistaRepository;
        private $artistaRedeRepository;
        private $artistaMultRepository;

        /**
         * OrdersService constructor.
         * @param $repository
         */
        public function __construct($serviceLocator)
        {
                $this->artistaRepository = new ArtistaRepository($serviceLocator);
                $this->artistaRedeRepository = new ArtistaRedeSocialRepository($serviceLocator);
                $this->artistaMultRepository = new ArtistaMultimidiaRepository($serviceLocator);
        }

        /**
         *
         * @param unknown $params
         * @return \ApiModule\V1\Rest\Artista\unknown[]
         */
        public function findAll($params)
        {
                return $this->artistaRepository->findAll($params);
        }

        /**
         *
         * @param unknown $params
         * @return \Api\V1\Rest\User\UserCollection
         */
        public function findAllPaginator($params)
        {
                return $this->artistaRepository->findAllPaginator($params);
        }

        /**
         *
         * @param unknown $id
         * @return mixed
         */
        public function find($id)
        {
                return $this->artistaRepository->find($id);
```

Fonte: Elaboração própria

Para fechar o ciclo de iteração entre as classes, foi codificado a classe "Artista*Repository*" (Imagem 09) responsável por tratar das regras de banco de dados, onde é solicitado ao método "*findAllPaginator*()" que efetiva a busca da lista de artista no banco de dados e retorna-os para "Artista*Service*" que por sua vez retorna para o "Artista*Resource*" em formato *JavaScript Object Notation* (*JSON*).

Imagem 09: Fragmento de código do Artista*Repository*.php

```php
    public function findAll($params)
    {

        $where = $this->prepararFiltro($params);

        $itens = parent::findAll($where, 'nome');

        foreach ($itens as $iten) {
                if (is_object($iten)) {

                        $this->prepararObj($item);

                }
        }

        return $itens;

    }

    /**
     *
     * @param unknown $params
     * @return \Api\V1\Rest\User\UserCollection
     */
    public function findAllPaginator($params)
    {
        $where = $this->prepararFiltro($params);

        $tableGatewary = $this->getTableGateway();
        $paginatorAdapter = new DbTableGateway($tableGatewary, $where, 'nome');

        return new ArtistaCollection($paginatorAdapter, $this->getServiceLocator());
    }
```

Fonte: Elaboração própria

2.1.2.4 Integração e Testes

O teste do sistema foi efetivado através dos "Testes Funcionais", comumente nomeando de testes caixa preta que tem como objetivo analisar as entradas e saídas de dados de um produto de *software*. "Ou seja, ele foca-se na parte externa do *software*, já que quem irá testá-lo não tem acesso aos códigos internos nem sabe sobre a operação interna do *software*. A interface é avaliada, bem como suas funcionalidades" (Testes de *Softwares*, 2016).

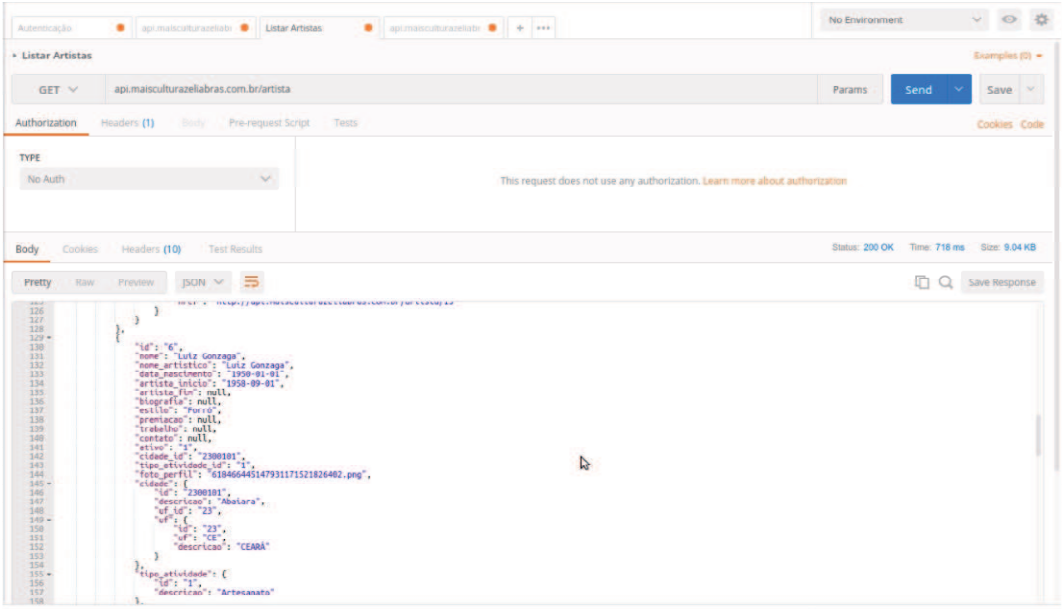
Essa metodologia foi aplicada primeiramente na validação da *API*, visto que sua função é a comunicação entre: o banco de dados, aplicação e a regra de negócio. Dessa forma, se fez necessário garantir seu funcionamento em completude. Para tanto, foi feito uso do aplicativo gratuito *Postman* (2016) que permitiu a realização de testes e depuração da *API REST* através de requisições via *HTTP*.

Por intermédio do *Postman,* foi possível verificar a conformidade do retorno da *API* informando dois dados: *URL* referente ao serviço da *API* de listar após; método *HTTP GET.* O procedimento, quando executado sem erros, oferece como retornou o *JSON* e o *status* da operação de retorno chamado de "*200 OK*", podendo ainda ser verificado o tempo de resposta

da consulta na *API* em milissegundos. No entanto, quando a execução apresenta erros o código “” é lançado sendo necessário a busca e reparo.

A Imagem 10 apresenta o teste caixa preta realizado no serviço “Listar Artistas”. O teste bem sucedido apresentou o *status* “*200 OK*” e um tempo de resposta de 718ms, dessa forma, foi possível validar, na esfera *API*, o requisito funcional RF07 e verificar sua conformidade com o requisito não-funcional RNF02 do projeto.

Imagem 10: Teste Caixa Preta através do *Postman* do RF07 e RNF02



Fonte: Elaboração própria

Para validar, no âmbito da entrada e saída de dados, o requisito funcional RF07, foi definido uma sequência de ações a serem realizadas e os resultados esperados que eram checados item a item. A Tabela 4 foi utilizada como instrumento dos testes caixa preta da RF07.

Tabela 04: Teste Caixa Preta do requisito funcional RF07

Pré-requisito	• Requisito funcional RF05 testado e funcional; • *Login* com usuário do tipo “Administrador” ou “Redator”.		
Passo	**Ação**	**Retorno Esperado**	**Retorno Efetivado**
01	Com nenhum cadastro de artistas gravado banco de dados, acessar a área de listagem de artista.	Mensagem “Nenhum artista cadastrado”.	Conforme ☒ Divergente ☐
02	Com apenas um cadastro de	Lista com apenas um	Conforme ☒

	artistas gravado banco de dados, acessar a área de listagem de artista.	nome de artista conforme cadastrado no banco.	Divergente ☐
03.1	Deletar o único usuário gravado no banco de dados e acessar a área de listagem de artista.	Mensagem "Nenhum artista cadastrado".	Conforme ☐ Divergente ■
03.2	Deletar o único usuário gravado no banco de dados e acessar a área de listagem de artista.	Mensagem "Nenhum artista cadastrado".	Conforme ■ Divergente ☐
04	Com vários cadastros de artistas gravados banco de dados, acessar a área de listagem de artista.	Lista com todos os nomes dos artistas conforme cadastrado no banco.	Conforme ■ Divergente ☐
05	Com vários cadastros de artistas gravados banco de dados, deletar apenas um usuário e acessar a área de listagem de artista.	Lista com todos os nomes dos artistas, com exceção do que foi deletado, conforme cadastrado no banco.	Conforme ■ Divergente ☐
06	Com vários cadastros de artistas gravados banco de dados, editar o nome de um usuário e acessar a área de listagem de artista.	Lista com todos os nomes dos artistas, incluindo o usuário com nome editado, conforme cadastrado no banco.	Conforme ■ Divergente ☐
07	Deslogar da Área Administrativa, limpar a cache do navegador e acessar novamente a área de "Listar Artista".	Lista com todos os nomes dos artistas conforme apresentado na passo 06.	Conforme ■ Divergente ☐
08	Com vários cadastros de artistas gravados banco de dados, deletar todos os usuários e acessar a área de listagem de artista.	Mensagem "Nenhum artista cadastrado".	Conforme ■ Divergente ☐
09	Deslogar da Área Administrativa, limpar a cache do navegador e acessar novamente a área de "Listar Artista".	Mensagem "Nenhum artista cadastrado" conforme apresentado na passo 08.	Conforme ■ Divergente ☐

Fonte: Elaboração própria

A validação de todos os requisitos foram realizadas mediante entradas e checagem das saídas conforme metodologia exemplificada com o requisito funcional e não-funcional RF07 e RNF02 respectivamente.

2.1.3 Cronograma

Para construir e acompanhar o cronograma das atividades do projeto, foi utilizado a ideia do Gráfico de *Gantt*, pois através dele é possível visualizar tarefas, o tempo estipulado para sua realização, bem como acompanhar sua execução observando possíveis atrasos, a fim de identificar facilmente problemas com a execução das atividades do projeto.

O cronograma final acordado junto ao cliente conteve a duração total de 79 dias corridos contando desde as definições de requisitos até a entrega do produto final. A estimativa de dias e real execução das fases do projeto pode ser visualizada do Gráfico 01.

Gráfico 01: Estimativa X Execução Real em quantidade de dias das fases do Projeto

ATIVIDADE	INÍCIO DO PLANO	DURAÇÃO DO PLANO	INÍCIO REAL	DURAÇÃO REAL	PORCENTAGEM CONCLUÍDA
Início	1	1	1	1	100%
Definição de requisito	2	5	2	6	100%
Projeto de sistema de software	8	10	8	10	100%
Implementação	19	53	19	81	100%
Integração e teste	72	7	100	6	100%
Sistema Final	79	1	106	1	100%

DURAÇÃO DO PLANO DO PROJETO (em dias)	79
DURAÇÃO REAL DO PLANO DO PROJETO (em dias)	106

Fonte: Elaboração própria

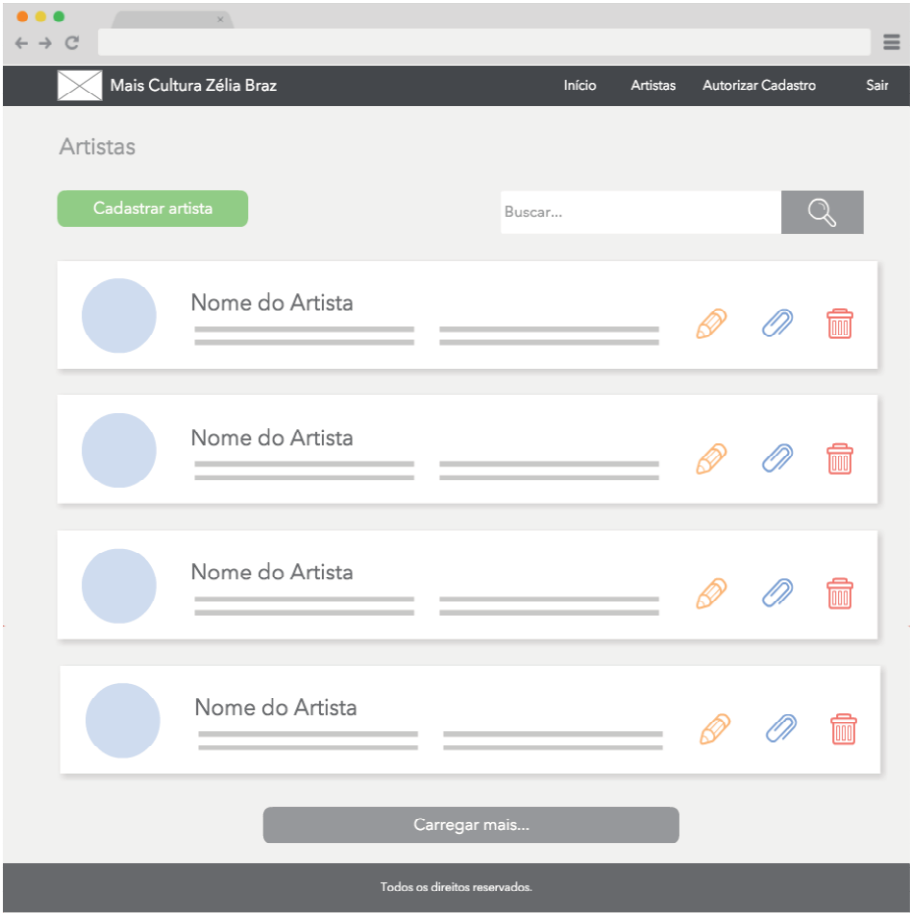
A fase “Implementação” exigiu dias além do planejado, esse fato acarretou atraso em sua finalização e, consequentemente, a fase de “Integração e teste” iniciou após a data prevista, no entanto, sua realização em dias foi menor do que o estipulado. É importante destacar que todas as fases foram concluídas em sua totalidade e, mesmo havendo atraso na entrega do Produto Final, o projeto atingiu os objetivos do cliente.

2.1.4 Sistema Final - Resultados e Discursões

Com a efetivação de todas as fases anteriormente descritas, foi obtido o Sistema Final e entregue ao cliente, no entanto, a entrada em produção não foi realizada devido motivos superiores ao mesmo. Esse fato impossibilitou que o comportamento do SMCZB fosse avaliado, assim como sua aceitação com os usuários, o que tornou o presente estudo puramente focado em questões técnicas da Engenharia de *Software*, visto que não há dados que possa avaliá-lo aos olhos do usuário, ou no âmbito de produção.

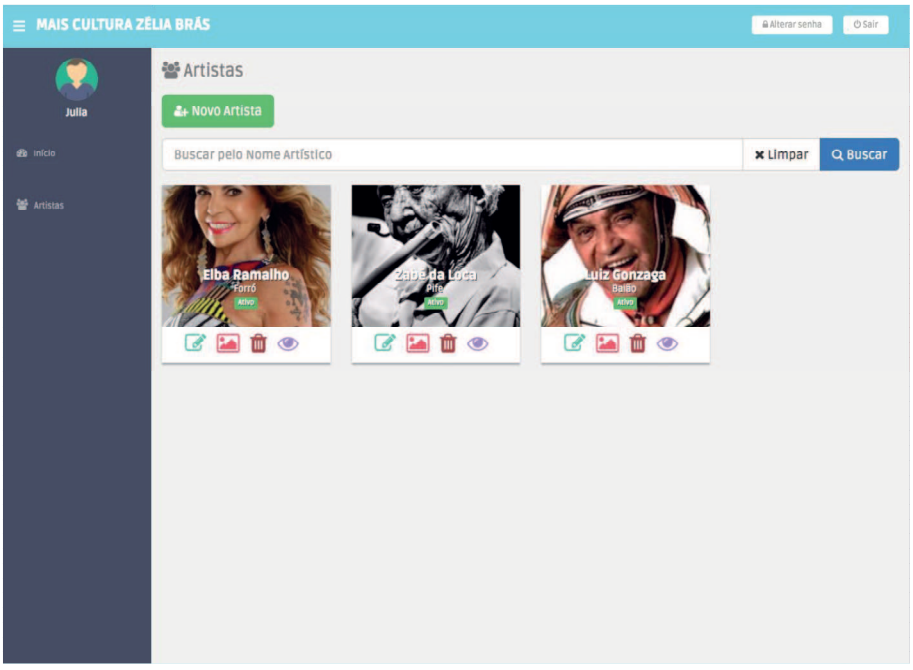
É importante destacar que no Sistema Final entregue, todas as funcionalidades com seus requisitos funcionais e não-funcionais levantados estão presentes e implementados em sua completude, atendendo dessa forma, as necessidades do cliente. Entretanto, durante o processo de implementação do *Fron-End* algumas orientações da prototipagem foram adaptadas para melhor atender as especificações dos requisitos funcionais e não-funcionais, além da necessidade de seguir as orientações das tecnologias utilizadas. Nas Imagens 11 e 12 é possível observar o comparativo entre a proposta prototipada e a versão final desenvolvida da Tela Listar Artista. As demais telas podem ser apreciadas nos Anexo B e C deste documento.

Imagem 11: Tela Listar Artista em formato prototipagem



Fonte: Elaboração própria

Imagem 12: Tela Listar Artista em formato final



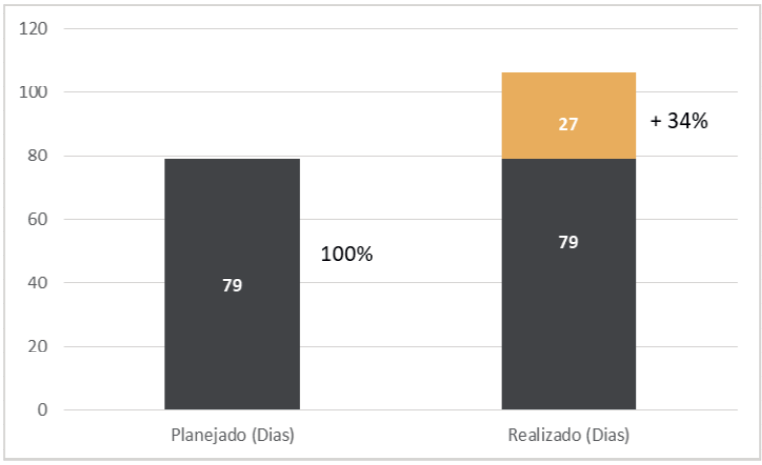
Fonte: Elaboração própria

Através desse paralelo é possível observar e reafirmar a função da prototipagem no processo de desenvolvimento de produtos de *software*. O uso da prática no projeto SMCZB foi importante em dois sentidos: aproximar a proposta de *layout* do sistema com às necessidade do cliente; nortear a implementação do *Fron-End* evitando retrabalho e melhorando o tempo de entrega do Sistema Final.

O principal desafio enfrentado no projeto foi o cumprimento com os prazos estipulados. Inicialmente, o cronograma apresentado para o cliente contemplou a entrega do Sistema Final em 110 dias corridos, mas a urgência para o fechamento do projeto devido as datas do edital ocasionou renegociação para entrega em 79 dias. A renegociação do cronograma acarretou retirada de detalhes importantes do sistema como acessibilidade, teste de unidade e construção da documentação do *software* completa, mas as necessidades do cliente no momento eram prioridades. Vale ressaltar que a negociação antecedeu a fase de "Definição de requisitos" significando que de fatos esses requisitos nunca fizeram parte do projeto.

Embora efetivada mudança no escopo do projeto, a estimativa do tempo foi subestimada necessitando de 27 dias a mais do que foi planejado, número que representa 34% de dia a mais, o Gráfico 02 apresenta os número comparativos do número de dias planejado e o realizado.

Gráfico 02: Dias Planejados e realizado para entrega do sistema final



Fonte: Elaboração própria

Por intermédio do gráfico de *Gantt* foi possível identificar a ocorrência do atraso a partir da fase de "Implementação", mas por não ter sido elaborado um plano de Gestão de Riscos a remediação levou um certo tempo a ser considerada. Como o projeto dispunha de

apenas um profissional que realizou a função de analista, desenvolvedor, *web designer* e programador, além de analisar as questões inerente à função educacional do sistema, viu-se a necessidade de contratação de um segundo programador para o suporte no desenvolvimento da *API*.

Contudo, obteve-se uma diminuição no atraso da entrega do Sistema Final que seria superior à 34% do tempo ultrapassado. Além disso, foi observado que embora o produto seja por encomenda, ou seja, os requisitos bem definidos, a negociação para atender às necessidade do cliente no quesito de "tempo" é um fator de risco para a qualidade do produto em desenvolvimento. Várias práticas que garantem a qualidade do produto ficaram de fora do processo e, outras como o plano de Gestão de Risco, não foram consideradas por falta de atenção da equipe devido a urgência da entrega, contribuíram para o atraso como um todo. Assim, é importante que equipes de desenvolvimento de produtos de *software* considerem todas as possibilidade e entrem em um acordo que possibilite um equilíbrio entre as necessidades do cliente e o tempo de trabalho da equipe, visto que além da qualidade do produto entregue, a referência como profissional se propaga com o artefato desenvolvido.

Em se tratando da arquitetura escolhida, o uso da *API* auxiliou na adversidade da estimativa do tempo, uma vez que serviu como fornecedora de dados para as duas áreas (SP e AA), reduzindo o tempo de implementação. Além disso, ela facilitou a gestão interna permitindo que os ajustes advindos da fase de "Testes e integração" fossem solucionados de maneira rápida, um dos motivos da conclusão da fase ter ocorrido em um tempo menor do que o estimado.

O sistema desenvolvido é uma mescla dos tipos de *software* educacional "Programa" ou "Aplicativo", "*Software* de Autoria" e "Multimídia e *Internet*". Ele possui processadores de textos, áreas de postagens e visualização de imagens, tendo a proposta de ser aplicado em ambiente escolar, no entanto, o projeto fixou o desenvolvimento apenas da ferramenta, a construção do conteúdo será realizada com sua entrada em produção. Isto posto, durante o desenvolvimento do SMCZB foram empregados conhecimentos específicos de concepção educacional de sistemas. Buscou-se obter um ambiente que aproxime o aluno através de uma interface amigável e intuitiva, seguindo as tendências de ferramentas consolidadas e que possivelmente fazem parte do dia-a-dia dos usuário. O objetivo foi promover o aprendizado das funcionalidades da ferramenta através do uso, e empregar um dos principais paradigmas de *software,* desenvolver produtos de software com qualidade dentro dos custos adequado.

A concepção do sistema como um todo foi pensada na facilidade do uso, no tempo de resposta e na qualidade de resposta para o usuário, tentando manter uma comunicação

didática, conceitos empregados graças aos conhecimentos pedagógico do Licenciado em Computação que executou o desenvolvimento do sistema aqui descrito. Embora haja muitas incertezas quanto ao campo de atuação do profissional com a formação citada, é importante destacar que o mercado de trabalho necessita de profissionais com habilidades equilibradas nas duas áreas (técnica e pedagógica), atualmente a tecnologia educacional evolui a passos largos e cada vez mais haverá espaço para atuação.

Por fim, mas não menos importante, a aplicação do modelo de desenvolvimento em cascata permitiu que o andamento das atividades do projeto andasse sem a dependência do cliente. As especificações inicialmente levantadas e validadas deram condições aos desenvolvedores de efetivar as etapas sem a dependências de fatores externos, embora tenha havido atraso é relevante deixar claro que o modelo aplicado não foi a causa do atraso, mas sim a estimativa de tempo. Ao contrário, a aplicação do modelo cascata no contexto auxiliou na redução do tempo. O uso de modelos que necessitassem de reuniões periódicas, acarretaria no aumento do atraso, uma vez que a indisponibilidade de agenda e distância entre cliente e equipe de desenvolvimento demanda tempo.

Desta forma, para a escolha do modelo de desenvolvimento de um projeto de *software*, é imprescindível o estudo do negócio do cliente para entender bem o problema que se deseja resolver. Com isso deve-se levar em consideração que a construção de produtos de *software* requer aplicação de conceitos da Engenharia de *Software,* mas, esses conceitos muitas vezes perpassam o uso de técnicas e diretrizes complexas. A boa aplicação da ES é fundamentada no equilíbrio entre o propósito do produto a ser desenvolvido e a metodologia aplicada, pois a finalidade é construir o produto de qualidade com custos apropriados.

CONCLUSÃO

O presente trabalho apresentou o processo de desenvolvimento do Sistema Mais Cultura Zélia Braz uma iniciativa de uma Unidade Municipal de Ensino da cidade de Sumé-PB, realizada com o apoio do Governo Federal. O sistema tem como objetivo oferecer um acervo digital multimídia dos artistas do cariri paraibano construído por alunos da escola, além de despertar nos alunos novas possibilidades, promovendo a inclusão digital, impulsionando o senso crítico permitindo que atuem de maneira interdisciplinar.

Para o desenvolvimento do sistema foi empregado o Modelo em Cascata, com as fases de “Definição de requisitos”, “Projeto de sistema e *software*”, “Implementação”, “Integração e testes” e “Sistema final”. Durante o processo, tentou-se aplicar conceitos da

Engenharia de *Software* de maneira equilibrada, considerando as necessidade do cliente as práticas e tecnologias mais indicadas, sendo demonstrado, através das boas práticas e lições aprendidas, que a boa aplicação da ES é fundamentada no equilíbrio entre o propósito do produto a ser desenvolvido e a metodologia aplicada.

Os resultados apresentados foram obtidos através da prática e dos artefatos produzidos durante o desenvolvimento do SMCZB. Com isso foi possível demonstrar uma aplicação do modelo de desenvolvimento em cascata; descrever de maneira detalhada as etapas de desenvolvimento de um produto de *software* destina à educação e cultura; discutir através do produto final, o potencial de um Licenciado em Computação para o mercado de desenvolvimento, tanto no tocante às questões técnicas quanto às educacionais.

Como trabalhos futuros, é pretendido acompanhar a entrada em produção do sistema para acompanhar as respostas mediante interação dos usuários e, consequentemente realizar a etapa de "Manutenção e Evolução" não contemplada no escopo do projeto aqui apresentado. Além disso, é almejado ainda investigar a opinião do usuário com intuito de receber *feedback* quanto a aceitação e validação da técnica utilizada no desenvolvimento da ferramenta.

DESENVOLVIMENTO DE UM SISTEMA PARA EDUCAÇÃO E CULTURA

ABSTRACT

This paper introduces the development process of the "Mais Cultura Zélia Braz" System (SMCZB), an initiative of a Municipal Education Unit in the city of Sumé- PB, held with the support of the Federal Government. The system aims to allow students from this Unit to build a multimedia digital collection of artists from the Cariri, in Paraíba. For the system development, was used the cascade model to 'define the requirements', 'system and software project', 'implementation', 'integration and testing', and 'the final system' phases. During its process, was also applied Software Engineering concepts (SE) in a balanced way, considering the needs of the client; the most appropriate techniques and technologies, demonstrating through good practices and lessons learned that the good application from the SE is based on a balance between the purpose of the product to be developed and the methodology used. The results obtained were extracted from the activities performed throughout the project and its artifacts produced, demonstrates the potential of cascade development model. In addition, it favors a discussion based on the final product about the potential of a licentiate in Computing Science for the development market, both in technical and educational matters. This work aims to assist software developers in making decisions about the choice of development model, technologies and architectures.

Keywords: Software Engineering. Cascade Model. Development.

REFERÊNCIAS

AngularJS. Disponível em < https://angularjs.org/> Acesso em: 28/11/2017.

Bootstrap. Disponível em <https://getbootstrap.com/docs/3.3/> Acesso em: 23/09/2016.

Cacoo. Disponível em < https://cacoo.com> Acesso em: 24/11/2017.

Emydio, M. M.; Rocha, R. F. **Gestão do Conhecimento na Área Educacional: a Tecnologia como Instrumento Facilitador**. In. Simpósio de excelência em gestão e tecnologia, 9., 2012. Resende. Anais eletrônicos... Resende: AEDB, 2012.

Fernandéz, D. M., Wagner, S., Kalinowski, M., Schekelmann, A., Tuzcu, A., Conte, T., Prikladnicki, R. (2015). **Naming the pain in requirements engineering: Comparing practices in brazil and germany**. IEEE Software..

Ferreira, M. A. D.; Rodrigues, A. N. (2015**). Interfaces Educativas: Implicações de design e processos cognitivos do jogo Nicetown,** In: Congresso brasileiro de informática na educação - CBIE, 2015.

Gomes, A. S. (2015**). A culpa é do celular?**. Disponível em: <http://www.abble.com.br/a-culpa-e-do-celular/> Acesso em: 27/10/2017.

GOVERNO FEDERAL. **Manual de Desenvolvimento das Atividades**. 2015. Disponível em: < https://goo.gl/MpmkFP > Acesso em: 27/10/2017.

Medeiro, H. **Introdução ao Modelo Cascata**. DAVMEDIA. Disponível em: <https://www.devmedia.com.br/introducao-ao-modelo-cascata/29843> Acesso em:28//10/2017.

Postman. **Developing APIs is hard Postman makes it easy** Disponível em <https://www.getpostman.com/> Acesso em: 28/11/2017.

Rodrigues, L. V.; Freitas, S. de; Mendes F. (2016**). Um estudo sobre o perfil das equipes de desenvolvimento de softwares educacionais,** In: Congresso brasileiro de informática na educação - CBIE, 2016, Uberlândia - MG. Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016), 2016.

Royce, W. W. (1970). **Man*API*ng the development of large software systems: concepts and tecniques**. Proc. IEEE WESTCON, Los Angeles CA: IEEE Computer Society Press.

Sommerville, Ian. **Engenharia de Software, 8ª edição** / Ian Sommerville; tradução: Selma Shin Shimizu Melnikoff, Reinaldo Arakaki, Edilson de Andrade Barbosa; revisão técnica: Kechi Kirama. – 8ª ed. – São Paulo: Pearson Addison – Wesley, 2007.

Teste de caixa preta: o que é, como funciona e exemplos práticos. Teste de Software. Disponível em: <http://testesdesoftware.com/teste-caixa-preta/> Acesso em: 28/11/2017.

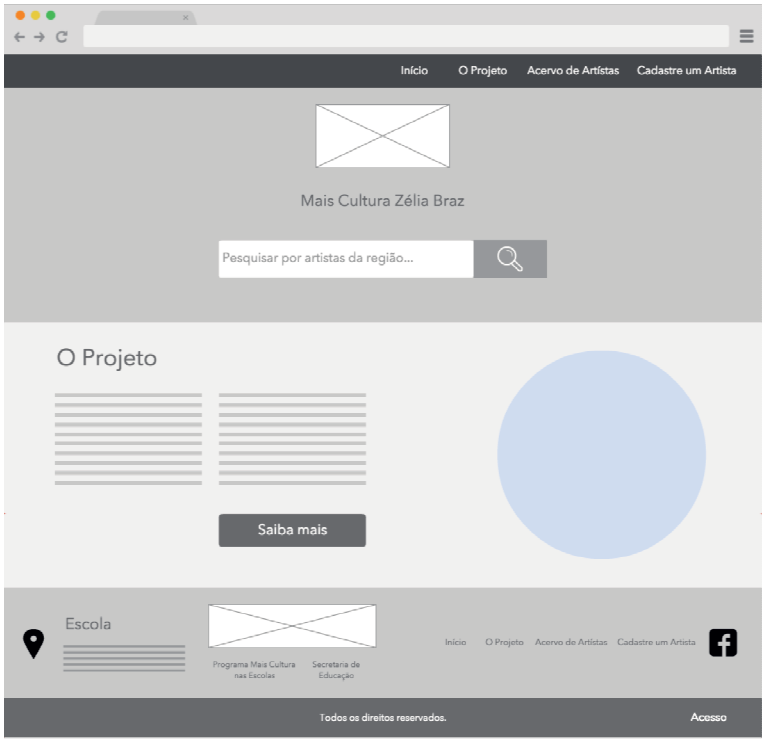
Ventura, P. **O que é Requisito Funcional.** Site Até o Momento. Disponível em: < http://www.ateomomento.com.br/o-que-e-requisito-funcional/> Acesso em: 16/11/2017.

Viana, D. (2017) **O que é front-end e back-end?** Blog TreinaWeb?. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-front-end-e-back-end/> Acesso em: 24/11/2017.

W3C. **WHAT IS CSS?** Disponível em: < https://www.w3.org/Style/CSS/Overview.en.html> Acesso em: 24/11/2017.

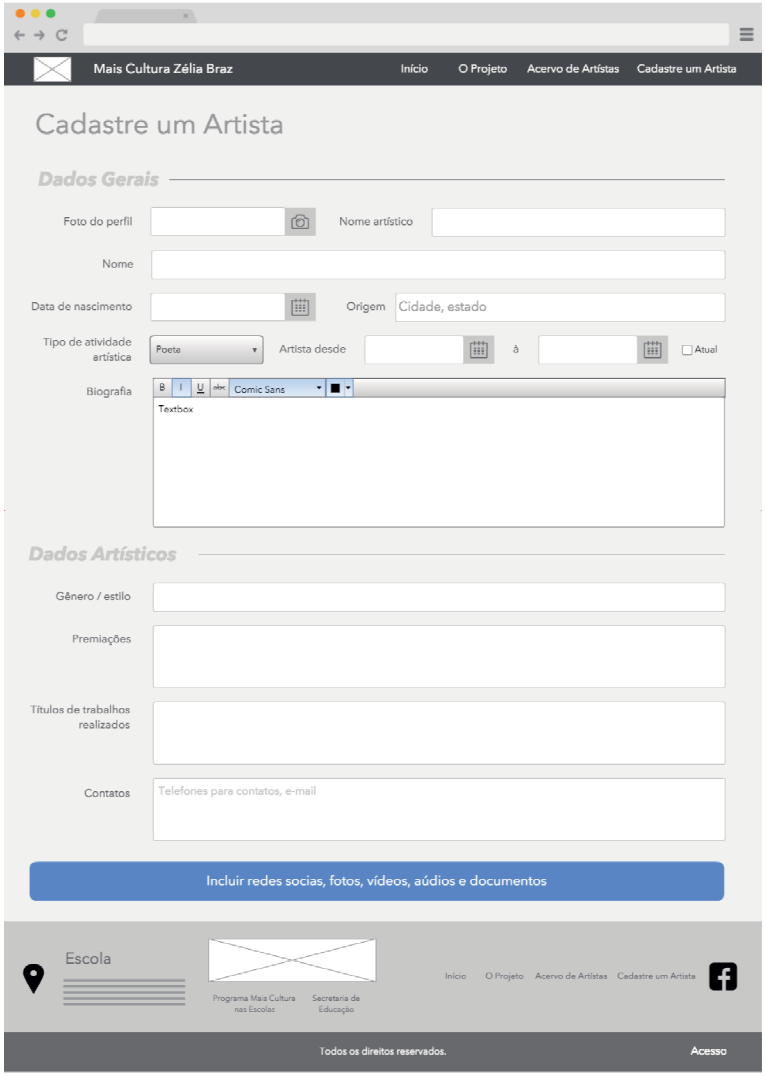
ANEXO

ANEXO A – PROTOTIPAÇÃO DO PROJETO

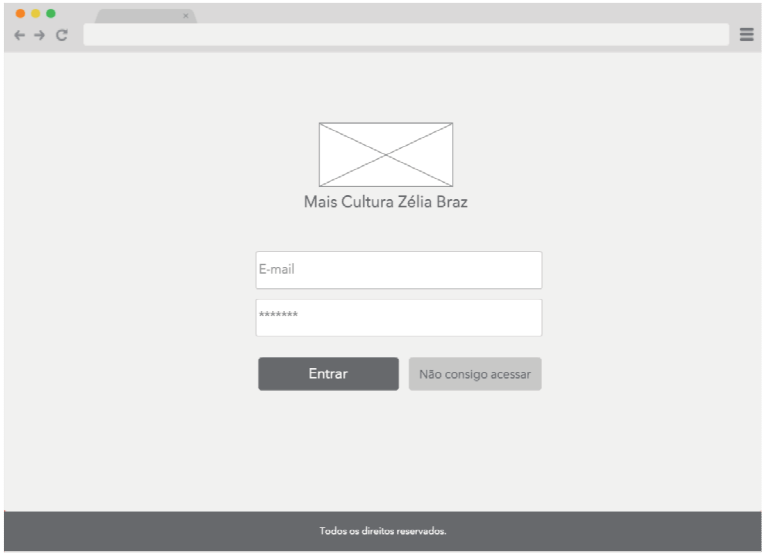


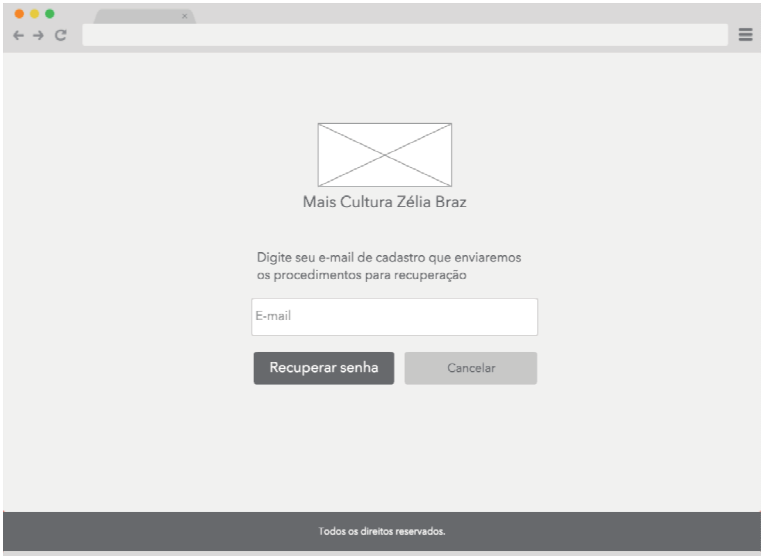

Mais Cultura Zélia Braz
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Acervo de Artistas
Pesquisar por artistas da região...
Nome do Artista
Visualizar
Nome do Artista
Visualizar
Nome do Artista
Visualizar
Nome do Artista
Visualizar
Carregar mais...
Escola
Programa Mais Cultura nas Escolas
Secretaria de Educação
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Todos os direitos reservados.
Acesso

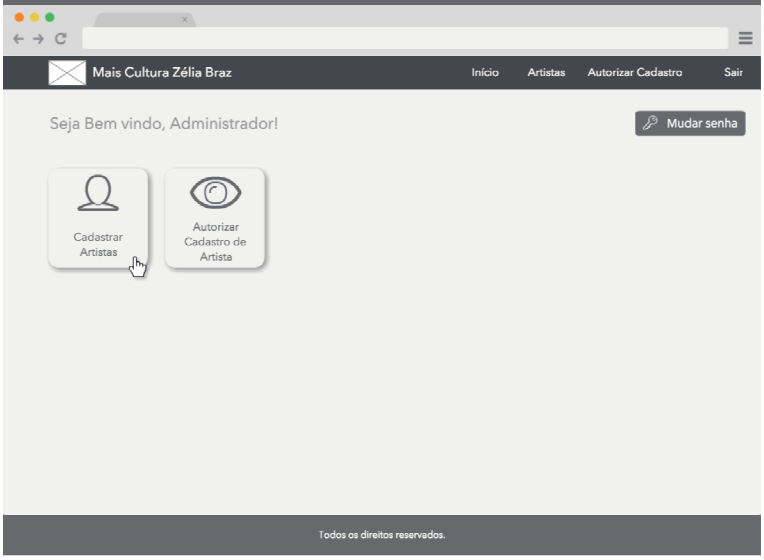

Mais Cultura Zélia Braz
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Nome do Artista
Pesquisar por artistas da região...
Dados Gerais
Dados Artísticos
Biografia
Acervo de arte
Escola
Programa Mais Cultura nas Escolas
Secretaria de Educação
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Todos os direitos reservados.
Acesso

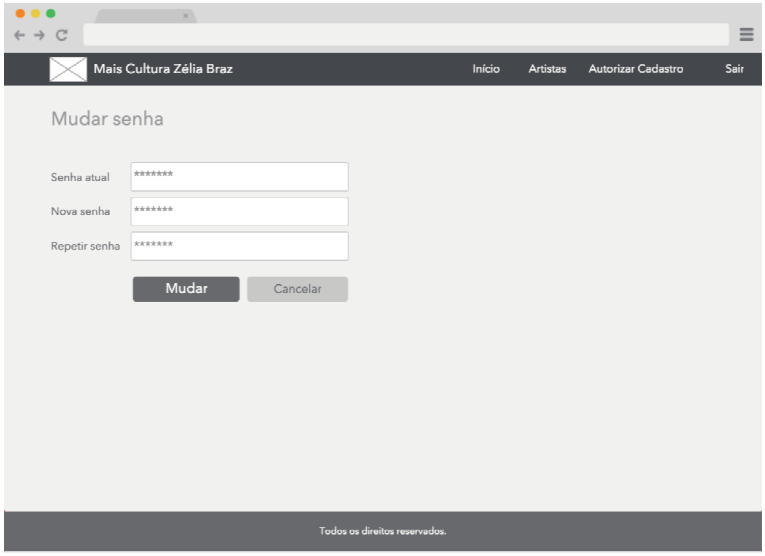

Mais Cultura Zélia Braz
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Cadastre um Artista
Dados Gerais
Foto do perfil
Nome artístico
Nome
Data de nascimento
Origem
Cidade, estado
Tipo de atividade artística
Poeta
Artista desde
à
Atual
Biografia
Comic Sans
Textbox
Dados Artísticos
Gênero / estilo
Premiações
Títulos de trabalhos realizados
Contatos
Telefones para contatos, e-mail
Incluir redes socias, fotos, vídeos, aúdios e documentos
Escola
Programa Mais Cultura nas Escolas
Secretaria de Educação
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Todos os direitos reservados.
Acesso

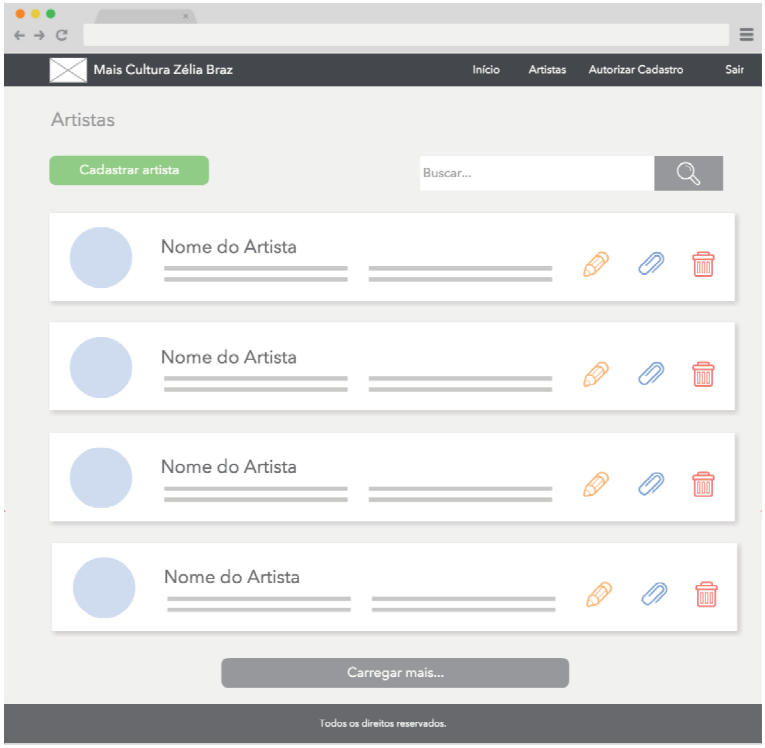

Mais Cultura Zélia Braz
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Nome do Artista
Cadastre os dados multimídias...
Voltar
Salvar
Cancelar
Redes Sociais
Facebook
Twitter
Instagram
SoundCloud
YouTube
Site oficial
Enviar arquivos
Enviar fotos...
Fotos já enviadas
Enviar vídeos...
Vídeos já enviados
Enviar documentos...
Documentos já enviados
Enviar áudios...
Áudios já enviados
Salvar
Cancelar
Escola
Início
O Projeto
Acervo de Artistas
Cadastre um Artista
Todos os direitos reservados
Acesso

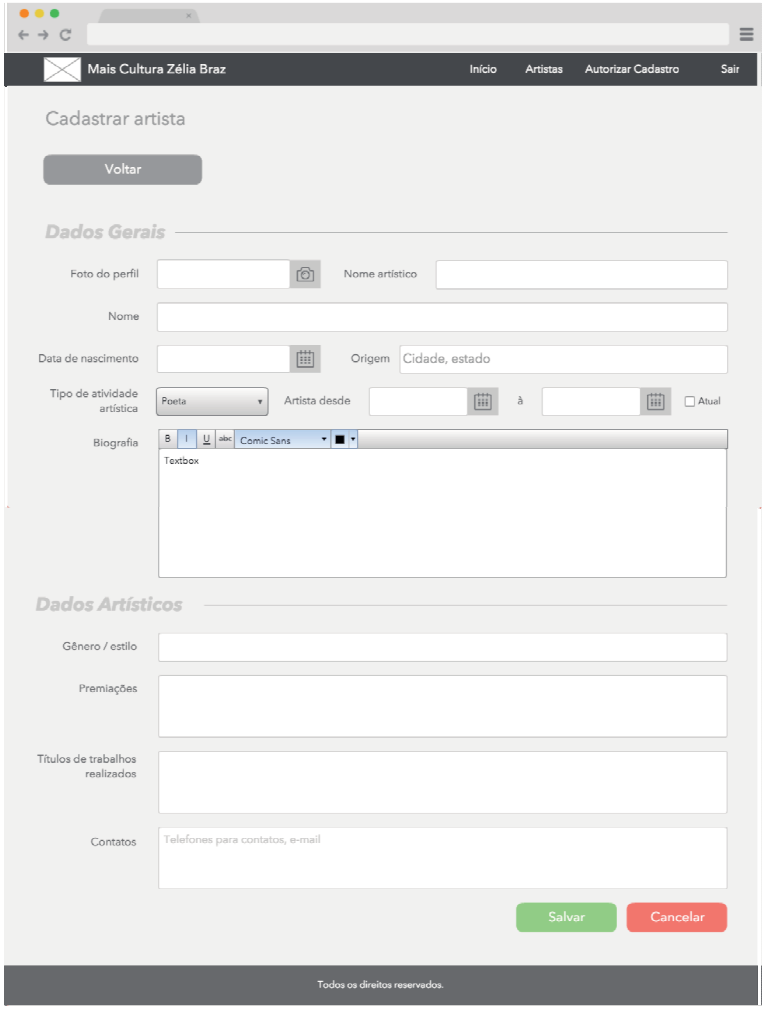

Mais Cultura Zélia Braz
E-mail
****
Entrar
Não consigo acessar
Todos os direitos reservados.

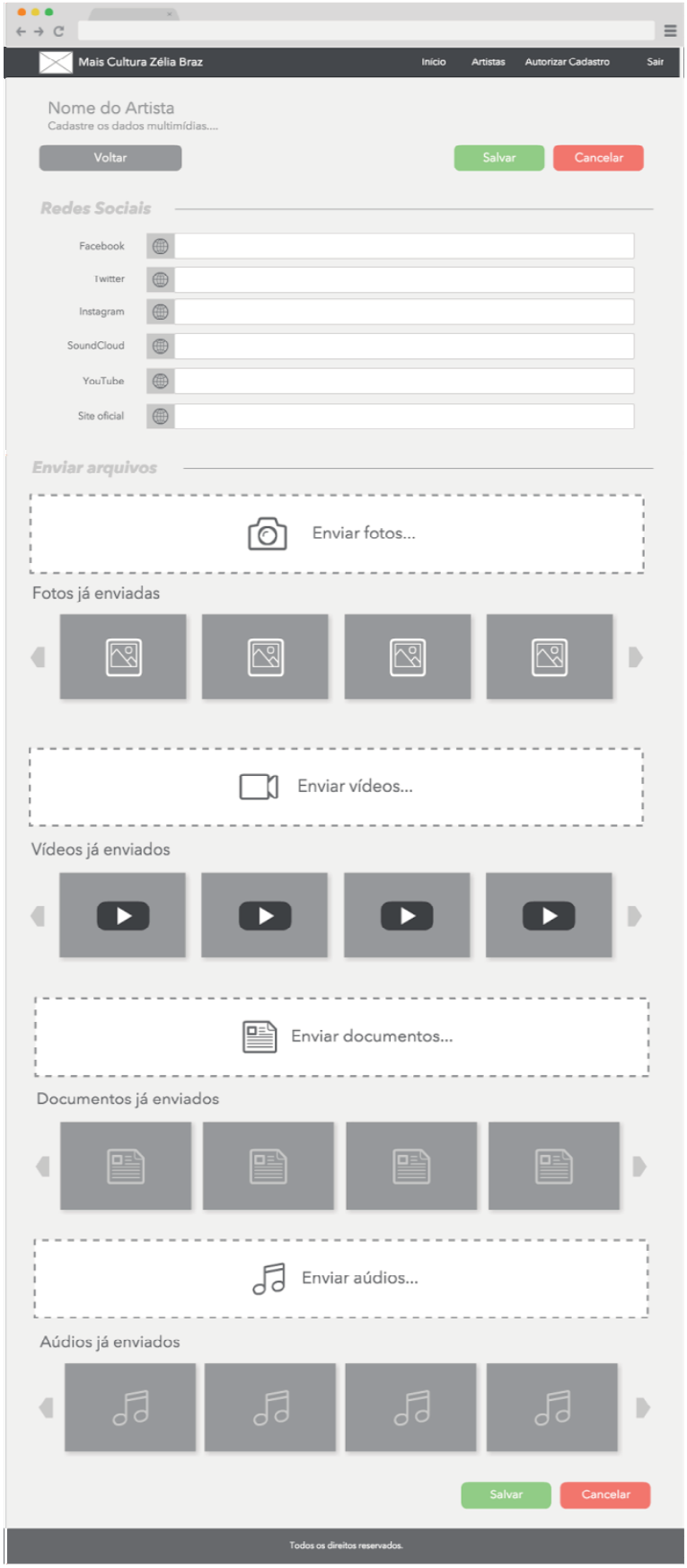

Mais Cultura Zélia Braz
Digite seu e-mail de cadastro que enviaremos os procedimentos para recuperação
E-mail
Recuperar senha
Cancelar
Todos os direitos reservados.

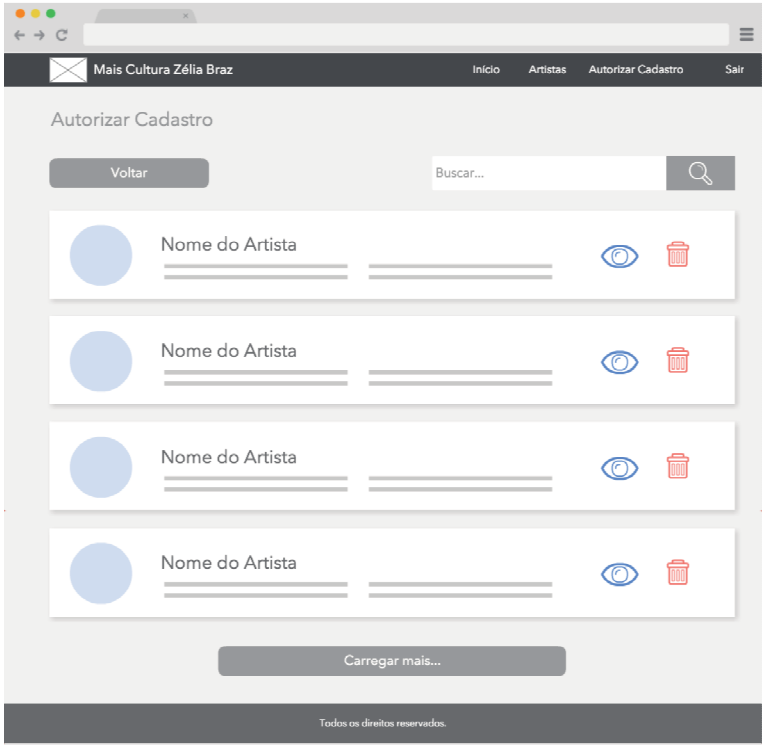

Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Seja Bem vindo, Administrador!
Mudar senha
Cadastrar Artistas
Autorizar Cadastro de Artista
Todos os direitos reservados.


Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Mudar senha
Senha atual
****
Nova senha
****
Repetir senha
****
Mudar
Cancelar
Todos os direitos reservados.


Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Artistas
Cadastrar artista
Buscar...
Nome do Artista
Nome do Artista
Nome do Artista
Nome do Artista
Carregar mais...
Todos os direitos reservados.

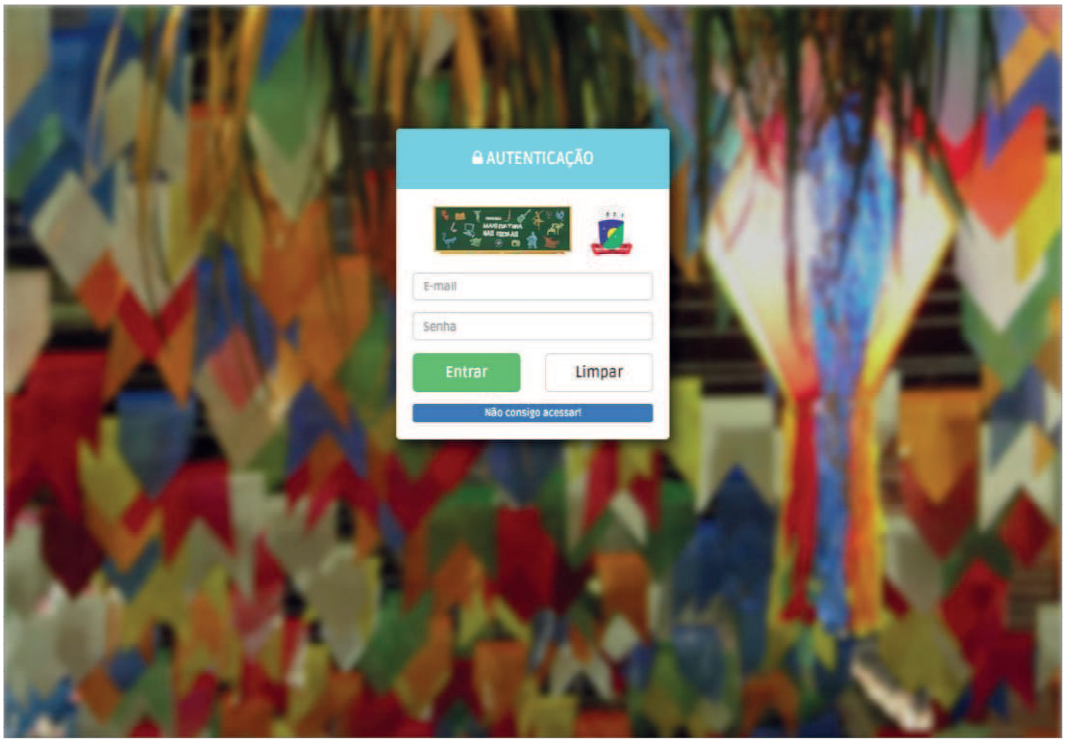

Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Cadastrar artista
Voltar
Dados Gerais
Foto do perfil
Nome artístico
Nome
Data de nascimento
Origem
Cidade, estado
Tipo de atividade artística
Poeta
Artista desde
à
Atual
Biografia
Comic Sans
Textbox
Dados Artísticos
Gênero / estilo
Premiações
Títulos de trabalhos realizados
Contatos
Telefones para contatos, e-mail
Salvar
Cancelar
Todos os direitos reservados.

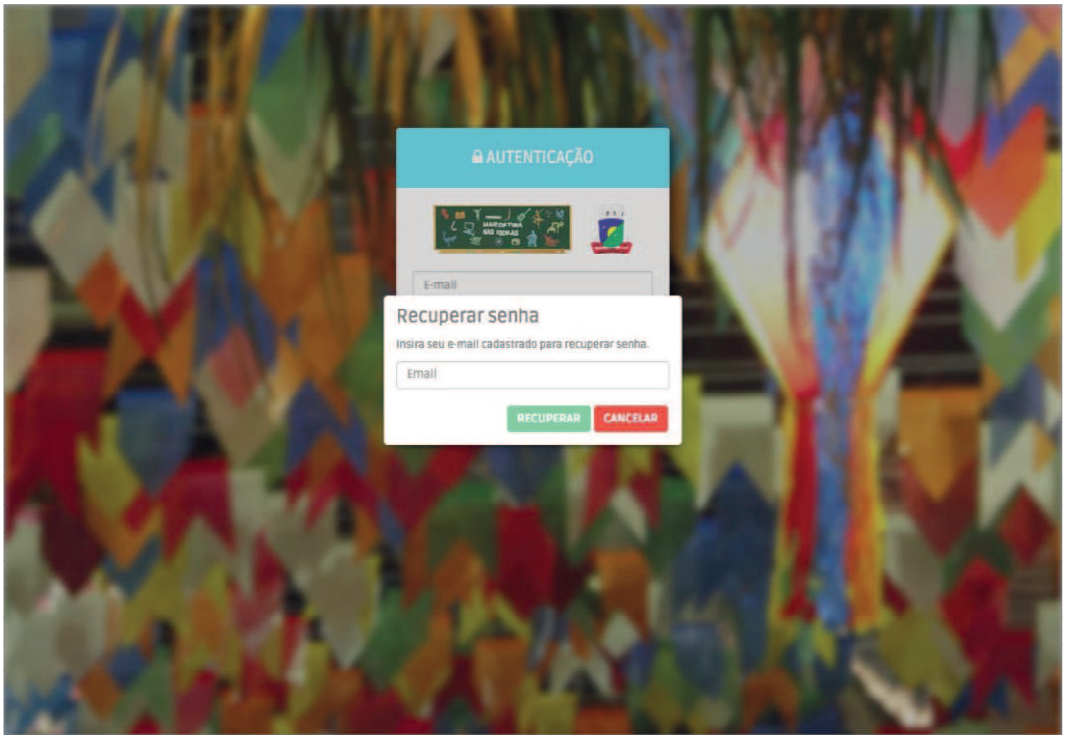

Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Nome do Artista
Cadastre os dados multimídias....
Voltar
Salvar
Cancelar
Redes Sociais
Facebook
Twitter
Instagram
SoundCloud
YouTube
Site oficial
Enviar arquivos
Enviar fotos...
Fotos já enviadas
Enviar vídeos...
Vídeos já enviados
Enviar documentos...
Documentos já enviados
Enviar aúdios...
Aúdios já enviados
Salvar
Cancelar
Todos os direitos reservados.

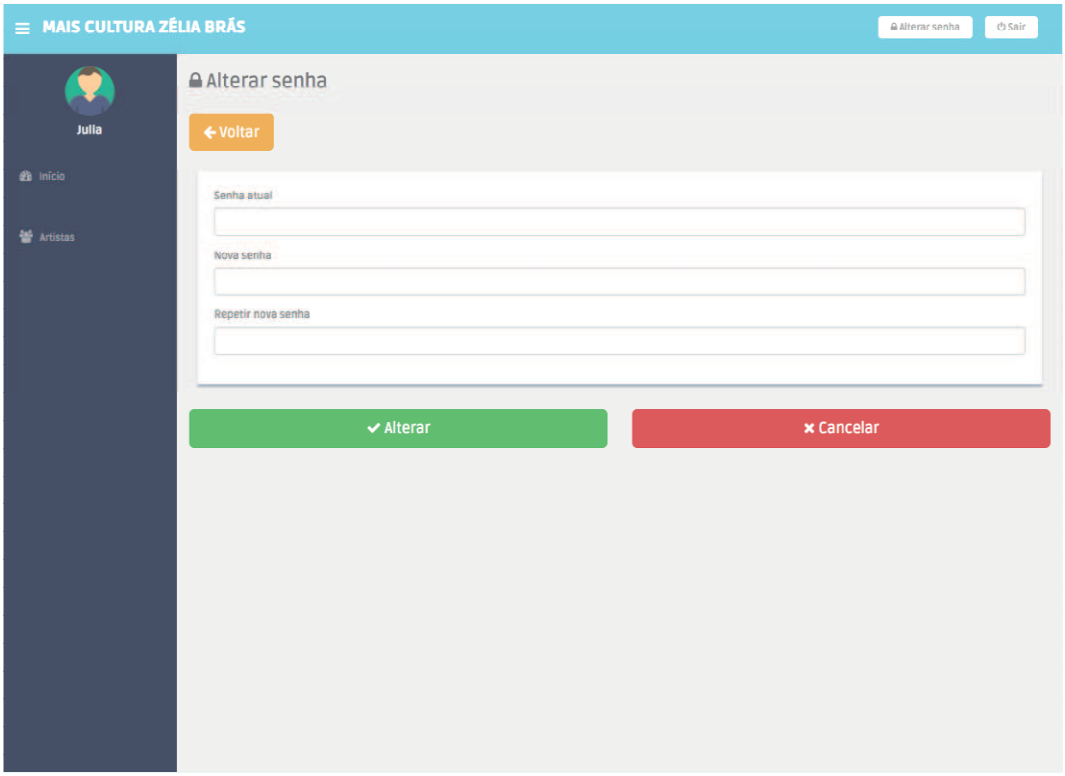

Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Autorizar Cadastro
Voltar
Buscar...
Nome do Artista
Nome do Artista
Nome do Artista
Nome do Artista
Carregar mais...
Todos os direitos reservados.

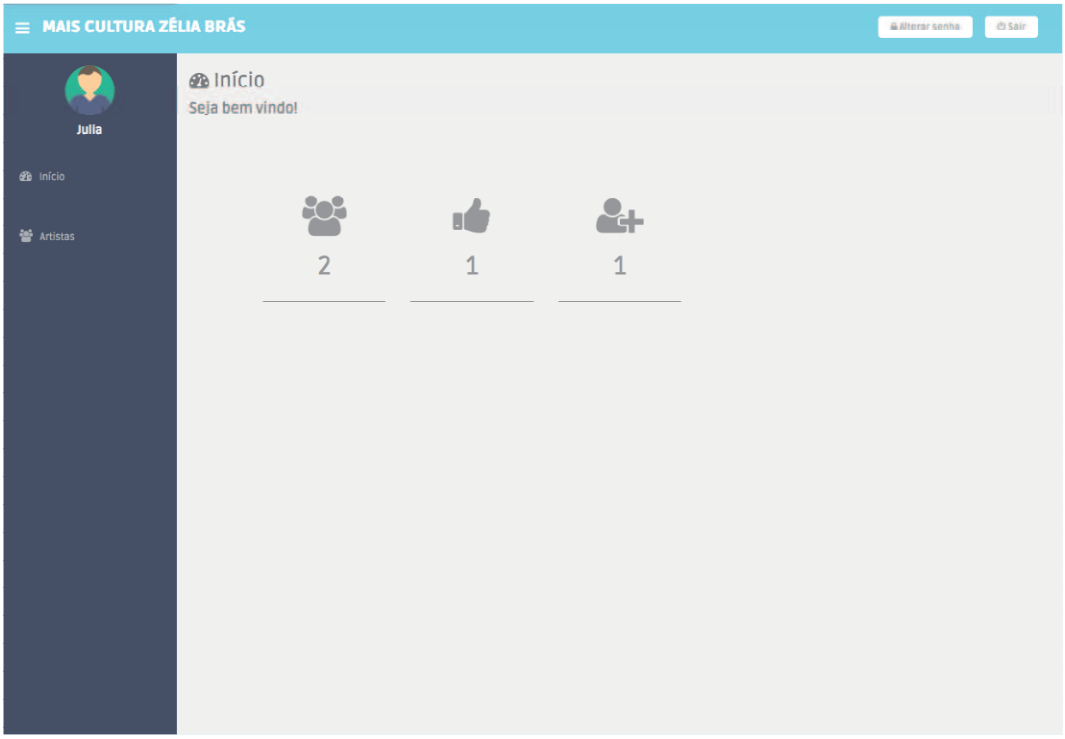

Mais Cultura Zélia Braz
Início
Artistas
Autorizar Cadastro
Sair
Nome do Artista
Autorizar Cadastro
Cancelar
Dados Gerais
Dados Artísticos
Biografia
Acervo de arte
Autorizar Cadastro
Cancelar
Todos os direitos reservados.

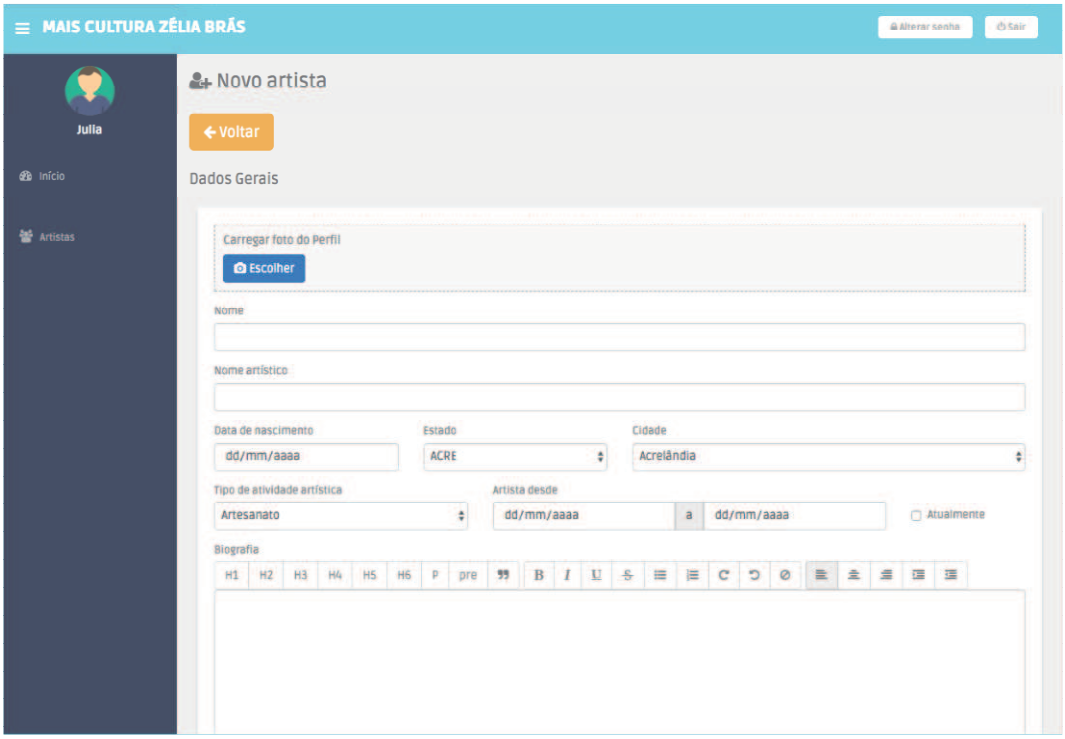
ANEXO B – TELAS DO PRODUTO FINAL – ÁREA ADMINISTRATIVA

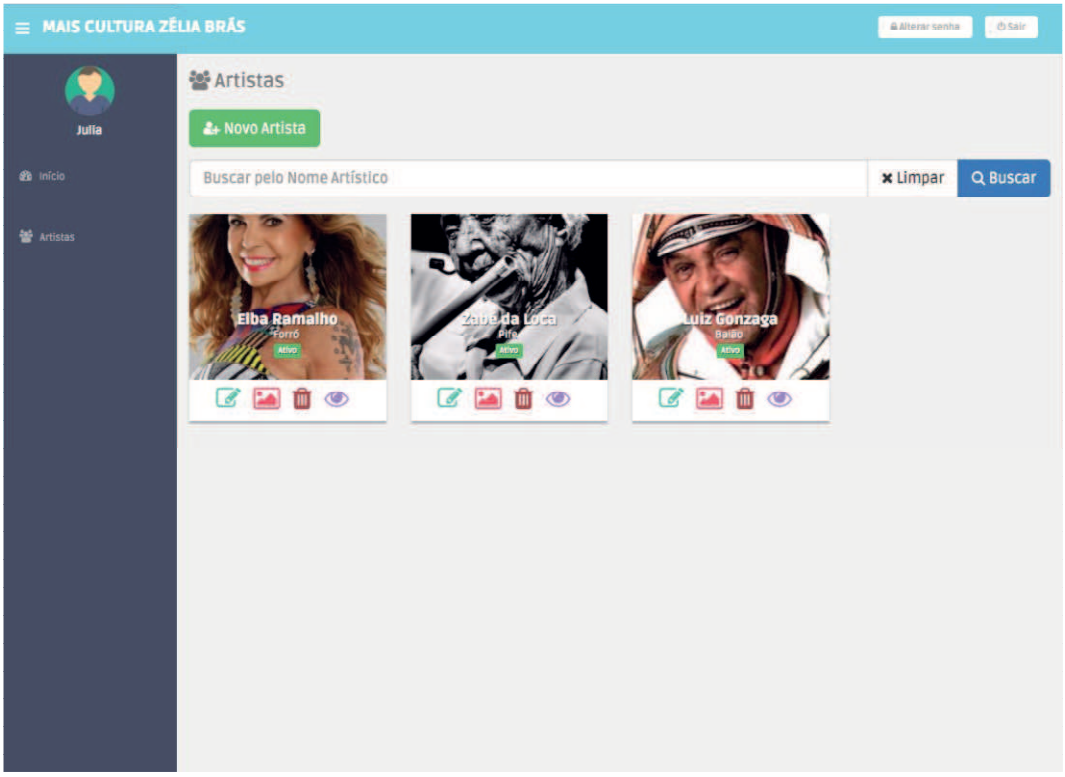




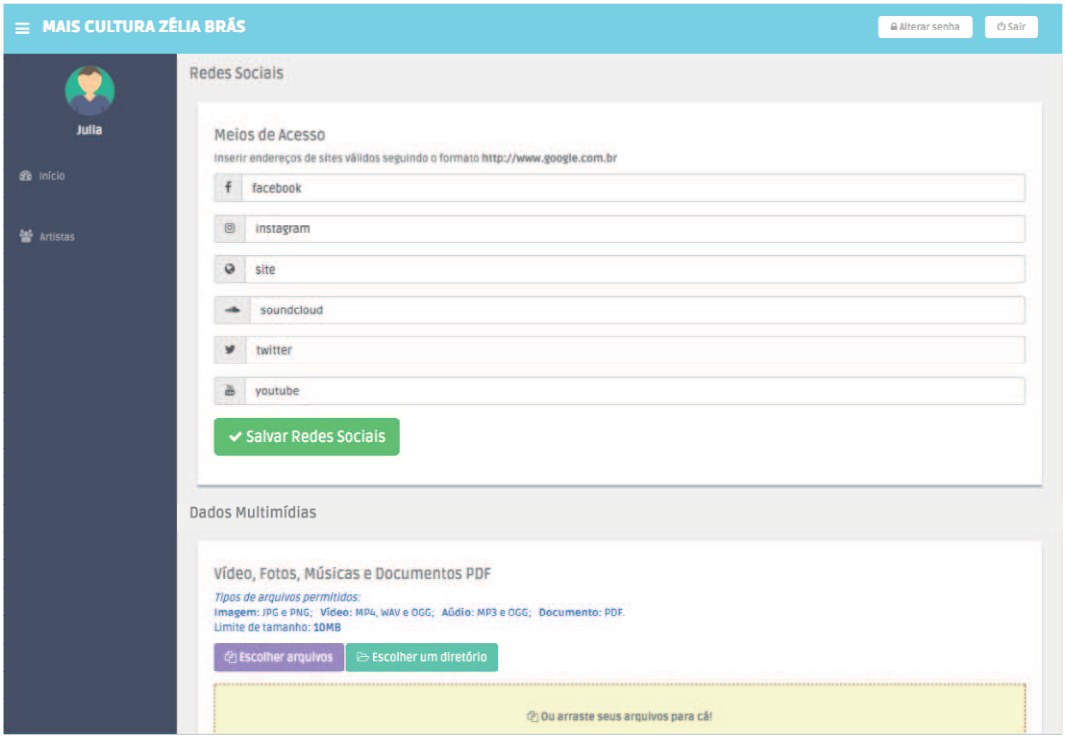

MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Alterar senha
Voltar
Senha atual
Nova senha
Repetir nova senha
Alterar
Cancelar


MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Início
Seja bem vindo!
2
1
1


MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Novo artista
Voltar
Dados Gerais
Carregar foto do Perfil
Escolher
Nome
Nome artístico
Data de nascimento
dd/mm/aaaa
Estado
ACRE
Cidade
Acrelândia
Tipo de atividade artística
Artesanato
Artista desde
dd/mm/aaaa
a
dd/mm/aaaa
Atualmente
Biografia
H1 H2 H3 H4 H5 H6 P pre


MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Artistas
Novo Artista
Buscar pelo Nome Artístico
Limpar
Buscar
Elba Ramalho
Forró
Ativo
Zabé da Loca
Ativo
Luiz Gonzaga
Baião
Ativo


MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Dados do artista
Voltar
Luiz Gonzaga do Nascimento Luiz Gonzaga
Baião
Data de nascimento:
13/12/1912
Origem:
Exu - PE
Tipo de atividade artística:
Música
Artista desde:
01/01/1940 até 02/08/1989
Premiações:
Em 13 de dezembro de 2012 o Correio Brasileiro, seguindo uma tradição filatélica, emitiu um selo postal em homenagem ao centenário de nascimento de Luiz Gonzaga.
Trabalhos:
A canção emblemática de sua carreira foi Asa Branca, composta em 1947 em parceria com o advogado cearense Humberto Teixeira.
Contato:
http://www.gonzagao.com.br/
Biografia:
Nasceu na sexta-feira, dia 13 de dezembro de 1912, numa casa de barro batido na Fazenda Caiçara povoado do Araripe, a 12 km da área urbana do município de Exu, extremo noroeste do estado de Pernambuco, cidade localizada a 610 km da capital pernambucana, Recife, a 69 km do Crato e a 80 km de Juazeiro do Norte (as duas últimas já situadas no Ceará, estado com o qual Exu faz divisa). Foi o segundo filho de Ana Batista de Jesus Gonzaga do Nascimento, conhecida na região por 'Mãe Santana', e oitavo de Januário José dos Santos do Nascimento. O padre José Fernandes de Medeiros o batizou na matriz de Exu em 5 de janeiro de 1920.[5][6]
Deveria ter o mesmo nome do pai, mas na madrugada em que nasceu, seu pai foi para o terreiro da casa, viu uma estrela cadente muito luminosa e mudou de ideia. Era também o dia de Santa Luzia e também mês do Natal, o que explica seu nome, "Luiz",que foi dado em homenagem a Santa Luzia, a estrela cadente e ao natal. Este nome tem tudo a ver com a época que nascera, e quer dizer "brilho, luz".
A cidade que nascera fica ao sopé da Serra do Araripe, e inspiraria uma de suas primeiras composições, "Pé de Serra". Seu pai trabalhava na roça, num latifúndio, e nas horas vagas tocava acordeão; também consertava o instrumento. Foi com ele que Luiz aprendeu a tocá-lo. Não era adolescente ainda quando passou a se apresentar em bailes, forrós e feiras, de início acompanhando seu pai. Autêntico representante da cultura nordestina, manteve-se fiel às suas origens mesmo seguindo carreira musical no sudeste do Brasil.[4] O gênero musical que o consagrou foi o baião.[2] A canção emblemática de sua carreira foi Asa Branca, composta em 1947 em parceria com o advogado cearense Humberto Teixeira.


MAIS CULTURA ZÉLIA BRÁS
Alterar senha
Sair
Julia
Início
Artistas
Redes Sociais
Meios de Acesso
Inserir endereços de sites válidos seguindo o formato http://www.google.com.br
facebook
instagram
site
soundcloud
twitter
youtube
Salvar Redes Sociais
Dados Multimídias
Vídeo, Fotos, Músicas e Documentos PDF
Tipos de arquivos permitidos:
Imagem: JPG e PNG; Vídeo: MP4, WAV e OGG; Áudio: MP3 e OGG; Documento: PDF.
Limite de tamanho: 10MB
Escolher arquivos
Escolher um diretório
Ou arraste seus arquivos para cá!

ANEXO C – TELAS DO PRODUTO FINAL – SITE DO PROJETO


Mais Cultura Zélia Brás
Início
O Projeto
Artistas
Adicionar Artista
Luiz Gonzaga do Nascimento Luiz Gonzaga
Baião
Data de nascimento:
13/12/1912
Origem:
Exu - PE
Tipo de atividade artística:
Música
Artista desde:
01/01/1940 até 02/08/1989
Premiações:
Em 13 de dezembro de 2012 o Correio Brasileiro, seguindo uma tradição filatélica, emitiu um selo postal em homenagem ao centenário de nascimento de Luiz Gonzaga.
Trabalhos:
A canção emblemática de sua carreira foi Asa Branca, composta em 1947 em parceria com o advogado cearense Humberto Teixeira.
Contato:
http://www.gonzagao.com.br/
Biografia:
Nasceu na sexta-feira, dia 13 de dezembro de 1912, numa casa de barro batido na Fazenda Caiçara povoado do Araripe, a 12 km da área urbana do município de Exu, extremo noroeste do estado de Pernambuco, cidade localizada a 610 km da capital pernambucana, Recife, a 69 km do Crato e a 80 km de Juazeiro do Norte (as duas últimas já situadas no Ceará, estado com o qual Exu faz divisa). Foi o segundo filho de Ana Batista de Jesus Gonzaga do Nascimento, conhecida na região por 'Mãe Santana', e oitavo de Januário José dos Santos do Nascimento. O padre José Fernandes de Medeiros o batizou na matriz de Exu em 5 de janeiro de 1920.
Deveria ter o mesmo nome do pai, mas na madrugada em que nasceu, seu pai foi para o terreiro da casa, viu uma estrela cadente muito luminosa e mudou de ideia. Era também o dia de Santa Luzia e também mês do Natal, o que explica seu nome, "Luiz",que foi dado em homenagem a Santa Luzia, a estrela cadente e ao natal. Este nome tem tudo a ver com a época que nascera, e quer dizer "brilho, luz".
A cidade que nascera fica ao sopé da Serra do Araripe, e inspiraria uma de suas primeiras composições, "Pé de Serra". Seu pai trabalhava na roça, num latifúndio, e nas horas vagas tocava acordeão; também consertava o instrumento. Foi com ele que Luiz aprendeu a tocá-lo. Não era adolescente ainda quando passou a se apresentar em bailes, forrós e feiras, de início acompanhando seu pai. Autêntico representante da cultura nordestina, manteve-se fiel às suas origens mesmo seguindo carreira musical no sudeste do Brasil.
O gênero


Mais Cultura Zélia Brás
Início
O Projeto
Artistas
Adicionar Artista
Acervo de Artistas
Buscar pelo Nome Artístico
Limpar
Buscar
U.M.E.I.E.F. Profª. Zélia Braz
Facebook


Mais Cultura Zélia Brás
Início
O Projeto
Artistas
Adicionar Artista
Adicionar Artista
Dados Gerais
Carregar foto do Perfil
Escolher
Nome
Nome artístico
Data de nascimento
dd/mm/aaaa
Estado
ACRE
Cidade
Acrelândia
Tipo de atividade artística
Artesanato
Artista desde
dd/mm/aaaa
a
dd/mm/aaaa
Atualmente
Biografia